



## Rapport de stage

Développement d'une application JEE  
pour une marque de prêt-à-porter

Sylvain Auzanne

Maître de stage : M. Christophe Le Du

Tuteurs enseignants : M. Gerson Sunyé, M. Christian  
Attiogbé

5 Septembre 2008

## Table des matières

<b>I</b>	<b>Remerciements</b>	<b>5</b>
<b>II</b>	<b>Introduction</b>	<b>6</b>
<b>III</b>	<b>Glossaire</b>	<b>7</b>
<b>IV</b>	<b>Présentation de Steria</b>	<b>10</b>
<b>1</b>	<b>Le groupe STERIA</b>	<b>10</b>
1.1	Les implantations de STERIA . . . . .	10
1.2	Les missions de STERIA . . . . .	11
1.3	Quelques chiffres . . . . .	11
<b>2</b>	<b>Les centres de services</b>	<b>13</b>
2.1	Les différentes générations des centres de services . . . . .	13
2.2	L'organisation des centres de services . . . . .	14
2.2.1	Front Office (FO) . . . . .	14
2.2.2	Back Office (BO) . . . . .	15
2.3	La répartition géographique des centres . . . . .	15
2.4	Les outils mis en oeuvre . . . . .	16
2.5	Exemples de projets gérés par le centre de services de Nantes . . . . .	18
<b>3</b>	<b>Le centre de services TMA de Nantes/ST Herblain</b>	<b>18</b>
3.1	Présentation . . . . .	18
3.2	Son rôle . . . . .	18
3.3	Son organisation . . . . .	19
<b>V</b>	<b>Le stage</b>	<b>20</b>
<b>4</b>	<b>Le sujet de stage</b>	<b>20</b>
<b>5</b>	<b>Mes objectifs</b>	<b>20</b>
<b>6</b>	<b>Le projet</b>	<b>21</b>
6.1	Présentation . . . . .	21
6.2	Les besoins . . . . .	23
6.3	La maîtrise des risques . . . . .	24
6.4	La gestion du projet . . . . .	25
6.5	Le chiffrage du projet . . . . .	26

6.6	Le planning du projet . . . . .	30
6.6.1	L'avant vente . . . . .	31
6.6.2	La phase d'études . . . . .	33
6.6.3	La phase de réalisation . . . . .	35
6.6.4	Mesures de la qualité de l'application . . . . .	36
6.7	L'architecture matérielle . . . . .	37
6.7.1	L'architecture matérielle de Steria . . . . .	37
6.7.2	L'architecture matérielle de Camaïeu . . . . .	38
6.8	L'architecture logicielle . . . . .	40
<b>7</b>	<b>Ma mission</b>	<b>43</b>
7.1	Le planning réalisé . . . . .	43
7.2	Le travail effectué . . . . .	48
7.2.1	La lecture de documents . . . . .	48
7.2.2	Etude d'un ETL : Oracle Data Integrator . . . . .	49
7.2.3	Le développement de l'application . . . . .	50
7.2.4	Développement de l'écran gestion des liens catégorie / taille magasins . . . . .	50
7.2.5	Développement de l'écran gestion des aléas entrepôt . . . . .	56
7.2.6	Les difficultés rencontrées . . . . .	63
<b>VI</b>	<b>Conclusion</b>	<b>65</b>
<b>VII</b>	<b>Annexes</b>	<b>66</b>
<b>8</b>	<b>Annexe 1 : Les outils de qualimétrie et de tests</b>	<b>66</b>
8.1	Cast : outil de qualimétrie . . . . .	66
8.2	Test Director (TD) . . . . .	67
8.3	La PIC (Plateforme d'Intégration Continue) . . . . .	67
<b>9</b>	<b>Annexe 2 : Les principaux frameworks JEE</b>	<b>69</b>
9.1	Struts . . . . .	69
9.2	Spring . . . . .	71
9.3	Hibernate . . . . .	74
9.4	Tiles . . . . .	75
<b>10</b>	<b>Annexe 3 : Outil de transformation de données</b>	<b>78</b>
10.1	ODI (Oracle Data Integrator) . . . . .	78
10.1.1	Créations des utilisateurs sur la base Oracle . . . . .	78
10.1.2	Création du Master Repository . . . . .	79
10.1.3	Connexion au référentiel . . . . .	80
10.1.4	Associer un Référentiel de travail au Master Repository . . . . .	81
10.1.5	Utilisation du designer . . . . .	83

<i>Rapport de stage</i>	4
<b>11 Annexe 4 : Sommaire de la proposition commerciale de Steria</b>	<b>85</b>
<b>12 Annexe 5 : sommaire du DAT</b>	<b>86</b>
<b>13 Annexe 6 : sommaire du DAL</b>	<b>87</b>
<b>14 Annexe 7 : sommaire du PPL</b>	<b>88</b>

## Première partie

# Remerciements

Je tiens à remercier M. Christophe Le Du, mon maître de stage, en sa qualité de Chef de Projet pour m'avoir accueilli au sein de Steria. Je tiens à le remercier chaleureusement pour la disponibilité et la confiance qu'il m'a apporté tout au long de mon stage.

Je souhaite remercier mon Directeur de Projet, M. Desiré Noumowé pour sa présence et sa disponibilité tout au long de mon stage.

Je tiens également à remercier toute l'équipe du projet Camaïeu, et tout particulièrement M. Franck, M. Miri, M. Lounes, M. Colin, M. Mace et M. Leduc, pour m'avoir donné l'assistance nécessaire à la réalisation et à l'aboutissement de ma mission.

Enfin, je remercie tous les collaborateurs de la société pour leur accueil chaleureux, et pour l'aide qu'ils ont pu m'apporter tout au long de mon stage.

## Deuxième partie

# Introduction

Dans le cadre de mon Master 2 ALMA, un stage de fin d'études de 5 mois est intégré au cursus. Ce stage s'est déroulé du 1er avril 2008 au 31 août 2008.

J'ai effectué mon stage au sein de la SSII Steria. La société est localisée à Saint Herblain (44) où elle dispose d'un centre de production.

Dès le début de mon stage, j'ai été affecté à une équipe composée de cinq développeurs et d'un chef de projet, sur le projet CPS commandée par la société Camaïeu. Au cours de ce stage il y a eu une forte montée en charge puisque l'équipe est passée de 6 à 14 personnes. Le projet a débuté en Novembre 2007 et est prévu pour être mis en production (pour une partie seulement) en Septembre 2008.

Ce projet consiste à développer une application pour la chaîne de prêt-à-porter Camaïeu. Cette application se dénomme CPS (Camaïeu Procurement System) ce qui signifie Système d'approvisionnement de Camaïeu. Comme son nom l'indique, cette dernière consiste à gérer l'approvisionnement des magasins de cette chaîne de prêt à porter, en prenant en compte divers paramètres, tels que le nombre d'articles vendus, le nombre de produits en stock, l'arrivée de nouvelles collections, la période commerciale (soldes) etc.

J'avais pour mission la spécification et la réalisation des différents écrans de l'application et je me suis occupé principalement de la partie présentation. Vers la fin de mon stage j'ai également eu l'occasion de m'occuper des aspects métiers.

Je vous ferais une présentation détaillée du groupe Steria, et plus particulièrement du site nantais, du projet Camaïeu et de ma mission. Puis nous étudierons les différentes phases du projet que j'ai dû étudier avant d'apporter ma contribution au développement de l'application. Enfin, je terminerai par la description du travail que j'ai effectué au cours de ce stage.

## Troisième partie

# Glossaire

**CPS-V2** : Camaïeu Procurement System Version 2 (Système d'approvisionnement de Camaïeu)

**TMA** : Tierce Maintenance Applicative

Cela signifie que la maintenance de l'application n'est pas réalisée par la société cliente même, mais par l'entreprise tierce. Cela consiste donc pour une entreprise, à confier l'infogérance d'une application à une société externe. Ce mode de fonctionnement se développe de plus en plus car les entreprises souhaitent se concentrer sur leur coeur de métier et confier la gestion de l'informatique à une société externe.

**TD** : Test Director

Outil permettant d'industrialiser la phase de tests, dans le but de diminuer au maximum le nombre d'anomalies du programme. Permet une fois les defects détectés de confier leur correction à une personne spécifique.

**PIC** : Plateforme d'intégration Continue

La plateforme d'intégration continue est un ensemble d'outils permettant de réaliser périodiquement une compilation totale du projet et de lancer des phases de tests sur ce dernier. Ainsi, cet outil permet de retourner un rapport d'état du projet.

**Cast** :

Cast est un outil de qualimétrie qui permet d'évaluer la qualité du code (copier/coller, commentaires) en donnant des indicateurs de performance et d'analyser les impacts en cas d'évolution du code

**JEE** : Java Enterprise Edition

C'est une spécification pour le langage de programmation Java de Sun, plus particulièrement destinée aux applications d'entreprises. Cette spécification contient un ensemble d'extensions au framework Java Standard (Java Standard Edition) afin de faciliter la création d'applications réparties.

**Framework** :

C'est un ensemble de bibliothèques, d'outils, et de conventions permettant le développement d'applications. Il fournit suffisamment de briques logicielles, et impose suffisamment de rigueur, pour pouvoir produire une application aboutie et facile à maintenir. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres.

### **Injection par dépendances ou Inversion de contrôle :**

#### Exemple :

Soit un objet A dépend d'un objet B. Habituellement, l'objet B est créé et instancié dans l'objet A. Ici, dans notre cas, l'objet B est seulement déclaré dans A, mais il sera instancié par injection de dépendances lorsqu'un appel à cet objet B sera effectué.

Ainsi, ce n'est plus l'application qui gère les appels au framework, mais le framework qui gère les appels à l'application.

### **Beans :**

Au sens de Spring, les Beans sont des objets gérés par son conteneur léger. Au sens de Struts, un Beans est un composant de type ActionForm, c'est-à-dire qu'il contient les différents champs d'un formulaire affichés sur une page JSP, ainsi que ceux dans lesquels l'utilisateur peut réaliser une saisie. Ce Bean permet donc d'utiliser ces paramètres aussi bien dans les pages JSP que dans l'Action correspondante.

### **DAO : Data Access Object**

Composant permettant de récupérer ou modifier les différents attributs d'un objet métier dans la base de données.

### **AOP/POA : Programmation Orientée Aspects**

Ce type de programmation sépare les considérations techniques (Aspect en anglais) des descriptions métier dans une application. Ceci permet donc d'extraire les dépendances entre modules et de les gérer depuis l'extérieur de ces modules en les spécifiant dans des composants du système à développer, nommés aspects.

### **RMI : Remote Method Invocation**

C'est une interface de programmation (API) pour le langage Java qui permet d'appeler des méthodes distantes, présentes sur des serveurs distants par exemple.

### **ETL : Extract Transform And Load**

Il s'agit d'une technologie informatique chargée d'effectuer des synchronisations massives d'informations entre plusieurs entrepôts de données. Il permet de transformer les données quand les bases de données n'ont pas la même technologie ou ont une structure de données hétérogènes. Un des ETL les plus connus est Oracle Data Integrator (anciennement nommé Sunopsis).



**DAL** : Dossier d'Architecture Logicielle

Il a pour objectifs de décrire l'ensemble de l'architecture logicielle de l'application, les technologies et les frameworks externes utilisés, ainsi que les frameworks développés pour le projet. Cet ensemble constitue le socle applicatif de l'application. Ce document est également un guide pour les concepteurs/développeurs sur les différentes couches techniques sur lesquelles ils interviennent.

**DAT** : Dossier d'Architecture Technique

Le DAT, présente l'architecture technique de l'application CPS-V2. Généralement, sont explicités dans un premiers temps les éléments structurants l'architecture avant de détailler l'architecture même de l'application. Par conséquent, ce document donne les choix macroscopiques en termes d'architecture et d'infrastructure.

**PPL** : Plan de Production Logiciel

Ce dossier explique de manière détaillée comment installer l'environnement de développement sur sa machine. En d'autres termes, il énumère les différents composants à installer et comment les paramétrer. Ceci évite que le développeur perde trop de temps sur cette installation, qui n'est pas toujours chose aisée. Ce document est très important car l'équipe d'un projet informatique évolue fréquemment et le temps de mise en place de l'environnement logiciel doit être le plus court possible. Les objectifs de ce document sont les suivants :

- Décrire les outils mis en place pour le développement du système (outils de développement et de mise au point, instructions d'installation des logiciels, ...)
- Décrire l'environnement de production (arborescence de travail, architecture des composants, règles de gestion de configuration, sauvegardes, ...)
- Décrire les règles de programmation et de nommage (entêtes des fichiers, gestion des traces, normes de programmation à adopter, ...)
- Décrire les outils et la stratégie utilisés pour réaliser les tests unitaires

**JAR** : Java ARchive

Un fichier JAR est utilisé pour distribuer un ensemble de classes Java. Ce format est utilisé pour stocker des classes et des métadonnées constituant un programme ou une application.

**SFD** : Spécifications Fonctionnelles détaillées

Document dans lesquelles sont détaillées l'ensemble des fonctionnalités à développer.

## Quatrième partie

# Présentation de Steria

Steria est un groupe de services informatiques faisant partie des 10 plus grands groupes de services en Europe. Elle a été fondée en 1969 par Jean Carteron sur la base d'un large actionnariat salarié. Les principaux marchés de ce groupe sont les suivants :

- Les services publics
- La finance
- Les télécommunications
- L'énergie
- Les transports

## 1 Le groupe STERIA

Au cours de cette partie, je présenterais dans un premier temps les implantations du groupe Steria, avant de m'attarder sur ses missions. Puis, dans un dernier temps, j'aborderais quelques données clés.

### 1.1 Les implantations de STERIA

Steria est un groupe international. En effet, celui-ci est présent dans 16 pays situés sur divers continents, mais principalement implanté en Europe. Depuis le rachat de Xansa en 2007, la société a acquis de fortes positions en Inde avec 5000 salariés.

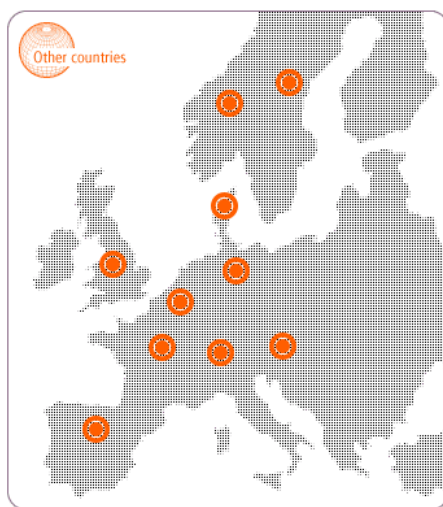


FIGURE 1 – Les implantations de STERIA en Europe

Comme le montre la figure ci-dessus, Steria est implanté dans les pays européens suivants : Allemagne, Autriche, Belgique, Danemark, Espagne, France, Luxembourg, Norvège, Royaume-Uni, Suède, et Suisse. Enfin, le groupe est également implanté au Maroc, en Chine, à Singapour, et en Inde.

Steria est dotée de 2 méga centres, et constituée au total de 11 centres de production, de 7 Service Desk, et de 11 centres de services (compétences et TMA).

Le siège de STERIA est localisé à Paris, c'est une société de droit français cotée sur la place boursière parisienne (mémo RIA).

## 1.2 Les missions de STERIA

L'entreprise Steria se positionne en opérateur de service global dans le secteur de l'informatique. Cela signifie qu'elle s'attèle à plusieurs tâches, que l'on peut résumer par :

- Le conseil, qui consiste à accompagner les clients dans la modification de leurs processus métiers ou fonctionnels.
- L'intégration de systèmes, qui consiste à faire communiquer des applications, des réseaux et des équipements hétérogènes dans les entreprises.
- L'infogérance, qui peut se faire de différentes façons : sur site (c'est à dire chez le client), ou externalisée (chez Steria).

## 1.3 Quelques chiffres

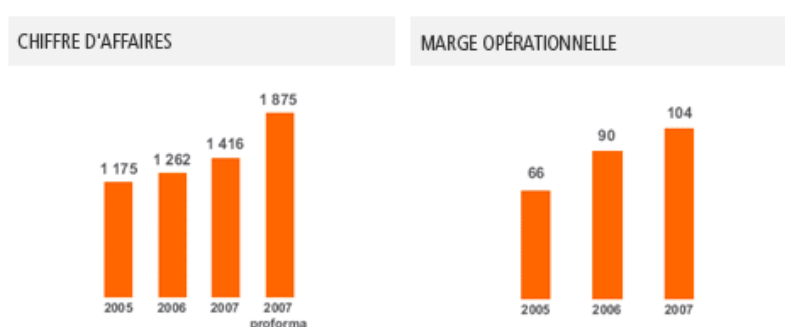


FIGURE 2 – Chiffre d'affaires et marge opérationnelle du groupe

Le chiffre d'affaires de Steria n'a cessé d'augmenter ces dernières années autant en croissance organique qu'en croissance externe. Cette croissance nécessite une politique de recrutement ambitieuse et volontariste.

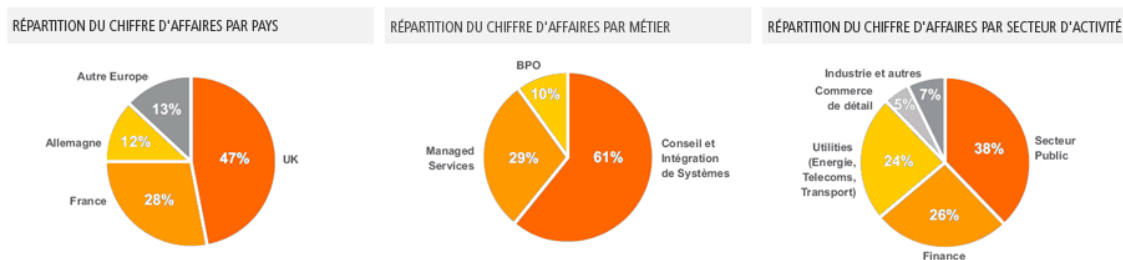


FIGURE 3 – Répartition du chiffre d'affaires

Le Royaume-Uni depuis l'intégration de Xansa est devenu le premier contributeur au CA du groupe avec près de 47% de l'activité. Les 2 principaux métiers du groupe sont l'infogérance, ainsi que le conseil et l'intégration de systèmes. Les 2 principaux secteurs où opère le groupe sont les services publics et la finance. La société a l'avantage d'avoir plus de 50% de contrats récurrents, ce qui lui assure une forte visibilité sur son activité future et permet de mieux absorber les périodes de ralentissement.

Voici une brève chronologie du groupe Steria :

- **1969 à 1986 :** De grands contrats multisectoriels en intégration de systèmes.
- **1986 à 1998 :** Développement de l'infogérance et internationalisation.
- **1998 à 2006 :** Montée en puissance d'un groupe européen, opérateur global de services informatiques.
- **2007 à 2008 :** Rachat et intégration de la société britannique Xansa, le groupe augmente sa taille de plus d'1/3 et passe par la même occasion dans le top 10 européen des services informatiques

La croissance du groupe est issue également d'une succession d'acquisitions :

- **Fin 2001 :** Steria a acquis l'essentiel d'Integris (activités européennes de services informatiques de Bull).
- **Janvier 2005 :** Acquisition de la société allemande Mummert Consulting.
- **Octobre 2007 :** Rachat de l'entreprise britannique Xansa

Steria possédait donc presque 10000 salariés en 2006, et aujourd'hui, elle a atteint les 18000 collaborateurs suite au rachat de Xansa, en 2007.

## 2 Les centres de services

Mon stage s'est déroulé dans l'entreprise Steria, localisée à Nantes, et plus particulièrement sur la plateforme (ou centre de services) TMA (Tierce Maintenance Applicative), anciennement dénommée AM3G (Application Management de 3ème Génération).

### 2.1 Les différentes générations des centres de services

Afin de mieux comprendre l'organisation de ce type de centres de services, détaillons les étapes qui ont conduit aux TMA. Il y a trois générations d'infogérance :

- La première génération consistait pour un client à s'adresser à une SSII afin d'obtenir des collaborateurs pour travailler sur un projet de maintenance.
- La seconde génération était celle des projets qui étaient gérés au forfait, ce qui apportait une garantie accrue pour les deux parties.
- Et la troisième génération est la Tierce Maintenance Applicative (TMA), qui est une maintenance au forfait, avec une externalisation des équipes de la SSII, et un contrat de service industrialisé. Cela signifie que la maintenance des applications est réalisée directement au sein des locaux de Steria

L'Application Management (infogérance) permet de décharger les équipes internes du client pour les dédier à de nouveaux projets, de mieux maîtriser les coûts, notamment par la forfaitisation, d'optimiser le processus de maintenance de ses applicatifs, etc.

Steria appelle Centre de Services de 3ème Génération une organisation et un mode de fonctionnement qui va beaucoup plus loin dans la démarche que celle qui est mise en place dans les Centres de Services dits de 2ème Génération. La démarche correspond à une volonté de mettre en oeuvre des plateformes totalement industrialisées en utilisant les mêmes outils, méthodes, et processus, quelque soit le plateau en France et en Europe.

Il faut noter qu'il y a peu de temps, le centre de services localisé à Nantes était encore un centre de troisième génération, dit AM3G. Mais de très récents changements font que la plateforme nantaise n'est plus en possession de ce grade. Elle se dénomme désormais AM (Application Management), ou TMA.

## 2.2 L'organisation des centres de services

Les centres de services s'appuient sur l'organisation suivante :

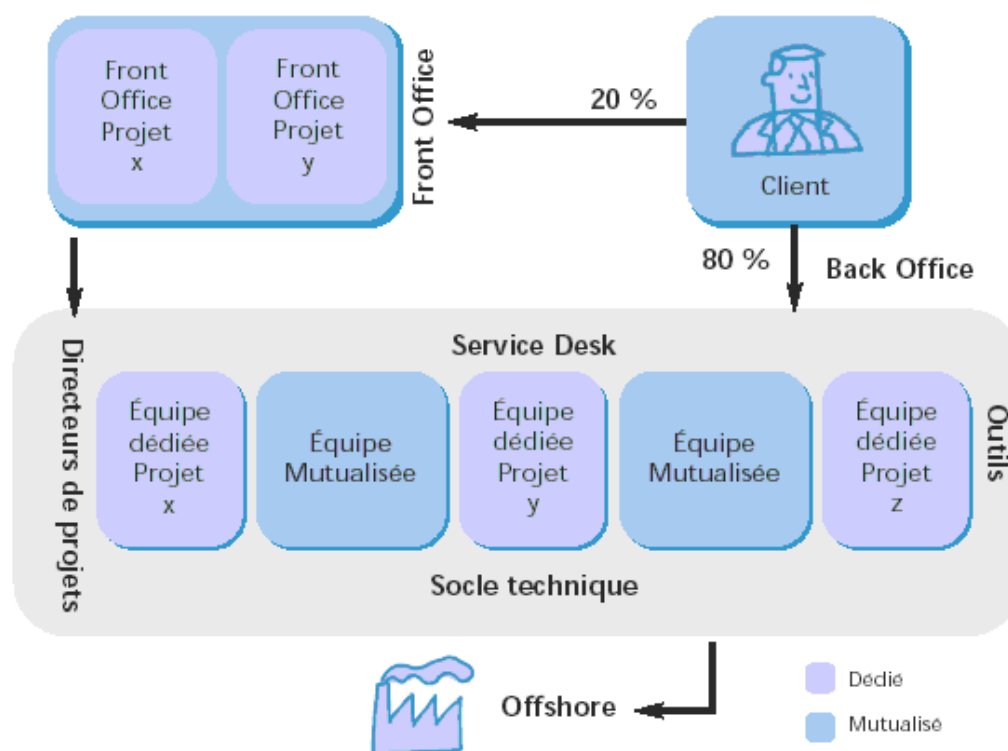


FIGURE 4 – L'organisation d'un centre de services

Deux importantes entités sont à souligner sur le schéma précédent : le Front Office et le Back Office.

### 2.2.1 Front Office (FO)

Le Front Office (FO) est principalement présent dans les locaux du client (dans le cadre du projet camaïeu le FO est située à Lille) pour garder la relation privilégiée de proximité. Il est constitué idéalement avec :

- Le Directeur de Projet Steria (interlocuteur privilégié du Client et garant des engagements pris)
- Les concepteurs travaillant sur le projet

### 2.2.2 Back Office (BO)

Le Back Office (BO) est le Centre de Services. Il est situé dans les locaux de Steria et se décompose en 5 grandes fonctions :

- **Le Service Desk** : Point d'entrée de la chaîne de soutien. Il est mutualisé et reçoit les appels pour des incidents fonctionnels autour des périmètres concernés. Il peut être de niveau 2 ou 3 selon les besoins.
- **Les Directeurs de projets BO** : Ils ont la responsabilité de la réalisation des projets sur le plateau. Ils rendent compte au Directeur de projet, situé au niveau du front office. Ils peuvent être mutualisés sur plusieurs projets.
- **Les experts du Socle Technique** : Ce sont des spécialistes des technologies utilisées sur le plateau. Ils sont mutualisés sur l'ensemble des projets réalisés dans le Centre de Services.
- **Les Outils** : Steria met en oeuvre un ensemble d'outils du marché permettant d'uniformiser le fonctionnement des TMA, et également d'optimiser la gestion de celles-ci à la fois en Interne et pour ses Clients.
- **Les Projets** : Les projets sont constitués d'équipes dédiées correspondant à un niveau moyen d'activités. Ces équipes sont renforcées à partir de cellules mutualisées qui sont autant de viviers technologiques, pour permettre d'absorber toutes les variations à la hausse, ou à la baisse, d'activités des projets.

