Élaboration d'un référentiel technique produit

Lionel Mancilla

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE FRANCHE COMTE

MEMOIRE

Présenté en vue d'obtenir

le DIPLOME d'INGENIEUR CNAM

SPECIALITE : INFORMATIQUE

OPTION : informatique, réseaux, systèmes et multimédia (IRSM)

par

Lionel MANCILLA

Elaboration d'un référentiel technique produit

Soutenu le 25 Septembre 2010

JURY

PRESIDENT :

MEMBRES : Kamel Barkaoui
Philippe Descamps
Alain-Jerôme Fougères
Jean Pierre Michaut

Remerciements

Mes premiers remerciements s'adressent à Monsieur Alain-Jérôme Fougères, Enseignant-Chercheur au CNAM de Franche-Comté, pour avoir été mon tuteur lors de la réalisation de mon mémoire, son aide fut précieuse lors de la définition du sujet ainsi que lors de la rédaction de mon mémoire.

Je remercie Jean Pierre MICHAUT et Valérie OUERIAGLI, ingénieur informaticien chez PSA Peugeot Citroën pour m'avoir accueilli au sein de leur service me donnant ainsi la chance de pouvoir réaliser ce mémoire d'ingénieur. J'ai notamment apprécié leur disponibilité lorsque j'ai rencontré des difficultés mais également les moyens matériels et financières mis en place pour le bon déroulement de ce projet.

Je remercie le CNAM, cette grande institution qui donne la chance à tous de suivre des études supérieures parallèlement à la vie active.

Une pensée très particulière va à ma famille pour leur soutien durant toutes ces années où j'ai suivi les cours du soir.

Table des matières

Résumé

Le groupe PSA Peugeot Citroën cherche à réduire les coûts d'infrastructures et d'exploitations de son architecture informatique. Il lui faut donc optimiser l'utilisation des ressources serveurs. Cette optimisation passe par la consolidation et la virtualisation des serveurs rendant possible le déplacement à chaud d'applications informatiques d'un serveur virtuel vers un autre.

La multiplication des déplacements d'applications demande aux exploitants de l'informatique PSA d'être de plus en plus souples. De ce besoin est née la création d'un référentiel technique produit capable de cartographier de façon dynamique la paramétrie de tous les produits d'infrastructures installés sur les quelques 4200 serveurs du parc informatique PSA.

Ce mémoire va illustrer les différentes étapes nécessaires lors de la réalisation d'un tel projet. Nous commencerons par l'expression des besoins auprès des utilisateurs puis j'illustrerais les choix d'architectures logiciels et applicatives mise en place pour ce référentiel. Par la suite je montrerais les procédures mise en place pour remonter de façon dynamique les configurations des produits installés sur les serveurs du groupe PSA et enfin nous verrons quelle interface utilisateur a été mise en place pour mettre en forme toutes ces données sur un site intranet.

Mot clé : Produit, infrastructure, référentiel, configuration, paramètre, intranet

Summary

Group PSA Peugeot Citroen seeks to reduce the costs of infrastructures and exploitations of its data-processing architecture. It thus should optimize the use as of its resources waiters. This optimization passes by the consolidation and the virtualisation of the waiters making possible the hot displacement of computer applications of a waiter virtual towards another.

The multiplication of displacements of implementations requires of the owners data processing PSA to be increasingly flexible. From this need from a technical reference frame produces able to chart in a dynamic way the parametry of all the products of infrastructures was born creation installed on the few 4200 waiters of information technology infrastructure PSA.

This memory will illustrate the various stages necessary at the time of the realization of such a project-base. We will start with the expression of the needs near the users then I would illustrate the choices of architectures software and application installation for this reference frame. Thereafter I would show the procedures installation for increase of way dynamic the configurations of the products installed on the waiters of group PSA and finally we will see which user interface was installation to format all these data on a site Intranet.

Keyword: Product, infrastructure, reference frame, configuration, parameter, Intranet

INTRODUCTION

En informatique, un **logiciel** ou un **produit** est un ensemble d'informations relatives à des traitements effectués automatiquement par un appareil informatique. Y sont incluses les instructions de traitement, regroupées sous forme de programmes, des données et de la documentation. Le tout est stocké sous forme d'un ensemble de fichiers dans une mémoire. Dans la suite de ce document nous parlerons fréquemment de produit informatique, vous vous demandez certainement quelle différence y a-t-il entre un produit et un logiciel ? En réalité c'est très simple, nous parlerons de logiciel pour un ordinateur (PC, MAC,…) et de produit pour les serveurs informatiques.

Chaque application informatique a besoin pour son bon fonctionnement de nombreux produits informatiques, ce sont plus de 1000 produits qui sont industrialisés chez PSA actuellement. Nous pouvons ainsi retrouver des produits très variés selon les besoins des applications. Prenons l'exemple d'une application web, celle-ci aura besoin d'être hébergée sur un serveur d'application. Nos amis d'IBM vendent un produit très robuste à PSA, celui-ci se nomme Websphere et héberge de nombreuses instances applicatives capables de supporter de nombreuses connexions simultanées.

Suite au contexte économique difficile dans lequel nous nous trouvons actuellement, le groupe PSA Peugeot Citroën cherche à réduire les coûts d'infrastructures et d'exploitations de son architecture informatique. Il lui faut donc optimiser l'utilisation des ressources serveurs. Cette optimisation passe par la consolidation et la virtualisation des serveurs rendant possible la flexibilité des applications. Ici le terme flexibilité décrit le processus qui permet le déplacement d'une application d'un serveur virtuel vers un autre. Cette flexibilité ne se limite pas au déplacement de l'application mais également aux déplacements des différents produits qui dépendent de celle-ci.

Avec la banalisation du processus de flexibilité, il devient de plus en plus difficile d'avoir une cartographie complète des produits d'infrastructures hébergés sur les quelques 4200 serveurs physiques du parc informatique du groupe PSA. De ce besoin est née la création d'un référentiel technique produit permettant de collecter de façon dynamique toutes les configurations des produits présents sur les différents serveurs du parc informatique.

Ce mémoire va ainsi illustrer la création d'un tout nouveau référentiel. Nous allons donc retrouver toutes les étapes d'une démarche projet avec pour débuter la définition d'un cahier des

charges basé sur l'analyse de l'existant. Par la suite je vous expliquerais pourquoi j'ai utilisé la filière LAMP pour développer ce référentiel. Nous parlerons également du processus mise en place pour collecter les configurations des produits informatiques sur tous les serveurs du parc informatique PSA.

PRESENTATION DU PROJET

Environnement de travail

L’entreprise PSA Peugeot Citroën

Le groupe PSA Peugeot Citroën est une holding réunissant un grand nombre de sociétés industrielles, commerciales, financières et de services. L'ensemble des activités du groupe converge vers l'automobile. L'importance de cette activité au sein du groupe, se traduit par quelques chiffres, puisque l'activité automobile représente 96 % du chiffre d'affaires, 90 % de ses investissements et 89 % de ses effectifs.

Le groupe PSA construit son développement sur deux marques généralistes fortes et différenciées, au rayonnement mondial, dans le cadre de stratégies internationales coordonnées. Celui-ci organise une gamme complète pour chacune des deux marques, assurant la complémentarité des styles et des concepts, et l’alternance des lancements pour une offre toujours renouvelée. Dans un souci d’efficacité et de recherche d’économies d’échelle, tout l’appareil technique, industriel, administratif et financier du groupe automobile est unifié. Néanmoins, chaque marque possède sa propre identité et dispose de l’autonomie nécessaire à la conduite d'une politique propre.

PSA Peugeot Citroën est un groupe qui poursuit une stratégie de croissance. Présent dans 140 pays, il continue d’accélérer son développement à l'international, et en particulier dans trois zones prioritaires dans lesquelles le marché automobile est en croissance : l’Europe centrale et orientale, l’Amérique du Sud et la Chine.

Leader de la voiture à basses émissions de CO2, PSA poursuit ses avancées dans ce domaine. En tant que constructeur généraliste, il s’est engagé dans une démarche pionnière de réduction des émissions de CO2 et de polluants sur l’ensemble de sa gamme. Les recherches portent notamment sur les motorisations, dont les techniques d’hybridation. Le groupe a ainsi décidé de généraliser le système « Stop and Start », et souhaite aussi étendre plus largement des solutions comme le filtre à particules, une technologie dans laquelle le groupe PSA est déjà le leader. Enfin, les véhicules du groupe acceptent une proportion de biocarburants allant jusqu’à 10 % avec l’essence et 30 % avec le gazole.

En Europe, le groupe commercialise deux véhicules flexfuel depuis 2007 : la Peugeot 307 SW et la Citroën C4, tandis qu'au Brésil, cette motorisation concerne déjà une large majorité des véhicules vendus.

L'activité du groupe PSA Peugeot Citroën peut se résumer en quelques chiffres :

- 207 800 hommes et femmes à travers le monde avec plus de 200 métiers différents
- 60,6 milliards d'euros de chiffre d'affaires
    - 3,43 millions de voitures particulières et de véhicules utilitaires vendus dans le monde, dont plus d'un tiers en dehors de l'Europe
- Le 2ème constructeur européen, avec une part de marché de 18,8%
- Plus de 140 marchés d'exportation dans le monde et 20 000 points de vente

Le site de Bessoncourt

Le site de Bessoncourt est situé à mi chemin entre les sites de production de Sochaux et de Mulhouse. Il est le principal centre d'hébergement et de traitement de services informatiques du groupe PSA. Celui-ci s'étend sur un terrain de 5,7 Ha et développe une surface de 12000 m² répartie d'une manière quasi équivalente entre des bureaux, des salles machines et des locaux techniques. Avec l'importance croissante des services informatiques, le site a déjà vécu 3 extensions depuis son ouverture en mai 1980, et compte aujourd'hui plus de 300 membres du personnel. Chaque jour, le site de Bessoncourt consomme l'équivalent de l'électricité d'une ville de 7000 habitants.

Figure 1 : Vue aérienne du site de Bessoncourt

La direction des systèmes d'informations

Le site de Bessoncourt dépend de la Direction des Systèmes d'INformation (DSIN). C'est cette entité administrative qui a en charge toute la gestion du parc informatique du groupe PSA Peugeot Citroën, soit environ 80000 postes de travail et 4200 serveurs physiques. Ses missions sont de garantir la cohérence des architectures fonctionnelles et techniques, d'assurer la meilleure adéquation entre les solutions techniques et les besoins du groupe, et de rechercher le meilleur ratio performance coût pour la gestion du système d'information.

La DSIN est elle même divisée en différentes entités caractéristiques des activités du groupe :

- **AIQP** : Audit Interne, Qualité et Performance
- **SIPP** : Système d'Information Produit Process, conception véhicule
- **SIFA** : Système d'Information Fabrication Logistique
- **SIDM** : Système d'Information Commerce, Marketing et Distribution
- **SISF** : Système d'Information Services de Financement, PSA banque
- **SDGA** : Système d'Information Direction Générale des Achats, RH et autres
- **CSMC** : Centre de Support, Méthode et Compétence
- **PGSB** : Planification, Gestion, Synthèse et Benchmarking
- **PDA** : Plateau de Développement Argentine
- **INSI** : INfrastructure des Systèmes d'Information, exploitation informatique

Présentation du service d'accueil

L'INSI a pour mission de fournir les moyens nécessaires à la mise en production de services répondant aux besoins exprimés par les autres entités de la DSIN. Ainsi sont regroupées différentes fonctions de supports aux utilisateurs, les fonctions d'industrialisations de produits (messagerie, services intranet, gestion de base de données, téléphonie, etc...) et les fonctions de gestion et de supervision de l'ensemble du parc informatique.

Au sein de l'INSI, une autre entité a en charge d'expertiser l'ensemble des moyens techniques et logiciels mis en œuvre afin d'assurer une certaine cohérence de l'ensemble du système d'information : l'ETSO « Expertise Technique et Support Opérationnel ». C'est dans cette entité que sont pensées les architectures logiques des logiciels déployés au sein du groupe. Ainsi, plus en amont, les personnes du service de l'ETSO sont réparties selon leurs domaines d'expertise afin de couvrir l'ensemble des besoins spécifiques inhérents à l'exploitation des services informatiques utilisés.

Pour résumer, lorsqu'une politique de déploiement d'un service a été décidée, dans un premier

temps, le produit et/ou logiciel, est passé en revue par toute une série d'experts et d'intégrateurs pour valider ses fonctionnalités. Puis, suite à cette phase de test, le produit est transmis à une équipe spécialisée dans la sécurité applicative pour une seconde validation. Ensuite, une autre équipe prend en charge la mise en production réelle du service pour le mettre à disposition des utilisateurs du groupe PSA. Enfin, d'autres équipes seront en charge de la supervision du fonctionnement de l'ensemble, et une dernière du support utilisateur.

Le service dans lequel l'étude a été menée et dans lequel je suis arrivé en Juillet 2009, s'occupe de l'exploitation des produits d'infrastructures de l'informatique PSA. Environ 40 personnes y travaillent. Chaque personne est responsable de plusieurs produits regroupés par domaine (Produits pour les serveurs d'applications, produits pour l'éditiques, produits concernant les transferts de fichiers …). La prise en charge de ces produits comprend tous les aspects techniques pour une mise à disposition aux BU (Business Unit) ayant besoin d'utiliser le produit (industrialisation, déploiement, ...), la maintenance de ces produits (mise à niveau, support, …) et la relation avec le fournisseur dans le cas d'un produit bénéficiant d'un contrat de maintenance avec le support fournisseur. Les responsables produits doivent également mettre leur expertise technique au service des projets applicatifs et d'infrastructures pour choisir et mettre en place l'architecture technique la plus adaptée, tant d'un point de vue fonctionnel qu'économique.

Environ 1000 produits distincts sont gérés par les différents responsables produits, ceux-ci étant installés sur les 4200 Serveurs du parc. C'est donc dans ce service nommé ETSO/PROD que la réalisation du projet Rachel a été effectuée.

Contexte du projet

Dans le contexte économique difficile, le groupe PSA Peugeot Citroën cherche à réduire ses coûts d'infrastructure et d'exploitation de son architecture informatique.

Il lui faut donc optimiser l'utilisation des ses ressources serveurs. Cette optimisation passe par la consolidation et la virtualisation des serveurs rendant possible la flexibilité des applications c'est-à-dire le déplacement d'une application d'un serveur vers un autre. Dans cette optique un projet d'envergure a été lancé au sein de la DSIN. Il s'agit du projet INFRA 2.0 qui vise à rendre l'infrastructure plus agile de façon à allouer les ressources plus rapidement, plus justement et au meilleur coût.

Ce projet fédérateur chez PSA pour les trois années à venir se compose de quatre grandes lignes

directrices complémentaires :

- ❖Développer des fermes d'infrastructures
- ❖Développer des fermes d'applications
- ❖Orchestrer les processus (automatiser)
- ❖Optimiser les moyens.

Pour des raisons de confidentialités, je ne rentrerai pas dans les détails du projet INFRA 2.0.

Objectif du projet

Avec le lancement du projet INFRA 2.0 qui va demander au service ETSO d'être de plus en plus souple, le besoin d'un référentiel unique au sein de ce service transversal se fait de plus en plus nécessaire afin d'avoir une visibilité en temps réels sur la paramètrie des produits d'infrastructure du parc informatique PSA. En effet à l'heure actuelle, les responsables produits gèrent dans leur coin, leurs propres référentiels techniques. Ceux-ci sont de toutes sortes, cela va du simple document Word stocké sur son ordinateur personnel à des documents plus complexe de type Excel avec mise à jour de façon dynamique via des macros. Avec la multiplication des déplacements des applications d'une zone virtuelle à une autre, il est devenu de plus en plus difficile pour les responsables produits de garder à jour de façon manuelle son propre référentiel technique produit. Ces référentiels isolés sont devenus peu fiables par leur contenu et la charge de travail pour les maintenir à jour est devenue trop importante.

C'est donc de ce besoin qu'est né en Juin 2009 le projet RACHEL, un référentiel technique produit unique qui permettra à chaque responsable produit de remonter les configurations des produits dont ils ont la charge sur un référentiel ergonomique et de façon dynamique. Le nom du projet « RACHEL » correspond en réalité à l'acronyme **R**emontée **A**utomatique de **C**onfiguration **H**étérogène et des **E**nvironnements **L**ogiciel.

Il est possible qu'à ce moment du mémoire, vous vous demandiez à quoi correspond une configuration d'un produit d'infrastructure ?

Il s'agit en réalité de l'ensemble des paramètres de configuration que le responsable choisit de remonter au référentiel RACHEL. Les paramètres de configuration permettent d'adapter le comportement d'un produit à une configuration matérielle, logicielle et réseau du système informatique dans lequel le logiciel est implanté. Nous verrons plus tard dans le mémoire quels processus ont été mis en place pour remonter les différents paramètres d'une configuration produit au référentiel RACHEL.

Ce nouveau référentiel sera ainsi utilisé par toutes les personnes responsables de produits d'infrastructure, que ce soit des produits systèmes, de base de données ou autres. Ce sont au total environ 100 personnes qui sont ainsi concernées par l'utilisation de ce nouveau référentiel.

Qu'est ce qu'un référentiel ?

Un référentiel est un ensemble de bases de données contenant les "références" d'un système d'informations. Ces références sont de deux types. Soit d'information dont les applications ont besoin pour fonctionner, mais qui, étant parfois mises à jour, sont stockées dans une base de données spéciale, les "données de référence", où les utilisateurs ou applications peuvent les retrouver chaque fois qu'ils en ont besoin, c'est le cas par exemple pour les annuaires (de l'organisation, des personnes, des équipements etc…). Soit d'information qui seront utilisées lorsque nous devons faire évoluer une application : nous parlons alors d'administration des données. Nous comprenons alors que le référentiel est la colonne vertébrale d'un système d'informations. Les règles auxquelles obéissent sa construction et sa gestion sont purement logiques, donc finalement assez simples.

Comment définir le niveau de détail souhaitable pour un référentiel ? Toute nomenclature est une suite de partitions emboîtées définies sur un domaine conceptuel, nous pouvons toujours la détailler en ajoutant un niveau mais il n'existe aucun critère formel permettant de définir le degré de détail auquel il convient de s'arrêter.

En pratique, nous nous arrêtons lorsque nous sommes lassés de formaliser les choses. Le niveau de détail utile n'apparaît que plus tard quand nous utilisons un référentiel. Ce qui est trop détaillé reste inutilisé, et les endroits où le détail manque, font l'objet de réclamations de la part des utilisateurs. Il paraît donc difficile de pouvoir anticiper toutes les réactions du public visé par un référentiel.

Revenons au point de départ de la démarche, tentons de nous la représenter de façon simple. Ce que nous cherchons à faire lorsque nous nous lançons dans la réalisation d'un référentiel, c'est mettre de l'ordre, introduire de la clarté, de la cohérence dans le système d'information. Nous sommes comme un étudiant qui range sa chambre : il fait le lit, met les pull-overs dans des tiroirs, les livres sur des étagères, les feuilles de cours dans des classeurs, etc. Quand l'ordre lui semble suffisant, il s'arrête. Un ordre suffisant, ce n'est pas l'ordre parfait. Il se peut que l'étudiant n'ait pas classé les livres sur l'étagère par nom d'auteur, ou par matière, et que son lit ne soit pas exactement "au carré". Qu'importe, si l'ordre même imparfait lui procure cette clarté dans les idées qui l'aide à définir les choses à faire en priorité.

Pour revenir plus spécifiquement au projet Rachel, voici la définition d'un référentiel

technique produit :

❖ Un **référentiel technique produit** est un site intranet collectant les différents paramètres de configuration des produits d'infrastructures installés sur les différents serveurs informatiques du groupe PSA

Rôle du candidat et sa méthode

J'avais pour mission de jouer le rôle de CPI (Chef de Projet Informatique). J'avais ainsi pour fonction de contrôler le bon déroulement du développement du référentiel RACHEL. En effet dans le cadre de projet informatique, il est nécessaire qu'une personne attitrée organise le déroulement du projet et sache inciter et motiver l'équipe pour qu'elle adopte des comportements qui permettront de mener à bien les tâches liées au développement du projet dans les délais (conception, développement, débogage, test…).

Afin de répondre correctement aux besoins de mes utilisateurs, il m'a fallut entretenir une relation quotidienne avec eux pour bien comprendre le besoin réel d'un tel projet.

Le développement du projet RACHEL a suivi la démarche projet dite de cycle en V. Cette méthode de développement est la plus classique et la plus connue mais c'est surtout la démarche projet reconnue au sein de PSA. Elle comprend, avant le développement, des phases d'études comprenant l'expression des besoins, puis les spécifications générales, et la conception détaillée.

Cette méthode est particulièrement bien adaptée aux projets pour lesquels les spécifications sont parfaitement connues et stables. Elle permet d'avoir une vue assez réaliste du planning et du coût du projet avant son développement, puisque toutes les fonctionnalités auront été décrites et spécifiées avec précision. Aucune modification ne peut être apportée au périmètre du projet une fois le projet lancé.

La méthode repose en tout cas dans une relation « client / fournisseur » assez prononcée, puisque la maîtrise d'ouvrage a une position claire de « cliente » responsable du seul périmètre du besoin, face à une maîtrise d'œuvre responsable du seul périmètre de la réalisation.

Il reste cependant possible d'intégrer au sein de cette démarche quelques comportements « agiles » qui permettront aux différentes phases du projet, d'optimiser la charge ici ou là.

Cette démarche projet s'est parfaitement adaptée au projet RACHEL pour lequel les

spécifications ont pu être parfaitement connues et stables grâce à la relation de proximité et quotidienne que j'ai pu entretenir avec les utilisateurs.

Grâce à cette démarche projet, j'ai pu définir un planning détaillé des différentes étapes du projet RACHEL. Nous allons y retrouver les deux phases importantes d'une démarche de cycle en V avec l'étude préalable puis le projet de réalisation. Ce planning a ainsi permis de clarifier les différents jalons à respecter tout au long du projet ainsi que chiffrer clairement le coût total d'un tel projet. Pour des raisons de confidentialité je ne donnerais pas le chiffrage complet qui a été fait pour le développement du projet RACHEL mais je vous propose de consulter le planning détaillé du projet qui a débuté le 15 Septembre 2009 et s'achèvera le 31 Décembre 2010

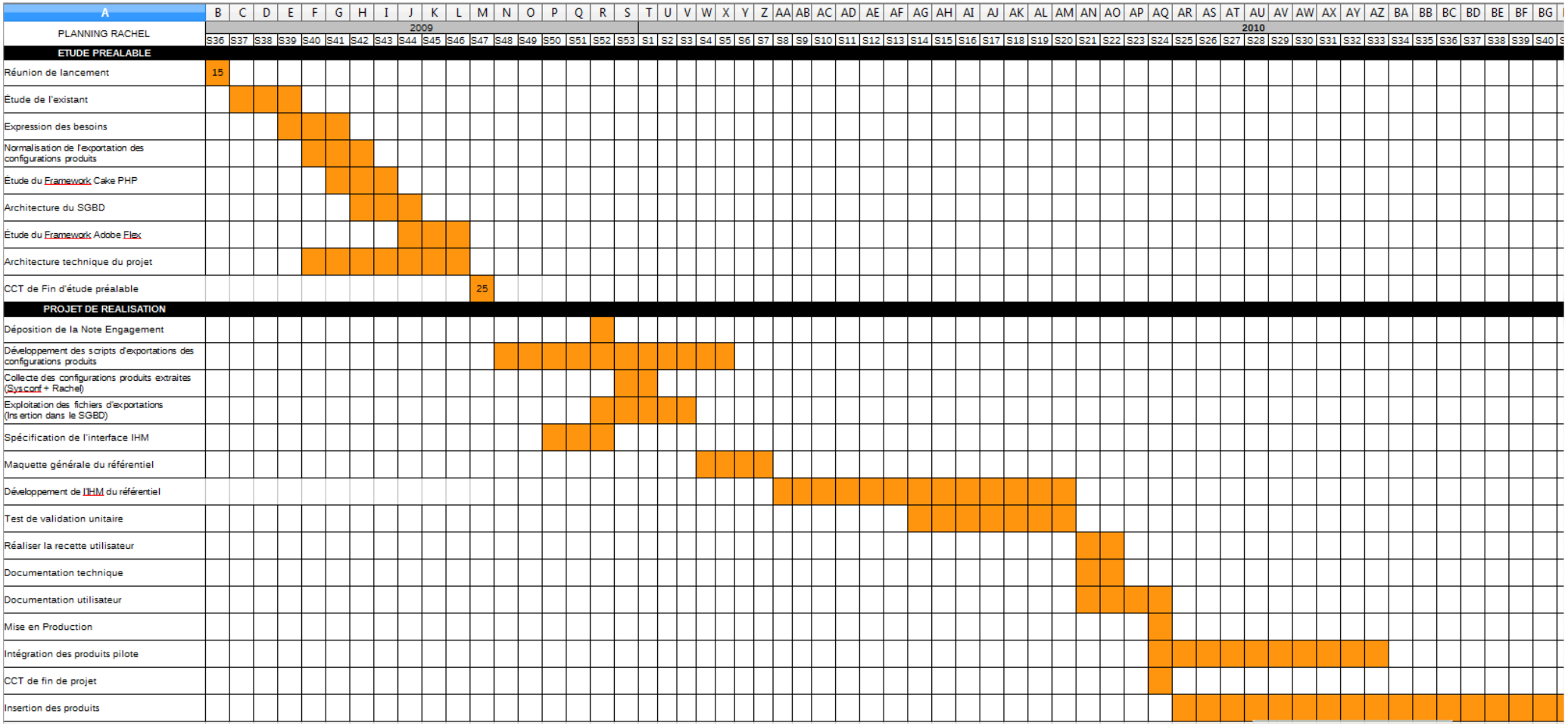


Figure 2 : Planning du projet RACHEL

Dans ce planning nous retrouvons les deux étapes importantes de la vie d'un projet informatique chez PSA :

1. **L'étude préalable**, dans laquelle nous allons clairement définir l'expression des besoins, puis les spécifications générales et la conception détaillée. Au terme de l'étude préalable un CCT (comité centrale technique) sera organisé afin de valider l'étude préalable du projet. Un CCT a pour objectif de réunir tous les intervenants techniques du projet. Durant ce CCT le Chef de projet Informatique présentera le mode de fonctionnement du projet, l'architecture technique ciblé, le planning et le budget de celui-ci. Si le CCT est validé, le CPI pourra débuter l'étape de réalisation du projet informatique.

