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

Cahier de spécification système & plan de développement			
Projet :	**Réalisation d'une application pour le Laboratoire Ligérien de Linguistique L.L.L. - TOURS**		
Emetteur :	Valentin DOULCIER	**E-mail :**	valentin.doulcier@gmail.com
Date d'émission :	7 janvier 2013		

Validation			
Nom	Date	Validé (O/N)	Commentaires
C. TACQUARD	*04/01/2013*	*OUI*	– –

Historique des modifications		
Version	Date	Description de la modification
1.0	12/2012	Version initiale : Définition des spécifications
2.0	01/2013	Version finale : Ajout des fonctionnalités

Table des matières

Cahier de spécification système

1.1 Introduction

Ce document présente les spécifications système du projet de fin d'études « 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 (L.L.L.) de l'université de Tours, que nous nommerons L.L.L. dans la suite de ce document.

Le L.L.L. est un laboratoire qui 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 environ 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)

1.2 Contexte de la réalisation

1.2.1 Contexte

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 la 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).

Aujourd'hui, la base de donnée est optimisée, simplement aucune application n'a été développée. Elle n'est donc pas utilisée à l'heure actuelle.

1.2.2 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.

1.2.3 Hypothèses

Certaines hypothèses peuvent être émises, décrivant les facteurs susceptibles de remettre en cause tout ou une partie de la réalisation des spécifications, ainsi que d'éventuelles solutions de repli.

- Si au cours du projet de nouvelles fonctionnalités à implémenter sont demandées par le client et qu'il n'est pas possible (dû à une contrainte de temps imposée par le calendrier prévisionnel) de les implémenter, alors il faudra redéfinir les priorités de développement.

- Si au cours du projet des fonctionnalités ne sont pas implémentables avec les contraintes de développement décrites dans ce présent document, on s'autorise à utiliser d'autres librairies et/ou langages qui s'adapteraient mieux.

- Si au cours du projet on se rend compte que les tables de la bases de données doivent être modifiées (pour l'ajout de fonctionnalités par exemple), il faudra modifier la base précautionneusement en vue de ne pas altérer le développement déjà fait.

- Si au cours du projet on voit apparaitre des retards sur le planning prévisionnel, il faudra réviser la priorité des tâches à effectuer.

1.2.4 Bases méthodologiques

Pour mener à bien ce projet, plusieurs procédures seront respectées, principalement la convention de nommage. Cette convention est présente en annexe de ce document. 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. Cela implique donc une réflexion approfondie en vue de déterminer ces nouveaux choix technologiques. Toutefois, la partie base de données ayant été refaite l'année dernière, on conservera le choix du langage ayant été fait (MySQL).

1.3 Description générale

1.3.1 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 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.

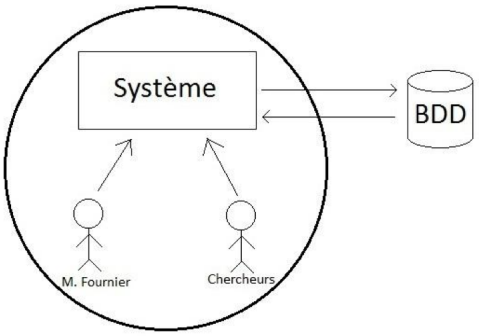

FIGURE 1.1 – Environnement du projet

1.3.2 Caractéristiques des utilisateurs

Concernant cette application, les utilisateurs finaux seront les chercheurs du L.L.L. ainsi que Jean-Michel FOURNIER. Elle ne sera donc utilisée qu'en interne.

Les connaissances requises en informatique sont quasi nulles. En effet, le résultat sera un logiciel dont l'ergonomie aura été validée, et qui sera aussi intuitif que n'importe quel logiciel que l'on utilise quotidiennement. Les interfaces seront quelque peu différentes en vue de s'adapter aux nouvelles contraintes, simplement un manuel les documentera et l'adaptation se fera en douceur.

L'application sera déployée sur les postes des chercheurs (avec l'approbation de Jean-Michel FOURNIER). Aucun compte spécial n'ayant besoin d'être créé, chaque utilisateur pourra l'utiliser, aussi bien de façon régulière que occasionnelle.

Toutefois, les utilisateurs n'auront pas tous les mêmes droits d'accès. Certains menus seront réservés à l'administration. En effet, Jean-Michel FOURNIER souhaite valider sur la base principale chacune des modifications qui seront proposées par les chercheurs. Il disposera donc d'un module particulier de supervision et de modification de la base maitre. Chaque utilisateur sera par contre maître de ses propres modifications sur sa base de données en local (sur sa machine respective).