## 2.3 La répartition géographique des centres

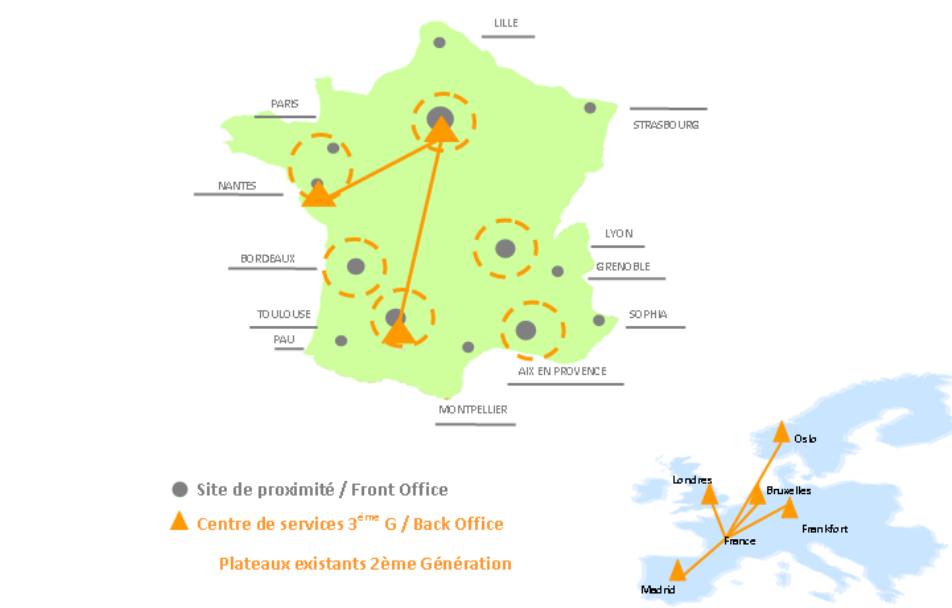


FIGURE 5 – Localisation française et européenne des centres de services

On remarque en observant les schémas précédents le souci de proximité de Steria vis-à-vis de ses clients. Cette organisation, associée à l'uniformité d'organisation des plateaux et à l'unicité des outils utilisés, permet de pouvoir organiser au mieux les ressources et les compétences.

## 2.4 Les outils mis en oeuvre

Steria met en place un portail (Web Service Client) permettant à l'ensemble des acteurs de la communauté (directeur projet côté client, directeur projet côté Steria,...) d'avoir accès aux données dont ils ont besoin pour piloter leur projet, et cela à tout instant. Ceci est stratégique pour une société de services car elle a tout intérêt à ce que les informations échangées restent en interne et ne soient pas diffusées sur des blogs ou sites sur internet. Cela permet également de capitaliser et de conserver les connaissances.

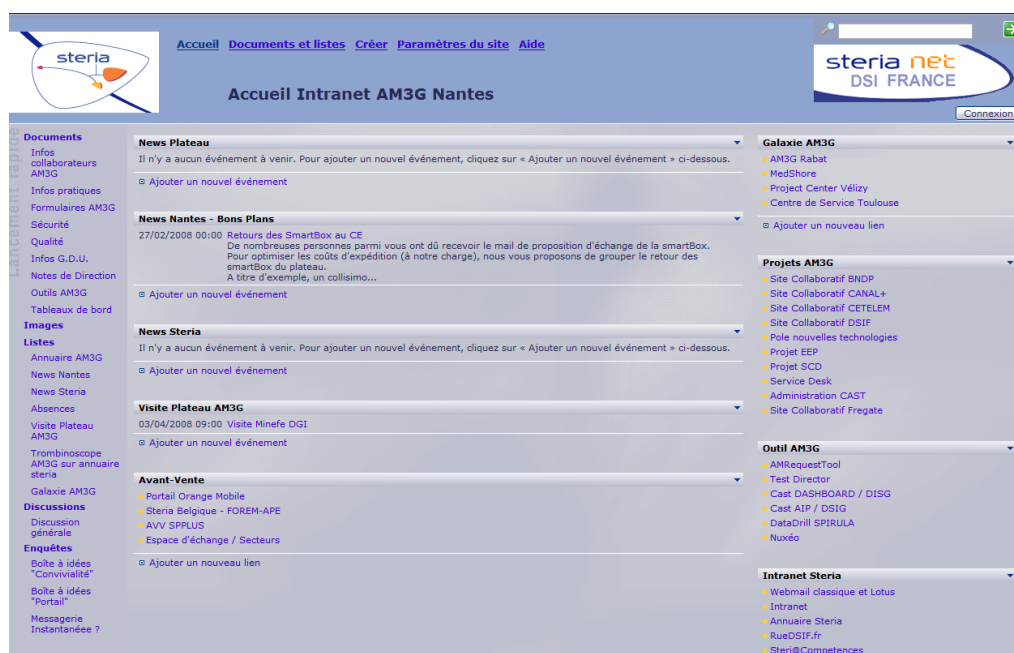


FIGURE 6 – Le portail de Steria

Ce portail paramétrable permet entre autre à l'utilisateur de :

- Mettre en place la Gouvernance du Système d'Information
- Suivre l'avancée des demandes
- Réaliser des estimations de charges macroscopiques
- Suivre les coûts
- Suivre la documentation du projet
- Si nécessaire, accéder à des spécificités après paramétrage (suivi d'un tiers,...)



Outre ce portail, Steria met en oeuvre un ensemble d'outils permettant d'industrialiser le fonctionnement des projets :

- AMRT : Outil de travail collaboratif permettant en outre :
  - La gestion des demandes : correctives, évolutives, de travaux, etc.
  - De générer de nombreux indicateurs, tels que les temps de prise en compte des anomalies sur une période donnée, pour une TMA donnée.



- Les outils de ACFO ou CAST, tels que System Code, permettent la gestion du patrimoine applicatif. Ce sont des outils puissants permettant d'analyser la qualité du code produit, de faire du reverse engineering et de la documentation. Ces outils permettent également d'analyser le code.



- Un outil de gestion documentaire



- Des outils de gestion de configuration tels que Clearcase, PVCS-Dimension, CVS, SVN :



- Des outils de gestion des tests en s'appuyant sur la suite Mercury et Princeton principalement (Winrunner, Loadrunner, Test Director,...)



- Des outils de modélisation (à partir de code) ou de génération de code (à partir d'une modélisation)



## 2.5 Exemples de projets gérés par le centre de services de Nantes

Nom du projet	Taille moyenne de l'équipe	Budget annuel(en j)
Cetelem	25 personnes	5000 jrs
Canal+	30 personnes	7000 jrs
DSIF Steria	25 personnes	6500 jrs
Fregate	15 personnes	7000 jrs

## 3 Le centre de services TMA de Nantes/St Herblain

Au cours de cette partie, sera présenté dans un premier temps, le centre de services de Nantes-St Herblain, avant de nous attarder sur son rôle dans le groupe Steria. Puis nous terminerons par nous intéresser à son organisation.

### 3.1 Présentation

Le centre de TMA de Nantes - St-Herblain est présent sur le site actuel depuis trois ans seulement.

Il compte actuellement 125 collaborateurs associés à la plateforme TMA. Théoriquement, ces derniers effectuent la maintenance des applications directement à partir des locaux nantais, contrairement aux salariés du département OIG. Le département Steria Ouest(OIG) est également présent sur le site, mais représente très peu de personnes à l'intérieur des locaux. En revanche, ce département gère deux à trois cents personnes présentes chez les clients.

### 3.2 Son rôle

Ce centre de TMA est chargé de divers projets de maintenance applicative. Les plus importants sont les suivants :

- DSIF constitue la maintenance d'une partie des applications informatiques de Steria France.
- CANAL+, ou encore CETELEM, sont des projets de maintenance d'applications clientes.

Le projet CAMAIEU auquel je suis affecté n'est pas un projet TMA mais un projet au forfait, bien que le développement de celui-ci s'effectue sur la plateforme TMA.

La maintenance d'applications peut se faire sous plusieurs formes, que l'on peut résumer suivant ces trois termes :

- **Maintenance de support** : Il s'agit d'assurer une assistance avec le client. Cette maintenance peut-être rapprochée d'une hotline. Elle est parfois appelée "Demande de travaux".
- **Maintenance corrective** : il s'agit ici d'apporter des modifications de correction à des applications sur lesquelles porte la maintenance.
- **Maintenance évolutive** : la maintenance évolutive ne porte pas sur la correction d'anomalies, mais sur la modification à plus grande échelle d'une application, comme par exemple l'ajout d'une fonctionnalité supplémentaire.

### 3.3 Son organisation

Voici l'organisation interne du centre AM de Nantes/St Herblain :

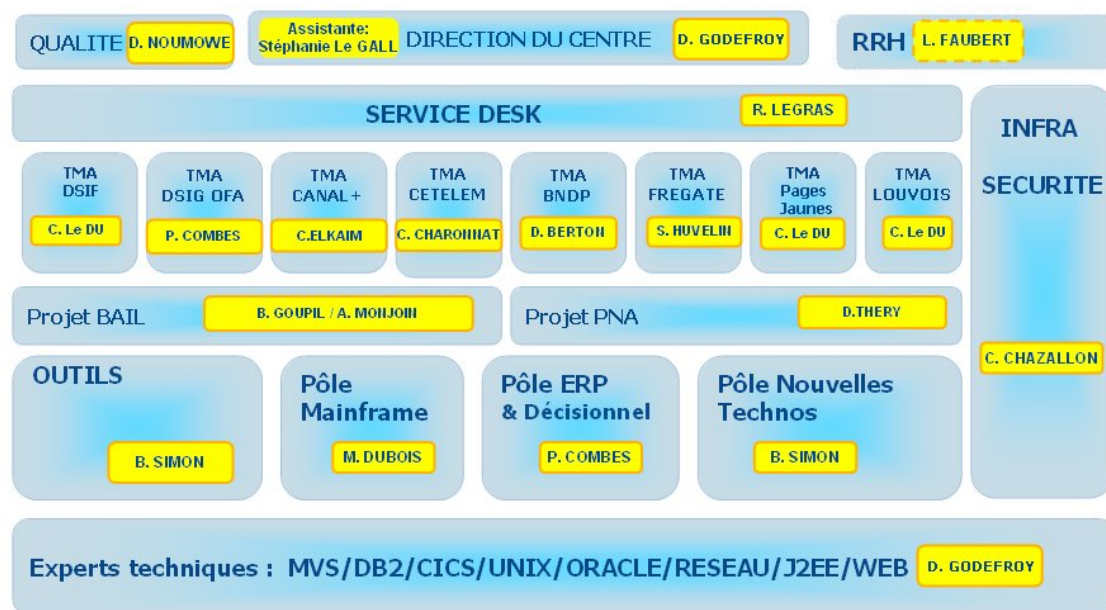


FIGURE 7 – Organisation du centre AM

On remarque sur ce schéma l'importance des projets TMA au sein du centre. Les projets TMA sont très intéressants par leur visibilité. Mais il existe également sur le centre quelques projets aux forfaits tels que PNA, Camaïeu et Bail. On notera aussi l'importance des fonctions transverses tels que le pôle mainframe, le pôle nouvelles technologies, et le pôle ERP et décisionnel. Ces fonctions transverses apportent leurs compétences fonctionnelles et organisationnelles à l'ensemble du plateau. Elles bénéficient d'une bonne souplesse puisque leur personnel peut être réaffecté sur des projets si ce dernier a de fortes montées en charges.

## Cinquième partie

# Le stage

Dans cette partie, je vous présenterai le sujet de mon stage, avant d'énumérer mes objectifs personnels sur le stage. Enfin, nous aborderons le projet en tant que tel en réalisant une présentation de celui-ci, en prenant en compte les différentes phases importantes du projet, et en s'attardant sur l'architecture matérielle et logicielle de l'application. Puis je terminerai par exposer le travail que j'ai effectué depuis le début de mon stage en m'appuyant sur le planning réalisé, également présenté dans cette partie.

## 4 Le sujet de stage

J'ai été affecté tout au long de mon stage à une équipe de développement qui a beaucoup évolué en terme d'effectifs puisqu'elle est passée de 6 à 14 personnes. L'intitulé de mon sujet est le suivant : "*Développement d'une application stratégique pour le système d'informations d'une marque de prêt-à-porter*". En d'autres termes, cela signifie que nous devons mettre en oeuvre une application pour la chaîne de prêt-à-porter Camaïeu, dont le siège est localisé à Lille, dans le but d'organiser la gestion des approvisionnements de l'ensemble de leur magasins.

Cette application a été développée dans un environnement JEE. Un panel de technologies et d'outils ont été utilisées pour réaliser notre projet. Parmi lesquelles :

- **Langages** : Java, JSP, HTML, CSS, Javascript, SQL
- **Frameworks** : Struts (dont le Validator), Spring, Hibernate, Tiles
- **Tests, compilation** : Maestro (Plateforme d'intégration continue), Maven
- **Environnement de développement** : Eclipse
- **Travail collaboratif** : SVN
- **Outil de modélisation** : Star UML
- **Serveurs** : Tomcat v5.5
- **Bases de données** : Oracle 10g, LDAP

Je reviendrais plus en détails sur ces technologies dans les chapitres suivants.

## 5 Mes objectifs

J'avais personnellement plusieurs objectifs sur ce stage de fin d'études et je vais les énoncer ici. Tout d'abord mon premier objectif était de se former sur des technologies et d'acquérir un bagage technique minimum pour pouvoir évoluer confortablement dans le métier d'informaticien. Cet objectif est pour moi atteint car j'ai

travaillé sur un riche panel de technologies qui sont en plus particulièrement recherchées sur le marché de l'emploi actuellement.

Mon second objectif était de participer à l'ensemble du cycle de vie d'un projet et notamment de pouvoir participer à différentes phases qui agrémentent la vie d'un projet. J'ai pu participer à différentes phases et ne pas seulement faire que du développement puisque j'ai également participé aux conceptions, aux livraisons client, et j'étais en relation avec le front (voir directement avec le client). Il a été particulièrement enrichissant de participer à ces différentes phases au cours de mon projet et de voir autre chose que du développement pur et dur.

Enfin mon dernier objectif était de m'intégrer et de participer à la vie d'une équipe de développement. Cet objectif a été pleinement atteint car je me suis très bien intégré à cette équipe. Il faut dire, et je remercie l'ensemble des membres de l'équipe de camaïeu pour cela, qu'ils ont été très accueillant dès le début avec moi et cela m'a beaucoup aidé pour progresser et prendre des responsabilités.

## 6 Le projet

Au cours de cette partie, je vais présenter dans un premier temps le projet. Puis dans un second temps, nous nous intéresserons aux différentes phases à prendre en compte pour mener à bien un projet, telles que l'identification des besoins du client, la maîtrise des risques, la gestion, le chiffrage et la planification du projet. Enfin, nous terminerons par présenter l'architecture matérielle et logicielle de l'application Camaïeu à développer.

### 6.1 Présentation

Comme énoncé auparavant, le projet Camaïeu consiste à développer une application permettant à la chaîne de prêt-à-porter d'organiser la gestion des approvisionnements de ses magasins. Camaïeu considère que l'approvisionnement fait partie de son coeur métier et que cette compétence est stratégique, doit être gérée en interne et gérée avec des outils de qualité. D'ailleurs on peut noter que la migration de cette application va entraîner le basculement de l'ensemble du SI vers une nouvelle version.

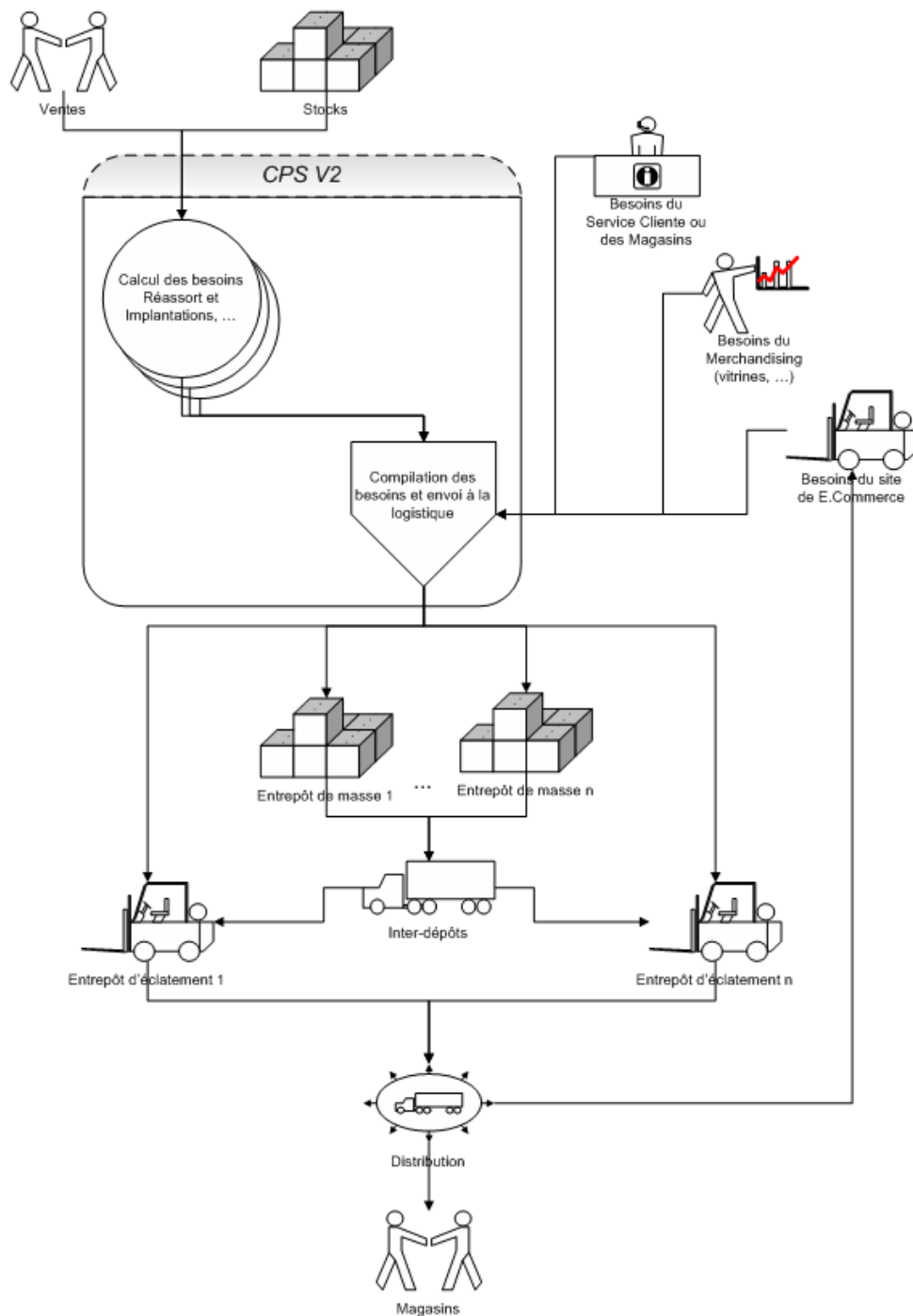


FIGURE 8 – Organisation de la chaîne CAMAÏEU

L'application développée par Steria se dénomme CPS-V2 (Camaïeu Procurement System Version 2). Camaïeu n'est pas satisfait de la première version, et souhaitait repartir sur une base saine en repartant d'une page blanche. Steria est donc parti de zéro pour développer ce projet.

Comme il est possible de le remarquer à partir de la figure précédente, l'application que nous devons développer est située au coeur de la supply chain de Camaïeu. CPS est en interconnexion avec l'ensemble du SI de Camaïeu. Il est alimenté par différentes bases en entrées telles que les ventes et les stocks des magasins et envoie en sortie des ordres de livraisons aux différents entrepôts de données après les avoir calculés. L'alimentation par un site de e-commerce est une évolution ultérieure souhaitée par Camaïeu.

Ainsi, pour réaliser l'ensemble de ces opérations, l'application que nous devons développer doit également s'adapter et interagir avec les applications déjà implantées dans le processus Camaïeu.

## 6.2 Les besoins

Comme je l'ai énoncé dans la partie précédente, la chaîne de prêt-à-porter Camaïeu disposait déjà d'une première version de son outil de gestion des approvisionnements, mais qui ne répond plus aujourd'hui à leurs attentes. En effet :

- L'ergonomie de l'application n'est pas adaptée. Les écrans de paramétrage sont trop nombreux et trop complexes pour obtenir des synthèses exploitables rapidement.
- La maintenance applicative est trop difficile à gérer. A plusieurs endroits dans le code, les mêmes règles de gestion et les mêmes algorithmes sont dupliqués.
- L'application n'est pas fiable. Les données affichées peuvent être incohérentes avec la base et selon le chemin pris dans l'application les résultats peuvent être incohérents entre eux
- L'application n'est pas adaptée aux évolutions de la société Camaïeu. Par exemple elle ne pourra s'adapter au rajout (envisagé par le client) de nouveaux entrepôts de masse et donc à l'évolution future du réseau de distribution de Camaïeu.
- L'application est difficilement déployable, c'est un client lourd et qui ne dispose pas de mécanismes de mise à jour automatique. Le passage sur un client léger et une application web facilite fortement le déploiement de l'application.
- L'application envoie ses ordres de livraisons seulement la nuit ce qui entraîne un fonctionnement heurté au moindre problème technique au niveau des centres de livraisons.
- Le code est très difficile à maintenir ce qui empêche Camaïeu de confier son évolution dans le cadre d'une TMA
- L'application n'explique pas ses choix et son raisonnement lorsqu'elle envoie les ordres de livraisons, ce qui entraîne une perte de confiance des utilisateurs.

Ces inconvénients ont évidemment des conséquences actuellement sur Camaïeu. Les managers de magasins ayant souvent des ratés dans les livraisons du fait des bugs qui existent, ils n'ont pas confiance et donc appellent très souvent l'équipe en charge du logiciel. Ces appels entraînent de lourdes pertes de temps aussi bien pour les managers de magasins que pour l'équipe utilisant CPS. Les magasins ne comprennent souvent pas les choix qui ont été faits par le logiciel lors de la livraison, et comme CPS n'historise pas et n'explique pas les choix, c'est l'équipe informatique de Camaïeu elle-même qui est obligée de chercher dans les bases de données les raisons qui ont conduits à une telle livraison. L'application actuelle est, plus grave encore, un frein au développement même de Camaïeu puisque elle ne peut s'adapter à l'ajout, inévitable, de nouveaux entrepôts de masse pour répondre à la forte croissance de cette société (ouverture de nombreux magasins).

La deuxième version de cet outil de gestion des approvisionnements doit donc reprendre les fonctionnalités de l'ancienne version en les améliorant et pour cela, Steria doit développer une application permettant à Camaïeu d'intégrer :

- Les règles de gestion déjà existantes
- Le traitement des opérations au fil de l'eau (traitement Just-in-time qui entraînera du coup un flux continu au niveau des centres d'approvisionnement)
- La gestion des activités en multi-entrepôts
- Une ergonomie adaptée à ses utilisateurs avec des écrans simples favorisant la lisibilité des informations et facilitant le paramétrage de l'application.
- une explication des choix effectués par le logiciel si l'utilisateur le souhaite

De plus, le projet CPS-V2 a également pour ambition de devenir, une fois l'outil développé, un projet TMA (Tierce Maintenance Applicative). C'est-à-dire que Steria assurera la maintenance du système ce qui lui assurera des revenus récurrents. Camaïeu pourra ainsi se concentrer sur son cœur de métier et externaliser les fonctionnalités informatiques à une société spécialisée.

### 6.3 La maîtrise des risques

Suite à l'analyse des besoins de la chaîne de prêt-à-porter Camaïeu, concernant la gestion des approvisionnements de ses magasins, quelques risques, plus ou moins importants, sont à prendre en compte pour le bon déroulement de ce projet. D'après l'analyse des besoins précédente, il ressort des risques fonctionnels, techniques, de délais et de transferts. Voici les moyens mis en oeuvre par Steria pour minimiser ces derniers :



Domaine	Niveau de risque	Description du risque	Moyens pour minimiser
Fonctionnel	*****	Difficultés de prise en mains de l'application par les utilisateurs	Intervention d'un ergonome Implication des utilisateurs dès la phase de spécifications Conduite du changement Formation métier du CP fonctionnel
Délais	*****	Retard de livraison de l'application	Dimensionnement conséquent de l'équipe dès le démarrage Force de production du Centre de Services (capacité de montée en charge) Suivi / pilotage rapproché du projet Lotissement
Technique	*****	Fiabilité de la nouvelle application	Contrôle qualité outillé (PIC, CAST) Gestion de configuration logicielle Tests approfondis et outillés (TD) Assistance à la recette métier
Transfert	*****	Perte de connaissance fonctionnelle et technique lors du transfert de l'application vers la TMA	Intervenants projet maintenus sur la TMA : continuité Projet -> TMA En cas de TMA Tiers : Processus de transfert de compétences Documentation adéquate Cartographie applicative (CAST) Référentiel de tests (TD)

FIGURE 9 – Moyens pour minimiser les risques

Comme il est possible de le voir sur la figure précédente, Steria fait appel à quelques outils pour minimiser les risques du projet. Parmi ces derniers, nous retrouvons : Cast, Test Director, et la PIC en l'occurrence Maestro (cf. : Annexe 1).