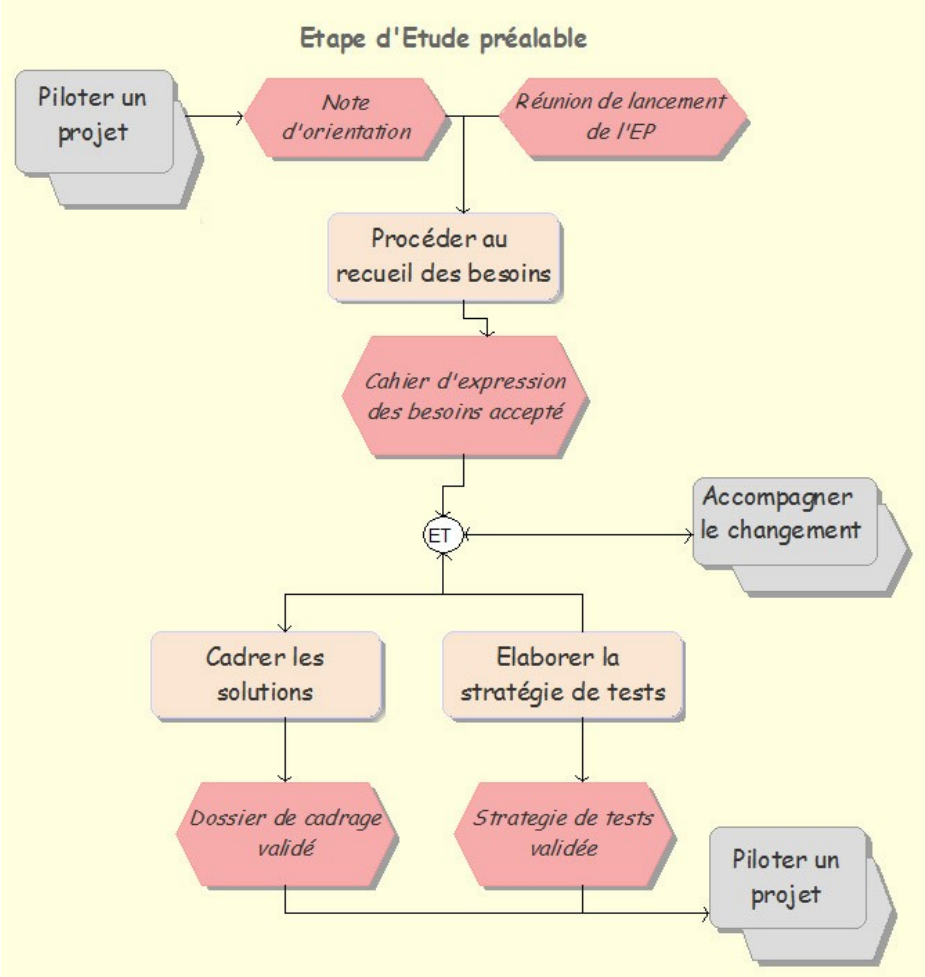


Figure 3 : Les différentes étapes de l'étude préalable

2. **L’étape de réalisation,** correspond quand à elle au développement du projet. Cela passe de la conception générale à la conception détaillée de celui-ci. Puis des tests unitaires seront réalisés sur chaque fonctionnalité du projet avant d’être qualifiée puis mise en production avec montée en charge de l’application.

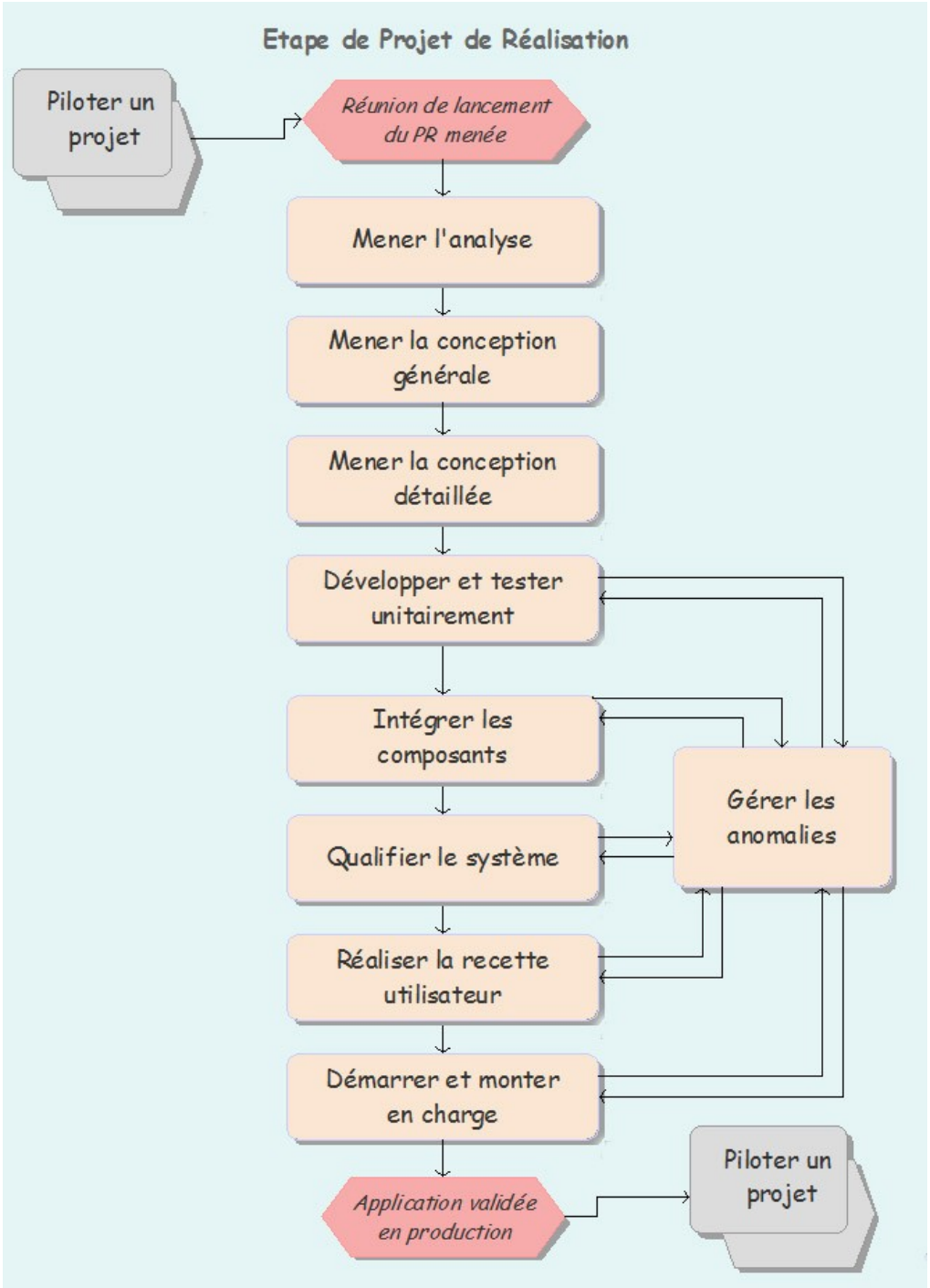


Figure 4 : Les différentes étapes du projet de réalisation

Concernant le développement de l'interface IHM du référentiel, un budget a été demandé pour faire intervenir une prestation extérieure. J'ai donc ainsi obtenu un budget pour acheter une prestation de 90 jours. La mission affectée à la prestation ne concernait que le développement de l'interface IHM, qui nous le verrons en détails ultérieurement a utilisé la technologie Flex. Un cahier des spécifications de l'interface IHM a ainsi été élaboré pour permettre à cette prestation d'être chiffrée et ainsi pouvoir ciblée le profil du prestataire à recruter.

J'ai donc consulté plusieurs prestataires proposant un profil similaire, en effet le profil demandé concernait essentiellement les compétences suivantes :

- Développement d'application web en PHP mais également en flash.
- Maitrise des bases de données MySQL
- Travaux sur l'environnement LAMP

La prestation a ainsi été actée en Mars 2010 pour une durée de 4 mois, du 15 mars 2010 au 15 Juillet 2010. A l'heure où je présente ce mémoire, le projet poursuit toujours son évolution et ce jusqu'au 15 juillet date à laquelle le prestataire aura terminé le développement de l'interface IHM, elle devra me former sur la technologie Flex utilisée lors du développement de l'interface IHM et fournir une documentation technique de l'architecture de l'interface IHM.

Tous les autres développements ont été réalisés par moi-même, nous les verrons plus en détails par la suite.

L'étude préalable

Analyse de l'existant

Avant de définir les besoins fonctionnels du projet, il était important de consulter les différents responsables produits possédant déjà un référentiel produit afin d'analyser avec eux le contenu de leur référentiel pour pouvoir les insérer au mieux dans le nouveau référentiel. Je vais vous présenter deux des référentiels analysés lors cette phase du projet, il s'agit du référentiel pour les produits SAP et celui pour le produit Websphere.

Le référentiel SAP

La grande famille des produits SAP installés sur différents serveurs qui communiquent entre eux, possède déjà un référentiel stockant de nombreuses informations. Toute l'information stockée dans ce référentiel ne concernait pas uniquement la paramètrie des produits SAP. Il a fallut faire du tri à l'intérieur de celui-ci afin d'extraire la paramètrie produit à intégrer dans le nouveau référentiel. L'analyse a permis de mettre en évidence la cible des paramètres produits à intégrer dans le référentiel. En effet les paramètres qui seront remontés au référentiel RACHEL ne concerneront que des paramètres consultables directement sur le serveur où est installé le produit.

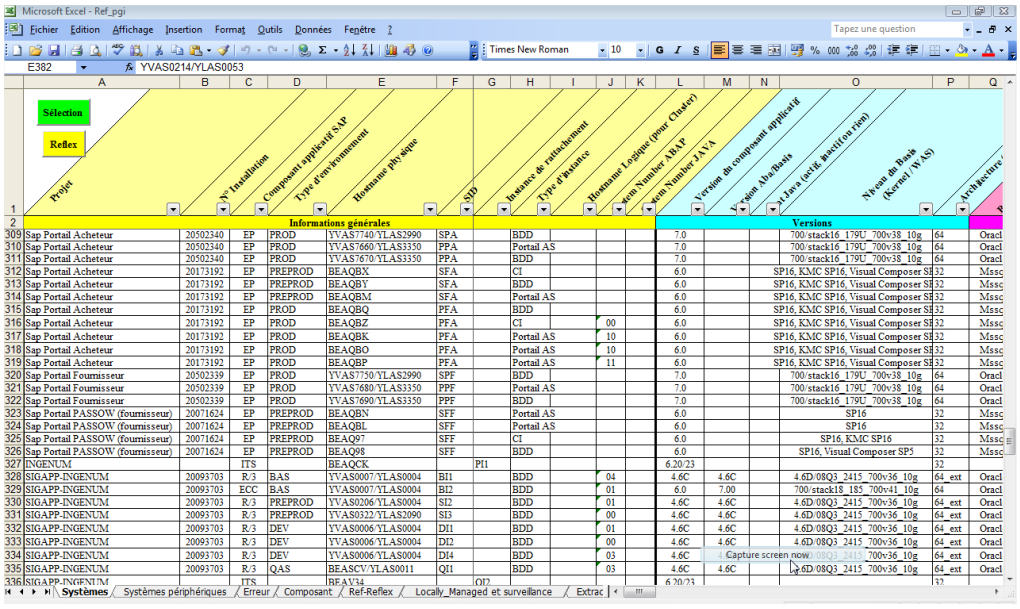
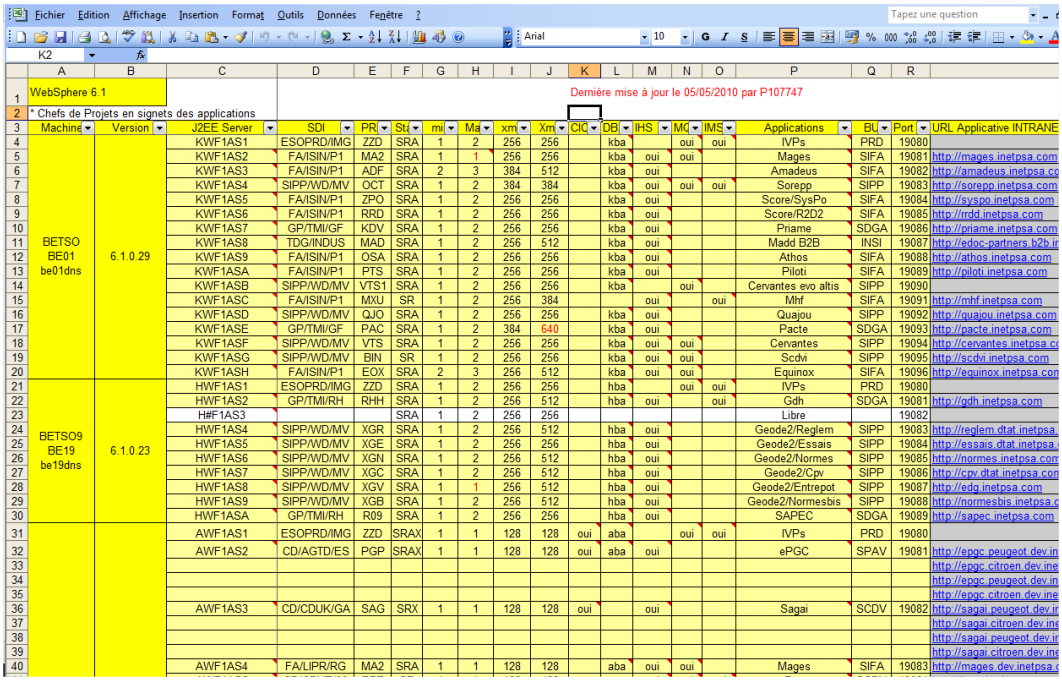


Figure 5 : Le référentiel SAP

Le référentiel Websphere

Le produit Websphere fait parti des produits les plus déployés chez PSA. Le serveur d'application WAS (Websphere Application Server) utilise des standards ouverts tels que Java EE, XML, et les Web Services. Il fonctionne avec de nombreux serveurs Web comme Apache HTTP Server, Netscape Enterprise Server, Microsoft Internet Information Services (IIS), IBM HTTP Server pour i5/OS, IBM HTTP Server pour z/OS, et IBM HTTP Server pour Z/OS/AIX/Linux/Microsoft Windows/Solaris.

Son grand déploiement chez PSA a obligé ses responsables produits à créer un référentiel sous forme de fichier Excel. L'extraction de quelques paramètres importants était faite de façon manuelle par ses responsables produits. Le suivi en temps réel de la paramétrie est devenu indispensable sur un parc où de nombreuses occurrences du produit sont installées.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	WebSphere 6.1											Dernière mise à jour le 05/05/2010 par P107747						
2	* Chefs de Projets en signets des applications																	
3	Machine	Version	J2EE Server	SDI	PR	St	mi	Ma	xm	Xm	CIC	DB	IHS	MQ	IMS	Applications	BU	Port	URL Applicative INTRANE
4			KWF1AS1	ESOPRD/IMG	ZZD	SRA	1	2	256	256		kba		oui	oui	IVPs	PRD	19080	
5			KWF1AS2	FA/ISIN/P1	MA2	SRA	1	1	256	256		kba	oui	oui		Mages	SIFA	19081	http://mages.inetpsa.com
6			KWF1AS3	FA/ISIN/P1	ADF	SRA	2	3	384	512		kba	oui			Amadeus	SIFA	19082	http://amadeus.inetpsa.co
7			KWF1AS4	SIPP/WD/MV	OCT	SRA	1	2	384	384		kba	oui	oui	oui	Sorepp	SIPP	19083	http://sorepp.inetpsa.com
8			KWF1AS5	FA/ISIN/P1	ZPO	SRA	1	2	256	256		kba	oui			Score/SysPo	SIFA	19084	http://syspo.inetpsa.com
9			KWF1AS6	FA/ISIN/P1	RRD	SRA	1	2	256	256		kba	oui			Score/R2D2	SIFA	19085	http://rrdd.inetpsa.com
10			KWF1AS7	GP/TMI/GF	KDV	SRA	1	2	256	256		kba	oui			Priame	SDGA	19086	http://priame.inetpsa.com
11	BETSO		KWF1AS8	TDG/INDUS	MAD	SRA	1	2	256	512		kba	oui			Madd B2B	INSI	19087	http://edoc-partners.b2b.ir
12	BE01	6.1.0.29	KWF1AS9	FA/ISIN/P1	OSA	SRA	1	2	256	256		kba	oui			Athos	SIFA	19088	http://athos.inetpsa.com
13	be01dns		KWF1ASA	FA/ISIN/P1	PTS	SRA	1	2	256	256		kba	oui			Piloti	SIFA	19089	http://piloti.inetpsa.com
14			KWF1ASB	SIPP/WD/MV	VTS1	SRA	1	2	256	256		kba		oui		Cervantes evo altis	SIPP	19090	
15			KWF1ASC	FA/ISIN/P1	MXU	SR	1	2	256	384			oui		oui	Mhf	SIFA	19091	http://mhf.inetpsa.com
16			KWF1ASD	SIPP/WD/MV	QJO	SRA	1	2	256	256		kba	oui			Quajou	SIPP	19092	http://quajou.inetpsa.com
17			KWF1ASE	GP/TMI/GF	PAC	SRA	1	2	384	640		kba	oui			Pacte	SDGA	19093	http://pacte.inetpsa.com
18			KWF1ASF	SIPP/WD/MV	VTS	SRA	1	2	256	256		kba	oui	oui		Cervantes	SIPP	19094	http://cervantes.inetpsa.co
19			KWF1ASG	SIPP/WD/MV	BIN	SR	1	2	256	256		kba	oui	oui		Scdvi	SIPP	19095	http://scdvi.inetpsa.com
20			KWF1ASH	FA/ISIN/P1	EOX	SRA	2	3	256	512		kba	oui	oui		Equinox	SIFA	19096	http://equinox.inetpsa.con
21			HWF1AS1	ESOPRD/IMG	ZZD	SRA	1	2	256	256		hba		oui	oui	IVPs	PRD	19080	
22			HWF1AS2	GP/TMI/RH	RHH	SRA	1	2	256	512		hba	oui		oui	Gdh	SDGA	19081	http://gdh.inetpsa.com
23			H#F1AS3			SRA	1	2	256	256						Libre		19082	
24			HWF1AS4	SIPP/WD/MV	XGR	SRA	1	2	256	512		hba	oui			Geode2/Reglem	SIPP	19083	http://reglem.dtat.inetpsa.
25			HWF1AS5	SIPP/WD/MV	XGE	SRA	1	2	256	256		hba	oui			Geode2/Essais	SIPP	19084	http://essais.dtat.inetpsa.
26	BETSO9		HWF1AS6	SIPP/WD/MV	XGN	SRA	1	2	256	512		hba	oui			Geode2/Normes	SIPP	19085	http://normes.inetpsa.com
27	BE19	6.1.0.23	HWF1AS7	SIPP/WD/MV	XGC	SRA	1	2	256	512		hba	oui			Geode2/Cpv	SIPP	19086	http://cpv.dtat.inetpsa.com
28	be19dns		HWF1AS8	SIPP/WD/MV	XGV	SRA	1	1	256	512		hba	oui			Geode2/Entrepot	SIPP	19087	http://edg.inetpsa.com
29			HWF1AS9	SIPP/WD/MV	XGB	SRA	1	2	256	512		hba	oui			Geode2/Normesbis	SIPP	19088	http://normesbis.inetpsa.c
30			HWF1ASA	GP/TMI/RH	R09	SRA	1	2	256	256		hba	oui			SAPEC	SDGA	19089	http://sapec.inetpsa.com
31			AWF1AS1	ESOPRD/IMG	ZZD	SRAX	1	1	128	128	oui	aba		oui	oui	IVPs	PRD	19080	
32			AWF1AS2	CD/AGTD/ES	PGP	SRAX	1	1	128	128	oui	aba	oui			ePGC	SPAV	19081	http://epgc.peugeot.dev.in
33																			http://epgc.citroen.dev.ine
34																			http://epgc.peugeot.dev.in
35																			http://epgc.citroen.dev.ine
36			AWF1AS3	CD/CDUK/GA	SAG	SRX	1	1	128	128	oui		oui			Sagai	SCDV	19082	http://sagai.peugeot.dev.ir
37																			http://sagai.citroen.dev.ins
38																			http://sagai.peugeot.dev.ir
39																			http://sagai.citroen.dev.ins
40			AWF1AS4	FA/LIPR/RG	MA2	SRA	1	1	128	128		aba	oui	oui		Mages	SIFA	19083	http://mages.dev.inetpsa.c

Figure 6 : Le référentiel Websphere

Expression des besoins

Le champ d'action du projet Rachel étant trop grand, un groupe réduit des produits les plus utilisés sur l'infrastructure informatique PSA a été défini afin de travailler plus

sereinement. Au total ce sont huit produits qui ont été désignés « produits pilotes » dans ce projet. En effet c'est auprès des responsables de ces huit produits que l'expression des besoins va être concentrée. A ces produits pilotes s'ajoute également le chef de projet du projet INFRA 2.0 à l' origine de la création de ce nouveau référentiel. Cette personne porte la casquette de Chef de Projet Utilisateur. Trois autres personnes au sein du service ETSO ont étés désignées pour rejoindre un comité technique de suivi de projet. En effet chaque mois avait lieu une réunion de suivi de l'avancement du projet et c'est avec ce comité technique que seront validées les décisions d'orientation du projet. (Expression des besoins, architecture technique, architecture matériel, Interface IHM)

Tableau I : Liste des produits pilotes

Nom du produit	Responsable de Produit
Websphere	YF
S1WS	VO
SAP	EP
$U	DH
Oracle	JP
Glassfish	NC
Colombus	RW
CFT	LB

Cahier d'expression des besoins

A la suite de nombreux groupes de travail avec les différents responsables des produits pilotes, un cahier des charges précis a pu être établi sur les différentes fonctionnalités nécessaires dans le cadre du développement de ce projet.

Besoins fonctionnels

Les utilisateurs ont besoin d'avoir accès à un référentiel technique dans lequel seront stockées toutes les configurations techniques de chaque produit installé sur les serveurs du groupe.

L'accès à cette base documentaire doit se faire via un site intranet où les données seront mises à jour de façon dynamique en collectant les informations sur les différents serveurs du groupe.

Voici ci-dessous le tableau récapitulatif des besoins fonctionnels :

Tableau II : Les besoins fonctionnels

	LIBELLE	PRIORITE	STATUT
1	Avoir un référentiel technique **unique** dans lequel seront stockées toutes les configurations techniques des produits d'infrastructures utilisés par PSA. **Service ciblé :** Service ETSO et ASII/DBDC **Multi OS :** Unix – Linux – Windows – Z/OS	Elevée	Approuvé
2	Stocker un ou des « **templates** » de configuration applicative ou produit réutilisable et distribuable.	Basse	Approuvé
3	**Extraire les données** de la configuration technique directement dans les **fichiers de configuration**. Ce qui implique la création d'un agent dynamique par RACHEL pour chaque version de produit.		Rejeté

4	**Extraire les données** de la configuration technique à partir d'un fichier généré lors de l'exécution périodique d'un script crée par le responsable du produit.	Elevée	Approuvé
5	Les informations sur le produit seront extraites de **REFLEX**. (Nom, version, responsable de produit, SYDID)	Elevé	Approuvé
6	Exécuter le script d'analyse de façon **périodique** en **local** sur le serveur. La réutilisation du cron de purge sera possible sur chaque serveur UNIX et Linux. Pour Z/OS, il faudra planifier le job sous OPC. Pour Windows, il faudra utiliser le scheduler pour planifier l'exécution des scripts d'analyse.	Moyenne	Approuvé
7	Lancer une analyse d'un serveur complet de façon **manuelle**. Cette action doit se faire directement depuis le référentiel RACHEL.	Basse	Approuvé
8	**Stocker** une nouvelle configuration produit en base de données que si celle-ci est différente de la version précédente.	Moyenne	Approuvé
9	Avoir un **historique des versions** de configuration stocké dans la BDD accessible depuis l'interface graphique. (5 versions par Instance)	Haute	Approuvé

10	Avoir un **module de statistiques** permettant l'affichage de rapport suivant différents critères (Instances, paramètres)	Moyenne	Approuvé
11	Pouvoir comparer les paramètres d'une configuration : - avec une autre configuration - avec une configuration préconisée - avec la version précédente de cette même configuration	Haute	Approuvé
12	Utilisation de la **carte grise** pour définir le path où sera présent le fichier de paramètrie à extraire par Rachel → le fichier de paramètre sera identifié grâce à la normalisation de son emplacement dans les arborescences des produits.		Rejeté
13	Remonter des **anomalies REFLEX** grâce à RACHEL. → Un comparatif entre les données remontées sur les serveurs par RACHEL et une extraction de la base REFLEX permettra de remonter d'éventuelles anomalies.	Moyen	Approuvé
14	**Rendre autonome** le référentiel afin que **chaque responsable produit** puisse mettre à jour les fiches produits dont il a la gestion. → Chaque responsable produit devra créer son propre script de récupération de la configuration technique.	Haute	Approuvé
15	**Harmoniser l'interface** du référentiel par **famille de produit**		Rejeté en état

	(ex : application server, produit de BDD) → Une étude sur la présentation des données sera à faire de façon plus détaillés. (Famille de paramètre comme par ex JVM) La codification des paramètres facilitera l'exploitation des données intégrer dans le SGBD.		Modifié dans le point 16
16	Regrouper les paramètres produits par **famille** afin de profiter d'une meilleure ergonomie de ceux-ci sur l'interface IHM.	Haute	Approuvé
17	Pouvoir faire une **extraction** des configurations techniques dans un fichier Excel.	Basse	Approuvé
18	Avoir un historique de cinq ans des configurations produits en base de données.	Moyenne	Approuvé
18 bis	Garder la configuration du produit pendant six mois après la suppression du produit.	Basse	Approuvé
19	Présentation d'un ensemble de configuration selon plusieurs paramètres de recherche. (paramètre remontée par RACHEL et ceux de REFLEX)	Basse	Approuvé
20	Lister les instances produits de la responsabilité d'un pôle en fonction d'une BU. EX : liste des produits de responsabilité ESO/PRD/PGI pour la	Basse	Rejeté

	BU SIFA → Ceci implique la liaison dans REFLEX entre une application attachée à une BU et le nom des produits utilisés par cette application. Cette fonction est déjà prévue dans REFLEX mais n'est pas renseigné.		
21	Ajouter dans le script d'analyse la variable du nom de l'application pour l'afficher dans la config.	Basse	Rejeté
22	Avoir une valeur préconisée en fonction d'un environnement (ex : SGP, INETPSA, PROD, PREPROD). → Il faudra faire une version différente pour chaque environnement. Voir 2 et 11	Moyenne	Rejeté
23	Avoir accès à la config d'un serveur ou d'un produit via une URL spécifique. (ex :http://rachel.inetpsa.com/serveur=yvas2710&produit=squid) → A prévoir dans les requêtes sur le SGBD	Basse	Approuvé

Besoins non fonctionnels

- Identification du contexte d'utilisation

Le référentiel technique sera utilisé par les responsables produits afin de consulter via une

interface intuitive la configuration technique de leur produit à un instant voulu et pouvoir ainsi comparer les données de celles-ci avec d'autres configurations. Trois types de comparaison seront ainsi possible :

- Une configuration préconisée pour un produit.
- La configuration d'un autre serveur pour le même produit.
- La même configuration historisée afin de voir les paramètres modifiés dans le temps.

