



IUT de Lannion
Département informatique
Rue Edouard Branly
22300 Lannion

RAPPORT DE STAGE

“Développement d'applications d'administration pour les comptes extranet de mutuelles”

Stage du 7 avril au 13 juin 2014

Tuteur professionnel : Monsieur Pierre-Yves Motreff

Tuteur universitaire : Madame Elisabeth Genaivre

Réalisé au Cimut



Corentin Quichaud

Année universitaire 2013-2014



IUT de Lannion
Département informatique
Rue Edouard Branly
22300 Lannion

RAPPORT DE STAGE

“Développement d'applications d'administration pour les comptes extranet de mutuelles”

Stage du 7 avril au 13 juin 2014

Tuteur professionnel : Monsieur Pierre-Yves Motreff

Tuteur universitaire : Madame Elisabeth Genaivre

Réalisé au Cimut



Corentin Quichaud

Année universitaire 2013-2014

Remerciements

Je tiens à remercier dans un premier temps tous les membres de l'équipe NTIC du Cimut pour leur excellent accueil et leur sympathie durant toute la durée du stage.

J'adresse particulièrement mes remerciements à Pierre-Yves Motreff et Guillaume Yclon pour m'avoir aidé et suivi lors du développement des applications. Leurs explications m'ont permis d'approfondir mes connaissances. J'ai pu apprécier leur disponibilité tout au long de mon stage.

Je tiens aussi à remercier Yoann Laot et Lionel Le Foll pour leur implication et leur soutien lors de mon stage, ainsi que Doriane Bon, pour son aide à la réalisation de ce rapport.

Enfin, j'adresse mes remerciements à l'IUT de Lannion, le département informatique et l'ensemble des professeurs, pour ces deux années d'enseignement.

Sommaire

Introduction

1. La structure d'accueil : le Cimut.....	07
1.1 Histoire du Cimut.....	07
1.2 Présentation de la structure d'accueil.....	08
1.3 Situation géographique et juridique.....	08
1.4 Finalité et ses objectifs.....	08
1.5 Positionnement.....	09
1.6 Insertion dans le tissu national.....	09
1.7 Ressources du Cimut.....	11
1.8 Organisation hiérarchique du Cimut.....	12
1.9 Starweb.....	16
2. La mission.....	17
2.1 Contexte.....	17
2.2 Objectif.....	18
2.3 Contraintes.....	19
2.4 Technologies utilisées.....	20
2.4.1 Environnement de travail.....	20
2.4.2 Langages utilisés.....	21
2.4.3 Librairies et plugins utilisés.....	23
2.4.4 Base de donnée et logs.....	24
2.4.5 Autres.....	25
2.5 L'existant : l'environnement Starweb.....	27
2.6 Gantt prévisionnel.....	27
3. Développement de l'application web.....	28
3.1 Le Web Service.....	28
3.1.1 WSDL.....	29
3.1.2 Développement du Web Service.....	30
3.1.3 Connexion LDAP.....	31

3.1.4 Recherche LDAP.....	31
3.1.5 Modification LDAP.....	33
3.1.6 Mise en ligne du Web Service.....	33
3.2 Environnement StarWeb: portail et mesICIP.....	33
3.3 L'application d'administration.....	36
3.3.1 Fonctionnement de l'application.....	36
3.3.2 La recherche.....	36
3.3.3 La servlet.....	38
3.3.4 L'affichage liste.....	41
3.3.5 Affichage des détails.....	43
3.3.6 Modification du compte.....	44
3.3.7 Réinitialisation de mot de passe.....	45
3.4 L'application d'historisation.....	45
3.4.1 Base de donnée et Log4j.....	45
3.4.2 Le Web Service.....	47
3.4.3 Création.....	47
3.4.4 la servlet HEXT.....	48
3.4.5 Affichage de la liste.....	49
3.5 Implémentation des applications dans l'architecture.....	50
3.6 Mise en ligne des applications.....	51
3.7 Test.....	51
3.8 Réclamations des utilisateurs.....	52
3.9 Gantt réel.....	52
Conclusion.....	53
Résumé.....	55
Abstract.....	57
Bibliographie.....	58
Liste des termes Techniques.....	59
Table des annexes.....	61

Introduction

Ce stage a été réalisé dans le cadre de ma formation pour l'obtention du DUT Informatique. Il s'est effectué entre le 7 avril et le 13 juin 2014 au CIMUT (centre informatique des mutuelles) à Quimper. J'ai rejoint l'équipe IHM (Interaction Homme Machine) sous la tutelle de Pierre-Yves MOTREFF, responsable de l'équipe. Pierre-Yves MOTREFF et Guillaume YCLON ont été les deux membres de l'équipe qui m'ont plus particulièrement encadré et suivi.

Mon choix s'est dirigé vers le service IHM du CIMUT car l'IHM est une approche que je trouve très intéressante, que ce soit d'un point de vue réalisation ou utilisation. De plus le JAVA est un langage de programmation qui m'a toujours séduit. Par ailleurs, les mutuelles font partie de notre quotidien, et travailler au sein de leur centre informatique constitue une expérience constructive et concrète.

J'ai rejoint l'équipe IHM du CIMUT, avec pour mission d'élaborer deux applications web ayant pour but l'administration de comptes extranet* d'adhérents et partenaires santé de mutuelles.

Ces applications seront mises à disposition des mutuelles afin de leur permettre d'administrer en toute autonomie leurs comptes extranet sans avoir recours aux services du Cimut.

1. La structure d'accueil : le Cimut

1.1 Histoire du Cimut

En 1975, 3 organismes bretons (Quimper, Vannes et Saint Brieuc) se sont retrouvés sans solution informatique, à la suite de la dissolution de l'ACMUR, à Nantes, qui gérait à la fois l'informatique des CMR et des Organismes Conventionnés.

Ces 3 organismes ont alors décidé de mettre leurs ressources en commun dans le but de développer leur propre solution informatique de gestion du régime obligatoire et de la complémentaire santé. Des dizaines de mutuelles vont dès lors utiliser ses services.

A partir de 2006, le CIMUT adopte une stratégie de partenariat, d'ouverture et de développement. Il se dote d'une nouvelle direction générale. Sous l'impulsion de Gilles Marty, nouveau Directeur Général, le Cimut élargit son champ d'action. Les fruits de sa forte croissance sont alors systématiquement réinvestis dans la maîtrise des coûts, l'infrastructure et l'enrichissement des services proposés.

En 2011, pour accompagner son développement et ses activités de formation (centre agréé depuis 2001), le Cimut ouvre une agence à Paris.

En 2012, le nouveau bâtiment du Cimut est inauguré. Avant-gardiste, écologique, hautement sécurisé, il est le vecteur d'une solution Starweb de pointe, s'appuyant sur des équipes opérationnelles efficaces.

En près de 40 ans, le CIMUT est devenu un pôle d'excellence informatique et technologique employant à Quimper plus de 100 ingénieurs et techniciens.

Environ 20 % relève de prestataires. 24 adhérents (mutuelles et organismes conventionnés), tels que Harmonie Mutuelle, Radiance ou en encore la Smeba, ont ainsi confié leur informatique à cette structure.

1.2 Présentation de la structure d'accueil

Le Cimut relève du secteur tertiaire, c'est un centre informatique pour les mutuelles. Ces services sont exonérés de TVA et refacturés à l'euro l'euro. C'est une organisation à but non lucratif, elle ne peut générer aucun bénéfice, donc ne dispose d'aucun capital.

Les clients payent les services à prix coûtant. L'origine des ressources du Cimut sont les prestations de service informatique facturées aux adhérents.

Le Cimut propose une prestation d'infogérance* qui regroupe l'ensemble des ressources et moyens permettant de développer et exploiter le système d'information commun Starweb. Ce système doit permettre aux mutuelles d'assurer la gestion des prestations santé auprès de leurs propres adhérents du régime social des indépendants (RSI), du régime complémentaire, ou encore du régime social étudiant.

Entre 2007 et 2014, le CIMUT a doublé le nombre de personnes protégées gérées par sa solution Starweb en passant en sept ans de un à deux millions de personnes protégées .

1.3 Situation géographique et juridique

Depuis 2 ans, le CIMUT s'est installé dans son nouveau bâtiment qui représente un coût d'investissement de 5 millions d'euros. Il est situé au 9 rue Félix Le Dantec à Quimper et peut accueillir jusqu'à 120 collaborateurs.

Depuis janvier 1998, le Centre a pris son indépendance juridique sous la forme d'une Union Technique Mutualiste (UTM) relevant du Livre I du Code de la mutualité.

1.4 Finalité et ses objectifs

La finalité du Cimut est de garantir l'excellence des solutions grâce à l'expertise métier, mettre au service des assurés et des adhérents une technologie de pointe et participer à la pérennité et à la réalisation des engagements mutualistes.

L'objectif majeur du CIMUT est de rester à la pointe de la technologie, au fait des mutations du monde de la santé pour offrir à ses adhérents les meilleures solutions informatiques et de proposer leur solution informatique au plus grand nombre de mutuelles pour développer l'activité du Cimut (prospérer). Le Cimut projette ainsi de progresser encore sur le marché de la complémentaire santé avec pour objectif d'atteindre des revenus proches de 13 millions d'euros d'ici 2016 avec la gestion de 2,5 millions de personnes protégées.

1.5 Positionnement

Le Cimut, dont le budget annuel approche 12 millions d'euros, figure dans le Top 10 des éditeurs de solutions de gestion en santé en France. Il est l'unique acteur mutualiste face à des sociétés nationales ou internationales concurrentes telles que Cegedim Activ, CSC, Mutua, CIM qui proposent des solutions aux mutuelles relevant du Code des Assurances. (Le Cimut ne relève pas du code des assurances).

Il occupe aujourd'hui la place de leader mutualiste dans le domaine de la gestion des artisans commerçants et professions libérales (RO/RSI) (plus de 50 % de part de marché). Le CIMUT est également une référence en matière de solution informatique RO/RC dédiée aux Sections Locales Mutualistes.

1.6 Insertion dans le tissu national

Les 24 organismes adhérents au Cimut, sont présents à différents endroits de la France, ils représentent un portefeuille de près de 2 millions de personnes protégées.

Des chiffres représentatifs :

Année	2016	2013	2011	2009
Nombre de personnes protégées gérées	2 300 000	1 810 000	1 690 000	990 000
Nombre d'adhérents (1)	20	14	22	30
Effectif au 31 décembre	125	109	116	70
Salariés	95	85	80	58
Prestataires	30	24	36	12

(1) 2009 - 20013: fusion des adhérents du CIMUT. (Source : document du Cimut)

Voici la carte avec la situation géographique des différents adhérents au Cimut :

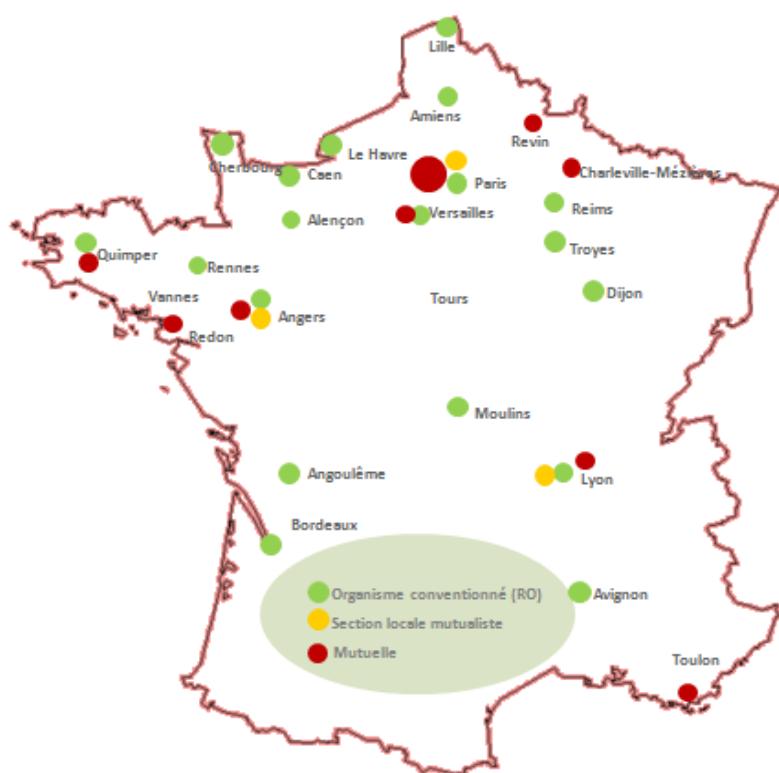


Figure 1 : Carte de répartitions des mutuelles (Source: document du Cimut)

1.7 Ressources du Cimut

Les ressources matérielles du Cimut sont dans un premier temps sa salle machine (Voir *figure 2*) contenant de nombreux serveurs que cela soit pour la mise en production de service ou le stockage de données, etc... cet investissement matériel et logiciel représente environ 4 millions d'euros.

La maintenance de cette architecture représente un budget annuel d'environ 500 000 €.

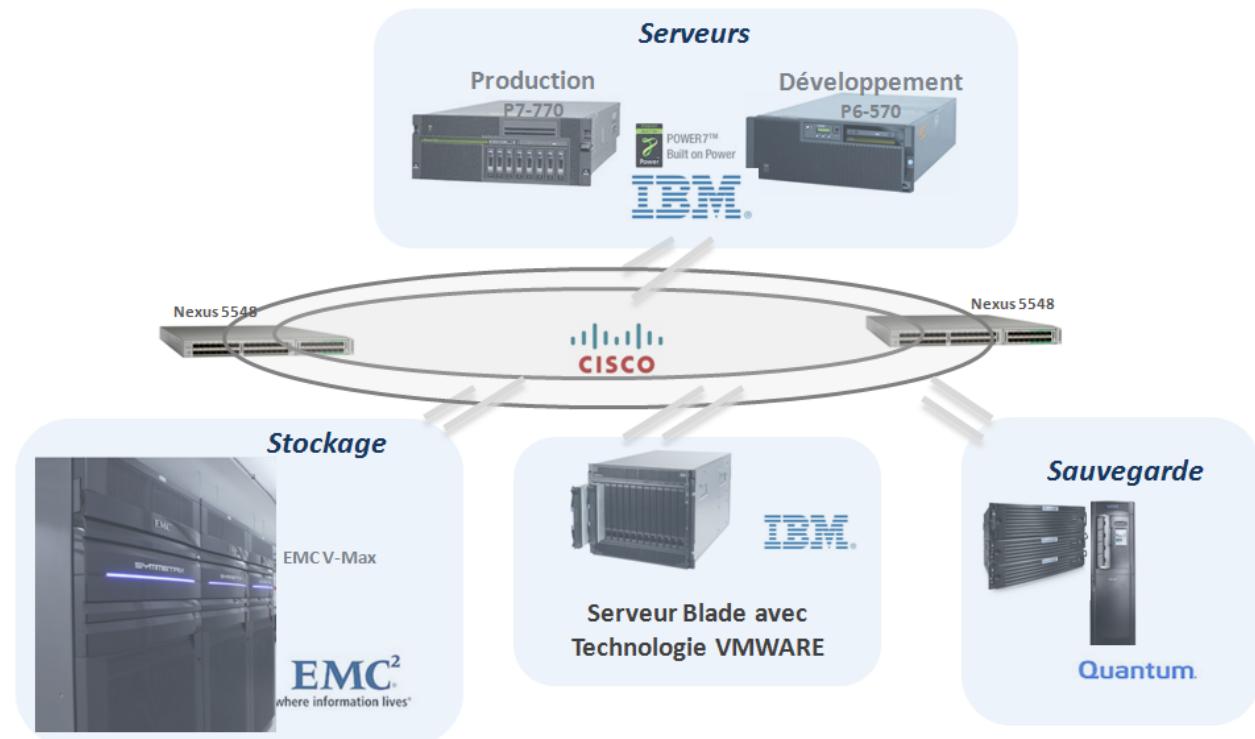


Figure 2 : Architecture technique du Cimut (Source: document du Cimut)

Ses ressources immatérielles sont constituées de sa solution Starweb évaluée à 5 millions d'euros.