## 6.4 La gestion du projet

Pour réaliser la gestion du projet Camaïeu, Steria utilise le modèle du cycle en V. Ce dernier est un modèle conceptuel de gestion de projet, imaginé suite au problème de réactivité du modèle en cascade. Il permet, en cas d'anomalies, de limiter un retour aux étapes précédentes.

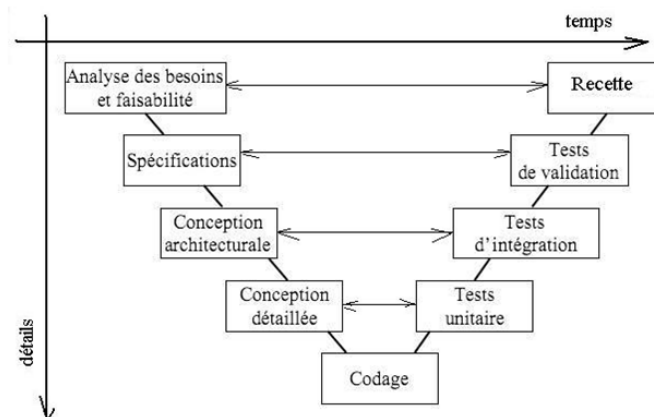


FIGURE 10 – Le cycle en V : plus on descend et plus on s'intéresse aux détails de l'application.

Les phases de la partie montante doivent renvoyer de l'information sur les phases en vis-à-vis lorsque des défauts sont détectés, afin d'améliorer le logiciel. Cela signifie

que lorsqu'on réalise les spécifications, il faut également imaginer au même moment les tests de validation. De même, lorsqu'on réalise la conception détaillée, il faut penser aux éventuels tests unitaires. Ce procédé est identique pour chaque niveau du cycle. Ainsi, une fois que le développement de l'application est effectué, les tests unitaires vont valider la conception détaillée, les tests d'intégration vont valider la conception architecturale et ainsi de suite.

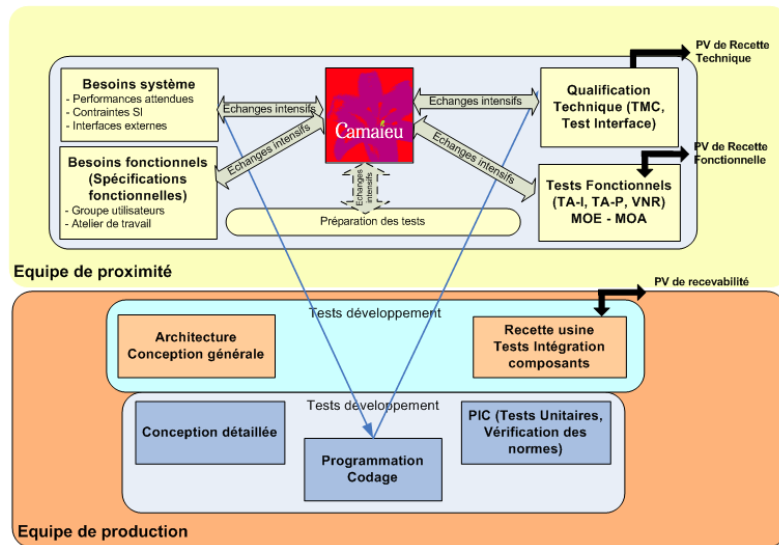


FIGURE 11 – Cycle en V du projet Camaïeu

Comme on le remarque sur la figure ci-dessus, les besoins systèmes et fonctionnels sont identifiés par l'équipe de proximité, en l'occurrence le personnel de l'agence Steria de Lille (le Front Office), situé à proximité du siège de Camaïeu. La conception générale et détaillée, ainsi que le développement de l'application sont réalisés par l'équipe de production à Nantes (le Back Office).

## 6.5 Le chiffrage du projet

Cette partie est consacrée à la description du processus utilisé par l'entreprise Steria pour chiffrer la charge d'un projet en j/h.

Pour cela, le chef de projet doit, dans un premier temps, dénombrer le nombre d'unités d'œuvres à partir des spécifications techniques, tout en leur associant un indice de complexité.

Voici le tableau utilisé pour réaliser cette opération :

Fonction	Tâche Paragraphe	NAVIGATION			CONSULTATION			SAISIE			IMPRESSION			CALCUL		
		Simple	Moyenne	Complexe	Simple	Moyenne	Complexe	Simple	Moyenne	Complexe	Simple	Moyenne	Complexe	Simple	Moyenne	Complexe
Menu de navigation	\$2.2.3			1												1
Mécanisme Annulation/Validation	\$1.6.1							1								
Liste des utilisateurs	\$2.2.4								1							
Création des utilisateurs	\$2.2.6							1				1				
Liste des écrans	\$2.3.4							1							1	
Recherche des utilisateurs	\$2.4.3							1	1							
Ajout d'un utilisateur	\$2.4.4								1							
Duplication d'un utilisateur	\$2.4.7								1							
<b>TOTAL</b>		0	0	1	0	0	0	4	4	0	0	1	0	0	1	1
VALORISATION PTU		0,00	0,00	3,00	0,00	0,00	0,00	4,00	12,00	0,00	0,00	4,00	0,00	0,00	5,00	10,00
VALORISATION TOTAL		0,00	0,00	6,86	0,00	0,00	0,00	9,14	27,42	0,00	0,00	9,14	0,00	0,00	11,43	22,95
<b>TOTAL</b>																

FIGURE 12 – Dénombrement des unités d'oeuvres

Comme le montre le tableau ci-dessus, sont énumérées, à gauche, toutes les fonctionnalités de l'application à développer, chacune étant référencée vers le paragraphe correspondant de la spécification technique. Puis, pour chacune de ces fonctionnalités, il faut déterminer le nombre d'écrans à développer, ainsi que leur type et leur complexité :

- **Un élément de navigation** : plan ou menu inspiré des sites web, il est utilisé pour naviguer dans une application web. Cet élément peut également être un cadre inspiré du monde Web. Un agencement de cadres permet de partitionner une fenêtre en plusieurs écrans et de garder à l'écran certains éléments (logo, plan de navigation,...)
- **Un élément de consultation** : Ecran, Page, Onglet, Popup : objet ne permettant pas de saisir des informations
- **Un élément de saisie** : Ecran, Formulaire, Onglet, Popup de saisie
- **Un élément d'impression** : Remise en forme d'un Ecran, d'une Page, d'un Onglet destiné à être imprimé
- **Un élément calcul** : Traitements batch ou interactifs complexes (hors écrans), et intervention sur la base de données.

Pour chacun de ces éléments, des indices de complexité sont définis en fonction des caractéristiques de ces derniers.

Une fois les unités d'oeuvres définies, la charge totale de développement (en j/h) est alors obtenue. Il faut savoir que deux totaux sont différenciés :

- La valorisation du PTU : le nombre de j/h de développement nécessaires sans les tests
- La valorisation du TOTAL : le nombre de j/h nécessaires pour la réalisation du projet avec les tests, l'analyse du projet, et la conception comprise.

Dans notre exemple, la valorisation du TOTAL est de 86 j/h, alors que la valorisation PTU est de 38 j/h.

Il est également important de savoir que pour obtenir cette valorisation PTU ou TOTAL, le total de chaque colonne du tableau est multiplié par un certain coefficient, qui est déterminé grâce à l'expérience de l'entreprise.

Charge en jours pour PTU			
Type de composants	Complexité des composants à		
	Simple	Moyen	Complexe
Navigation	3	4	5
Consultation	2	3	4
Saisie	3	4	5
Impression	1	2	3
Calcul	4	5	6

Charge totale en jours			
Type de composants	Complexité des composants à		
	Simple	Moyen	Complexe
Navigation	3,00	5,20	7,00
Consultation	0,50	1,00	2,00
Saisie	3,00	3,50	4,00
Impression	3,80	4,50	5,00
Calcul	5,00	5,50	7,00

FIGURE 13 – Les coefficients de calcul de charge en jours

Le même processus est utilisé pour définir le coût du développement de l'application. Le total de chaque colonne du tableau de la figure 15 est multiplié par un certain coefficient, déterminé également grâce à l'expérience de l'entreprise.

Coût Total en €			
Type de composants	Complexité des composants à créer		
	Simple	Moyen	Complexe
Navigation	1 000,00	2 000,00	3 000,00
Consultation	1 100,00	2 100,00	3 100,00
Saisie	1 200,00	2 200,00	3 200,00
Impression	1 300,00	2 300,00	3 300,00
Calcul	1 400,00	2 400,00	3 400,00

FIGURE 14 – Les coefficients de calcul des coûts

Steria possède également d'autres outils permettant de chiffrer le projet de manière plus détaillée, comme connaître le nombre de j/h par tâche ou bien par plateforme de développement, ou encore le coût de chaque phase de réalisation.

Concernant le projet Camaïeu, il a été initialement chiffré à 300 j/h, mais ce chiffrage a complètement explosé puisqu'aujourd'hui nous en sommes déjà à 1100 j/h simplement pour la partie production et celle ci devrait atteindre 1500 j/h à la livraison.

## 6.6 Le planning du projet

Une fois le chiffrage du projet effectué, il est possible de définir son planning

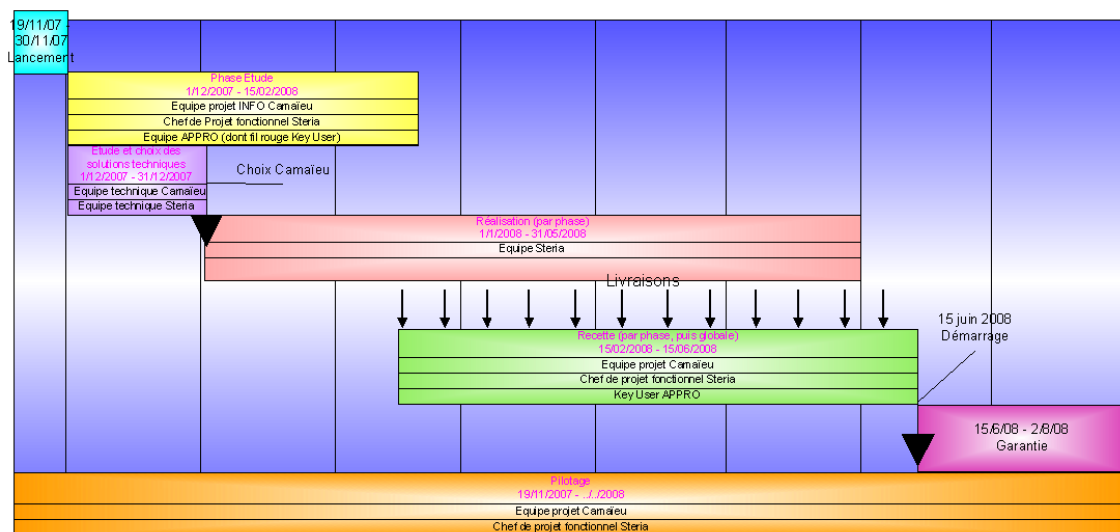


FIGURE 15 – Planning général du projet Camaïeu

Comme il est possible de le remarquer sur le planning ci-dessus, le projet à été lancé au mois de novembre 2007 et devait se terminer en Juin 2008. Du fait du fort accroissement des charges, la première livraison a été fixée finalement pour Septembre suivie d'une 2ème livraison en Octobre.

Il est possible de distinguer trois phases principales : La phase d'études avec le choix des solutions techniques, la phase de réalisation, puis la phase de Recette qui est pratiquement effectuée en parallèle avec la réalisation. En effet, l'application CPS-V2 est découpée en plusieurs modules, qui feront chacun office après développement d'une recette.

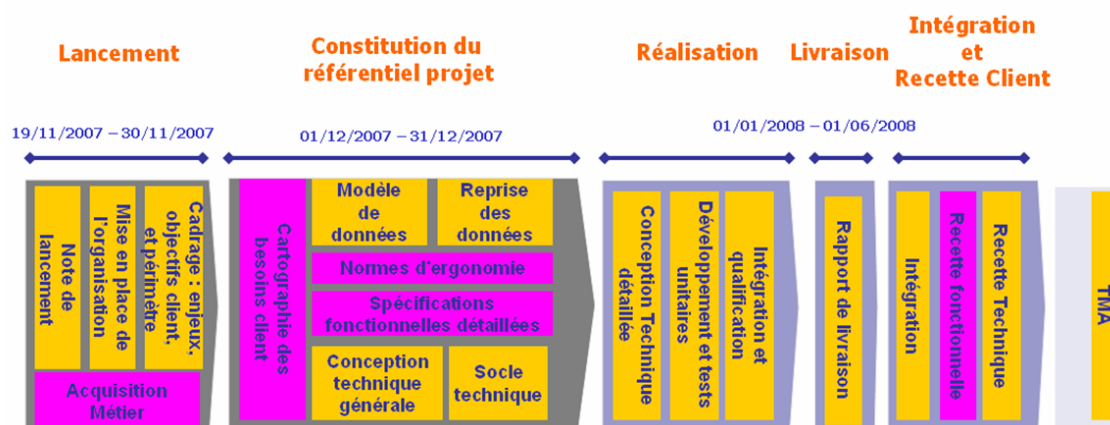


FIGURE 16 – Planning détaillé du projet Camaïeu

Lorsque je suis arrivé au début du mois d'avril 2008 le projet était déjà bien lancé et les développements avaient commencé depuis un petit moment. Le socle technique ainsi que plusieurs écrans étaient en place et j'ai donc pu assez rapidement après le début de mon stage commencé à développer des écrans côté présentation.

Comme le montre la figure ci-dessus, la globalité du projet Camaïeu, de l'appel d'offre jusqu'au recettage, se découpe en cinq phases bien distinctes :

- L'avant vente ou lancement du projet
- La phase d'études ou constitution du référentiel projet
- La phase de réalisation ou de développement
- La phase de livraison
- La phase de recette

Une sixième phase, non négligeable, peut être mentionnée, soit celle de la maintenance applicative. L'objectif est que le projet devienne une TMA ultérieurement.

### 6.6.1 L'avant vente

Comme le montre la figure 17, cette phase se découpe également en quatre étapes :

- L'appel d'offre
- La proposition commerciale
- L'étape des questions/réponses
- Le contrat

Voici comment ces différentes étapes interagissent entre elles :

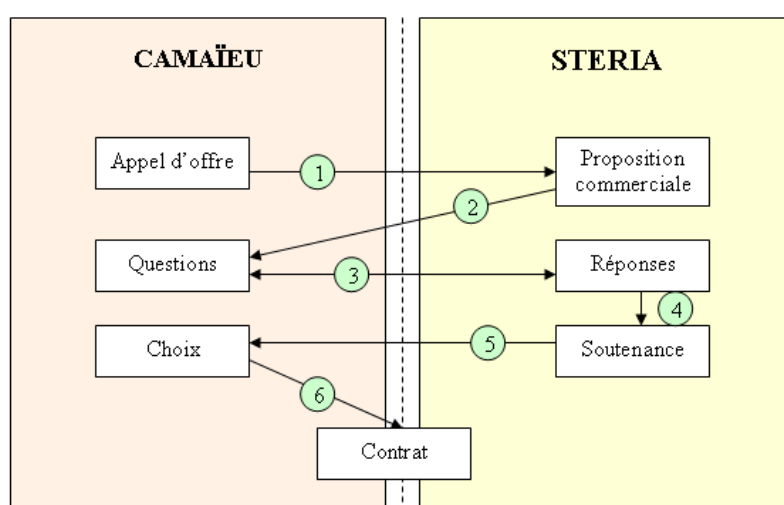


FIGURE 17 – Diagramme de séquence de la phase d'avant vente

1. Comme le montre la figure ci-dessus, la phase d'avant vente débute lorsque le client, en l'occurrence Camaïeu, envoie son appel d'offre. Ce dernier est donc un dossier rédigé par le client où sont présents les éléments suivants : le cadre, les attentes du projet, les besoins du client, les exigences générales, et le planning du projet.
2. Lorsque le prestataire, reçoit l'appel d'offre du client, il rédige une proposition commerciale dans laquelle il présente dans un premier temps son entreprise, les aspects financiers, les outils qu'il utilise, et les compétences qu'il dispose. Dans un second temps, le prestataire montre également qu'il a bien compris les enjeux du projet, et essaye de répondre aux différentes questions du client présentes dans le document constituant l'appel d'offre. Enfin, dans un dernier temps, le prestataire essaye de proposer une solution, de répondre en termes d'organisation, en découpant le projet en plusieurs phases, également présentées dans cette proposition commerciale.

Vous trouverez dans l'annexe 4 le sommaire de la proposition commerciale rédigée par Steria pour répondre à l'appel d'offre de Camaïeu.

3. Une fois que le client a reçu cette proposition commerciale du prestataire, on va arriver dans une phase d'échanges, principalement constituée de questions/réponses.
4. Ensuite, le prestataire continue en réalisant une soutenance chez le client durant laquelle il résume et explicite tous les choix effectués dans la proposition commerciale.
5. Suite à cette soutenance, le client réfléchit et fait son choix
6. Enfin, si la réponse du client est positive, un contrat est rédigé entre le client et le prestataire. Généralement, l'appel d'offre et la proposition commerciale sont mis en annexe.

Une fois le contrat rédigé et signé, la phase d'études peut débiter.



### 6.6.2 La phase d'études

Comme le montre la figure 17, de nombreux documents sont réalisés au cours de cette phase. En effet, une majorité de ces derniers peuvent être rédigés parallèlement, mais certains doivent l'être selon un certain ordre, comme le montre la figure ci-dessous :

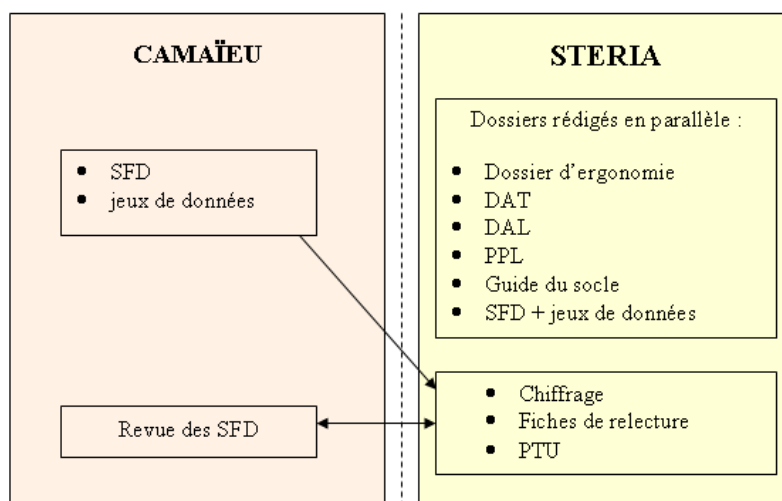


FIGURE 18 – Schéma de la phase d'études

Dans le cas du projet Camaïeu, le dossier d'ergonomie a été réalisé par un ergonome de Steria Paris, suivant les attentes du client.

Comme le montre la figure précédente, c'est durant cette phase d'études que l'architecture de l'application est décidée à travers le DAT et le DAL.

Vous pourrez retrouver dans l'annexe 5 le sommaire du DAT rédigé pour le projet Camaïeu.

Vous pourrez retrouver dans l'annexe 6 le sommaire du DAL rédigé pour le projet Camaïeu.

Le PPL (Plan de Production Logicielle) présente les différentes règles de production logicielle appliquées aux logiciels produits utilisés par Steria dans le cadre du projet.

Vous pourrez retrouver dans l'annexe 7, le sommaire du PPL rédigé pour le projet Camaïeu.

Le Guide du socle est un document rédigé pour les concepteurs/développeurs afin de leur faciliter l'utilisation du socle applicatif en fonction des différentes couches techniques. Le socle applicatif constitue une base sur laquelle est développée l'application. En d'autres termes, ce socle fournit un ensemble de classes et de méthodes de

base utilisées pour mettre en oeuvre une application JEE. Ce guide permet donc de guider le concepteur, ou le développeur, dans son utilisation du socle et indique les règles à respecter lors de l'implémentation. Il donne des exemples concrets d'utilisation pour chaque couche technique avec notamment des exemples d'implémentation des objets métiers, des services métiers, des façades, des DAO, etc.

Les documents décrits ci-dessus peuvent être rédigés simultanément. En effet, aucun ordre n'est nécessaire pour leur bonne réalisation. De plus, comme le montre la figure précédente, ces derniers sont rédigés par le prestataire. Mais il faut savoir qu'en parallèle, le client rédige les SFD (Spécifications Fonctionnelles détaillées) qui sont généralement rédigées pour un écran de l'application (c'est le cas dans le cadre de CPS). Une SFD est également constituée d'un MCD (Modèle Conceptuel de Données), et d'un MPD (Modèle Physique de données). Ces derniers permettent de réaliser une description de la base de données.

Le client fournit donc les SFD, mais également des jeux de données significatives avec ces dernières. Celles-ci permettent aux développeurs et aux concepteurs d'avoir un exemple de données qu'il est possible de retrouver dans les écrans à développer. Ces jeux de données peuvent également être utilisés pour effectuer des tests à la fin des développements.

Une fois que le prestataire reçoit les SFD du client, il peut effectuer le chiffrage du développement concernant les écrans spécifiés. La méthode de chiffrage utilisée par Steria a été décrite dans la partie précédente. Egalement, et parallèlement à ce chiffrage, des fiches de relecture peuvent être rédigées après avoir parcouru les SFD reçues. Dans ces fiches vont être signalés des éléments ou parties des SFD à éclaircir, des contradictions présentes au sein des SFD, ou des informations manquantes. Ces fiches de relecture sont alors retournées au client afin de corriger ou de compléter les SFD, qui seront à nouveau transférées au prestataire.

Lorsque tous les points sombres des SFD ont été éclaircis, le PTU (Plan de Tests Unitaires) correspondant à la SFD est réalisé. Cette étape consiste à réfléchir et à imaginer les tests à effectuer une fois que le développement de l'écran, spécifié par la SFD, soit terminé. Le processus du cycle en V est donc retrouvé ici. En effet, les tests sont décrits avant le commencement des développements.

Une fois toutes ces étapes terminées, les dossiers de conception peuvent être rédigés à partir des SFD. Il faut savoir qu'à chaque SFD va correspondre un dossier de conception. Au sein de ce dernier, le développeur pourra retrouver précisément tous les détails techniques qui lui seront utiles pour effectuer le développement de l'écran en question. Nous entrons alors dans la phase de réalisation.

### 6.6.3 La phase de réalisation

Comme je viens de l'énoncer, la phase de réalisation débute par la rédaction du dossier de conception. cette phase se découpe en trois étapes :

- Rédaction du dossier de conception
- Développement
- Recette interne

Ainsi, un dossier de conception correspond à une SFD, soit à un écran. Une fois que ce dossier est rédigé, le développeur s'appuie sur ce dernier pour développer l'écran en question. Le développement d'un écran sera explicité ultérieurement dans ce rapport, lors de la description du travail que j'ai effectué sur la partie présentation.

Enfin, lorsque l'écran a été développé, l'étape de recette peut débiter. Plusieurs niveaux de tests sont réalisés au cours de cette dernière, comme le montre la figure suivante :

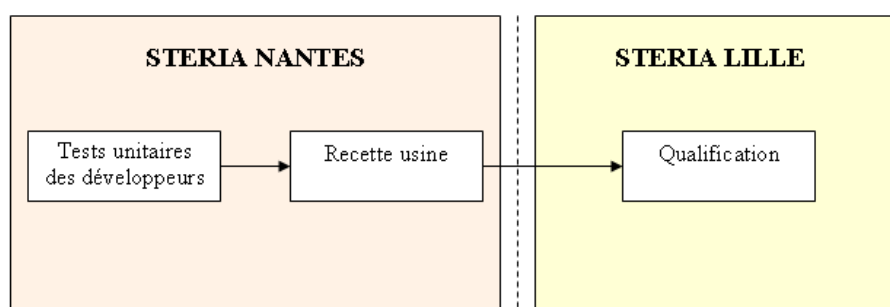


FIGURE 19 – Schéma de la phase de Recette

Ainsi, le premier niveau de tests est effectué par les développeurs. En effet, ces derniers réalisent toute une série de tests unitaires sur l'écran juste développé. Une fois que le développeur considère l'écran terminé, c'est-à-dire que tous ces tests unitaires sont positifs, l'écran en question subit le deuxième niveau de tests, soit la recette usine.

