Agence ou Service : IS-TAS

Projet : ULISSE/SITOOLS2

DOSSIER D'ARCHITECTURE SITOOLS2 2.0

Rédigé par : Equipe AKKA Sitools2	Diffusé à : Jean-Christophe MALAPERT - CNES
Approuvé par : Jean-Pascal BOIGNARD	

GROUPE AKKA Technologies – Pôle Informatique & Système
Siège social 3 rue Thierry Le Luron – 92300 LEVALLOIS-PERRET
SITE DE TOULOUSE – 6, rue Roger Camboulives – BP 13633 – 31036 TOULOUSE CEDEX 1 – Tél. 05.34.61.92.92 – Fax : 05.34.61.92.22

LISTE DES MODIFICATIONS DU DOCUMENT

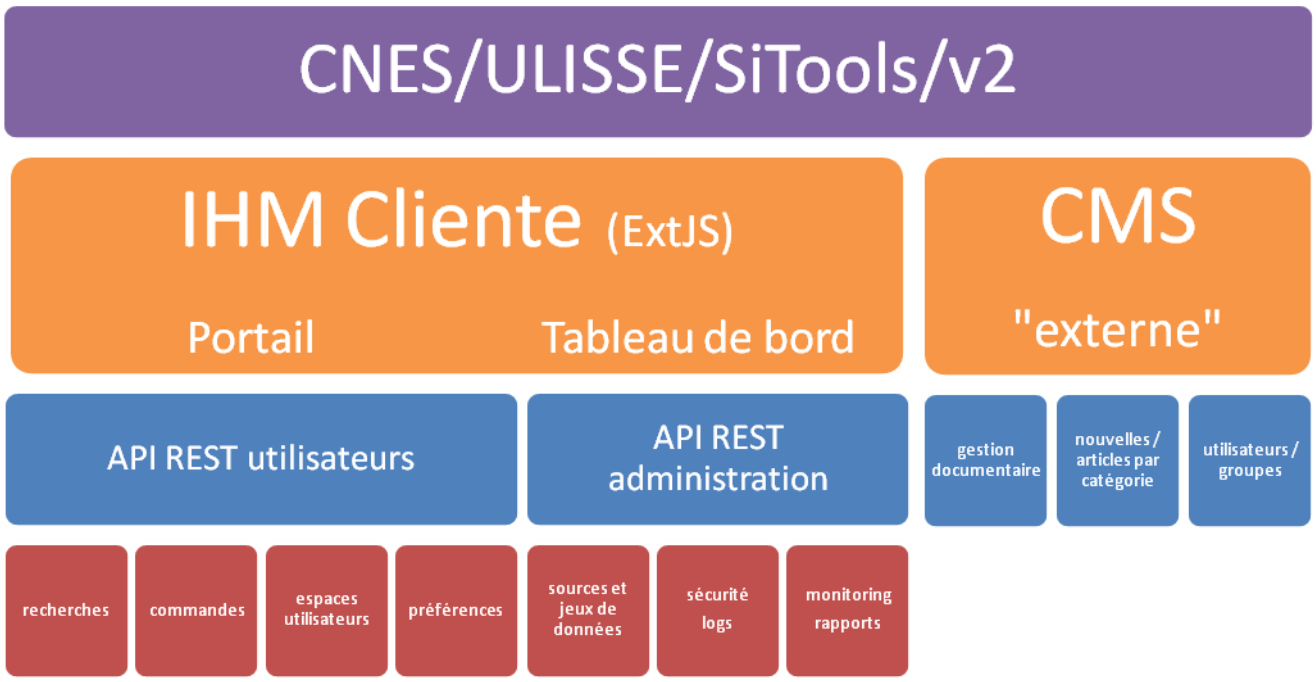
Vers.	Date	Paragraphe	Description de la modification
01.0	12/11/2012	tous	Création du document
01.1	13/02/2012	2.4.3	Nouvelle politique d'authentification sur les requêtes avec mauvaise authentification

SOMMAIRE

1 INTRODUCTION

Le Dossier d'Architecture décrit la vue technique du système. Il répond aux exigences techniques du système, et définit l'architecture matérielle et logicielle.

Pour rappel, cette architecture doit permettre de répondre dans les meilleures conditions, aux fonctionnalités décrites dans le schéma suivant :



2 ARCHITECTURE LOGICIELLE

2.1 ARCHITECTURE RESTFUL

SITools2 est construit sur les principes d'architecture RESTful, rassemblés sous le terme de ROA pour Rest Orientée Architecture par opposition à SOA Service Oriented Architecture.

Ce type d'architecture est très bien adapté au Web en reprenant les principes qui ont fait le succès d'HTTP, mais est aussi applicable à d'autres protocoles.

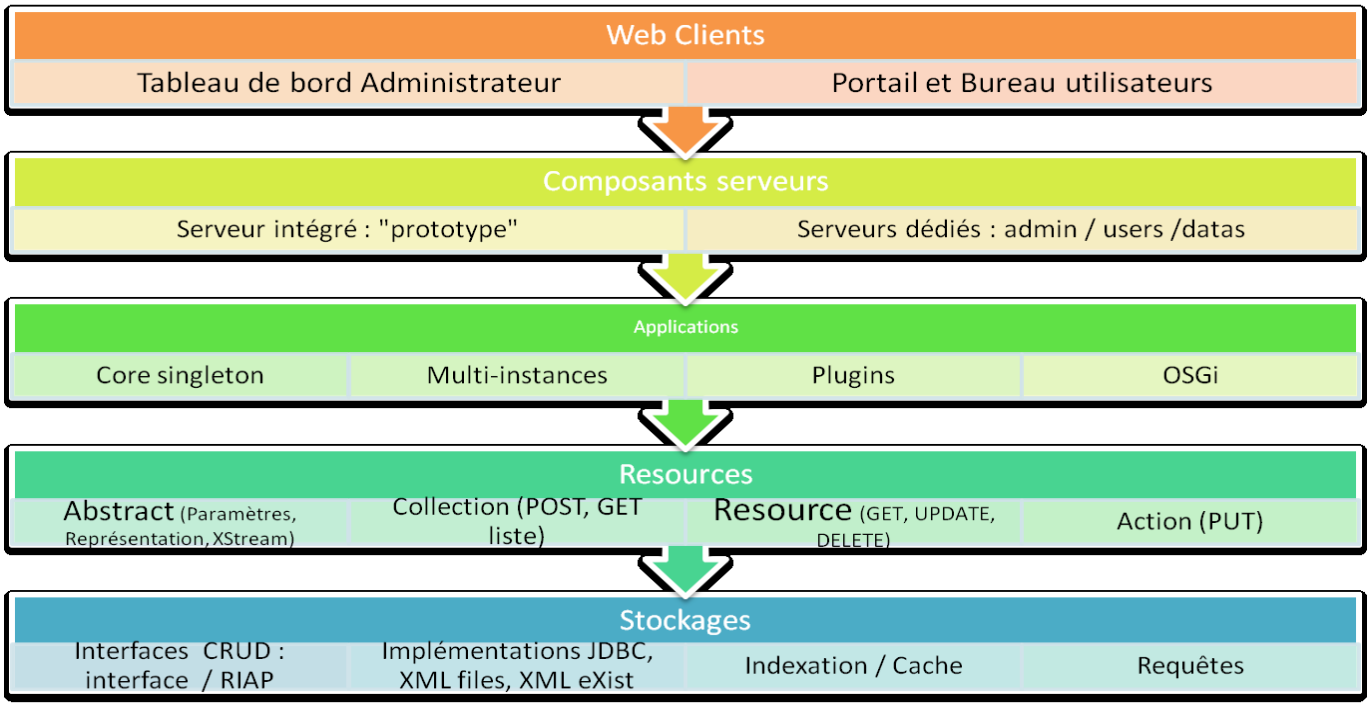
Les principales contraintes sont les suivantes :

- Les éléments ou objets manipulés sont des ressources accessibles par une URL « bien formée » et intelligible. En première intention, on n'expose pas des services mais des ressources.
- Le mode de fonctionnement est sans état, chaque requête comporte tous les éléments permettant d'identifier l'utilisateur si besoin, et tous les autres paramètres permettant au serveur d'effectuer les actions sur la ou les ressources concernées.
- Les informations échangées dans les requêtes sont les représentations de ressources exposées par le serveur. C'est l'origine du terme REST : Representational State Transfert

Pour faire vite voir l'article REST sur wikipédia

2.2 MODELE DES COUCHES LOGICIELLES

Le modèle de SITools2 reprend le découpage d'architecture en couches classique avec la mise en œuvre des concepts d'Application, de Ressources et de Representation tels que définis dans l'approche RESTful et dans le framework RESTLET qui est utilisé.



