# Méta-annuaire LDAP-NIS-Active Directory

Auteur : Michel LASTES
Co-auteur Bernard MÉRIENNE

Date: 15 octobre 03

LIMSI (Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur)
Bât 508 Université Paris Sud
91503 Orsay CEDEX

michel.lastesATlimsi.fr

**Résumé**
Notre application s'inscrit dans une démarche de centralisation de l'ensemble des informations professionnelles concernant les membres d'un établissement, même si dans un premier temps elle ne met l'accent que sur la gestion des informations relatives aux comptes informatiques et au courrier électronique des utilisateurs.

Notre réalisation est motivée par le désir d'apporter des améliorations à une administration informatique relativement traditionnelle basée sur un annuaire NIS (Network Information Service) pour la gestion des compte Unix et sur un annuaire Active Directory pour la gestion des comptes Windows. Afin d'apporter plus de souplesse quant à la nature des informations susceptibles d'être enregistrées, nous utilisons un annuaire complémentaire de type LDAP (Lightweight Directory Access Protocol). Nous recopions dans cet annuaire, les données du NIS concernant les utilisateurs, cependant l'ancien système NIS est maintenu par souci de compatibilité avec les machines les plus anciennes.

Afin de faciliter la gestion et de maintenir la cohérence des informations enregistrées dans chacun des annuaires, nous avons conçu une application méta-annuaire LDAP-NIS-Active Directory.

Il s'agit d'une application Web, basée sur des servlets Java et sur JNDI (Java Naming Directory Interface). Elle permet à un utilisateur de rechercher des informations, de s'authentifier et de modifier certains de ses propres attributs (shell et répertoire de connexion sous Unix) et son mot de passe sur chacun des annuaires où il possède un compte. Nous avons créé dans l'annuaire, un groupe d'administrateurs auxquels l'application méta-annuaire propose des opérations complémentaires :

-Création d'un nouveau compte dans LDAP et NIS
-Enregistrement ou modification des informations relatives à la gestion des comptes
-Création d'un compte sur les serveurs Active Directory
-Modification du mot de passe associé à un compte dans tous les annuaires où il est référencé.

La mise en place d'une structure ouverte permet d'envisager de multiples extensions, tant dans le domaine de la gestion des profils des utilisateurs que dans les autres domaines de la logistique d'un établissement.

## Mots clés
méta-annuaire, LDAP, NIS, Active Directory, Servlets, JNDI

# 1. Organisation initiale 'pré méta-annuaire'

Initialement les informations concernant les membres de l'établissement sont enregistrées dans plusieurs bases indépendantes. Un annuaire NIS permet de centraliser les caractéristiques des comptes UNIX et du courrier électronique. En complément un fichier texte, permet de consigner les informations relatives à la gestion des comptes (nom du responsable, nom de la machine de travail, date d'arrivée, etc.) qui ne peuvent s'intégrer facilement dans le NIS. Par ailleurs les comptes Windows sont gérés par des serveurs Active Directory.

Le nouvel arrivant est systématiquement enregistré dans le NIS car cet annuaire gère l'accès au courrier électronique. Pour l'utilisation d'une autre ressource et notamment celle de machines sous Windows, le nouvel utilisateur doit se présenter auprès du responsable de cette ressource pour se faire enregistrer.

# 2. Inconvénients de l'organisation initiale :

- Dissémination des informations dans des bases qui ne peuvent inter opérer.
- Manque de cohérence des informations enregistrées dans les différentes bases.
- Difficulté d'accéder aux informations et/ou aux bases (par exemple si le responsable d'une ressource est absent).
- Difficulté pour un utilisateur de savoir en cas de problème à qui s'adresser et où trouver l'information sur une ressource particulière.

**En ce qui concerne plus précisément le domaine informatique :**

Les outils de gestion d'annuaire pour Unix et Windows résident évidemment sur des plates-formes différentes, ce qui rend difficile pour les administrateurs de chacun des mondes, d'accéder ou de mettre à jour les informations de l'autre.

Inconvénients plus spécifiques à chaque unité de mémorisation :

**NIS**

- La nature des informations enregistrées est prédéfinie et limitée.
- Ces informations ne sont disponibles que sur les machines Unix, ou pour les utilisateurs qui ont ouvert une session sur ces machines à condition qu'ils sachent utiliser les commandes de consultation.

**Fichier texte**

- Manque de cohérence dans le format du fichier, pas de normalisation des champs.
- La nature des informations enregistrées évolue au cours du temps.

**Pour les deux entités:**

- La mise à jour est effectuée par édition manuelle avec les coûts en temps significatifs et les risques d'erreurs, parfois très pénalisant pour le NIS car cela peut entraîner le blocage des systèmes. Pour résoudre ce problème, nous avons mis en place au laboratoire le système RCS (Revision Control System). Ce système permet, en cas d'incohérence d'un fichier source du NIS, par exemple après une erreur lors de l'édition, de restaurer une version antérieure. Toutefois, avant que le problème ne soit identifié et le retour en arrière effectué, le service est perturbé.
- Pour modifier leur mot de passe dans le NIS, les utilisateurs doivent appeler un utilitaire qui n'est disponible que sur les machines Unix. Pourtant les utilisateurs non Unix utilisent ce mot de passe car les serveurs de courrier sont des machines Unix et les outils qui permettent de rapatrier les nouveaux messages requièrent ce mot de passe.

**Active Directory**

- Les utilisateurs n'ont pas systématiquement de compte dans Active Directory. Les utilisateurs sous Unix sont parfois appelés à utiliser les ordinateurs en libre service sous Windows. ils ne savent alors pas sous quel compte ouvrir une session. Ils essayaient souvent leur nom d'utilisateur et leur mot de passe Unix, lequel se voit rejeté. Ils effectuent généralement l'opération plusieurs fois avant de faire appel à un administrateur Windows.
- Le nom d'utilisateur n'est pas obligatoirement le même que dans le NIS, ce qui entraîne quelquefois des incohérences. Un même utilisateur peut être enregistré avec un identifiant différent dans les mondes Unix et Windows.
- Active Directory n'est pas très souple pour la mémorisation d'informations autres que celles qui sont utiles à Windows.
- Et cependant le service fourni par Active Directory est indispensable, car à partir de la version Windows 2000 les machines sont configurées pour utiliser cet annuaire.

