Rapport de stage
Juillet - Octobre 2011

Ecole

SUPINFO
The International Institute of Information Technology

Stagiaire

MAITRE Cyril
Seconde Année (B2)
SUPINFO - Campus Reims

Entreprise

MAITRE Luc et Annick
Clinique Vétérinaire Gonzague
19, avenue Charles de Gaulle
08300 Rethel

Table des matières

1. Introduction

1.1. Avant Propos

Ce rapport de stage s'inscrit dans la continuité de mon stage d'été de seconde année dans la Clinique Vétérinaire Gonzague. Il a pour but de renseigner tout le travail effectué durant ces deux mois et demi et permettra, au Directeur de la Clinique, et éventuellement à de futures stagiaires, de savoir ce qui a été fait et ce qu'il reste à faire.

Ainsi je commencerai par la présentation de l'entreprise et du personnel avec qui j'ai travaillé. Puis, après une présentation générale du projet réalisé, j'en ferai le détail en expliquant les quatre étapes qui le composent. Enfin, je conclurai par une synthèse de mon expérience de ces deux mois et demi de stage.

1.2. Présentation de l'entreprise et de l'environnement de travail

La Clinique Vétérinaire Gonzague, a été créée en 1975 à Rethel. Elle a été reprise par les Docteurs MAITRE Luc et MAITRE Annick en 1990. L'entreprise a su faire preuve de rigueur et de performance pour se forger une très bonne réputation au sein de sa clientèle qui vient de toutes les Ardennes.
La Clinique est dirigée par deux vétérinaires (Dr MAITRE Luc et Dr MAITRE Annick) et emploie quatre salariés, elle fait appel à des remplaçants et accueille des stagiaires tout au long de l'année. Elle tient donc une place importante dans la santé animale Ardennaise et l'économie Rethéloise.

CLINIQUE VÉTÉRINAIRE GONZAGUE

Logo de la Clinique Vétérinaire Gonzague

Durant mon stage j'ai eu l'occasion de travailler avec plusieurs personnes.
Tout d'abord les deux Docteurs MAITRE (que je nommerai plus tard, le Directeur), avec qui j'ai composé le cahier des charges du projet et qui m'ont assisté tout au loin de ce dernier.
Ensuite avec les secrétaires de la Clinique, pour obtenir des informations ainsi que leurs avis.
Enfin, j'ai travaillé avec des personnes extérieures à l'entreprise, notamment sur internet. En effet, travaillant la plupart du temps seul, pour les questions techniques je me suis tourné vers des forums spécialisés tel que developpez.com (http://www.developpez.com/).

1.3. Présentation du projet

La Clinique Vétérinaire, comme toute bonne entreprise, possède un système informatique lui permettant de gagner en productivité. Ce système informatique est accompagné d'un logiciel de gestion, un ERP (Enterprise Resource Planning), qui permet de manager tous les éléments de la Clinique. Cela va des fiches clients, aux bons de commandes envoyés aux laboratoires, en passant bien sur par une multitude d'autres fonctionnalités tel que la gestion des factures, des stocks, des relances vaccins... etc.

Ce logiciel très important ne répond plus, à l'heure actuelle, aux attentes de la Clinique. En effet, son développement a été arrêté laissant derrière lui un logiciel bogué, manquant de fonctionnalité et peu performant.
Ces défauts sont une vraie contrainte pour la Clinique, qui entre perte de données capitales et baisse de productivité, à décidé de changer de logiciel.

C'est à ce moment que j'interviens. Mon travail étant de développer un nouvel ERP qui satisfera les attentes de la Clinique.
Pour ce faire, nous avons décidé le Directeur et moi de tout reprendre à zéro. J'ai décomposé ce projet en quatre parties: L'Analyse, La Modélisation, Le Développement et enfin Le Débogage et Optimisation.

Plus d'informations sur les Enterprise Resource Planning: http://fr.wikipedia.org/wiki/Enterprise_Resource_Planning

2. Le projet

2.1. Introduction

Comme nous l'avons vu précédemment, l'objectif de ce projet est de remplacer le logiciel de gestion que possède actuellement la Clinique.
Ce nouveau logiciel devra obligatoirement proposer certaines caractéristiques. Tous d'abord la résolution de bug, et notamment au niveau des pertes de données. Il devra également offrir les fonctionnalités déjà présentes, en améliorer certaines et en proposer de nouvelles. Enfin ce nouvel outil devra posséder une interface ergonomique, fluide et simple d'utilisation.