Comme le montre la figure ci-dessus, la recette usine est réalisée au sein de Steria Nantes, soit par le back office. C'est pour cette raison que cette étape est également appelée recette interne. Il faut savoir que ce sont des tests de type technique qui sont effectués à ce niveau, comme tester l'ensemble des boutons de l'écran, ou comme vérifier l'affichage des informations. Chaque élément présent sur l'écran est donc testé. De plus, il est important de savoir que ces tests sont réalisés en s'appuyant sur le PTU (Plan de Tests Unitaires) rédigé précédemment. Ce PTU et les résultats correspondants sont ensuite envoyés au client, afin de lui apporter une assurance sur le bon fonctionnement de l'application. Mais avant de livrer l'ensemble au client, cette dernière subit le troisième et dernier niveau de tests, soit l'étape de qualification.

Comme le montre la figure précédente, la qualification est réalisée par Steria Lille en tant que front office. Ce niveau de tests consiste à vérifier la présence et le bon fonctionnement de l'ensemble des fonctionnalités attendues par le client, donc spécifiées dans les SFD. Ainsi, pour réaliser la qualification, le front office doit parcourir à nouveau les SFD correspondantes. De plus, ce sont les jeux de données fournis avec les SFD, donc les jeux de données envoyés par le client, qui sont utilisés pour réaliser toute la série de tests de qualification.

Enfin, lorsque ces trois étapes de tests ont été effectuées, le front office, Steria Lille, réalise la livraison de l'application au client. Celui-ci peut alors tester à son tour les écrans livrés via sa phase de recette. Une fois cette dernière effectuée, le client retourne un PV (Procès Verbal) à Steria dans lequel sont référencées la livraison ainsi que les différentes remarques faites par le client sur l'application, telles que les anomalies détectées.

#### **6.6.4 Mesures de la qualité de l'application**

Transversalement à la réalisation même de l'application, toute une étude de qualité est réalisée. Au cours de cette dernière, trois principaux dossiers sont utilisés :

- Le PAQ (Plan d'Assurance Qualité)
- La CNS (Convention de Niveaux de Services)
- Le dossier de qualimétrie

Le PAQ permet de décrire les dispositions spécifiques et les moyens mis en oeuvre par le prestataire pour la maîtrise de la qualité des produits et des services du projet. Il décrit également les organisations respectives des intervenants, notamment du prestataire et du client, ainsi que la gouvernance du projet. Le client participe ainsi à la maîtrise de la qualité de l'application en approuvant les exigences et en appliquant les procédures qui le concernent.

Quant au CNS, il permet de compléter le PAQ applicable entre le prestataire et le client. En effet, celui-ci contient les objectifs convenus en termes de qualité de services entre les deux partis, ainsi que les critères d'évaluation et leurs moyens de mesure. Ainsi, seront retrouvés dans ce document de nombreux indicateurs permettant de mesurer la qualité de l'application. Ces indicateurs peuvent être de type organisationnel, tels que les délais de livraison ou les délais de prise en compte des anomalies. Ils peuvent être également de type technique, tels que le nombre de boucles imbriquées dans le code (complexité cyclomatique) ou la quantité de commentaires. Généralement, ces indicateurs techniques sont mesurés à partir de l'outil CAST (cf. : annexe 1).

Enfin, le dossier de qualimétrie CAST est un document dans lequel sont détaillés et explicités les différents indicateurs techniques énumérés dans le CNS. Le client

peut ainsi avoir un aperçu de la qualité du code fourni par le prestataire. En réalité, ce document est mis à jour à chaque fois qu'une livraison chez le client est réalisée. De ce fait, cette organisation permet au prestataire de rectifier rapidement un indicateur si les résultats de celui-ci ne sont pas satisfaisants. Le prestataire dispose ainsi d'une plus grande réactivité. Quant aux deux premiers dossiers, ils sont généralement rédigés au début du projet.

## 6.7 L'architecture matérielle

Au cours de cette partie, je vais présenter l'architecture matérielle de l'entreprise Steria, dédiée au développement des applications, avant d'exposer celle de la chaîne de prêt-à-porter Camaïeu.

### 6.7.1 L'architecture matérielle de Steria

L'architecture matérielle présentée dans cette section est celle utilisée par l'entreprise Steria pour réaliser le développement de l'ensemble des applications dont elle a la charge.

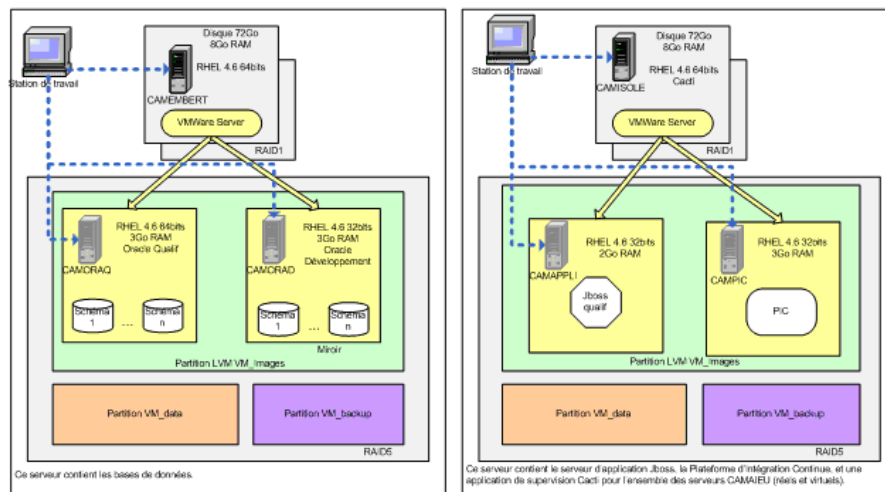


FIGURE 20 – L'architecture matérielle de l'entreprise Steria

Comme le montre la figure ci-dessus, l'architecture matérielle de Steria est constituée de deux serveurs physiques différents :

- Un serveur contenant la base de données
- Un serveur applicatif

Chacun de ces deux serveurs possède deux machines virtuelles, gérées par WMWare. Concernant le serveur de la base de données, la première machine virtuelle est réservée pour le développement, c'est-à-dire que c'est dans cette base de données que seront stockées des données rentrées par les développeurs pour tester les fonctionnalités mises en oeuvre. La seconde machine virtuelle est réservée à la base de données de validation, c'est-à-dire que c'est dans cette base que le client rentrera ses propres données pour effectuer ses propres tests. De plus, chacune de ces machines virtuelles peut utiliser plusieurs instances de la base de données. Par exemple, une instance est remplie par la couche Métier, et la seconde par la couche Présentation. Ceci permet d'éviter que les données de tests des deux couches se superposent ou s'entrecroisent.

Concernant le serveur applicatif, la première machine virtuelle est utilisée pour stocker le serveur Tomcat (et non plus JBoss), alors que la seconde est consacrée à la plateforme d'intégration continue (PIC).

Ces deux serveurs physiques sont architecturés de la même manière :

- Un disque de 72 Go utilisé pour stocker le système d'exploitation UNIX (Red Hat Enterprise Linux). Ce dernier utilise la technologie RAID1. Il fait donc appel à la notion de miroir, ce qui signifie qu'il est totalement dupliqué sur un second disque de même capacité.
- Quatre autres disques, de 160 Go chacun, sont utilisés pour stocker les machines virtuelles et leurs données associées. Ces derniers font appel à la technologie RAID5. Par conséquent, les quatre disques ne sont pas utilisés à 100% de leur capacité pour stocker les données utiles. Le quart de cette capacité est utilisée pour sauvegarder les données redondantes permettant de retrouver les données perdues si un disque est défectueux.

### 6.7.2 L'architecture matérielle de Camaïeu

L'architecture ici présentée est celle de Camaïeu. Il faut savoir que la chaîne de prêt-à-porter n'a pas souhaité nous communiquer beaucoup d'informations sur celle-ci, car il n'est pas attendu que nous intervenions directement sur leurs serveurs. Ils nous ont juste communiqué les technologies qu'ils utilisent, soit :

- Une base de données Oracle (version 10g)
- Un annuaire LDAP Oracle
- Un serveur Tomcat (version 5.5.25)

Voici leur architecture matérielle simplifiée :

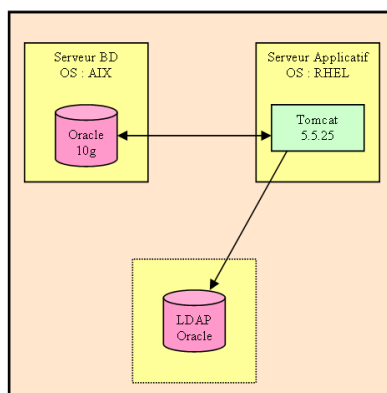


FIGURE 21 – L’architecture matérielle de Camaïeu

Comme le montre la figure ci-dessus, Camaïeu distingue également le serveur de base de données, dont le système d’exploitation UNIX est AIX (Advanced Interactive eXecutive), du serveur applicatif, qui utilise Red Hat Enterprise Linux (RHEL) comme système d’exploitation.

De plus, n’ayant pas beaucoup d’informations sur ces derniers et ne pouvant y avoir accès, Steria doit développer une solution possédant une grande portabilité, afin que celle-ci soit compatible avec le matériel de Camaïeu.

## 6.8 L'architecture logicielle

Tout d'abord, il faut savoir que l'application CPS-V2 possède une architecture n-tiers classique, découpée en 3 couches principales + une couche d'intermédiation située entre la présentation et le métier, comme le montre la figure ci-dessous :

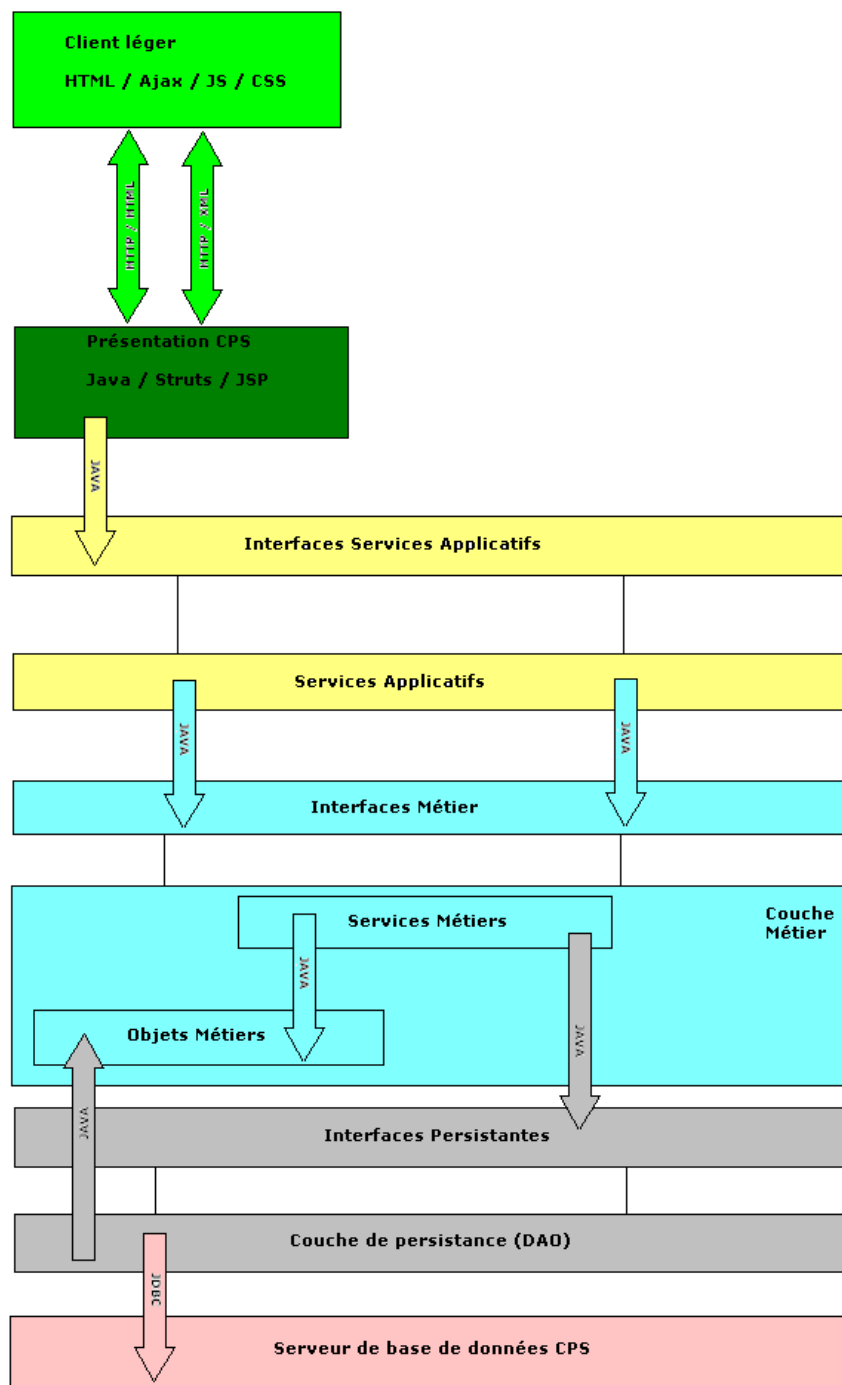


FIGURE 22 – Les couches logicielles de l'application CPS-V2



- **Couche de Présentation :**

Cette couche correspond à la présentation de l'interface graphique, l'enchaînement des pages et la logique applicative. En d'autres termes, c'est la partie visuelle de l'application que pourra observer l'utilisateur. Elle est basée sur le modèle MVC2 et utilise la technologie Struts et les JSP. Cette couche est interconnectée avec la couche applicative.

- **Couche de Services Applicatifs :**

Cette couche assure l'interconnexion entre la présentation et le métier. Par exemple, elle va convertir les objets métiers en objets compréhensibles par la présentation et vice versa. Cette couche permet une séparation très stricte entre présentation et métier puisqu'elle évite qu'une des 2 couches se charge de la conversion des objets.

- **Couche Métier :** C'est véritablement au sein de cette couche que sont réalisées toutes les actions de traitement de l'application CPS-V2. Cette dernière est décomposée en :

- ***Objets Métiers :***

Ce sont les objets du domaine de CPS. Ils correspondent au modèle objet (objets généralement persistants en base de données).

- ***Services Métiers :***

C'est une couche comportant des classes de services. En d'autres termes, cette couche offre un ensemble de services adéquats aux différentes fonctions réalisées par l'application. Ces services sont donc des façades fonctionnelles permettant la manipulation des objets métiers par la couche intermédiaire des services applicatifs.

- ***Les Interfaces métiers :***

comprenant les interfaces des services métiers et des objets métiers.

- **Couche de Persistance :**

Le rôle de cette couche est d'insérer, de mettre à jour, de supprimer, ou de rechercher les objets métiers dans la base de données. Elle contient donc le gestionnaire de persistance des objets métiers (DAO). Cette couche de persistance utilise le framework Hibernate. La connexion de Hibernate avec la base se fait via un driver JDBC.

La couche de Présentation ne dialogue pas directement avec la couche Métier, mais via la couche services applicatifs qui agit en tant qu'intermédiaire pour interconnecter présentation et métier. La couche de présentation appelle les services applicatifs via une interface. On retrouve les mêmes principes sur le dialogue entre la couche métier et la couche de persistance, puisque la couche métier appelle l'interface de persistance. Cette couche de persistance met ensuite à jour les objets métiers en fonction de ce qu'elle a récupéré en base.

Mais en réalité, une autre technologie intervient au niveau de ces interfaces pour rendre possible le dialogue entre ces différentes couches : la technologie Spring.

Reportez-vous à l'annexe 2 pour avoir de plus amples informations sur les principaux frameworks JEE utilisés pour le développement de l'application CPS-V2 : Struts, Hibernate, Spring et Tiles.

## **7 Ma mission**

Au cours de cette partie, je présenterai dans un premier temps le planning réalisé concernant mon stage. Puis je m'appuierai sur ce dernier pour expliciter le travail que j'ai effectué depuis le début de mon stage au sein de l'entreprise Steria.

### **7.1 Le planning réalisé**

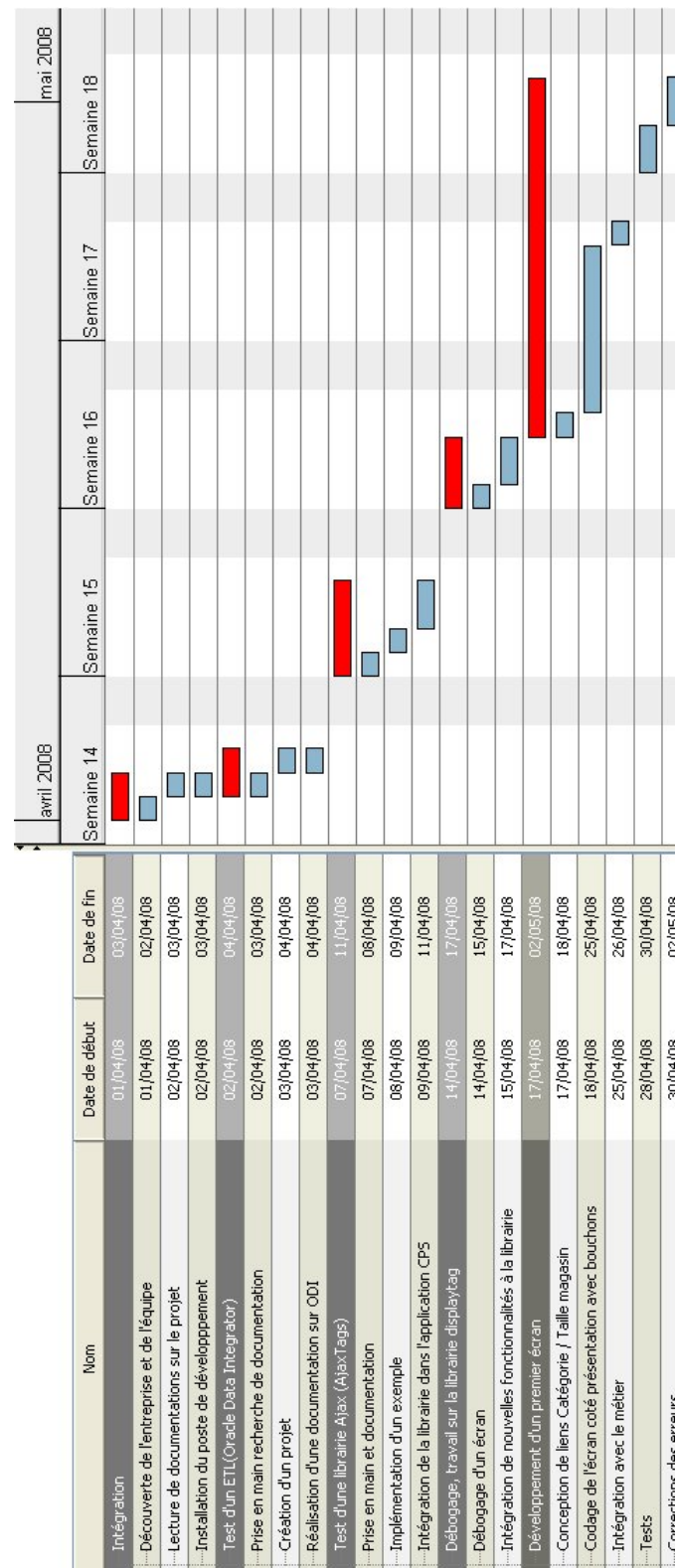


FIGURE 23 – Planning réalisé 1/4

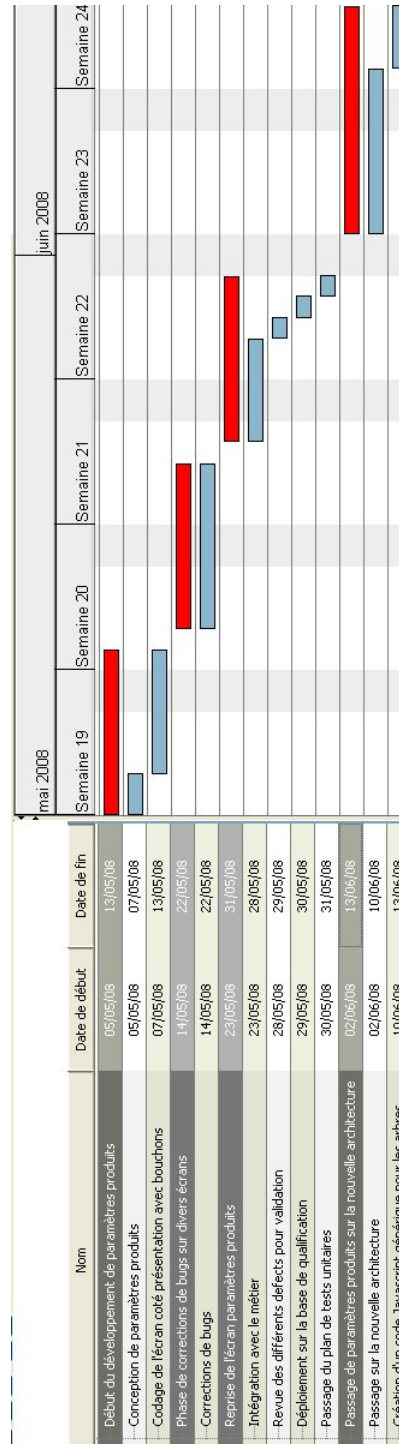


FIGURE 24 – Planning réalisé 2/4

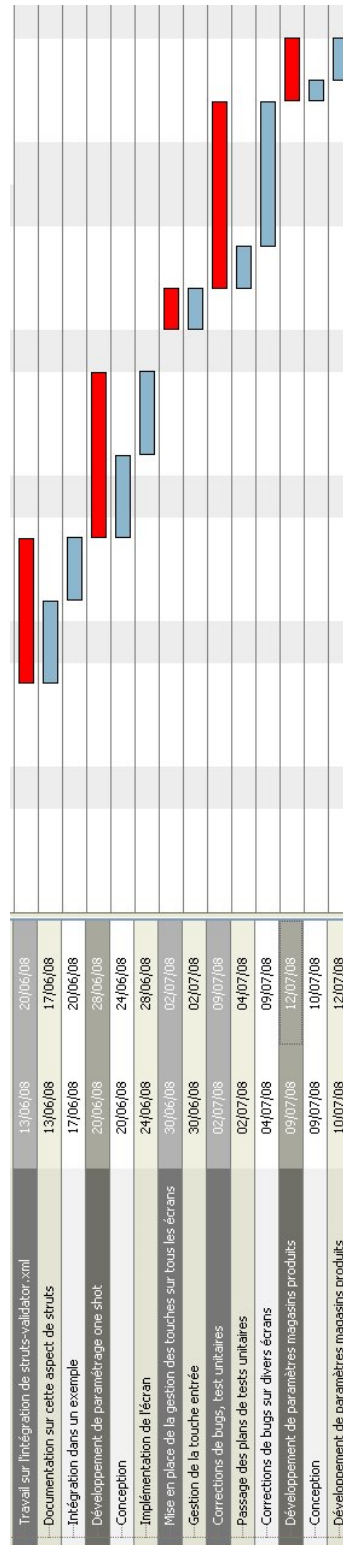


FIGURE 25 – Planning réalisé 3/4

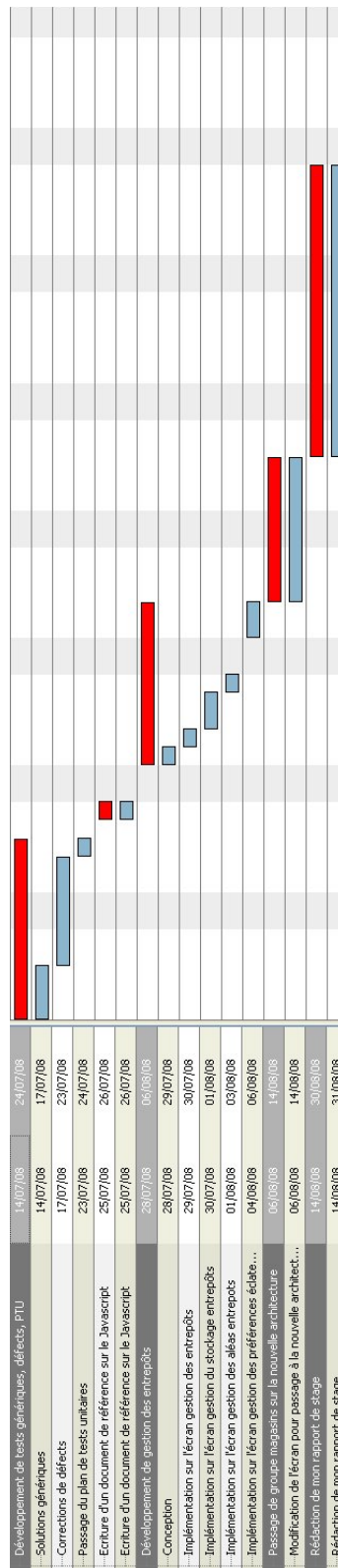


FIGURE 26 – Planning réalisé 4/4

## **7.2 Le travail effectué**

Comme le montre le planning réalisé ci-dessus, après mon arrivé au sein de l'entreprise Steria le 01/04/2008, et après avoir découvert cette dernière ainsi que l'équipe de développement à laquelle j'ai été affecté, j'ai étudié divers documents afin de mettre en place mon poste de développement et de prendre connaissance du projet Camaïeu.

