Conception et réalisation d'un système de gestion des habilitations pour Apogée : APOBILITATION

Marie-Christine Duffour

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

PARIS

_____

MEMOIRE

présenté en vue d'obtenir

le DIPLOME d'INGENIEUR CNAM

SPECIALITE : informatique

OPTION : réseaux, systèmes et multimédia (IRSM)

par

DUFFOUR COLLAS Marie-Christine

_____

Conception et réalisation d'un système de gestion des habilitations pour Apogée : APOBILITATION

Soutenu le 30 septembre 2013

_____

JURY

PRESIDENT : Elisabeth METAIS

MEMBRES: Luc-Henri BARTHELEMY
Eric CHEREL
Eric GRESSIER-SOUDAN
Stéphane ROVEDAKIS
Françoise SAILHAN

Remerciements

Mes remerciements vont en tout premier lieu à M. François Cadé, directeur du département de l'Informatique et des Systèmes d'Information au moment où le projet a démarré, qui m'a autorisée à réaliser ce projet dans le cadre de mon travail à l'Université Paris-Descartes, à M. Luc-Henri Barthélémy directeur du département des études recherches et développement ,et à Mme Corinne Claisse, responsable du service d'aide aux utilisateurs de vie étudiante et scolaire qui a été l'instigatrice de ce projet et mon supérieur hiérarchique. De nombreux collègues m'ont apporté leur contribution à des degrés divers, sur les plans technique, fonctionnel, historique, ainsi que de nombreux encouragements.

Je remercie également les différents enseignants rencontrés tout au long du cursus menant au diplôme d'ingénieur. Tous leurs enseignements ont contribué à me hisser à un niveau de connaissances indispensables pour me permettre de reconstruire ma carrière d'informaticienne. Parmi eux M Eric Gressier-Soudan a été mon professeur dans trois unités d'enseignement de l'option Ingénierie Réseaux Systèmes et Multimédias, et mon tuteur dans la rédaction de ce mémoire. Il a été lui aussi une source d'encouragement importante. Je remercie également Mme Elisabeth Métais, son homologue dans le domaine des systèmes d'informations qui a bien voulu revoir l'ensemble sous un angle différent.

Ce mémoire, aboutissement d'un long parcours n'aurait pu se faire sans la compréhension et le soutien de mon mari et mes fils, et de mes amis, qui ont dû supporter mes absences en soirée et une part non négligeable de jours de vacances consacrés à projets et révisions d'examen. Je leur suis extrêmement redevable. J'ai également une pensée émue pour mon défunt ingénieur de père, qui m'a donné depuis longtemps ce rêve de devenir ingénieur à mon tour.

Je remercie également Edwige Bombaron, dont le travail de thèse porte sur l'écriture, et dont l'écoute a été bienfaisante.

Abréviations

AMUE	Agence de Mutualisation des Universités et des Etablissements d'enseignement supérieur
API	Application Programming Interface
APOGEE	Application Pour l'Organisation et la Gestion des Etudes et des Examens. APOGEE est un logiciel national édité par l'AMUE.
CAS	Central Authentication Service
CGE	Centre de Gestion. Ici à Paris Descartes (anciennement Paris V), c'est UP5.
CIP	Centre d'Inscription Pédagogique
CSS	Cascading Style Sheet ou feuille de style
CTN	Centre de Traitement
CXF	Celtic XFire (l'acronyme CXF vient de la fusion des deux projets Celtic et XFire)
DAO	Data Access Object.
DBA	Data Base Administrator (Administrateur de base de données)
DEVU	Direction des Etudes et de la Vie Universitaire
DGS	Direction Générale des Services
DISI	Direction de l'Informatique et des Systèmes d'Information
ENT	Environnement Numérique de travail, portail de l'intranet.
GRHUM	Base de données du référentiel (Gestion des Ressources HUmaines et Morales)
HARPEGE	HARmonisation de la GEstion des PERsonnels HARPEGE est un logiciel national élaboré par l'AMUE pour la gestion des ressources humaines
HTML	Hyper Text Markup Language
Java / JEE	Java Entreprise Edition
JDBC	Java DataBase Connectivity
JSF	Java Server Faces

LDAP	Lightweight Directory Access Protocol (service d'annuaire)
OIDDAS	Oracle Internet Directory Delegated Administration Services Service d'annuaire LDAP géré par l'application web Oracle Identity Management
PHP	Hypertext Pre Processor. Langage de script pour le web.
PRES	Pôle de Recherche et d'Enseignement Supérieur : regroupement d'établissements d'enseignement supérieurs et de recherche.
SAUVES	Service d'Aide aux Utilisateurs de la Vie Etudiante et Scolaire
SQL	Structured Query Language
SSO	Single Sign On (authentification unique)
SUPANN	Recommandation en matière d'**ann**uaire pour l'enseignement **sup**érieur
UFR	Unité de Formation et de Recherche
UID	Abréviation pour User IDentifier
XML	eXtensible Markup Language, méta langage.

Glossaire

APOBILITATION	Système de gestion des Ha**bilitation**s sur **Apo**gée : nom donné à cette application.
Bean	Composant Java réutilisable. Ou Java Bean.
Central Authentication Service (CAS)	Logiciel Open Source utilisé dans la plupart des établissements pour mettre en œuvre un système d'authentification à mot de passe unique (SSO).
Composante	Terme désignant une des parties de l'université qui peut être une faculté ou une UFR ou un IUT.
Cron	Dérivé du grec kronos (le temps), cron est le nom d'un programme permettant de planifier le lancement de tâches à heure fixe. Ce programme est utilisé dans Quartz d'OpenSymphony.
DBLink	Lien entre bases de données : un DBLink est un objet d'une base de données permettant d'exécuter des requêtes sur une autre base de données, qu'elle se trouve physiquement sur la même machine ou qu'elle soit distante.
Eclipse	Atelier de Génie Logiciel très répandu chez les développeurs en java
Esup-commons	Framework de développement du consortium Esup portail
Esup-Portail	Consortium d'universités
Framework	En français socle d'applications, cadriciel, cadre de travail. Cet outil de développement impose au programmeur par construction le respect de certains patterns (canevas).
Hibernate	Outil d'accès aux bases de données concurrent de JDBC. Hibernate possède son propre langage HDL, concurrent du SQL. Hibernate a été intégré à Java sous le nom JPA.
Jasig	Consortium d'institution d'enseignement pour promouvoir des projets open source pour l'enseignement supérieur.
Java	Langage de programmation
Java JEE	Java adapté aux applications sur le web
JPA	Java Persistance API : intégration des API d'Hibernate dans Java à partir de java 1.6.
JSF	Java Server Faces connu aussi sous le nom d'Apache My Faces. Cet

	outil succède aux JSP et à Struts pour créer des pages web compatibles html et java.
JUnit	Framework de tests unitaires pour la programmation en java
LDAP	Annuaire
Maven	Logiciel libre pour la gestion et l'automatisation des projets logiciels java
OpenSymphony	Editeur de Quartz, logiciel open source pour la planification des tâches en java, malheureusement disparu en 2010.
ORACLE	Système de gestion de base de données relationnel
Spring	Outil de conteneur léger d'ejb
Tomcat	Serveur d'applications web distribué par Apache.

Table des matières

TABLE DES MATIERES

Chapitre 1. Introduction

1.1. PROBLEMATIQUE

L'Université Paris Descartes utilise pour gérer l'ensemble des traitements liés aux métiers de la scolarité le logiciel de gestion intégrée APOGEE (Application Pour l'Organisation et la Gestion des Etudes et des Examens). Ce logiciel, qui est installé dans de nombreuses universités, existait d'abord en tant qu'application client-serveur isolée. Puis il a figuré au sein d'un système d'information avec référentiel. Il est enfin accessible par le web par l'ensemble des utilisateurs. Aujourd'hui Apogée arrive à un niveau d'entropie nécessitant la recherche d'une solution pour automatiser le processus d'habilitation des utilisateurs d'Apogée et maintenir la cohérence dans le système d'information. La solution proposée d'une application de gestion des habilitations pour Apogée est donc la réponse qui permet de simplifier le processus des enregistrements multiples sur différents systèmes. Le nom choisi pour cette application est APOBILITATION, contraction d'**APO**gée et d'ha**BILITATION**.

1.2. MON ROLE

En septembre 2010, en tant qu'ingénieur de développement attachée au Service d'Aide aux Utilisateurs de la Vie Etudiante et Scolaire (SAUVES) de la Direction de l'Informatique et des Systèmes d'Information (DISI) de l'université Paris-Descartes, sous les ordres de Corinne CLAISSE , responsable du service SAUVES, j'ai été chargée de la conception, de la réalisation (développement, tests, validation, déploiement), et de la formation des utilisateurs finaux d'un système permettant d'automatiser des tâches manuelles auparavant effectuées par le service SAUVES à la demande des responsables de scolarités de l'université Paris Descartes, afin de permettre aux responsables de scolarité d'avoir l'initiative et la saisie des demandes d'habilitations à utiliser ou à ne plus utiliser l'application APOGEE. Cette application doit être accessible sur le web, par les membres des personnels autorisés après qu'ils se soient identifiés sur le portail de l'université.

1.3. IMPORTANCE DU SUJET

1.3.1. Un projet d'automatisation de processus et de gestion d'identité

Le projet APOBILITATION pose la question de la gestion des identités lorsqu'au fil du temps un même individu se trouve avoir plusieurs identités pour accéder à différents services et que le système peine à faire le lien entre les identités appartenant au même individu. Les solutions apportées dans le passé ont multiplié les opérations de saisie et l'on se retrouve avec tout un enchaînement de processus d'identification à jouer pour habiliter une nouvelle personne, avec le risque d'oublier des opérations au moment de lui retirer son habilitation.

Puisqu'on parle de processus, le projet va prendre en charge l'automatisation de ce processus.

1.3.2. Un projet permettant de réaliser des économies

Une personne de l'équipe passe une partie de son temps à réaliser des tâches de saisie à la demande de l'utilisateur. L'automatisation de ces tâches lui permettra de se consacrer à d'autres activités, tout en donnant des outils de contrôle à l'utilisateur.

Par ailleurs le contrôle de cohérence des informations à l'intérieur d'Apogée ainsi qu'entre Apogée et d'autres systèmes est fastidieux, à la portée uniquement des informaticiens maitrisant le SQL, si bien que les anomalies sont décelées souvent par hasard, ou après réclamation d'un usager. Il manque d'un outil de gestion. Cette absence d'outil fait que le contrôle sur les utilisateurs qui ne devraient plus être habilités n'est pas fait. Ces utilisateurs non désactivés alourdissent les tables et rendent les requêtes en base de données plus longues. Avec cette application le travail de l'administrateur de la base de données s'en trouvera simplifié.

1.3.3. Un projet complexe

Le projet est complexe car il s'insère au-dessus d'applications existantes, elles-mêmes assez complexes. Deux bases de données et deux annuaires LDAP sont impliqués. Certaines jointures sont inexistantes ou difficiles. Une des bases nécessiterait un bon nettoyage. Pour savoir comment sont réalisées les opérations manuelles, il faut interviewer ceux qui les font, et certaines règles de gestion ne se trouvent que dans le cerveau d'une seule personne. La partie conception demande un investissement réel.

1.3.4. Un projet à faible priorité

Faible priorité parce que les utilisateurs qui sont les responsables de scolarité sont peu intéressés de savoir comment le service SAUVES habilite les personnels à utiliser APOGEE, alors que le service SAUVES est fortement intéressé de ne plus avoir à faire ces opérations d'enregistrement qui lui prennent du temps au détriment d'activités plus informatiques. Mais le service SAUVES doit faire passer en priorité d'autres projets dont le besoin semble plus urgent aux scolarités. Cette faible priorité a eu pour conséquences que le projet a dû être mis en attente plusieurs fois puis repris pour enfin arriver à son terme en octobre 2012. Il totalise à ce jour plus de 210 jours répartis sur 2 ans.

1.3.5. Un projet pour un public réduit

Le nombre d'utilisateurs de l'application devrait se stabiliser autour de 25 à 30 personnes en moyenne. Toutes ces personnes ont de grandes responsabilités dans l'université, et un pouvoir de décision significatif.

1.3.6. Un projet en autonomie

J'ai fait seule l'essentiel de la conception et de la réalisation, avec une grande autonomie, sous la supervision de Mme Corinne Claisse.

1.3.7. Un projet pour le web en Java

La partie réalisation a été faite à l'aide du framework[1] Esup-Commons V2, avec les technologies java, JSF, Spring, JPA et le déploiement s'est fait sur serveurs Tomcat. Le projet a bénéficié de l'expérience des autres projets utilisant les mêmes technologies.

1.3.8. Contenu du mémoire

Dans les pages qui vont suivre nous verrons tout d'abord dans quel contexte ce projet s'est déroulé puis l'analyse du besoin qui a été exprimé (Chapitre 2). La suite expose la façon dont j'ai interprété le besoin pour proposer et concevoir une solution (Chapitre 3), poursuivre l'analyse (Chapitre 4) et mettre en œuvre la réalisation (Chapitre 5 et Chapitre 6) jusqu'à son achèvement (Chapitre 7).

Nous conclurons par un bilan de l'opération (Chapitre 8).

[1] Framework : on a proposé comme traduction française socle d'applications, cadre de travail, cadriciel. Comme aucune de ces traductions ne me semble satisfaisante, je conserve le mot « framework » tout-au-long de ce mémoire, le lecteur me pardonnera.

Chapitre 2. Le besoin ressenti à l'Université Paris Descartes

2.1. PRESENTATION DU CONTEXTE DE L'UNIVERSITE

2.1.1. Introduction

Depuis mai 2010, je travaille au sein de l'Université Paris-Descartes, à la Direction Informatique et des Systèmes d'information, en qualité d'ingénieur de développement contractuel. Ce chapitre décrit l'environnement dans lequel s'inscrit mon travail et mon rôle dans le projet. Le projet décrit ici est le deuxième sur lequel je suis intervenue depuis mon arrivée.

2.1.2. L'université Paris Descartes

L'université Paris-Descartes, ou Paris V, a son siège administratif 12 rue de l'école de Médecine à Paris 6e. Elle regroupe plusieurs « composantes » réparties sur plusieurs sites qui sont :

- Unité de Formation et de Recherche (UFR) biomédicale des Saints Pères
- Faculté de chirurgie dentaire
- Faculté de droit
- Institut Universitaire de Technologie (IUT)
- UFR de mathématiques et informatique
- Faculté de médecine
- Faculté des sciences pharmaceutiques et biologiques de Paris

- Institut de psychologie
- Faculté des sciences humaines et sociales
- UFR de Sciences et Techniques des Activités Physiques et Sportives de Paris (STAPS)

L'Université fait partie du Pôle de Recherche et d'Enseignement Supérieur (PRES) Sorbonne Paris-Cité.

L'Université a pour ambition de se classer parmi les premières de France et sur le plan international, et pour cela vise l'excellence dans tous les domaines.

L'Université est dirigée par un président élu (M. Axel Kahn, puis M. Frédéric Dardel depuis 2012) assisté par le Directeur Général des Services (M. François Paquis).

2.1.3. La Direction Informatique et des Systèmes d'Information

2.1.3.1. L'organigramme de la DISI

La Direction Informatique et des Systèmes d'information (DISI) est directement rattachée au président et au directeur général des services. Elle a été dirigée par M. François Cadé, puis par M. Eric Chérel depuis septembre 2012.

Elle est composée de 4 départements qui sont :

- le département des moyens informatiques
- le département TICE
- le département EDRE (Etudes et Développement)
- le département informatique de proximité

Chacun de ces départements comprend plusieurs services (Figure 1 Organigramme de la DISI (septembre 2011)). Pendant le projet, j'ai été amenée à avoir de nombreux contacts avec le service RESYST (réseaux et systèmes) du département des moyens informatiques, particulièrement au moment du déploiement et de la mise en production.

Une fois par trimestre une réunion de toute la DISI permet de connaître les grandes orientations prises par la direction et de savoir sur quels projets travaillent les collègues. Cette réunion permet de mieux connaître le système d'information dans lequel nous évoluons.

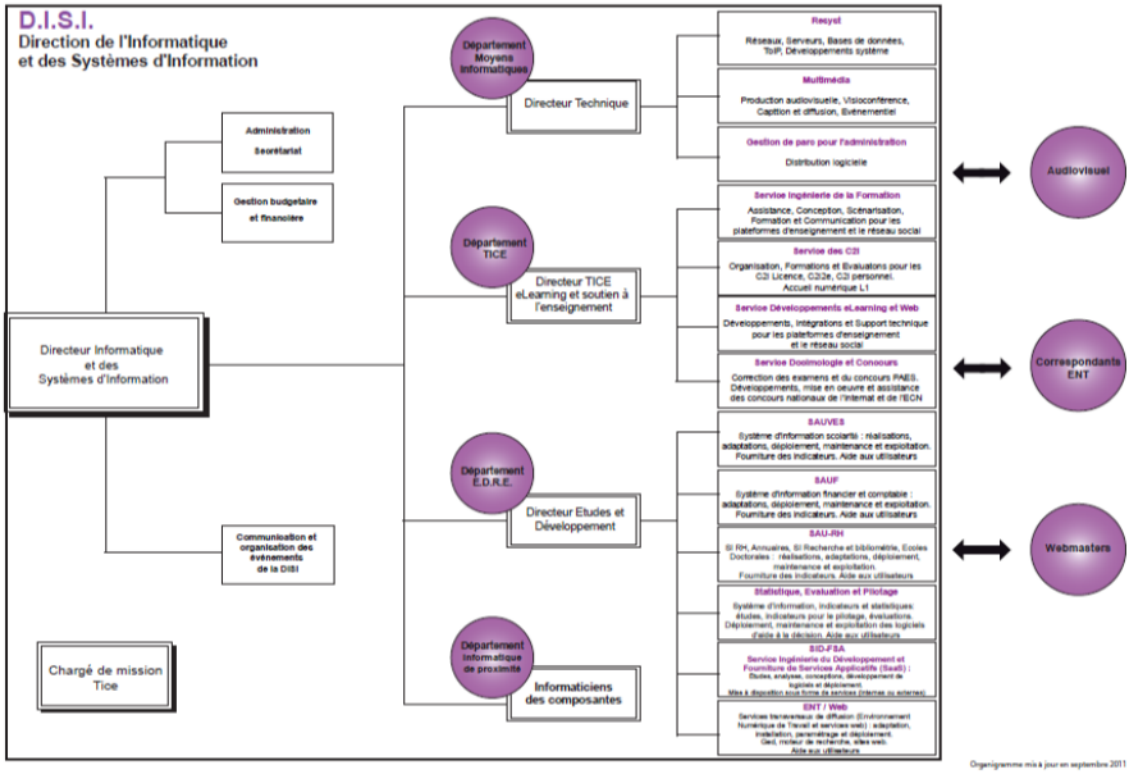


Figure 1 Organigramme de la DISI (septembre 2011)
Source : Université Paris-Descartes

On trouvera en annexe 1 une version en plus grand format de cet organigramme (Figure **91** Organigramme de la DISI).

2.1.3.2. Le département EDRE

Le département EDRE, dirigé par M. Luc-Henri Barthélémy est le plus important en termes d'effectifs. Il est chargé du développement d'applications, ou de l'intégration d'applications développées par des organismes privés ou publics.

Le service DEV (développement), rebaptisé SID-FSA (Service Ingénierie du Développement et Fourniture de Services Applicatifs (SaaS)) en raison de la nouvelle orientation vers le Cloud Computing, est dédié au développement. En tant que développeurs, nous avons des relations d'entraide et de capitalisation d'expérience.

Les services SAU-xx (Service d'Aide aux Utilisateurs- de la Vie Etudiante et Scolaire/ des Relations Humaines/ des Finances) sont affectés à un service utilisateurs particulier dont il connaît bien le métier.

Le service ENT / Web est en charge du site web et du portail ENT (Environnement Numérique de Travail) de l'université. Les applications que nous offrons aux utilisateurs ne doivent pas

être accessibles directement, mais après que ceux-ci se soient identifiés à travers le portail ENT[2] .

2.1.4. Le service SAUVES : service d'aide aux utilisateurs de la vie étudiante et scolaire

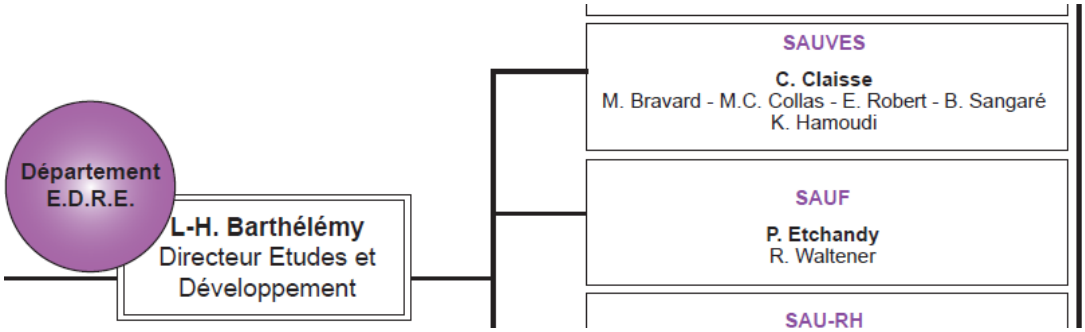


Figure 2 Extrait de l'organigramme de la DISI
Source : Université Paris-Descartes

2.1.4.1. Le fonctionnement du service SAUVES et ses attributions

Le service SAUVES (Service d'Aide aux Utilisateurs de la Vie Etudiante et Scolaire) est consacré à l'informatisation des services de scolarité. Il est dirigé par Mme Corinne Claisse.

Le service se charge de l'intégration d'applications achetées et du développement d'applications spécifiques.

La principale de ces applications est **APOGEE** (Application Pour l'Organisation et la Gestion des Etudes et des Examens). Avant d'être sous le contrôle du service SAUVES, l'application était sous celui de la « Cellule APOGEE ». C'est dire l'importance de cette application. Ce logiciel est développé et largement distribué dans les Universités françaises par l'AMUE (Agence de Mutualisation des Universités et des Etablissements d'enseignement supérieur)[3]. Régulièrement des patchs doivent être passés pour la mise à jour de ce logiciel. La base de données d'APOGEE s'appelle PRP5. Il existe une base de test nommée APOTEST. Le service a la responsabilité de l'administration de la base de données. Il a aussi la responsabilité d'organiser des formations pour les utilisateurs.

Plusieurs applications web sont greffées sur Apogée. Certaines sont développées et distribuées par l'AMUE, comme divers services web (récupération de données en lecture et

[2] Le système d'authentification *Single Sign On* sera décrit plus loin, dans la partie technique.
[3] www.amue.fr

plus rarement en mise à jour), APOWEB (réinscriptions administratives par le web), IA-Primo (inscriptions administratives des primo-entrants par le web), (IP-Web (inscriptions pédagogiques par le web), SNW (saisie de notes par le web), etc... D'autres sont développées par d'autres universités au travers du consortium ESUP (comme Esup-rendezvous développée par l'université de Montpellier, et modifiée dans le service SAUVES). Ou comme ARIA, interface de pré-candidatures en ligne développée par l'Université de Strasbourg en PHP. Ou comme AMETHIS (Accès Multi-Etablissements aux THèses, à l'International et au Suivi des doctorants/docteurs), pour la gestion des thèses, des doctorants et écoles doctorales, développée par l'Université Européenne de Bretagne (UEB). D'autres sont développées sur place (comme « IEJ-concours », « IEJ-gestionconcours », « oral-exam »).

En dehors de l'univers d'Apogée, le service est responsable des applications :

- ADE, distribuée par ADE Soft, pour la gestion des emplois du temps, la réservation de salles.
- ARIA, pour la gestion des pré-candidatures.

L'intégration des logiciels, l'administration des bases, la formation des utilisateurs et le support de ces derniers constitue l'essentiel des activités du service, le développement est une activité plus récente.

Tout projet, toute activité du service est soumise à l'approbation du directeur de la DISI. Le coût des projets est imputé soit à la DEVU, soit à une scolarité.

2.1.4.2. La DEVU client privilégié du service SAUVES

Le service SAUVES a vocation à travailler avec la DEVU (Direction des Etudes et de la Vie Universitaire). Cette direction est directement rattachée à la Direction Générale des Services.

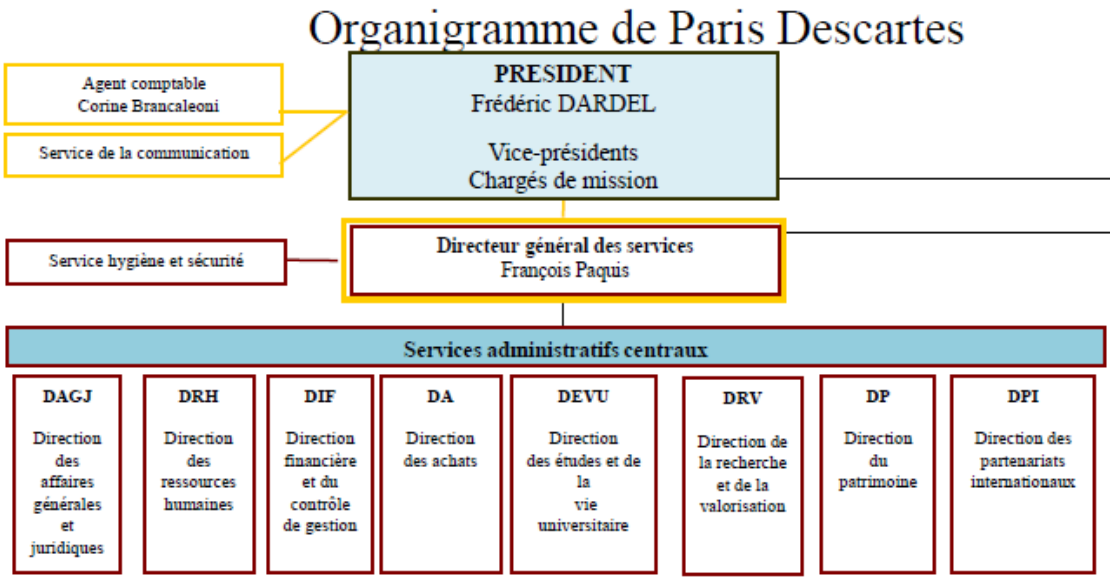


Figure 3 Extrait de l'organigramme général
Source : Université Paris-Descartes, septembre 2011

La DEVU est dirigée par M. Régis Lechenault, directeur, secondé par Mme Mariel Fayemi, chef de la scolarité centrale, puis par Mme Sophie Fernandez qui l'a remplacée. La DEVU exerce un contrôle sur les scolarités des différentes composantes de l'université. Chaque composante a un responsable de scolarité qui dirige le service de scolarité de la composante.

L'université essaie d'imposer un fonctionnement commun à toutes les scolarités, mais elles ont une certaine liberté d'adaptation pour gérer leurs particularités.

2.1.5. Les personnes qui ont participé au projet

2.1.5.1. De l'expression du besoin à la prise de décision

- Mme Corinne Claisse (DISI/EDRE/SAUVES) a vu le besoin, l'a exprimé et a rédigé le cahier des charges.
- M. François Cadé (direction DISI) et M. Luc-Henri Barthélémy (DISI/EDRE) l'ont approuvé et donné le feu vert.
- M Régis Lechenault et Mme Mariel Fayemi (DEVU) l'ont accepté.

2.1.5.2. Pendant la conception

Plusieurs personnes m'ont aidée à faire l'étude de l'existant :

- Mme Fayemi (DEVU) pour les processus à travers les scolarités,
- Mme Claisse (SAUVES) pour sa connaissance d'Apogée et des processus manuels, et les précisions sur le cahier des charges,

- Mme Ghyslaine Coquillette, et M. Emmanuel Robert (SAUVES) qui a pris sa suite, pour leur connaissance des opérations d'enregistrement existantes,
- Mme Monique Lagarde (DISI/EDRE/SAURH) pour sa connaissance de l'annuaire et de GRHUM,
- M. Richard Vatré (DISI/RESYST), pour sa connaissance des autres systèmes d'information (OIDDAS) et l'implantation sur les serveurs.

2.1.5.3. Pendant la réalisation et le déploiement

Pendant la réalisation, j'ai reçu l'aide de :

- la société Orsys, pour sa formation sur Spring,
- les membres participant au projet Esup-commons, pour un support permanent via le site, le forum et une formation sur Esup-commons v2,
- les autres développeurs du service DISI/EDRE/DEV, pour leur aide occasionnelle sur les subtilités du développement,
- M. Yves Gerday (DISI/RESYST) qui gère le réseau et les serveurs de l'université et les adresses des URL,
- Mme Claisse (SAUVES) pour son aide en tant qu'administrateur de base de données (DBA) sur Apogée,
- Mme Claisse et M. Robert (SAUVES) pour leur participation à l'étape de validation,
- Mme Dominique Houdet-Joly (DISI/EDRE/ENT) qui gère les accès aux applications via le portail de l'ENT.

2.1.5.4. Après la mise en service

Une fois l'application mise en service, ont participé au projet :

- M. Lechenault et Mme Fernandez (DEVU) pour la recette de l'application, et le paramétrage des tables,
- M. Robert (DISI/EDRE/SAUVES) pour l'aide au paramétrage qu'il leur a apporté et son aide à l'élaboration de la formation des utilisateurs.

2.1.6. Mon rôle dans le projet

2.1.6.1. L'autonomie

Dans cette application, j'ai joué un rôle d'architecte concepteur développeur, conduisant le projet de bout en bout avec les aides mentionnées ci-dessus, mais seule dans la réalisation et la rédaction des documents.

2.1.6.2. Le partage du temps

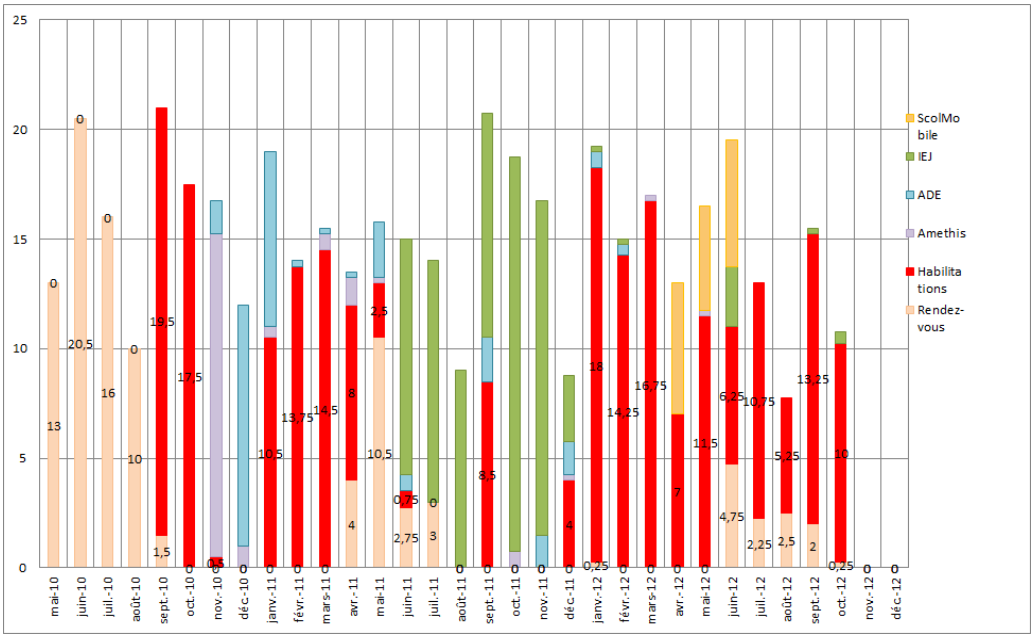


Figure 4 Partage du temps entre projets
Source : M. C. Collas

Le projet figure en rouge sur le graphique ci-dessus (voir l'annexe 2 en grand format). Il montre comment le temps s'est réparti : de septembre 2010 à octobre 2012, il représente environ 10 mois de travail. Plusieurs fois il a dû être interrompu pour céder la priorité à :

- Amethis (intégration d'un système de gestion de thèses réalisé par l'Union des Etablissements de Bretagne), ou
- ADE (intégration du logiciel d'emploi du temps réalisé par ADE Soft), ou
- IEJ (développements pour la scolarité de la Faculté de droit et la préparation des concours à l'Institut d'Etudes judiciaires)
- Rendez-vous (maintenance d'esup-rendezvous, bâti sur esup-commons, créé par l'université de Montpellier et modifié par mes soins, et support aux utilisateurs en période d'inscriptions)
- ScolMobile (étude préalable en vue d'une application mobile sur iPhone et Androïd)

Ce partage du temps impose un rythme différent de celui que l'on pourrait observer sur une plate-forme de développement dédiée à un seul logiciel. Il a permis que des compétences acquises sur un projet bénéficient à un autre. Ainsi le projet Habilitation a bénéficié de l'expérience acquise sur rendez-vous, Amethis et IEJ.

2.1.6.3. Les compétences mises en œuvre

En ce qui concerne la conception et l'analyse, les principaux outils sont UML et MERISE.

Le développement s'est fait en java sur l'atelier Eclipse. Il nécessite la connaissance du framework[4] esup-commons, les API de Spring, JSF, JPA (Hibernate), JDBC, CXF. L'application s'exécute sur serveur Tomcat. Les serveurs d'exploitation sont sur Linux. Le développement s'est fait sur une machine Apple (MacBook Pro).

Ces compétences sont celles sur lesquelles j'ai été recrutée, l'équipe SAUVES ne disposant pas de développeur java /jee auparavant.

2.1.7. Conclusion

Dans une telle organisation, prendre connaissance de la culture générale de l'université et de son système d'information prend du temps, et se fait au sein des projets.

Le projet APOBILITATION m'a amenée à faire des enquêtes au sein des services décrits dans ce chapitre pour récolter les informations métier, fonctionnelles et techniques nécessaires pour intégrer la future application au sein des architectures techniques et fonctionnelles existantes.

2.2. LE BESOIN RESSENTI

Cette section présente le besoin d'automatiser le processus d'habilitations, ce besoin étant ressenti essentiellement par l'équipe des informaticiens. Je m'inspire des schémas de M. Volle pour décrire l'évolution (Volle, 2002).

[4] Framework : voir §1.3.7 et la note

2.2.1. Un besoin ressenti dans le service SAUVES

2.2.1.1. Augmentation de la complexité et des niveaux d'imbrication

Comme tout logiciel, le logiciel Apogée gère une liste d'utilisateurs habilités à utiliser l'application. L'administrateur de la base de données Apogée, investi des droits maximum, est celui qui inscrit ces utilisateurs dans la liste.

Au départ Apogée était utilisé comme un logiciel de type client-serveur. Les personnels utilisateurs s'authentifiaient avec l'identifiant Apogée qui se trouve être le code utilisateur de la base de données Apogée sur Oracle. A cette époque-là (1997) les applications étaient organisées en parallèle, chacune avec leurs propres identifiants. Apogée gère les étudiants, Harpège gère le personnel. L'administrateur attribuait à chaque gestionnaire d'Apogée un compte Oracle qui était son code utilisateur d'Apogée et un mot de passe (Figure 5 Période client-serveur 1997).

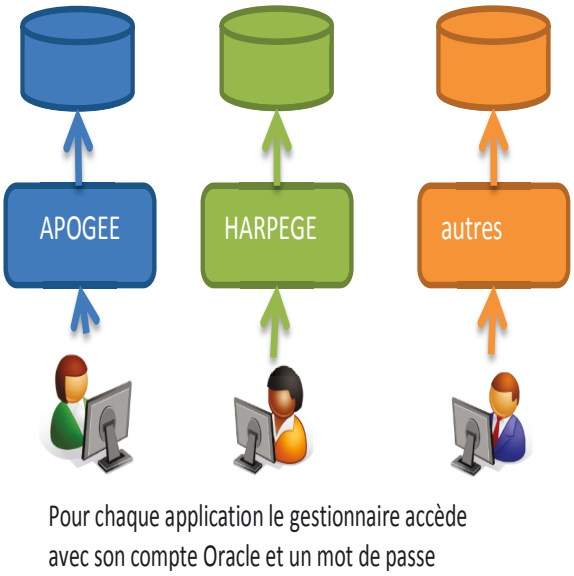


Figure 5 Période client-serveur 1997
Source : M.C. Collas

Puis on ajouta un premier annuaire Ldap attribuant un login et un mot de passe à tous les personnels connus par Harpège et tous les étudiants connus d'Apogée (Figure 6 Ajout de l'annuaire LDAP (2000)). L'annuaire fait le lien entre le login et le code étudiant Apogée ou le code personnel Harpège, mais ne connaît pas les codes des gestionnaires.

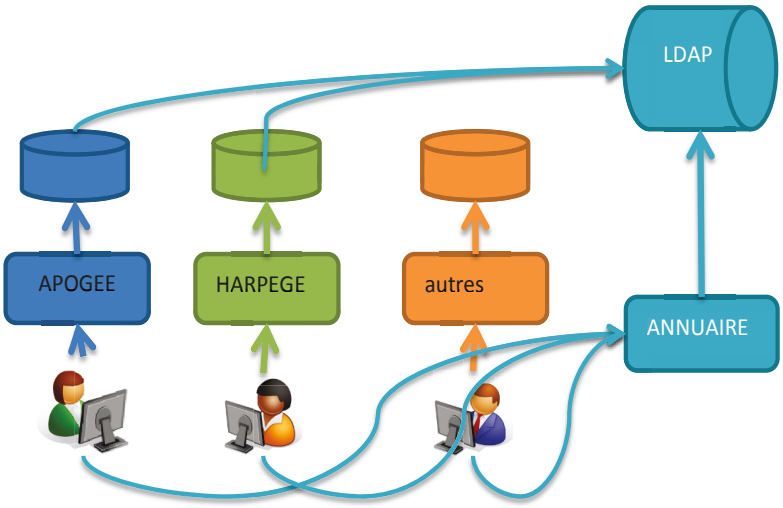


Figure 6 Ajout de l'annuaire LDAP (2000)
Source : M.C. Collas

Puis vint le moment de l'intranet (Figure 7 Ajout du SSO, du portail et d'une nouvelle version d'Apogée (2002)). La direction a souhaité que toutes les applications soient portées sur le web. On a ajouté un système SSO (Single Sign On) afin que chacun ne s'identifie qu'une seule fois en arrivant sur le portail (Voir §3.4.5 pour plus de détails). Le SSO consulte l'annuaire pour vérifier le login de celui qui se connecte.

L'AMUE a réalisé un programme d'interface permettant d'accéder à Apogée sur le web. Un annuaire spécial (OIDDAS) permet de faire la connexion entre le login du portail et l'identifiant et mot de passe d'Apogée. L'administrateur qui crée le gestionnaire dans Apogée doit aussi vérifier dans l'annuaire OIDDAS qu'il a bien un login connu dans l'annuaire, puis il raccorde à ce login le compte Oracle et le mot de passe de la base Apogée et de la base de test Apotest. Il fait de même pour les gestionnaires d'Harpège ou d'autres applications de la même génération. (Remarque : les applications ultérieures utiliseront le login de l'annuaire pour identifier leurs utilisateurs, se passant de l'intermédiaire d'OIDDAS.)

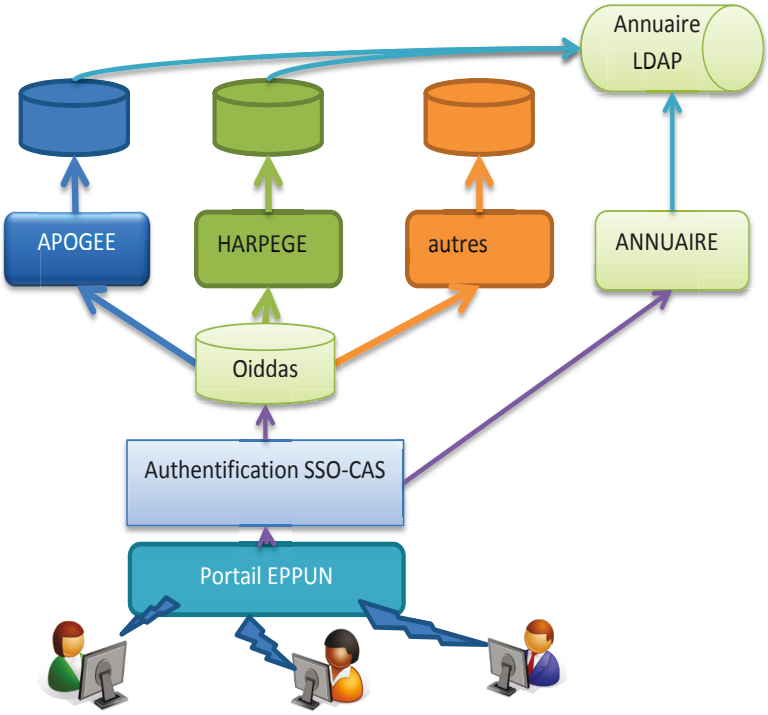
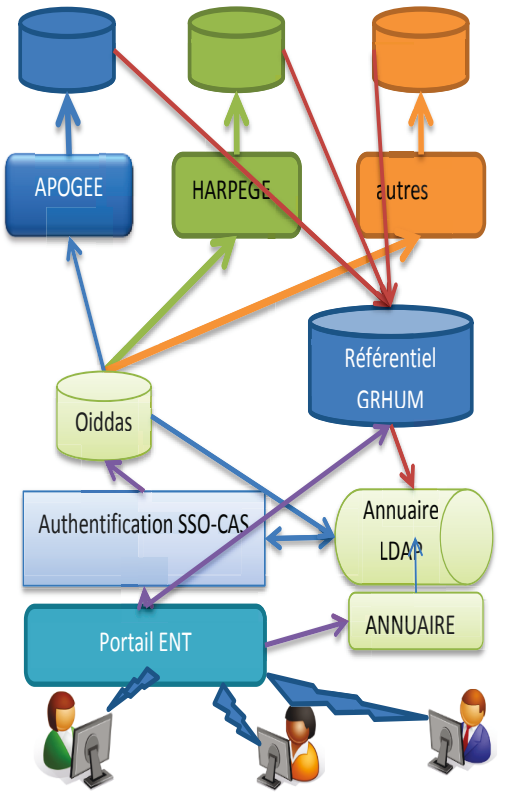


Figure 7 Ajout du SSO, du portail et d'une nouvelle version d'Apogée (2002)
Source : M.C. Collas

Puis on ajouta un référentiel (Figure 8 Mise en place du référentiel GRHUM (2005) et changement de portail (2006)). Ce référentiel copie les données des systèmes Apogée, Harpège, et d'autres pour les mettre à disposition des autres applications en lecture. C'est ce référentiel qui va alimenter le nouvel annuaire, chaque matin. Dans ce référentiel se trouve une gestion de groupes, appelée « Forum ». Chaque individu appartient à plusieurs groupes, que ce soit une structure, ou une liste de diffusion, ou un groupe d'utilisateurs d'une application. L'appartenance à un groupe va conditionner les accès aux applications dans les menus du portail ENT.

L'administrateur qui crée un gestionnaire Apogée doit donc à présent inscrire ce gestionnaire dans le groupe des utilisateurs d'Apogée.



Mise en place d'un référentiel alimenté par Apogée, Harpège et qui est le seul à alimenter l'annuaire. Le référentiel fait correspondre les codes individus (étudiant, personnel) avec leur login

Figure 8 Mise en place du référentiel GRHUM (2005) et changement de portail (2006)
Source : M.C. Collas

On est donc passé d'une inscription à un système à trois inscriptions à trois systèmes différents, elles-mêmes soumises à une inscription préalable dans l'annuaire.

2.2.1.2. Augmentation des volumes et des cas particuliers

Avec le temps, le nombre d'utilisateurs a augmenté. Lorsqu'un utilisateur sort des effectifs, il devrait être mis hors service, mais les responsables de scolarité qui réclament la création de nouvelles habilitations pour leurs collaborateurs oublient de signaler leur départ, surtout s'ils s'en vont eux-mêmes.

Le logiciel Apogée donne des droits différents selon le type d'utilisateur auquel on est attaché. Ces droits sont nombreux et variés et les combinaisons de ces droits peuvent se multiplier à l'infini. Avec le temps, le nombre de types d'utilisateurs différents est devenu ingérable.

Le service SAUVES constate le nombre d'utilisateurs qui devraient être hors service, mais ne peut de son propre chef les mettre hors service, car c'est la responsabilité de la DEVU d'ajouter ou d'enlever les personnes habilitées.

2.2.1.3. Augmentation du temps passé par le service SAUVES

La charge d'inscrire un nouvel utilisateur dans Apogée a été déléguée à une personne qui doit donc vérifier que la nouvelle personne est inscrite en tant que membre du personnel dans l'annuaire de l'établissement (ce qui prouve qu'elle a été enregistrée par les services de Ressources Humaines comme membre du personnel et par le service annuaire qui attribue le login tant qu'elle n'est pas partie) avant de la créer dans Apogée, dans l'annuaire OIDDAS et dans le forum.

Au moment des inscriptions universitaires l'administrateur inscrit de nombreux vacataires recrutés pour aider aux inscriptions universitaires. Puis lors de l'intégration des nouveaux personnels, une ou deux fois par an. Enfin tout le reste de l'année, de façon ponctuelle. A chaque fois, il doit réfléchir au type d'utilisateur qui devra être attribué, sans compter les autres données de saisie.

Le service SAUVES commence à penser que son personnel aurait mieux à faire que de faire de la saisie, tâche qui pourrait être déportée dans les services de scolarité. La responsable du service commence à réfléchir à une application qui ferait toutes ces opérations en une seule fois. Mais il serait essentiel que la DEVU soit partie prenante.

2.2.2. Un besoin partagé à la DEVU

Quel intérêt la DEVU aurait-elle au projet ? Le projet est présenté lors d'une des réunions régulières DEVU-SAUVES. Des discussions il ressort que les types utilisateurs se sont multipliés de façon anarchique, chaque composante ayant les siens. L'application serait l'occasion de restreindre la liste de ces types et d'harmoniser les pratiques entre toutes les composantes. Les scolarités des composantes pourraient visualiser la liste de leurs utilisateurs, ce qu'elles ne peuvent pas faire pour l'instant.

Plus tard j'ai appris qu'au départ la cellule Apogée était composée d'informaticiens et de référents dans les composantes. Puis elle a été divisée entre le service SAUVES et la DEVU. Pendant quelques années (5 ans) les composantes géraient leurs utilisateurs sur Apogée.

C'est à ce moment-là que les types utilisateurs ont proliféré de façon anarchique. C'était à l'époque d'Apogée en client-serveur. Avec la version Apogée sur Internet, le service SAUVES a repris la gestion d'Apogée.

Toutefois, bien qu'ayant donné son accord au projet, la DEVU n'a pas considéré ce projet comme urgent, la situation actuelle leur semblant suffisamment satisfaisante.

2.2.3. Un besoin exprimé par un objectif

Au moment où le projet m'a été exposé, le cahier des charges se résumait à la description de l'objectif sur deux pages (voir le texte complet en annexe 4 Cahier des charges). Il contient :

- un objectif :

  « Objectif : créer une application web permettant aux responsables de scolarité dans les composantes de demander des habilitations Apogée pour leurs collègues. Après validation par le directeur de la DEVU, création des droits pour ces utilisateurs. »

- une définition d'une habilitation sur Apogée
- les fonctionnalités attendues
- les tables impactées dans Apogée.

Ce cahier des charges est une invitation à mener une enquête approfondie, pour comprendre comment cette application peut s'inscrire dans la continuité du système d'information existant.

2.2.4. Un besoin détaillé exprimé oralement et recueilli par interviews

Pour pouvoir enrichir le cahier des charges, il a été nécessaire de faire l'étude de l'existant en interrogeant différentes personnes. En effet la connaissance est disséminée au sein de la DISI dans plusieurs services, chacun maîtrisant les paramétrages des applications dont il est responsable.

Mme Corinne Claisse m'a précisé comment elle voyait le scénario de base, la signification des termes employés dans le cahier des charges, et les personnes qui pouvaient être interrogées :

- Mme Ghyslaine Coquillette sur la manière dont on pratique actuellement la création d'habilitations

- Mme Monique Lagarde, responsable du service SAU-RH qui gère l'annuaire et GRHUM
- M. Richard Vatré, DBA du service RESYST, sur l'annuaire OIDDAS (ou « annuaire Form Access » dans le cahier des charges).

En interrogeant ces personnes et en étudiant l'existant, j'ai découvert que les règles de gestion concernant l'attribution d'un type utilisateur n'existaient pas sous forme écrite, mais uniquement dans le savoir-faire et l'expérience de Mme Corinne Claisse. Il fallait donc arriver à recueillir toute cette connaissance pour pouvoir l'intégrer dans l'application. Comme par exemple pourquoi deux personnes qui font le même métier et ont le même rôle dans Apogée n'ont pas le même type utilisateur. En fait chaque composante a développé des habitudes de travail différentes, et ne souhaite pas en changer, malgré le désir au niveau de la scolarité centrale d'harmoniser les pratiques au sein des UFR de Paris-Descartes.

