PROJET INFORMATIQUE JAVA
Gestion informatique d'un centre hospitalier

Kocupyr Vincent , Grosjean Léopold, Le Cloirec Youen

ING3 TD9

2013-2014

Sommaire

I)	Présentation	p 2
II)	Planning des taches effectuées	p 3
III)	Phase de Conception finale	p 4
IV)	Analyse détaillée des problèmes rencontrés	p 7
V)	Scénario de fonctionnement du système	p 8
VI)	Bilans	p 13
VII)	Bibliographie	p 15

I) Présentation

Dans un hôpital, la gestion du personnel et des malades n'est pas une tâche facile. En effet, il y a plusieurs bâtiments, comportant de nombreuses chambres, composés d'un nombre prédéfini de lits...etc

Comment savoir ou doivent aller les médecins et les infirmiers pour s'occuper de leurs patients ? Comment orienter un visiteur qui passe voir un ami ou un membre de sa famille ? Comment savoir si une chambre à un lit de libre afin qu'un malade puisse s'y installer ?

Grâce à ce projet, nous allons pouvoir facilement accéder a la base de donnée de l'hôpital et obtenir de nombreuses informations tels :

Quel patient est affilié à une mutuelle « X ».
Quels sont les infirmières travaillant pendant la rotation de nuit.
Ainsi que de nombreuses autres recherches qui vous faciliterons la vie en quelques clics !

Il sera aussi possible d'ajouter, supprimer et éditer toutes les informations présentes dans la base de données, créant ou supprimant ainsi des employés, des services, des malades et permettant facilement de choisir quel médecin soignera quel patient.

Le tout ce veut efficace et simple d'utilisation. Pour ce faire, chaque activité de l'application sera clairement détaillé et explicite pour que l'utilisateur puisse se retrouver facilement.

II) planning

Ordre et répartition du travail de l'aval à l'amont du projet :
L : Léopold Grosjean
Y : Youen Le cloirec
V : Vincent Kocupyr

Lecture et découverte du projet : L / Y / V

-Prendre connaissance des enjeux du projet, et de l'environnement de programmation.

Rédaction du MVC : L / Y / V

- MVC pour « Modèle Vue Contrôleur » est une technique de programmation visant à séparer l'application en trois grande catégorie. L'objectif était ici de s éparer les taches à réaliser en trois catégorie pour mieux se repérer dans la charge de travail.

Code « Connexion package » : V

-Implémentation de la connexion grâce au fichier fournis par Mr. Segado. Ajustement de ces dernier pour mettre en place une connexion local, ainsi qu'un objet unique d'instance utilisé par la suite pour les DAO

Code « BD Class » : L

-Création des classes Objet reprenant les caractéristiques des différentes table de la base de donnée pour travailler sur ses objets. Nous éviterons ainsi les mauvaises manipulation SQL par l'utilisateur.

Code « DAO package » : V

-Implémentation des classes DAO : « Data Access Object » ces objets sont une surcouche faisant le lien entre les objets java de l'application, et les données en BD. C'est ici que sont enregistré toute les requêtes qui seront utilisés.

Rédaction de la Phase de conception : L / Y / V

-Rédaction du premier dossier compte-rendu. Ce dernier s'attarde sur l'explication de la méthode de développement voulu : MVC système de DAO.

Code Interface graphique « Hôpital package » : Y

-Création de l'interface graphique en fonction de notre ébauche papier lors de la phase de conception. Elle n'est pas terminé, et utilise plusieurs fenêtres.

Code « table package » : L / V

-Implémentation des objets manipulé dans un modèle de tableau pouvant ainsi être manipulé plus facilement par l'utilisateur. Ce dernier permet le trie automatique, la suppression simplifié, ou l'édition simplifié.

Correction des erreurs de code I : L / Y / V

-Première phase de correction. Axé sur les fonctions utilisé pour le travail sur les données afin de gérer les exceptions.

Amélioration Interface graphique « Hôpital package » : L / Y

-Amélioration de l'interface graphique. Tout fonctionne maintenant sur une seule fenêtre, la mise en page est plus clair, des couleurs et un fond d'écran ont été rajouté, et des messages d'erreur en cas de mauvaise manipulation.

Code « JOB package » : L / V

-JOB package est une couche supplémentaire au DAO qui permet d’appeler à bon escient les différentes fonctions de traitement des données. Vérifie si le traitement est valide.