1.3.3 Fonctionnalités et structure générale du système

Les principales fonctions utilisateurs du système seront la consultation de la base de données, la modification ou l'ajout d'informations sur un mot (consultation et édition des tables).

Le système est seulement constitué de la base de données, et des interfaces (fenêtres du logiciel) permettant d'y accéder.

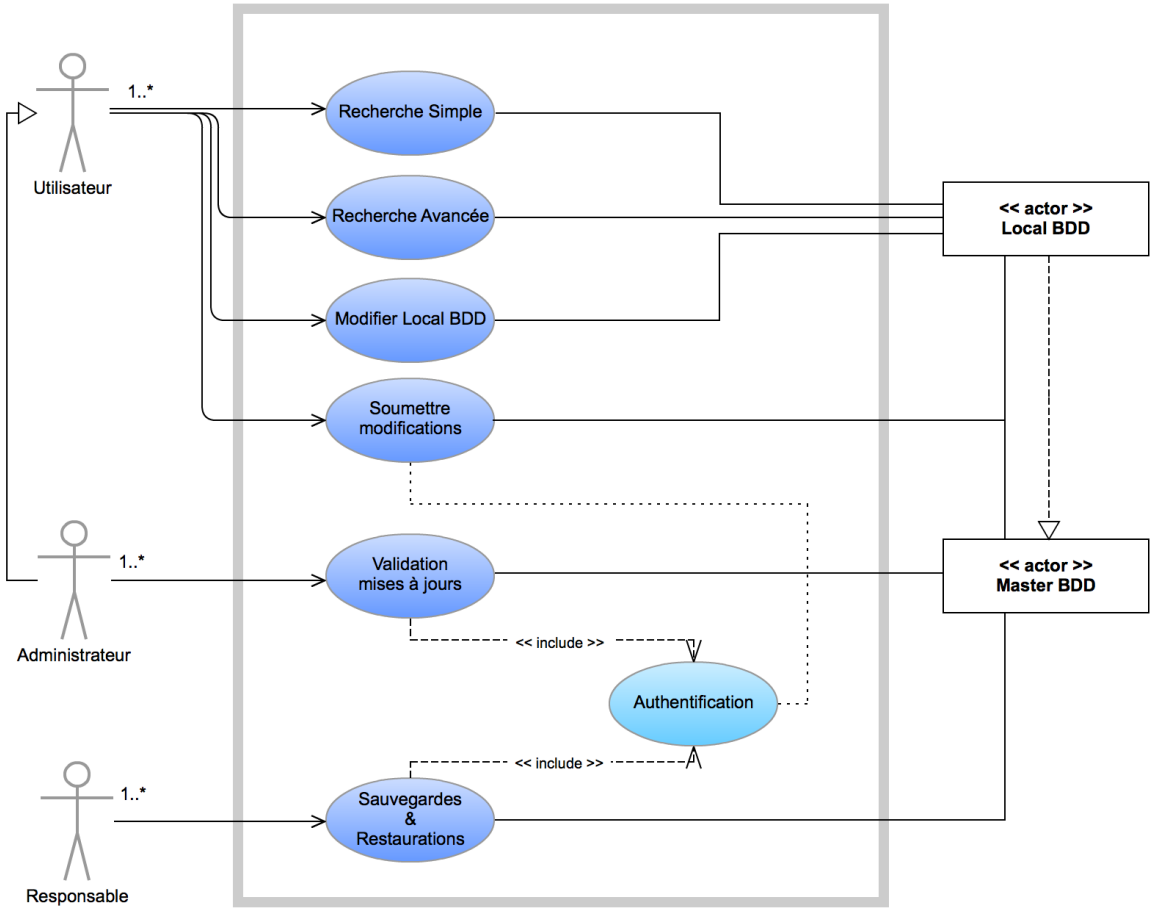

FIGURE 1.2 – Diagramme de cas d'utilisation

1.3.4 Contraintes de développement, d'exploitation et de maintenance

Contraintes de développement

Ce projet reprend un projet déjà existant. Toutefois, ne gardant que la partie base de données, il convient de ré-analyser le problème posé et de choisir des solutions adaptées. A ce titre, les contraintes propres au développement de ce projet sont les suivantes :

- Les langages à utiliser pour le développement sont principalement le java et le SQL (lié au framework Hibernate et DAO).
- La solution développée doit fonctionner sur Windows ainsi que sur Mac OS X.
- L'ensemble doit fonctionner indépendamment d'une quelconque connexion (internet ...).
- La convention de nommage se doit d'être respectés,
- L'environnement sera plus simple que la solution précédente. En effet, MySQL Server ainsi que java seront les 2 seuls outils à installer (voir manuel d'utilisation).
- La date de livraison prévue est indiquée dans les consignes données en début de projet (voir présentation).