## 3. Les contraintes

Nous avons étudié une solution qui permette de remédier aux inconvénients présentés ci-dessus, et qui garantisse les points suivants :

- Continuité du service grâce au maintien des systèmes d'annuaire en place.
- Souplesse dans la nature des informations enregistrées
- Enregistrement d'informations qui permettent de suivre le parcours des utilisateurs (changement de statut, de machine).
- Cohérence des informations enregistrées.
- Gestion et accès aux informations concernant une ressource par au moins un responsable et un suppléant.
- Mise en place d'une solution, qui permette aux personnes autorisées, de consulter, de mettre à jour, de manière simple et sécurisée les informations mémorisées.
- Extensibilité facile.

## 4. Caractéristiques générales de l'application

La réalisation de l'application a nécessité :

- L'intégration d'un nouvel annuaire, permettant à terme de remplacer le NIS et contenant en conséquence les informations définies sur cet ancien annuaire, mais aussi toutes les informations nécessaires à la gestion des comptes informatiques absentes du NIS et enfin toutes les informations concernant les autres ressources attribuées à chaque membre de l'établissement (numéros des clés qui lui sont fournies, ….).
- La conservation du NIS car les machines Unix les plus anciennes ne savent utiliser que cet annuaire.
- La conservation des serveurs Active Directory et des outils Windows servant à leur mise à jour, car nous savons que Microsoft « devance » régulièrement les standards. Si tel était le cas, et que cela conduisait à une incompatibilité avec l'application que nous avons mise en place, il faudrait que le service d'annuaire Windows continue d'être rendu.
- La mise en place d'une application Web **méta-annuaire**, accessible depuis toutes les machines de l'intranet, qui permette :
  - de gérer en parallèle les informations dans chacun des annuaires,
  - de mémoriser, dans chacun des annuaires, les informations relatives à la gestion des comptes informatiques,
  - d'offrir une architecture qui permette à terme, de mémoriser et de gérer tout type d'information sur un utilisateur. Et ainsi de centraliser les informations initialement réparties dans plusieurs bases,
  - d'assurer une cohérence des données enregistrées,
  - de gérer les modifications des informations par des instances concurrentes de l'application,
  - d'assurer la confidentialité des données enregistrées,
  - de protéger les flux sensibles (transmission de mot de passe sur le réseau),
  - d'authentifier la personne qui se connecte,
  - de reconnaître la catégorie de l'utilisateur lorsqu'il s'authentifie, et de lui proposer les opérations, spécifiques à sa catégorie, sur les annuaires en place ; et, pour le moins, de permettre à chacun de changer son mot de passe de courrier électronique,
  - en fonction de l'opération choisie, de fournir des formulaires de saisie et de réaliser les traitements nécessaires sur chacun des annuaires, de manière automatique, que ce soit pour visualiser, modifier les informations ou pour enregistrer un nouvel arrivant, de manière simple et intuitive,
  - d'enregistrer, une trace de toute opération de modification, de manière à assurer le suivi, à posteriori, en cas de problème ou d'erreur,
  - d'enregistrer, parmi les attributs de l'utilisateur, la trace des changements sur les informations qui le concernent de manière à pouvoir en effectuer l'historique.

## 5. Les éléments sur lesquels s'appuie notre solution

Afin de nous affranchir des faiblesses du NIS et de pouvoir enregistrer les informations sur les utilisateurs que nous désirons, nous avons décidé d'intégrer un serveur LDAP. Les machines Unix savent utiliser ce type de serveur en remplacement du NIS, ce que ne permettrait pas un SGBD. De plus, LDAP est en plein essor et nous aurons acquis une expérience qui, nous l'espérons, nous permettra de nous rattacher à un annuaire global LDAP s'il est mis en place.
Pour assurer la cohérence des informations mémorisées dans chacun des annuaires et afin de simplifier la mise à jour de ces informations, nous avons développé une application méta-annuaire LDAP-NIS-Active Directory.

Nous avons décidé que le client du méta-annuaire, pour les utilisateurs ou les administrateurs, serait un navigateur (**Netscape Navigator** ou **Internet Explorer**) car aujourd'hui la quasi-totalité des ordinateurs en est dotée.
Afin de ne pas charger les machines clientes, l'application est mise en œuvre côté serveur.

L'application est basée sur des servlets Java et utilise JNDI (Java Naming Directory Interface) pour l'accès aux annuaires. JNDI permet d'accéder en lecture/écriture sur la plupart des annuaires mais il ne permet que la consultation sur un serveur NIS. Pour modifier les attributs enregistrés dans cet annuaire, nous utilisons JNI (Java native Interface). JNI nous permet également d'utiliser des utilitaires du système hôte de l'application, par exemple la journalisation, grâce à laquelle nous enregistrons une trace des opérations de modification.

**LDAP :**
Rappelons que LDAP est issu de DAP, protocole d'accès à l'annuaire X500, annuaire complexe et basé sur des protocoles réseaux peu utilisés. LDAP a été développé afin de permettre l'accès à X500 par des ordinateurs de faible puissance connectés à l'Internet. Par la suite, ce qui n'était initialement qu'un protocole d'accès est devenu un annuaire à part entière, spécifié par la version 3 du protocole, la version actuelle.
LDAP V3 précise la manière de mémoriser les informations et les moyens pour les gérer et y accéder de manière sécurisée. Le succès de LDAP se mesure au grand nombre d'implémentations de serveurs ainsi qu'à l'utilisation de LDAP dans de multiples domaines, aussi bien par les systèmes d'exploitation que par les applications.
LDAP est devenu le standard des annuaires électroniques qui prennent de plus en plus d'importance dans les systèmes d'information des entreprises (http://www.cru.fr/ldap).

**SERVLETS JAVA :**
Les **servlets** sont des objets Java utilisables par toute application client/serveur.
Cependant elles sont généralement utilisées dans les applications Web pour fournir une solution de rechange, basée sur Java, aux CGI. Aussi servlet est-il communément utilisé pour désigner une servlet HTTP.
Afin de pouvoir utiliser les servlets, un moteur de servlets doit être associé au serveur Web. Dans certaines distributions, le moteur de servlets fait partie intégrante du serveur Web.
Lorsqu'un navigateur appelle une servlet il spécifie dans l'URL les caractéristiques de la classe de la servlet désirée. Un objet de cette classe n'est instancié par le moteur que lors du premier appel de l'URL. A ce moment, des initialisations (ouverture connexion base de données, etc.) peuvent être effectuées. Ensuite, et ceci est également le cas pour tous les appels suivant de l'URL, en fonction de la méthode spécifiée dans la requête HTTP, le moteur appelle la méthode correspondante développée par le programmeur dans la servlet (doGet() pour les méthodes GET, doPost() pour POST, etc.).
Le moteur associe deux paramètres à cet appel, un objet de la classe HttpServletRequest et un autre de la classe HttpServletResponse. Le premier contient les paramètres provenant de la requête HTTP, notamment les arguments transmis par l'intermédiaire d'un formulaire, et des informations complémentaires ajoutées par le moteur de servlet. Le second permet principalement à la servlet de renvoyer ses résultats. La servlet effectue les traitements pour lesquels elle est programmée et prépare une page Web qui est renvoyée au navigateur.