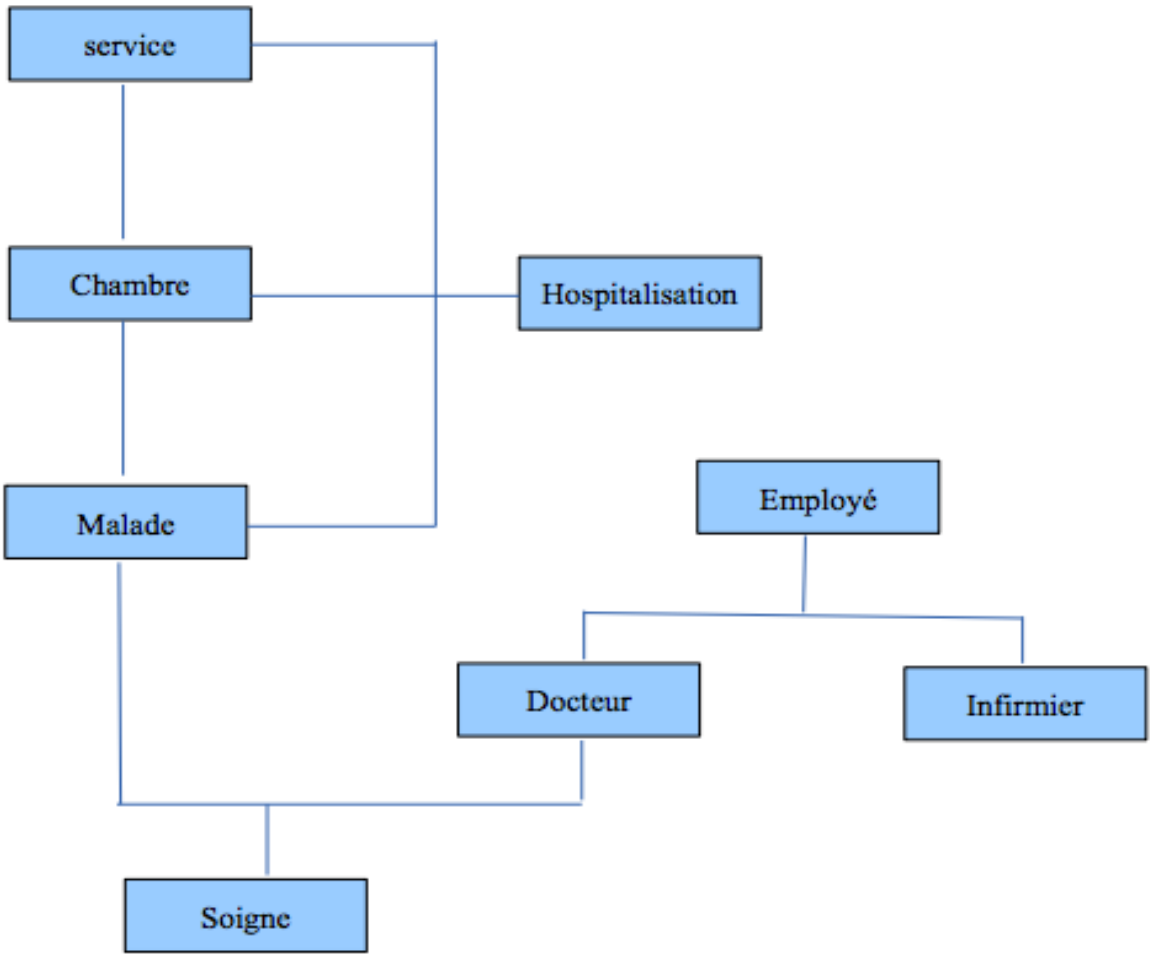
Correction des erreurs de code II : L / Y / V

-Correction des dernières erreur. Notamment sur les erreurs provenant après la compilation du fichier jar.