Le référentiel devra mettre en évidence les différences détectées lors de la comparaison d'une ou plusieurs configurations.

Ce référentiel doit faciliter la flexibilité des produits associés à nos applications sans perte d'information.

- Disponibilité

Le référentiel devra dans la mesure du possible être accessible 24h/24 et 365 jours /365. Néanmoins le temps maximum pendant lequel les utilisateurs ou clients peuvent supporter le non fonctionnement du système est estimé à plus de 2 jours.

- Intégrité – Sauvegarde – Restauration des données

L'intégrité est évaluée par rapport au coût de remise en ordre des enjeux affectés par le sinistre, y compris les coûts induits chez les partenaires éventuellement impactés.
Le coût indicatif de la remise en ordre des enjeux est évalué à moins de 150 k€

- Confidentialité

L'information est réservée aux personnes, entités ou partenaires qui en ont besoin pour agir. Sa divulgation à des personnes non autorisées peut entraîner des préjudices importants mais réparables.

Deux services applicatifs seront mises en place par une authentification LDAP, l'un pour les responsables produits et l'autre pour les administrateurs du référentiel.

- Facilité d'utilisation

Une interface IHM adaptée aux besoins des utilisateurs sera mise en place. Une documentation et présentation seront fournies auprès de chaque utilisateur.

Le référentiel technique ne sera utilisé que par les responsables de produit, les utilisateurs seront donc des informaticiens habitués à consulter des interfaces web.

- Portabilité

Entrant dans le cadre du projet d'infrastructure 2.0, l'application devra être capable d'être porté d'un serveur à un autre en utilisant la procédure de flexibilité.

1.1 Architecture

Quelles filières de développement ?

Trois technologies majeures représentent les grands standards du marché : Java, LAMP, .NET. Chez PSA, les orientations retenues prennent en compte plusieurs principes directeurs: indépendance technique (portabilité, pérennité...), maîtrise économique (mise en concurrence, Open Source...), réduction de la diversité (universalité, couverture fonctionnelle...), disponibilité des compétences.

Ainsi pour les nouveaux projets, la stratégie de développement défini par PSA repose sur deux filières : une cible Java J2EE et une alternative PHP (LAMP). Des solutions complémentaires ou des exceptions sont néanmoins nécessaires, respectivement pour des développements propriétaires (ex. SAP R3, CATIA...) ou des besoins spécifiques (.NET, Domino, Uniface...) Chaque solution technique de développement correspond à des cas d'usage appropriés. Ces solutions prédéfinies constituent les « Filières d'architectures ».

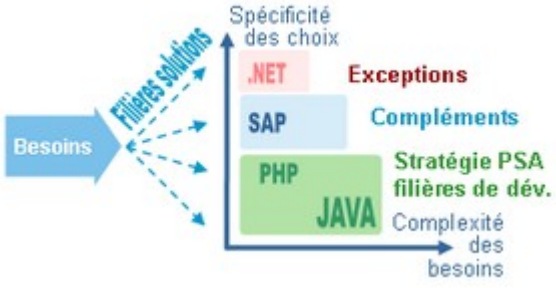


Figure 7 : Les différentes filières de développement

Chez PSA, le langage PHP est indissociable de la filière LAMP dont l'acronyme signifie Linux (système d'exploitation), Apache (serveur Web), Mysql (base de données) et PHP (Langage). Ces composants sont issus du monde Open Source, il peut exister toutefois des restrictions légales en cas de redistributions commerciales d'applications de ce type. La plateforme de production retenue par PSA s'appuie sur du matériel x86.

La filière LAMP fait partie des filières de développement retenues par PSA. Elle constitue une alternative à la filière privilégiée Java. Son périmètre de mise en œuvre concerne principalement le développement de besoins simples et isolés (sites Web sans criticité de disponibilité ou de sécurité, gestion de contenu, LSI collaboratif sous maîtrise d'œuvre des utilisateurs, hébergement de sites Web...). La facilité de mise en œuvre de ce type de développement permet d'engager rapidement des projets de taille modeste.

Après analyse du besoin technique du projet Rachel, mon choix de filière s'est dirigé vers la filière LAMP qui correspond le mieux à notre besoin technique.

La solution retenue est la filière LAMP avec l'utilisation d'un serveur Linux sur plate forme économique x86, un moteur Apache, une Base de donnée Mysql et l'utilisation du langage PHP pour le développement du site Web

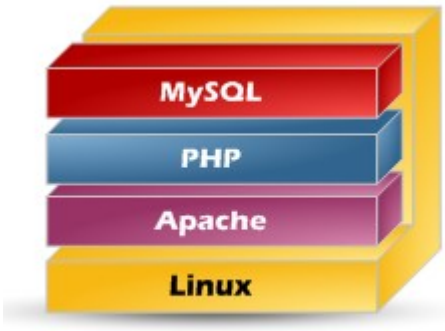


Figure 8 : La filière LAMP

Architecture logicielle

A ce jour, deux typologies d'architecture peuvent être mises en place, en fonction des besoins en termes de fréquentation ou criticité. L'ensemble de ces architectures reposent sur la Comut, afin d'offrir un investissement compétitif pour les équipes projet d'une part et d'autre part permettre une flexibilité maximum en cas d'accroissement des besoins initiaux.

La **première architecture** type se répartit sur deux serveurs physiques distincts, le premier assurant le rôle de serveur web avec Apache et PHP, tandis que le second héberge la base de données MySql. Cette architecture est calibrée pour assurer près de 120 utilisateurs distincts simultanés. Actuellement, cette architecture représente 75% des environnements mis en place chez PSA

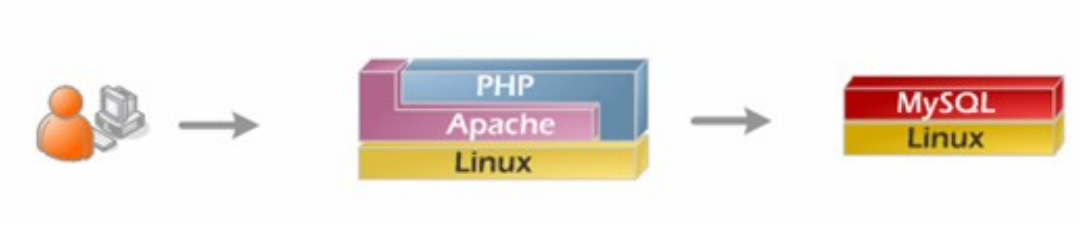


Figure 9 : 1ère architecture logicielle

L

La **seconde typologie d'architecture** est réservée aux sites étant soumis à de fortes sollicitations. S'appuyant sur une ferme de serveur web, avec, en amont, un répartiteur et sur un serveur dédié, la base de données. Cette architecture est actuellement mise en place pour des applications sensibles.

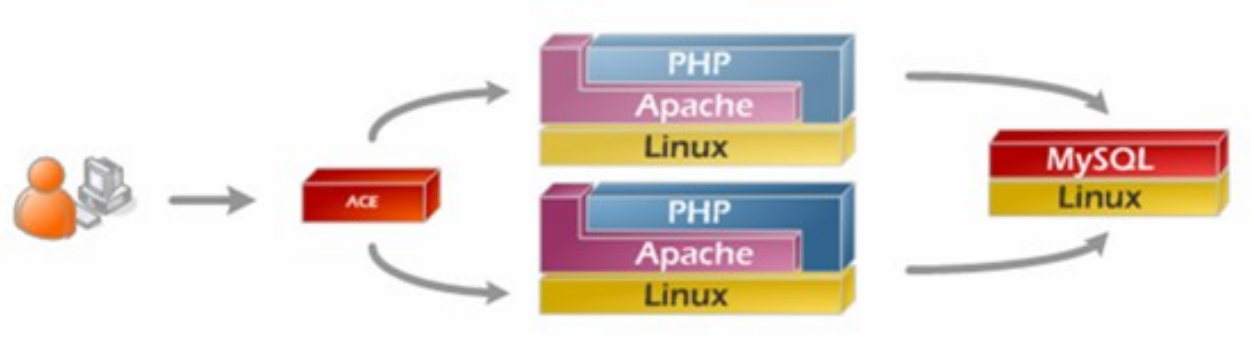


Figure 10 : 2ème architecture logicielle

Le projet Rachel correspond à la première typologie d'architecture. Le nombre de connexion au référentiel est estimé à 100, ce qui correspond à l'échelle défini dans la première architecture. Quant à la criticité du projet, celle-ci en est réduite. Mon choix d'architecture logicielle s'est donc dirigé vers le premier choix avec un serveur assurant le rôle de serveur web avec Apache et PHP, tandis que le second hébergera la base de données Mysql.

Architecture Applicative

Même si PHP ne repose pas « encore » sur une conception objet, au même titre que les filières JAVA/J2EE et .NET, il est préconisé un découpage des applications en 3 couches, en vue d'anticiper les prochaines évolutions de la plateforme LAMP au sein de PSA.
Ce découpage garantit la bonne adéquation aux principes attendus : robustesse, performance, sécurité, exploitabilité et maintenabilité.

Les 5 mots-clés des architectures applicatives :

Sécurité : La problématique de la sécurité doit être prise en compte sur l'ensemble des phases de la conception, de l'étude d'architecture à la mise en production, en passant par la phase de développement. A cet effet, des bonnes pratiques sont préconisées pour réduire la couverture d'exposition aux attaques de Sql Script Injection d'une part, et d'autre part permettre un profiling optimisé des données via l'utilisation de composants LDAP.

Maintenabilité : Réduire les coûts liés aux évolutions et gestion des anomalies fonctionnelles et techniques en vie courant passe aussi par une découpe intelligente des applications. Même si PHP n'offre pas encore un découpage aussi fin que les autres filières de développement Orientées Objet, il est préconisé de séparer clairement les interfaces graphiques des traitements métiers. L'utilisation de Framework permettra, tout en maintenant la simplicité de cette filière, de réduire encore plus la conception des applications web.

Performance : Augmenter les performances d'une application doit être un des leitmotivs appliqués sur l'ensemble des phases de conception. Par ailleurs, des mécanismes de caches et d'optimisations spécifiques peuvent être déployés à la demande par IDVS et ETSO. La combinaison des caches MySql, PHP et Typo3 et des optimisations/tuning sur Apache peuvent augmenter les performances d'une application avec un facteur de x20.

Scabilité : La mise en place des infrastructures Commut chez PSA illustre bien la volonté d'offrir une scabilité accrue aux applications LAMP. Ainsi, si nécessaire et en fonction des évolutions des applications existantes, la mise en place d'une architecture simple à une architecture en ferme peut être réalisée aisément. De plus, depuis peu, les architectures WIPE permettent de déployer des solutions web LAMP sur une infrastructure virtualisée.

Simplicité : La force de la filière repose sur sa simplicité d'utilisation. L'axe de professionnalisation visé sur la plateforme LAMP ne remet pas en cause le principe de simplicité.

Volumétrie de l'application

Afin de pouvoir chiffrer correctement le coût d'achat du serveur auprès de la direction des

achats, une volumétrie de l'espace disque nécessaire au projet a du être réalisée.

❖ **Volumétrie de la base de données**

Le calcul de volumétrie est fait à partir d'un produit moyen. J'ai donc décidé d'utiliser le produit Websphere afin de faire une estimation du nombre de paramètres à collecter. Nous avons pris le cas d'un produit sur l'OS Z/OS où de nombreuses occurrences sont présentes.
Cette estimation est donc réalisée avec le responsable de ce produit et en s'aidant du référentiel déjà existant. Nous avons estimé que 7000 paramètres seront à remonter par serveur :

❖ 7000 paramètres sur 1 serveur
❖ 10 serveurs Z/OS
=> 70 000 paramètres pour tous les produits Websphere sur Z/OS

Les 7000 paramètres seront collectés dans un fichier texte qui sera par la suite exploité par le référentiel Rachel dans la base de données.
Taille des fichiers :

- ❖ 700 ko pour les 7000 paramètres d'un serveur
- ❖ 7 Mo pour tous les serveurs Z/OS
- ❖ On doit garder 5 versions historiées par instances
- ❖ 7 Mo * 5 = 35 Mo pour le produits Websphere sur Z/OS

Nous ciblons environ 800 produits qui remonteront leurs paramètres dans le référentiel Rachel, ce qui nous fait : 35 * 800 = 28 Go d'espace disque pour les fichiers de configurations à stocker dans la Base de données.

A ceci doit s'ajouter les traitements annexes faits sur la base de données (Import/Export avec Reflex, gestion des traitements, gestion des alertes) estimée à 10 Go

Enfin, nous pouvons ajouter 20 Go d'espace disque nécessaire pour les différents Index mis en place sur la base de données.

Ce qui nous fait une volumétrie de 60 Go d'espace disque nécessaire pour la base de données.

❖ **Volumétrie du serveur**

Des besoins en espace disque sont nécessaire afin de stocker les fichiers d'exportations collectés des différents serveurs avant exploitation par les batchs pour les intégrer en base de données chaque jour, ainsi que les différents fichiers et scripts de l'interface IHM. Cette volumétrie est estimée à 10 Go.

La volumétrie totale nécessaire du serveur est **de 70 Go.**

❖ **Puissance du serveur**

Celle-ci est définie par les experts en développement et se calcule en nombres de requêtes par secondes. En effet, à partir du nombre d'utilisateurs potentiellement connectés sur le serveur, les experts vont nous donner une estimation du nombre de requêtes/secondes exécutées sur notre serveur.

Pour le projet Rachel, le nombre de requêtes a été estimé à **25.**

Base de données

Le principe des bases de données est de stocker, d'organiser de manière structurée et typée une grande quantité d'informations. Les SGBD (Système de Gestion de Base de Données) permettent de naviguer dans ces données et d'extraire (ou de mettre à jour) les informations voulues au moyen de requêtes. Dans notre cas, l'utilisation de la filière LAMP chez PSA nous a de facto obligé d'utiliser une base de données Mysql.

MySQL est un serveur de base de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. Il est multi-thread et multi-utilisateur. Ce choix correspond à notre besoin dans lequel nous aurons de nombreuses lectures à faire sur la base de données pour afficher les nombreux paramètres d'une configuration. Pour information, nous retrouverons dans les cas extrêmes des configurations avec 9000 paramètres à afficher.

Ci-dessous, voici la première version de la base de données élaborée pour le projet RACHEL :

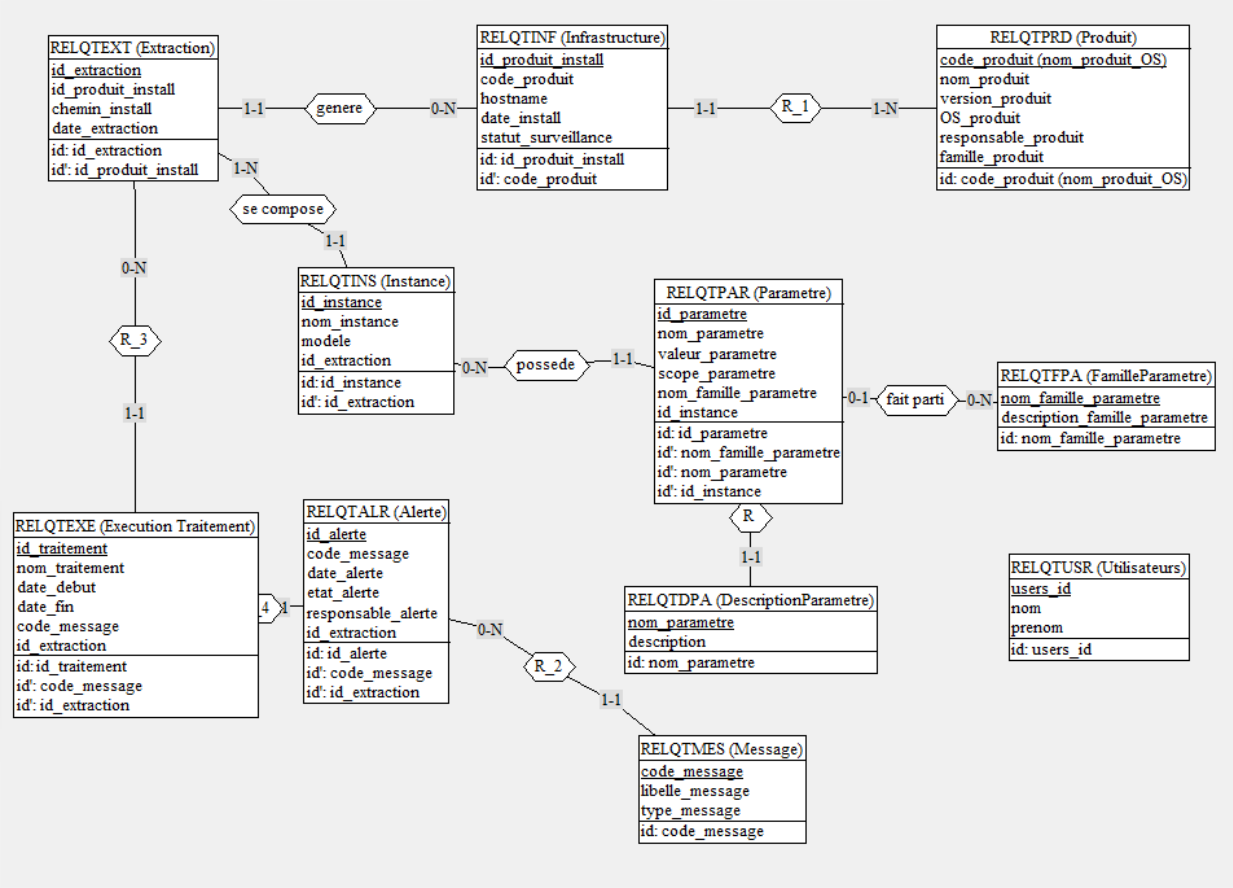


Figure 11 : La première version de la base de données

Très rapidement cette première version du schéma de la base de données fut inadaptée au projet Rachel. C'est notamment l'utilisation du Framework CakePHP sur la couche serveur du référentiel qui a imposé de nombreuses modifications dans le SGBD. En effet, de nombreuses règles de nommage sont imposées par le Framework. Voici la liste non exhaustive des conventions de nommages utilisées par CakePHP. Il est possible de ne pas respecter ces conventions mais cela impliquera une surcharge de travail car il faudra paramétrer chaque information. De plus, nous perdrons certains automatismes de fonctionnement intégrés au Framework :

❖ Les tables sont en **minuscule** et au **pluriel**, par exemple ordinateurs, imprimantes, lunettes, maisons, etc. Si nous avons un nom composé, il faut séparer les mots par _ pour que le dernier soit au pluriel. Par exemple, prenons une table des chaînes d'or, nous allons l'écrire or_chaînes. Si, nous avons la catégorie des articles, nous écrirons article_categories, les grandes maisons au bord de la mer, bord_de_la_mer_grandes_maisons.

❖ Les champs des tables sont en **minuscule**

- La clé primaire se nomme **id**, elle est de type **entier** et **auto_increment** ou équivalent

- Les champs **created** et **modified** de type **datetime** seront utilisés s'ils sont présents dans la table pour gérer automatiquement les dates de création et les dates de modification.

- Un champ **name** ou **title** sera automatiquement utilisé dans les vues comme libellé dans les menus déroulant de type SelectBox.

- Une clé étrangère vers une table poissons se nommera poisson_id, remarquez poisson est au **singulier** et un suffixe **_id** est ajouté. Une autre clé étrangère vers une table pays se nommera pays_id ce n'est pas donc juste une affaire de « s » à la fin, il faut spécifier le singulier correctement. Cependant cela ne fonctionne que pour l'anglais. Mais pas de panique il est possible d'ajouter nos propres mots et règles si besoin.

- S'il y a une jointure entre deux tables, par exemple, produits, catégories en (N, N) alors le nom de la table de jointure doit être les **noms des deux tables séparées par _ et dans l'ordre alphabétique,** donc catégories_produits et non produits_catégories. Cette table de jointure doit contenir au minimum deux champs, category_id et produit_id, les conventions des clés étrangères sont appliquées ici. Remarquez que c'est category_id et non categorie_id, parce qu'il s'agit des règles de pluralisation anglaise, mais bien sûr il est possible de définir des cas spécifiques.

Toutes ces règles de nommage, nous ont donc obligés à reprendre complètement l'architecture du SGBD. Voici ci-dessous une vue d'ensemble de celui-ci :

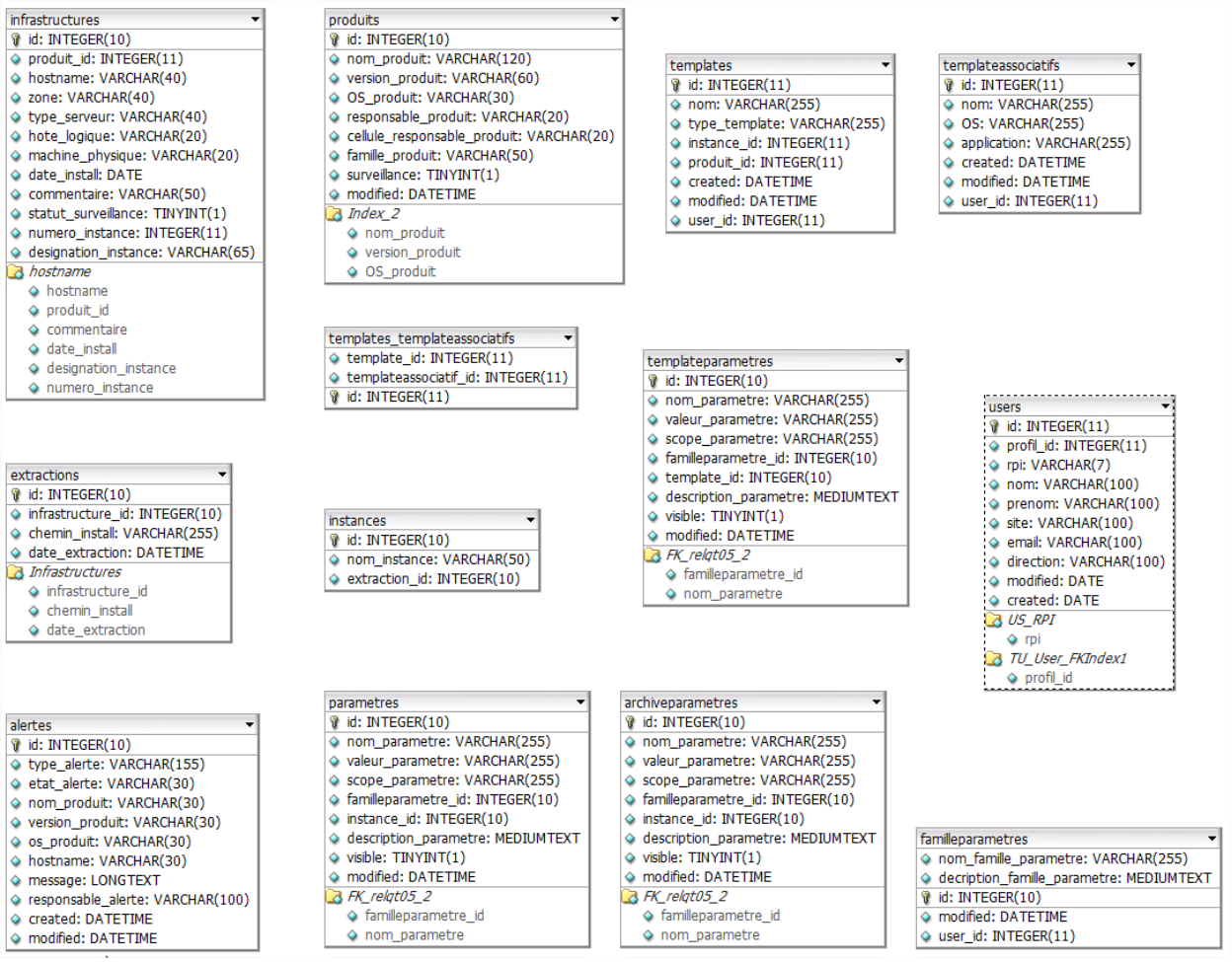


Figure 12 : La deuxième version de la base de données

Vous remarquerez que la base de données ne possède aucune relation visible, en effet la gestion des clés étrangères se fait par le Framework CakePHP comme nous l'avons vu précédemment dans les différentes règles de nommage.

