École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Laboratoire Ligérien de Linguistique
3, Rue des Tanneurs BP 4103
37041 TOURS CEDEX 1
Tél. +33 (0)2 47 36 66 54
http://lettres.univ-tours.fr/

Département Informatique
5e année
2012 - 2013

Projet de fin d'études

Réalisation d'une application pour le Laboratoire Ligérien de Linguistique L.L.L.

Encadrant
Claudine TACQUARD
claudine.tacquard@univ-tours.fr

Étudiant
Valentin DOULCIER
valentin.doulcier@etu.univ-tours.fr

Clients
Jean-Michel FOURNIER
jean-michel.fournier@univ-tours.fr
Marjolaine MARTIN
marjolaine.martin@univ-tours.fr

DI5 2012 - 2013

Version du 29 avril 2013

Table des matières

Table des figures

Liste des tableaux

Introduction

Au cours de notre dernière année au Département Informatique de Polytech' Tours, nous devons réaliser un projet de fin d'études, plus connu sous l'acronyme P.F.E.. Ce projet, de part sa durée, est l'un des plus conséquent que nous devons réaliser au cours de notre formation. Ainsi, il nous permet de gérer en autonomie les grandes phases du cycle de vie d'un projet, de l'analyse à la mise en production, en passant par le développement et la phase de test.

Ce document présente mon rapport de Projet de Fin d'Études, résultant du travail que j'ai effectué tout au long de l'année scolaire 2012 - 2013.

« Réalisation d'une application pour le Laboratoire Ligérien de Linguistique »

Comme son nom l'indique, ce projet est en rapport avec le Laboratoire Ligérien de Linguistique de l'université de Tours, que nous nommerons L.L.L. dans la suite de ce document.

Le Laboratoire Ligérien de Linguistique (L.L.L.) traite du langage et des langues dans tous ses aspects, de la conception d'enquête au traitement automatique des langues et notamment dans une perspective contrastive. Le L.L.L. regroupe 28 enseignants-chercheurs, 24 doctorants. Il est implanté sur 2 sites, Tours et Orléans.

Le L.L.L. de Tours utilise une application sur la prononciation des unités lexicales en anglais contemporain qui regroupe trois dictionnaires existants. Cette application interagit directement avec une base de données qui regroupe les informations syntaxiques, lexicales, morphologiques, ainsi que des données de fréquence, d'usage et de variation. Cette base de données compte aujourd'hui approximativement 700 000 entrées provenant des trois dictionnaires commerciaux.

Différents acteurs interviennent dans ce projet. Je serai en contact avec chacun d'entre eux tout au long de ce projet.

- Le L.L.L. représenté par Jean-Michel FOURNIER et Marjolaine MARTIN (Client et MOA)
- Claudine TACQUARD (Encadrante pédagogique)
- Valentin DOULCIER (MOE)

Présentation du projet

2.1 Contexte de réalisation

Ce projet s'inscrit dans le cadre de la recherche linguistique effectuée par le L.L.L. de Tours. Il concerne une application utilisée par les chercheurs, application qui regroupe toutes les données de trois dictionnaires anglais :

- The Cambridge English Pronouncing Dictionary (EPD)
- The Longman Pronouncing Dictionary (LPD)
- The Macquarie Dictionary (MCQ)

Une application a été développée en 2009 dans son ensemble. Suite à cette première version d'application, deux P.F.E. réalisés respectivement par Elodie BACCONNET en 2010 et Ludovic DUPRAZ en 2011 ont été mis en place. Le but principal de ces P.F.E. était (entre autre) la refonte de la base de données. En effet, celle-ci n'était pas conçue de façon générique et n'était pas modulaire ; il était alors impossible d'espérer une évolution (ajout d'un autre dictionnaire par exemple).

La base de données à donc été refaite. Toutefois, aucune application n'est venue valider l'intégrité de celle-ci. A ce titre, ce rapport détaille les changements qui ont eu lieu.

2.2 Objectifs

Dans ce sens, ce P.F.E. a pour but premier de réaliser une application interagissant avec cette nouvelle base, de vérifier son intégrité et d'implémenter les fonctionnalités désirées par le L.L.L..

La liste des fonctionnalités désirées par le L.L.L. n'est pas exhaustive. En effet, elle dépendra aussi de l'avancement et des difficultés rencontrées au cours du projet. Elle est susceptible d'être complétée à tout moment, son évolution sera prise en compte et le calendrier prévisionnel sera modifié dans la mesure du possible.

2.3 Environnement du projet

Ce projet possède un existant. En effet, il existe un logiciel que les chercheurs du L.L.L. utilisent déjà pour leurs recherches. Toutefois, l'architecture du projet Java ainsi que l'absence de commentaire nous oblige, comme nous le verrons dans ce rapport, à repartir de zéro. Essayer de comprendre l'organisation du code et le réadapter pour la nouvelle base de données serait trop complexe et trop long.

La refonte complète de l'application étant nécessaire, la question de l'architecture est donc essentielle pour construire ce projet sur de bonnes bases.

Pour ce qui est de l'application, le développement se fera sous un environnement Mac OS. Aussi, elle sera développée en Java. A ce titre, il faudra obligatoirement que les utilisateurs disposent d'un JRE (Java

Runtime Environment) afin que le programme puisse s'exécuter.

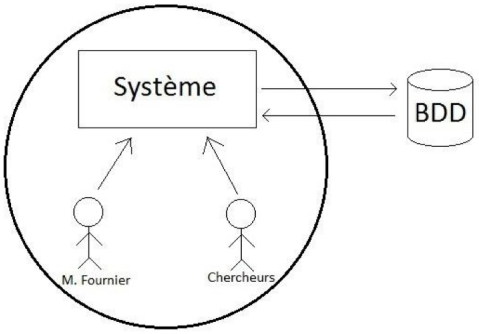


FIGURE 2.1 – Environnement du projet

2.4 Bases méthodologiques

Pour mener à bien ce projet, plusieurs procédures seront respectées, principalement la convention de nommage. Cette convention est présente dans l'archive de documents résultant de mon projet. Elle stipule les règles qui devront être respectées lors du développement de l'application.

Pour ce qui est des langages utilisés, la refonte intégrale de l'existant étant prévue, on ne s'appuiera pas sur ce qui a pu être fait auparavant.

Suite à une analyse des contraintes énoncées par le client, le langage de programmation sera Java. En effet, il permet le développement de logiciels multi-plateformes facilement déployables. La partie base de données ayant été refaite l'année dernière, on conservera le choix du langage ayant été fait (SQL).

FIGURE 2.2 – Bases méthodologiques

Planning

3.1 Planning prévisionnel

La gestion et surtout la planification d'un projet n'est pas évident. Chiffrer un projet et l'estimer en terme de charge (jours/hommes) est loin d'être trivial. Il s'agit d'une compétence qui ne s'acquiert qu'avec l'expérience. Dans un projet complet, il s'agit de prendre en compte plusieurs facteurs, plus ou moins critiques, tels que les risques, les difficultés que l'on pourrait rencontrer lors du développement du projet, etc..

Le planning prévisionnel a été établi en début de projet. Il permet avant tout d'identifier les tâches à réaliser ainsi que leurs durées respectives.

	Nom de la tâche	Durée	Début	Fin
1	Prise en main de l'existant	20 jours	Jeu 27/09/12	Mer 24/10/12
2	Recherche solution technologique	20 jours	Jeu 25/10/12	Mer 21/11/12
3	Rédaction Cahier de Spécifications	35 jours	Lun 12/11/12	Ven 28/12/12
4	Réalisation des documents (Posters, ...)	3 jours	Ven 28/12/12	Mar 01/01/13
5	Correction du Cahier de Spécifications	4 jours	Mer 02/01/13	Lun 07/01/13
6	Mise en place de l'environnement	12 jours	Mar 08/01/13	Mer 23/01/13
7	Développement du squelette de l'IHM	15 jours	Jeu 24/01/13	Mer 13/02/13
8	Module de Recherche	15 jours	Jeu 14/02/13	Mer 06/03/13
9	Module "Edition d'un mot"	15 jours	Jeu 07/03/13	Mer 27/03/13
10	Module "Modification d'un mot"	15 jours	Jeu 28/03/13	Mer 17/04/13
11	Module "Edition des liens valides"	22 jours	Jeu 18/04/13	Ven 17/05/13
12	Serveur - Analyse Technique	20 jours	Lun 04/02/13	Ven 01/03/13
13	Serveur - Mise en place BDD Principale	20 jours	Lun 04/03/13	Ven 29/03/13
14	Tests Application	35 jours	Lun 01/04/13	Ven 17/05/13
15	Rédaction du rapport	90 jours	Lun 14/01/13	Ven 17/05/13

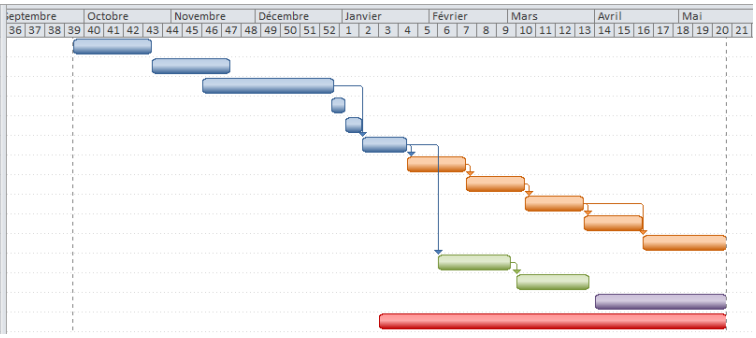


FIGURE 3.1 – Diagramme de Gantt

3.2 Planning réel

Le planning réel ne peut être présenté dans ce rapport. En effet, suite au développement des fonctionnalités initiales du projet, je me suis rendu compte que la base de données ne modélisait pas ce que l'on attendait. Ainsi, l'intégrité des données a été compromise, et il a fallu stopper tout le développement du projet pour s'orienter vers une restructuration de la base. Ce changement est intervenu mi-février, et à partir de ce moment là, plusieurs tâches ont été menées en parallèle, rendant impossible l'élaboration d'un nouveau diagramme de GANTT.

Travail réalisé

4.1 Reprise de l'existant

La première étape de mon P.F.E. fut d'étudier le fonctionnement des cinq logiciels déjà en place. En effet, ce projet s'inscrivant dans la continuité de 2 P.F.E. précédemment réalisés, il a été nécessaire de chercher à comprendre l'imbrication et le sens donné aux 5 logiciels.

Toutefois, la refonte de la base de données m'a empêché de tester ces logiciels, et l'absence de tout commentaire n'a en aucun cas favoriser ma compréhension. Je me suis vite aperçu que le code était incompréhensible, se dispersait dans tous les sens ; un remaniement était alors nécessaire.

4.1.1 Présentation de l'existant

Suite au premier projet réalisé en 2009, l'application fonctionnait comme suit. Le poste de Jean-Michel FOURNIER servait de serveur pour héberger la base de données principale de l'application. Aussi, l'application était déployée sur chacun des postes des chercheurs, doctorants qui le souhaitaient. Déployer l'application ne revenait pas simplement à installer le logiciel, il s'agissait aussi de configurer l'environnement, d'installer la base de données en local sur chacun des postes.

L'application était alors décomposée en 5 projets bien distincts :

- **MajBddClient & MajBddServeur**

  Ces deux programmes permettaient la synchronisation avec la base de données centrale située sur l'ordinateur de Jean-Michel FOURNIER.
  Une fois l'application mise en place sur un nouveau poste de travail, l'utilisateur devait utiliser MajBddClient afin de mettre à jour sa base de données locale. Suite à une modification de cette base (mise à jour d'un mot, apport d'informations complémentaires, ..) l'utilisateur devait alors générer un fichier sql contenant les requêtes à exécuter pour mettre à jour la base de données principale de l'application. Ce fichier était donc envoyé à M. FOURNIER qui sauvegardait ou non les modifications proposées, et ce en utilisant MajBddServeur.

- **Parseur XML**

  Ce programme permettait au développeur de remplir la base de données à l'aide des fichiers XML issus des 3 principaux dictionnaires commerciaux. Seules les tables de références, ne nécessitant aucune création d'association ou de liens (ex : headword, composes, liste_categories, etc.) étaient remplies.

  A ce stade, la base de données était alors partiellement remplie, et les classes du projet principal (Projet Bddictionnairique) étaient partiellement instanciées.