Contraintes d'exploitation

Les contraintes par rapport à l'exploitation du projet sont les suivantes :

* Une équipe interviendra pour la mise en production d'une base de données gérée de façon externalisée. En attendant, les tests se feront sur une base en locale. L'avancement de cette tâche est indépendante de l'application, la bascule se fera quand la base externe aura été portée en production.
* Concernant la gestion de projet, aucune méthode organisationnelle particulière ne sera mise en place. En effet, les périodes de travailles sont disposées de telle façon qu'il est très délicat de suivre une de ces méthodes.
* Pour ce qui est de la méthode de gestion de projet, la démarche I.T.I.L. (Information Technology Infrastructure Library) plaçant l'utilisateur final au centre des préoccupations liées au développement (démarche qualité de service) sera utilisée. Chaque fonctionnalité sera ainsi développée selon la roue de Deming (plan / do / check / act).

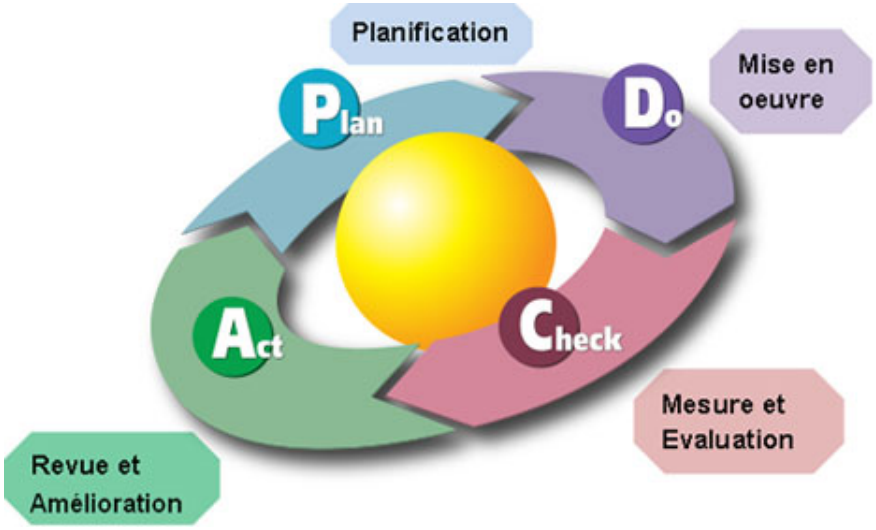

FIGURE 1.3 – Roue de Deming - Démarche Qualité - I.T.I.L.

Maintenance et évolution du système

Les contraintes liées aux procédures de maintenance sont les mêmes que celles liées au développement. Toutefois, un manuel utilisateur sera réalisé, et le code sera commenté, ce qui permettra à l'application d'être reprise / mise à jour sans difficultés.

1.4 Description des interfaces externes du logiciel

1.4.1 Interfaces matériel/logiciel

L'interface matériel/logiciel n'est ici pas très conséquente. Un simple ordinateur avec la configuration nécessaire installée (java, MySQL Server) ainsi que la base de données installée en local permet une utilisation complète de l'application.

La base de données maître quant à elle, sera hébergée dans une structure adaptée, non définie à ce jour (intervention équipe externe). Elle sera accessible depuis n'importe quel poste relié à internet pour permettre les mises à jour, mais seul M. FOURNIER pourra modifier cette base.

1.4.2 Interfaces homme/machine

L'interface homme/machine est le cœur de mon projet de fin d'études. Voici ci-dessous la maquette de la fenêtre principale. L'interface a été repensée de telle sorte que l'application soit le plus simple et le plus ergonomique possible. Le nombre de fenêtre a donc été considérablement réduit, facilitant l'utilisation.