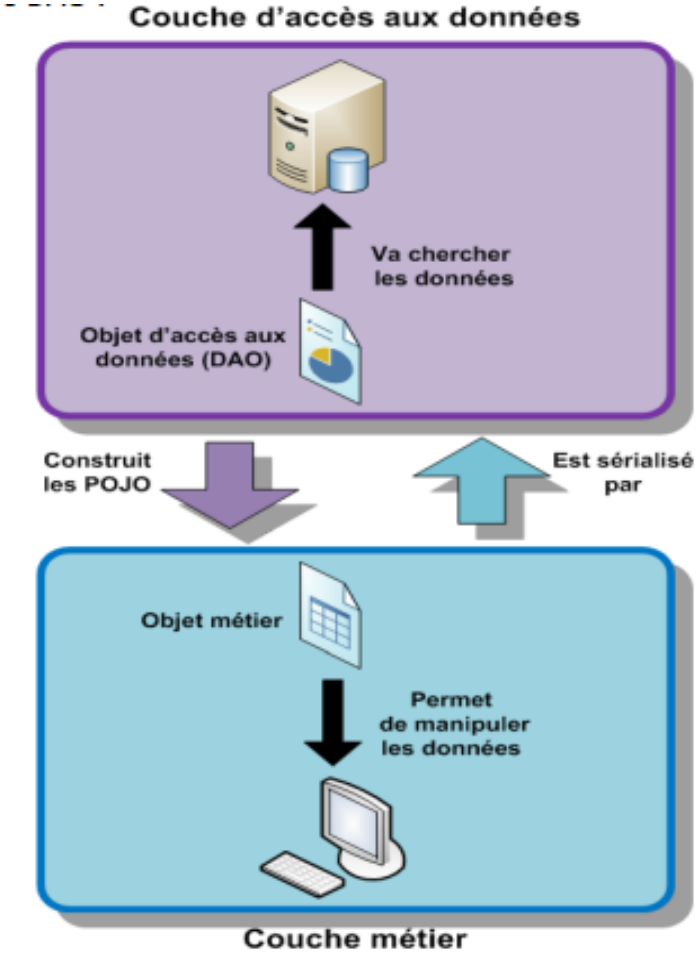
Rédaction du Rapport Final : L / Y / V

III) Phase de conception finale

La base de données est l'élément central de l'application. Cette dernière, stocké sur un serveur SQL permettra d'obtenir toutes les informations relative à l'hôpital. Voici son schéma :



La pratique de connexion – requête est une notion simple, mais dangereuse pour le logiciel. Il est plus intéressant, pour le principe de MVC, et la sécurité du programme, de mettre en place une connexion à la base de données sur la méthode DAO.
La méthode DAO : "Data Access Object" bien que plus fastidieuse à réaliser, permet de projeter la base de données. En effet, celle ci n'est pas traité directement par l'utilisateur avec des requêtes, mais par des fonctions stockées dans des classes supplémentaires faisant office de lien entre l'utilisateur et la base de données.
Schéma d'explication des DAO :



Ainsi, on dissociera 3 Types de classe :
-Les classes de vue : Pour les fenêtres et les boutons.
-Les classes de contrôle : Les DAO utiliser pour les demandes de l'utilisateur
-Les classes du modèle : Les classes objets reprenant la base de données

Nous allons maintenant nous attarder en détail sur chacune des classes identifiés :

Les classes de contrôle :

Nous avons créés trois types de modules qui permettent de lire ou écrire dans la base de données , ces derniers sont :

- Module de Recherche d'informations
- Module d'édition et de suppression
- Module d'Ajout des données

-Le Module de Recherche d'informations est la partie du logiciel qui effectuera les requêtes de type recherches. Son objectif principal est de recueillir afin de se renseigner sur certains point du centre hospitalier. Les résultats sont affichés dans un tableau variable grâce à des modèles pré-enregistrés pour chaque table de la BD.

-Le Module d'édition et de Suppression est directement lié avec le module de Recherche d'informations. Une fois les résultats de la recherche affiché, nous pouvons sélectionner un objet du résultat, et choisir de le modifier ou de le supprimer. Attention toutefois, l'édition et la suppression n'est pas valable sur tout les objets ! En fonction des liens existant entre les objets certain ne peuvent pas être supprimé au risque de casser ses liens et corrompre la BD. La fonction permettant la validité du traitement est aussi dans ce module.

-Le Module d'Ajout des données est indépendant au deux autres. Celui ci permet l'ajout dans la BD de n'importe qu'elle élément dans les tables. Cependant certaines contrainte existe. Pour éviter au plus possible erreurs. Les identifiants unique de chaque table n'est pas rentré par l'utilisateur, mais sont écrie automatiquement par la fonction pour être sur que l'identifiant soit bien unique. (la méthode utilisé consiste à donner au nouvelle objet un identifiant = valeur de l'identifiant maximum existant+1).

Pour effectuer nos requêtes, le plus important est d'avoir une construction facile permettant un accès aux données sans risque. Pour un fonctionnement rapide et sans erreur, nous avons mis en place une couche de type DAO ("Data Access Object") permettant un accès au source des données afin que l'utilisateur lui même y accède. Ainsi, pour chaque classe de la Base de données, une classe DAO existe permettant de faire toute sorte de manipulation sur l'objet.

Les classes de de Vue :

Nous avons 3 type de classe différentes pour la vue qui permettent le meilleur affichage possible de l'application :

-Les différentes fenêtre
-La Gestion de la fenêtre unique
-Le Tableau

-Les différentes fenêtre correspondent à toutes les fenêtre que l'ont peut voir dans l'application, ce qui comprend : La connexion, le menu, l'interface de recherche, le tableau de résultat, l'interface d'ajout, le menu d'historisation, l'interface des rendez-vous, l'interface des opérations. Chaque fenêtre à son propre comportement en fonction de quelle table est en cours de traitement. Pour ce faire nous utilisons une variable dans un switch qui permet de différencier le traitement des 8 tables différentes de la BD.

-La Gestion de la fenêtre unique fonctionne sur le principe du trie de carte. L'application tourne avec une seule fenêtre, et contient différents panneau qui correspondent à toutes les fenêtre de l'application. À chaque changement de fenêtre, on change la carte (le panneau) afficher dans la fenêtre avec celui voulu, que l'on récupère à l'aide d'un identifiant nominatif.