La couche **« Web Clients »**

Cette couche comprend plusieurs applications clientes, dont le tableau de bord d'administration, le portail et le bureau utilisateurs. Ces applications sont construites avec le framework javascript Sencha – ExtJS.

Il s'agit d'applications à part entière, dont l'architecture repose sur le modèle MVC.

La partie modèle du MVC est représentée par des classes Store ExtJS.

Le rôle de contrôleur est joué par les classes et méthodes chargées du traitement des événements utilisateur, et d'échange avec le serveur. Il s'agit essentiellement de méthodes javascript appelées lors d'événements levés par ExtJS (click souris, retour d'appel Ajax …)

La partie Vue du MVC est représentée par l'assemblage des composants ExtJS ou extensions tels que les classes graphiques Form, Panel, Grid...

La couche **« Composants Serveurs »**

Dans la version initiale, Sitools est composé d'un seul serveur intégrant l'ensemble des services d'administration, et d'utilisation des jeux de données.

Dans une version ultérieure, nous aurons pour objectif de répliquer différents serveurs d'exposition de données pour assurer une meilleure montée en charge.

La couche **« Applications Serveurs »**

Cette couche rassemble les composants fonctionnels de haut niveau correspondant aux fonctionnalités majeures du système. Certaines applications indispensables au fonctionnement font partie du cœur du système. D'autres applications « optionnelles » devront pouvoir se greffer par un mécanisme de « plugin ». Nous considérons aussi que l'exposition d'un jeu de données élémentaire, constitue une application à part entière, sur laquelle on peut appliquer la même stratégie de sécurité.

Les applications sont des objets non volatiles, qui possèdent les informations nécessaires pour le traitement des requêtes par les ressources. Une application pourra ainsi posséder un modèle de données en cache, et la référence des objets des couches inférieures de persistance.

La couche **« Ressources & Représentations »**

Cette couche rassemble les classes chargées du traitement de chaque requête. Il s'agit plus précisément, de décoder la représentation fournie dans le corps de la requête, d'invoquer les traitements métiers adéquats et de restituer dans le corps de la réponse la meilleure représentation.

Cette couche est fortement liée à la couche applications. Elle y fait d'ailleurs appel pour obtenir son contexte et la référence vers les interfaces de la couche de persistance des objets qui l'intéressent.

La couche **« Métier et Stockage »**

Elle est chargée de restituer ou de modifier les objets de l'application.

Chaque sous domaine possèdera sa propre interface de stockage. On pourra ainsi assembler le système en choisissant une implémentation de chaque interface. Par exemple, dans la version initiale de Sitools, la gestion des inscriptions utilisateurs et la gestion des utilisateurs et des groupes reposent sur des stockages différents, fichier XML pour les inscriptions, base de données SQL pour les utilisateurs et les groupes.

2.3 DESCRIPTION DES COUCHES LOGICIELLES

2.3.1 Couche présentation – Client ExtJS

L'architecture d'une application ExtJS consiste à assembler des composants ExtJS.

Pour plus d'information sur la conception d'une application ExtJS, se reporter au site Sencha.

Une particularité de l'interface utilisateur Sitools 2 est le chargement dynamique de composants basé sur le mécanisme javascript de « Dynamic Script Loading ». Pour plus d'information se reporter au manuel développeur.

2.3.2 Client Couche présentation – Vue du serveur

Les ressources nécessaires à la présentation sur le client web sont services par le serveur par le biais d'applications « statiques ». Une application statique est construite autour de la classe Directory de Restlet pour exposer en REST le contenu d'un répertoire de fichier.

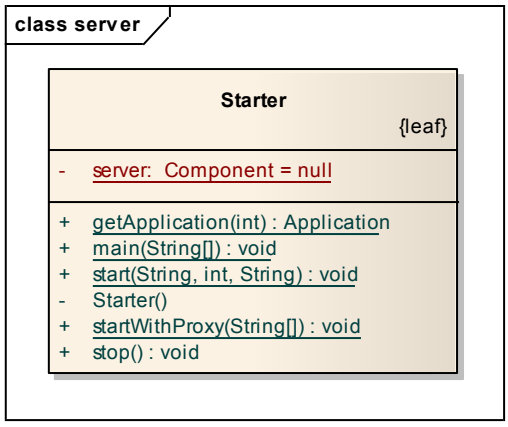
D'autres ressources HTML dynamiques sont produites par le serveur par le biais de templates freemarker. Il s'agit des pages d'accueil du portail, du projet, ainsi que la page de status d'erreur.

Ces templates personnalisables sont situés dans le dossier data/freemarker.

2.3.3 Couche Components / serveurs

Le serveur est un composant Restlet et un VirtualHost permettant de préciser le nom de domaine, le port, récupéré de la configuration.

Avec Restlet, le serveur http utilisé (actuellement jetty) est découvert au lancement de l'application.



Le démarrage du serveur consiste à créer les instances d'applications et à les attacher au serveur selon la configuration décrite dans le fichier de propriétés **sitools.properties**.

Note : Le fichier sitools.properties est généré via l'utilitaire Ant, à partir du fichier sitools-reference.properties pour prendre en compte les variables spécifiques à chaque plateforme et décrites dans les fichiers fr.cnes.sitools.core/conf/build/properties/build-{plateforme].properties.

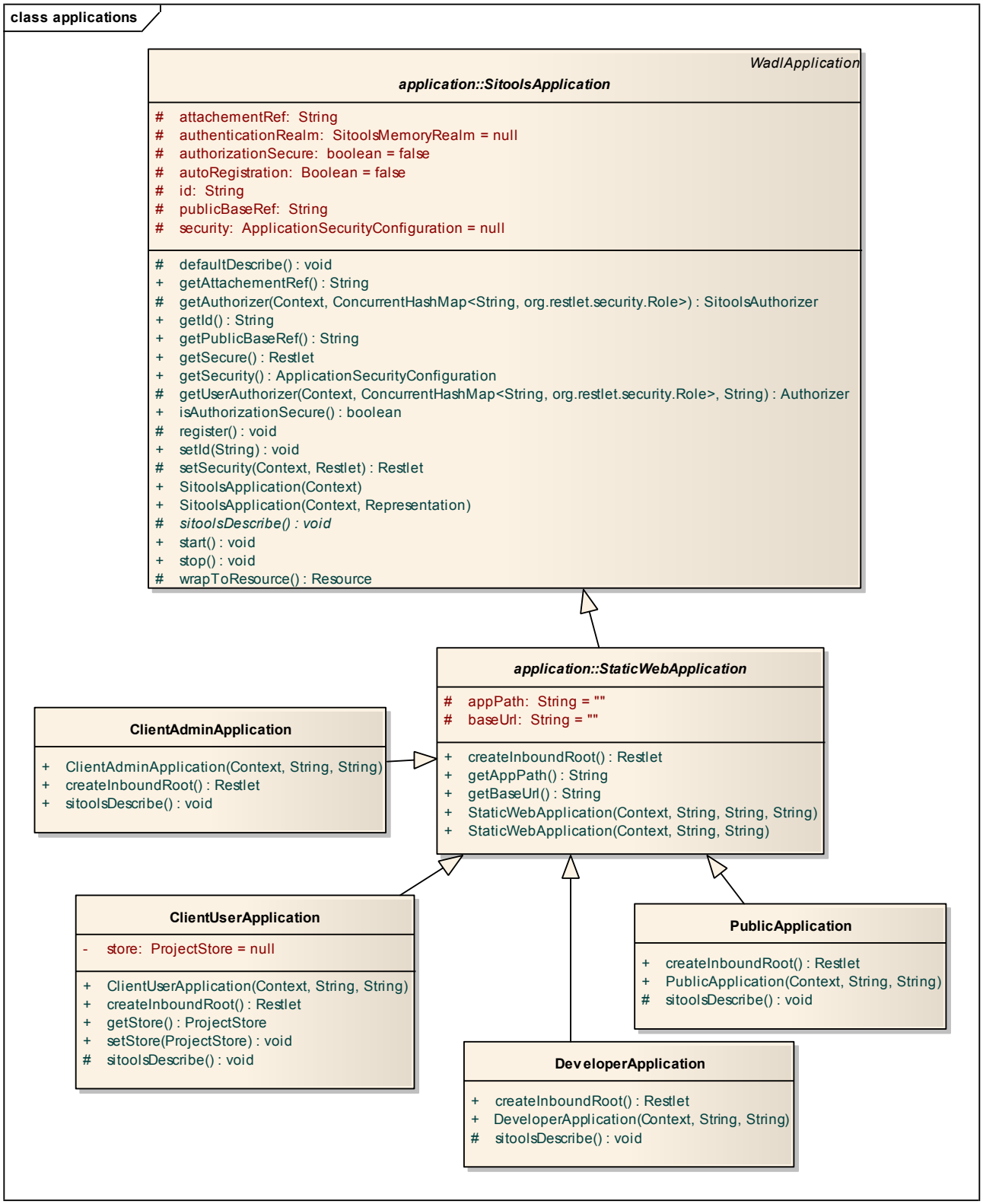
Se reporter au document d'installation et au guide du développeur pour plus de renseignements sur la configuration de ces fichiers.

