

Compte Rendu

Mise en place d'une base de données avec Oracle

**Erwan AZZOUG, Arnaud CHESNAY, Ludovic CHEVRAUX, Florian GUISNEL, Corentin JEZEQUEL,
David MORVAN**

2013 - 2014

Ce document est un compte rendu du projet « Mise en place d'une base de données avec Oracle » du semestre 2 du DUT Informatique. Il vous y sera expliqué par quelles étapes nous sommes passés et quels problèmes le groupe L a rencontré lors du développement.

Sommaire

I-	Mise en place d'une base de données.....	2
1)	<i>Les bases</i>	2
2)	<i>Les déclencheurs</i>	3
3)	<i>Les procédures</i>	3
II-	L'interface.....	4
1)	<i>Les recherches préliminaires</i>	4
2)	<i>L'interface</i>	4
3)	<i>Quelques précisions</i>	6
4)	<i>En ce qui concerne l'exportation en PDF</i>	7
III-	Identifications des utilisateurs	8
1)	<i>Problématique</i>	8
2)	<i>Identification CAS (Central Authentication Service)</i>	8
3)	<i>Serveur LDAP (Lightweight Directory Access Protocol)</i>	8
4)	<i>Présentation du système de création des comptes</i>	9
	Note concernant l'installation du script PHP	9
	Bilan	10
	Annexe 1 : Modèle EAP	11
	Annexe 2 : Images de l'interface (1/2)	12
	Annexe 3 : Images de l'interface (2/2)	13
	Annexe 4 : Bibliographie.....	14

Dans notre groupe, nous sommes tous étudiants à l'IUT de Nantes, dans la spécialité Informatique, ce qui est assez évident. A l'issue des deux années de DUT, nous devons faire un choix : choisir des poursuites d'études ou travailler. Jusqu'à aujourd'hui, le traitement des demandes d'études supérieures se faisait à la main par la secrétaire. Cependant, il y a plus de 80% des étudiants qui veulent poursuivre leurs études et nous sommes en général 80 en deuxième année, ce qui fait beaucoup de travail pour la secrétaire. C'est donc non seulement une perte de temps mais aussi une fatigue évitable, avec en plus la possibilité d'erreur des suites de la fatigue. On nous a alors proposé de faire un programme pour faciliter cette étape de la vie étudiante. Le projet n'était pas très compliqué : Nous devions créer une base de données avec toutes les informations nécessaires à la création des demandes d'études (avec oracle), faire une interface permettant aux étudiants de faire leur demande (en Java) et que les demandes soient utilisables par les professeurs et la secrétaire après leur formulation. Ce projet a été réalisé à l'aide de M. Nachouki, professeur à l'IUT de Nantes. Nous vous expliquerons donc dans un premier temps comment nous avons fait la base de données. Puis comment a été réalisée l'interface et comment elle communique avec la base de données. Enfin, nous avons pensé que l'entrée d'informations sur les élèves une par une serait laborieuse, nous avons donc proposé un script en PHP pour faciliter « l'inscription », on vous l'expliquera en dernière partie.

I- Mise en place d'une base de données

1) Les bases

Pour faire une application ayant une utilité sur le long terme, il fallait créer une base de données. Nous concevions l'application de sorte que nous ayons besoin des entités suivantes : des étudiants, des formations, un administrateur (pour tous les travaux d'administrations) et des vœux. Cette base de données devait donc contenir : des informations sur les étudiants, des informations sur les formations disponibles, un administrateur et des vœux. Comme vous pouvez le voir dans l'annexe 1, notre base de données est constituée de quatre tables et correspond aux besoins de énumérés précédemment. Chaque étudiant est différencié de l'autre par son identifiant, de même pour l'administrateur (normalement, ce sera un compte administrateur unique), les formations par un numéro, ce numéro est incrémenté de 1 à chaque création d'une nouvelle formation et les vœux se différencient à partir de l'identifiant de l'étudiant qui a formulé le vœu et du numéro de la formation que concerne le vœu. L'identifiant est donc la clé primaire de la table Etudiant et Administrateur, munie de la clé primaire de la table Formation, qui est numéro, forment la clé primaire de la table Vœu, elles sont toutes deux clés secondaires aussi. Comme vous avez pu le voir dans le modèle EAP, il y a une propriété de formation qui est "alternance", cette propriété n'est pas réellement un booléen, elle n'a pas la valeur true ou false, mais elle ne peut avoir que les deux chaînes de caractères "Oui" et "Non" comme valeur.

2) Les déclencheurs

Les tables étant créées, certaines situations n'ont pas le droit d'avoir lieu, c'est pourquoi nous avons créé des déclencheurs. Le premier est que toutes les informations d'un étudiant doivent être là, nous avons retenu que les informations les plus pertinentes, c'est pourquoi, à chaque création, aucune données ne doit avoir une valeur nulle. De même pour les tables Formation et Administrateur.