### **7.2.1 La lecture de documents**

Au cours de cette phase, je me suis intéressé, dans un premier temps, au DAL (Dossier d'Architecture Logicielle) de l'application. Ce dernier m'a permis de bien comprendre les différentes couches constituant l'application avec les technologies associées (cf. : 2.3.7. L'architecture logicielle). Dans un second temps, le PPL (Plan de Production Logiciel) m'a été très utile pour installer mon environnement de développement sur la machine qui m'a été confiée. Enfin, dans un dernier temps, j'ai dû me concentrer sur la lecture du guide du socle. En effet, il est important de savoir que l'application que Steria doit développer pour Camaïeu s'appuie sur une base, un socle applicatif, qui a déjà été mise en oeuvre pour des projets de même envergures.



Ce socle technique est constitué :

- D'un framework interne, mis en oeuvre pour développer des applications JEE. Son architecture a été modifiée au cours du projet pour rajouter notamment une couche de services applicatifs. Ce dernier contient les classes de base de chacune des couches techniques :
- **Couche de Présentation :**
  - Classes de base pour les Actions et Forms Struts
  - Classes de base pour les Actions Ajax
  - Ensemble des scripts et feuilles de style par défaut
- **Couche de Services Applicatifs :**
  - Classes de base pour les services applicatifs (Interfaces, classes d'exceptions, classes génériques)
  - Classes de base pour les OP (objets de présentation)
- **Couche Métier :**
  - Classes de base pour les objets métiers
  - Classes de Fabrique (factory) des objets métiers
  - Classes d'administration
  - Classes de base pour les Services métiers
- **Couche de Persistance :**
  - Interfaces de Persistance
  - Classes techniques encapsulant la librairie Hibernate
- D'un ensemble de frameworks externes encapsulés ou assemblés (Struts, Hibernate, Spring)
- D'un ensemble de librairies externes (displaytag, ajaxtags...)
- Des composants d'administration de l'application.

### 7.2.2 Etude d'un ETL : Oracle Data Integrator

Au début de mon projet, on m'a confié l'étude d'un ETL (cf. Glossaire) en l'occurrence Oracle Data Integrator. Le client envisageait à cette époque de nous confier la gestion d'un ETL pour la réalisation des synchronisations massives de données entre deux de ses entrepôts de données qui étaient d'une technologie hétérogène. Cette demande est finalement sortie de notre périmètre mais j'ai quand même étudié cette technologie. L'étude a été réalisée en intégrant un exemple complet de synchronisation de données entre deux de nos bases. Les deux bases utilisées pour l'exemple étaient en SQL et tournaient sous Oracle 10g, la base source et la base de

destination étaient donc exactement similaires en terme de technologie. La transformation consistait donc à la migration des données entre deux structures de tables différentes (j'ai essayé un exemple assez simple ou on migrerait les tuples d'une table vers une autre table possédant des champs différents). L'essentiel de la difficulté de l'utilisation de l'ETL ne venait pas de la transformation en elle même mais plutôt de la configuration très délicate de l'accès à la base et des droits d'accès à cette même base (j'ai eu beaucoup de problèmes liés à l'oubli de l'exécution de certaines requêtes SQL). Voir l'annexe pour un exemple de configuration avec Oracle Data Integrator.

### 7.2.3 Le développement de l'application

Une fois acclimaté au projet je suis passé sur le développement. J'ai particulièrement oeuvré sur la partie présentation et services applicatifs du projet (la partie services applicatifs n'était pas présente avant le changement d'architecture). C'est pourquoi dans cette partie je présenterais deux écran que j'ai développé moi même pour la partie présentation et services applicatifs : l'écran liens catégorie / taille magasins et l'écran gestion des aléas entrepôts.

Voici les écrans où j'ai développé la partie présentation et / ou la partie services applicatifs :

- L'écran liens catégorie / taille magasins.
- L'écran de gestion des paramètres produits
- L'écran de gestion du paramétrage one shot
- L'écran de gestion des paramètres magasins produits
- L'écran de gestion des entrepôts
- L'écran de gestion des aléas entrepôts
- L'écran de gestion des stockages entrepôts
- L'écran de gestion des préférences éclatement

### 7.2.4 Développement de l'écran gestion des liens catégorie / taille magasins

L'écran de gestion des liens catégorie / taille magasins est un écran relativement simple. Il est composé de l'élément d'affichage que j'ai le plus vu jusqu'à présent c.à.d un arbre (j'ai également beaucoup travaillé sur les arbres en développant notamment des javascripts pour eux). C'est le premier écran que j'ai réalisé seul et il est basé sur l'ancienne architecture, ce qui fait qu'il n'y a pas de services applicatifs et que j'ai donc seulement développé le côté présentation.

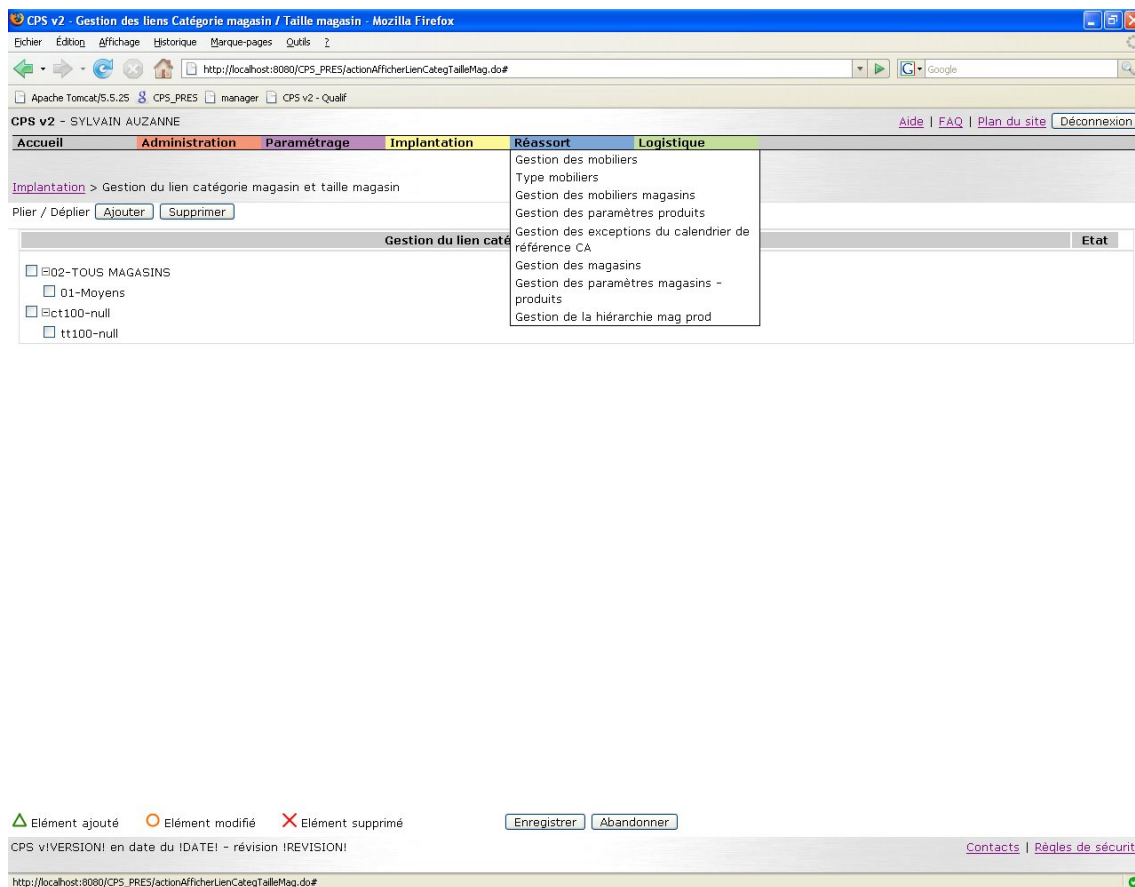


FIGURE 27 – L'écran liens catégorie taille / magasins au démarrage

On voit que l'écran est divisé en 3 parties principales, à savoir :

- Un entête fixe comportant le menu, les infos de connexion (ici Sylvain Auzanne)
- Le corps de la page avec le contenu de l'écran
- Un bas de page fixe lui aussi avec la légende et les boutons pour enregistrer et abandonner

Les bas de pages et entêtes de pages sont fixes lorsque l'on redimensionne la fenêtre ce qui donne un aspect de client lourd malgré l'exécution dans un contexte web.

Comme on peut le voir en haut à gauche de la page le nom et le prénom de la personne connecté sont rappelés. En haut à droite on va retrouver des liens vers de l'assistance technique et le bouton déconnexion pour fermer sa connexion et revenir sur la page de connexion. Juste en dessous on a le menu avec un regroupement par grandes fonctionnalités. On a ensuite une ligne d'information pour repréciser la catégorie et le nom de la page. L'entête de la page suit le même modèle sur toute page du site mis à part l'écran de connexion.

Vient ensuite le corps de la page proprement dit avec les boutons suivant :

- Le bouton Plier / Déplier : il permet lorsque l'on a coché une case (ou plusieurs) dans l'arbre de plier ou de déplier la ou les branche(s) en question
- Le bouton Ajouter : il permet d'ajouter un nouveau lien entre une catégorie et taille magasins et ouvre une popup
- Le bouton supprimer permet de supprimer un ou des liens catégorie / taille magasins

Puis s'affiche l'arbre proprement dit, on peut cocher des cases pour supprimer ou ajouter un nouveau lien, plier ou déplier les branches de l'arbre.

On trouve enfin le bas de la page avec la légende et les boutons enregistrer et abandonner.

Ajoutons maintenant un nouveau lien. On ne fait pas d'appel serveur lorsque l'on affiche juste la popup.

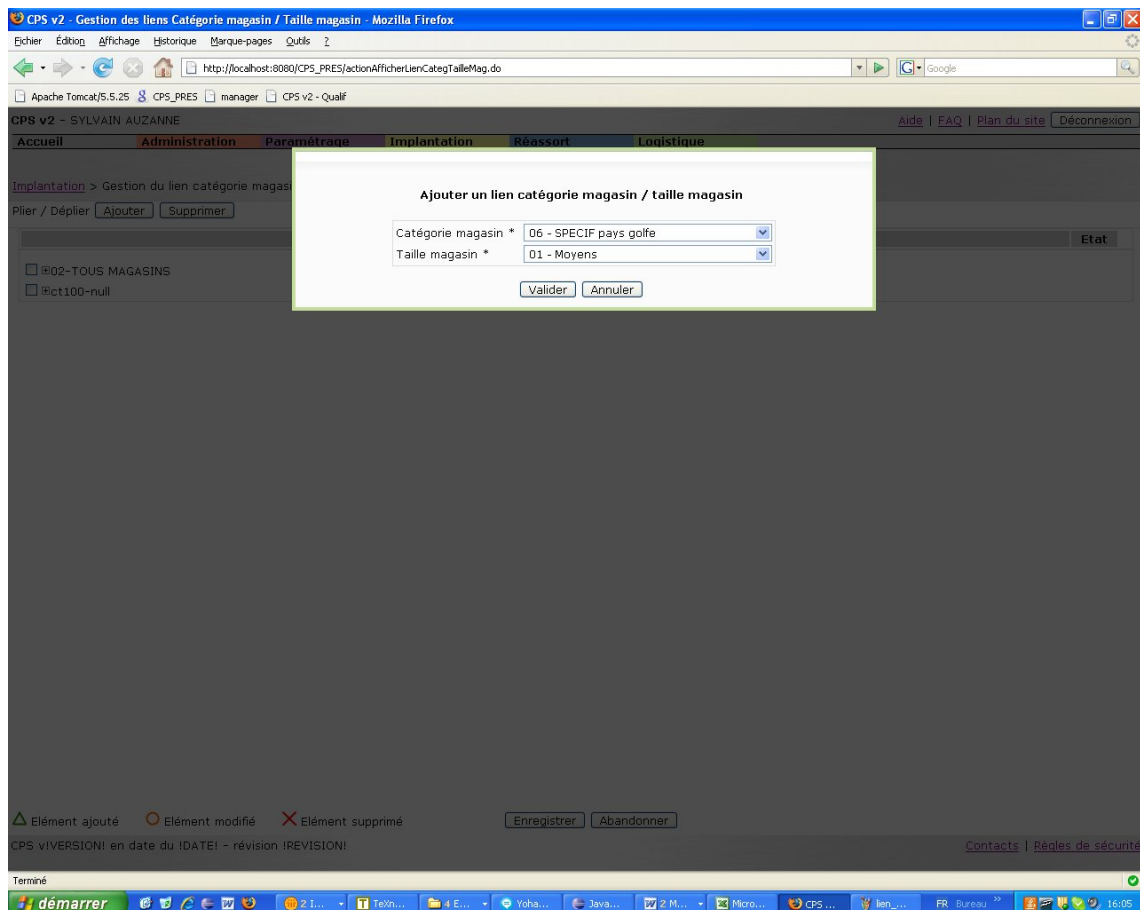


FIGURE 28 – La popup d'ajout d'un nouveau lien

Cette popup permet donc d'ajouter un nouveau lien entre une catégorie et une taille magasin. Comme on peut le voir l'affichage de la popup obscurcit tous les autres éléments et elle empêche effectivement toute action de l'utilisateur en dehors de la popup. L'utilisateur sélectionne une catégorie, lorsqu'il fait sa sélection une requête Ajax est envoyée au serveur et celui-ci va chercher la liste des tailles magasins qui peuvent être associée à la catégorie sélectionnée (il fait pour cela une requête en base à partir de la requête transmise par Ajax). Et cette fonction retourne un flux xml contenant la liste des tailles. Pour mettre à jour la liste déroulante on utilise la librairie AjaxTags (on fait l'appel dans notre page JSP) qui va récupérer le contenu du fichier xml et ajouter ces éléments à la liste.

Une fois que l'on a sélectionné une catégorie et une taille de magasins, on peut alors valider (en cliquant ou en appuyant sur entrée). Lorsque l'on valide, on fait un appel au serveur pour traiter l'action. On va récupérer coté Java les données que l'utilisateur a sélectionné via une remontée d'informations par Struts.

L'arborescence est alors réaffichée :

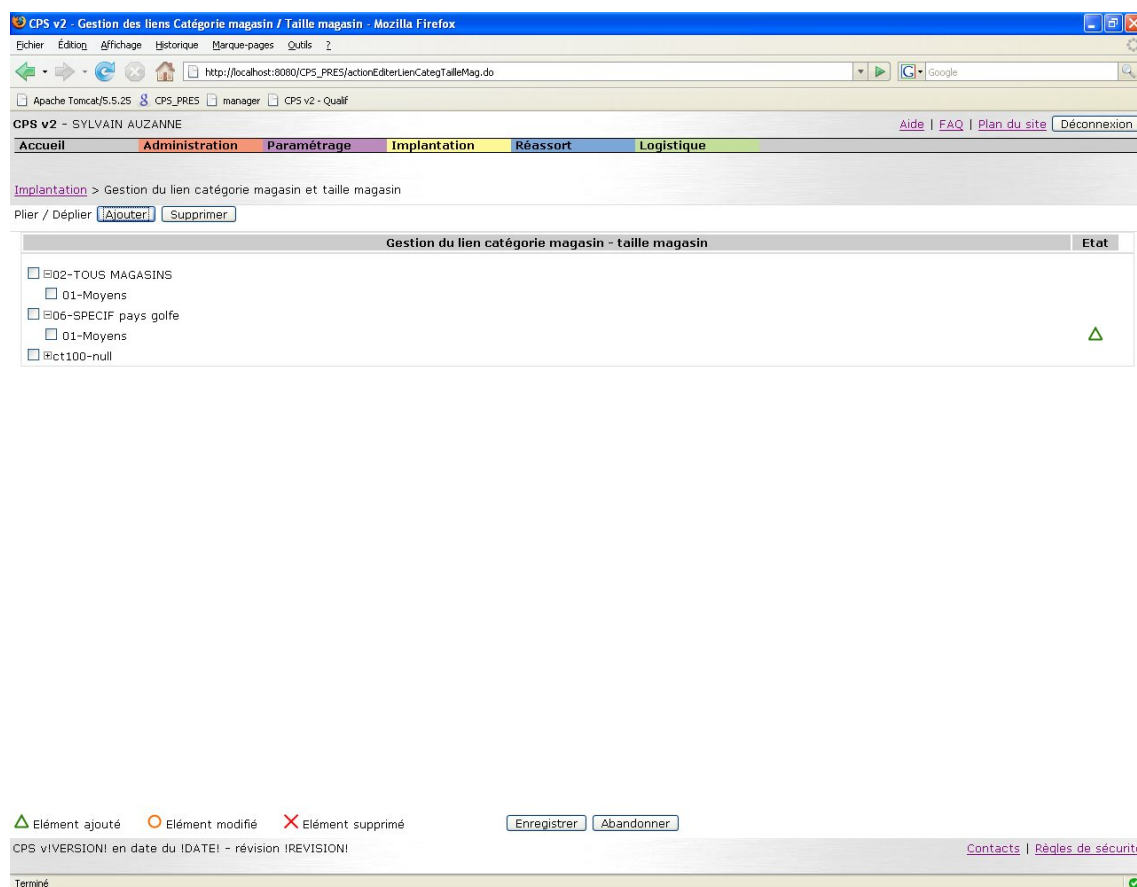


FIGURE 29 – Réaffichage de l'arborescence après ajout

Comme on peut le voir le nouveau lien ajouté apparaît à l'état crée (triangle vert). Cette état de modification est enregistré dans un champ caché et une fonction javascript lancée à chaque affichage de la page affiche l'icone associée. On peut maintenant appuyer sur enregistrer, pour stocker définitivement en base les données saisies.

On va maintenant enregistrer les données saisies :

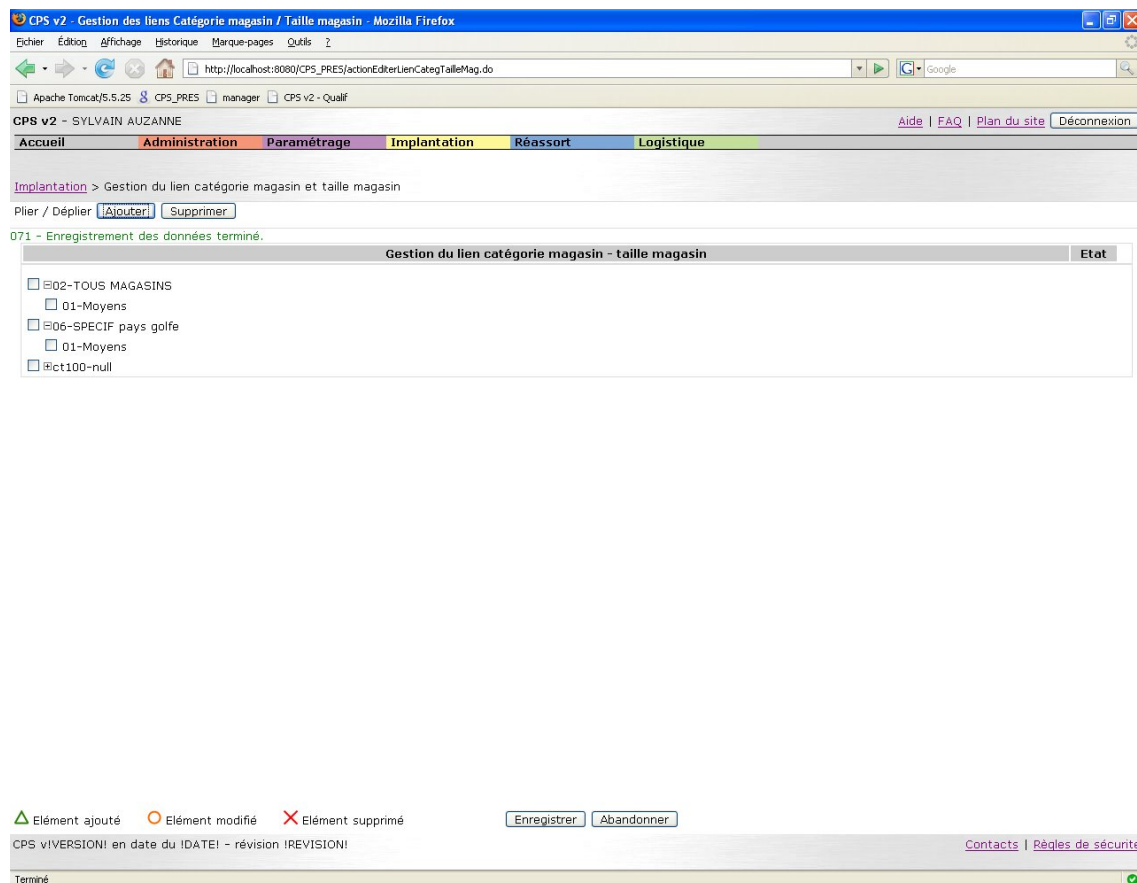


FIGURE 30 – Réaffichage de la page après enregistrement

Une fois l'enregistrement effectué, le message "Enregistrement des données terminées" apparaît, les données ont été insérées en base et on voit que les icones d'état ont disparu.

Regardons maintenant comment sont traitées les erreurs lorsque l'utilisateur fait une saisie éronnée.

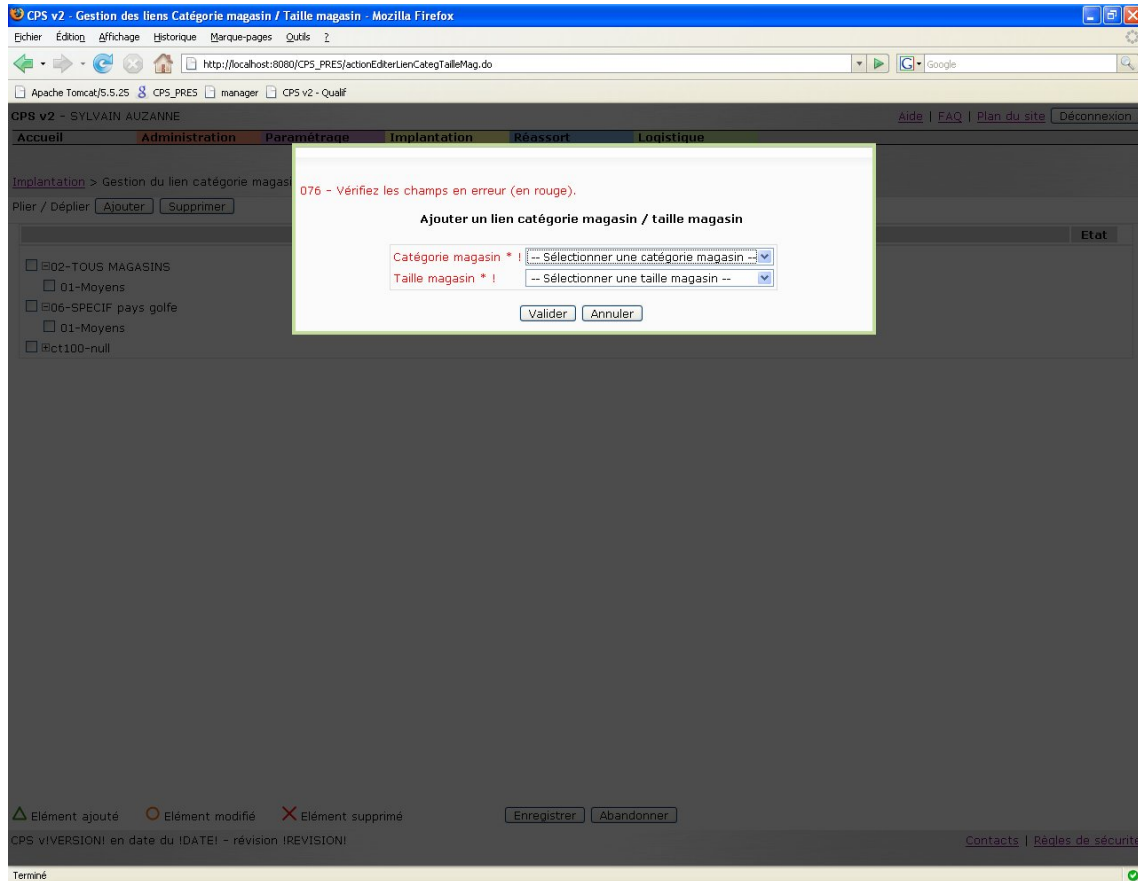


FIGURE 31 – Résultat après saisie éronnée

Ici l'utilisateur n'a sélectionné ni catégorie de magasin ni de taille magasin. Le mécanisme est le suivant : on passe dans l'action d'édition coté Java et celui-ci se rend compte que l'objet rempli par Struts contient des valeurs vides, il lève alors une exception. Dans la page JSP il y a un tag chargé de traiter les erreurs celui-ci se rend compte qu'une erreur a été levée et on affiche alors l'erreur et les champs concernés en rouge. Il y a 2 erreurs affichée et dès qu'il y a 2 erreurs ou plus, le message est générique, on affiche "Vérifiez les champs en erreur (en rouge)".