2.3.4 Couche Applications

Cette couche repose sur la notion d'Application Restlet.

Pour Sitools, nous distinguons plusieurs types d'applications comme présenté dans l'arborescence de classes ci-dessous, dont la principale est la **SitoolsApplication**.

Cette classe comprend plusieurs mécanismes standards pour la description, le monitoring (start / stop) et la gestion de la sécurité qui rendent plus simple l'intégration de nouvelles applications dans Sitools, en leur permettant de bénéficier de ces mécanismes.



Classes principales :

StaticWebApplication Exposer un répertoire de ressources fichier.

SitoolsApplication

Oblige à implémenter la méthode sitoolsDescribe pour fournir la description de l'application.

La mise en place de la sécurité se fait uniformément pour toutes les applications. Cf § Sécurité.

2.3.5 Couche Ressources

Cette couche est fortement couplée avec la couche supérieure (Applications)

Le principe est d'attacher à chaque application un ensemble de classes de ressources avec un template d'URL.

Chaque classe ressource répond ensuite aux différentes méthodes GET, PUT, POST, DELETE, … et en fonction des variantes (types de media JSON, XML, …) attendues en retour.

En fonction du résultat attendu, les classes ressources utilisent des classes de représentations dédiées à chaque MediaType.

2.3.6 Couche Métier

La couche **« métier »** comprend l'ensemble des classes conteneur de données de l'application.

Schématiquement, on peut la résumer aux classes POJO décrivant les entités métier manipulées.

On retrouvera ces classes dans les packages fr.cnes.sitools.<feature>.model.

Pour l'heure, il n'a pas été jugé utile de dissocier les classes persistantes des classes retournées par le serveur.

La configuration XStream permet de rendre invisible certaines propriétés.

2.3.7 Couche Stockage

Le stockage est réalisé via XStream et un mécanisme de stratégie de persistance.

Pour l'heure, la stratégie utilisée est le stockage dans des fichiers sur le disque.

On peut envisager une évolution de Sitools2 pour changer de stratégie et stocker l'ensemble des informations dans une BD XML.

2.4 SECURITE

2.4.1 Matrice de flux

Pour un déploiement au CNES, il est demandé une matrice de flux qui énumère la liste des canaux (protocoles/ports ouverts) entre les différentes machines physiques/virtuelles du CNES, désignées par leurs adresses IP, et utiles au fonctionnement de SITools2.

Ce document constitue la demande du Responsable de Projet CNES, au support informatique CNES, pour la mise en œuvre d'une installation de SITools2 répondant à un projet particulier.

A chaque besoin d'une nouvelle installation, pour la rédaction de la matrice de flux, on pourra se reporter sur le diagramme d'architecture §3.2.3 pour énumérer et préciser les flux.

2.4.2 Sécurité – Filtrage-IP

Dans Sitools, on distingue 2 domaines réseaux que l'on appelle Intranet et Extranet.

- Intranet correspond à un ensemble de sous réseaux restreints
- Extranet correspond à tous les réseaux possibles (aucun filtrage).

Cela permet de bloquer les accès à certaines applications sur l'adresse IP d'origine de l'appel, avant toutes les autres vérifications de sécurité (autorisation, authentification …).

Il s'agit d'un filtre Restlet (classe SecurityFilter du package fr.cnes.sitools.security.filter) que l'on attache par défaut avant chaque application. La catégorie de l'application permet de savoir si une application est en Intranet ou en Extranet.

La configuration du filtrage s'effectue dans le fichier sitools.properties grâce aux propriétés suivantes :

- Security.Intranet.net : Contient la liste des adresses réseaux du domaine Intranet séparées par des | (pipe)
- Security.Intranet.mask : Contient le masque de sous réseaux commun aux adresses réseaux
- Security.Intranet.<*catégorie d'application*> : (true ou false) pour spécifier si une catégorie d'application appartient au domaine Intranet ou non. (<*catégorie d'application*> correspond à une entrée de l'énum `fr.cnes.sitools.common.model.Category`)

2.4.3 Authentification

Le modèle d'authentification repose sur les notions suivantes certaines étendues de Restlet :

- User
- Authenticator >> SitoolsChallengeAuthenticator
- ChallengeScheme (HTTP_BASIC / HTTP_DIGEST)
- Verifier
- Realm >> SitoolsMemoryRealm

La stratégie d'authentification se configure initialement dans le fichier sitools.properties :.

La propriété Starter.AUTHENTICATION_SCHEME peut en théorie prendre l'une des 2 valeurs http_basic et http_Digest (En V0.8.RC1 seule l'authentification http_Basic est opérationnelle).

La propriété Starter.AUTHENTICATION_DOMAIN est utilisée dans l'information retournée au client comme domaine d'authentification et dans la fonction l'encryptage des mots de passe (mode digest). Un seul domaine est utilisé pour l'ensemble de Sitools2.

La propriété Starter.AUTHENTICATION_ALGORITHM est utilisée pour le cryptage des mots de passe utilisateur dans le store des inscriptions et des utilisateurs.

Les services d'administration de la sécurité accessibles via l'interface d'administration permettent ensuite de

- Créer / Modifier / Supprimer des utilisateurs
- Créer / Modifier Supprimer des groupes

- Créer / Modifier / Supprimer des rôles
- Définir des stratégies d'accès aux applications pour les rôles des utilisateurs

Le rôle « public » est le seul rôle prédéfini dans Sitools2 permettant de gérer les utilisateurs non authentifiés. Il est rajouté à chaque requête.

Le service d'authentification repose sur le référentiel des utilisateurs pour constituer un SitoolsMemoryRealm à partir duquel on récupère un objet Verifier (dans le cas d'une BasicAuthentification). Ci-après une liste non exhaustive des différentes opérations réalisées pour sécuriser Sitools2 :

1. fr.cnes.sitools.security.model.User >> wrap to >> org.restlet.security.User
2. ajouter les utilisateurs au MemoryRealm
3. AuthentifcationStrategy >> wrap to >> ChallengeAuthenticator
4. MemoryRealm.getVerifier() >> associé au ChallengeAuthenticator
5. attacher le ChallengeAuthenticator au serveur et au router du scope des applications partageant la même stratégie d'authentification.

A Noter qu'à partir de la version 2.0, s'il y a une erreur dans l'authentification (mauvais mot de passe ou mauvais login) une erreur avec un code 403 est renvoyée au client. Dans les versions précédentes, la requête était exécutée avec un rôle « public ».

2.4.4 Autorisations

Le modèle d'autorisation repose sur les notions suivantes :

User >> org.restlet.security.User

Group >> org.restlet.security.Group

Role >> org.restlet.security.Role

Method (GET, PUT, POST, OPTIONS, HEAD, DELETE)

Les services d'administration permettent de

- Créer, Modifier, Supprimer des groupes
- Créer, Modifier, Supprimer des rôles
- associer à un rôle des utilisateurs et des groupes.
- définir une stratégie d'autorisation combinant rôles et méthodes autorisées.

