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

Département Informatique
5e année
2010 - 2011

Rapport de Projet de Fin d'Etudes

Réalisation d'une base de données de dictionnaires linguistiques

Encadrants
Claudine TACQUARD
claudine.tacquard@univ-tours.fr
Jean-Michel FOURNIER
jean-michel.fournier@univ-tours.fr

Université François-Rabelais, Tours

Étudiants
BACCONNET Elodie
elodie.bacconnet@etu.univ-tours.fr

DI5 2010 - 2011

Table des matières

Table des figures

Liste des tableaux

Remerciements

Tout d'abord je tiens à remercier les personnes qui m'ont permis de réaliser ce PFE, de le comprendre et qui m'ont aidée à me sortir de situations où je n'avais pas assez de recul pour pouvoir le résoudre : Mme TACQUARD, qui, en tant qu'encadrant, a été présente, disponible, et qui a su répondre à mes questions, notamment sur la partie réorganisation de la base de données de mon projet. Je tiens également à remercier M. FOURNIER, qui, malgré un emploi du temps chargé, a eu la sympathie de libérer du temps pour que nous puissions nous rencontrer et ainsi faire le point sur le projet, répondre à mes interrogations, et m'expliquer le contexte de l'application.

Introduction

2.1 Objectifs du projet de fin d'études

Durant notre cinquième année, nous devons réaliser un projet de fin d'études (PFE), nous permettant de vraiment avoir à gérer un projet seul, de l'analyser, faire les études nécessaires, ainsi que réaliser le développement associé. Cela nous permet d'appréhender la gestion de projet comme si nous étions dans le monde du travail, et cela nous sensibilise aux attentes de clients, avec des délais, des demandes précises, tout comme les attentes qu'un professionnel peut avoir à combler.
Le but pédagogique de ce PFE est également de nous confronter à la gestion de délais, de planning, et surtout de respect de ce planning, et le cas échéant de comprendre et justifier pourquoi les délais n'ont pas pu être respectés.

Pour pouvoir réaliser nos PFE, deux jours par semaine y sont consacrés dans notre emploi du temps. Ces deux jours nous servent à bien sûr travailler nos projets, mais également à rencontrer les clients, faire le point avec nos encadrants. Mais ces deux jours nous permettent surtout de travailler dans des vraies conditions professionnelles, dans le sens où nous avons la des journées entières à consacrer à nos projets, et non pas de simples créneaux dans une journée. C'est à nous de gérer notre journée, nos éventuelles rencontres avec le client, l'encadrant, nos phases de documentation, de codage, et de test.

2.2 Présentation du projet

Le projet que j'ai du gérer porte sur la réalisation d'une base de données de dictionnaires linguistiques pour le Laboratoire Ligérien Linguistique de l'université de Tours, que nous nommerons LLL dans la suite de ce rapport.

Actuellement, le LLL utilise une application lui permettant d'étudier la construction des mots anglais, et de chercher une corrélation entre certains mots, des points de construction similaires. Cependant cette application n'est pas totalement finie, elle n'est pas totalement intuitive pour les chercheurs du LLL. Elle ne permet pas d'étudier certains aspects de la phonétique des mots par exemple, ou certaines erreurs subsistent, ce qui fait que les chercheurs doivent continuer à faire des vérifications à la main, voire même de corriger certains éléments à la main, ce qui n'est évidemment pas le but de l'application.

Mon travail a donc consisté à améliorer cette application, en restructurant la base de données, mais également en corrigeant certains bugs de l'application, et en répondant à certaines attentes supplémentaires de M. FOURNIER.

Voici quelques captures d'écran de l'application actuelle :

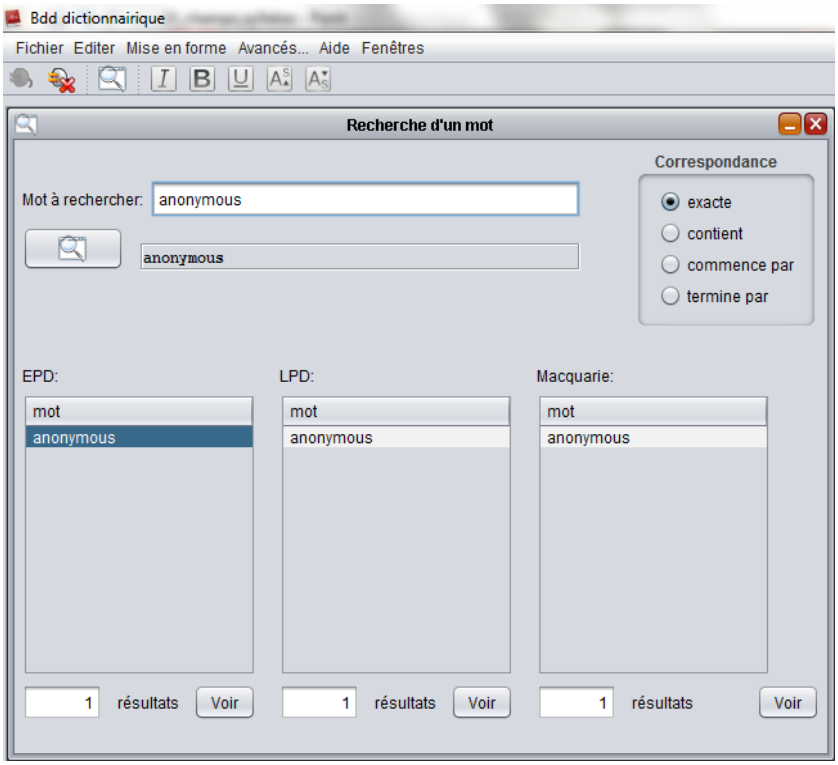


FIGURE 2.1 – Fenetre de recherche d'un mot

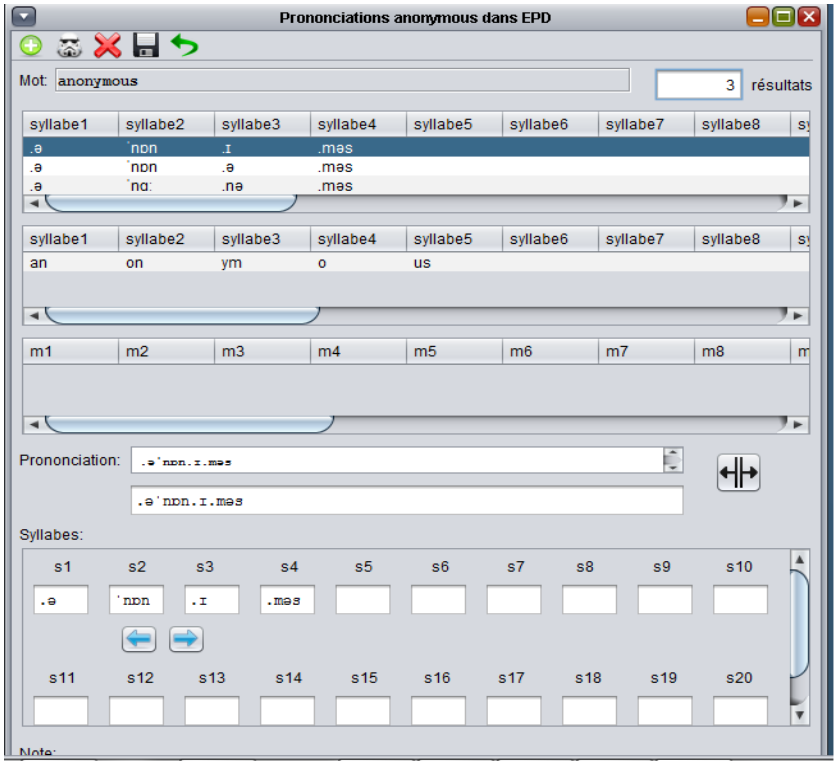


FIGURE 2.2 – Fenetre présentant le mot et sa construction

Avec cette application, on peut également effectuer des recherches avancées :