Une deuxième évolution importante a eu lieu plus tard dans le déroulement du projet, un travail d'optimisation a été effectué sur la table paramètre, celle-ci stocke tous les paramètres des différentes configurations des produits d'infrastructures. C'est une table qui devient assez rapidement volumineuse. De plus, nous rappelons que nous souhaitons garder cinq versions historiées de chacune des configurations produits. Il a donc fallut trouver une idée afin d'améliorer les temps de réponses lors des requêtes sur cette table. Nous nous sommes aperçus que les informations demandées très fréquemment correspondaient aux paramètres de la configuration active, c'est-à-dire la dernière configuration présente sur le serveur. Les versions historiées ou

archivées ne sont consultées qu'en cas de problèmes pour faire une analyse ou détecter d'éventuelles anomalies entre deux versions d'une même configuration. Nous avons décidé de séparer les paramètres d'une configuration active de ceux d'une configuration archivée, nous allons retrouver deux tables différentes dans notre SGBD (paramètre et archive_paramètre) chacune d'entre elle vont donc stocker respectivement tous les paramètres de la configuration active et les paramètres des quatre versions historiées des configurations archives. Les temps de réponses se sont retrouvés nettement améliorés lors de l'affichage de grande configuration (> 6000 paramètres)

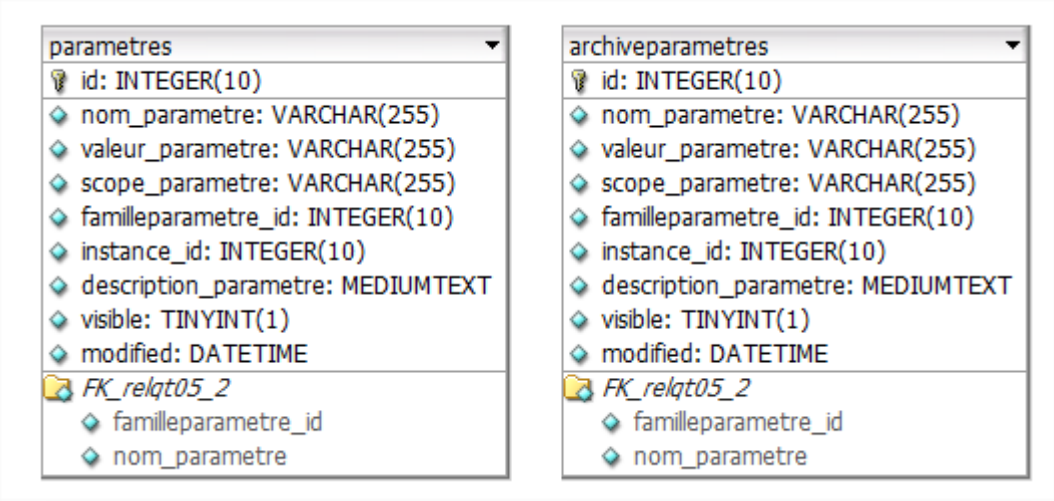


Figure 13 : Les tables paramètres et archive_parametres

La collecte des données

Principe de fonctionnement

Chaque produit installé sur les serveurs du parc informatique de PSA possède sa propre paramètrie produit. C'est cette paramètrie que nous souhaitons collecter et exploiter dans ce nouveau référentiel.

Cette paramètrie sera extraite à partir des différents fichiers de configuration présent sur le serveur. C'est donc grâce à un script que l'on nommera « script d'analyse » que nous allons pouvoir extraire la paramètrie produit qui sera mise en forme dans un fichier normalisé que l'on nommera « fichier d'exportation » portant comme nom Rel_extraction_produit.dat. Voici le schéma de la génération du fichier d'exportation :

Figure 14 : Génération du fichier d'exportation

Chaque responsable produit aura donc à sa charge la création du script d'analyse des produits dont il a la charge. Le langage utilisé pour écrire ce script n'est pas limité à une technologie. En effet, vu la diversité des systèmes d'exploitations utilisés chez PSA, différents langages de programmations seront utilisés. A noter, nous avons 4 OS distincts chez PSA avec Linux, Unix, Windows et Z/OS.

Tableau III : Langage de programmation conseillé

OS	Langage de programmation conseillé
Unix	Shell Script ou PERL
Linux	Shell Script ou PERL
Windows	PERL
Z/OS	JAVA ou JPYTHON

Chaque script d'analyse, devra être déployé sur tous les serveurs du parc informatique PSA pour y être exécuté chaque jour de façon périodique. Un autre référentiel PSA dans lequel sont référencées toutes les installations de produit existe déjà chez PSA. Ce référentiel se nomme

REFLEX. C'est sur ce référentiel REFLEX que RACHEL va pouvoir savoir où seront installés les produits à un instant T et ainsi pouvoir déployer les scripts d'analyse sur les bons serveurs.

Concernant les nouvelles installations de produit, l'ajout de ce script d'analyse sera à intégrer par le responsable produit dans le package industrialisé du produit.

Interaction avec REFLEX

REFLEX est le référentiel PSA par excellence. En effet, il s'agit ici en réalité d'un regroupement de plusieurs référentiels au sein d'un seul. On va donc retrouver les données des référentiels suivants :

- **PNMS (gestion financière)**,
- **GPMS (gestion des postes de travail et pilotage PSAV3)**,
- **REFLOG (gestion des serveurs)**,
- **REFAPPLI (gestion des applications)**,
- **TRESORUS (Gestion réseau des matériels informatiques).**

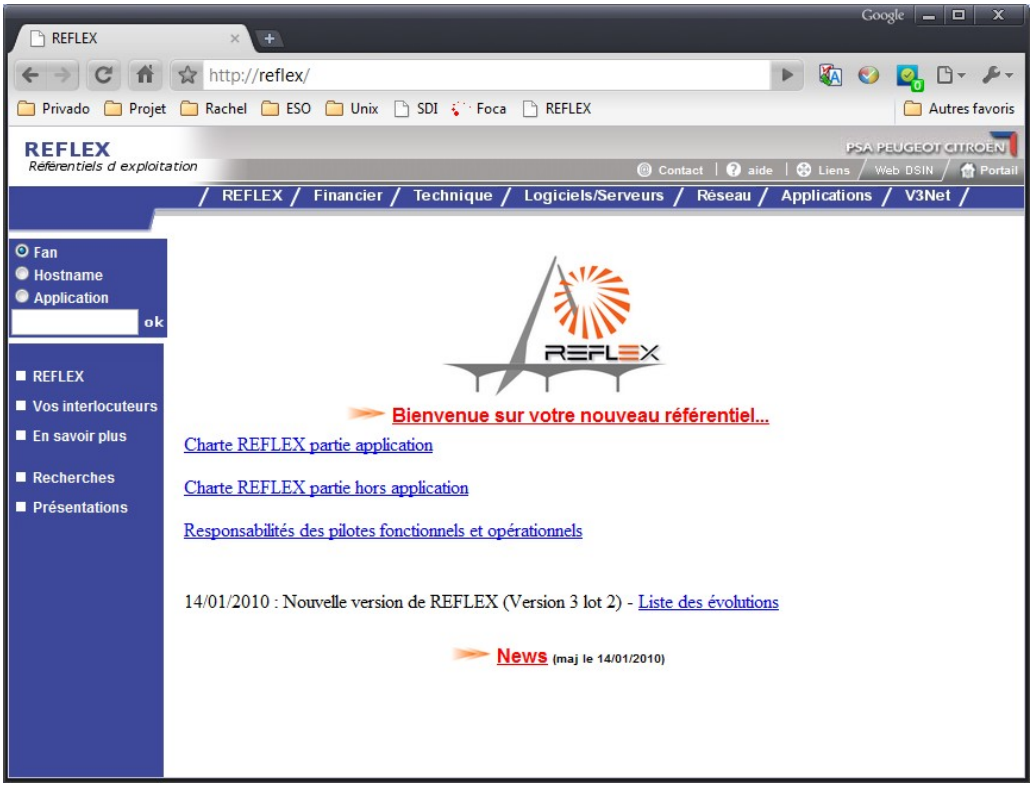


Figure 15: Le référentiel REFLEX

La section à laquelle RACHEL va s'intéresser est la section REFLOG du référentiel REFLEX. C'est dans cette section que nous allons retrouver toutes les données concernant les produits installés sur les différents serveurs du groupe. Nous allons retrouver sur un serveur la liste

de tous les produits installés sur celui-ci :

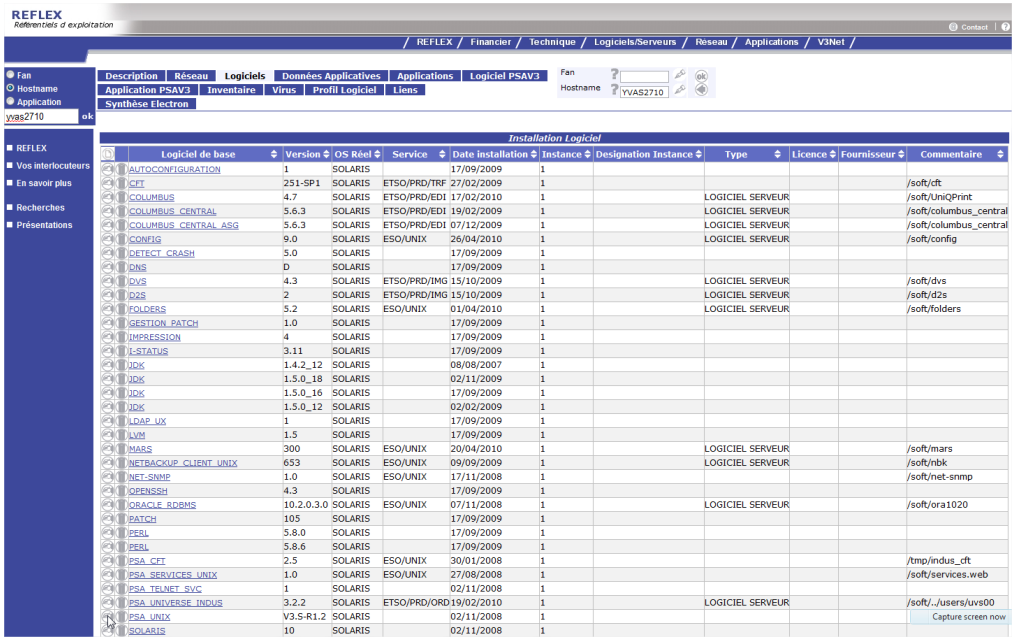


Figure 16 : La section REFLOG

C'est à partir de ces données que nous allons faire une interaction avec le référentiel RACHEL.

❖ Principe de fonctionnement de l'interaction avec REFLEX

Pour faire interagir REFLEX et RACHEL, nous allons faire des imports/exports quotidien des données de la base de données de REFLEX dans celle de RACHEL. Chaque jour, un script va s'exécuter pour extraire sous forme de fichier les données en provenance de REFLEX. Nous avons quotidiennement deux fichiers qui seront mis à disposition de RACHEL afin d'être exploités et intégrés dans la base de données. C'est ainsi que deux fichiers vont être mis à jour, le premier va concerner la liste des produits dans lequel nous allons retrouver les informations relatives aux produits et le deuxième va concerner la liste de l'infrastructure PSA dans lequel nous allons retrouver la liste des serveurs associés aux produits installés sur ceux-ci.

❖

La liste des produits

Un premier script va permettre de faire l'extraction de la liste des produits existant chez PSA. Pour chaque produit, plusieurs informations vont être recensées. Les voici :

Tableau IV : Extraction de la liste des produits

Nom	Description
Nom du produit	
Version du produit	
OS du produit	
Service responsable du produit	Il s'agit du nom du service auxquels est rattaché le produit
Responsable du produit	Il s'agit du nom du responsable produit
Famille de produit	Nom de la famille à laquelle se rattache le produit (ex : Gestion des travaux informatique)

Ces informations seront stockées dans la table Produits de notre Base de données, elles nous seront utiles pour la gestion des droits. En effet le responsable d'un produit aura tous les droits sur les produits dont il a la charge. Ce sont 3700 instances de version de produits qui sont ainsi stockés dans cette table. Voici quelques lignes du fichier généré lors de l'extraction de la liste des produits, chaque ligne correspondant à un produit. Les différents informations recensées pour chaque produit sont séparées par le caractère séparateur « | » :

```
Nom | Version | OS | Service responsable | ID responsable Produit | Famille du Produit

$UNIVERSE|5.0.1.0|WINDOWS/NT|ETSO/PRD/ORD|J118315|Gestion des travaux informatique
$UNIVERSE|5.00|AIX|ETSO/PRD/ORD|J118315|Gestion des travaux informatique
$UNIVERSE|5.00|HP-UX|ETSO/PRD/ORD|J118315|Gestion des travaux informatique
$UNIVERSE|5.00|SOLARIS|ETSO/PRD/ORD|J118315|Gestion des travaux informatique
$UNIVERSE|5.00|WINDOWS/NT|ETSO/PRD/ORD|J118315|Gestion des travaux informatique
$UNIVERSE|5.1.3|AIX|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3|HP-UX|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3|LINUX|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3|SOLARIS|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3_FX24451|AIX|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3_FX24451|HP-UX|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3_FX24451|LINUX|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
$UNIVERSE|5.1.3_FX24451|SOLARIS|ETSO/PRD/ORD|J624314|Gestion des travaux informatique
```

- La liste de l'infrastructure PSA

Le second fichier va concerner la liste des serveurs associés aux différents produits installés sur chacun d'eux, chaque ligne va associer un serveur à un produit installé, nous y retrouverons également d'autres informations utiles sur ces installations de produits comme la date d'installation de ceux-ci ou le répertoire d'installation du produit.

Voici le détail de chaque information collectée de REFLEX :

Tableau V : Information collectée depuis REFLEX

Nom	Description
Nom du serveur	
Nom du produit	
Version du produit	
OS du serveur	
Date d'installation	Date à laquelle le produit a été installé
Service responsable du produit	
Type de serveur	Ici nous allons définir le type du serveur (Production, pré-production, test, …)
Nom du serveur hôte logique	Un serveur virtuel est hébergé sur un serveur logique, c'est ici que nous renseignerons cette information
Nom de la machine physique	Un serveur logique est hébergé sur un serveur physique, c'est ici que nous renseignerons cette information
Zone du serveur	C'est la zone où est situé le serveur (Interne, DMZ, ZSR, …)

Toutes ces informations seront stockées dans la table Infrastructure de la base de données RACHEL. Ces informations couplées à la table des produits vont nous permettre d'avoir en temps réel l'état des installations logicielles de tout le parc PSA. La table infrastructure possède environ 140 000 occurrences.

Vous trouverez ci-dessous quelques lignes du fichier concernant l'infrastructure des produits :

Serveur | Nom Produit | OS | repertoire d'install | date_install | Service responsable | Type de serveur

BEAI98|GLANCE|380|AIX|/soft/glance|2003-11-20|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|IBM HTTP SERVER|2.0|AIX|CDD|2003-12-03|ETSO/DBDC|1||INTERNE|CDD||BEAI98|
BEAI98|MARS|100|AIX|/soft/mars|2004-09-02|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|MARS|200|AIX|/soft/mars|2007-03-19|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|NET-SNMP|1.0|AIX|/soft/net-snmp|2008-11-12|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|NETBACKUP_CLIENT_UNIX|653|AIX|/soft/nbk|2009-11-03|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|OPENSSH|4.3|AIX|/soft/OpenSSH|2007-09-06|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|ORACLE_RDBMS|8.1.7.4.0|AIX|/soft/ora817|2004-03-16|ETSO/DBDC|1||INTERNE|CDD||BEAI98|
BEAI98|ORACLE_RDBMS|9.2.0.3.0|AIX|/soft/ora920|2004-03-16|ETSO/DBDC|1||INTERNE|CDD||BEAI98|
BEAI98|PERL|5.8.0|AIX|/soft/perl-5.8.0|2006-02-27|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|PSA_CFT|2.7|AIX|/tmp/indus_cft|2009-04-07|ESO/UNIX|1||INTERNE|CDD||BEAI98|
BEAI98|PSA_CONFIG|5|AIX|/soft/config|2009-01-05|ESO/UNIX|1||INTERNE|CDD||BEAI98|

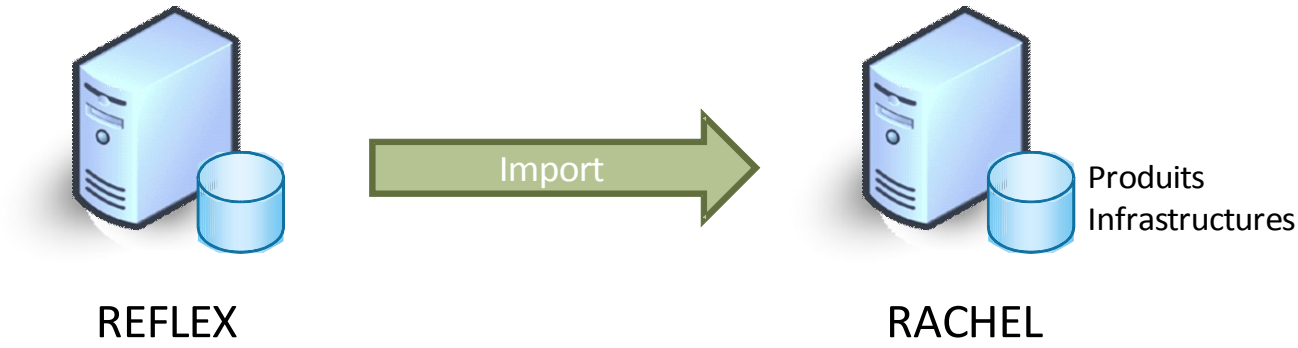


Figure 17: Interaction REFLEX / RACHEL

Ces deux fichiers extraits de la base de données de Reflex vont être exploités par un batch PHP afin d'être intégrés dans la base de données RACHEL. Je ne vais pas vous illustrer toutes les fonctions de ce traitement batch mais je vais vous illustrer deux fonctions de ce script qui vont vous permettre de comprendre son fonctionnement.

Pour faire l'import/export des données extraites du référentiel REFLEX, j'utilise deux tables temporaires, celles-ci sont en réalité les sœurs jumelles des tables produits et infrastructures et se nomment produitsTMP et infrastructuresTMP.

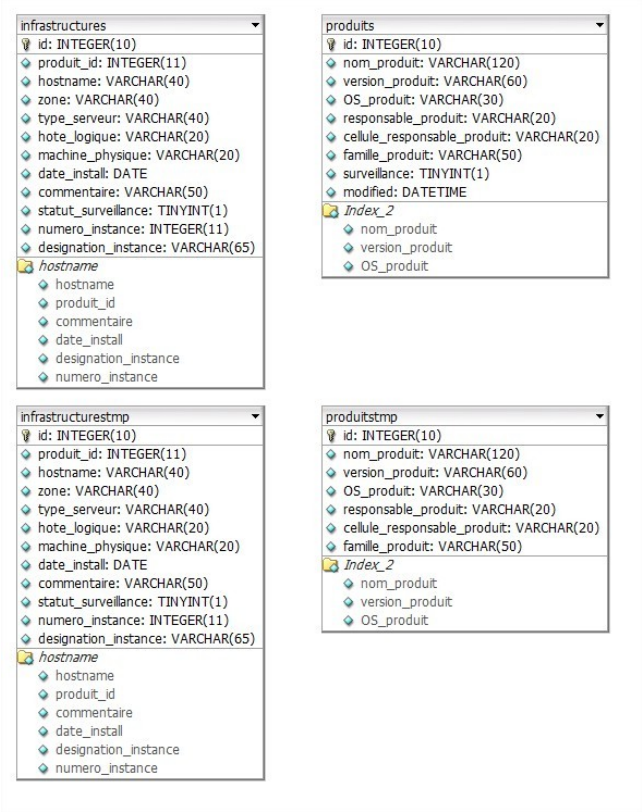


Figure 18 : Les tables temporaires Produits et Infrastructures

Les données directement extraites du référentiel REFLEX sont ainsi intégrées dans chacune de ces deux tables temporaires avec les fonctions extractionProduitReflex() et extractionInfraReflex(). Voici ci-dessous la fonction permettant la lecture du fichier texte dans lequel ont été extraites les données en provenance de REFLEX puis l'intégration de ces données dans le référentiel RACHEL :

```
// Fonction permettant l'extraction l'infrastructure présent dans Reflex
function extractionInfraReflex(){

// Accès en lecture au fichier
$fp = fopen($this->fichier, "r");

      if($fp){

      // Tant que que le fichier n'est pas terminé
            while (!feof($fp))
                  {

// On récupère dans la variable $ligne le contenu de la première ligne

                        $ligne = fgets($fp, 2048);

// La fonction split permet de récupérer des parties de caractères dans une
ligne : ici on indique que l'on prend tous les caractères depuis le début du
curseur jusqu'au caractère "$this->caractereSeparateur"

                        $split=split($this->caractereSeparateur,$ligne);
```

```
                        //Recherche de l'id du produit

                        $nom_produit=$split[1];
                        $version_produit=$split[2];
                        $os_produit=$split[3];

                        $donnees="id";
                        $table_produit="produits";
                        $conditions="`nom_produit`=\"$nom_produit\" and
`version_produit`=\"$version_produit\" and `OS_produit`=\"$os_produit\"";

                        $this->utilitaireDB->openConnexion();
                        $requete="SELECT $donnees FROM $table_produit WHERE $con-
ditions";

                        $resultat = mysql_query($requete);
                        $row = mysql_fetch_row($resultat);

                        // Récuperation de l'ID du produit

                        $produit_id=$row[0];

                        if ($produit_id==0){


// Le produit n'existe pas dans la table Produit, génération d'une alerte RACHEL
de type REFLEX_PRODUIT_ABSENT

                                $hostname=$split[0];
                                $date = date("Y-m-d H:i:s");
                                $type_alerte="REFLEX_PRODUIT_ABSENT";
            $message = "Le produit n'éxiste pas dans la table des produits";
            $responsable_alerte="ADMIN";

            $table_alerte="alertes";

$champs="type_alerte,etat_alerte,nom_produit,version_produit,os_produit,host-
name,message,responsable_alerte,created";
                                                $valeurs="'$type_alerte','INI-
TIAL','$nom_produit','$version_produit','$os_produit','$hostname','$message','$r
esponsable_alerte','$date'";

      // Insertion de l'alerte dans la table Alerte
            $requete="INSERT INTO $table_alerte ($champs) VALUES ($valeurs)";
            $resultat=mysql_query($requete);

            if (!$resultat) {
                  die('Requête invalide : ' . mysql_error());
                  echo "pb d'insertion de l'alerte";
                              }

                        }else{

//Le produit est présent, ajout de l'infra dans la table Infrastructure

                        $hostname=$split[0];
                        $commentaire=$split[4];
                        $date_install=$split[5];
                        $numero_instance=$split[7];
                        $designation_instance=$split[8];
                        $zone=$split[9];
```

```
                        $type_serveur=$split[10];
                        $hote_logique=$split[11];
                        $machine_physique=$split[12];

                        $table_infra="infrastructuresTMP";
                  $champs="produit_id,hostname,zone,type_serveur,hote_logique,ma-
chine_physique,date_install,commentaire,numero_instance,designation_instance";
                  $valeurs="'$produit_id','$hostname','$zone','$type_ser-
veur','$hote_logique','$machine_physique','$date_install','$commentaire','$nume-
ro_instance','$designation_instance'";

      $requete="INSERT INTO $table_infra ($champs) VALUES ($valeurs)";
      $resultat=mysql_query($requete);

            if (!$resultat) {
                  die('Requête invalide : ' . mysql_error());
                  echo "pb d'insertion de l'infra";
                        }
                        }// Fin du ELSE

                  }


            }


      }
```

Une fois les tables temporaires alimentées, deux nouvelles fonctions vont permettre d'updater le contenu des tables produits et infrastructures. En effet pour chaque entrée de la table temporaire, nous allons vérifier si celle-ci est présente dans la table en production. Si celle-ci est déjà présente nous ne touchons à rien et passons à la suivante, par contre si celle-ci n'existe pas nous l'ajouterons dans la table de production. Nous aurons la même procédure dans le sens contraire, c'est-à-dire que nous vérifierons pour chaque ligne de la table de production que celle entrée est bien présente dans la table temporaire, si celle-ci est absente de la table temporaire cela voudra dire que soit l'infra soit le produit a été supprimé et donc cette information sera à supprimer de la table de production. Voici ci-dessous la fonction updateTableProduit() qui a pour fonction de faire les updates entre la table temporaire et la table de production :

```
function updateTableProduit(){

      echo "Update de la table produit\n";
                  $don-
nees="nom_produit,version_produit,OS_produit,responsable_produit,cell  ule_res-
ponsable_produit,famille_produit";
      $table_produit="produits";
      $donnees_controler="id";
      $table_produitTMP="produitsTMP";
      $produitSupprimer=0;
      $produitAjouter=0;
```

```
// Suppression des produits n'apparaissant plus dans la table ProduitsTMP

      $this->utilitaireDB->openConnexion();
      $requete="SELECT $donnees FROM $table_produit";
      $resultat=mysql_query($requete);

// Lecture de chaque ligne présente dans la table de production Produits

      while ($row = mysql_fetch_array($resultat, MYSQL_ASSOC)) {

            $nom_produit=$row["nom_produit"];
            $version_produit=$row["version_produit"];
            $OS_produit=$row["OS_produit"];

// La condition de la requête vérifie que chaque champs possède la même valeur

            $conditions="`nom_produit`=\"$nom_produit\" and
`version_produit`=\"$version_produit\" and `OS_produit`=\"$OS_produit\"";

            $requeteTMP="SELECT $donnees_controler FROM $table_produitTMP WHERE
$conditions";

      $resultatTMP = mysql_query($requeteTMP);
            $tableau = mysql_fetch_row($resultatTMP);

// La valeur de la variable $tableau va nous permettre de savoir si le produit
est déjà présent dans la table temporaire

                  if ($tableau[0]==''){
                        $produitSupprimer++;

// Suppression du produit dans la table Produit car celui ci n'est pas présent
dans la table produitTMP

      $requeteSUPPRIME="DELETE FROM $table_produit WHERE $conditions";
            $resultatSUPPRIME = mysql_query($requeteSUPPRIME);

// Prevoir la suppression de tous les produits dans la table Infra

                  }else{
                        $present++;
                        }
            }

            echo $produitSupprimer." produits supprimes ";

// Ajout des nouveaux produits présent dans la table ProduitsTMP

      $this->utilitaireDB->openConnexion();
      $requete="SELECT $donnees FROM $table_produitTMP";
      $resultat=mysql_query($requete);

      while ($row = mysql_fetch_array($resultat, MYSQL_ASSOC)) {


      $nom_produit=$row["nom_produit"];
      $version_produit=$row["version_produit"];
      $OS_produit=$row["OS_produit"];
      $responsable_produit=$row["responsable_produit"];
                  $cellule_responsable_produit=$row["cellule_responsable_pro-
duit"];
      $famille_produit=$row["famille_produit"];
```

```
// La condition de la requête vérifie que chaque champs possède la même valeur

      $conditions="`nom_produit`=\"$nom_produit\" and `version_produit`=\"$ver-
sion_produit\" and `OS_produit`=\"$OS_produit\"";

      $requetePROD="SELECT $donnees_controler FROM $table_produit WHERE $condi-
tions";

      $resultatPROD = mysql_query($requetePROD);
      $tableau = mysql_fetch_row($resultatPROD);

// La valeur de la variable $tableau va nous permettre de savoir si le produit
est absent dans la table temporaire

                        if ($tableau[0]==''){
                              $produitAjouter++;

// Suppression du produit dans la table Produit car celui ci n'est pas pré©sent
dans la table produitTMP

      $requeteAJOUTER="INSERT INTO $table_produit
(`id`,`nom_produit`,`version_produit`,`OS_produit`,`responsable_produit`,`cel-
lule_responsable_produit`,`famille_produit`) VALUES (NULL,\"$nom_pro-
duit\",\"$version_produit\",\"$OS_produit\",\"$responsable_produit\",\"$cellule_
responsable_produit\",\"$famille_produit\")";
      $resultatAJOUTER = mysql_query($requeteAJOUTER);

            }else{
                              $present++;
                        }
                  }
            echo "\n";
            echo $produitAjouter." produits ajoutes ";

      }//Fin de la fonction UpdateProduit
```