Le service d'authorisation repose sur le référentiel des utilisateurs, des groupes et des rôles pour constituer une MemoryRealm (idem authentification) à partir duquel on récupère un objet Enroler.

1. fr.cnes.sitools.security.model.User >> wrap to >> org.restlet.security.User

   fr.cnes.sitools.security.model.Group >> wrap to >> org.restlet.security.Group

   fr.cnes.sitools.security.model.Role >> wrap to >> org.restlet.security.Role

2. ajouter les utilisateurs, groups, roles et mappings au MemoryRealm

3. model.AuthorizationStrategy (*) >> wrap to >> restlet Authorizer

4. MemoryRealm.getEnroler() >> associé au ChallengeAuthenticator (idem Verifier)

5. attacher le ChallengeAuthenticator au serveur et au router du scope des applications partageant la même stratégie d'authentification.

(*) Au niveau IHM, la définition d'une stratégie d'autorisation consiste à définir pour chaque Rôle les méthodes autorisées.

Pour éviter de devoir définir une stratégie d'autorisation pour chaque application :

- Définir une stratégie d'autorisation par défaut
- Définir une stratégie d'autorisation pour un ensemble d'application (suppose aussi que la même stratégie d'authentification est appliquée pour cet ensemble d'applications) >> Evolution non encore implémentée en 0.8.RC1.

2.4.5 Extension des capacites de sécurisation

Pour spécialiser la sécurité des éléments de SITools2 plusieurs

- Dans le fichier de properties, il est possible de spécifier une classe de filtre pour toutes les applications. La configuration par défaut est celle d'un filtre BlackList et Intranet/Extranet.
- Pour les DataStorage (exposition de ressources fichiers), l'administrateur peut spécifier pour chacun, une classe de filtre d'accès. Par défaut, aucun filtre n'est configuré. Un filtre de test est fourni en exemple pouvant être spécialisé selon les besoins d'une instance utilisatrice.

2.4.6 HTTPS

SITools2 peut être configuré en mode sécurisé avec HTTPS. Cette configuration est globale pour tout le serveur.

2.5 LOGGING

2.5.1 Logs applicatifs

Toutes les applications héritant de SitoolsApplication possèdent un logger applicatif, défini par défaut à la création de l'instance à partir du nom de l'application (Application.getName()) ou par défaut à partir du nom de la classe de l'application (< ? extends SitoolsApplication>.class.getName()).

Par défaut le nom de l'application est utilisé pour rechercher le logger.

2.5.2 Logs d'accès

Les logs d'accès se configurent dans le fichier sitools-reference.properties.

Les propriétés configurables sont les suivantes :

```
--= configuration des logs d'acces
Starter.LogService.outputFile=./logs/sitools-log-service.log
Starter.LogService.levelName=FINEST
```

```
Starter.LogService.logName=sitools.server
Starter.LogService.logFormat={cia} {m} {S} {rp}     AGENT:{cig}     REF:{fp}
Starter.LogService.active=true
```

2.5.3 Logs d'accès pour application métier

Les logs d'accès pour application se configurent dans le fichier sitools-reference.properties.

Les propriétés configurables sont les suivantes :

```
--= configuration des logs d'acces
Starter.AppLogService.outputFile=../fr.cnes.sitools.core/logs/sitools-log-
application-service.log
Starter.AppLogService.levelName=FINEST
Starter.AppLogService.logName=fr.cnes.sitools.logging.application
Starter.AppLogService.logFormat=
Starter.AppLogService.active=true
```

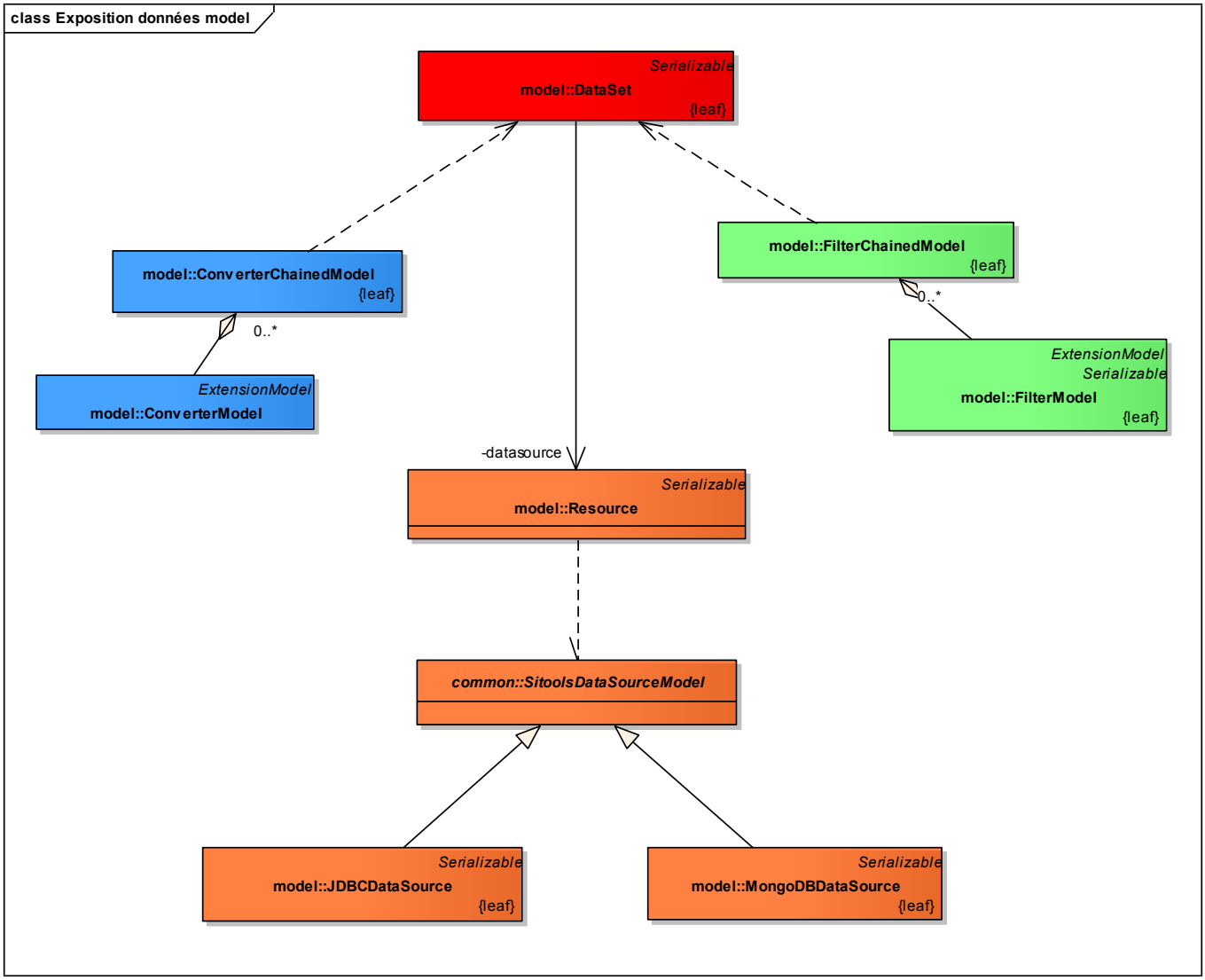
De plus, il faut mettre dans le « Context » de l'application la propriété « ContextAttributes.LOG_TO_APP_LOGGER » avec la value « Boolean.TRUE ». Les accès à ces applications seront donc loggés dans un fichier particulier et pourront être traités par Analog par exemple.

2.6 MODELE D'EXPOSITION DE DONNEES

2.6.1 Modèle objet

La fonction principale de SITools2 est l'exposition de données. Cette partie présente la façon dont cette fonctionnalité est mise en place au sein de l'application.

