République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études
Pour l'obtention du diplôme de Master en Informatique

Option: Génie Logiciel (G.L.)

Thème

Réalisation d'une plateforme de gestion des patients en pré-greffe rénale avec Struts et EJB

Réalisé par :

- **Meriem ZEGGAI**
- **Selma SEBIANE**

Présenté le 22 Juin 2016 devant le jury composé de MM.

- *S.M. CHOUITI* *(Président)*
- *Yassamine SELADJI* *(Encadrant)*
- *H. MAHFOUD* *(Examinateur)*
- *F. BENMANSOUR* *(Examinatrice)*
- *Mahfoud CHERIF BENMOUSSA* *(Co-encadrant)*

Année universitaire: 2015-2016

REMERCIEMENTS

Nous tenons tout d’abord à remercier notre encadrant Mme Yassamine SELADJI pour le temps qu’elle nous a consacré ainsi que ses encouragements. Merci de nous avoir accompagnés tout au long de ce projet avec beaucoup de rigueur scientifique et de précieux conseils et remarques qui nous ont permis de réaliser ce travail.

L’aboutissement de ce projet n’aurait pu se faire sans l’aide d’un néphrologue. Nos remerciements s’adressent ici à Mr. Mahfoud CHERIF BENMOUSSA. Malgré ses multiples responsabilités, il nous a accordé de son temps. Sans les détails qu’il nous a fournis et ses précieuses remarques, ce projet n’aurait jamais pu aboutir à terme. Nous vous remercions infiniment pour votre collaboration et votre disponibilité.

Nous remercions les membres du jury d’avoir pris le temps de lire et d’évaluer notre travail.

Nous remercions également les enseignants qui nous ont suivis tout au long de notre formation, et plus particulièrement Mrs. Mohamed MESSABIHI et Mohammed TADLAOUI pour leurs précieux conseils et les connaissances dont ils nous ont fait part tout au long de notre cursus.

Nous ne saurons remercier assez nos parents. Vous trouverez ici le résultat de longues années de sacrifices et de privations. Merci pour les valeurs nobles, l'éducation et le soutien permanent venu de vous.

Nous exprimons aussi notre gratitude envers tous ceux qui nous ont accordé leur soutien, tant par leur gentillesse que par leur dévouement.

Nous ne pouvons nommer ici toutes les personnes qui, de près ou de loin, nous ont aidé et encouragé mais nous les en remercions vivement.

TABLE DES MATIERES

1.INTRODUCTION GENERALE

Le phénomène informatique qui n'est plus exclusivement un outil de bureautique, s'est progressivement et rapidement répandu dans tous les domaines, notamment dans le domaine de la santé publique, où son utilisation a été longtemps appréhendée.

De nos jours, dans le milieu de la santé, l'informatique prend petit à petit une place importante et grandissante, ainsi la gestion des opérations hospitalières devient beaucoup plus sûre et efficace ce qui a suscité l'intérêt d'un bon nombre d'établissements pour cette informatique plus « moderne ».

Nous nous intéressons au service de néphrologie du centre hospitalo-universitaire de Tlemcen. Le service de néphrologie s'intéresse aux maladies des reins et des techniques de suppléances quand les reins ne fonctionnent plus en proposant la dialyse et la transplantation rénale. Le service est organisé en unités dont l'unité de transplantation, où ils prennent en charge les personnes qui veulent se faire greffer et les greffés. La greffe rénale est une méthode de choix pour palier à l'insuffisance rénale. Le processus de don est un long processus où le médecin doit établir un ensemble d'examens pour s'assurer de la faisabilité de la transplantation. Ces examens doivent être rigoureux, répétés et enregistrés; la faisabilité et la sécurité du don rénal pour une personne saine, et la réussite de la transplantation chez le malade exige une analyse rigoureuse d'un semble important de données.

Le service de néphrologie accueille toute personne désireuse de se faire greffer. L'épidémiologie de l'insuffisance rénale est très importante ; la demande de soin et l'exigence se développent. Leur gestion est sensible par le nombre important de paramètres à traiter. Une plateforme informatisée prenant en compte toutes les données de manière systématisée améliorerait la rentabilité de la prise en charge et une gestion efficace des dossiers réduirait le risque d'erreur.

C'est donc en essayant de rénover la gestion de l'unité de greffe rénale que l'informatisation de la pratique médicale s'impose ; l'instauration d'une base de données numérique est une nécessité exprimée par une équipe médicale. Ce système accompagne le néphrologue dans ses pratiques quotidiennes pour améliorer ses services aux bons soins des patients souffrant d'insuffisance rénale. Le système est nommé « Greffon », terme qui signifie le rein qui est greffé. L'application qui enregistre tous les patients, futurs receveurs de rein et les éventuels donneurs. Cette dernière est une version numérisée de tout le dossier médical.

Nous présenterons dans le premier chapitre le contexte de travail à savoir le CHU de Tlemcen, le service de néphrologie et particulièrement l'unité de greffe rénale. Nous expliquerons plus en détail le processus de travail dans cette unité.

Quant au deuxième chapitre, il présente le processus employé par le corps médical de l'unité de greffe rénale afin de gérer tous les dossiers des patients, les besoins dérivés de l'étude de l'existant et ce qui pousse les médecins au besoin de l'informatisation de leur système de gestion.

Le troisième chapitre donne un aperçu détaillé de la conception du système conçu, « Greffon », ainsi que les différents outils utilisés.

Le quatrième et dernier chapitre présente les outils qui ont servi au développement de l'application ainsi que la réalisation de « Greffon » et expose quelques interfaces du système conçu.

Enfin, nous finirons par une conclusion en mettant le point sur l'importance des technologies utilisées.

2. CONTEXTE DE TRAVAIL

2.1 Introduction

L'unité de greffe rénale du service de néphrologie au niveau du CHU de Tlemcen nécessite une gestion optimale des dossiers des patients ce qui a engendré le besoin de développer une application permettant de répondre à ce dernier. Pour mieux comprendre ce besoin, on présente dans ce chapitre le service de néphrologie ainsi que le processus de fonctionnement de leur unité de greffe.

2.2 Présentation du CHU de Tlemcen

Le centre Hospitalo-universitaire de Tlemcen a été construit dans les années 1950. Il prend le nom du docteur TIDJANI DAMERDJI, médecin, patriote de la 1ère heure, martyr de la révolution algérienne, tombé au champ d'honneur le 17 avril 1957. Il est composé de 44 services et laboratoires spécialisés et dispose d'une capacité d'accueil de 646 lits.

2.3 Service de néphrologie

Le centre hospitalo-universitaire de Tlemcen contient un service de néphrologie sous la direction du Professeur Benmansour.

Ce service est composé de quatre unités principales : hospitalisation, consultation et dialyse péritonéale, greffe rénale et l'unité d'hémodialyse.

Notre sujet de travail porte sur l'unité de greffe rénale.

La néphrologie est la spécialité médicale qui concerne le diagnostic et le traitement de l'ensemble des pathologies humaines touchant le rein, jouant un rôle fondamental dans la filtration du sang.

2.3.1 Insuffisance rénale

Les reins sont des machines à épurer très sophistiquées capables de filtrer jusqu'à 170 litres de sang par jour. En filtrant le sang, ils produisent l'urine.

Outre la filtration du sang et la production d'urine, les reins remplissent également d'autres fonctions physiologiques importantes comme le maintien de la teneur en eau et en sel du corps à un niveau d'équilibre. [1]

L'insuffisance rénale est diagnostiquée seulement lorsque les trois quarts des néphrons, unités fonctionnelles situées dans le rein qui filtrent, sont perdus.

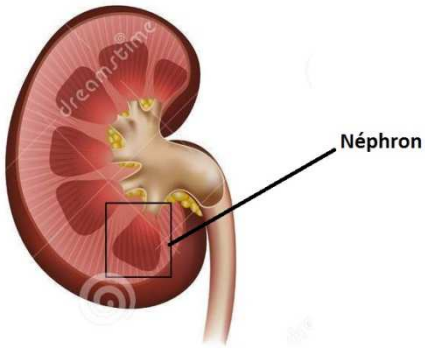


Figure 2.1 : Schéma d'un rein

La figure 2.1 représente un rein indiquant l'emplacement d'un néphron.

On distingue deux types d'insuffisance rénale : aigüe et chronique.

C'est à la détérioration graduelle et irréversible que l'insuffisance rénale est définie comme chronique. L'insuffisance rénale chronique évolue fréquemment vers une perte totale de la fonction rénale ; un traitement de substitution de la fonction rénale, la dialyse ou la transplantation, devient alors nécessaire.

Dans un premier temps, la dialyse péritonéale, dialyse sans machines qui permet de filtrer le sang des personnes dont les reins ne fonctionnent plus correctement, est utilisée comme traitement.

Si le patient ne répond pas à ce traitement intervient l'unité d'hémodialyse. Cette unité est munie de machines pour épurer le sang du patient à reins défaillants. Le sang du patient est envoyé par un petit tuyau vers le rein artificiel, qui renferme les membranes de filtration et la solution de dialyse. Le sang, une fois filtré, est renvoyé vers le patient. [1]

La figure 2.2 représente le processus d'hémodialyse.

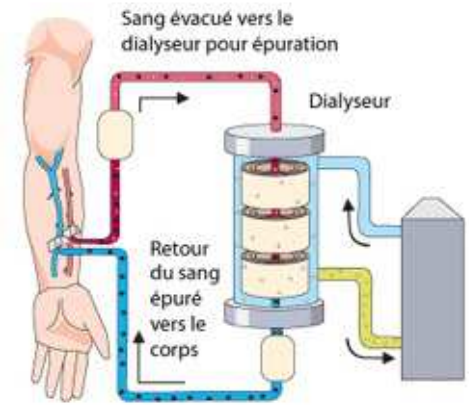


Figure 2.2 : Schéma du processus d'une hémodialyse

Si le patient ne répond plus aux séances d'hémodialyse, une greffe du rein s'impose.

2.3.2 Unité de greffe rénale

L'unité de greffe rénale gère la transplantation des reins qui consiste à placer chez le receveur, qui est en phase terminale d'insuffisance rénale chronique, un rein sain, provenant d'un donneur vivant, placé dans l'abdomen par le biais d'une opération chirurgicale.

Dans cette unité, le travail est partagé en deux sections: la section de pré-greffe et la section de post-greffe rénale.

Nous nous intéressons à la section de pré-greffe rénale où un bilan de santé physique et psychologique complet du receveur est effectué par le néphrologue afin de s'assurer de l'absence de contre-indication absolue ou temporaire à la transplantation. Une fois accepté, le patient pourra bénéficier d'une greffe rénale en provenance d'un donneur vivant.

Une liste d'attente est mise en place par les néphrologues. Elle contient les patients qui doivent effectuer une transplantation rénale. L'attente peut s'avérer longue et le renouvellement des bilans de santé peut être nécessaire.

Le receveur devra présenter son donneur qui sera aussi soumis à un bilan de santé en tous genres.

Suite à ces examens, le néphrologue va trancher sur la compatibilité entre le donneur et le receveur ; si elle est positive, le receveur sera apte à effectuer la transplantation.

Afin de limiter les risques de rejet, les néphrologues essaient de greffer des reins les plus compatibles possibles, c'est pour cela que le donneur doit obligatoirement être un membre de la même famille.