3) Les procédures

La façon la plus pratique que nous ayons trouvé pour faire des modifications, ou tout simplement des lectures, dans la base de données, ce sont les procédures. Pour mieux ordonner le nombre assez conséquent de procédures et pour anticiper sur une modification possible de l'application dans le futur, nous avons créé des packages. Dans le premier package, celui pour créer des entités, on retrouve toute les procédures de création : d'étudiant, d'administrateur, de vœu et de formation. Ces procédures contiennent une simple requête "INSERT INTO", cependant, la procédure "creer_formation" prend un paramètre "date" qui est de type varchar2 (varchar2 = string) et non de type date, ceci facilitant la transaction de données depuis l'application, nous avons utilisé la fonction SQL "to_date(varchar2, 'DD-MM-YY')" qui permet de mettre en type date un varchar2. Le deuxième package est celui de la suppression. Lorsque l'on supprime un étudiant ou une formation, il faut réajuster chaque table car un problème est levé : Si un étudiant est supprimé, comment fait-on pour ces vœux ? De même pour une formation. Dans ces procédures, nous avons contré le problème par une suppression préventive dans la table vœu, si l'étudiant a déjà formulé des vœux alors, ils seront tous automatiquement supprimés, de même si la formation a des vœux qui la concerne, ils seront automatiquement supprimés et l'étudiant/la formation avec. Avec la possibilité de supprimer toutes les formations et tous les étudiants en entrant -1 à la place d'un numéro ou d'un identifiant. Le dernier package est celui dit "informatif", il contient la procédure pour lire les vœux des étudiants. La procédure retourne tous les vœux d'un étudiant dont l'identifiant est donné (sachant que seul les 5 premiers seront affichés directement sur l'application). Pour ce faire, nous avons utilisé les curseurs, le résultat de la requête "SELECT" est stockée dans le curseur et est retourné à l'application dans lequel des méthodes permettent de récupérer tous les résultats un par un, ainsi toutes les informations essentiel du, ou des, vœux effectués seront affichés. De plus pour une meilleure lisibilité, nous avons utilisé la fonction "to_char(date, varchar2)", ainsi la date ne s'affichera pas "20/12/15" mais "20/12/2015", ceci n'étant qu'une question d'esthétisme. La seconde procédure présente de ce package est celle pour définir un avis. Dans cette dernière, nous avons juste utilisé la fonction "UPDATE" de SQL.

Avec toutes ces étapes, la base de données est fonctionnelle et est facile d'utilisation pour une application JAVA. Nous allons donc passer à l'interface qui fait l'intermédiaire entre la base de données et les utilisateurs.

II- L'interface

1) Les recherches préliminaires

Avant de commencer la programmation, nous nous sommes renseignés sur les IHM puisque nous n'avions pas encore reçu de cours à ce sujet. Nous avons donc cherché comment celles-ci fonctionnent en Java et avons programmé des petites applications en de nous entrainer, notamment un petit jeu qui utilise des boutons et une application qui utilise JDBC pour permettre de faire des lectures de bases de données.

Pour nous aider nous avons utilisé les sites openclassrooms.com et developpez.com pour apprendre comment interagir avec nos applications, en utilisant des boutons, des menus, etc...

2) L'interface

L'interface de notre application est constituée de plusieurs fenêtres, qui ont toutes été faite en utilisant le layout manager BorderLayout :

a) La fenêtre de connexion

C'est grâce à cette fenêtre que l'on peut se connecter avec JDBC à la base de données. On utilise un seul compte Oracle pour tous les utilisateurs. Elle est constituée notamment d'un champ de texte pour l'identifiant, d'un deuxième pour le mot de passe, un bouton de connexion et un bouton d'inscription.

Au moment où l'utilisateur clique sur le bouton de connexion, l'application lance une requête SQL afin de récupérer dans la base de donnée le mot de passe correspondant à l'identifiant écrit dans le premier champ de texte. Ensuite, le mot de passe récupéré par la requête et celui entré dans le deuxième champ de texte sont comparés. S'ils sont identiques, alors l'utilisateur peut accéder au reste de l'application. Sinon, si le mot de passe est incorrect, ou même si l'identifiant l'est, la connexion est refusée.

De plus, selon s'il s'agit d'un étudiant ou de l'administrateur, l'application n'ouvrira pas la même fenêtre après la connexion. Pour cela, elle «regarde» le premier caractère de l'application. Si c'est un «E», c'est qu'il s'agit d'un étudiant, et si c'est un «A», c'est qu'il s'agit de l'administrateur.

Afin le bouton d'inscription sert lors de la première utilisation de l'application par un étudiant, le détail se situe dans la partie «Inscription».

b) La fenêtre de l'étudiant

La fenêtre de l'étudiant contient un menu, un label indiquant l'identifiant de l'utilisateur actuel, un tableau affichant 5 vœux de l'étudiant, et un bouton qui ouvre une fenêtre pour afficher l'ensemble des vœux de l'étudiant.