2.2.5. Le niveau de priorité qui en découle

Le fait que le besoin d'une telle application soit ressenti plus fortement du côté des informaticiens (SAUVES) que du côté du service utilisateur (DEVU) rend le projet non prioritaire. Les utilisateurs des scolarités font connaître aux services informatiques quels sont les projets qui leur semblent prioritaires, urgents, utiles et importants.

La question a déjà été évoquée dans le paragraphe 2.1.6.2 sur la gestion du temps, montrant comment le projet s'est mis en veille à plusieurs reprises au profit d'autres projets. Toutefois les projets urgents ne sont pas forcément les plus importants. Le projet gestion des habilitations est très important parce qu'il va régler un problème de fond.

2.3. CONCLUSION

Le besoin d'un projet qui simplifie le processus d'enregistrement d'un utilisateur Apogée et qui en même temps garde le système en ordre et cohérent, est fortement ressenti du côté des informaticiens, et faiblement du côté des utilisateurs. Il faudra donc tout au long de ce projet communiquer avec ces derniers d'une façon qui les intéresse et qui mette en avant l'avantage qu'ils pourront en tirer.

Avec le recul, nous voyons bien qu'il s'agit d'un cas *d'entropie des systèmes d'information* dans la gestion des identités, qui va trouver une solution par l'automatisation d'un processus (Volle, 2002). J'entends par entropie ce désordre qui va grandissant si l'on ne fait rien pour

l'enrayer : de plus en plus de doublons, d'utilisateurs considérés comme actifs dans Apogée mais absents de l'annuaire, de plus en plus de types d'utilisateurs, de plus en plus de particularités selon les composantes, et de plus en plus de mal à s'y retrouver.

Le projet d'automatisation du processus va me permettre dans un premier temps de bien comprendre comment il se déroule dans le système existant pour pouvoir proposer un nouveau processus automatique plus performant.

Chapitre 3. Conception générale de l'automatisation d'un processus

3.1. INTRODUCTION

Nous comprenons par conception générale cette phase qui va consister à rechercher les informations évoquées dans le cahier des charges, à faire plusieurs investigations dans le système informatique existant, à analyser le processus géré manuellement et à en établir les défaillances, et à proposer une maquette des écrans à l'utilisateur final. Les documents livrables à l'issue de cette phase sont supposés être compréhensibles par l'utilisateur non informaticien.

3.2. CYCLE EN V ET APPROCHE AGILE

L'expérience de Mme Corinne Claisse, maître d'œuvre du projet pour la DEVU et maître d'ouvrage pour moi, lui a montré que la plupart des utilisateurs ont des difficultés à visualiser d'avance ce que sera le résultat final. Elle préconise donc de faire déjà une maquette dans l'étape de conception pour aider les décideurs à voir la future application, mais surtout de réaliser rapidement l'application correspondant à sa description du besoin, de la mettre entre les mains des utilisateurs finaux, puis de laisser venir les demandes d'améliorations du produit dans un deuxième temps.

La première version du projet va se dérouler selon un cycle en « V » classique : analyse générale, conception détaillée, développement et tests, validation, mise en production et formation des utilisateurs. Avec toutefois des itérations entre phases de développement, de tests et de validation, comme il se doit.

L'université privilégie une démarche qui s'apparente aux approches agiles. Elle n'est pas agile dans le sens qu'elle mettrait en œuvre consciemment des méthodes estampillées comme XP (e**X**treme **P**rogramming©) ou Scrum©. Mais de fait la pratique respecte les valeurs de l'agilité[5] (Boisvert *et al.*, 2011), citant (Wikipedia contributors, 2013):

- Il n'y a pas de contrat signé entre les parties, comme on le ferait avec une entreprise extérieure, puisque cela sera réalisé en interne. Les délais sont très dépendants du temps qui peut être consacré à ce projet plutôt qu'à un autre.
- Les seuls documents exigés sont le manuel utilisateur, le dictionnaire des données et le guide de déploiement, ceci s'expliquant par le fait que la plus grosse partie de l'activité est de l'intégration.
- De plus le circuit est très court entre le donneur d'ordre et le réalisateur, l'épaisseur d'une cloison entre deux bureaux voisins.
- Je suis libre du choix de mes outils et méthodes de conception, ce qui compte est le résultat.
- Il est possible de livrer partiellement l'application en environnement de test pour mettre à disposition de l'utilisateur SAUVES, tout en continuant de développer la suite, et en intégrant au fur et à mesure les retours des utilisateurs.

En ce qui concerne les méthodes de conception, j'ai mobilisé essentiellement l'approche objet et UML pour représenter les systèmes et processus existants et à venir. Une grande partie de ces travaux est restée au brouillon ou dans mes documents privés et seule une petite partie utile a été documentée et partagée.

3.3. LES DOCUMENTS DE CONCEPTION

3.3.1. Les habitudes de l'université Paris-Descartes en termes de documentation

Quels sont les documents à produire et à livrer ? Après enquête auprès des collègues du service SAUVES seuls les documents suivants sont exigés :

- dictionnaire des données
- guide de déploiement

[5] Valeurs de l'agilité selon le Manifeste agile http://fr.wikipedia.org/wiki/Manifeste_agile

- manuel utilisateur
- éventuellement support de formation qui peut être le manuel utilisateur

Le service SAUVES fait essentiellement de l'intégration et il a l'habitude de recevoir ce type de documents de ses fournisseurs. Mais qu'en est-il du service DEV qui est consacré au développement de nouvelles applications ? A-t-il d'autres documents ? Des modèles ?
Le service DEV n'a pas d'autres documents à délivrer, ce qui ne les empêche pas d'utiliser des outils UML par exemple. La documentation s'adapte au cas par cas, projet par projet. Donc une certaine liberté est offerte sur la forme à donner aux documents. Ce qui est attendu de la part des développeurs et intégrateurs est qu'ils livrent des applications qui fonctionnent, testées et fiables.

3.3.2. Le choix des documents à produire

Partant de là, le choix des documents livrables a consisté à prendre comme modèles les documents existants pour d'autres projets semblables et à créer de toutes pièces les documents complémentaires, selon les usages en vigueur dans la profession. Les documents suivants se sont inspirés de l'application Rendez-vous précédemment intégrée :

- Dictionnaire des données avec Scripts SQL en annexe
- Guide de déploiement
- Manuels utilisateurs

Les documents suivants se sont inspirés des supports de formation utilisés dans le service pour d'autres applications telles qu'Apogée, ADE, etc. ….

- Support de formation (présentation PowerPoint)

Les documents suivants ont été inspirés de mon expérience et de recherches sur Internet :

- Dossier de conception générale accompagné d'une maquette
- Dossier d'étude détaillée
- Dossier de validation

J'ai appelé dossier de conception générale le document qui complète ce qui manque au cahier des charges et qui contient le résultat de mon travail d'analyse fonctionnelle. Il est volontairement peu technique pour être accessible aux non informaticiens. Les informations très techniques relatives au système d'information existant n'ont pas été diffusées.

3.4. L'ETUDE DE L'EXISTANT

3.4.1. Le processus de demande de création d'une habilitation

Par interview des différents acteurs, on peut reconstituer le processus suivant (Figure 9 Ancien processus d'habilitation pour Apogée) :

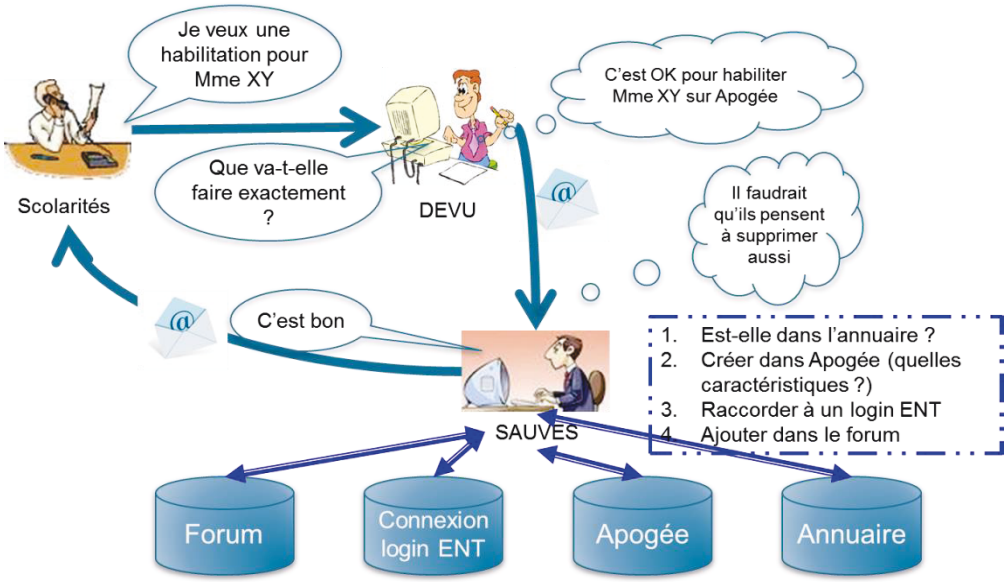


Figure 9 Ancien processus d'habilitation pour Apogée
(Source : M.C. Collas)

1. Un responsable de scolarité demande une habilitation à utiliser Apogée au directeur de la DEVU.

2. Le directeur de la DEVU lui envoie un formulaire à compléter pour identifier l'usage qui sera fait d'Apogée. (Voir annexe 5 Formulaire de demande d'habilitation) Avant cela se faisait par téléphone.

3. Le directeur de la DEVU donne son accord après vérification.

4. Le directeur de la DEVU demande par email au service SAUVES de créer une habilitation.

5. Le service SAUVES vérifie si l'habilitation existe déjà pour la personne, ce qui serait possible dans un cas de mutation de poste. Le service demande des compléments d'information au demandeur pour pouvoir renseigner tous les champs et déterminer le type d'utilisateur.

6. Le service SAUVES se connecte en tant qu'administrateur dans le logiciel Apogée et crée (ou modifie) un utilisateur dans Apogée, avec code utilisateur et mot de passe dans le sens du user Oracle (Figure 10 Création d'un utilisateur dans Apogée-. Tous les champs utiles de l'utilisateur sont renseignés. Il refait la même chose sur la base Apotest.

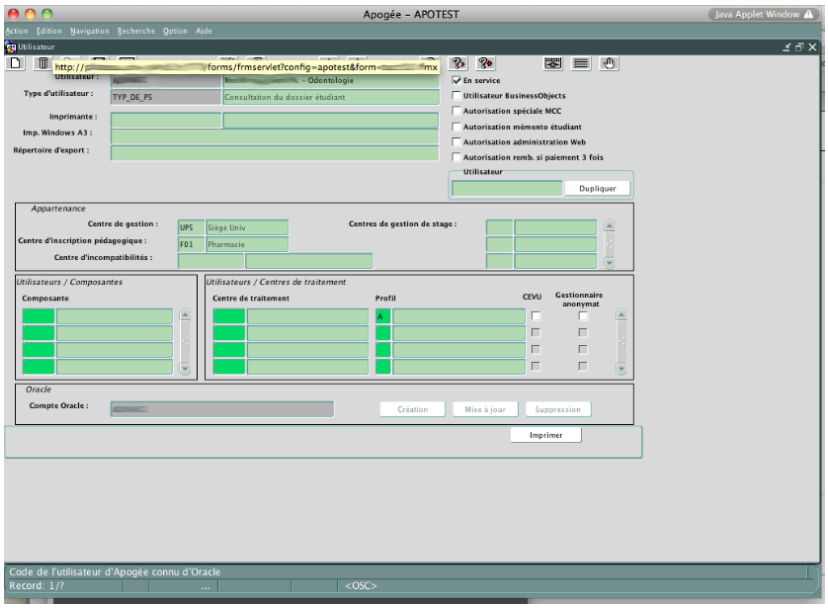


Figure 10 Création d'un utilisateur dans Apogée-
Source : Université Paris Descartes

7. Le service SAUVES vérifie dans l'annuaire si la personne fait partie du personnel (Figure 11 Recherche dans l'annuaire de l'université). Si ce n'est pas le cas il faut attendre que la personne soit arrivée et qu'elle ait reçu un login pour accéder à l'ENT. Le service SAUVES se met en relation avec le service SAU-RH qui gère l'annuaire en cas d'interrogation.

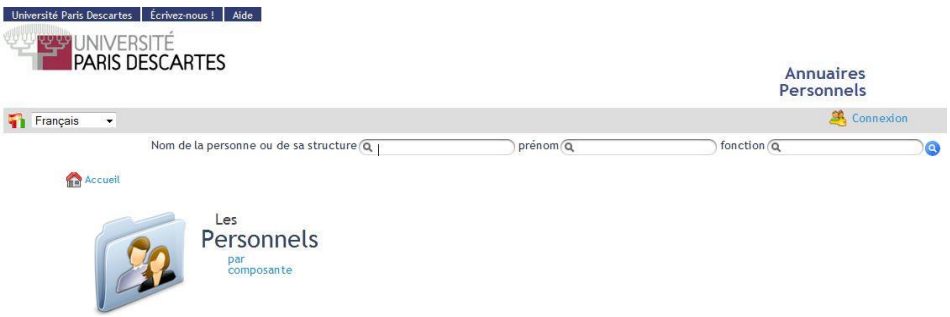


Figure 11 Recherche dans l'annuaire de l'université
Source : Université Paris Descartes

8. Le service SAUVES se connecte en tant qu'administrateur dans le logiciel OIDDAS (Oracle Identity Management) et crée ou modifie un compte pour le login trouvé dans l'annuaire (Figure 12, Figure 13, Figure 14). A ce compte est associée une liste de configurations. Une configuration est : le nom d'une base de données + le code d'accès de l'utilisateur à cette

base + le mot de passe. Un membre du personnel peut avoir accès à une dizaine de bases de production ou de tests de cette façon : Apogée, Apotest, Harpège, Helico, etc… Le service SAUVES crée donc un accès à Apogée et un accès à Apotest pour le login donné (Figure 15).

Figure 12 OIDDAS Recherche d'une personne
Source : Université Paris Descartes

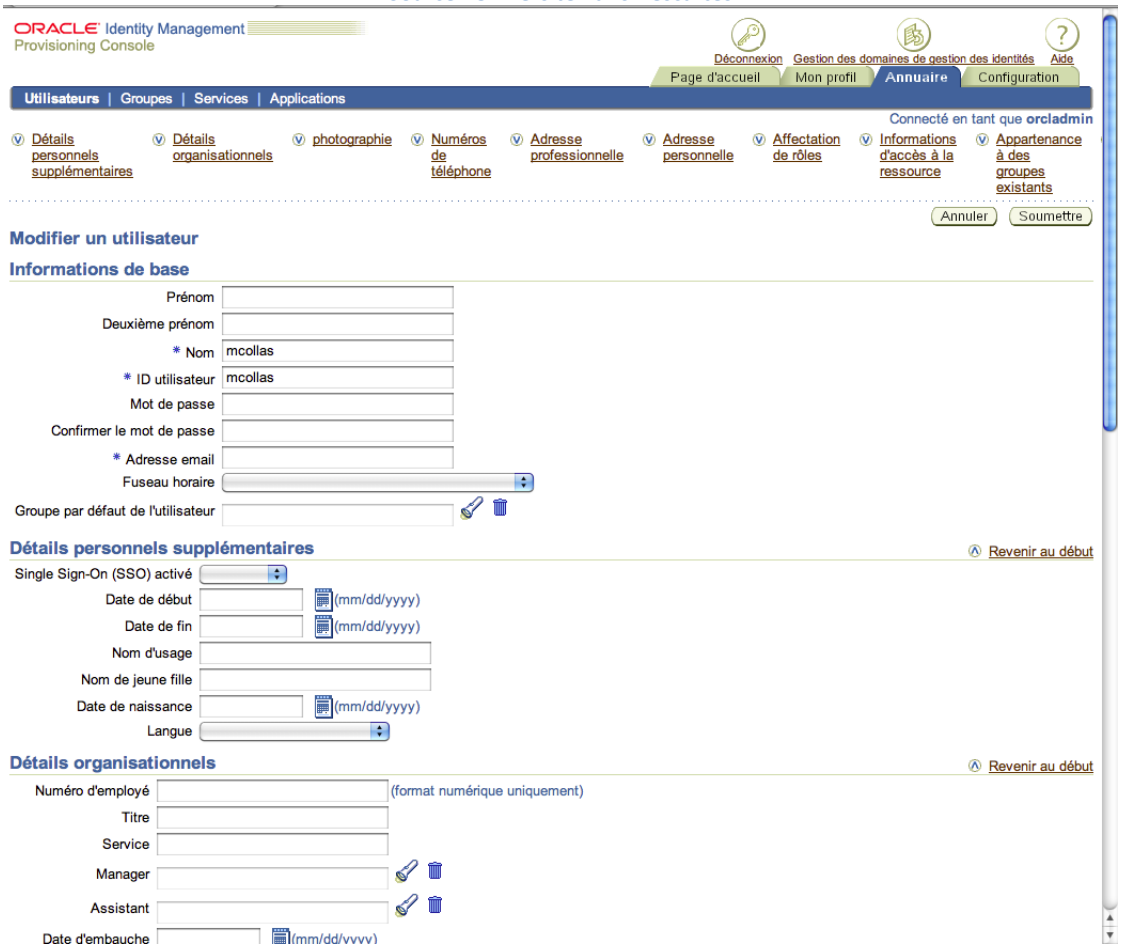


Figure 13 OIDDAS Création d'une entrée
Source : Université Paris Descartes

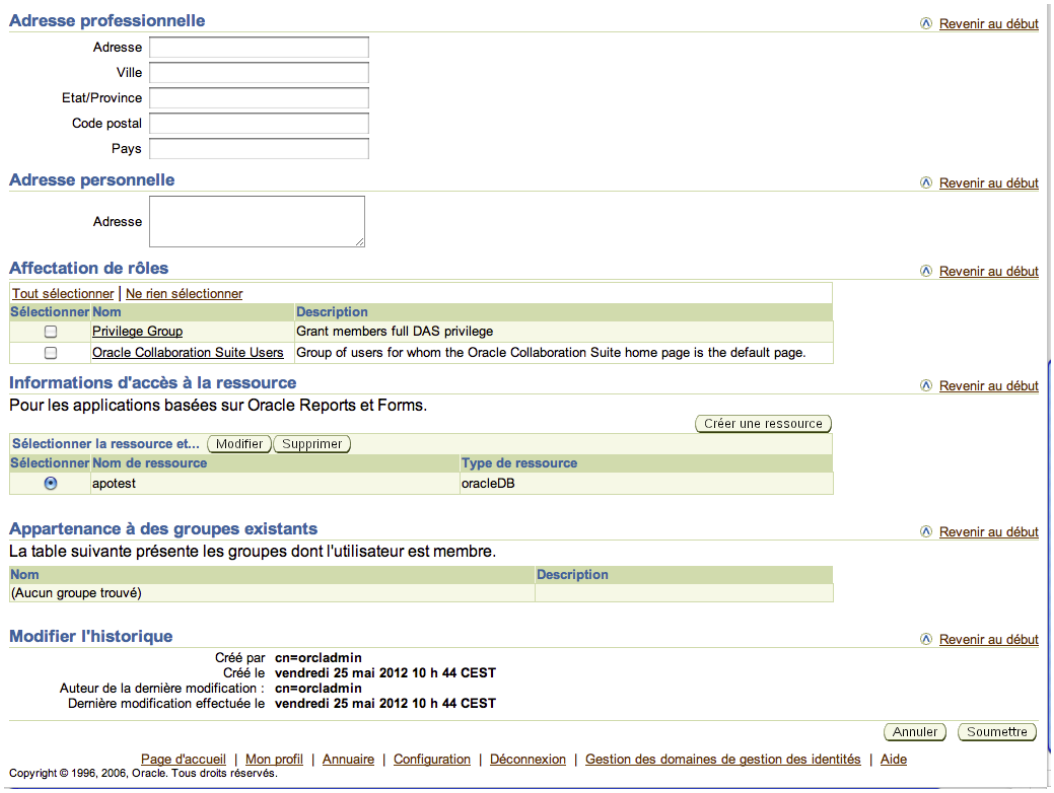


Figure 14 OIDDAS Création d'une entrée (suite)
Source : Université Paris Descartes

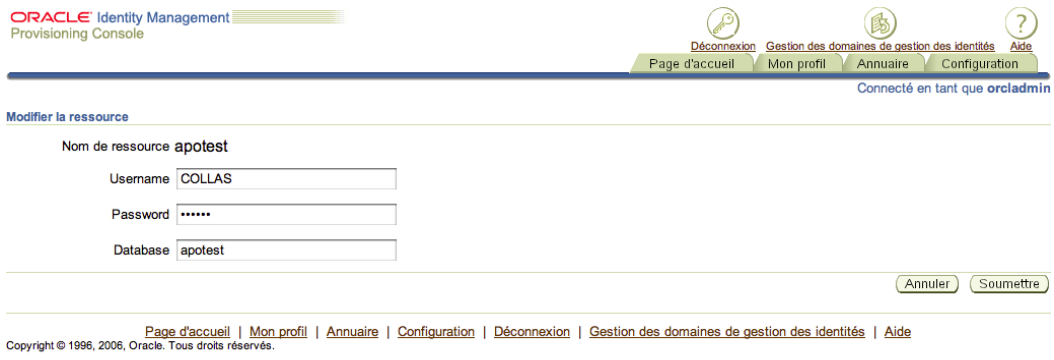


Figure 15 OIDDAS Création d'une ressource
Source : Université Paris Descartes

9. Le service SAUVES se connecte en tant qu'administrateur sur le logiciel qui gère les groupes dans l'annuaire, qu'on appelle le Forum, car certains de ces groupes sont aussi des listes de diffusion (Figure 16). Une personne identifiée par son login peut appartenir à plusieurs groupes, et un groupe peut être inclus dans un autre. Il choisit le groupe des utilisateurs d'Apogée (*code 33373*) et rajoute le login de la personne habilitée (Figure 17).

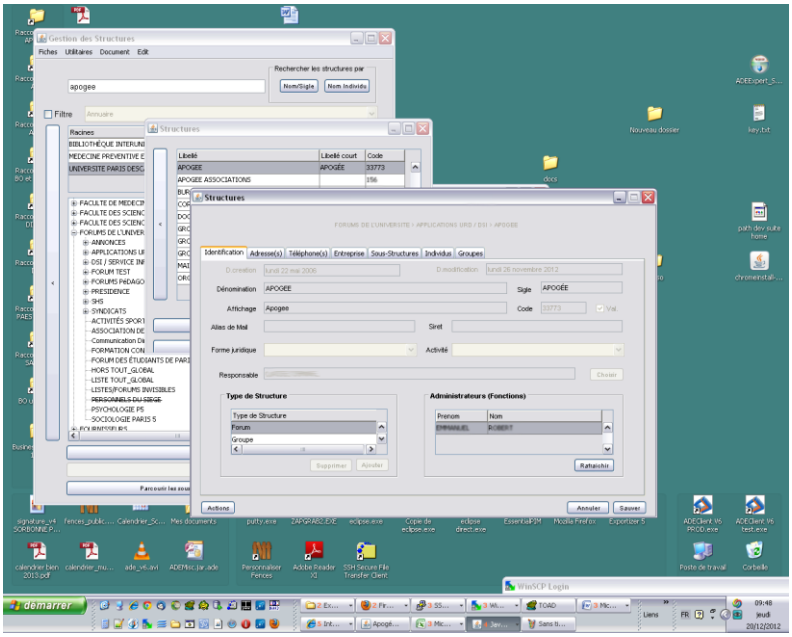


Figure 16 Gestion des structures, recherche de la structure
Source : Université Paris Descartes

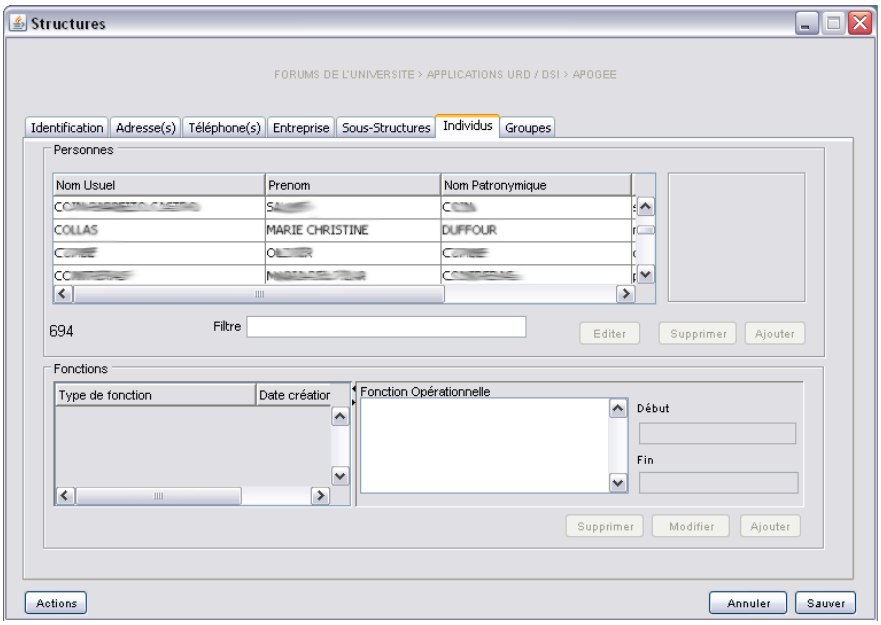


Figure 17 Gestion des structures : membres du groupe
Source : Université Paris Descartes

10. Le service SAUVES informe le responsable de scolarité que la personne peut maintenant utiliser Apogée.

Que fait la personne nouvellement habilitée à utiliser Apogée ?

1. Sur le site de l'université elle essaie d'entrer dans l'ENT (Figure 18), intranet du personnel et des étudiants, à l'aide de son login et mot de passe (Figure 19).

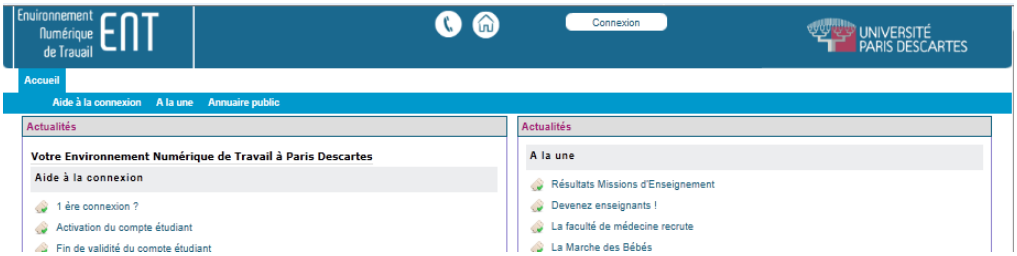


Figure 18 Portail de l'ENT
Source : Université Paris Descartes

Figure 19 Portail de l'ENT authentification
Source : Université Paris Descartes

2. Dans l'ENT elle cherche parmi les menus celui qui mène à Apogée, et grâce à l'inscription sur le forum le trouve et clique sur l'onglet (Figure 20).

Figure 20 Portail de l'ENT: les onglets
Source : Université Paris Descartes

3. Un formulaire (Form Web Access d'Oracle) lui demande de choisir sa configuration (Apogée ou Apotest ou autres..). Le logiciel OIDDAS va vérifier qu'il connaît le code utilisateur et le mot de passe d'Apogée et sinon lui demande (Figure 21).

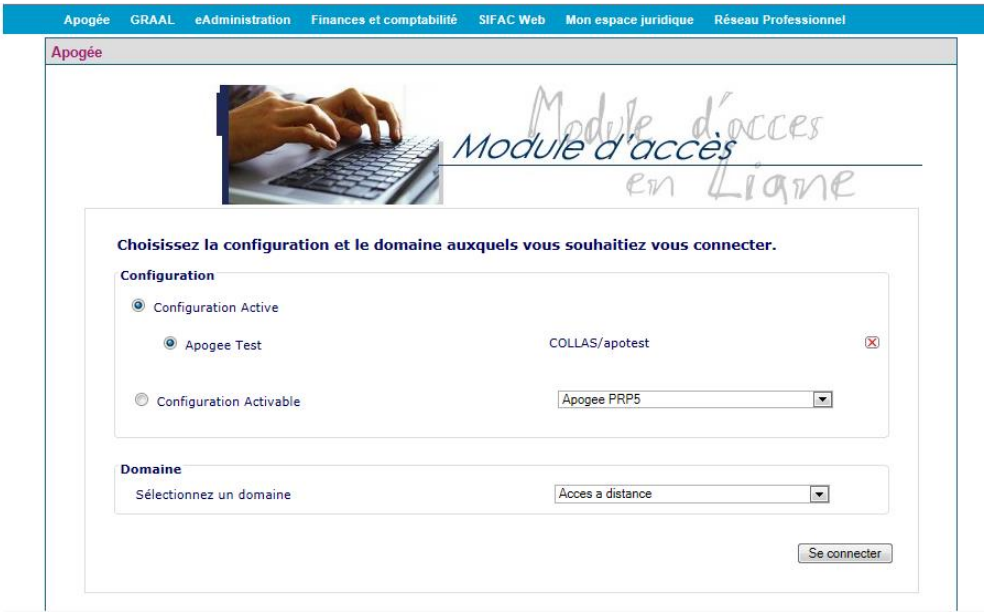


Figure 21 Web Form Access d'Apogée
Source : Université Paris Descartes

4. En cas de succès, plusieurs fenêtres successives s'ouvrent (Figure 22): ce sont des adaptateurs transformant en applets pour le web les fenêtres d'Apogée initialement conçues en mode client-serveur.

Figure 22 les applets d'Apogée
Source : Université Paris Descartes

5. Enfin la fenêtre d'accueil d'Apogée s'ouvre et l'utilisateur peut commencer à travailler (Figure 23).

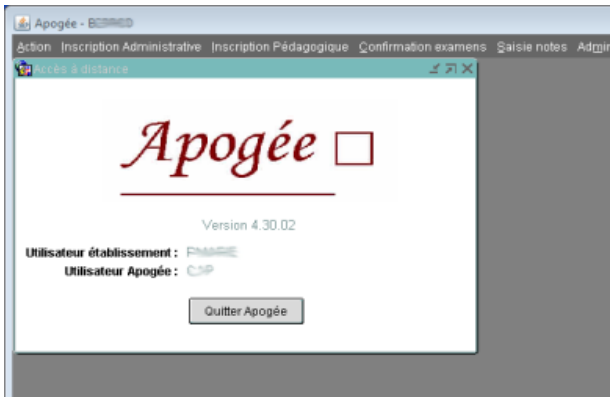


Figure 23 Ecran d'accueil Apogée
Source : Université Paris Descartes

6. Si la personne rencontre un blocage pour se connecter, elle appelle l'assistance du service SAUVES : onglet Apogée absent, mot de passe inconnu, accès à la configuration détruit par erreur, droits d'utilisation ne correspondant pas à ses attentes, etc….

3.4.2. Le système d'information de Paris-Descartes impliqué dans le projet

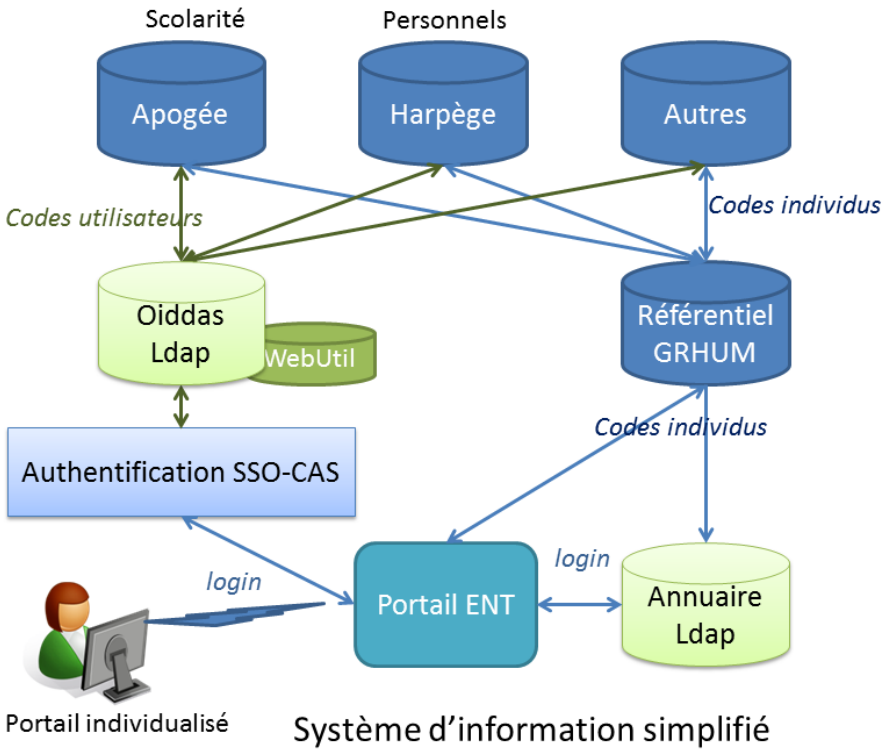


Figure 24 Système d'information simplifié
Source : MC Collas

Les applications qui ont besoin d'informations venant d'un autre système s'adressent au référentiel qui collecte les informations individuelles : les informations des étudiants à partir

d'Apogée, les informations du personnel à partir d'Harpège. L'annuaire est mis à jour à partir du référentiel. L'authentification s'appuie sur l'annuaire avec le login.

Les gestionnaires d'Apogée ou d'Harpège utilisent des codes d'accès qui sont encore différents (hérités de l'époque client-serveur). Le gestionnaire d'identité OIDDAS délivre l'accès à Apogée ou Harpège ou autres à partir du login.

3.4.3. Les applications qui interviennent dans ce processus

Comme le montre le processus, quatre applications interviennent : l'annuaire, Apogée, OIDDAS et le forum. Ces applications sont accessibles en étant authentifié via le portail de l'ENT.

1. **Le portail de l'ENT** : accessible à partir du site web de l'université, le portail ENT (Environnement Numérique de Travail) nécessite une authentification SSO sur un serveur CAS. Pour pouvoir passer l'étape d'authentification avec succès il faut s'identifier avec le login et le mot de passe de l'annuaire. Le portail de l'ENT est géré par le service ENT.

Toutes les applications accessibles par le web à destination des étudiants inscrits ou des personnels doivent être protégées par le système d'authentification, ce qui est le cas des applications désignées ci-dessous.

2. **Apogée** est à la fois le nom du logiciel géré par le service SAUVES.

La base de données existe en version de production (PRP5) et en version de test (Apotest). Les utilisateurs ont accès à ces deux versions, la version de test leur servant de bac à sable et pour la formation. La base de test est aussi utilisée par les services informatiques pour les tests, elle est copiée à partir de la base de production plusieurs fois par an.

La base de données d'Apogée contient plusieurs centaines de tables et de vues. Seules quelques tables du module « Référentiel » sont impliquées pour ajouter un nouvel utilisateur (12 tables exactement, dont 4 en écriture).

Le logiciel Apogée développé par l'AMUE est considéré comme en fin de vie. Une nouvelle version est en préparation[6].

[6] Actualités sur le site de l'Amue (Amue, 2012)

Autour d'Apogée gravitent des applications diverses qui utilisent la base de données Apogée, soit venant de l'AMUE, ou d'une autre université ou développées en interne.

3. **L'annuaire** est une base de type LDAP qui se met à jour toutes les nuits à partir du référentiel. Cet annuaire est géré par le service SAU-RH. C'est un annuaire SUPANN dont les caractéristiques sont communes aux établissements d'enseignement supérieur.

L'annuaire fournit des informations de connexion (login et mot de passe), et des informations générales sur l'étudiant, le membre du personnel, et d'autres catégories (anciens, fournisseurs, abonnés bibliothèque, visiteurs,...). Les informations générales qui nous intéressent sont les noms, prénoms, statut, structures (ou groupes). Une personne ne peut entrer dans l'ENT si elle ne figure pas dans l'annuaire.

Le référentiel qui alimente l'annuaire est alimenté à partir du logiciel et base de données Harpège pour les membres du personnel, d'Apogée pour les étudiants et d'Hélico. Sa base de données dans Oracle s'appelle GRHUM.

4. L'annuaire **OIDDAS** est une autre base de type LDAP dont le but est de gérer différentes identités. Cet annuaire est géré par le service RESYST.

L'identité principale est le login de l'annuaire. Dans la branche Extended Properties se trouvent les différentes configurations autorisées pour l'individu concerné. Une configuration est l'ensemble composé du nom de la base, du code d'accès sur cette base et du mot de passe.

Il existe une base de données (WebUtil) attachée à cet annuaire qui ne contient pas de tables, mais seulement des procédures d'appel pour les formulaires d'accès. Cette absence de table qui pourrait servir de table de jointure entre Apogée et le référentiel est un des problèmes qu'il faudra surmonter.

5. Le **forum** est un sous-ensemble du **référentiel GRHUM**. Comme l'ensemble du référentiel, il est géré par le service SAU-RH.

L'application «Gestion des structures » permet de le mettre à jour. L'université est un groupe racine et tous les autres groupes ou structures sont ses branches organisées de façon hiérarchique. Les structures peuvent être une UFR, ou le siège, ou un service, mais aussi un

groupe d'utilisateurs d'un logiciel, ou une liste de diffusion. Chaque membre du personnel peut faire partie de plusieurs groupes. En tant qu'administrateur, le service SAUVES ajoute au groupe des utilisateurs d'Apogée ceux qui sont autorisés à utiliser l'application. Ce groupe est aussi une liste de diffusion permettant d'informer par e-mail les utilisateurs que l'application Apogée sera fermée pour maintenance par exemple.

Le portail ENT utilise le forum pour afficher ou non les menus auxquels un individu a droit après s'être authentifié.

3.4.4. L'analyse des données d'Apogée, de GRHum et des annuaires

3.4.4.1. Apogée

Apogée est un système complexe assez long à prendre en main en tant qu'utilisateur. C'est pourquoi il existe des formations dispensées par le service SAUVES et des supports de formation. Les formations dépendent des modules qui doivent être utilisés et qui sont organisés en domaines :

- Dossier étudiant
- Epreuve
- Exploitation
- Inscription administrative
- Inscription pédagogique
- Modalités de contrôle
- Pilotage
- Référentiel
- Résultats
- Stage
- Structure des enseignements
- Thèses HDR-DRT

C'est dans le domaine référentiel qu'un nouvel utilisateur est ajouté, sur un seul écran (Figure 25 Saisie d'un utilisateur Apogée) :

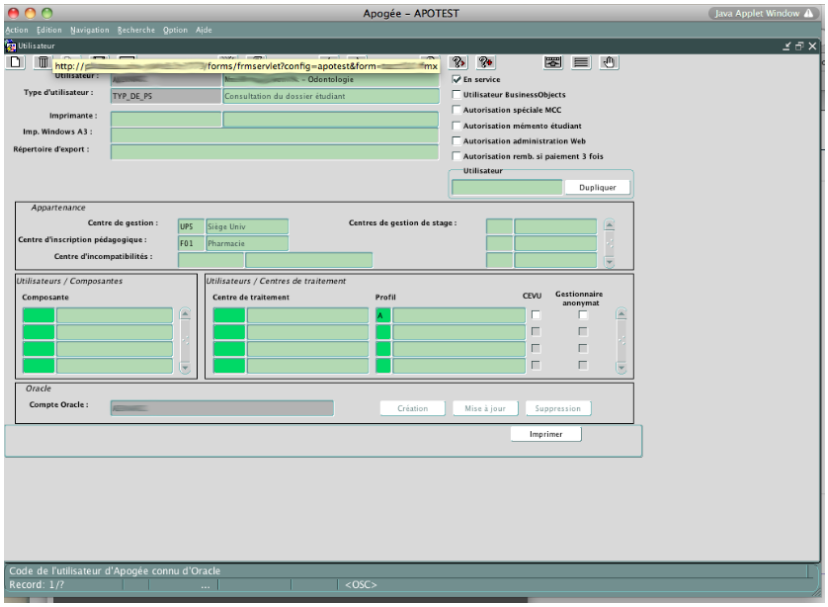


Figure 25 Saisie d'un utilisateur Apogée
Source : Université Paris Descartes

La mise à jour de cette page impacte directement la table UTILISATEURS dans la base de données (voir Figure 26 Extrait de la base de données Apogée). Cette table correspond à un compte Oracle dont le code est l'identifiant de cette table.

La table UTILISATEURS contient plusieurs codes qui sont des clés étrangères vers des tables de libellés : TYP_UTILISATEUR, CENTRE_GESTION, CENTRE_INS_PED, CENTRE_INCOMP.

Un utilisateur peut aussi gérer plusieurs composantes, plusieurs centres de stage et plusieurs centres de traitement des notes. Ce qui implique les tables de données COMPOSANTE, CENTRE_GES_STG, CENTRE_TRAITEMENT, et les tables de jointure UTI_CMP, UTI_CGS, UTI_COLLECTER_CTN.

Le centre d'inscription pédagogique permet de contrôler une ou plusieurs composantes, ce qui est reflété par la table CMP_GERER_CIP.

Lorsqu'un utilisateur est créé ou modifié (jamais supprimé, mais mis hors service), seulement les tables UTILISATEURS, UTI_CMP, UTI_CGS, UTI_COLLECTER_CTN sont modifiées.

La table TYP_UTILISATEUR est d'une grande importance : un type utilisateur détermine des droits d'utiliser ou non un module, soit en consultation, soit en modification. A l'intérieur de ces grands domaines, certaines opérations peuvent être autorisées ou modifiées. Des limitations peuvent être portées sur le nombre de composantes dont la modification ou la consultation est autorisée.

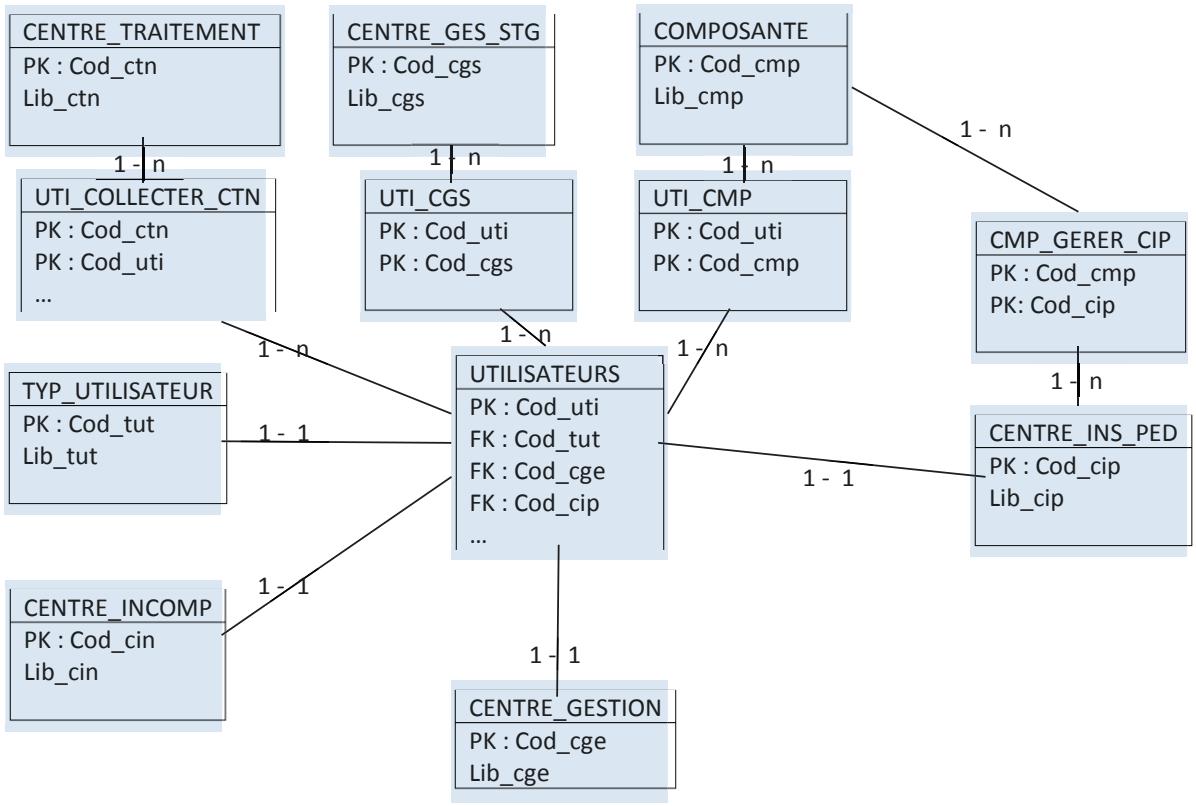


Figure 26 Extrait de la base de données Apogée
Source : MC COLLAS

L'examen du contenu des tables montre que sur environ 75 types utilisateurs existants, 54 différents sont utilisés, dont 2 par plus de 100 personnes, 11 par 10 à 50 personnes, 21 par 2 à 9 personnes, 18 par une seule personne. C'est manifestement trop, certains types ne différant que par peu de détails et le souhait de l'équipe SAUVES est de pouvoir réduire le nombre de types utilisateurs à une quinzaine correspondant aux grandes fonctions exercées dans l'université.

Les tables d'Apogée peuvent être consultées directement par un informaticien à l'aide d'outils tels que SQL Developer ou TOAD[7] et par requêtes SQL.

[7] Outils de consultation et d'administration de bases de données Oracle.

3.4.4.2. Référentiel Grhum

En ce qui concerne le référentiel GRHUM, l'examen des tables a été complété par l'interview de la responsable du service SAU-RH (ci-dessous Figure 27 Extrait de la base de données GRHUM).

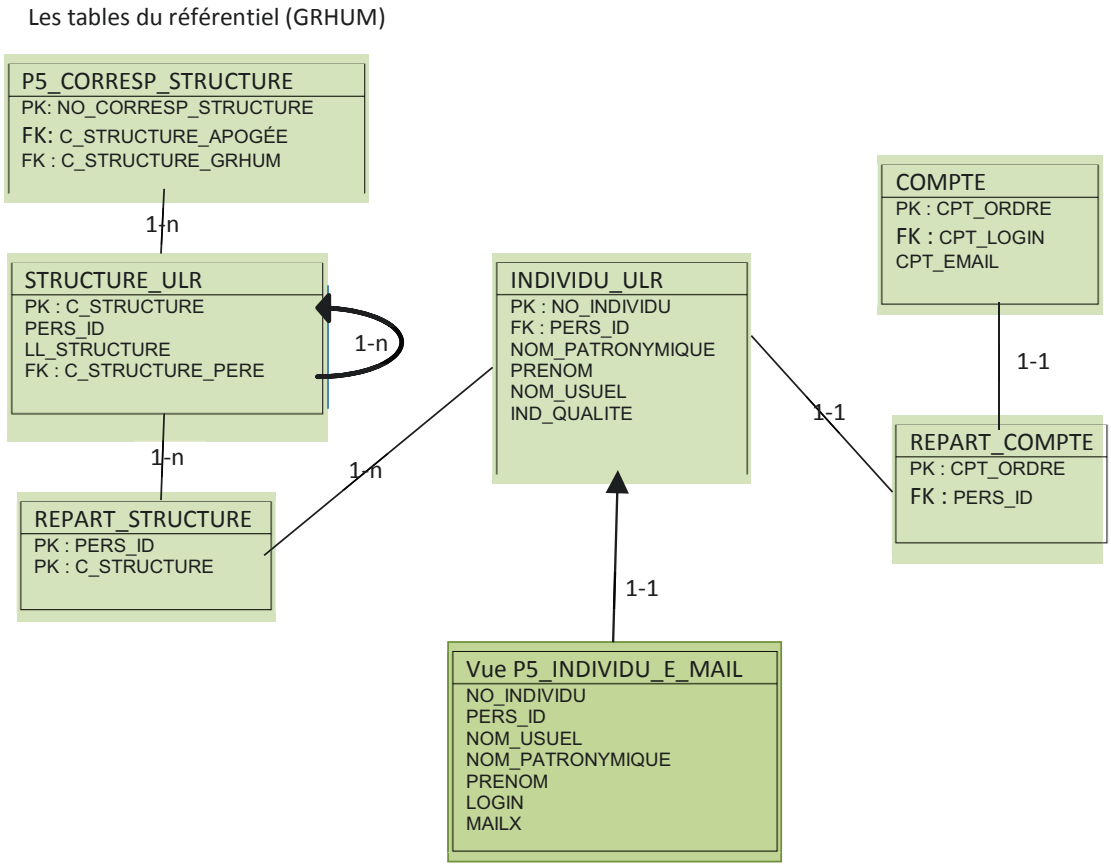


Figure 27 Extrait de la base de données GRHUM
Source : MC COLLAS

Les besoins sont les suivants :

- comment connaître la composante dans laquelle travaille un membre du personnel ?