La chirurgie de la transplantation rénale dure de 2 à 3 heures. Le receveur et donneur sont opérés en même temps, par deux équipes différentes, dans deux salles d'opération voisines; le rein est de meilleure qualité car il est greffé immédiatement après le prélèvement.

Après la chirurgie, le receveur en post-greffe, passe environ 48h dans le service des soins intensifs avant d'être transféré dans celui de transplantation où son séjour sera d'en moyenne 15 jours.

Le receveur va être surveillé en cas de rejet du greffon, rein greffé. Ceci peut se manifester dans les heures qui suivent la chirurgie de transplantation.

Le donneur est aussi mis sous surveillance, et il pourra reprendre le cours de ses activités 3 à 4 semaine après l'intervention chirurgicale.

2.4 Conclusion

L'étude du milieu dans lequel travaillent les néphrologues est essentielle à la bonne compréhension des services attendus et des problèmes rencontrés. Cela va permettre d'extraire les besoins essentiels afin de concevoir un produit satisfaisant.

3.ANALYSE DES BESOINS

3.1 Introduction

L'étude du contexte de travail nous a permis de comprendre les difficultés rencontrées par les néphrologues. Nous présentons dans ce chapitre les principales problématiques ainsi que les besoins qui vont nous permettre de placer les services attendus en priorité de notre travail.

3.2 Etude de l'existant

Ce projet est une collaboration entre la faculté des sciences de l'université Abou Bakr Belkaid de Tlemcen et l'hôpital universitaire Tijani Damerdji.

Un cahier de charges a été remis par l'unité de greffe rénale. On y explique le processus de fonctionnement de la pré-greffe rénale.

Une réunion supplémentaire avec un néphrologue a été nécessaire afin de relever tous les besoins nécessaires à la compréhension du système.

Le néphrologue a exposé son mode de travail qui se résume en un document Microsoft Office Word qu'il remplit lui-même et les difficultés qu'il rencontre. Il doit également gérer la liste d'attente des receveurs qui désirent effectuer une transplantation rénale. L'attente peut être longue et donc les dossiers des patients doivent être à mis à jour périodiquement. De ce fait, les données doivent être toujours disponibles, ce qui n'est pas toujours le cas.

Nous avons proposé des idées sur le nouveau mode de fonctionnement et les fonctionnalités qui pourraient être ajoutées au nouveau système de gestion. Enfin, nous nous sommes mis d'accord avec le médecin sur les données qui devront se trouver dans le système et la manière de les présenter. Plusieurs discussions par messages électroniques ont été nécessaires afin d'aboutir à un produit satisfaisable et validé par le client.

3.2.1 Processus de pré-greffe rénale

Un patient en phase d'hémodialyse et ne répondant plus aux soins donnés passe à l'étape de greffe rénale.

Durant cette étape, le patient souffrant d'insuffisance rénale est enregistré comme un receveur de rein d'un donneur potentiel. Le receveur ne peut avoir qu'un seul donneur et ce dernier doit être obligatoirement un membre de sa famille.

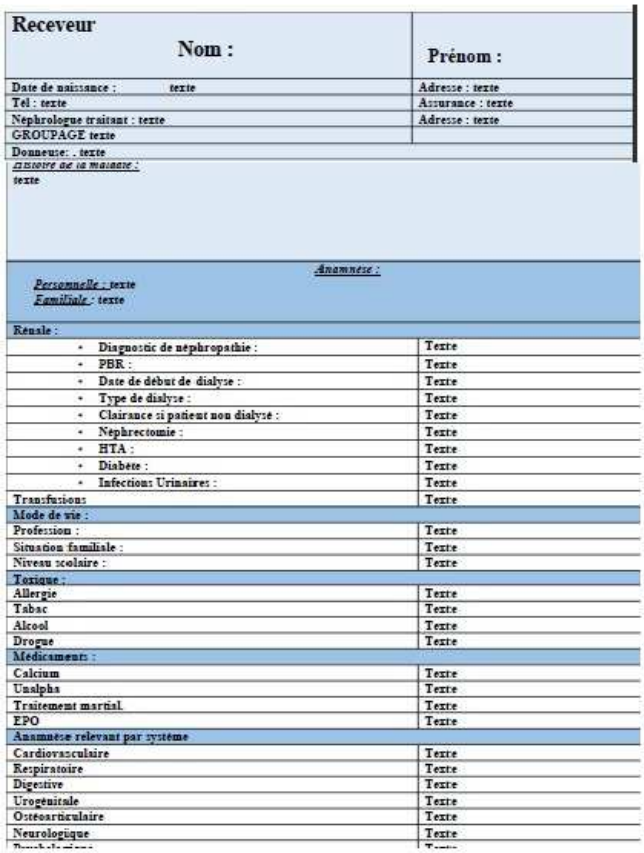
Etapes de pré-greffe rénale :

- **Arrivée des patients:** le néphrologue accueille le patient qui va recevoir le rein et le futur donneur. Si le receveur amène plusieurs donneurs potentiels, le néphrologue n'en choisira qu'un seul, en se basant sur certains critères, et va juger qui sera donneur le plus approprié.
- **Création des fichiers d'examens:** le néphrologue enregistre les informations personnelles qui concernent le receveur et le donneur comme le mode de vie, antécédents médicaux, anamnèse[1] familiale et personnelle et le groupe sanguin.
- **Soumission aux examens:** le néphrologue soumet le receveur ainsi que le donneur à un ensemble d'examens et questionnaires. Ces examens sont répartis en plusieurs dossiers. Il y a le dossier médical qui est composé d'un examen physique qui consiste à prendre la taille
- et le poids du patient, de tests immunologiques, examens système... Il y a également le dossier biologique, qui est principalement constitué de résultats d'analyses médicales, et le dossier morphologique composé de radiographies, scanners. Ces dossiers ne sont pas forcément remplis à la création des patients, ils seront remplis au fur et à mesure des consultations chez le néphrologue.
- **Mise à jour des dossiers:** le néphrologue met à jour les dossiers des patients suite aux examens qui seront effectués périodiquement à la demande de ce dernier.
- **Suppression des dossiers:** le néphrologue peut supprimer un dossier donneur si celui-ci n'est pas compatible avec son receveur, ce qui entrainera la suppression de tous ses examens et dossiers. Le receveur devra trouver un autre donneur et le présenter au médecin. Le néphrologue peut aussi supprimer le receveur si ce dernier n'est définitivement pas habilité à recevoir de rein. Cette opération entrainera non seulement la suppression de tous les examens et dossiers du receveur mais aussi celle de son donneur, s'il en possède, et de ses examens et dossiers.

[1] Informations médicales concernant la famille ou la personne même relatives au passé

Certains des examens pourraient être effectués à plusieurs reprises, à des dates différentes, afin de permettre au néphrologue de suivre l'évolution de l'état de santé du patient.

Pour gérer les données récoltées, le néphrologue a mis en place un document WORD composé de tableaux où il enregistre ces données. Dans la figure 3.1, on donne un exemple du document utilisé pour l'enregistrement d'un dossier receveur.



Receveur Nom :	Prénom :
Date de naissance : texte	Adresse : texte
Tel : texte	Assurance : texte
Néphrologue traitant : texte	Adresse : texte
GROUPAGE texte	
Donneur: . texte	
Histoire de la maladie : texte	
Anamnèse : Personnelle : texte Familiale : texte	
Rénale :	
• Diagnostic de néphropathie :	Texte
• PBR :	Texte
• Date de début de dialyse :	Texte
• Type de dialyse :	Texte
• Clairance si patient non dialysé :	Texte
• Néphrectomie :	Texte
• HTA :	Texte
• Diabète :	Texte
• Infections Urinaires :	Texte
Transfusions	Texte
Mode de vie :	
Profession :	Texte
Situation familiale :	Texte
Niveau scolaire :	Texte
Toxique :	
Allergie	Texte
Tabac	Texte
Alcool	Texte
Drogue	Texte
Médicaments :	
Calcium	Texte
Unalpha	Texte
Traitement martial	Texte
EPO	Texte
Anamnèse relevant par système	
Cardiovasculaire	Texte
Respiratoire	Texte
Digestive	Texte
Urogénitale	Texte
Ostéoarticulaire	Texte
Neurologique	Texte

Figure 3.1 : Document pour l'enregistrement d'un dossier patient

Cette figure représente le document WORD réalisé par le néphrologue. Il est réparti en plusieurs sections : informations personnelles, rénal, mode de vie… Ce sont toutes les informations qui vont trancher sur l'habilité d'un patient à donner ou recevoir un rein.

3.3 Problématiques

Le service de néphrologie connait une grande difficulté dans la gestion des dossiers patients, puisque ces derniers se trouvent sur des documents textes ou parfois même sur

papier. Cela engendre des pertes d'informations, et ainsi un retard dans la gestion de la pré-greffe.

Ci-dessous un ensemble de ces problèmes :

- Risque de perte des dossiers des patients et cela à cause de l'enregistrement désordonné des fichiers Word.
- Recherche difficile des dossiers et ce à cause de l'enregistrement des dossiers sur différents ordinateurs.
- Accès non contrôlé des médecins : n'importe quelle personne pourrait accéder à l'ordinateur de service et enregistrer modifier ou supprimer des données.
- Absence de vision globale de l'ensemble des patients enregistrés car il est impossible d'afficher une liste complète de tous les patients.
- Pas de lien entre les fichiers qui correspondent aux dossiers du receveur et de son donneur.
- Perte de temps dans la mise en page des documents, le remplissage des données et la recherche.

3.4 Spécification des besoins

La spécification des besoins constitue la phase de départ de toute application à développer. Cette phase nous permet d'identifier les besoins de notre application. Nous distinguons des besoins fonctionnels qui présentent les fonctionnalités attendues du nouveau système de gestion. Ainsi que les besoins non fonctionnels qui ne concernent pas spécifiquement la logique métier du système, mais plutôt l'identification des contraintes internes et externes de ce dernier.

3.4.1 Besoins fonctionnels

Les services proposés par notre application se résument dans les lignes suivantes :

- **Accès contrôlé des utilisateurs :** les néphrologues devront s'authentifier avec un nom d'utilisateur et mot de passe propres à chaque utilisateur.
- **Création d'un dossier receveur :** création d'un nouveau receveur en remplissant obligatoirement ses informations personnelles afin de pouvoir le créer et l'enregistrer.

Les autres dossiers, médical, biologique et morphologique sont systématiquement crées mais le remplissage des examens qu'ils contiennent est optionnel.

- **Création d'un dossier donneur :** créer un nouveau donneur pour un receveur qui existe déjà ; un donneur ne peut être créé que si on lui affecte un receveur. Un receveur ne peut avoir qu'un seul donneur enregistré.
- **Consultation et remplissage des dossiers :** le néphrologue doit pouvoir remplir et consulter le dossier médical, dossier biologique et dossier morphologique pour chaque patient (receveur et donneur) ainsi que les examens qui les constituent.
- **Modification des dossiers :** le néphrologue peut modifier les informations personnelles, dossier médical, dossier biologique et dossier morphologique pour chaque patient (receveur et donneur) ainsi que les examens qui les constituent.
- **Suppression des dossiers :** le néphrologue pourra supprimer un donneur, s'il le juge inapte à donner son rein au receveur correspondant, ce qui entrainera la suppression de tous de ses dossiers et examens. Le médecin peut aussi supprimer un receveur, dans ce cas, non seulement ses dossiers et examens seront supprimés mais les dossiers et les examens de son donneur, s'il en a, seront supprimés.
- **Recherche des patients :** le néphrologue pourra chercher un patient (receveur ou donneur) par son nom et son prénom et de là la possibilité d'afficher son dossier, le supprimer ou affecter un donneur dans le cas d'un patient de type receveur.
- **Vue globale des patients :** le néphrologue peut afficher une liste de tous les patients (receveurs et donneurs).
- **Enregistrement et impression :** le néphrologue peut enregistrer les informations personnelles, dossier médical, dossier biologique et dossier morphologique pour chaque patient (receveur et donneur) sous un format PDF ou les imprimer.
- **Plusieurs utilisateurs à la fois :** il n'y a qu'un ordinateur dans l'unité dans un premier temps, mais les néphrologues doivent pouvoir utiliser l'application simultanément.