- **Remplissage Tables**

  Ce programme permettait d'effectuer la seconde étape de remplissage de la base de données appelée

l'édition des liens. Ce module générait des fichiers .csv ou encore .xls, créant alors les associations plusieurs à plusieurs entre certaines tables et en remplissait d'autres grâce aux informations déjà contenues dans la base à l'issue du parseur XML.

– **Projet Bddictionnairique**

Ce programme était le programme principal, utilisé par les chercheurs du laboratoire pour leur travail. Il permettait l'affichage des informations contenues dans la base mais aussi la modification ou bien l'ajout d'informations.

Voici de façon illustrée l'enchainement des projets qui étaient alors en place :

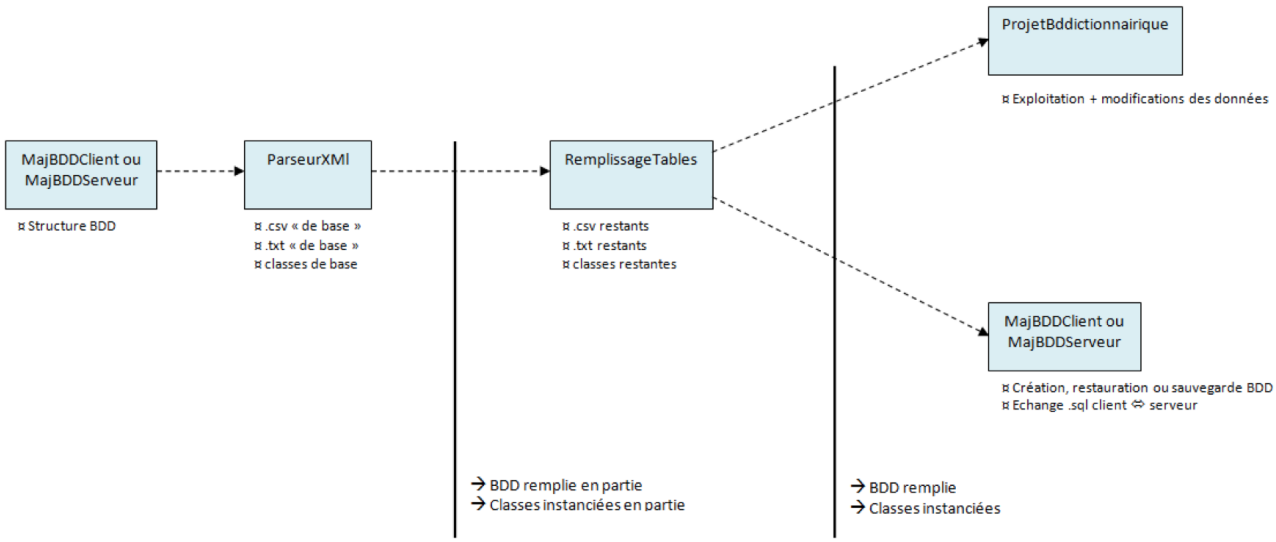


FIGURE 4.1 – Relations entre les différents projets constituant l'application

L'architecture était donc trop peu fiable. Aucune solution n'était alors en place pour permettre à l'utilisateur d'envoyer ses modifications sur le poste de Jean-Michel FOURNIER, servant alors de serveur principal. Aussi, une machine personnelle utilisée en temps que serveur est une configuration assez risquée. En effet, cet ordinateur ne bénéficiait d'aucune protection contre les attaques, ni même de solution de sauvegarde et de restauration. Enfin, il aurait fallu que ce poste soit en permanence en fonctionnement, afin de garantir un accès en haute disponibilité à chacun des utilisateurs voulant mettre à jour sa base.

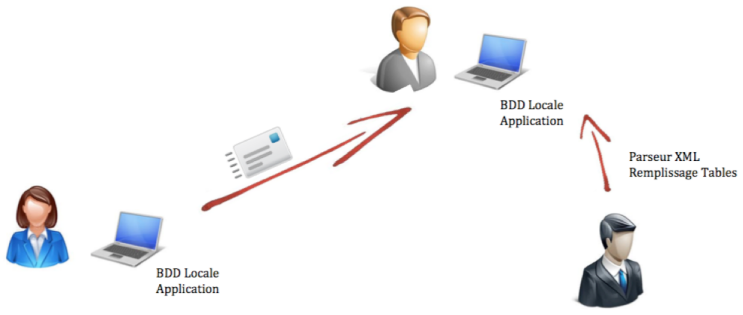


FIGURE 4.2 – Schéma de l'ancienne architecture

4.1.2 L'application Bddictionnairique

Mon P.F.E. ayant pour principal but d'effectuer la refonte de l'application Bddictionnairique, je me suis d'abord penché sur l'analyse de l'application existante. Dans un premier temps, j'ai donc mis en place une machine virtuelle sous Windows 7, afin de pouvoir tester l'actuelle version de cette application. J'ai pour cela importé un jeu de donné préalablement parsé.

Grâce à cette phase d'analyse poussée de la première version, j'ai pu me faire un avis sur ce qui allait et n'allait pas. J'ai découvert une application quelque peu brouillon de part son ergonomie. L'utilisateur se perd facilement dans les différents écrans, des fenêtres s'ouvrent dans tous les sens, se superposent ; Autant de choses qui ne facilitent pas l'expérience utilisateur.

Aussi, j'ai rencontré Mme MARTIN, chercheur et utilisatrice de l'application. Je lui ai demandé dans un premier temps de me faire part de l'ensemble des critiques qu'elle pouvait émettre au sujet de l'application. Ce premier constat m'a permis de voir les points faibles de l'ancienne application, et de proposer une maquette plus ergonomique et répondant davantage aux besoins des utilisateurs, l'expérience utilisateur étant vraiment le point central de la nouvelle application.

Le résultat des réunions que nous avons eu ainsi que les maquettes proposées sont à retrouver dans les comptes rendus de réunion ainsi que dans le cahier de spécifications systèmes.

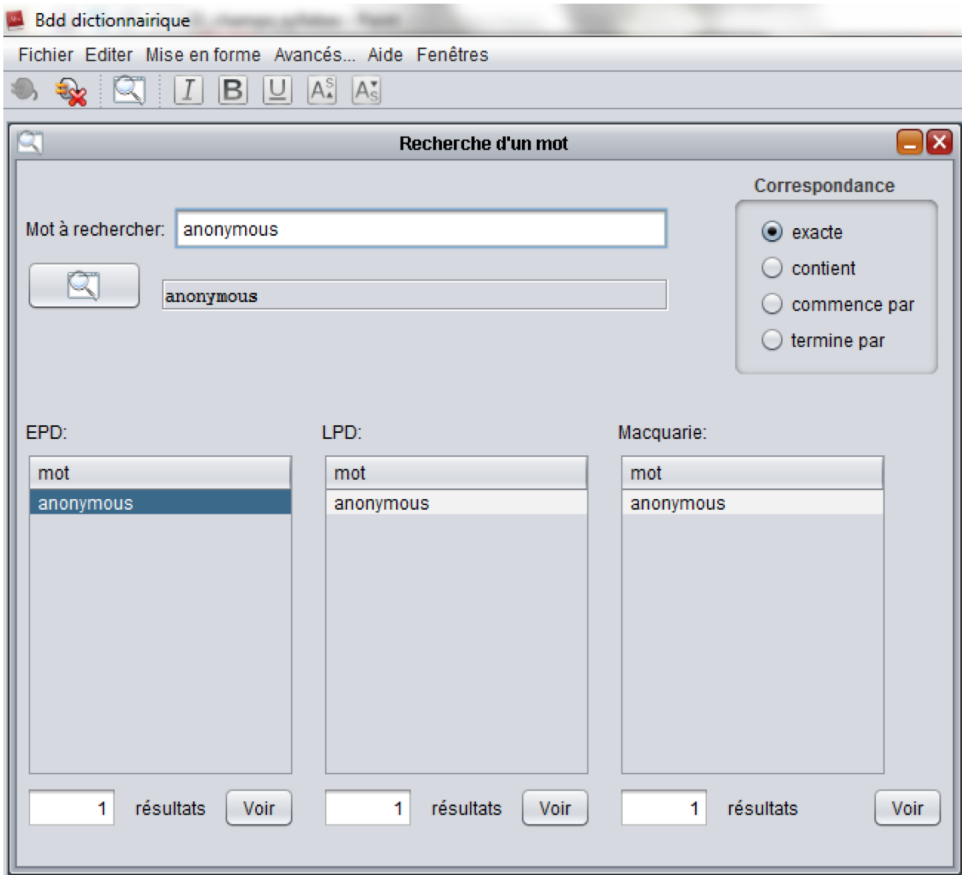


FIGURE 4.3 – Résultat recherche simple

Voici une illustration de la première version de l'application. Ci-dessus, on peut voir le résultat d'une recherche simple.

En premier lieu, on remarque qu'il manque certaines correspondances, telles que les négations "ne commence pas par", "ne contient pas", "ne termine pas par", etc.. Aussi, le mot est recopié juste sous le champ de saisi, mais cette information n'a pas d'intérêt. Pour finir, l'application n'est en rien générique et ne permet pas ici d'imaginer un nouveau dictionnaire. L'écran de résultat n'est pas adapté, il faudrait mettre les mains dans le code pour prévoir une nouvelle zone de résultat.

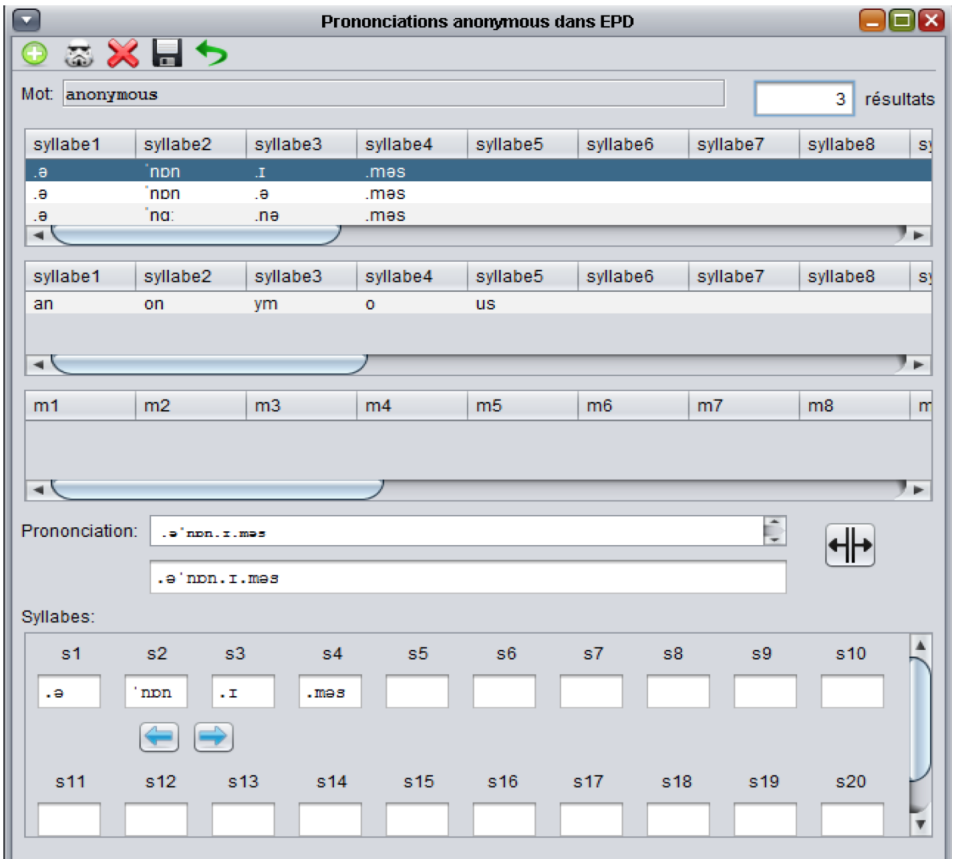


FIGURE 4.4 – Affichage du détail de la prononciation

Sur l'écran ci-dessus, on observe un manque d'information concernant les caractéristiques affichées. En effet, les informations présentes sur cet écran ne sont pas intuitives en terme de sens.

Prenons un dernier exemple d'écran avec celui ci-dessous. Cette fois, on observe l'écran permettant à l'utilisateur de renseigner les critères désirés pour une recherche avancée.

L'ergonomie de cet écran n'est pas très adaptée et on conçoit qu'il puisse être difficile pour un utilisateur de s'y retrouver dans l'ensemble de ces onglets.