Le modèle de données est le suivant :

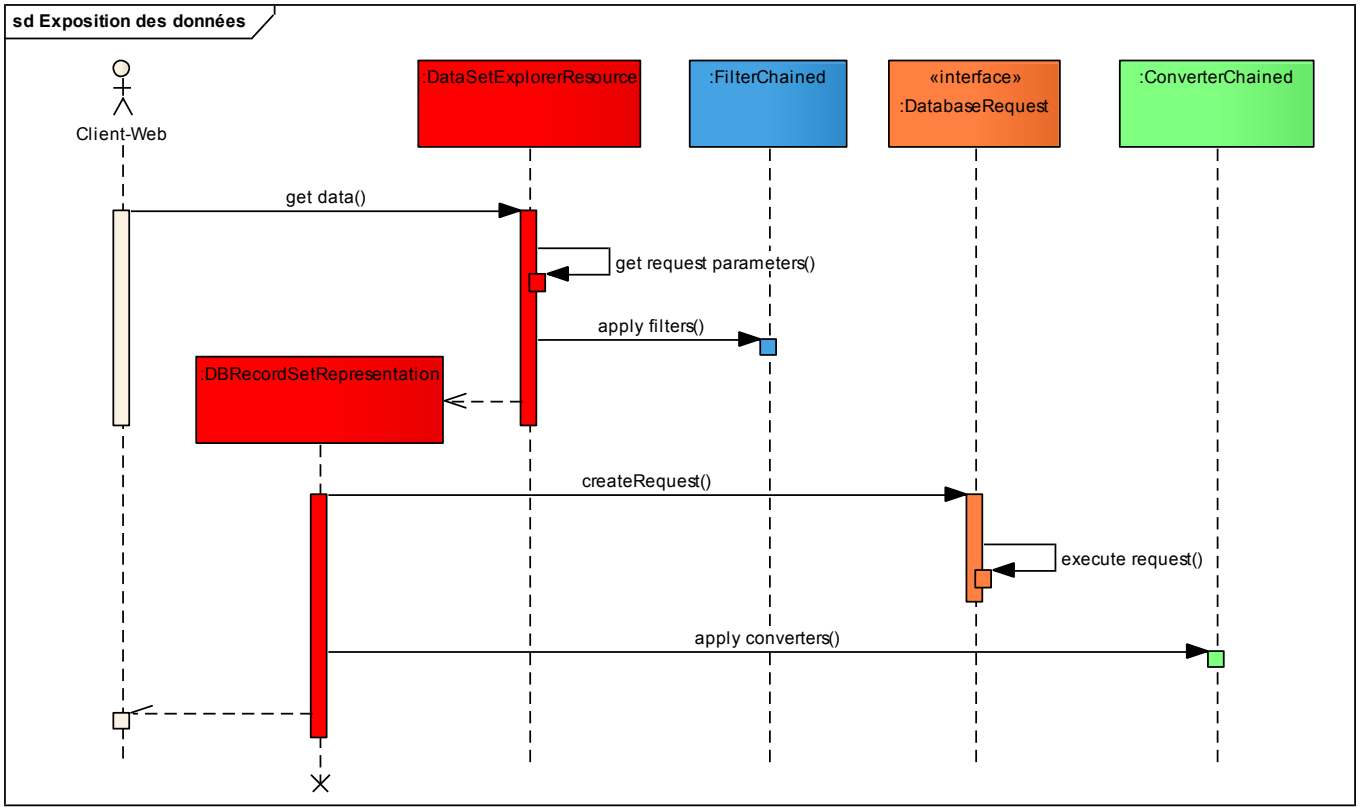


La classe ***DataSet*** regroupe l'ensemble des informations concernant l'exposition de données. Elle est liée, indirectement via une classe ***Resource***, à une ***SitoolsDataSourceModel***. Cette classe contient les informations de connexion à une source de données. Il existe actuellement 2 implémentations de source de données, à savoir une source de données basée sur JDBC pour PostgreSQL et MySQL et une autre pour MongoDB.

Le ***DataSet*** est également lié à une liste de filtres (***FilterChainedModel***) dont le but va être d'ajouter des filtres de requête avant qu'elle soit envoyée à la source de données. Il est également lié à une liste de convertisseurs (***ConverterChainedModel***) dont le but sera de convertir les données une fois la requête effectuée par la base de données.

2.6.2 Modèle dynamique

Le principe de l'exposition des données dans SITools2 est le suivant :



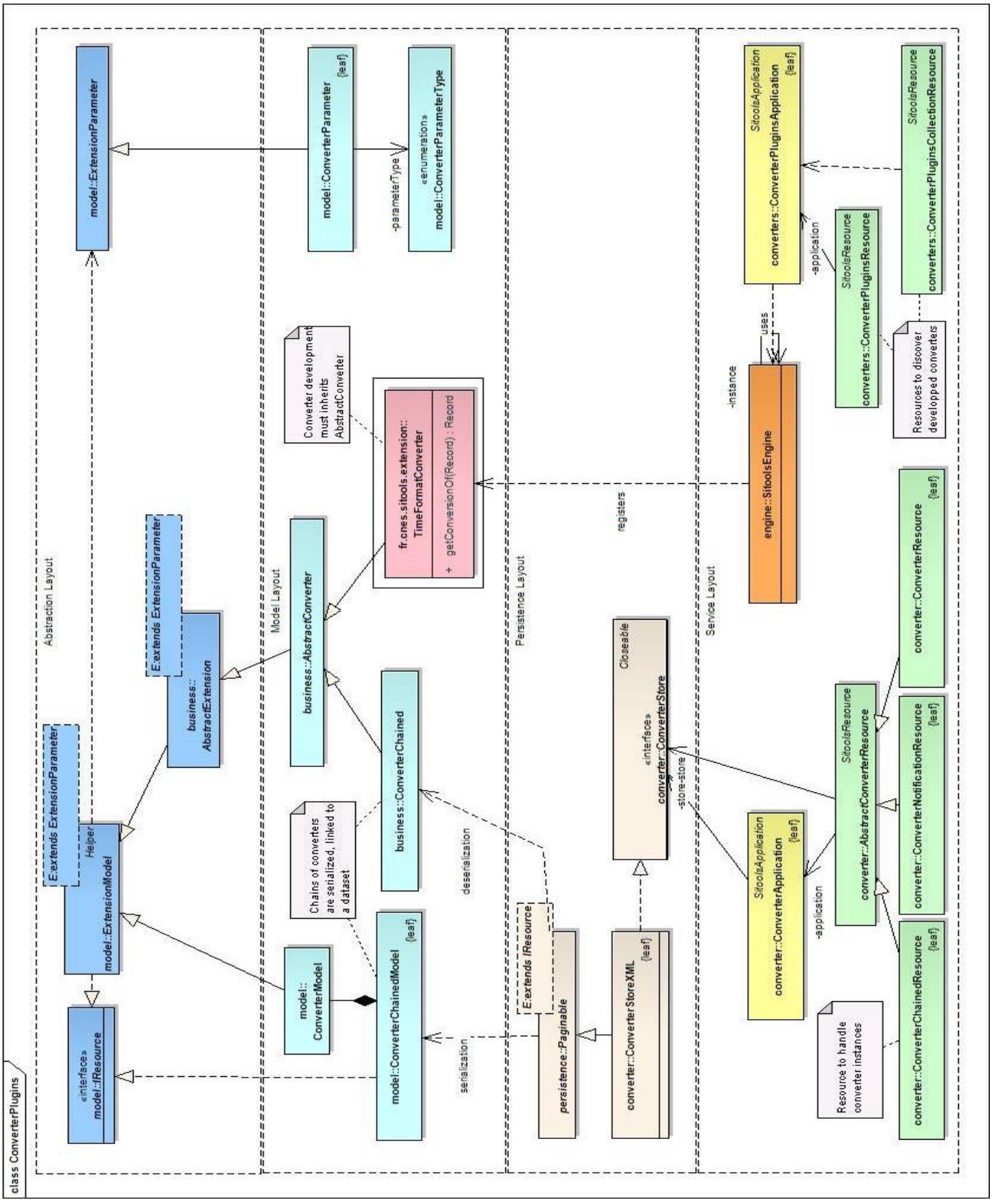
- Le client interroge une ressource serveur en REST avec une liste de paramètres correspondants à des critères de recherche.
- Cette ressource applique une liste de filtres qui analysent ces critères de recherche ajoutent des ***Predicats*** (condition de requête générique) à la requête
- Cette ressource crée une Représentation (***DBRecordSetRepresentation*** dans notre exemple) qui sera en charge d'effectuer la requête et de retourner un flux de donnée.
- Cette représentation demande à la requête de s'exécuter au niveau de la base de données de façon transparente (elle ne sait pas s'il s'agit de MySQL, PostgreSQL ou MongoDB)
- Cette représentation applique une liste des convertisseurs sur les résultats.
- Cette représentation formate le résultat au format XML ou JSON et l'envoi **en streaming** au client

Cette architecture est optimale du point de vue des performances, du respect des principes REST (représentations selon le Header Accept-Content http, possibilités de cache, …), et de l'extensibilité (ajout de nouvelles sources de données, de filtres de requêtes, de post-traitements de record via convertisseur) .