3.4.2 Besoins non fonctionnels

Les besoins non fonctionnels décrivent toutes les contraintes auxquelles est soumis le système pour sa réalisation et son bon fonctionnement.

- **La sécurité:** nous devons prendre en considération la confidentialité des données des patients, pour cela nous devons restreindre l'accès à ces informations à l'administrateur et aux utilisateurs concernés.
- **La fiabilité et la rapidité:** notre système doit garantir la rapidité et la fiabilité de la recherche des informations, ainsi qu'une gestion optimale des ressources.
- **L'ergonomie:** l'application offre une interface conviviale et facile à utiliser.
- **Une solution ouverte et évoluée :** le code doit être clair pour permettre des futures évolutions ou améliorations.
- **Robustesse et maintenabilité :** l'application doit permettre le stockage des informations de tous les patients enregistrés et les différents traitements utiles pour le fonctionnement correct, ainsi qu'assurer une gestion exhaustive des erreurs.

3.5 Conclusion

L'étape d'analyse de l'existant est une étape complexe mais essentielle au lancement d'un projet car elle permet de comprendre les besoins attendus afin d'aboutir à des spécifications générales décrivant, en langage naturel, les données manipulées et les traitements à effectuer sur celles-ci. Cette étape permet d'esquisser une modélisation à grosses mailles de ces données et traitements que nous verrons dans le chapitre suivant.

4.CONCEPTION DU SYSTEME GREFFON

4.1 Introduction

Après avoir tracé les grandes lignes de phase de spécification de besoins, mettons l'accent maintenant sur une phase fondamentale dans le cycle de vie d'un logiciel, la phase de conception. Cette phase permet de modéliser le fonctionnement futur du système, de manière claire afin d'en faciliter la réalisation.

Dans ce chapitre nous allons présenter les trois diagrammes de modélisations en nous basant sur le langage de modélisation UML.

4.2 Présentation d'UML

En regardant les objectifs fixés pour la réalisation du projet, nous remarquons que nous sommes face à une application modulaire et qui devra rester ouverte pour les améliorations futures. De ce fait, il est très important d'utiliser un langage universel pour la modélisation afin de clarifier la conception et de faciliter les échanges.

4.2.1 Définition

UML (Unified Modeling Language ou « langage de modélisation unifié ») est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ».Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes et ne se limitant pas au domaine informatique [2].

4.2.2 Les avantages d'UML

La modélisation UML à plusieurs avantages :

- Universel : elle est connue par tous les concepteurs du monde.
- Adopté par les grandes entreprises : les grandes entreprises utilisent UML pour modéliser divers types de systèmes et de taille quelconque.
- Notation unifiée.
- Facile à comprendre.
- Limite les risques d'erreur.

- Concerne tous les domaines d'application et pas limités seulement au domaine informatique.

4.2.3 Logiciel de modélisation

Pour la modélisation des besoins, nous utilisons les diagrammes UML suivant : Diagramme de cas d'utilisation, diagramme de séquence et diagramme de classes, et pour cela on a choisi « Entreprise Architect » comme logiciel de modélisation.

Enterprise Architect est un logiciel de modélisation et de conception UML, édité par la société australienne Sparx Systems. Couvrant, par ses fonctionnalités, l'ensemble des étapes du cycle de conception d'application, il est l'un des logiciels de conception et de modélisation les plus reconnus [2].

4.3 Les diagrammes du système GREFFON

4.3.1 Diagramme de cas d'utilisation

Bien souvent, les utilisateurs ne sont pas des informaticiens. Donc il leur faut un moyen simple d'exprimer leurs besoins. C'est précisément le rôle des diagrammes de cas d'utilisation qui permettent de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Il s'agit donc de la première étape UML d'analyse d'un système.

En prenant en compte les besoins spécifiés dans le chapitre précédent, on présente le diagramme de cas d'utilisation.

La figure 4.1 représente le digramme de cas d'utilisation.

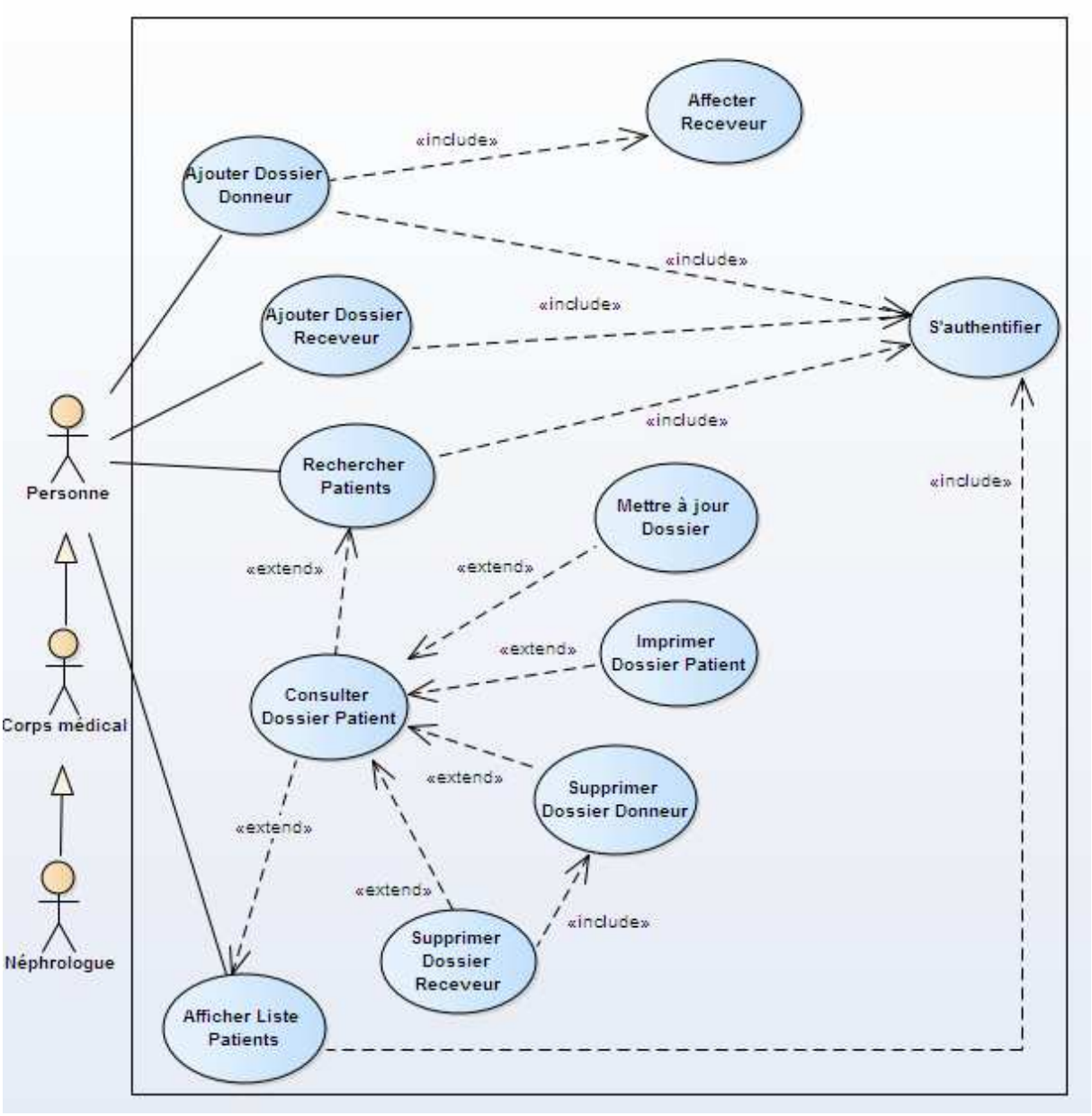


Figure 4.1 : Diagramme de cas d'utilisation

❖ *Identification des acteurs*

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié [3].

Suite à la demande de notre client, nous avons intégré trois acteurs qui vont interagir avec notre système qui sont : personne, corps médical et néphrologue.

❖ ***Identification des cas d'utilisations***

Un cas d'utilisation représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier [3].

Voici les cas d'utilisation de notre système :

- **L'authentification :** l'application vérifie que l'utilisateur est bien ce qu'il prétend être et lui donne ensuite l'autorisation d'accès.
- **Ajouter dossier receveur:** pouvoir ajouter des nouveaux receveurs et leurs dossiers.
- **Ajouter dossier donneur:** la possibilité d'ajouter un donneur seulement si son receveur existe déjà dans le système.
- **Lister patients :** voir la liste des patients (un patient peut être ou donneur ou receveur).
- **Rechercher patients :** pouvoir rechercher un receveur ou un donneur par son nom et son prénom.
- **Consulter dossier patient :** le néphrologue peut consulter les dossiers de ses patients (receveurs et donneurs).
- **Imprimer dossiers patient :** informations personnelles, dossier médical, dossier morphologique et dossier biologique.
- **Mise à jour des dossiers patients :** le néphrologue peut mettre à jour les dossiers à n'importe qu'elle moment.
- **Supprimer dossiers patients:** il peut aussi supprimer un dossier patient (la suppression d'un dossier receveur inclut la suppression du dossier de son donneur).

❖ ***Les relations entre acteurs***

La seule relation entre acteurs est la relation de ***généralisation***. Quand un acteur fils hérite d'un acteur père, il hérite en réalité de toutes les associations du père [3].

Suite à la demande du client, notre application va permettre des futures évolutions, et donc l'ajout de nouveaux utilisateurs par la suite.

Pour cela, on a utilisé deux relations de généralisations dans notre diagramme.

- L'acteur fils «corps médical» hérite de l'acteur père «personne» : le corps médical peut faire avec le système tout ce que peut faire une personne, plus d'autres choses.

- L’acteur fils «néphrologue» hérite de l’acteur père «corps médical» : le néphrologue peut faire avec le système tout ce que peut faire le corps médical, plus d’autres choses.

❖ ***Les relations entre cas d'utilisation***

Dans notre diagramme de cas d’utilisation nous avons utilisé les relations d’inclusion et d’extension qui servent à enrichir des cas d’utilisation par d’autres cas d’utilisation.

- L'inclusion « include » : Cela implique obligatoirement l'inclusion d'un cas d'utilisation dans un autre comme ici « Ajouter dossier donneur » fait obligatoirement appel à « S'authentifier », «Affecter receveur ».
- L'extension « extended » : Cela permet éventuellement l'extension d'un cas d'utilisation par un autre comme ici « Consulter dossier patient » peut étendre « mettre à jour dossier ».

4.3.2 Diagrammes de séquences

Il s'agit d'une explication détaillée d'un cas d'utilisation et donne une représentation temporelle des objets et leurs interactions.