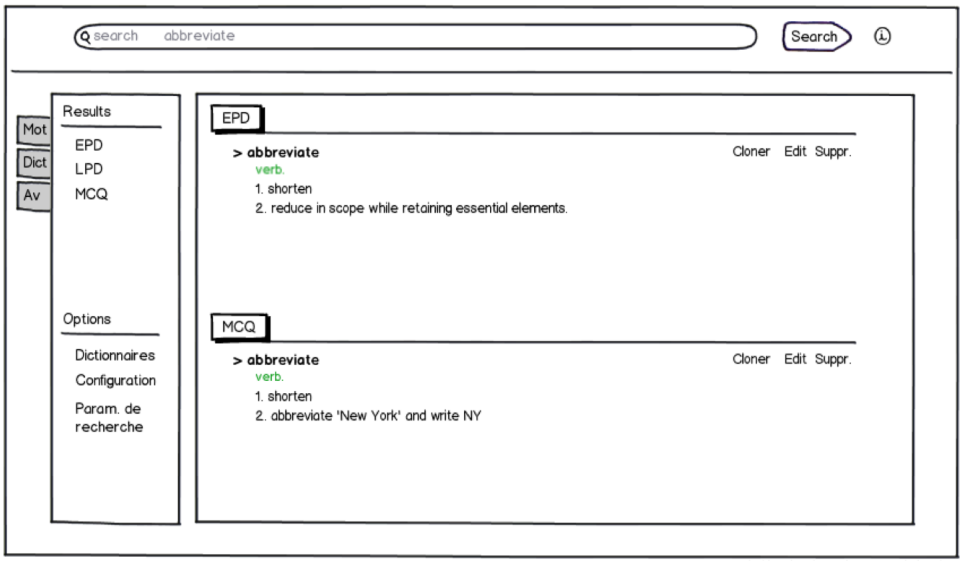

FIGURE 1.4 – IHM - Interface de Recherche

La fenêtre principale contient un champ de saisie qui permettra d'effectuer la recherche de mots. Cette fonctionnalité élémentaire sera au centre de l'application. Il sera également possible d'affiner sa recherche à l'aide d'une reconnaissance de mots pour les correspondances (exacte, contient, commence par, termine par, différent de, etc.).

Le panneau de gauche pourra être considéré comme le menu de l'application. Il contiendra principalement 3 onglets, permettant de spécifier le type de recherche que l'on veut effectuer (mot, liens, avancée). Selon le type de recherche, les options diffèreront. Les options seront présentées dans le panneau en lui même. Bien entendu, ces options agiront directement sur les résultats de recherche.

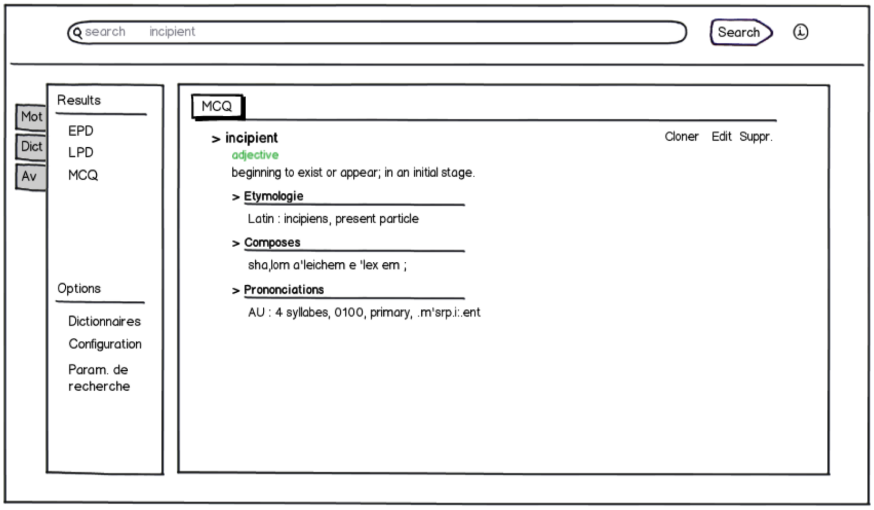

FIGURE 1.5 – IHM - Interface de Résultats

Ce panneau des résultats (carré principal) contient donc la liste des mots résultants de la recherche effectuée. Le niveau de détail de affiché pour le résultat sera paramétrable dans les options. Cette nouvelle organisation permet d'alléger l'interface et reste évolutive (ajout d'un nouveau dictionnaire). L'édition d'un mot se fera en cliquant sur ce mot, et un écran contenant toutes les informations apparaîtra.

Le choix des dictionnaires dans lesquels on souhaite effectuer une recherche se fera par le menu « dictionnaire » dans les options de gauche. En cochant les cases devant les dictionnaires présents dans la base, on choisit de chercher ou non l'information dans ceux-ci.