2.7 EXTENSION DE L'EXPOSITION DE DONNEES DE DATASOURCES

2.7.1 Plugins de convertisseurs

- La couche service se décompose en deux applications : la première expose les convertisseurs qui ont été développés ; la seconde administre ces convertisseurs pour les assembler en chaine et les rattacher à un dataset.
- La couche de persistance se base sur la sérialisation d'un modèle de chaine de convertisseur simple. La déserialisation se base sur le nom de la classe de convertisseur enregistré pour créer une nouvelle instance de celui-ci, puis le rajouter dans le modèle de chaine de convertisseur.
- Le développement se situe au niveau de la couche modèle, les convertisseurs devant hériter de la classe abstraite AbstractConverter. Ils sont ensuite enregistrés dans le fichier Helper dans src/META-INF des extensions pour pouvoir être découverts par le SitoolsEngine.
- La couche abstraite est constituée de l'architecture de base pour tous les objets ayant pour but d'être étendu, notamment autour de la classe abstraite ExtensionModel.



2.7.2 Plugins de filtres

Un mécanisme similaire aux convertisseurs est mis en place pour la gestion et l'utilisation de filtres.

2.8 MODELE D'EXTENSION DE SITOOLS2

2.8.1 Modèle d'extension de services (API)

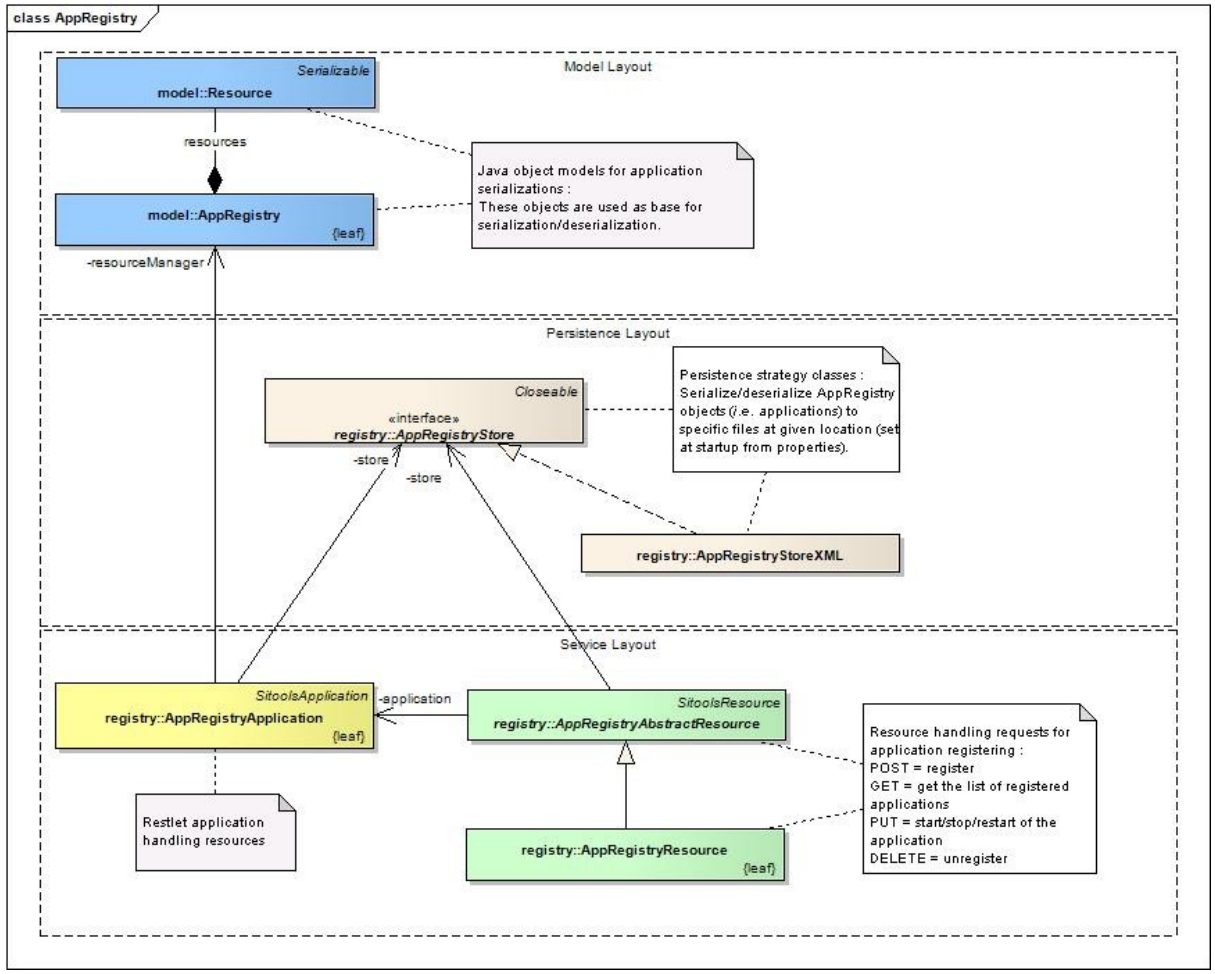
Nous présentons ici le modèle général d'extension de SITools2, en présentant les éléments pouvant être spécialisés par des classes développées à l'extérieur du core, pour ajouter de nouvelles fonctionnalités, de nouveaux services.

2.8.2 AppRegistry : application d'enregistrement et de découverte des applications

- La couche service est principalement dédiée à l'enregistrement de nouvelles applications au niveau du registre. Elle sert également à gérer les actions sur l'application (START/STOP/RESTART).
- La couche de persistance enregistre des éléments de registre.
- La couche model est composée d'un objet AppRegistry (registre entier) qui embarque une liste de ressources (applications).