Au niveau financier, le Cimut fonctionne par budget annuel présenté et voté en Conseil d'Administration. Il s'élève en 2014 à environ 12 millions d'euro. Exclu des bénéfices, le Cimut ne dispose pas de réserves financières.

Enfin ses ressources humaines affichent 86 salariés et 19 prestataires.

1.8 Organisation hiérarchique du Cimut

Le Cimut dispose d'un Directeur général et d'un comité de direction. Ce comité est en charge des 5 pôles suivants:

- Direction administrative et financière (Comptabilité, Contrôle de gestion)**

Le service Direction Administrative et Financière établit les budgets annuels, assure la comptabilité et le contrôle de gestion de l'union.

- Direction études (Administratif, Prestations, Cotisations, Architecture et urbanisation):**

La direction études informatiques est chargée d'élaborer les études préalables ou de faisabilité, de participer à la conception des schémas directeurs, d'assister et conseiller les utilisateurs dans leurs choix, d'identifier les besoins du marché et d'inspirer la conception et la réalisation du métier des mutuelles.

- Direction technique (Réseau, Système et Sécurité, Production et Exploitation, Ingénierie logicielle et Intégration) :**

La direction technique encadre les départements production, systèmes/réseaux et ingénierie de développement. La synergie entre ces trois entités garantit le bon fonctionnement du système d'information. Dans un environnement de haute technologie, ces pôles unissent leurs compétences techniques afin de garantir aux mutuelles adhérentes une qualité de service conforme à leurs attentes. La direction technique met en œuvre les grands projets d'évolution du système informatique et est responsable des appels d'offre et des contrats avec les fournisseurs d'équipements. Dans le cadre de son activité d'infogérance, la continuité de service est assurée par la mise en œuvre d'un PRA (plan de reprise d'activité) contractualisé avec un acteur reconnu de la sécurité et régulièrement testé en conditions réelles.

- Direction des relations extérieures (Hotline, Assistance à Maîtrise d'ouvrage, Contrôle Interne et Méthodes, Communication/développement):**

La direction des relations extérieures a en charge :

- l'activité commerciale, la mise en œuvre du plan de communication externe. Elle poursuit deux objectifs : augmenter la notoriété du Cimut et valoriser sa solution informatique afin de pérenniser et développer son activité.
- Le service Assistance à Maîtrise d'Ouvrage occupe un rôle clé dans son organisation. Il est l'interface entre le Cimut et les mutuelles adhérentes et joue un rôle moteur auprès des services internes par l'apport d'une solide expertise métier.
- Le service hotline, ouvert de 7h30 à 18h00 du lundi au vendredi, est chargé de recueillir toutes les demandes en provenance des mutuelles adhérentes : évolutions, modifications, corrections, informations réglementaires, assistance…

• **Direction des applications transversales (Nouvelles technologies (IHM, éditique, extranet), Décisionnel, Gestion relation adhérent/GED, Base de données, Mouvements d'organismes):**

La direction des applications transversales prend en charge tous les projets qui présentent une composante technique forte (bases de données, NTIC) ou requièrent des compétences spécifiques car basés sur des progiciels (décisionnel, GRC, GED, éditique, comptabilité, …).

Les missions qui lui sont confiées sont les suivantes :

Bases de données et NTIC

- mise au point et maintenance évolutive des processus de fabrication industrielle et des outillages génériques
- conception et administration des bases de données
- production des composants IHM et BD, en fonction des demandes formalisées par le service étude
- conception, développement et maintenance du portail applicatif Starweb, de la gestion des utilisateurs et de leurs droits, des plateformes extranet

- veille technologique

Progiciels (décisionnel, GRC, GED, éditique, comptabilité)

- paramétrage, développement, adaptation des produits aux métiers des mutuelles
- prise en compte des évolutions fonctionnelles et techniques
- conception et développement des interfaces entre les suites progicielles et les applications métiers.

Voici l'organigramme de l'UTM :

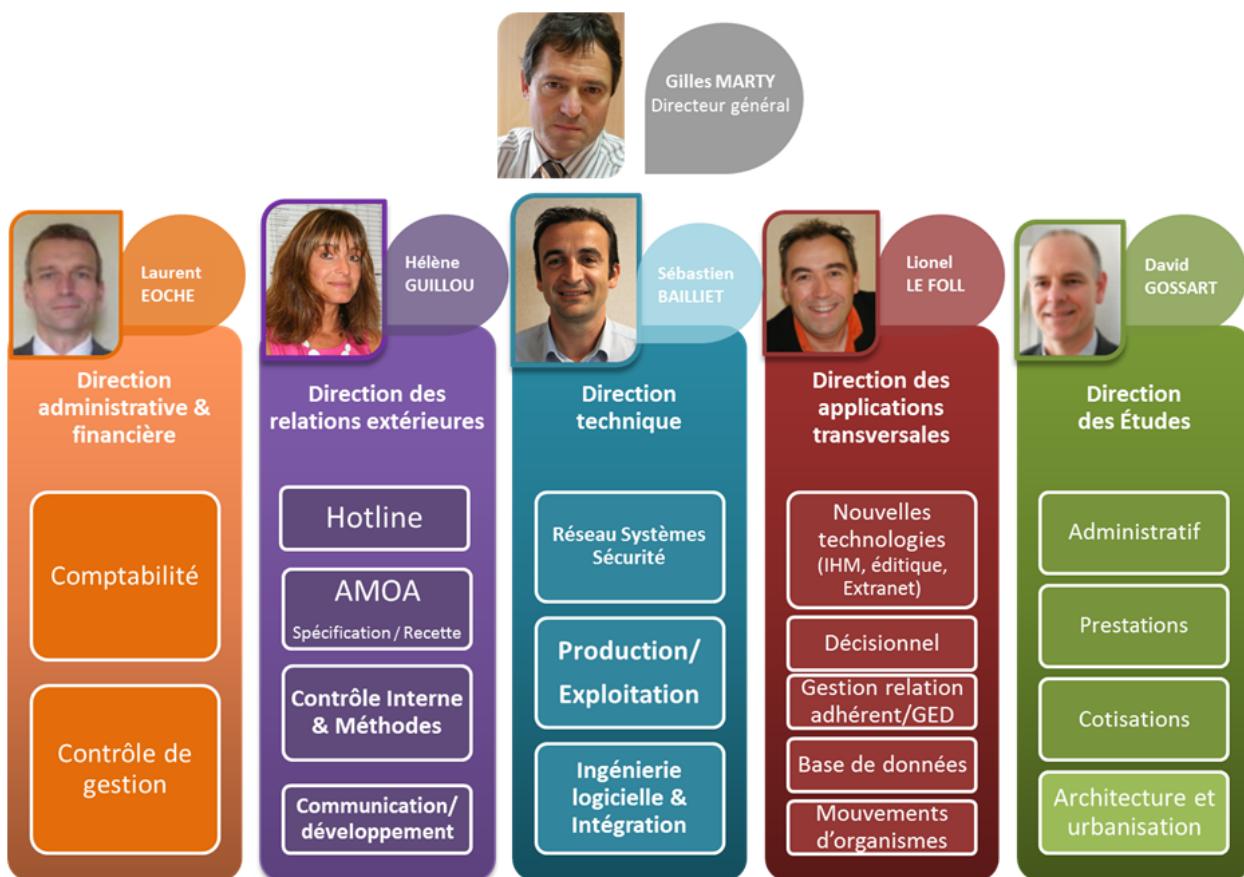


Figure 3 : Organigramme du Cimut (Source: document du Cimut)

Quant à moi, j'ai intégré le service IHM, qui fait partie de la direction des applications transversales, dirigé par Lionel LE FOLL (*Figure 4*).

Je travaille au quotidien dans un bureau composé de Pierre-Yves Motreff, responsable du service et tuteur de mon stage, Thomas Boenisch, analyste et Guillaume Yclon, prestataire, dans le but de créer de nouvelles IHM.

Les clients, (on parle ici plutôt d'adhérents au CIMUT), je n'ai aucun contact avec eux.

Les utilisateurs sont les gestionnaires de ces mutuelles.

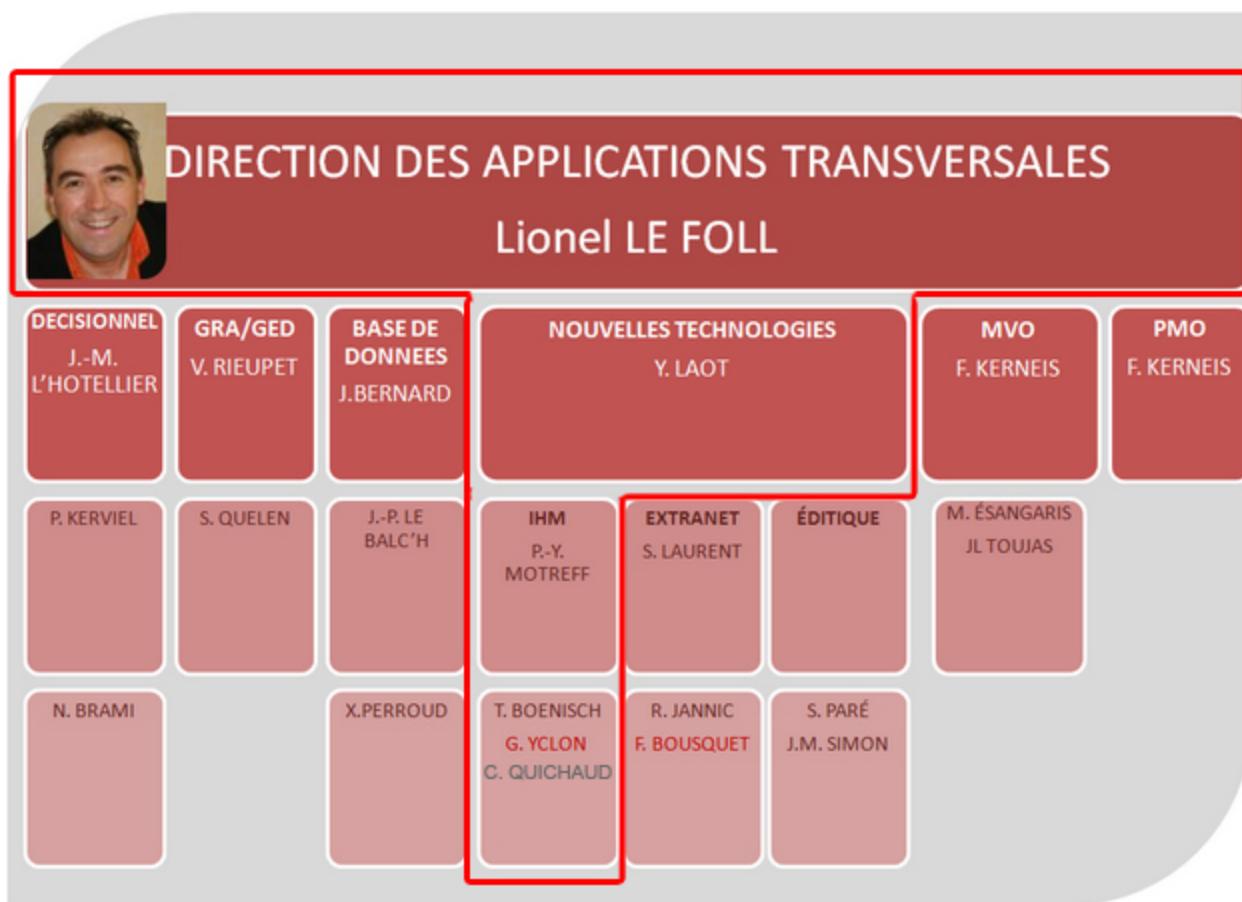


Figure 4 : Organigramme du service (Source: document du Cimut)

1.9 Starweb

Starweb, est une solution informatique globale de gestion des prestations santé. Il s'agit d'une gamme complète de solutions (le Décisionnel, la Gestion électronique de documents, la Comptabilité, la Gestion de la Relation Adhérent, une solution Éditique complète, des extranets & Web Services*), des services opérationnels (conseil, infogérance, maintenance, hotline, ···) et une interconnexion avec les opérateurs de santé.

2. La mission

2.1 Contexte

Le CIMUT met à disposition des organismes des portails extranet pour leur adhérents qui offrent différents services comme la consultation des remboursements, la situation de l'adhérent etc…

Depuis cette année l'extranet a évolué vers une plateforme CMS* Liferay. Des services mis à disposition par le Cimut peuvent être ajoutés à ces espaces, ce qui permet aux organismes de construire leurs espaces en toute autonomie. Ils peuvent ainsi créer des onglets et placer les différents services où ils veulent, ils peuvent également bâtrer le design de leurs espaces. (CMS de la smeба, annexe1).

Néanmoins l'architecture existante ne permet pas d'utiliser l'interface d'administration des comptes via la console d'administration Liferay. Il y a actuellement plus de 800 000 comptes extranet, ces comptes sont référencés dans une base LDAP interfacée avec Liferay, dont l'administration se fait uniquement par une console dédiée et externe donc inaccessible par les organismes.

Actuellement seul le service NTIC du Cimut a la main sur la gestion des comptes extranet. Par conséquent toutes les demandes de modification de comptes font l'objet d'un incident (FAE*). Il y a plus de 10 demandes de réinitialisation de mot de passe de compte extranet par jour, ce qui génère une charge de travail non négligeable pour le service NTIC. De plus le temps de réactivité est de plus de 24 h.

L'application qui sera mise directement à disposition des organismes supprimera cette charge de travail du service NTIC.

Pour ces différentes raisons le CIMUT m'a confié la mission de créer une passerelle d'administration des comptes extranet directement insérée dans le

catalogue de services de la solution Starweb. Cette application ne sera accessible que par les utilisateurs habilités, parmi les 2500 gestionnaires au total.