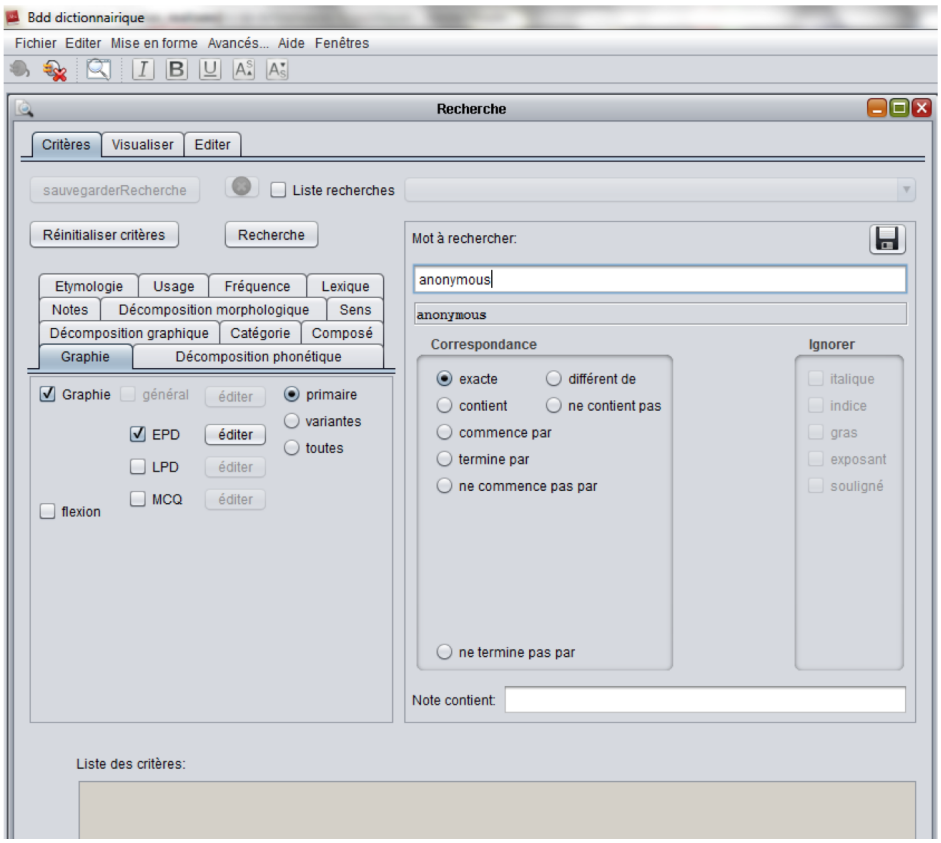


FIGURE 4.5 – Détail de la recherche avancée

On se rend bien compte que de nouvelles maquettes sont à proposer, visant principalement à effectuer une refonte complète de l'application.

Un autre élément que j'ai pris en compte a été le code source de l'application réalisé par le précédent développeur. Malheureusement, l'architecture de l'application n'était pas adaptée et optimale. Je n'y ai trouvé aucun commentaire, aucune documentation expliquant ces choix d'architecture et le rôle de chacune des fonctions.

Suite à l'analyse plus poussée de l'application en l'état, l'absence de tout commentaire et de tout document qui auraient pu m'aider à la compréhension m'a obligé à repartir de zéro. Il était impossible de continuer le projet débuté. Il s'agissait déjà d'un programme patché dans tous les sens, l'adaptation des applications à la nouvelle base de données aurait été bien trop compliquée. Cela m'a toutefois laissé la possibilité d'utiliser certains plugins bien utiles que je détaille par la suite.

4.1.3 Nouvelle architecture

L'avantage à repartir de zéro en ce qui concerne l'application est certain. Cela permet de se reposer certaines questions fondamentales d'architecture.

Suite à une réunion organisée avec le client, j'ai pu prendre bonne note des principales contraintes qu'il désirait, comme pouvoir accéder à l'application sans être connecté au net, application multi-plateforme. Nous avons donc opté pour une application en java avec un déploiement MySQL systématique sur chacun des postes utilisateurs.

Le projet a une architecture générale plutôt simple. En effet, le système est composé d'une base de données locale, qui contient tous les mots, et les informations relatives à ces derniers. Cette base de données communique avec l'application en elle-même via des requêtes SQL (Criteria).

Une base de données maître est quant à elle stockée sur un serveur externe (prestataire). Les bases de données locales sur les postes de chaque utilisateur sont une réplication de cette base principale. Ces bases contiennent les informations contenues dans les 3 dictionnaires.

Toutefois, les droits d'accès à ces bases ne sont pas identiques : il existe deux types d'utilisateurs. Tous les utilisateurs auront un accès total à leur base de données locale (lecture, écriture) mais un accès en lecture uniquement à la base de données maître. Seul l'administrateur (Jean-Michel FOURNIER) aura accès en écriture à cette base (notamment pour valider les modifications).

Le principe est que les utilisateurs non administrateurs travaillent en local, et dès que ceux-ci le souhaitent, ils proposent de mettre à jour les données de la base maître avec leur travail local. Cette procédure est soumise à approbation par l'utilisateur administrateur (qui joue ici le rôle de modérateur).

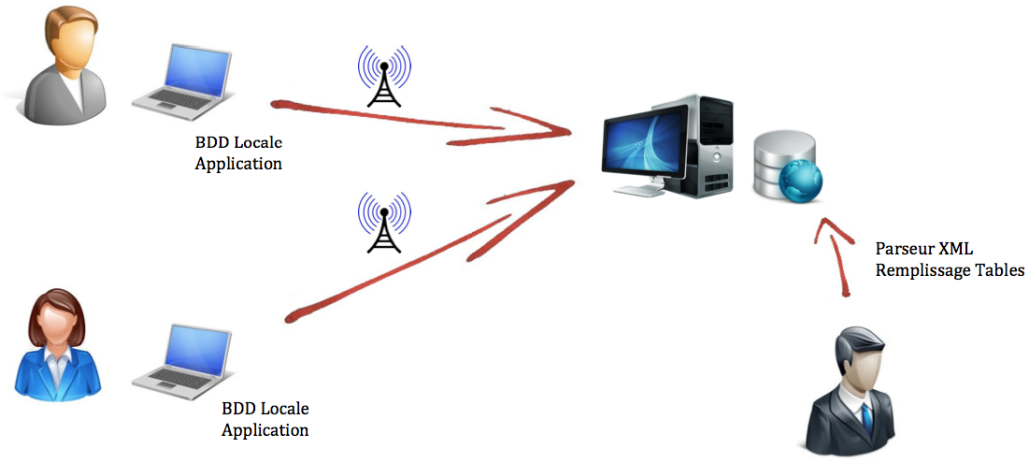


FIGURE 4.6 – Architecture retenue pour le projet

4.2 Mise en place de la nouvelle base de données

L'analyse de l'existant terminé, j'ai commencé par configurer l'environnement de travail en vue de développer la base de l'application. Pour se faire, la première chose que j'ai importé a été le fruit des 2 P.F.E réalisés précédemment, à savoir la base de données. Le SGBD étant MySQL, j'ai donc installé le serveur sur ma machine personnelle, puis j'ai utilisé MySQLWorkbench afin de configurer et d'administrer la base de données.

J'ai du faire face à une première difficulté concernant l'import de cette base de données. En effet, le fichier contenant la structure même de la base de données ne voulait pas s'importer. S'agissant d'un travail effectué l'année passée et l'étudiant n'étant plus joignable, j'ai donc du résoudre l'ensemble des erreurs indiquées par l'outil MySQLWorkbench afin de pouvoir commencer l'application au plus vite.

Les erreurs se concentraient toutes sur les clés étrangères. En effet, elles ne voulaient pas se mettre à jour lors de l'import du script de la structure de la base. Pour le résoudre, j'ai du ré-écrire à la main l'ensemble des requêtes de création de table, en prenant soin de respecter les standards MySQL en terme de notation.

Cette première difficulté, non prévue initialement, a engendré un léger retard que j'ai pu rattraper par la suite.

4.3 Refonte de l'application

Suite à l'installation de la nouvelle base de données, je me suis tourné du côté de l'application principale, à savoir Bddictionnairique. Comme je l'ai présenté plus haut, j'ai fait le choix de repartir de zéro en ce qui concerne la nouvelle application. J'ai toutefois proposé des maquettes, qui sont consultables dans le cahier de spécifications systèmes.

Une des décisions importantes qui a été prise a consisté en la fusion de l'ensemble des 5 programmes existants en un seul. L'application dispose d'une partie administration, accessible via login et mot de passe. Il faut bien sûr avoir le profile Administrateur pour pouvoir s'y connecter. Mon premier travail a été de répartir les programmes existants pour savoir si oui ou non ils relevaient des droits administrateurs.

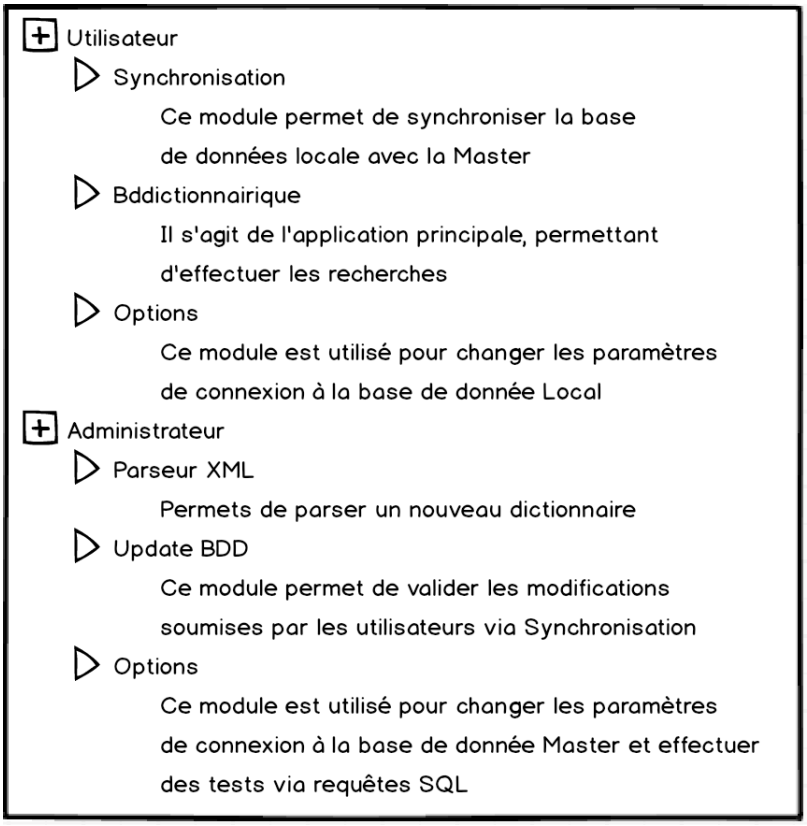


FIGURE 4.7 – Répartition des programmes

Ce fonctionnement permet de ne pas déployer 5 programmes différents, et améliore l'expérience utilisateur. Tout peut être fait via un seul programme, simplifiant ainsi l'utilisation des fonctionnalités.

Nous allons à présent détailler les interfaces d'accueil de l'utilisateur ainsi que de l'administrateur.

Voici donc ci-après l'écran d'accueil de l'application. Il est composé d'un titre, d'un logo (non à jour mais libre de droits), des boutons d'accès aux différents programmes, ainsi que d'un bouton pour accéder à l'interface d'administration, situé dans le footer.

Lorsque l'utilisateur passe sa souris sur un des carrés représentant un programme, le détail sur la fonctionnalité de ce dit programme s'affichera. C'est ce que l'on peut voir sur l'illustration, où la souris de l'utilisateur survolait le carré "Synchronisation".

On remarque également l'absence de menu sur cette application. En effet, je n'ai pas trouvé le besoin de mettre en place ce menu, toutefois, il est prêt à l'implémentation avec un exemple que j'ai laissé dans le code. Il pourra donc être complété, pour rajouter certaines fonctionnalités génériques (augmentation de la taille de la police, imprimante, ...) au libre choix du client.

Sous la plateforme MacOS (à partir de la version Lion), l'application bénéficie d'une fonctionnalité très intéressante et utilisée par de nombreux logiciels, à savoir la possibilité d'étendre l'application dans un nouveau bureau. Il s'agit de la double flèche que l'on retrouve en haut à droite, dans le bandeau de l'application. Cette fonctionnalité n'existe pas sous Windows, on pourra toutefois la comparer au mode "plein écran".