Voici quelques notions de base du diagramme :

Scénario: Représente une succession particulière d'enchaînements, s'exécutant du début à la fin du cas d'utilisation, un enchaînement étant l'unité de description de séquences d'actions [3].

Ligne de vie : Représente l'ensemble des opérations exécutées par un objet [3].

Message: Un message est une transmission d'information unidirectionnelle entre deux objets, l'objet émetteur et l'objet récepteur. Dans un diagramme de séquence, deux types de messages peuvent être distingués [3]:

- **Message synchrone** : Dans ce cas l'émetteur reste en attente de la réponse à son message avant de poursuivre ses actions.
- **Message asynchrone** : Dans ce cas, l'émetteur n'attend pas la réponse à son message, il poursuit l'exécution de ses opérations.

Ce diagramme est la représentation du dialogue qui se fait entre le néphrologue, le système et la base de données.

- ***Diagrammes de séquence de Greffon***

Dans cette section, on présente quelques diagrammes de séquences de notre système.

Diagramme de séquences pour le cas d'utilisation «Ajouter dossier receveur»: le néphrologue remplit les informations spécifiques au nouveau receveur arrivé. Le système vérifie la présence des données obligatoires pour l'inscription d'un patient et le types de ces données. Si les informations sont valides, le formulaire va être envoyé et un nouveau dossier receveur sera créé sinon, un message d'erreur s'affiche à l'écran. Nous prenons en compte également la possibilité d'échec dans le cas d'un problème technique. Ce diagramme est représenté dans la figure 4.2.

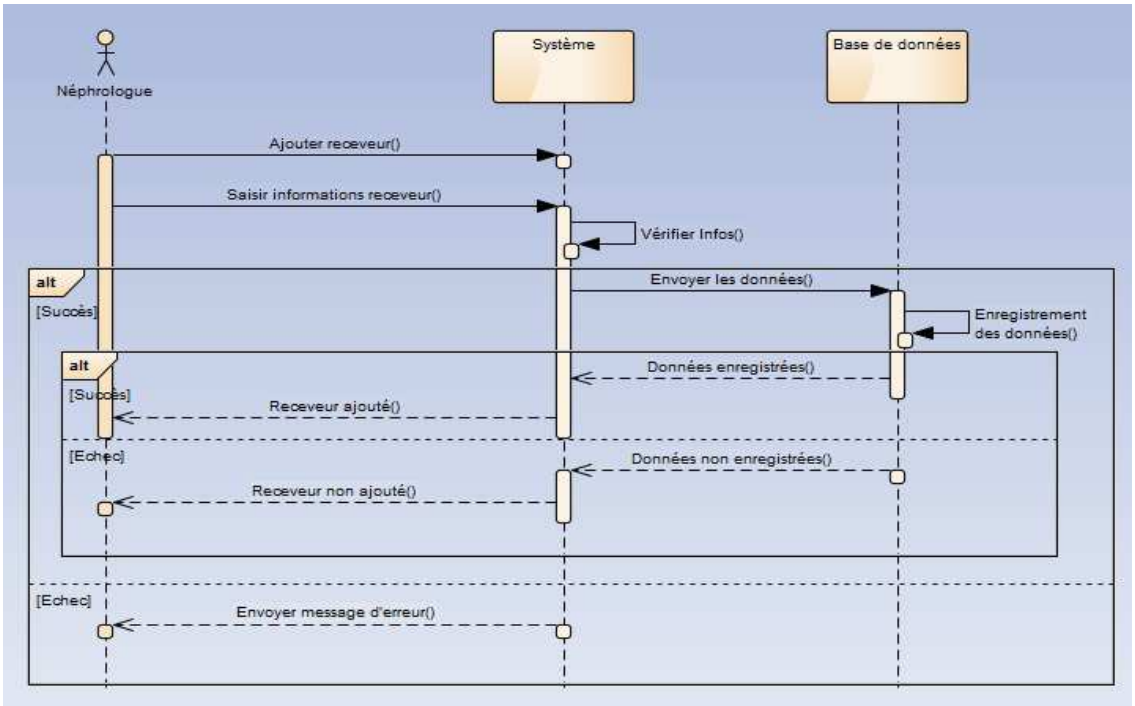


Figure 4.2 : Diagramme de séquence "Ajouter Dossier Receveur"

Diagramme de séquences pour le cas d'utilisation « Rechercher patient»: un patient représente ou le donneur ou le receveur.

Le néphrologue entre le nom et prénom du patient qu'il souhaite rechercher, une vérification de la validité des champs entrés est faite, si les champs ne sont pas vides et ils sont bien des

caractères, la rechercher est lancée et un résultat, positif ou négatif, est envoyé au néphrologue, sinon un message d'erreur est affiché dans le cas d'un échec technique..

La figure 4.3 représente le digramme de séquence pour le cas d'utilisation « Rechercher patient».

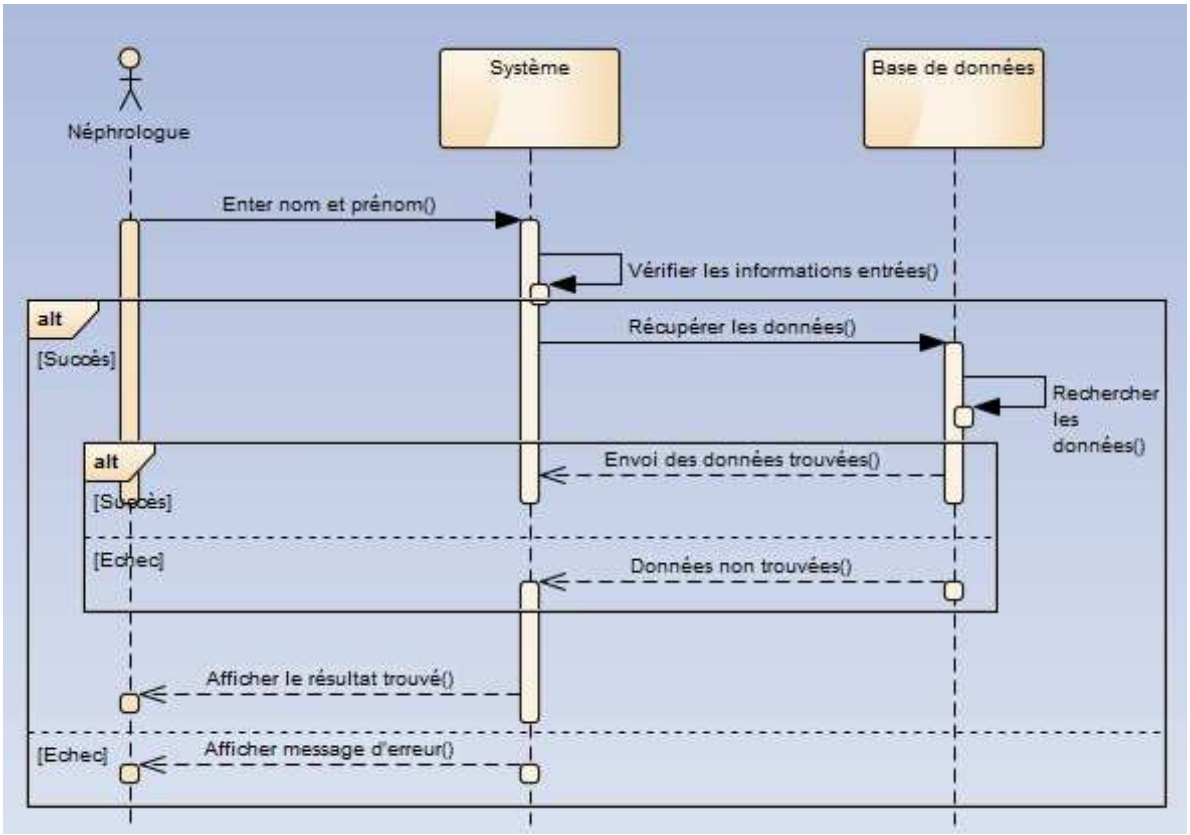


Figure 4.3 : Diagramme de cas d'utilisation "Rechercher patient"

Diagramme de séquences pour le cas d'utilisation «Modifier dossier patient»: le néphrologue recherche le patient à modifier et une fois trouvé, il entre les données à modifier qui sont vérifiées et enregistrées en cas de succès, sinon un message d'erreur est affiché.

La figure 4.4 représente le digramme de séquence pour le cas d'utilisation « Modifier dossier patient».

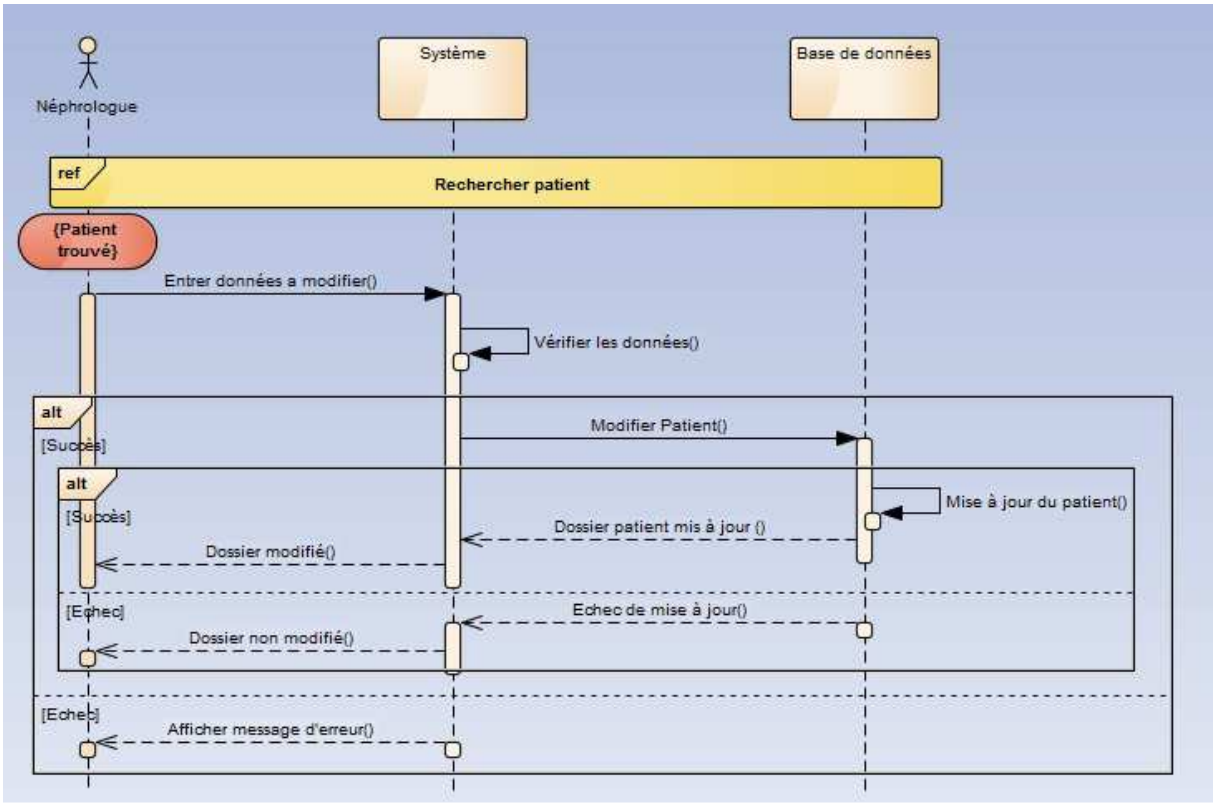


Figure 4.4 : Diagramme de séquence "Modifier dossier patient"

4.3.3 Diagramme de classes

Le diagramme de classes exprime la structure statique du système en termes de classes et de relations entre ces classes.