2.2 Objectif

Cette application interagira directement avec la base LDAP et devra permettre aux organismes de faire une recherche d'un adhérent selon différents critères tels que le code d'accès de l'adhérent, son INSEE ou son nom. L'utilisateur devra préciser le type de compte. Il existe trois types de compte :

- abonnés individuels qui correspond à un adhérent,
- abonnés partenaires, ce sont les partenaires de santé
- abonnés entreprises, ce sont les entreprises adhérentes

Si sa recherche contient plusieurs résultats il arrive sur la liste des résultats comportant les champs, INSEE, code d'accès, nom prénom. Depuis cette liste en cliquant sur un résultat il pourra accéder au détail de cet adhérent. Si lors de la recherche il n'y a qu'un seul résultat, il accédera directement au détail de l'adhérent.

Les champs du détail sont :

- | | | |
|----------------------|------------------------------|-------------------------------|
| • INSEE | • Code d'accès | • Nom et prénom |
| • Adresse email | • Question secrète | • Réponse à la question |
| • CGU acceptée* | • Dématérialisation* | • Dématérialisation en cours* |
| • Date d'inscription | • Date de dernière connexion | • Inscription en cours* |
| • Populations | | |

Depuis le détail l'utilisateur pourra accéder à la modification qui permettra de modifier les champs :

- INSEE
- Dématérialisation
- Dématérialisation en cours

Il pourra accéder aussi à une fenêtre permettant la réinitialisation du mot de passe (en précisant et confirmant le nouveau mot de passe). Enfin depuis le détail, l'utilisateur pourra aussi désactiver le compte.

Toutes actions réalisées sur cette application d'administration seront enregistrées dans une base de donnée ou un fichier plat.

En effet, je devrai aussi créer une interface de traçabilité des modifications horodatées et identifiées par gestionnaire.

Cette interface permettra de faire une recherche sur l'historique des actions sous différents critères (Date de début - Date de fin, Code gestionnaire, Action), et affichera la liste des résultats comportant :

- Le code d'accès du compte modifié
- L'action réalisée (modification de dématérialisation, modification de l'INSEE, etc)
- Le code gestionnaire ayant fait l'action
- La date et l'heure

Ces applications seront disponibles dans la télécommande des services Starweb et seront utilisables par les gestionnaires habilités.

2.3 Contraintes

Voici les différentes contraintes de ce projet:

- L'interrogation au serveur LDAP permettant la recherche et la modification se fera à l'aide de Web Services en utilisant Axis2.
- Une servlet* devra être mise en place pour chaque application afin de faire le lien entre l'application et les Web Services. Cette servlet devra retourner un flux en JSON à l'application.
- Un lien vers l'application d'administration sera aussi disponible au niveau de l'application NMAB (Nouvelle Maintenance Administrative) dans l'onglet contact.
- L'enregistrement des actions devra se faire avec Log4j.
- L'application devra respecter la charte graphique des IHM existantes.

- Les normes de codage devront être respectées

2.4 Technologies utilisées

2.4.1 Environnement de travail

Eclipse

J'ai essentiellement travaillé avec l'environnement de développement intégré Eclipse en utilisant Apache Maven.



L'environnement Eclipse est distribué depuis 2001. Il est à l'origine un projet IBM, qui est passé par la suite dans la « communauté du libre ». Depuis, plusieurs versions se sont succédées, la dernière version 4.3 Kepler est sortie en Juin 2013 ; pour ma part j'ai utilisé la version 4.2 Juno.

Eclipse est très lié au développement Java, mais il est très polyvalent. En fait, il permet de développer dans de nombreux langages de programmation. Je l'utiliserais d'ailleurs pour le développement intégral de mon application, et ainsi programmer en HTML et Javascript.

De nombreux plugins* sont développés, pour réaliser toutes sortes de développements dans un même environnement. Eclipse est un environnement extensible.

Par exemple au cours de ce stage j'ai utilisé le plugin m2eclipse pour l'utilisation de maven.

Maven

Maven est un outil logiciel libre pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier.



Maven utilise un paradigme connu sous le nom de Project Object Model (POM) afin de décrire un projet logiciel. Maven dispose d'un grand nombre de tâches prédéfinies, comme la compilation de code Java ou encore sa modularisation. Une dépendance sert à ajouter une librairie (.jar) dont on a besoin pour notre projet, par exemple pour créer un Web Service, nous avons besoin

de la librairie axis2, nous ajoutons donc la dépendance axis2. Les plugins permettent l'ajout de fonctionnalités. C'est dans le POM que nous déclarerons toutes les dépendances et les modules externes. La configuration du POM se fait en XML. Les plugins et les dépendances sont disponibles sur le site de Maven, ou peuvent être développées.

Jboss



JBoss Application Server (ou WildFly) est un serveur d'applications Java EE Libre écrit en Java, publié sous licence GNU LGPL. Grâce à son écriture en Java, JBoss Application Server peut être utilisé sur tout système d'exploitation fournissant une machine virtuelle Java (JVM). Je l'ai utilisé pour y réaliser les tests de mon application.

2.4.2 Langages utilisés

Java

Java est le nom de marque d'une technique informatique développée par Sun Microsystems : la « technologie Java ». Java est utilisé dans une grande variété de plates-formes depuis les systèmes embarqués et les téléphones mobiles, les ordinateurs individuels, les serveurs, les applications d'entreprise... Défini à l'origine comme un langage, « Java » a évolué au court du temps pour devenir un ensemble cohérent d'éléments techniques et non techniques.

Des standards définis par le Java Community Process (JCP), en trois éditions :

- Java SE (standard edition)
- Java EE (enterprise edition), s'appuyant sur Java SE
- Java ME (micro edition), indépendante des deux précédentes

Mon application se servira de Java EE. Effectivement j'ai utilisé le java pour le développement des servlets et des Web Service.

HTML

L'Hypertext Markup Language(HTML), est le format de données conçu pour gérer et organiser le contenu d'une page web. C'est un langage de balisage qui permet d'écrire de l'hypertexte, d'où son nom. C'est un langage de description de données, et non un langage de programmation.

 Je l'ai utilisé pour créer la partie statique de l'application web.

CSS

 CSS (Cascading Style Sheets : feuilles de style en cascade) est un langage informatique qui sert à décrire la présentation des documents HTML (et XML). Les standards définissant les CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, le CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

JavaScript

 Javascript est un langage de programmation de type script, non compilé, orienté objet, principalement utilisé dans les pages Web. C'est un langage exécuté côté client, c'est-à-dire par le navigateur de l'utilisateur. Il a pour but de dynamiser les sites Internet.

jQuery

En JavaScript j'ai utilisé plus particulièrement jQuery, qui est une bibliothèque  JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant Ajax) et HTML, et a pour but de simplifier des

commandes communes de JavaScript. C'est avec cette technologie que j'ai réalisé la partie dynamique de l'application web.

XML

XML(Extensible Markup Language) pour la définition du POM de Maven. XML,



est un langage de balisage générique standardisé par le World Wide Web Consortium (à l'origine du HTML) le 10 février 1998, il définit un ensemble de règles syntaxiques pour la présentation structurée de l'information, c'est une norme de structuration de données. Tout le monde peut créer son langage à partir des règles de syntaxes dictées par cette norme. Le XML est un méta-langage, il permet d'en créer d'autres, tel que le WSDL.

WSDL

WSDL (Web Services Description Language), est une grammaire XML permettant de décrire un Service Web. WSDL 1.1 a été proposé en 2001 au W3C pour standardisation. Le WSDL décrit une Interface publique d'accès à un Service Web, notamment dans le cadre d'architectures de type SOA (Service Oriented Architecture). C'est une description fondée sur le XML qui indique « comment communiquer pour utiliser le service ».

Le WSDL sert à décrire le protocole de communication (SOAP* RPC ou SOAP orienté message), le format de messages requis pour communiquer avec ce service, les méthodes que le client peut invoquer et la localisation du service.

2.4.3 Librairies et plugins utilisés

Les librairies contiennent l'ensemble des éléments nécessaires. Cela correspond généralement à un ou plusieurs fichiers contenant le code (généralement *.jar). Elles peuvent ainsi contenir des classes ou des méthodes qui ne sont pas décrites dans l'API

(parce qu'elles ne doivent pas être utilisées directement par exemple), voire même du code natif…

Axis2

Axis2 pour le développement des Web Services. C'est un outil permettant la



création de Web Services. Il y a deux implémentations d'axis2 : axis2/java et axis2/C. J'ai utilisé l'implémentation en Java.

Unboundid

Unboundid pour la connexion et l'interaction à l'annuaire LDAP. Unboundid est



une librairie java permettant la gestion de connexion avec les serveurs d'annuaires LDAP.

Datatables

Datatables est un plugin JQuery. C'est un outil très flexible, permettant d'ajouter des contrôles avancés sur un tableau HTML tels que le tri, la



recherche, la pagination, etc. Nous allons l'utiliser pour gérer l'affichage des résultats lors de la recherche dans l'historisation des actions.

2.4.4 Base de donnée et logs

LDAP

Le protocole Ldap (*Lightweight Directory Access Protocol*) pour interagir avec



l'annuaire Ldap des comptes extranet. C'est un protocole permettant l'interrogation et la modification des services d'annuaire. Les annuaires

Ldap sont des annuaires électroniques de type “base de données” spécialisées permettant de stocker des informations de manière hiérarchique.

SQLite

 SQLite est une bibliothèque écrite en C qui propose un moteur de base de données relationnelle. La base de données est stockée dans son intégralité (tables, déclarations, données) dans un fichier avec comme extension “db”. Les requêtes se font dans le langage SQL. J'ai utilisé SQLite pour l'enregistrement des actions réalisées dans l'application d'administration.

LOG4J

 Log4j est un projet open source distribué sous la licence Apache Software. Cette API permet aux développeurs d'utiliser et de paramétrier un système de gestion de journaux (logs*). Il est possible de fournir les paramètres dans un fichier de configuration ce qui rend sa configuration facile et souple. Log4j peut être configuré pour rediriger ces “logs” dans une base SQLite à l'aide d'un appender* JDBC, c'est ce que l'on va faire. Le logging permet de gérer des messages émis par une application durant son exécution et de permettre leur exploitation immédiate ou a posteriori. J'ai utilisé Log4j pour la gestion de l'enregistrement des actions.

2.4.5 Autres

JSON

 JSON (*JavaScript Object Notation*) est un format de données textuel, générique. Il permet de représenter de l'information structurée. Nous l'utiliserons pour Sérialiser les résultats des interrogations LDAP.

FileZilla

Concernant l'envoi de l'ensemble de mes fichiers, ainsi que mes Web Services



vers l'hébergeur web, j'ai transféré mes fichiers avec le client FTP de FileZilla. FTP signifie File Transfert Protocol ou Protocole de Transfert de Fichiers. C'est un protocole de communication qui permet l'échange de fichiers sur internet avec un réseau TCP/IP.

Le FTP fonctionne selon le modèle client-serveur. C'est-à-dire, un client depuis lequel on envoie les fichiers et un serveur FTP, sur lequel on envoie les fichiers. Un client FTP est donc un logiciel qui permet de faire la liaison entre le client et le serveur.

Pour des raisons de sécurité, FileZilla propose un mode dans lequel il ne conserve aucune trace des mots de passe sur l'ordinateur. Ainsi, une authentification est nécessaire à chaque connexion aux serveurs.

Servlet

Une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML. Ces servlets (une pour chaque application), vont servir à faire le lien entre l'application et les Web Services.

Web Services

Les Web Services sont des ensembles de fonctionnalités accessibles depuis Internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel. Je vais en créer plusieurs pour l'interaction avec l'annuaire LDAP.

2.5 L'existant : l'environnement Starweb

Starweb est divisé en deux parties, mesICIP et Portail.

Le Portail permet l'authentification des gestionnaires et permet l'affichage de la télécommande de services et mesICIP sert à la présentation et à l'appel métier.

La télécommande Starweb mise à disposition des organismes permet aux gestionnaires d'accéder aux services dont ils ont besoin, créés par le Cimut. (Voir annexe1)

Les autorisations d'accès des gestionnaires aux services sont définies par l'organisme lui-même.

L'application dont on m'a confié la tâche sera disponible sur cette télécommande Starweb.

2.6 Gantt prévisionnel

Voici le Gantt prévisionnel de ce projet :

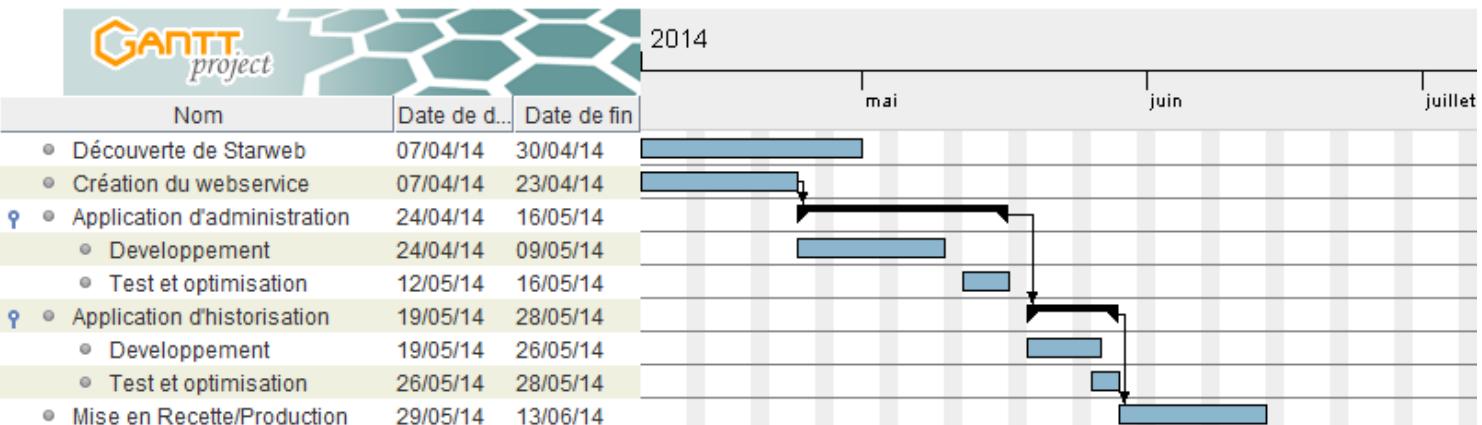


Figure 5 : Gantt prévisionnel

3. Développement de l'application web

3.1 Le Web Service

Pour faire la recherche et la modification dans la base LDAP, j'ai du créer un Web Service contenant deux services, un pour la recherche et un pour la modification.

J'ai utilisé la librairie axis2 et unboundid. Pour cela j'ai tout d'abord créé deux projets Maven.

Un première partie, "cliente", qu'on appellera « Stub » qui permettra la génération du jar qu'on utilisera dans la classe (servlet) qui fait appel à ce Web Service.