Voici les deux tables qui seront ainsi alimentées quotidiennement :

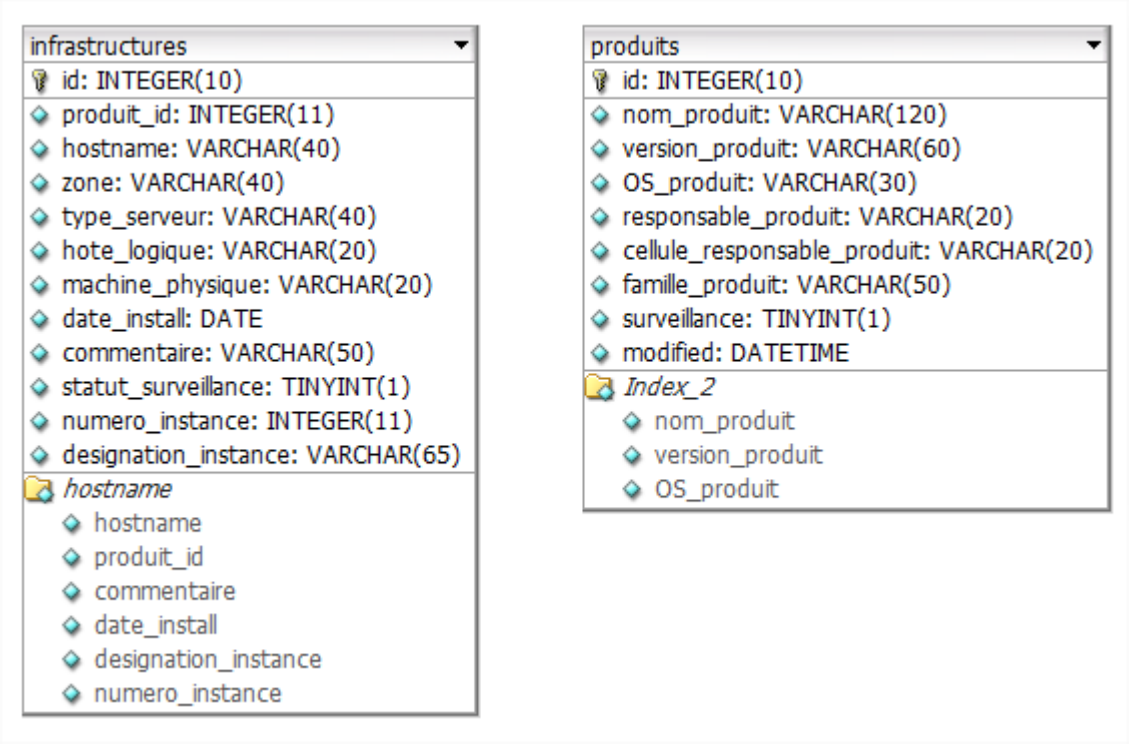


Figure 19 : Les tables Produits et Infrastructures

Le référentiel REFLEX possède néanmoins une petite faiblesse : la mise à jour de celui-ci doit se faire de façon manuelle. Avec REFLEX, aucune connexion n'existe avec les serveurs pour s'assurer de l'exactitude des données dont il fait référence. Prenons un exemple simple : un responsable produit vient de supprimer un produit sur un serveur X et oublie de mettre à jour le référentiel REFLEX afin de supprimer ce produit, nous aurons donc une information erronée dans ce référentiel sans que personne ne le sache.

Grâce à l'interaction entre RACHEL et REFLEX, nous allons pouvoir remonter des anomalies sur le référentiel REFLEX. En effet RACHEL collecte ses données directement depuis les serveurs du parc informatique PSA, celles-ci sont donc de source fiable, nous allons ainsi pouvoir les exploiter afin de remonter d'éventuelles anomalies. Deux types d'anomalies vont ainsi pouvoir être identifiés :

❖ Un produit n'est plus installé sur un serveur

Si d'après REFLEX, un produit est installé sur un serveur, nous allons être en attente de la remontée d'une configuration produit en provenance de ce même serveur. Si ce n'est pas le cas, une alerte va être affectée au responsable pour qu'il vérifie sur le serveur que le produit ne soit effectivement pas installé et qu'il fasse le nécessaire pour supprimer ce produit du référentiel REFLEX.

❖ Un produit est présent sur un serveur mais le référentiel REFLEX ne le sait pas.

Nous allons recevoir une configuration produit d'un serveur sur lequel, d'après REFLEX, ce produit n'est pas installé. Il semble donc que le produit soit bien installé sur le serveur mais que la personne qui a installé le produit ait oublié de mettre à jour le référentiel REFLEX de l'installation de celui-ci. Nous aurons une nouvelle alerte qui sera affectée au responsable de ce produit.

Normalisation des fichiers d'exportations

Script d'analyse

Le script d'analyse est le script executé en local sur le serveur permettant l'extraction des données de la configuration d'un produit pour les écrire dans un fichier .dat que nous nommerons fichier d'exportation. Ces données seront collectées par la suite afin d'être remontées au serveur central du référentiel RACHEL.

- Unique à chaque produit et version

Un script d'analyse correspond à un produit et version, celui-ci est mis en place à chaque installation du produit.

- Création du script

La création du script d'analyse de la configuration produit sera développée par chaque responsable produit. Celui-ci ira chercher les différents paramètres qu'il souhaite remonter à RACHEL dans les différents fichiers de configuration existant pour son produit. Une documentation a été mise en place auprès des responsables produit pour leur donner plusieurs méthodes et chercher les paramètres associés à leurs valeurs dans les fichiers de configuration (lecture de fichier plat, de fichier XML, …)

- Emplacement du script

L'emplacement du script d'analyse doit être placé dans une arborescence précise. Celle-ci est à mettre en place dans le répertoire FILESO du produit sous l'arborescence suivante :

/soft/nom_produit_install/fileso/rachel/REL_analyse.sh

- Distribution du script

Le script doit être intégré dans le package distribué par CADIX, outils d'installation des produits chez PSA sur les serveurs lors de l'installation d'un produit.

Néanmoins afin de rattraper le parc dans lequel ce script n'est pas présent, un script de rattrapage a été mis à disposition des responsables de produit afin de distribuer le script sur tous les serveurs où le produit concerné est installé.

- Exécution du script

Le script d'analyse sera exécuté de façon périodique (jour, semaine, mois ou année) selon le besoin du responsable de produit. Pour cela plusieurs outils seront proposés aux responsables produits :

➢ Pour Unix/Linux :

Il sera possible d'utiliser le cron UNIX. En effet, le script d'analyse pourra ainsi être appelé dans le script de purge correspondant au besoin du responsable de produit :

- CRON.sh = exécution journalière
- CRONhebdo.sh = exécution hebdomadaire
- CRONmensuel.sh = exécution mensuelle
- CRONannuel.sh = exécution annuelle

➢ Pour Z/OS

Il sera possible de planifier l'exécution d'un job via OPC.

➢ Pour Windows

Il sera possible d'utiliser le scheduler windows pour lancer l'exécution d'un script.

- Lancement manuel de l'analyse :

L'utilisateur aura la possibilité de lancer manuellement l'analyse complète d'un serveur depuis le référentiel RACHEL.

Après la demande d'activation de cette fonctionnalité par l'utilisateur depuis le référentiel RACHEL, un processus lancera tous les scripts d'analyse présents sur le serveur cible et les données extraites viendront être collectées pour les intégrer au référentiel RACHEL.

- Extraction des données dans un fichier d'exportation normalisé

Toutes les données exportées par le script d'analyse Rachel devront être stockées dans un fichier d'exportation normalisé dans le point 3 de ce document.

Fichier d'exportation Rachel

Définition : Il s'agit du fichier dans lequel seront stockées toutes les informations de paramètrie que nous souhaitons remonter au référentiel Rachel pour un produit. Un fichier d'exportation sera généré par version de produit installé sur tous les serveurs du groupe PSA.

1. Emplacement du fichier :

Ce fichier sera stocké dans le répertoire suivant :

/users/eso00/status/rachel/REL_extraction_NomProduitInstall.dat

NomProduitInstall correspond au nom du répertoire d'installation du produit sur le serveur

2. Droits sur le fichier d'exportation

Il faut mettre les droits 644 sur le fichier pour que le groupe Other puisse lire le fichier d'exportation.

3. Les entêtes d'identifications

Voici les paramètres indispensables à la bonne identification du fichier d'exportation par RACHEL lors de la collecte.

Nom_Produit | WEBSPHERE APPLICATION SERVEUR| → Il s'agit ici du nom exact renseigné sous REFLEX

Version_Produit | 2.7.3|
OS_Produit | AIX |
Chemin_Install | /soft/was273 |
Hostname | yvas2710|
Famille_Produit | OUTILS ET LOGICIELS POUR LES ENVIRONNEMENTS WEB|
Date_analyse | 2010-02-08 15:22:50|

4. Les valeurs des paramètres

Chaque ligne va correspondre à une valeur de paramètre associée à une instance et à son scope s'il existe. Chaque ligne se présentera sous la forme suivante :

Instance | Scope | Libelle | Valeur | Famille_Paramètre

a. Instance

Il s'agit ici du nom de l'instance auquel correspond le paramètre (exemple : http-serv01). Cette information n'est pas obligatoire notamment pour les produits n'utilisant pas la notion d'instance.

b. Scope

Dans ce champ, nous allons pouvoir rattacher un paramètre à une arborescence au sein d'une instance de produit. C'est notamment grâce à ce champ que nous allons pouvoir gérer la notion d'appartenance à un groupe.

Exemple avec la hiérarchie sous Websphere :

Instance | DOMAINE → NOEUD→ CELLULE | Libelle | Valeur | Famille_Parametre

Voici un exemple concret d'un fichier d'exportation :

AppServerNodeE1||name|bwf1as1|SERVER
AppServerNodeE1|dmcelbwf:bnnodee1|enableMultipleServerInstances|true|SERVER
AppServerNodeE1dmcelbwf:bnnodee1|maximumNumberOfInstances|2|SERVER

c. Libelle

Il s'agit du nom du paramètre, nous verrons plus tard dans l'interface que nous aurons la possibilité d'ajouter une description longue au libellé d'un paramètre.

d. Valeur du paramètre

Il s'agit ici de la valeur du paramètre.

e. Famille de Paramètre

Il s'agit des familles de paramètre à laquelle est associé ce paramètre. Cette information sera grandement utile lors de l'affichage d'une configuration, nous pourrons ainsi classer les paramètres de la configuration par Famille de paramètre.

Exemple de fichier d'exportation

```
----------
```

```
DEBUT ENTETE |
-------------
Nom_Produit|SUN ONE WEB SERVER|
Version_Produit|6.19|
OS_Produit|SOLARIS
Chemin_Install|/soft/nes619|
Hostname|yvas1640|
Famille_Produit|OUTILS ET LOGICIELS POUR LES ENVIRONNEMENTS WEB|
Date_analyse|2010-02-08 15:22:50|
-------------
FIN ENTETE  |
-------------
https-fidauto||RqThrottle|128|SERVEUR|
https-fidauto||DUREE_REQUETE_ACTIVE|0|LOGS|
https-fidauto||CGI_ACTIVE|1|CGI|
https-fidauto||STATS_ACTIVEES|1|STATISTIQUES|
https-fidauto||javahome|/users/nes619/bin/https/jdk|JVM|
https-fidauto||Xmx|256m|JVM|
https-fidauto||INTROSCOPE_ACTIVE|0|JVM|
https-fidauto||TYPE_DRIVER_JDBC|ojdbc14.jar|JDBC|
https-fidauto||maxpoolsize|32|JDBC_POOL_FIX_dr|
https-fidauto||idletimeout|300|JDBC_POOL_FIX_dr|
https-fidauto||maxwaittime|60000|JDBC_POOL_FIX_dr|
https-fidauto||transactionisolationlevel|read-committed|JDBC_POOL_FIX_dr|
https-fidauto||URL|jdbc:oracle:thin|
```

Collecte des fichiers d'exportations

Une fois les fichiers d'exportations générés par les différents scripts d'analyse des produits,

nous allons pouvoir procéder à la collecte des fichiers d'exportations pour les exploiter par le serveur central du référentiel Rachel et ainsi les intégrer dans la base de données.

Sur chaque serveur, en fonction du nombre de produits installés nous allons retrouver autant de fichier d'exportation à collecter et exploiter par un traitement batch afin d'être intégré dans la base de données et ainsi être accessible via l'interface IHM du référentiel.

La collecte des fichiers va être de centraliser tous les fichiers sur un seul et unique serveur utilisé par RACHEL pour faire les traitements. A partir d'un serveur nous allons nous connecter sur chaque serveur du parc informatique PSA pour collecter les différents fichiers d'exportations présents sur chaque serveur et les déplacer sur le serveur collecteur.

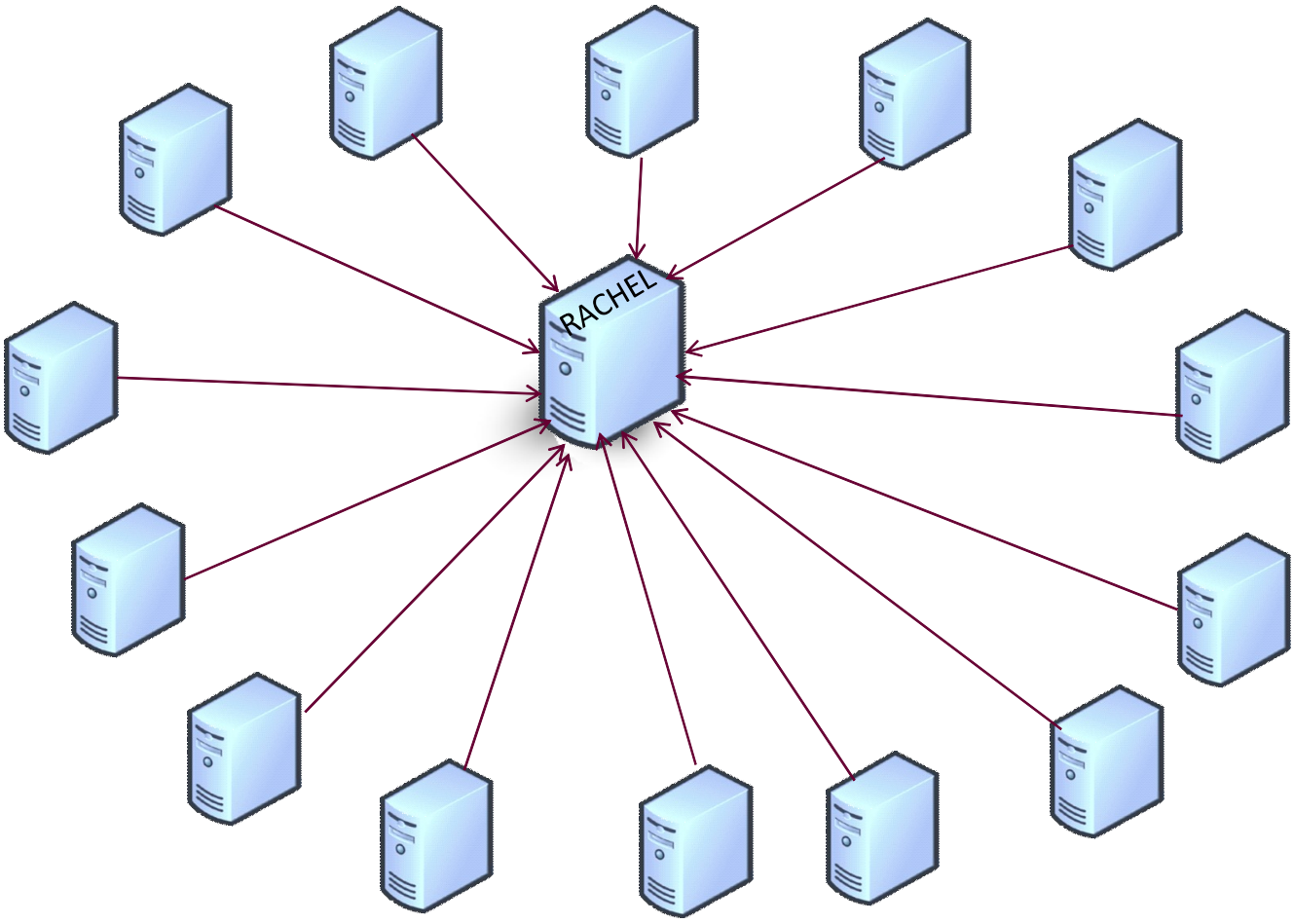


Figure 20 : Collecte des fichiers d'exportations

Une fois les fichiers centralisés sur le serveur Rachel, un premier script va pouvoir être exécuté sur le serveur afin d'éliminer les fichiers d'exportations qui n'ont pas été modifiés d'un jour

à l'autre. En effet, une copie de chaque fichier d'exportation va être gardée sur le serveur. La copie aura deux fonctions :

- ❖ La première est de comparer ce fichier avec le nouveau fichier qui vient d'être collecté. La comparaison se fera grâce à la fonction checksum qui nous dira précisément si le contenu du fichier est identique à la copie du fichier d'exportation présente dans la base de données. Si le fichier est le même alors ce nouveau fichier collecté sera supprimé car il n'y aura pas d'intérêt à l'intégrer de nouveau dans la base de données puisque le contenu de celui-ci est identique.
- ❖ La deuxième fonction de la copie du fichier d'exportation permet de réalimenter la base de données en cas de crash ou de corruption de celle-ci. Ce sera en sorte une deuxième sécurité qui s'ajoutera à la sauvegarde journalière du serveur et de la base de données.

Insertion des fichiers d'exportations dans le SGBD

Après la collecte sur le serveur central Rachel de tous les fichiers d'exportations, ceux-ci doivent être exploités afin d'être intégrés dans la base de données du référentiel RACHEL. Un traitement batch a donc été développé afin d'exploiter chaque fichier collecté et ainsi intégrer leur contenu dans le SGBD. Ce script a été développé en PHP, nous n'allons pas rentrer dans les détails de chaque fonction du script mais voici ci-dessous la fonction main de ce script qui permet de visualiser le principe de fonctionnement de celui-ci :

```
function main (){

      // Lecture du fichier à exploiter
      $this->lireFichier();

      // Extraction de l'entête du Fichier
      $this->extractEntete();

      // Récupération de l'ID du produit
      $this->recupIdProduit();

      // Vérification de la présence du produit sur cette Infra
      $this->verifPresenceProduitSurInfra();

      // Vérification de l'existence d'une extraction sur cette infra
      $this->verifExistanceExtractionSurInfra();

//-------------------------------------------------------------------//
// Etude des cas suivant le nombre d'extractions déjà présentes en Base //
//-------------------------------------------------------------------//
```

```
      if ($this->nombre_extraction==0){

            // Pas d'extraction sur cette Infra
            echo "<br> Pas d'extraction existante sur cette Infra";

            // Ajout de l'extraction
            $this->insertionExtraction();

            // Ajout de la parametrie
            $this->detectDebutParametrie();
            $this->insertionParametrie();

      }else{

            // Au moins une extraction existe
      switch ($this->nombre_extraction){

            // Présence d'extraction mais la limite n'est pas atteinte

            case ($this->nombre_extraction > 0 and $this->nombre_extraction <
5):
            echo "<br>Une extraction existe deja sur cette Infra a la date
$this->date_extraction";

            // Archivage de la paramétrie de l'extraction dans la table archive
paramètres
            $this->archivageParametrie();

            // Ajout de l'extraction
            $this->insertionExtraction();

            // Update de la parametrie de la config
            $this->detectDebutParametrie();
            $this->updateParametrie();

            // Nettoyage des parametres disparus
            $this->nettoyageParametrieObsolette();


            break;

            // Presence d'extraction avec atteinte de la LIMITE --> Suppression
de la version OBSOLETTE
            case ($this->nombre_extraction==5):

            echo "<br> 5 extraction existe deja pour cette infra <br> --> su-
pression de l'infra en date du $this->date_extraction pour l'extraction $this-
>extraction_a_supprimer";

            // Archivage de la parametrie de l'extraction dans la table archive-
parametres
            $this->archivageParametrie();

            // Suppression de la config devenue obsolète
            $this->deleteParametrie();

            // Ajout de l'extraction
            $this->insertionExtraction();

            // Update de la parametrie de la config
            $this->detectDebutParametrie();
```

```
            $this->updateParametrie();

            // Nettoyage des parametres disparus
            $this->nettoyageParametrieObsolette();


            break;

            // ANOMALIE RACHEL

            default :
            echo "<br> ALERTES RACHEL : Trop de version historisée";
            break;

            }
    }
```

La fonction main s'exécute pour faire appel aux différentes fonctions du script, dans une première étape nous allons lire le fichier pour extraire l'entête du fichier de configuration et identifier à quel produit et serveur correspond la configuration contenu dans ce fichier. Par la suite nous allons lire ligne par ligne le contenu du fichier pour intégrer les valeurs de la configuration dans les différentes table de la base de donnée. C'est également par ce script que nous allons archivés les paramètres d'une configuration historisé dans le cas ou celle-ci existe. Enfin nous pouvons également noté que ce script va générer des alertes lors de la détection d'anomalie dans une configuration.

L'interface IHM

Utilisation du framework CakePHP

Un framework est un ensemble d'outils et de composants logiciels organisé conformément à un plan d'architecture et des design patterns. L'ensemble forme un *squelette* de programme. Il est souvent fourni sous la forme d'une bibliothèque logicielle, et accompagné du plan de l'architecture cible du framework.

Avec un framework orienté objets, le programmeur qui utilise le framework pourra personnaliser les éléments principaux du programme par extension, en utilisant le mécanisme d'héritage, créer de nouvelles classes qui contiennent toutes les fonctionnalités que met en place le framework, et en plus ses propres fonctionnalités, créées par le programmeur en fonction des besoins spécifiques à son programme. Le mécanisme d'héritage permet également de transformer les fonctionnalités existantes dans les classes du framework.

Un framework est conçu en vu d'aider les programmeurs dans leur travail. L'organisation du framework vise la productivité maximale du programmeur qui va l'utiliser, gage de baisse des coûts de construction du programme. Le contenu exact du framework est dicté par le type de programme et l'architecture cible pour lequel il est conçu.

CakePHP est le framework PHP conseillé chez PSA pour développer les applications web dans la filière LAMP. Il n'est pas obligatoire et il est toujours possible de le développer de zéro. La charge de prise en compte d'un tel framework est estimée entre 1 et 5 jours selon la connaissance du modèle MVC, les concepts objets et du PHP.

Il s'agit d'un framework récent datant de 2005 inspiré du framework Rails. Ses concepts de base sont :

- Conventions plutôt que configurations : le but est d'avoir le moins possible de fichiers de paramétrage, des conventions de nommage vont permettre de rendre facultatif le maximum d'actions de configurations.
- Modèle MVC : ce framework est basé sur le pattern MVC «Modèle Vue Contrôleur» fréquemment utilisé pour les applications web.
- Scaffolding (échafaudage) : un script utilitaire va générer pour le développeur la majorité des squelettes des classes nécessaires et permet la mise au point des tables de base de données de l'application de vues dynamiques par interrogation et analyse automatique du modèle de base de données. Ce script utilitaire a été personnalisé par PSA pour permettre la génération d'une application à la charte PSA et reposant sur l'annuaire PSA pour l'authentification.

Modèle MVC

Le framework CakePHP repose sur le modèle MVC, il s'agit d'une architecture en 3 couches permettant de séparer les données, la logique et la présentation :

❖ **Modèle** : comportement de l'application : traitement des données, interaction avec la base de données, règles métiers, etc.

❖ **Vue** : interface avec laquelle l'utilisateur interagit (souvent du HTML ou du XML), la vue n'effectue aucun traitement, elle affiche l'interface et les données provenant des classes modèles.

❖ **Contrôleur** : répond aux événements émis par l'utilisateur, contient toute la partie logique de navigation de l'application et de sélection d'enregistrements.