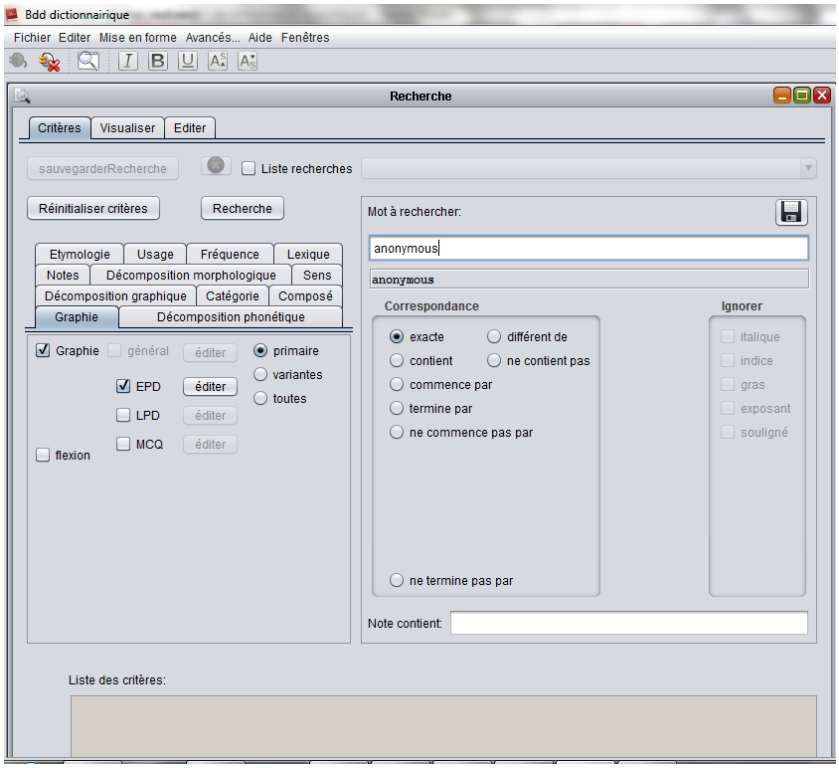


FIGURE 2.3 – Fenetre de recherche avancée

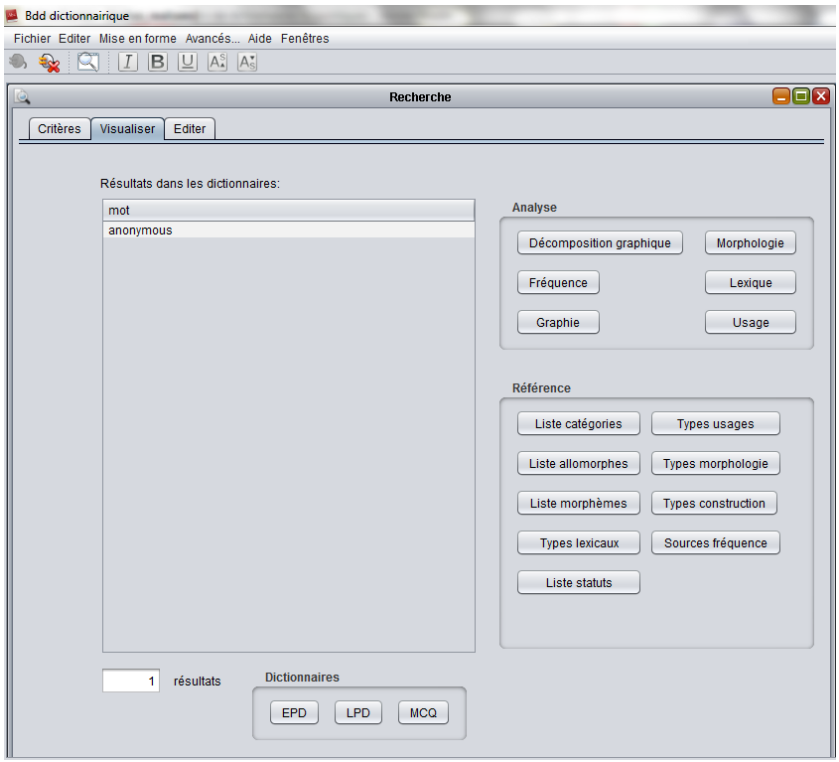


FIGURE 2.4 – Fenetre du résultat de la recherche avancée

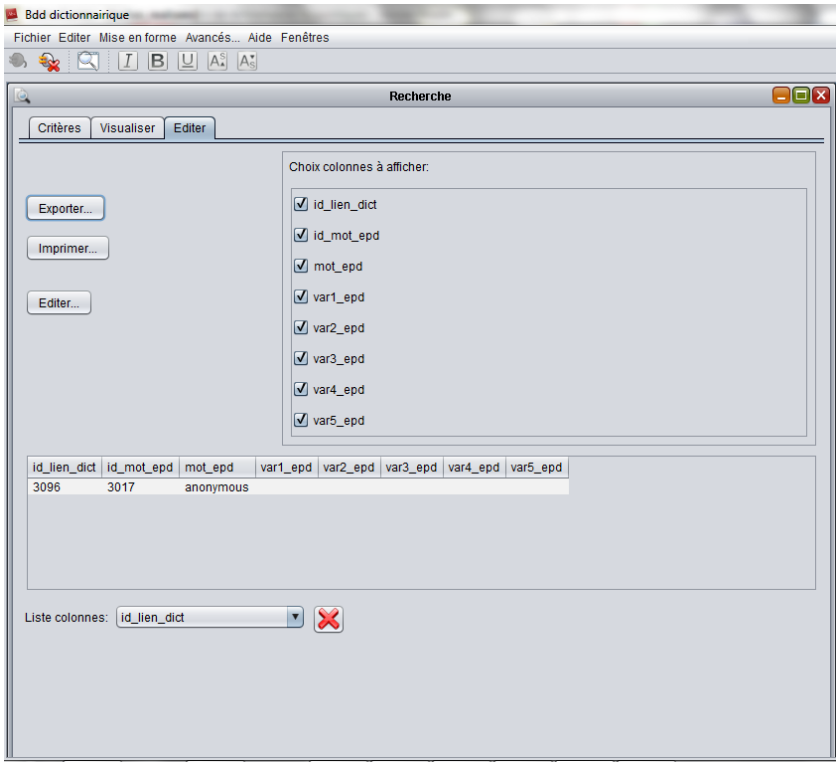


FIGURE 2.5 – Fenetre d'édition du résultat de la recherche avancée

Comme vous pouvez le voir sur cette dernière capture d'écran, il est possible d'exporter ou d'imprimer le résultat de la recherche avancée. Les colonnes que l'on veut éditer sont également sélectionnables, ce qui permet de choisir ce qui nous intéresse dans les différents critères du mot recherché.

Planning et gestion du projet

3.1 Points essentiels du projet

3.1.1 Planning prévisionnel

Le planning prévisionnel a été établi en début de projet, afin de pouvoir fixer des délais aux tâches à réaliser, mais également pour pouvoir organiser tout le projet et le déroulement de sa conception.

On peut voir ici le planning :

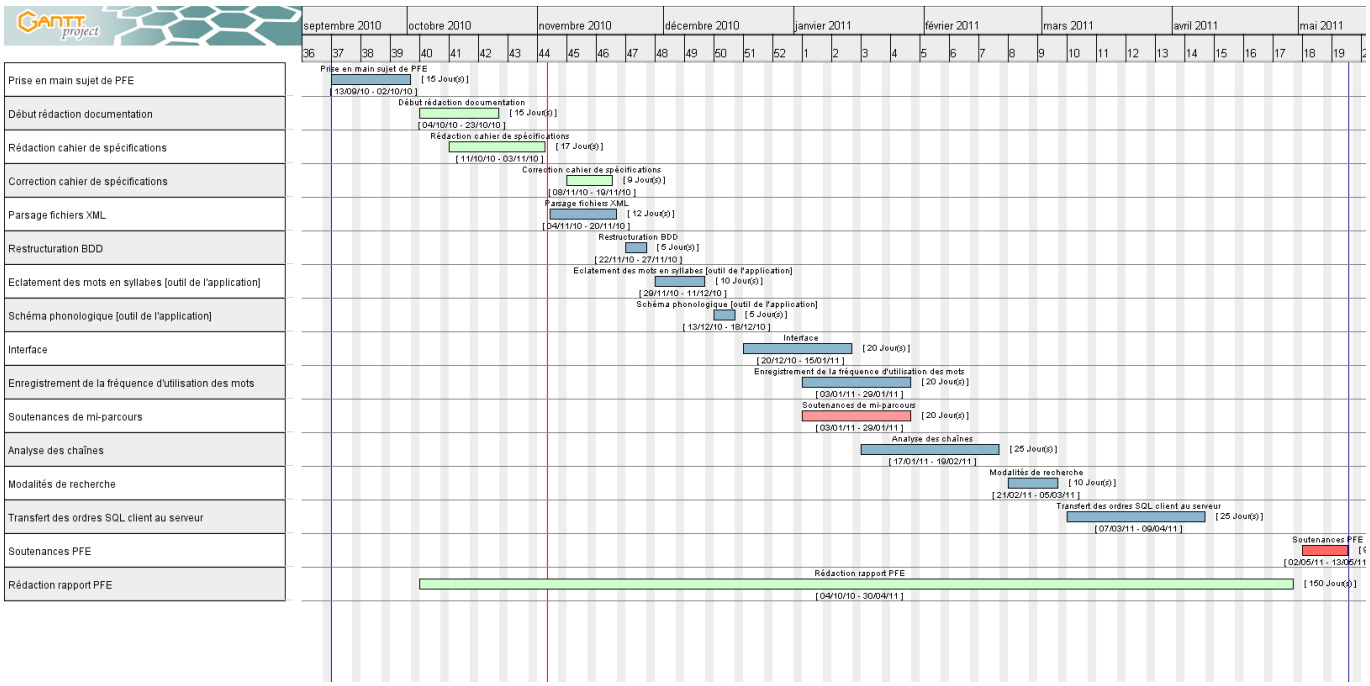


FIGURE 3.1 – Planning prévisionnel

On peut observer les points principaux du projet :

* parsage des fichiers XML
* restructuration de la base de données
* éclatement des mots en syllabes
* schéma phonologique
* interface
* enregistrement de la fréquence d'utilisation des mots
* analyse des chaînes
* modalités de recherche
* transfert des ordres SQL client au serveur

Ces différents points vont être expliqués ci-dessous.

3.1.2 Points du projet

Le parsage des fichiers XML

La base de données de l'application actuelle qu'utilise le LLL repose sur la fusion de trois fichiers XML, chacun de ces fichiers rassemblant les données de trois dictionnaires anglais :

* The Cambrigde English Pronouncing Dictionary
* The Longman Pronounciation Dictionnary
* The Macquarie Dictionary

Le parsage de ces fichiers est déjà existant, seulement il est défaillant sur certains points, et nécessite donc d'être repris en partie.

Restructuration de la base de données existante

La base de données de l'application existe également, mais certaines tables ne sont pas optimales : en effet, la base contient une table rassemblant les différents liens entre les trois dictionnaires sur de mêmes mots, et cette table n'est pas vraiment efficace, la création des liens n'est pas intuitive, et leur modification/suppression n'est pas réelle.

Mon travail consistera donc à reprendre cette table, à revoir la gestion de liens, mais il faudra également que je réfléchisse à propos de la table « pron », qui contient les prononciations des mots, et qui nécessite peut-être d'être éclatée, pour être mieux gérée ensuite, de façon plus claire.

De plus, la base actuelle n'est pas flexible en cas d'ajout d'un nouveau dictionnaire. Je dois donc également restructurer cette base de façon à ce que les futurs ajouts de nouveaux dictionnaires se fassent le plus simplement possible, et sans modification ou ajout de tables.

L'éclatement des mots en syllabes

Ce point fait partie des outils de l'application. Ces outils existent, mais ne fonctionnent pas vraiment. Il va donc falloir que je les reprenne, voire même que je les redéveloppe complètement.

Pour cette partie, il va falloir que les mots soient éclatés en syllabes, mais pas seulement en syllabes syntaxiques : il faudra également éclater le terme phonétique correspondant au mot. Il faudra donc gérer ces deux sortes d'éclatement, qui ne vont pas se dérouler de la même façon, mais également gérer la correspondance entre ces deux termes éclatés.

Pour l'éclatement des termes phonétiques, il va également falloir que je gère l'accentuation. Par exemple, une syllabe à accent principal sera précédée du signe « ' », une syllabe à accent secondaire du signe « ˌ », et une syllabe sans accent n'aura pas de signe. Cependant, le LLL a l'habitude de précéder les syllabes sans accent du signe « . ». Il faut donc gérer ces signes « . », les générer pour les syllabes sans signe particulier.

Le schéma phonologique

Tout comme l'éclatement des mots en syllabes, ce point fait partie des outils de l'application.

Le schéma phonologique se définit comme tel : « 1 »pour les syllabes à accent principal, « 2 »pour les syllabes secondaires, et enfin « 0 »pour les syllabes sans accent. Ce schéma s'appuiera bien entendu sur le point précédent.

L'interface

Dans l'application actuelle, quelques points de l'interface sont à revoir, comme des champs en trop pour l'éclatement des syllabes, des fenêtres d'interrogation/édition qui ne sont pas judicieusement placées, ...

L'enregistrement de la fréquence d'utilisation des mots

M. Fournier voudrait enregistrer la fréquence d'utilisation des mots, en collaboration avec une équipe de recherche américaine, qui a établi une base de données d'environ 400 millions de mots, avec leur fréquence d'utilisation. Voici le lien du site internet de cette équipe de recherche : http://www.americancorpus.org/.

L'analyse des chaînes

Cet outil n'existe pas pour le moment. Il faudra donc le développer intégralement.
Pour comparer avec l'application courante, la recherche de mots peut se faire avec des spécificités précises : commence par, se finit par ...
Le nouvel outil à développer se comportera de la même façon, c'est-à-dire que l'utilisateur pourra spécifier qu'il cherche un mot dont la première syllabe est accentuée, sans consonne, avec ou sans « e », ...

Des outils effectuant ce genre de tâche doivent exister sur le marché, il suffira de les adapter aux besoins du LLL.