Pour afficher les vœux dans le tableau, nous utilisons la procédure SQL `getVoeuxEtu` du package `package_info`. Celle-ci récupère les informations que nous voulons afficher, puis nous ajoutons ces informations dans le tableau. Nous rajoutons à cela un titre pour les colonnes.

Le menu est constitué de plusieurs parties :

- Fichier :

- A propos: Ouvre une fenêtre qui donne des informations sur l'application
- Support: Ouvre une fenêtre qui renseigne l'adresse mail de la personne à contacter en cas de problème avec l'application

- Modifier :

- Ajouter choix : Ouvre une fenêtre qui permet d'ajouter un vœu. On sélectionne pour cela une formation dans la liste déroulante, puis on clique sur le bouton Ajouter. Si la formation a déjà été choisie, on ne peut pas l'ajouter une deuxième fois. Sinon on appelle la procédure SQL `creer_voeux` de `package_creation` pour ajouter le vœu dans la liste. Ensuite, le tableau est actualisé.
- Supprimer choix: Ouvre une fenêtre qui sert à supprimer un vœu. Pour cela, on sélectionne le vœu à supprimer dans la liste déroulante, puis on clique sur le bouton Supprimer. De façon analogue à Ajouter Choix, la procédure `suppr_voeux` de `package_suppression` est appelée, puis le tableau est actualisé.

- Arrêter :

- Quitter : Permet de quitter l'application. Une fenêtre est ouverte pour confirmer que l'on veut fermer l'application.

c) La fenêtre de l'administrateur

Cette fenêtre reprend des éléments de la fenêtre de l'étudiant, mais est aussi plus complète. Nous avons notamment ajouté un champ de texte et un bouton qui servent à afficher dans le tableau les vœux de l'étudiant que l'on a recherché. Le bouton Liste étudiant sert quant à lui à afficher la liste des étudiants, rangés dans l'ordre alphabétique de leur nom, afin d'obtenir notamment leur identifiant.

Dans le menu nous retrouvons les mêmes sous-menus que pour la fenêtre de l'étudiant, mais nous en avons ajouté d'autres:

-Fichier:

-Exporter:

- Pour tous les étudiants : Permet d'exporter dans un fichier .csv l'ensemble des vœux de tous les étudiants. Par défaut, le fichier est nommé «voeux_tous.csv». Le dossier par défaut dans lequel on souhaite l'enregistrer est à modifier (puisque le chemin actuel a été choisi sur un ordinateur personnel).
- Pour un étudiant : Permet d'exporter les vœux de l'étudiant que l'on a recherché dans le champ de texte dans un fichier .csv. Par défaut, le fichier est nommé «voeux_[identifiant de l'étudiant].csv».

- Modifier :

- Ajouter une formation : permet de créer une nouvelle formation. Pour cela, il faut renseigner dans les champs de texte différentes informations (nom de l'école, ville...) puis cliquer sur le bouton Ajouter. La procédure `creer_formation` est ensuite appelée et la formation est alors ajoutée à la base de données.
- Supprimer une formation: permet de supprimer une formation. On choisit la formation que l'on veut supprimer dans la liste déroulante, et on clique sur le bouton Supprimer. La procédure `suppr_formation` est appelée et la formation est enlevée de la base de données.
- Ajouter un étudiant: permet d'ajouter un étudiant à la base de données. De la même manière que pour ajouter une formation, on renseigne les différentes informations sur l'étudiant, puis on clique sur le bouton Ajouter. La procédure `creer_etudiant` est alors appelée pour l'ajouter à la base de données.
- Supprimer un étudiant: permet d'enlever un étudiant de la base de données. De la même manière que pour supprimer une formation, on sélectionne l'étudiant que l'on souhaite enlever de la base de données, puis on clique sur le bouton Supprimer. La procédure `suppr_etudiant` est alors appelée.

- Jury :

- Ajouter un avis : permet de donner un avis sur un vœu fait par l'étudiant recherché. Pour cela, on choisit le vœu dont on veut donner un avis, ainsi que l'avis, puis on clique sur le bouton Valider. La procédure `setAvis` de `package_info` est appelée pour modifier la valeur de l'avis dans la base de données, puis le tableau est actualisé.

3) Quelques précisions

Concernant l'export des vœux en .csv, il faut modifier le chemin par défaut du dossier dans lequel on souhaite enregistrer le fichier. En effet, celui-ci est actuellement «C:/Users/Arnaud/Desktop/Resultat_Projets2/», or il doit être modifié pour être utilisé sur les machines de l'IUT. Ce chemin est à modifier dans les classes `Export_CSV_Tous` et `ExportCSV_Un`.

Les identifiants de connexion à la base de données sont aussi à modifier, ils se trouvent dans la classe `Connexion`.

Un seul problème persistait, comment obtenir les informations des étudiants à leur première connexion. La première solution était de créer un script sql permettant de rentrer tous les étudiants dans la base, mais cette technique montre très rapidement ces limites, c'est pourquoi nous proposons une autre solution que nous allons vous présenter dans la prochaine partie.