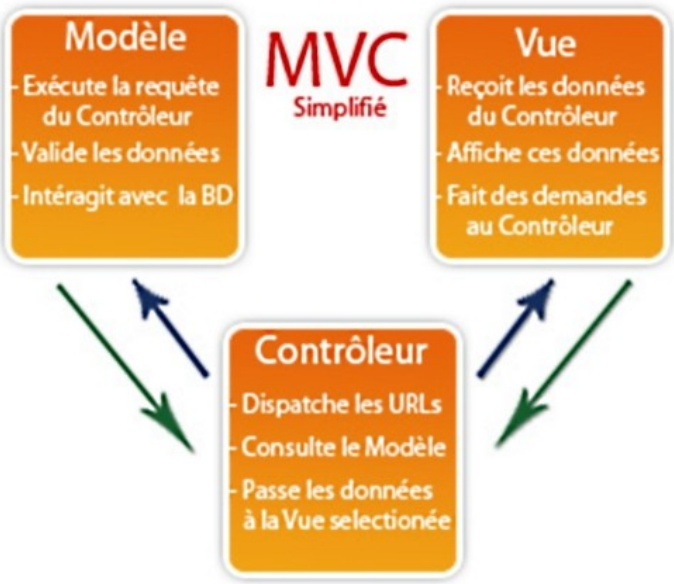


Figure 21 : Le modèle MVC

Fonctionnalités du framework CakePHP

De nombreuses fonctionnalités sont intégrées à ce Framework. Voici la liste des fonctionnalités intéressantes et utilisées lors du développement du projet Rachel.

❖ Compatibilité PHP4-PHP5

❖ Génération CRUD (interface de Création, Lecture (Read), MAJ (Update) et de Suppression (Delete)) et des requêtes usuelles.

❖ Url rewriting permettant l'utilisation d'url compréhensible humainement.

❖ Moteur de template ayant l'avantage d'être rapide à prendre en main (du pur PHP)

❖ Facile à déployer et à héberger

- Fonctionnalités de validation des données de propriétés des classes du modèle.
- Scaffolding (échafaudage) permettant rapidement la mise au point d'un prototype ou même d'un squelette d'application.
- Optimisation des performances avec moteur de cache pour les vues (html) et les requêtes SQL.
- Connectivité à de multiples SGBD.

A ces fonctionnalités, la cellule IDVS (support d'aide aux développeurs chez PSA) a rajouté des spécifications PSA à la distribution de ce framework :

- Lien avec Réunis (annuaire ldap PSA) pour la gestion des utilisateurs au niveau du login/password. Attention seule l'authentification des logins password est gérée par Réunis. Le concept de profil est quant à lui géré de façon applicative.
- Squelette d'application permettant de gérer la gestion des utilisateurs et des profils associés de l'application.
- Charte graphique PSA intégrée à l'application.
- De nouveaux générateurs CakePHP ont été implémentés afin de créer de façon interactive à partir des tables de l'application un squelette d'application respectant la charte PSA et permettant les opérations de base CRUD (Création, Read (Lecture), Update (MAJ), Deleted (Destruction).

De nombreuses règles sont néanmoins à respecter lors de l'utilisation d'un tel Framework, notamment le respect de nombreux concepts de nommage.

Installation du Framework

CakePHP repose sur le concept de conventions en place du paramétrage. Des conventions de nommage non obligatoires vont permettre de gagner en vitesse de développement et de profiter d'automatismes intégrés dans le Framework. Voici des exemples de conventions :

- le nom d'une table est le nom de la classe au pluriel.
- la colonne id est obligatoirement la clef de votre table en entier en auto-incrément.
- la colonne created est automatiquement renseignée par la date de création.
- la colonne modified est automatiquement renseignée par la date de modification.
- Le nommage de colonne avec le nom d'une table externe au singulier suivi de _id indique une relation vers cette table.

Le développement d'une application CakePHP est basé sur le cycle suivant et utilise le script fourni par le support PSA aux développeurs nommés bakePsa.

- créer la base de données et les tables de l'application.
- créer le répertoire de son application (déjà crée par le squelette PSA)
- créer la structure de son application dans le répertoire (effectué par le squelette PSA).
- Configurer la base de données
- Générer des classes modèles
- Tant que la base de données n'est pas figée
  - Génération des classes vues et contrôleurs avec le prototypage dynamique scaffold (désactivée chez PSA)
  - Saisie d'un jeu d'essai et test de navigation dans l'application
- Avec le script bakePsa
  - Génération des sources des classes vues et contrôleurs
  - Personnalisation de ces classes ainsi que des classes métiers pour développer l'application cible.

La mise en place d'un squelette d'application à partir du Framework CakePHP nous a permis de gagner du temps dans le développement de fonction simple. Voici une capture d'écran de ce squelette en utilisant la version distribuée par le support PSA :

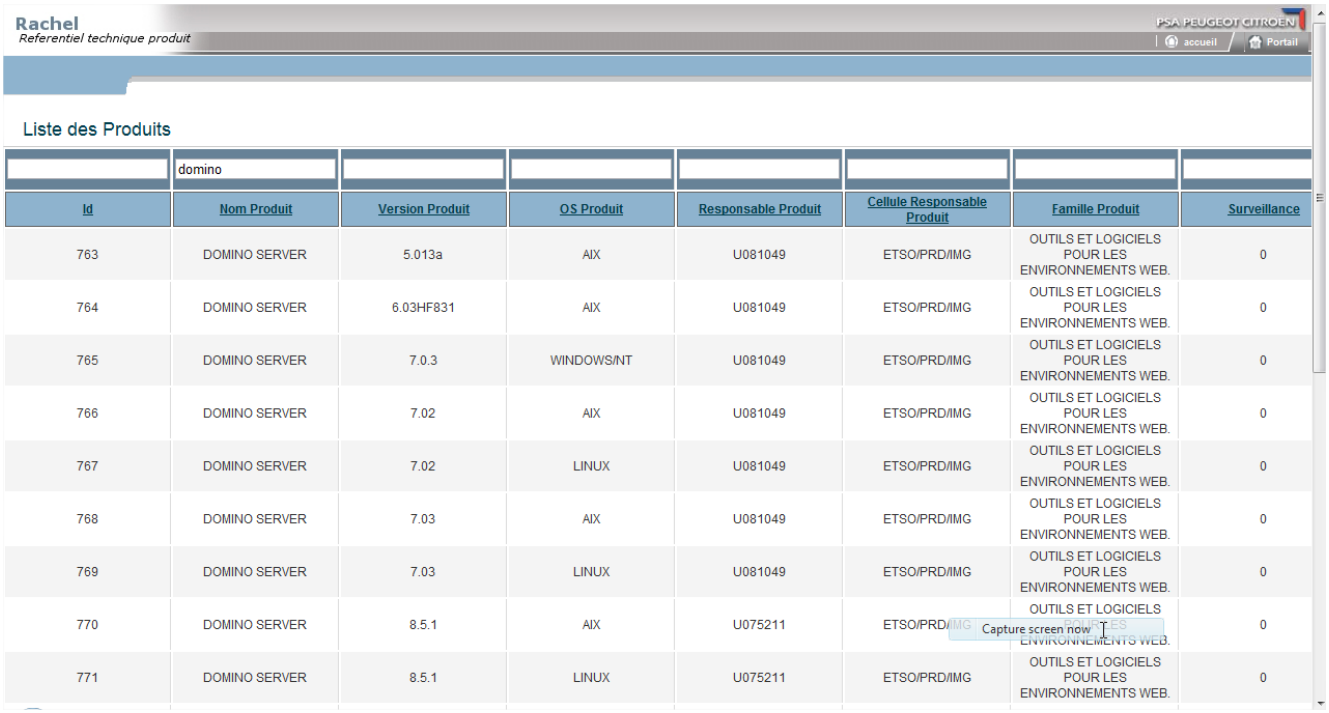


Figure 22 : Maquette cakePHP

Les différentes classes métiers seront ainsi développées dans le Framework cakePHP. Pour l'interface IHM, nous avons fait le choix d'utiliser une nouvelle technologie montante dans le monde du web avec Flex, une technologie orientée développeurs qui permet de réaliser des interfaces Web basées sur le lecteur Flash.

Utilisation d'une RIA (Rich Internet Application) avec FLEX

Une ***Rich Internet Application*** (RIA), ou **application internet riche** est une application web qui offre des caractéristiques similaires aux logiciels traditionnels installés sur un ordinateur. La dimension interactive et la vitesse d'exécution sont particulièrement soignées dans ces applications web.

Les applications web traditionnelles s'articulent souvent sur une architecture utilisant des clients légers : les traitements sont réalisés sur le serveur (distant), le client (local) ne fait que réaliser une présentation (exemple : HTML). Le client envoie ses données au serveur, celui-ci effectue le/les traitement(s) puis une page de réponse est renvoyée au client. Le serveur est donc sollicité à chaque interaction, hormis quelques cas spécifiques comme la saisie dans un formulaire.

Les RIA s'efforcent de rapatrier chez le client (local) une partie des traitements normalement dévolus au serveur. Les standards Internet ont évolué lentement et continuellement à travers le temps pour s'accommoder à ces techniques. Il est difficile de définir clairement ce qui constitue une RIA et

ce qui n'en constitue pas une. Généralement, ce qui peut être effectué au moyen d'une RIA est limité par les capacités du système client. Parce que les RIA utilisent les ressources du système client, elles offrent aux applications web des possibilités d'interfaces utilisateur plus réactives, ce qui serait impossible avec des balises HTML standards.

Nous pouvons déporter sur le client de nombreuses fonctionnalités, comprenant le glisser-déposer, l'utilisation de barres d'outils pour modifier les données, des calculs, données n'ayant pas nécessairement besoin d'être renvoyées au serveur.

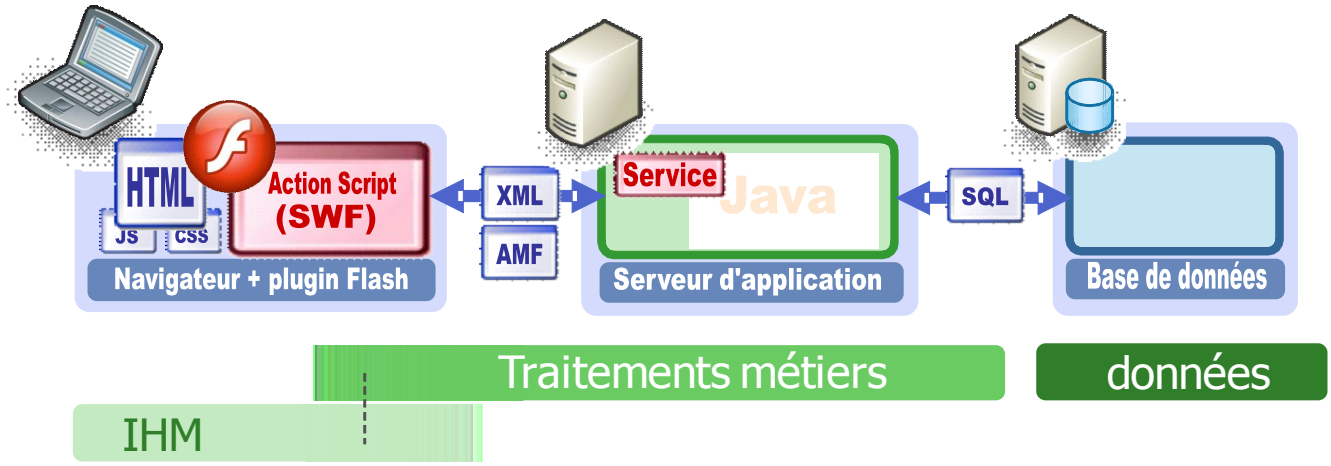


Figure 23 : Architecture RIA d'une application

Flex est un framework Open Source gratuit qui permet de créer des applications web ultra-interactives et expressives se déployant de manière identique sur la plupart des navigateurs, postes de travail et systèmes d'exploitation.

Flex offre un modèle de programmation évolué qui repose sur des langages standards et gère les modèles de conception courants. MXML, langage déclaratif basé sur XML, sert à décrire l'agencement et le comportement de l'interface utilisateur tandis que le langage de programmation orienté objet ActionScript 3.0 est employé pour la création de fonctions de traitement côté client. Flex inclut en outre une riche bibliothèque de composants comprenant plus d'une centaine d'éléments d'interfaces utilisateurs évolutifs réservés à la création d'applications Internet riches (RIA) ainsi qu'un débogueur interactif d'applications Flex.

Les RIA créées avec Flex s'exécutent dans le navigateur à l'aide d'Adobe Flash Player ou en dehors de celui-ci via l'environnement d'exécution multiplateforme Adobe AIR, qui garantissent une

réelle homogénéité sur la plupart des navigateurs et postes de travail. Installé sur plus de 98 % des ordinateurs connectés à Internet, Flash Player est un moteur d'exécution client de qualité professionnelle, comprenant des fonctionnalités vectorielles avancées, capable de gérer les applications plus exigeantes à fort pourcentage de données sans ralentir la cadence. En mettant à profit AIR, les applications Flex ont accès aux données en local et aux ressources système.

Cette filière de développement commence à se développer chez PSA pour les applications orientées WEB, en 2010 un projet de lancement de cette nouvelle filière a été lancé au sein de PSA. Je me suis portés pilotes sur cette nouvelle technologie et le projet RACHEL est devenu projet pilote pour le développement d'une application web sous FLEX.

Architecture technique de l'interface

Comme les traitements métiers s'exécutent sur le serveur php, il faut créer tous les controleurs métiers dans Cakephp afin de distribuer les données collectées au navigateur client qui traitera les données pour les mettre en forme. Pour cela, le plugin AMFPHP a été installé sur le serveur et permettra d'utiliser les controlleurs de cakePHP comme des services, il fonctionne aussi avec les RemotingObject de Flex.

Utilisation d'un Framework pour Flex

En pratique, le problème se posant à tout développeur Flex est le même que celui qui se pose à tout développeur tout court : comment éviter que le code ne se transforme en un spaghetti inextricable et devienne impossible à maintenir. Nous avons du faire des recherches afin de trouver une méthode de bon développement.

Quel framework choisir ?

De nombreux framework Flex suivant le motif MVC (Modèle Vue et Controleur) existe depuis quelques années mais deux sortent particulièrement du lot. Il s'agit de **PureMVC** et **Cairgorn**. Une étude de ces deux framework a du être réalisée afin de sélectionner le framework le plus adapté à notre projet.

Nous allons tout d'abord commencé par analyser le framework **Cairngorm**. C'est le framework MVC historique de Flex et le plus connu. Il reprend les codes du monde Java et se concentre

sur trois domaines: la gestion des actions de l'utilisateur, les interactions avec le serveur et la logique métier tout en gérant le contexte de l'utilisateur. Les classes principales sont le ModelLocator, le ServiceLocator, la logique métier et le FrontController. La force principale de Cairngorm est sa réputation dans la communauté, le fait qu'il soit un projet Open Source supporté par Adobe et avec une communauté active. Sa faiblesse se situe au niveau du nombre élevé de classes à coder (chaque événement est relié à une commande, donc une classe à écrire par événement). Une autre limitation est le fait que chaque événement doit avoir sa propre classe de commande, donc nous sommes limités à un répondeur par événement.

Son rival le plus fervent est **PureMVC**, à l'origine, ce framework n'a pas été conçu pour Flex. La philosophie MVC est donc respectée à l'extrême. Comme Cairngorm, un projet PureMVC découpe votre code en plusieurs packages et en de nombreuses classes. PureMVC profite aussi d'une large communauté de plus en plus active. Il est aussi très adapté au développement par équipes. La faiblesse de PureMVC est qu'il ne tire pas parti des forces du framework Flex *(il ne profite pas par exemple des forces du langage MXML)*. PureMVC est jugé plus difficile à appréhender que d'autres frameworks, la courbe d'apprentissage est plus étendue.

Le choix du Framework a été fait en collaboration avec le service IDVS (Support aux développements chez PSA). La filière de développement FLEX étant toute récente, il était important de faire le bon choix afin de mieux guider les développeurs Flex tout au long de leur projet. Notre choix s'est porté sur Cairngorm qui est beaucoup plus orienté vers Flex. En effet, nous avons vu précédemment que la grande faiblesse de PureMVC était qu'il ne tirait pas parti des grandes forces de Flex avec notamment la gestion du langage MXML. Le framework Cairngorm a donc été homologué et nous l'avons mis en place lors du développement du référentiel RACHEL.

Fonctionnement et utilisation du framework Cairmgorn

Dans le cas des applications frontales, le problème a été résolu depuis de nombreuses années : il faut séparer les données (et leur traitement) de leur représentation. Il y a évidemment des centaines de manières de procéder, mais en l'occurrence Adobe nous en propose une, à savoir Cairngorm, présentée comme un framework de développement. Ici, il faut entendre framework non pas dans le sens de boite à outils (ce qu'est déjà Flex) mais en tant qu'implémentation de « best practices ». Les concepteurs de Cairngorm ne disent d'ailleurs pas autre chose. Ce framework se divise en différentes couches, voici les fonctions de chacune :

- **Business :**

Cette couche va contenir toutes les informations concernant les services et les appels des procédures distantes.

❖ **Les déléguées :**

Cette couche est le magasin de services. Elle va contenir les définitions des méthodes distantes d'accès aux données. Elle consiste en une liste de classes auxquelles nous avons délégué la responsabilité d'encapsuler les appels des services distants.

❖ **Les services locateurs :**

C'est l'annuaire de toutes les sources de données, nous les appelons aussi les services distants. Chaque service possède un identifiant, un lien d'accès et d'autres informations qui permettent de le définir. Elles peuvent être un lien vers un fichier XML, un serveur de base de données…

❖ **Command :**

Le pattern commande est l'implémentation des fonctionnalités de l'application, chacune dans une classe. Chaque commande aura une mission (charger une liste d'exercices par exemple) bien précise qu'elle doit exécuter et gérer la réception et/ou l'envoi du résultat.

❖ **Control :**

Cette couche est basée sur le pattern FrontController. C'est une composition entre le modèle MVC et celui du Commande. Il permet d'associer un évènement à un type de traitement. Ce qui demande l'encapsulation des traitements dans des classes commandes où chaque commande possède une classe évènement qui servira à son déclenchement. C'est un annuaire dans lequel nous allons lister tous les traitements possibles que l'application peut offrir.

❖ **Event :**

Cette couche est celle qui s'occupe de la gestion d'évènements et du transfert des données entre la couche présentation et commande. Par exemple, si nous voulons ajouter un nouvel exercice, le déclenchement de l'évènement « ajouter exercice » doit être accompagné par une instance de l'exercice.

❖ **Model :**

Dans cette couche nous allons centraliser les accès à nos structures de données. C'est un annuaire

d'instances de nos classes objets. Cette couche se base sur le pattern Singleton qui nous permet de résoudre le problème « Comment m'assurer que nous avons bien une instance unique de telle objet ?».

- **View :**

Cette couche est la présentation des données pour l'utilisateur. Elle possède aussi sa logique de fonctionnement, elle s'occupe des contrôles des saisies et le déclenchement des évènements. Elle s'occupe de l'acheminement des informations depuis l'utilisateur et la couche évènement.

- **Vo :**

Ce pattern « Value Object » également appelé « Data Transfert Object » permet de définir la structure de nos objets que nous allons utiliser dans l'application. Il repose sur une classe qui définit un objet. Cette notion est prise de la programmation orientée objet. La structure de données dans ce pattern n'est pas une définition technique mais plutôt sémantique. Nous n'allons pas parler de tableau, de chaîne de caractères ou d'entiers mais plutôt de professeur, classe, exercice, série… Les objets doivent avoir un sens. La nouveauté par rapport à la programmation orientée objet est la centralisation des objets dans une seule couche.

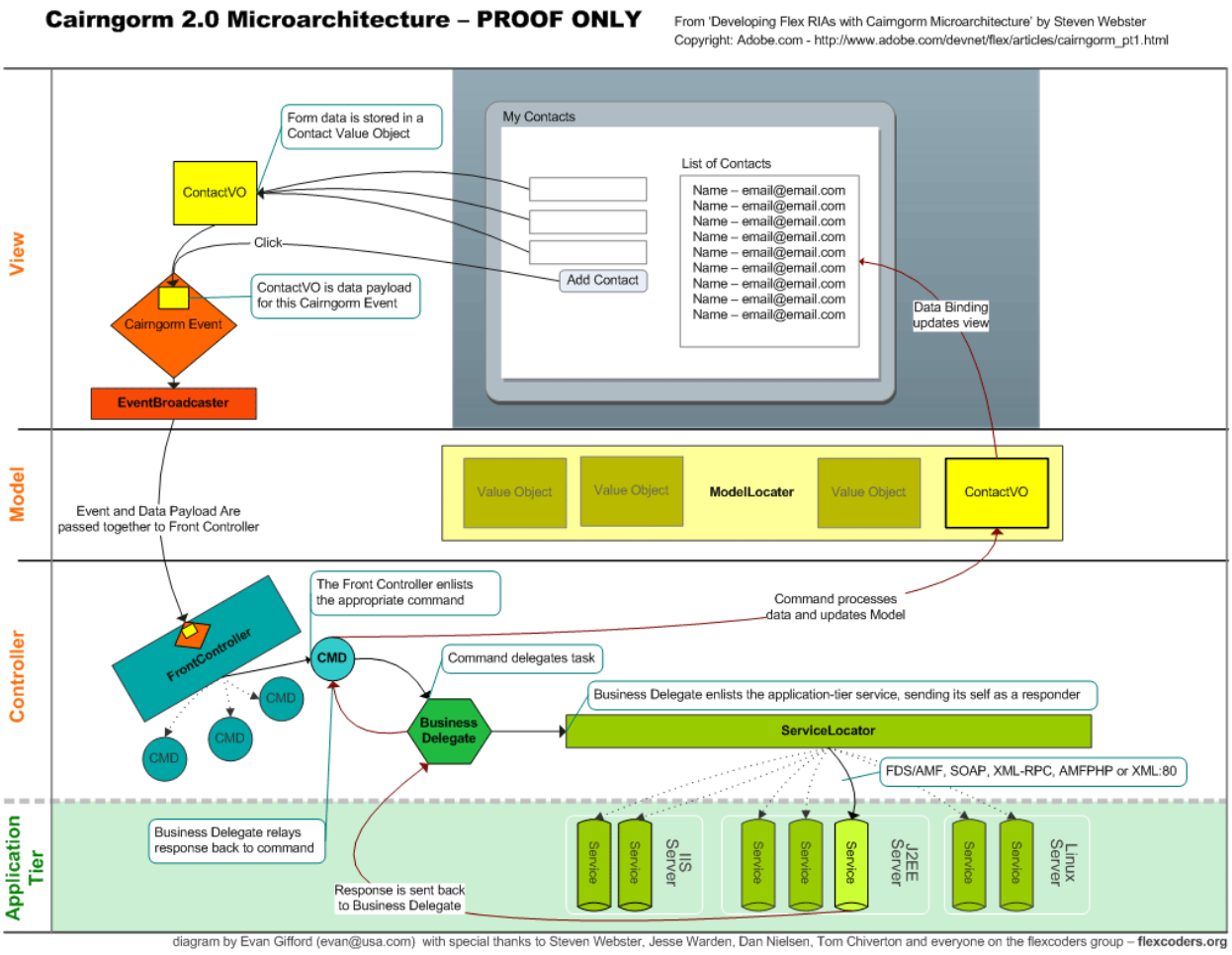


Figure 24 : Principes de fonctionnement de Cairngorm, WIKI PSA

Je vais vous décrire ci-dessous la procédure pour faire communiquer le serveur php avec l'interface IHM Flex.

Fonction Php

On va commencer par créer la fonction php chargée de remonter les données. Dans le controller Cake adapté (en fonction des tables à interroger), on créé une nouvelle fonction php. Les éléments à renvoyé sont transmis dans le « return ».

```
function FunctionName($parametre1 = '',$parametre2 = ''){

	return $Result;

}
```

Après avoir mis en ligne le controller, on peut tester la fonction dans un navigateur. Le nom du controller s'écrit alors sans le « _controller »

Adresse : http://rachel.dev.inetpsa.com/nom_du_controller/functionname

Ensuite, on teste la fonction dans le browser de Cpamf à l'adresse : http://rachel.dev.inetpsa.com/cpamf/browser

Si les données arrivent correctement dans le browser, la fonction est prête à être appelée depuis Flex.