FIGURE 4.8 – Menu d'accueil de l'utilisateur

Passons maintenant à l'écran de login. Pour rappel, on arrive sur cet écran lorsque l'utilisateur souhaite s'authentifier afin d'arriver sur le mode administrateur de l'application.

Cet écran est très simple d'utilisation et de fonctionnement. Il s'agit de préciser son login et son mot de passe, et l'application valide ou non la transaction.

Lorsqu'une saisie est incorrecte, un message apparaît indiquant la raison de l'échec. Ainsi, des messages tels que "Utilisateur inconnu", "Erreur de mot de passe" ou encore "Droits insuffisants" apparaissent en rouge, permettant à l'utilisateur de corriger son erreur.

En ce qui concerne les données, une table Utilisateur existe dans la base de données. Elle contient plusieurs informations qui sont le nom, l'adresse email, le mot de passe et un booléen pour savoir si l'utilisateur est administrateur ou non.

Concernant le champ mot de passe, le but n'était pas de faire une application ultra sécurisée, mais bien de protéger l'accès à la console d'administration, afin d'en restreindre les modérateurs. Ainsi, le mot de passe n'est bien évidemment pas stocké en clair, mais est simplement crypté en MD5. Lors de la saisie du mot de passe dans le champ et suite à la validation, l'application va crypter le mot de passe saisi par l'utilisateur, et vérifier l'adéquation du résultat obtenu avec le contenu de la base. Cela évite ainsi de pouvoir pirater la base et d'observer le mot de passe en clair, même si un cryptage MD5 est facilement décryptable sur un mot de passe simple.

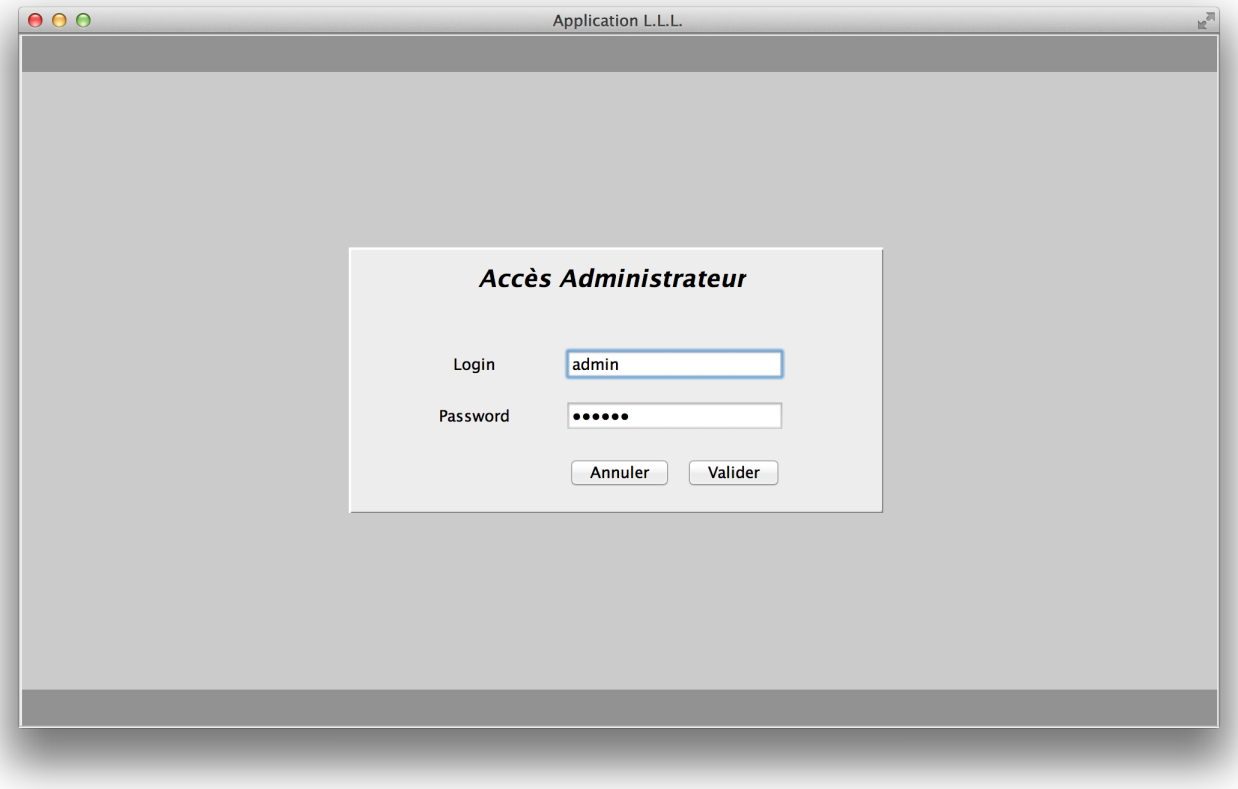


FIGURE 4.9 – Écran de login

Terminons ce tour des écrans de menu par celui d'administration. Comme vous pouvez le remarquer sur l'illustration ci-dessous, l'ergonomie reste inchangée et le fonctionnement est semblable.

On retrouve ainsi le titre "Administration", un logo libre de droits, les différents programmes, et le bouton de déconnexion dans le header en haut à gauche.

A noter que la connexion ne gère pas de cookie, ainsi, lorsque l'utilisateur clique sur le bouton déconnexion, il se retrouve bel et bien déconnecté, et n'a pas d'autre choix que de se ré authentifier pour revenir sur la console d'administration.

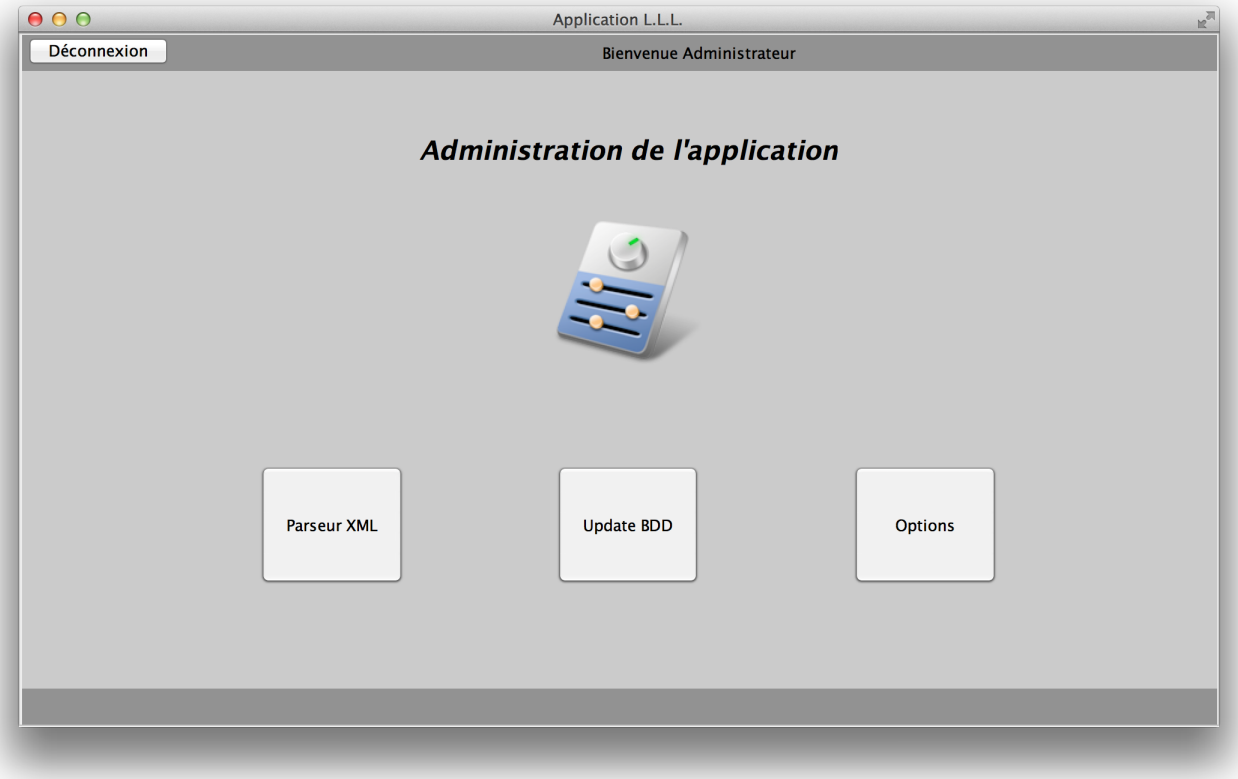


FIGURE 4.10 – Menu d'accueil de l'administrateur

4.3.1 L'accès aux données

Avant de s'attaquer à l'application à proprement parler, il a fallu régler une question cruciale : **comment accéder aux données ?**

En effet, l'application interagissant avec 2 bases de données, chacune d'entre elles comportant plus de 25 tables assez complexes, il a fallu trouver un moyen à la fois simple et générique pour accéder aux données. Suite à l'étude de plusieurs solutions, je me suis très rapidement tourné vers Hibernate. S'agissant d'un outils que je connais, et qui est optimisé pour le traitement de gros volumes de données, mon choix s'est très vite justifié et je l'ai donc implémenté rapidement.

Hibernate est un framework open source, gérant la persistance des objets en base de données relationnelles. Hibernate est complètement modulable en terme d'architecture. Il est dans notre cas utilisé dans un développement de client lourd. Hibernate apporte une solution aux problèmes d'adaptation entre le paradigme objet et les SGBD en remplaçant les accès à la base de données par des appels à des méthodes objets de haut niveau.

Concrètement, la génération des classes par Hibernate permet de mapper un objet java avec une table dans la base de données, et s'occupe tout seul de faire le lien entre les objets.

FIGURE 4.11 – Logo Hibernate

4.3.2 Les modules

Nous allons à présent détailler l'avancement et le travail effectué dans chacun des 6 modules.

Il est important de noter que les 6 modules ne sont pas terminés. Cependant ils disposent tous au minimum d'une base solide sur laquelle se baser pour poursuivre le développement. Tous les modules sont basés sur le même "principe". Une application dispose d'un volet, et d'un "content". On peut très facilement masquer le volet.

Les applications se chargent et se déchargent comme des modules indépendants. Il est très important de décharger le précédent module avant d'en charger un nouveau. En effet, cela pourrait poser des problèmes d'accès à des pages désallouées.

4.3.2.1 Synchronisation

Le module synchronisation permet à l'utilisateur de gérer les mises à jour de son application. Ce module se décompose en 3 parties, accessibles respectivement via le volet à gauche de l'application.

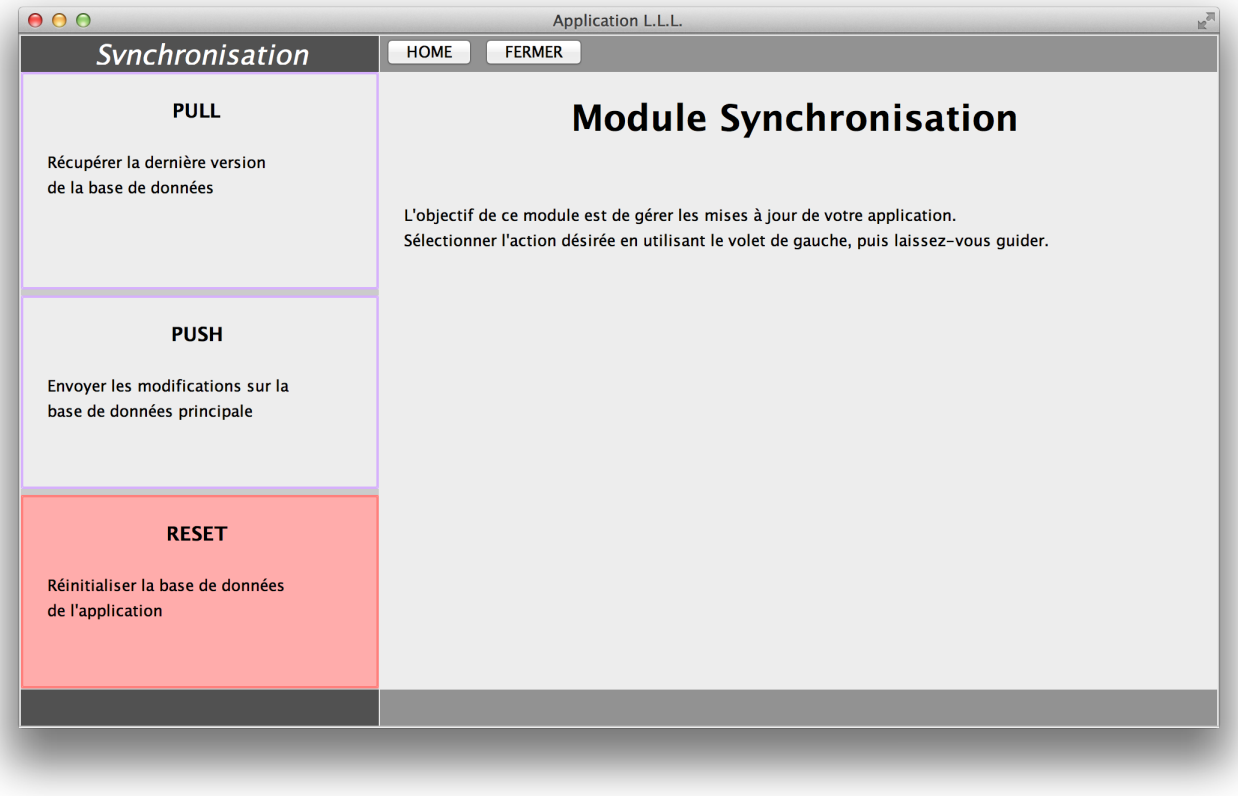


FIGURE 4.12 – Module Synchronisation

L'entrée **PULL** permet de vérifier si la base de données locale est à jour. Ainsi, l'application va chercher sur le serveur la version de la base de données Master, et va donc vérifier si une nouvelle version est sortie. Pour se faire, je voulais initialement récupérer le champ Update_Time grâce à la requête show table status. Toutefois, ce champ n'est pas mis à jour avec le type de stockage Innodb. Il faudra donc réfléchir à une autre solution pour vérifier la version de la base.