-Le Tableau à une gestion bien différentes des autres fenêtres. Ce dernier doit s'adapter à la taille en fonction du nombre d'élément, et doit pouvoir changer le nombre de colonne et leur nom en fonction de l'affichage de table voulu. Comme expliqué au dessus, chaque table de la BD à son propre modèle de tableau, stocké dans le package « tablepackage » qui permet d'identifier comment doit être le tableau que nous allons afficher.

Remarque : C'est suite à l'implémentation du tableau et la compréhension de la classe AbstractTableModel que nous avons décidé de mettre en place les modules d'édition et de suppression dans cette fenêtre. Les fonctions fournis par cette classe nous permettent de récupérer facilement les valeurs sélectionnés dans le tableau, permettant d'identifier plus rapidement quel élément doit être modifié ou supprimé.

Les classes de de Modèle :

Les classes de modèle sont des implémentations de chaque table de la BD dans des objets Java qui seront instanciés. Ainsi notre modèle correspond fidèlement à la BD, et même si devons changer de langage ou de plate-forme de programmation, son intégrité restera la même, favorisant donc la sécurité de l'application.

IV) Analyse détaillée des problèmes rencontrés

Le plus gros problème que l'on a rencontré au cours de ce projet est la gestion du temps.

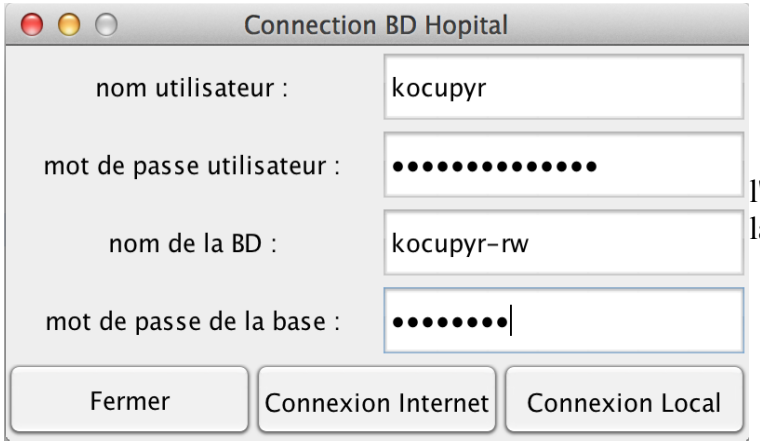
-Pour commencer, nous nous sommes trop attardés à améliorer ce qui marchait déjà en sous-estimant le temps que cela allait prendre: la mise en page des différentes fenêtres du logiciel reste longue et fastidieuse selon le souhait de la disposition. Bien entendu, le BorderLayout est intéressant mais lorsqu'on souhaite avoir un nombre de Jlabel , Jbutton, JTextField .. plus important, le fait de dimensionner un à un ses éléments reste plus intéressant pour l'esthétisme mais est plus long.

-Nous avons également rencontré des problèmes de portabilité. En effet deux d'entre nous codant sur Macbook et l'autre sur Windows avec en plus des versions différentes de NetBeans à posé quelques soucis. Ce qui fut une perte de temps mais aussi de l'apprentissage sur les problèmes auxquels sont confrontés les professionnels.

-Lors de la conception de l'application, nous avons rencontrer des problèmes lors de la connexion. Le fonctionnement en DAO demande une instance de connexion à chaque création. Or, l'objet connexion était initialement bloqué dans la fenêtre de connexion sans portabilité sur le reste de l'application. Il a fallu faire un élément static en singleton afin de palier ce soucie.

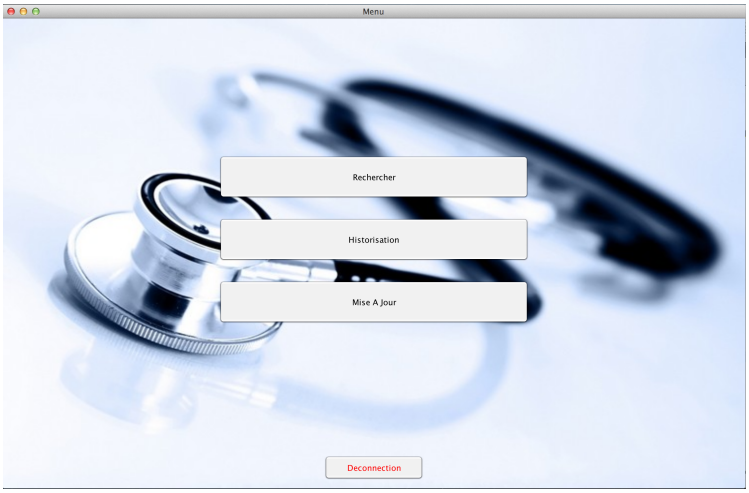
-Sur la finalisation, lors de l'export de l'application en fichier exécutable .jar, l'image de fond ne s'affichait par, car cette dernière n'était pas inclus dans le projet. Pour résoudre ce problème, nous avons ajouter l'image dans un package « lib » inclus dans le .jar, et cette dernière est appelé en InputStream depuis sont positionnement dans une classe vide, permettant uniquement l'accès au chemin de l'image.