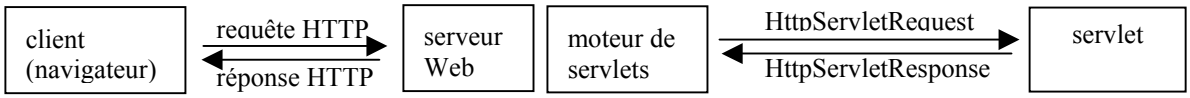


*Figure : Architecture d'une application Web basée sur les servlets.*

**Mécanisme de session :**
Une servlet peut demander au moteur de servlet d'ouvrir une session pour elle. Les caractéristiques du navigateur sont associées à la session et la servlet peut demander au moteur d'enregistrer des attributs dans cette session, attributs que le moteur inclut dans les objets httpServletRequest lors des appels suivants provenant de ce navigateur. Plusieurs sessions peuvent être ouvertes parallèlement avec plusieurs navigateurs, chaque session contenant ses propres attributs.

**Chaînage de servlets :**
L'opération de chaînage permet de spécialiser des servlets, de manière à ce que chacune d'entre elles effectue une tâche spécifique, ce qui facilite la maintenance et la réutilisation du code. Lorsqu'une requête est traitée par une suite de servlets spécialisées, les paramètres d'appel (HttpServletRequest et HttpServletResponse) de la première servlet, éventuellement modifiés par cette dernière, sont transmis à la suivante, et ainsi de suite.

**Avantages des servlets**
Les servlets sont persistantes entre les requêtes sous forme d'objets. Le moteur de servlets ne crée qu'une seule instance de la classe de chaque servlet, ce qui présente les avantages suivants :

- limitation de l'espace mémoire nécessaire au traitement de requêtes multiples sur une même servlet,
- accélération de la réponse au client. Si la servlet est déjà lancée, une requête peut être traitée immédiatement,
- mémorisation d'informations utilisables lors des appels suivants,
- les servlets, objets Java, peuvent utiliser toutes les API existant dans ce langage, pour accéder, par exemple, à des bases de données, à des annuaires ou au système de fichiers du système hôte, etc.
- les servlets bénéficient de la portabilité du langage Java. Elles peuvent être mises en place sur des moteurs de servlets différents installés éventuellement sur des plates-formes différentes.
- enfin, le temps de réponse d'une servlet, dont l'exécution a lieu sur le serveur, et qui renvoie uniquement des pages HTML, sera équivalent quelle que soit la version du navigateur ou les performances de la machine cliente.

**JNDI**
Afin d'accéder aux annuaires du laboratoire, les servlets utilisent JNDI (Java Naming Directory Interface). JNDI est l'interface Java conçue par Sun pour les systèmes d'annuaire.
**Architecture de JNDI**
JNDI est constitué de l'API (Application Programming Interface) et de la SPI (Service Provider Interface).
JNDI fournit aux applications Java une interface unique (JNDI API) pour accéder aux annuaires.
L'interaction réelle avec les annuaires est réalisée par des composants logiciels (des classes), les fournisseurs de service (SP Service Provider) de la SPI.

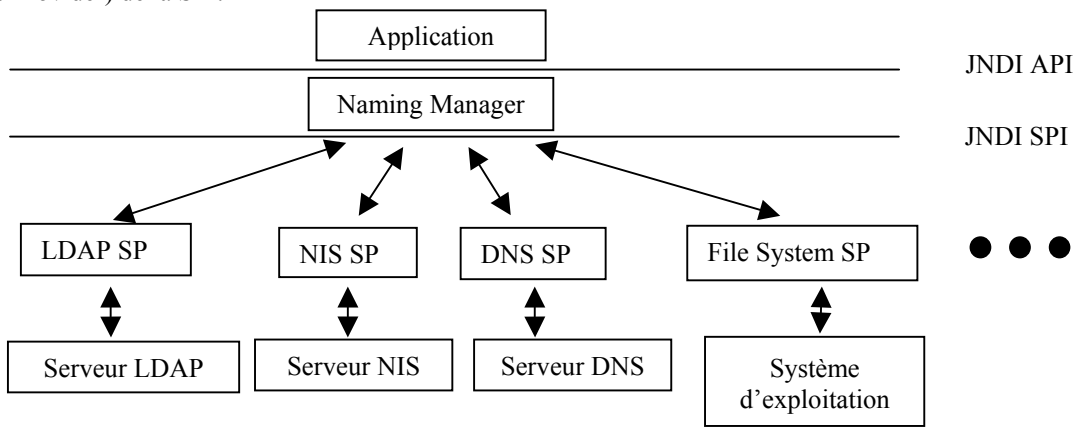


*schéma : architecture de JNDI*
*L'application demande au Naming Manager de connecter le service provider qu'elle désire, elle peut ensuite utiliser les méthodes fournies par l'API JNDI, que le service provider implémente.*

**Contexte JNDI**
Le préalable à l'utilisation des méthodes de JNDI est l'ouverture d'un contexte (ceci est réalisé par l'instanciation d'un objet d'une classe de JNDI implémentant l'interface Context). Afin d'ouvrir un contexte, l'application doit renseigner certaines variables, définissant dans un premier temps les caractéristiques du Service Provider désiré, que le Naming Manager connecte à l'application. Il convient également de spécifier les caractéristiques de l'annuaire à contacter, par exemple le nom de la machine serveur et le port sur lequel est proposé le service et, en fonction du type d'annuaire, les informations pertinentes pour l'utiliser (informations d'authentification, etc.)
Un contexte offre les opérations de recherche, de modification des entrées ou de leur nom, d'inventaire du contenu d'une entrée. Ces opérations ne sont évidemment réalisables que si l'annuaire cible le permet. On ne peut, par exemple, modifier directement le contenu d'une table du NIS. Cette opération sous-entend la modification d'un fichier source et l'appel de l'utilitaire makedbm, sur le serveur NIS, ce que ne peut faire un Service Provider.