Les modalités de recherche

Actuellement, l'application permet d'effectuer des recherches « en cascade » : l'utilisateur sélectionne d'abord un groupe de mots dans la base de données, puis il effectue une recherche sur ce groupe de mots, isole le résultat, effectue une recherche dessus et ainsi de suite.
Avec l'application courante, tous ces groupes de résultats sont regroupés sur un seul et même fichier, ce qui oblige à parcourir tout le fichier pour trouver le résultat, et ce qui est handicapant lorsqu'un chercheur du LLL veut fournir ses résultats à quelqu'un, mais sans lui fournir tous les groupes de recherche intermédiaires.

M. Fournier voudrait que ces différents groupes de résultats soient placés dans des fichiers différents, pour n'avoir à la fin que les résultats finaux sur un fichier indépendant.

Le transfert des ordres clients SQL au serveur

Actuellement, la structure informatique de l'application est la suivante : M. Fournier possède la base « serveur »sur son poste, et il la distribue aux autres chercheurs lors de modifications. Les chercheurs, qui sont donc clients de M. Fournier, lui transmettent ensuite un fichier contenant tous les ordres effectués pour modifier la base de données, que M. Founier applique ensuite à sa base de données serveur.

Selon l'application actuelle, le ficher des ordres SQL client n'est généré qu'à la fermeture de l'application, ce qui peut être risqué en cas de coupure innoportune de l'application.
M. Fournier souhaiterait donc que le ficher soit généré au fur et à mesure, comme ça en cas de coupure de l'application, le travail effectué en amont n'est pas perdu.

3.2 Tâches réalisées

3.2.1 Restructuration de la base de données existante

La base de données utilisée actuellement par le LLL a été totalement refondue.
En effet, la base de données mise en place n'était pas flexible du tout : elle était totalement dépendante des dictionnaires utilisés par le LLL. Le LLL utilisant actuellement les données issues de trois dictionnaires différents, chaque dictionnaire avait sa table dans la base de données, et les tables liées à ces dictionnaires en étaient donc multipliées.
Cette structure ne permet pas de faire durer dans le temps la base de données ; en effet, à chaque rajout ou suppression de dictionnaires, il faut modifier totalement la structure de la base, et le code qui fait appel aux données présentes dans cette base. Ce n'est évidemment pas envisageable pour une application utilisée par des professionnels, ne serait-ce qu'en coûts de refonte du code et de la base, ainsi qu'au niveau des délais nécessaires pour réaliser ces refontes, délais durant lesquels le LLL ne peut pas exploiter les données du nouveau dictionnaire.

Le premier but de mon PFE a donc été de refondre totalement cette base de données. Elle est maintenant beaucoup plus flexible, et les ajouts ou retraits de dictionnaires peuvent se faire sans avoir à modifier toute la structure de la base, ce qui permet que cette dernière soit pérenne.
Pour avoir un ordre d'idée, le nombre de tables de la base est passé de 44 à 28, sans supprimer aucune informations ou liaisons entre les tables.
Les représentations de ces deux base de données, la base initiale et celle restructurée par mes soins sont disponibles en annexe de ce rapport.

Cette refonte de MCD ne s'est pas faite en une seule fois. Il a fallu beaucoup de temps et de MCD intermédiaires avant d'arriver à un résultat concluant. En effet, le MCD initial n'était vraiment pas prévu pour avoir une base de données souple dans le temps : comme vous pouvez le voir dans l'annexe, la plupart des tables étaient spécifiques à chaque dictionnaire, ce qui n'est bien sûr pas envisageable pour une base de données censée durer dans le temps, et ce malgré la suppression ou l'ajout de dictionnaires.

La première étape de cette refonte a été de supprimer les tables en double ou en triple, notamment celles spécifiques à chaque dictionnaire. Seules les tables headword_XXX (par exemple headword_epd pour les mots issus du dictionnaire epd) ont été conservées ; mais toutes celles qui pouvaient être rendues génériques immédiatement (les tables concernant les notes, les sens, les syllabes et autres) l'ont été dès cette première étape.
Cette première étape a également permis de supprimer les attributs en trop ; en effet, lors de la création de l'application, cette dernière prenait en compte le fait qu'un mot pouvait avoir au maximum 20 syllabes. Il s'est avéré qu'un mot anglais ne pouvait avoir au maximum que 10 syllabes ; des champs étaient donc en trop dans la base de données, et ont été supprimés.

Dans la deuxième étape, les tables headword_XXX ont été fusionnées en une seule et même table, qui contiendra les mots de tous les dictionnaires. Pour pouvoir garder l'origine de ces mots, la table dictionnaires a été rajoutée au MCD : elle contient les noms des différents dictionnaires dont les données sont présentes dans la base, et est en relation avec les tables headword, syllabes, et toutes les autres tables dont on a besoin de garder l'origine des données.

La dernière étape de cette refonte s'est concentrée sur la table syllabes ; en effet, on voit dans le premier MCD qu'il y avaient des syllabes pour le mot en lui-même (syllabesheadword_XXX), et pour tous la version phonétique de ce mot (syllabesflexion_XXX). Lors des précédentes étapes, ces deux tables avaient été réunies dans cette seule et même table syllabes.

On peut voir dans le nouveau MCD que cette table est en relation avec les tables headword et flexions, et que les coordonnées 0,1 qui lui sont attribuées permettent d'indiquer que les valeurs de cette table concernent soit les syllabes d'un mot, soit les syllabes de sa version phonétique.
Bien évidemment, cette dernière étape a également servi à affiner les relations entre les tables, notamment entre les tables headword et liens_dict.

3.2.2 Le parsage des fichiers XML - l'analyse

L'analyse du parsage des fichiers XML a été effectuée. Elle a permis de mettre en évidence les liens entre les différents projets qui constituent l'application, ainsi que les points épineux des projets à modifier avec attention.
Ces points seront détaillés dans ce rapport, mais voici tout d'abord un schéma permettant de voir les relations entre les différents projets :

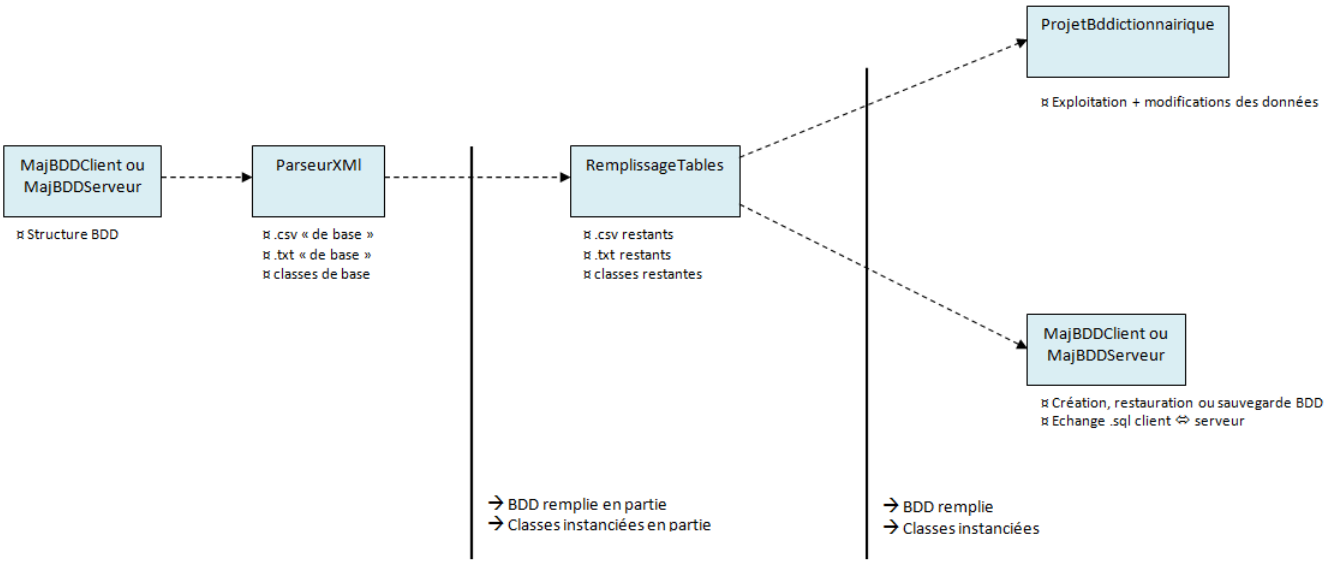


FIGURE 3.2 – Relations entre les différents projets constituant l'application

Dans ce schéma, on voit que la construction de la base de données passe par les projets MajBDDClient ou MajBDDServeur. Pour rappel, dans le cas du LLL, le serveur c'est l'ordinateur portable de M. FOURNIER, et les clients sont les postes des autres chercheurs du LLL.

Une fois cette base de données construite, le projet ParseurXML permet de générer les fichiers .csv ou .txt « de base », c'est-à-dire les fichiers contenant les données des tables « simples », qui ne nécessitent pas d'analyse, dont les informations sont directement tirées du dictionnaire (exemple : la table « Headword »- ses données sont de base dans le dictionnaire).

À ce stade, la base de données est partiellement remplie, et les classes du projet principal (ProjetBddictionnairique) sont partiellement instanciées.

Le projet RemplissageTables permet ensuite de générer les fichiers .csv ou .txt restants, c'est-à-dire ceux qui nécessitent une analyse ou un traitement supplémentaire avant d'être générés (exemple : la table « Syllabes »- ses données sont issues d'un découpage du mot présent dans le dictionnaire). Une fois tout ceci effectué, la base de données est totalement remplie, et les classes du projet principal sont toutes instanciées.

L'utilisateur peut ensuite soit utiliser le projet ProjetBddictionnairique pour exploiter - voire modifier - les données des différents dictionnaires présents dans la base de données, ou bien faire des sauvegardes ou autres enregistrements sql via les projets MajBDDClient ou MajBDDServeur selon l'utilisateur. Ces deux derniers projets permettent également de faire communiquer les ordres clients sql vers le serveur, ou inversement.

Voici un exemple du fonctionnement du projet ParseurXML, qui génère les fichiers .csv à partir du fichier XML dictionnaire :

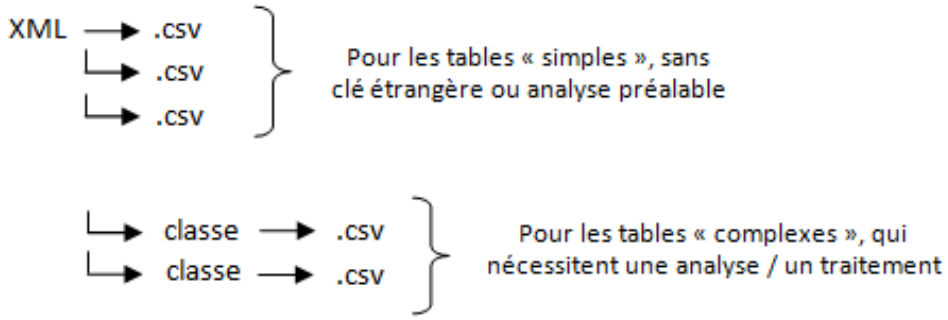


FIGURE 3.3 – Exemple de fonctionnement du projet ParseurXML

On voit bien comment sont construits les fichiers .csv (ou .txt) selon le degré de traitement qu'ils nécessitent : les fichiers qui recensent les données « pures »du dictionnaire sont créés directement (ainsi que les classes liées), alors que ceux qui contiennent les données nécessitant un traitement supplémentaires sont faits ensuite, en passant par une classe supplémentaire de traitement.