L'entrée **PUSH** permet à l'utilisateur de proposer ses modifications. Une fois les modifications envoyées, il faudra bien entendu que le modérateur, jouant le rôle d'administrateur, valide les modifications. La fonction de push n'étant pas implémenter, il faudra trouver un moyen fiable et simple d'effectuer l'envoi d'informations sur la base master. Une étude de cette fonction est disponible dans le compte rendu de réunion.

Enfin, l'entrée **RESET** est un peu particulière et doit être utilisée avec précaution. Elle permet de réinitialiser la base de données. Cette fonction procède à un effacement des tables du schéma bddictionnairique puis elle les recrée. Il faudra ensuite procéder au remplissage de la base de données vierge, en utilisant la fonction PULL par exemple.

Ce module n'est pas achevé. La navigation entre les entrées est implémentée, mais les fonctions en elles-mêmes ne le sont pas.

4.3.2.2 Bddictionnairique

Ce module est le programme principal de l'application. En effet, l'ensemble des modules annexes ne servent qu'à permettre le bon fonctionnement de celui-ci ou bien le rajout des fonctionnalités. On pourrait le qualifier comme étant le cœur de cette application.

Voici ci-après une illustration de ce module :

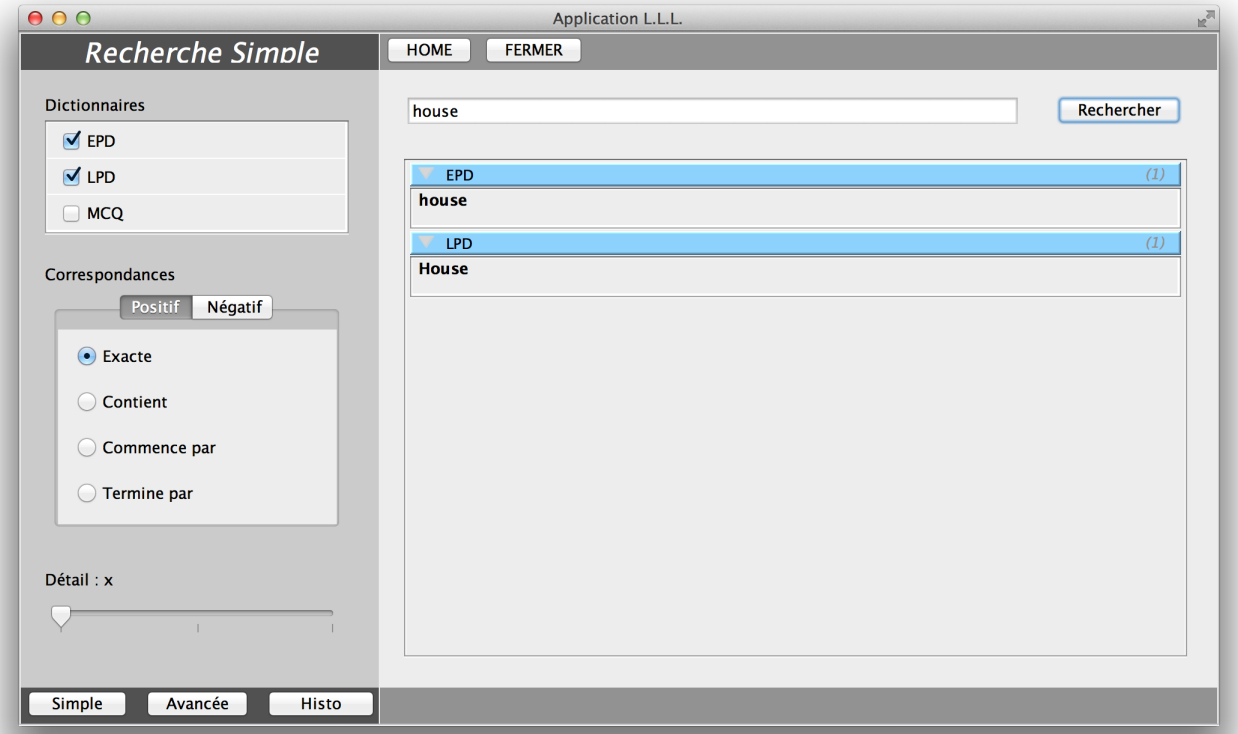


FIGURE 4.13 – Module Bddictionnairique

Ergonomiquement similaire aux autres modules, on retrouve donc le volet ainsi que le content. Dans le footer du volet, on accède aux fonctions suivantes :

- Recherche Simple
  Réaliser des recherches simples, utilisant les notions de correspondances
- Recherche Avancée
  Réaliser des recherches se basant sur des propriétés bien spécifiques
- Historique
  Accéder à l'historique des recherches

Le développement de ce programme était initialement l'objectif principal de mon P.F.E.. Toutefois, certains problèmes de modélisation de la base de données m'ont obligé à stopper le développement et à retravailler sur des fonctions annexes. Dans le module Bddictionnairique, l'historique est totalement implémenté, la recherche simple est en cours, et la recherche avancée n'a pas été commencée.

Commençons par l'entrée Recherche Simple. On retrouve dans ce volet la liste des dictionnaires présents en base de données. Ainsi, l'utilisateur peut à sa guise sélectionner les dictionnaires dans lesquels effectuer la recherche. Ensuite, le bloc "Positif / Négatif" liste l'ensemble des correspondances que l'on peut mettre en œuvre lors d'une recherche. Ainsi, dans le Positif, on retrouve "Exact", "Contient", "Commence par", "Termine par", alors que dans le négatif, "Différent", "Ne contient pas", "Ne commence pas par", "Ne termine pas par". L'ensemble des correspondances est selon moi couvert. On peut toutefois en rajouter si le client le demande.

Pour finir, on propose un niveau de détail, qui agit sur le niveau de détail du résultat. Ainsi, le niveau 1 n'affiche que le mot en lui même, alors que le niveau 2 va afficher le mot, les catégories, le découpage syllabique, la région, le schéma et le type. Le niveau 3 quant à lui est pour le moment idem au niveau 2, il s'agit d'ajouter par la suite la fréquence et l'occurrence.

Lorsqu'une recherche est lancée, la liste des résultats s'affiche dans la partie "content" de l'écran. Les résultats sont regroupés par dictionnaire dans des listes dépliantes. Lorsque l'on clique sur un mot, celui-ci apparait en surbrillance et 2 boutons s'affichent sur la même ligne.

Masquer permet d'effacer le résultat de la liste de recherche. Ainsi, il n'apparaîtra ni dans la liste de résultat ni dans l'historique. On pourrait très bien imaginer stocker ces résultats masqués afin de les ré afficher à la demande de l'utilisateur (Liste des mots masqués par recherche).

Éditer permet l'édition des mots. Ainsi, cliquer sur éditer fera apparaître une nouvelle fenêtre, dans laquelle l'ensemble des informations du mot édité seront affichées et modifiables. L'édition peut très bien se faire via le résultat d'une recherche (simple ou avancée) ou bien via l'historique.

4.3.2.3 Options

Le menu option a pour principal but de paramétrer l'accès aux bases de données. On retrouve ce programme aussi bien du côté utilisateur que du côté administrateur.

On peut donc influer sur les paramètres de la base de données locale. Les paramètres modifiables sont les suivants :

- Driver SGBD
- Url de la base de données
- Nom d'utilisateur
- Mot de passe

La case à cocher "Se souvenir" permet de préciser si l'on souhaite appliquer les modifications pour l'instance de l'application ou bien garder ces nouveaux paramètres comme paramètres par défaut.

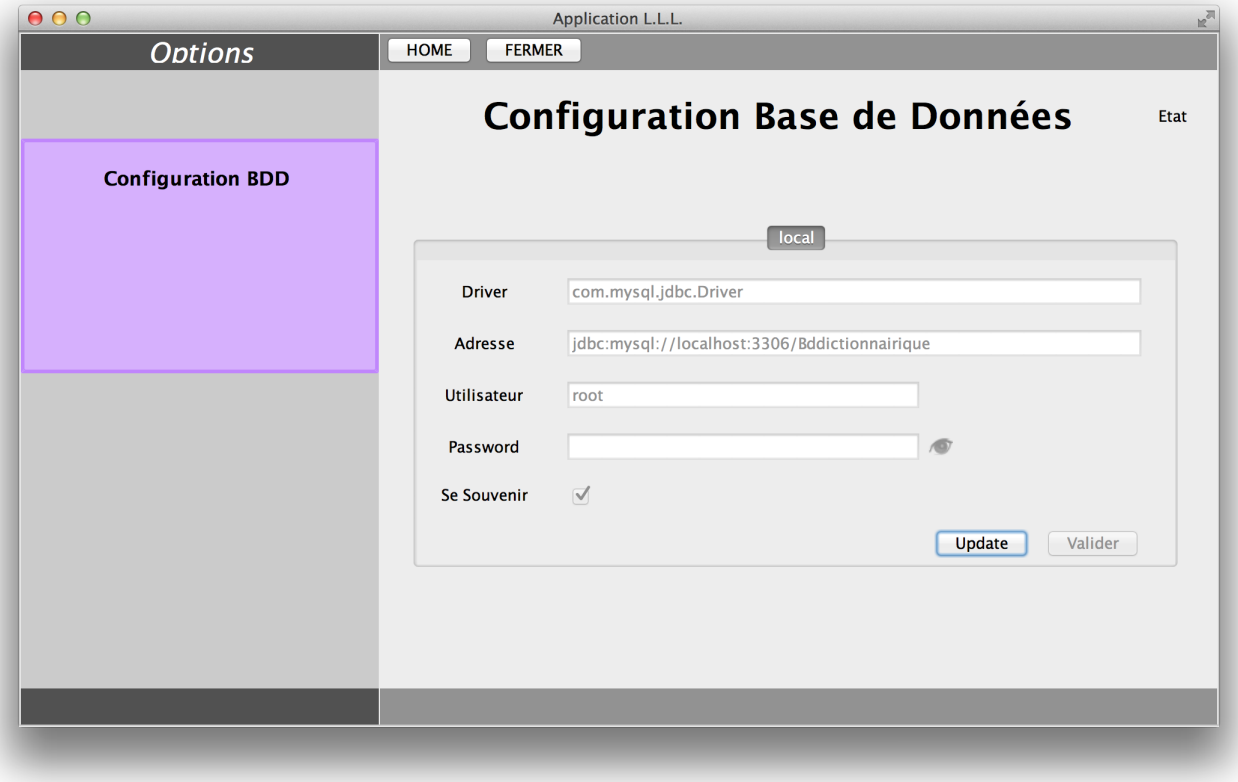


FIGURE 4.14 – Module Option - User

Lorsque l'on clique sur Update, les champs deviennent éditables. Pour le mot de passe, maintenir le clic gauche de la souris sur le petit œil permet de voir le mot de passe saisi ; relâcher ce clic permet de re masquer la saisie.