En plus des dictionnaires déjà présents dans la base de données, on pourra via cet écran rajouter un dictionnaire très simplement. Aussi, bon nombre d'options seront proposées (type du fichier XML du dictionnaire, balises pour les prononciations, les entrées, les mots, etc.). Ci-dessous, la maquette de cet écran :

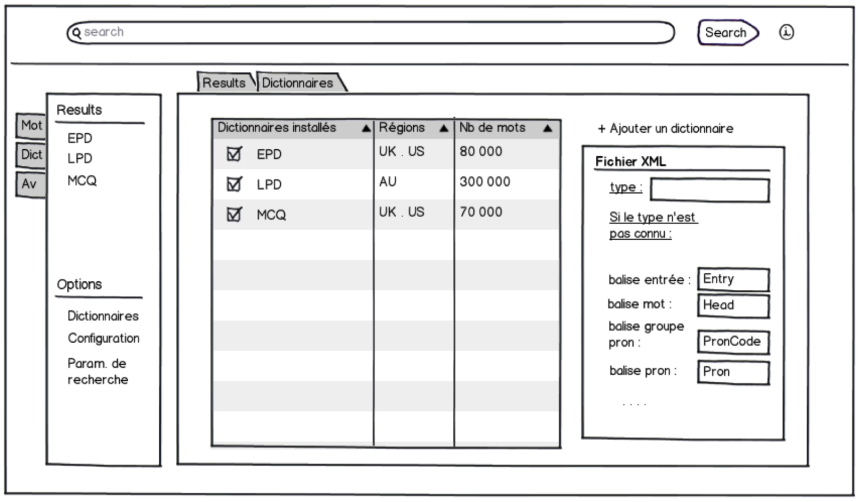

FIGURE 1.6 – IHM - Choix des dictionnaires

La sélection de l'onglet "Recherche Avancée" fera apparaître un onglet "paramètres de recherche". Il sera possible d'affiner la recherche selon des critères très spécifiques.

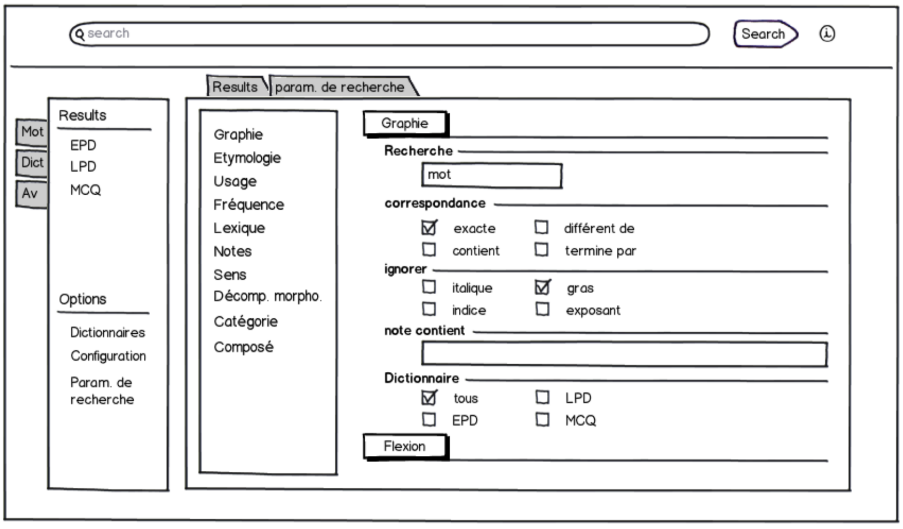

FIGURE 1.7 – IHM - Interface de Recherche Avancée

1.4.3 Interfaces logiciel/logiciel

Pour accéder aux bases de données, l'application utilisera MySQL Server. L'ancienne application était basée sur XAMP, toutefois les nouveaux choix d'architecture permettent de grandement simplifier les outils à utiliser.

Chaque utilisateur disposera d'une base de données locale sur sa propre machine. Ainsi, l'application fera des requêtes sur cette base de données. Toutefois, lorsqu'un utilisateur le souhaitera, il pourra (en étant relié à internet) récupérer les éventuelles mises à jour, et proposer les siennes (soumises à l'approbation de l'utilisateur administrateur - Jean-Michel FOURNIER).

La mise à jour se fera par insertion dans la base de données, de données temporaires (en attente de validation). Ce processus permets d'éviter l'envoie de fichier XML par mail (risque de corruption du fichier, fichier vérolé, mauvaise adresse ...).

1.5 Architecture générale du système

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 MySQL Server. 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).