**JAVA IO**
Si le service provider LDAP permet la lecture et l'écriture sur un annuaire qui supporte ce protocole, le service provider NIS autorise uniquement la consultation. Les maps NIS sont créées à partir de fichiers texte. Pour modifier les informations du NIS, il faut dans un premier temps modifier ces fichiers. Puisque les servlets s'exécutent sur la machine serveur NIS, les fichiers source du NIS peuvent être modifiés par l'intermédiaire du package d'entrée-sortie : java.io.

**JNI**
Afin de construire les tables NIS à partir des fichiers texte contenant les informations, il faut appeler l'utilitaire makedbm. makedbm est un exécutable écrit en langage C fourni avec l'implémentation d'un serveur NIS. Il est possible d'appeler de tels programmes à partir de Java en utilisant JNI (Java Native Interface).
JNI est fourni en standard avec le JDK (Java Development Kit) de Sun. JNI permet à du code Java qui s'exécute à l'intérieur d'une machine virtuelle Java d'interagir avec des applications et des librairies écrites dans d'autres langages, tels C et C++.
L'application utilise également JNI pour appeller la fonction Unix crypt(), qui transforme une chaîne de caractères en chaîne cryptée, utilisable aussi bien dans le NIS que dans LDAP.
JNI permet également l'appel de la méthode système **log** fournie par Linux, qui permet d'enregistrer des informations dans le fichier de journalisation du système.

# 6. Mise en œuvre de la solution

## 6.1 Choix des outils nécessaires à la mise en œuvre de la solution

**Le système d 'exploitation hôte : Debian GNU-Linux**
Nous avons choisi Linux car c'est un système fiable, relativement compatible avec les UNIX propriétaires et facile à maintenir. Ce système d'exploitation est utilisé depuis des années dans le secteur enseignement-recherche et il remplace de plus en plus les systèmes UNIX propriétaires.

**L'annuaire LDAP OpenLDAP**
Il existe de nombreux serveurs LDAP notamment **Iplanet Directory Server**, **OpenLdap**, **IBM Secure Way,** etc. Nous avons choisi **Openldap** pour plusieurs raisons :

- beaucoup d'organisations ont mis en place OpenLDAP, produit Open Source dérivé du serveur développé à l'Université du Michigan, et en sont satisfaites,
- ce produit est fiable, implémente la version 3 de LDAP, est maintenu et continue d'être amélioré.

Bien sûr, les tests parus dans la presse informatique montrent que le serveur développé par **Iplanet** est plus performant. Mais il est payant.

**Serveur Web : Apache**
Comme nous l'avons indiqué précédemment, le méta-annuaire est une application Web qui s'exécute côté serveur.
Le serveur Web que nous avons choisi est le serveur **Apache** car c'est un logiciel libre, très performant, qui est maintenu et continue d'être amélioré. Son succès est tel que plus de 50% des serveurs Web installés dans le monde sont des serveurs Apache.

**Moteur de servlets : Tomcat**
Pour réaliser l'application Web, nous avions le choix entre scripts CGI et servlets Java.
En raison des avantages que nous avons présentés précédemment, mais aussi parce que nous désirions approfondir la connaissance du langage, nous avons choisi les servlets Java. Seule la vérification des formulaires proposés aux administrateurs est réalisée sur le navigateur de la machine cliente, par l'intermédiaire de petits programmes Javascript.
Le moteur de Servlets que nous avons choisi est **Tomcat**, développé par l'équipe d'Apache, il présente globalement les mêmes avantages que ce serveur Web et il est facilement inter opérable avec celui-ci.

**Sécurisation des transferts d'information**
Les annuaires NIS et OpenLDAP sont installés sur la plate-forme qui héberge l'application méta-annuaire. Les flux entre cette dernière et ces annuaires sont ainsi sécurisés. Mais certaines opérations nécessitent le transit du mot de passe sur le réseau (authentification, changement de mot de passe). La communication entre navigateur et serveur Web se fait via HTTPS (Hyper Text Transfert Protocol Secure), protocole HTTP sécurisé, basé sur TLS (Transport Layer Security) qui assure le chiffrement des données transmises.
Parallèlement, tous les flux avec les serveurs Active Directory Windows 2000 passent par un canal chiffré. L'application utilise les packages Java JSSE (Java Secure Socket Extension), inclus dans le Java 2 SDK (Software Development Kit), Standard Edition, V1.4 pour réaliser ce chiffrement.

## 6.2 Adaptations, paramétrages nécessaires des outils

**NIS**

Nous avons porté le serveur **NIS**, mis en place initialement sur un SUN OS sur un système **Debian GNU-Linux**. Cette distribution fournit un serveur NIS facilement configurable.

Le système **RCS (Revision Control system)**, appelé par l'intermédiaire de JNI, continue d'être utilisé pour enregistrer les versions successives des fichiers source du NIS.

**OpenLDAP**

Nous avons recopié dans l'annuaire LDAP les informations sur les utilisateurs, préalablement enregistrées dans le NIS. Nous avons utilisé pour cette opération les utilitaires fournis par le site www.padl.com. Nous avons également enregistré dans cet annuaire des informations complémentaires, utiles à la gestion des comptes informatiques des utilisateurs, dont une partie se trouvait initialement dans un fichier texte. Nous avons utilisé le langage Perl pour transformer ce fichier dans un format exploitable par le serveur LDAP ( format LDIF LDAP Data Interchange Format ).

**-extension du schéma** :

Afin d'enregistrer des attributs non standard LDAP (i.e. : qui ne font pas partie des schémas standard LDAP), nous avons créé nos propres attributs et classes d'objets. Dans ce but, nous avons demandé et obtenu un OID (Object IDentifier) pour le laboratoire auprès de l'IANA (Internet Assigned Numbers Authority ) et ajouté au schéma LDAP nos propres définitions.

**-organisation des données (DIT Directory Information Tree)**

Les informations enregistrées dans un annuaire LDAP sont organisées dans une structure arborescente appelée DIT. Comme il est préconisé dans les standards (RFC 2247), nous avons utilisé notre nom de domaine DNS (Domain Name Service) comme racine de l'arbre, en utilisant l'attribut dc (domain component), ce qui donne pour notre laboratoire, dc=limsi,dc=fr. En dessous nous avons créé une branche ou sont enregistrées les informations concernant les utilisateurs, le RDN (Relative Distinguished Name) de cette branche est ou=people (ou : organizational unit). On trouve ici une entrée par utilisateur référencé. Cette entrée contient les attributs se rapportant à l'utilisateur et prend pour RDN : uid=username. Ainsi pour un utilisateur dont l'identifiant est michel dans la table passwd du NIS, le DN (Distinguished Name) de l'entrée LDAP qui le représente sera : uid=michel,ou=people,dc=limsi,dc=fr.