Pour ce faire j'ai découpé le projet en quatre grandes parties qui auront chacune une importance primordiale à sa réalisation.

2.2. L'Analyse

2.2.1. Introduction

La première partie de ce projet était l'analyse. Je savais où je devais aller, mais je ne savais ni par où, ni comment.
Cette étape était très importante, voir même la plus importante, puisque c'est sur elle que toute la suite du projet allait se reposer.

Je savais quel était mon objectif: réaliser un outil de gestion. Mais que devait-il gérer ? Comment ? Quelles étaient les fonctionnalités attendues ? Tant de question auxquelles il fallait répondre afin de réaliser un cahier des charges qui me permettrait de savoir comment modéliser le logiciel.

2.2.2. Récupération des informations

Pour réaliser cette analyse je me suis d'abord appuyé sur le logiciel existant. Je me suis muni d'un bloc-notes et d'un crayon et j'ai noté toutes les fonctionnalités qu'il proposait. J'ai appris à l'utiliser et j'ai également pris du temps à comprendre son fonctionnement en faisant du reverse-engineering.
Une fois le logiciel passer au crible, je me suis attardé sur chacune des fonctionnalités et je me suis posé la question suivante: Que pourrais-je modifier afin de l'améliorer ?
Pour m'aider j'ai demandé l'avis des employés de la Clinique. L'avis des secrétaires m'a paru très important puisque ce sont elles qui passent le plus de temps sur le logiciel.

Après cela je commençais à voir à quoi ressemblerait le logiciel et j'avais une liste détaillée des fonctionnalités qu'il devait posséder. Cependant je ne me suis pas arrêté là. Dans un souci

d'amélioration et d'évolutivité, j'ai réfléchi avec les utilisateurs du futur outil, quelles étaient les fonctionnalités que l'on pourrait ajouter.

2.2.3. Mes impressions sur l'analyse

Cette étape fut très difficile puisque rien ne devait m'échapper. Quelle serait la réaction du client si on lui disait que l'on ne peut pas lui éditer de facture car le logiciel ne le permet pas ? Des petits détails peuvent avoir de graves conséquences par la suite, et en se sens j'ai passé beaucoup de temps à vérifier que toutes les fonctionnalités étaient énumérées.
Une autre difficulté fut le filtrage des informations reçues par les employées de la Clinique. En effet il fallait faire la différence entre des informations pertinentes et moins importantes. Je me suis très vite rendu compte qu'un utilisateur avait une conception toute autre du logiciel que les personnes qui le développent. Ainsi une information que le développeur trouve très importante, peut paraitre anodine aux yeux de l'utilisateur, et vise versa. Le tout était donc de faire des compromis, trouver "le juste milieu".

2.3. La Modélisation

2.3.1. Introduction

L'analyse effectuée en début de projet m'a apportée énormément d'informations. Ces informations devaient maintenant être filtrées, triées et je devais commencer à créer des liens logiques entre chacune d'entre elles.

2.3.2. Filtrage des informations et création de liens logiques

J'ai commencé cette étape de modélisation en refaisant un filtrage des informations obtenues. Un premier filtrage avait déjà été effectué tout au long de la phase d'analyse mais cela ne suffisait pas. Il restait encore des informations superflues qui venaient entacher les autres. Pour réaliser ce second filtrage j'ai demandé de l'aide aux Docteurs Maitre afin d'avoir leur avis sur la pertinence de chacune d'elles.

Une fois le second filtrage effectué, et les informations qui allaient me servir tout au long du projet entre les mains, j'ai commencé à regrouper les fonctionnalités entre elles. Ce regroupement a permis de dessiner le début d'une certaine logique. Malheureusement, les ordinateurs actuels ne peuvent pas comprendre ce type d'information. Il faut les traduire dans un langage qui sera compris par la machine.
Cette traduction est représentée par la modélisation et la création de la base de données, qui va contenir toutes les données exploitées par le logiciel.

2.3.3. Modélisation et création de la base de données

Avant de créer la base de données, je devais d'abord la modéliser, c'est à dire définir les différentes tables et champs qui allaient la composer. Pour réaliser cette modélisation, j'ai utilisé plusieurs outils.
Le premier outil utilisé, est la méthode Merise. Cette méthode permet, à l'aide de règles simples, de modéliser la base de données proprement, c'est à dire éviter d'avoir des incohérences entre les tables, tel que la duplication de données.
Le second outil utilisé est MySQL WorkBench. C'est un logiciel qui permet de modéliser la base de donnée sur un diagramme. Ce diagramme sera ensuite utilisé pour générer automatiquement la base de données. Ce logiciel très simple d'utilisation, offre un gain de temps très important.