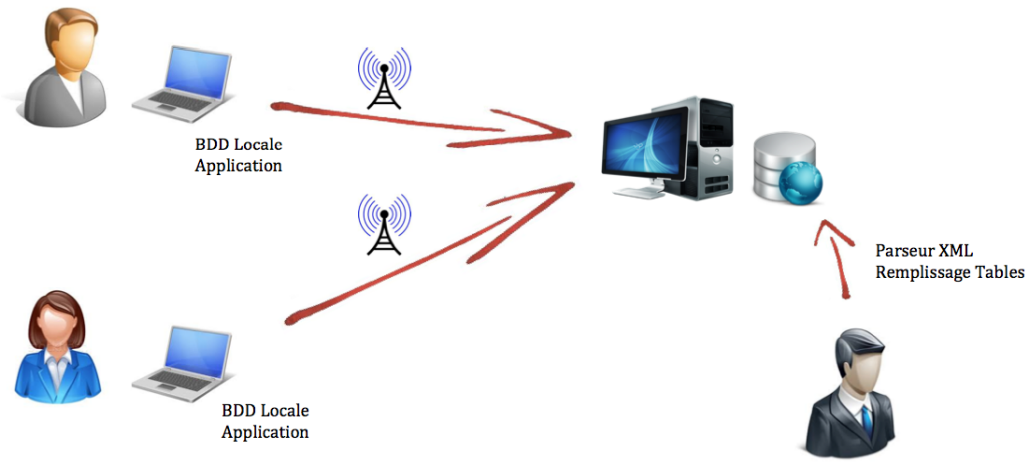

FIGURE 1.8 – Description de l'architecture

1.6 Description des fonctionnalités

Il s'agit de l'expression des besoins fonctionnels. Cette partie a donc comme objectif de décrire l'ensemble des fonctionnalités du système en précisant avec quels composants de la partie 1.5 elles interagissent.

1.6.1 Fonction de recherche

Identification de la fonction

Présenter la fonction :
- Nom : Recherche de mots dans la table.
- Rôle : Le but de cette fonction est de gérer la recherche des mots dans la base de données.
- Priorité : Primordiale.

Description de la fonction

Cette fonctionnalité répond à l'un des besoins les plus "urgents". Il s'agit d'éditer un mot, et de présenter toutes ses caractéristiques (décompositions, ...). Il faudra cependant réfléchir à un moyen pour ne pas permettre la modification lorsqu'une simple consultation est demandée (prévenir des modifications non désirées).

1.6.2 Fonctions relatives aux liens

Identification de la fonction

Présenter la fonction :
- Nom : Édition des liens valides.
- Rôle : Le but de cette fonction est de gérer les liens entre différents mots.
- Priorité : Normale.

Description de la fonction

Les liens sont le cœur de l'application. Il s'agit d'un regroupement de mots issus de dictionnaires différents. Les liens entre ces mots ne sont pas triviaux. A ce titre, les liens construits par défauts ne sont pas forcément justes, et seule une personne qualifiée peut traduire l'intégrité de l'information.

C'est la raison pour laquelle ces liens doivent pouvoir être construits manuellement et déconstruit intelligemment de telle sorte à ne pas perdre d'information. Bien entendu, une fonction d'édition basique des liens sera implémentée, permettant aussi bien la consultation "lecture seule" que la modification.

1.7 Conditions de fonctionnement

Il faut dans ce paragraphe décrire les dispositions qu'il est nécessaire de prendre en compte pour les différentes conditions de fonctionnement su systèmes.

1.7.1 Performances

Du point de vue de l'utilisateur : l'application doit être assez rapide quant à l'interrogation et aux mises à jour de la base de données.

Du point de vue de l'environnement : à chaque fois qu'un utilisateur souhaite appliquer une modification sur la base de données, cette dernière sera enregistrée dans une table temporaire (ou à l'aide d'un flag), qui sera soumise à validation par l'utilisateur maître.

1.7.2 Capacités

En ce qui concerne les limites de capacités de l'application, il n'y a pas de contrainte vis à vis du nombre maximum de terminaux, idem concernant le nombre maximum de points d'acquisition ou encore du nombre maximum de transactions simultanées.

En revanche, la taille maximum des données traitées par requête sera peut-être soumise à limitation de taille. Toutefois, ce paramètre se règle dans les fichiers de configuration de mySql.

Cette application fonctionnant en local, il n'y aura pas de mode reprise de secours. Toutefois, la base maitre étant hébergée sur un serveur, le mode dégradé sera géré par le service prestataire, et un export manuel des fichiers de modification pourra être prévu (voir avec le client).

1.7.3 Contrôlabilité

L'envoie de mise à jour ou l'échange avec le serveur hébergeant la base de données pourra donné lieu à des traces dans des fichiers de log (à voir avec le prestataire travaillant avec le L.L.L.).

1.7.4 Sécurité

Tous les utilisateurs du L.L.L. ayant accès à l'application auront sur leurs postes respectifs une duplication de la base de données principale. Cette base de données verra son accès restreint par authentification, idem pour la base maître hébergée par le prestataire. Aussi, ce prestataire aura surement des mesures de protections adaptées (anti-piratage).

1.7.5 Intégrité

La base principale étant hébergée par un service informatique, des mesures de protections (sauvegardes, restauration, ...) sont forcément mises en place. Toutefois, ces méthodes ne seront pas détaillées ici, je ne suis pas au courant des procédés qu'ils utilisent.