Un des plus gros problèmes qui est ressorti de cette analyse concerne justement la génération de ces fichiers .csv : en effet, la nouvelle base de données contient beaucoup plus de clés étrangères que la précédente. Ces clés étrangères sont le lien entre plusieurs tables ; elles permettent d'indiquer que telles informations contenues dans une table sont reliées avec d'autres informations contenues dans d'autres tables. La valeur de ces clés étrangères correspond à la clé primaire de la table avec laquelle on est relié.

Le problème qui se présente est que si on enregistre des informations dans une table (qui est censée contenir une clé étrangère) alors que la valeur de la clé primaire représentant cette clé étrangère n'est pas renseignée la clé étrangère ne peut pas être instanciée, et le lien entre ces tables n'est plus.

3.2.3 Solutions possibles pour palier au problème de clés étrangères - Les différents cas

Cette analyse, bien que fastidieuse, m'a permis de faire ressortir certaines solutions possibles pour gérer ce problème d'enregistrement de clés étrangères.

Première solution

La première solution consisterait à mettre les clés étrangères à NULL, et de les remplir ensuite lors de l'enregistrement des données de la table référencée. Trois cas sont alors possibles : en voici la présentation.

Cas 1 : une relation sans clé étrangère (une seule table)

Nous sommes dans le cas où la table possède des cardinalités 0,N (ou 1,N), ce qui signifie qu'elle ne contient pas de clé étrangère :

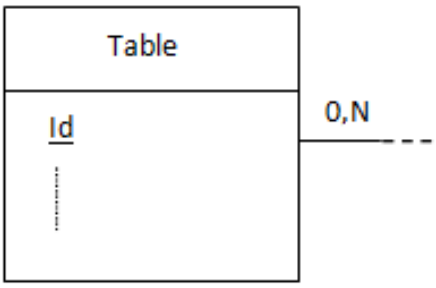


FIGURE 3.4 – Une table avec une relation sans clé étrangère

Dans ce cas précis, il n'y a pas de problème pour enregistrer les données, car il n'y a pas de lien clé primaire/clé étrangère avec une autre table.

Cas 2 : une relation avec clé étrangère (deux tables)

Nous sommes dans le cas où une des deux tables possède des cardinalités 0,N (ou 1,N), et l'autretable possède quant à elle des cardinalités du type 1,1. La table avec la cardinalité 1,1 possèdera donc une clé étrangère, qui aura la même valeur que la clé primaire de l'autre table :

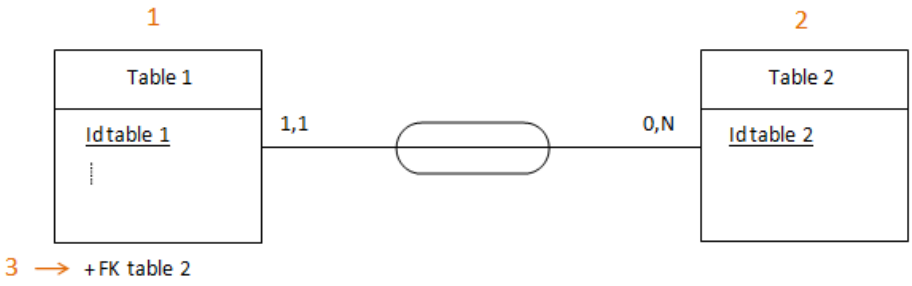


FIGURE 3.5 – Deux tables avec une relation clé primaire/clé étrangère

Comme on peut le voir sur le schéma, la solution consisterait à enregistrer tout d'abord les données de la table 1 mais sans renseigner la clé étrangère, et d'enregistrer ensuite les données de la table 2.

Une fois que les données de la table 2 sont enregistrées, on peut récupérer la valeur de sa clé primaire et l'enregistrer en tant que clé étrangère dans la table 1 (FK dans le schéma (foreign key)).

Cas 3 : une relation avec table porteuse et clé multiattribut (trois tables)

Nous sommes dans le cas où deux tables possèdent des cardinalités 0,N (ou 1,N), et où la dernière table possède deux cardinalités de type 1,1. Cette table possèdera donc deux clés étrangères, chacune de ces clés référant à une des deux tables en 0,N :

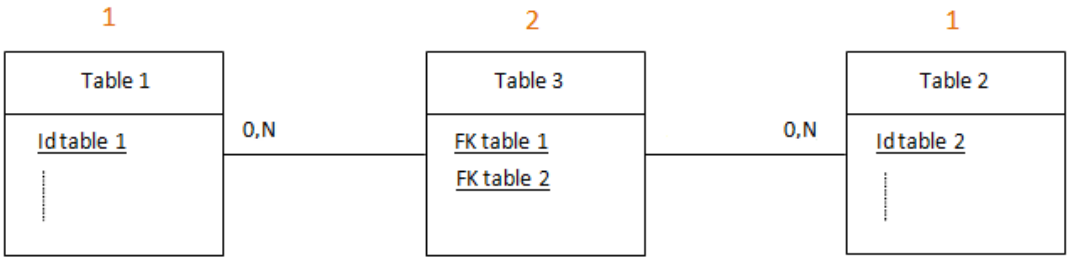


FIGURE 3.6 – Trois tables avec deux relations clé primaire/clé étrangère

Une des solutions pourrait consister à enregistrer les données des deux tables en 0,N en premier, ainsi leur clés primaires sont renseignées, et ensuite on renseigne les données de la table porteuse (en 1,1 deux fois), avec comme clé primaire les deux clés étrangères référant aux deux autres tables (FK dans le schéma (foreign key)).

Deuxième solution

Comme vu précédemment, les fichiers .csv sont générés d'une certaine façon :

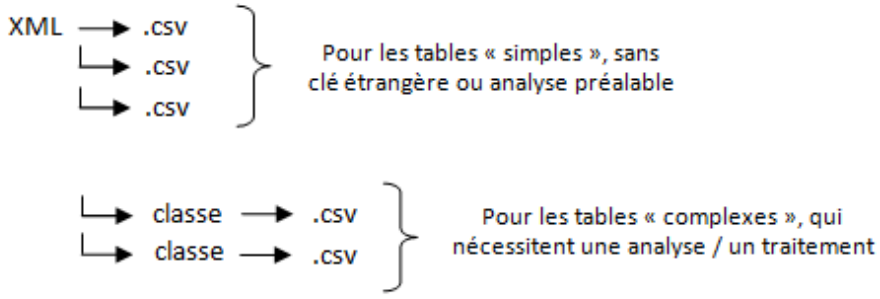


FIGURE 3.7 – Exemple de fonctionnement du projet ParseurXML

La deuxième solution consisterait à utiliser une table intermédiaire :

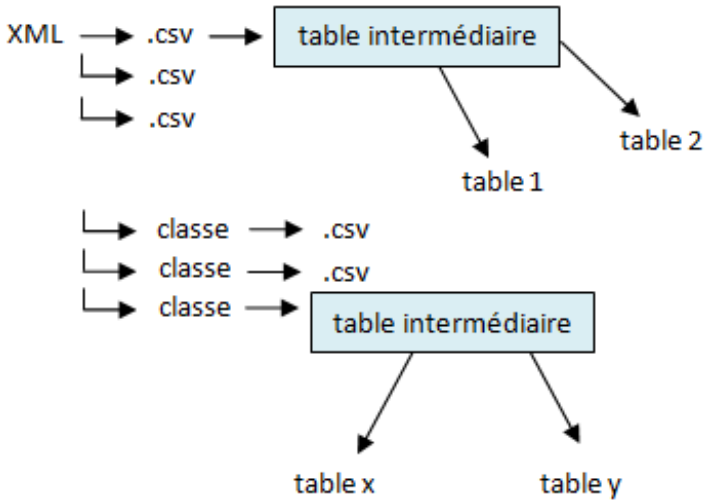


FIGURE 3.8 – Deuxième solution : utiliser une table intermédiaire

Cette table serait créée au besoin : comme le montre le schéma, on mettrait dedans toutes les données, et ensuite on redistribuerait dans les différentes tables de la base de données.
Une fois que toutes les données seraient redistribuées à leur place, on supprimerait cette table intermédiaire de la base de données.

3.2.4 Modification de l'interface

Dans l'interface, les champs contenant les syllabes des mots étaient au nombre de 20. Comme expliqué précédemment, suite à de nombreuses recherches le LLL a estimé que les mots anglais ne pouvaient pas avoir plus de 10 syllabes.
J'ai donc supprimé ces champs en trop dans l'interface, comme vous pouvez le voir ci-dessous :

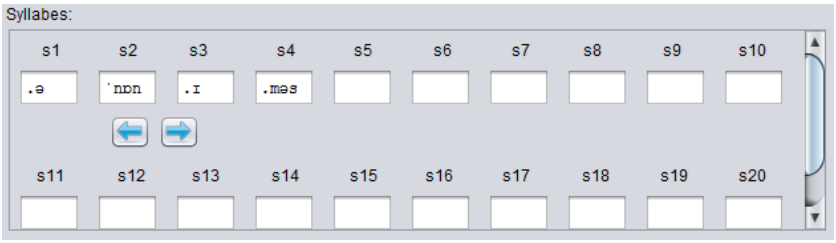


FIGURE 3.9 – Ancienne application - 20 champs pour les syllabes d'un mot

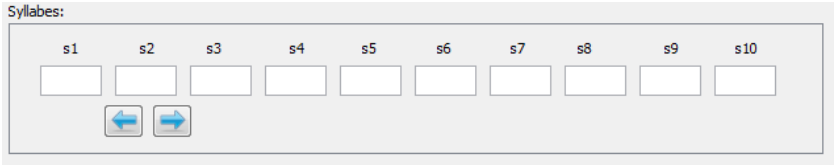


FIGURE 3.10 – Nouvelle application - 10 champs pour les syllabes d'un mot

3.3 Tâches non réalisées

La restructuration de la base de données et l'analyse des différents projets constituant l'application ayant pris beaucoup de temps, tous les points définis en début d'année scolaire n'ont pas pu être réalisés. Il reste des éléments à faire :

Le parsage des fichiers XML

L'analyse ayant été effectuée, il faut maintenant modifier le code pour qu'il corresponde à la nouvelle base de données et au principe de modularité de cette base : il faut pouvoir ajouter un dictionnaire sans avoir à modifier tout le code.
J'ai précisé en annexe ce qui a été fait comme code et ce qu'il reste à faire par projet et par package.

L'éclatement des mots en syllabes

Ce point fait partie des outils de l'application.
Cette partie va être à modifier pour corriger les bugs existants, mais également pour correspondre à la nouvelle base de données.

Le schéma phonologique

Ce point fait également des outils de l'application.
Ce schéma s'appuiera sur le point précédant, et devra être adapté à la nouvelle base de données.

L'interface

Dans l'application actuelle, certains champs étaient en trop pour l'éclatement des syllabes, des fenêtres n'étaient pas judicieusement placées, ce qui rendait l'application peut intuitive.
Les champs en trop ont été corrigés à certains endroits de l'application, mais pas partout. Il faudra également corriger les autres points.