Une seconde partie, "serveur", qui permettra la définition des services et la génération du fichier aar. Afin de mettre en place ce nouveau Web Service, j'ai réalisé le déploiement de l'archive aar généré depuis le code source sur le serveur d'application au travers de son module axis2. Ceci afin de pouvoir accéder au Web Service depuis une url.

On différencie ces projets dans la définition du POM (Maven), en effet dans le deuxième projet on ajoute le plugin permettant la génération du aar.

Dans un premier temps, il faut créer un fichier WSDL qui sera identique aux deux projets. Ce fichier sert à déclarer les services, c'est à dire, leurs noms, les paramètres qu'ils prennent en entrée (ce qu'on envoie au Web Service) et les paramètres qu'ils prennent en sortie (ce qu'ils nous répondent).

3.1.1 WSDL

Voici un schéma des composants d'un fichier WSDL :

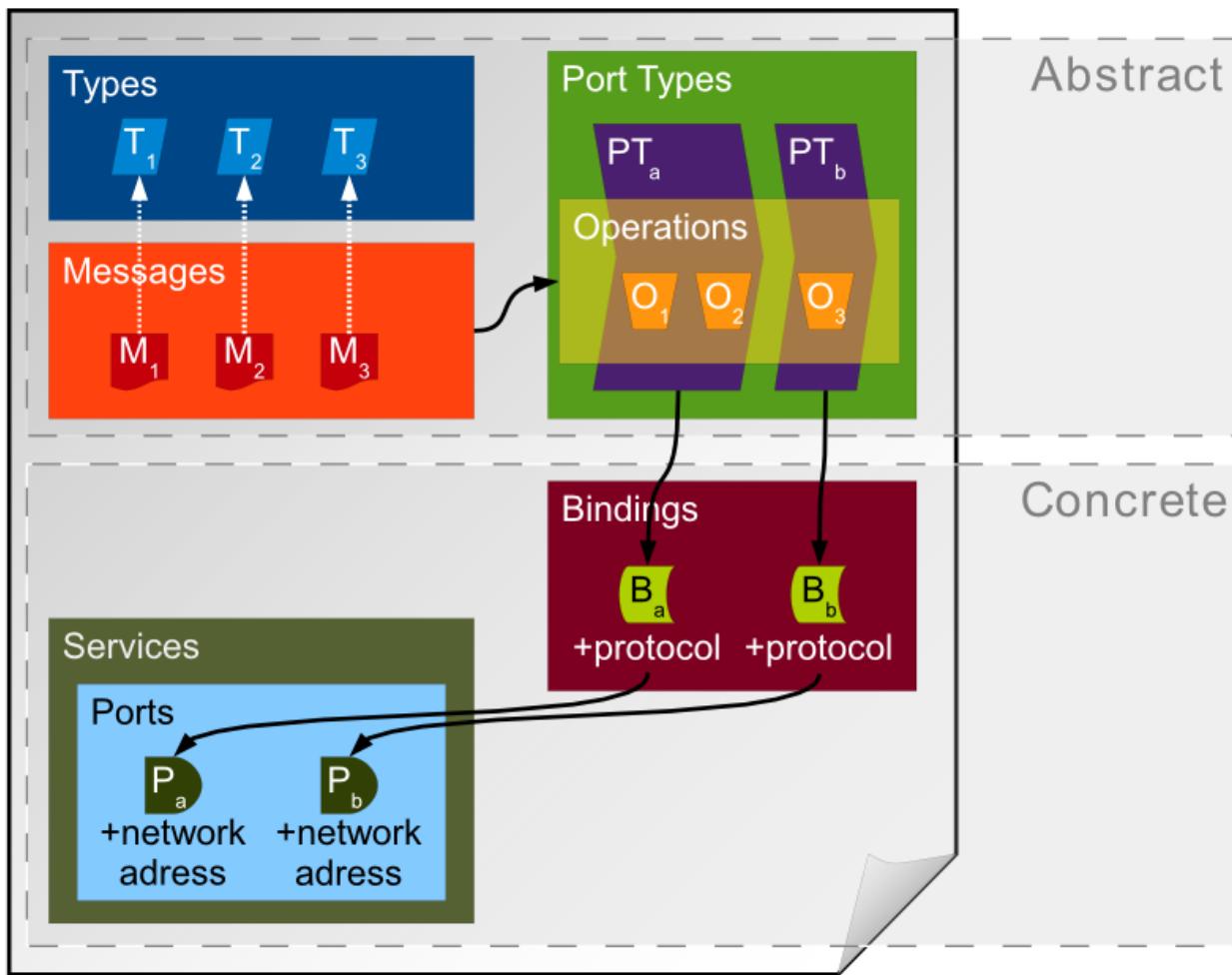


Figure 6 : Schémas WSDL (Source: <http://www.w3.org/TR/wsdl>)

Dans un premier temps, comme on peut le voir il y a les éléments “Types”, ces éléments “Types” correspondent aux types de données que nous voulons utiliser dans ces Web Services. Des types prédéfinis tels que “String” ou “int” peuvent être utilisés. Dans un second temps il faut définir les éléments “messages”. Ces éléments sont composés d’un ou plusieurs types. Un Web Service comprend deux éléments “messages”, un message d’entrée et un message de sortie.

Puis l'élément “portType” qui définit le nom de l'opération, ainsi que son message d'entrée et son message de sortie.

Ensuite, c'est dans l'élément “binding” que l'on détaille les informations nécessaires pour se connecter physiquement au service Web. Il décrit les spécifications concrètes de la manière dont le service sera implémenté. On y précise le protocol (SOAP), qui est un protocole d'invocation de méthodes sur des services distants basé sur XML, il a pour principal objectif d'assurer la communication entre machines. Le protocole permet d'appeler une méthode RPC et d'envoyer des messages aux machines distantes via HTTP. Ce protocole est très bien adapté à l'utilisation des services Web, car il permet de fournir au client une grande quantité d'informations récupérées sur un réseau de serveurs tiers.

Enfin on y indique l'opération avec laquelle il sera lié.

Dans un dernier temps on définit l'élément service qui sert à préciser les ports soutenus par le service Web on y précise le binding utilisé et l'adresse à laquelle le Web Service sera définit.

3.1.2 Développement du Web Service

Une fois le pom et le wsdl définis, nous utilisons Maven pour générer des classes à partir de ses fichiers. Maven permet de générer des classes JAVA. Parmi ces classes, les seules sur lesquelles nous allons intervenir sont les classes “Skeleton”, en effet c'est dans celles-ci que nous allons définir ce que font nos services (La recherche ou la modification dans l'annuaire LDAP).

Une méthode dans ces classes est pré-crée, elle prend en paramètres le message d'entrée défini dans le WSDL et retourne le message de sortie. Chaque Web Service dispose d'un “Skeleton” et donc d'une méthode avec en paramètre ses messages d'entrée.

3.1.3 Connexion LDAP

Pour pouvoir faire des échanges avec l'annuaire LDAP, il faut créer une connexion pour interfaçer le Web Service avec l'annuaire LDAP. Une connexion sera réalisée dans le Web Service de recherche et de modification.

Pour créer cette connexion, j'ai utilisé Unboundid. Comme dit précédemment Unboundid est une librairie java permettant la gestion de connexion avec les serveurs d'annuaires LDAP.

Cette bibliothèque contient un type connexion LDAP qui permet de créer une connexion en passant en paramètres l'host et le port de l'annuaire. Ensuite dans cette connexion on précise les identifiants pour accéder à cet annuaire grâce à la méthode bind, en y passant en paramètres le login et le mot de passe. Puis on définit un pool de connexion pour permettre plusieurs connexions simultanées. En effet, il faut gérer le cas où plusieurs gestionnaires utiliseraient l'application d'administration en même temps. Cela est géré grâce au type ConnectionPool, on lui passe en paramètre la connexion que l'on a défini juste au-dessus et sa taille (le nombre de connexions simultanées possible).

Enfin, j'ai créé une instance de cette classe et une méthode qui récupère l'instance si elle est créée, ou la crée dans le cas inverse. Ainsi il suffit d'utiliser cette méthode pour avoir l'instance et récupérer son pool de connexion pour pouvoir l'utiliser pour la recherche ou la modification dans l'annuaire LDAP.

3.1.4 Recherche LDAP

Le but du Web Service de recherche est de :

- faire une recherche sur les comptes dans l'annuaire LDAP d'après les paramètres d'entrée.
- renvoyer une liste de résultats.

Pour faire une recherche dans le serveur d'annuaire LDAP, il faut indiquer l'endroit de la recherche, c'est à dire le numéro de l'organisme (CMROC*) et le type de compte (abonnés individuels, abonnés entreprises, abonnés partenaires). En effet la

base LDAP du cimut est découpée en plusieurs parties. Chaque organisme a une section propre. Voici un schéma de la disposition de la base (9017 est le CMROC de la Smeba, il est différent pour chaque organisme) :

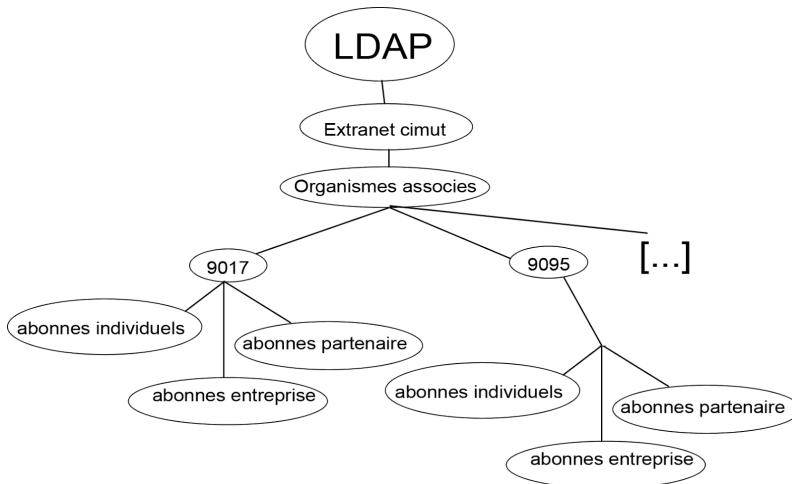


Figure 7 : Architecture de l'annuaire LDAP

Mais il faut aussi envoyer au Web Service les critères de recherche (mode et code), par exemple “uid” et “aecadyyb”.

J'ai créé une méthode qui prend en paramètre le pool de connection et les éléments précisés ci-dessus.

Ensuite on effectue la recherche sur le pool de connexion en donnant le mode, le code et l'endroit de la recherche, à l'aide du type `searchResult`.

Par la suite nous récupérons les données des comptes correspondant à la recherche à l'aide de la méthode `getAttributeValue(clé)`, où “clé” est le nom de l'attribut.

Ces données (uid, nom, mail...) correspondent au message de retour du Web Service. Elle seront renvoyées pour pouvoir par la suite être utilisées et affichées dans l'interface d'administration.

3.1.5 Modification LDAP

Le but du Web Service de modification est de faire une modification sur le compte dont l'uid aura été passé en paramètre. Et de renvoyer les données du compte comprenant les modifications. On utilise le type ModifyRequest, et on lui précise l'endroit exact du compte, le type de changement (modification), l'attribut qu'on va modifier, et la nouvelle valeur. Par la suite on applique ces modifications à l'aide du pool de connexion. (Voir *figure 8*)

```
ModifyRequest modifyRequest = new ModifyRequest(  
    "dn: uid="+uid+", "+dn,  
    "changetype: modify",  
    "replace: "+mode,  
    mode+": "+code);  
connectionPool.modify(modifyRequest);
```

Figure 8 : Modification LDAP

Une fois la modification effectuée, nous faisons une recherche sur l'uid grâce aux classes créées auparavant pour la recherche. Puis nous envoyons les résultats de cette recherche en message de retour.

3.1.6 Mise en ligne du Web Service

J'ai ensuite mis en ligne le Web Service sur le serveur de développement du Cimut. Pour cela j'ai utilisé FileZila. En se connectant sur le serveur du Cimut, j'ai pu y déposer le fichier ““aar” ainsi que la bibliothèque “jar”. Le serveur se met à jour automatiquement, et met en ligne le Web Service. A chaque modification du Web Service j'ai du re-déposer le nouveau fichier “aar”. La bibliothèque “jar”, n'est à re-déposer seulement si on a fait une modification dans le WSDL.

3.2 Environnement StarWeb: portail et mesICIP

Comme dit précédemment le Cimut dispose de son propre environnement pour le développement d'IHM.

Tout d'abord il a fallu créer une nouvelle application dans Starweb. Pour cela il

faut aller sur la page configuration Starweb, on y accède depuis le wiki du Cimut. Pour commencer j'ai créé l'application sur le serveur de développement. Depuis la page de configuration Starweb, il y a un onglet pour créer une nouvelle application. On arrive sur un formulaire (Annexe X) où il faut détailler plusieurs informations, telles que le nom de l'application, son libellé, sa date de début et de fin ou encore son code métier. Une fois correctement rempli, on peut le valider. L'application est alors directement créée.

Par la suite il a fallu installer sur mon poste cet environnement, comprenant les deux projets java "portail" et "mesICIP". Ces projets englobent l'intégralité du produit Starweb. Tout l'environnement est déjà défini et aide à la construction d'IHM.

La présentation d'une IHM sur StarWeb est répartie en plusieurs zones. Tout d'abord il y a une zone concernant la recherche, nommée "vRecherche", cette zone contient un formulaire situé en haut de la page.

Ensuite il y a une zone dite "supérieur", nommé "vSuperieur", située juste en-dessous de la zone de recherche. Finalement, une zone dite "inférieur", nommé "vInferieur". Dans ces deux dernières nous affichons les résultats de la recherche. Dans notre cas "vSupérieur" servira à l'affichage des détails, de l'affichage liste, ou encore pour les modifications. "vInferieur" quant à elle, servira pour l'affichage de l'historique des actions réalisées sur un compte.

A noter, tout en haut de la page, une zone de message permettant d'afficher des indications à destination de l'utilisateur, comme par exemple "Modification de mot de passe effectué". Cette zone se présente sous la forme d'une petite pop-up qui peut, à tout moment être fermée par l'utilisateur.

Seule la zone de recherche est systématiquement présente tandis que les autres zones ne sont affichées que suivant les besoins.

Voici le schéma d'une IHM dans starweb ci-dessous.