Logo du logiciel MySQL WorkBench

Concernant le moteur de la base de donnée, je me suis tourné vers MySQL. Ce moteur, gratuit, est très utilisé dans le domaine de la base de donnée. Il propose de très bonnes performances ainsi qu'une grande fiabilité, ce qui est en adéquation avec les attentes de la Clinique. De plus MySQL WorkBench est optimisé pour ce moteur.

Logo du Système de Gestion de Base de Données MySQL

2.3.4. Mes impressions sur la modélisation

L'étape de modélisation m'a paru très importante. En effet à l'instar de la phase d'analyse, c'est sur ce travail qu'allait se baser la suite du projet. J'ai passé beaucoup de temps à réfléchir comment je pourrais modéliser chaque fonctionnalité de la meilleure façon possible.
J'ai fait en sorte d'avoir une modélisation très modulaire, c'est à dire que je pourrais ajouter ou supprimer des éléments très facilement, et ce sans changer ce qui existe déjà.

A l'heure actuelle, la base de données contient 64 tables.

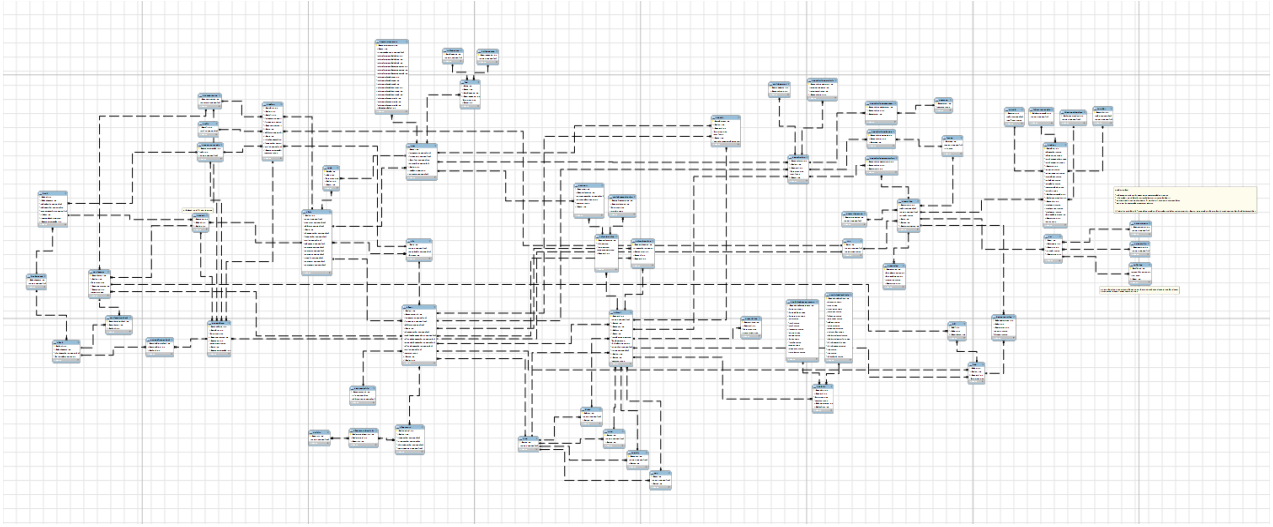


Diagramme réalisé avec MySQL WorkBench, représentant les 64 tables de la base de données

2.4. Le Développement

2.4.1. Introduction

L'étape de modélisation achevée, je pouvais commencer la phase de développement de l'application. Cette phase consistait à créer véritablement le logiciel, c'est à dire coder les différentes parties qui allaient le composer. Cela allait de l'interface utilisateur, à la gestion des données récupérées dans la base de données.
Comme nous l'avons vu dans l'introduction, le logiciel devait proposer un certain nombre de fonctionnalités, tel que la fluidité et l'ergonomie de l'interface. C'est lors de cette étape qu'allaient être mise en place ces fonctionnalités.

2.4.2. Sélection des outils utilisés

Avant de commencer à coder le programme, je devais d'abord sélectionner plusieurs outils qui me serviraient tout au long de cette étape de développement.