Agence ou Service : IS-TAS

Projet : ULISSE/SITOOLS2



Agence ou Service : IS-TAS

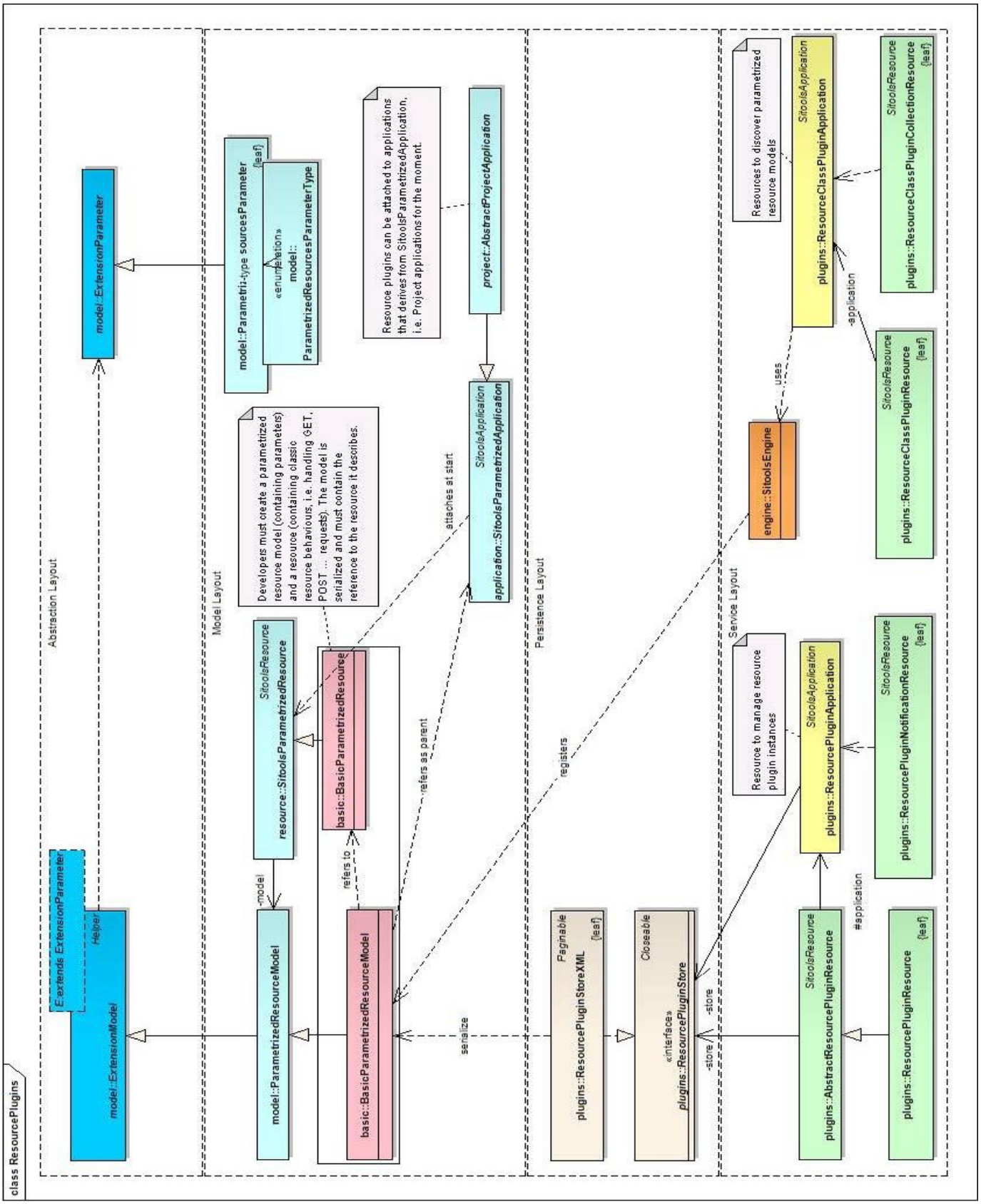
Projet : ULISSE/SITOOLS2

2.8.3 Plugins d'application

Un mécanisme similaire aux convertisseurs est mis en place pour la gestion et l'utilisation d'applications.

2.8.4 Plugins de ressources

- La couche service se décompose en deux applications : la première expose les modèles de ressources qui ont été développés ; la seconde administre ces modèles pour les paramétrer et les associer à des applications qui acceptent ce type de ressource.
- La couche de persistance se base sur la sérialisation d'un modèle ressource héritant de ParametrizedResourceModel et développé par l'utilisateur de SITools2. Ce dernier contient tous les éléments nécessaires à l'instanciation de la ressource et à son attachement à l'application.
- Le développement se situe au niveau de la couche modèle, les modèles de ressources étant accompagnés de la classe de ressource qu'ils réfèrent. Cette ressource contient tous les éléments nécessaires pour qu'elle se comporte comme une ressource Restlet classique, l'avantage étant qu'elle a accès aux paramètres du modèle. Cette ressource doit obligatoirement hériter de SitoolsParametrizedResource. Le modèle de son côté, est référencé dans le fichier Helper de src/META-INF pour pouvoir être découvrable par le SitoolsEngine.
- La couche abstraite est constituée de l'architecture de base pour tous les objets ayant pour but d'être étendu, notamment autour de la classe abstraite ExtensionModel.



2.9 MODELE D'EXTENSION DE L'IHM

2.9.1 Extension de l'IHM UTILISATEUR

2.9.1.1 Modules de projets

SITools2 peut être étendu simplement pour couvrir des besoins utilisateurs spécifiques par le biais de modules de projets. Un module de projet est un ensemble de fichiers javascript répondant à certaines règles et placés dans l'arborescence du client-user.

Le scénario général de mise en œuvre d'un module de projet dans SITools2 est le suivant :

Depuis l'interface utilisateur client-admin, il faut ensuite déclarer les modules avec en particulier la classe principale.

Ensuite au niveau de la configuration d'un projet particulier, il est possible d'associer certains modules, de définir les droits d'accès et les propriétés spécifiques au module pour le projet en question.

2.9.1.2 Dataset Views

SITools2 peut être étendu pour offrir des représentations de jeux de données (DataSetView) plus adaptées au contexte utilisateur. A l'heure actuelle, il en existe plusieurs, avec /sans pagination, avec cartographie, vue composite avec formulaire de recherche par défaut.

C'est une possibilité d'extension essentielle de SITools2 pour mettre en œuvre rapidement des visualisations adaptées au besoin utilisateur.

Le scénario suit le même principe que celui d'un module de projet :

- Copie dun répertoire des fichiers de l'extension dans l'arborescence client-user.
- Déclaration dans l'interface d'administration, de la datasetview avec sa classe
- Association d'un dataset avec sa vue par défaut et définition des propriétés spécifiques.

2.9.2 Extension de l'IHM ADMINISTRATEUR

2.9.2.1 Fonction d'administration

SITools2 peut être étendu par des nouvelles fonctions d'administration. Cet aspect des fonctions d'administration interne est pour l'instant réservé à des fonctions « core » de SITools2. Nous estimons que les contributions les plus courantes s'effectueront par le biais des modules de projets.

2.10 INSTALLATION ET MISE EN ŒUVRE

L'architecture de SITools2 doit permettre une installation simple. Les éléments de configuration liés à cette installation doivent être réduits au minimum. Les valeurs par défaut doivent correspondre à un résultat de fonctionnement le plus implicite pour la majorité des cas.

2.10.1 Première installation

SITools2 comprend un installateur IzPack simplifiant l'installation en demandant à l'utilisateur les configurations nécessaires.

IzPack vérifie l'existence de la base de données, et permet de créer le schéma et les tables sur demande de l'utilisateur dans la procédure d'installation.

Il n'existe pas de fonction automatique qui permette de contrôler que la configuration d'installation est correcte. Les logs de démarrage doivent permettre de lever la plupart des dysfonctionnements dus à une mauvaise configuration.

2.10.2 Installation de mise à jour

SITools2 ne comprend pas de procédure automatique de mise à jour par rapport à une configuration existante, qui permette de garantir la reprise intégrale de tous les paramétrages.

Cependant, un mode de démarrage « sans échec » a été mis en place depuis la version 2.0. Ce mode a été conçu pour répondre aux attentes des laboratoires qui disposent de nombreux datasets et qui naturellement ne souhaitent pas ré-effectuer le travail de configuration à chaque version.

La configuration de SITools2 est stockée sur une arborescence de fichiers sous le répertoire /data.

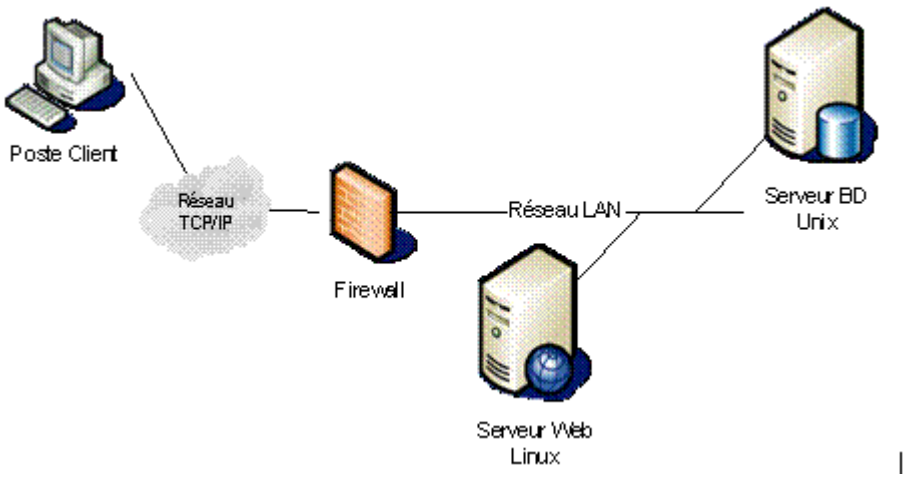
La reprise de ce répertoire dans une nouvelle version est rendue possible sous réserve de compatibilité ascendante des modèles de données. Cette information est fournie par l'équipe de développement dans le README de chaque distribution.

2.10.3 Installation avec migration

SITools2 ne comprend pas à l'heure actuelle de service automatique de migration d'une instance de SITools2 à une autre.

3 ARCHITECTURE PHYSIQUE

3.1 SCHEMA D'ARCHITECTURE MATERIELLE



3.2 DESCRIPTION DE L'ARCHITECTURE PHYSIQUE

3.2.1 Les composants matériels

Serveur Web : destiné à accueillir l'application serveur REST Sitools2.