Le code composante dans Apogée correspond au code structure Harpège dans le référentiel (l'inverse n'est pas vrai).

La table P5_CORRESP_STRUCTURE donne pour chaque code structure Harpège le code composante Apogée s'il existe, sinon rien.

La table REPART_STRUCTURE donne pour chaque personne le code PERS_ID et les structures auxquelles elle appartient.

La table STRUCTURE_ULR donne pour chaque code structure la structure parent.

- Comment connaître le forum des utilisateurs d'Apogée ?

Le code structure 33773 correspond aux utilisateurs d'Apogée

- Comment rattacher le code personne à son login ?

La table INDIVIDU_ULR permet d'obtenir le code NO_INDIVIDU à partir de PERS_ID.

La table REPART_COMPTE permet d'avoir le code CPT_ORDRE à partir de PERS_ID.

La table COMPTE permet d'avoir le CPT_LOGIN et l'e-mail à partir de CPT_ORDRE.

La table P5_HARPEGE_NUMBER permet d'avoir le nom et le prénom à partir du code NO_INDIVIDU.

La vue P5_INDIVIDU_E_MAIL fait la requête qui lie toutes ces tables pour donner le login, nom, prénom, e-mail, NO_INDIVIDU et PERS_ID.

Toutes ces tables sont utiles si l'on veut obtenir des informations sans passer par l'annuaire, lorsque par exemple on veut utiliser dans la même requête SQL des informations qui viennent d'Apogée et du référentiel. Dans ce cas précis on utilisera un DBLink (lien entre bases de données) pour accéder à la base GRHUM en étant identifié comme utilisateur Oracle sur Apogée. Un DBLink est le moyen offert par Oracle lorsqu'on est connecté sur une base, de se connecter sur une autre base de données afin de faire une requête sur les tables des deux bases à la fois.

3.4.4.3. Annuaire Ldap

Le logiciel utilitaire Gawor Ldap Browser permet d'inspecter n'importe quel LDAP pourvu que l'on ait les codes d'accès (Figure 28).

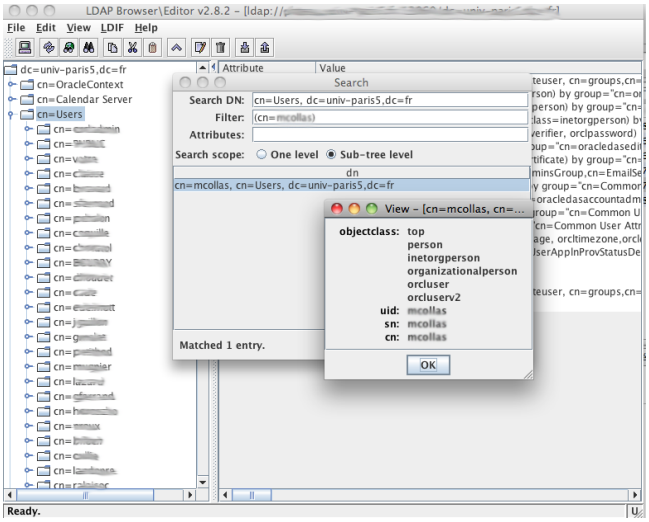


Figure 28 Outil Gawor Ldap Browser
Source : Université Paris Descartes

L'annuaire est organisé selon les recommandations de SUPANN de la façon suivante (Figure 29):

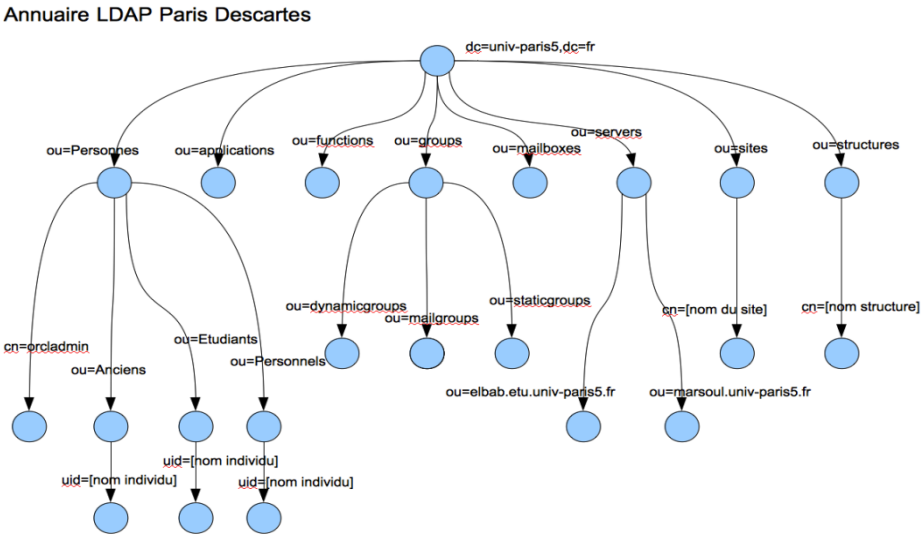


Figure 29 Structure simplifiée de l'Annuaire Ldap
Source : MC Collas

En suivant la branche Personnes / Personnels, et l'uid étant égal au login, on obtient les attributs d'une personne appartenant au personnel. Les attributs susceptibles de nous intéresser sont le login, nom, prénom, le numéro Harpège, l'e-mail, les codes structures.

3.4.4.4. Oracle Identity Management ou OIDDAS

Toujours avec l'outil Gawor Ldap Browser on peut inspecter le Ldap OIDDAS et découvrir la structure suivante (Figure 30) :

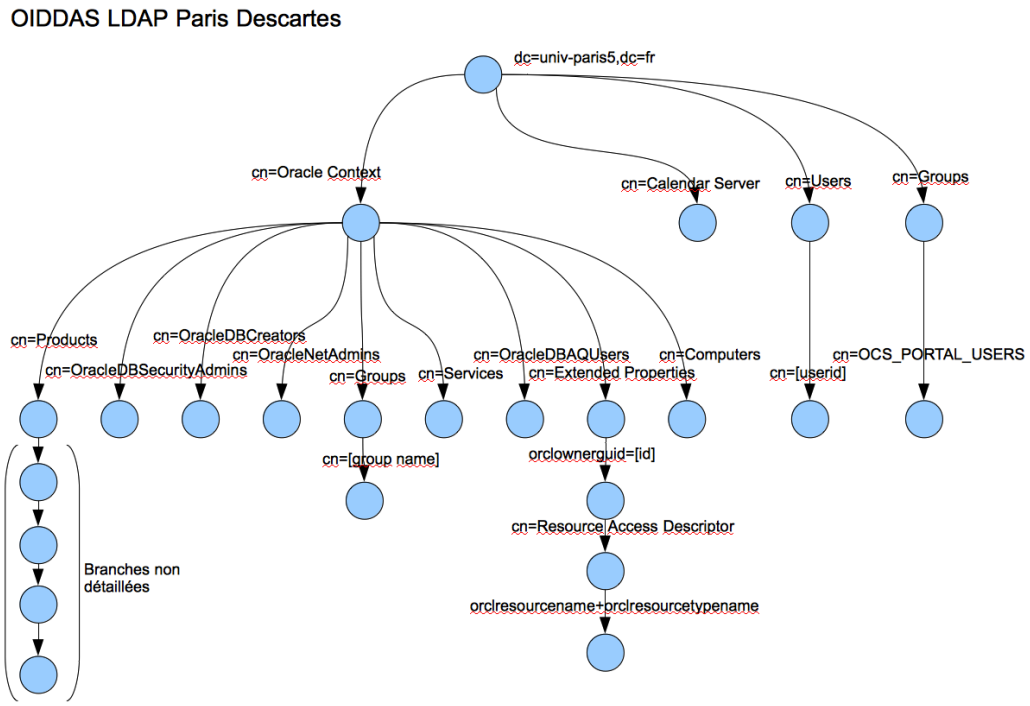


Figure 30 Structure simplifiée du Ldap OIDDAS
Source MC Collas

L'administrateur utilise l'outil Oracle Identity Management pour créer, modifier et supprimer dans cet annuaire (voir les figures déjà présentées plus haut Figure 13 OIDDAS Création d'une entrée, Figure 14 OIDDAS Création d'une entrée (suite) , Figure 15 OIDDAS Création d'une ressource).

La création d'un *user* dont l'identifiant est le login de l'annuaire crée un enregistrement dans la branche *Users*. L'ajout à cet *user* d'un premier accès à Apogée ou Apotest ou Harpège ou Hélico, va créer un point d'entrée dans la branche Oracle Context / Extended Properties, il y aura création d'un nœud '*orclownerguid*' + identifiant attribué par le système.

A l'intérieur, un nœud Resource Access Descriptor (RAD).

A l'intérieur une ou plusieurs feuilles correspondant à une configuration de base de données, Apogée par exemple. Les attributs seront le code utilisateur de cette base et le mot de passe pour se connecter, ainsi que le chemin pour accéder à la branche *Users*. Chaque fois qu'on ajoute une entrée pour une base de données (RAD Entry), il y a création d'une feuille '*orclresourcename* '.

Contrairement à l'annuaire il n'y a pas de base de données permettant d'obtenir ces résultats par requête SQL qui soit accessible. Il y a une base de données qui contient des procédures

stockées FORMS_RAD_MANAGER permettant de créer une session, fermer la session, créer un RAD entry, supprimer un RAD Entry, obtenir un RAD entry et donner une nouvelle définition de configuration.

Le recours à la procédure de création de RAD Entry permet de créer une configuration activable. C'est-à-dire que lorsque l'utilisateur se connectera sur Apogée la première fois, on lui demandera d'entrer son code et son mot de passe Apogée pour activer sa configuration qui deviendra alors une configuration activée. Beaucoup d'utilisateurs n'activent que leur configuration de production et non celle de test, ce qui explique qu'il y ait moins d'utilisateurs dans la base de tests.

3.4.4.5. Inspection croisée par programmation

Le nombre d'utilisateurs en service semble anormalement élevé par rapport au nombre attendu. Aussi j'ai réalisé un petit programme batch pour effectuer quelques contrôles croisés, et aussi tester les modes d'accès à ces différentes bases. Ce programme lit OIDDAS en sélectionnant uniquement les utilisateurs d'Apogée ou Apotest pour se créer une table interne de jointure login/code utilisateur Apogée. Puis il va comparer cette table à la table UTILISATEURS d'Apogée, aux tables P_TEMP_INDIVIDU et REPART_STRUCTURE dans le référentiel. Il édite des listes Excel d'individus considérés en anomalie. Les anomalies recherchées ont été les suivantes :

- existe-t-il des utilisateurs dans Apogée, étant en service mais dont on ne peut connaître le login par OIDDAS (orphelins) ?
- existe-t-il des utilisateurs donnés par OIDDAS introuvables dans Apogée (problème de connexion) ?
- existe-t-il des utilisateurs dans Apogée étant en service, mais dont le référentiel signale qu'ils n'appartiennent pas ou plus à la catégorie du personnel ?
- existe-t-il des individus dans Apogée étant en service, mais dont le référentiel signale qu'ils ne sont pas inscrits dans le forum ?

Le rapport d'exécution de ce programme a donné les résultats suivants :

```
nb employes en service: 2337
nb employes inscrits au forum: 311
nombre de users oiddas: 956

nb config apogee dans oiddas : 565
```

```
nb config apotest dans oiddas : 165
nb utilisateurs dans prp5 : 893 dont 637 en service
nb utilisateurs dans apotest : 874 dont 623 en service
nb utilisateurs ayant un pb de connexion dans Apogee : 21
nb utilisateurs ayant un pb de connexion dans Apotest : 12
nb utilisateurs Apogee a desinscrire : 277
nb utilisateurs Apotest a desinscrire : 59
nb utilisateurs Apogee a inscrire dans le forum: 20
nb utilisateurs Apogee a desinscrire dans le forum: 48
nb de comptes Apogee orphelins (sans login): 176
nb de comptes Apotest orphelins (sans login): 511
```

Le grand nombre d'anomalies constatées amène les remarques suivantes :

- Les responsables de scolarité ne signalent pas toujours qu'un utilisateur est sorti de leurs effectifs. Il reste donc en service dans Apogée et dans le forum. Mais son login est sorti de l'annuaire ce qui fait qu'il ne peut plus se connecter à l'ENT et donc à l'application. Et dans le référentiel il a changé de catégorie, devenant « visiteur ».
- Les écarts entre Apogée et Apotest s'expliquent par le fait que nombre d'utilisateurs n'utilisent que leur configuration Apogée, et n'ont pas activé la configuration Apotest.
- Sur le formulaire d'accès à Apogée, il est possible de supprimer une configuration et beaucoup d'utilisateurs suppriment par erreur leur accès à Apogée. Ils essaient de la rétablir avec des identifiants qui sont faux, ce qui crée des problèmes de connexion.
- Il n'est pas souhaitable avec les listes ainsi éditées de faire faire le ménage dans la base de données par l'équipe SAUVES. Il faut que les responsables de scolarité demandent les suppressions comme ils demandent les créations.

D'autres curiosités ont été relevées, comme par exemple plusieurs personnes (login) se servant du même compte utilisateur Apogée. C'est par exemple le cas des vacataires qui utilisent des comptes VAC-1, VAC-2, etc. Chaque année les vacataires sont différents, mais les comptes utilisés sont les mêmes. Mais c'est aussi Madame B qui utilise le compte de Monsieur A dont elle a pris la succession, sans en informer les gestionnaires d'Apogée.

3.4.5. La recherche des règles de gestion

Si le processus existant est assez clair et semble-t-il facile à reproduire, il apparaît pourtant que la principale difficulté apparaît au moment de choisir le type utilisateur de la personne habilitée. Parmi les 75 types existants, lesquels ne sont jamais utilisés ? Lesquels sont largement utilisés ? Lesquels sont utilisés par des personnes qui ne sont plus dans l'université ? Une fois éliminés les types inutilisés il restait encore beaucoup de types (54).

Certains se ressemblent beaucoup, ne diffèrent que d'un détail. L'enquête a fait ressortir les faits suivants :

- d'une UFR à une autre, la même fonction au sein de l'université amène à des types différents.
- dans le même service, on souhaite que sur deux personnes faisant pratiquement la même chose, l'une ne puisse voir ce que fait l'autre.
- certains postes n'ont besoin que de consulter les informations, mais la personne concernée ayant eu dans le passé le pouvoir de modification accepte mal d'y renoncer.
- les responsables de scolarité perdent la notion même de type utilisateur et demandent seulement « le même type que M. ou Mme Untel ».
- les personnes qui travaillent dans les scolarités ont pris l'habitude d'avoir accès à certains modules qu'elles n'utilisent pas, mais se dépêchent de protester si cet accès disparaît.

Il est donc apparu essentiel de pouvoir identifier « les métiers » dans l'utilisation d'Apogée, et de pouvoir les associer à un type utilisateur adéquat. Ce travail relève essentiellement des compétences de la scolarité centrale et il est souhaitable qu'elle soit associée à cet effort de structuration.

3.4.6. Les contraintes du système d'information

Certaines contraintes sont imposées par des choix pris au niveau de l'université tels que :

- l'application doit être accessible par le web, sur le portail de l'ENT. Le portail de l'ENT est un portail ESUP-Portail s'appuyant sur uPortal.
- l'authentification sur toute application accessible par l'ENT doit passer par le système SSO (Single Sign On ou authentification unique) sur le serveur CAS (Central Authentification Service) de l'université. Le Central Authentification Service est un système SSO pour le web développé par l'université Yale, partenaire majeur dans le développement d'uPortal.
    - Lorsqu'un client s'authentifie le serveur CAS crée un cookie sur le navigateur accessible par lui seul.

- Lorsque une application web est lancée, elle s'adresse au serveur CAS qui lui délivre un ticket et l'identifiant de la personne. Dès que le ticket est périmé (déconnexion ou temps dépassé), il faut obtenir un nouveau ticket ce qui oblige la personne à s'authentifier à nouveau.

UPortal, CAS et autres sont portés par Jasig, un consortium d'universités pour le développement d'outils pour l'enseignement supérieur en open source. En France, le consortium ESUP-Portail porte le projet ENT sur les solutions de Jasig (uPortal, CAS, Ldap)[8].

- L'application doit être installée sur des serveurs Apache Tomcat fonctionnant sous Linux dont un est réservé aux web services.
- Les tables propres à l'application font partie de l'espace de nom d'Apogée sur Oracle.

3.4.7. La sécurité du système d'information

3.4.7.1. La protection des adresses

La question de la sécurité est prise en charge par un autre service, le service RESYST, qui gère les différents droits accordés aux utilisateurs. Cette question échappe donc à mon périmètre. Il suffira d'appliquer les règles communes à toutes les applications.

Comme toute application métier, APOBILITATION n'est accessible qu'après authentification. Le système SSO-CAS est fiable et n'a encore connu aucune faille à ce jour, à moins qu'un utilisateur (ou pire, un informaticien), ne divulgue son login et mot de passe.

Il n'est pas possible de l'attaquer par injection de SQL puisque le système interroge un annuaire LDAP et non la base de données. En effet le principe de cette attaque repose sur le présupposé que l'application va lancer une requête SQL après la saisie du login qui pourra être piratée, or ici ce n'est pas le cas puisqu'un annuaire Ldap ne s'interroge pas avec des ordres SQL.

Les url des applications mises en production ne doivent jamais être communiquées sous la forme *http:// <nom du serveur> :<port>/monapplication*.

[8] Voir (Esup Portail, 2004)

Le service RESYST attribue à l'application une url du type : *http://app.parisdescartes.fr/monapplication* qui masque les informations du serveur physique. L'accès direct à cette adresse doit renvoyer sur le serveur CAS si l'utilisateur n'est pas authentifié.

Le portail ENT effectue une nouvelle transposition de l'URL et celle-ci prend la forme : *https://ent.univ-paris5.fr/render.userLayoutRootNode.uP?uP_sparam=activeTab&activeTab=8&uP_root=nxxx.*

Enfin les services web ne sont accessibles que par les applications installées sur les serveurs de l'université.

3.4.7.2. La protection des mots de passe

Deux catégories de mots de passe sont envisagées, se confondant parfois : les mots de passe personnels (celui du login, celui d'Apogée), et les mots de passe des applications qui doivent se connecter aux annuaires LDAP et aux bases de données.

Le mot de passe du login est modifiable à tout moment par son possesseur. Les applications qui interrogent l'annuaire LDAP n'ont pas les droits suffisants pour lire ces mots de passe. Seul le serveur CAS peut le faire. En cas de perte il doit être changé. Par sécurité le service Annuaire a donné des droits restreints aux informaticiens et aux applications qui interrogent l'annuaire, en cas de divulgation accidentelle.

Un utilisateur d'Apogée ne connaît pas et ne doit pas connaître son mot de passe Apogée depuis qu'il accède à Apogée avec son login de l'ENT. Ce mot de passe Apogée est attribué par l'administrateur au moment de la création de son compte. Si ce mot de passe est oublié ou perdu il faut en recréer un autre. C'est un mot de passe Oracle, aucune lecture ne peut en être faite.

Le système OIDDAS a pour objet de stocker ce mot de passe pour pouvoir retrouver le compte Apogée attaché au login de l'utilisateur. Il faut avoir les droits d'administrateur pour renseigner OIDDAS. Très peu de personnes ont les droits permettant de voir en clair cette

information. L'application Oracle Forms « Forms Web Access » qui lance Apogée dans le portail ENT[9] a ce droit.

Les mots de passe nécessaires aux développeurs pour développer se communiquent de développeur à développeur dans la plus grande discrétion. Ils sont renouvelés de temps en temps.

3.5. LA PROPOSITION D'UNE SOLUTION D'ENSEMBLE

3.5.1. Le nouveau processus envisagé

Dans le nouveau processus on souhaite que le responsable de scolarité renseigne toutes les informations nécessaires à la création ou la modification d'un utilisateur Apogée, les autres informations existantes étant récupérées dans le système d'information. Les informations sensibles ou trop techniques pour être décidées par un responsable de scolarité auront une valeur par défaut définie dans des fichiers de paramètres. Le nouveau processus de création ressemble donc à ceci :

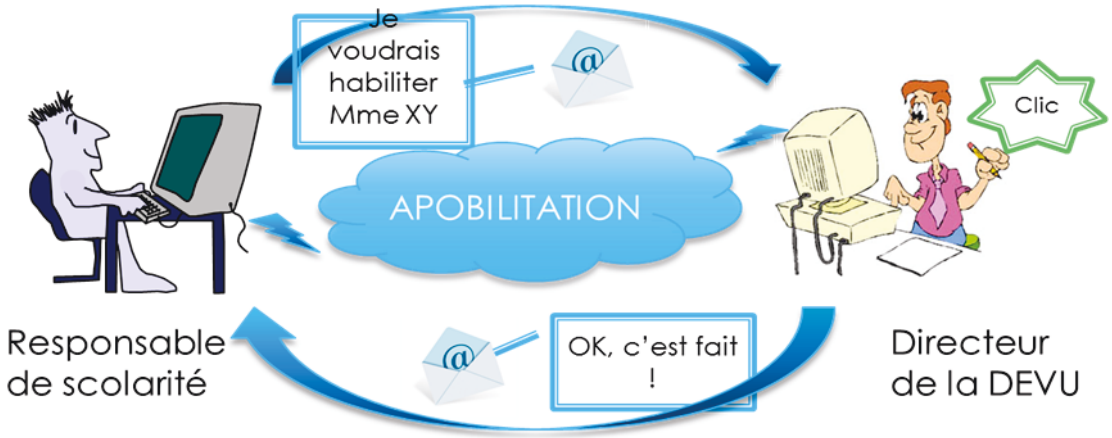


Figure 31 Nouveau processus de création
Source MC Collas

1. Le responsable de scolarité choisit la personne à habiliter parmi le personnel de sa composante.
2. Automatiquement un email est envoyé au directeur de la DEVU qui va se connecter sur l'application à son tour.
3. Le directeur de la DEVU donne son accord et automatiquement les différents systèmes (Apogée, OIDDAS, forum) sont mis à jour.

[9] Voir Figure 21 Web Form Access d'Apogée

4. Dès que l'opération est terminée un email est envoyé automatiquement au responsable de la scolarité pour l'informer du succès de l'opération.

Le processus de suppression ou modification est presque semblable :

1. Le responsable de scolarité dispose de la liste des personnes de sa composante qui sont habilitées. Il peut de lui-même constater quelles sont les personnes qui doivent être supprimées de cette liste (départs, mutations). Il peut aussi constater des anomalies dans les données. En choisissant une personne de cette liste, il fait une demande de suppression ou de modification.
2. Cette demande est envoyée au directeur de la DEVU qui la valide et qui la fait exécuter, et le responsable de scolarité est informé du succès de l'opération.

Ce processus nous amène directement à identifier les acteurs :

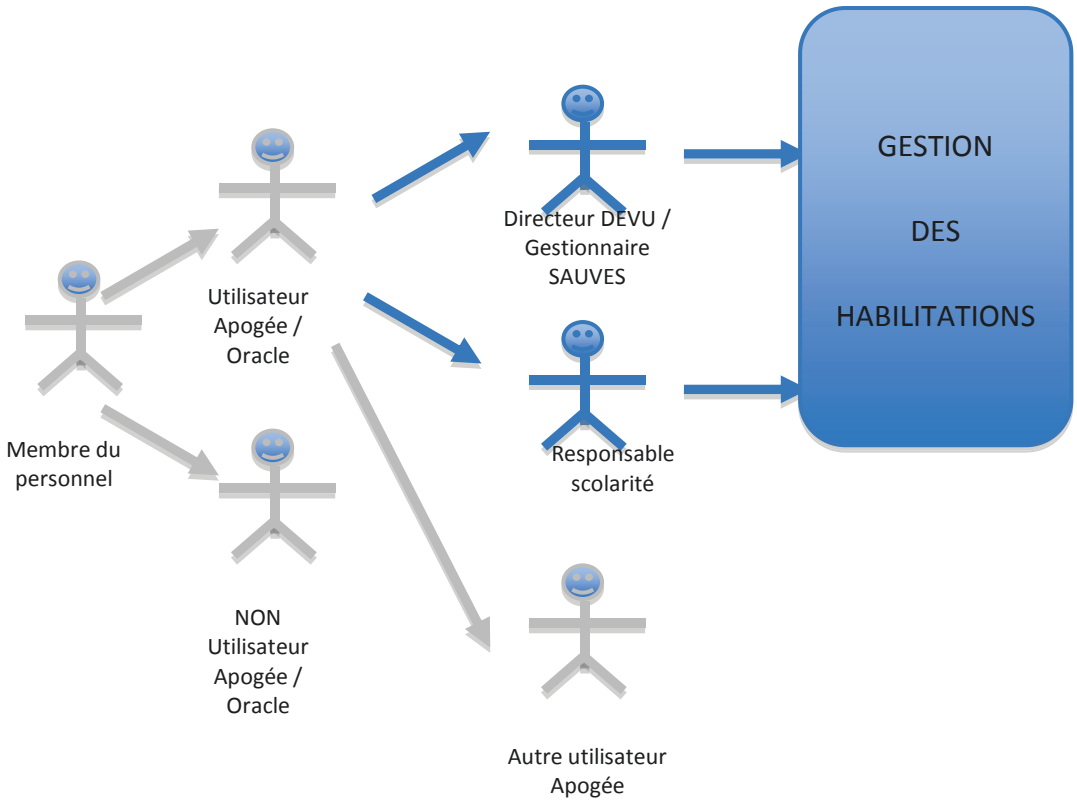


Figure 32 Acteurs
Source MC COLLAS

Et il est possible de définir les cas d'utilisation :

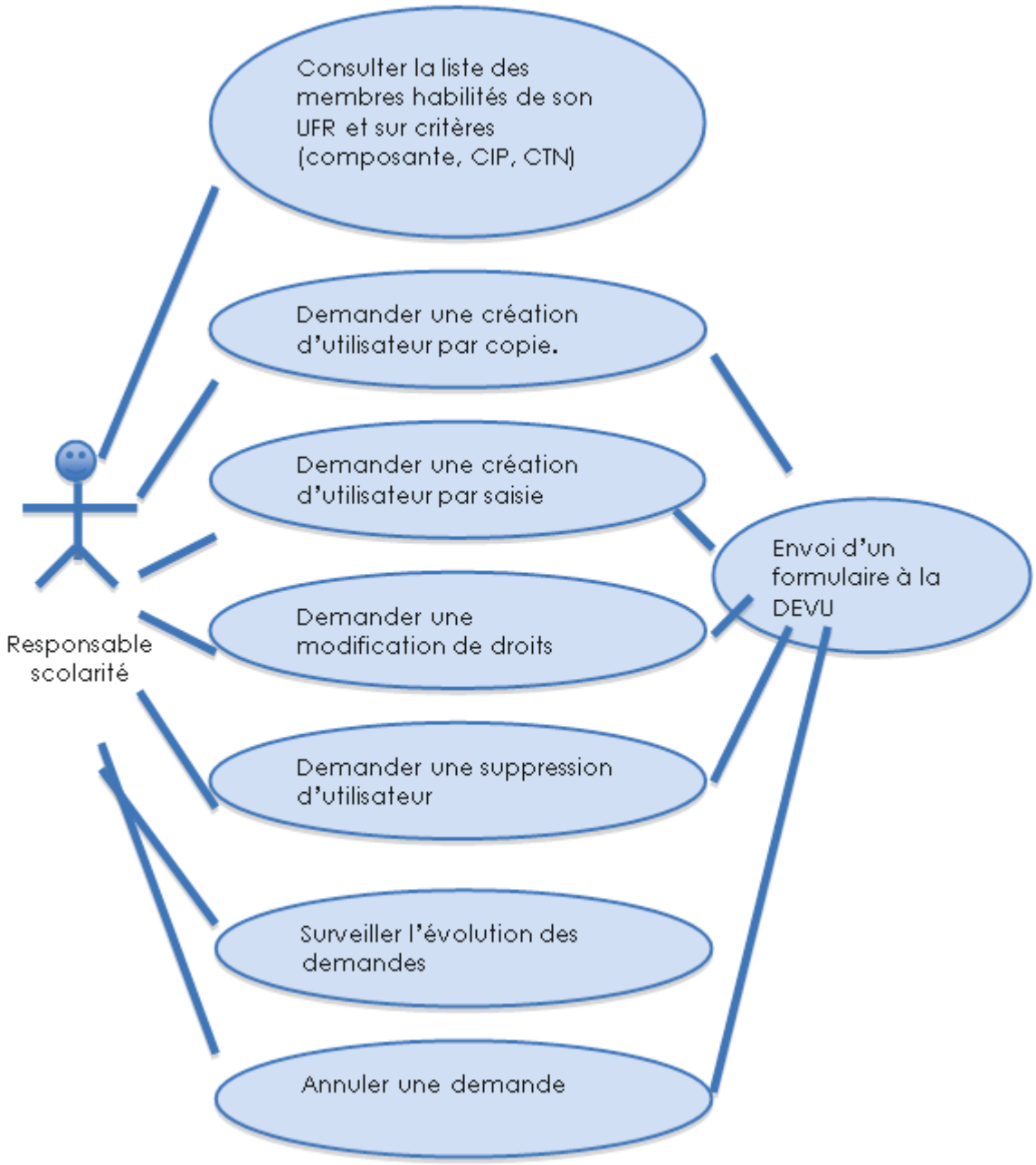


Figure 33 Cas d'utilisation des responsables de scolarité
Source MC COLLAS

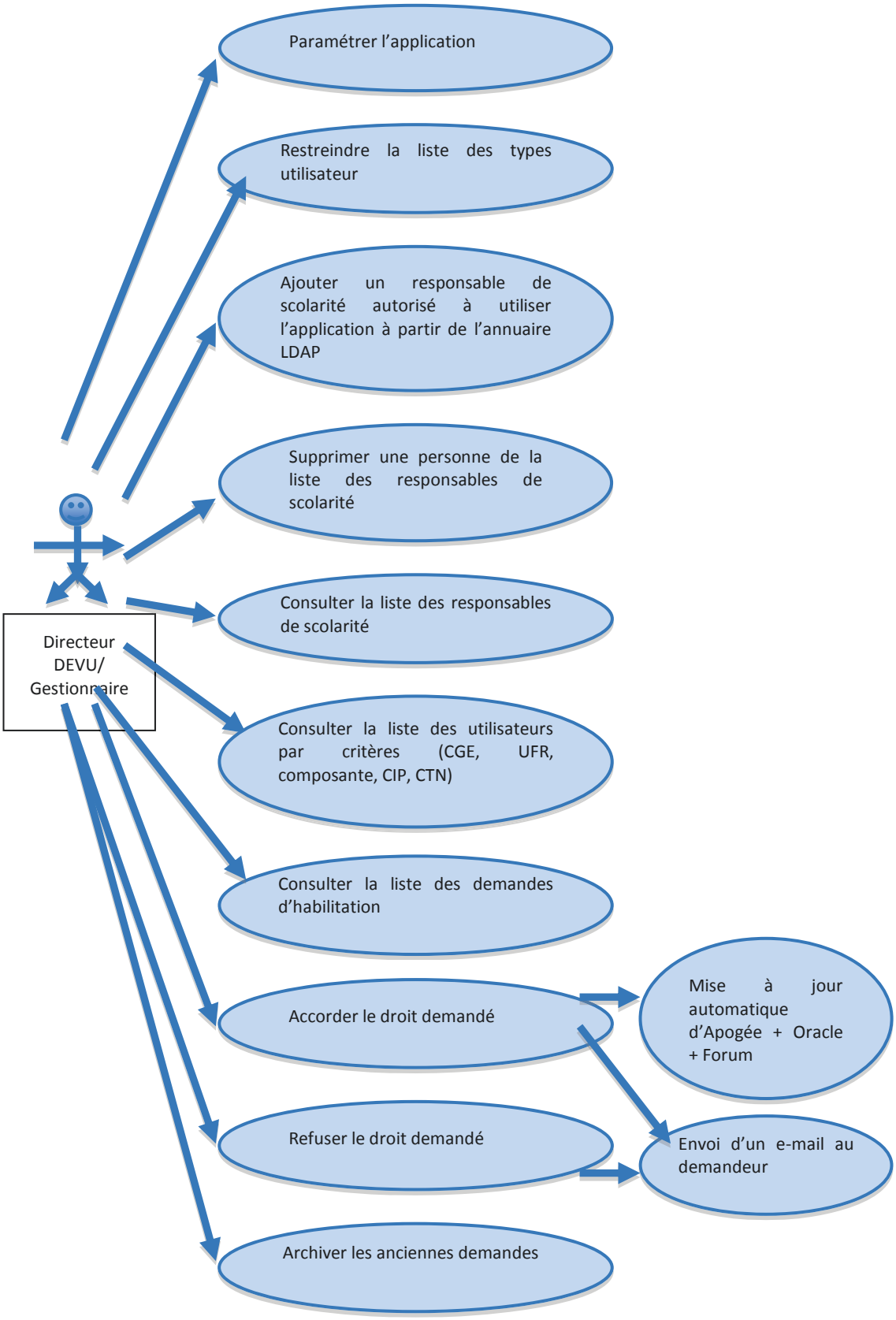


Figure 34 Cas d'utilisation des administrateurs
Source MC COLLAS

3.5.2. Une architecture générale basée sur le recours à un service web

Depuis longtemps, il est question du recours à une nouvelle version d'Apogée dont la version actuelle est vieillissante. Une exigence apparaît pour que les opérations de mise à jour d'Apogée, Grhum, et OIDDAS soient isolées sous forme de recours à un service web pour pouvoir plus facilement les modifier.

Par ailleurs Apogée propose toute une série de services web permettant d'obtenir les données en lecture, mais aucun ne correspond aux besoins de l'application. Donc pour la lecture d'Apogée on aurait le choix entre une lecture classique en table, ou l'écriture d'un service web spécifique. Les questions de performance font pencher en faveur d'une lecture directe.

Le recours au service web n'est pas la seule solution. L'utilisation de vues pouvant être modifiées facilement en est une autre, dans le cas de la lecture.

D'emblée il est apparu qu'une difficulté se pose à cause de l'absence d'une table de jointure entre le login de l'ENT et le code utilisateur Apogée. Jusqu'à présent, il faut passer par l'annuaire OIDDAS. Or il n'est pas possible de faire des requêtes complexes mettant en œuvre à la fois des tables Oracle et des annuaires Ldap. Il faudrait que toutes les informations soient sous forme de tables Oracle.

Pour ce faire, il faudra créer cette table de jointure. En faisant cela, on crée une redondance d'information qui risque donc d'être en décalage. Ce décalage n'est pas très fréquent dans le temps, car les opérations de cette nature ne sont ni très fréquentes, ni réalisées par un grand nombre de personnes. Toutefois, le décalage est possible. Il sera donc nécessaire de faire des opérations de synchronisation. Cette table sera appelée HAB_LOGIN_UTI.

La répartition des rôles entre l'application web et le service web serait donc la suivante :

Tableau I Répartition des tâches entre application cliente et service web
Source MC COLLAS

Application web	Web service
Création et mise à jour des tables propres à l'application	Création de la table HAB_LOGIN_UTI à partir d'OIDDAS et Apogée
Gestion des demandes de création d'utilisateurs Apogée. Mise à jour des tables de demandes	Exécution des demandes de création dans Apogée et mise à jour d'Apogée, OIDDAS, Grhum, HAB_LOGIN_UTI
Appel ponctuel de synchronisation	Synchronisation régulière de HAB_LOGIN_UTI à partir d'OIDDAS et Apogée
Appel des listes d'anomalie	Recherche d'anomalies sur les tables

3.5.2.1. Quelles seraient les tables propres à l'application ?

Puisque nous voulons créer un utilisateur Apogée ou le modifier, non en temps réel, mais en deux temps, l'idée générale est de créer une demande contenant l'utilisateur Apogée modifié. Cette demande serait lue par le service web qui extrairait les données de l'utilisateur et exécuterait la mise à jour.

On a donc : une demande = un numéro de demande + un utilisateur + des informations utiles au traitement (table HAB_DEM_UTI).

Il se trouve qu'un utilisateur Apogée (table UTILISATEURS) contient des listes de composantes (table UTI_CMP), de centres de gestion de stage (table UTI_CGS) et de centres de traitement (table UTI_COLLECTER_CTN) attachés dans Apogée. Donc de la même façon des listes de composantes, de centres de gestion de stage et de centres de traitement sont attachées à la demande concernant un utilisateur (tables HAB_DEM_UTI_CMP, HAB_DEM_UTI_CGS, HAB_DEM_UTI_CTN).

Il a été dit avec insistance que la notion de type utilisateur est complexe, mal comprise et utilisée par des non-informaticiens. Le souhait des responsables SAUVES est de masquer la complexité du type utilisateur en le simplifiant. A la place il sera proposé un « profil » d'utilisateur dont le libellé sera évocateur et correspondra à une fonction comme « responsable de scolarité » ou « secrétaire pédagogique », ou « vacataire d'inscriptions administrative », etc…

Pour chacun de ces profils on choisira un type utilisateur, par exemple « tous les droits sur une UFR » pour un responsable de scolarité, et un certain nombre de valeurs par défaut dans les données difficiles à expliquer. Ces profils seront gérés dans l'application par un utilisateur habilité à administrer l'application, en particulier le responsable de la scolarité centrale (table HAB_PROFIL).

Ces profils dépendent donc directement des types utilisateurs, mais nous avons besoin de données supplémentaires qui ne figurent pas dans la table TYP_UTI d'Apogée. Ces données sont d'une part un témoin qui dit si on laisse le droit ou non d'utiliser ce type utilisateur (rappelons-nous que nous souhaitons en réduire le nombre), d'autre part, un résumé des droits compris dans ce type, ce qui est une information utile pour aider un responsable de scolarité à choisir. Ainsi une table HAB_TYP_UTI vient étendre celle d'Apogée.

L'examen de la base de données d'Apogée a montré que certains comptes d'Apogée sont propres au logiciel d'Apogée et ne doivent même pas être visibles ou pris en compte dans des rapports d'anomalie. Une table HAB_UTI_EXCLUS en fait la liste.

Enfin il est nécessaire d'avoir une table des utilisateurs de cette application, qui permette de restreindre l'accès et de définir les rôles de chacun. Ce sera la table HAB_USERS.

3.5.2.2. Les états de la demande

Dans les informations utiles au traitement véhiculées dans la table HAB_DEM_UTI se trouve le statut de la demande. Celui-ci prendra les valeurs suivantes :

- EC pour une demande en cours, qui n'est pas encore acceptée ni refusée, et qui peut donc être modifiée
- V comme validée, qui n'est plus modifiable mais peut être consultée. Elle peut donc être exécutée.
- R comme refusée par l'administrateur, qui n'est ni modifiable, ni exécutable, mais archivable.
- X comme exécutée par le web service, qui n'est pas modifiable, mais consultable et archivable.
- F (Failed) comme en échec au moment de l'exécution, qui n'est pas modifiable mais peut-être à nouveau validée ou refusée pour une nouvelle tentative.

Le témoin d'archivage permet de signaler les demandes dont l'affichage à écran n'est plus nécessaire.

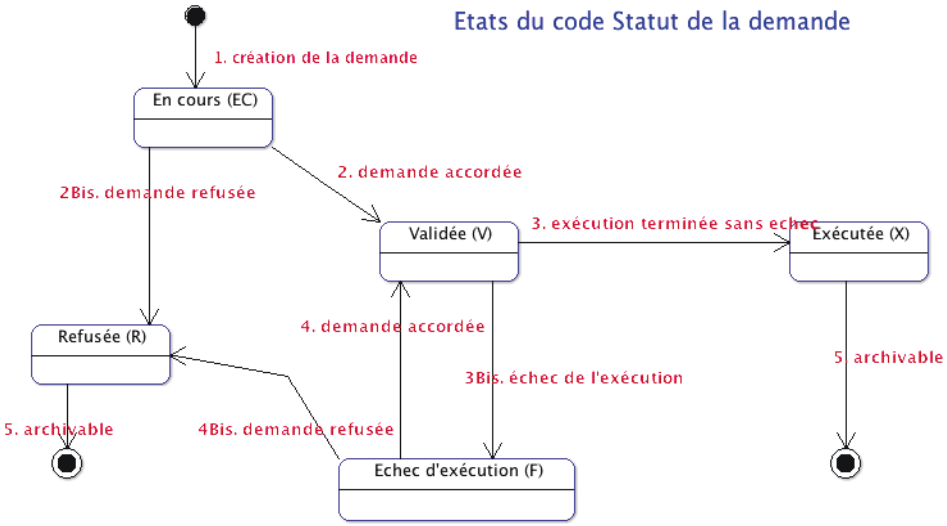


Figure 35 Les états de la demande
Source MC COLLAS

3.5.3. La maquette des écrans

Des maquettes d'écran ont été réalisées en html afin de permettre au responsable de la DEVU de visualiser l'application web. Les informations exemples étaient figées et le design non définitif. Voici quelques-unes des vues proposées.

La page d'authentification habituelle de l'ENT de Paris Descartes :

Figure 36 Authentification par l'ENT
Source : Université Paris Descartes

Ci-dessous les pages du responsable de scolarité d'une composante :

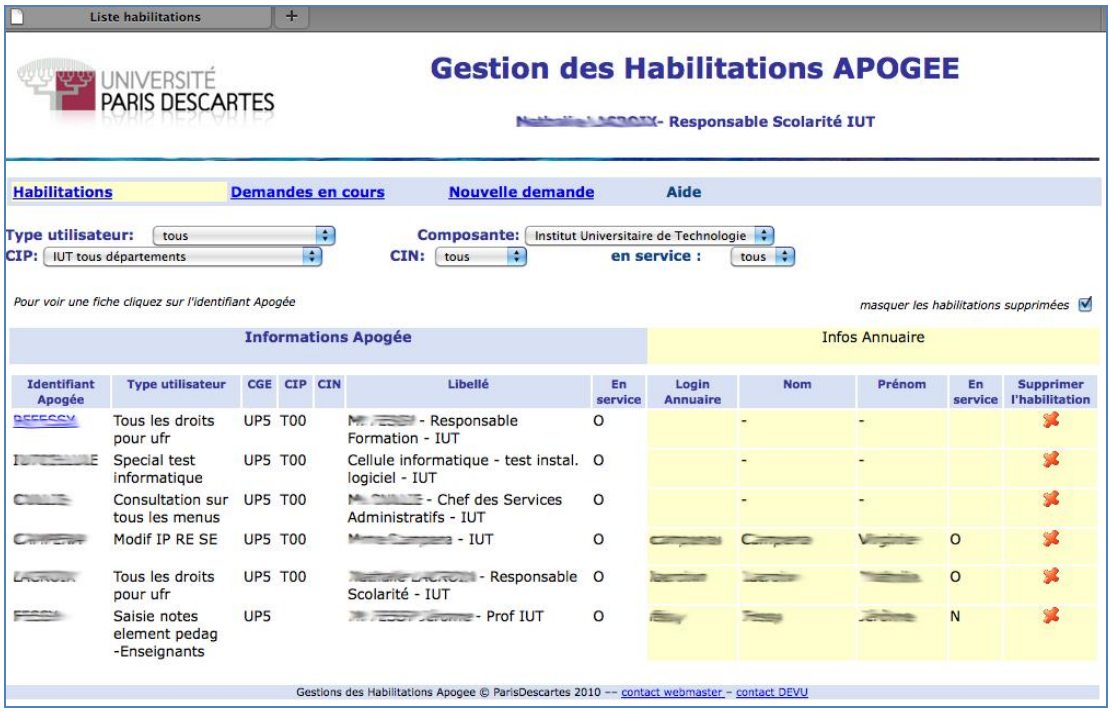


Figure 37 Maquette: page liste des habilitations (responsable de scolarité)
Source MC COLLAS

Le responsable de scolarité peut voir les personnes habilitées de sa composante et voir si elles ont un login dans l'annuaire. Cet écran montre une anomalie : les personnes qui n'ont pas de login ne devraient plus être en service dans Apogée. Les responsables de scolarité vont s'en apercevoir et demander la suppression. Ils peuvent visualiser la fiche complète de l'utilisateur.

Figure 38 Fiche de l'utilisateur dans Apogée
Source MC COLLAS

Figure 39 Maquette: recherche d'un personnel dans l'annuaire
Source : MC COLLAS

Le responsable de scolarité peut chercher dans l'annuaire le personnel qu'il souhaite faire habiliter.

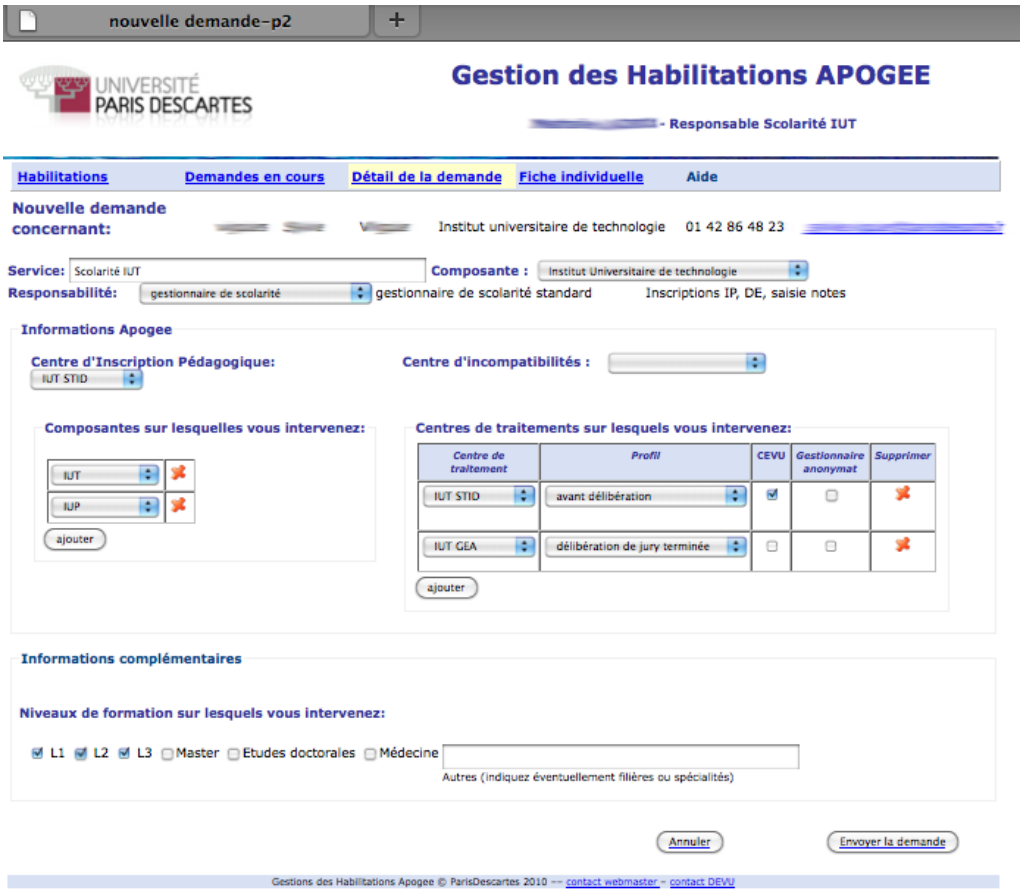


Figure 40 Maquette: formulaire de demande d'une habilitation
Source : MC COLLAS

Un formulaire permet de renseigner les données nécessaires.

Figure 41 Maquette: liste des demandes d'habilitations émises par le responsable de scolarité
Source : MC COLLAS

Une fois créée la demande se rajoute à la liste existante. Il est nécessaire de connaître le statut de la demande.

L'administrateur de l'application (directeur de la DEVU, service SAUVES) dispose de pages avec des informations supplémentaires et de pages de paramétrage de l'application. Sa liste des habilitations est identique mais sa portée couvre toute l'université plutôt qu'une composante.

Figure 42 Maquette: liste des demandes (responsable DEVU)
Source : MC COLLAS

Seul celui qui a un droit de validation (le directeur de la DEVU) dispose des colonnes « accepter » et « refuser ». C'est le droit attaché à sa fonction. Il peut déléguer ce droit à d'autres personnes de la DEVU.

Les administrateurs de l'application disposent d'un accès qui leur permet de gérer les paramètres de l'application : liste des utilisateurs autorisés à utiliser l'application, liste des types utilisateurs autorisés, liste des profils d'utilisateurs.