Lien PHP/Flex

Il y a deux fichiers à éditer pour rendre une fonction de Cakephp appelable par Flex dans le répertoire Rachel_Flex > src > business : Services.mxml et ServicesDelegate.as

Dans le Services.mxml on doit avoir des nœuds comme suit :

Le nom de la source doit être construit avec comme suit : infrastructures_controller => InfrastructuresController

```
<mx:RemoteObject id="servicesControllerName" destination="amfphp"
source="NameController">
        <mx:method name="FunctionName"/>
</mx:RemoteObject>
```

Dans ServicesDelegate.as :

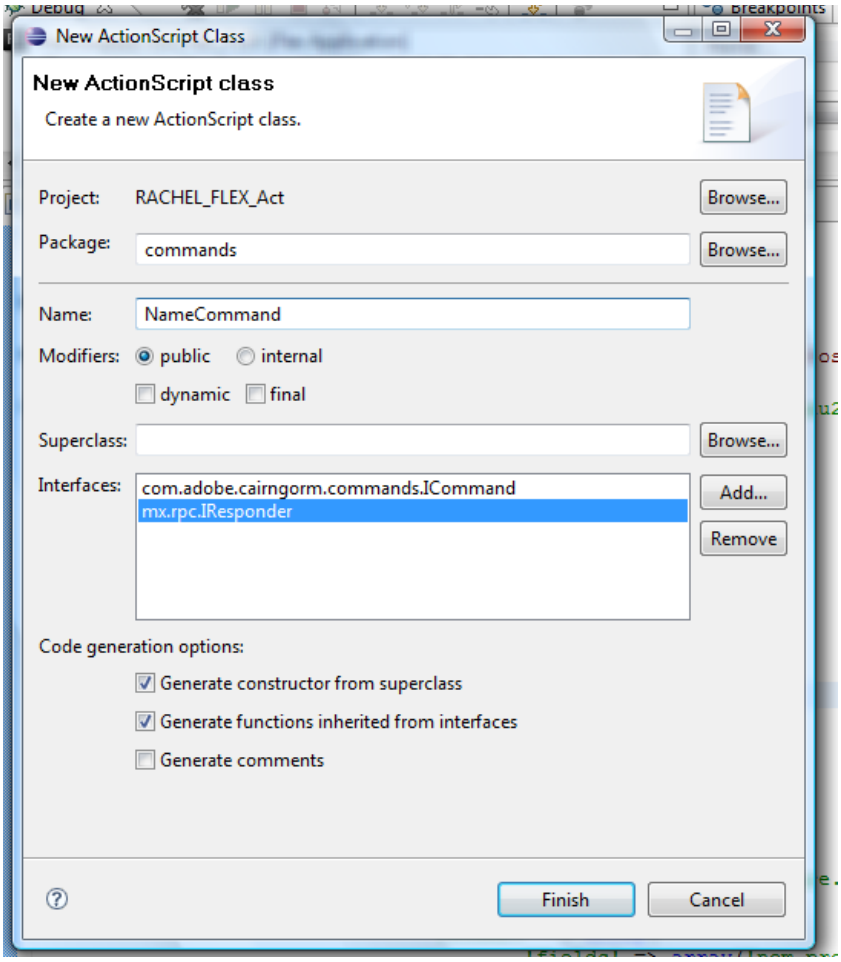
```
private var servicesControllerName:Object;

public function ServicesDelegate(responder:IResponder)
{
    this.responder = responder;
    this.servicesControllerName =
ServiceLocator.getInstance().getRemoteObject("servicesControllerName");
}
public function FunctionName (parametre1:String, parametre2:String):void{
    var call : Object = servicesControllerName. FunctionName (parametre1,
parametre2);
    call.addResponder(responder);
}
```

Fichier Command (étape 1)

Dans le répertoire Rachel_Flex > src > commands on créé une nouvelle Class ActionScript

Pour cette nouvelle Class on ajoute les interfaces Icommand (de cairngorm) et IResponder grâce au bouton « Add… »



Le nouveau fichier a cette forme là :

```
package commands
{
	import com.adobe.cairngorm.control.CairngormEvent;
	import mx.rpc.IResponder;
	import com.adobe.cairngorm.commands.ICommand;

	public class nameCommand implements ICommand, IResponder
	{
		public function nameCommand()
		{
		}
```

```
            public function execute(event:CairngormEvent):void
            {
            }

            public function result(data:Object):void
            {
            }

            public function fault(info:Object):void
            {
            }

      }
}
```

La fonction qui porte le même nom que la classe ne nous servira pas. On va tout d'abord travailler sur la fonction `execute` qui appelle la fonction php.

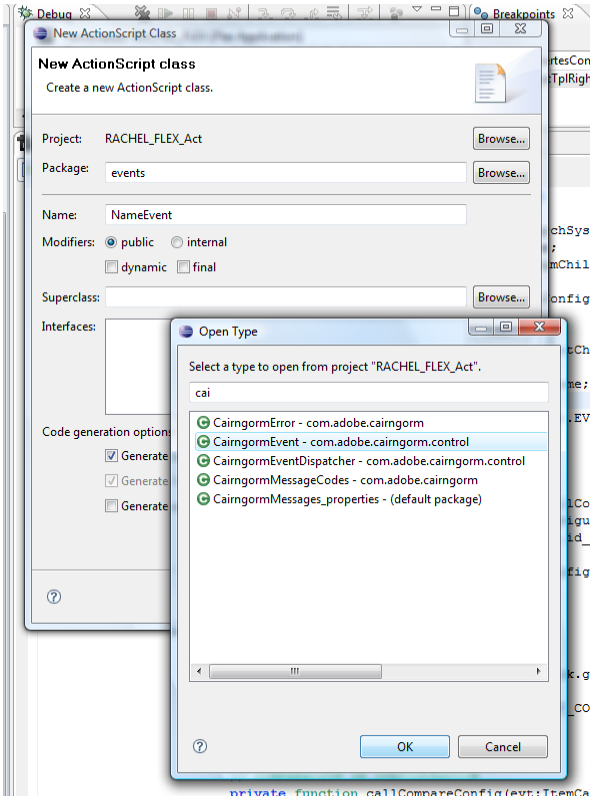
Voici les lignes de la fonction pour appeler une fonction définie précédemment :

```
public function execute(event:CairngormEvent):void
{
      var parametre1:String = "";
       var parametre2:String = "";

       var delegate:ServicesDelegate = new ServicesDelegate ( this );
      delegate.FunctionName(parametre1, parametre2);
}
```

La fonction « `execute` » va appeler le ServicesDelegate et lui passer les paramètres nécessaires. Le résultat arrivera dans la fonction « result ». En cas d'erreur, on tombe dans la fonction « `fault` ».

Fichier Event



Dans le répertoire `Rachel_Flex > src > events` on créé une nouvelle Class ActionScript

Pour cette nouvelle Class on ajoute la Superclass « CainrgormEvent » (champ Superclass, bouton « Browse »)

Le nouveau fichier a cette forme là :

```
package events
{
	import com.adobe.cairngorm.control.CairngormEvent;

	public class NameEvent extends CairngormEvent
	{
		public function NameEvent(type:String, bubbles:Boolean=false,
cancelable:Boolean=false)
		{
			super(type, bubbles, cancelable);
		}

	}
}
```

On va commencer par supprimer les paramètres bubbles et cancelable, puis on va ajouter une variable Static

⇨ `static public var EVENT_NAME:String = "myNameEvent";`

On va également prévoir les paramètres dont on a besoin. D'abord les paramètres nécessaires pour interroger la base de données : `parametre1` et `parametre2` puis un paramètre « `ui` » du type « `UIComponent` » qui contiendra une référence à l'objet qui attend la réponse (vue, composante…) Pour en savoir plus sur UiComponent, voir la doc en ligne.

Les paramètres sont passés à la fonction principale avec des valeurs par défaut au besoin

=> `public function NameEvent(type:String, parametre1:String = "", parametre2:String = "", ui:UIComponent = null)`

Ils sont également déclarés en variables public de la fonction.

```
=>    public var parametre1:String;
      public var parametre2:String;
      public var ui:UIComponent;
```

Ces variables sont mises à jours avec les valeurs passées en paramètre dans la fonction.

```
=>    this.parametre1 = parametre1;
      this.parametre2 = parametre2;
      this.ui = ui;
```

Au final, on obtient une classe qui ressemble à ça :

```
package events
{
      import com.adobe.cairngorm.control.CairngormEvent;

      import mx.core.UIComponent;

      public class NameEvent extends CairngormEvent
      {
            static public var EVENT_NAME:String = "myNameEvent";

            public var parametre1:String;
            public var parametre2:String;
            public var ui:UIComponent;

            public function NameEvent(type:String, parametre1:String = "",
```

```
parametre2:String = "", ui:UIComponent = null)
            {
                  super(type);
                  this.parametre1 = parametre1;
                  this.parametre2 = parametre2;
                  this.ui = ui;
            }

      }
}
```

Controller

Le lien entre la class NameCommand et la class NameEvent va se faire dans le controller « RELController.as » situé dans le répertoire suivant : `Rachel_Flex > src > control`

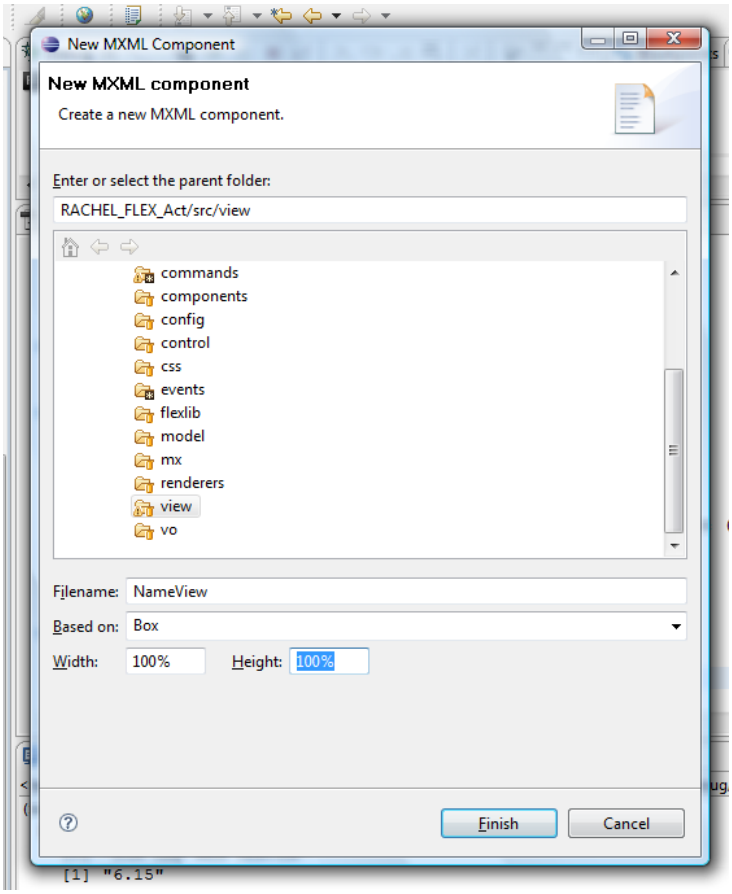
Il faut tout d'abord importer les deux classes puis les lier grâce à un « addCommand ».

```
import commands.NameCommand;
import events.NameEvent;

addCommand(NameEvent.EVENT_NAME, NameCommand);
```

De cette façon Cairngorm va lancer l'exécution de la command dès que l'event lié sera dispatché.

Fichier View



Dans le répertoire `Rachel_Flex > src > view` on créé un nouveau composant MXML basé sur « Box », width et Height à 100%

Le nouveau fichier a cette forme là :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Box xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%">


</mx:Box>
```

Voilà une forme de base adaptée au projet RACHEL. On a modifié l'entête pour mettre des marges de 30 pixels tout autour.

Ensuite, on ajoute une balise `<mx:Script>` qui contiendra le code As3.

En dessous, on a la partie visuelle de la vue avec les différents objets :

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Box xmlns:mx="http://www.adobe.com/2006/mxml" width="100%" height="100%"
     direction="vertical"
    paddingTop="30"
    paddingBottom="30"
    paddingLeft="30"
    paddingRight="30">

     <mx:Script>
          <![CDATA[
               import mx.collections.ArrayCollection;
               /*
               Zone des imports et des Script AS3
               */
               [Bindable]
               public     var     ListeResultats:ArrayCollection     =     new
ArrayCollection();

          ]]>
     </mx:Script>
     <!-- Zone pour les composants en mxml -->
     <mx:DataGrid dataProvider="ListeResultats">

     </mx:DataGrid>
</mx:Box>
```

Appel de la vue dans un nouvel Onglet

C'est dans le fichier principal de l'application RACHEL_FLEX.mxml que se fait l'appel des différents éléments visuels de l'application, en particulier l'appel d'un nouvel onglet.
La création d'un nouvel onglet est définie dans une fonction qui sera appelée au moment voulu.

```
private function AfficheNameView():void{
      // instanciation de la vue dans un nouvel onglet
      var myOnglet:NameView = new NameView();
      // définition du label de l'onglet
     myOnglet.label = "My Label";
```

```
      // le nouvel onglet est ajouté aux autres
      viewStack.addChild(myOnglet);
      // et il est sélectionné comme actif
      viewStack.selectedIndex = viewStack.getChildIndex(myOnglet);

      var parametre1:String = "value1";
      var parametre2:String = "value2";
      // envoi de l'event pour demander les données
      (new
NameEvent(NameEvent.EVENT_NAME,parametre1,parametre2,myOnglet)).dispatch()
}
```

Pour envoyer l'événement « `NameEvent`» et donc récupérer les données depuis CakePhp, la syntaxe est la suivante :

⇨ `(new NameEvent(NameEvent.EVENT_NAME,parametre1,parametre2,myOnglet)).dispatch() ;`

❖ *VO **ou** Value Object*

Le **VO** est un pattern de Cairngorm. Il va nous permettre d'encapsuler dans un objet, toutes les informations correspondant à un enregistrement dans la base de données. Un VO n'est rien de plus qu'un objet contenant un ensemble de propriétés (une colonne de notre table dans la BDD correspondant à une propriété de notre objet).

Dans le répertoire `Rachel_Flex > src > vo` on créé une nouvelle Class ActionScript puis on ajoute en dehors de la fonction les déclarations de variables qui sont les propriétés du vo.

Exemple :

```
package vo
{
      public class Exemple_vo
      {
            public function Exemple_vo()
            {
            }

            public var nom_parametre:String = "";
            public var description_parametre:String = "";
            public var visible:uint = 0;
      }
}
```

Fichier Command (étape 2)

Pour passer les paramètres depuis l'appel de la vue jusqu'à Cakephp, on va faire évoluer notre classe Command. L'event lié est instancié pour récupérer les valeurs des paramètres. De même, on garde dans une variable locale le UIComponent auquel il faudra transmettre la réponse.

```
public class NameCommand implements ICommand, IResponder
{
private var parametre1:String ="";
private var parametre2:String ="";
private var ui:UIComponent;

public function execute(event:CairngormEvent):void
{
      //instanciation de l'event lié
      var e:NameEvent = event as NameEvent;

      //récupération des données
      parametre1 = e.parametre1;
      parametre2 = e.parametre2;
      ui = e.ui;

      var delegate:ServicesDelegate = new ServicesDelegate ( this );
      delegate.FunctionName(parametre1, parametre2);
}
```

Pour passer les paramètres depuis l'appel de la vue jusqu'à Cakephp, on va faire évoluer notre classe.

Le résultat de la fonction « `execute`» arrive dans la fonction « `result` » de Command. Les données sont contenues dans la vairable data passée en paramètre, précisément dans `data.result` qui contient le tableau des résultats.

```
public function result(data:Object):void
{
       var ListeResultats:ArrayCollection = new ArrayCollection();

      for each(var o:Object in data.result)
      {
            // Création de la ligne à partir du vo
            var elt:Exemple_vo = new Exemple_vo();
```

```
            // Remplissage des valeurs grâce aux données reçues
            elt.nom_parametre = o.param.nom_parametre;
            elt.description_parametre = o.param.description_parametre;
            elt.visible = o.param.visible;

            // Ajout de la ligne au tableau de résultat
            ListeResultats.addItem(elt);
      }

      // Transmission à la vue
      (ui as NameView).ListeResultats = ListeResultats;
}
```

En cas d'erreur dans ce que transmet le ServicesDelegate, c'est la fonction « `fault` *» qui est appelée. Pour savoir en que l'exécution a échoué, on pourra mettre un «* `trace` *» et un point d'arrêt et phase de développement.*

```
public function fault(info:Object):void
{
      trace("erreur NameCommand");
}
```

Ergonomie de l'interface IHM

La charte PSA

Tous les sites intranet PSA doivent suivre la charte PSA. Une charte PSA est déjà définie et doit être identique pour tous les sites intranet PSA. Seuls les couleurs de telle charte peuvent être modifiés selon les besoins et envies, tout en respectant l'utilisation maximum de 4 couleurs.

Voici à quoi ressemble la charte PSA mise en place pour la création de site intranet :

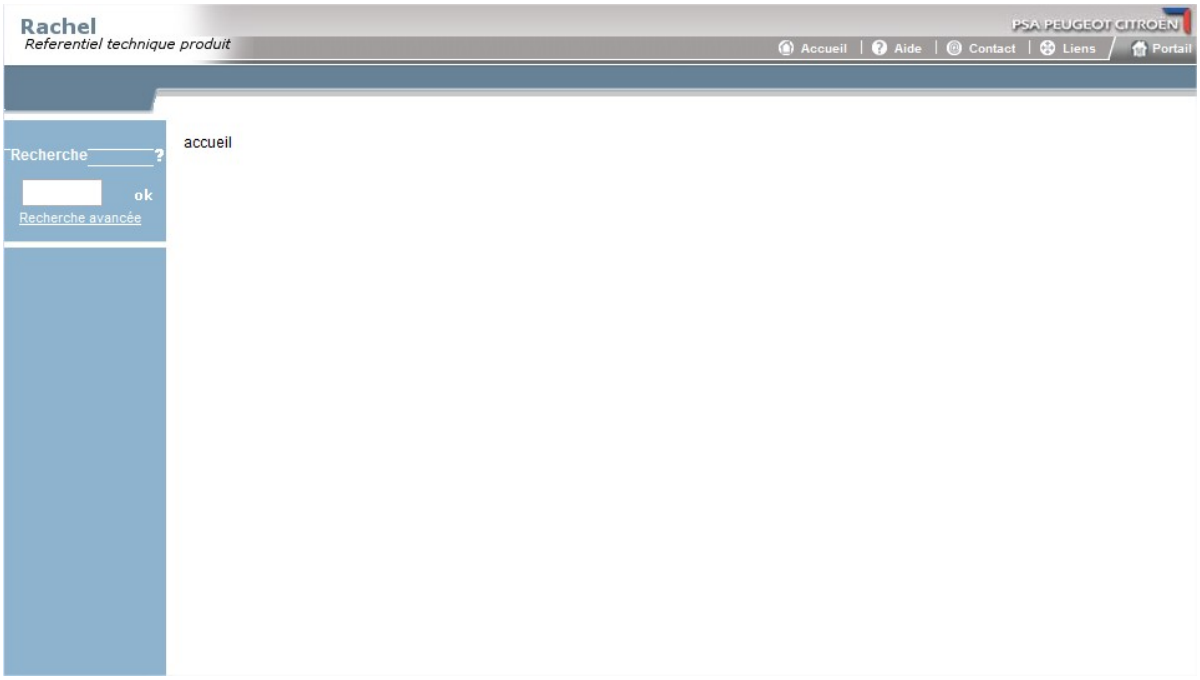


Figure 25 : Exemple de la charte PSA

Or dans le cadre de notre projet, la charte PSA pour les sites Intranet développés en html ne s'adaptait pas à une application Flex. Nous avons donc eu « carte blanche » pour proposer une nouvelle charte PSA qui serait mis à disposition des autres sites intranet du groupe développés dans la filière Flex.

Certes, nous n'avons reçu aucune limitation pour l'élaboration de cette nouvelle charte, cependant un cahier des charges des éléments indispensables nous a été fourni. Nous pensons notamment à l'utilisation du logo du groupe, ou bien à la mise en place d'un lien vers le portail de communication du groupe. Une fois toutes ces informations prises en compte nous avons pu travailler sur l'élaboration de cette nouvelle charte.

Pour donner un ton encore plus dynamique et stylisé du référentiel, nous avons décidé d'élaborer pour celui-ci un logo.

Elaboration d'un logotype Nous avons certes eu « carte blanche »

Notre nouveau référentiel se devait d'avoir son propre logotype. Celui-ci se devait dynamique, à l'image de l'utilisation d'une technologie innovante comme celle de Flex. De nombreuses maquettes du logo (voir ci-dessous) ont été proposées au comité de validation qui devait choisir celle qui servirait de logo au référentiel Rachel.

Figure 26 : Logo n°1

Figure 27 : Logo n°2

Figure 28 : Logo n°3

Le choix du comité de validation s'est porté sur le logo n°3. Quelques modifications ont été apportées pour le rendre plus lisible lorsqu'il est de taille plus réduite. Une nouvelle version a été proposée, validée et retenue pour devenir le logotype du référentiel Rachel. Voici donc la dernière version du logo :

Figure 29 : Logo RACHEL

Les différentes sections du référentiel

Selon les besoins exprimés lors de l'étude préalable, nous avons divisé le référentiel en huit grands tiroirs, chaque tiroir correspond à un besoin concret:

❖ Module de recherche :

C'est sur cette section que l'utilisateur sera dirigé lors de l'accès au site intranet juste après son authentification. Il accèdera à un module où plusieurs champs de formulaire lui permettront de rechercher la configuration qu'il souhaite visualiser. Dans le champ supérieur, celui-ci pourra procéder à la recherche d'une configuration en fonction de différents paramètres de recherches :

- Le nom du serveur
- Le nom du produit filtré sur la version de celui-ci et l'OS

Le deuxième formulaire de recherche lui permettra de comparer la configuration de deux serveurs, nous proposons à l'utilisateur de saisir deux noms de serveur distincts pour pouvoir les comparer l'un à l'autre. Afin d'aider l'utilisateur lors de la saisie des champs, une fonction d'auto completion a été mise en place sur chacun des champs du formulaire. Cette fonction complète automatiquement un champ de saisi lors que l'utilisateur commence a taper sur son clavier. Ce module de recherche à l'accueil du site intranet ressemble à cela:

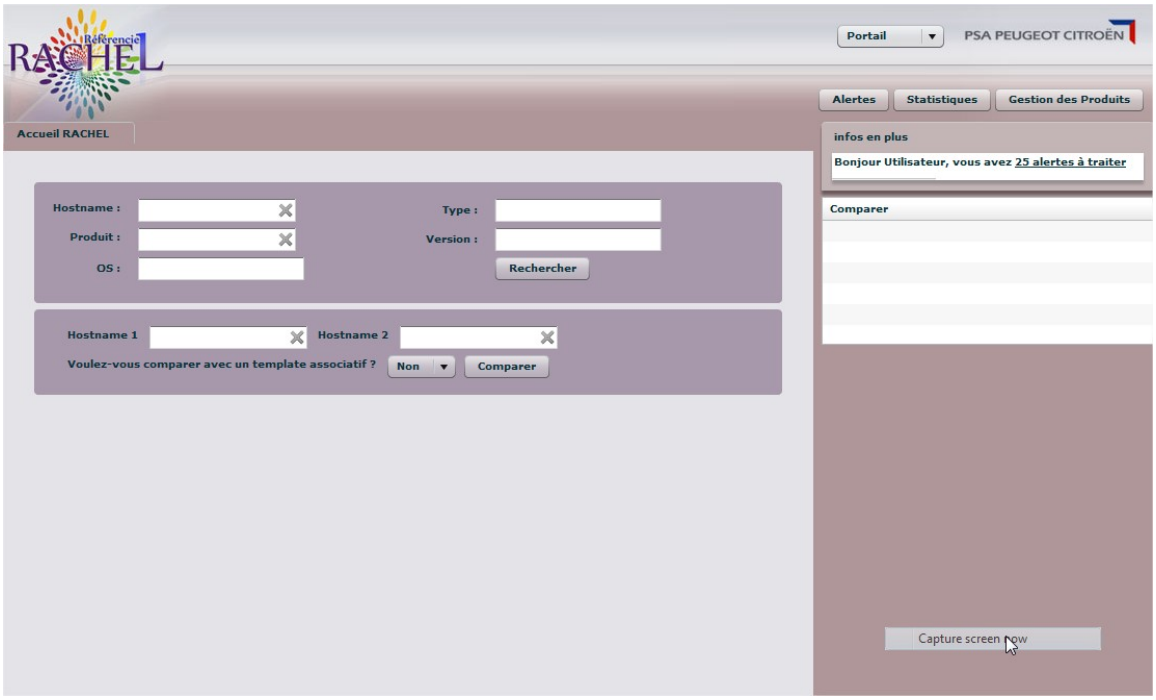


Figure 30 : Module de recherche

❖ Affichage d'une configuration :

Dans cette section nous allons pouvoir afficher le contenu d'une configuration que nous avons au préalable sélectionnée. Celle-ci va mettre en forme les multiples paramètres remontés au référentiel pour chaque configuration. Afin de catégoriser la configuration et ainsi la rendre plus lisible pour l'utilisateur, celle-ci est classée dans plusieurs niveaux d'arborescences.

Le premier niveau correspond au nom de l'instance du produit à laquelle le paramètre est rattaché. Par la suite nous afficherons les différentes familles de paramètres qui ont été recensées dans la configuration. Enfin à l'intérieur de celle-ci, nous allons pouvoir retrouver le nom de chaque paramètre associé à sa valeur. En survolant le paramètre nous aurons accès à la description longue de ce paramètre.

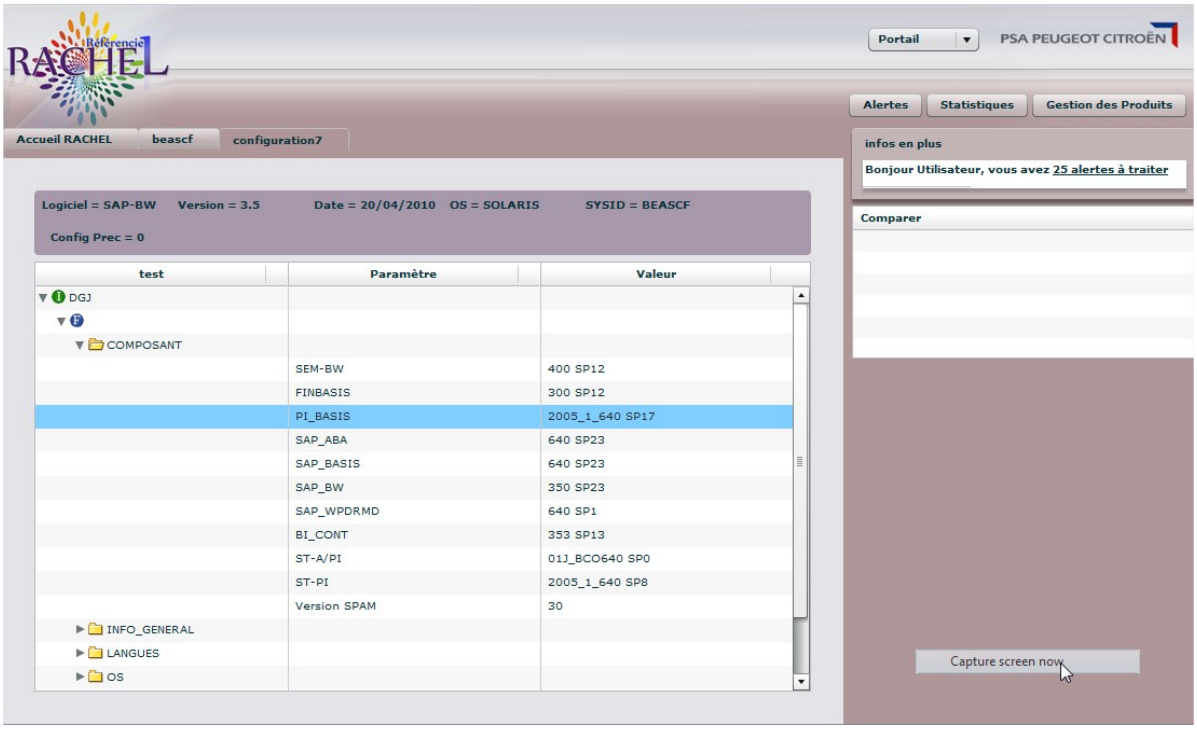


Figure 31 : Affichage d'une configuration

❖ Comparaison d'une configuration :

Ce module est une extension du module d'affichage d'une configuration, l'utilisateur qui va consulter plusieurs configurations aura besoin de visualiser les différences entre deux versions de configuration. Cette comparaison ne sera possible que pour comparer deux configurations d'un même produit, toute autre comparaison de configuration de produit différent n'aurait aucun sens.