**-création de groupes administratifs du méta-annuaire :**

Sous la racine, nous avons également défini une branche nommée ou=metaGroups, ou seront enregistrés les groupes définis pour le méta-annuaire. On ne trouve actuellement ici qu'une seule entrée, de RDN ou=infoAdm. Cette entrée est une liste de personnes à qui sont attribués les droits d'administration des annuaires.
Lors de l'extension du méta-annuaire à des informations autres que celles relatives à l'informatique (gestion des clés de bureau…) de nouveaux groupes seront créés dans cette branche.

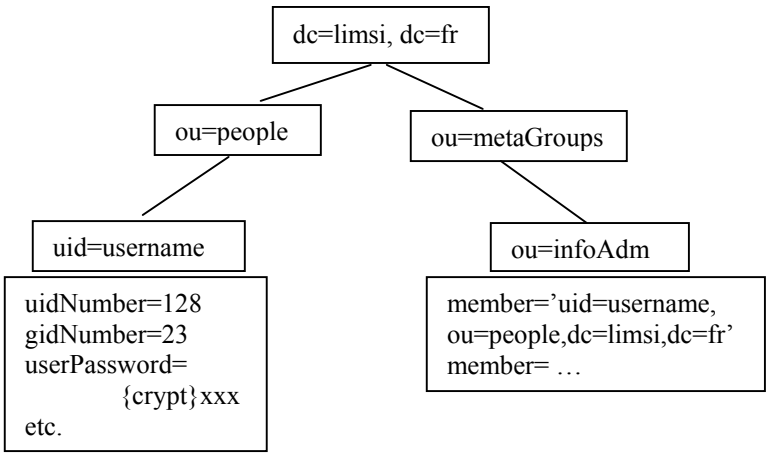


*Schéma : DIT du limsi.*

*Lorsque l'utilisateur s'authentifie, l'application vérifie s'il fait partie d'un groupe administratif du méta-annuaire afin de lui proposer les opérations offertes aux membres de ce groupe.*

**Windows 2000 :**
Nous avons installé le service de certificat (Certificate Service) et le MS Hight Encryption Pack sur les serveurs Windows 2000. Le premier élément permet à l'application méta-annuaire, pour laquelle nous mettons à disposition les certificats des autorités de certification, d'accepter les certificats des serveurs lorsqu'elle ouvre des communications chiffrées avec ces serveurs. Le second élément permet les communications SSL/TLS chiffrées en 128 bits, préalable requis par Active Directory pour donner le droit de modifier un mot de passe via le protocole LDAP.

## 6.3 Définitions des catégories d'utilisateurs :

**Les utilisateurs connectés au méta-annuaire**

**Utilisateur anonyme :**

C'est l'état dans lequel un utilisateur se trouve lorsqu'il appelle l'application par l'intermédiaire de son navigateur. L'application ouvre alors pour cet utilisateur un contexte JNDI avec le serveur NIS et un autre avec le serveur OpenLDAP, enregistrés en tant qu'attributs de session.
L'une des premières opérations proposées à cet utilisateur est l'authentification. Cette opération invite alors l'utilisateur à fournir un username et un mot de passe. L'application transmet ces valeurs au serveur OpenLDAP qui les vérifie. Si elles sont valides, l'utilisateur change de statut et passe dans l'une des deux catégories suivantes.

**Administrateur des ressources informatiques:**

Si le DN correspondant à cet utilisateur (i.e. uid=username, ou=People, dc=limsi, dc=fr) est enregistré en tant que membre de l'entrée ou=infoAdm, ou=metaGroups, dc=limsi, dc=fr et que cet utilisateur possède une entrée sur chacun des serveurs Active Directory et que chacune de ces entrées fait partie du groupe des administrateurs Windows, alors cet utilisateur se voit proposer la totalité des opérations de gestion de compte : création de compte, accès à tous les attributs et en particulier au mot de passe.

**Utilisateur authentifié :**

Dans le cas où l'utilisateur est authentifié mais qu'il n'est pas administrateur ses droits se limitent aux opérations de recherche et de modification de certains de ses attributs.

**Utilisateurs gérés par le méta-annuaire :**

L'application crée d'abord des entrées pour les nouveaux utilisateurs, dans LDAP et NIS puis sur les serveurs Active Directory ensuite. Les comptes peuvent être désactivés en fonction des mouvements de personnel. Ainsi, nous ne supprimons aucune entrée dans nos annuaires. En effet, de nombreux chercheurs effectuent des passages successifs au laboratoire et leur compte doit être réactivé à chacun de leurs séjours. D'autre part, il arrive que les travaux réalisés par des personnels temporaires doivent être exploités bien après leur départ. Il faut donc que les données et les comptes restent accessibles.

Caractéristiques des utilisateurs enregistrés dans les annuaires du LIMSI

**Utilisateur valide :**

Un utilisateur valide possède :

- un compte sur le NIS
  - avec une entrée valide dans la table passwd,
  - éventuellement une entrée définissant le serveur de courrier électronique dans la table aliases.
- un compte sur le serveur LDAP sous le DN :uid=username, ou=people, dc=limsi, dc=fr
- éventuellement un compte sur un ou plusieurs des serveurs Active Directory.

**Utilisateur désactivé:**

Pour désactiver un utilisateur, sous OpenLDAP et NIS on ajoute un tiret à la fin du username, ce qui force le renvoi avec la mention « user unknown » de tout courrier adressé à username d'une part et, d'autre part, attire l'attention lors du 'ls –l' d'un répertoire, sur le fait que les fichiers sont ceux d'un utilisateur ancien.
Ensuite certaines opérations sont effectuées dans chacun des annuaires:

- sur le NIS :
  - dans la table passwd: on ajoute le caractère étoile « * » en début du champ mot de passe, l'authentification est alors impossible.
  - dans la table aliases: on efface, si elle existe, la définition du serveur de courrier de cet utilisateur.
- Dans LDAP
  - on fait également précéder le mot de passe par une étoile,
  - l'attribut mail est effacé, mais un attribut sauvegardeMailServeur reste renseigné, pour utilisation en cas de réactivation du compte.
  - la désactivation est signalée dans l'attribut historique de l'entrée.