3.5.4. Des règles de gestion étoffées

L'application veut contraindre les utilisateurs d'Apogée en limitant les possibilités qu'offre le logiciel Apogée. Elle veut standardiser la manière de créer un nouvel utilisateur dans Apogée, tout en tenant compte des pratiques actuelles de l'université. La liste de ces règles est la suivante :

- On veut désormais que les nouveaux utilisateurs d'Apogée aient un code utilisateur d'Apogée identique au login de l'ENT, mais en majuscules.
- Une exception à cette règle est faite pour les vacataires qui utilisent des codes utilisateurs tels qu'INS1, INS2, INS3, ou VAC01, VAC02, etc… On veut pouvoir garder ces comptes et juste rattacher un login différent le temps du contrat du vacataire. Il s'agit généralement des vacataires qui aident au moment des inscriptions et qui ont des contrats d'un mois en moyenne.
- Le login est une information de nature confidentielle et on veut que les administrateurs puissent le voir, mais pas les responsables de scolarité
- On veut que le type utilisateur soit masqué aux responsables de scolarité qui n'en maitrisent pas la complexité, et qu'ils n'aient accès qu'aux profils qui leur sont destinés. Certains profils ne pourront être utilisés que par les administrateurs.
- On veut que les utilisateurs puissent avoir accès à Apogée en version de test, comme auparavant, en particulier les nouveaux qui suivent une formation à l'utilisation d'Apogée. Pour cela on veut que lorsqu'une création / modification / suppression d'un utilisateur est faite dans Apogée, cela soit répercuté dans Apotest. L'inverse n'est pas vrai : les opérations effectuées dans Apotest ne sont pas répercutées dans Apogée.
- En ce qui concerne la gestion du mot de passe, autrefois le service SAUVES attribuait un mot de passe associé au compte Oracle qui est aussi le code utilisateur. Lorsqu'un utilisateur appuyait par mégarde ou ignorance sur la croix rouge du formulaire d'accès à Apogée, la configuration passait du statut d'activée à activable et pour l'activer à nouveau il devait rentrer son code utilisateur et son mot de passe. Pour connaître ce dernier il s'adressait au service SAUVES. Or on ne veut pas qu'il connaisse ce mot de passe. On souhaite que l'utilisateur ait toujours accès à une configuration activée et non activable, de façon qu'il ne connaisse pas le mot de passe.
- Régulièrement le service SAUVES copie la base Apogée sur Apotest, il doit en être tenu compte.
- L'université n'utilise pas les centres de gestion de stage, le répertoire d'export, l'imprimante A3.
- Une difficulté réside dans la notion de composante : dans Apogée un utilisateur peut gérer une ou plusieurs composantes, sans en faire partie forcément. Les personnes qui travaillent au siège peuvent avoir une visibilité sur plusieurs voire toutes les

composantes. Il existe une composante abstraite dont le code est « UNI » qui signifie « toute l'université ».

- Le code CIP est rattaché à une ou plusieurs composantes. Les personnes qui travaillent au siège et qui veulent avoir une vision de toute l'université, utilisent le code CIP « U01 » qui est lié à toutes les composantes. Les autres codes CIP sont normalement attachés à une composante et cette composante peut avoir plusieurs codes CIP.
- Le code CIP « K01 » présente une singularité : il s'agit du Service Commun de Formation Continue (SCFC). Ses membres peuvent être dispersés dans plusieurs autres services. Il ne représente pas une composante. On veut que le responsable de ce CIP puisse utiliser l'application comme un responsable de scolarité, mais aucun code composante ne peut lui correspondre sauf « UNI » et le CIP « K01 ». Il fera l'objet d'un traitement adapté.
- On souhaite que chaque responsable de scolarité ne voit sur son écran que les personnes qui font partie de sa composante, mais la difficulté est de savoir comment obtenir ce filtre puisque :
    - On ne connaît pas la composante de l'utilisateur Apogée, seulement les composantes qu'il gère.
    - Tous les utilisateurs ne sont pas forcément rattachés à un login, le lien est rompu pour ceux qui sont partis, et plusieurs logins peuvent être attachés à un code utilisateur générique ou non.
    - Même en enlevant de la liste d'un responsable d'une scolarité de composante les utilisateurs qui ont un code CIP « U01 », on aura une cote mal taillée, car certains ont la valeur du CIP nulle.
    - Si les données seront plus propres une fois créées avec le nouveau logiciel, il restera quand même les utilisateurs hors service.

3.5.5. Le dossier de conception générale

Le dossier de conception générale contient les éléments suivants :

Figure 43 Sommaire du dossier de conception générale
Source : MC COLLAS

Ce document est accessible aux gestionnaires et utilisateurs d'Apogée. Le formalisme UML est utilisé pour les acteurs et les cas d'utilisation. Plus de complexité est apportée au dictionnaire des données qui reprend les tables existantes ou à créer, ainsi que la description des annuaires Ldap. Le dossier de conception générale et les maquettes d'écran ont servi de base de discussion lors des réunions avec la DEVU. Les notes plus techniques sont restées en réserve pour le dossier de spécifications détaillées.

Au moment de la rédaction du dossier de conception détaillée un certain nombre d'éléments n'étaient pas très clairs et se sont affinés pendant la réalisation elle-même. Par exemple le cas particulier du service commun de la formation continue (SCFC), que l'on doit traiter comme une composante alors qu'il n'est qu'un des services du siège de l'université, n'a été vu qu'après, lorsque les pages web donnant les listes d'utilisateur ont été écrites. A ce moment-

là plusieurs cas de figure qui n'avaient pas été décelés ont été découverts grâce à ces listes. On a vu aussi certains abus tels que l'usage inapproprié du code Apogée d'un collègue, ce qui amenait des doublons inattendus. Naturellement la conception a évolué pour intégrer ces nouvelles informations, sans que l'édifice ne soit remis en cause. De la même façon certaines attentes de la responsable du service SAUVES ont pu être affinées une fois la première version de test disponible.

D'autres aspects n'apparaissent pas dans le dossier de conception générale. Toute la réflexion sur l'histoire de la gestion des identités à l'université Paris-Descartes a été faite a posteriori, avec le recul donné par la réalisation d'Apobilitation.

3.6. CONCLUSION

La conception générale est donc une activité intellectuelle qui se mène une première fois de façon initiale. Puis elle revient de façon itérative lorsque débute la rédaction des spécifications détaillées qui peut remettre en cause ou faire évoluer certains aspects, et lors de la réalisation lorsqu'elle se bute sur la réalité technique, et enfin lors de la validation lorsqu'il y a évolution des attentes. Dans ces cas le dossier est remis à jour dans une nouvelle version.

Dans la mesure où la même personne réalise le projet entièrement : conception, réalisation, vérification, et où les personnes qui font la recette de l'application sont proches géographiquement, cela ne pose pas de problème et présente un facteur de souplesse intéressant.

Cette conception générale a été discutée et validée avec les responsables de la DEVU, avant d'aller plus loin.

La rédaction de ce dossier a été d'un grand secours lors de la réalisation qui est arrivée presque un an après. Le dossier a été utile pour remettre en mémoire rapidement et relancer le travail.

Chapitre 4. Les spécifications détaillées

4.1. INTRODUCTION

L'étape de spécifications détaillées commence en principe lorsque le document de conception générale a été approuvé par l'utilisateur client. En pratique, rédiger des spécifications détaillées permet de clarifier les idées et de se poser de bonnes questions aussi le document de spécification détaillée a été ouvert avant que ne soit finalisé le document de conception générale.

4.2. LA REDACTION DU DOSSIER DE CONCEPTION DETAILLEE

Voici le sommaire du dossier :

Figure 44 Sommaire du dossier de conception détaillée
Source : MC COLLAS

Dans le document de conception détaillée qui n'est pas destiné à l'utilisateur mais à celui qui va réaliser, on utilisera les outils fournis par UML ou par MERISE pour décrire les interactions

entre les différentes parties du système. Le document s'appuie sur les autres documents cités en référence.

4.3. LES ACTEURS

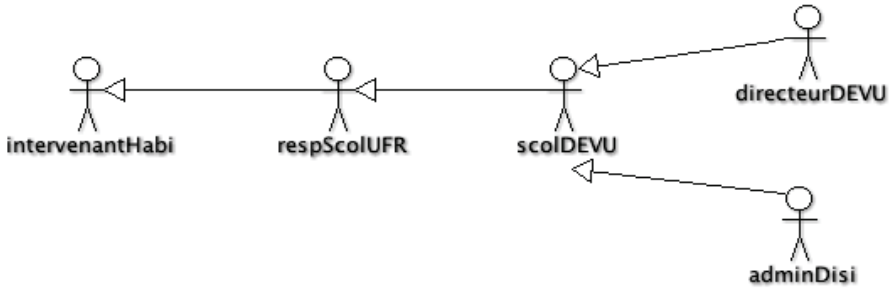


Figure 45 Schéma UML des acteurs
Source : MC COLLAS

L'intervenant qui utilise l'application est au minimum un responsable de scolarité qui peut gérer toute une UFR. S'il appartient au siège il peut être membre de la scolarité centrale ou de la DISI et avoir une vue sur l'ensemble de l'université. Certaines personnes de la DEVU ou de la DISI possèdent des droits d'administrateur pour gérer les paramétrages. Enfin le directeur de la DEVU possède le droit de valider les demandes d'habilitation.

4.4. LE PROCESSUS D'HABILITATION

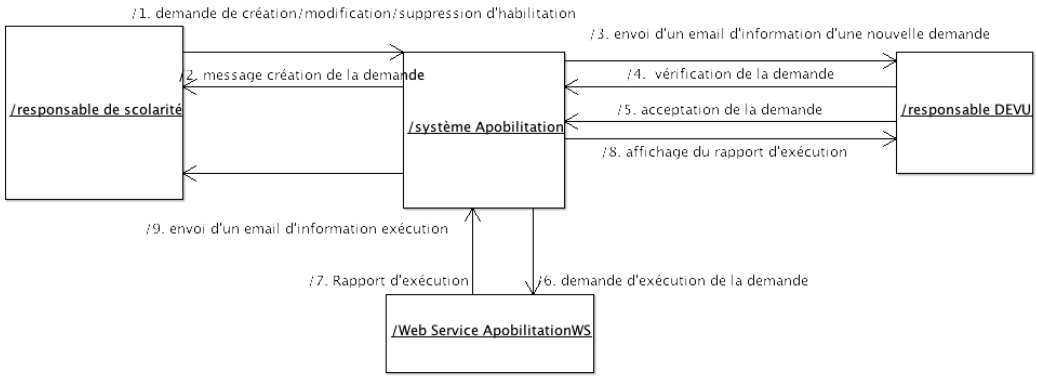


Figure 46 Le nouveau processus d'habilitation
Source : MC COLLAS

Alors que la Figure 31 Nouveau processus de création (p 57) présentait ce nouveau processus comme une boîte noire mettant en évidence la disparition de l'intermédiaire SAUVES, la figure ci-dessus (Figure 46 Le nouveau processus d'habilitation) montre ce que fait en interne l'application : l'application orchestre les différentes étapes du processus en s'assurant que

chaque étape se passe dans l'ordre logique et que chaque acteur sache où en est le déroulement du processus et intervienne à son tour.

4.5. LES CAS D'UTILISATION

La liste des cas est la suivante, la présentation graphique a été présentée dans les figures suivantes : Figure 33 Cas d'utilisation des responsables de scolarité (p59), Figure 34 Cas d'utilisation des administrateurs (p 60).

4.5.1. Les cas d'utilisation de tous les intervenants

L'intervenant veut :

1. s'identifier
2. voir la liste des personnes habilitées Apogée de sa composante
3. voir le détail d'une habilitation

4.5.2. Les cas d'utilisation des responsables de scolarité UFR

Le responsable de scolarité veut :

1. déshabiliter une personne qui a quitté son service
2. demander une habilitation pour une nouvelle personne
3. modifier les caractéristiques d'une personne habilitée
4. suivre sa demande
5. annuler une demande en cours

4.5.3. Les cas d'utilisation du directeur de la DEVU

Le directeur de la DEVU veut :

1. voir la liste de toutes les personnes habilitées du CGE
2. déshabiliter une personne qui a quitté le CGE
3. demander une habilitation pour une nouvelle personne du CGE
4. modifier les caractéristiques d'une personne habilitée
5. voir l'ensemble des demandes d'habilitation
6. accepter une demande
7. refuser une demande
8. autoriser un responsable de scolarité ou autre à utiliser le système de gestion des habilitations (paramétrage autorisations)
9. modifier la liste des types utilisateurs autorisés (paramétrage type utilisateurs)
10. modifier la liste des profils dans Apogée (paramétrage profils)

4.5.4. Les cas d'utilisation des responsables de scolarité centrale de la DEVU

Les mêmes que le directeur de la DEVU sauf 6 et 7.

4.5.5. Les cas d'utilisation de l'administrateur DISI

L'informaticien de la DISI veut en plus :

1. installer l'application
2. migrer les données la première fois (éventuellement recommencer)
3. s'assurer de la cohérence des données vis à vis d'autres systèmes externes
4. gérer les anomalies constatées

4.6. DESCRIPTION TEXTUELLE DES CAS D'UTILISATION

Chacun des cas d'utilisation est décrit comme l'exemple ci-dessous, de façon textuelle.

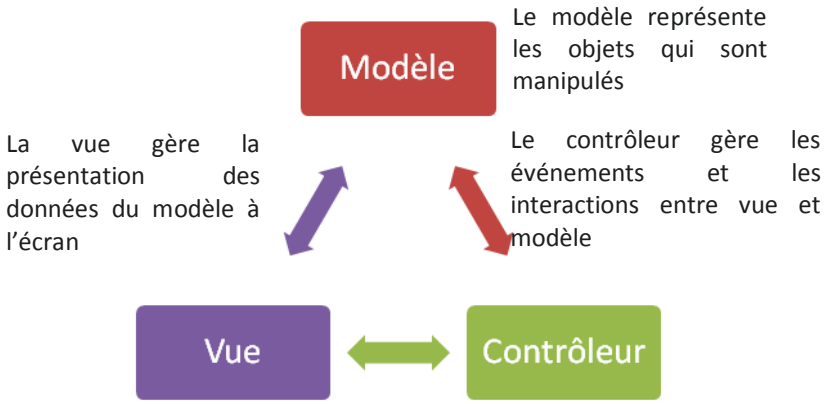
Tableau II Exemple de cas d'utilisation textuel

Cas d'utilisation	A1 S'identifier
Acteur principal	L'intervenant
Acteurs secondaires	Le système d'authentification CAS
Objectif	Utiliser le système Apobilitation
Pré conditions	Avoir un login et un mot de passe L'application a été correctement installée
Post conditions	Les en-têtes et options de menus reflètent le profil de l'intervenant : responsable de scolarité, personnel DEVU, directeur DEVU, admin DISI
Scénario nominal	1. L'intervenant appelle l'application 2. Il entre son login et son mot de passe ENT 3. Le système CAS vérifie qu'il est habilité à Paris Descartes 4. Le système vérifie qu'il fait partie de la liste des personnes autorisées à utiliser l'application 5. Le système vérifie les droits et en déduit à quel type d'utilisateur il a affaire. Le système affiche ou masque les options de menu selon les droits 6 Le système affiche la page Liste des utilisateurs par défaut et les menus. Son nom et/ou login apparaît d'ans l'en-tête
Scénario d'exception	E1. Le login ou le mot de passe ne sont pas corrects, l'intervenant est refoulé par le CAS. (non gérée par l'application mais par le CAS) E2. L'intervenant s'est bien authentifié mais il n'est pas autorisé à utiliser l'application. Un message d'erreur lui est adressé.
Scénarios alternatifs	A1. L'utilisateur s'est authentifié avec succès par l'ENT indépendamment de l'application Il accède directement à l'application qui reprend le scénario nominal au point 3 A2. L'intervenant accède directement dans l'application et constate que la session a été ouverte avec un autre login. Il clique sur le bouton déconnexion pour invalider la session et reprendre au début.
Spécification particulière	Le scénario A2 est surtout rencontré par les informaticiens en période de test.

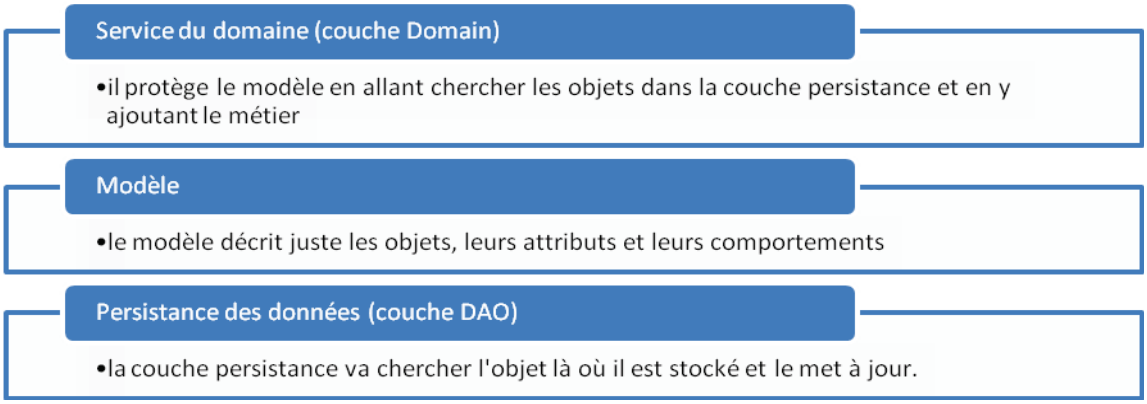
L'utilisation de grilles de description des scénarios nominaux ou d'exception est une technique très efficace pour décrire le fonctionnement du système. Lors de la validation et des tests, il suffit de reprendre ces fiches pour obtenir des fiches de tests correspondantes. 22 fiches ont été décrites dans le dossier de spécifications détaillées.

4.7. UNE ARCHITECTURE EN COUCHES

Les pratiques actuelles du développement d'applications pour le web recommandent une architecture en couches selon le schéma MVC (Modèle – Vue –Contrôleur).



Une autre pratique qui s'est imposée au niveau du modèle est celle qui sépare le modèle en 3 couches :



Enfin il faut rajouter une ou plusieurs couches de services :

- Services d'accès aux LDAP
- Service d'accès SSO
- Service d'accès aux web services

Nous avons donc besoin d'une architecture ressemblant à ceci

4.8. LE MODELE DES DONNEES

Le modèle des données est représenté sous forme simplifiée page suivante (Figure 47 Les classes du modèle). Les objets représentés en rose correspondent à des objets se trouvant dans Apogée et utilisés en lecture seule. De même les objets bleus correspondent à des entités se trouvant dans le référentiel GRHUM en lecture seule aussi. Les objets en jaune sont les objets créés spécialement pour répondre aux besoins de notre système.

Tous ces objets sont simplifiés dans le sens que seules les clés primaires et étrangères pour les accès aux tables sont notées de façon explicite. Ces objets contiennent en outre comme attributs supplémentaires les informations contenues dans les colonnes des tables. Ces attributs ne sont pas systématiquement mentionnés pour ne pas alourdir le schéma.

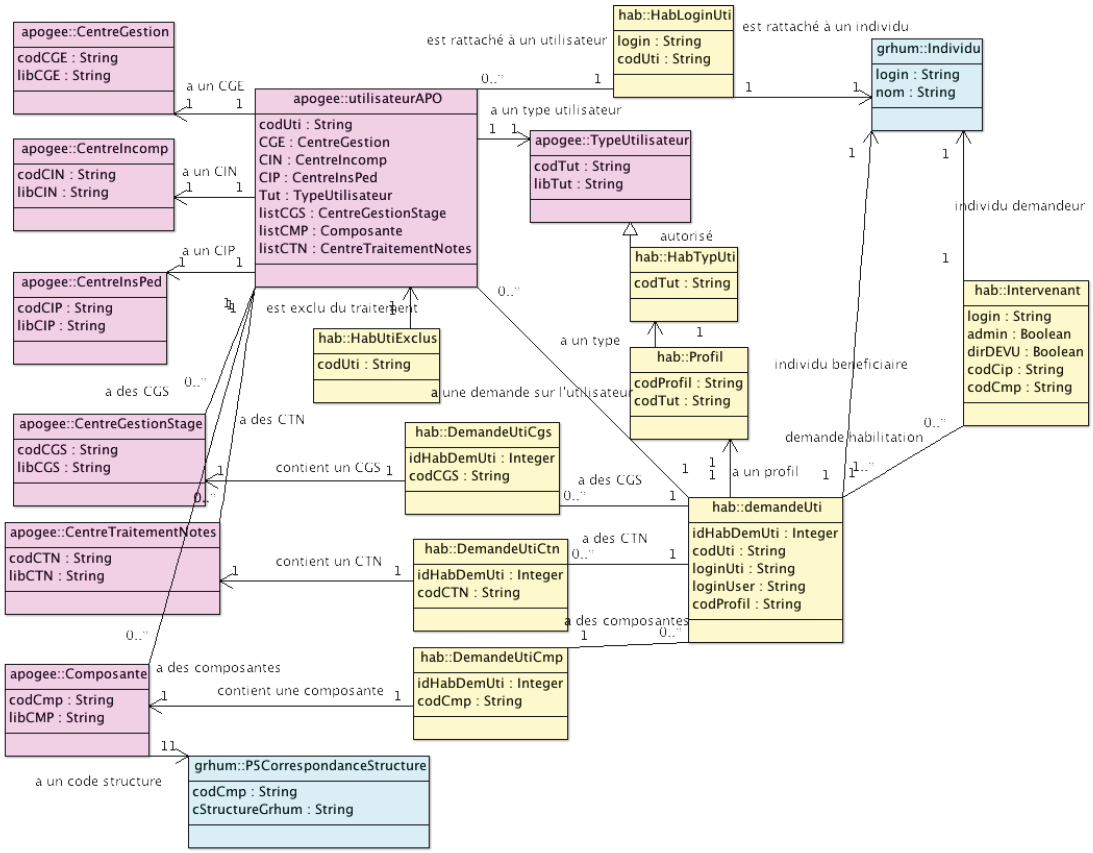


Figure 47 Les classes du modèle
Source : MC COLLAS

Ce modèle fait apparaître la logique de la conception : puisque l'objet utilisateur d'Apogée contient des codes renvoyant à des tables de libellés d'une part et trois collections de composantes, centre de traitement des notes et centres de gestion des stages, alors l'objet DemandeUti de notre application va contenir un objet utilisateur avec ses trois collections. En plus de cela il va aussi véhiculer d'autres informations relatives à l'individu (login) et à la demande elle-même (numéro de demande, demandeur, validateur, profil, dates, statut, etc…).

La logique de l'application a aussi besoin d'étendre le type utilisateur en y ajoutant des informations supplémentaires, et de créer l'objet profil qui masque la complexité du type utilisateur.

Puisque nous souhaitons déléguer à un service web la modification des tables qui sont dans Apogée, OIDDAS et Grhum, le modèle des données du service web diffère assez peu pour la partie mise à jour du modèle. L'objet DemandeUti est envoyé au Service web qui se charge de la mise à jour et renvoie des informations sur l'exécution.

En revanche la synchronisation a besoin du modèle suivant. La synchronisation a besoin de créer la première fois la table HAB_LOGIN_UTI, et ensuite de la mettre à jour pour intégrer des modifications survenues dans Apogée et OIDDAS et provenant d'autres applications.

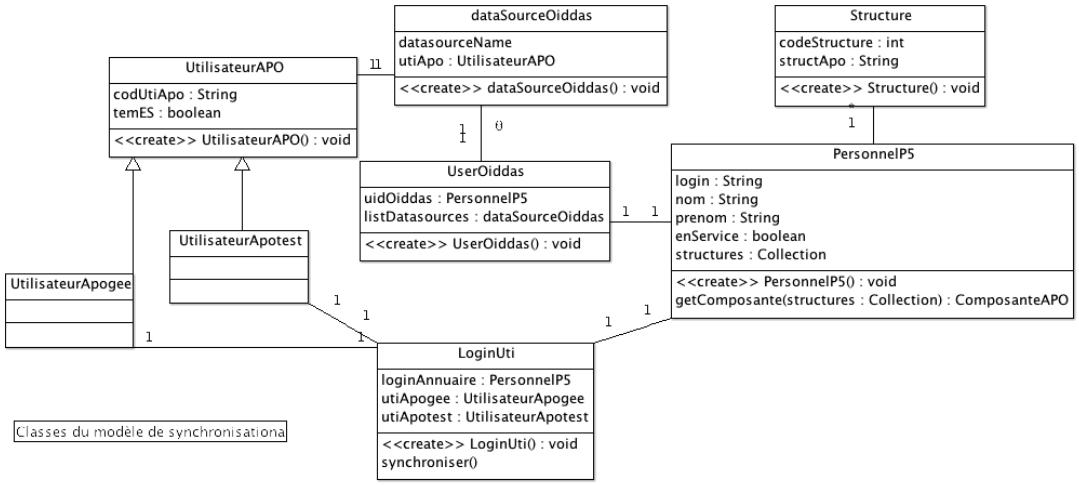


Figure 48 Classes du modèle du service web
Source : MC COLLAS

Pourquoi créer une table HAB_LOGIN_UTI qui double OIDDAS ? Doubler ainsi volontairement les informations est-il une bonne idée ? La réponse est que nous nous serions bien passés de cette solution, mais que pour le moment nous ne savons pas obtenir une table offrant le code apogée de l'utilisateur et le ou les logins qui l'utilisent en faisant des jointures sur des tables Oracle et un LDAP dans la même requête. Pour pouvoir faire des requêtes en SQL qui ont besoin de faire cette jointure nous devons avoir cette table.

Une fois cette table créée pouvons-nous nous passer d'OIDDAS ? En pratique OIDDAS est utilisé de façon symétrique pour les accès au logiciel Harpège et donc doit rester en service. De plus lorsqu'un utilisateur appelle Apogée, un formulaire géré par OIDDAS Management apparait avant l'affichage du logiciel Apogée. Cela fait partie de la solution donnée par l'AMUE pour l'accès à Apogée par le Web.

4.9. LA PERSISTANCE DES DONNEES

4.9.1. La cohérence des données et les cas d'anomalies

Un utilisateur d'Apogée correctement inscrit doit :

- Avoir un compte Apogée, ce qui correspond à un « user » dans le sens d'Oracle
- Avoir un utilisateur Apogée avec un témoin « en service »

- Avoir un login valide dans le Ldap et dans le référentiel
- Avoir dans OIDDAS pour ce login une ressource Apogée (PRP5) et éventuellement une ressource Apotest
- Etre inscrit dans le forum des utilisateurs d'Apogée

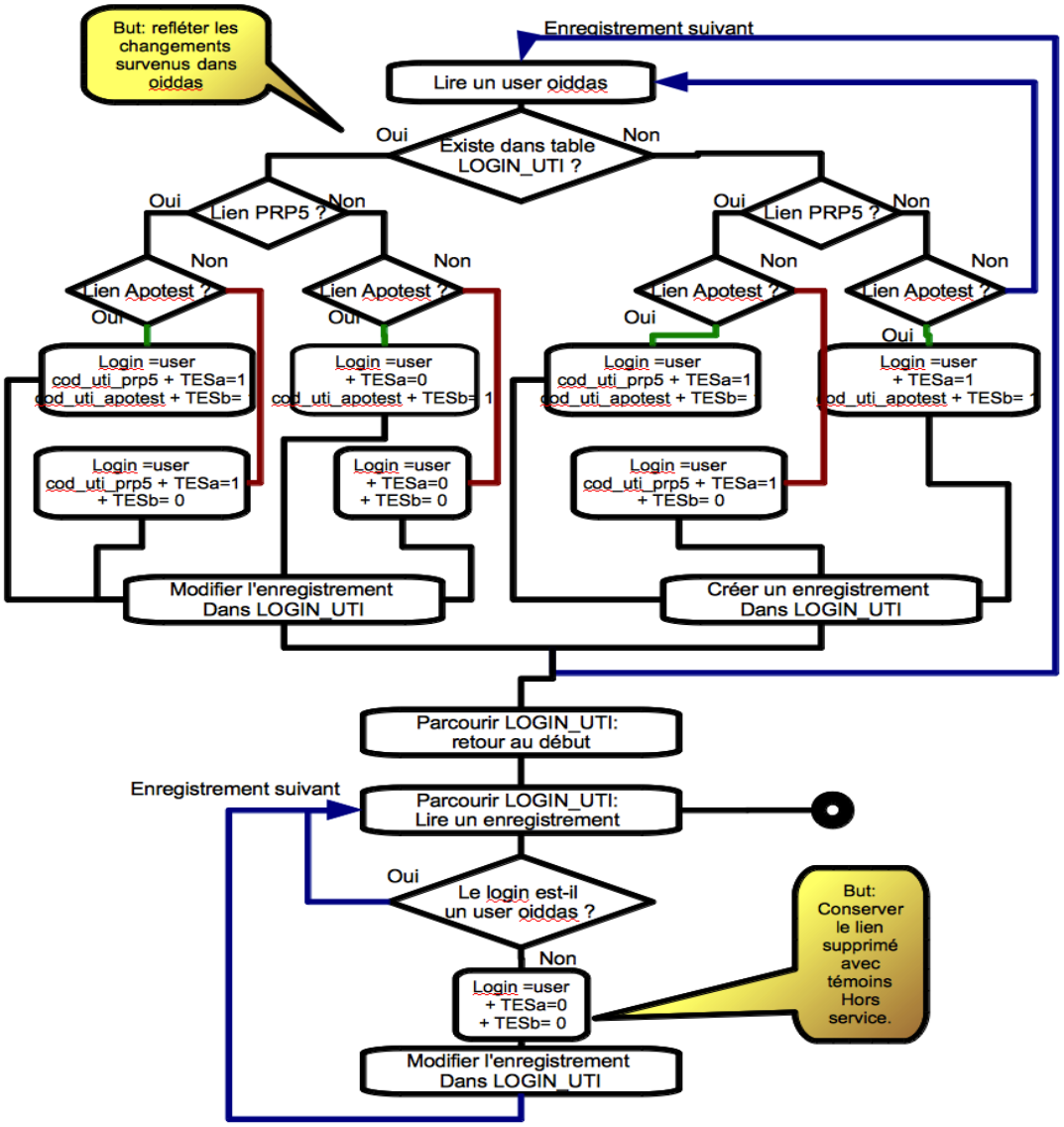
Un utilisateur correctement désinscrit d'Apogée doit :

- Avoir un compte Apogée avec un utilisateur dont le témoin est « hors service »
- Si le login est encore dans OIDDAS, les ressources apogée (PRP5) et Apotest ne doivent plus être rattachées
- Il ne doit plus être inscrit dans le forum des utilisateurs d'apogée.

Tous les autres cas sont des anomalies. Une table de décision a été constituée pour déceler les anomalies, qui sont assez nombreuses au commencement de l'étude (voir en annexe 6)

4.9.2. La synchronisation des données en batch

La Figure 49 La synchronisation des données en batch suivante représente l'organigramme du traitement batch de synchronisation. Ce traitement doit être lancé au moins une fois par jour au démarrage des serveurs après l'arrêt de la nuit. Il pourra être lancé à la demande en appelant le traitement sous forme de web service, pour un rafraichissement des données supplémentaire. Comme dit précédemment, ce traitement est nécessaire pour intégrer des événements extérieurs à l'application.



Traitement batch de synchronisation Oiddas vers HAB_LOGIN_UTI

Figure 49 La synchronisation des données en batch
Source : MC COLLAS

4.9.3. La synchronisation des données en temps réel

Les événements qui sont gérés par l'application doivent être mis à jour en temps réel afin qu'ils soient reflétés immédiatement lorsque le demandeur d'une création / modification / suppression voudra vérifier son exécution sur la page des habilitations. La Figure 50 Le traitement d'une demande par le rend compte du traitement.

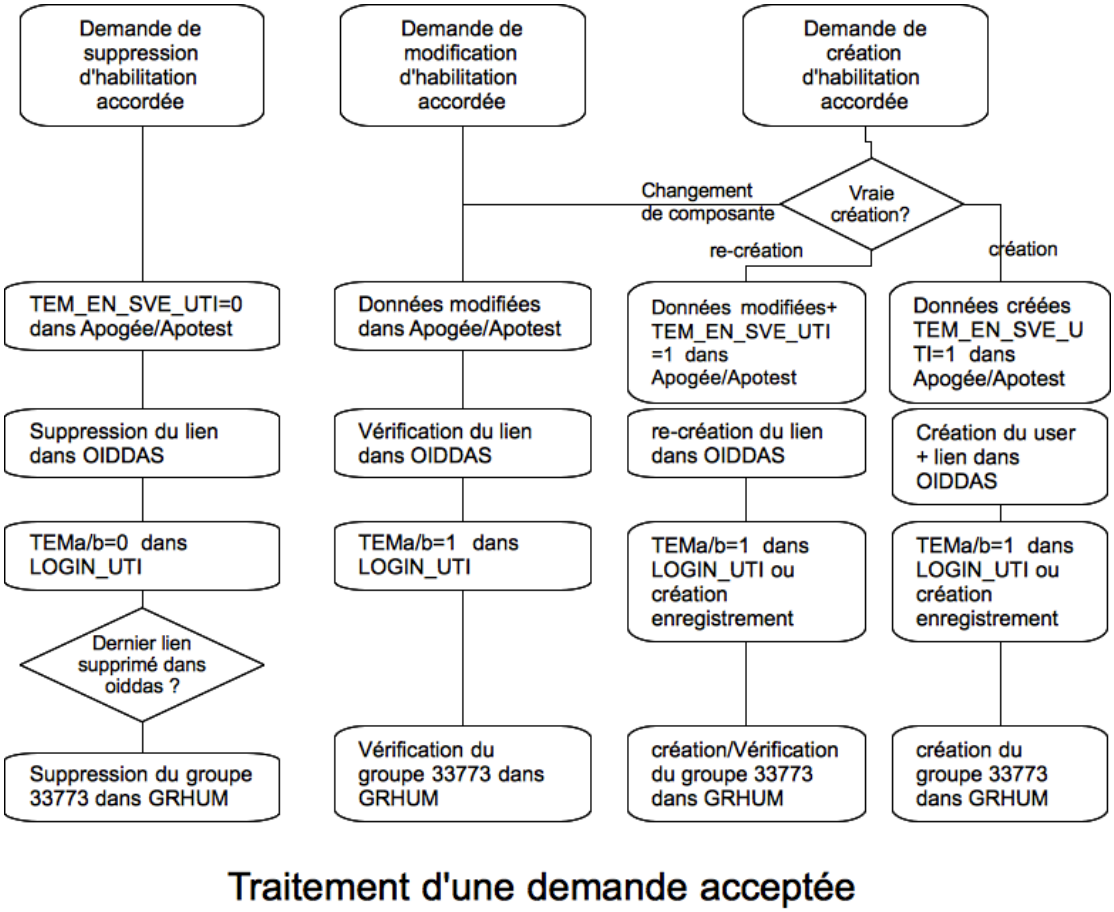


Figure 50 Le traitement d'une demande par le service web
Source : MC COLLAS

4.10. LA GESTION DES MOTS DE PASSE

On souhaite que l'utilisateur d'Apogée ne connaisse qu'un seul mot de passe, celui qui est attaché à son login et qu'il est seul à connaître. Les anciens utilisateurs d'Apogée, du temps où celui-ci s'utilisait en mode client-serveur connaissaient leur compte Apogée et le mot de passe associé. Par la suite, le service SAUVES en utilisant OIDDAS affectait un mot de passe, toujours le même par facilité, parce que l'utilisateur n'avait plus besoin de le connaitre.

L'application APOBILITATION consulte OIDDAS pour connaître le mot de passe du compte Oracle s'il existe, et vérifie s'il est valide. S'il n'est pas valide il est alors forcé avec un nouveau mot de passe.

Avec Apobilitation, il est apparu que donner toujours le même mot de passe représentait une faille de sécurité car il pouvait "s'éventer", mais qu'il n'était pas utile de conserver toujours le même. En cas de perte ou déconnexion, il suffit de demander à APOBILITATION de refaire une connexion. A partir de là, il a été décidé que le mot de passe serait généré à la volée de façon aléatoire.

4.11. LE CHOIX DES OUTILS ET LE FRAMEWORK ESUP-COMMONS

Le choix des outils de réalisation n'a pas fait l'objet d'un long débat. Au contraire, une solution s'est imposée d'elle-même. L'université Paris-Descartes a développé des projets pour le web dans les langages suivants : java, Web Object (java pour Apple), PHP, Oracle Forms. Mes compétences en java m'ont conduite naturellement à choisir cet outil de développement.

Au moment de la conception je venais de terminer un projet d'intégration d'une application réalisée avec le framework esup-commons qui utilise les outils suivants : java pour le code, JSF pour les vues, XML pour les paramétrages, Spring pour l'architecture, Hibernate pour la persistance, le framework[10] dispose de services pour accéder au Ldap, à SSO pour l'authentification sur un serveur CAS.

Il semble que ce soit plus rentable d'utiliser les compétences existantes que de chercher à repartir à zéro. Mais esup-commons répondra-t-il à tous les besoins ?

Qu'est-ce qu'esup-commons ?

Esup-commons[11] est un framework de développement conçu et maintenu par le consortium Esup-Portail[12]. Ce consortium est porteur du projet ENT, s'appuyant sur le portail uPortal et l'authentification CAS du consortium Jasig, sur l'annuaire Ldap Supann. Le consortium a pour but de fournir des outils communs à toutes les universités, et l'université Paris-Descartes a

[10] Framework : voir §1.3.7 et la note
[11] Présentation d'esup-commons (Renater (CRU), 2007a)
[12] Présentation d'Esup-Portail sur www.esup-portail.org (Esup Portail, 2010) et (Wikipedia contributors, 2012)

adopté Esup-Portail tout en ayant largement contribué aux solutions d'annuaire SUPANN et de SSO[13]. Esup-Portail favorise donc l'utilisation d'esup-commons pour la réalisation de projets partagés ou non entre universités.

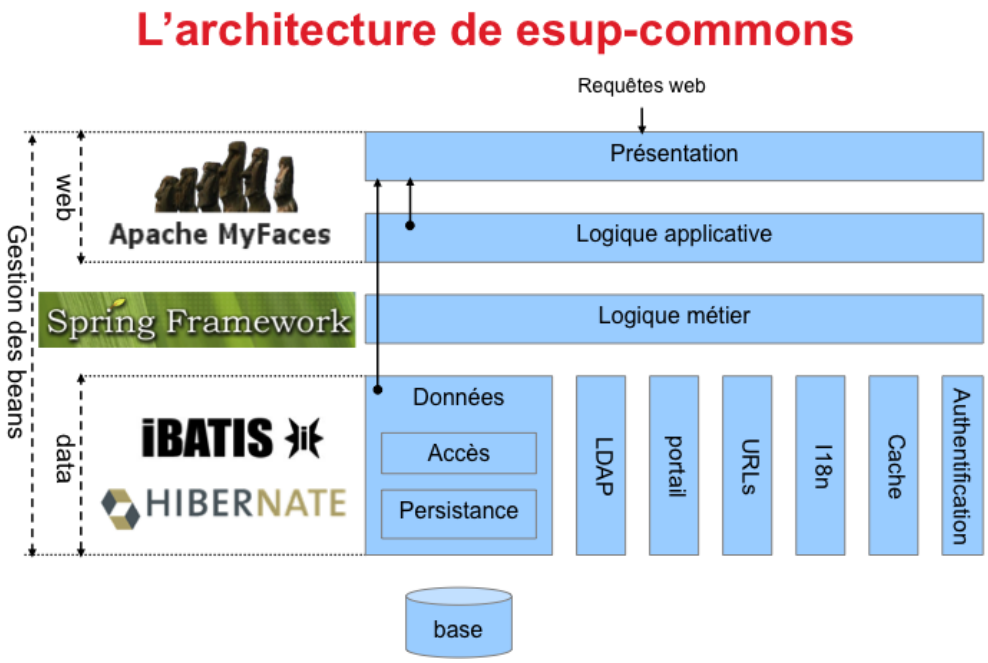


Figure 51 Architecture d'esup-commons v1
Source Esup-Portail : Guide du développeur Esup-commons v1 (Esup Portail *et al.*, 2007)

L'architecture d'esup-commons convient à l'architecture souhaitée dans la conception.

4.12. CONCLUSION

Nous avons donc défini une application web appelant un service web dédié, et devant accéder à un service SSO, des annuaires Ldap, des bases de données Oracle. L'application web sera une application java web. Elle utilisera le framework esup-commons qui lui-même est construit sur les briques JSF, Spring, Hibernate, Ant, etc…

[13] Après avoir mis en place un premier annuaire LDAP à l'université dès 2000, Mme Lagarde (Paris V) en fait une présentation qui retient l'attention des partenaires. En 2003 et les années suivantes elle fait partie du groupe de travail Supann supervisé par le CRU (Comité Réseau des universités) qui livrera Supann v1 en 2003 (CRU, 2003). Dans le même temps M. Vatré (Paris V) met en place le système SSO à l'université dès 2002 et en 2003 participe aux travaux de l'AAS (Authentification, autorisations et SSO), interuniversitaire sous l'égide du CRU et du ministère de la jeunesse, de l'éducation nationale et de la recherche (Esup Portail, 2003).

Chapitre 5. Mise en œuvre du framework esup-commons

5.1. INTRODUCTION

Une partie de la réalisation a été faite dans la foulée du travail de conception, à savoir l'installation de l'environnement de développement, la réalisation de la mise à jour des tables de paramétrage « types utilisateurs » et « profils » en s'appuyant sur l'expérience acquise sur un autre projet « esup-rendezvous ». La réalisation a été suspendue pour permettre le développement d'autres projets, ce qui a renforcé l'expertise sur les différents outils, puis le travail a repris jusqu'à la fin. Le temps de réalisation est aussi un temps d'apprentissage pour comprendre et maîtriser les différents outils proposés. Le framework esup-commons permet de développer sans avoir forcément la maîtrise de toutes les briques qui composent ce framework, ce qui permet de gagner du temps en avançant dans le développement et en comprenant les subtilités de chaque technologie employée au fur et à mesure de la réalisation.

5.2. L'INSTALLATION DE L'ENVIRONNEMENT DE DEVELOPPEMENT

5.2.1. Les bases de données en test

Il est nécessaire d'avoir des bases de tests pour le temps de réalisation. Mais il n'est pas toujours possible d'avoir un environnement complet de test.

5.2.1.1. Apotest : la problématique de cette base mi- test, mi- production

Apogée a une version de production (PRP5) et une version de test (Apotest). Cette version de test est une copie de la base de production réalisée environ chaque trimestre. Cette base de test sert à la fois aux développeurs pour leurs essais et pour les formations à l'utilisation d'Apogée. Ensuite, les utilisateurs continuent d'y avoir accès s'ils le souhaitent.

Or on veut que lorsqu'un nouvel utilisateur est créé en production, il soit aussi créé en test. La solution imaginée pour tenir compte de ce cas particulier est la suivante : nous avons deux configurations prp5 et Apotest que le programme connaîtra sous les noms config1 et config2. Le programme travaille seulement sur la config1 sauf dans les cas de figure où il doit tenir

compte de la config2. Pendant le développement et les tests la config1 sera Apotest et la config2 sera Apotest, faute de mieux.

5.2.1.2. Grhum

GRHUM peut être accédé de 2 façons : soit directement avec des identifiants propres à Grhum qui permettent de lire et d'écrire. Soit avec les identifiants d'accès à apogée et un dB Link sur Grhum qui permettent de réaliser des requêtes avec jointure sur les deux bases. La première solution a été retenue pour le service web et la deuxième pour l'application cliente du web service. Il existe une base de test de GRHUM qui est mise à jour toutes les nuits et très peu différente de la base de production. C'est celle-ci qui est utilisée en version de test.

5.2.1.3. Les annuaires LDAP

L'annuaire Ldap n'a pas de version de test. Il existe cependant un étudiant de test et un professeur de test dont les identifiants sont acceptés par l'authentification. Nous n'utilisons cet annuaire qu'en lecture. Donc c'est cet annuaire qui est utilisé en test et en production. Il sert à l'authentification. L'annuaire OIDDAS n'a pas de version de test non plus. En test on ne considère que la config 2 (Apotest).

5.2.1.4. L'outil Oracle SQL Developer

L'Outil SQL Developer fourni par Oracle permet d'explorer les tables des bases de données et d'utiliser des ordres SQL directement et de tester les requêtes SQL.

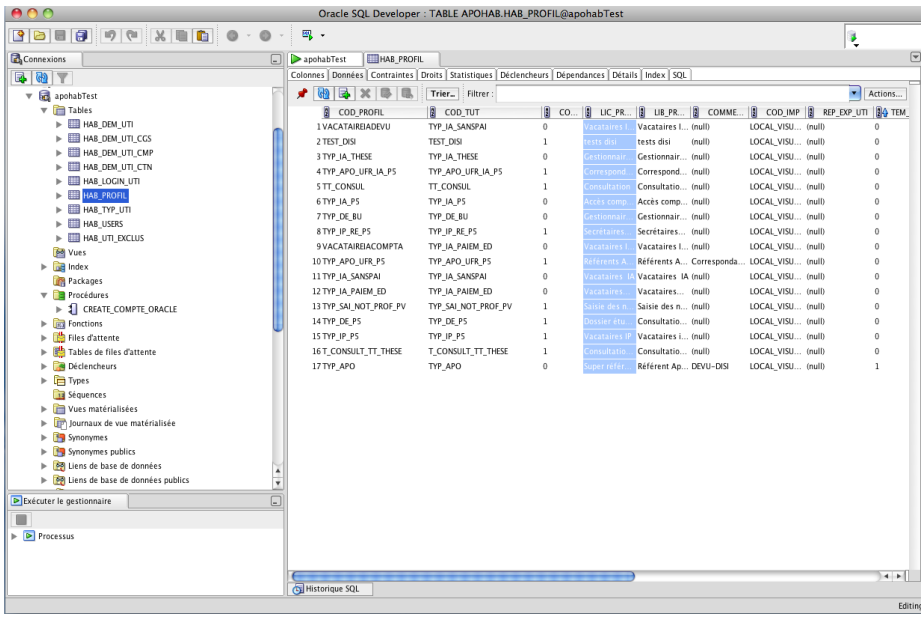


Figure 52 L'outil Oracle SQL Developer

5.2.1.5. L'outil GAWOR

L'outil Gawor Ldap Browser permet d'explorer les annuaires Ldap et de tester les requêtes Ldap.

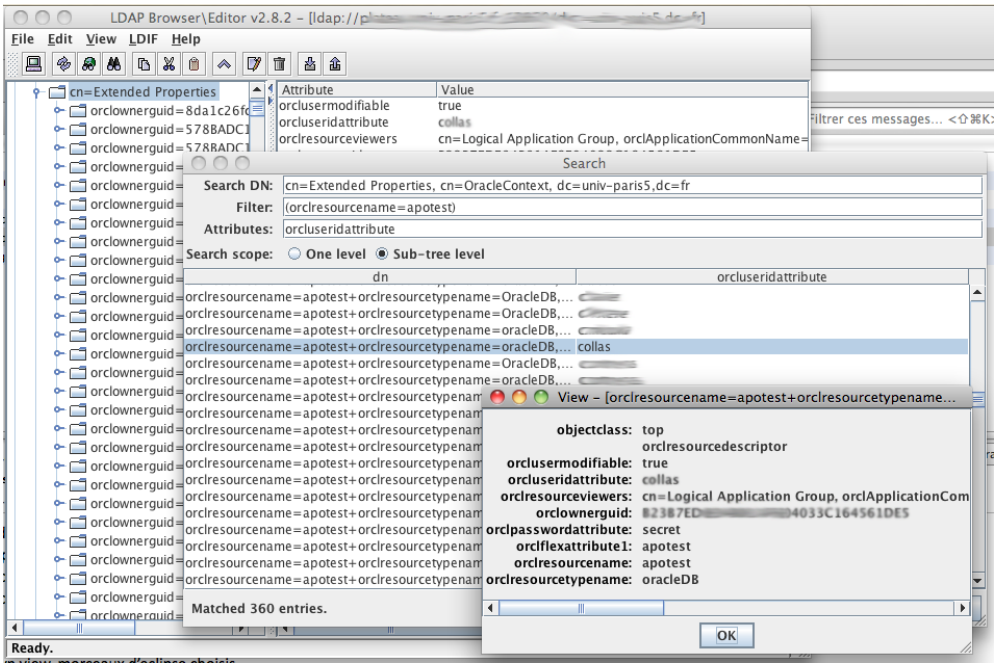


Figure 53 L'outil Gawor Ldap Browser
Source : Université Paris Descartes

5.2.2. Le framework esup-commons : de la version 1 à la version2

Le principe du framework esup-commons est simple. Une procédure d'installation fournie avec le framework crée un nouveau projet appelé esup-blank, un squelette de projet qui utilise les classes créées par le module esup-commons. Esup-blank est renommé du nom que l'on souhaite donner au projet. On peut aussi chercher sur le dépôt SVN le projet esup-example pour s'en inspirer. Esup-example montre un exemple de page de création et mise à jour de la table des utilisateurs.

Esup-commons gère les aspects difficiles tels que la gestion du cache, les procédures d'envoi d'e-mail lors des exceptions, les accès au Ldap, l'authentification, etc… Esup-blank possède une structure en couche où il suffit d'alimenter les couches qui vont être utilisées et de supprimer celles qui ne le seront pas.

Esup-commons fonctionne sur l'atelier de génie logiciel Eclipse. Il s'obtient sur le dépôt SVN du CRU : *https://subversion.cru.fr/esup-commons*.