Pour mettre en évidence rapidement les différences entres deux configurations, nous avons décidé d'afficher à l'utilisateur uniquement les paramètres présentant des valeurs de paramètres différentes. Vous trouverez un exemple ci-dessous :

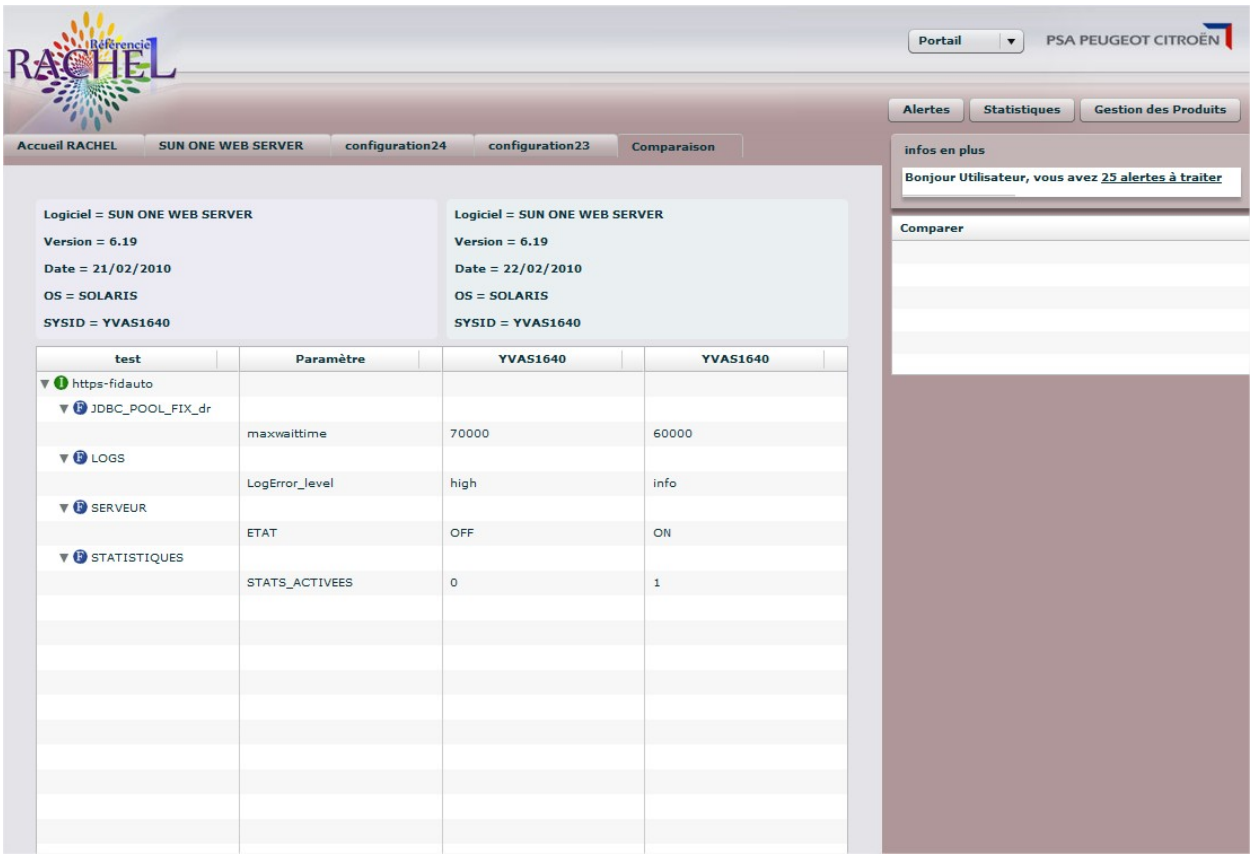


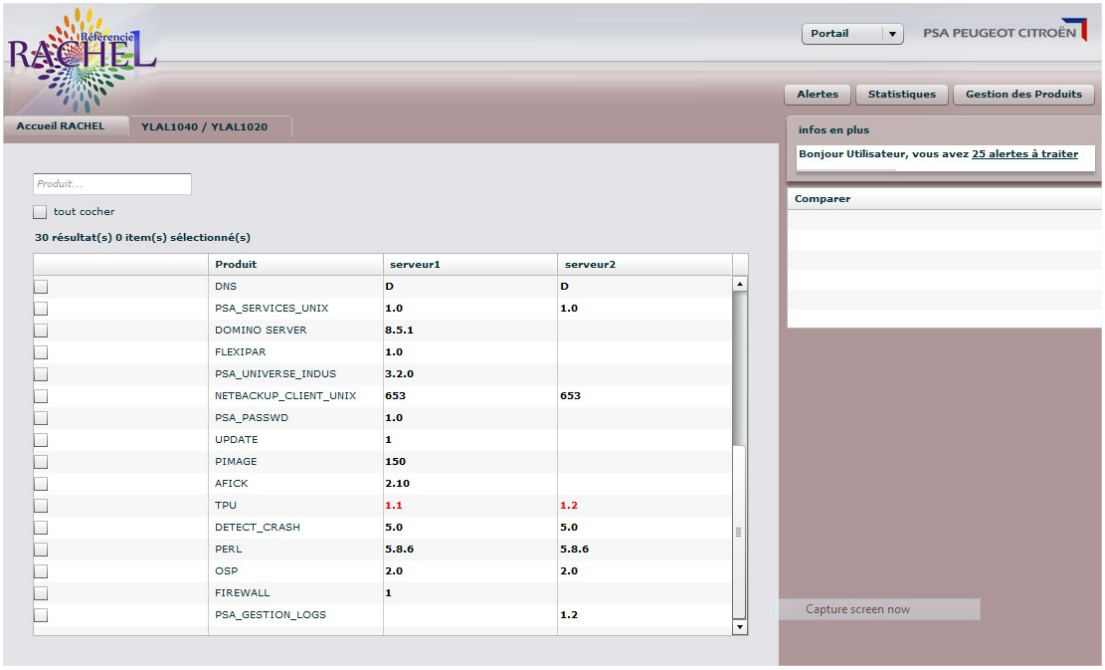
Figure 32 : Comparaison d'une configuration de produit

Il sera ainsi facile pour un responsable produit de visualiser quels paramètres ont étés modifiés d'un jour à l'autre. Prenons un exemple concret d'utilisation de ce module .Nous avons un incident sur ce produit et suite à celui-ci, il est impossible de redémarrer le produit. Bien entendu, les utilisateurs du produit n'ont modifié aucun paramètre, enfin c'est ce que ceux-ci expliquent auprès du responsable produit. Le responsable produit n'étant pas convaincu, décide d'utiliser son tout nouveau référentiel technique produit afin de visualiser si aucun paramètre n'a été modifié ces derniers jours. A la stupeur générale, il se rend compte qu'un paramètre important permettant la

relance du produit avait été modifié la semaine précédente. Ouf !!!! Heureusement que le référentiel RACHEL était là, le responsable produit a pu modifier la valeur de ce paramètre et relancer son produit en un temps très inférieur à ce qui aurait pu être sans la mise en évidence d'une telle modification.

❖ Comparaison de serveur :

Dans cette section, nous comparons la configuration produit de deux serveurs. Après avoir sélectionnés les deux serveurs que nous souhaitons rechercher, l'interface va nous afficher chaque produit installé sur ce serveur ainsi que la version. Ces informations sont extraites des tables infrastructures et produits, elles-mêmes extraites du référentiel REFLEX vu dans le chapitre 3.2. Les différences de version entre deux serveurs seront mises en valeur en utilisant la couleur rouge. Certaines différences de version seront normales, d'autres beaucoup moins. Pour chaque configuration produit présent dans le référentiel RACHEL, l'utilisateur pourra cliquer sur la version du produit pour afficher la configuration active de ce produit sélectionné sur le serveur demandé.



	Produit	serveur1	serveur2
☐	DNS	D	D
☐	PSA_SERVICES_UNIX	1.0	1.0
☐	DOMINO SERVER	8.5.1	
☐	FLEXIPAR	1.0	
☐	PSA_UNIVERSE_INDUS	3.2.0	
☐	NETBACKUP_CLIENT_UNIX	653	653
☐	PSA_PASSWD	1.0	
☐	UPDATE	1	
☐	PIMAGE	150	
☐	AFICK	2.10	
☐	TPU	1.1	1.2
☐	DETECT_CRASH	5.0	5.0
☐	PERL	5.8.6	5.8.6
☐	OSP	2.0	2.0
☐	FIREWALL	1	
☐	PSA_GESTION_LOGS		1.2

Figure 33 : Comparaison de serveur

❖ **Gestion des produits :**

Ce module sert d'administration des produits, chaque responsable produit va ainsi pouvoir administrer les produits dont il a la charge. Grâce à celui-ci nous allons pouvoir activer ou désactiver la surveillance d'un produit par le référentiel RACHEL. Lorsqu'un produit sera surveillé par le référentiel, RACHEL générera des alertes sur le produit en cas d'anomalie avec le référentiel REFLEX. Concernant les configurations des produits, par ce module, le responsable produit pourra ajouter une description longue à un paramètre. Cette description longue apparaît dans l'interface IHM lors du survol d'un paramètre, c'est donc une aide pour l'utilisateur qui ne sait pas à quoi fait référence chaque nom de paramètre (Voir exemple ci-dessous avec le paramètre port)

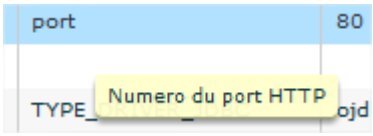


Figure 34 : InfoBulle sur la description d'un paramètre

Dans ce module, le responsable du produit pourra également désactiver la visibilité de certains paramètres dans l'affichage des configurations, cela afin d'épurer la vue d'une configuration produit qui parfois apparaît trop confuse. Enfin la dernière fonctionnalité possible est la création d'un template produit. Le responsable produit pourra ainsi définir une configuration modèle de son produit. Il sera ainsi possible à tout moment de comparer une configuration lambda avec cette configuration modèle appelée template produit.

❖ Statistiques :

Le module statistique propose aux utilisateurs du référentiel de mettre sous forme de graphique certaines informations exploitées par le référentiel RACHEL. Nous pouvons par exemple suivre le nombre de produits installés en fonction du temps, suivre le nombre de licences utilisées par un produit sur tout le parc. Des graphiques sur les valeurs des paramètres d'une configuration pourront également être mise en place. L'idée de ce module est de proposer à l'utilisateur un configurateur de statistique. Celui-ci pourra ainsi demander un graphique selon ses besoins et l'extraire au format image pour une éventuelle réutilisation dans une présentation. Nous vous proposons un exemple de graphique généré par le référentiel RACHEL.

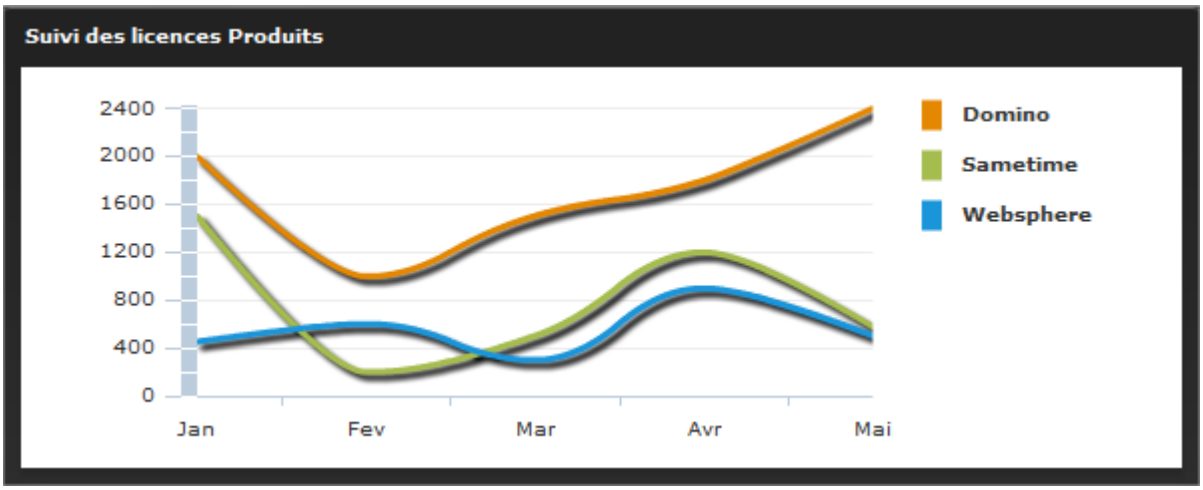


Figure 35 : Graphique dans le module statistique

❖ Gestion des alertes :

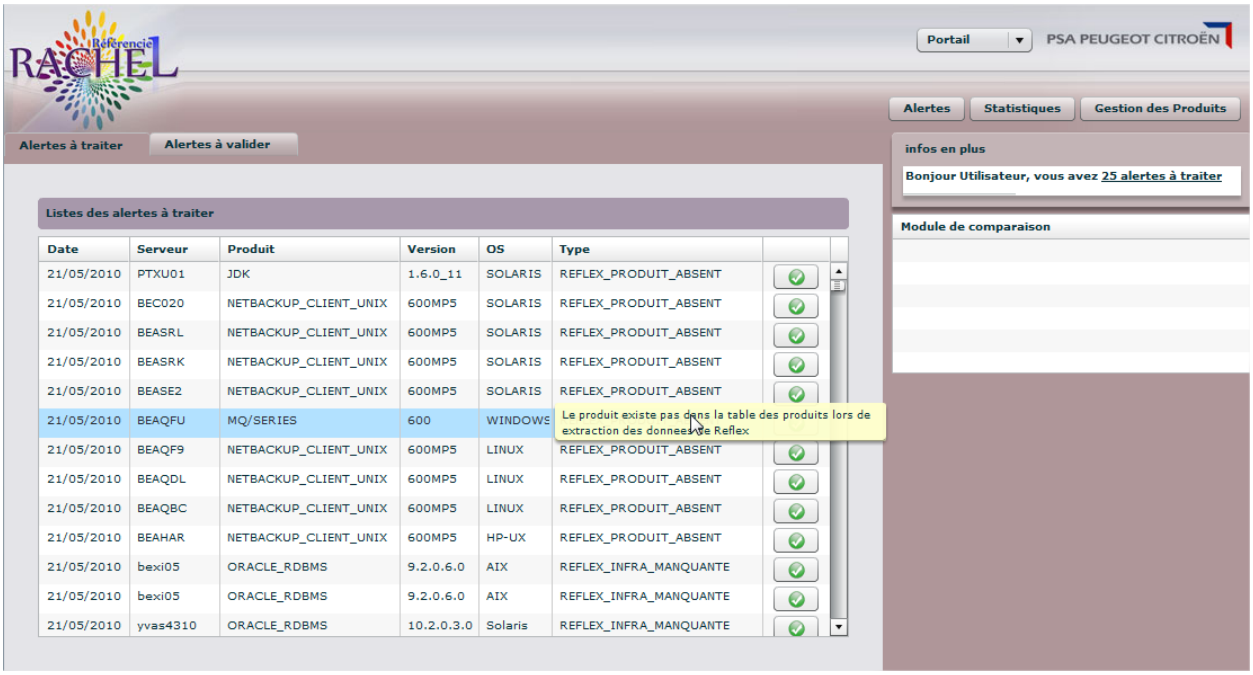
Ce module nous permet de gérer les différentes alertes remontées par le référentiel RACHEL. Certaines seront d'ordre administratif et seront visibles uniquement par les administrateurs du référentiel. D'autres seront des irrégularités remontées pour cause d'anomalie avec le référentiel REFLEX. En effet, nous avons vu précédemment que le référentiel REFLEX était mise à jour manuellement, c'est-à-dire que chaque installation ou désinstallation de produit entraine la mise à jour de façon manuelle par le réalisateur de l'action du référentiel REFLEX. Quant à RACHEL, il reçoit des informations directement depuis les serveurs pour mettre à jour son référentiel. Deux sortes d'alertes vont ainsi pouvoir être analysées :

- Produit non présent dans REFLEX :

Il s'agit du cas d'une configuration produit qui est remontée au serveur RACHEL alors que d'après Reflex, ce produit n'est pas installé sur le serveur en question. Une alerte sera ainsi affectée au responsable du produit en question, celui-ci devra ajouter l'installation de ce produit dans le référentiel REFLEX de façon manuelle. Il validera l'alerte RACHEL depuis l'interface du référentiel RACHEL qui passera au statut RESOLVED. Dans le traitement batch suivant, c'est-à-dire la nuit suivante, le référentiel RACHEL vérifiera le traitement de l'alerte par l'utilisateur et passera l'alerte au statut CLOSED si celle-ci a correctement été traitée.

- Pas de remontée de config pour un produit sur un serveur :

Il s'agit du cas contraire, nous sommes en attente d'une configuration produit d'un serveur. En effet d'après REFLEX, le produit est installé sur ce serveur mais aucun fichier de configuration ne remonte au serveur RACHEL. Une alerte sera donc affectée au responsable du produit afin de vérifier que le produit soit bien présent sur le serveur. Si ce n'est plus le cas, il devra supprimer l'installation de ce produit sur le référentiel REFLEX et valider le traitement de l'alerte RACHEL.



Date	Serveur	Produit	Version	OS	Type
21/05/2010	PTXU01	JDK	1.6.0_11	SOLARIS	REFLEX_PRODUIT_ABSENT
21/05/2010	BEC020	NETBACKUP_CLIENT_UNIX	600MP5	SOLARIS	REFLEX_PRODUIT_ABSENT
21/05/2010	BEASRL	NETBACKUP_CLIENT_UNIX	600MP5	SOLARIS	REFLEX_PRODUIT_ABSENT
21/05/2010	BEASRK	NETBACKUP_CLIENT_UNIX	600MP5	SOLARIS	REFLEX_PRODUIT_ABSENT
21/05/2010	BEASE2	NETBACKUP_CLIENT_UNIX	600MP5	SOLARIS	REFLEX_PRODUIT_ABSENT
21/05/2010	BEAQFU	MQ/SERIES	600	WINDOWS	REFLEX_PRODUIT_ABSENT
21/05/2010	BEAQF9	NETBACKUP_CLIENT_UNIX	600MP5	LINUX	REFLEX_PRODUIT_ABSENT
21/05/2010	BEAQDL	NETBACKUP_CLIENT_UNIX	600MP5	LINUX	REFLEX_PRODUIT_ABSENT
21/05/2010	BEAQBC	NETBACKUP_CLIENT_UNIX	600MP5	LINUX	REFLEX_PRODUIT_ABSENT
21/05/2010	BEAHAR	NETBACKUP_CLIENT_UNIX	600MP5	HP-UX	REFLEX_PRODUIT_ABSENT
21/05/2010	bexi05	ORACLE_RDBMS	9.2.0.6.0	AIX	REFLEX_INFRA_MANQUANTE
21/05/2010	bexi05	ORACLE_RDBMS	9.2.0.6.0	AIX	REFLEX_INFRA_MANQUANTE
21/05/2010	yvas4310	ORACLE_RDBMS	10.2.0.3.0	Solaris	REFLEX_INFRA_MANQUANTE

Figure 36 : Module de gestion des alertes

❖ Administration :

Dans le référentiel RACHEL nous n'avons pas de module proprement dédié à l'administration. Par le bief d'un service applicatif attribué aux administrateurs du référentiel, ceux-ci aurons accès au même module que ceux décrit dans le module gestions de produit et alertes à la seule différence que les administrateurs auront une vision sur tous les produits du parc ainsi que sur toutes les alertes générés par le référentiel.

CONCLUSION

Situation actuelle.

Remonter la configuration de tous les produits d'infrastructures au sein d'un même référentiel était un projet audacieux, la plus grande difficulté était de définir une procédure homogène pour chaque produit.

La grande diversité des produits utilisés par le groupe PSA m'a obligé à mettre en place des procédures les plus homogènes possible afin d'être exploitable sur le long terme. En effet, chaque produit ne se comporte pas de la même manière mais surtout chacun d'entre eux possède son propre type de configuration. Il a donc été nécessaire de recenser et prévoir la collecte des différentes architectures de produit, nous pensons notamment aux produits gérant les serveurs d'applications. Ces produits sont installés une seule fois par serveur mais peuvent créer un nombre non défini d'instanciation de celui-ci. Afin de visualiser correctement la configuration d'un produit de ce type, nous avons du définir une procédure réutilisable par d'autres produit n'utilisant pas d'instanciation de produit. Une autre difficulté à laquelle je me suis confrontés est la diversité des utilisateurs de ce projet. Ce référentiel est amené à être utilisé par une centaine d'utilisateurs, chacun d'entre eux ayant leurs propres besoins vis-à-vis de leurs métiers. Il nous a donc été très difficile de satisfaire tous les besoins, il m'a dont fallut parfois proposé des solutions de contournement pour garder une homogénéité dans le développement de ce référentiel.

Evolution à venir

Débuté en septembre 2009, le projet a été planifié jusqu'en Décembre 2010. A l'heure actuelle, le développement des différents modules de l'interface IHM sont en cours de finalisation, le site intranet va ainsi pouvoir être mis en pré-production fin août 2010. Cette première version du référentiel dans laquelle nous allons retrouver toutes les fonctionnalités identifiées dans le cahier des charges va être alimentée par les huit produits pilotes (Sun One Web Server, Glassfish, Websphere, Oracle, CFT, $Universe, Columbus). Ces fonctionnalités vont ainsi pouvoir être testées dans un environnement de pré-production dans lequel chaque fonctionnalité sera testée par les responsables produits. Ces derniers vont pouvoir tester les différentes fonctionnalités de l'interface Web du référentiel (Consultation de configuration, comparaison avec une autre configuration, gestion des templates, gestion des alertes, génération de statistique) mais cela va également être l'occasion de

tester à plus grande échelle les traitements batch de fond mis en œuvre pour exploiter tous les fichiers de configurations collectés chaque jour sur le serveur central.

Bilan personnel

Ce fut ma première expérience professionnelle en qualité de chef de projet informatique, une mission à la fois enrichissante et pleine de responsabilité. Je me suis confronté avec la responsabilité de répondre aux besoins de mes utilisateurs dans un délai imparti. Au début du projet, je me suis rapidement retrouvé avec une multitude de besoins identifiés auprès des responsables de produit sans pour autant savoir comment faire pour les satisfaire, ni par ou commencer. J'ai réussi à traiter tous les problèmes de front en me basant sur la démarche projet de cycle en V. J'ai ainsi pu documenter un vrai cahier des charges que j'ai exploité tout au long du projet. L'utilisation de cette démarche projet m'a également permis d'élaborer un planning complet des différentes étapes de la vie du projet avec des délais à respecter afin de pouvoir livrer mon projet en temps et en heure.

Une autre facette du projet qui m'a plu était le libre choix de la technologie informatique à utiliser pour le développement du projet. J'ai du étudier différentes possibilités afin de choisir le langage de programmation qui m'a parut la plus approprié à mon projet. De plus la technologie utilisée pour l'interface IHM (Flex) était innovatrice chez PSA, ce qui a rendu ce projet d'autant plus agréable. Mais attention, le fait de choisir une technologie innovatrice m'a confronté à de nombreuses contraintes. Par exemple, aucun support d'aide n'était disponible au développeur sur ce type de technologie, une filière de développement est tout de même née en début d'année mais celle-ci doit encore acquérir de l'expérience pour être exploitable pour les développeurs d'application web.

Pour les besoins du projet, nous avions identifié en amont le besoin de faire appel à une prestation extérieure de 4 mois pour m'aider dans le développement du projet. J'avais décidé de lui donner pour mission le développement de plusieurs modules de l'interface IHM en utilisant la technologie FLEX. Hors, là encore je me suis confronté a de nombreuses difficultés car cette prestataire ne connaissait pas la technologie Flex, celle-ci a donc du s'approprier cette technologie en début de mission. Ces difficultés ont donc généré du retard dans le développement du projet auxquels j'ai du m'adapter. A l'heure actuelle, la prestation arrive à son terme au 15 Juillet, toutes les fonctionnalités dont j'avais demandées le

développement à la prestataire ne seront pas réalisées à temps. J'ai donc du adapter mon planning et prendre en charge le développement de certains modules comme le module des alertes et celui des statistiques. J'en ai ainsi profité pour m'auto former sur cette nouvelle technologie.

Le développement du projet arrive à son terme mais de nouveaux besoins arrivent déjà jusqu'à moi par la maitrise d'ouvrage comme par exemple la possibilité, à partir du référentiel, d'aller configurer sur un serveur un produit de façon automatique. La mise en production de ce nouveau référentiel, tous les utilisateurs vont pouvoir apprécier les fonctionnalités de celui-ci mais également détecter et demander de nouveaux besoins. Nous pouvons donc déjà imaginer une deuxième version du référentiel RACHEL.

Tables des figures

Tables des tableaux

Bibliographie

[REF01] http://www.cakephp-fr.org/ :

Site communautaire sur le FrameWork Cake PHP

Date de Consultation : Mars 2010

[REF02] http://blog.kapit.fr/ria/2007/04/11/utiliser-cairngorm-1/ :

Explication du fonctionnement du framework Cairngorm

Date de Consultation : Avril 2010

[REF03] http://www.flex-tutorial.fr/ :

Site communautaire sur Flex

Date de Consultation : Avril 2010