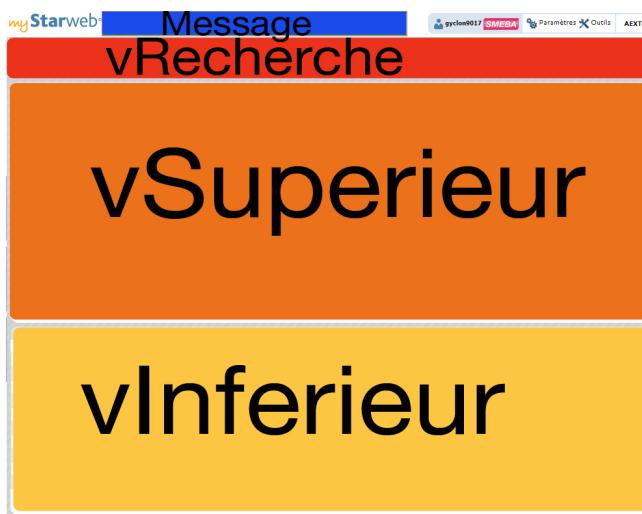


Figure 9 : Schema IHM starweb

Dans mesICIP, l'organisation du développement se fait dans plusieurs fichiers. Tout d'abord l'index (NOMindex.htm) : c'est ici que l'on charge les différents scripts que nous allons utiliser, tels que jQuery ou l'environnement Starweb. C'est aussi là que l'on réalise le lien avec le fichier qui servira à la recherche (NOMrecherche.htm). Dans ce dernier, on y construira le formulaire de recherche et l'action à réaliser lors de sa soumission, c'est à dire l'appel à une servlet et à un fichier "js" pour l'affichage du résultat.

La servlet correspond à la partie qui va faire le lien entre Web Service et les fichiers "js". Ces fichiers "js" afficheront le résultat à l'écran à l'aide de HTML et javascript.

Dans l'environnement du Cimut, les fichiers "js" sont regroupés en 3 catégories : "nom_L" "nom_D" et "nom_E".

L (Liste)	Affichage d'une liste de résultat
D (Détail)	Affichage de détail d'un résultat
E (Edition)	Affichage du mode édition (modification)

Dans notre cas lors d'une recherche, s'il y a plusieurs résultats nous irons sur l'affichage liste, puis nous pourrons afficher les détails en cliquant sur un résultat. Si il

n'y a qu'un résultat nous passons directement en mode détail. Depuis le mode détail nous pouvons passer en mode édition.

3.3 L'application d'administration

3.3.1 Fonctionnement de l'application

J'ai créé un diagramme de séquence pour montrer ce qui devra se passer lors de chaque action (affichage des détails, affichage d'une liste, modification).

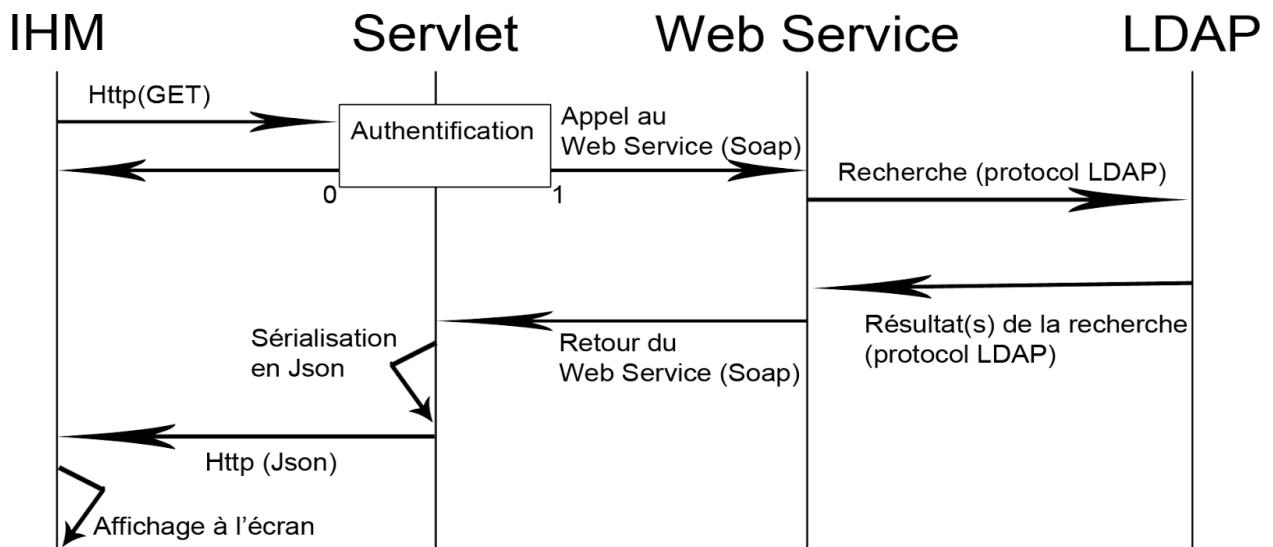


Figure 10 : Diagramme de séquence de l'application

3.3.2 La recherche

Il nous faut dans notre application un formulaire de recherche ; dans celui ci, il y aura plusieurs champs. Tout d'abord trois champs permettant d'entrer un INSEE, un code d'accès ou un nom. Il faut savoir que la recherche ne se fait que sur un champ, l'utilisateur ne pourra donc faire sa recherche que sur l'un de ces trois champs. Ensuite dans ce formulaire une liste déroulante est présente, permettant de détailler le type de compte. En effet il y a trois types de compte, "abonnes individuels" (simple adhérent), "abonnes partenaire" (partenaire de santé), et "abonnes entreprise" (une entreprise). La valeur de la liste déroulante sera par défaut "abonnes individuels", car cela représente

la grande majorité des comptes.

Une fois un de ces champs rempli et le compte détaillé, l'utilisateur pourra effectuer la recherche.

Comme précisé auparavant, lors d'une recherche, deux modes de résultats sont possibles. Nous partirons sur un affichage des résultats en mode liste pour l'affichage de plusieurs résultats et directement en mode détail si un seul s'affiche. Il existe un fichier "JS" pour chacun de ces types d'affichage.

Pour faire le lien entre les différentes parties de l'application nous utilisons une servlet. Cette servlet nous renvoie le ou les résultats de la recherche. La recherche s'effectue depuis l'IHM en s'adressant à cette servlet via un appel Ajax. Un appel Ajax est composé de plusieurs éléments, en particulier :

- Une URL, celle qui pointe sur notre servlet.
- Le type de message retour (JSON dans notre cas)
- Les paramètres à transmettre à notre servlet.
- méthode (dans notre cas un GET).

En cas de succès de la requête, la réponse de notre servlet sera récupérée sous un format de type JSON. En cas d'erreur, les traitements appropriés seront effectués.

C'est l'exécution du code javascript associer au mode d'affichage, en conjonction des informations transmises par notre servlet qui conditionne l'affichage du résultat.

On précise aussi la fenêtre "frame" dans laquelle sera chargé le script. Dans notre cas cela sera "vSuperieur". Enfin nous utilisons la méthode "displayData()", avec le JSON en paramètres. C'est cette méthode, définie dans chaque script qui va servir pour l'affichage du ou des résultats.

Vous remarquerez aussi l'utilisation des fonctions "commencerChangement" et "terminerChangement". Ces deux fonctions sont définies dans l'environnement starweb. Cela permet d'afficher le message "Changement en cours" et de bloquer l'accès à tous les boutons. Lorsque qu'un chargement est commencé, nous bloquons

toute action coté IHM jusqu'à ce que la transaction soit effectuée.

Voici l'appel Ajax qui est réalisé lors de la recherche.

```

$.gestionnaireTransaction.commencerChargement({cible: 'vSuperieur'});
$.ajax({url : url ,
        type : 'GET',
        async : true,
        dataType: 'json',
        success : function(json, textStatus, jqXHR){
            var script;
            if (json.detail!=null){ //1 résultat => mode détail
                script="JS/FonctionsJavaScriptIHM_AEXT_D.js";
            }else{ //Plusieurs résultats => mode liste
                script="JS/FonctionsJavaScriptIHM_AEXT_L.js";
            }
            $.getScript(script, function(){
                $.gestionnaireTransaction.terminerChargement({cible: 'vSuperieur'});
                var $frame = $('#vSuperieur');
                displayData(json);

            });
        },
        error : function(jqXHR, textStatus, errorThrown) {
            $.gestionnaireTransaction.terminerChargement({cible: 'vSuperieur'});
            $.jMessage({
                message : "Une erreur technique est survenue",
                type : 3
            });
        }
    });
}

```

Figure 11 : Un exemple d'appel ajax

3.3.3 La servlet

La servlet est ce qui va permettre de faire le lien entre l'IHM et les Web Services.

La servlet fera appel au Web Service et transmettra les résultats sous la forme d'un flux JSON à l'IHM.

Tout d'abord j'ai utilisé une méthode d'authentification de starweb permettant d'identifier un gestionnaire, de récupérer son code de gestionnaire, son CMROC (cela permet de savoir à quel organisme il appartient) et son CMSID. Le CMSID correspond aux droits dont dispose le gestionnaire sur cette application. Suivant ces droits, il

n'aura pas accès à l'intégralité de l'application. Les droits des gestionnaires seront définis par l'organisme, via la configuration de Starweb. Il y a trois types de droit :

“I” (Interrogation)	Pour avoir accès à l’application en elle même
“M” (Modification)	Pour avoir accès à la modification du compte
“S” (Suppression)	Pour avoir accès à la désactivation du compte

Le code de gestionnaire et le CMROC nous serviront lors de l'appel au Web Service. Cette authentification me servira aussi pour détecter et afficher un message si la session est expirée.

C'est le paramètre d'appel "ecran" de la servlet qui détermine le mode de fonctionnement de celle ci.

Par exemple, dans le cas où l'écran est défini à "AEXT_L", on cherche à obtenir le résultat de la recherche en mode liste. Cette liste regroupera uniquement l'INSEE, le code d'accès et le nom de chaque compte. Si la liste ne contient qu'une seule occurrence, le résultat est retourné en mode détail. Dans ce mode détail, le JSON contiendra toutes les informations du compte.

Pour cela nous avons d'abord récupéré, les différents paramètres envoyés, grâce à la fonction "getParameter()". Avec cela nous avons déterminé si l'utilisateur a fait une recherche avec un INSEE, un code d'accès ou un nom.

Nous avons défini l'endroit du Web Service grâce à l'url de celui ci (Voir figure x). Puis on lui envoie le type de la recherche (INSEE, uid...), la valeur associée (1891222...), le code du gestionnaire, le CMROC, et enfin le type de compte. En mettant ces derniers dans une variable d'un type spécifique créée dans le wsdl du Web Service. On effectue alors la recherche en passant en paramètre cette nouvelle variable.

Cette recherche nous renvoie en réponse les résultats.

```
ConsultationUidStub stub3 = new ConsultationUidStub("http://lci9.cimut.fr:94/axis2/services/consultationUid");
RechercheUid recherche = new RechercheUid();
recherche.setGest(codeGes);
recherche.setCode(code);
recherche.setMode(mode);
recherche.setCmroc(cmroc);
recherche.setCompte(compte);
RechercheUidResponse user = new RechercheUidResponse();
user = stub3.rechercheUid(recherche);
```

Figure 12 : Appel au Web Service

On récupère les résultats grâce au méthode créé dans le Web Service.

```
UserOut utilisateur = user.getUser();
if(utilisateur.getUserOutSequence()!=null){
    if(utilisateur.getUserOutSequence().length==1){//1 seul résultat
        UserOutSequence seq =utilisateur.getUserOutSequence()[0];
        //Récupération des informations du compte trouvé
        String uid1=seq.getUid();
        String assureinsee = seq.getAssureinsee();
        String cn=seq.getCn();
        String mail=seq.getMail();
        String assurequestion=seq.getAssurequestion();
        String assurereponse=seq.getAssurereponse();
        String iscgu=seq.getIscgu();
```

Figure 13 : Récupération des résultats

Dans les spécifications, nous avons vu que la servlet devait renvoyer un JSON à l'IHM. J'ai du ainsi sérialiser les résultats en JSON. Pour cela j'ai utilisé une librairie java, "org.JSON.simple" permettant ceci.

Dans la deuxième partie de la servlet. C'est le cas où l'on vient de réaliser une modification ou lorsqu'on était en mode liste, et que l'utilisateur veut afficher les détails d'un résultat. Si on est en mode liste, on enregistrera dans le JSON deux nouveaux attributs "codeR" et "modeR" qui correspondent au code et au mode de la recherche. Ces derniers serviront si l'utilisateur clique sur le bouton "Retour liste", présent dans le mode détail.

Dans ce cas, la servlet appelle le Web Service de modification si une modification a été effectuée, puis nous appelons le Web Service de recherche pour effectuer la recherche des détails du compte.

Lors d'une modification, la servlet renvoie aussi un nouvel attribut dans le JSON qui est "msgRetour", il correspond au message renvoyé par le Web Service à propos de la modification. Ce message indique si la modification s'est bien réalisé ou non.

Puis la servlet renvoie en JSON les détails du compte. La dernière partie de la servlet représente le cas où on ne fait pas d'appel au Web Service, par exemple quand on passe de la page de détail à la page de modification. Cette partie de la servlet permet de retransmettre le JSON à l'application, mais avec quelques modifications. En effet on réinitialise l'attribut "msgRetour" du JSON.

3.3.4 L'affichage liste

L'affichage de la liste (résultat annexe 4) des résultats s'effectue sous la forme d'un tableau. Afin de créer ce tableau dans la zone "vSuperieur", on crée un tableau avec quatre colonnes : INSEE, code d'accès, nom prénom, et la dernière qui comprendra un bouton (pour permettre de passer à l'affichage des détails du compte sélectionné).

Une fois que ce tableau rempli est affecté à la variable "str", on utilise la fonction jquery "append" appliquée à "str" sur la zone "vSuperieur", ce qui permet d'ajouter ensuite un élément du contenu HTML. Ainsi le tableau s'affichera dans cette zone "vSuperieur" de

l'application. Voici le code de cette explication :

```

var str = '<div id="noformSup">', i, color=['paire','impaire'];
//Si il y a des résultats à la recherche: affichage en mode liste
if(ListeMLDAP.length > 1) { //ListeMLDAP correspond à la liste des résultat présent dans le json
    //On met la variable testRet à 1, ce qui servira pour définir les variables pour le retour liste
    testRet=1;
    str += '<table class="tabliste" id="tabliste_'+ lFrame +' width="100%">'; //(lFrame=vSupérieur)
    str += '<tr>';
    str += '<th></th>';
    str += '<th>INSEE</th>';
    str += '<th>Code d\'accès</th>';
    str += '<th>Nom Prénom</th>';
    str += '</tr>';
    //Pour chaque résultat une ligne
    for (i=0;i<ListeMLDAP.length;i++) {
        str += '<tr class="'+color[i%2]+'">';
        //On passe en mode détail quand l'utilisateur click sur l'icone: icone-visualiser
        str += '<td width="15px">' +
        '<a style="cursor: pointer;" class="ci-icon ci-icon-visualiser" onclick="goDetail('+i+');"></a></td>';
        str += '<td>' +ListeMLDAP[i].insee+'</td>';
        str += '<td>' +ListeMLDAP[i].uid+'</td>';
        str += '<td>' +ListeMLDAP[i].cn+'</td>';
        str += '</tr>';
    }
    str += '</table>';
    //Si il n'y a pas de résultat on affiche: aucun résultat
} else
    str +=$.aucunResultat();
str += '</div>';
//On ajoute dans vSupérieur str
$('#'+lFrame).append(str);
//Définition du nom
$('#noformSup', '#'+lFrame).zone({titre:'Liste des résultats de la recherche'});

```

Figure 14 : Affichage des résultats (Liste)

Pour passer à l'affichage des détails, on réalisera un nouvel appel Ajax à la servlet qui, en cas de succès exécutera le script de détail. Voici un schéma de cette appel.