4) En ce qui concerne l'exportation en PDF

Nous avons pris l'initiative que l'application puisse exporter toutes les données dans un fichier, en format pdf. Des recherches sur ce sujet ont été entreprises. Mais cette idée n'a finalement pas aboutie, puisque M. Nachouki a préféré que l'on exporte les données en format csv, c'est-à-dire un fichier Excel. Nous allons tout de même développer les recherches qui ont été effectuées.

Tout d'abord, créer un pdf n'est pas une chose des plus courantes en Java. Il faut donc importer un nombre assez conséquent de librairies, la plupart se trouvant dans `com.itextpdf.text` et `com.itextpdf.text.pdf`. Nous aurons aussi besoin de `java.util.Scanner` et de deux librairies d'exception se situant dans `java.io`.

Pour pouvoir créer un document en format pdf, il ne faut pas vouloir instancier un document de ce type-là, mais il faut au préalable instancier un nouveau document, grâce à la classe `Document`. C'est à ce moment-là que l'on doit créer son type. Cela se fait grâce à la classe `PdfWriter` et à sa méthode `getInstance()`, qui comprend deux paramètres : le document que l'on a créé juste avant, et l'endroit où l'on enregistre le fichier. Cet enregistrement s'effectue directement sur le terminal, où l'on récupère le chemin et le nom du fichier avec la méthode `nextline()`, qui appartient à la classe `Scanner`, qui prend en paramètre `System.in` et dont on aura instancié une variable auparavant. Le document doit alors être ouvert avec la méthode `open`.

Nous pouvons désormais écrire dans le fichier. L'ajout de texte dans ce pdf est plutôt simple, puisqu'il faut créer une variable de type `Paragraph`, de mettre en paramètres le texte que l'on veut afficher, et utiliser la méthode `add()` du document pour pouvoir l'ajouter.

Etant donné qu'il aurait fallu présenter les informations de la même façon pour tous les étudiants, les boucles comme la boucle `for` ou `while` doivent être utilisées.

Afin de pouvoir mettre le logo de l'IUT, il faut savoir insérer une image. La classe correspondante est `Image` et l'on utilise la méthode `getInstance()` et y mettre le lien de l'image en paramètre. Il est possible de redimensionner l'image pour qu'elle soit de la taille que l'on souhaite, la classe `Image` a pour cela une méthode `scaleToFit()` qui prend en paramètres la longueur et la largeur. On ajoute, de la même façon que le texte, l'image au document. Cet ajout d'image peut être source de problèmes, c'est pourquoi il faut faire un `try/catch`. Une exception `DocumentException` ou `IOException` est levée si nécessaire.

La génération d'un pdf peut aussi causer des problèmes. Ne serait-ce qu'en énonçant un mauvais emplacement de fichier. Un `try/catch/finally` est donc indispensable. Le `catch` lève une exception de type `Exception`. Et le `finally` comprend permet de fermer le document avec la méthode `close` de `Document`.

III- Identifications des utilisateurs

1) Problématique

L'identification des utilisateurs sur notre application était une problématique assez importante : en effet, chaque année, il faudrait ajouter entre 60 et 80 nouveaux utilisateurs dans la base de données de l'application. Bien que cette tâche puisse être faite à l'aide d'un script de génération des comptes, il y aurait toujours une opération assez importante de communication des identifiants aux étudiants. Le gain de temps acquis grâce à l'utilisation de l'application serait donc réduit.

2) Identification CAS (Central Authentication Service)

Après quelques heures de réflexions nous avons remarqué que tous les web-services de l'université utilisaient le même système d'identification pour connecter un étudiant. Que ce soit sur MaDoc, le webmail ou encore l'intranet, chaque étudiant utilise le même identifiant, avec le même mot de passe. Nous avons donc commencé à étudier ce système avec les éléments dont on disposait. Le code source transmis au navigateur lors de l'accès à une page de connexion était une première piste de recherche. Le sigle CAS revenait assez fréquemment dans le code source, des recherches sur internet nous ont permis d'obtenir quelques informations sur ce système. Bien que très mal documenté, des bibliothèques PHP sont disponibles pour interagir avec le système. Nous avons procédé à l'implantation de ce système sur une simple page web, les tests étaient concluants.

Note : nous avons tenté d'adapter cette bibliothèque pour Java mais ce système étant assez complexe, nous n'avons pas obtenu de résultat positif.