Serveur BD : destiné à accueillir un SGBD de données métier.

Serveur de fichiers : destiné à accueillir les espaces de stockages utilisateurs exposés par l'application REST Sitools2.

Ces 3 serveurs peuvent être concentrés sur la même machine si elle est proportionnée au nombre d'utilisateurs et à la volumétrie des bases de données.

SITools2 n'a pas été encore testé dans une configuration répliquée sous Apache par exemple pour pallier à des pics de charge.

3.2.2 Les composants logiciels déployés

Serveur Web :

- Arborescence du projet avec
  - cots
    - restlet
    - extjs
  - data
  - workspace (ou autre nom « distribution »)

- client-admin
- client-user
- client-public
- fr.cnes.sitools.core
- fr.cnes.sitools.extensions
- fr.cnes.sitools.ext.test
- libraries
  - net.sf.saxon_9.3
  - org.apache.solr_3.1.0
  - org.apache.xalan_2.7.1
  - org.codehaus.jackson_1.8.0
  - org.gjt.mm.mysql_5.1.7
  - org.xmlpull_3.1
  - com.ice.tar_2.5
  - com.sun.syndication_1.0
  - fr.cnes.sitools.captcha_1.0
  - javax_measure_0.9.5
  - org.postgresql_8.3
- org.restlet.solr
- org.restlet.wadl
- org.restlet.patched
- org.restlet.security

- Serveur Apache (optionnel)

On peut mettre en place un frontal Apache, pour exposer les ressources statiques (cots et workspace/client-*), pour assurer les mécanismes de cache, et de répartition de charge entre plusieurs instances de serveur REST Sitools.

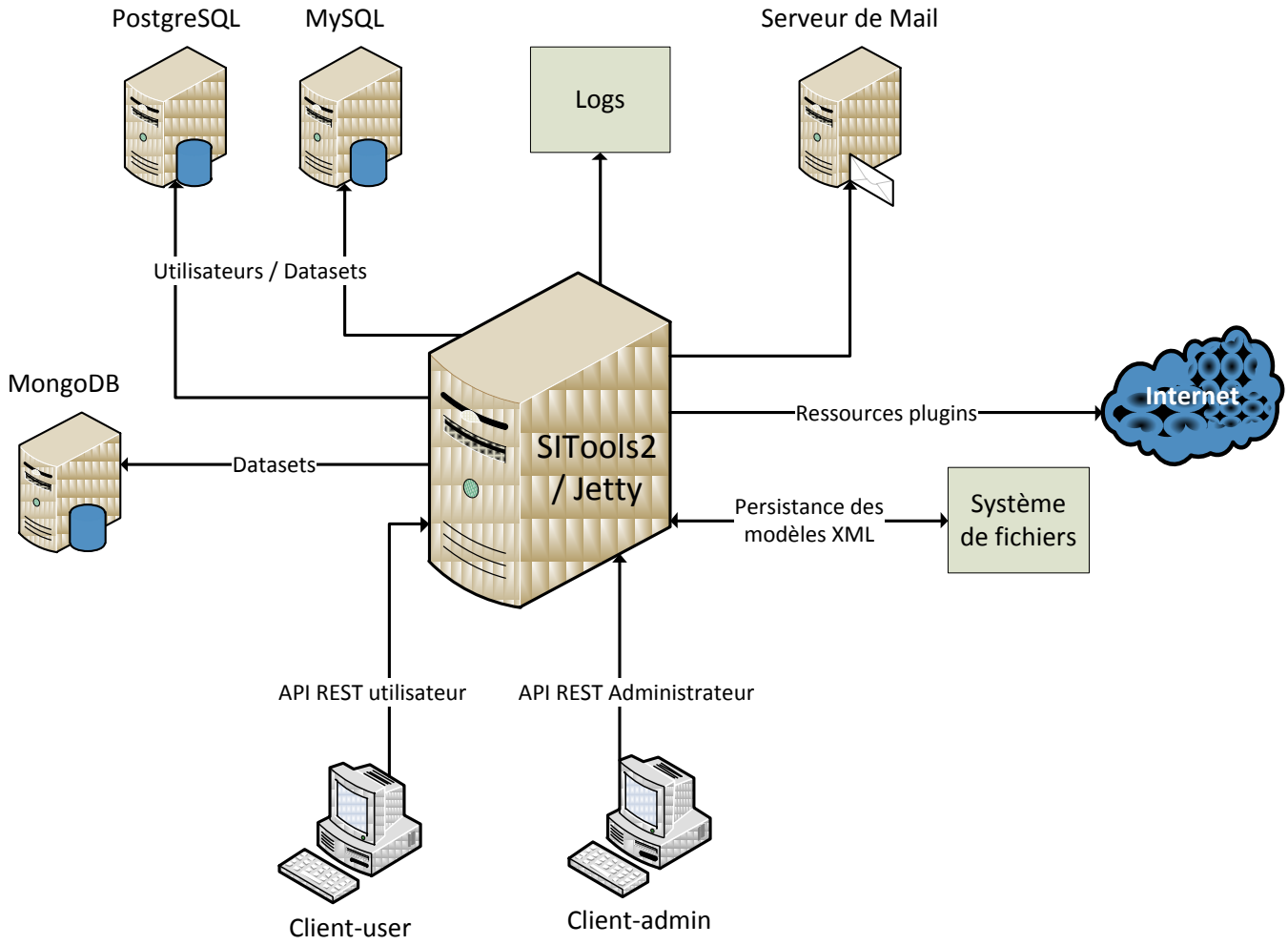
Dans la version actuelle, la configuration du serveur Sitools repose sur un système de stockage de fichiers XML, qui ne supporte pas le partage entre plusieurs instances serveurs. Dans le futur, la mise en place d'une solution alternative pour mutualiser ces données de configuration comme une BD XML partagée, devrait permettre de distribuer Sitools2 sur plusieurs machines, avec par exemple une dédiée pour l'administration et plusieurs pour la consultation.

Serveur BD :

- BD Postgresql
- BD MySQL
- BD NOSQL MongoDB
- Evolutions futures : connecteurs BD Oracle, BD XML, BD Objet, NOSQL CouchDB, BD RDF …

Il existe des spécificités de langage SQL pour assurer la compatibilité des requêtes sur chaque type de base de données. L'objectif est d'offrir la même API générique de recherche et de récupération de données quelle que soit la source. Ce travail de généricité a été réalisé sur la MongoDB avec cependant certaines limitations, mais préservant les avantages de l'utilisation de MongoDB.

3.2.3 Echanges entre SITools2 et les systèmes externes



4 DOCUMENTS APPLICABLES ET DE REFERENCE (A/R)

A/R	Référence	Titre
A	DO2-PRC-10910-SAD	PRC 3.0

5 GLOSSAIRE ET ABREVIATIONS

5.1 GLOSSAIRE

Terme	Définition
Acteur	Ensemble cohérent de rôles qu'une entité, humaine ou non, peut jouer lorsqu'elle interagit avec le système étudié. Un acteur peut jouer des rôles différents en fonction du cas d'utilisation auquel il participe. La notion « d'acteur » ne doit pas être confondue avec l'entité physique à laquelle elle est reliée.
Acteur principal	Un acteur est qualifié de principal lorsqu'il déclenche des évènements qui réalisent le cas d'utilisation.
Acteur secondaire	Un acteur est qualifié de secondaire lorsqu'il ne fait que recevoir des informations suite à la réalisation du cas d'utilisation.
Cas d'utilisation	Il permet de définir le comportement du système. Chaque cas d'utilisation spécifie un scénario, pouvant comporter des variantes, que le système est susceptible de réaliser, en interaction avec ses acteurs.
Package	Groupement logique d'éléments du modèle. Un package peut lui-même contenir d'autres packages. Les packages permettent d'organiser le modèle suivant une structure en arbre.
Scénario	Séquence d'échanges d'évènements entre les acteurs et le système dont la finalité est de réaliser un objectif précis.

5.2 ABREVIATIONS

Abréviation	Nom détaillé
DEBM	Dossier d'Expression des Besoins Métier
DSF	Dossier de Spécifications Fonctionnelles
DST	Dossier de Spécifications Techniques