L'enregistrement de la fréquence d'utilisation des mots

Cet outil, en relation avec une équipe de recherche américaine (http://www.americancorpus.org/), n'existe pas de base, et est donc à faire en totalité.

L'analyse des chaînes

Cet outil inexistant à l'heure actuelle permet de faire des recherches très précises du type éditeur de code dans la base de données. Des outils effectuant ce genre de tâche doivent exister sur le marché, il suffira de les adapter aux besoins du LLL.

Les modalités de recherche

Ce module permettant d'enregistrer les étapes de recherche dans différents fichiers n'a pas été réalisé.

Le transfert des ordres clients SQL au serveur

Cette partie passe par les projets MajBDDClient ou MajBDDServeur. Ces projets génèrent un fichier contenant les ordres sql que l'utilisateur a appliqué sur sa base de données. Ce fichier est ensuite transmis au serveur, qui applique ainsi les modifications sur sa base de données.
Ce fichier doit être généré au fur et à mesure des ordres sql.

Organisation du travail

4.1 Mises au point

Durant toute la durée du projet, des mises au point ont été nécessaires, aussi bien avec mon encadrant qu'avec le client du projet.

4.1.1 Mises au point avec l'encadrant

Mon encadrant, Mme TACQUARD, était disponible assez souvent pour que je puisse aller la voir à n'importe quel moment de la journée, que ce soit durant les deux jours consacrés au PFE ou non.
Cette proximité et cette disponibilité m'ont permis d'aller la voir assez facilement pour pouvoir lui poser des questions, me débloquer dans des situations complexes, et rediriger mon analyse quand je m'écartais un peu trop de l'objectif.

4.1.2 Mises au point avec le client

Plusieurs rendez-vous de mise au point ont été faits avec M. FOURNIER durant l'année. Ils nous ont permis de communiquer plus facilement, de mettre à plat certains problèmes, et de présenter comment avançait le projet.
Si jamais les rencontres n'étaient pas possibles ou pas nécessaires, nous communiquions par mail, notamment pour que je puisse lui poser une question qui ne nécessitait pas de déplacement à l'école.
Malgré des difficultés pour organiser des rendez-vous, dues aux emplois du temps de chacun, nous avons tout de même pu nous voir environ tous les deux-trois mois, pour pouvoir faire avancer le projet de façon convenable.

4.1.3 Mises au point avec le client et l'encadrant

Durant toute l'année scolaire, des rendez-vous entre le client, l'encadrant et moi-même ont été effectués. Ces rencontres m'ont permises d'informer tout le monde de mon avancement dans mon projet, mais également de réfléchir ensemble à certains problèmes, ou de faire valider une étape importante de mon projet (par exemple la base de données).
Une soutenance de mi-parcours a également été faite ; tout d'abord pour présenter l'état de mon projet à cette période-ci, mais également pour avoir un avant-goût de ma soutenance finale de PFE.

4.2 Outils utilisés

Pour pouvoir mener ce projet à bien, certains outils ont été utilisés. Pour la plupart je n'ai pas eu à choisir lesquels prendre, car reprenant un projet déjà existant, j'ai du reprendre l'environnement de développement utilisé au préalable.

4.2.1 Editeur

Le précédent développeur ayant commencé à développer l'application avec l'éditeur NetBeans, j'ai continué sur ce même éditeur.

4.2.2 Système de gestion de base de données relationnelle, outil d'administration et installateur

Le SBGD choisi est MySQL, l'outil d'administration PhpMyAdmin et l'installateur XAMPP. Pour plus de précisions quant à ces choix, je vous renvoie au manuel de conception et développement du précédent développeur.

4.2.3 Subversion

Afin d'avoir un serveur de sauvegarde mais également pour gérer les versions, tout mon projet était déposé sur Assembla, un dépôt SVN de projets, et était synchronisé avec les données sur mon ordinateur via TortoiseSVN.

4.2.4 Gestion du planning

Pour pouvoir établir le planning prévisionnel, j'ai utilisé GanttProject, qui a l'avantage d'être un outil gratuit et simple d'utilisation.

Bilan personnel

5.1 Compétences acquises

Ce projet de fin d'études est un projet de grande envergure par rapport aux projets précédents que nous avons effectués.
Le fait d'être seule aux commandes de ce projet m'a permis de vraiment voir comment un chef de projet se doit de gérer les différentes étapes, les différentes tâches d'un projet, mais également de faire face aux imprévus liés à ce projet.
En effet, un projet ne se déroule jamais vraiment comme prévu, et il faut tout le temps faire face à divers imprévus, tels qu'une durée de tâche mal planifiée, un aspect du projet qu'on ne pensait pas si complexe ou si important, ou bien même une validation de tâche du projet difficile à obtenir, car le client n'est pas toujours disponible comme on le souhaiterait.

Pour ma part cette responsabilité de chef de projet m'a vraiment permis de comprendre ces difficultés auxquelles on peut avoir à faire face. Ce projet à gérer tout au long de l'année nous montre comment se déroule une gestion de projet à « long terme » : cela nous permet de voir que les phases les plus importantes d'un projet ne sont pas forcément le développement, mais que les phases d'analyse et de conception sont bien plus importantes. En effet, sur un projet à long terme, il est important de consacrer du temps à la partie analyse et conception, car on ne peut pas travailler sur le développement d'un projet sans mettre sur papier ses grandes lignes de conception.

Ce projet de fin d'études m'a permis également de préciser mes ambitions au niveau de ma carrière : dans ce projet nous voyons les différentes grandes phases d'un projet, et par conséquent les grands rôles d'une équipe autour de ce projet. Cela m'a permis de comprendre si plus tard je me voyais plus dans le rôle du développeur ou dans le rôle de l'analyste-concepteur, ou bien même dans le rôle de chef de projet.
Même si je ne peut pas vraiment définir ma carrière maintenant, cela m'a permis d'entrevoir les différentes possibilités s'offrant à moi, et ainsi de mieux cibler les postes auxquels je pourrait m'intéresser.

5.2 Difficultés rencontrées

Tout au long de cette année scolaire, la gestion de ce projet de fin d'études n'a pas été sans embûches. En tout premier temps, il a fallu établir le planning de ce projet : cela a impliqué le découpage en tâches. Cette tâche nécessite de dégager les grandes lignes, les grandes étapes du projet, ce qui n'est pas forcément évident, même si dans mon cas M. Fournier m'avait déjà fait ressortir les grandes étapes.

Une fois ces grandes étapes établies, il faut estimer la durée de chacune. Dans mon cas, cela s'est avéré être une vraie difficulté, car les phases d'analyse et de conception ont été beaucoup plus importantes que prévues. Je n'ai pas su estimer correctement ces phases, et cela a beaucoup handicapé le déroulement du reste du projet, car les retards causés ont été très importants.
Le reste du projet ayant été très retardé à cause de ces mauvaises estimations, il n'a pas pu vraiment être mené à terme. La gestion de ces retards a été l'autre grosse difficulté du projet, car il a fallu gérer tout cela, gérer toutes ces causes à effet.

Le planning ci-dessous montre les grands écarts entre les durées prévues et les durées réelles des tâches du projet :

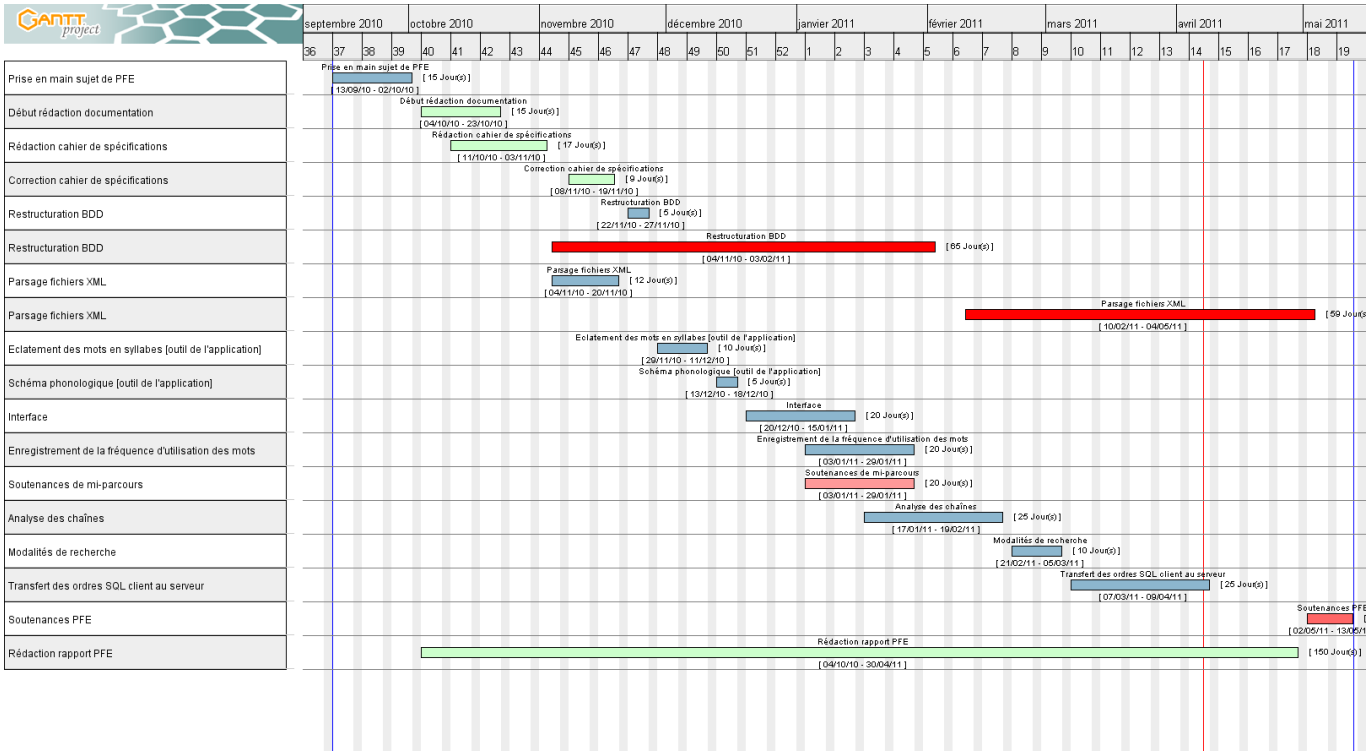


FIGURE 5.1 – Comparatif planning prévu - planning réel

On peut voir sur ce comparatif que l'estimation de la durée des deux premières tâches était fausse, et que le retard occasionné par ces durées n'a pas permis d'effectuer les autres tâches du projet.

Lors de la soutenance de mi-parcours, les tâches concernant l'analyse des chaînes, l'interface et la fréquence des mots ont d'ailleurs été mis de côté volontairement suite à ce retard.

Une autre difficulté de ce projet n'est pas de l'ordre de la gestion de projet en soi, mais c'est un aspect qui est quand même très important. La communication avec le client n'est pas toujours une chose facile, et respecter ses attentes l'est encore moins.
En effet, gérer les attentes du client tout en gérant les imprévus, les retards dans le projet n'est pas évident : le client a des attentes, et du côté du chef de projet il faut savoir travailler avec cet aspect, et avec les imprévus du projet.