V) Scénario de fonctionnement du système

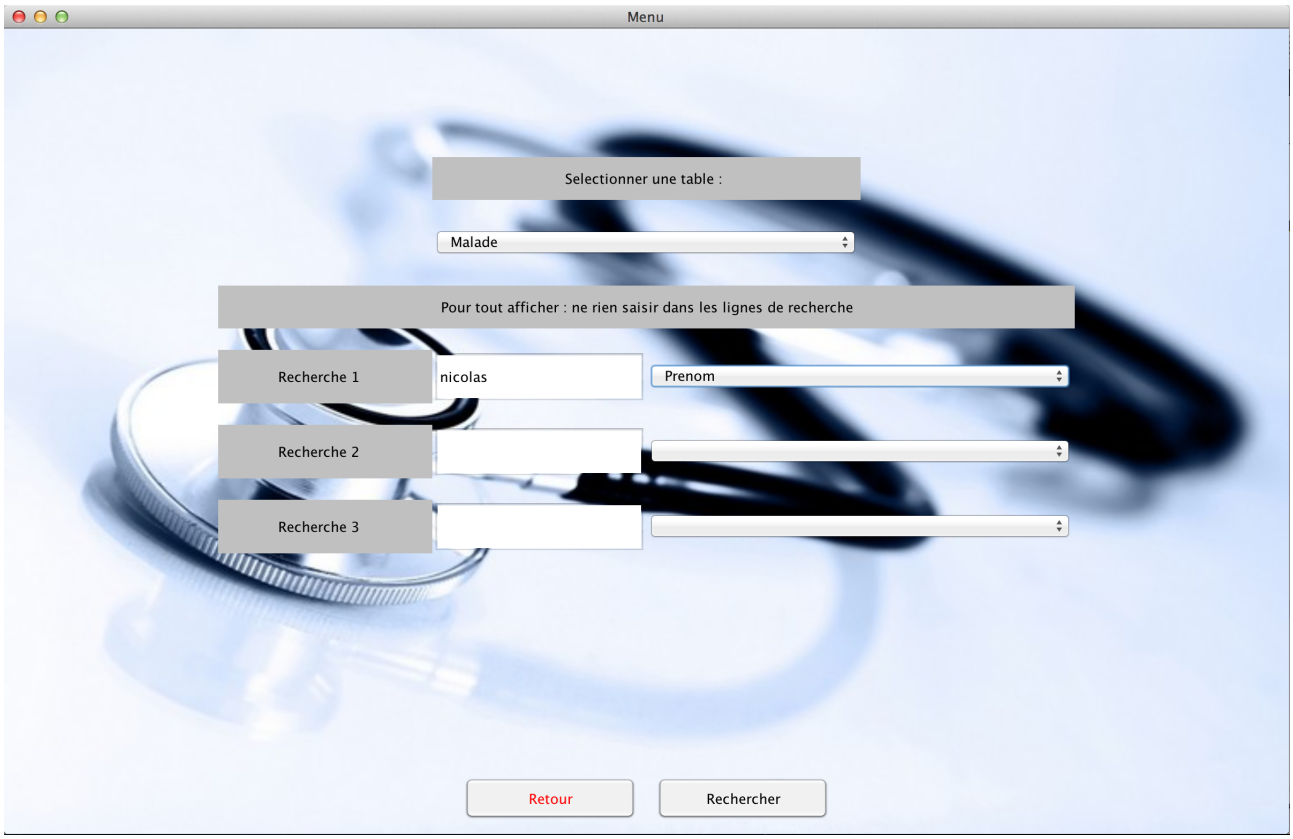


On commence le scénario par l'ouverture d'une page de connexion à la base de données .