Le langage:
Pour coder un programme, il nous faut un langage de programmation. Il en existe énormément qui propose chacun des avantages et des inconvénients. Je devait donc sélectionner le langage qui serait selon moi le plus approprié.
Pour réaliser cette sélection j'ai utilisé plusieurs critères tel que la connaissance que j'avais pour ce langage, sa fiabilité et ses performances ou encore sa facilité à l'utiliser pour réaliser une interface graphique.
A l'issue de cette sélection, j'ai retenu le langage Java qui est très utilisé pour le type de logiciel que je devais développer et qui répond à toutes mes attentes.

Logo du langage Java

L'Environnement de Développement Intégré (IDE):
Suite à la sélection du langage Java, je devais maintenant trouver un outils permettant de développer facilement avec ce langage. Une fois de plus j'ai utilisé plusieurs critères de sélection pour trouver cet IDE.
Je me suis tourné vers IntelliJ IDEA, que j'avais déjà utilisé au cours de ma seconde année à SUPINFO. Ce nouvel IDE est en phase montante puisqu'il offre de réel avantage par rapport à la concurrence, tel que son "IntelliSense" très performant.

Logo de l'IDE IntelliJIDEA

Logiciel de gestion de version:
Pour la gestion des fichiers sources du projet j'ai choisi Subversion (SVN). J'utilise cet outil depuis plusieurs années déjà. Il permet le "versionnage" des fichiers sources d'un projet, ce qui permet en cas de problème, d'accéder à d'ancien version du programme.

Logo du système de gestion de versions SVN

Logiciel de gestion des tâches:
Afin de m'organiser, de lister les différentes tâches du projet ainsi que leurs avancements, j'ai utilisé le logiciel Acunote. Cet outil est utilisable via une interface web, et permet d'utiliser la méthode de management Scrum.

Logo de l'outil de management Acunote

2.4.3. Création d'un moteur

Une fois les outils sélectionnés, j'ai débuté le codage de l'application. J'ai commencé par coder un moteur, c'est à dire un ensemble de méthodes et d'outils permettant de manager les différentes vues du programme facilement.
Pour cela je me suis appuyé sur le modèle MVC (Model View Controller). Ce modèle préconise de diviser notre code en 3 parties. Tous d'abord le Model qui représente les données manipulées. Le model est représenté par les classes représentant les différentes tables de la base de données.
Ensuite la View dont le rôle est d'afficher les données à l'utilisateur. Concrètement cela représente l'interface graphique qui permet d'afficher des éléments à l'écran et permet à l'utilisateur d'entrer des informations.
Pour finir le Controller qui est utilisé pour faire le lien entre le Model et la View. Il va récupérer les informations fournies par le Model, et va après un certain nombre de calculs, les envoyer à la View, et vice versa.

2.4.4. Codage de différents modules

Lorsque le moteur était terminé, j'ai débuté le codage des différents modules du logiciel. Pour ce faire j'ai utilisé le moteur précédemment construit.
Le codage de chaque module suivait un schéma plus ou moins similaire.

Conception du module:
La conception du module consistait à construire une interface permettant de manipuler les données précédemment déterminées lors de la phase de modélisation.
Pour cela j'utilisais simplement du papier et un crayon et je dessinais quelques esquisses de à quoi pourrait ressembler le module.

Création du Model:
La création du model était représentée par le codage de deux classes. La première représentant la table dans la base de données. La seconde permettant au logiciel de communiquer avec la base de données.

Création de la Vue:
La création de la vue était réalisée avec un GUI Builder, c'est à dire un logiciel permettant de construire notre interface graphique. Cela permet un gain de temps considérable.
L'interface graphique était composée de divers éléments fournis par Java. Par exemple les JTextField qui permettent à l'utilisateur d'entrer du texte ou les JButton qui représentent un bouton cliquable