- Sur chacun des serveurs Active Directory où l'utilisateur possède un compte,
    - la valeur de userAccountControl est placée à 514, et
    - l'attribut userprincipalName (de la forme 'username@nom_domaine_AD') prend un tiret en fin du username.

**Nouvel utilisateur sous OpenLDAP et NIS :**

A la création du compte et jusqu'à sa mise en service effective :

- L'attribut mot de passe est une étoile, ce qui empêche toute authentification,
- uidNumber et gidNumber sont des valeurs valides, alors que le shell et le directory de connexion prennent des valeurs inexistantes (/noshell, /nodir),
- Le nom et le prénom et les attributs qui en découlent (nom et prénom sans accent, gecos) sont renseignés.
- Les autres valeurs ne sont pas renseignées.

**Nouvel utilisateur sous Active Directory :**

Tous les attributs nécessaires au compte, récupérés sur le serveur LDAP, sont renseignés. L'application utilise les mécanismes Java de création de chaînes de caractères aléatoires et c'est une valeur de ce type qui est fournie comme mot de passe du compte, interdisant ainsi l'authentification sous Windows, en attendant la mise à jour effective.

## 6.4 Fonctionnement de l'application méta-annuaire :

Les opérations du méta-annuaire sont réalisées par une succession de servlets chaînées entre elles.
L'utilisateur appelle l'URL du méta-annuaire dans son navigateur. La servlet méta-annuaire renvoie à cet utilisateur la page des commandes proposées aux utilisateurs non authentifiés (anonymes). Cette page offre la possibilité d'effectuer une recherche ou de s'authentifier. Dans ce dernier cas, la servlet responsable de l'authentification fait demander par le navigateur un username et un mot de passe. La servlet transmet ces valeurs à OpenLDAP pour vérification. Si ces valeurs sont valides, l'utilisateur passe dans le statut authentifié ou administrateur et c'est alors la servlet qui prépare la page contenant la liste des opérations disponibles pour ce type d'utilisateur qui est appelée. Lorsque l'utilisateur fait une sélection dans la page, la première servlet qui intervient dans la réalisation de l'opération choisie est appelée. Celle-ci prépare généralement un formulaire, qui renvoie, lorsque la validation de l'opération est demandée, vers une autre servlet (ou une succession de servlets) qui réalise les traitements et rappelle la servlet qui prépare la page des opérations disponibles.
Le préalable au fonctionnement du méta-annuaire est que tous les annuaires gérés par l'application soient accessibles. Dans ce but, la servlet méta-annuaire ouvre un contexte JNDI non authentifié avec chacun d'eux.
Lorsqu'elle reçoit un nom d'utilisateur et un mot de passe, la servlet qui a la charge de l'authentification engage l'ouverture d'un contexte authentifié avec le serveur OpenLDAP. Si cette opération réussit, elle tente d'ouvrir un contexte authentifié avec chacun des serveurs Active Directory. Les contextes ouverts sont enregistrés sous la forme d'attributs de session de l'utilisateur et sont utilisés pour effectuer les opérations sur les annuaires respectifs.