Plan de développement

2.1 Découpage du projet en tâches

2.1.1 Tâche 1 : Prise en main de l'existant

Description de la tâche

Cette tâche consiste en la compréhension de ce qui a déjà été fait. En effet, pour rappel, ce P.F.E. fait suite à 2 précédents P.F.E.. Un travail initial a donc été réalisé. Et s'agissant d'un projet de taille conséquente, cette tâche est donc inévitable.

Estimation de charge

L'estimation selon le diagramme de Gantt est de 20 Jours.

Contraintes temporelles

Il n'existe pas de contraintes temporelles fortes pour cette tâche.

2.1.2 Tâche 2 : Rédaction du cahier de spécifications

Description de la tâche

Cette tâche consiste en la rédaction du cahier de spécifications système.

Livrables

Le cahier de spécifications doit être livré en fin de tâche, à savoir pour le 7 janvier 2013. Il s'agit de ce présent document, qui décrit l'environnement du projet ainsi que les fonctionnalités à implémenter.

Estimation de charge

L'estimation selon le diagramme de Gantt est de 35 Jours.

Contraintes temporelles

La version finale du cahier de spécifications doit être déposé sur l'intranet avant le 7 janvier 2013.

2.1.3 Tâche 3 : Mise en place de l'environnement

Description de la tâche

La refonte de l'application impliquant un changement de technologies, la mise en place de l'environnement est une tâche initiale fondamentale pour le bon déroulement du développement. A savoir que cette mise en place de l'environnement devra également être faite sur chacun des postes utilisateurs.

Livrables

Un manuel d'installation sera rédigé en parallèle de cette tâche. En effet, chaque poste utilisateur doit être pourvu de l'environnement logiciel adéquat pour le bon fonctionnement de l'application.

Estimation de charge

L'estimation selon le diagramme de Gantt est de 12 Jours.

Contraintes temporelles

Il n'existe pas de contraintes temporelles fortes pour cette tâche.

2.1.4 Tâche 4 : Développement du squelette de l'application

Description de la tâche

Le squelette de l'application correspond à la base du logiciel. Il s'agit de la fenêtre "brute", qui va servir de base pour l'élaboration des fonctionnalités définies.

Estimation de charge

L'estimation selon le diagramme de Gantt est de 15 Jours.

Contraintes temporelles

Il n'existe pas de contraintes temporelles fortes pour cette tâche.

2.1.5 Tâche 5 : Module Recherche + Édition des mots

Description de la tâche

Cette fonctionnalité répond à l'un des besoins les plus "urgents". Il s'agit d'éditer un mot, et de présenter toutes ses caractéristiques (décompositions, ...). Il faudra cependant réfléchir à un moyen pour ne pas permettre la modification lorsqu'une simple consultation est demandée (prévenir des modifications non désirées).

Estimation de charge

L'estimation selon le diagramme de Gantt est de 45 Jours.

Contraintes temporelles

Il n'existe pas de contraintes temporelles fortes pour cette tâche.

2.1.6 Tâche 6 : Module Édition des liens valides

Description de la tâche

Les liens sont le cœur de l'application. Il s'agit d'un regroupement de mots issus de dictionnaires différents. Les liens entre ces mots ne sont pas triviaux. A ce titre, les liens construits par défauts ne sont pas forcément justes, et seule une personne qualifiée peut traduire l'intégrité de l'information.

C'est la raison pour laquelle ces liens doivent pouvoir être construits manuellement et déconstruit intelligemment de telle sorte à ne pas perdre d'information. Bien entendu, une fonction d'édition basique des liens sera implémentée, permettant aussi bien la consultation "lecture seule" que la modification.

Estimation de charge

L'estimation selon le diagramme de Gantt est de 22 Jours.

Contraintes temporelles

Il n'existe pas de contraintes temporelles fortes pour cette tâche.

2.2 Planning

	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

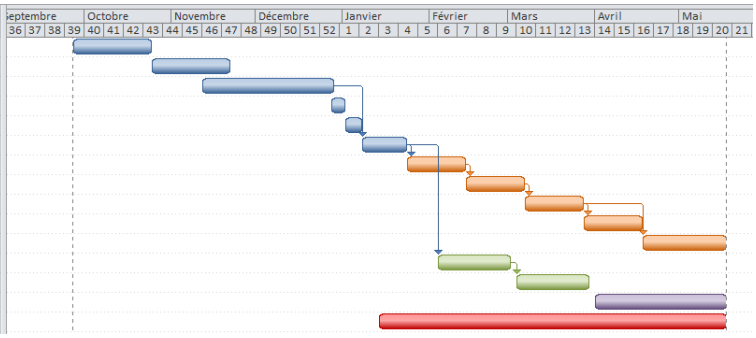

FIGURE 2.9 – Diagramme de Gantt

Annexes

Convention de programmation
Laboratoire Ligérien de Linguistique
P.F.E. – Développement d'une application

Le suivante convention sera suivie par
chacun d'entre nous jusqu'à la fin de ce projet.
Le code ***JAVA*** sera documenté en
utilisant des commentaires ***Javadoc***.