La version 1 d'esup-commons a été créée en 2006 sous la direction de Pascal Aubry, de l'université de Rennes (Esup Portail, 2006). Esup-commons v1 possédait les caractéristiques suivantes :

- Java : en tant que langage de base
- JSF : Java Server Faces pour gérer les vues sur le web
- Spring : pour gérer le conteneur de beans
- Hibernate ou Isatis : pour gérer l'accès aux données
- Accès LDAP
- SSO : authentification par le portail Esup-Portail et le service CAS
- Ant : pour déployer le projet
- Eclipse et Tomcat : AGL et serveur

Esup-commons est en évolution constante. A partir de fin 2009/2010 le souhait de suivre l'évolution des outils open source les plus utilisés du marché conduit à des évolutions majeures dans esup-commons v1 qui passe en v2 début 2011. Esup-commons v2 a eu aussi des évolutions successives (Esup Portail, 2011d) :

- Maven remplace Ant
- JPA remplace Hibernate (évolution de java intégrée dans ESCV2)
- RichFaces ou Primeface s'introduisent dans le JSF
- CXF remplace XFire pour les services web SOAP
- Le serveur Jetty est utilisé en développement
- Un framework encore instable au moment où la réalisation a commencé

Lorsque la réalisation a commencé, ESCV1 n'était plus très utilisable, il y avait des incohérences entre les versions de java, d'éclipse et d'ESCV1, ce qui m'a conduite à abandonner ESCV1. ESCV2 était dans une version intermédiaire relativement stable, ESCV2 était encore fourni comme un module unique (version 0.2.2 ou 0.2.3). Par la suite ESCV2 a encore évolué pour être fourni sous forme de modules multiples (versions > 0.2.5), et cette version a été utilisée dans d'autres projets. Mais pour Apobilitation, en tenant compte du volume de code déjà écrit nous sommes restés en version 0.2.3. Une version 0.3.n a été annoncée en 2012 qui utilise le dépôt GIT. Comme le forgeron, cent fois sur le métier Esup remet son ouvrage…

5.2.3. Les serveurs locaux

5.2.3.1. Tomcat

ESCV1 utilisait le serveur Tomcat par défaut. Une procédure Ant permettait de créer soit un projet de test soit le projet de production. L'université Paris-Descartes utilise des serveurs Tomcat aussi sur des machines Linux. Il existe un serveur Tomcat pour les applications qui utilisent la version de Tomcat 5.5. Deux serveurs Tomcat sont dédiés aux applications qui utilisent Tomcat 6.0. Un serveur Tomcat est réservé aux web services. Sur chacun de ces serveurs on peut mettre une application *test-monapplication* en plus de l'application officielle. Pendant le développement deux serveurs locaux ont été installés sur ma machine (Tomcat version 6.0), l'un pour l'application cliente (port 8080) l'autre pour le service web (port 8081).

5.2.3.2. Jetty

Le serveur Jetty est fourni avec ESCV2. Il a l'avantage de permettre de corriger très vite et relancer l'exécution d'une application en cours de développement. Il est embarqué avec Maven 2, ce qui explique sa présence dans ESCV2. Ce serveur est très léger et très rapide. En revanche il est un peu juste en taille mémoire et présente quelques bugs.

5.2.4. Le matériel

Tout le développement a été fait sur un ordinateur portable Mac Book Pro et sous le système Snow Leopard. Les serveurs de production sont sous Linux sur des serveurs Apache.

5.2.5. Le dépôt sous subversion (SVN)

Le framework esup-commons est obtenu sur le dépôt de Subversion (SVN). Pour créer un nouveau projet on utilise un archétype esup-blank sur le dépôt Maven.

L'université possède son propre espace pour y déposer ses projets sous subversion.

Les applications APOBILITATION et ApobilitationWS sont aussi sur leur dépôt SVN.

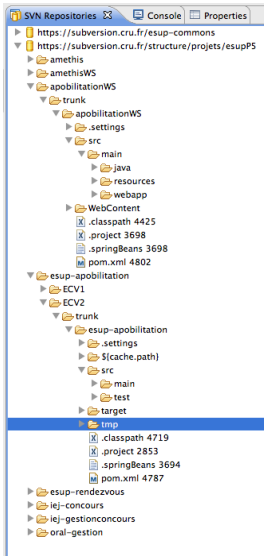


Figure 54 Le repository de Subversion
Source : MC COLLAS

L'utilisation de Subversion permet de sauvegarder les sources de

l'application à l'abri du risque de perte de données. Elle facilite la transmission d'un projet d'une personne à une autre. Elle permet aussi de revenir en arrière sur une version précédente, ou de comparer des versions et de retrouver la trace d'une correction. Elle permet aussi (ce qui n'est pas le cas dans ce projet) le travail collaboratif au sein d'une équipe.

5.3. LA CREATION DES PROJETS ET L'ORGANISATION DES REPERTOIRES

Pour créer le projet il faut créer un projet esup-blank et le renommer ensuite du nom choisi, comme esup-example. Au départ il a été nommé esup-apobilitation. On peut aussi s'inspirer du projet esup-example fourni par le projet esup-commons pour avoir un noyau de départ qui fonctionne.

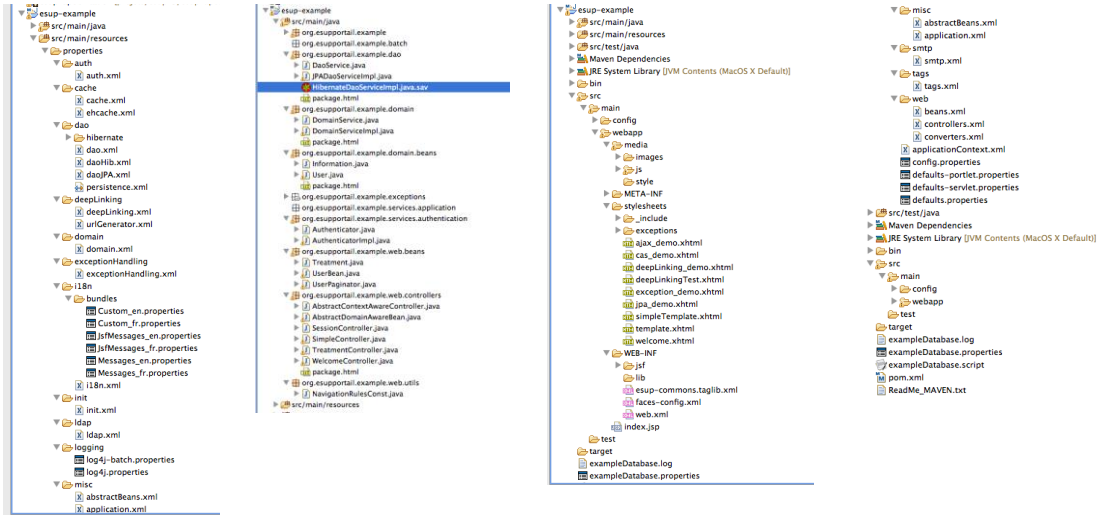


Figure 55 Organisation d'esup-example
Source : MC COLLAS

5.3.1. Apobilitation

Le projet esup-apobilitation ainsi créé va être enrichi fonctionnalité après fonctionnalité jusqu'à atteindre son développement final.

Il est à noter que ce projet hérite par construction d'un certain nombre d'attributs.

Esup-apobilitation est un projet Maven. Donc on retrouve dans l'arborescence des fichiers le fichier ***pom.xml*** qui contient d'une part les dépendances du

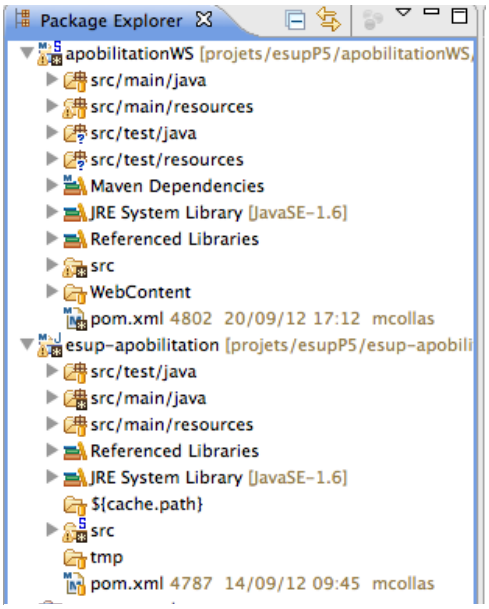


Figure 56 Attributs Maven des projets
Source : MC COLLAS

projet à d'autres projets, d'autre part les paramétrages de différents profils de déploiement (test sur Jetty, développement sur Tomcat, développement en production, etc..).

Sur la Figure 56, on observe les répertoires main et test. Ces derniers accueillent les tests jUnit.

Maven distingue aussi entre les classes java et les fichiers ressources qui sont essentiellement des fichiers XML ou properties. Maven crée aussi un répertoire *target* (non visible sur l'image). Ce répertoire *target* peut être supprimé. Il est recréé à chaque déploiement et contient les war des objets exécutables et des sources. Le répertoire *tmp* est dû aussi à Maven.

Esup-apobilitation est un projet web. Donc on retrouve dans les sources les répertoires *webapp, META-INF, WEB-INF* et les fichiers *index.jsp* et ***web.xml*** qui sont lus par le serveur Tomcat ou autres. Le répertoire *stylesheets* contient les vues jsf (Figure 57).

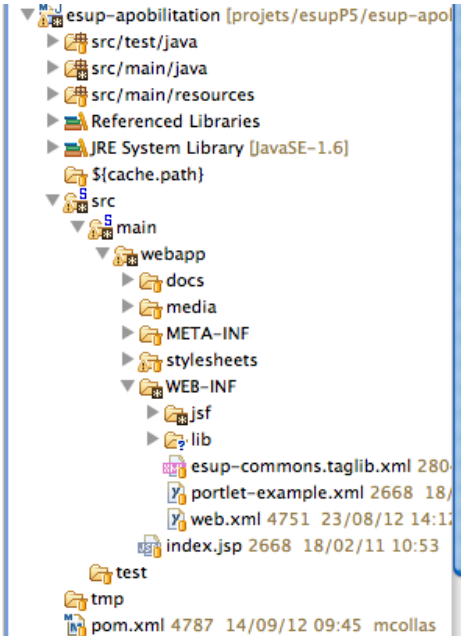


Figure 57 Attributs web du projet
Source : MC COLLAS

Esup-apobilitation est un projet JSF (Myfaces). A ce titre il doit contenir les fichiers ***application.xml*** et ***navigation-rules.xml*** qui définissent les règles de navigation et les paramètres nécessaires à JSF. Ces fichiers sont dans le répertoire WEB-INF/jsf.

Les vues qui utilisent la syntaxe de JSF sont dans le répertoire ***stylesheet***.

Esup-apobilitation est un projet Spring. A ce titre il doit contenir le fichier ***applicationContext.xml*** qui déclare quels sont les beans que Spring doit gérer. *ApplicationContext.xml* inclut d'autres fichiers XML spécialisés par fonctions. Ci-dessous les fichiers XML déclarant des beans :

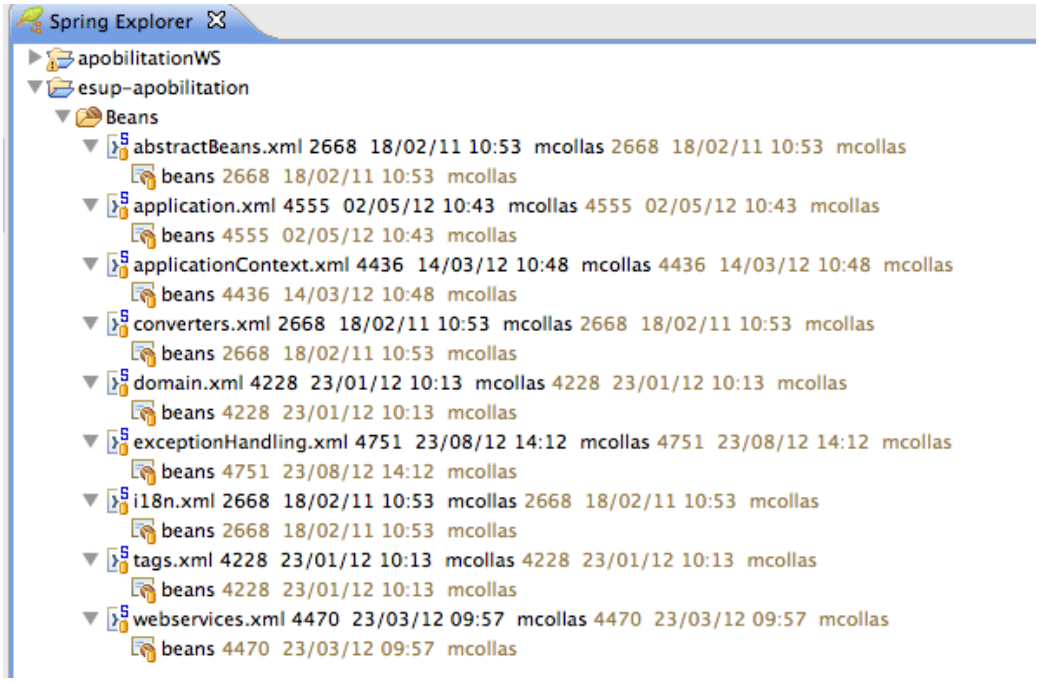


Figure 58 Attributs Spring du projet
Source : MC COLLAS

L'extrait du fichier ApplicationContext.xml montre quels sont les imports :

```
    <import resource="auth/auth.xml" />
    <import resource="exceptionHandling/exceptionHandling.xml" />
    <import resource="dao/dao.xml" />
    <import resource="domain/domain.xml" />
    <import resource="cache/cache.xml" />
        <!-- <import resource="deepLinking/deepLinking.xml" /> -->
    <!-- <import resource="deepLinking/urlGenerator.xml" /> -->
    <import resource="i18n/i18n.xml" />
    <import resource="ldap/ldap2.xml" />
    <import resource="smtp/smtp.xml" />
    <import resource="misc/abstractBeans.xml" />
    <import resource="misc/application.xml" />
    <import resource="tags/tags.xml" />
    <import resource="web/beans.xml" />
    <import resource="webservices/webservices.xml" />
    <import resource="web/controllers.xml" />
```

Tous ces fichiers servent à passer des paramètres aux beans au moment de l'exécution. Ils se trouvent dans les répertoires src/ main/resources/properties.

Esup-apobilitation est un projet Esup-commons.

Toutes ces fonctionnalités sont livrées pré-paramétrées par esup-commons. Il suffit d'adapter celles-ci aux besoins du projet. [14]

Le projet esup-example est fourni avec une fonctionnalité qui fonctionne bien : la mise à jour d'une table qui est celle des utilisateurs de l'application. On peut ajouter une ligne dans la table, la modifier, la supprimer, et paginer la table dès qu'elle dépasse 10 lignes. Ce type de fonctionnalité se retrouve tout le temps dans une application et peut être reproduit autant de fois que nécessaire.

5.3.2. ApobilitationWS

Le projet ApobilitationWS est le service web qui sera appelé par l'application et est aussi l'application batch qui est appelée par un cron chaque jour. Il ressemble beaucoup à esup-apobilitation, mais a quelques différences.

Les points communs :

- ApobilitationWS est un projet Maven
- ApobilitationWS est un projet web
- ApobilitationWS est un projet Spring

Les différences :

- ApobilitationWS n'est pas un projet JSF car il ne gère aucune couche de présentation, il n'y a aucune vue, il n'y a pas de fichier application.xml
- ApobilitationWS n'est pas un projet esup-commons.

La décision a été prise parce que d'une part ApobilitationWS n'a pas besoin de toutes les fonctionnalités offertes par esup-commons. Il est plus simple d'écrire uniquement ce dont nous avons besoin.

La seconde raison est que pour les traitements de mise à jour sur la base de données en batch ou des transactions sur plusieurs bases et annuaires Ldap ou des requêtes complexes pour déceler des anomalies, il est plus simple (de mon point de vue) d'utiliser le langage SQL et l'API JDBC plutôt que JPA qui est assez contraignant.

[14] Grâce au Guide du développeur v2 en ligne ! (Esup Portail, 2011c)

5.4. LE SCHEMA D'UNE FONCTIONNALITE

Un exemple de fonctionnalité simple est l'édition et la mise à jour d'une table de paramètres, par exemple la table des types utilisateurs. Les classes qui vont collaborer à cette fonctionnalité sont :

- une classe du modèle : ***TypeUtilisateur.java***. Cette classe a un bean Spring associé qui mappe automatiquement la table TYP_UTILISATEUR.

- une vue : ***typesutilisateurs.jspx***. Cette vue présente l'affichage de la liste et un bouton modifier sur chaque ligne.

- un contrôleur : ***TypesUtilisateursController.java*** qui est chargé d'écouter les événements en provenance de la vue, de les analyser, de les traiter et de préparer une réponse. Le traitement consiste à repérer les erreurs, mettre à jour la table selon le bouton activé, rafraîchir la nouvelle liste et la réafficher.

- un service de domaine : l'interface ***DomainService.java*** et son implémentation ***DomaineServiceImpl.java***. Ce service est la façade unique qui reçoit toutes les demandes d'objets du modèle, à l'unité ou en liste. C'est la couche Domain. En général ce service se contente de s'adresser à la couche DAO pour fournir l'information, mais il peut y avoir des compléments d'information à trouver ici ou là.

- un service de persistance : l'interface ApoDaoService et son implémentation ApoDaoServiceImpl. Dans cette couche sont réunis tous les accès en base de données: interrogations, créations, modifications, suppressions.

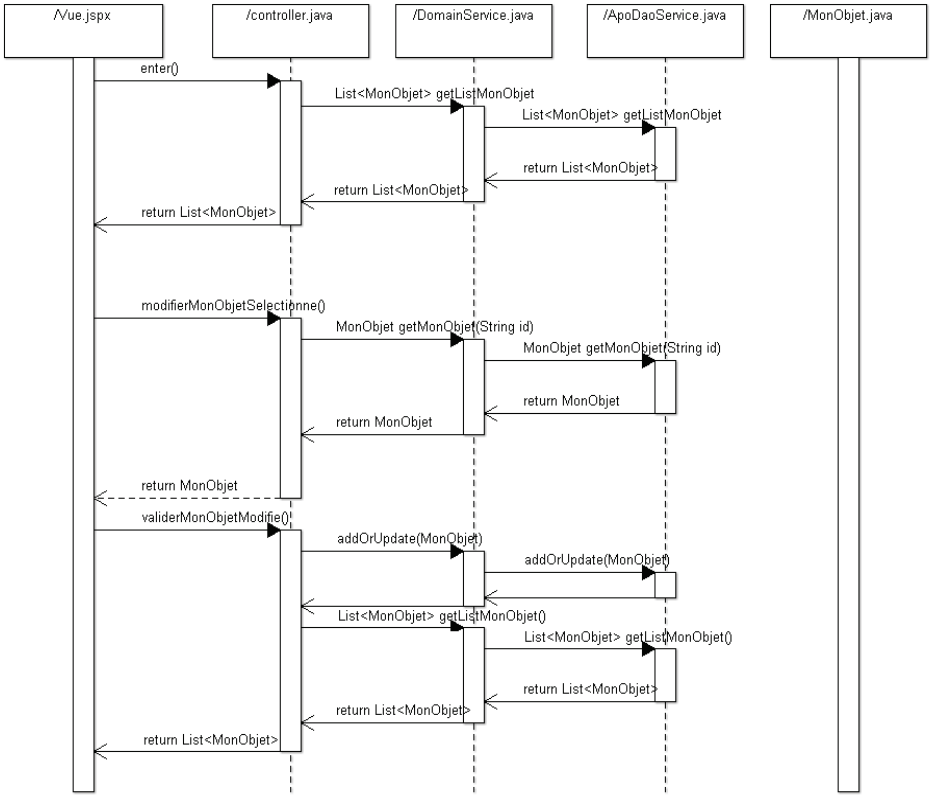


Figure 59 Diagramme de séquence d'édition et mise à jour de table.
Source : MC COLLAS

Le diagramme de séquence met en évidence la collaboration de 5 classes dans une fonctionnalité simple.

Il y a autant de vues jsf (extension jspx) que de pages établies dans la maquette et même un peu plus pour la navigation. Pour chacune de ces vues il y a un contrôleur dédié.

Il y a au moins autant de classes du modèle qu'il y a de tables utilisées en base de données, mappées par Spring.

Mais il n'y a qu'un Domain Service et qu'un DAO service qui sont toujours appelés. Développer une fonctionnalité consiste donc à écrire la classe du modèle, la vue, le contrôleur, et à ajouter dans les classes de DomainService et ApoDaoDomainService les méthodes que l'on souhaite appeler. A cela il faut ajouter la déclaration des beans dans les fichiers de Spring.

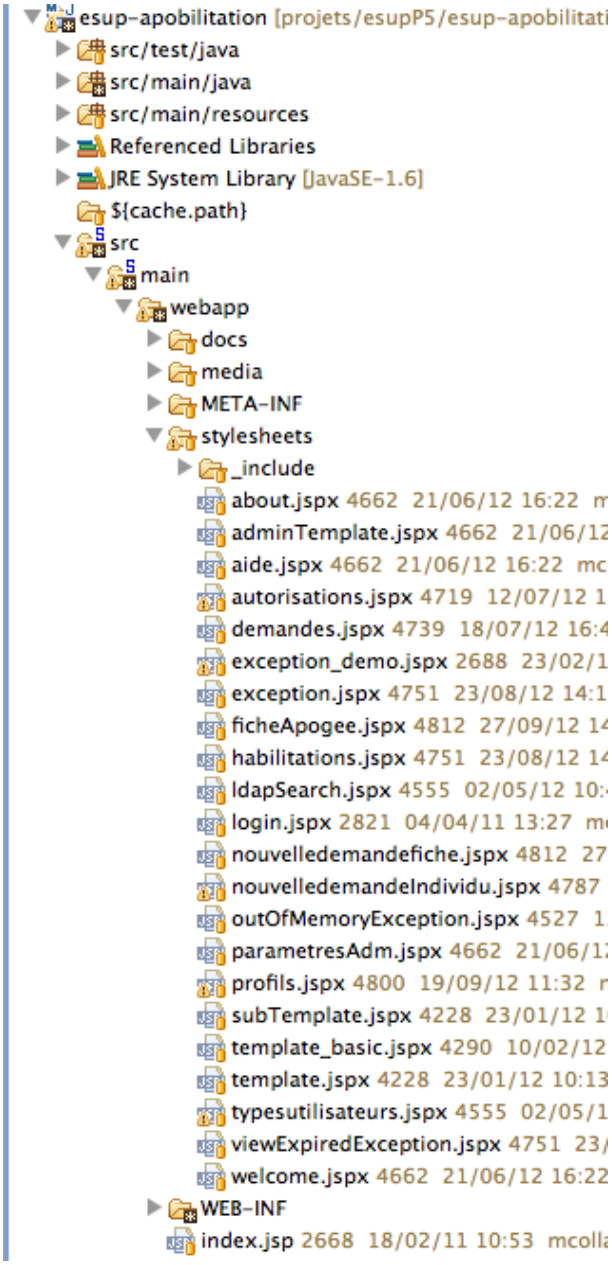


Figure 60 Liste des vues
Source : MC COLLAS

5.5. L'UTILISATION DE L'HERITAGE

5.5.1. L'héritage dans les vues

Avec JSF on peut inclure des sous-parties de pages à l'intérieur d'une page. On va utiliser cette possibilité pour mutualiser les en-têtes, les pieds de page et menu de navigation.

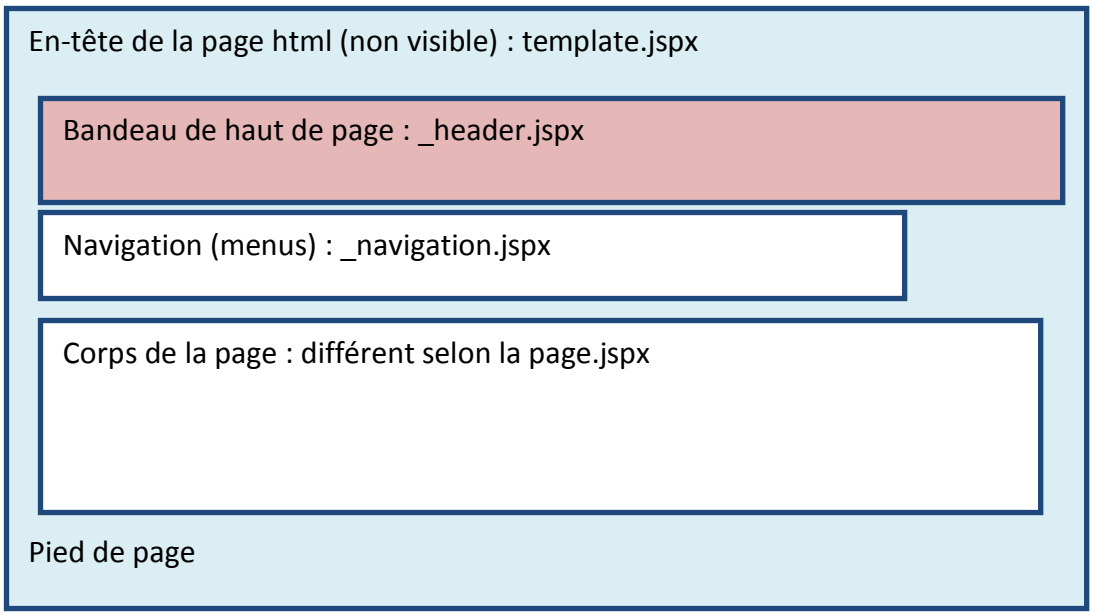


Figure 61 Modèle de vue JSF
Source : MC COLLAS

5.5.2. L'héritage dans les contrôleurs

A l'instar de WelcomeController tous les contrôleurs de pages héritent du contrôleur abstrait AbstractContextAwareController qui lui-même hérite de AbstractDomainAwareBean (Figure 62). De cette façon les comportements attendus sur toutes les pages peuvent être factorisés. Seul le SessionController se trouve placé plus haut dans la chaîne d'héritage afin d'être obtenu de tous les autres par un simple getSessionController.

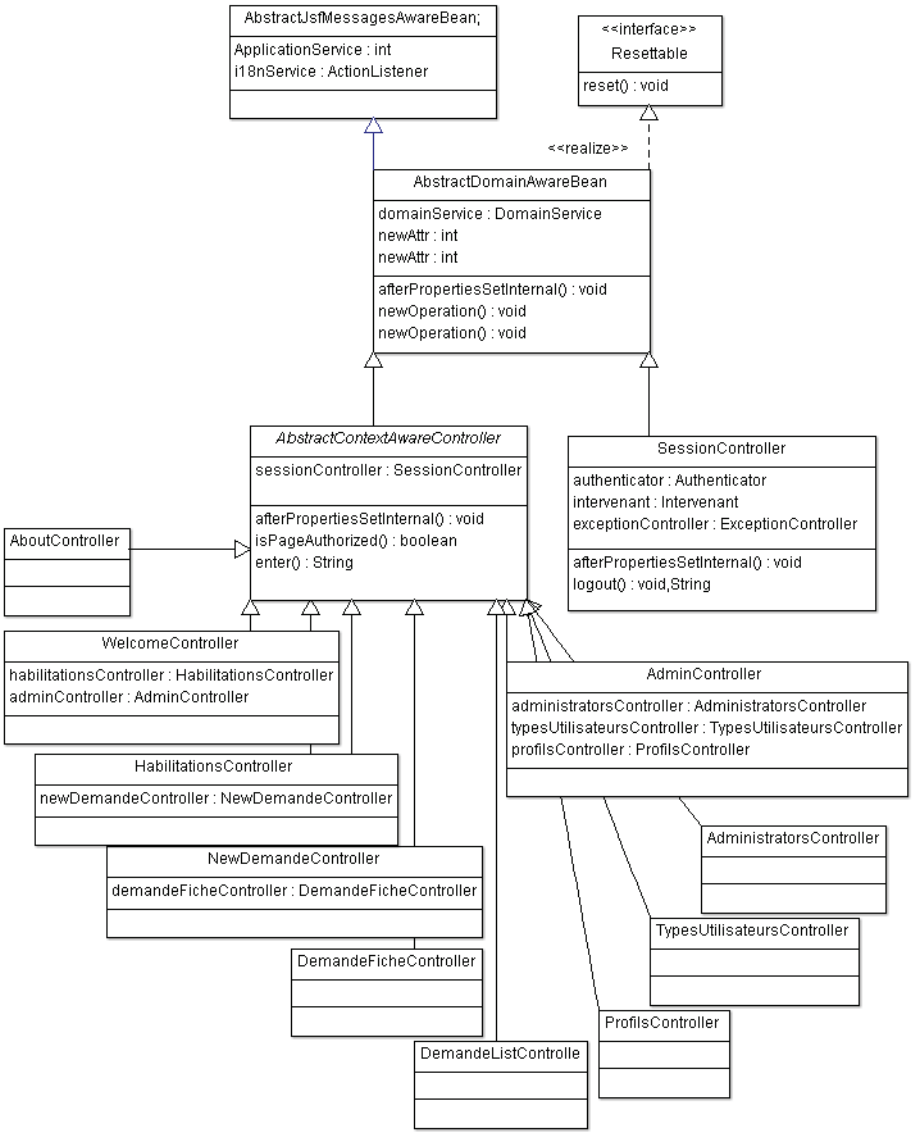


Figure 62 L'héritage des contrôleurs
Source : MC COLLAS

5.6. CONCLUSION

L'installation de tout un environnement de développement esup-commons est un investissement qui prend un certain temps au départ, d'autant plus qu'il s'est fait dans une période transitoire où Esup-commons n'était plus en version 1 et pas tout-à-fait dans une version 2 définitive. Une fois le socle du projet mis en place, le développement proprement dit peut s'effectuer, ce que nous voyons au chapitre suivant.

Chapitre 6. Développement des fonctionnalités

6.1. INTRODUCTION : ORDONNANCEMENT DU DEVELOPPEMENT

Le développement d'une application se construit tout doucement, en vérifiant pas après pas que cela fonctionne au point où l'on est parvenu. Il est déconseillé d'écrire une grande quantité de code sans contrôler qu'il peut s'exécuter. C'est pourquoi il vaut mieux écrire une fonctionnalité à la fois. Voici dans quel ordre les différentes parties ont été développées, chacune ayant besoin de la précédente pour pouvoir se faire :

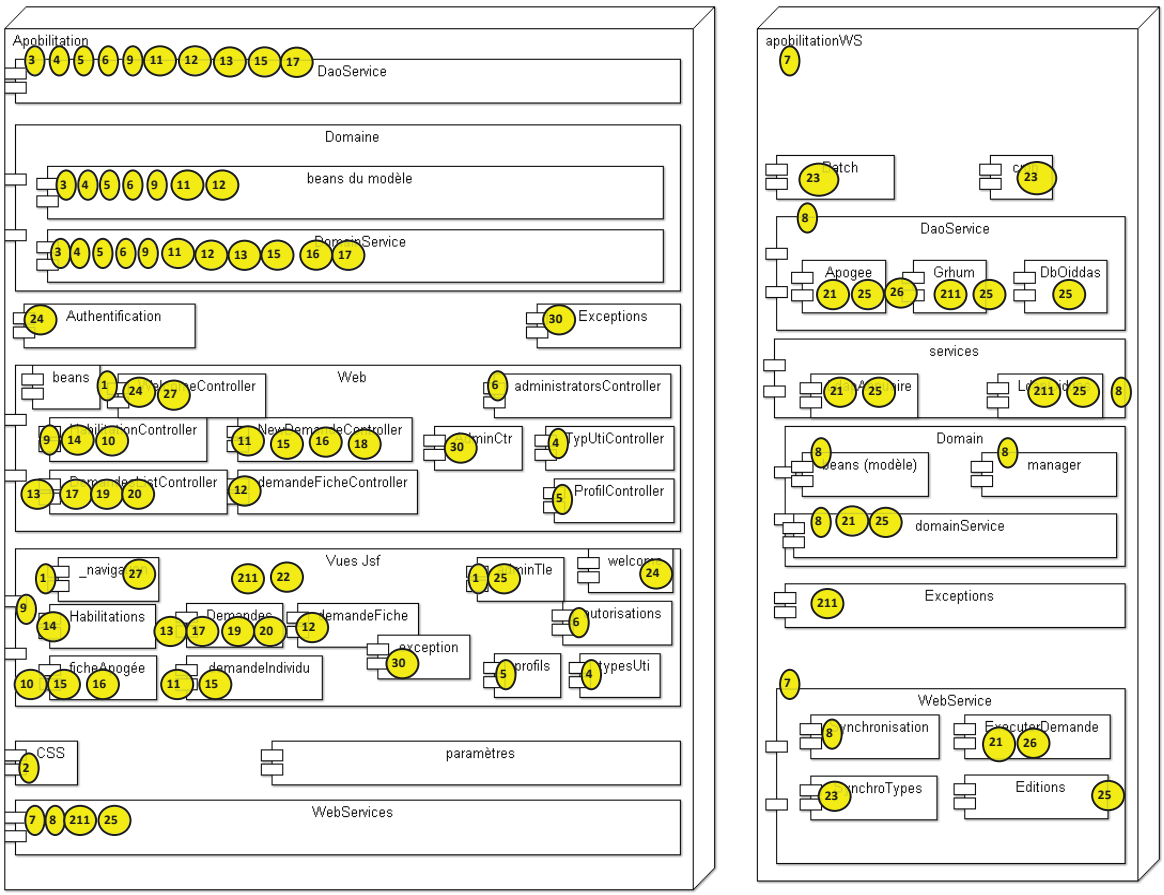


Figure 63 Ordre du développement – Impact sur les composants
Source : MC COLLAS

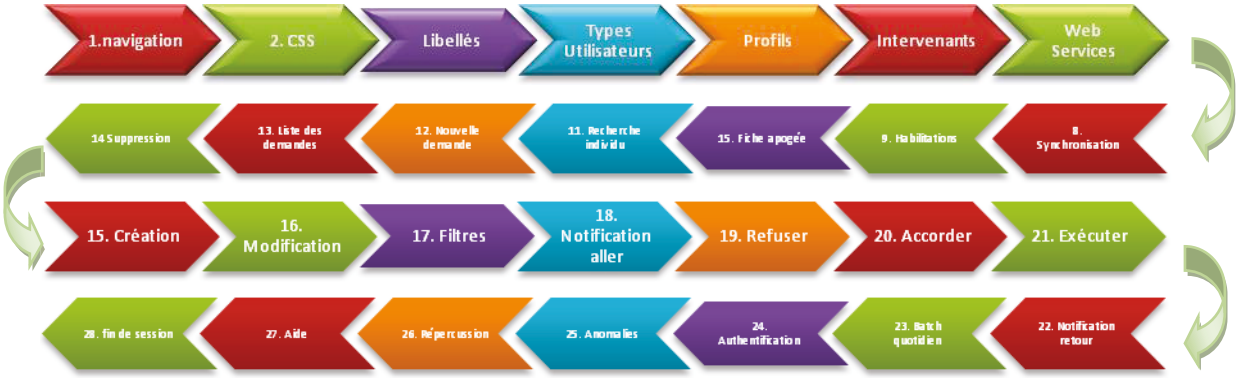


Figure 64 Ordre de développement - Processus
Source : MC COLLAS

6.2. DEVELOPPEMENT DES FONCTIONNALITES

6.2.1. Navigation

Au départ toutes les pages sont créées dans le répertoire `src/main/webapp/stylesheets` à partir du même modèle (`/stylesheets/template.jspx`) qui inclut l'en-tête de la page (`/stylesheets/_include/_header.jspx`), le pied de page, le menu de navigation horizontal page (`/stylesheets/_include/_navigations.jspx`). Dans chaque page (`/stylesheets/welcome.jspx`, etc…) on laisse les informations de la maquette en html. Le but est de vérifier que la mise en page se fait bien.

Les pages de paramétrage ont un deuxième modèle (`/stylesheets/ adminTemplate.jspx`) qui inclut en plus un menu vertical gauche.

On renseigne le fichier `WEB-INF/jsf/navigation-rules.xml` qui va indiquer à JSF quelle page il doit appeler en fonction du mot de navigation qui lui est fourni. On crée aussi la classe `web.utils.NavigationRulesConst` qui contient les constantes de navigation.

On teste ainsi la navigation entre toutes les pages. Puis tous les contrôleurs sont créés à l'identique par héritage de `AbstractContextAwareController`, dans le package `org.esupportail.apobilitation.web.controllers`. On met à jour `src/main/resources/properties/web/controllers.xml` en ajoutant chaque contrôleur. Progressivement le contenu HTML sera remplacé par du contenu JSF, au fur à mesure du développement.

On part du principe qu'on réalise d'abord la navigation du point de vue de l'administrateur, sachant qu'il suffira d'interdire certaines pages au responsable de scolarité.

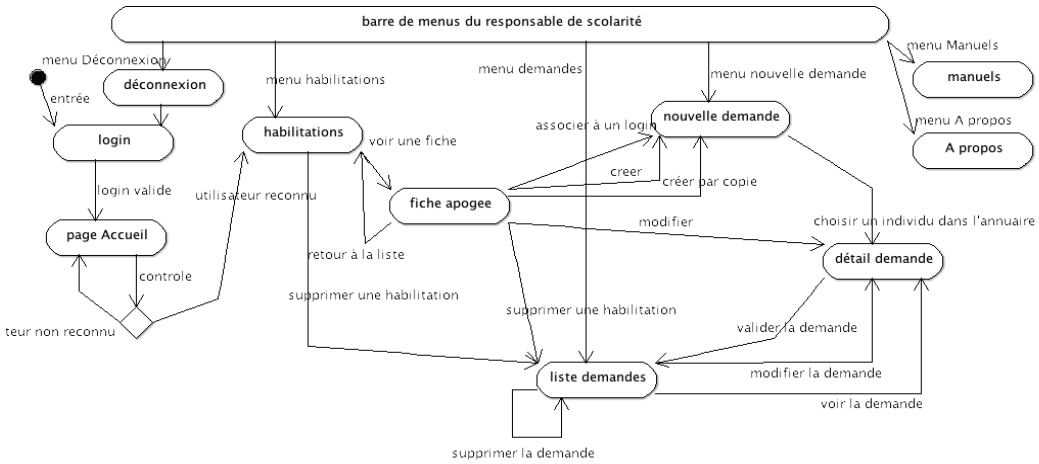


Figure 65 Diagramme de navigation d'un responsable de scolarité
Source : MC COLLAS

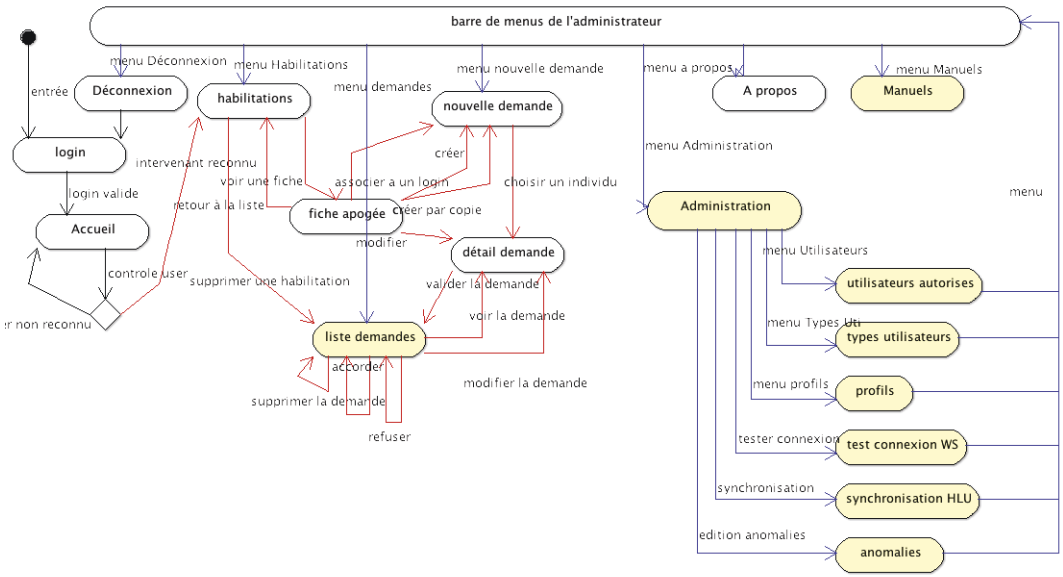


Figure 66 Diagramme de navigation de l'administrateur (en jaune menus réservés ou différenciés)
Source : MC COLLAS

6.2.2. CSS

La mise au point des feuilles de style (css) est faite dans la foulée, car il est agréable de voir évoluer le travail réalisé sous son apparence finale. Cela ne constitue pas une obligation, d'autres choisiraient de le faire à la fin, ou de le déléguer à un spécialiste.

Esup-commons est fourni avec un jeu de CSS de Fluid Skinning System (FSS) et il y a peu de choses à modifier pour obtenir un résultat satisfaisant. Les feuilles de style se trouvent sous `src\main\webapp\media\style`.

6.2.3. Libellés

Les données utilisées manipulent un grand nombre de codes, mais ce sont les libellés et non les codes qu'il est souhaitable d'afficher à l'écran. Or ces tables changent très peu, mais il est nécessaire de lire ces libellés à chaque affichage de page. Pour limiter les accès au serveur de base de données, il est préférable de lire ces tables une seule fois au lancement de la session. Dans ce cas, le temps réel n'est pas une obligation. Ces tables dans Apogée sont :

- les Centres de Gestion (CGE),
- les Centres de Gestion de Stage (CGS),
- les Centres d'incompatibilité (CIN),
- les Centres d'inscription pédagogiques (CIP),
- les Centres de traitement des notes (CTN),
- les composantes,
- les types Utilisateurs,
- les correspondances de structures entre Grhum et Apogée (dans Grhum).

L'objet `domain.UP5.java` appartenant à `domain.DomainServiceImpl.java` est chargé de lire et conserver ces tables en mémoire. Les jsf, classes et interfaces impactées sont :

- `Dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `domain.UP5`
- `domain.beans.apogee.CentreGestion`
- `domain.beans.apogee.CentreGestionStage`
- `domain.beans.apogee.CentreIncomp`
- `domain.beans.apogee.CentreInsPed`
- `domain.beans.apogee.CentreTraitementNotes`
- `domain.beans.apogee.Composante`
- `domain.beans.apogee.TypeUtilisateur`
- `domain.beans.apogee.UtiCollecterCtn`
- `domain.beans.grhum.P5CorrespondanceStructure`

6.2.4. Types d'Utilisateurs

La fonctionnalité de mise à jour de la table HAB_TYP_UTI des types utilisateurs étendus (modèle, domainService, DaoService, contrôleur, vue jsf, paginateur). Cette table est nécessaire à la fonctionnalité profil. Elle étend la table d'Apogée TYP_UTILISATEUR que nous

ne voulons pas modifier, mais rajoute les informations nécessaires à la constitution des profils. Les jsf, classes et interfaces impactées sont :

- Dao.ApoDaoService et ApodaoServiceImpl
- domain.DomainService et DomainServiceImpl
- domain.beans.apogee.TypeUtilisateur
- domain.beans.hab.HabTypUti
- web.controller.TypesUtilisateursController
- web.beans.TypeUtiPaginator
- stylesheets/ typesutilisateurs.jspx

6.2.5. Profils

La fonctionnalité de création, mise à jour et suppression de la table HAB_PROFIL (modèle, domainService, DaoService, contrôleur, vue jsf, paginateur). Cette fonctionnalité s'appuie sur les types utilisateurs. Les jsf, classes et interfaces impactées sont :

- Dao.ApoDaoService et ApodaoServiceImpl
- domain.DomainService et DomainServiceImpl
- domain.beans.hab.Profil
- web.controller.ProfilsController
- web.beans.ProfilPaginator
- web.beans.ProfilBean
- stylesheets/profils.jspx

6.2.6. Intervenants

La fonctionnalité de création, mise à jour et suppression de la table HAB_USERS des utilisateurs autorisés (ou intervenants). Cette fonctionnalité s'appuie sur les Composantes, les CIP et l'annuaire. Elle inclut une recherche dans l'annuaire, et définit les droits des intervenants.

A ce stade, comme on n'a pas encore utilisé l'authentification, on saisit sur la page d'accueil le login et le rôle de l'intervenant pour tester ce qui est écrit selon les différents cas. Les jsf, classes et interfaces impactées sont :

- Dao.ApoDaoService et ApodaoServiceImpl
- domain.DomainService et DomainServiceImpl
- domain.beans.hab.Intervenant
- web.controller.AdministratorsController
- web.controller.LdapSearchCaller
- web.controller LdapSearchController
- web.beans.IntervenantPaginator
- stylesheets/autorisations.jspx

6.2.7. Web Service

Pour continuer le développement on a besoin de la table HAB_LOGIN_UTI. Or celle-ci doit être créée par le programme de synchronisation qui se trouve dans le web service.

Au départ on crée une application web basée sur Spring. On y ajoute un HelloWorld trouvé dans la documentation de Spring (Apache, 2011b) :

- Dans le package org.esupportail.apobilitation.demo.spring
  - `Interface HelloWorld`
  - `Classe HelloWorldImpl`
- Dans les resources/properties/spring
  - Le fichier beans.xml avec les appels à cxf et la description du bean `HelloWorldImpl`
- Dans la section test, on ajoute une classe JUnit `HelloWorldTest` qui appelle la fonction depuis le service web lui-même pour le tester.

Puis dans l'application APOBILITATION on ajoute l'appel de ce helloWorld : `String sayHi(String text);`

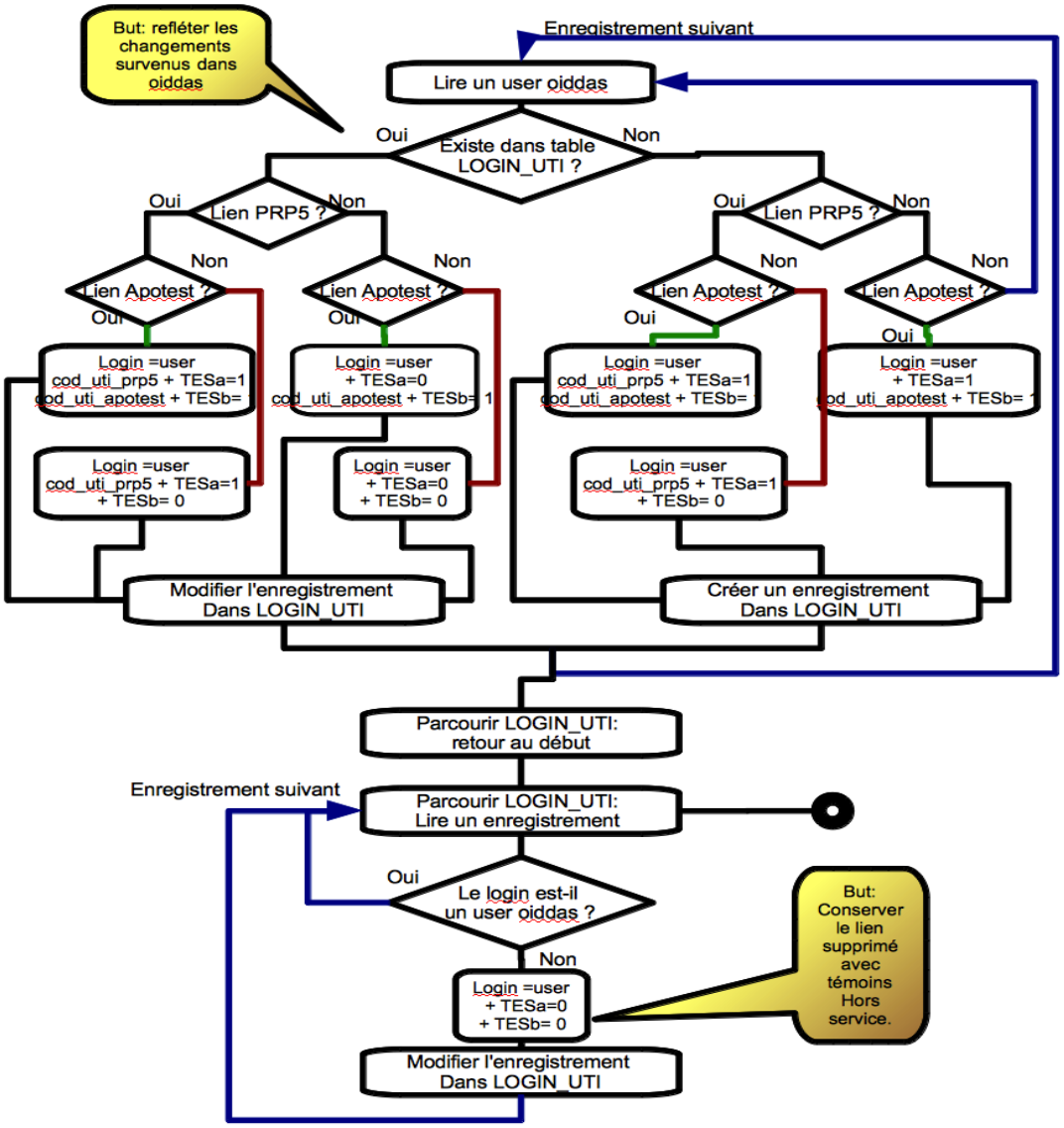
- Dans le package `org.esupportail.apobilitation.demo.spring`
  - Interface `HelloWorld`
  - Classe `Client`
- Dans les `resources/webservices`
  - Le fichier `webservices.xml` avec les appels à cxf qui référence côté client le bean `helloWorld`
- Dans le package `org.esupportail.apobilitation.web.controllers`
  - On ajoute un bouton à la page jsf `admintemplate.jspx` pour appeler le service web `helloWorld`
  - On modifie la classe `AdminController` pour qu'elle appelle le service web `helloWorld`

Tout ceci permet de vérifier que cela marche bien. Ce HelloWorld destiné à être supprimé par la suite sera conservé comme fonctionnalité pour vérifier que la connexion avec le service web est établie.