### 7.2.5 Développement de l'écran gestion des aléas entrepôt

Développé vers la fin de mon stage cet écran est plus complexe et utilise entièrement la nouvelle architecture. Il a pourtant été développé en beaucoup moins de temps (mais entre temps il s'était écoulé 4 mois...). L'écran est donc légèrement différent et n'a pas été développé avec les mêmes méthodes.

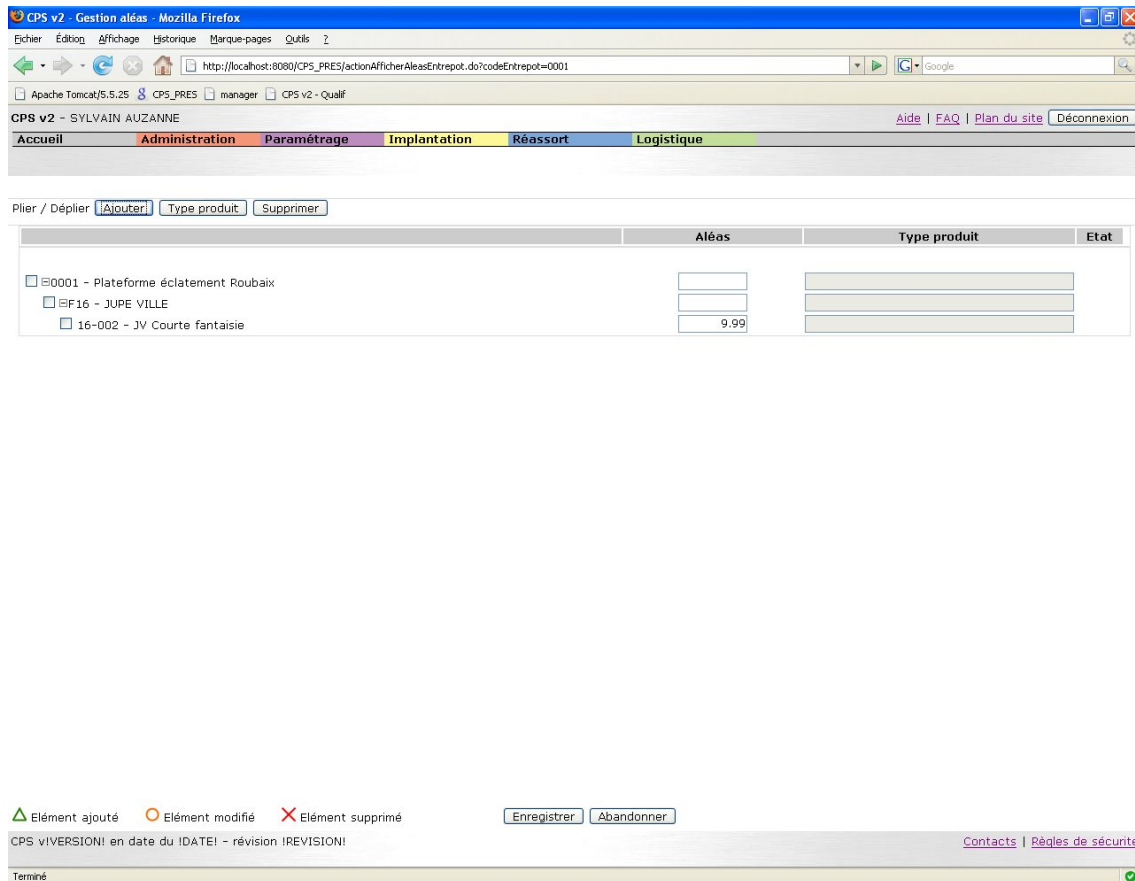


FIGURE 32 – Résultat de l'affichage de la page

Je ne reviendrais pas sur l'organisation de cet écran qui est en tout point similaire à celle de l'écran précédent. On peut juste noter qu'il y a plus de niveaux d'arborescence et qu'il y a un bouton (Type produit) en plus. On peut noter également qu'il y a un paramètre dans la requête pour arriver sur l'écran (codeEntrepot=001). Cet écran a la particularité d'être appelé depuis un autre écran (gestion des entrepôts) et on ne peut pas y accéder sans avoir préalablement passé dans gestion des entrepôts. Il est forcément associé à un entrepôt existant. Lorsque la requête est effectuée, on cherche dans le code Java associé à la requête actionAfficherAleasEntrepot.do, on récupère la requête et on cherche l'entrepôt associé. Notons enfin sur cet écran que le bouton abandonner renvoie sur gestion des entrepôts.



En ce qui concerne la représentation des données, il faut bien comprendre que la représentation est "virtuelle" dans le sens où chaque niveau d'arborescence ne correspond pas à une entrée dans la base. Là sur la capture d'écran spécifié un seul tuple comportant un code entrepôt, un code famille et un code sous famille avec une valeur de 9.99 est enregistré. Le dernier niveau d'arborescence comporte forcément une valeur. En fait, ici, à chaque ligne correspond un enregistrement si et seulement si une valeur y est associée.

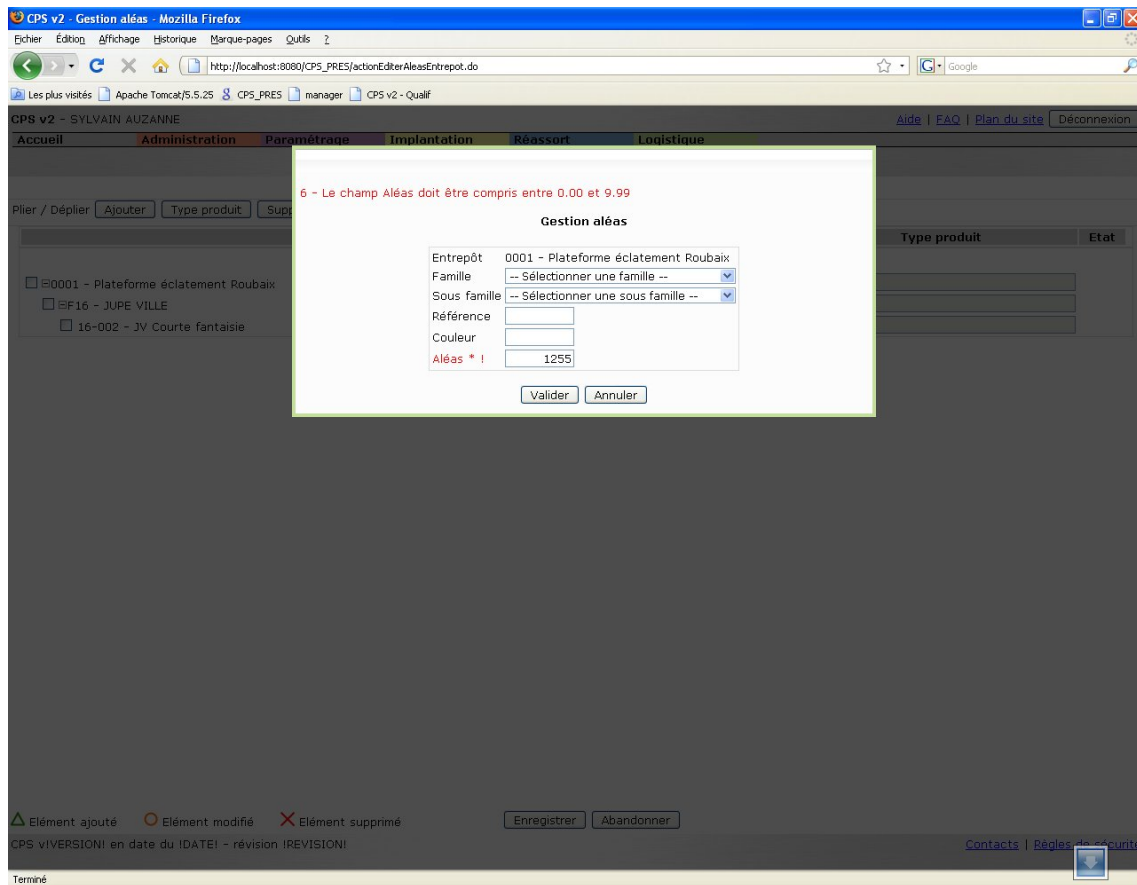


FIGURE 33 – Résultat après une saisie erronée dans la popup d'ajout

La popup d'ajout ressemble beaucoup à celle de liens catégorie taille magasins. Mais la différence vient du fait qu'il y a un contrôle de surface qui est effectué avec struts validator. Struts validator s'écrit en XML. Contenu de validation.xml :

```
<form name="formAleasEntrepot">
  <field property="objetPlatPresentation.valeur.aleas" depends="required,double,
    <arg0 key="error.champ.aleas"/>
    <arg1 name="doubleRange" key="{var:min}" resource="false"/>
    <arg2 name="doubleRange" key="{var:max}" resource="false"/>
    <var><var-name>min</var-name><var-value>0.00</var-value></var>
    <var><var-name>max</var-name><var-value>9.99</var-value></var>
  </field>
</form>
```

On utilise également un fichier `cps_frameworks.properties` (où l'on stocke les messages d'erreurs) :

```
errors.range=6 - Le champ {0} doit être compris entre {1} et {2}
error.champ.aleas=Aléas
```

Le nom du formulaire correspond au nom du formulaire défini dans le fichier de configuration struts. Le field property définit le nom (dans la page jsp) du champ qui va être vérifié, l'argument depends indique les validation qui vont être faites dessus (required = l'utilisateur doit saisir quelque chose, double=la saisie doit être un double, doubleRange=ce nombre doit être dans un intervalle spécifique). Les balises arg vont venir compléter les champs 0, 1 et 2 avec leur attribut key. Tout les contrôles de surfaces sont effectués avec struts validator et pour les autres contrôles (tels que la présence de doublons) on les effectue dans le code Java.

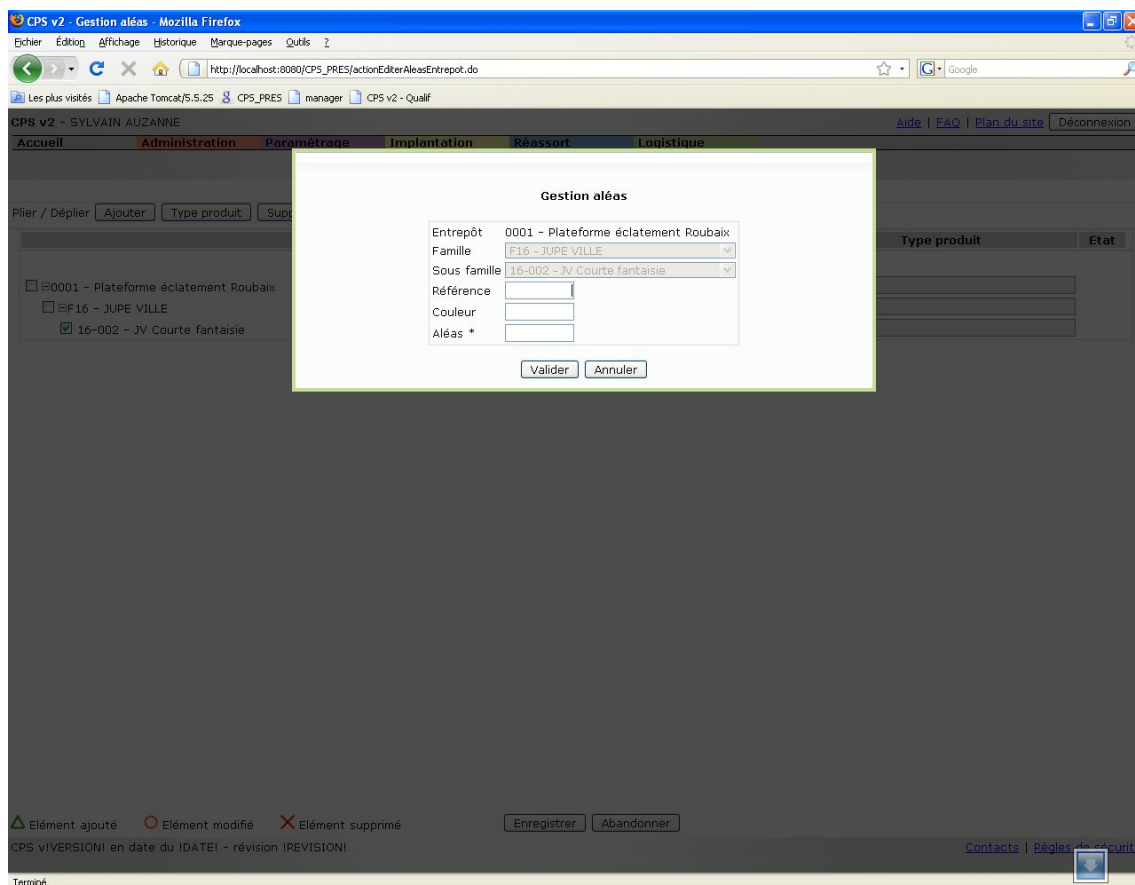


FIGURE 34 – Ouverture d'une popup en ayant sélectionné des éléments dans l'arbre

Si l'utilisateur sélectionne un niveau d'arborescence (un et un seul) les éléments de la popup vont être présélectionnés et grisés. On utilise pour cela un script générique (que j'ai développé) et qui déclenche également les requêtes ajax (ici les sous familles sont chargées dynamiquement en fonction de la famille sélectionnée).

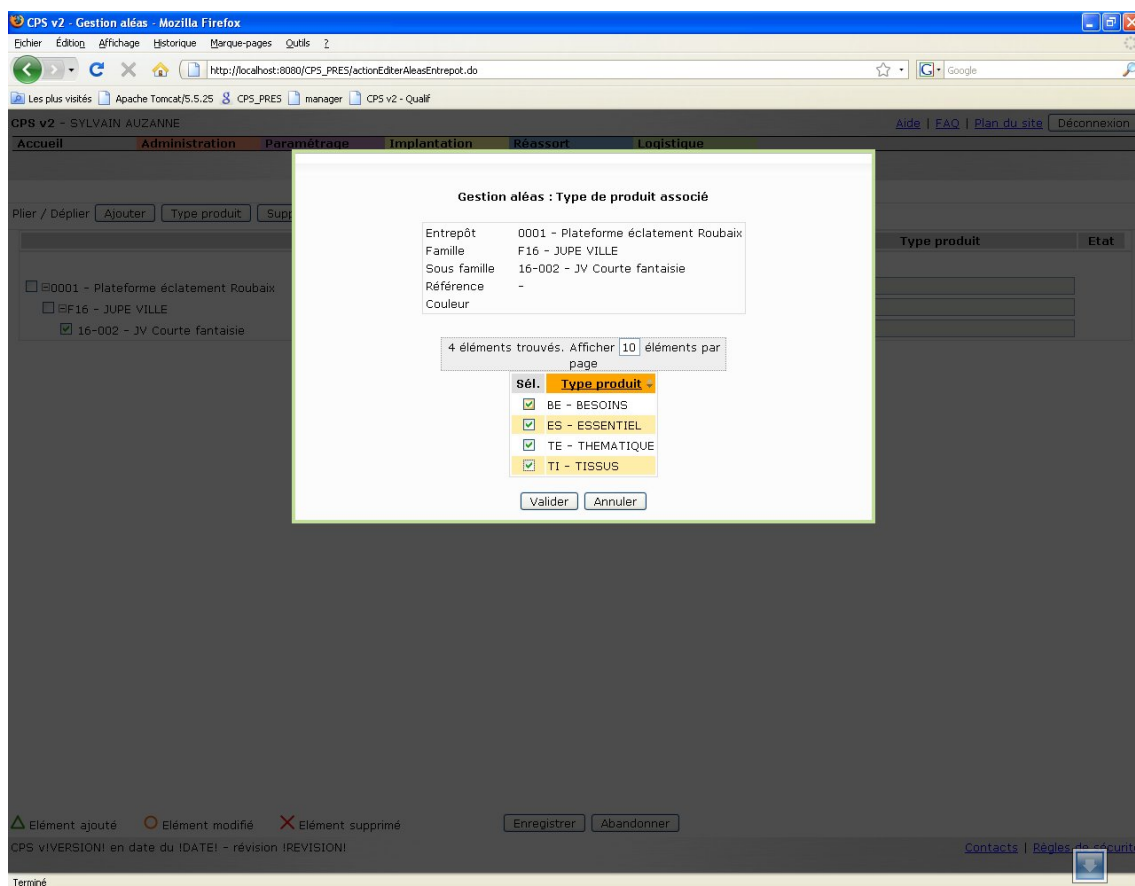


FIGURE 35 – Popup pour associer un type produit

La popup d'association d'un type produit permet d'associer plusieurs type produits au niveau d'arborescence sélectionné. On utilise la librairie displaytag pour générer le tableau (ce tableau est dynamique car on peut sélectionner sa page, le nombre d'éléments affichés par page, la page en cours). Cette popup a également pour particularité de ne pas faire appel au serveur lorsque l'on valide, ici tout est fait en Javascript.

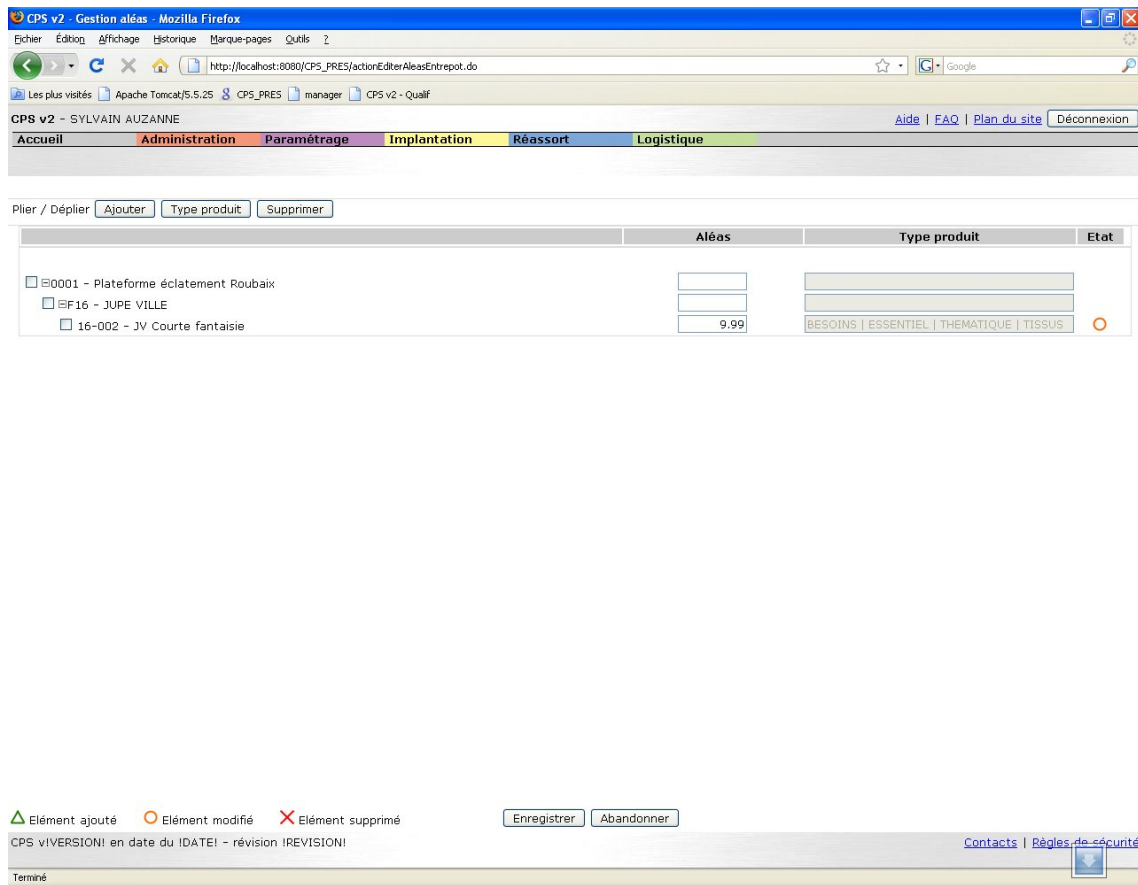


FIGURE 36 – Contenu de la page après validation de la popup Type produit

Comme on peut le voir des types produits ont été associés au niveau d'arborescence sélectionné (on indique ici les libellés). La ligne concernée comporte un rond orange qui indique qu'il y a eu des modifications.

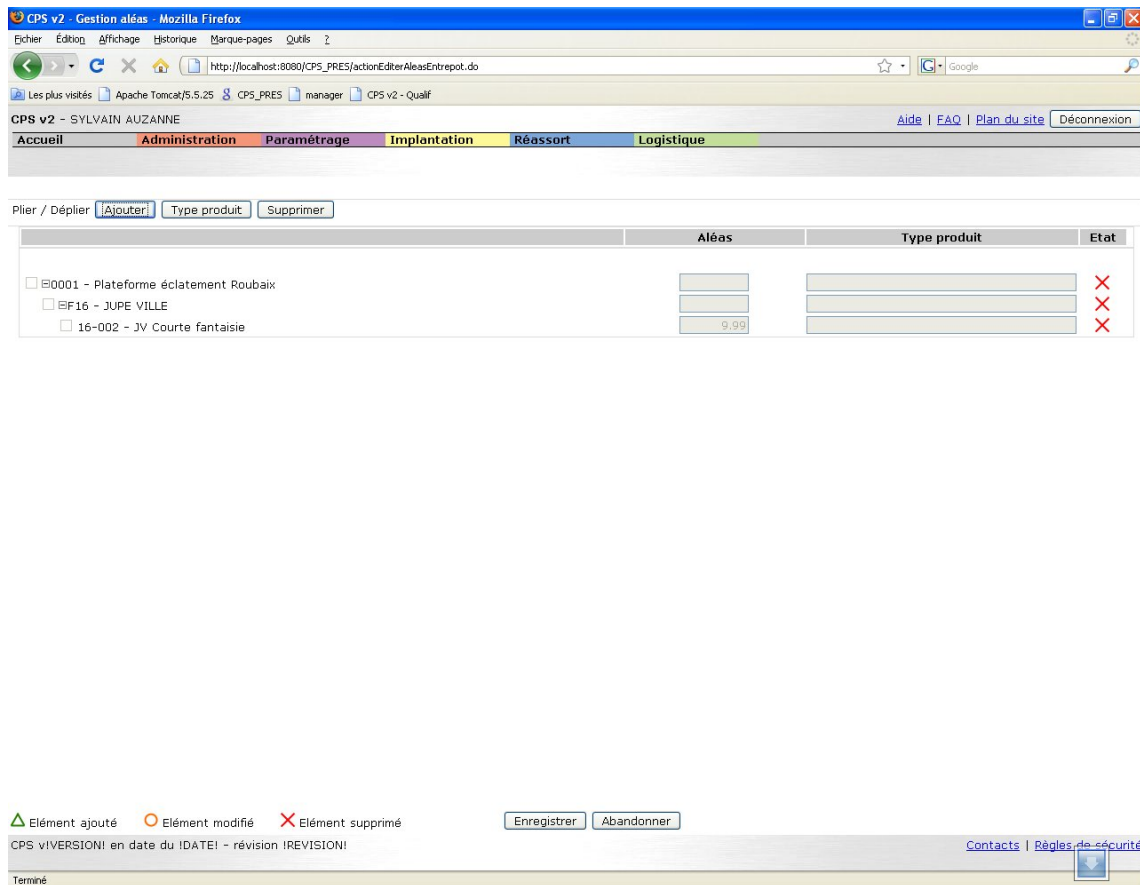


FIGURE 37 – Suppression de niveaux de paramétrages

L'utilisateur a cliqué ici sur la case à cocher du premier niveau de l'arborescence et a cliqué ensuite sur supprimer. La suppression est ici récursive, les niveaux enfants sont mis à supprimés jusqu'à la racine. Il faut bien noter qu'on ne supprime pas un entrepôt, une famille ou une sous famille, mais bien un niveau de paramétrage combinant les 3 paramètres. La suppression n'est effective qu'après avoir cliqué sur enregistrer. Pour l'utilisateur tout est transparent mais le code (contenu dans un champ caché) ne sera pas toujours le même (on a les primitives SUPPR\_SOULHAITEE, SUPPR\_CREE). Par exemple si l'utilisateur vient de créer un élément et que l'utilisateur veut annuler en les supprimant, l'état sera SUPPR\_CREE un état particulier ou l'on effectue aucune action sur les objets étant donné qu'ils n'ont jamais été enregistré en base.

### 7.2.6 Les difficultés rencontrées

J'ai évidemment rencontré des difficultés sur le projet qui étaient de 2 types : la compréhension des specs et des problèmes techniques divers. Mais j'avais la chance d'être dans une équipe très sympathique et toujours prête à m'aider.