L'intérêt du diagramme de classe est de modéliser les entités du système d'information.

Voici quelques notions de base du diagramme :

- **Une classe :** représente la description abstraite d'un ensemble d'objets possédant les mêmes caractéristiques. On peut parler également de type [3].
- **Un attribut :** représente un type d'information contenu dans une classe [3].
- **Une opération:** représente un élément de comportement (un service) contenu dans une classe [3].
- **Une association**: représente une relation sémantique durable entre deux classes [3].
- **Une superclasse** : est une classe plus générale reliée à une ou plusieurs autres classes plus spécialisées (sous-classes) par une relation de généralisation. Les sous-classes

«héritent» des propriétés de leur superclasse et peuvent comporter des propriétés spécifiques supplémentaires [3]

- ***Design pattern composite***

Pour avoir une bonne structuration de l'application, nous avons utilisé le design pattern « composite ».

Description :

Le but du pattern Composite est d'offrir un cadre de conception d'une composition d'objets dont la profondeur est variable, cette conception étant basée sur un arbre. Par ailleurs, cette composition est encapsulée vis-à-vis des clients des objets qui peuvent interagir sans devoir connaître la profondeur de la conception.

Avantages :

- L'ajout de nouveaux composants est simple cela grâce à la hiérarchie des classes.
- Le client ne se soucie pas de l'objet accédé.
- La structure de l'application est moins complexe.

La figure 4.5 représente le digramme de classes que nous avons proposé.

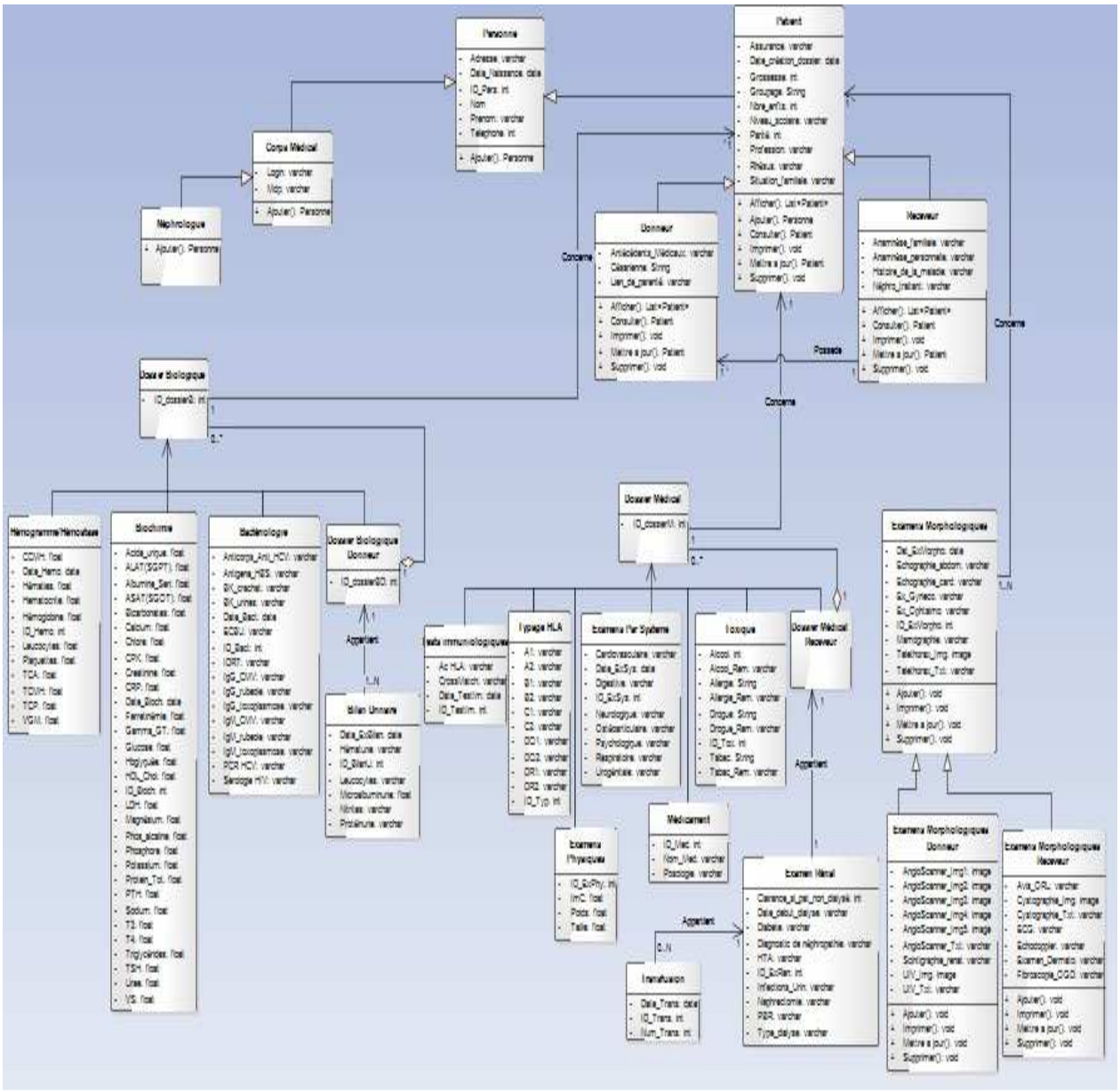


Figure 4.5 : Diagramme de classes

Nous allons décrire le diagramme de classes représenté dans la figure 4.5 en le découpant en cinq niveaux pour une meilleure lisibilité.

Niveau 1 :

Notre diagramme contient la classe « Personne », cette classe présente le sommet de la hiérarchie.

Niveau 2 :

Les deux classes qui héritent de cette classe sont : la classe « Corps médical » et la classe « Patient » via la relation de généralisation.

Niveau 3:

La classe « Néphrologue » hérite de « corps médical ».

Les classes « receveur » et « donneurs » héritent de la classe « patient » qui contient les informations en commun entre eux.

- L'héritage se fait par une relation de généralisation qui nous permet par la suite d'ajouter de nouveaux types de personne, patient ou corps médical au système.

Niveau 4 :

Les classes suivantes : dossier médical, dossier biologique et examens morphologique ont une relation d'association « concerne » avec « patient ».

- Un dossier médical concerne un patient.
- Un dossier biologique concerne un patient.
- Un examen morphologique concerne un patient.

Niveau 5 :

- Les classes « examens morphologique receveur » et « examens morphologique donneur » héritent de la classe « examens morphologique ». Cette dernière contient les examens morphologiques en commun entre le receveur et le donneur.
- Le dossier médical contient les examens suivants : test immunologique, examen physiques, examen système, toxique, médicament et typage hla.

Ces examens sont communs entre le receveur et le donneur.

Le dossier médical receveur, en plus de ces examens, il contient l'examen rénal.

Pour une meilleure organisation des classes et pour faciliter la manipulation de cette dernières par l'utilisateur, nous avons choisi le design pattern « composite » qui est présenté dans la figure 4.6.

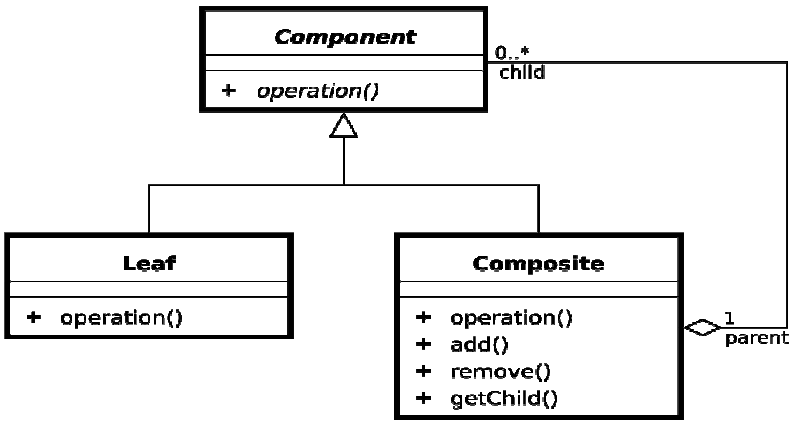


Figure 4.6 : Design Pattern Composite

On considère le dossier médical une classe abstraite (composant).

Les classes : test immunologique, examen physique, examen système, typage hla, médicaments, toxique et dossier médical receveur appartient à cette classe.

Comme le dossier médical receveur (composite) et un composant potentiel, il peut être composé d'autres éléments composites (dossier médical) et d'autres éléments simples « examen rénal ».

Le même principe pour le dossier biologique.

4.3.4 Le modèle relationnel

A partir du diagramme de classe que nous avons effectué, on peut réaliser le modèle relationnel ; vue que le système d'information ne peut pas le manipulé directement, et cela en utilisant les règles de passage suivantes :

Règle1: présence de la cardinalité (?..1) d'un côté de l'association :

- Chaque classe se transforme en une table.
- Chaque attribut de classe se transforme en un champ de table.
- L'identifiant de la classe qui est associée à la cardinalité (?..1) devient la clé étrangère de l'autre classe.

Règle2: présence de (?..N) des deux côtés de l'association :

- Chaque classe se transforme en une table

- Chaque attribut de classe se transforme en un champ de table
- L'association se transforme en une table. Cette table a comme champs l'identifiant de chacune des deux classes, plus d'éventuels autres attributs.

Règle3: présence d'une généralisation :

Méthode 1:

- Création d'une table avec tous les attributs des classes.
- L'ajout d'un attribut pour distinguer les types des objets.

Méthode 2 :

- Création d'une table pour chaque sous type, chaque table se compose des attributs génériques et d'attributs spécifiques.

Méthode 3 :

- Création d'une table par classe et des associations.

La figure 4.7 représente le modèle relationnel de notre système.