Sur cette base on pourra créer les autres services web qui seront placés dans le package `org.esupportail.apobilitation.webservices`.

6.2.8. Synchronisation

La fonctionnalité de création et synchronisation de la table HAB_LOGIN_UTI.



Traitement batch de synchronisation Oiddas vers HAB_LOGIN_UTI

Figure 67 Organigramme de synchronisation
Source : MC COLLAS

Cette fonctionnalité se trouve dans SynchroHabLoginUtiServiceImpl.java. Elle applique l'organigramme de synchronisation décrit dans la conception de façon procédurale (Figure 67 Organigramme de synchronisation). Elle envoie en retour un objet RapportWsBean qui contient un code retour et une liste de messages d'exécution, à afficher.

Puisque cette fonctionnalité parcourt les bases et annuaires Ldap, les couches DAO, domain et services sont implémentées à cette occasion.

Les jsf, classes et interfaces impactées dans apobilitationWS sont :

- Webservices.SynchroHabLoginUtiService et webservices.SynchroHabLoginUtiServiceImpl (coté serveur)
- Webservices.RapportWSBean qui est l'objet de retour
- Webservices.WSErrorCodeConst (liste des codes retour)
- dao.ApoDaoService et dao.ApoDaoServiceImpl
- dao.DbOiddasService et dao.DbOiddasServiceImpl
- domain.DomainService et domain.DomainServiceImpl
- domain.beans.apogee.CentreGestion;
- domain.beans.apogee.CentreIncomp;
- domain.beans.apogee.CentreInsPed;
- domain.beans.apogee.UtilisateurAPO;
- domain.beans.hab.HabLoginUti;
- domain.beans.hab.HabUtiExclus;
- domain.beans.oiddas.Configuration;
- domain.beans.oiddas.UserOiddas;
- domain.manager.BeanOiddasDataManager;
- exceptions.ConnexionsClosedException;
- exceptions.UserExistsException;
- services.LdapOiddasService et services.LdapOiddasServiceImpl;
- services.LdapAnnuaireService et services.LdapAnnuaireServiceImpl
- services.LdapUser;
- services.LdapAnnuaireContextException
- services.LdapOiddasContextException
- utils.ApplicationContextHolder
- properties/dao/dao.xml
- properties/domain/domain.xml
- properties/ldap/ldap.xml
- Properties/webservices/webservices.xml

Pour pouvoir tester cette fonctionnalité on utilise des classes de test JUnit. Puis on utilise l'appel du Service web par APOBILITATION pour lancer la fonctionnalité : RapportWSBean synchroniserLoginUti(). Les JSF, classes et interfaces impactées dans APOBILITATION sont :

- Webservices.SynchroHabLoginUtiService
- Webservices.RapportWSBean
- Webservices.WSErrorCodeConst
- exceptions.ExecutingWSException
- exceptions.TempsDepasseException
- web.controllers.AdminController
- stylesheets/adminTemplate.jspx
- Properties/webservices/webservices.xml

On dispose maintenant de la table HAB_LOGIN_UTI correctement renseignée.

6.2.9. Habilitations

On veut écrire la fonctionnalité d'affichage des habilitations. Cela consiste à compléter la table HAB_LOGIN_UTI avec les informations venant de diverses tables d'Apogée et de Grhum.

A partir du login on va chercher les noms et prénoms et le code structure. A partir du codeUtiApogee on va chercher l'utilisateur Apogée et tous les libellés associés à des codes. Comme on veut aussi sélectionner sur les composantes il faut que toutes les composantes de chaque utilisateur soient présentes. Il y a une ligne par code utilisateur. Mais pour chaque code utilisateur il peut n'y avoir aucun login, ou un login ou plusieurs logins. Donc il y a une ligne par couple code Utilisateur + login.

Un premier essai pour compléter l'objet HabLoginUti en imbriquant dedans les objets UtilisateursAPO, et tous les autres objets dedans et en laissant JPA (Hibernate) remplir les objets à sa manière a montré un effondrement des performances. Pour chaque objet habLoginUti il fait plusieurs select successifs pour compléter les informations. Il a donc fallu changer de stratégie.

Finalement la solution suivante a été retenue : un bean spécialement attaché à cette fonctionnalité HabilitationListBean est rempli par un select unique qui retourne une table temporaire avec tous les champs de libellés renseignés. Cette table contient une ligne par composante de chaque utilisateur. Le select retient les utilisateurs possédant la composante qui intéresse. Une fois la table obtenue on crée un objet HabilitationListBean pour chaque utilisateur/login avec une collection de composantes.
La liste de ces HabilitationListBean peut être affichée. Cette opération qui utilise la puissance de la base de données améliore grandement les performances. Il faut penser qu'un administrateur qui a une vue sur l'ensemble du CGE va voir s'afficher la liste de tous les utilisateurs en service, ce qui fait environ 750 lignes à afficher. Le soin de la pagination a été abandonné à l'outil RichFaces sur la page JSF.

Pour le filtre d'affichage, plutôt que de passer un grand nombre de paramètres à la requête, on préfère renseigner un objet `UtilisateurApoFilter.` Cet objet est transmis et lu par la méthode du Dao Service qui modifiera la requête en conséquence. Les JSF, classes et interfaces impactées dans APOBILITATION sont :

- web.controllers.HabilitationsController
- web.beans.HabilitationListBean;
- web.beans.UtilisateurApoFilter
- domain.beans.apogee.CentreIncomp;
- domain.beans.apogee.CentreInsPed;
- domain.beans.apogee.Composante;
- domain.beans.apogee.TypeUtilisateur;
- domain.beans.apogee.UtilisateurAPO;
- apobilitation.domain.beans.hab.Intervenant;
- dao.ApoDaoService et ApodaoServiceImpl
- domain.DomainService et DomainServiceImpl
- stylesheets/habilitations.jspx

6.2.10. Fiche Apogée

L'affichage d'une fiche Apogée s'obtient à partir de la liste des habilitations en cliquant sur le bouton  (voir la fiche). A ce moment-là, à partir du code utilisateur de la ligne, on va chercher un objet Utilisateur complet, avec tous ses objets imbriqués et ses listes de composantes, de centres de stage, et de centres de traitement des notes, avec tous les libellés utiles.

Cette page n'est pas modifiable, elle est seulement pour la modification. Il y a plusieurs boutons commandant des fonctionnalités (supprimer, créer par copie, modifier, connecter sur un login et retour à la liste). Pour l'instant seul le bouton « retour à la liste » est programmé. Les autres fonctionnalités restent en attente car il y a d'autres chemins pour les atteindre.

Le code nécessaire pour contrôler l'édition de la fiche reste dans HabilitationController, car cette page n'est accessible que par la page Habilitations. Que ce soit pour supprimer, créer, modifier, reconnecter une habilitation à un login, il faut dans tous les cas créer une demande. Les JSF, classes et interfaces impactées dans APOBILITATION sont :

- web.controllers.HabilitationsController
- domain.beans.apogee.CentreIncomp;
- domain.beans.apogee.CentreInsPed;
- domain.beans.apogee.Composante;
- domain.beans.apogee.TypeUtilisateur;
- domain.beans.apogee.UtilisateurAPO;
- dao.ApoDaoService et ApodaoServiceImpl
- domain.DomainService et DomainServiceImpl
- stylesheets/ ficheApogee.jspx

6.2.11. Recherche

Pour créer une demande il faut d'abord implémenter la fonctionnalité de recherche d'une personne avec son login. Cette recherche peut se faire soit dans l'annuaire, soit dans Grhum. Les deux ont été écrites, la recherche dans Grhum a été conservée car l'information est plus fraîche d'une journée ce qui est un atout pour rechercher les futurs arrivants.

La recherche dans l'annuaire Ldap se fait sur le personnel actif. Elle est déjà implémentée dans esup-commons avec son module `esup-commons.service.ldap.jar`. Un exemple se trouve aussi dans `esup-exemple` qui consiste à rechercher un individu dans l'annuaire LDAP à partir du login ou du nom, et il a été légèrement modifié pour la mise à jour de la table des intervenants.

Dans cette recherche, nous avons mis comme critères de recherche le nom, le prénom, le login et la composante. Si l'intervenant est un administrateur la recherche se fait sur l'ensemble de l'université. Si l'intervenant est un responsable de scolarité, la recherche est limitée à la composante ou le CIP sur lequel il a des droits. Or l'information Composante ou CIP n'existe pas à l'identique dans l'annuaire, mais seulement une liste de structures. Il faut donc rechercher dans la table des correspondances structures de GRHUM quelle structure correspond à la composante. Puis faire la recherche dans LDAP avec ce code structure. Le code structure du CIP n'appartient pas à cette table. Il est donc entré en paramètre du fichier de configuration pour le CIP K01 qui est le seul qui nous intéresse.

La recherche sort une liste de résultats affichés sous forme de table. Un bouton sur chaque ligne permet de choisir parmi ces résultats celui qui est retenu. Avant d'aller sur une nouvelle page :

- On met dans l'objet demande toutes les informations en provenance de l'annuaire. Cet objet demande a été créé de plusieurs manières. Une seule est développée pour le moment : le bouton de menu « nouvelle demande » crée un nouvel objet demande, sans numéro, vide.
- Puis on regarde dans la table HAB_LOGIN_UTI s'il existe déjà un utilisateur apogée pour ce login.

- S'il existe on va le chercher et on renseigne la demande avec toutes les données en provenance d'Apogée, et on met comme type de demande « M » comme modification.
- S'il n'existe pas on crée un objet Utilisateur avec comme code utilisateur le login en majuscule et le type de demande à « C » comme création.

La recherche dans GRHUM est étendue sur les personnels entrants. Puisque le personnel entrant ne se trouve pas dans l'annuaire avant la date de démarrage de leur contrat et qu'il y a une demande des services utilisateurs pour préparer d'avance tout ce dont ils auront besoin dès le premier jour, en particulier les vacataires, la fonctionnalité a été implémentée de cette deuxième façon en cherchant dans une vue constituée à partir de plusieurs tables dans GRHUM. Cette deuxième façon de procéder est celle qui a été retenue.

La recherche et les méthodes de pré remplissage de nouvelle demande sont dans `NewDemandeController`. Les JSF, classes et interfaces impactées dans APOBILITATION sont :

- `Web.controllers.NewDemandeController`
- `domain.beans.apogee.CentreGestionStage;`
- `domain.beans.apogee.CentreInsPed;`
- `domain.beans.apogee.Composante;`
- `domain.beans.apogee.UtiCollecterCtn;`
- `domain.beans.apogee.UtilisateurAPO;`
- `domain.beans.grhum.P5CorrespondanceStructure;`
- `domain.beans.hab.DemandeUti;`
- `domain.beans.hab.DemandeUtiCgs;`
- `domain.beans.hab.DemandeUtiCmp;`
- `domain.beans.hab.DemandeUtiCtn;`
- `domain.beans.hab.HabLoginUti;`
- `domain.beans.hab.Intervenant;`
- `domain.beans.hab.Profil;`
- `web.beans.HabilitationListBean;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `org.esupportail.commons.services.ldap.*`
- `stylesheets/nouvelledemandeIndividu.jspx`

6.2.12. Nouvelle demande

La page de saisie d'une demande d'habilitation s'affiche selon plusieurs provenances. On considère d'abord qu'on vient de rechercher un login en vue d'une création ou modification.

Les fonctionnalités développées sont de remplir toutes les zones de saisie avec les données d'un utilisateur existant, de permettre d'ajouter ou supprimer une composante, un centre de traitement de notes, de faire réagir certaines données selon le profil choisi, de contrôler la

saisie. La demande qui a été pré-remplie à l'affichage de la page doit s'enregistrer lorsqu'on appuie sur le bouton « valider ». Puis l'intervenant est dirigé vers la liste des demandes. On peut contrôler la création de la demande en regardant directement dans la base de données.

Pour pouvoir mieux profiter de cette liste des demandes, il faut abandonner un moment l'implémentation de la création d'une demande pour aller mettre au point la page qui affiche la liste des demandes. Les JSF, classes et interfaces impactées dans APOBILITATION sont :

- `Web.controllers.DemandeFicheController`
- `domain.beans.apogee.CentreGestionStage;`
- `domain.beans.apogee.CentreInsPed;`
- `domain.beans.apogee.Composante;`
- `domain.beans.apogee.UtiCollecterCtn;`
- `domain.beans.apogee.UtilisateurAPO;`
- `domain.beans.hab.DemandeUti;`
- `domain.beans.hab.DemandeUtiCgs;`
- `domain.beans.hab.DemandeUtiCmp;`
- `domain.beans.hab.DemandeUtiCtn;`
- `domain.beans.hab.Intervenant;`
- `domain.beans.hab.Profil;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `stylesheets/nouvelledemandefiche.jspx`

6.2.13. Liste des demandes

L'affichage de la liste des demandes rend un grand service pendant le développement. On implémente l'affichage de toutes les demandes, la possibilité de supprimer, voir, éditer, modifier une demande. L'édition ou la modification d'une demande à partir de cette liste renvoie sur la page de saisie d'une demande.

Le contrôleur `DemandeListController` est implémenté en grande partie, sauf le bouton  (accorder), qui a besoin du web service. Les JSF, classes et interfaces impactées dans APOBILITATION sont :

- `Web.controllers.DemandeListController;`
- `domain.beans.apogee.Composante;`
- `domain.beans.hab.DemandeUti;`
- `domain.beans.hab.Intervenant;`
- `web.beans.DemandeUtiFilter;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `stylesheets/demandes.jspx`

Au point où nous sommes parvenus, toutes les pages JSF ont été implémentées ainsi que tous les contrôleurs. La navigation se fait bien de page en page, mais il reste les principales fonctionnalités à ajouter dans les classes existantes.

6.2.14. Suppression

La demande de suppression peut se faire soit à partir de la page Habilitations en cliquant sur le bouton (supprimer) de la ligne choisie, soit à partir de la fiche apogée en cliquant sur le bouton « supprimer ». Dans les deux cas la page est gérée par `HabilitationController`. Mais `NewDemandeController` sait créer une demande, la renseigner, l'ajouter à la liste et envoyer une notification. On va donc regrouper toutes ces méthodes chez `NewDemandeController`. Et `HabilitationController` va utiliser les méthodes de `NewDemandeController`.

`HabilitationController` vérifie quand même si la demande de suppression concerne un doublon, c'est-à-dire le cas où plusieurs lopins sont attachés à un seul compte apogée. Dans ce cas la demande de suppression est transformée en demande de déconnexion du login.

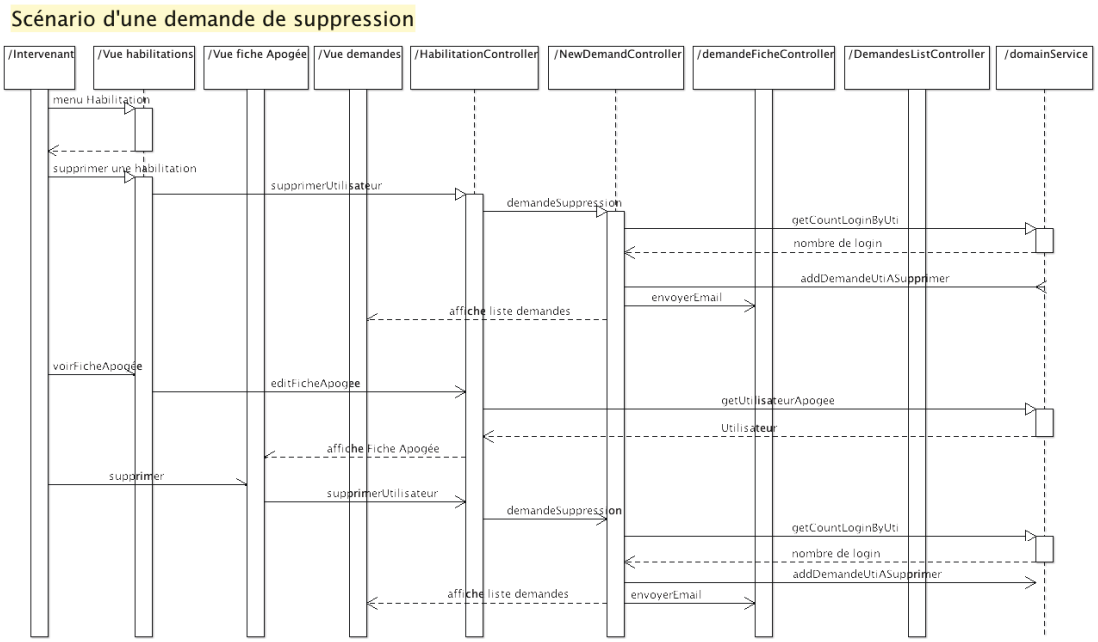


Figure 68 Diagramme de séquence d'une demande de suppression
Source : MC COLLAS

Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `Web.controllers.NewDemandeController;`
- `Web.controllers.HabilitationController;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `stylesheets/habilitations.jspx`
- `stylesheets/ficheApogee.jspx`

6.2.15. Création

La demande de création peut être demandée par différentes voies :

- Menu Nouvelle demande
- A partir de la fiche apogée, bouton créer un nouvel utilisateur
- A partir de la fiche apogée, bouton créer un nouvel utilisateur par copie

Suivant les cas la demande va être pré remplie par le contrôleur qui intervient le premier `HabilitationController` ou `NewDemandeController`, Ensuite `NewDemandeController` se charge de finir la création. Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `Web.controllers.NewDemandeController;`
- `Web.controllers.HabilitationController;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `stylesheets/habilitations.jspx`
- `stylesheets/ficheApogee.jspx`

6.2.16. Modification

La demande de réactivation d'un compte est un cas particulier. Du point de vue de l'utilisateur c'est une création. Du point de vue technique, c'est une modification puisqu'il suffit de remettre le code en service à « O » plutôt que « N ». Pour l'affichage c'est une création. Pour la mise à jour en base les cas de création et de modification sont traités ensemble et la différence ne tient qu'au test d'existence fait juste avant la mise à jour.

La demande de connexion d'un login sur un compte existant (type de demande « D ») est faite depuis la fiche apogée en cliquant sur le bouton « connecter à un login ». Elle renvoie sur la page recherche d'un individu pour chercher le login. Puis l'intervenant choisit sur la page de saisie pour saisir le profil et valider. Si un compte existant dont on demande la modification n'a pas de login on considère qu'il faut le traiter de la même façon. Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `Web.controllers.NewDemandeController;`
- `Web.controllers.HabilitationController;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `stylesheets/habilitations.jspx`
- `stylesheets/ficheApogee.jspx`

6.2.17. Filtres

La fonctionnalité de filtrer les demandes sur différents critères est implémentée en utilisant un filtre `DemandeUtiFilter` qui est transmis jusqu'à la méthode qui sait créer l'ordre SELECT

selon les critères passés par le filtre. L'intérêt de cette solution est de n'avoir qu'une seule méthode à appeler : `getDomainService().getDemandeUtis(filter);`

Sinon il faudrait plusieurs méthodes selon le nombre de critères qu'on veut faire passer. Le filtre est utilisé à la demande de l'utilisateur selon ses envies, mais aussi en application des restrictions de droits pour les responsables de scolarité. Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `Web.controllers.NewDemandeController;`
- `Web.controllers.HabilitationController;`
- `dao.ApoDaoService et ApodaoServiceImpl`
- `domain.DomainService et DomainServiceImpl`
- `stylesheets/demandes.jspx`

6.2.18. Notification aller

La fonctionnalité de notifier la création, la modification ou la suppression d'une demande depuis le demandeur vers le validateur s'appuie sur le service de esup-commons `org.esupportail.commons.services.smtp.SmtpService`. Dès que la demande est validée et mise à jour en base de données dans `DemandeFicheController`, la notification est préparée et envoyée à l'aide de ce service d'envoi d'e-mail. Un message signalant cet envoi s'affiche sur la page des listes de demandes. Les JSF, classes et interfaces modifiées et utilisées dans APOBILITATION sont :

- `Web.controllers.DemandeFicheController`
- `stylesheets/demandes.jspx`
- `org.esupportail.commons.services.smtp.SmtpService.`

6.2.19. Refuser

La fonctionnalité de refus d'une demande et la saisie du motif de refus est mise en oeuvre dans `DemandeListController`. Lorsque l'utilisateur clique sur le bouton  (refuser) une zone de saisie réclame le motif du refus. Le statut de la demande passe de « EC » (en cours) à « R » (refusé). La demande est mise à jour avec le nouveau statut et le motif du refus. Une notification est préparée pour signaler le refus puis envoyée en se servant d'une méthode commune dans `DemandeFicheController`. Les JSF, classes et interfaces modifiées et utilisées dans APOBILITATION sont :

- `Web.controllers.DemandeListController`
- `Web.controllers.DemandeFicheController`
- `stylesheets/demandes.jspx`
- `org.esupportail.commons.services.smtp.SmtpService`

6.2.20. Accorder

La fonctionnalité de validation d'une demande, le workflow et la modification du statut s'implémente aussi dans `DemandeListController`. Lorsque l'utilisateur appuie sur le bouton  le statut de la demande passe à « V » et la demande est mise à jour en base immédiatement. Cela a pour effet de bloquer la mise à jour ou la suppression de cette demande, elle pourra seulement être vue.

Le service web (à implémenter) est appelé avec la demande en paramètre IN et un rapport en paramètre OUT. Le rapport est affiché au-dessus de la liste. Si le code retour n'est pas OK ou si le service web tarde à rendre la main, le statut de la demande est mis à « F » (failed, échec). Les boutons accorder ou refuser sont à nouveau disponibles. Si le code retour est OK, le statut de la demande passe à « X » (exécuté). Les boutons accorder ou refuser disparaissent, la demande peut être archivée.

Si plusieurs demandes se suivent concernant le même compte Apogée, la plus ancienne dans l'ordre des numéros de demande est disponible pour être accordée ou refusée, les autres sont indisponibles (grisées). Dès qu'une demande est exécutée, elle libère donc la suivante.

Plus tard on a observé dans le comportement des utilisateurs qu'ils faisaient des demandes de suppression suivies de demandes de modification. Au moment de valider la demande l'application contrôle les demandes précédentes pour qu'une création suive une suppression même si la suppression n'est pas encore exécutée.

Les JSF, classes et interfaces modifiées et utilisées dans APOBILITATION sont :

- `Web.controllers.DemandeListController`
- `Web.controllers.DemandeFicheController`
- `exceptions.ExecutingWSException`
- `exceptions.TempsDepasseException`
- `webservices.RapportWSBean`
- `webservices.WSErrorCodeConst`
- `webservices.ExecuteDemandeUtiService`
- `stylesheets/demandes.jspx`

6.2.21. Exécuter

L'exécution de la demande par le service web est implémentée dans ApobilitationWS en appelant l'interface `ExecuteDemandeUtiService`.

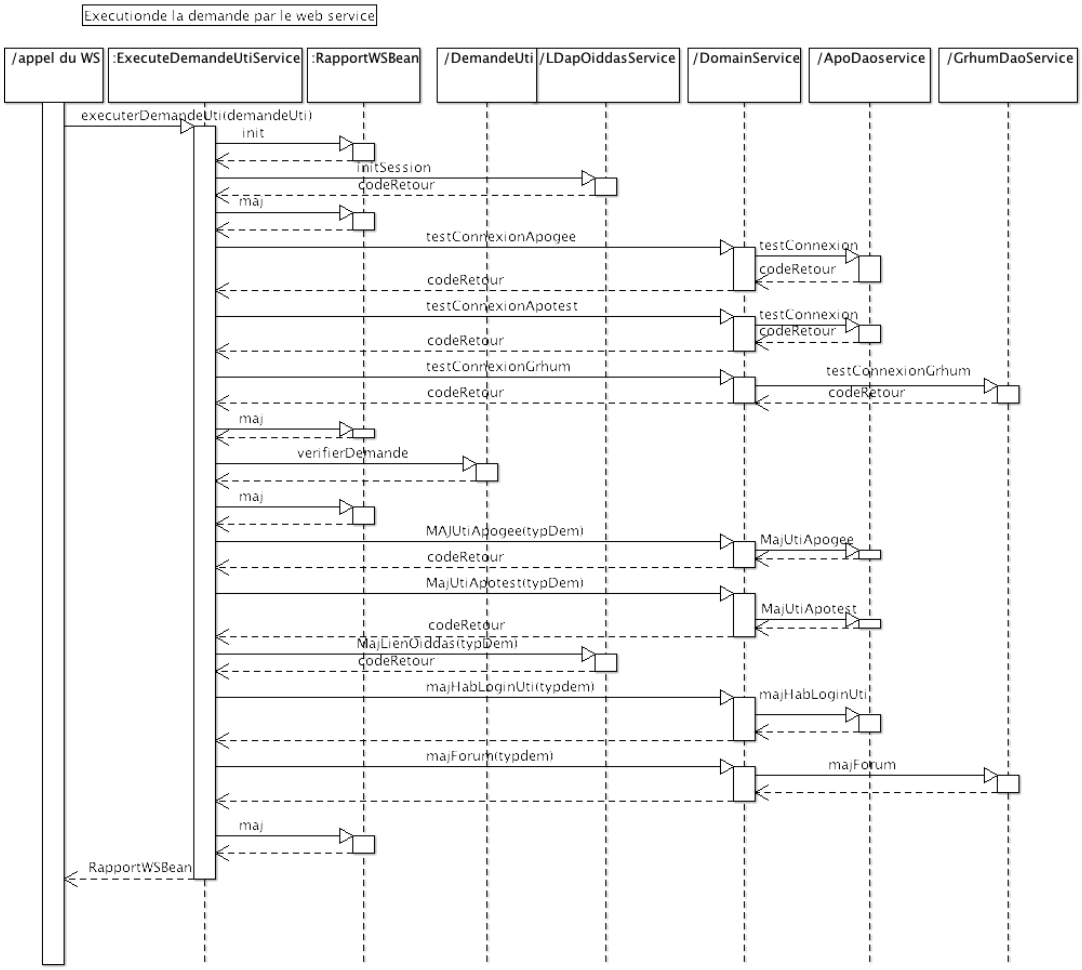


Figure 69 Diagramme de séquence de l'exécution d'une demande
Source : MC COLLAS

Le rapport du WS et les codes de retour :

En paramètre IN on passe une demande, en retour on obtient un objet RapportWSBean qui contient un code retour et une liste de messages d'exécution. La méthode suit une logique procédurale. On positionne le code retour à 99 et il est décrémenté à chaque étape jusqu'à arriver à O en fin de parcours. On sait ainsi où la procédure s'est arrêtée. A chaque étape on écrit une ligne ajoutée à la liste des messages indiquant si l'étape s'est bien déroulée ou non. Avant de quitter la méthode, le rapport est écrit dans le fichier de log pour permettre à l'administrateur de retrouver ce qui s'est passé.

Les codes retours sont :

Tableau III Codes retours du web service

OK	0	valeur de retour OK
OK1	1	valeur de retour OK mais user Oiddas a activer
WARN_DUPLICATE_KEY	20	Tentative de creation sur enregistrement existant
WARN_STATUT_EC	21	Tentative d'executer une demande non validee
WARN_STATUT_R	22	Tentative d'executer une demande refusee
WARN_STATUT_X	23	Tentative d'executer une demande deja executee
ERR_CON_DBOIDDAS	30	Connexion impossible sur DB Oiddas
ERR_INIT_CONF_ACTIVES	31	parametres configurations invalids
ERR_CLOSE_CON_OIDDAS	32	Deconnexion impossible sur Oiddas
ERR_DELETE_OIDDAS	33	Echec de suppression du radEntry dans oiddas
ERR_CREATE_OIDDAS	34	Echec de creation du radEntry dans oiddas
ERR_CON_LDAPOIDDAS	35	Connexion impossible au ldap oiddas
ERR_HAB_EXCLUS	40	Table des exclusions vide ou absente
ERR_SQL_EXCEPTION_APO	50	Toutes SQL Exceptions sur Apogee/Apotest
ERR_SQL_EXCEPTION_GRHUM	51	Toutes SQL Exceptions sur Grhum
ERR_STATUT_INCONNU	80	Statut inconnu
ERR_TYPDEM_INCONNU	81	Type de demande inconnu
ERR_LOGIN_INCONNU	82	Login inconnu et indispensable
ERR_TRT_GRHUM_MAJ_OK	92	Traitement inacheve grhum mis a jour
ERR_TRT_HAB_MAJ_OK	93	Traitement inacheve hab-login_uti mis a jour
ERR_TRT_OID_MAJ_OK	94	Traitement inacheve oiddas mis a jour
ERR_TRT_ORA_CRE_OK	95	Traitement inacheve user oracle apogee cree ou verifie
ERR_TRT_APO_MAJ_OK	96	Traitement inacheve apogee mis a jour
ERR_TRT_CON_VERIFE	97	Traitement inacheve connexions verifiees
ERR_TRT_DEBUT	98	Debut du traitement inacheve donc recommencer possible
ERR_WS_FERME	99	pas de reponse du serveur

L'idée générale est qu'à partir du moment où le service web est sollicité il doit régler le maximum de problèmes seul et n'avouer un échec que s'il n'a pas d'autre choix.

Le traitement se protège lui-même avec l'idée qu'une demande d'exécution peut être présentée plusieurs fois. Donc si un enregistrement est créé la première fois, il se modifie la deuxième fois. Si un enregistrement est déjà supprimé, il est inutile de le supprimer. S'il y a un rollback sur une étape de mise à jour, il n'y aura pas de rollback sur l'étape précédente, car il s'agit de bases de données différentes.

Les exceptions levées

Les exceptions sont détectées et transformées en code retour. Seules deux exceptions sont levées par le côté appelant :

- `java.net.ConnectException` si le service web est indisponible,
- `java.net.SocketTimeoutException` si le WS n'a pas répondu dans le temps imparti.

Les autres exceptions sont :

- `java.sql.SQLException;`
- `javax.naming.NamingException;`
- `apobilitation.exceptions.BadLoginException;`
- `apobilitation.exceptions.BadStatutException;`
- `apobilitation.exceptions.ConnexionsClosedException;`
- `apobilitation.exceptions.TypeDemException;`
- `apobilitation.exceptions.UserExistsException;`
- `apobilitation.exceptions.UserOiddasACreerException;`
- `apobilitation.services.LdapAnnuaireContextException;`
- `apobilitation.services.LdapOiddasContextException;`

Les vérifications préliminaires

Le service vérifie les connexions aux différentes bases de données et s'arrête dès qu'il détecte une connexion fermée. Le service vérifie que la demande est conforme à ce qui est attendu sur les champs type de demande et statut de la demande.

La création, mise à jour ou mise hors service dans apogée

Apogée est créé ou mis à jour selon le type de demande et après tests d'existence. En cas de vraie création il faudra créer un user Oracle aussi du même nom. Une transaction couvre l'ensemble des opérations sur la table UTILISATEUR et les tables UTI_CGS, UTI_CMP, UTI_COLLECTER_CTN. A chaque mise à jour de la table UTILISATEUR, les tables de jointure sont supprimées et recréées, pour prendre en compte les cas où la modification consisterait à supprimer une composante de la liste par exemple.

Il n'y a pas de suppression dans Apogée aussi la suppression consiste à mettre le témoin « en service » hors service.

La création, mise à jour ou suppression dans OIDDAS

Il existe deux façons d'utiliser OIDDAS : soit en utilisant des procédures stockées dans WebUtil, soit en utilisant l'API Ldap. Finalement c'est la deuxième solution qui a été utilisée, la première ne permettant pas de créer un user OIDDAS.

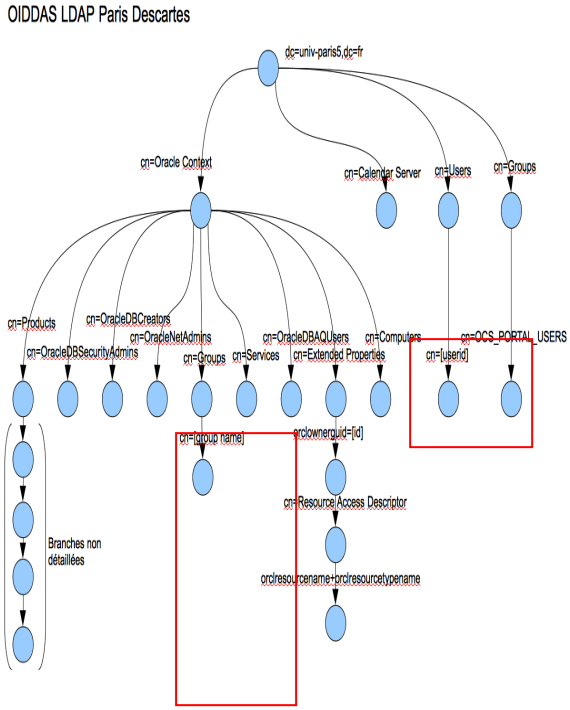


Figure 70 Détail LDAP OIDDAS
Source : MC COLLAS

En cas de création, il faut créer un user OIDDAS ayant le même login que dans l'annuaire. Puis récupérer son « guid » et se créer dans la branche « Extended properties » un nœud « orclownerguid », puis un nœud « Resource Access Descriptor », puis enfin une feuille « orclresourcename » avec le nom de la configuration (prp5 ou Apotest), le user Oracle et le mot de passe pour accéder à Apogée.

Si un user existe déjà, parce qu'il est utilisé aussi par Harpège ou Hélico, il suffira de créer la feuille « orclresourcename ». Et si cette feuille existe déjà parce que créée précédemment, il faut vérifier le mot de passe. Il se peut qu'il ne fonctionne pas. Le seul moyen est de l'essayer. Si Oracle envoie une exception il est faux et dans ce cas on modifie l'existant en forçant sa valeur à une nouvelle valeur générée aléatoirement.

En cas de suppression il suffit d'enlever la feuille « orclresourcename ».

La création, mise à jour ou suppression dans Grhum (le forum)

Une procédure stockée existe déjà qu'il suffit d'appeler en lui passant les bons paramètres.

La création, mise à jour ou suppression dans Hab_login_uti

HAB_LOGIN_UTI est mis à jour en dernier pour refléter les mises à jour faites sur OIDDAS, afin que l'utilisateur voie immédiatement le changement opéré lorsqu'il rafraichît la page habilitation.

La mise à jour du statut dans HAB_DEMANDE_UTI

Enfin on met à jour le statut dans HAB_LOGIN_UTI : soit « X » en cas d'exécution normale, soit « F » en cas d'échec en cours de route. Les classes et interfaces impactées dans ApobilitationWS sont :

- `Webservices.ExecuteDemandeUtiService et webservices.ExecuteDemandeUtiServiceImpl`
- `Webservices.RapportWSBean`
- `Webservices.WSErrorCodeConst`
- `dao.ApoDaoService et dao.ApoDaoServiceImpl`
- `dao.DbOiddasService et dao.DbOiddasServiceImpl`
- `domain.DomainService et domain.DomainServiceImpl`
- `domain.beans.hab.DemandeUti;`
- `domain.beans.hab.HabLoginUti;`
- `exceptions.BadLoginException;`
- `exceptions.BadStatutException;`
- `exceptions.ConnexionsClosedException;`
- `exceptions.TypeDemException;`
- `exceptions.UserExistsException;`
- `exceptions.UserOiddasACreerException;`
- `services.LdapAnnuaireContextException;`
- `services.LdapAnnuaireService;`
- `services.LdapOiddasContextException;`
- `services.LdapOiddasService;`
- `services.LdapUser;`

6.2.22. Notification retour

Lorsque la classe `DemandeListController` reçoit le retour du service web et que celui-ci indique que l'exécution s'est déroulée sans échec, une notification est préparée pour être envoyée par e-mail à celui qui a fait la demande. L'email est préparé dans la méthode `notifierReponse(DemandeUti du)`, puis la méthode de `DemandeFicheController.notifierEmail(…)` est sollicitée pour faire l'envoi. Un message signale à l'utilisateur qu'un email a bien été envoyé.

Les JSF, classes et interfaces modifiées et utilisées dans APOBILITATION sont :

- `Web.controllers.DemandeListController`

- `web.controllers.DemandeFicheController`
- `stylesheets/demandes.jspx`
- `org.esupportail.commons.services.smtp.SmtpService.`

6.2.23. Batch quotidien

Pour lancer une *synchronisation quotidienne*, on a utilisé un « cron ». Pour cela il faut utiliser le produit Quartz d'OpenSymphony [15]qui permet de programmer un trigger. Dans le pom.xml on ajoute la dépendance :

```
<dependency>
      <groupId>opensymphony</groupId>
      <artifactId>quartz</artifactId>
      <version>1.6.3</version>
</dependency>
```

On crée une classe qui va faire le batch `org.esupportail.apobilitation.batch.MyJob.` Une annotation la fait reconnaitre par Spring : `@Repository("myJob").`

Cette classe lance une tâche et dans notre cas elle lance les services web déjà créés , à savoir HelloWorld et SynchroHabLoginUtiService.

```
@Inject
private HelloWorld hello;
public void runHello() {
      logger.debug("I'm the hello Job");
      logger.debug(hello.sayHi("cron task"));
      }
```

Parmi les fichiers Spring on ajoute batch.xml

```
<context:component-scan base-package="org.esupportail.apobilitation.batch"/>
      <task:scheduled-tasks> <task:scheduled cron="${cron.hello}"
      method="runHello" ref="myJob"/>
 </task:scheduled-tasks>
```

`${cron.hello}` sera remplacé à la compilation par la valeur réelle qui est dans le pom.xml

```
<cron.hello>0 0 9 * * ?</cron.hello>
```

Il s'agit d'une expression cron qui représente : 1. Seconds, 2. Minutes, 3. Hours, 4. Day-of-Month, 5. Month, 6. Day-of-Week, 7. Year (optional field). Dans l'exemple Hello, on lance Hello chaque jour à 9 heures.

Finalement on a 3 crons utilisés :

```
<cron.hello>0 0 9 * * ?</cron.hello>
<cron.syncTut>0 0 6-14/7 * * ?</cron.syncTut>
<cron.synchro>0 1 6-14/7 * * ?</cron.synchro>
```

[15] Voir (NetApsys, 2009) et (Developpez.com, 2010).

Ce qui signifie Hello tous les jours à 9 h, syncTut tous les jours à 6h et 13h, synchro tous les jours à 6h01 et 13h01.

Les xml, classes et interfaces modifiées et utilisées dans ApobilitationWS sont :

- batch.MyJob
- Properties/batch/batch.xml
- Pom.xml

La fonctionnalité de *synchronisation des types utilisateurs* n'avait pas été écrite jusqu'à présent. Il s'agit de détecter si un type utilisateur a été créé dans Apogée, dans la table TYP_UTILISATEUR. Si c'est le cas il faut le rajouter à la table `HAB_TYP_UTI` avec des valeurs par défaut.

L'interface `SynchroTypUtiService` et la classe `SynchroTypUtiServiceImpl` sont créées pour cet usage comme un service web supplémentaire, bien qu'il ne soit pas appelé par l'application Apobilitation, mais pourrait l'être éventuellement.

Les xml, classes et interfaces modifiées et utilisées dans ApobilitationWS sont :

- batch.MyJob
- webservices/SynchroTypUtiService et SynchroTypUtiServiceImpl
- domain.DomainService et DomainServiceImpl
- dao.ApoDaoService et ApoDaoServiceImpl
- Properties/batch/batch.xml
- Pom.xml

6.2.24. Authentification

La fonctionnalité d'authentification s'implémente le plus tard possible, car elle limite les possibilités de test. Une fois implémentée, je dois donc m'authentifier même en test avec mon propre login, et pour simuler d'autres identités et cas de figure il faut modifier à la main les tables de test. Donc avant d'implémenter cette phase, une validation de ce qui est déjà implémenté doit être faite.

Dans la période précédente un login, saisi sur la page d'accueil et un rôle permettait de vérifier si l'intervenant faisait partie de la table user et selon le rôle permettait de vérifier si les menus administrateurs étaient bien masqués ou affichés selon les cas, ainsi que les données réservées sur les autres pages. On a pu vérifier si les filtres fonctionnaient correctement selon la composante sur laquelle l'intervenant avait droit de regard.

Esup-commons permet d'implémenter facilement le service d'authentification. Le projet esup-blank est fourni avec les interfaces, classes, et XML suivants :

- services.authentication.Authenticator
- services.authentication.AuthenticatorImpl
- domain.beans.User
- web.controllers.SessionController
- properties/auth.xml
- WEB_INF/web.xml

Chaque fois que le code fait appel à sessionController.getUser(), le service authenticator.getUser() est appelé et fait appel au service AuthenticationService d'esup-commons qui traite la question. Si la personne est authentifiée, son login est renvoyé en retour, sinon la valeur est nulle. Le service AuthenticationService est paramétré dans auth.xml :

```
<bean id="authenticator" lazy-init="true"
class="org.esupportail.apobilitation.services.authentication.Authentic
atorImpl">
<property name="authenticationService" ref="${auth.bean}" />
</bean>

<bean id="servletAuthenticationService" lazy-init="true"
class="org.esupportail.commons.services.authentication.CasFilterAuthen
ticationService">
</bean>

<bean id="OfflineFixedUserAuthenticationService"
class="org.esupportail.commons.services.authentication.OfflineFixedUse
rAuthenticationService">
      <property name="authId" value="" />
      <property name="authType" value="cas" />
</bean>
```

Dans le fichier config.properties on remplace :

```
auth.bean= OfflineFixedUserAuthenticationService
```

par :

```
 auth.bean=servletAuthenticationService
```

Ce qui signifie qu'au lieu d'avoir une valeur sous la forme de constante dans le programme, l'authentification va réellement interroger le serveur CAS. Celui-ci doit être correctement nommé dans le fichier web.xml en remplaçant le texte surligné.

```
<filter>
      <filter-name>CAS Authentication Filter</filter-name>
      <filter-class>org.jasig.cas.client.authentication.AuthenticationFilter
      </filter-class>
            <init-param>
                  <param-name>casServerLoginUrl</param-name>
```

```
                  <param-value>
https://(leServeurCAS).univ.fr/cas/login</param-value>
            </init-param>
            <init-param>
                  <param-name>serverName</param-name>
                  <param-value>http://localhost:8080</param-value>
            </init-param>
      </filter>
      <filter-mapping>
            <filter-name>CAS Authentication Filter</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>
      <filter>
            <filter-name>CAS Validation Filter</filter-name>
            <filter-class>
org.jasig.cas.client.validation.Cas10TicketValidationFilter
            </filter-class>
            <init-param>
                  <param-name>casServerUrlPrefix</param-name>
                  <param-value>https://(leServeurCAS).univ.fr/cas</param-
value>
            </init-param>
            <init-param>
                  <param-name>serverName</param-name>
                  <param-value>http://localhost:8080</param-value>
            </init-param>
      </filter>
      <filter-mapping>
            <filter-name>CAS Validation Filter</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>
</filter>
```

Ensuite WelcomeController doit être modifié: le login n'est plus saisi sur cette page mais lorsque cette page s'affiche il a déjà franchi l'étape de l'authentification dans l'ENT à Paris Descartes. Il ne reste plus qu'à contrôler qu'il fait partie des users autorisés et à envoyer un message poli de refus aux autres.

Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `Web.controllers.WelcomeController`
- `stylesheets/welcome.jspx`
- `properties/auth.xml`
- `WEB_INF/web.xml`

6.2.25. Anomalies

En ce qui concerne la recherche d'anomalies par le web service, pourquoi placer cette fonctionnalité en service web ?

- Le code écrit lors de la phase d'étude de données pour explorer les bases a été repris pratiquement à l'identique. Une partie de ce code utilise des méthodes communes (méthode `lectureOiddas()`) avec le code de synchronisation, en ce qui concerne la lecture d'OIDDAS. Il suffit donc d'exposer ce code comme un service et de l'appeler.

- L'accès à OIDDAS ne se fait jusqu'à présent que dans ApobilitationWS, et non dans Apobilitation, il est plus propre de conserver cette logique.

La démarche consiste à lire OIDDAS pour le transformer en table interne, puis de comparer avec les listes d'utilisateurs Apogée, du forum en recherchant les anomalies du Tableau IV Anomalies et cas normaux des identités (en annexe p 157).

Chaque recherche renvoie un objet EditionWSBean qui contient le nombre de lignes transmises, un tableau de résultats, et une liste d'entêtes. Le programme appelant le service web transforme ce résultat en feuille Excel. Selon la recherche les colonnes peuvent être différentes, la liste des entêtes en donnera le sens.

Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `Webservices.EditionService`
- `Webservices.EditionWSBean`
- `web.controllers.AdminController`
- `stylesheets/adminTemplate.jspx`
- `properties/webservices.xml`

Les XML, classes et interfaces modifiées dans ApobilitationWS sont :

- `Webservices.EditionService et EditionServiceImpl`
- `Webservices.EditionWSBean`
- `Domain.DomainService et DomainServiceImpl`
- `Dao.ApoDaoService et ApoDaoServiceImpl`
- `properties/webservices.xml`

6.2.26. Répercussion dans Apotest

La fonctionnalité de répercuter dans Apotest les demandes faites dans Apogée a été demandée plus tard. L'idée étant la suivante : lorsqu'on crée en production un nouvel utilisateur on veut lui donner accès aussi bien à Apotest qu'à Apogée. Dans OIDDAS chaque droit donné pour un accès revient à créer une entrée avec le nom de la configuration (prp5 ou apotest ou helico ou harpège), le compte oracle et le mot de passe. La table HAB_LOGIN_UTI suit la même logique avec un jeu de colonnes Apogée et un jeu Apotest.

Dans le fichier de paramètres, nous avons entré deux jeux de paramètres pour les configurations 1 (apogée) et 2 (apotest). Lorsque nous travaillons sur l'application de test nous avons une configuration 1 (apotest) et une configuration 2 (apotest). Cette notion de configuration va être réutilisée pour répercuter les actions faites sur Apogée dans Apotest

(l'inverse n'est pas vrai). Chaque étape de l'exécution du service web est doublée pour travailler d'abord sur la configuration1 puis sur la configuration2. La modification introduite a été de passer le numéro de configuration en paramètre des méthodes utilisées. Par exemple :

```
UtilisateurAPO getUtilisateurAPO(String codUti) throws SQLException;
```

est remplacée par :

```
UtilisateurAPO getUtilisateurAPO(int configNb, String codUti) throws
SQLException;
```

Les classes et interfaces modifiées dans ApobilitationWS sont :

- `Webservices.ExecuteDemandeUtiService`
- `Webservices.ExecuteDemandeUtiServiceImpl`
- `Domain.DomainService et DomainServiceImpl`
- `Dao.ApoDaoService et ApoDaoServiceImpl`

6.2.27. Aide

J'ai réalisé des manuels utilisateurs en PDF à partir des copies d'écran. L'un est destiné aux responsables de scolarité, l'autre aux administrateurs. Une option de menu `Manuels` teste le rôle de l'utilisateur connecté et fait un lien sur le bon manuel. Selon les navigateurs le document s'affiche dans un nouvel onglet ou est téléchargé directement en zone de téléchargement. On utilise le contrôleur `sessionController` qui sait déjà identifier le rôle de l'utilisateur en lui ajoutant une méthode `aide()`. Les JSF, classes et interfaces modifiées dans APOBILITATION sont :

- `web.controllers.SessionController`
- `stylesheets/_include/_navigation.jspx`
- `properties/webservices.xml`

6.2.28. Fin de session

L'expiration d'une session après 30 minutes d'inactivité posait problème dans la navigation, car au lieu de signaler une erreur de session expirée, le système signalait une erreur de type `NullPointerException`, parce que le contrôleur devenait subitement nul. Cette erreur était attrapée par `ExceptionController` d'esup-commons et affichait un message laissant croire à un gros bug dans l'application. Or cette exception peut aussi signifier vraiment une erreur.