J'ai rencontré évidemment le plus de difficultés au début de mon stage. Tout d'abord il fallait appliquer les étapes du PPL (plan de production logicielle) pour installer mon environnement logiciel. Cette étape m'a posé pas mal de problèmes car le PPL n'était pas tout à fait à jour (il avait été fait au début du projet et évidemment certaines évolutions n'étaient pas indiqués). J'ai eu beaucoup de difficultés pour stabiliser mon environnement Eclipse (gros problèmes de synchronisation avec SVN notamment).

Il fallait s'habituer à l'environnement de développement car même si j'avais vu la plupart des outils en cours, je ne n'avais pas forcément l'expérience nécessaire pour en faire un usage correct. Il fallait également découvrir et comprendre le fonctionnement de nombreux frameworks (Struts, Spring) que je n'avais jamais pratiqué. Au départ j'ai beaucoup procédé par mimétisme mais cette stratégie a des limites et je me suis retrouvé assez vite obligé de poser des questions.

Tout au long du projet j'ai eu des difficultés avec les conceptions car elles étaient peu précises et n'utilisaient aucun formalisme particulier (elle étaient en Français). Leur problème principal, c'est qu'elles n'abordent que le cadre général mais jamais les problèmes (et la réponse que l'on doit apporter) qui peuvent survenir dans le cas d'utilisations particulières de l'écran. Les conceptions étaient réalisées par le client mais validées par l'équipe de Steria au contact avec le client à Lille. Cette situation entraînait pas mal d'interprétation erronée et j'ai dû écrire un certains nombres de FQR (fiche de questions réponses qui permettent le contact avec le client) pour comprendre les conceptions.

Dans le développement des écrans il y avait 2 difficultés principales : la mise en conformité avec l'ergonomie générale et l'intégration avec le métier. Pour ce qui est de la mise en conformité avec l'ergonomie ce qu'il faut voir c'est que bien qu'exécuté dans un client léger, l'équipe de camaïeu a une culture de client lourd. Du coup la présentation est très contrainte (entête et pied de pages fixes, scrolling possible que sur certaines zones de l'écran) et impose des comportements peu naturels dans un navigateur web (utilisation du clavier). Parallèlement à ça le cadre d'exécution du logiciel n'est pas fixé entre IE 6 et IE 7 et cela pose pas mal de problème d'affichage (utilisation de fichiers CSS différent selon le navigateur). Cela a nécessité la réalisation de très gros fichiers Javascript avec de nombreuses fonctions et il était évidemment assez facile d'oublier certains éléments. La deuxième difficulté était l'intégration avec le métier. Au départ de mon stage je faisais beaucoup de bouchons lorsque le métier n'était pas développé. Malheureusement dans les écrans issus de l'ancienne architecture il y avait une très mauvaise séparation entre le métier et la

présentation (par exemple Struts faisait remonter les infos directement sur un objet métier) et du coup le retrait des bouchons et l'intégration avec le métier était souvent très couteuse et longue à réaliser.

Les difficultés techniques que j'ai pu rencontrer au début de mon stage se sont fortement estompées avec l'acquisition de connaissances et compétences dans le domaine du JEE, notamment sur de nombreux frameworks le constituant, principalement ceux concernant la couche Présentation, tels que Struts, Tiles, DisplayTag, AjaxTag, et Spring. Les autres difficultés telles que l'interprétation des conceptions ou la conformité de l'écran avec les règles d'ergonomie générale restent encore aujourd'hui des difficultés sérieuses mais qui se sont estompées avec mon expérience sur le projet.



## Sixième partie

# Conclusion

Ces 5 mois de stage au sein de Steria m'ont permis de découvrir la réalité d'un projet en milieu professionnel. J'ai dû affronter les difficultés liées à un projet informatique en général et celles plus spécifiques au projet Camaïeu. Le projet Camaïeu était un projet court mais très intense et il a fallu produire des livraisons de qualité dans des délais impartis relativement tendus.

J'ai pu découvrir en participant au développement du projet la rigueur nécessaire à la satisfaction du client et les bonnes pratiques nécessaires à la réalisation d'un logiciel de qualité. Je me suis également rendu compte de l'importance de la coopération dans l'équipe et du dialogue avec le client. D'un point de vue technologique, le projet a été très enrichissant puisque j'ai étudié et utilisé un vaste panel de technologies (Struts, Spring, Hibernate, Tiles) et d'outils (Maven, Tomcat, SVN) autour du JEE mais aussi des ETL avec l'utilisation de Oracle Data Integrator. Mes missions ont été variées et intéressantes et même si j'ai beaucoup oeuvré sur la partie présentation, j'ai également travaillé sur des aspects métiers notamment à la fin de mon projet.

Ce projet a donc été extrêmement enrichissant pour moi. J'ai énormément appris sur le plan technologique mais aussi en termes d'organisation de projet, d'autant plus que le rythme était extrêmement tendu. Il était également très intéressant de participer depuis le début des développements (peu d'écrans avaient été développés à mon arrivée) jusqu'aux portes de la livraison (livraison importante le 22 septembre). Je n'aurais jamais pu m'intégrer aussi facilement sur ce projet si l'équipe n'avait pas, tout au long du projet, été extrêmement soudée et toujours là pour m'aider. Ce stage a donc été une excellente assise pour ma future vie professionnelle et je pense que les connaissances acquises me serviront aussi bien à court terme dans le cadre de la poursuite du projet Camaïeu qu'à plus long terme sur d'autres expériences de développement.

## Septième partie

# Annexes

## 8 Annexe 1 : Les outils de qualimétrie et de tests

Dans cette annexe sont présentés les principaux outils utilisés par Steria pour mesurer la qualité de l'application et maîtriser les risques du projet. Parmi ces derniers, nous retrouvons :

- L'outil de qualimétrie CAST
- Test Director (TD)
- La plateforme d'intégration continue (PIC)

### 8.1 Cast : outil de qualimétrie

Cet outil permet de gérer le patrimoine applicatif. Il permet de mesurer la qualité du code, tel que le respect des normes et des conventions de codage. Pour cela, il produit automatiquement une batterie d'indicateurs agrégés en facteurs de santé, comme la portabilité, la robustesse, la performance, la maintenabilité, et la sécurité du code.

Cet outil permet également de créer une cartographie de l'application, dans le but de montrer et de connaître les impacts d'une modification du code.

Cet outil est principalement utilisé à trois instants clé au cours de l'avancement du projet :

- A la fin de la mise en place de l'architecture :  
Sont ainsi relevés les éventuels problèmes du socle technique et des modèles de développement, avant la généralisation à l'ensemble du logiciel
- Deux semaines après la production en masse du logiciel :  
Ceci permet de s'assurer que tous les développeurs maîtrisent les principes et consignes mis en place lors de la phase de conception.
- Avant toutes livraisons majeures à Camaïeu :  
Ceci permet de s'assurer de la qualité finale du logiciel et ainsi, donner à Camaïeu la visibilité sur les facteurs de santé de l'application produite.

## 8.2 Test Director (TD)

Cet outil permet d'industrialiser la phase de tests, dans le but de diminuer au maximum le nombre d'anomalies que Camaïeu pourrait détecter lors de sa propre phase de recette.

Il permet tout d'abord de gérer des référentiels de tests à partir des exigences fonctionnelles et techniques, définies au préalable. L'équipe projet doit également définir auparavant la stratégie de tests, créer les jeux de tests, et les saisir dans Test Director.

Cet outil permet aussi de gérer les anomalies décelées. Chaque campagne de tests est tracée. Test Director peut alors élaborer des fiches d'anomalies.

Enfin, utilisé avec Cast, il permet d'identifier des cas de tests à rejouer en cas de modification de code.

## 8.3 La PIC (Plateforme d'Intégration Continue)

La plateforme d'intégration continue est un ensemble d'outils permettant de réaliser périodiquement une compilation totale du projet et de lancer des phases de tests sur ce dernier. Ainsi, cet outil permet de retourner un rapport d'état du projet. L'équipe de développement peut ainsi connaître l'état actuel du projet, de relever les erreurs de compilation, ainsi que les éventuels dysfonctionnements de l'application. Attention, la PIC ne permet pas de valider les fonctionnalités de l'application, du fait que ces dernières soient spécifiques à chaque projet.

Voici les différentes technologies que la PIC peut être amenée à utiliser :

Technologie	Descriptif
Maven	Framework de management projet capable de réaliser les tests automatiques via les classes Junit, de créer les scripts de compilation, de déploiement, de mesurer la couverture des tests lancés
<i>Cobertura Plugin</i>	Mesure la couverture des tests
<i>CheckStyle plugin</i>	Vérifie le respect des standards et des bonnes pratiques
<i>PMD plugin</i>	Audit de code
Continuum	Intégration en continu
Archiva	Repository de librairie, permettant d'unifier les librairies sur l'ensemble des projets ou projet par projet.
JDpend	Vérifie les dépendances entre les couches
Cactus	Extension de Junit pour simplifier les tests avec le serveur
JunitPerf	Extension de Junit orientée mesures de performances
DBUnit	Extension de Junit pour tester les bases de données
JMeter	Test de charge et de stress des applications
HttpUnit	Simulation d'un navigateur

FIGURE 38 – Les technologies de la PIC

Dans le cas du projet Camaïeu, la plateforme d'intégration continue utilise comme outil principal : Maestro.

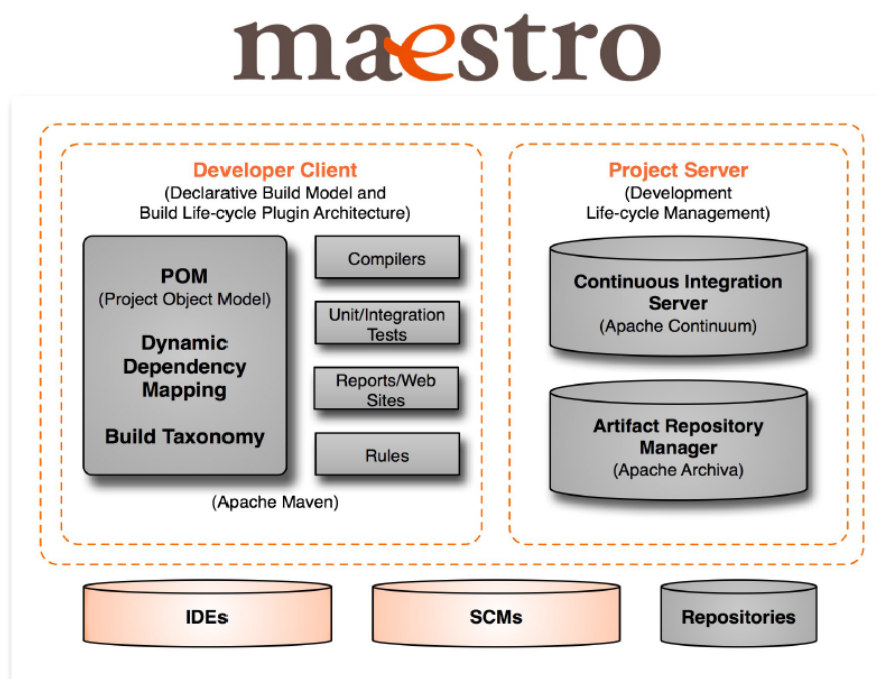


FIGURE 39 – Schéma de l'outil Maestro

Comme le montre la figure ci-dessus, Maestro possède un côté serveur, sur lequel peut se connecter chaque projet de Steria, et un côté client associé à chaque projet. Le côté client fait appel à l'outil Maven qui compile le code récupéré à partir de SVN. Quant au côté serveur, il utilise deux autres outils importants : Continuum et Archiva. L'outil Archiva permet de récupérer les différentes librairies nécessaires à la compilation du projet effectuée par Maven à partir des Repositories. Enfin, l'outil Continuum constitue l'ordonnanceur de la plateforme d'intégration continue. Ce qui signifie que c'est lui qui donne l'ordre de compilation, et qui lance les phases de tests réalisées par des outils externes venant se greffer à Maestro. Le tout constitue la PIC.

## 9 Annexe 2 : Les principaux frameworks JEE

Les principaux frameworks JEE présentés dans cette annexe sont les suivants :

- Struts : framework de la couche Présentation
- Hibernate : framework de la couche de Persistance
- Spring : framework réalisant les interactions entre les différentes couches de l'application
- Tiles : frameworks de la couche Présentation

### 9.1 Struts

Struts, sous son véritable nom, "Apache Struts", est un framework open source développé par la Apache Software Foundation utilisé pour faciliter le développement des applications web JEE. Son but premier est de faciliter la mise en oeuvre d'une architecture MVC (Modèle-Vue-Contrôleur). Pour cela, Struts combine deux technologies : JSP et Servlets, dans le but de séparer la présentation, les données, et les transactions. Ceci permet donc d'obtenir une meilleure subdivision et structuration du code d'une application web. Par conséquent, la maintenabilité et la modularité de l'application pour des développements futurs sont facilitées.

Pour notre application, Struts est essentiellement utilisé dans la couche Présentation. Ceci, permet donc de réaliser, à ce premier niveau, une première distinction entre l'IHM et les traitements.

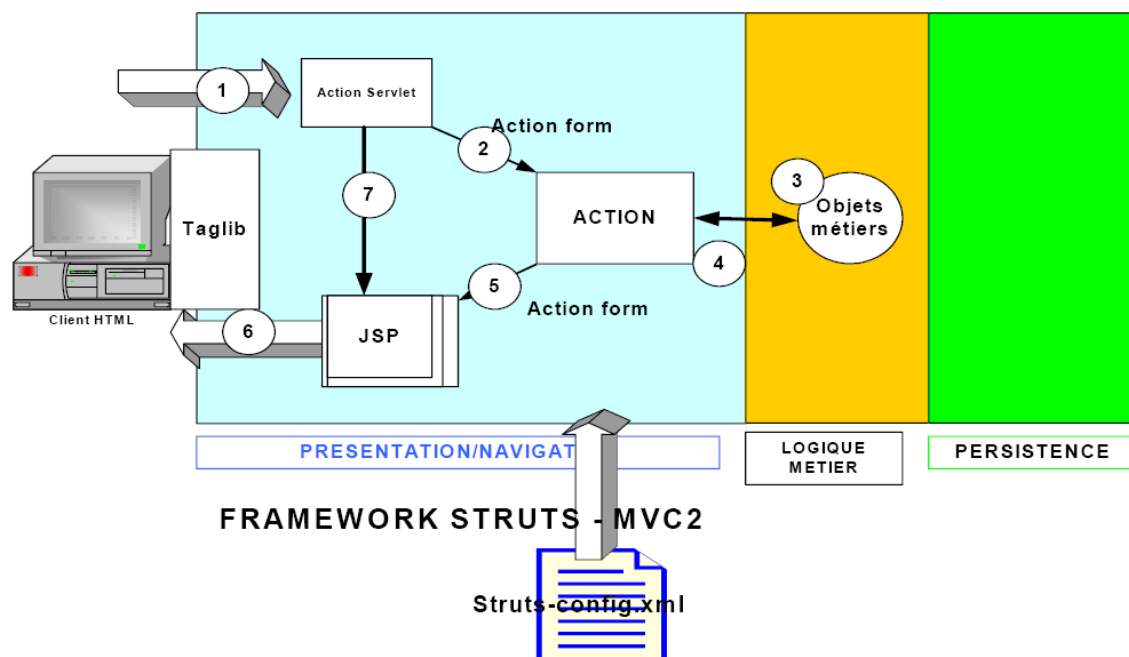


FIGURE 40 – Schéma du framework Struts

Si on associe la figure ci-dessus avec le modèle MVC, nous avons les correspondances suivantes :

- L’Action Servlet constitue le Contrôleur
- La JSP constitue la Vue
- L’Action constitue le Modèle

**1)** Le contrôleur, soit l’Action Servlet, représente le coeur de la couche Présentation, car toutes les requêtes du client transitent par lui. L’Action Servlet consulte le fichier de configuration Struts-config pour savoir vers quelle action rediriger la requête

**2)** Si la requête du client contient des paramètres, notamment lors de saisies de champs d’un formulaire, les paramètres sont envoyés dans un objet de type ActionForm.

**3-4)** Le modèle, appelé Action, réalise les différents traitements en fonction de la requête, et peut faire appel à la couche Métier si nécessaire.

**5-7)** Une fois de plus, le contrôleur fait appel au fichier Struts-config pour savoir vers quelle JSP rediriger la réponse. De plus, si l’action a besoin de renvoyer des paramètres à la JSP, elle peut le faire, via un objet de type ActionForm.

**6)** La réponse est renvoyée au client.

C’est toujours à partir du fichier de configuration Struts-config que le contrôleur sait quel objet de type ActionForm est associé à quelle Action.

Les objectifs de ce framework de Présentation sont donc de fournir un cadre de travail constitué d’un ensemble de classes génériques afin de :

- Faciliter les développements en implémentant dans les classes de base les comportements génériques standards et les méthodes squelettes, en laissant aux sous-classes le soin d’implémenter les méthodes abstraites.
- Homogénéiser les développements.
- Permettre une meilleure évolutivité et maintenance : une modification sur les classes de base du framework permettra à toutes les sous-classes de bénéficier des nouvelles fonctionnalités.
- Gérer de manière homogène les erreurs (exceptions renvoyant sur les pages d’erreurs).

## 9.2 Spring

Spring est un framework open Source JEE pour les applications n-tiers facilitant leur développement ainsi que leurs phases de tests. Il faut savoir que Spring est considéré comme un conteneur léger. En voici, une définition :

*"SPRING est un conteneur dit "léger", c'est-à-dire une infrastructure similaire à un serveur d'application JEE. Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets. Le gros avantage par rapport aux serveurs d'application est qu'avec Spring, vos classes n'ont pas besoin d'implémenter une quelconque interface pour être prises en charge par le framework (au contraire des serveurs d'applications JEE et des EJBs). C'est en ce sens que SPRING est qualifié de conteneur "léger". "*

Spring s'appuie principalement sur l'intégration de trois concepts clés :

- Inversion de contrôle ou injection de dépendances (cf : Glossaire)
- Programmation orientée aspect
- Couche d'abstraction

La programmation orientée aspects sépare les considérations techniques des descriptions métiers dans une application. Elle va donc permettre d'extraire les dépendances entre modules et de gérer depuis l'extérieur ces modules en les spécifiant dans des composants du système à développer, nommés "aspects".

Quant à la couche d'abstraction, elle permet d'intégrer d'autres frameworks et bibliothèques avec une plus grande facilité. En effet, grâce à cette dernière, Spring ne concurrence pas d'autres frameworks dans une couche spécifique d'un modèle architectural MVC, mais s'avère être un framework multi-couches pouvant s'insérer au niveau de toutes les couches : modèle, vue, et contrôleur. Cette couche d'abstraction permet donc à Spring d'intégrer Hibernate pour la couche de Persistance, ou encore Struts pour la couche Présentation.

Il faut savoir que Spring propose une multitude de façon d'utiliser son conteneur léger. Une des plus utilisée est celle faisant appel aux notions de fabrique de Beans et de contexte d'application, permettant de dialoguer et de configurer son conteneur léger.

La fabrique de Beans (BeanFactory) est l'interface de base permettant aux applications reposant sur le conteneur léger de Spring d'accéder à ce dernier. Elle définit les fonctionnalités minimales dont dispose l'application pour interagir avec celui-ci. Les méthodes proposées par cette interface permettent donc d'interroger le conteneur concernant les objets dont il à la charge.

### **Pourquoi utiliser Spring ?**

Spring est composé de plusieurs modules disponibles sous forme de fichiers jar. Ainsi, on peut n'ajouter au projet que la partie que l'on souhaite utiliser. De plus, Spring fournit des services de type fonctionnel comme par exemple la gestion de transactions.

Voici quelques uns des avantages de Spring :

- Découplage des composants logiciels. Moins de dépendances entre les différents modules
- Diminue la quantité de code par l'intégration des Frameworks tiers (Struts, Hibernate dans le cadre de l'application CPS)
- Permet de mettre en oeuvre la programmation orientée aspect
- Un système transactionnel au niveau de la couche métier.
- Rend simple les tests des applications multicouches (n-tiers)
- Pas de dépendances entre le code et Spring grâce à la notion d'injection de dépendances. Ce qui permet de remplacer une couche sans impacter sur les autres
- Déploie et consomme des services Web
- Echange des objets par le protocole HTTP
- Facilite la maintenabilité de l'application
- Facilite la modularité et l'extensibilité de l'application

Ainsi, la séparation des interfaces de leur implémentation permet d'utiliser par exemple, un framework différent pour la couche Présentation ou Persistance sans impacter sur les autres couches.



## L'architecture de Spring

La figure suivante représente l'organisation des modules et des fonctionnalités utilisés dans Spring :

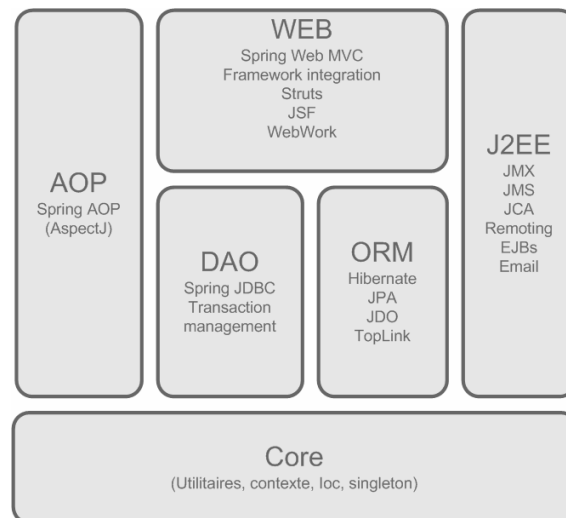


FIGURE 41 – Architecture du framework Spring

Comme le montre la figure ci-dessus, Spring repose sur un socle technique constitué de deux modules principaux :

- **Spring Core** : le conteneur léger. Ce module implémente le concept d'injection de dépendances (ou inversion de contrôle). Il est également responsable de la gestion de la configuration du conteneur.
- **Spring AOP** : ce module permet d'intégrer la programmation orientée aspect (POA).

Sur ce socle viennent se greffer d'autres modules de plus haut niveau destinés à intégrer des Frameworks tiers, ou à fournir des fonctions de support. Ces modules sont les suivants :

- Modules d'intégration de la persistance de données :
  - **Spring DAO** : permet de rendre abstrait la notion d'objets d'accès aux données.
  - **Spring ORM** : permet d'intégrer un framework de persistance.
  - **Spring JDBC** : permet de simplifier l'utilisation de JDBC.
- **Spring Context** : Module de gestion de contexte applicatif  
Permet d'assurer le dialogue entre Spring et l'application, indépendamment de la plateforme technique sur laquelle fonctionne cette dernière.

- **Spring Web** : Module d'intégration de Framework Web (Struts par exemple)
- **Spring Remoting** : Module de distribution d'objets.  
Permet de transformer de simples classes Java en objets distribués RMI, Web Services ou autres.
- Modules de support de la différente technologie JEE (JMX, JMS, JCA et EJB).

### 9.3 Hibernate

Hibernate est un framework open source gérant la persistance des objets en base de données relationnelle. Ce dernier remplace les accès à la base de données par des appels à des méthodes objet de haut niveau. De plus, il permet de tirer partie des concepts de la programmation objet, tels que, l'héritage, le polymorphisme, les relations entre les classes, etc.

La figure suivante décrit l'architecture du framework Hibernate :

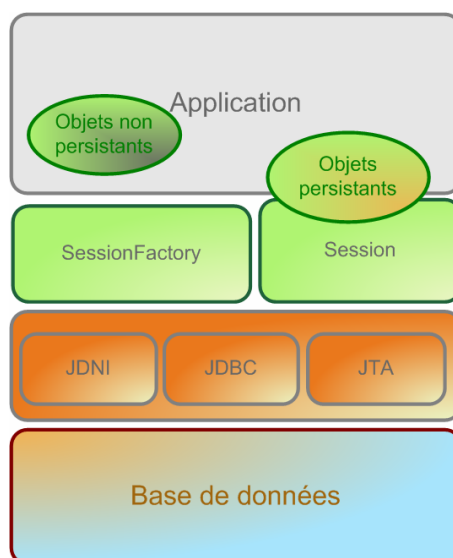


FIGURE 42 – Architecture du framework Hibernate

- **SessionFactory** : Permet de créer des objets sessions, puis réalise le mapping entre la session créée et la base de données relationnelle. Il est important de souligner que la session ne peut être mappée qu'avec une seule base de données.
- **Session** : C'est un objet à courte durée de vie représentant la conversation entre l'application et l'entrepôt de persistance. Ce module permet d'encapsuler une connexion JDBC et de créer des objets de type transaction.

- **Objets persistants** : Ce sont des objets à courte durée de vie, contenant l'état de persistance et la fonction métier. Ce sont des objets de type JavaBeans ou POJOs associés à une seule session. Ils deviennent libres dès la fermeture de la session. Hors connexion, ils peuvent être utilisés par n'importe laquelle des couches de l'application.
- **Objets non persistants** : Ce sont des objets non associés à une session (ils peuvent en réalité être des instances de classes persistantes ou métiers, mais en mode déconnecté).