## 6.5 Architecture de l'application méta-annuaire

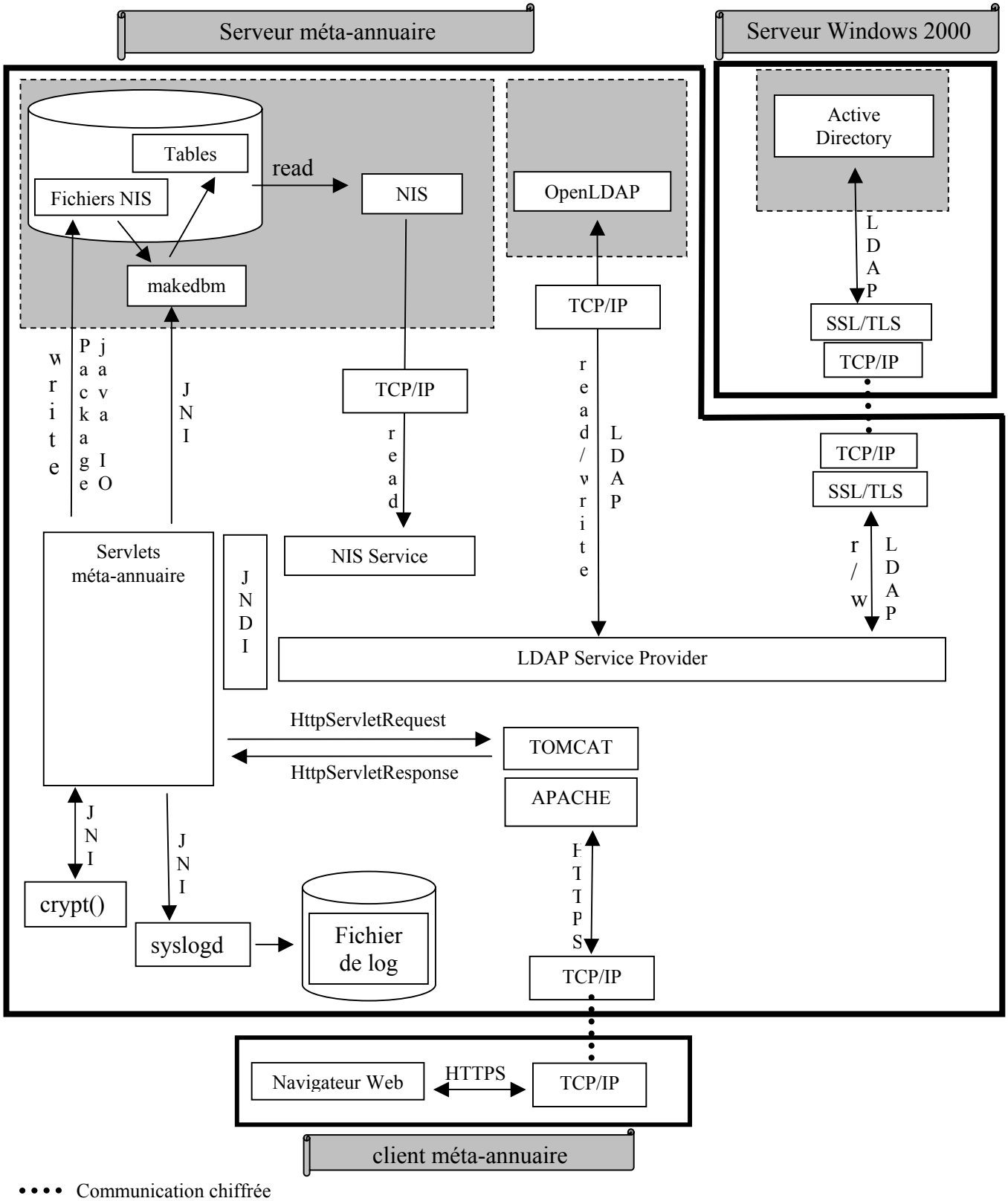


*Figure : Architecture générale du méta-annuaire*

## 6.6 Opérations proposées par le méta-annuaire :

Le préalable à la réalisation d'opérations est que la totalité des annuaires gérés soit accessible. Cette condition est vérifiée à chaque nouvel appel du méta-annuaire, par la tentative d'ouverture d'un contexte non authentifié avec chacun d'eux.
Le méta-annuaire propose ensuite un certain nombre d'opérations en fonction du type d'utilisateur connecté.

**Opérations proposées à un utilisateur anonyme :**

-Recherche d'une personne en fonction du nom ou d'une partie du nom :

L'application fournit une IHM par laquelle l'utilisateur peut saisir le nom ou une partie du nom de la personne recherchée. A partir de ces critères, l'application lance la recherche sur le serveur OpenLDAP et affiche le résultat sous forme d'un tableau comportant autant de lignes que d'individus répondant aux critères fournis. Chaque ligne contient les attributs nom, prénom et adresse de courrier électronique de l'entrée. L'adresse de courrier est un hyperlien qui permet d'envoyer un courrier électronique à l'utilisateur référencé.
Il est à noter que la consultation se déroule en deux phases : une phase de recherche dans les annuaires, suivie d'une phase de préparation de l'affichage.

-Authentification :

L'application utilise les mécanismes HTTP pour demander par l'intermédiaire du navigateur, les noms et mot de passe de l'utilisateur. Puis ces valeurs sont transmises au serveur OpenLDAP pour vérification. Si ces valeurs sont valides, elles sont successivement transmises à chacun des serveurs Active Directory par le biais d'une demande d'ouverture de contexte avec ces annuaires. L'utilisateur passe dans la catégorie authentifié ou administrateur.

**Opérations proposées à un utilisateur authentifié :**

-Opération de recherche :

Elle se déroule de la même manière que pour un utilisateur anonyme.

-Modification de certains attributs OpenLDAP et NIS :

Un formulaire permet la saisie des nouvelles valeurs. L'application met à jour les attributs correspondants dans l'entrée OpenLDAP et la table NIS passwd et lance la reconstruction de cette table. Les caractéristiques de la modification sont enregistrées dans l'entrée historique et dans le fichier de journalisation du système hôte. En cas de problème lors d'une modification, un mécanisme de retour en arrière est lancé afin de rétablir la cohérence initiale des informations dans tous les annuaires et un message indique la cause du problème à l'utilisateur.

-Modification du mot de passe :

Un formulaire invite l'utilisateur à saisir avec confirmation, son nouveau mot de passe. Si les deux valeurs saisies sont identiques, l'application appelle la fonction Unix crypt() et récupère la chaîne chiffrée résultante. Puis elle remplace par cette valeur l'ancien mot de passe dans le fichier source de la table passwd. La table est reconstruite et l'application transmet ensuite au serveur OpenLDAP la demande de remplacement de la valeur de l'attribut userPassword par :"{crypt}chaîne chiffrée". Puis le nouveau mot de passe en clair est transmis par le canal chiffré spécifique, à chacun des serveurs Active Directory sur lequel l'utilisateur possède un compte.

**Opérations proposées à un administrateur :**

-Enregistrement d'un nouvel utilisateur dans NIS et OpenLDAP :

Le formulaire proposé lors de la demande d'un tel enregistrement ne comprend que peu de champs. Le nom et le prénom de la personne, le nom d'utilisateur choisi, le nom d'utilisateur du responsable (un nouvel arrivant est accompagné de son responsable lors de son inscription), le département et le groupe dont il va faire partie. Est-il permanent ou non, et éventuellement la date de départ prévue et un champ de remarques.
A partir de ces valeurs, l'application vérifie d'abord que le nom d'utilisateur proposé n'existe ni dans OpenLDAP, ni dans le NIS, ni dans aucun des serveurs Active Directory. Nous rappelons que l'administrateur possède obligatoirement un contexte sur chacun des annuaires ce qui permet d'effectuer ces opérations.
Si le nom d'utilisateur proposé convient, l'application détermine, à partir de la table passwd, un numéro d'identifiant d'utilisateur (uidNumber) disponible. Puis elle ajoute une ligne dans le fichier source de la table passwd.
La table NIS est ensuite reconstruite.

Une entrée de DN : uid=username, ou=people, dc=limsi, dc=fr est créée dans OpenLDAP où tous les attributs sont enregistrés. On enregistre également la date de création du compte.

-Validation d'un nouvel utilisateur :

Il s'agit simplement d'enregistrer le premier mot de passe de l'utilisateur. Ce qui est réalisé en appelant l'opération de modification du mot de passe.

-Recherche d'une personne à partir d'une partie du nom :

Les opérations de saisie des critères et de recherche dans OpenLDAP se passent de la même manière que précédemment. Cependant, l'affichage du résultat diffère. D'une part, les résultats représentant des entrées désactivées sont affichés. D'autre part, dans chaque ligne du résultat, un champ supplémentaire est ajouté. Ce champ est le username de l'utilisateur correspondant aux critères de recherche et ce champ est un hyperlien. La sélection de ce lien envoie une page affichant les attributs de l'utilisateur dans le NIS et dans OpenLDAP. Au bas de cette page se trouvent des boutons qui permettent l'appel d'opérations supplémentaires détaillées ci-dessous :

-Edition des attributs OpenLDAP et/ou NIS de l'utilisateur :

Un formulaire d'édition des attributs de l'utilisateur est proposé à l'administrateur. Dans le cas d'attributs redondants dans les deux annuaires, ce sont les valeurs contenues dans le NIS qui sont proposées en tant que valeurs initiales communes. Lors de la demande d'enregistrement des modifications, les valeurs initiales et celles provenant du formulaire sont comparées. Si elles diffèrent, la modification est enregistrée, et une trace de la modification est ajoutée dans l'attribut historique de l'entrée. Lorsque la comparaison est terminée, les pages NIS sont reconstruites si nécessaire et l'opération est journalisée sur le système hôte.