Du côté administrateur, on observe le même module, avec toutefois une entrée supplémentaire. Le tronc commun de ce programme reste le paramétrage de l'accès à la base de données (master ici). Aussi, l'administrateur dispose d'une console afin de saisir directement des requêtes dans la base de données (locale ou master au choix).

L'entrée paramétrage des accès bases de données est implémenté. Si la case "Se souvenir" n'est pas cochée, alors on renouvelle l'instance avec les nouveaux paramètres temporaires. Si la case est cochée, alors on écrit les nouveaux paramètres dans des fichiers de propriétés, qui seront chargés au prochain lancement de l'application.

L'entrée requêtes SQL n'est quant à lui pas implémenté.

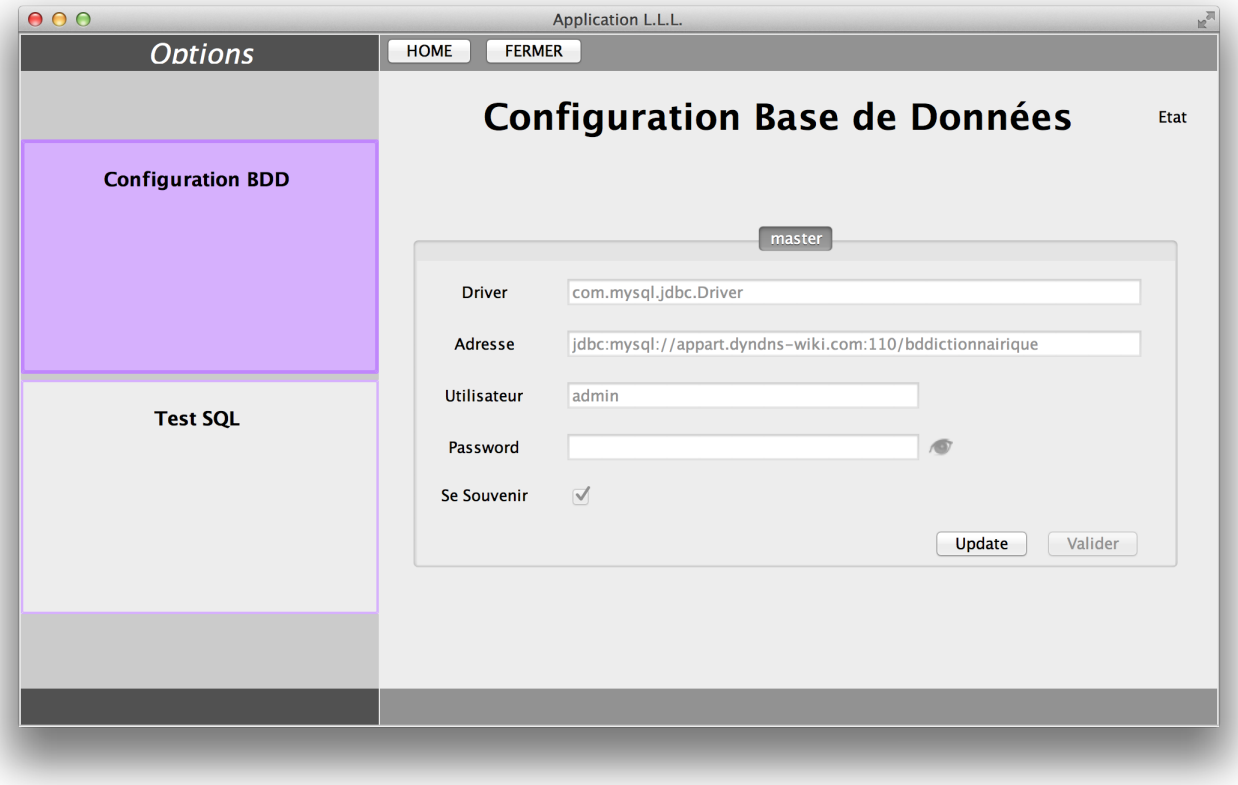


FIGURE 4.15 – Module Option - Administration

4.3.2.4 Parseur XML

Le parseur XML est très important pour le fonctionnement de l'application. En effet, il permet de convertir les données des dictionnaires issues des fichiers XML en données que l'on peut insérer dans les bases de données.

Le parseur XML n'est pas totalement implémenté. En effet, ce module fait l'objet d'un stage de fin d'année pour un étudiant de 4ème année. Je vais tout de même le détailler sommairement.

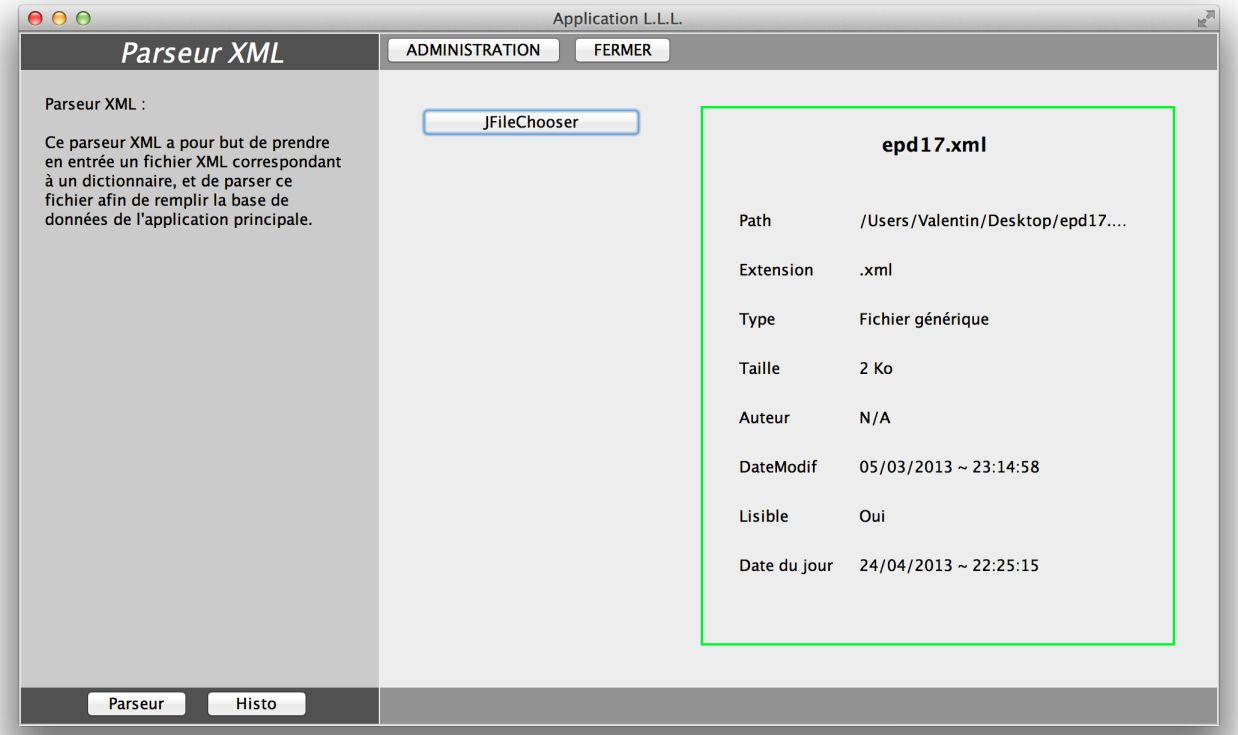


FIGURE 4.16 – Module Parseur XML

J'ai donc implémenté la base de ce module. L'utilisateur peut donc, grâce à un file chooser, sélectionner un fichier. Lorsqu'un fichier est sélectionné, on récupère l'ensemble des informations le concernant (nom, chemin absolu, extension, taille, ...). A ce moment là, on vérifie simplement que l'extension est bien "XML", le cas échéant on affiche "Fichier non supporté". Il s'agit d'un mini contrôle rapide à effectuer, et évitant tout problème de parse.

Chaque fichier XML valide sélectionné par l'utilisateur est répertorié dans un historique, accessible via le footer du volet.

Il s'agit donc maintenant d'implémenter la fonction de parsing à proprement parler. Il faudra peut être incorporer un Document Type Definition (D.T.D.) afin de vérifier la bonne structure du fichier XML. Attention toutefois, il faudra bien préciser un D.T.D. par dictionnaire potentiel, la structure n'étant pas identique à chaque fois.

4.3.2.5 Update BDD

Le module Update BDD a pour but de valider les modifications soumises par les utilisateurs. Ce programme est uniquement accessible par le menu administrateur. Je n'ai fait que que commencer ce module, ainsi la base est faite. Il faut désormais construire les fonctions qui vont lister les différences. Une vraie réflexion est à mener à ce sujet, et des réponses doivent être apportées notamment sur le "comment faire ?".

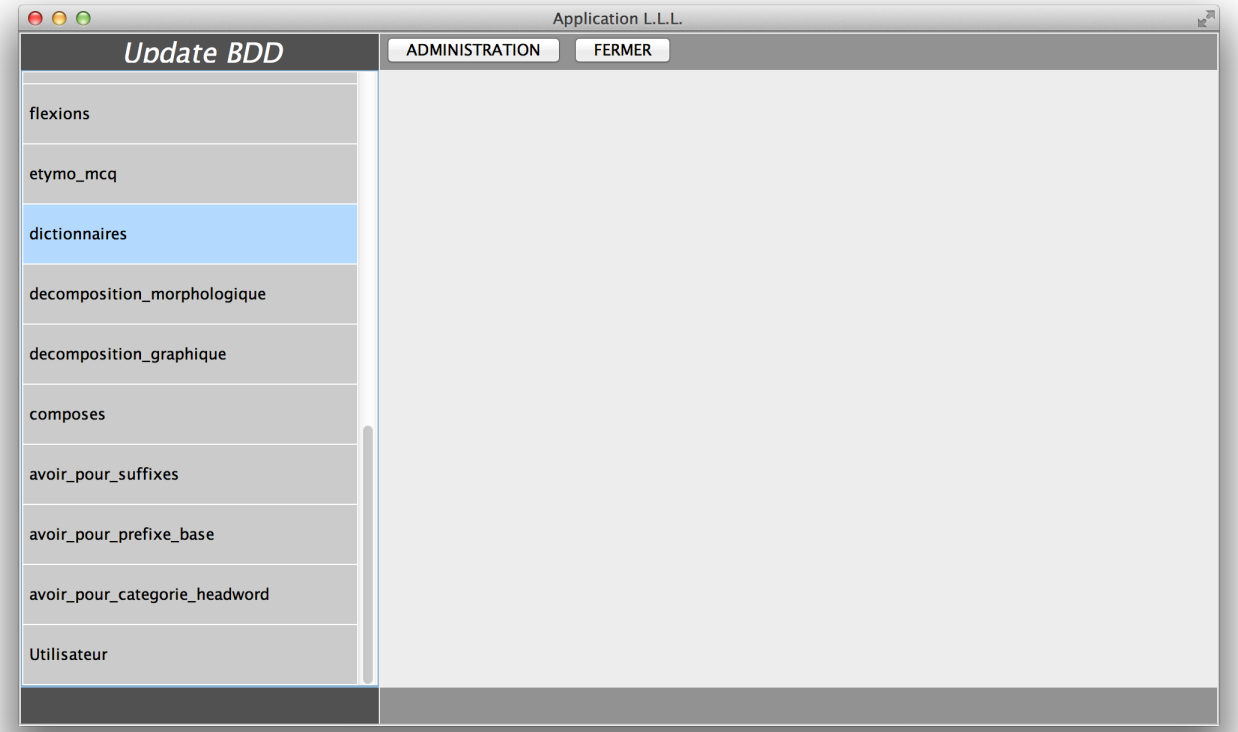


FIGURE 4.17 – Module Update BDD

On retrouve, dans le volet, la liste des tables du schéma bddictionnairique de la base de données locale. La base de l'update s'arrête ici.

4.4 Collaboration avec le CNRS d'Orléans

Dans le cadre de la mise en production de la base de données Master, j'ai eu l'occasion de collaborer avec l'équipe informatique du CNRS d'Orléans. En effet, pour rappel, la base de données Master de la nouvelle application ne sera plus hébergée sur le poste de Jean Michel FOURNIER mais sur les serveurs du CNRS.