Il faut savoir qu'Hibernate utilise :

- **L'API JDBC** : permet de gérer la connexion à la base de données relationnelle.
- **L'API JNDI** : permet de gérer la connexion aux sources de données
- **L'API JTA** : permet de gérer la connexion aux serveurs d'applications

## 9.4 Tiles

Ce framework est utilisé au niveau de la couche Présentation et permet notamment de compartimenter une page JSP.

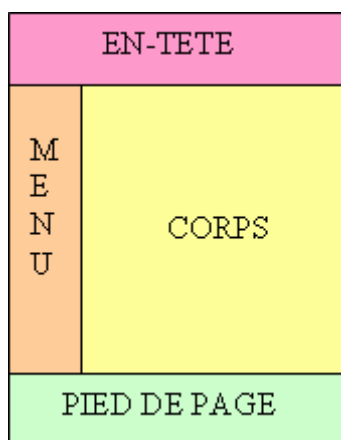


FIGURE 43 – Exemple du framework JEE Tiles

Ce framework est très pratique lorsqu'une application possède beaucoup de similarités entre ces différentes pages. En effet, si nous prenons l'exemple de la figure ci-dessus, l'en-tête, le menu, et le pied de page de chaque page de l'application sont identiques. Il suffit donc de les définir une seule et unique fois. Seul le corps de chacune de ces pages sera différent.

Une telle configuration est définie dans le fichier de configuration de Tiles nommé "*tiles-defs.xml*". Voici le code nécessaire :

```
<definition name="tiles.body_template"
  path="/jsp/body_template.jsp">
  <put name="header" type="page" value="/jsp/header.jsp"/>
  <put name="menu" type="page" value="/jsp/menu.jsp"/>
  <put name="body" />
  <put name="footer" type="page" value="/jsp/footer.jsp"/>
</definition>
```

Il faut, dans un premier temps, définir un modèle, appelé "*body-template.xml*". Notre page JSP est constituée de 4 parties :

- **L'en-tête** correspond à la page "*header.jsp*". Dans cette dernière, il est possible de retrouver toutes les importations nécessaires pour la page JSP, et particulièrement celles pour les fichiers Javascript
- **Le menu** correspond à la page "*menu.jsp*", que l'on retrouve dans l'ensemble des pages JSP
- **Le corps** de la page JSP (le body). Celui-ci est différent pour chacune des pages JSP. C'est pourquoi il n'est défini explicitement dans l'exemple ci-dessus.
- **Le pied de page** correspondant à la page "*footer.jsp*"

Mais il ne faut pas oublier que ce modèle reste tout de même une page JSP à part entière. Celle-ci est d'ailleurs précisée via le champ "*path*". Nous intégrons ensuite au sein de cette page les quatre parties décrites précédemment. Voici le code du fichier "*body-template.jsp*".

```
<html>
  <head>
    <tiles:insert attribute="header"/>
  </head>

  <body>
    <tiles:insert attribute="menu"/>
    <tiles:insert attribute="body"/>
    <tiles:insert attribute="footer"/>
  </body>
</html>
```

L'intégration d'une partie est effectuée grâce au tag "*tiles:insert attribute=header*". Champ que l'on retrouve dans le fichier xml de définition.

Une fois ce modèle mis en place, il suffit de déclarer la nouvelle page JSP de l'application dans le fichier de configuration "*tiles-defs.xml*". Voici le code nécessaire :

```
<definition name="tiles.ecrans" extends="tiles.body_template">
  <put name="body" type="page" value="/jsp/gestionDroits/ecrans.jsp" />
</definition>
```

Dans cet exemple, on définit la page `"ecrans.jsp"` qui étend du modèle `"body-template"` mis en place précédemment. Seul le `"body"` est déclaré ici, car c'est la seule partie de la page qui n'avait pas été définie dans le modèle. Je souligne également que le `"name=tiles.ecrans"` va être utilisé dans le fichier de configuration struts `"struts-config.xml"`, lors d'une redirection `"forward success"`. En héritant du modèle `"body-template"`, cette page possède automatiquement l'en-tête, le menu, et le pied de page associé.

## 10 Annexe 3 : Outil de transformation de données

L'outil d'extraction et de transformation de données (ETL en anglais) utilisé par Steria est Oracle Data Integrator.

### 10.1 ODI (Oracle Data Integrator)

La configuration de Oracle Data Integrator sera présentée par un exemple dans le cadre du projet Camaïeu. Ce projet utilise une base de donnée Oracle, le driver de connexion utilisée est JDBC.

#### 10.1.1 Créations des utilisateurs sur la base Oracle

Il est nécessaire de créer 2 utilisateurs pour que ODI puisse créer les tables dont il a besoin. Les commandes suivantes sont à exécuter, elles peuvent évidemment être différentes selon le SGBD que vous allez utiliser.

```
create tablespace snpm_tablespace datafile 'snpm.dbf' size 50M autoextend on maxsize 200M;
create temporary tablespace temp_snpm_tablespace tempfile 'temp_snpm.dbf' size 50M autoextend on maxsize 200M;
create tablespace snpw_tablespace datafile 'snpw.dbf' size 50M autoextend on maxsize 200M;
create temporary tablespace temp_snpw_tablespace tempfile 'temp_snpw.dbf' size 50M autoextend on maxsize 200M;
create user snpm identified by snpm default tablespace snpm_tablespace temporary tablespace temp_snpm_tablespace;
create user snpw identified by snpw default tablespace snpw_tablespace temporary tablespace temp_snpw_tablespace;
grant connect, resource to snpm;
grant connect, resource to snpw;
```

FIGURE 44 – Commandes SQL à exécuter

On crée 2 utilisateurs , l'un correspond au Master Repository et l'autre au Work Repository. Les tailles utilisés pour les tablespaces sont celles conseillées sur le document de référence fournit par Oracle.

### 10.1.2 Création du Master Repository

lancer "Master Repository Creation" dans le menu démarrer.

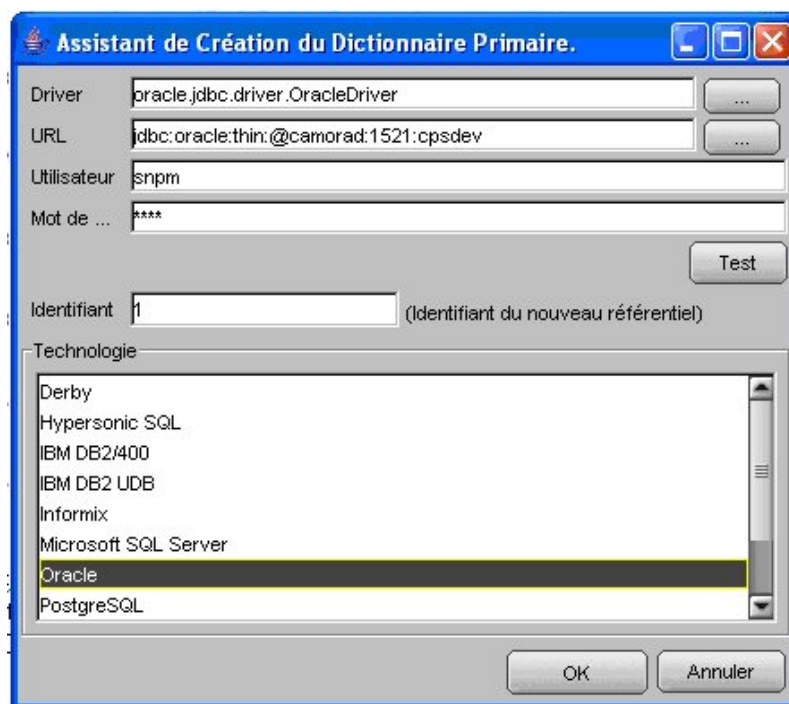
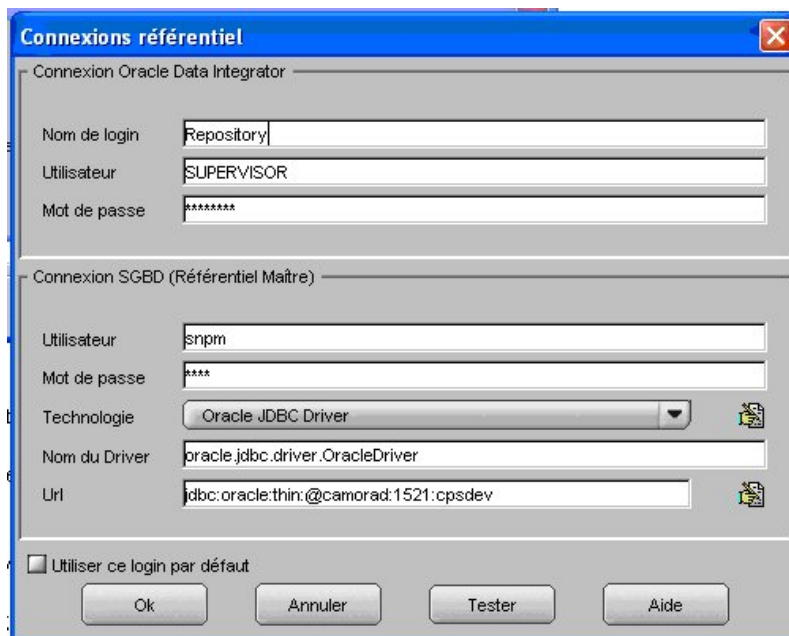


FIGURE 45 – Compléter la fenêtre tel qu'indiqué

Il faut utiliser en nom d'utilisateur celui créé pour le Master Repository (user : snpm, mdp : snpm) L'identifiant doit avoir un nombre de caractères supérieur ou égal à 1. La configuration de votre URL et de votre driver dépend de votre SGBD. Cliquez ensuite sur "ok", un batch se lance et crée les fichiers nécessaires (fichiers xml et tables dans votre base de donnée).

### 10.1.3 Connexion au référentiel

lancer "Topology Manager" dans le menu démarrer. Créer une nouvelle connexion en cliquant sur "nouveau".



The screenshot shows a Windows-style dialog box titled "Connexions référentiel". It is divided into two main sections. The top section, "Connexion Oracle Data Integrator", contains three input fields: "Nom de login" with the value "Repository", "Utilisateur" with the value "SUPERVISOR", and "Mot de passe" with masked characters "\*\*\*\*\*". The bottom section, "Connexion SGBD (Référentiel Maître)", contains five input fields: "Utilisateur" with "shpm", "Mot de passe" with "\*\*\*\*", "Technologie" with a dropdown menu showing "Oracle JDBC Driver", "Nom du Driver" with "oracle.jdbc.driver.OracleDriver", and "Url" with "jdbc:oracle:thin:@camorad:1521:cpsdev". At the bottom of the dialog, there is a checkbox labeled "Utiliser ce login par défaut" which is currently unchecked, and four buttons: "Ok", "Annuler", "Tester", and "Aide".

FIGURE 46 – Compléter la fenêtre tel qu'indiqué

Le mot de passe par défaut pour le SUPERVISOR est SUNOPSIS, les paramètres de connexion au SGBD sont les mêmes que dans la fenêtre précédente. Cliquez sur "ok" puis à nouveau sur "ok" dans la fenêtre de connexion.



### 10.1.4 Associer un Référentiel de travail au Master Repository

Cliquer sur l'onglet ou il est marqué AB en bas à gauche de la fenêtre. Puis cliquer avec le bouton droit sur "Référentiel de travail" et faites "Insérer référentiel de travail"

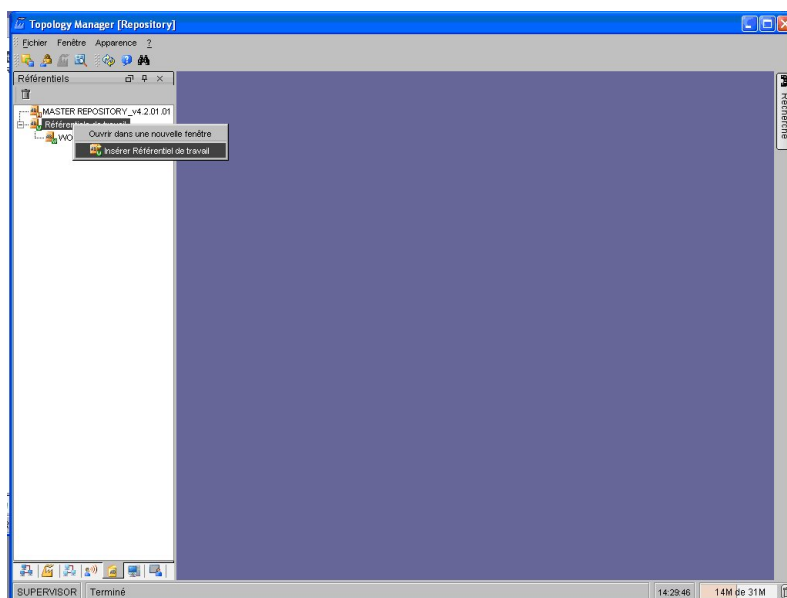


FIGURE 47 – Insertion d'un référentiel de travail

Cliquer sur l'onglet ou il est marqué AB en bas à gauche de la fenêtre. Puis cliquer avec le bouton droit sur "Référentiel de travail" et faites "Insérer référentiel de travail".

The screenshot shows a Java Swing window titled "Serveur de données : Work repository". It has five tabs: "Définition", "JDBC", "Version", "Privileges", and "FlexFields". The "Définition" tab is active. The form contains the following fields and controls:

- Nom:** A text field containing "Work repository".
- Technologie:** A dropdown menu with "Oracle" selected.
- Instance / dblink (Serveur de données):** A text field containing "cpsdev".
- Connexion:** A group box containing:
  - Utilisateur:** A text field containing "snpw".
  - Mot de passe:** A password field containing "\*\*\*\*".
  - JNDI connexion:** An unchecked checkbox.
- Taille de l'array fetch:** A text field containing "30".
- Taille de batch update:** A text field containing "30".

At the bottom, there are five buttons: "Ok", "Annuler", "Appliquer", "Aide", and "Tester".

FIGURE 48 – Définition du référentiel de travail

La fenêtre suivante apparaît. Choisir snpw (le 2ème compte utilisateur que nous avons crée sur la base de donnée) et saisir le mot de passe associé (dans l'exemple snpw). Cliquez également sur l'onglet JDBC et configurez comme précédemment (voir l'écran précédent). Cliquez ensuite sur "ok".

The image shows a software window titled "Référentiel travail :WORKREP1". It has three tabs: "Définition", "Version", and "Privileges". The "Définition" tab is active. Inside the tab, there are several input fields and buttons. The "ID" field contains the number "1". The "Type" dropdown menu is set to "Répertoire". The "Nom" field is filled with the text "WORKREP1". Below it, the "Mot de passe" field is empty, and there is a button labeled "Changer le mot de passe". The "Identifiant Externe" field contains the number "1207129897370", and there is a button labeled "Connexion". At the bottom of the window, there are four buttons: "Ok", "Annuler", "Appliquer", and "Aide".

FIGURE 49 – Remplir le nom

Remplissez juste le "nom" puis cliquez sur "ok".

#### 10.1.5 Utilisation du designer

Cliquez sur "Nouveau" pour créer une nouvelle connexion.

**Connexions référentiel**

Connexion Oracle Data Integrator

Nom de login: Repository

Utilisateur: SUPERVISOR

Mot de passe: \*\*\*\*\*

Connexion SGBD (Référentiel Maître)

Utilisateur: snpm

Mot de passe: \*\*\*\*

Technologie: Oracle JDBC Driver

Nom du Driver: oracle.jdbc.driver.OracleDriver

Url: jdbc:oracle:thin:@camorad:1521:cpsdev

Référentiel de travail

Nom du référentiel: WORKREP1

☒ Utiliser ce login par défaut

Ok Annuler Tester Aide

FIGURE 50 – Paramétrage de la connexion

Le remplissage du "Connexion Oracle Data Integrator" et de "Connexion SGBD (Référentiel Maître)" se font de la même façon qu'auparavant. Bien noter que l'utilisateur est snpm (celui du Master Repository). La seule chose qui change dans cette fenêtre c'est l'apparition d'une zone de saisie pour le référentiel de travail. Y ajouter le référentiel de travail défini à l'étape précédente "WORKREP1". La configuration est terminée vous pouvez maintenant saisir les extractions et transformations que vous souhaitez réaliser entre vos bases.

## 11 Annexe 4 : Sommaire de la proposition commerciale de Steria

- Compréhension et perception des besoins
- La Réponse STERIA
- Solution proposée = Solution industrielle
- Organisation proposée : une équipe intégrée
- Gouvernance
- Conseil et Expertises
- Phase d'Analyse
- Phase de Réalisation
- Outils mis en œuvre
- Méthode d'évaluation des charges
- Transfert
- Tierce Maintenance Applicative : réponses aux questions
- Proposition financière
- Compétences
- Annexes : nos références

## 12 Annexe 5 : sommaire du DAT

### 1 GLOSSAIRE TECHNIQUE

### 2 PREAMBULE

#### 2.1 OBJET DU DOCUMENT

- 2.1.1 *Contexte et enjeux*
- 2.1.2 *Périmètre fonctionnel*

#### 2.2 LES ACTEURS DU DOCUMENT

#### 2.3 CYCLE DE VIE DU DOCUMENT

### 3 ELEMENTS STRUCTURANTS POUR L'ARCHITECTURE TECHNIQUE

#### 3.1 PRINCIPES DIRECTEURS D'ARCHITECTURE GENERALE

#### 3.2 ARCHITECTURE APPLICATIVE

- 3.2.1 *Présentation succincte des composants logiciels du socle*
- 3.2.2 *Communication entre les couches*
- 3.2.3 *Format de livraison*

#### 3.3 POINTS SPECIFIQUES

- 3.3.1 *Choix du serveur d'application*
- 3.3.2 *Choix du langage de programmation des batchs*
- 3.3.3 *Souhait d'homogénéisation du SI Camaieu*

### 4 ARCHITECTURE TECHNIQUE

#### 4.1 ROLES ET TECHNOLOGIES DES ENVIRONNEMENTS UTILISES

- 4.1.1 *Intégration*
- 4.1.2 *Qualification*
- 4.1.3 *Pré production*
- 4.1.4 *Production*

#### 4.2 SCHEMA PHYSIQUE DE LA PLATE FORME CIBLE

- 4.2.1 *Composants techniques du serveur d'application*
- 4.2.2 *Composants techniques du serveur de base de données*
- 4.2.3 *Plate forme de qualification*
- 4.2.4 *Plate forme de pré-production*
- 4.2.5 *Plate forme de production*

#### 4.3 GESTION DES FLUX

- 4.3.1 *Tableau des flux*
- 4.3.2 *Explications des technologies*

#### 4.4 SOCLES CLIENTS

- 4.4.1 *Navigateur et résolution*
- 4.4.2 *Connexion réseau*

## 13 Annexe 6 : sommaire du DAL

### **1 PREAMBULE**

### **2 INTRODUCTION**

- 2.1 APPLICATION CONCERNEE ET PERIMETRE
- 2.2 LEGENDE DES DIAGRAMMES

### **3 CADRE GENERAL D'ARCHITECTURE LOGICIELLE**

- 3.1 PRINCIPES TECHNIQUES DIRECTEURS
- 3.2 CONTRAINTES TECHNIQUES
- 3.3 REFERENCES DES NORMES ET PRECONISATIONS APPLICABLES
- 3.4 OBJECTIFS QUALITE
- 3.5 CARACTERISATION DE L'ARCHITECTURE LOGICIELLE
- 3.6 MODELES DE CONCEPTION ET PROGRAMMATIQUES

### **4 DECOMPOSITION DE L'ARCHITECTURE LOGICIELLE**

- 4.1 SCHEMA GENERAL
- 4.2 SERVICES COMMUNS
- 4.3 COUCHE « OBJETS METIER »
- 4.4 COUCHE SERVICES METIER
- 4.5 PERSISTANCE
- 4.6 COUCHE DE PRESENTATION
- 4.7 INTEGRATION AVEC SPRING

### **5 CONSTRUCTION DE L'ARCHITECTURE LOGICIELLE**

- 5.1 PROCESSUS DE DEVELOPPEMENT
- 5.2 LES LANGAGES ET OUTILS DE MODELISATION UTILISES
- 5.3 LES LANGAGES ET OUTILS DE DEVELOPPEMENTS UTILISES
- 5.4 UTILISATION DE FRAMEWORKS ET BIBLIOTHEQUES
- 5.5 INTERACTION AVEC L'UTILISATEUR
- 5.6 TRAITEMENTS CALCULATOIRES
- 5.7 DEPLOIEMENT

## 14 Annexe 7 : sommaire du PPL

### **1 INTRODUCTION**

- 1.1 OBJET DU DOCUMENT
- 1.2 RESPONSABILITES
- 1.3 DOCUMENTS APPLICABLES
- 1.4 DOCUMENTS DE REFERENCE
- 1.5 TERMINOLOGIE

### **2 ENVIRONNEMENT DE DEVELOPPEMENT**

- 2.1 LES OUTILS DE DEVELOPPEMENT
- 2.2 LES OUTILS DE MISE AU POINT
- 2.3 INSTALLATION D'UN POSTE DE DEVELOPPEMENT
- 2.4 INSTALLATION DE BORLAND TOGETHER

### **3 NORMES DE DEVELOPPEMENT**

- 3.1 STRATEGIE DE CONTROLE

### **4 REGLES DE PRODUCTION**

- 4.1 CONVENTION DE NOMMAGE
- 4.2 ORGANISATIONS
- 4.3 GESTION DES DIFFÉRENTS MAPPING SPRING

### **5 TESTS**

- 5.1 TESTS UNITAIRES

### **6 GESTION DE CONFIGURATION**

- 6.1 PRESENTATION
- 6.2 COMPOSITION DE LA CONFIGURATION

### **7 CONCEPTION**

- 7.1 CONCEPTION GENERALE
- 7.2 CONCEPTION DETAILLEE

### **8 ANNEXE**

- 8.1 GESTION DU REPOSITORY MAVEN ET ARCHIVA



## Table des figures

1	Les implantations de STERIA en Europe . . . . .	10
2	Chiffre d'affaires et marge opérationnelle du groupe . . . . .	11
3	Répartition du chiffre d'affaires . . . . .	12
4	L'organisation d'un centre de services . . . . .	14
5	Localisation française et européenne des centres de services . . . . .	15
6	Le portail de Steria . . . . .	16
7	Organisation du centre AM . . . . .	19
8	Organisation de la chaîne CAMAÏEU . . . . .	22
9	Moyens pour minimiser les risques . . . . .	25
10	Le cycle en V : plus on descend et plus on s'intéresse aux détails de l'application. . . . .	25
11	Cycle en V du projet Camaïeu . . . . .	26
12	Dénombrement des unités d'oeuvres . . . . .	27
13	Les coefficients de calcul de charge en jours . . . . .	29
14	Les coefficients de calcul des coûts . . . . .	29
15	Planning général du projet Camaïeu . . . . .	30
16	Planning détaillé du projet Camaïeu . . . . .	30
17	Diagramme de séquence de la phase d'avant vente . . . . .	31
18	Schéma de la phase d'études . . . . .	33
19	Schéma de la phase de Recette . . . . .	35
20	L'architecture matérielle de l'entreprise Steria . . . . .	37
21	L'architecture matérielle de Camaïeu . . . . .	39
22	Les couches logicielles de l'application CPS-V2 . . . . .	40
23	Planning réalisé 1/4 . . . . .	44
24	Planning réalisé 2/4 . . . . .	45
25	Planning réalisé 3/4 . . . . .	46
26	Planning réalisé 4/4 . . . . .	47
27	L'écran liens catégorie taille / magasins au démarrage . . . . .	51
28	La popup d'ajout d'un nouveau lien . . . . .	52
29	Réaffichage de l'arborescence après ajout . . . . .	53
30	Réaffichage de la page après enregistrement . . . . .	54
31	Résultat après saisie erronée . . . . .	55
32	Résultat de l'affichage de la page . . . . .	56
33	Résultat après une saisie erronée dans la popup d'ajout . . . . .	58
34	Ouverture d'une popup en ayant sélectionné des éléments dans l'arbre . . . . .	59
35	Popup pour associer un type produit . . . . .	60
36	Contenu de la page après validation de la popup Type produit . . . . .	61
37	Suppression de niveaux de paramétrages . . . . .	62
38	Les technologies de la PIC . . . . .	67
39	Schéma de l'outil Maestro . . . . .	68
40	Schéma du framework Struts . . . . .	69
41	Architecture du framework Spring . . . . .	73

42	Architecture du framework Hibernate . . . . .	74
43	Exemple du framework JEE Tiles . . . . .	75
44	Commandes SQL à exécuter . . . . .	78
45	Compléter la fenêtre tel qu'indiqué . . . . .	79
46	Compléter la fenêtre tel qu'indiqué . . . . .	80
47	Insertion d'un référentiel de travail . . . . .	81
48	Définition du référentiel de travail . . . . .	82
49	Remplir le nom . . . . .	83
50	Paramétrage de la connexion . . . . .	84