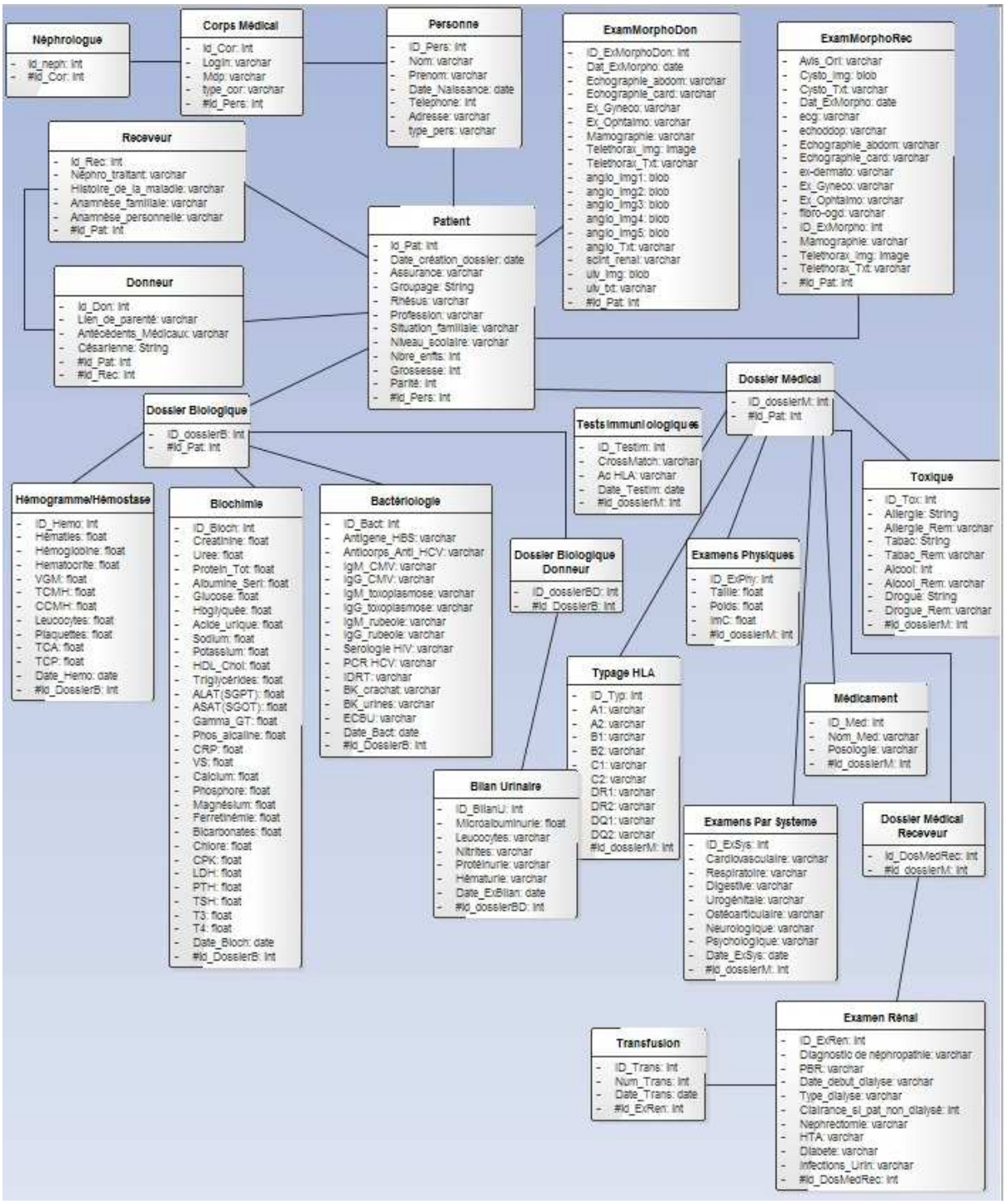


Figure 4.7 : Modèle logique de données relationnelles

4.4 Conclusion

L'étape de conception permet de mettre le point sur les éléments importants du système à réaliser et de ce fait diminuer les risques et maintenir l'entropie[2] du projet à un niveau raisonnable afin d'entamer l'étape de développement sur de bonnes bases.

[2]Degré de désorganisation ou de manque d'information d'un système

5. REALISATION DU SYSTEME GREFFON

5.1 Introduction

Après avoir tracé l'architecture du système, défini ses fonctions et les ensembles qui le composent, nous allons présenter dans ce chapitre la réalisation de l'application Greffon ainsi que les ressources mobilisées et les outils qui y ont contribué.

5.2 Outils de structuration du code

A la demande du client, notre application doit pouvoir être utilisée par plusieurs néphrologues simultanément. C'est pour cela que nous optons pour une architecture client/serveur, où plusieurs machines clientes, connectées à un serveur d'applications, pourront utiliser l'application en même temps.

5.2.1 L'architecture 3-tiers

Notre application, se basant sur le principe client/serveur, nous adoptons donc l'architecture 3-tiers.

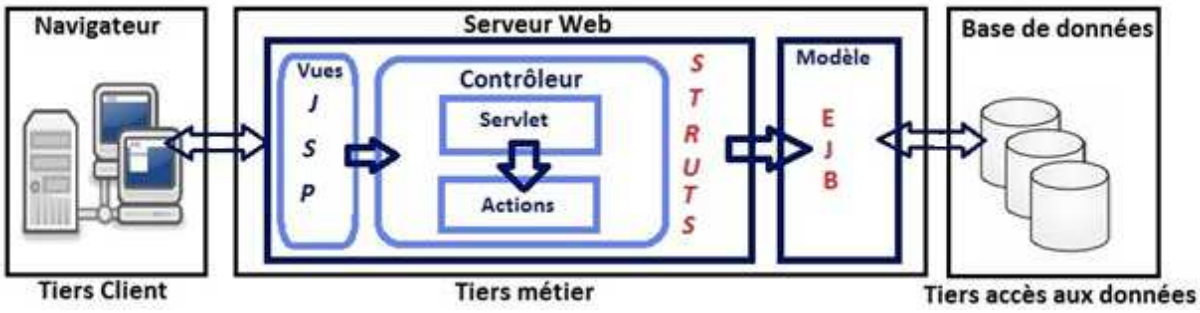


Figure 5.1 : Architecture 3-tiers avec Strus et EJB

Ce modèle représenté par la figure 5.1 est une évolution du modèle d'application client/serveur. L'architecture *3-tiers* est divisée en trois niveaux :

- Tiers client qui correspond à la machine sur laquelle l'application cliente est exécutée.
- Tiers métier qui correspond à la machine sur laquelle l'application centrale est exécutée.
- Tiers accès aux données qui correspond à la machine gérant le stockage des données.

Chaque tiers doit être indépendant des autres et peut, par conséquent, être remplacé sans engendrer des modifications dans les autres tiers [4]. Pour faciliter cette répartition, nous choisissons d'utiliser l'architecture MVC.

5.2.2 Architecture MVC

Notre application repose sur le modèle en couche. Cependant la connexion de ces couches entre-elles n'est pas une mince affaire. En effet, pour rester dans le principe de l'autonomie, il faut que les couches soient indépendantes. C'est donc à ce niveau que les problèmes apparaissent. Donc, pour obtenir une telle architecture il est indispensable d'utiliser des modèles de conception qui facilitent cette tâche fastidieuse. Le modèle de conception que nous allons choisir et qui satisfait notre besoin est le MVC.

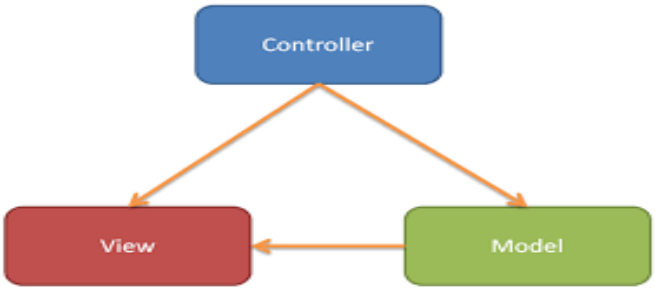


Figure 5.2 : Modèle-Vue-Contrôleur

Le modèle MVC (*Model/View/Controller*) représenté sur la figure 5.2 est un schéma de programmation qui prend en compte toute l'architecture d'un programme et classe les différents types d'objets qui composent l'application dans 3 catégories [5]:

- **La vue :** elle affiche le contenu du modèle. Elle prend les contenus du modèle et spécifie comment ces données peuvent être présentées aux utilisateurs.
- **Le modèle:** il présente la partie métier de l'application. Il gère l'accès et la modification des données de l'application.
- **Le contrôleur :** il définit le comportement de l'application. Il interprète les requêtes des utilisateurs et définit les actions que le modèle devra exécuter.

5.2.3 Tiers accès aux données

Comme nous l'avons présenté dans le troisième chapitre, le modèle relationnel a été utilisé pour la conception de la base de données. Nous avons choisi le SGBD Oracle Database 10g Express.

- ***Oracle Database 10g XE***

Oracle est un système de gestion de base de données relationnelle. Cette édition permet aux entreprises de tester et développer des solutions Oracle sans investir dans les frais de licences et a une capacité de 10 Go ce qui est largement suffisant pour notre application. Il est aussi très fiable et robuste et a un stockage des données images de bonne qualité pour l'enregistrement des radiographies, échographies et scanners.

5.2.4 Tiers métier

Pour créer la connexion entre la base de données et le serveur, nous choisissons d'utiliser les EJB.

- ***Entreprise JavaBeans***

Notre choix s'est porté sur les EJB 3 car ils intègrent la JPA (*Java Persistance API*) qui est utilisée au niveau de la couche de persistance pour créer une séparation entre l'application et la base de données. Ils sont parfaitement adaptés pour être intégrés dans une architecture trois tiers ou plus.

Enterprise JavaBeans (EJB) est une architecture de composants logiciels côté serveur pour la plateforme de développement Java EE [2] qui possèdent certaines caractéristiques comme la réutilisabilité, la possibilité de s'assembler pour construire une application.

Une des principales caractéristiques des EJB est de permettre aux développeurs de se concentrer sur les traitements orientés métiers car les EJB et l'environnement dans lequel ils s'exécutent prennent en charge un certain nombre de traitements tel que la gestion des transactions, la persistance des données, la sécurité, ...

Les types de composants EJB sont :

- **Session bean :** un Session Bean est une application côté serveur permettant de fournir un ou plusieurs services à différentes applications clientes.

- **Entité bean :** les Entity Beans ont été créés pour simplifier la gestion des données au niveau d'une application, mais aussi pour faciliter la sauvegarde en base de données.

Contrairement aux Session Beans, les données d'un Entité Bean sont conservées, même après l'arrêt de l'application.

Pour créer nos entités bean, nous avons utilisé le plugin fourni par l'IDE Eclipse, Dali.

❖ ***Plugin Dali***

Dali est un plug-in proposant de fournir des outils pour faciliter le mapping objet/relationnel et est un sous projet du projet WTP(Web Tools Platform) dont le but est de faciliter la mise en œuvre de l'API JPA (Java Persistence API).

En effet cette plateforme inclut des outils pour créer un projet JPA et faire du **reverse engineering** pour créer automatiquement les entités beans. Cela correspond à l'approche BottomUp qui consiste à d'abord créer le schéma de la base de données puis à développer les entités beans. Nous avons utilisé donc le modèle logique de données relationnelles pour la création de la base de données, puis nous avons généré les entités beans par la suite.

Dans l'architecture adoptée, sur le serveur, les objets métiers contiennent les traitements. Les EJB sont spécialement conçus pour constituer de telles entités.

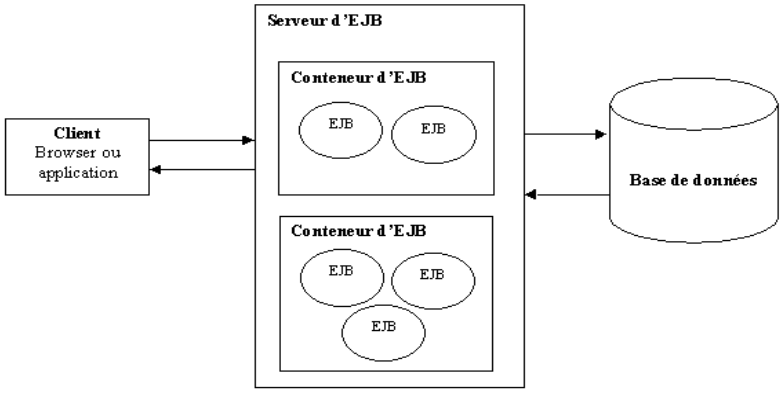


Figure 5.3 : Schéma d'une architecture trois tiers et EJB

La figure 5.3 représente l'architecture 3-tiers tout en intégrant les EJB. Ils s'exécutent dans un environnement particulier : le serveur d'EJB. Celui-ci fournit un ensemble de fonctionnalités

utilisées par un ou plusieurs conteneurs d'EJB qui constituent le serveur d'EJB. En réalité, c'est dans un conteneur que s'exécute un EJB et il lui est impossible de s'exécuter en dehors.