3) Serveur LDAP (Lightweight Directory Access Protocol)

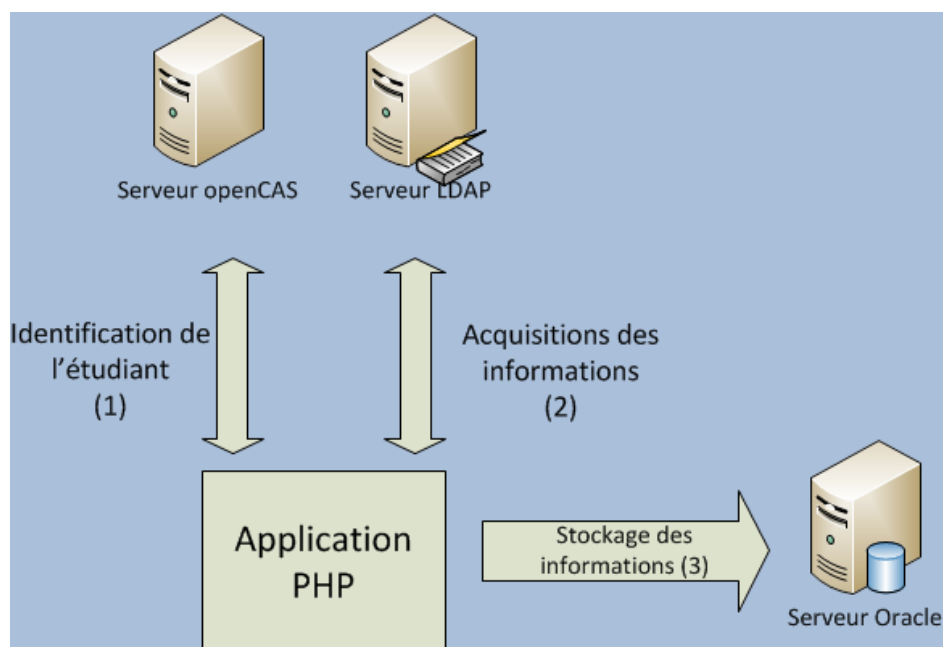
Nous avons vu précédemment que l'on pouvait authentifier un utilisateur à l'aide de son identifiant universitaire (Exxxxxxx). Cependant nous n'avons aucune information concernant l'identité de cet étudiant. Nous avons donc cherché un moyen d'obtenir des informations sur un étudiant, comme son nom, prénom et adresse e-mail. Après investigations, nous avons remarqué la présence d'un serveur LDAP sur le réseau de l'IUT. Des tests ont été réalisés à l'aide d'un client LDAP (JXplorer) afin de se connecter à ce serveur. Nous avons maintenant accès à toutes les données que nous cherchions c'est-à-dire le nom, prénom, e-mail, et identifiant.

4) Présentation du système de création des comptes

Grace au système openCAS et au serveur LDAP de l'IUT, nous avons tous les éléments pour créer les comptes étudiant sur notre application. Nous avons donc imaginé le fonctionnement suivant :

- 1) lors de la toute première utilisation de l'application par un étudiant, il devra se connecter sur une page web à l'aide de ses identifiants universitaire.
- 2) une page récapitulative des informations personnelles de l'étudiant s'affiche (les informations sont récupérées depuis le serveur LDAP), un formulaire de création de mot de passe est proposé. Lorsque l'étudiant valide cette page, les informations sont envoyées dans notre base de données Oracle.
- 3) l'étudiant se connecte à l'application Java avec son identifiant universitaire et le mot de passe qu'il a défini précédemment.

Ci-dessous un schéma récapitulatif :



Note concernant l'installation du script PHP

Pour fonctionner convenablement le script PHP que nous avons développé requiert :