A ce titre, tout au long de mon P.F.E., j'ai été en contact avec Richard Walter (Responsable du service "Supports aux projets numériques" de l'université d'Orléans) et plus particulièrement Ly A Phat (Intégrateur). Nous avons beaucoup échangé sur l'hébergement que j'avais étudié. A Phat a donc géré le côté administratif pour l'hébergement sur les serveurs du CNRS. Je lui ai ensuite envoyé le MCD de l'application, pour qu'il puisse déployer la base de données.

En ce qui concerne les problèmes d'accès, c'est le point sensible de cet hébergement. Pour le moment, l'adresse IP publique de l'université Site Portalis permet en toute logique d'accéder à la base de données.

Suite à un problème on ne peut plus bloquant survenu lors du développement, j'ai du ré étudier la base de données. Ainsi, notre collaboration s'est interrompue le temps que je remette les choses en place. Un nouveau script de base de données a été établi. Si ce script passe les phases de tests (intégration + intégrité), alors il conviendra de faire parvenir le script sql à A Phat qui se chargera de le déployer.

Travail restant

Ce chapitre a pour objectif de présenter le travail restant à effectuer. En effet, comme je l'ai expliqué précédemment dans ce rapport, je n'ai pas pu tenir le planning prévisionnel pour plusieurs raisons.

Nous allons donc étudier les tâches à réaliser. Attention, la liste n'est en aucun cas exhaustive, elle résulte des principaux objectifs qui m'ont été fixé en début de P.F.E. ou que j'ai pu répertorier au cours de mes analyses.

5.1 Parseur XML

Le parseur XML fait partie des modules dont l'implémentation n'est pas terminée. Lors de mon P.F.E., j'ai du apporter plusieurs modifications importantes sur la base de données, modifications altérant la structure même de la base (M.C.D. revu). Dès lors, le parseur XML réalisé par l'étudiant l'année dernière était obsolète.

Le parseur XML fait l'objet d'un stage de fin d'année pour un étudiant de 4ème année. Le principal objectif de ce stage sera d'adapter le parseur XML à la nouvelle base de données. Comme nous l'avons vu plus haut, le parseur XML est vraiment un maillon essentiel, intervenant tôt dans le schéma représentant les relations entre les différents modules du projet.

Ce module dispose d'une base que j'ai réalisé. Pour plus d'information, se référer au chapitre "Travail réalisé". Le travail restant à faire sur ce module est donc la fonction de parse, qui prendra en entrée un fichier XML valide, puis qui insèrera en base de données les données extraites.

Également, en complément de ce module, il conviendra de réaliser plusieurs procédures stockées, qui seront charger d'effectuer un traitement sur les données insérées via le parseur. Les procédures stockées sont un ensemble d'instructions SQL pré-compilées, stockées dans une base de données et exécutées sur demande par le S.G.B.D. qui manipule la base de données.

Le choix des procédures stockées se justifie. Les requêtes dites "classiques" envoyées à un serveur SQL font l'objet d'une 'analyse syntaxique' puis d'une interprétation avant d'être exécutées. Ces étapes sont très lourdes si l'on envoie plusieurs requêtes complexes.
Les procédures stockées répondent à ce problème : une requête n'est envoyée qu'une unique fois sur le réseau puis analysée, interprétée et stockée sur le serveur sous forme exécutable (pré-compilée). Pour qu'elle soit exécutée, le client n'a qu'à envoyer une requête comportant le nom de la procédure stockée. On peut ainsi passer des paramètres à une procédure stockée lors de son appel, et recevoir le résultat de ses opérations comme celui de toute requête SQL.

Le but des procédures stockées sera de créer une entrée dans le dictionnaire principal pour chacune des entrées des dictionnaires parsés. Pour rappel, le but principal de l'application est de regrouper sous une unique entrée (entrée du dictionnaire principal, ex table_liens) plusieurs entrées de plusieurs dictionnaires (epd, lpd, mcq). Ainsi, les données de ces entrées se verront fusionnées en une seule entrée, simplifiant l'utilisation et permettant une analyse généralisées et plus complète de chacune des entrées (mots).

EPD	LPD	MCQ	Proc Stockée (Traitement)
house	house	house	-> entrée 1
House	house	house	-> entrée 2
house	House		-> entrée 3
	house		-> entrée 4

FIGURE 5.1 – Traitement des procédures stockées

Un autre traitement que devront effectuer les procédures stockées est le découpage syllabique. Suivant une règle très précise (déjà mise en place), elles devront remplir les tables de données calculées (découpages_xXx).

L'ensemble des traitements annexes aura pour but de compléter la base de données en calculant des données à partir des données brutes issues des fichiers XML traités par le parseur. Il est très important d'être méticuleux et très précis sur l'analyse de ces données afin d'éviter d'enregistrer en base des données erronées, pouvant cour-circuiter et rendre incohérente les recherches effectuer.

Pour faciliter le traitement, l'analyse des balises pour les 3 dictionnaires commerciaux utilisés actuellement a été faite. Attention toutefois, il serait très intéressant de lister l'ensemble des balises, et voir si l'analyse déjà effectuée est correcte, certaines balises ayant été oubliées dans la dernière version du parseur.

5.2 Synchronisation

Le module synchronisation dispose également d'une base. Le traitement Reset est implémenté (peut-être à remettre en forme), mais les requêtes de PULL et PUSH ne le sont pas. En effet, il faudra réfléchir à un moyen d'insérer en base des données temporaires (temporary tables ?). Le but étant, quand l'utilisateur décide d'envoyer ses modifications sur la base principale, de les enregistrer sans les valider, ou du moins de pouvoir les repérer en vue de les supprimer si le modérateur ne les valide pas.

Aussi, il faudra étudier le merge, fonction Hibernate permettant la "fusion" de 2 bases de données en vue de leur synchronisation. Cette fonctionnalité s'apparente au fonctionnement maître / esclave, où recopie d'une base de back up.

J'ai, au cours de mon P.F.E., mené une première analyse qui peut servir de base à l'élaboration de ce module. Rapidement, j'avais écarté l'utilisation de fichier et l'envoie par mail, car de trop nombreux inconvénients peuvent subvenir à ces méthodes. L'ajout en base de données temporaires via requêtes est, selon moi, la meilleure solution.

5.3 Hibernate

L'utilisation d'Hibernate facilite grandement les transactions dans ce type de projet. En effet, la multitude de table m'ont obligé à utiliser ce genre d'outils, rendant l'application plus modulable et générique. Coupler cet outils à des Data Access Object (D.A.O.) rends le travail bien plus simple.

Toutefois, paramétrer Hibernate n'est pas toujours évident, surtout lorsque l'on manipule 2 bases de données bien distinctes. Ainsi, il faut gérer 2 connexions différentes.

Le travail à faire sur Hibernate est d'améliorer les constantes de connexions. Ainsi, des modules tels que c3p0 peuvent être utiliser, rendant plus viable l'utilisation d'Hibernate. Le chargement de pilote *jdb* à chaque demande de connexion est couteuse. Il est donc préférable d'utiliser un pool de connexion qui est un mécanisme permettant de créer un ensemble de connexions (instancier les connexions une fois pour toutes) et de les réutiliser à la demande par un utilisateur. Hibernate utilise un mécanisme de gestion de pool très basique. Il est est donc conseillé d'utiliser des pools performant afin de remplacer celui d'hibernate. C3P0 est l'un des plus connu, avec DBPC. http://ouedmouss.blogspot.fr/2008/02/hibernate-et-pool-de-connexion-c3p0.html

Aussi, il faudra gérer les "erreurs" telles que l'interruption du serveur MySQL, connexion impossible, session fail, la reconnexion suite à un changement de paramètres, etc..

5.4 Mais aussi...

Bien entendu, l'application principale est à continuer. Les priorités seront à redéfinir avec le client. L'expression des besoins est disponible dans mon cahier de spécifications, mais n'est surement pas complète.

Ainsi, la recherche simple est quasiment terminée, il ne reste que les niveaux de détails à revoir, le niveau 3 n'étant pas implémenté. Aussi, la recherche avancée doit se calquer sur la recherche simple, avec dans le volet, les critères regroupés. Pour avoir une vision globale, il convient de reprendre les maquettes de mon cahier de spécifications.

Tout au long de mon P.F.E., je me suis attaché à déployer l'application sur une machine MacOS de la salle TP Système. Ainsi, les clients pouvaient consulter l'application lorsqu'ils le souhaitaient, et pouvaient ainsi me remonter leurs remarques ou des bugs. Il existe donc beaucoup de petites fonctionnalités à implémenter, telles que :

- L'ajout de raccourcis clavier (entrée lance la recherche si le champ n'est pas vide)
- Bloquer la recherche si le champ est vide ou si aucun dictionnaire n'est sélectionné
- Permettre d'effectuer une nouvelle recherche ou de rechercher dans une recherche déjà effectuée (les lister dans une liste déroulante)
- Placer les méthodes importantes dans des threads
- Passer les D.A.O. en singleton permettrait de ne pas réinstancier à chaque requête
- Permettre de sauvegarder l'historique et de le recharger au lancement de l'application

Bilan Personnel

6.1 Compétences acquises

Ce P.F.E. est un projet de grande envergure si on le compare à ceux que nous avons réalisés au cours de notre formation.

Le concept de ce projet est en effet un peu particulier. Le fait d'être seul est quelque peu déstabilisant au début, mais cela nous permet de nous placer dans la peau d'un chef de projet qui doit sans cesse gérer les différentes phases (cycle en V), et de réagir pro-activement à chacun des problèmes qui survient. Je me suis rendu compte, au fur et à mesure de l'avancement, que ces problèmes peuvent être de différentes natures. De la durée de la tâche mal planifiée, à la priorité des tâches en elles mêmes, en passant par la compréhension du sujet, ou bien la reprise d'un existant. Ce dernier point a été très délicat dans le cadre de mon projet de fin d'études.

Ce P.F.E. m'est apparu comme très intéressant du point de vue de mon projet professionnel. Ainsi, j'ai pu découvrir et approfondir les différents profil métiers interagissant en permanence dans une équipe de projet informatique.

Aussi, ce projet a été très bénéfique pour moi. Il m'a permis de me rendre compte de l'enchainement des process, ainsi que des difficultés préalables à prendre en compte de façon systématique. Le projet sur lequel j'ai travaillé a réuni un ensemble de phases, passant de l'analyse à la conception, de la rédaction au développement, des tests aux livrables. Ce mélange de compétence a en quelques sortes résumé les 3 années d'ingénierie passées au département informatique de Polytech' TOURS.

Pour finir, ce projet ayant un aspect "Recherche", j'ai été amené à chercher diverses solutions technologiques à de nombreuses problématiques rencontrées. Ainsi, j'ai été amené à travailler avec des outils, des frameworks et mêmes des plugins que je n'avais jamais utilisé auparavant.

6.2 Difficultés rencontrées

Au fur et à mesure de l'avancement de mon projet de fin d'études, j'ai dû faire face à de nombreuses difficultés. Certaines solvables assez simplement, d'autres nécessitant plus d'analyse, et d'autres encore que je n'aurais pas du rencontrer.

En premier lieu, mon P.F.E. s'inscrivant dans la continuité de plusieurs projets (existants), j'ai été surpris de ne voir aucune documentation particulière. Aussi, l'absence de tout commentaire dans un projet de développement a considérablement ralenti mon avancement. Je tiens à souligner tout particulièrement ce point car il a été très bloquant pour moi, ne serait-ce que pour comprendre l'environnement et le sujet de mon projet.

L'étude de l'existant s'est donc avérée difficile. Le code n'étant pas optimisé ni même ordonné, je n'ai pas eu d'autre choix que d'en faire abstraction et de repartir de zéro en ce qui concerne le développement

de l'application.

Aussi, la modélisation de la base de données, résultant des 2 P.F.E. des années passées, n'ayant pas été testée dans une application quelconque, l'intégrité des données pouvait être remis en cause. A ce sujet, je me rendu compte après avoir débuté le développement que la modélisation ne convenait pas. J'ai donc du stopper la phase de développement, et reprendre l'analyse de la base de données, fondement de l'application. Cette difficulté, très critique, a impacté de manière très conséquente mon planning prévisionnel. Ainsi, il a fallu redéfinir de nouvelles tâches et essayer tant que possible de combler ce contretemps.

Conclusion

Ce projet de fin d'études n'a pas été très évident. Je n'avais pas réalisé la difficulté à reprendre un existant conséquent en l'absence de tout document. Cette notion de "reprise" fait la spécificité de ce projet. Il est alors très différent d'un projet de recherche, car il nécessite une période d'adaptation pendant laquelle il faut tenter de maîtriser le cheminement du précédant développeur afin de comprendre l'ensemble de ses choix.