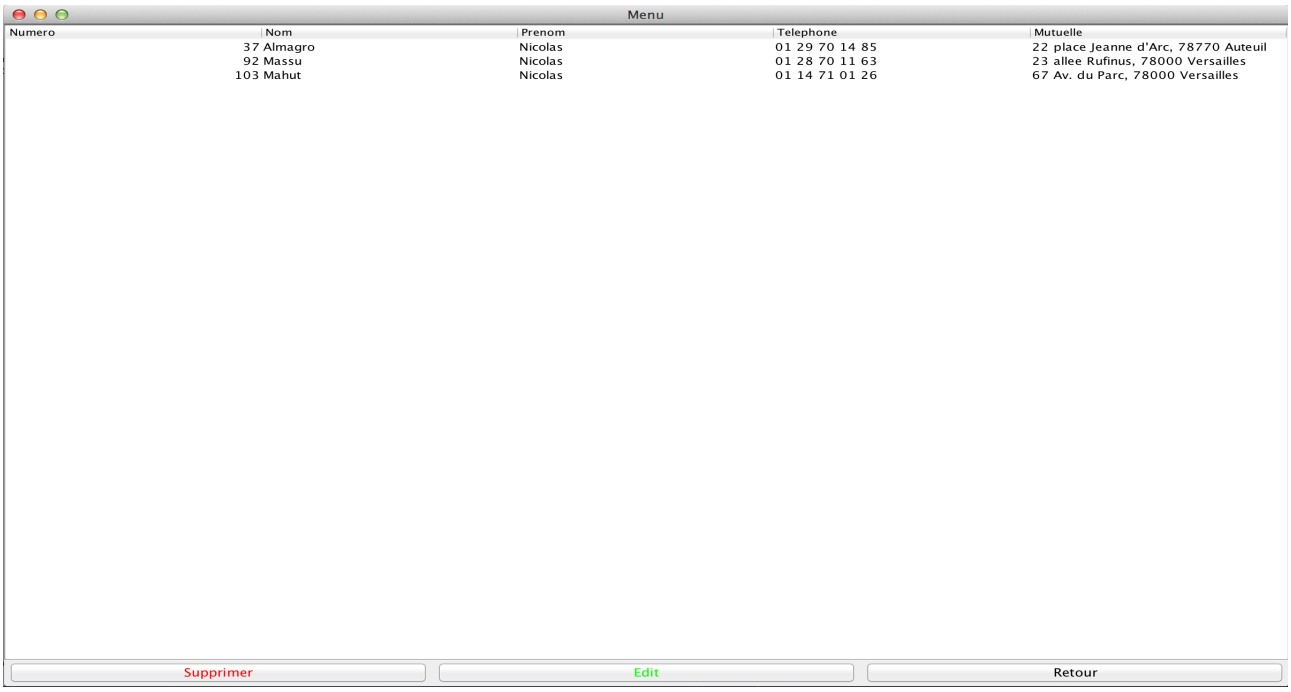
Ce qui nous renvoie directement sur la page d'accueil de l'application .
Il suffit de cliquer sur « Continuer » pour rejoindre la page d'accueil .



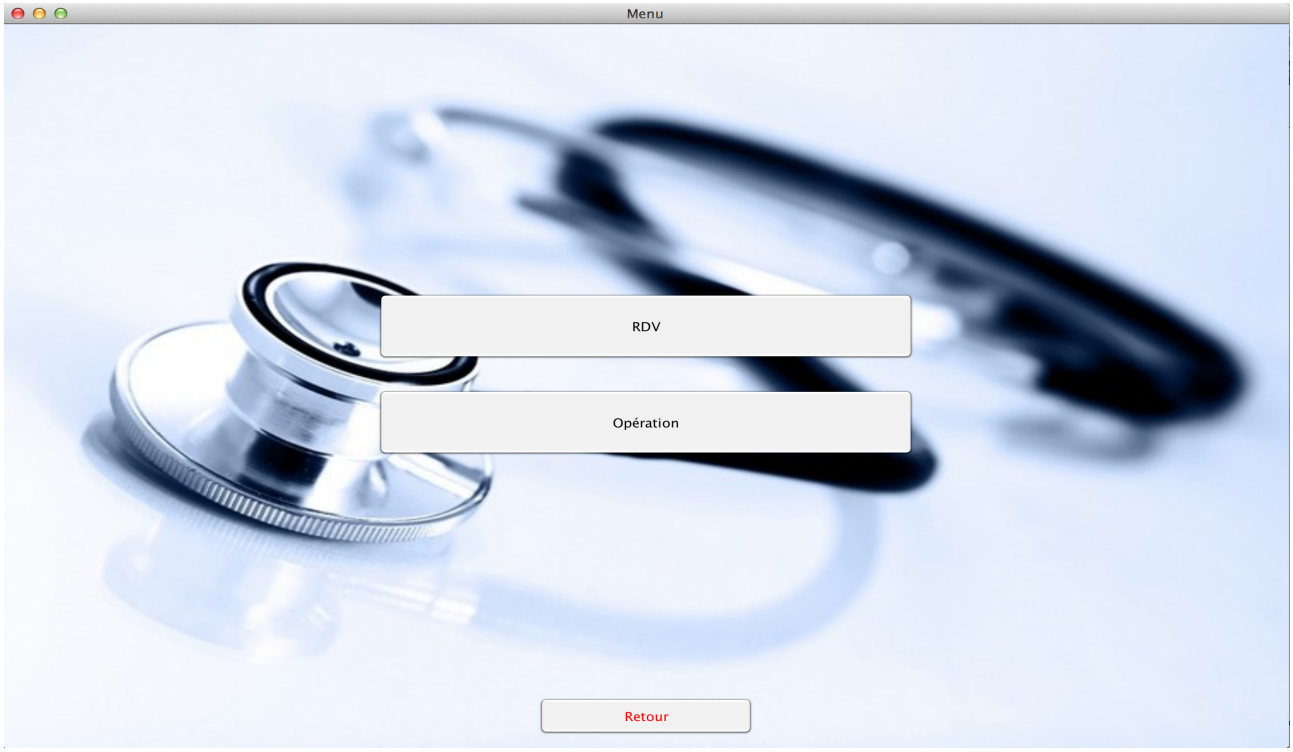
Cette page d'accueil nous propose trois fonctions : Rechercher, Historisation, Mise à Jour. (renommer ajouter par la suite)
Dans un premier temps, nous allons effectuer une recherche.