Il faut donc savoir exposer ses éventuels problèmes au client, pour pouvoir lui faire comprendre que toutes ses attentes ne seront pas forcément remplies. Mais il faut cependant lui expliquer que même si les résultats concrets qu'il attend ne se feront pas forcément, toutes les phases de recherche, d'analyse et de conception effectuées ne sont pas vaines, et qu'elles permettront par la suite de réaliser de manière beaucoup plus rapide et beaucoup plus efficace la phase de développement.

Conclusion

Conclure sur ce projet n'est pas une chose si simple que cela ne le paraît.
Ce projet m'a apporté tout un tas de choses, aussi bien sur la technique en elle-même que sur mes attentes à propos du monde du travail.

Du point de vue technique, développement, ce projet a été très riche en apports et en remises en questions.
Gérer tout un projet sur une année scolaire entière n'est pas un grand exploit dans le monde professionel, mais cela m'a tout de même permis de voir et comprendre comment se déroulait un projet de moyenne envergure. Il ne faut pas s'asseoir devant son ordinateur et commencer à travailler sur le code dès le premier jour.
Il faut tout d'abord prendre le temps de bien comprendre le sujet, discuter avec le client pour qu'il nous explique au mieux l'environnement dans lequel s'applique le projet, pour comprendre à la fois les attentes du client, mais également les comportements possibles des utilisateurs avec l'application.

Quand nous sommes à l'école, cette phase de « réflexion »sur papier ne nous semble pas primordiale. Cependant, sur un projet qui commence à être conséquent, elle l'est. Elle nous permet de bâtir les grandes lignes du projet, de voir sans coder quelles directions vont être prises, et quelles éventualités vont être éliminées dès le début, pour ne pas ensuite perdre de temps lors de la phase de code.
Le développeur saura dès le début quelle direction prendre, et il ne sera pas perdu au milieu de différentes possibilités, il aura déjà un fil rouge à suivre.

Hormis cet apprentissage sur l'importance de la réflexion, de la phase d'analyse dans un projet, ce travail durant l'année nous a permis de voir un aspect que nous n'avions pas vraiment vu jusque-là : le rôle du chef de projet.
Même si nous étions notre propre équipe dans ce projet, nous avons du gérer un planning, estimer des durées de tâches, discuter avec le client, se faire comprendre de lui, et gérer ses attentes.
Ce projet m'a vraiment permis de voir quelle place a le chef de projet dans une équipe.
Il n'est peut-être pas en train de réaliser le code de l'application, mais c'est lui qui récupère les informations venant du client, ses besoins, et c'est lui qui planifie tout le projet selon des tâches précises. C'est un rôle essentiel dans l'équipe, et gérer ce projet m'a vraiment permis de réaliser tout cela.

D'un point de vue beaucoup plus personnel, ce projet m'a également beaucoup apporté.
Le fait d'être sa propre équipe, d'endosser plusieurs rôles m'a permis de me projeter dedans, et de voir quel rôle me conviendrait le mieux dans ma vie professionnelle. Cela reste bien sûr un projet de moyenne envergure, et j'était ma propre équipe, mais être présente dans les différents aspects d'un projet va - je pense - orienter mes choix de carrière, et partir dans la vie professionnelle avec une vision, ne serait-ce que minime, des différents rôles dans un projet est pour moi un avantage, un point fort.

Lexique

Dictionnaires

EPD : The Cambridge English Pronouncing Dictionary.
LPD : The Longman Pronouncing Dictionary.
MCQ : The Macquarie Dictionary.

Morphème

Morphème : en linguistique, on définit généralement un morphème comme la plus petite unité porteuse de sens qu'il soit possible d'isoler dans un énoncé. De même que le phonème, le morphème est une entité abstraite susceptible de se réaliser de plusieurs manières dans la chaîne parlée (source Wikipédia).

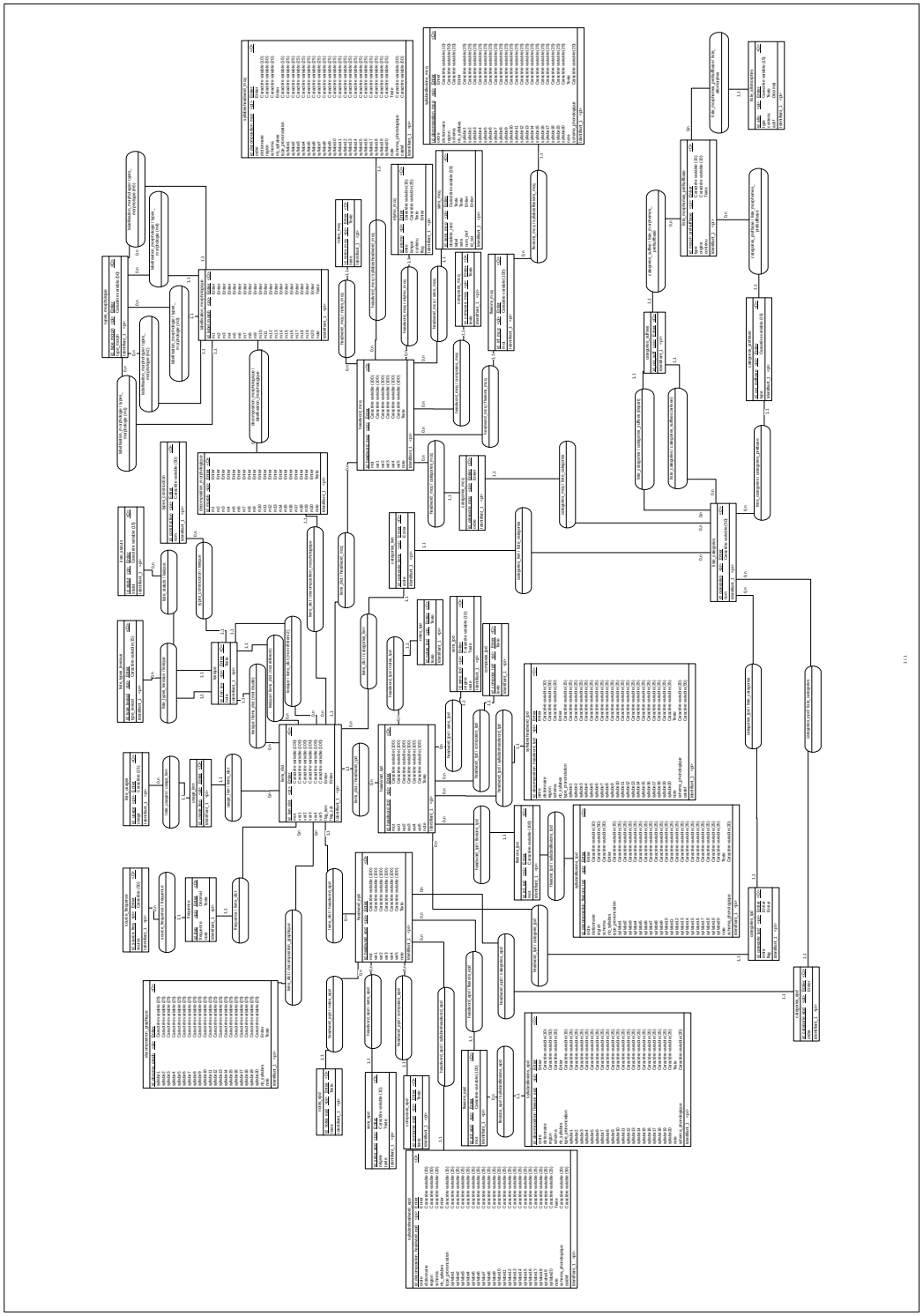
Graphie

Graphie : représentation écrite d'un mot.

Allomorphe

Allomorphe : l'allomorphe est l'équivalent en morphologie de l'allophone (réalisation sonore possible d'un phonème (plus petite unité discrète ou distinctive (c'est-à-dire permettant de distinguer des mots les uns des autres) que l'on puisse isoler par segmentation dans la chaîne parlée)).
De même que plusieurs réalisations d'un phonème sont possibles, un morphème peut prendre plusieurs formes. Il s'agit alors soit de variantes libres soit de variantes contextuelles (source Wikipédia).

MCD de l'application actuelle



Description des tables du MCD de l'application actuelle

categories_epd

Cette table représente la catégorie à laquelle appartiennent les mots issus du dictionnaire EPD.
Elle fait le lien entre la table des différentes catégories de mots (liste_categories : adverbe, adjectif, ...), et la table des mots de dictionnaire (headword_epd).

categories_lien

Cette table contient les différentes catégories grammaticales d'un lien entre les différents dictionnaires (EPD, LPD, et MCQ).
Elle contient entre autre l'ordre des différentes catégories pour un même lien.

categories_lpd

Cette table représente la catégorie à laquelle appartiennent les mots issus du dictionnaire LPD.
Elle fait le lien entre la table des différentes catégories de mots (liste_categories : adverbe, adjectif, nom propre, ...), et la table des mots de dictionnaire (headword_lpd).

categories_mcq

Cette table contient la catégorie à laquelle appartiennent les mots issus du dictionnaire MCQ.
Elle fait le lien entre la table des différentes catégories de mots (liste_categories : nom propre, abréviation, adjectif, ...), et la table des mots de dictionnaire (headword_mcq).

categories_prefbase

Cette table enregistre la catégorie d'un morphème de type préfixe ou de type base.
Cette table fait le lien avec la table liste_categories, qui permet de savoir à quelle catégorie appartient un mot, et la table liste_morphemes_prefsuffbase, qui contient la liste des morphèmes de tous types (préfixe, base, ou suffixe).

categories_suffixes

Cette table contient la catégorie de mot de départ d'un morphème de type suffixe, puis sa catégorie de mot d'arrivée.
Elle fait le lien entre la table liste_categories, qui contient toutes les catégories de mots (abréviation, adverbe, ...), et la table liste_morphemes_prefsuffbase, qui contient la liste des morphèmes de tous types (préfixe, base, ou suffixe).

composes_epd

Cette table contient les formes composées associées à une graphie d'un mot issu du dictionnaire EPD (ex : acid (bonbon pour cet exemple) est contenu dans la table headword_epd, et peut avoir comme composant drop (acid drop : bonbon acidulé). Le composant drop est donc contenu dans la table composes_epd).

composes_lpd

Tout comme composes_epd, cette table contient les formes composées associées à une graphie d'un mot issu du dictionnaire LPD.

composes_mcq

Cette table fonctionne comme les tables composes_epd et composes_lpd, elle contient les formes composées associées à un mot issu du dictionnaire MCQ (ex : abacuses (bouliers) a comme composant abacus (boulier). Un composant peut également être une forme différente du mot contenu dans la table headword_epd/lpd/mcq, comme le pluriel de ce mot par exemple).

decomposition_graphique

Cette table est un des rouages de la base de données : en effet, elle contient la décomposition graphique d'un mot, c'est-à-dire les différentes syllabes graphiques de ce mot.

decomposition_morphologique

Cette table contient la décomposition morphologique (ensemble des morphèmes) d'un mot.