I - Attribution des noms et capitalisation

1 – Les packages

Le nom d'un package doit respecter les conventions suivantes. Toute lettre doit être en minuscule et on ne peut utiliser que les caractères [a-z], [0-9] et le point '.'. Les caractères '-', '_', espace, ou autres caractères ($, *,...) ne doivent pas être utilisés.

Tout package doit avoir comme premier élément un des mots suivants : com, gov, mil, net, org ou les deux lettre identifiants un pays.

Exemple: **fr.atos.java**

2 – Les classes

La première lettre d'une classe doit être en majuscule. Si la classe est composée de plusieurs mots, chaque mot doit être attaché et avoir sa première lettre en majuscule également. Les noms doivent être simple et descriptifs (il faut éviter les acronymes).

De plus, on ne pourra utiliser que les caractères [a-z], [0-9] et le point '.'. Les caractères '-', '_', espace, ou autres caractères ($, *,...) ne doivent pas être utilisés.

Exemple: **class MainWindows**

3 – Les interfaces

Pour les interfaces, nous adopterons les mêmes conventions que pour les classes. La première lettre doit être un I majuscule. On ne peut utiliser que les caractères [a-z], [0-9] et le point '.'. Les caractères '-', '_', espace, ou autres caractères ($, *,...) ne doivent pas être utilisés.

Exemple: **interface IUser**

4 – Constantes

Les constantes seront écrites entièrement en majuscule et si il y a plusieurs mots nous les relierons avec un '_'.

Exemple: **TAILLE_MAXIMUM_TABLEAU**

5 – Variables

Les variables doivent commencer par une minuscule. Si la variable est composée de plusieurs mots, chaque mot doit être attaché et avoir sa première lettre en majuscule. Les noms doivent être simple et descriptifs (il faut éviter les acronymes).

Exemple: **int nombreTaches**

6 – Méthodes

Tout comme les variables, les méthodes commencent par une minuscule. De même, si le nom de la méthode est composé de plusieurs mots, chaque mot doit être attaché et avoir sa première lettre en majuscule. Le premier mot est généralement un verbe.

Exemple: **trouverChemin()**

II – Les commentaires

1 – Utilisation

Les commentaires sont ici utilisés pour expliquer une partie de code en vue de sa maintenance, mais aussi pour décrire les subtilités qui peuvent être présentes à certains endroits (méthode compliqué à utiliser, comprendre,...)

2 - Implémentation

Les commentaires peuvent être implémentés de plusieurs fasons. En effet il existe des blocs de commentaires, souvent utilisés lorsque l'on a besoin d'écrire plusieurs lignes (pour expliquer l'utilité d'un fichier par exemple), et aussi des lignes de commentaires seules.

Bloc : /*

* Un commentaire

* sur deux lignes

*/

Ligne : // un commentaire sur une seule ligne

ou

/* un commentaire sur une seule ligne */

Les commentaires pourront être placés soit à côté d'une ligne à commenter, soit au dessus de cette ligne, mais pas en dessous :

/*commentaire de l'action A*/

[Action A]

[Action B] // Commentaire de l'action B

III – Les classes et Méthodes

1 - Déclaration

Les déclarations des classes et méthodes se font dans un ordre logique. On les classera en mettant par exemple les getters et setters ensemble, ou encore en plaçant les fonctions de moindre importance en dernier.

Exemple : classe maClasse{

```
private String maChaine;

private concatChaines(String, String);
private String getMaChaine();
private void setMaChaine(String);
}
```

2 – Méthodes d'accès aux attributs

Les attributs étant toujours déclarés **private** il nous faudra créer des accésseurs. Ceux-ci seront nommés à l'aide des préfixes "get" et "set", réspectivement pour la lecture et l'écriture. Une exception sera faite pour la lecture d'un attribut de type **boolean** : le préfixe sera "is".

Exemples: **public boolean isActif()**

public int getIdPersonne()

public void setIdPersonne(int unIdPersonne)

IV – Conventions de portabilité

Pour une raison de portabilité, nous veillerons à ne pas mettre de chemins en dur dans le code.

Glossary

DAO 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.. 8

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.. 8

L.L.L. Laboratoire Ligérien de Linguistique. 5

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