La version de JSF déclenchant une `javax.faces.application.ViewExpiredException` est postérieure à celle utilisée et donc l'exception n'est pas lancée. La solution trouvée a été d'utiliser l'attribut `destroy-method` de Spring. La méthode « `avantExpiration` » a été

définie comme destroy-method dans le bean AbstractContrextAwareController qui est le contrôleur parent de tous les autres. Lorsque le time-out va arriver cette méthode est appelée.

```
-<bean scope="session" parent="abstractDomainAwareBean" destroy-
method="avantExpiration" id="abstractContextAwareController"
abstract="true">
      <description> An abstract bean to factorize the declaration of beans
of which class inherits from AbstractContextAwareController. </description>
      <property ref="sessionController" name="sessionController">
            <description>The session controller.</description>
      </property>
</bean>
```

Elle lance une exception ExpiredBeanException. Lorsque l'utilisateur appuie sur une touche après expiration au lieu d'avoir un message d'erreur, il obtient une fenêtre lui signalant que sa session est expirée.

Enfin, en s'appuyant sur les habitudes de travail des utilisateurs, le temps disponible avant time-out a été allongé afin de ne pas exaspérer les utilisateurs.

6.3. LES COMMENT-FAIRE ?

6.3.1. Les difficultés rencontrées dans le développement et les solutions trouvées

- Premier projet Maven : Maven est un outil merveilleux de gestion des versions de librairies utilisées, mais il nécessite un temps d'apprentissage conséquent. Heureusement les tutoriels d'ESCV2 [16]y aident beaucoup.
- Premier service web : les tutoriels de Spring, Apache [17] et l'exemple du projet Amethis ont permis de passer d'une connaissance abstraite et conceptuelle à la mise en pratique par l'exemple.
- Premier utilisation de cron avec Quartz : là encore Amethis est venu en exemple[18].
- Percer les secrets d'OIDDAS : grâce à l'outil Gawor d'inspection du Ldap et au programme FormsWebAccess fourni en exemple par le service RESYST.

[16] Voir la formation (Esup Portail, 2011b) , mais voir aussi (Company, 2008) en anglais et sa traduction française en ligne (Maven-guide *et al.*, [s d]).
[17] Voir en ligne (Apache, 2011a), (Apache, 2011b), (McCann, 2009), (Objis, 2011)
[18] Voir aussi (Developpez.com, 2010), (NetApsys, 2009)

- Les problèmes de place mémoire avec Eclipse et Jetty : le lancement de l'application en mode développement sur le serveur Jetty est très pratique mais déclenche régulièrement des erreurs de type `OutOfMemory`. Malgré l'augmentation de place mémoire dans les paramètres d'éclipse, l'erreur reste. D'après les recherches faites sur les forums d'Internet, nous avons compris qu'il s'agit d'un bug de Jetty qui a du mal à nettoyer dans la mémoire les versions de programme qui ont été modifiées à chaud. Toutefois ce problème ralentit fortement le développement.
- L'optimisation des accès à la base de données : les bases de données Apogée et GRHUM sont liées par un DBLink, ce qui est très pratique pour faire des jointures entre les tables de l'une et de l'autre. Mais les temps s'effondrent si l'on ne s'y prend pas bien.
    - Un premier essai de select avec les tables A, B, C dans Apogée et X, Y, Z dans Grhum ne donne pas de bons temps.
    - Un deuxième essai avec constitution de la vue D(X, Y, Z) dans Apogée puis un select avec les tables A, B, C, plus la vue D dans Apogée améliore un peu.
    - Un troisième essai avec constitution de la vue V(X, Y, Z) dans Grhum puis un select de A, B, C dans Apogée et V dans Grhum améliore très sensiblement les performances, car il limite l'usage du dB Link et laisse GRHUM créer la vue dans son propre espace de nom.[19]

6.3.2. L'expérience acquise dans les autres projets

6.3.2.1. L'exemple de rendez-vous avec ESCV1

Etre intervenue en évolution sur une autre application esup-commons a permis de comprendre comment utiliser le framework et comment il était organisé.

[19] Merci à R. Vatré pour la solution !

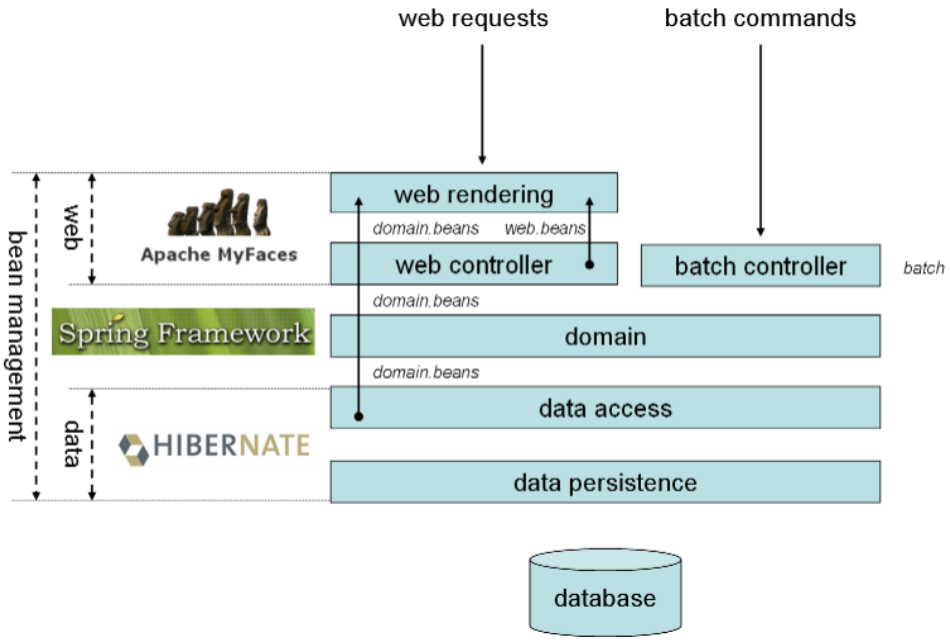


Figure 71 Couches logicielles d'Esup-Commons
Source Renater (https://sourcesup.cru.fr/esup-commons/main/layers.html) (Renater (CRU), 2007b)

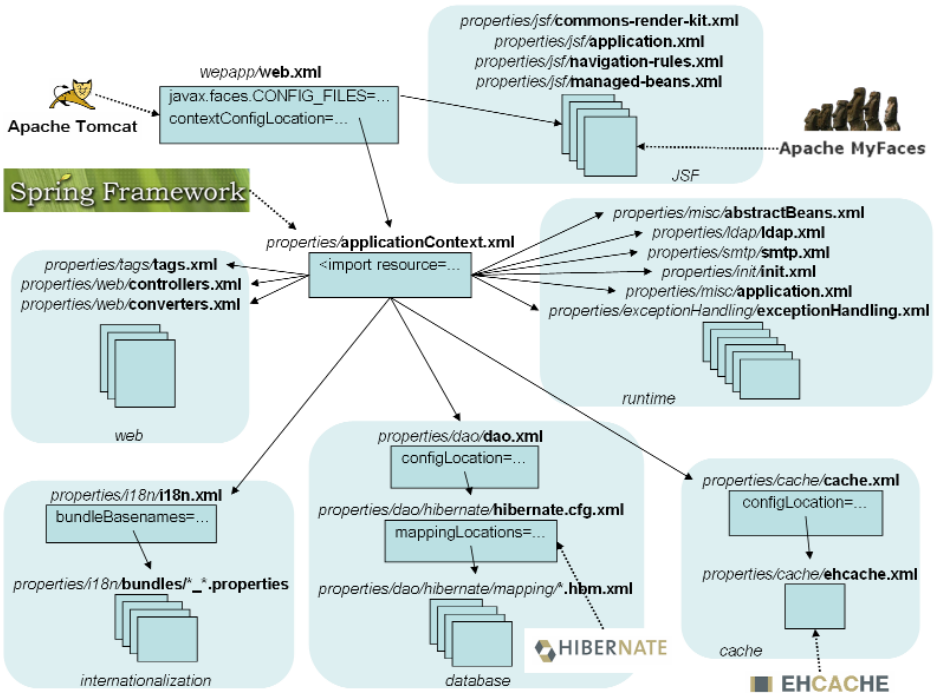


Figure 72 Configuration d'Esup-Commons
Source Renater (https://sourcesup.cru.fr/esup-commons/main/config.html) (Renater (CRU), 2007c)

6.3.2.2. La maitrise d'ESCV2 à travers les projets IEJ

D'autres applications basées sur esup-commons ont été réalisées entre temps sur la version V2 d'esup-commons. Cette version est livrée sous la forme de plusieurs modules et non d'un seul. On choisit les modules qui seront utilisés et les autres peuvent être enlevés.

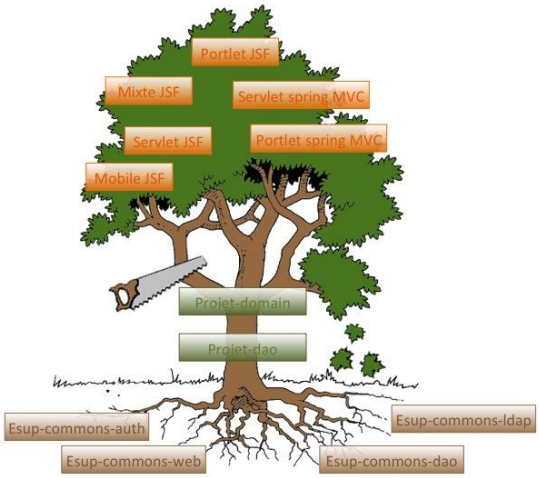


Figure 73 l'arbre de Esup Commons V2
Source Esup-Portail :http://www.esup-portail.org/pages/viewpage.action?pageId=100663437 (Esup Portail, 2011a)

Ainsi en utilisant plusieurs applications ESCV1, ESCV2 abouties et notre version ESCV2 intermédiaire j'ai pu acquérir une certaine aisance dans l'utilisation de Maven.

6.3.2.3. Un exemple de service web à travers le projet Améthis

Le projet Amethis[20] nous a été confié pour intégration, avec les sources. Ce projet utilise un service web qui collecte les données d'Apogée et les réinjecte dans une base de données propre à Amethis. Ce projet a donc fourni un exemple concret sur la manière de mettre en œuvre un web service.

6.3.2.4. Un exemple d'appel de cron à travers le projet Améthis

Le même projet Amethis permet de programmer le lancement de la collecte à une heure donnée. Un cron utilise cette heure pour lancer le travail de collecte, indépendamment de tout appel extérieur. Cet exemple a permis de retrouver quelles API ont été utilisées, comment retrouver la documentation du fournisseur et comment paramétrer le cron.

6.3.2.5. Un exemple d'accès à OIDDAS à travers le projet FormsWebAccess

Le projet FormsWebAccess de l'AMUE permet d'interroger OIDDAS et d'afficher une page de connexion à Apogée à partir du login en masquant la saisie du code utilisateur. Ce programme

[20] Une présentation d'Amethis sur le site de l'Université Européenne de Bretagne (Bretagne, 2013)

utilise des procédures stockées dans WebUtil qui accèdent à OIDDAS et renvoient l'url de la page à utiliser. Le code de ce projet a été utile pour comprendre comment fonctionnait OIDDAS et pour l'utiliser en mode lecture. En revanche il a été insuffisant en mode écriture dans OIDDAS et là il a fallu changer d'approche en utilisant l'API LDAP de préférence.

6.4. DE L'AGILITE EN COURS DE REALISATION

Si le modèle des données, la présentation, l'architecture des composants sont définis au moment de la conception, l'architecture fine n'est pas vraiment terminée au début de la réalisation. Elle s'affine au fur et à mesure que l'on découvre comment doit s'utiliser chaque outil : esup-commons, Spring, cron, etc… Des difficultés techniques imposent de faire des changements au choix initial de programmation afin de contourner l'obstacle.

Dans le même temps, des oublis, voire des erreurs faites au moment de la conception, lorsque des cas imprévus se présentent sont facilement corrigés puisque le concepteur et le développeur sont la même personne. Par exemple lorsqu'il a fallu inclure le service commun de formation continue qui n'est pas une composante mais que nous voulons traiter comme s'il l'était. Il a fallu reprendre l'analyse et la modifier.

Procéder ainsi de façon itérative et corriger au niveau supérieur dès que le besoin est ressenti relève d'un processus agile. Le coût de cette façon de procéder est bien inférieur à celui que nous aurions eu si nous avions persisté dans les choix premiers en attendant que l'utilisateur demande des modifications coûteuses.

6.5. CONCLUSION

Lorsque l'on est le seul développeur java d'un service, il est agréable de pouvoir s'appuyer sur un framework adapté et une communauté, ce qui permet d'appliquer les bonnes pratiques et de se former en le faisant. Le framework esup-commons s'est révélé être un bon choix pour moi.

Chapitre 7. Validation et déploiement

7.1. INTRODUCTION

Dans ce chapitre on traitera plus à fond la question de la validation de façon séparée de la réalisation. En pratique la validation est un exercice continu lorsque l'on est à la fois concepteur réalisateur, validateur et formateur.

Un plan de validation a été rédigé au moment d'entrer dans cette phase, lorsqu'une partie du projet a été mise à la disposition de certains utilisateurs, en version de test. Il n'est pas demandé dans l'établissement et n'a été réalisé que dans le but personnel de clarifier les idées.

7.2. LES PRINCIPES D'AGILITE ET DE VALIDATION ITERATIVE ET CONTINUE

Le principe du cycle en « V » est celui que l'on a en tête tout au long du projet. Ci-dessous le schéma qui correspond au projet, avec les documents produits en italique:

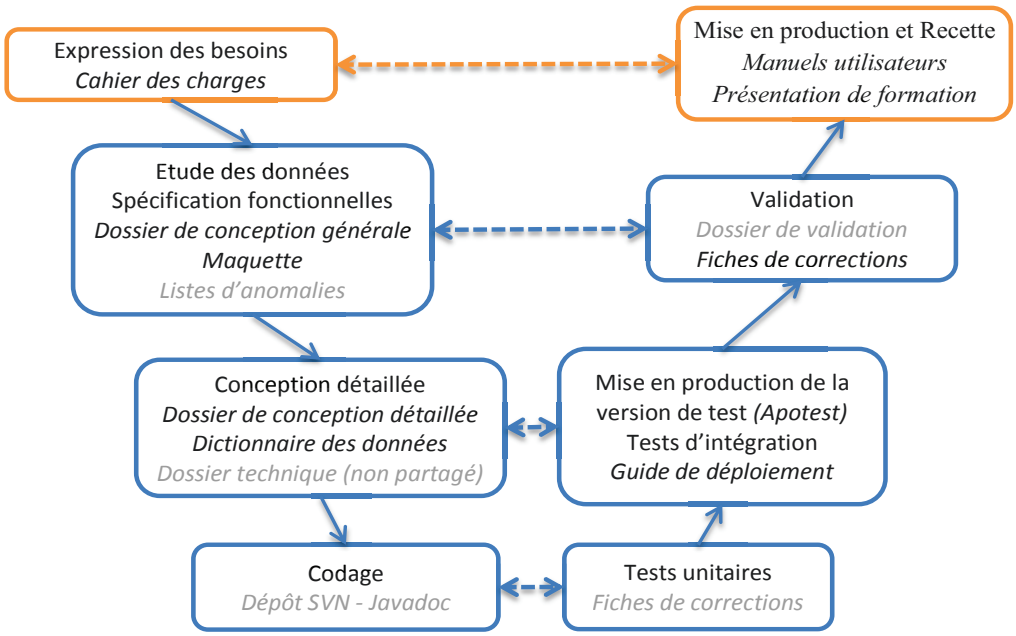


Figure 74 Le cycle en V d'APOBILITATION avec les documents
Source: MC COLLAS

En pratique le déroulé des opérations ressemble davantage à un processus itératif tel que décrit par les méthodes agiles :

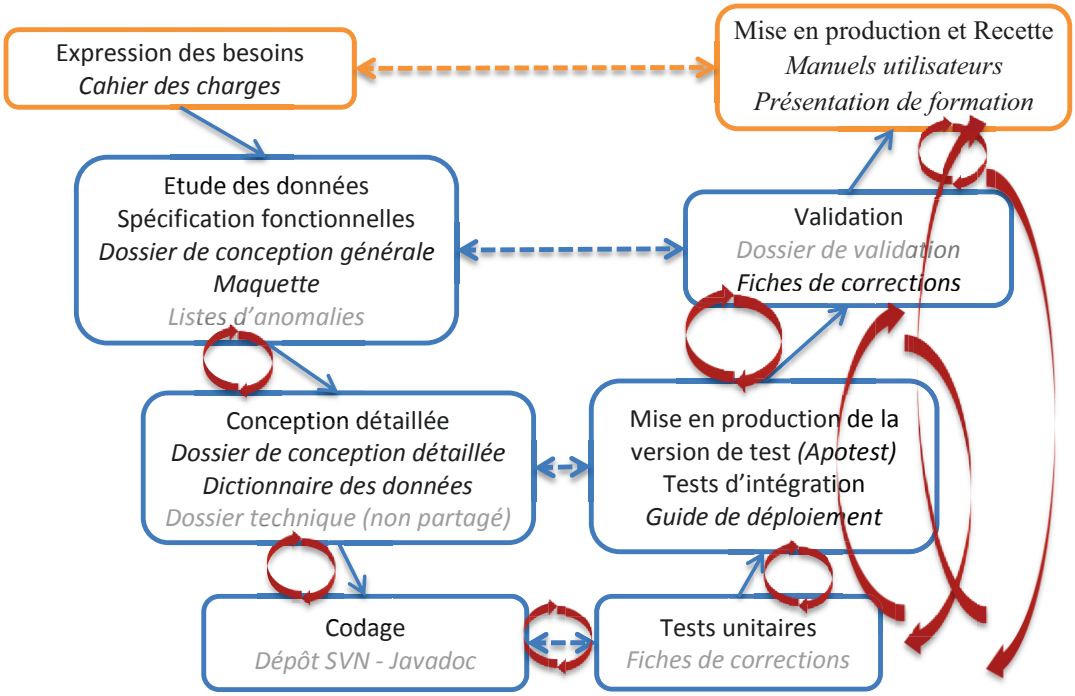


Figure 75 Itérations du projet
Source MC Collas

Comme il a été montré dans la partie réalisation, celle-ci a été conduite fonctionnalité par fonctionnalité. Chaque fonctionnalité a été testée avant de passer à la réalisation de la suivante, sur le plan du fonctionnement. Et ainsi jusqu'à l'ensemble des fonctionnalités. Après quoi d'autres tests ont été réalisés. Dès qu'une partie de l'application a pu être mise à disposition à d'autres personnes, cela a permis de faire progresser en parallèle le codage de certaines fonctionnalités et les tests sur les fonctionnalités livrées.

7.3. LES TYPES DE TESTS DU PLAN DE VALIDATION

Les types de tests mentionnés dans le plan de validation sont :

- les ***tests unitaires*** sont réalisés tout au long du développement et de la maintenance par le développeur pour vérifier le fonctionnement normal de l'application
- les ***tests de déploiement*** pour vérifier le fonctionnement sur le serveur de production

- les ***tests de performance, de robustesse, de vulnérabilité*** peuvent s'ajouter aux premiers en vue d'obtenir une application plus rapide, moins sensible aux montées en charge ou aux attaques hostiles.
- les ***tests de fonctionnement*** sont réalisés à l'aide des fiches de test par le développeur, le concepteur, l'utilisateur pour vérifier le fonctionnement normal et la non - régression de l'application.
- les ***tests de conformité*** sont réalisés par le concepteur et le rédacteur du cahier des charges pour vérifier la conformité au cahier des charges et aux documents de conception.
- les ***tests utilisateurs*** sont réalisés par les utilisateurs finaux ou leurs représentants.

Les personnes intervenant dans la phase de test sont :

- **Corinne Claisse** : en tant que demandeur du projet, rédacteur du cahier des charges, chef du service de développement, et DBA de la base Apogée, a vocation à vérifier la conformité de la livraison aux attentes du cahier des charges.
- **Emmanuel Robert** : en tant qu'utilisateur administrateur de l'outil réalisé et réalisant les opérations à la main avant la livraison de l'outil, en tant que formateur des futurs utilisateurs de la scolarité, a vocation à vérifier que ce produit réalise bien le travail manuel précédent, que son ergonomie sera comprise des utilisateurs de la scolarité. Il a de plus la charge d'initialiser les tables de paramétrages avec un responsable de la DEVU.
- **Sophie Fernandez (succédant à Mariel Fayemi)**: en tant qu'interlocutrice privilégiée de la DEVU avec le service SAUVES, et bénéficiaire du service rendu par l'outil, a vocation à paramétrer les tables du système avec l'assistance d'E. Robert. Elle fait remonter les problèmes rencontrés par elle-même ou les autres utilisateurs mettant en évidence soit des bugs résiduels, soit des incompréhensions de fonctionnement, soit des problèmes non pris en charge par l'application.
- **Marie-Christine Collas** : en tant que rédactrice des dossiers de conception et réalisatrice de l'application, j'ai vocation à réaliser les tests unitaires, de non régression, de performances, de déploiement et autres tests techniques, à vérifier la conformité aux documents de conception et au cahier des charges.

7.4. LES TESTS UNITAIRES EN CONTINU

Les tests unitaires comprennent :

- élimination des bugs, des arrêts sur exception, exécution sans erreurs
- vérification dans les tables avant et après écriture
- vérification du code
- conformité à la maquette
- tests de concurrence d'accès

Les outils utilisés sont les fonctionnalités de débogage d'Eclipse, la fonctionnalité d'interception d'exception d'esup-commons envoyant un e-mail avec informations utiles à chaque arrêt sur exception, dans certains cas l'utilisation de programmes JUnit, les outils d'inspection de base de données (SQL Developer) et de Lap (Gawor), l'inspection des logs écrits par les programmes s'exécutant.

7.5. LES TESTS DE DEPLOIEMENT

Le but est de vérifier que l'application fonctionne aussi bien sur le serveur en environnement de production que sur le serveur local.

Lors de la réalisation et la mise au point on a utilisé le serveur Jetty incorporé avec Maven sur le port 8090. On va porter l'application dans les environnements suivants avec les objectifs de tests suivants, en définissant des profils :

- Sur un Tomcat local port 8080 et avec une configuration de test (base Apotest) :
  - On vérifie que l'application se comporte sur Tomcat comme sur Jetty. Par exemple le test pour savoir si une donnée est nulle sur Tomcat donne un résultat différent sur Jetty dans certains cas.
  - On vérifie que les paramètres dynamiques passés à l'application sont corrects, tels que les chemins relatifs, ou les noms de fichiers,
  - On vérifie que la redirection sur le serveur CAS et le retour sur le serveur d'application utilise les bonnes url.

- Sur un serveur local port 8080 et avec une configuration de production (base Apogée) :
  - En plus des vérifications précédentes, on vérifie que les informations d'accès aux bases pointent bien vers les bases de production.
  - On vérifie les temps de réponse sur les volumes réels.
- Sur un serveur d'application distant ayant un autre port et avec une configuration de test
  - On vérifie que l'application se déploie bien sans oubli de librairies indispensables, et que toutes les dépendances retrouvent ce dont elles ont besoin (images, css, librairies…).
  - Après avoir demandé au service de production d'incorporer l'application (versions de test et de production) dans leurs tables de redirection, on vérifie que l'url http://app.parisdescartes.fr/test-apobilitation fonctionne en renvoyant bien l'application
  - On vérifie la navigation
  - On vérifie que l'on est bien sur la base de test
  - On vérifie que les instructions de logging s'écrivent dans les bons fichiers et qu'ils ne sont pas restés en version debug
- Sur le serveur d'application de production distant et avec une configuration de production (http://app.parisdescartes.fr/apobilitation)
  - En plus des tests précédents, on vérifie que l'on est bien sur une configuration de production
  - On vérifie que le portail ENT fait bien appel à cette version.

Les tests de déploiement ne sont pas faits une fois pour toutes, mais doivent être refaits à chaque fois que l'on met une nouvelle version en production, il y a toujours une possibilité d'erreur, comme par exemple un paramètre resté en position de test.

Lorsque la première version utilisable en test est mise sur le serveur distant, cela permet aux autres utilisateurs participant aux tests de pouvoir tester l'application de leurs points de vue respectifs.

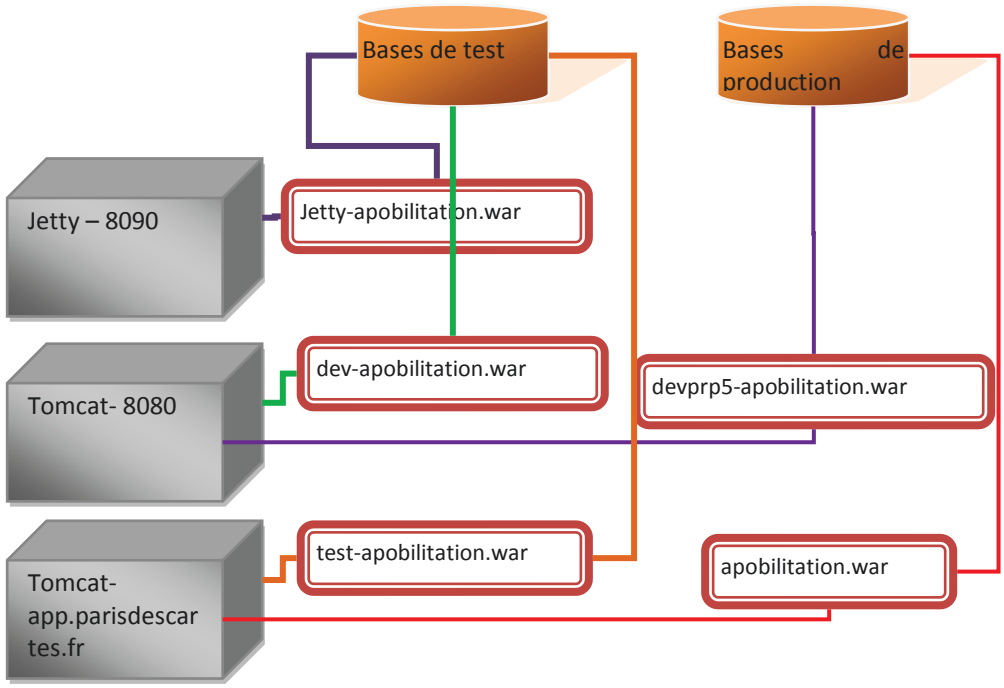


Figure 76 Les profils de déploiement en test et production
Source : MC COLLAS

Le service web a un serveur d'application particulier qui a un port différent. Il n'est pas représenté sur la figure pour éviter la surcharge.

7.6. REDACTION DES DOSSIERS DE DEPLOIEMENT

Le manuel de déploiement contient les instructions permettant à un autre développeur ou un administrateur système de pouvoir réinstaller sur son poste les sources des applications et de pouvoir les redéployer à nouveau. Une fois les premiers déploiements exécutés avec succès, il est rédigé immédiatement.

Le manuel de déploiement est aussi destiné à l'équipe système qui gère les serveurs.

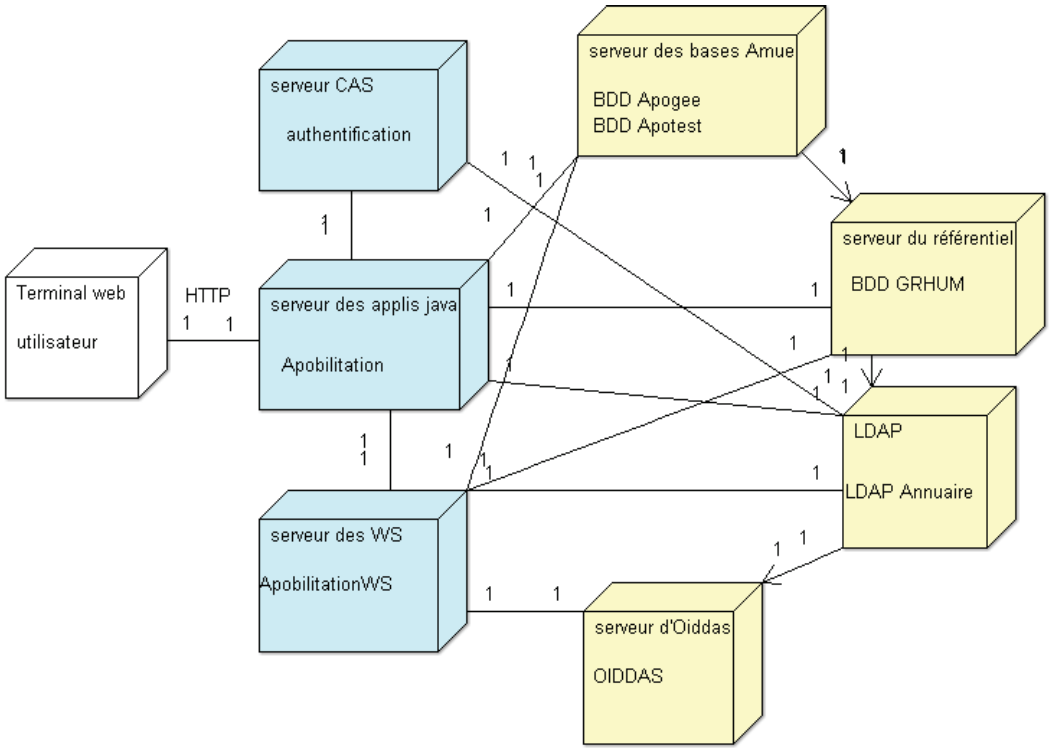


Figure 77 Diagramme de déploiement
Source : MC COLLAS

7.7. LES TESTS DE PERFORMANCE

Le but est de vérifier les problèmes liés aux volumes qui se répercutent sur les temps de réponse.

Le temps d'accès à la page Habilitations a été considérablement réduit après avoir :

- Changé la table visible par une table à plat avec select en JDBC plutôt qu'une requête JPA
- Fait exécuter les jointures par un dB Link dans Grhum plutôt que dans Apogée.
- Il faut vérifier que la table est rendue en moins de 10 secondes.

L'accès au service web :

- Si la base de données est ouverte et le service web ouvert, l'exécution d'une demande prend moins de 10 secondes.
- L'appel à la synchronisation dure environ 50 à 60 secondes. Au-delà ce n'est pas normal.

Normalement la synchronisation est appelée par un cron et non par l'utilisateur, car il s'agit d'un traitement batch qui dure un certain temps de l'ordre de 50 à 60 secondes. Toutefois on laisse la possibilité à l'administrateur de faire une synchronisation intermédiaire dans la

journée. Il a fallu demander au service système d'augmenter le temps de réponse maximum pour une réponse du serveur, car par défaut il était trop court.

7.8. LES TESTS DE FONCTIONNEMENT

Le but de ces tests est de vérifier le bon fonctionnement de l'application après chaque mise à jour, afin de vérifier que la mise à jour n'a pas introduit de régression quelque part. 13 tests ont été définis avec leurs fiches de test qui sont :

"

LISTE·DES·FICHES·DE·TEST¶

Figure 78 Liste des fiches de tests
Source : MC COLLAS

L'idée générale est de vérifier la navigation et les fonctionnalités de chaque page sans déclencher un plantage de l'application. Ces tests se font en continu sur le poste de développement et sont refaits sur chaque installation déployée sur le serveur, l'installation de production en dernier.

7.9. LES REMONTEES DE BUGS ET REMARQUES DE L'EQUIPE

Une version de test installée sur le serveur distant (test-apobilitation) a été mise à la disposition de Corinne Claisse et d'Emmanuel Robert dès que les fonctionnalités de paramétrage ont été disponibles, et mise à jour régulièrement à chaque déploiement. Cela a permis à Emmanuel Robert de pouvoir tester l'outil en fournissant à la base de données les profils dont l'application a besoin. Il n'a pas manqué à l'occasion de signaler les bugs et erreurs rencontrés. Une liste des corrections à faire a été mise en place.

	A	B	C	D	E	F
	date	origine	description de l'anomalie	page	date de correction	mise en prod
	25/04/2012	erobert	probleme de remise à zero du libellé et du résumé de profil dans nouvelle demande	nouvelle demande	27/04/2012	02/05/2012
	25/04/2012	erobert	dans recherche ldap, manque etoile pour data obligatoire	nouvelle demande	26/04/2012	02/05/2012
	26/04/2012	erobert	on ne peut pas modifier le code profil	profil	26/04/2012	02/05/2012
	26/04/2012	erobert	erreur si le résumé fait plus que 200 caractères	type utilisateur	26/04/2012	02/05/2012
	26/04/2012	erobert	? Inutile sur la page profil	profil	26/04/2012	02/05/2012
	26/04/2012	erobert	ajouter un motif de refus et notification du refus	demande en cours	11/05/2012	15/05/2012
	26/04/2012	erobert	ajouter version de test en test	header	26/04/2012	02/05/2012
	26/04/2012	erobert	total page faux dans les manuels	manuels	26/04/2012	02/05/2012
0	26/04/2012	mcollas	amelioration des messages d'erreur	toutes	26/04/2012	02/05/2012
1	26/04/2012	erobert	filtre demandes ne marche pas	demande en cours	02/05/2012	02/05/2012
2	15/05/2012	mcollas	ajouter test si operation manuelle créer user	demande en cours	15/05/2012	15/05/2012
3	15/05/2012	claisse	changer esup-apobilitation en apobilitation	pom.xml	15/05/2012	15/05/2012
4	14/05/2012	mcollas	optimisation requete SQL	habilitations	14/05/2012	15/05/2012
5	16/05/2012	claisse	enlever Pascal Aubry	a propos	16/05/2012	16/05/2012
6	16/05/2012	claisse	enlever demande de formation	a propos	16/05/2012	16/05/2012
7	16/05/2012	claisse	adresse de retour: apobilitation@parisdescartes.fr	a propos	16/05/2012	16/05/2012
8	18/05/2012	claisse	supprimer operation manuelle creation user dans oiddas	webservice	25/05/2012	25/05/2012
9	30/05/2012	claisse	créer dans apogee et apotest	webservice	04/06/2012	04/06/2012
0	05/06/2012	erobert	erreur sur le bouton soumettre la demande	nouvelle demande fiche	12/06/2012	12/06/2012
1	03/07/2012	mcollas	signaler absence de validateur	utilisateurs autorisés	04/07/2012	05/07/2012
2	03/07/2012	mcollas	filtre nouvelle demande sur code structure: prendre la structure correspondant à la composante de l'intervenant et non celle de l'annuaire	nouvelle demande	04/07/2012	05/07/2012
3	06/07/2012	dhoudet	reduire le logo quand la fenetre est dans l'ent	header	06/07/2012	17/07/2012
4	10/07/2012	claisse	masquer le login aux responsables de scol: information confidentielle	habilitations, demandes, nouvelle demande	11/07/2012	17/07/2012
5	10/07/2012	claisse	remplacer par nom prénom	demandes	11/07/2012	17/07/2012
			remplacer label "informations annuaire oiddas" par			

Figure 79 Liste des corrections
Source : MC COLLAS

Cette liste permet de vérifier l'origine de la demande, l'exécution de la correction, la date de mise en production.

Par la suite des colonnes concernant la mise à jour des manuels a été rajoutée, car certains peuvent nécessiter une mise à jour, comme par exemple le manuel utilisateur si le rendu visuel d'un écran a changé.

7.10. LA VALIDATION FACE AUX DOSSIERS DE CONCEPTION GENERALE ET DETAILLEE

Avant de décider que l'application est achevée, il est nécessaire de vérifier en relisant les dossiers de conception que tout a été fait, conformément aux dossiers de conception générale et détaillée.

Certaines modifications apportées à l'application n'avaient pas lieu d'être, ou des oublis ont été constatés, ce qui renvoie sur une nouvelle itération : développement – tests.

D'autres modifications ont été apportées qui doivent être conservées soit pour des raisons techniques, soit par demande du supérieur, soit parce que l'oubli était au niveau de la conception. Dans ces cas de figure, la documentation est mise à jour (conception générale, dictionnaire des données).

De la même manière le dossier de conception générale a été mis à jour.

Tous ces dossiers sont versionnés ce qui permet de savoir où des modifications ont été apportées. Il est tout de même intéressant d'avoir une documentation à jour.

7.11. LA PRESENTATION DE L'APPLICATION A LA DEVU ET SON APPROPRIATION

La rentrée universitaire est synonyme de mutations et nouveaux recrutements au sein du personnel dont certains doivent être habilités pour Apogée. A ce moment-là la DEVU comme utilisateur a ressenti l'importance de pouvoir disposer de ce nouvel outil et de se l'approprier, malgré un emploi du temps particulièrement chargé dans les services de scolarité en raison des inscriptions universitaires.

Sophie Fernandez nouvellement arrivée à la DEVU a reçu la mission d'être le principal interlocuteur des responsables de scolarité et d'être le validateur de leurs demandes d'habilitations. Elle a donc été formée sur l'outil la première, ce sont des cas réels qui ont été exécutés, vers mi-septembre 2012.

Les types utilisateurs retenus ont été enregistrés, les profils aussi, et les vacataires de scolarités habilités à l'aide de cette application.

Durant cette phase quelques derniers bugs sont remontés correspondant à des cas de figure rares ou à une utilisation imprévue de l'outil, qui ont été corrigés rapidement.

7.12. LA FORMATION DES UTILISATEURS FINAUX

Il était prévu une formation début octobre 2012 pour les responsables de scolarité qui devaient utiliser APOBILITATION (2 par scolarité). A cet effet une présentation a été réalisée comme support de formation.

Malheureusement les responsables de scolarité sont des personnes très occupées et trop peu de gens se sont inscrits. Ceci a eu pour effet l'annulation de la formation.

En revanche, grâce aux manuels utilisateurs et à la présentation l'ensemble des responsables de scolarité ont pu s'auto-former en suivant les supports et ont trouvé cela facile et intuitif. Le suivi des demandes d'habilitations a montré que la plupart en ont profité pour nettoyer les listes en enlevant les habilitations périmées, ce qui était l'un des buts de l'opération, et pour demander d'autres habilitations.

Après la levée des derniers bugs résiduels jusqu'à mi-octobre, l'application a fonctionné sans souci. Les cas de bugs les plus significatifs ont été joints au dossier de validation pour penser à vérifier la non-régression en cas d'évolutions futures.

7.13. LE RESULTAT FINAL

Voici les écrans que peuvent voir les utilisateurs.

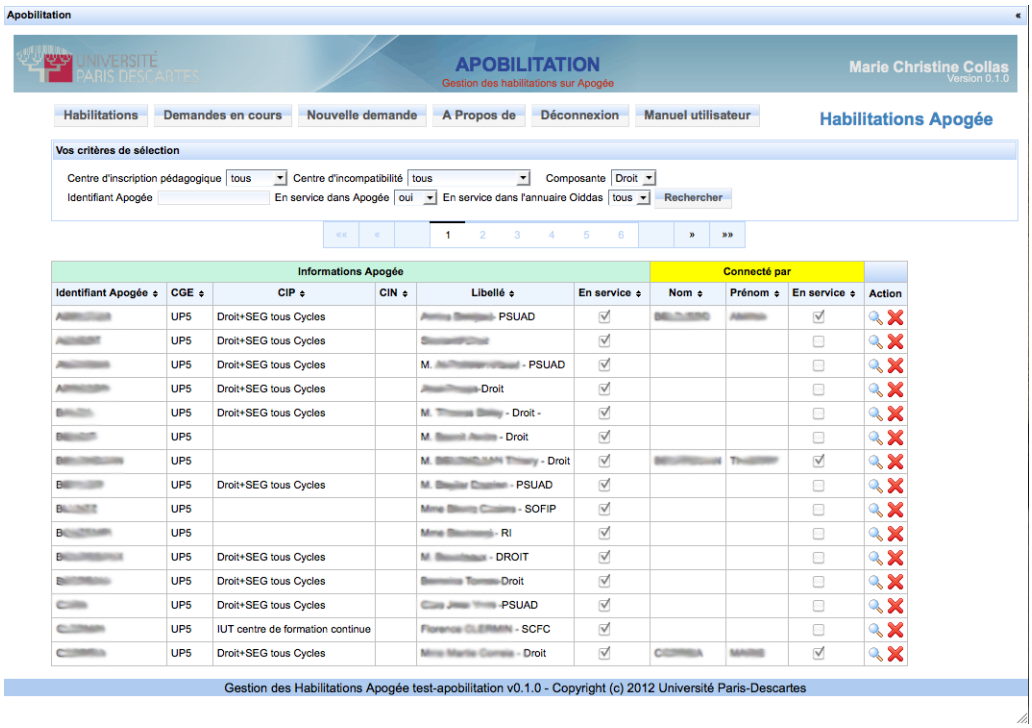


Figure 80 La page habilitation d'Apobilitation

Ci-dessus la page qui s'affiche après l'authentification. Un responsable de scolarité ne verra que sa composante. S'il clique sur le bouton voir d'une ligne, il obtient la page ci-dessous, reflet du contenu d'Apogée.

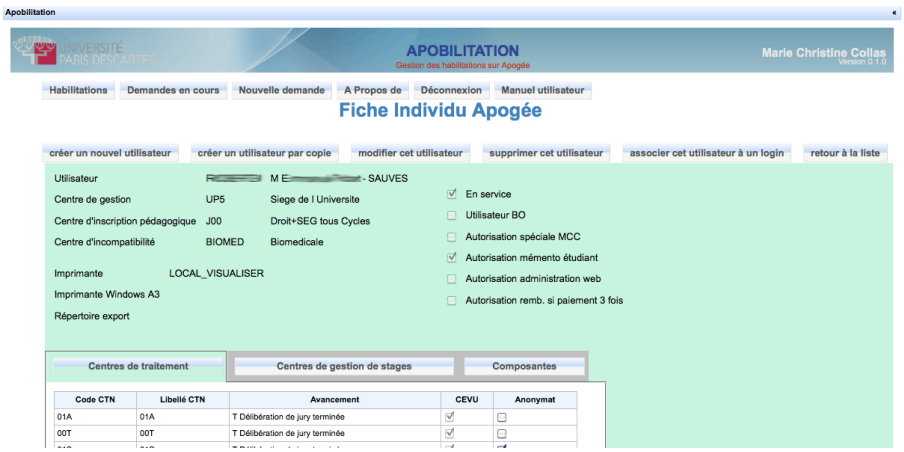


Figure 81 La fiche individu d'Apobilitation

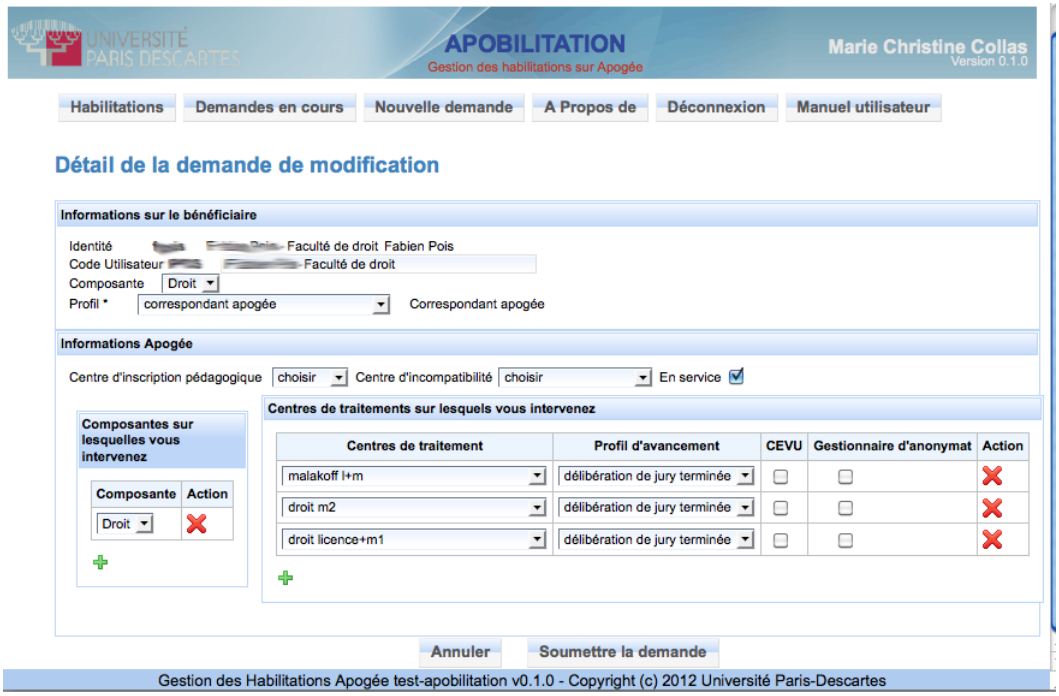


Figure 82 Page détail d'une demande d'Apobilitation

Lorsqu'une demande de création ou modification est faite, la page ci-dessus doit être saisie. L'appui du bouton « soumettre la demande » renvoie à la liste ci-dessous.

Figure 83 Page liste des demandes avec notification d'Apobilitation

On signale l'envoi d'une notification au directeur de la DEVU. Celui-ci se rend sur cette même page qui contient quelques fonctionnalités supplémentaires, comme le pouvoir d'accorder ou refuser la demande (ci-dessous colonnes accorder et refuser).

A l'appui du bouton accorder la demande est envoyée au service web qui exécute tout le processus d'habilitation et renvoie un rapport qui s'affiche au-dessus de la liste, ainsi que des notifications (figure ci-dessous).

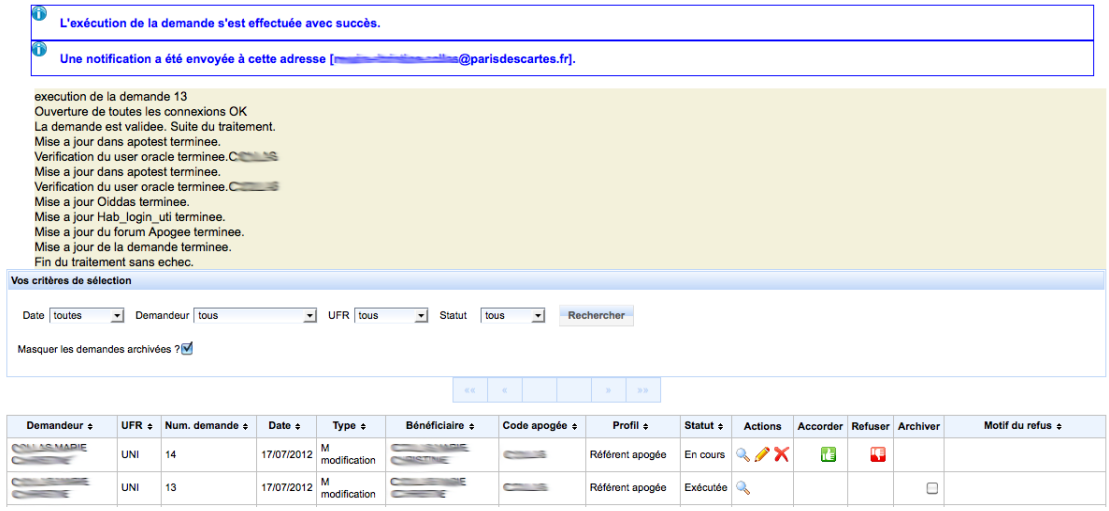


Figure 84 Page de rapport d'exécution et notifications après une validation (extrait) d'Apobilitation

L'administrateur dispose en outre de tout un menu d'administration (Figure 85).