Le serveur permettant d'exécuter les EJB que nous avons choisi est JBoss.

- ***Serveur d'applications JBoss Open Source***

Nous avons porté notre choix sur JBOSS car il fournit toute sorte de services standardisé: conteneur d'EJB, gestion de mail, de transactions, de gestion de la sécurité, gestion du déploiement…

JBOSS est un serveur d'application dit « libre ». Ce serveur d'application est dédié au développement d'applications compatible J2EE. En effet, les développements sous JBOSS deviennent portables sur d'autres serveurs. Cela garantie une pérennité importante des développements et une moindre maintenance.

De plus, JBOSS permet de se connecter à la plupart des standards du marché (ORACLE, MySQL…). [6]

Pour exécuter des requêtes directement sur les entités bean, on utilise le langage JPQL (Java Persistence Query Language). JPQL est proche de la syntaxe SQL mais orienté objet. En effet, les requêtes JPQL opèrent sur des classes et des objets persistés (entités) au lieu de tables et colonnes.

Un exemple d'une requête JPQL :

```
@PersistenceContext em;
Query query = em.createQuery("SELECT r FROM Receveur r WHERE r.nephroTrait = :valeur");
q.setParameter("valeur", nephro);
Receveur rec = q.getSingleResult();
```

L'interface EntityManager fournit la méthode **createQuery()** pour créer des requêtes. Cette méthode retourne une instance Query. La requête sélectionne les objets de type Receveur qui ont pour néphrologue traitant la valeur indiquée dans la requête. La requête va être exécutée à l'appel de la méthode **getSingleResult()**. Cette méthode ne retourne qu'un seul objet.

- Les opérations d'insertion INSERT ne sont pas supportées par JPQL. De nouvelles lignes peuvent être insérées en base en utilisant la méthode **EntityManager.persist()**[3].

Afin d'intégrer le design pattern MVC au développement de notre application, nous avons choisi d'utiliser le framework STRUTS car il s'appuie sur ce modèle.

❖ ***Struts et MVC2***

Struts s'appuie non seulement sur le modèle MVC mais permet aussi de créer l'abstraction de l'accès aux données. Ce sont les EJB Entity qui rempliront cette fonction.

Struts est un framework MVC utilisé pour développer des applications WEB. Il s'agit donc d'un squelette d'application s'appuyant sur le modèle vue contrôleur et fournissant des outils supplémentaires pour aider le développeur à réaliser ses applications.

On distingue donc un contrôleur principal qui aiguille les requêtes reçues du client vers des sous contrôleurs qui vont alors effectuer le traitement correspondant.

- ***Le modèle :*** Struts ne fournit aucune implémentation pour le modèle. Le développeur doit donc définir lui-même le(s) modèle(s). Dans notre application, elle représente ici la partie métier (EJB).
- ***Les vues :*** dans Struts, ce sont principalement des pages JSP.
- ***Le contrôleur :*** Struts implémente un contrôleur principal (représenté par la classe ActionServlet) et des sous contrôleurs (correspondant aux classes Action).

Le schéma de la figure 5.4 représente la structure de Struts et son fonctionnement. On retrouve sur ce schéma les éléments principaux de Struts, à savoir :

- ***ActionServlet*** : c'est le contrôleur principal. Il joue le rôle de servlet et reçoit donc les requêtes du client qu'il renvoie ensuite vers les sous contrôleurs.
- ***Action*** : il s'agit des sous contrôleurs. Ils effectuent le traitement correspondant à une requête précise.
- ***ActionForm*** : ce sont des containers pour les pages JSP. Ils peuvent être utilisés par les Action lors du traitement.
- ***Le fichier de configuration 'Struts-config.xml'*** : C'est le composant principal de Struts puisqu'il permet de faire le lien entre tous les composants précédents.

[3]Méthode de l'interface javax.persistence.EntityManager (API JPA)

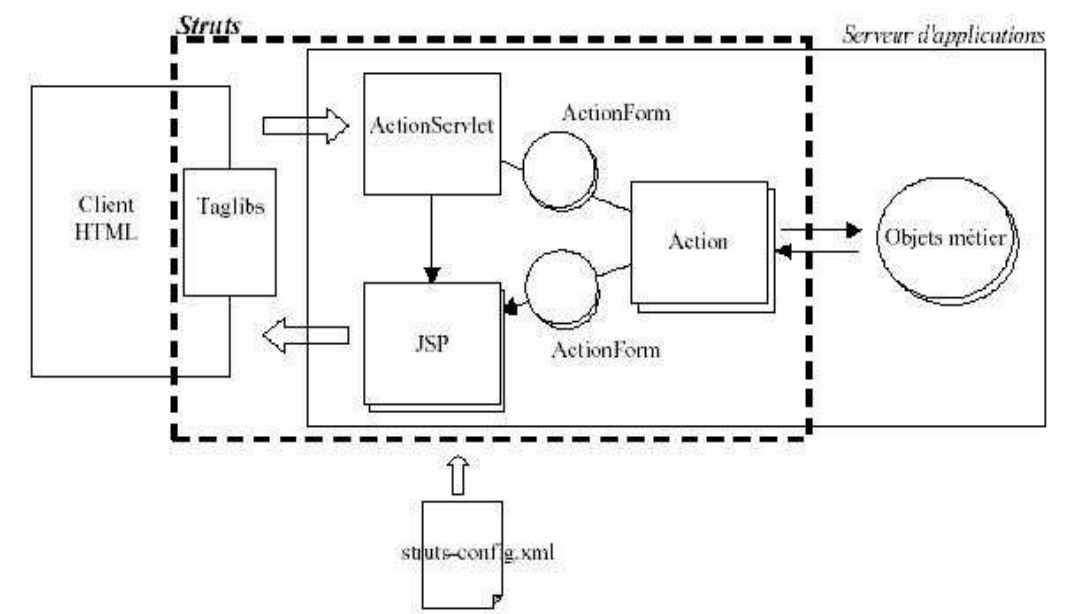


Figure 5.4 : Schéma Arnaud Buisine, Sysdeo

Selon le schéma représenté dans la figure 5.4, le cycle de vie d'un service de l'application en utilisant Struts se déroule comme suit :

Le client envoie une requête à l'ActionServlet.

- Grâce au fichier de configuration 'Struts-config.xml', l'ActionServlet aiguille la requête vers l'Action appropriée.
- L'Action réalise alors le traitement adéquat. Si besoin, cette Action utilise les ActionForm nécessaires et effectue les opérations utiles sur le modèle.
- L'action renvoie ensuite le résultat du traitement (réussite, échec…).
- A partir de cette valeur, l'ActionForm est alors capable de déterminer le résultat à renvoyer au client (redirection vers une autre page JSP…).

❖ ***Intégration de Struts2 avec EJB3 :***

Dans cette section, nous exposons le fonctionnement des EJB3 avec STRUS2. Le schéma de la figure 5.5 illustre l'implémentation du modèle MVC 2 dans Struts2 et l'utilisation des EJB :

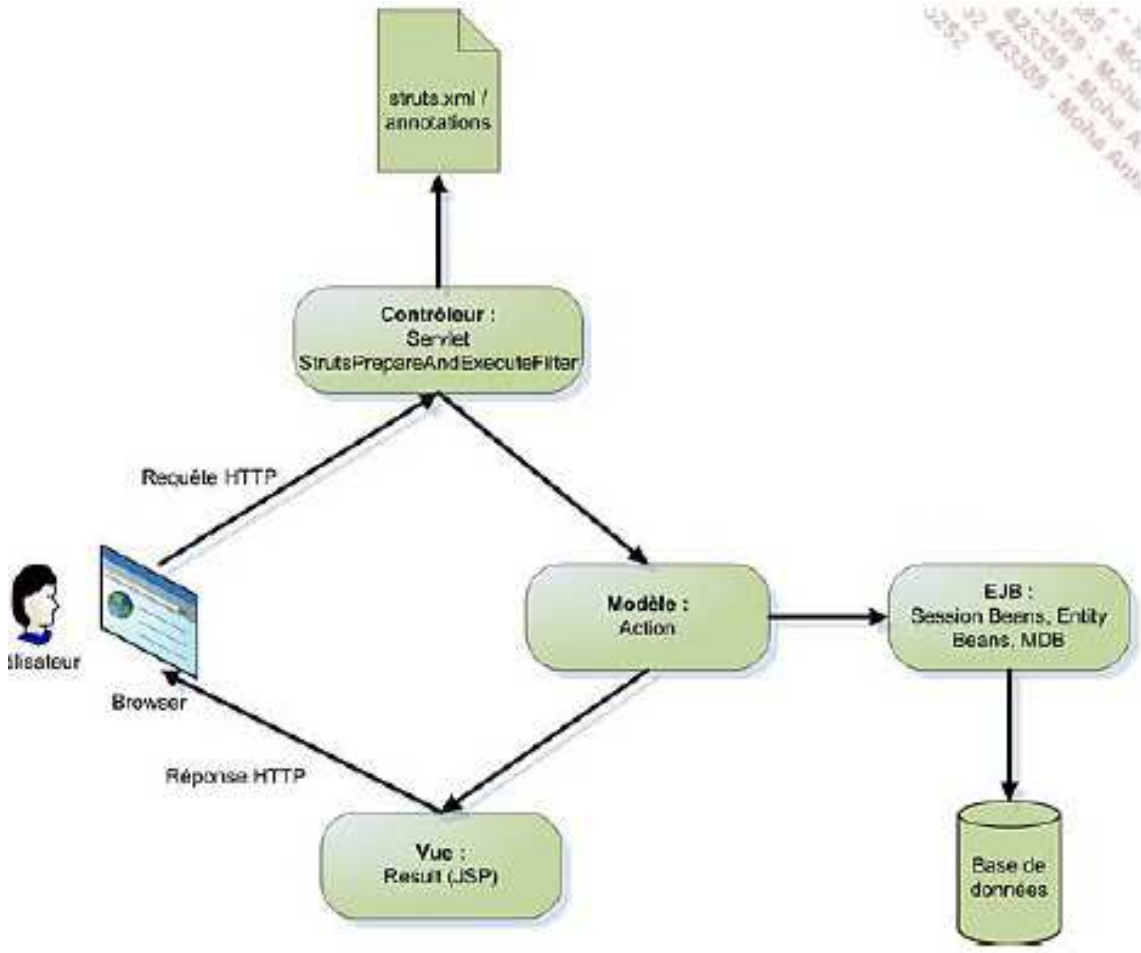


Figure 5.5 : Intégration Struts2 avec EJB3

Le Framework **Struts** qui se base sur le modèle **MVC** couvre le premier et le deuxième tiers de l'architecture client/ serveur.

Dans un premier temps, le client envoie une requête, celle-ci est interceptée par le contrôleur représenté par la servlet **StrutsPrepareAndExecuteFilter**.

Cette servlet invoque l'action du modèle qui va traiter la requête en fonction de la configuration définie dans le fichier struts.xml ou suivant les annotations ou par convention.