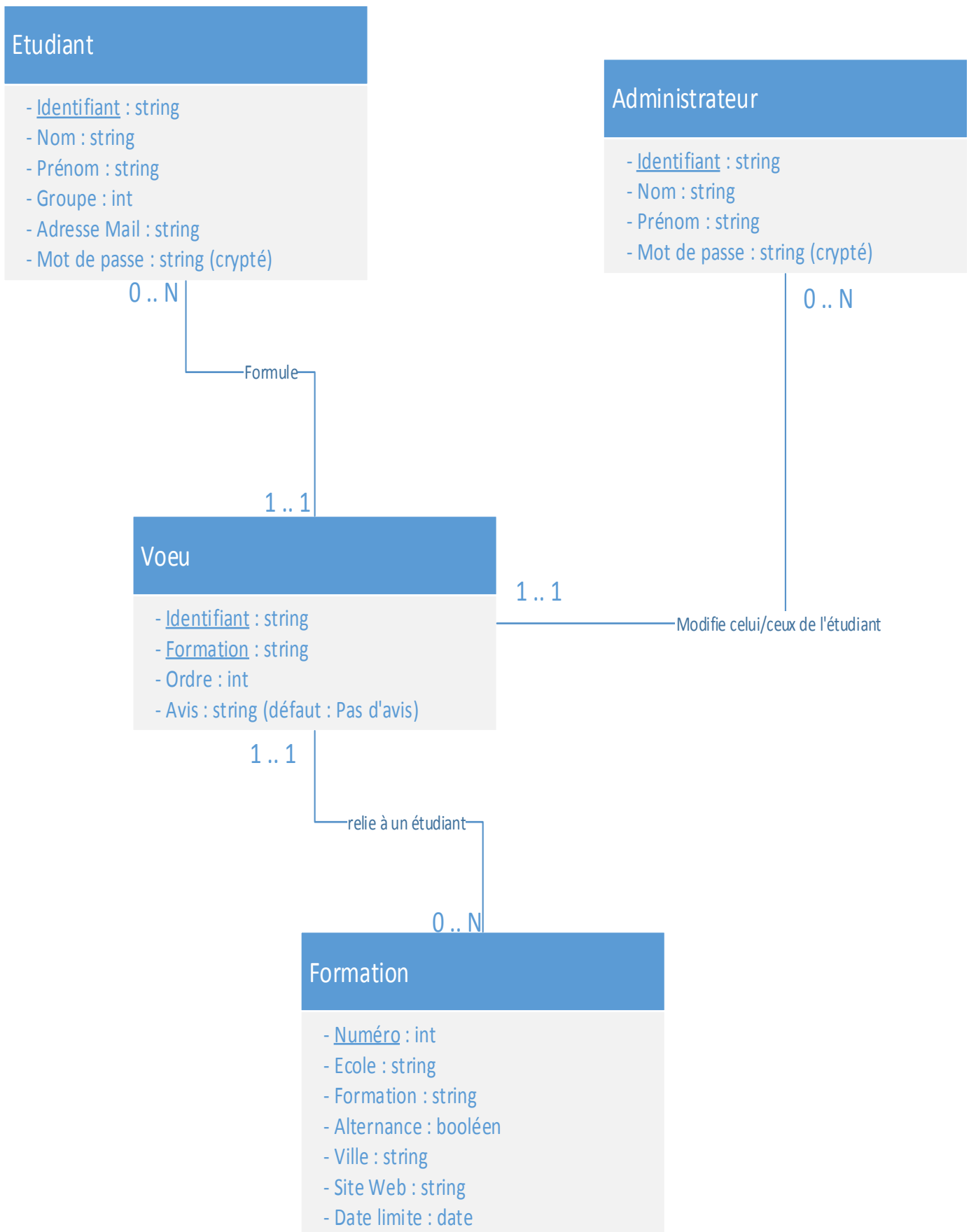
- Un serveur Web équipé de PHP 5.4 ou supérieur.
- L'extension php5 ldap (connexion LDAP)
- L'extension php5 oci (connexion ORACLE)
- Le serveur web doit pouvoir se connecter au serveur LDAP de l'IUT. Effectivement la politique de sécurité de l'IUT empêche toute connexion à ce serveur depuis l'extérieur du réseau.

Bilan

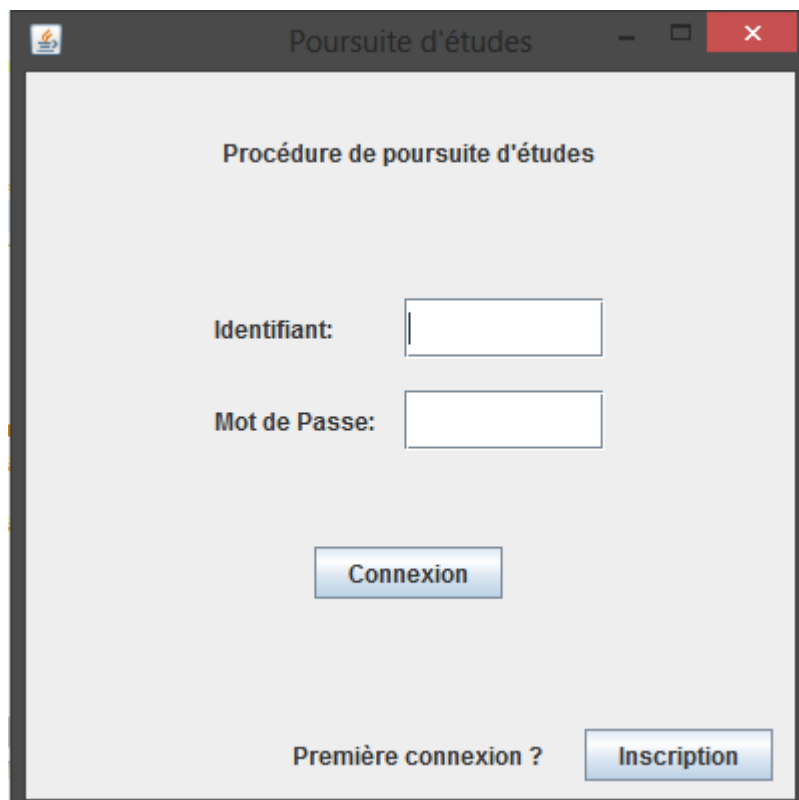
Nous vous avons expliqué toutes les étapes ainsi que toutes les complications que nous avons rencontrées, nous pouvons donc faire le bilan. Notre application est fonctionnelle, cependant, nous pouvons toujours y implémenter plusieurs fonctions pour faciliter l'utilisation de cette dernière. Peu de fonctions ont été abandonnées à cause de leur complexité, nous en avons une seule majeure qui est l'implémentation en JAVA de l'identification CAS, cette dernière a été conçu pour être utilisé avec PHP mais pas en JAVA. Nous voulions aussi à la base exporter en PDF les vœux des étudiants, mais nos recherches (faites par Florian et David) dans le domaine n'ont pu être implémentée. L'esthétisme de l'application reste assez basique mais nous restons fier du résultat. Pour en revenir aux difficultés, la principale difficulté rencontrée dans la base de données (principalement gérée par Erwan et aidé par Corentin) aura été la construction d'une procédure contenant une requête "SELECT", mais une fois le problème étudié en cours, il put être résolu, ainsi que le fait de gérer chaque clé étrangère (surtout lors des suppressions). Dans l'interface (gérée par Arnaud et aidé par Corentin), ce sont les interactions avec la base de données qui auront donné quelques soucis suite aux cas particuliers des curseurs et des procédures. Et pour l'inscription des utilisateurs dans la base de données (gérée par Ludovic), nous restons très déçus de ne pas avoir pu implémenter cette fonction en JAVA, le niveau demandé étant très élevé pour des étudiants en première année. Avoir fait, tous ensemble, une partie théorique (disponible dans le cahier des charges) avant de programmer nous aura permis de gagner beaucoup de temps et tous nous mettre d'accord sur notre vision de l'application.