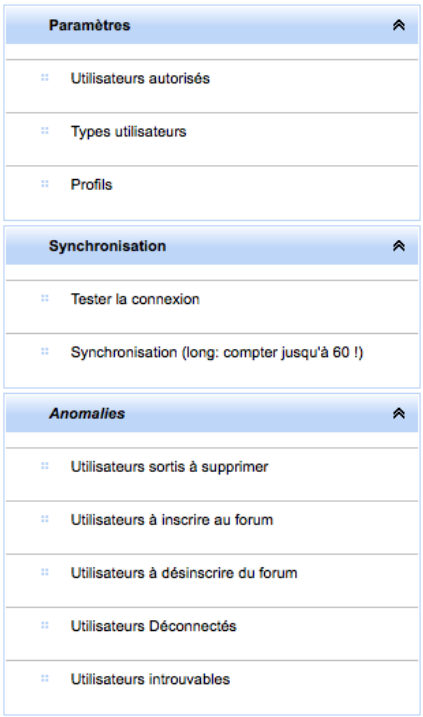


Figure 85 Menu administration déployé

Figure 86 Menu et aide d'administration

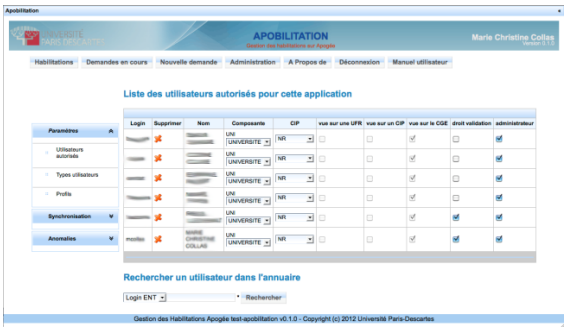


Figure 87 Page des utilisateurs d'Apobilitation

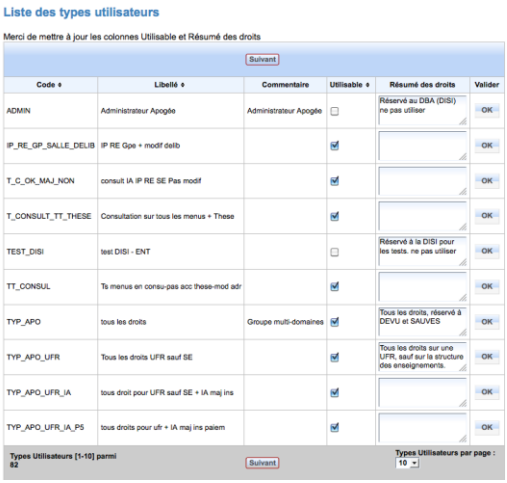


Figure 88 Page des types utilisateurs

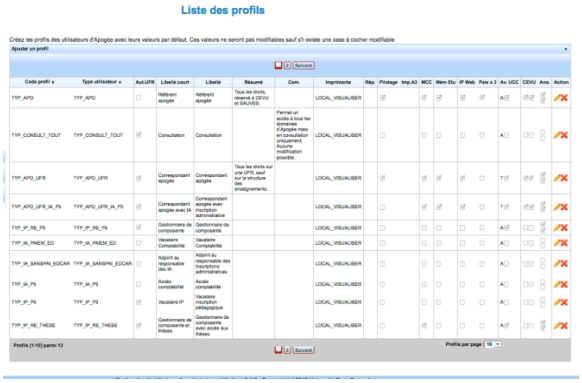


Figure 89 Page des profils

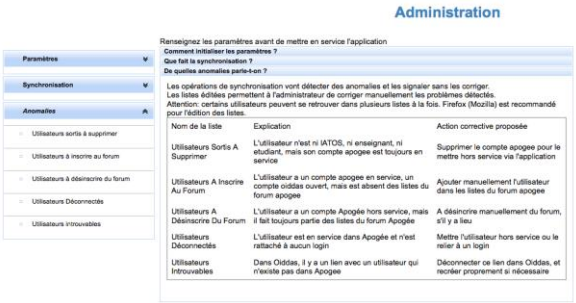


Figure 90 Lancement des listes d'anomalies

7.14. CONCLUSION

Il est toujours préférable de pouvoir compter sur une équipe de testeurs autres que les développeurs, qui ont le recul suffisant et l'esprit suffisamment affûté pour penser à des situations qui n'arrivent presque jamais. Malheureusement ce n'est pas toujours possible et il ne reste que la rigueur personnelle pour compenser.

La démarche itérative trouve ici toute sa pertinence.

Chapitre 8. Conclusion

8.1. BILAN DU POINT DE VUE DE L'UNIVERSITE

L'université Paris-Descartes a maintenant une application qui permet d'automatiser le processus d'habilitation. Cela concerne chaque année une centaine de personnes environ, entre les nouveaux arrivants et de nombreux vacataires pendant la période des inscriptions.

Les avantages et bénéfices apportés par APOBILITATION sont ceux de tout système de gestion d'identité (Fischer, 2011), et donc les suivants :

- Visibilité : les responsables de scolarité et la DEVU sont repositionnés au centre, ce sont eux qui décident qui doit être habilité et qui ont la main sur l'exécution. C'est une avancée majeure puisqu'ils disposent d'une visibilité (listes de collaborateurs) qu'ils n'avaient pas avant.
- Gain de productivité : les coûts d'administration sont réduits. Si le soutien du service SAUVES a été encore important cette année pour former à cette application, M. Robert déjà dispose de temps disponible pour d'autres usages.
- L'efficacité et la réactivité sont améliorées. La conséquence est la réduction des délais de mise à disposition des droits d'accès et une réduction des sources d'erreur.
- Cohérence : Petit à petit les gestionnaires suppriment des listes les anciens collaborateurs en les mettant hors service, ce qui contribue aussi à rendre la base de données plus saine et plus cohérente. La cohérence est maintenue entre les systèmes considérés par l'application.
- La sécurité est améliorée par un meilleur contrôle des droits accordés.
- L'application permet d'avoir une traçabilité des demandes d'habilitation.

Un axe d'amélioration pour l'université serait de considérer la gestion des identités sur un plan supérieur et sur un plus large périmètre. Ce qui a été fait pour Apogée ne pourrait-il pas être considéré pour d'autres applications comme Harpège (gestion des ressources humaines) ou Hélico (gestion des heures complémentaires) datant de la même époque ?

8.2. BILAN DU POINT DE VUE DU PROJET

Ce projet a coûté 213 jours (soit 10,7 mois) pour moi, plus quelques jours pour mes collègues. Quel est le retour sur investissement ?

L'application est destinée à un public réduit : les responsables de scolarité. Elle est basée sur le logiciel Apogée dont on dit qu'il est en fin de vie. Faire les choses à la main ne prenait pas tant de temps pour une personne expérimentée. Autant d'arguments de la part de ceux qui ne voyaient pas l'intérêt d'un tel projet.

Pourtant ce projet a aussi des atouts :

- Il a permis de dégager le service SAUVES d'une charge de saisie qui ne devrait pas être celle d'un service informatique.
- Il a permis de réfléchir à nos processus de gestion d'identité et d'avoir une vue clarifiée sur la question.
- L'expérience acquise sur ce projet est capitalisée sur d'autres projets en relation avec l'univers d'Apogée d'une part, et le framework esup-commons d'autre part.
- Il est fortement probable qu'Apogée ne soit pas remplacé par un autre logiciel, mais par une version ultérieure d'Apogée. L'utilisation d'un service web permet de pouvoir suivre les évolutions dans une certaine mesure. A ce moment-là on pourra réfléchir à un couplage encore plus lâche avec Apogée.
- Enfin il est possible que d'autres universités soient intéressées par cette solution et qu'une version interuniversitaire voie le jour.

Il y aurait des pistes d'amélioration de ce projet, dont l'idée m'est venue après et avec l'acquisition de nouvelles compétences :

- Le découplage d'avec Apogée pourrait être renforcé davantage. L'occasion d'y réfléchir viendra peut-être lors du dévoilement du nouvel Apogée.
- Le paramétrage pourrait être mieux réfléchi en ayant à l'esprit une application utilisée par plusieurs universités, localement ou partagée sur le cloud.
- Certaines anomalies signalées par liste informative pourraient être traitées de façon automatique sans qu'il y ait besoin d'une demande explicite d'un responsable de scolarité suivie d'une validation, comme par exemple la mise hors service en cascade des comptes orphelins, option qui n'était pas souhaitée au départ.

8.3. BILAN DU POINT DE VUE PERSONNEL

Du point de vue personnel, ce projet, déjà fortement avancé d'avant d'être retenu comme projet de mémoire CNAM, a été pour moi un sujet de fierté. L'opportunité de réaliser à la fois la conception, la réalisation et la validation m'a donné la stature de responsable de projet et beaucoup appris.

Du point de vue technique, tout en m'appuyant sur des techniques déjà acquises, le projet m'a donné l'opportunité de découvrir d'autres techniques. J'ai pu augmenter mes compétences, capitaliser de l'expérience aussi bien en architecture et en développement, qu'en connaissance du métier de l'université. Je dispose maintenant de statistiques personnelles de temps de réalisation qui vont me permettre de faire des estimations plus fines.

La rédaction de ce mémoire et l'obligation de réfléchir à la problématique rencontrée par mes supérieurs a été un moyen de prendre du recul et de considérer le système d'information dans son ensemble, de retracer l'historique du système d'information à l'université, et de m'élever au-dessus de la simple réalisation.

La formation suivie au CNAM dans la filière IRSM (Ingénierie de Réseaux, Systèmes et Multimédias) ne correspond pas exactement à mon travail quotidien qui relèverait plutôt d'une filière AISL (Architecture et Intégration des Systèmes Logiciels) ou ISI (Informatique et Systèmes d'Information) aussi il est fortement probable que je sois passée à côté d'autres solutions, mais je n'ai pas ressenti cela comme une gêne insurmontable, la formation d'ingénieur poussant constamment à rechercher par soi-même l'information manquante et à identifier ses besoins en formation.

Annexes

1. Organigramme de la DISI
2. Partage du temps entre les projets
3. Organigramme général
4. Cahier des charges
5. Formulaire de demande d'habilitation
6. Types d'anomalies

1. ORGANIGRAMME DE LA DISI

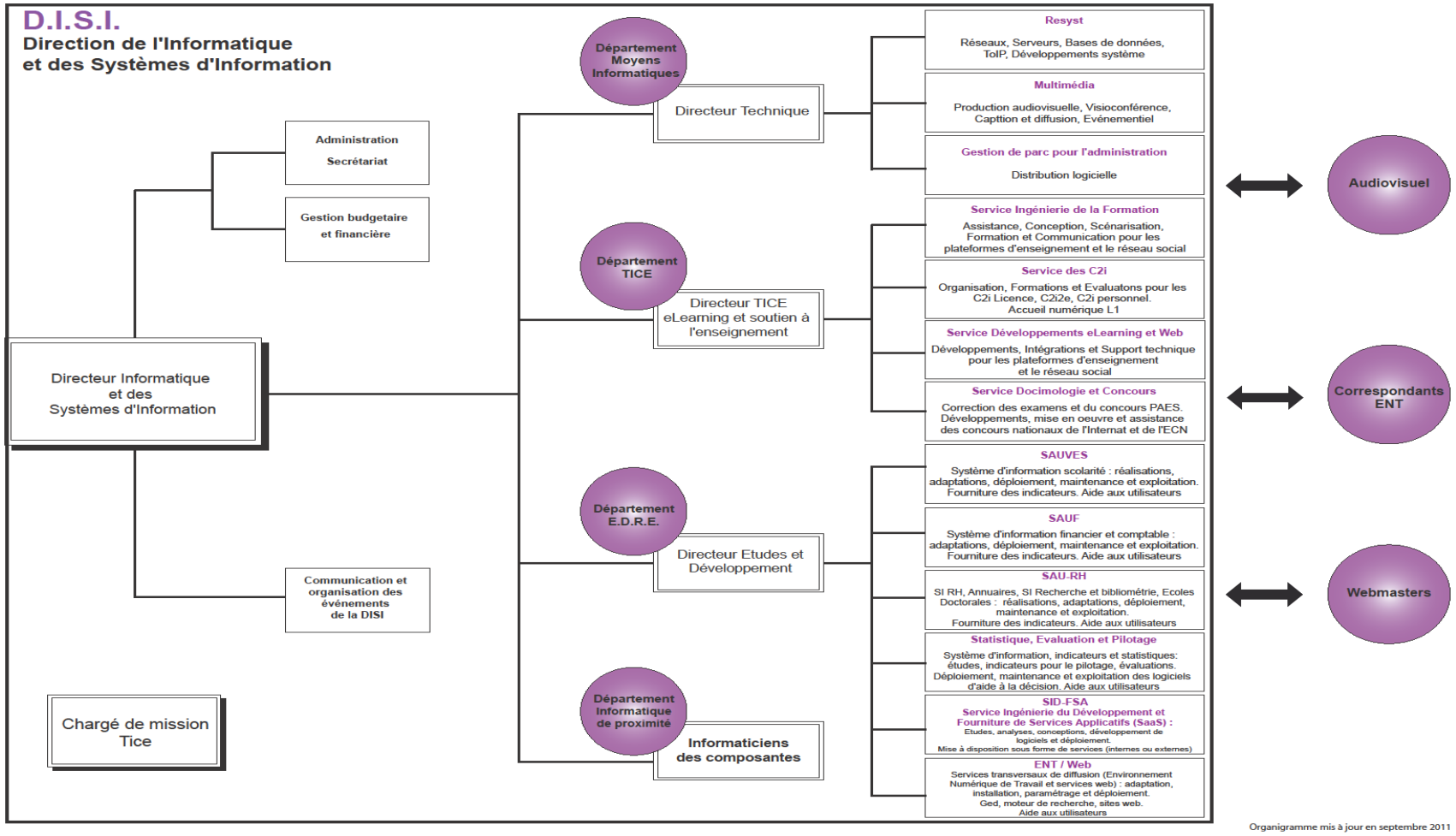


Figure 91 Organigramme de la DISI Source : Université Paris Descartes

2. PARTAGE DU TEMPS ENTRE LES PROJETS

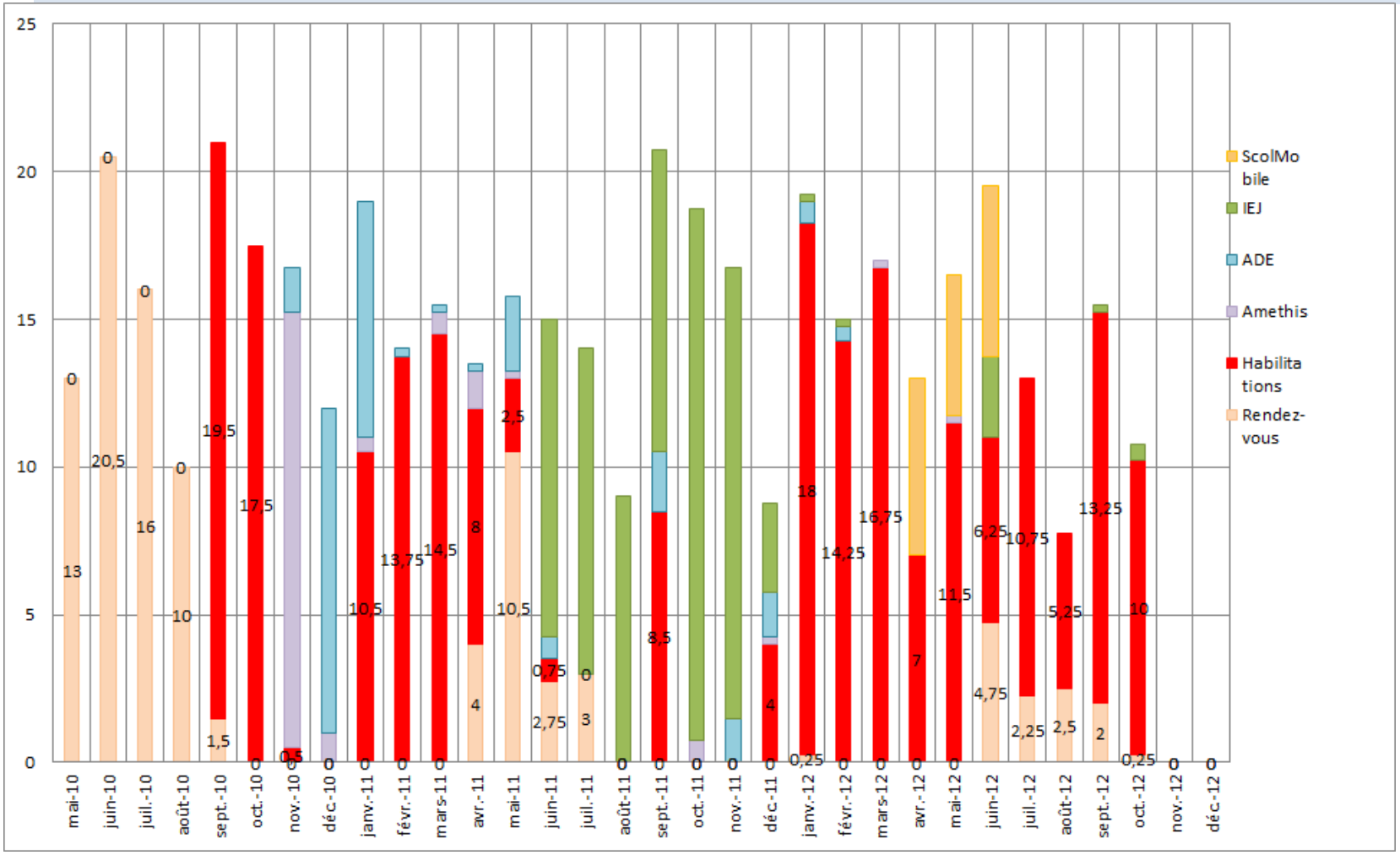


Figure 92 Répartition du temps par projet
Source : MC COLLAS

3. ORGANIGRAMME GENERAL

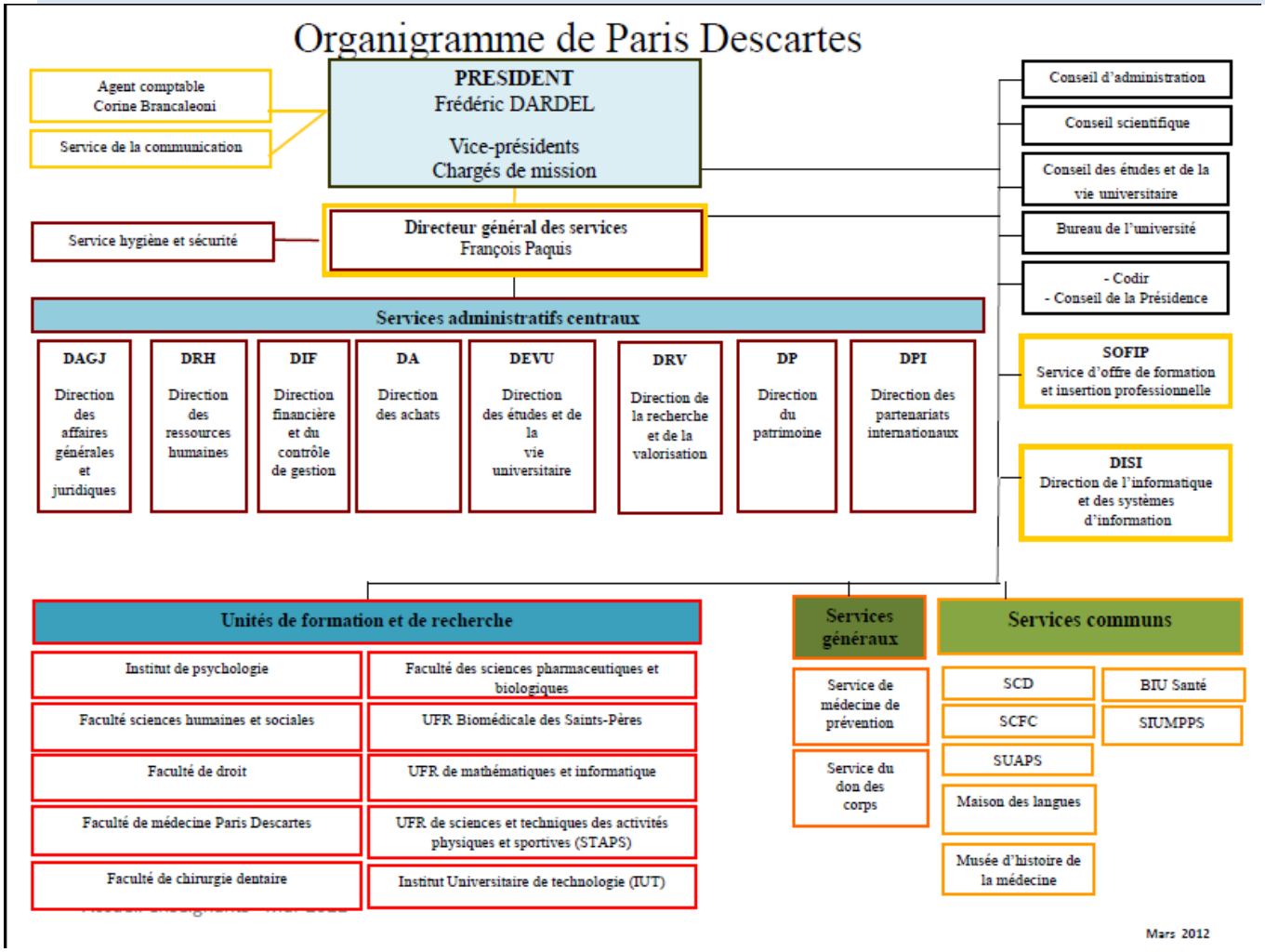


Figure 93 Organigramme général
Source Université Paris Descartes

4. CAHIER DES CHARGES

Application gestion des HABILITATIONS

Objectif : créer une application web permettant aux responsables de scolarité dans les composantes de demander des habilitations Apogée pour leurs collègues. Après validation par le directeur de la DEVU, création des droits pour ces utilisateurs.

1) Définition d'une habilitation dans Apogee.

Une habilitation se compose :
- d'un identifiant
- d'un type d'utilisateur
- d'un centre de gestion CGE
- d'un centre d'inscription pédagogique CIP
- d'un centre d'incompatibilité
- d'un ou plusieurs centres de gestion de stage
- d'une ou plusieurs composantes
- d'un ou plusieurs centres de traitement CTN
- d'un compte Oracle

a) type utilisateur

Un type d'utilisateur définit les droits d'accès aux modules, aux traitements et aux traitements personnalisés.
La table Typ_utilisateur contient tous les types d'utilisateurs existants dans Apogée.

NB : Il faudra, en accord avec la DEVU, restreindre la liste des types d'utilisateur aux plus utilisés et leur attribuer un nom générique de type secrétariat pédagogique, responsable de scolarité etc….(utilisation du champ commentaire ?)

b) centre de gestion

Les centres de gestion sont présents dans la table centre_gestion. Pour Paris Descartes, on se limite à UP5.

c) centre d'inscription pédagogique

Les CIP sont présents dans la table centre_ins_ped.

d) centre d'incompatibilité

Ils sont dans la table centre_incomp

e) centre de stage
Ils sont dans la table centre_ges_stg

f) Composante
Elles sont dans la table composante.

g) centre de traitement

Ils sont dans la table centre_traitement.

2) Fonctionnalité de l'application

a) Une gestion des responsables de scolarité dans les composantes doit être créée dans Apogée. Un responsable peut octroyer des droits pour sa composante, le CIP et les CTN rattachés à son UFR.

b) Un responsable de scolarité pourra consulter l'ensemble des habilitations pour son UFR. Il aura la possibilité de demander des modifications, des créations ou des suppressions. La création nécessite la vérification que cet utilisateur n'existe pas déjà avec des droits pour une autre composante. Elle impose également la recherche de cet individu dans le référentiel établissement (GRHUM).

c) Après validation, un mail sera envoyé à la Devu pour les informer que de nouvelles demandes d'habilitations sont demandées.

d) Le directeur de la Devu pourra alors se connecter à l'application pour donner ou non son accord. En cas d'avis favorable, les habilitations seront soit créées soit modifiées dans la base Oracle Apogée. La création engendrera une création de compte dans « l'annuaire FormsAccess ». La suppression consiste à mettre à 'N' le témoin en service dans la table Utilisateur, à supprimer l'agent du forum apogée et à supprimer ces connexions FormsAcces.

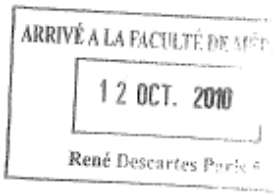
e) Pendant toute la durée de l'opération, le responsable d'Ufr pourra consulter l'évolution de ces demandes.

f) tous les utilisateurs doivent être présents dans le forum Apogée du référentiel

Liste des tables Apogée affectées :

Utilisateur
Uti_cmp
Uti_collecter_ctn
Uti_cgs

5. FORMULAIRE DE DEMANDE D'HABILITATION



UNIVERSITE PARIS DESCARTES
DEVU

FORMULAIRE D'INSCRIPTION D'UN NOUVEL UTILISATEUR DANS APOGEE

Pour toute demande de connexion à Apogée, ce formulaire doit être rempli.
Service / UFR : scolarité
Nom : O
Prénom : L
Téléphone : 01.44.41.22.13
Mail : @parisdescartes.fr
Si vous n'avez pas d'adresse électronique, demandez à votre correspondant technique (informaticien de votre Composante) la procédure pour en obtenir une.

Composante : (la composante dont vous dépendez)
☐ UFR **Sciences Biomédicales** ☐ UFR **Droit Economie Gestion** ☒ UFR **Médecine**
☐ UFR **STAPS** ☐ **UFR Psychologie** ☐ UFR **Pharmacie**
☐ **IUT** ☐ **UFR mathématiques informatique** ☐ **UFR Sciences Humaines et Sociales**
☐ **UFR Chirurgie Dentaire**

☐ **Services Centraux** ☐ **Autre – préciser :** .....

Votre rôle dans Apogée : (cocher une ou plusieurs cases)
☐ Inscriptions Administratives ☐ Référentiel
☐ Inscription Pédagogiques ☒ Consultation du Dossier de l'Etudiant
☒ Saisie/Calcul des Notes et Résultats ☐ Consultation tous les domaines d'Apogée
☒ Gestion des Thèses/HDR/DR ☐ Structure des Enseignements
☐ Gestion des Stages ☐ Modalités de Collecte des Connaissances

Les niveaux de formation sur lesquels vous intervenez : (cocher une ou plusieurs cases)
☐ L1 ☐ L2 ☐ L3 ☐ MASTER ☒ ETUDES DOCTORALES
☐ **Autre – préciser :** .....
- Indiquer éventuellement les filières ou spécialités :

Informations complémentaires :
Votre fonction :
☐ Responsable de scolarité ☒ Gestionnaire de scolarité ☐ Secrétaire pédagogique
☐ Enseignant ☐ Service central – lequel : .....
☐ Autre – préciser : .....
- Votre responsable administratif ou enseignant : M
- Votre correspondant pour Apogée :
Avez-vous suivi des formations Apogée ?
Si oui, lesquelles ? NON

Il est indispensable de suivre des formations pour pouvoir utiliser Apogée dans de bonnes conditions. Il peut S'agir de formation en interne auprès de vos collègues utilisant déjà Apogée, ou des sessions de formations Organisées par le Service de la Formation des Personnels.

Le : 11 octobre 2010
Visa du responsable APOGEE de la composante ou de votre responsable direct : Avis favorable,

Figure 94 Ancien Formulaire de demande d'habilitation
Source Université Paris Descartes

6. TYPES D'ANOMALIES

Tableau IV Anomalies et cas normaux des identités
Source : MC COLLAS

Indicateurs	user Oracle	UTI PRP5	UTI APOTEST	uid Oiddas	Oiddas lien PRP5	Oiddas lien Apotest	uid Annuaire	statut Annuaire	33173	Action corrective
Etats possibles	O	N	N	O	O	O	O	staff	O	
	N	ES	ES	N	N	N	N	faculty	N	
		HS	HS					student		
								affiliate		
cas normaux										
n'existe pas à P5	N	N/A	N/A	N	N/A	N/A	N	N/A	N/A	
n'utilise pas Apogée	N	N/A	N/A	N	N/A	N/A	O	?	N	
n'utilise plus Apogée	O	HS	HS	?	N	N	O	?	N	
n'est plus à P5	O	HS	HS	N	N/A	N/A	O	affiliate	N	
utilise Apogée seul	O	ES	HS	O	O	N	O	staff/faculty	O	
utilise Apotest seul	O	HS	ES	O	N	O	O	staff/faculty	O	
utilise Apogée et Apotest	O	ES	ES	O	O	O	O	staff/faculty	O	
impossibilités										
user oracle	**N**	**O**	**O**	?	?	?	?	?	?	
User Oiddas	?	?	?	**N**	**O**	**O**	?	?	?	
annuaire	?	?	?	?	?	?	**N**	**O**	**O**	
oiddas/annuaire	?	?	?	**O**	?	?	**N**	?	?	
Anomalies										
Apogée et affiliate	O	**ES**	?	?	?	?	O	**affiliate**	?	A supprimer dans Apogée
Apotest et affiliate	O	?	**ES**	?	?	?	O	**affiliate**	?	A supprimer dans Apotest
Apogée et pas	O	**ES**	?	O	O	?	O	staff/faculty	**N**	A inscrire dans le forum

Indicateurs	user Oracle	UTI PRP5	UTI APOTEST	uid Oiddas	Oiddas lien PRP5	Oiddas lien Apotest	uid Annuaire	statut Annuaire	33173	Action corrective
forum										
Apotest et pas forum	O	?	**ES**	O	?	O	O	staff/faculty	**N**	A inscrire dans le forum
pas Apogée et forum	O	**HS**	**HS**	O	?	?	O	staff/faculty	**O**	A désinscrire dans le forum
Apogée et pas Oiddas (orphelins)	O	**ES**	?	O	**N**	**N**	O	staff/faculty	?	Lien rompu à recréer manuellement si l'application ne l'a pas conservé
Apotest et pas Oiddas (orphelins)	O	?	**ES**	O	**N**	**N**	O	staff/faculty	?	Lien rompu à recréer manuellement si l'application ne l'a pas conservé
Apogée et pas Oiddas (orphelins)	O	**ES**	?	**N**	N/A	N/A	O	staff/faculty	?	User et lien à créer dans Oiddas ou supprimer dans Apogée
Apotest et pas Oiddas (orphelins)	O	?	**ES**	**N**	N/A	N/A	O	staff/faculty	?	User et lien à créer dans Oiddas ou supprimer dans Apotest
Apogée et Oiddas sortie	O	**HS**	?	O	**O**	**?**	O	staff/faculty	?	A désinscrire dans Oiddas
Apotest et Oiddas sortie	O	?	**HS**	O	**?**	**O**	O	staff/faculty	?	A désinscrire dans Oiddas
pas Apogée et Oiddas	**N**	N/A	N/A	O	**O**	**?**	O	staff/faculty	?	A désinscrire dans Oiddas
pas Apogée et Oiddas	O	**N**		O	**O**	**?**	O	staff/faculty	?	A désinscrire dans Oiddas
pas Apotest et Oiddas	O		**N**	O	**?**	**O**	O	staff/faculty	?	A désinscrire dans Oiddas

Bibliographie

BIBLIOGRAPHIE

Boisvert, Mathieu, et Sylvie Trudel. 2011. *Choisir l'agilité- Du développement logiciel à la gouvernance*. Paris : Dunod, 320 p. ISBN : 2100558501.

Cogoluègnes, Arnaud, Thierry Templier, Julien Dubois, et Jean-Philippe Retaillé. 2009. *Spring par la Pratique Spring 2.5 et 3.0*. 2e Revue et augmentée. Paris : Eyrolles, 686 p. ISBN : 221212421X.

Company, Sonatype. 2008. *Maven: The Definitive Guide*. 1re éd. Sébastopol : O'Reilly Media, Inc, USA, 470 p.(Précis et concis). ISBN : 0596517335.

Edlich, Stefan. 2002. *Ant précis & concis*. Sébastopol : O'Reilly, 110 p.(Précis et concis). ISBN : 2841771598.

Esup Portail, Pascal Aubry, et Raymond Bourges. 2007. *Formation à esup-commons, ESUP-PORTAIL* [En ligne]. Disponible sur : < (lien rompu) > (Consulté le 8 juin 2010).

Fischer, Frantz. 2011. [Thèse]. *IdMe : un socle pour la gestion d'identités en entreprise* [En ligne]. Mémoire d'Ingénieur en Informatique. Paris : CNAM Paris - Conservatoire National des Arts et Métiers, 224 p. Disponible sur : < http://dumas.ccsd.cnrs.fr/dumas-00711781 > (Consulté le 17 juillet 2012).

Freeman, Eric, Elisabeth Freeman, Kathy Sierra, et Bert Bates. 2005. *Design patterns*. Sébastopol : O'Reilly Editions; 2010 Digit Book pour l'édition française, 639 p.(Tête la première). ISBN : 2841773507.

Gardarin, Georges. 2003. *Bases de données*. 5e éd. Paris : Eyrolles, 787 p.(Best of). ISBN : 2212112815.

Lafosse, Jérôme. 2009. *Java EE - Guide de développement d'applications web en Java*. Saint-Herblain : Editions ENI, 610 p.(Epsilon). ISBN : 2746047152.

Muller, Pierre-Alain, et Nathalie Gaertner. 2003. *Modélisation objet avec UML*. 2e éd. Paris : Eyrolles, 514 p.(Best of). ISBN : 2212113978.

Sennesal, François-Xavier. 2009. *Java Server Faces (JSF) avec Eclipse - Mise en œuvre pour la conception d'applications web*. Saint-Herblain : Editions ENI, 299 p. ISBN : 2746048124.

WEBOGRAPHIE

Amue. 2012a. « Accueil Amue - Amue ». In : *Amue* [En ligne]. Disponible sur : < http://www.amue.fr/ > (Consulté le 9 juillet 2012).

---. 2012b. « APOGEE - Amue ». In : *Amue* [En ligne]. Disponible sur : < http://www.amue.fr/formation-vie-de-letudiant/logiciels/apogee/ > (Consulté le 9 juillet 2012).

---. 2012c. « Base de connaissance - Amue ». In : *Amue* [En ligne]. Disponible sur : < http://www.amue.fr/formation-vie-de-letudiant/logiciels/apogee/assistance-et-conseil/base-de-connaissance/ > (Consulté le 9 juillet 2012).

Apache. 2011a. « Apache CXF -- Using CXF with maven ». In : *Apache* [En ligne]. Disponible sur : < http://cxf.apache.org/docs/using-cxf-with-maven.html > (Consulté le 12 septembre 2011).

---. 2011b. « Apache CXF -- Writing a service with Spring ». In : *Apache* [En ligne]. Disponible sur : < http://cxf.apache.org/docs/writing-a-service-with-spring.html > (Consulté le 12 septembre 2011).

---. 2011c. « Apache OpenJPA User's Guide ». In : *Apache* [En ligne]. Disponible sur : < http://openjpa.apache.org/builds/1.0.2/apache-openjpa-1.0.2/docs/manual/ > (Consulté le 4 mars 2011).

Bretagne, UEB-Université européenne de. 2013. « UEB - Université européenne de Bretagne - uneActu ». Disponible sur : < http://www.ueb.eu/InfosServices/Actualites/actualite/Avec_Amethis__l_UEB_am_liore_le_suivi_des_doctorants.cid16900 > (Consulté le 12 mai 2013).

CRU. 2003. *RECOMMANDATIONS POUR LES ANNUAIRES DE L'ENSEIGNEMENT SUPÉRIEUR : SUPANN- Référence : annuaire SUPANN-V10* [En ligne]. Disponible sur : < www.cru.fr/_media/documentation/supann/supann-v10.pdf > (Consulté le 11 mai 2013).

Developpez.com. 2004a. « Gestion des utilisateurs ». In : *Developpez.com* [En ligne]. Disponible sur : < http://oracle.developpez.com/guide/administration/adminuser/#L1.7 > (Consulté le 14 février 2012).

---. 2004b. « Introduction au framework Spring ». In : *Developpez.com* [En ligne]. Disponible sur : < http://ego.developpez.com/spring/ > (Consulté le 7 février 2011).

---. 2010. « Java Quartz avec Spring ». In : *Developpez.com* [En ligne]. Disponible sur : < http://thebigjim.developpez.com/tutoriels/java/java-quartz-avec-spring/ > (Consulté le 12 mars 2012).

Doudoux, JM. 2010. « Développons en Java - Hibernate ». In : *JM Doudoux* [En ligne]. Disponible sur : < http://www.jmdoudoux.fr/java/dej/chap-hibernate.htm > (Consulté le 4 juin 2010).

---. 2012. « Développons en Java - JNDI (Java Naming and Directory Interface) ». In : *JM Doudoux* [En ligne]. Disponible sur : < http://www.jmdoudoux.fr/java/dej/chap-jndi.htm#jndi-6 > (Consulté le 27 février 2012).

Esup Portail. 2011a. « 1.2 Méthodologie de développement - Projet esup-commons - Esup Portail ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/pages/viewpage.action?pageId=100663437 > (Consulté le 22 octobre 2012).

---. 2011b. « Auto-formation - Projet esup-commons - Esup Portail ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/display/PROJCOMMONS/Auto-formation > (Consulté le 16 novembre 2011).

---. 2010. « Bienvenue sur le site du consortium ESUP-Portail ! - ESUP-Portail - Esup Portail ». In : *Esup-Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/display/ESUP/Bienvenue+sur+le+site+du+consortium+ESUP-Portail+%21 > (Consulté le 17 mai 2010).

---. 2006. « esup-commons - Framework de développement - Projet esup-commons - Esup Portail ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/pages/viewpage.action?pageId=1867791 > (Consulté le 7 juin 2010).

---. 2003a. « ESUP-Portail - Interactions avec le groupe AAS (Authentification, Autorisations et SSO) ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/consortium/espace/SSO_1B/aas/index.html > (Consulté le 28 août 2012).

---. 2004. « ESUP-Portail - Présentation de CAS (Central Authentication Service) ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/consortium/espace/SSO_1B/cas/ > (Consulté le 17 février 2012).

---. 2003b. « ESUP-Portail -Eclipse :: Debug distant ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/consortium/espace/Normes_1C/tech/eclipse/debug.html > (Consulté le 2 juin 2010).

---. 2011c. « Formation V2 - Projet esup-commons - Esup Portail ». In : *Esup Portail* [En ligne]. Disponible sur : < https://www.esup-portail.org/display/PROJCOMMONS/Formation+V2 > (Consulté le 10 novembre 2011).

---. 2011d. « Guide du développeur V2 - Projet esup-commons - Esup Portail ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/pages/viewpage.action?pageId=100663416 > (Consulté le 7 février 2011).

---. 2011e. « Présentation de ESUP-Commons V2 - Projet esup-commons - Esup Portail ». In : *Esup Portail* [En ligne]. Disponible sur : < http://www.esup-portail.org/pages/viewpage.action?pageId=111214616 > (Consulté le 23 mai 2011).

Esup Portail, Pascal Aubry, et Raymond Bourges. 2007. *Formation à esup-commons, ESUP-PORTAIL* [En ligne]. Disponible sur : < (lien rompu) > (Consulté le 8 juin 2010).

JBoss. 2009. « HIBERNATE - Persistance relationnelle en Java standard ». In : *JBoss* [En ligne]. Disponible sur : < http://docs.jboss.org/hibernate/core/3.3/reference/fr/html/index.html > (Consulté le 10 juin 2010).

Maven-guide, et Erwan Alliaume. [s d]. « Maven, le guide Ultime - Chapitre 11. Profils de Build ». In : *Maven:The definitive guide FR* [En ligne]. Disponible sur : < http://maven-guide-fr.erwan-alliaume.com/maven-guide-fr/site/reference/profiles.html#profiles-sect-what > (Consulté le 5 mars 2012).

McCann, Ben. 2009. « Web Services Tutorial with Apache CXF | Ben McCann ». Disponible sur : < http://www.benmccann.com/blog/web-services-tutorial-with-apache-cxf/ > (Consulté le 12 septembre 2011).

Mirtain, Laurent (Inria). 1999. « Tutorial LDAP ». In : *Inria* [En ligne]. Disponible sur : < http://www-sop.inria.fr/members/Laurent.Mirtain/ldap-livre.html > (Consulté le 27 février 2012).

NetApsys. 2009. « Spring-Quartz : Planifier une tâche batch java en 30 minutes - Netapsys Blog ». In : *NetApsys* [En ligne]. Disponible sur : < http://blog.netapsys.fr/index.php/post/2009/06/06/Spring-Quartz-%3A-Planifier-une-tache-batch-java-par-lexemple > (Consulté le 12 mars 2012).

ObjectDB. 2011. « ObjectDB - JPA Manual - Getting Started, Entities, CRUD, JPQL Queries, Settings ». In : *ObjectDB* [En ligne]. Disponible sur : < http://www.objectdb.com/java/jpa > (Consulté le 4 mars 2011).

Objis. 2010. « Tutoriel hibernate N°13 : stratégies de fetching et optimisations requêtes générées ». Disponible sur : < http://www.objis.com/formation-java/tutoriel-hibernate-3-strategie-fetch-fetching-optimisations-requetes-hibernate-3.html > (Consulté le 4 juin 2010).

---. 2011. « Tutoriel web services N°2 : création service web Java 6 ». Disponible sur : < http://www.objis.com/formation-java/tutoriel-webservice-creation-web-service-java-6-jax-ws.html > (Consulté le 12 septembre 2011).

OIDDAS. [s d]. « Glossaire ». In : *OHW (Oracle Help for the Web) 2.0 Bêta* [En ligne]. Disponible sur : < https://oiddas1.epa.gov/oiddas/help/state/content/locale.fr/navId.3/navSetId._/vtTopicFile.hs19%7Cglos~htm/ > (Consulté le 1 mars 2012).

DBA-Ora. 2010. « Encrypter vos données avec DBMS_CRYPTO - Oracle 10G SQL Pour les Nuls ». In : *ORACLE SQL 10G Pour les Nuls & Zéro* [En ligne]. Disponible sur : < http://www.dba-ora.fr/article-encrypter-vos-donnees-avec-dbms_crypto-42771236.html > (Consulté le 1 mars 2012).

Oracle. [s d]. « 2 Introduction to LDAP and Oracle Internet Directory ». In : *Oracle Internet Directory Administrator's Guide* [En ligne]. Disponible sur :

< http://docs.oracle.com/cd/B28196_01/idmanage.1014/b15991/intro.htm#i35990 > (Consulté le 1 mars 2012a).

---. [s d]. « DBMS_CRYPTO ». Disponible sur : < http://docs.oracle.com/cd/B19306_01/appdev.102/b14258/d_crypto.htm#BJFCGDIC > (Consulté le 1 mars 2012b).

---. [s d]. « Managing Oracle Internet Directory ». Disponible sur : < http://docs.oracle.com/cd/A97329_03/core.902/a92171/security.htm > (Consulté le 1 mars 2012c).

---. [s d]. « Oracle Internet Directory ». In : *Oracle® Application Server Release Notes* [En ligne]. Disponible sur : < http://docs.oracle.com/cd/B28196_01/relnotes.1014/b28191/oid.htm > (Consulté le 1 mars 2012d).

DBA-Oracle. 2012a. « Implementing the SHA1 Algorithm ». In : *Burleson Consulting* [En ligne]. Disponible sur : < http://www.dba-oracle.com/forensics/t_forensics_sha1.htm > (Consulté le 1 mars 2012).

---. 2012b. « Oracle Internet Directory (OID) tips ». In : *Burleson Consulting* [En ligne]. Disponible sur : < http://www.dba-oracle.com/application_server/t_app_server_oid.htm > (Consulté le 1 mars 2012).

Paris Descartes. [s d]. « Annuaires ». In : *Paris Descartes* [En ligne]. Disponible sur : < http://app.parisdescartes.fr/cgi-bin/WebObjects/AnnuaireWeb.woa/ > (Consulté le 2 septembre 2010a).

---. [s d]. « ENT Paris Descartes - [ent] ». In : *ENT Paris Descartes* [En ligne]. Disponible sur : < https://ent.univ-paris5.fr/ > (Consulté le 2 septembre 2010b).

---. [s d]. « L'Université Paris Descartes ». In : *Paris Descartes* [En ligne]. Disponible sur : < http://www.parisdescartes.fr/ > (Consulté le 2 septembre 2010c).

---. [s d]. « Université Paris Descartes : CAS – Central Authentication Service ». In : *Paris Descartes* [En ligne]. Disponible sur : < https://servauth.univ-paris5.fr/cas/login?service=http%3A%2F%2Fapp.parisdescartes.fr%2Fapobilitation%2F > (Consulté le 2 septembre 2010d).

Renater (CRU). 2010. « esup-commons - Revision 2271: / ». In : *CRU* [En ligne]. Disponible sur : < https://subversion.cru.fr/esup-commons/ > (Consulté le 7 juin 2010).

---. 2007a. « esup-commons overview ». In : *SourceSup* [En ligne]. Disponible sur : < https://sourcesup.renater.fr/esup-commons/main/index.html > (Consulté le 22 octobre 2012).

---. 2007b. « esup-commons overview: architecture ». In : *SourceSup* [En ligne]. Disponible sur : < https://sourcesup.cru.fr/esup-commons/main/layers.html > (Consulté le 22 octobre 2012).

---. 2007c. « esup-commons overview: configuration files ». In : *SourceSup* [En ligne]. Disponible sur : < https://sourcesup.cru.fr/esup-commons/main/config.html > (Consulté le 22 octobre 2012).

---. 2012. « SourceSup: Bienvenue ». In : *SourceSup* [En ligne]. Disponible sur : < https://sourcesup.renater.fr/ > (Consulté le 22 octobre 2012).

Tech on the Net. [s d]. « SQL ». In : *Tech on the Net* [En ligne]. Disponible sur : < http://www.techonthenet.com/sql/index.php > (Consulté le 6 mars 2012).

Tigris. 2005. « argouml-fr.tigris.org ». In : *Tigris* [En ligne]. Disponible sur : < http://argouml-fr.tigris.org/ > (Consulté le 7 février 2011).

UML.free.fr, et Laurent Piechocki. [s d]. « UML en français ». In : *Cours UML* [En ligne]. Disponible sur : < http://uml.free.fr/index-cours.html > (Consulté le 13 janvier 2011).

Volle, Michel. 2002. « Entropie du système d'information ». Disponible sur : < http://www.volle.com/ulb/021116/textes/entropie.htm > (Consulté le 16 juillet 2012).

---. 2010. « volle.com: L'ingénierie des processus ». In : *Volle.com* [En ligne]. Disponible sur : < http://michelvolle.blogspot.fr/2010/07/lingenierie-des-processus.html > (Consulté le 16 juillet 2012).

Wikipedia contributors. 2012a. *Agence de mutualisation des universités et des établissements* [En ligne]. *Wikipédia*. Disponible sur : < http://fr.wikipedia.org/w/index.php?title=Agence_de_mutualisation_des_universit%C3%A9s_et_des_%C3%A9tablissements&oldid=74424187 > (Consulté le 8 juillet 2012).

---. 2012b. *Apogée (logiciel)* [En ligne]. *Wikipédia*. Disponible sur : < http://fr.wikipedia.org/w/index.php?title=Apog%C3%A9e_(logiciel)&oldid=79062343 > (Consulté le 8 juillet 2012).

---. 2012c. *Cycle de développement (logiciel)* [En ligne]. *Wikipédia*. Disponible sur : < http://fr.wikipedia.org/w/index.php?title=Cycle_de_d%C3%A9veloppement_(logiciel)&oldid=84272917 > (Consulté le 19 novembre 2012).

---. 2012d. *Cycle en V* [En ligne]. *Wikipédia*. Disponible sur : < http://fr.wikipedia.org/w/index.php?title=Cycle_en_V&oldid=83408946 > (Consulté le 19 novembre 2012).

---. 2012e. *Esup-Portail* [En ligne]. *Wikipédia*. Disponible sur : < http://fr.wikipedia.org/w/index.php?title=Esup-Portail&oldid=78091518 > (Consulté le 8 juillet 2012).

---. 2013. *Manifeste agile* [En ligne]. *Wikipédia*. Disponible sur : < http://fr.wikipedia.org/w/index.php?title=Manifeste_agile&oldid=92218013 > (Consulté le 2 mai 2013).

LISTE DES FIGURES

LISTE DES TABLEAUX

RESUME

Le système d'information autour de l'application Apogée (logiciel de gestion de scolarité édité par l'AMUE) s'est complexifié avec le temps et l'accumulation de couches logicielles telles que référentiel de données, portail ENT, annuaire, gestionnaire d'identités. Si bien que le processus d'habilitation des utilisateurs d'Apogée est devenu complexe aussi. L'étude des données montre une augmentation de l'entropie du système.

La solution d'automatisation de ce processus à l'université Paris Descartes est présentée chronologiquement depuis l'expression des besoins, l'étude de l'existant, la conception d'une application web et sa réalisation. La solution mise en œuvre doit s'interfacer avec deux bases de données, deux annuaires Ldap, un système d'authentification et un portail web dans le système d'information existant. Elle réalise la synchronisation nécessaire entre les systèmes de données. Elle utilise les technologies Java/J2EE, Spring, JSF, JPA et JDBC, Oracle, LDAP, le framework esup-commons et le recours au web service.

La mise en production de cette solution a permis un gain de temps passé à habiliter, a rendu au gestionnaire de scolarité le contrôle sur ce processus et a amené la disparition progressive des incohérences dans les données existantes.

Mots clés : Systèmes répartis, architecture logicielle, automatisation de processus, gestion d'identité, habilitation, Apogée

SUMMARY

The information system around Apogee, (Schooling Management software edited by AMUE), has become more complex with time and the accumulation of software layers such as data repository, ENT portal, identities manager. Therefore the process of end users empowerment became very complex too. The data study shows the system entropy. Increasing

The solution of automation process, in Paris Descartes University is presented chronologically from requirement analysis, detail design for web application and implementation. The implemented solution must interface with two databases, two LDAP directories, authentication and a web portal system into the existing information system. It achieves the necessary synchronization between data systems. It uses the Java/J2EE, Spring, JSF, JPA and JDBC technologies, Oracle, LDAP, the framework esup-commons and web service utilities.

The implementation of this solution has enabled a gain of accreditation time, made the school managers control over this process and has led to the gradual decrease of inconsistencies in the existing data.

Keywords: Distributed systems, software architecture, process automation, identity management, empowerment, Apogee