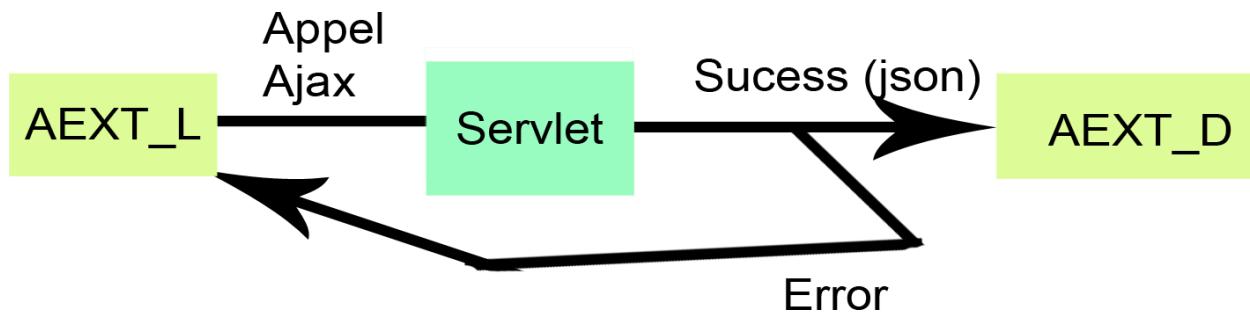


Figure 15 : Schéma d'un appel Ajax

Lors de la définition du JSON dans la liste on a créé les attributs “uid”, “INSEE” et “cn”.

Par la suite on remplit ce tableau en récupérant ces attributs à l'aide d'une boucle "for" jusqu'à que tous les résultats de la liste aient été bien traités.

3.3.5 Affichage des détails

L'affichage des détails (résultat annexe 5) se fait lui aussi sous la forme d'un tableau. Mais dans celui-ci, les noms des attributs s'afficheront sur la première colonne et la valeur de ses attributs sur la deuxième colonne, comme ci-dessous:

Attribut 1	Valeur 1
Attribut 2	Valeur 2

Pour cela on reprend le même principe que pour l'affichage de la liste, on définit le contenu HTML dans une variable "str", puis on l'ajoute à "vSuperieur".

Dans l'affichage des détails, plusieurs boutons seront présents, pour passer à la modification du compte ou à la réinitialisation du mot de passe, pour désactiver/activer le compte (selon si le compte est activé ou désactivé, le bouton change) et pour faire un retour à l'affichage liste (Bouton présent que si on est passé par l'affichage liste).

Pour créer ces boutons, j'ai utilisé une méthode déjà présente sur starweb nommée boutonJS.

Cette méthode permet d'ajouter un bouton en passant en paramètre le libellé du bouton, l'icône du bouton (icône définie dans un fichier css), et l'évènement à réaliser lors du clic (Script javascript). On peut aussi y définir une condition pour que le bouton soit actif (visible). Tout les boutons doivent être créés dans un tableau "btn" puis ajoutés dans leur zone (Dans notre cas "vSuperieur"). Voici un exemple, correspondant au bouton permettant d'aller à la réinitialisation de mot de passe.

```
btn[btn.length] = $('#'+lFrame)
.boutonJS({"texte":"R&acute;initialiser mot de passe","icone":"unlock","actif":autorise("M")}).on('click',goModifMdp);
```

Endroit où ajouter le bouton
Libellé du bouton Icône du bouton Condition pour que le bouton soit visible Script javascript à exécuter

Figure 16 : Création d'un bouton

Le script goModifMdp réalise un appel Ajax (Comme pour la recherche).

Le bouton n'est actif que si l'utilisateur a le droit "M", ainsi la fonction autorise (droit), renvoie "true" ou "false", suivant si l'utilisateur a le droit nécessaire.

Dès qu'une modification est réalisée, on revient toujours à l'affichage en mode détail. C'est donc dans celui-ci que nous allons afficher un message, pour dire si la modification a bien été effectuée. Ce message est, comme dit précédemment, défini dans le JSON. Pour afficher ce message, nous allons nous servir de la partie jMessage de starweb.

Voici comment on le définit :

```
if(lejson.msgRet!=null){  
    $.jMessage({  
        message : lejson.msgRet, //Libellé du msg  
        type : 0 //Type de msg 0 = Information  
    });  
}
```

Figure 17 : Affichage d'un message

3.3.6 Modification du compte

Une fois que l'utilisateur a cliqué sur le bouton "Modification" depuis l'affichage du mode détail, il arrive dans ce mode de modification (résultat annexe 6).

Ce mode reprend la même structure que le mode d'affichage détail. A noter, qu'à la place de l'INSEE il y a une zone de texte (La valeur par défaut de cette zone est l'INSEE), et à la place de la "dématérialisation" et "dématérialisation en cours" il y a des boutons radios "Oui" "Non" (La valeur du compte est présélectionnée par défaut). Et seuls deux boutons sont présents "Valider" et Annuler".

Lorsque l'utilisateur valide les modifications, on compare les anciennes valeurs des champs aux nouvelles pour voir si il y a eu des modifications. Ainsi dans une variable on ajoute les valeurs "De", "En" ou "In", puis on fait encore un appel Ajax. On envoie à la servlet cette variable ainsi que les valeurs entrées dans les champs. La servlet regardera les valeurs présentes dans cette variable pour savoir quelle(s) modification(s)

réalisée(s). Une fois la modification traitée, la servlet renvoie (en JSON) le détail du compte avec les nouvelles valeurs. On retourne ainsi dans le mode détail.

Si l'utilisateur annule, il repasse directement dans le mode détail.

3.3.7 Réinitialisation de mot de passe

Une fois que l'utilisateur a cliqué sur “réinitialiser le mot de passe” depuis le mode détail, il arrive dans ce mode “Réinitialisation” (résultat annexe 7).

La page de réinitialisation de mot de passe est comme les autres sous la forme d'un tableau, mais ne comprend que l'uid du compte et deux champs “Nouveau mot de passe” et “Confirmer nouveau mot de passe” avec pour chacun d'eux une zone de texte permettant d'écrire le mot de passe.

J'ai créé une indication en face de la deuxième zone de texte, pour permettre à l'utilisateur de savoir en temps réel si les deux mots de passe sont identiques.

S'ils sont identiques, une icône verte “√” apparaît, dans le cas contraire, il y a une icône rouge “X”. Pour ce faire, j'ai créé une fonction qui compare les deux valeurs, et change l'icône en fonction. Cette fonction est appelée lors de l'évènement “onChange” de javascript sur les deux zones de texte.

3.4 L'application d'historisation

3.4.1 Base de donnée et Log4j

J'ai d'abord créé une base de donnée SQLite dans laquelle j'ai créé une table, nommée “log”, comprenant la date, un code d'accès, un code gestionnaire, une action, un CMROC et le type de compte.

Par la suite pour pouvoir entrer les modifications dans cette table, j'ai utilisé log4j, qui permet de logger des informations de différentes manières.

J'ai créé un appender JDBC, pour logger dans une base de donnée SQLite. (Voir figure 18).

Dans cette appender j'ai du y détailler l'url, c'est à dire l'emplacement de la base de donnée. Ainsi que l'user et le password (malgré que cela soit inutile, car SQLite ne gère

pas de session). J'ai enfin défini la requête SQL à effectuer lors de l'appel à cet appender.

```
<appender name="JDBC_SQLITE" class="org.apache.log4j.jdbcplus.JDBCAppender">
    <param name="url" value="jdbc:sqlite:${jboss.server.log.dir}/LogAEXT/logAEXT.db" />
    <param name="dbclass" value="org.sqlite.JDBC" />
    <param name="username" value="aaa" />
    <param name="password" value="" />
    <param name="sql" value="INSERT INTO log (Date, Uid, Action, Gest, Compte, Cmrc)
VALUES('@LAYOUT:3@ @LAYOUT:4@', '@MDC:Uid@', '@MDC:Action@', '@MDC:Geste@', '@MDC:Compte@', '@MDC:Cmrc@') " />
    <param name="layoutPartsDelimiter" value="#-#" />
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="#{t} #--##-#%d{yyyy-MM-dd}#-#%d{HH:mm:ss}"/>
    </layout>
</appender>
```

Figure 18 : Appender Log4J

Les éléments de type “@LAYOUT:X@” correspondent à la Xème partie dans le ConversionPattern. Dans ce dernier la troisième partie correspond à la date sous la forme aaaa/mm/jj, et la quatrième partie correspond à l'heure. Je les utilise pour entrer la date et l'heure de la modification dans la table.

Les éléments de type @MDC:attribue@ correspondent à la valeur donnée à cette attribut.

En effet dans le Web Service de modification j'ai ajouté un logger configuré avec cette appender et avant de faire appel à ce logger j'ai défini plusieurs valeurs dans le MDC. Ci-dessous le Logger et la méthode appelée lors d'une modification pour l'enregistrer dans la Table. Après l'appel au Logger, je “remove” les différents attributs du MDC.

```
private static Logger LOG1 = Logger.getLogger("LoggerSQL");
public void traceModif2(String uid, String action, String codeGes, String compte, String cmroc) {
    //Log dans une base SQLite avec log4j
    MDC.put("Uid", uid);
    MDC.put("Action", action);
    MDC.put("Gest", codeGes);
    MDC.put("Compte", compte);
    MDC.put("Cmroc", cmroc);
    LOG1.info("appel");
    MDC.remove("Uid");
    MDC.remove("Action");
    MDC.remove("Gest");
    MDC.remove("Compte");
    MDC.remove("Cmroc");
}
```

Figure 19 : Logger dans le Web Service de modification.

3.4.2 Le Web Service

Pour faire une recherche dans la base de donnée, j'ai créé un nouveau Web Service. J'ai défini le WSDL, pour que ce service prenne en paramètre un CMROC, un code Gestionnaire, un code d'accès (uid), et une plage horaire composée d'une date de début et d'une date de fin.

Tout les champs ne sont pas obligatoires pour effectuer la recherche. Le CMROC est obligatoire ainsi que le code gestionnaire ou le code d'accès. Cependant, on peut préciser la recherche en détaillant une date de début et/ou une date de fin, ou encore en indiquant le code gestionnaire et le code d'accès. Le CMROC est utilisé pour effectuer la recherche seulement sur les actions réalisées par les gestionnaires de l'organisme de l'utilisateur.

On crée d'abord une connexion à la base de donnée, puis suivant ce que le Web Service reçoit, il construit sa requête SQL différemment (Figure 20). Une fois la requête créée, nous l'exécutons pour qu'elle nous renvoie les résultats. Les résultats de cette recherche seront insérés dans une liste avant d'être renvoyés.

```
Connexion connex = new Connexion("/opt/jboss-5.1.0.GA/server/cimutWebSrvcsDEV/log/LogAEXT/logAEXT.db");
connex.connect();
String query = "SELECT * FROM log where Cmroc = '"+cmroc+"' and ";
if(gest!=null & uid!=null){
    query=query+"Uid= '"+uid+"' and Gest = '"+gest+"' and date BETWEEN '"+debut+"' AND '"+fin+"'";
}else if(gest!=null){
    query=query+"Gest = '"+gest+"' and date BETWEEN '"+debut+"' AND '"+fin+"'";
}else if(uid!=null){
    query=query+"Uid = '"+uid+"' and date BETWEEN '"+debut+"' AND '"+fin+"' ";
}
ResultSet resultSet = connex.query(query);
```

Figure 20 : Création de la requête SQL

3.4.3 Crédation

Pour cette application j'ai créé une nouvelle IHM nommée "Historique administration des comptes extranet". Avec le même principe que l'administration des comptes extranet. Ainsi, j'ai créé une nouvelle servlet pour faire appel au nouveau Web Service. Un fichier de recherche, un fichier "index" et aussi un fichier JS, "AEXT_L". Cette IHM ne renvoyant qu'une liste et n'ayant par conséquent qu'un mode d'affichage ne nécessitant qu'un seul fichier JS.

Le formulaire de cette application est composé de quatre zones de texte. Une pour saisir un code de gestionnaire, une autre pour saisir un code d'accès et deux pour saisir une plage horaire (date de début et date de fin).

Pour la saisie des dates j'ai aussi ajouté un datepicker grâce à une méthode présente dans starweb, pour permettre à l'utilisateur de choisir sa date dans un calendrier.

Une fois le formulaire rempli, lorsque l'utilisateur l'envoie, on garde le même principe que pour l'application d'administration. On réalise un appel Ajax, mais cette fois-ci vers une nouvelle servlet HEXT, créée essentiellement pour cette application.

3.4.4 la servlet HEXT

Cette servlet a pour seule fonctionnalité la réalisation d'une recherche dans la base de donnée SQLite, contenant l'historique des dernières actions.

Cette servlet, comme la servlet AEXT, renverra à l'application un flux de JSON.

On utilise à nouveau la méthode d'authentification pour récupérer le CMROC.

La servlet récupère en paramètre GET les différents champs du formulaire de recherche. Pour commencer, ces paramètres sont enregistrés dans des variables distinctes. Ensuite, si une date de début ou de fin (ou les deux) a été envoyée on modifie son format. En effet, lors de la saisie du formulaire, elle se présente sous la forme de “jjmmaaaa” alors que dans la base de donnée elle affiche un format de type “aaaa-mm-jj”.

Pour ce faire j'ai utilisé la fonction substring de java(Figure x).

```
if(dateD!=null){  
    temp1=dateD.substring(0,2);  
    temp2=dateD.substring(2,4);  
    temp3=dateD.substring(4,8);  
    dateD=temp3+"-"+temp2+"-"+temp1;  
}
```

Figure 21 : Changement du format de la date

Par la suite on utilise ces variables pour faire appel au Web Service. Le Web Service nous renvoie la liste des résultats qui seront sérialisés dans un tableau de JSON. Ce tableau de JSON aura la forme [[Uid, Gest, Action...],[...],[...]]. Nous verrons par la suite pourquoi j'ai choisi ce format.

3.4.5 Affichage de la liste

Pour afficher la liste (résultat annexe 8) des résultats de la recherche, on garde toujours le même principe que pour l'application d'administration. En effet on insère dans vSuperieur un tableau. Mais cette fois-ci nous allons utiliser un plugin jquery, nommé Datatables, pour pouvoir paginer et trier ce tableau suivant les différentes colonnes. Datatables va aussi nous servir pour insérer des valeurs dans le tableau. Pour utiliser ce plugin il suffit d'importer le fichier “jquery.dataTables.js” dans l'index, qu'on aura téléchargé auparavant.