etymo_mcq

Cette table contient les indications éthymologiques des mots contenus dans le dictionnaire MCQ.

flexions_epd

Cette table contient les graphies des flexions, c'est-à-dire les graphies des différentes formes possibles d'un mot issu du dictionnaire EPD (ex : aardwolf (loup de terre ; insectivore hyène) dans la table headword_epd a comme flexion aardwolves , abandon a comme flexions abandons, abandoning, et abandoned).

flexions_lpd

Tout comme flexions_epd, cette table contient les graphies des flexions du mot issu du dictionnaire LPD.

flexions_mcq

Comme flexions_epd et flexions_lpd, cette table contient les graphies des flexions du mot issu du dictionnaire MCQ.

frequence

Cette table contient la liste des sources de fréquence (COCAE, COCAE sport, BNC, ...).

headword_epd

Cette table contient les graphies et leurs variantes d'un mot issu du dictionnaire EPD.

headword_lpd

Tout comme headword_epd, cette table contient les graphies et leurs variantes d'un mot issu du dictionnaire LPD.

headword_mcq

Cette table contient les différentes graphies et leurs variantes d'un mot issu du dictionnaire MCQ.

labellisation_morphologique

Cette table contient les labels de la décomposition morphologique d'un mot.
Par exemple, si un mot est décomposé en trois morphèmes, la labellisation de ces morphèmes peut être préfixe - base - suffixe.

lexique

Cette table contient tous les liens expliquant le lien entre deux mots : par exemple le mot displacement vient du mot displace (ex. tiré de la documentation développeur du projet actuel).
La table fait le lien entre la table liens_dict pour le mot étudié (ex : displacement) et cette même table liens_dict pour le(s) mot(s) référent(s) (ici displace). Elle est également liée avec la table liste_types_lexicaux, qui contient les différents types lexicaux (suffixé, préfixé, composé), ainsi que les tables liste_statuts (séparable, inséparable, savant, ...) et types_construction (types de construction lexicaux : juxtaposition, substitution, fusion, ...).

liens_dict

Cette table est le pilier de la base de données. En effet, elle contient les liens entre les mots des trois dictionnaires.
Elle permet également d'indiquer s'il y a des doublons de mot au sein d'un même dictionnaire, ainsi que s'il y a une catégorie présente dans deux des trois dictionnaires, mais pas dans le dernier.

liste_allomorphe

Cette table contient la liste des allomorphes (préfixe, suffixe, base), ainsi qu'un possible lien de parenté vers un morphème (ex : co- est parent de col).

liste_categories

Cette table est très importante dans la base de données : en effet, elle contient les différents types de catégorie grammaticale d'un mot (adjectif, adverbe, nom propre, abréviation, ...).

liste_morphemes_prefsuffbase

Cette table contient la liste des morphèmes (préfixe, suffixe, base). Elle est le point central des tables categories_suffixe, categories_prefbase, et liste_allomorphes.

liste_statuts

Cette table contient les statuts pour la table lexique (séparable, inséparable, savant, ...).

liste_types_lexicaux

Cette table contient la liste des types lexicaux possibles : suffixé, préfixé, composé, ...
Elle est en lien avec la table lexique.

liste_usages

Cette table contient les différents usages dont peu faire l'objet un mot : savant, général, discutable, ou rien.

notes_epd

Cette table permet de recenser toutes les notes liées aux mots contenus dans la table headword_epd.

notes_lpd

Comme la table notes_epd, cette table contient toutes les notes relatives aux mots de la table headword_lpd.

notes_mcq

Cette table permet également de regrouper toutes les notes sur les mots présents dans la table headword_mcq.

sens_epd

Cette table contient le sens associé à une graphie d'un mot issu du dictionnaire EPD.

sens_lpd

Cette table permet de connaître le sens associé à une graphie pour les mots issus du dictionnaire LPD.

sens_mcq

Tout comme les tables sens_epd et sens_lpd, cette table permet d'obtenir le sens associé à une graphie de mot issu du dictionnaire MCQ, ainsi que certaines indications (obsolète, label dans le dictionnaire, ...).

source_frequence

Cette table permet d'avoir la liste des sources de fréquence (ex : COCAE, COCAE sport, BNC, ...).

syllabesflexions_epd

Cette table contient l'éclatement en syllabes de la version phonétique du mot issu du dictionnaire EPD.

syllabesflexions_lpd

Cette table contient également l'éclatement en syllabes de l'équivalent phonétique du mot issu du dictionnaire LPD.

syllabesflexions_mcq

Tout comme les tables syllabesflexions_epd et syllabesflexions_lpd, cette table contient l'éclatement phonétique en syllabe d'un mot issu du dictionnaire MCQ.

syllabesheadword_epd

Cette table contient la décomposition en syllabes du mot issu du dictionnaire EPD.

syllabesheadword_lpd

Cette table contient l'éclatement syllabique du mot issu du dictionnaire LPD.

syllabesheadword_mcq

Cette table contient également la décomposition syllabique du mot issu du dictionnaire MCQ.

types_construction

Cette table contient les types de construction lexicaux possibles : juxtaposition, substitution, fusion, ...
Elle est liée avec la table lexique.

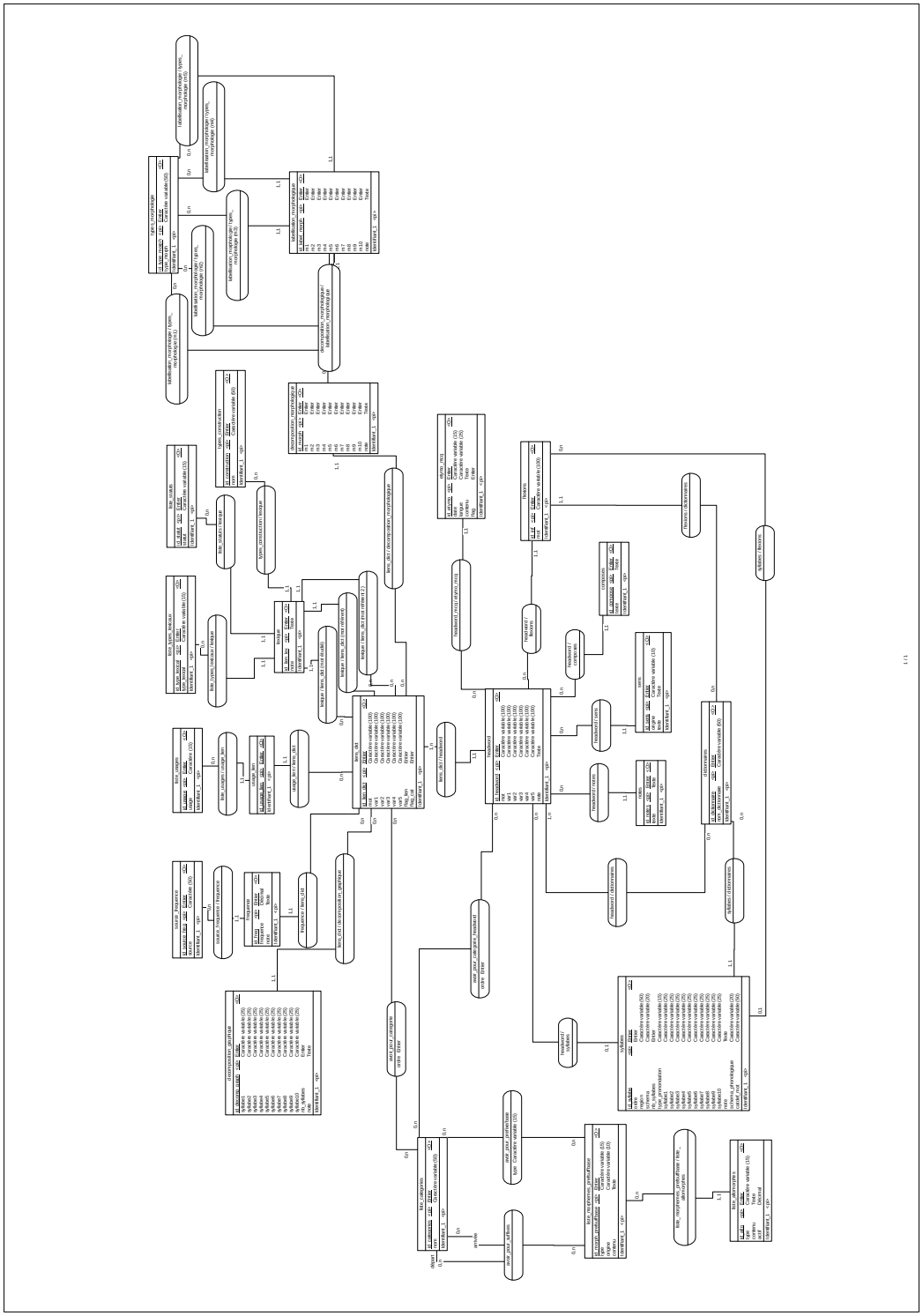
types_morphologie

Cette table contient les différents types de morphologie possibles : prefixe, base libre, suffixe lexical, ...
Elle est liée à la table labellisation_morphologique.

usage_lien

Cette table contient les usages faits d'un mot issu de la table liens_dict.
Elle fait le lien entre les tables liste_usages et liens_dict.

MCD restructuré



Description des tables du MCD restructuré

avoir_pour_categorie

Cette table fait le lien entre la table des différentes catégories de mots (liste_categories : adverbe, adjectif, ...), et la table des liens entre les mots des dictionnaires (liens_dict).
Elle contient un entier (ordre) qui représente l'ordre des différentes catégories pour un même lien.

avoir_pour_categorie_headword

Cette table fait le lien entre la table des différentes catégories de mots (liste_categories : adverbe, adjectif, ...), et la table des mots des dictionnaires (headword).
Elle contient l'ordre des différentes catégories pour un même mot.

avoir_pour_prefixe/base

Cette table enregistre la catégorie d'un morphème de type préfixe ou de type base.
Cette table fait le lien avec la table liste_categories, qui permet de savoir à quelle catégorie appartient un mot, et la table liste_morphemes_prefsuffbase, qui contient la liste des morphèmes de tous types (préfixe, base, ou suffixe).

avoir_pour_suffixes

Cette table contient la catégorie de mot de départ d'un morphème de type suffixe, puis sa catégorie de mot d'arrivée.
Elle fait le lien entre la table liste_categories, qui contient toutes les catégories de mots (abréviation, adverbe, ...), et la table liste_morphemes_prefsuffbase, qui contient la liste des morphèmes de tous types (préfixe, base, ou suffixe).

composes

Cette table contient les formes composées associées à une graphie d'un mot issu d'un dictionnaire (ex : acid (bonbon pour cet exemple) est contenu dans la table headword, et peut avoir comme composant drop (acid drop : bonbon acidulé). Le composant drop est donc contenu dans la table composes).

decomposition_graphique

Cette table est un des rouages de la base de données : en effet, elle contient la décomposition graphique d'un mot, c'est-à-dire les différentes syllabes graphiques de ce mot.

decomposition_morphologique

Cette table contient la décomposition morphologique (ensemble des morphèmes) d'un mot.

dictionnaires

Cette table contient les noms des différents dictionnaires présents dans la base de données.