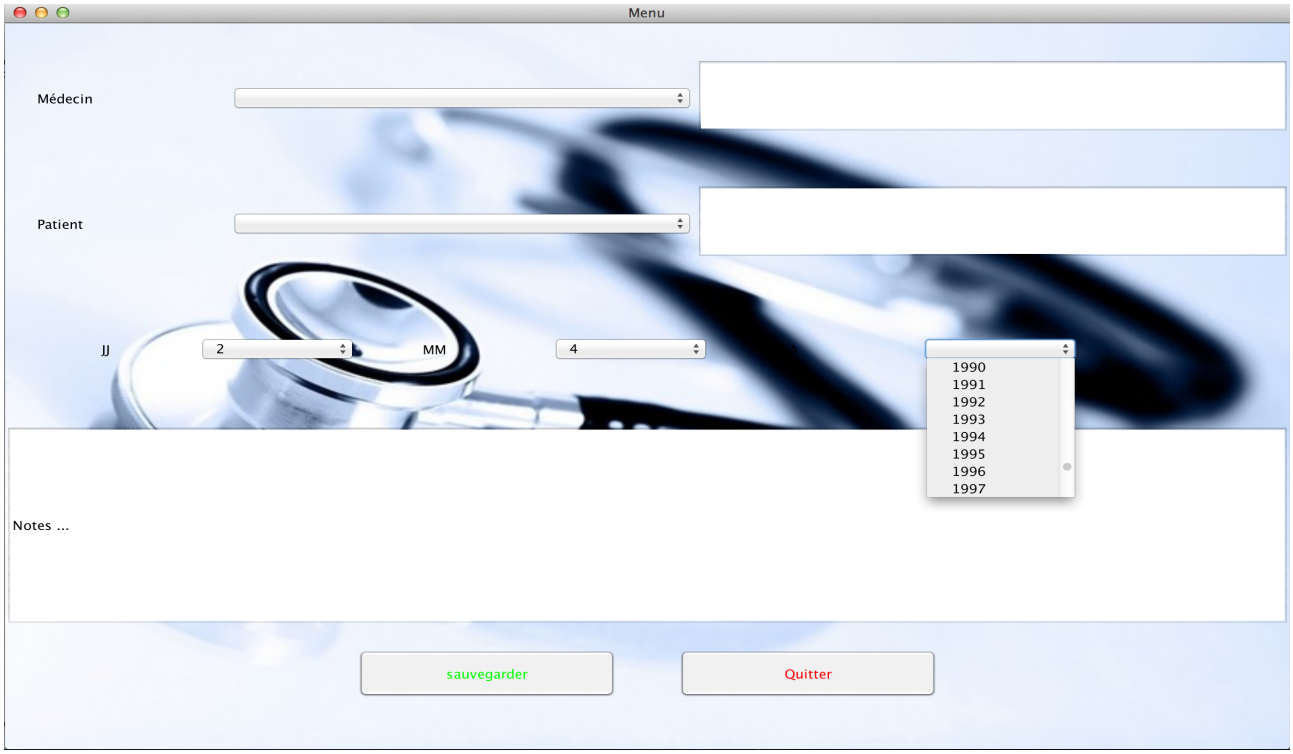
La fonction « Recherche » nous amène sur une page où l'on peut sélectionner une table (nous avons le choix entre : Docteur, Malade, Chambre, Employé, Infirmier, Service, Hospitalisation, Soigne). Ensuite, on peut effectuer la recherche suivant plusieurs critères. on peut ne rien saisir pour afficher toute la table souhaitée ou alors effectuer une recherche comme ici où l'on connaît le prénom du malade pour ainsi réduire le nombre de personne à l'affichage. Cette recherche nous amènera au résultat suivant :



Nous allons maintenant utiliser la fonction Historisation qui permet soit de créer un rendez-vous ou alors de créer une opération :



Nous allons ici créer un rendez-vous, on sélectionne un médecin, un patient déjà inscrit, la date du rendez-vous et on peut y inscrire des notes auxquelles le médecin aura accès.