Pour commencer on insère avec un “append” un tableau dans vSuperieur, ce tableau aura pour id “tabliste”. Ensuite on définit ce tableau grâce à dataTable, comme sur extrait de code ci-après. Dans un premier temps j'avais ajouté les lignes dans le tableau manuellement au moment de sa déclaration. Puis par la suite, j'ai changé de

méthode et j'ai utilisé l'attribut "aaData" de dataTable, qui sert à ajouter des valeurs dans le tableau depuis un JSON qui doit être sous la forme [[...],[...][...]] C'est pour cela que j'ai sérialisé la liste des résultats dans la servlet, dans un JSON de cette forme.

```
$('#tabliste').dataTable( {
    "aaSorting": [[0, 'desc']], //On tri de base sur la 1er colonne(Date) par ordre décroissant
    "aaData": Liste, //On ajoute les données depuis le Json
    "bFilter": false, //On désactive le filtre de recherche
    "bPaginate": true, //On active la pagination
    "bLengthChange": false, //Le nombre de résultat par page n'est pas modifiable
    "sPaginationType": "full_numbers", //Pour naviguer entre les page << < 0 1 2 > >>
    "iDisplayLength": 20, //Nombre de résultats par page
    "oLanguage" : {
        "sInfo": "R&eacute;sultats _START_ &grave; _END_ sur _TOTAL_",
        "oPaginate": {
            //On supprime le nom des boutons pour naviguer entre les pages, on les remplacera par un icone
            "sFirst": " ",
            "sPrevious": " ",
            "sNext": " ",
            "sLast": " "
        }
    }
});
```

Figure 22 : Configuration DataTable

3.5 Implémentation des applications dans l'architecture

Pour réaliser ces deux applications, j'ai du respecter l'architecture du Cimut.

Les IHM du cimut fonctionnent avec une servlet “NonRepeatableExchangeController”, cette servlet ne répondant pas au besoin de l'application, j'ai du en créer de nouvelles. Voici un schéma de l'implémentation dans l'architecture, on peut voir que je suis intervenu à tous les niveaux de l'architecture :

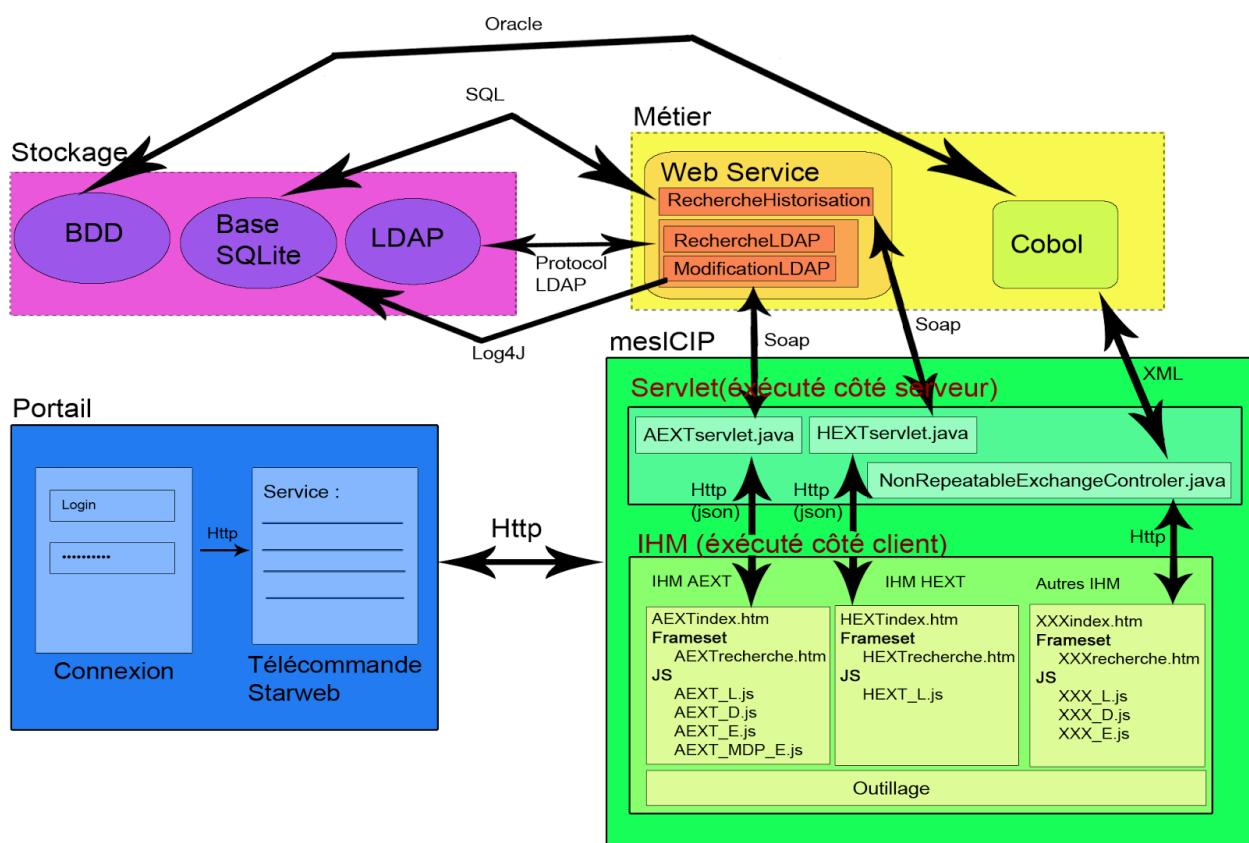


Figure 23 : Schéma de l'implémentation dans l'architecture

3.6 Mise en ligne des applications

La mise en production passe par plusieurs étapes, tout d'abord une phase de développement, une phase de recette et enfin une phase de mise en production. L'application a été développée en local. Pour l'installer sur le serveur de développement du Cimut j'ai utilisé FileZilla. J'ai inséré tous les fichiers que j'ai créé en gardant la même arborescence que sur le schéma ci-dessus. Par exemple j'ai inséré tous mes fichiers "js" dans le dossier JS.

L'application sera très prochainement mise en recette, puis mise en production.

3.7 Test

Suite aux tests, toutes les vérifications ont été faites pour que l'utilisateur soit obligé de saisir au moins un élément dans les formulaires. En effet dans l'application

d'administration l'utilisateur doit saisir un INSEE ou un code d'accès ou un nom (au choix) et dans l'application d'historisation il doit au moins saisir un code gestionnaire ou un code d'accès (Il peut également saisir les deux).

Les vérifications pour la validité de la date et de l'INSEE dans les formulaires ont aussi été réalisées (en javascript).

Un message prévient l'utilisateur lorsqu'un problème a été rencontré.

3.8 Réclamations des utilisateurs

Fin mai, dans le cadre d'une réunion du groupe de travail extranet à Paris, monsieur LAOT a présenté cette l'application aux utilisateurs. Les utilisateurs étaient dans l'ensemble satisfaits, mais plusieurs nouvelles exigences ont été formulées. Tout d'abord les utilisateurs veulent avoir dans les détails de l'application un résumé des dernières actions faites sur ce compte, cela sans avoir à passer par la fonction d'historisation. Ils veulent également que la dernière modification soit affichée au-dessus des boutons (Modification de Mot de passe, etc) et qu'en cas de modification effectuée, un message d'alerte s'affiche. J'ai donc ajouté dans "vlnferieur" les modifications réalisées sur ce compte en me servant de la servlet de l'application d'historisation. Puis j'ai affiché la dernière modification suivant sa date, soit en petit au dessus des boutons soit en gros et visible au dessus des informations de détail.

3.9 Gantt réel

Voici le Gantt réel de ce stage :

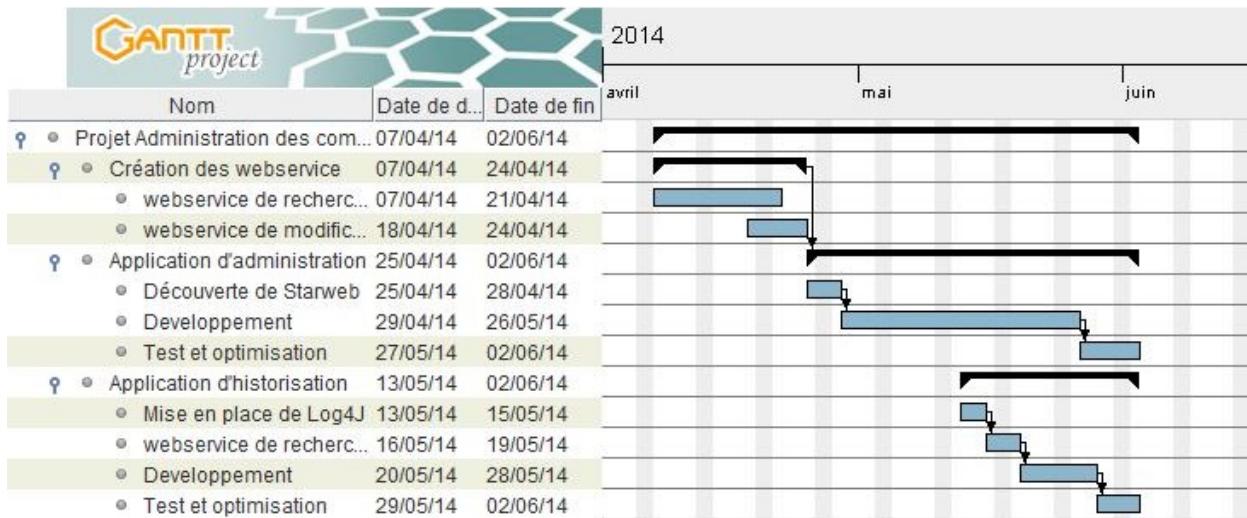


Figure 24 : Gantt Réel

Conclusion

L'administration des comptes extranet des organismes se fait actuellement grâce à une console externe disponible uniquement au Cimut, ce qui ne permet pas aux organismes de réaliser l'administration eux-même, et induit une charge de travail supplémentaire au service NTIC du Cimut. Le Cimut m'a donc confié la mission de créer une application qui sera mise à disposition des organismes, pour qu'ils puissent gérer leurs comptes extranet et une seconde application pour qu'ils puissent consulter les modifications réalisées sur les comptes extranet.

Les deux applications sont finalisées. En effet toutes les spécifications faites par le Cimut et par les utilisateurs ont été prises en compte. Ainsi les applications sont fonctionnelles, l'administration des comptes extranet est donc opérationnelle via cette application. La mise en production est prévue dans le cadre d'une livraison qui doit intervenir d'ici mi-juillet.

Ce stage s'est merveilleusement bien passé. Il a été très enrichissant, j'ai pu y découvrir de nouvelles technologies que je n'ai pas eu l'opportunité de voir lors de mon cursus à l'université telles que le Javascript (dont Jquery), le XML, WSDL, Maven, Log4J, SQLite, Axis2 pour la création de Web Service. De plus j'ai découvert un environnement professionnel intéressant grâce à sa solution Starweb.

J'ai appris à m'adapter aux normes de développement d'un environnement. Ce stage aboutit sur un CDD d'un mois et demi, ce qui me permettra entre autre de réaliser la mise en production des deux applications.

Résumé

J'ai réalisé ce stage dans le cadre de ma formation pour l'obtention du DUT Informatique. Il a eu lieu entre le 7 avril et le 13 juin 2014 au CIMUT à Quimper, qui est le centre informatique des mutuelles. J'ai rejoint l'équipe IHM, dans l'objectif d'élaborer deux applications web qui seront mises à disposition des organismes. Ces applications permettront l'administration de comptes extranet d'adhérent et de partenaires santé de mutuelles.

Le Cimut est une organisation qui propose une prestation d'infogérance aux mutuelles. Il regroupe l'ensemble des ressources et moyens permettant de développer et exploiter le système d'information commun Starweb. Ce système doit permettre aux mutuelles adhérentes d'assurer la gestion des prestations santé auprès de leurs adhérents du régime indépendant, complémentaire ou étudiant.

Le Cimut est composé de plus de 100 ingénieurs et techniciens, dont environ 20 % de prestataires. Il y a actuellement 24 adhérents au Cimut, ce qui correspond à près de 2 millions de personnes protégées. Le Cimut est composé de plusieurs directions, dont la direction des applications transversales elle-même divisée en plusieurs services, dont le service IHM où j'ai effectué mon stage.

Actuellement les comptes extranet des mutuelles sont référencés dans un annuaire LDAP et leur administration se fait via une console externe disponible uniquement au Cimut, ce qui empêche les organismes de faire l'administration eux-même. Ainsi chaque demande de modification de comptes fait l'objet d'une FAE. Il y a plus de 10 demandes de modifications de compte extranet par jour, ce qui génère une charge de travail non négligeable. Le Cimut m'a donc confié la mission de créer une application qui sera mise à disposition des organismes, pour qu'ils puissent gérer eux-même leurs comptes extranet. Cette application va permettre aux gestionnaires des organismes d'effectuer une recherche de comptes extranet dans la base LDAP.

Il pourra ainsi voir les informations du comptes telles que l'INSEE, le nom, le code d'accès, la question secrète, etc. Puis il pourra réinitialiser le mot de passe du compte, ou encore modifier différents critères du compte (l'INSEE, la "dématérialisation" et la "dématérialisation en cours"). Enfin il aura également la possibilité de désactiver le compte. Une deuxième application a été créée dans le but de pouvoir consulter les modifications réalisées sur les comptes extranet.

Ces applications ont été développées avec Eclipse et principalement codées en Java, Javascript (dont JQuery) et HTML. J'ai réalisé plusieurs Web Services permettant de faire une recherche dans l'annuaire LDAP ou de faire une modification. Ces Web Services ont été développés en java à l'aide de Axis2 et d'Unboundid pour la communication avec l'annuaire LDAP. Par la suite j'ai créé une servlet pour faire le lien entre l'application et les Web Service. Lors d'une recherche ou d'une modification, le servlet fait un appel à un des Web Service, qui lui même fait appel à l'annuaire LDAP; enfin la servlet renvoie le résultat sous la forme d'un JSON. Par la suite, j'ai utilisé Html et JQuery pour faire l'application web en elle-même; en effet une fois le résultat récupéré, le JSON est traité, puis la réponse est affichée sur l'application. Cette application respecte la charte graphique des interfaces existantes au Cimut.

Ce stage s'est très bien déroulé, les applications sont fonctionnelles, toutes les exigences du Cimut et des utilisateurs ont été respectées et mises en oeuvre.

La mise en production sera réalisée mi-juillet, l'application n'est donc pas encore disponible auprès des organismes. Une fois que l'application sera mise en production, les organismes pourront réaliser l'administration des comptes extranet de leurs adhérents. Mon stage se poursuit sur un CDD de un mois et demi, ce qui me permettra entre autre, de suivre la mise en production.