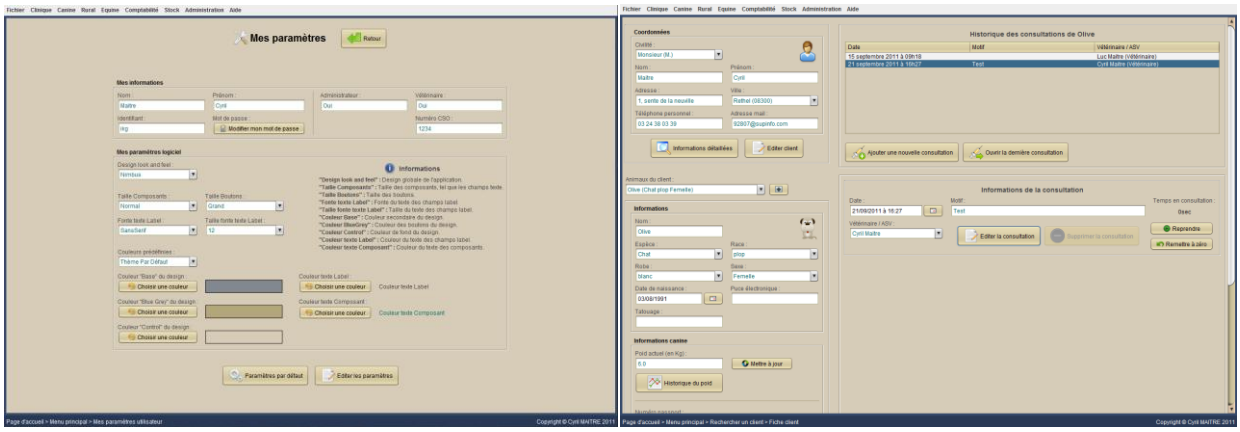
Création du Controller:
Pour finir la création du Controller. Le Controller était représenté par une classe, qui comme expliqué précédemment permet de faire le lien entre le modèle et la vue.

Le Controller possède également diverses méthodes dont le rôle est d'analyser les données transitant par elles. Par exemple vérifier qu'une adresse e-mail est valide.

2.4.5. Mes impressions sur le développement

Cette phase de développement fût très enrichissante. Elle m'a fait apprendre beaucoup de choses.
Tous d'abord au niveau informatique avec l'utilisation du langage Java et d'outils tel que IntelliJIDEA et SVN.
Mais aussi au niveau management avec l'utilisation de la méthode Scrum et l'outil Acunote.
J'ai également vu que le codage d'une application devient très vite redondant. En effet chaque module suivait un schéma presque similaire.
Enfin, une des grosse difficulté que j'ai rencontrée, concerne les droits d'utilisations. En effet j'ai utilisé des images, librairies et d'autres objets qui sont soumis à certains droits. Je devais donc faire attention à ne pas utiliser d'objets sous licence GPL ou Apache par exemple (sous peine de devoir publier le code source), mais sous licence commerciale.

A l'heure actuelle, l'application possède 35 Vues (écrans) différentes.



Screenshots de deux des 35 vues que propose le logiciel: "Mes Paramètres" et "Consultation"

2.5. Le Débogage

A l'heure où j'écris ces lignes, l'application contient prêt de 30.000 lignes de code, et on comprend facilement qu'une erreur peut rapidement s'introduire dans celles-ci. C'est pourquoi, pour chaque module, je réalisais une batterie de tests complets et poussés me permettant de voir si tous fonctionnaient comme je le souhaitais. Ces tests prennent du temps et ne sont pas particulièrement intéressants mais ils sont, selon moi, très importants. En effet, quelle serait la réaction de l'utilisateur si tous les modules étaient bogués?

Afin de m'assister dans la détection de ces bogues, j'ai utilisé un outil, Java VisualVM, qui propose le suivi de la consommation CPU et mémoire du programme au cours de son exécution. Mais aussi d'autre fonctionnalités tel qu'un "Dump" de la mémoire qui permet de vérifier les objets qui ont été créés par exemple.

Logo de la Java VisualVM

3. Conclusion

Ce stage fût très enrichissant. Tous d'abord au niveau de l'informatique puisque j'ai approfondis mes connaissances en Java, j'ai également appris à me servir de nouveaux outils tel que la Java VisualVM. J'ai aussi appris à analyser une problématique et à la modéliser de le la meilleure manière qu'il soit.
Ensuite au niveau du management, j'ai appris à m'organiser, à utiliser des outils et des méthodes de managements comme Acunote et Scrum. Je me suis aussi rendu compte de l'importance de l'organisation d'un projet, qui sans elle ne peut pas perdurer.

Même si déjà beaucoup a été réalisé au cours de ce stage, il reste encore beaucoup à faire. Plusieurs modules sont encore à développer et j'ai plusieurs idées d'ajout et d'améliorations. J'espère pouvoir continuer mon travail durant les 3 années qu'il me reste à faire à SUPINFO afin de terminer ce projet.

Cela m'apporte encore plus de motivation que j'en avais déjà pour poursuivre mes études dans le secteur informatique.

Enfin je tenais à remercier toutes les personnes avec qui j'ai travaillés et qui m'ont apporté leur aide tout au long de ces deux mois et demi de stage.