La fonction Opération fonctionne de la même manière que « Rendez-vous ».

La dernière fonction auquel donne accès le menu est Ajouter. Cette fonction permet comme son nom l'indique de d'ajouter un élément dans n'importe quelle table sélectionnée dans une liste.

Il suffit de sélectionner une table (l'affichage apparaît selon la table sélectionnée) puis de remplir les informations voulu. Si l'ajout n'est pas bon, un message d'erreur apparaît (ici nous avons oublier de saisir un code de service, qui est pourtant obligatoire pour cette table en BD)

VI) Bilan

1) Bilan individuel

LE CLOIREC Youen :

Ce projet m'a permis de faire une corrélation entre les bases de données et le JAVA, ce qui auparavant me semblais plutôt abstrait. Cela m'a également apporté une connaissance en SWING que je trouve très intéressante : ce qui me pousse à vouloir faire mon sujet de PPE autour des applications mobiles.
Ce projet m'a montré à quel point l'informatique était et sera présent dans la vie de tous les jours.
De plus, le fait de travailler en équipe me prépare à mon futur métier d'ingénieur.
Cependant, le seul bémol, selon moi, fut que ce projet ne soit pas synchronisé avec les cours de SWING en cours et en TP...

GROSJEAN Léopold :

J'avais déjà eu la chance au premier semestre de travailler avec Vincent sur le « Snake », notre premier projet d'info de l'année, qui fut un vrais succès. Nous avons d'un commun accord décidé de réitérer l'expérience en incluant Youen, un ing3 nouveaux sérieux et motivé.
Pour ce projet, j'appréhendais le fait de se connecter en ligne à sql.ece.fr pour gérer « notre » BDD mais cela ne nous a, finalement, pas posé de soucis.
Grâce à une réelle motivation et une implication dépassant mes espérances, j'ai grâce a ce projet développé exponentiellement mes connaissances en Java et « NetBeans », logiciel n'ayant, presque, plus de secrets pour moi.
J'ai choisi de faire SI l'année prochaine, j'aspire à continuer de progresser en code afin de pouvoir développer des applications sur smartphones et tablettes, « Swing » m'a en effet ouvert les yeux sur l'interface graphique. Peut-être sortirai-je le prochain « Flappy Bird », en attendant le succès, j'essayerai de trouver du temps pendant mon stage pour continuer et améliorer ce projet : j'aimerai en effet mettre en place dans la classe «Ajout» un algorithme permettant d'afficher selon le numéro sélectionné, le nom de l'employé ou du malade dans un Jlabel.

KOCUPYR Vincent :

Ayant déjà travaillé sur un projet similaire à l'IUT, j'ai abordé celui ci comme un complément dans ma formation. Ce que je trouve dommage est le manque de temps passé en cours ou TD sur le visuel (swing et awt) car le travail sur la base de donnée est très redondant... Mais je pense qu'une fois les librairies swing et awt totalement maîtrisés, l'application aurait pu avoir un aspect encore plus réussie. Je regrette aussi de pas avoir eu le temps de tout terminer comme je l'imaginais, mais cela ne m'étonne plus maintenant d'entendre que 70 % des projets informatiques n'aboutissent pas ou sont retardés. Il faut persévérer et poursuivre le travail afin de réaliser des projets de plus en plus abouti !

2) Bilan Collectif

Au final ce projet fut l'un des plus compliqué que nous ayons eu à réaliser à l'ECE. Ce dernier utiliser toute les techniques vue en cours de programmation JAVA, et même plus pour aller plus loin. Le plus difficile fut la gestion du temps et des taches prioritaires. Mais une fois le travail lancé, celui ci c'est effectué de manière limpide jour après jour.
Nous pouvons tout de même regretter de ne pas avoir réaliser une application « parfaite » avec toutes les fonctionnalités opérationnelles. Mais au lieu de vouloir faire le plus possible moyennement, nous avons focalisé nos efforts à améliorer et rendre plus stable ce que nous avions. Néanmoins la parti graphique étant réaliser sur les fonctions non opérationnel, cela peut très bien servir de démo visuel pour une présentation à un potentiel acheteur si nous nous plaçons dans le rôle du vendeur.

Nous pouvons tous les trois affirmer que ce fut un travail passionnant, demandant beaucoup d'énergie, mais qui en valait la peine, spécialement pour vivre le moment ou l'on voit son application fonctionner pour la première fois, et ce rendre compte que nous avons réaliser quelque chose qui pourrait servir. Le travail porte ses fruits !

VII) Bibliographie

http://stackoverflow.com/

http://fr.openclassrooms.com/

http://docs.oracle.com/javase/7/docs/api/

http://en.wikipedia.org