Abstract

I've made this student's training course as part of my graduation in DUT IT specialities. I realised this training between april 7th and june 13th 2014 on CIMUT at Quimper, which is the informatic center of complementary health insurance.

At this time the extranet account of these companies are referenced in a directory LDAP. There administration is made by an external terminal usable only at CIMUT. The mutual cannot do the administration by herself. I join the GUI team for creating two web applications for the companies. Those applications will allow the administration of members account's and health partner.

Those applications were developed with Eclipse and mainly coded in Java, Javascript (including JQuerry) and HTML.

The first one allows the administration. It means that it gives the user the possibility to search an existing account. Then the user can modify this account, change the password or disable it. Every action will be recorded in an SQLite database.

The second one will allow the search of carried out actions on the account. The user, after a search, can access the list of the actions made by any administrator on any account.

Those applications work with a servlet and a Web Service. I've made Web Services for the search and the modification in the LDAP database. The servelet made the link between the Web Services and the application.

This student's training course went great. Both applications are functional and comply with all the demands in terms of specifications; However the deployment is not realised yet.

This professional experience drives me to a month and a half fixed-term contract in order to, among other things, deploy the applications.

Bibliographie

- Site web d'Apache, <http://www.apache.org/>
- Site web du W3C, <http://www.w3c.org/>
- Site web de Developpez.com, <http://www.developpez.com/>
- Site web de l'encyclopédie libre Wikipedia, <http://www.wikipedia.fr/>
- Site web d'eclipse, <http://www.eclipse.org/>
- Site web de guide pour les CMS, <http://www.guidecms.com/>
- Site web d'un développeur, <http://www.jmdoudoux.fr/>
- Les différents documents électronique et papier de présentation présent au CIMUT

Liste des termes Techniques

Appender : les flux qui vont recevoir les messages de log.

CGU acceptées : Conditions Générales d'Utilisation acceptées

CMS : Cela désigne des systèmes de publication et de mise à jour de sites Internet, ne requérant pas de connaissances spécifiques en programmation ou en HTML. D'une manière générale, les CMS permettent la création, le stockage et l'édition dynamique de contenu. Ils gèrent également les utilisateurs et leurs droits, l'indexation et la recherche, l'interface utilisateur. Les CMS s'adressent notamment aux utilisateurs non-spécialistes mais permettent également aux plus avertis de gagner du temps dans les phases de conception et de maintenance, et de garder une cohérence dans la présentation, notamment au niveau de la conservation de la charte graphique (séparation du contenu et du graphisme).

CMROC : Numéro d'identification des organismes. Ce numéro d'identification est interne au Cimut.

Dématérialisation : L'utilisateur opte pour la réception de ses relevés de décomptes santé par mail et ne les reçoit plus par papier.

Dématérialisation en cours : La dématérialisation demandée est en cours de traitement. Le traitement dure 48 heures.

Extranet : Un extranet (ou réseau interne étendu) est un réseau de télécommunications de type internet conçu pour faciliter les échanges entre une organisation sociale et ses correspondants extérieurs.

FAE : Fiche d'anomalie et d'évolutions, les gestionnaires remontent les anomalies au Cimut pour qu'elles soient traitées. Lors d'une demande de modification de compte par un adhérent, les organismes envoient une FAE au Cimut.

Infogérance : Phénomène qui consiste pour une entreprise à externaliser auprès d'un prestataire de services spécialisés, tout ou partie de la gestion de son système d'informations.

Inscription en cours : correspond à une valeur numérique.

0	Utilisateur inscrit sur la nouvelle plateforme
-1	Utilisateur en cours d'inscription, il n'y a pas eu de réponse au mail envoyé lors de l'inscription et le lien dans le mail est désactivé.
>0	Utilisateur en cours d'inscription, le lien dans le mail est toujours actif.
Autre	Utilisateur inscrit sur l'ancienne plateforme

INSEE : C'est le numéro de sécurité sociale, il est unique.

Log : Le logging consiste à ajouter des traitements dans les applications pour permettre l'émission et le stockage de messages, appelé log, suite à des événements.

Plugin : Un plugin, aussi nommé module d'extension, module externe ou add-on, est un paquet qui complète un logiciel hôte pour lui apporter de nouvelles fonctionnalités.

SOAP : ancien acronyme de Simple Object Access Protocol est un protocole de RPC orienté objet bâti sur XML. Il permet la transmission de messages entre objets distants, ce qui veut dire qu'il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur. Le transfert se fait le plus souvent à l'aide du protocole HTTP, mais peut également se faire par un autre protocole, comme SMTP.

Table des annexes

Annexe1:	CMS	de	la	mutuelle
Smeba.....			62	
Annexe	2	:		Environnement
Starweb.....			62	
Annexe	3:	Formulaire	pour	la
d'IHM.....			63	création
Annexe	4	:	IHM	AEXT
liste.....			63	mode
Annexe	5	:	IHM	AEXT
détail.....			64	mode
Annexe	6	:	IHM	AEXT
(modification).....			65	mode
Annexe	7	:	IHM	AEXT
mode			reinitialisation	du
passe.....				mot de
Annexe	8	:	IHM	HEXT
mode			liste	(avec
pagination).....				tri et
Annexe	9	:	IHM	NMAB
Contact).....				(onglet
Annexe	10	:	SNI	de l'application
(AEXT).....				d'administration
			68	

NB : Certaines données sont grises volontairement.

Annexe 1 : CMS de la mutuelle Smeba

The screenshot shows the Smeba CMS extranet interface. At the top, there's a banner with a profile picture of a young man and the text "Malin et complet, mon espace personnel me permet de bénéficier de tous mes services en ligne !". Below the banner, it says "Julien, 19 ans, étudiant". On the left, there's a sidebar with sections like "Mes remboursements", "Ma situation d'assuré", and "Mes messages". The main area has three tabs: "Mes derniers remboursements", "Mes messages", and "Mes messages". The "Mes derniers remboursements" tab displays a table of recent reimbursements:

Date	Fin des soins	Dépense engagée	Montant remboursé
27/09	22/09	22.00	14.40
16/07	12/07	22.00	14.40
05/02	04/02	45.79	29.55
09/12	07/12	22.00	1.90
26/08	25/08	22.00	14.40

Annexe 2 : Environnement Starweb

The screenshot shows the Starweb environment. On the left, there's a sidebar with "Mes services favoris" (highlighted with a red box), "Mes liens favoris", and "Ressources". The main area has a header with "Bienvenue gyclon9017" and "9017 - Smeba - localhos". It includes a "Changer d'organisme" button with a red arrow pointing to it, and a "Valider" button. Below the header, there are sections for "COMMUNICATION", "CODE GESTIONNAIRE, Cmroc, Organisme" (highlighted with a red box), "Les services", and "CONTACT". The "COMMUNICATION" section shows "Communication CIMUT" and "Communication interne", both with "Aucun nouveau message". The "CONTACT" section shows "Annuaire interne" and "Coordonnées organismes".

Annexe 3: Formulaire pour la création d'IHM

Nom (*)	AEXT	Libellé (*)	Administration des comptes extranets (AEXT)		
Code (*)	AEXT				
Date de début (*)	15042014	<input type="button" value=""/>	Date de fin (*)	31122099	<input type="button" value=""/>
Order by (*)	42	Url http (*)	/mesICIP/AEXTindex.htm		
Tuxedo (*)	SVCMESL	classe (*)	C		
CMSI max (*)	CMSID	Regime (*)	3		
Type application (*)	Nouvelle grille	<input type="button" value=""/>			
Code du service métier (*)	M-AEXT-AEXT				

Annexe 4 : IHM AEXT mode liste

Recherche					
INSEE	Code d'accès	Nom	Type de compte	Adhérent	
Liste des résultats de la recherche					
	INSEE	Code d'accès		Nom utilisateur	
189	9081	ae yyb		ROUSSE FRANCOIS	
283	1025	ae udp		ROUSSE AMELIE	
292	0317	ah edp		ROUSSE LEA	
285	4042	ac chc		ROUSSE AURELIE	
291	7655	ah pxa		ROUSSE ELODIE	
292	3213	au aex		ROUSSE JUSTINE	

Annexe 5 : IHM AEXT mode détail

Recherche
INSEE
Code d'accès
Nom
Type de compte
Adhérent

Détail du compte (Compte désactivé)

⚠ Modification effectuée aujourd'hui (Compte désactivé) par gyclon9017

Code d'accès	a[REDACTED]yb
INSEE	18[REDACTED]1
Nom Prénom	ROUSSE FRANCOIS
Adresse email	rc[REDACTED].fr
Question secrète	Q[REDACTED]n?
Réponse secrète	V[REDACTED]
CGU acceptée	Oui
Dématérialisation	Oui
Dématerrialisation en cours	Oui
Date d'inscription	02/07/2009 à 15:24:46
Date de dernière connexion	05/05/2014 à 17:27:31
Inscription en cours	Inscription finalisée
Populations	DEMATERIALISE, DEFAUT, RO

Retour liste
 Réinitialiser mot de passe
 Modifier
 Activer le compte

Historique

Date et heure	Code gestionnaire	Action
2014-06-02 16:32:37	gyclon9017	Compte désactivé
2014-06-02 09:55:09	gyclon9017	Modification du Mot de passe
2014-05-28 17:54:27	gyclon9017	Compte activé
2014-05-28 17:54:23	gyclon9017	Compte désactivé
2014-05-28 17:54:15	gyclon9017	Modification de "dematerialisation"
2014-05-28 17:54:09	gyclon9017	Modification de "dematerialisation"
2014-05-28 17:38:41	gyclon9017	Modification de "dematerialisation"
2014-05-28 17:38:32	gyclon9017	Modification de "dematerialisation"
2014-05-28 17:38:19	gyclon9017	Modification de "dematerialisation"
2014-05-28 17:38:06	gyclon9017	Modification de "dematerialisation"

Résultats 1 à 10 sur 630

Annexe 6 : IHM AEXT mode édition (modification)

The screenshot shows the 'Modification du compte' (Account Modification) screen. The user has selected a record with INSEE 18 081. The form fields include:

Champ	Valeur
INSEE	18 081
Nom Prénom	ROUSSE FRANCOIS
Adresse email	ro...@fr
Question secrète	C...en?
Réponse secrète	V...t
CGU acceptée	Oui
Dématérialisation	<input checked="" type="radio"/> Oui <input type="radio"/> Non
Dématérialisation en cours	<input checked="" type="radio"/> Oui <input type="radio"/> Non
Date d'inscription	02/07/2009 à 15:24:46
Date de dernière connexion	05/05/2014 à 17:27:31
Inscription en cours	Inscription finalisée
Populations	DEMATERIALISE, DEFAUT, RO

Buttons at the bottom: Valider (Validate) with a checkmark icon, Annuler (Cancel) with a cancel icon.

Annexe 7 : IHM AEXT mode réinitialisation du mot de passe

The screenshot shows the 'Modification du mot de passe' (Password Modification) screen. The user has selected a record with INSEE 18 081. The form fields include:

Champ	Valeur
Nouveau mot de passe	*****
Confirmer nouveau mot de passe	*****

A green checkmark icon is displayed next to the confirmation field, indicating the password match. Buttons at the bottom: Valider (Validate) with a checkmark icon, Annuler (Cancel) with a cancel icon.

Annexe 8 : IHM HEXT mode liste (avec tri et pagination)

Recherche

Code gestionnaire	Code d'accès	aecadyyb	Traçabilité du	27052014	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>
-------------------	--------------	----------	----------------	----------	---------------------------------	---------------------------------	---------------------------------

Historique

Date et heure	Code gestionnaire	Action	Code d'accès
2014-05-28 17:21:20	gyclon9017	Modification du Mot de passe	ae vb
2014-05-28 15:44:49	gyclon9017	Modification de "dematerialisation en cours"	ae vb
2014-05-28 15:44:48	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 15:44:42	gyclon9017	Modification de "dematerialisation en cours"	ae vb
2014-05-28 15:44:41	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:26:04	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:25:57	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:21:40	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:21:30	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:20:33	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:20:27	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:18:12	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 14:11:47	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 13:58:14	gyclon9017	Modification de "dematerialisation"	ae vb
2014-05-28 13:55:35	gyclon9017	Modification du Mot de passe	ae vb
2014-05-28 12:09:05	gyclon9017	Modification du Mot de passe	ae vb
2014-05-28 10:00:10	gyclon9017	Compte activé	ae vb
2014-05-28 09:59:41	gyclon9017	Compte désactivé	ae vb
2014-05-27 17:53:28	gyclon9017	Modification du Mot de passe	ae vb
2014-05-27 17:52:25	gyclon9017	Modification de "dematerialisation"	ae vb

Résultats 1 à 20 sur 38
 1 2

Annexe 9 : IHM NMAB (onglet Contact)

Recherche

Assuré 18 81 Action Interrogation ▾

Synthèse | Courrier | Notes | Clé insee | Appel RC | Ville | Saisie accompte | RNIAM | Mutation | Alertes | Affiliation

Assuré							
Insee				Organisme de Tutelle			
Nom et Prénom	M ROUSSE FRANCOIS			Centre de Gestion			
Adresse	M ROUSSE FRANCOIS			Etablissement			
Tél. Fixe / Portable				Droit / CMU			

Bénéficiaires

Rang	Adr.	Dest. Paiement	Date naissance	Jum.	Sexe	Nom	Prénom	RO Régime Obligatoire				RC Régime Complémentaire			
								Ouverture	Fermeture	Motif Rad	Motif Ferm	RNIAM	Médecin Traitant	Ouverture	Fermeture
01	✓	✓													

Détail Dossier	Changement INSEE	Adresses Assurés	Contact	Remarques	Détail Bénéficiaire	Modalités de paiement	Oppositions	Décomptes	Accord Médical	Archivage	Documents
Prestations Plafonnées	Certificat Médical	Lien Etablissement	Demandes Assurés	Echéanciers	Calendrier Paiement	Compte Cotisant	Vitale	Mouvements RNCPS			
Produits Bénéficiaire	Participation (base dettes)	Aides	Règles Cotis. Bénéficiaire	Majorations Minorations	Renouvellement Rég. Etudiant	Droits RC	Devis / PEC	Renouvellement RC			
Participation		Mouvements Noémie RC	Caisses Noémie RC	Organisme Tutelle	Echéancier RC	Dossiers Contentieux	Prestations RO Plafonnées				

TELEPHONE

N° Fixe	N° Portable	N° Fax	N° Hors Métropole	N° SMS / Envoi autorisé
Personnel				
Professionnel				-

E-MAIL

E-mail Personnel / Envoi autorisé / Certification	E-mail Professionnel / Envoi autorisé
<input type="checkbox"/> Adresse e-mail non certifiée	

Code d'Accès Extranet/Inscription/Acception/CGU validées le Lien vers l'application d'administration

Dématérialisation Relevés

Dématérialisation Com. avec l'organisme

a_yb Lien vers l'application d'administration

Modifier

Annexe 9 : SNI de l'application d'administration (AEXT)