Nous remercions nos professeurs (et tous les sites de la bibliographie) pour leur aide sans laquelle ce projet n'aurait sûrement pas abouti, ou n'aurait pas pu être aussi complet.

Annexe 1 : Modèle EAP



Annexe 2 : Images de l'interface (1/2)



Poursuite d'études

Procédure de poursuite d'études

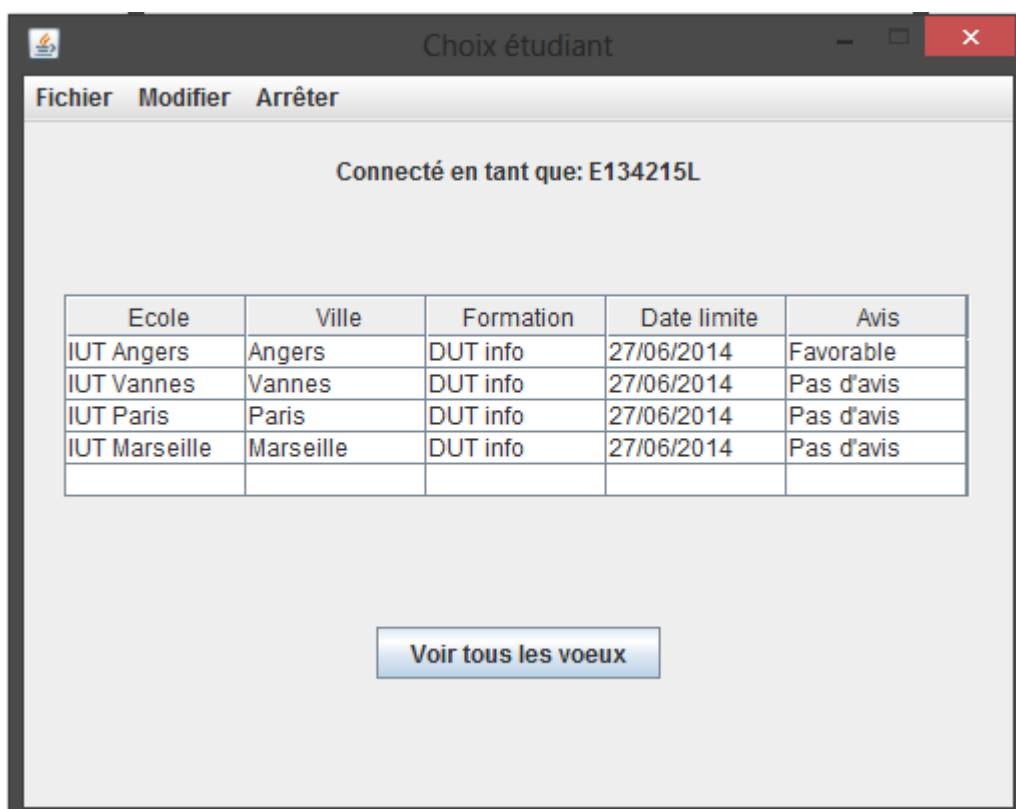
Identifiant:

Mot de Passe:

Connexion

Première connexion ? Inscription

Page de connexion



Choix étudiant

Fichier Modifier Arrêter

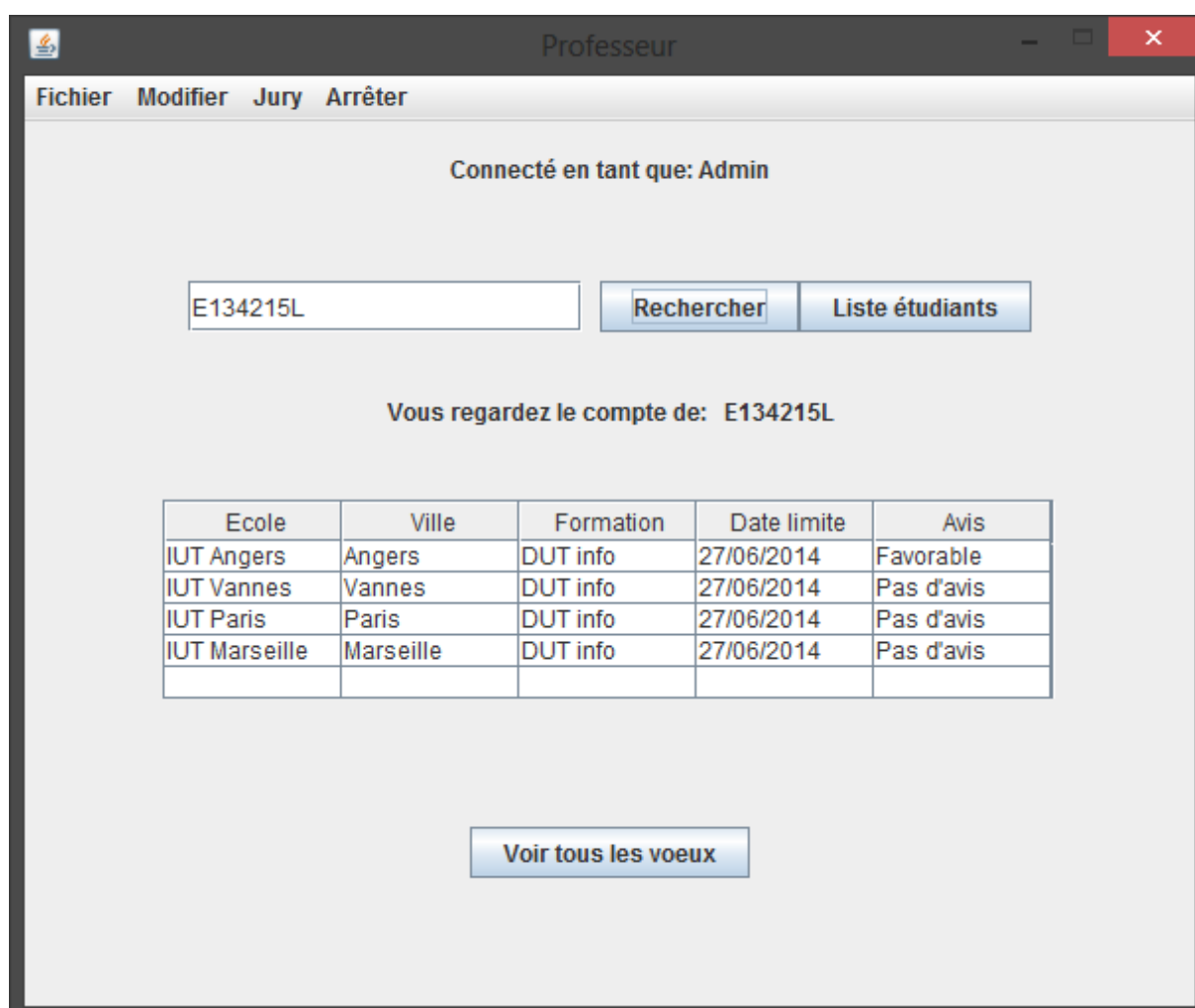
Connecté en tant que: E134215L

Ecole	Ville	Formation	Date limite	Avis
IUT Angers	Angers	DUT info	27/06/2014	Favorable
IUT Vannes	Vannes	DUT info	27/06/2014	Pas d'avis
IUT Paris	Paris	DUT info	27/06/2014	Pas d'avis
IUT Marseille	Marseille	DUT info	27/06/2014	Pas d'avis

Voir tous les vœux

Page sur laquelle arrive l'étudiant quand il se connecte, il peut y voir ces 5 premiers vœux et a accès à toutes les fonctionnalités citées précédemment

Annexe 3 : Images de l'interface (2/2)



Page sur laquelle arrive l'administrateur quand il se connecte et qu'il a choisi un étudiant dont il veut les vœux

Annexe 4 : Bibliographie

Sites web :

fr.openclassrooms.com/
www.docs.oracle.com/
www.zen-workshop.com/
www.commentcamarche.net/
www.stackoverflow.com/
www.sql.sh/
www.oracle.developpez.net/

Logiciels :

SQLDeveloper – *Oracle*
Microsoft Word – *Microsoft*
Notepad++ - *Notepad++ team*
Eclipse – *Eclipse*
WampServer – *Romain Bourdon*
Java – *Oracle*