-Modification du mot de passe de l'utilisateur :

Une fenêtre de saisie de mot de passe avec confirmation est envoyée au navigateur de l'administrateur qui laisse alors l'utilisateur saisir la nouvelle valeur. Cette opération se déroule de la même manière que lorsqu'elle est réalisée en simple mode 'utilisateur authentifié'.

-Opérations sur les attributs enregistrés sur les serveurs Active Directory :

Pour chacun des serveurs Active Directory, la page des commandes propose soit d'afficher les attributs de cet utilisateur, soit, si ce dernier ne possède pas d'entrée, la création d'un compte sur le serveur.

-Création d'un compte sur un serveur Active Directory :

Lorsque cette opération est sélectionnée, l 'application récupère dans OpenLDAP les attributs nécessaires et crée le compte avec ces attributs. Le mot de passe de ce compte est une chaîne de caractères aléatoire fournie par Java, interdisant ainsi la connexion Windows jusqu'à la prochaine opération de modification du mot de passe.

-Désactivation d'un compte :

Nous réalisons les opérations suivantes.

- Dans NIS et OpenLDAP :
  - le caractère '-' est ajouté à la fin du username
  - le caractère '*' est ajouté en début du mot de passe.
  - la ligne définissant le serveur de mail est retirée de la table aliases. Cependant l'attribut privé sauvegardeMailServeur reste renseigné dans l'entrée OpenLDAP de cet utilisateur.
- Sur chacun des serveurs Active Directory ou l'utilisateur possède un compte :
  - le caractère '-' est ajouté à la fin du username dans l'attribut userPrincipalName
  - la valeur 514 est enregistrée dans l'attribut userAccessControl.

-Réactivation d'un compte :

- Dans NIS et OpenLDAP,
  - les caractères '-' et '*' ajoutés lors de la désactivation sont retirés.
  - Dans le cas ou l'attribut sauvegardeMailServeur est renseigné dans OpenLDAP, la ligne définissant le serveur de mail dans aliases est recréée.

- Sur les serveurs Active Directory :
  - le '-' est supprimé dans l'attribut userPrincipalName,
  - la valeur 512 est enregistrée dans l'attribut userAccessControl.

### 6.7 Fonctionnement des opérations de modification :

**Enregistrement des valeurs initiales des attributs sur chacun des annuaires :**

Lorsqu'un utilisateur authentifié visualise la page des commandes à sa disposition ou lorsqu'un administrateur affiche le contenu des entrées d'un utilisateur dans les annuaires, l'application place dans des variables de session, la valeur des attributs de l'utilisateur dans chacun des annuaires. Lors de l'enregistrement de nouvelles valeurs d'attributs, le processus qui réalise l'enregistrement obtient un *monitor*, fait une nouvelle recherche sur les annuaires et compare les valeurs trouvées aux valeurs mémorisées. Si les valeurs ont changé, un message est affiché dans la page du navigateur du demandeur, l'informant du problème et l'invitant à réitérer sa demande. Sinon, la suite de l'enregistrement peut avoir lieu.

**Processus de retour à l'état initial :**

Lorsqu'une opération de modification d'attribut sur un annuaire pose problème (par exemple lorsqu'une exception est levée), les modifications effectuées précédemment sont annulées.
Ceci n'est malheureusement pas possible dans le cas des mots de passe Windows. Si l'application connaît la valeur initiale du mot de passe chiffré dans le NIS et LDAP, elle n'a pas la possibilité de récupérer, à notre connaissance, cette valeur sur les serveurs Active Directory. Aussi, en cas de problème, un message invite à vérifier la cohérence des mots de passe dans tous les annuaires.

**Enregistrement de l'historique des opérations de modification :**

La trace des opérations (modification d'attribut, création de compte Active Directory, etc.) qui concernent un utilisateur est enregistrée dans l'attribut : « historique » de l'entrée OpenLDAP représentant cet utilisateur.

**Journalisation des opérations :**

Lorsque des opérations de modification ont lieu, elles sont consignées en utilisant la journalisation du système hôte du méta-annuaire. Ceci permet de suivre à posteriori le déroulement des opérations et de corriger les éventuelles erreurs.

## 7. Extensions envisageables

Le méta-annuaire ne sait gérer pour l'instant que des informations en rapport avec le profil informatique des utilisateurs. Mais la structure du DIT OpenLDAP permet de créer aisément de nouvelles familles d'attributs et de nouveaux groupes administratifs pour en assurer la gestion. Ceci nous permet d'envisager d'étendre l'usage du méta-annuaire à d'autres services de la logistique du laboratoire : secrétariat pour la gestion du personnel, responsable des clefs et autres ressources diverses.

## 8. Conclusion

Nous avons proposé une architecture qui permette de centraliser la gestion des informations professionnelles concernant les membres d'un établissement. Nous l'avons conçue dans le but obtenir un plus, autant en manière de sécurité qu'en manière d'ergonomie. Nous savons l'appliquer à la gestion du profil informatique des utilisateurs. Il nous reste à convaincre nos collègues des autres services de l'intérêt d'utiliser le méta-annuaire comme base de données de référence, et d'y intégrer leurs propres informations.

Par ailleurs, nous nous employons actuellement à développer une interface complémentaire de type API (Application Programming Interface) destinée à permettre au méta-annuaire d'inter opérer avec les autres applications informatiques qui peuvent en avoir besoin.

Enfin nous espérons avoir donné un nouvel exemple de la possibilité de réaliser de nouveaux développements significatifs sans qu'il soit besoin de faire appel à autre chose qu'aux ressources du monde de l'informatique libre, qu'il s'agisse des logiciels d'accès libres ou des forums de discussion associés, et que cela nous a même permis d'aboutir à une meilleure utilisation de ressources notoirement propriétaires tel que NIS et Active Directory.

# 9. Références

**JNDI :**
Rosana Lee, http://java.sun.com/products/jndi/tutorial , Novembre 2002.
Forum JNDI : http://forum.java.sun.com/forum.jsp?forum=51
**JNI:**
Beth Stearns, http://java.sun.com/docs/books/tutorial/native1.1
**LDAP et JNDI :**
Sameer Tyagi, http://www.javaworld.com/javaworld/jw-03-2000/jw-0324-ldap.html, Mars 2000.