Cette difficulté a toutefois été un avantage sur certains points. Elle m'a permis de repartir sur des bases saines, technologiquement à jour, et surtout facilement compréhensibles et évolutives. Ce projet m'a également beaucoup apporté sur le plan technique. J'ai ainsi pu accroitre mes compétences en programmation ainsi qu'en framework spécifiques aux bases de données.

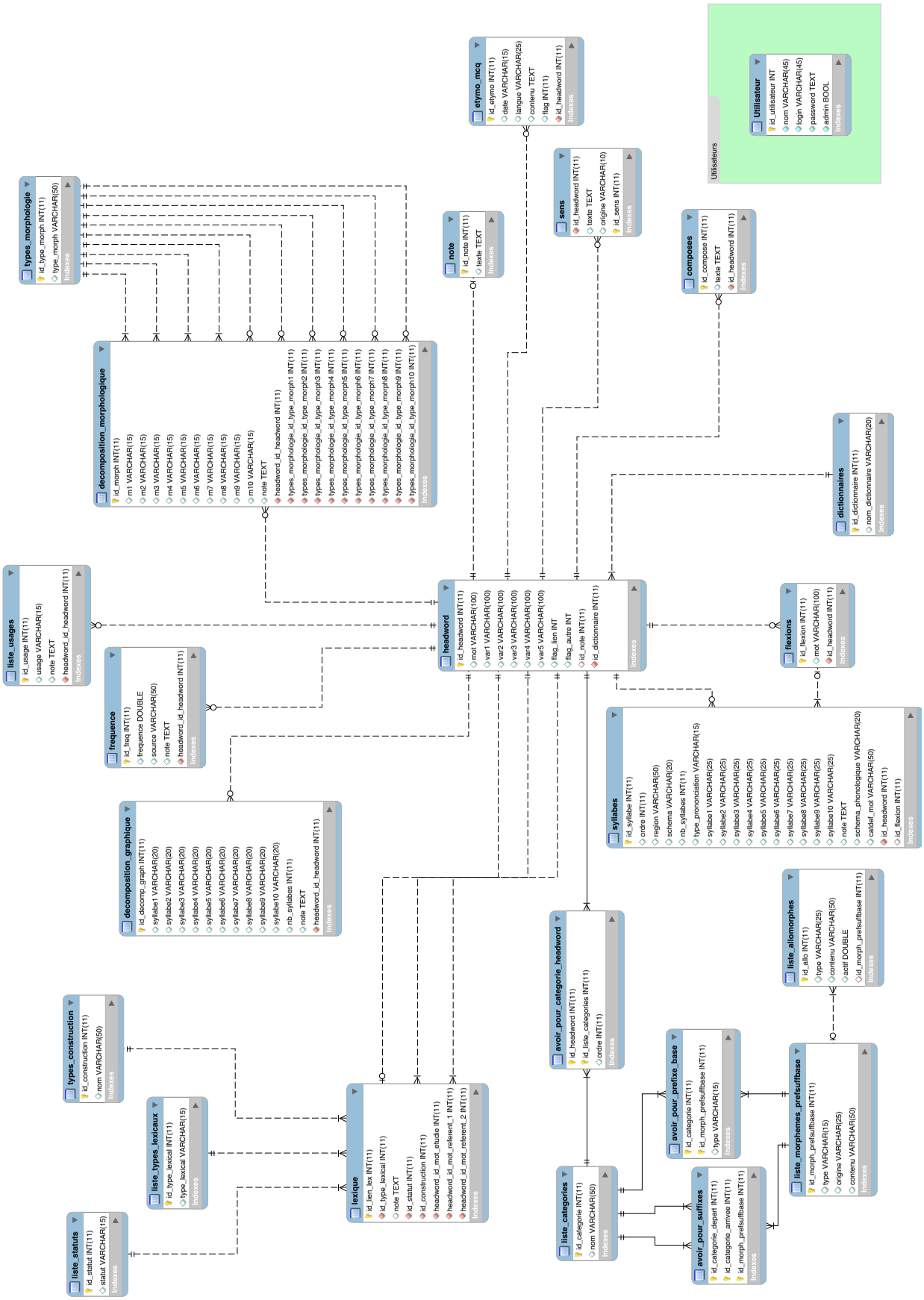
Un autre point important concerne le domaine d'application du projet que j'ai trouvé très intéressant.

"L'informaticien n'a pas de métier, il a le métier de ses clients"

Il est vrai que pour un informaticien, le vaste domaine de la linguistique n'est pas compréhensible au premier abord. Le vocabulaire très précis ne facilite pas une maîtrise rapide des besoins. Il a donc fallu organiser plusieurs réunions afin de bien comprendre la problématique de base avant de se lancer dans les spécificités de l'application, touchant alors à des notions de linguistiques plus complexes.

Mon projet de fin d'études s'est bien déroulé dans l'ensemble. Merci à tous ceux qui sont intervenus et qui ont pris le temps de collaborer, de répondre à mes interrogations avec patience, de guider mes choix. Cela m'a permis de faire avancer ce projet de façon pérenne et sur de bonnes bases. Le planning prévisionnel n'ayant pas pu être respecté pour les raisons évoquées précédemment, la documentation établie permettra certainement au prochain développeur de prendre en main rapidement l'existant en vue de mener le projet actuel vers son terme.

Annexes


Utilisateurs
Utilisateur
id_utilisateur INT
nom VARCHAR(45)
login VARCHAR(45)
password TEXT
admin BOOL
etymo_mcq
id_etymo INT(11)
date VARCHAR(15)
langue VARCHAR(25)
contenu TEXT
flag INT(11)
id_headword INT(11)
types_morphologie
id_type_morph INT(11)
type_morph VARCHAR(50)
note
id_note INT(11)
texte TEXT
sens
id_headword INT(11)
texte TEXT
origine VARCHAR(10)
id_sens INT(11)
composes
id_compose INT(11)
texte TEXT
id_headword INT(11)
decomposition_morphologique
id_morph INT(11)
m1 VARCHAR(15)
m2 VARCHAR(15)
m3 VARCHAR(15)
m4 VARCHAR(15)
m5 VARCHAR(15)
m6 VARCHAR(15)
m7 VARCHAR(15)
m8 VARCHAR(15)
m9 VARCHAR(15)
m10 VARCHAR(15)
note TEXT
headword_id_headword INT(11)
types_morphologie_id_type_morph1 INT(11)
types_morphologie_id_type_morph2 INT(11)
types_morphologie_id_type_morph3 INT(11)
types_morphologie_id_type_morph4 INT(11)
types_morphologie_id_type_morph5 INT(11)
types_morphologie_id_type_morph6 INT(11)
types_morphologie_id_type_morph7 INT(11)
types_morphologie_id_type_morph8 INT(11)
types_morphologie_id_type_morph9 INT(11)
types_morphologie_id_type_morph10 INT(11)
dictionnaires
id_dictionnaire INT(11)
nom_dictionnaire VARCHAR(20)
liste_usages
id_usage INT(11)
usage VARCHAR(15)
note TEXT
headword_id_headword INT(11)
headword
id_headword INT(11)
mot VARCHAR(100)
var1 VARCHAR(100)
var2 VARCHAR(100)
var3 VARCHAR(100)
var4 VARCHAR(100)
var5 VARCHAR(100)
flag_lien INT
flag_autre INT
id_note INT(11)
id_dictionnaire INT(11)
flexions
id_flexion INT(11)
mot VARCHAR(100)
id_headword INT(11)
frequence
id_freq INT(11)
frequence DOUBLE
source VARCHAR(50)
note TEXT
headword_id_headword INT(11)
decomposition_graphique
id_decomp_graph INT(11)
syllabe1 VARCHAR(20)
syllabe2 VARCHAR(20)
syllabe3 VARCHAR(20)
syllabe4 VARCHAR(20)
syllabe5 VARCHAR(20)
syllabe6 VARCHAR(20)
syllabe7 VARCHAR(20)
syllabe8 VARCHAR(20)
syllabe9 VARCHAR(20)
syllabe10 VARCHAR(20)
nb_syllabes INT(11)
note TEXT
headword_id_headword INT(11)
syllabes
id_syllabe INT(11)
ordre INT(11)
region VARCHAR(50)
schema VARCHAR(20)
nb_syllabes INT(11)
type_prononciation VARCHAR(15)
syllabe1 VARCHAR(25)
syllabe2 VARCHAR(25)
syllabe3 VARCHAR(25)
syllabe4 VARCHAR(25)
syllabe5 VARCHAR(25)
syllabe6 VARCHAR(25)
syllabe7 VARCHAR(25)
syllabe8 VARCHAR(25)
syllabe9 VARCHAR(25)
syllabe10 VARCHAR(25)
note TEXT
schema_phonologique VARCHAR(20)
cattel_mot VARCHAR(50)
id_headword INT(11)
id_flexion INT(11)
liste_allomorphes
id_allo INT(11)
type VARCHAR(25)
contenu VARCHAR(50)
actif DOUBLE
id_morph_prefsuffbase INT(11)
avoir_pour_categorie_headword
id_headword INT(11)
id_liste_categories INT(11)
ordre INT(11)
avoir_pour_prefixe_base
id_categorie INT(11)
id_morph_prefsuffbase INT(11)
type VARCHAR(15)
liste_morphemes_prefsuffbase
id_morph_prefsuffbase INT(11)
type VARCHAR(15)
origine VARCHAR(25)
contenu VARCHAR(50)
types_construction
id_construction INT(11)
nom VARCHAR(50)
liste_types_lexicaux
id_type_lexical INT(11)
type_lexical VARCHAR(15)
lexique
id_lien_lex INT(11)
id_type_lexical INT(11)
note TEXT
id_statut INT(11)
id_construction INT(11)
headword_id_mot_etude INT(11)
headword_id_mot_referent_1 INT(11)
headword_id_mot_referent_2 INT(11)
liste_categories
id_categorie INT(11)
nom VARCHAR(50)
avoir_pour_suffixes
id_categorie_depart INT(11)
id_categorie_arrivee INT(11)
id_morph_prefsuffbase INT(11)
liste_statuts
id_statut INT(11)
statut VARCHAR(15)

Glossary

D.A.O. De son vrai nom Data Access Object, ce patron de conception (ou pattern) permet de séparer la couche d'accès aux données de la couche logique applicative. Son utilisation permet de s'abstraire de la façon dont les données sont stockées au niveau des objets métier.. 33

Hibernate Framework open source Java qui simplifie la persistance des objets en base de données relationnelle. Son utilisation remplace la mise en place de JDBC et propose entre autres le cache des objets, les sessions et les transactions.. 22, 32, 33

L.L.L. Laboratoire Ligérien de Linguistique. 7

Références

[1] Emmanuel Puybaret, **Java SE 5, AWT/Swing, Java 3D, Java Web Start, SWT/JFace** *Exclusivité ebook*, EYROLLES, 2nd Edition, 2008.

[2] Anthony Patricio, **Hibernate 3.0** *Gestion optimale de la persistance dans les applications Java/J2EE*, EYROLLES, 2011.

Index

Réalisation d'une application pour le Laboratoire Ligérien de Linguistique L.L.L.

Département Informatique
5e année
2012 - 2013

Projet de fin d'études

Résumé : Ce rapport présente le projet de fin d'études que j'ai réalisé au cours de ma 5ème année au département informatique de Polytech'TOURS. Ce projet porte sur la reprise et la refonte d'une application existante ainsi que de sa base de données, application utilisée par le Laboratoire Ligérien Linguistique de l'université de Tours.

Mots clefs : P.F.E., L.L.L., Projet, Polytech'TOURS, Java, Hibernate, MySQL, Gestion de projet

Abstract: This report presents the project that I realized during my fifth year at Polytech'Tours. This project deals with the recovery and renewal of an existing application and its database, application used by the Laboratoire Ligérien Linguistique of the University of Tours.

Keywords: P.F.E., L.L.L., Projet, Polytech'TOURS, Java, Hibernate, MySQL, Project management

---

Encadrant
Claudine TACQUARD
claudine.tacquard@univ-tours.fr

Étudiant
Valentin DOULCIER
valentin.doulcier@etu.univ-tours.fr

Clients
Jean-Michel FOURNIER
jean-michel.fournier@univ-tours.fr
Marjolaine MARTIN
marjolaine.martin@univ-tours.fr

DI5 2012 - 2013