etymo_mcq

Cette table contient les indications éthymologiques des mots contenus dans le dictionnaire MCQ.
Cette table est uniquement propre aux mots du dictionnaire MCQ : en effet, le format XML des différents fichiers n'étant pas encore standardisé, certains dictionnaires ont des informations en plus que tous les autres n'ont pas. C'est le cas du dictionnaire MCQ ici, qui est le seul à contenir des informations du type étymologique.

flexions

Cette table contient les graphies des flexions, c'est-à-dire les graphies des différentes formes possibles d'un mot issu d'un dictionnaire (ex : aardwolf (loup de terre ; insectivore hyène) dans la table headword a comme flexion aardwolves , abandon a comme flexions abandons, abandoning, et abandoned).

frequence

Cette table contient la liste des sources de fréquence (COCAE, COCAE sport, BNC, ...).

headword

Cette table contient les graphies et leurs variantes des mots issus des dictionnaires.

labellisation_morphologique

Cette table contient les labels de la décomposition morphologique d'un mot.
Par exemple, si un mot est décomposé en trois morphèmes, la labellisation de ces morphèmes peut être préfixe - base - suffixe.

lexique

Cette table contient tous les liens expliquant le lien entre deux mots : par exemple le mot displacement vient du mot displace (ex. tiré de la documentation développeur du projet actuel).
La table fait le lien entre la table liens_dict pour le mot étudié (ex : displacement) et cette même table liens_dict pour le(s) mot(s) référent(s) (ici displace). Elle est également liée avec la table liste_types_lexicaux, qui contient les différents types lexicaux (suffixé, préfixé, composé), ainsi que les tables liste_statuts (séparable, inséparable, savant, ...) et types_construction (types de construction lexicaux : juxtaposition, substitution, fusion, ...).

liens_dict

Cette table est le pilier de la base de données. En effet, elle contient les liens entre les mots des trois dictionnaires.
Elle permet également d'indiquer s'il y a des doublons de mot au sein d'un même dictionnaire, ainsi que s'il y a une catégorie présente dans deux des trois dictionnaires, mais pas dans le dernier.

liste_allomorphes

Cette table contient la liste des allomorphes (préfixe, suffixe, base), ainsi qu'un possible lien de parenté vers un morphème (ex : co- est parent de col).

liste_categories

Cette table est très importante dans la base de données : en effet, elle contient les différents types de catégorie grammaticale d'un mot (adjectif, adverbe, nom propre, abréviation, ...).

liste_morphemes_prefsuffbase

Cette table contient la liste des morphèmes (préfixe, suffixe, base). Elle est le point central des tables avoir_pour_suffixes, avoir_pour_prefixe/base, et liste_allomorphes.

liste_statuts

Cette table contient les statuts pour la table lexique (séparable, inséparable, savant, ...).

liste_types_lexicaux

Cette table contient la liste des types lexicaux possibles : suffixé, préfixé, composé, ...
Elle est en lien avec la table lexique.

liste_usages

Cette table contient les différents usages dont peu faire l'objet un mot : savant, général, discutable, ou rien.

notes

Cette table permet de recenser toutes les notes liées aux mots contenus dans la table headword.

sens

Cette table permet d'obtenir le sens associé à une graphie de mot issu d'un dictionnaire, ainsi que certaines indications (obsolète, label dans le dictionnaire, ...).

source_frequence

Cette table permet d'avoir la liste des sources de fréquence (ex : COCAE, COCAE sport, BNC, ...).

syllabes

Cette table contient la décomposition en syllabes d'un mot ou de sa version phonétique.
Les cardinalités 0,1 qui lui sont attribuées permettent de déterminer si les syllabes correspondent à un mot ou à sa version phonétique.

types_construction

Cette table contient les types de construction lexicaux possibles : juxtaposition, substitution, fusion, ...
Elle est liée avec la table lexique.

types_morphologie

Cette table contient les différents types de morphologie possibles : prefixe, base libre, suffixe lexical, ...
Elle est liée à la table labellisation_morphologique.

usage_lien

Cette table contient les usages faits d'un mot issu de la table liens_dict.
Elle fait le lien entre les tables liste_usages et liens_dict.

Code fait / pas fait

Cette annexe regroupe tout ce qui a déjà été fait au niveau du code, et ce qu'il reste à faire à ce niveau d'analyse.

F.1 Ce qui est fait

F.1.1 Projet MajBDDClient

- remplacement du nom de la base de données (bddictionnairique –> bddictionnairique2)

F.1.2 Projet ParseurXML

Package parseur.ui

- mise en place d'un message d'erreur si l'export en .csv/.txt ne se passe pas bien (données trop volumineuses par exemple)

=> VERIFIER LE BON FONCTIONNEMENT

F.1.3 Projet ProjetBddictionnairique

Package dao

- modification de toutes les classes pour correspondre aux nouvelles tables de la base de données
- création de certaines classes manquantes (DictionnairesDAO)

Package entities

- modification de toutes les classes pour correspondre aux nouvelles tables de la BDD
- création de certaines classes manquantes (Dictionnaires)

Package gestion.criteres

- modification des noms de classes importées et des noms de classes et leurs attributs utilisés

Package mditest

- modification de certaines classes et leurs attributs suite à la nouvelle BDD (FenetreGestionLiensdict, FenetreRechercheMot, FenetreRechercheMotLiensDict)

Package uipackage.consultation

- modification de certaines classes et leurs attributs suite à la nouvelle BDD (FenetreAjouterHeadword, FenetreAjouterLienDict, FenetreCategoriesMorphemesPrefsuffbase, FenetreComposes, FenetreDecompositionGraphique, FenetreDecompositionMorphologique, FenetreFlexions, FenetreFrequence, FenetreHeadword, FenetreLienDict, FenetreNotes, FenetrePrononciation, FenetreRechercheGenerale, FenetreSensEpdLpd, FenetreSensMcq, FenetreUsage)

Package uipackage.criteres

- modification de certaines classes et leurs attributs suite à la nouvelle BDD (CriteresCategorie)

F.1.4 Projet RemplissageTables

Package par défaut

- fichiers xml modifiés pour pouvoir interroger la bonne BDD et faire la correspondance avec les bonnes tables

Package remplissagetables

- classe modifiée pour prendre en compte les noms des nouvelles classes/tables en import

Package test01.entity

- modification de toutes les classes pour prendre en compte la nouvelle BDD

Le script sql structure.sql appelé par les projets MajBBDClient et MajBDDServeur à été modifié et prend maintenant en compte la structure de la nouvelle BDD

F.2 Ce qu'il reste à faire

F.2.1 Projet MajBDDServeur

- remplacement du nom de la base de données (bddictionnairique –> bddictionnairique2)

F.2.2 Projet ParseurXML

Package parseur.app

- modifier les classes et leurs attributs en fonction des nouvelles tables de BDD)

Package parseur.epd

- modifier les classes et leurs attributs en fonction des nouvelles tables de BDD)

Package parseur.lpd et parseur.mcq

- idem package parseur.epd

F.2.3 Projet ProjetBddictionnairique

Package gestion.edition

- modifier les requêtes et les classes selon la nouvelle BDD

Package jtabletest01

- voir s'il faut modifier les classes suite à la nouvelle BDD

Package mditest

- modifier les classes restantes suite à la nouvelle BDD
- finir de corriger certaines erreurs dans les classes dont les tables ont déjà été mises à jour (FenetreGestionLiensdict, FenetreRechercheMotLiensDict)

Package uipackage.consultation

- modifier les classes restantes suite à la nouvelle BDD
- finir de corriger certaines erreurs dans les classes dont les tables ont déjà été mises à jour (FenetreAjouterHeadword, FenetreAjouterLiendict, FenetreDecompositionGraphique, FenetreDecompositionMorphologique, FenetreFlexions, FenetreHeadword, FenetreLienDict)

Package uipackage.criteres

- vérifier si les classes autres que CriteresCategorie n'ont pas besoin d'être modifiées pour pouvoir correspondre à la nouvelle BDD mise en place

Package uipackage.edition

- vérifier si les classes n'ont pas besoin d'être modifiées pour pouvoir correspondre à la nouvelle BDD mise en place

F.2.4 Projet RemplissageTables

Package remplissagetables

- finir de corriger la classe RemplirTables pour qu'elle corresponde bien à la nouvelle BDD
- ajouter une méthode qui permet de connaître les dictionnaires présents en BDD et de les parcourir

Une fois la BDD remplie, il faudra générer un script sql bddictionnairique2.sql qui permet ainsi de faire une sauvegarde de la BDD et de ses données, et placer ce script dans les répertoires sqldata/mise_à_jour pour le projet Client et sqldata/base_originale pour le projet Serveur, ceci afin d'avoir une copie de la base originale avant toutes modifications effectuées par l'utilisateur.

Les sauvegardes tout au long de l'utilisation de l'application seront à placer dans le dossier /sauvegarde.

Etapes à réaliser en cas d'ajout d'un nouveau dictionnaire

En cas d'ajout d'un nouveau dictionnaire, la base de données ne sera pas à modifier. Cependant, il faudra rajouter certains éléments dans le code, notament dans le projet ParseurXML. En effet, pour l'instant chaque dictionnaire a son propre package dans ce projet (par exemple : package « parseur.epd »), car actuellement, il n'existe pas de norme sur l'édition du fichier XML, ce qui fait que chaque dictionnaire a sa propre construction XML, ses propres balises XML, ce qui n'est pas gérable pour l'instant en un seul traitement.

Dans ce même projet ParseurXML est présent un package « parseur.app », qui contient deux classes par dictionnaire.
Ces classes permettent en réalité d'appeler toutes les méthodes de traitement de chaque package propre à un dictionnaire, et de générer ainsi les fichiers .csv issus de chaque dictionnaire.
Ce package contient également la classe Main, qui permet de lancer l'exécution de ce projet.

Pour certaines exceptions, un ajout de tables peut être possible dans la base de données, comme cela est fait actuellement pour le dictionnaire MCQ, car il est le seul dans son fichier XML à contenir des informations sur l'étymologie du mot.
On peut donc voir dans la base de données une table nommée « etymo_mcq », qui est liée uniquement aux mots issus de ce dictionnaire.

Réalisation d'une base de données de dictionnaires linguistiques

Département Informatique
5e année
2010 - 2011

Rapport de Projet de Fin d'Etudes

Résumé : Ce rapport présente le projet de fin d'études que j'ai réalisé durant ma troisième année à Polytech' Tours.
Ce projet porte sur la reprise et la refonte d'une application existante et de sa base de données, application actuellement utilisée par le Laboratoire Ligérien Linguistique de l'université de Tours.

Mots clefs : Laboratoire Ligérien Linguistique de l'université de Tours, base de données, application, analyse, planning, gestion de projet

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

Keywords: Laboratoire Ligérien Linguistique of the University of Tours, database, application, analysis, schedule, project management

---

Encadrants
Claudine TACQUARD
claudine.tacquard@univ-tours.fr
Jean-Michel FOURNIER
jean-michel.fournier@univ-tours.fr

Étudiants
BACCONNET Elodie
elodie.bacconnet@etu.univ-tours.fr

DI5 2010 - 2011

Université François-Rabelais, Tours