Le traitement qui se fait par cette action appelle le session bean (**EJB**). C'est à ce niveau-là qu'on intègre les EJB qui vont faire le lien entre le deuxième et le troisième tiers métier/ base de donnée.

Les sessions bean utilisent les entités bean pour exécuter les requêtes **JPQL et les opérations d'insertion, de suppression et de modification**, ces dernières permettent l'accès à la base de données.

En fonction de la valeur renvoyée par la requête et les informations du fichier de configuration, l'Action détermine la **vue** à présenter en résultat au client.

5.2.5 Tiers client

Dans notre application, et selon l'architecture utilisée, le client est un client web. Ainsi, selon Struts, les vues, qui sont le client, sont représentées par des pages JSP.

Les JSP (Java Server Pages) sont une technologie Java qui permet la génération de pages web dynamiques.

La technologie JSP permet de séparer la présentation sous forme de code HTML et les traitements écrits en Java sous la forme de JavaBeans ou de servlets. Ceci est d'autant plus facile que les JSP définissent une syntaxe particulière permettant d'appeler un bean et d'insérer le résultat de son traitement dans la page HTML dynamiquement.

Les JSP créées pour notre application sont composées de :

- Données et tags HTML.
- Tags fournis par les bibliothèques «taglibs» de Struts permettant d'ajouter dans les JSP des balises spécifiques à Struts.
- **Code JavaScript :** langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs [2]. Il nous permet ici de faire les tests de validation avant l'envoi des formulaires.
- **JQuery :** framework Javascript sous licence libre qui permet de faciliter des fonctionnalités communes de Javascript. L'utilisation de cette bibliothèque permet de gagner du temps de développement lors de l'interaction sur le code HTML. [2]
- **CSS :** est un langage informatique utilisé sur l'internet pour mettre en forme les fichiers HTML ou XML. Ainsi, les feuilles de style, aussi appelé les fichiers CSS, comprennent du code qui permet de gérer le design d'une page en HTML.

Nous utilisons également les intercepteurs qui sont une forme de programmation par aspect. Ils nous permettent de vérifier l'authenticité des utilisateurs de l'application « Greffon ».

5.2.6 Les intercepteurs

Un intercepteur est une méthode qui intercepte l'appel d'une méthode métier.

Les intercepteurs fonctionnent à la manière des filtres avec les servlets. Ils sont automatiquement appelés, juste avant que les méthodes métiers du bean soient appelées.

Les classes intercepteurs ont le même cycle de vie que les EJB qu'elles interceptent.

Les avantages des intercepteurs sont multiples :

- Un contrôle plus fin dans le déroulement d'une méthode.
- La possibilité d'ajouter des fonctionnalités à une méthode sans en modifier le code.
- On peut de façon élégante analyser et manipuler des paramètres et autoriser ou interdire l'exécution d'une méthode métier.

Afin d'intégrer tous ces outils dans l'implémentation de notre application et de planifier notre travail, nous avons eu recours à la conception de diagrammes regroupant les classes participant à chaque service. Cela nous a permis d'effectuer un travail modulaire et de bien répartir les tâches.

5.2.7 Diagrammes de classes conception

Afin de mieux comprendre la structure du code utilisé en intégrant Struts et EJB, nous l'avons modélisé à l'aide de diagrammes de classes de conception. Nous les avons intégrés dans cette section car, sans comprendre ce que sont les EJB et Struts, les diagrammes n'auront pas été clairs.

Le diagramme de classes de conception représente la vue de conception statique d'un système. Il permet d'identifier les objets à utiliser pour la réalisation de chaque service. Nous donnons dans la figure 5.6 le diagramme de classes de conception pour le service d'authentification.

- Diagramme de classes de conception pour les cas d'utilisation « Authentification » :

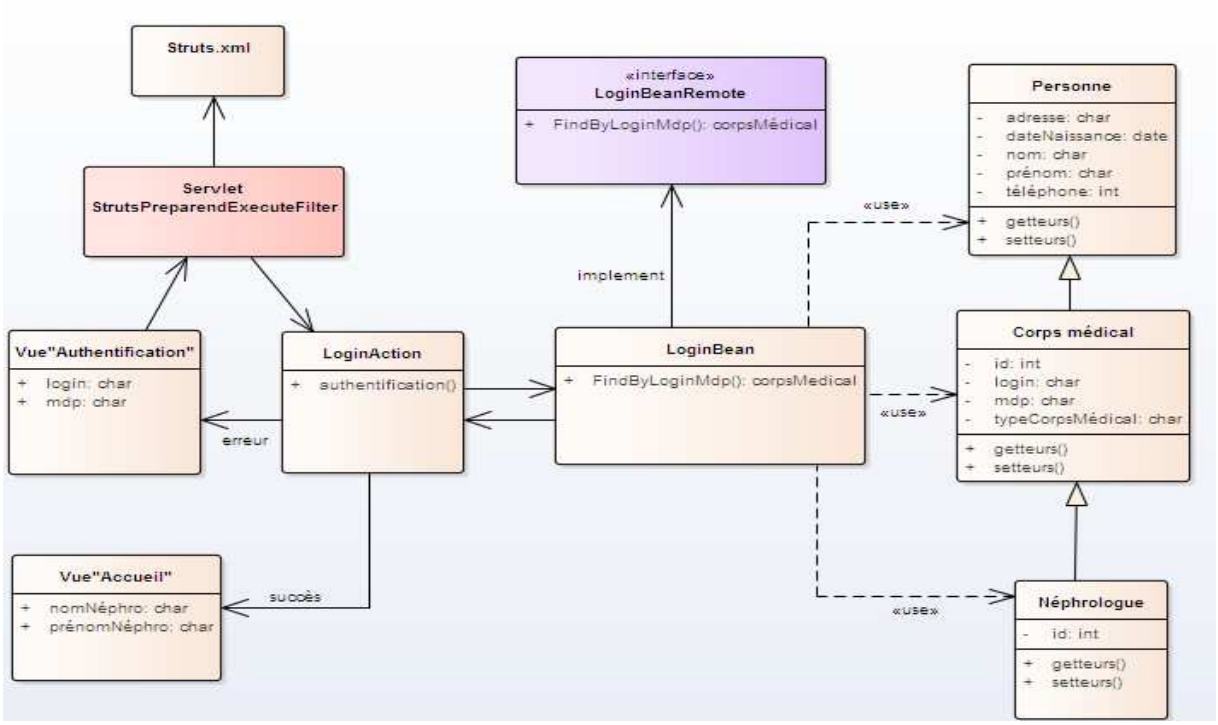


Figure 5.6 : Diagramme de conception pour "Authentification"

Scénario :

Le néphrologue appelle la page d'authentification, saisit son nom d'utilisateur, son mot de passe et valide le formulaire.

L'ActionServlet « **StrutsPrepareAndExecuteFilter** » intercepte la requête pour la traiter en effectuant les actions suivantes :

- Détermination de l'Action à utiliser en fonction des informations contenues dans le fichier struts.xml.
- Appel de la méthode authentification() de la classe LoginAction qui contient les traitements à effectuer pour répondre à la requête.
- Appel de la méthode FindByLoginMdp() qui se trouve dans le sessionBean « LoginBean ».

Les accès aux sessions beans sont facilités par la création d'une classe qui centralise ces accès, conformément au design pattern Service locator. Le Service Locator est utilisé pour faire l'abstraction des appels de méthodes au niveau du serveur d'application ; Il va créer un

cache qui permet un gain de performance car si un service (session bean) est déjà présent dans le cache, on ne va pas le chercher à nouveau.

De plus, les appels aux entity beans se font via les sessions beans afin de ne pas exposer les entity beans à la couche présentation.

- En fonction de la valeur renvoyée par la méthode authentification() et des informations du fichier de configuration, l'ActionServlet détermine la page à présenter au néphrologue : si les informations saisies sont justes, la page déterminée est la page d'accueil, sinon le néphrologue reste dans la page d'authentification.

5.3 Développement de « Greffon »

Pour récapituler la réalisation de Greffon, nous allons exposer les étapes de déroulement de notre travail.

Nous allons commencer par le choix des outils. En effet, les outils qui ont permis le développement de notre application ont été choisis pour des soucis de compatibilités, notamment dûs à la récence de certaines technologies.

La réalisation du projet s'est déroulée en plusieurs étapes :

Nous avons utilisé le modèle logique de données relationnelles afin de concevoir le schéma de la base de données. C'est en nous basant sur le schéma de la base de données que nous avons généré automatiquement les entités beans en faisant du reverse engineering.

Un premier projet de test a été développé afin de tester les connexions à la base de données, le bon déploiement du projet sur le serveur d'applications ainsi que les opérations de base comme l'ajout et la suppression. Nous avons eu recours au framework ***JUnit*** pour ces tests. Nous avons ensuite créé quelques pages JSP et servlets pour nous assurer qu'il y avait une bonne communication entre les vues et les servlets. Après nous être assurés de la bonne connexion entre toutes les parties du projet, nous avons pu continuer le développement du projet en intégrant Struts.

Parmi les services que l'application doit fournir l'impression des rapports d'examens. Pour cela nous utilisons l'outil iReport pour créer les rapports et le framework JasperReports pour les générer sous format PDF et, éventuellement, les enregistrer ou les imprimer.

5.3.1 IReport

iReport est un outil de design de type WYSIWYG (*What You See Is What You Get*). Il s'adresse aux développeurs et aux utilisateurs qui utilisent la librairie JasperReports pour créer des rapports.

Il est développé en Java et permet d'éditer de façon visuelle des rapports au format XML contenant des diagrammes, des images, des sousrapports, etc. Modifier les fichiers de type JRXML à la main n'est pas une opération facile, ce qui fait d'iReport un outil indispensable pour la mise en forme des rapports. [7]

Nous utilisons ici la version standalone de iReport.

La figure 5.7 montre un exemple d'un fichier .jrxml créé avec iReport :

Figure 5.7 : Exemple de fichier jrxml

La figure 5.7 représente l'un des rapports créés qui est la fiche du receveur. Elle est découpée en plusieurs sections :

- Title (<title>) : le titre apparaît une seule fois, au début du rapport.
- Page Header (<pageHeader>) : cette section apparaît au début de chaque page. C'est l'entête de la page.
- Detail (<detail>) : les valeurs des champs de la source de données qui est l'ensemble des informations concernant le receveur sont affichées ici.
- Page Footer (<pageFooter>) : c'est le pied de page qui apparaît au bas de chaque page et peut afficher un compteur de pages.

5.3.2 JasperReports

Le framework JasperReports est mis en œuvre dans l'application afin de générer des rapports. Lorsque le néphrologue consulte le dossier d'un patient, il a la possibilité de générer à la volée une fiche du dossier dans un format PDF.

JasperReports est un framework Open Source de reporting très populaire. Il permet de générer des fichiers aux formats PDF, XML, HTML, Excel, Word, OpenOffice (ODF) ou CSV à partir de données provenant de sources diverses : une base de données relationnelle (par exemple Oracle)de JavaBeans, de requêtes EJBQL…Il peut être intégré dans une application Java EE pour fournir au développeur un moyen puissant de créer facilement des rapports personnalisés contenant des données de tout type : textes, images, diagrammes…

Le framework JasperReports se compose d'une librairie Java. La compilation du fichier de design .jrxml est effectuée par la méthode compileReport(). [7]

La figure 5.8 montre les étapes de génération d'un rapport en le framework JasperReports et l'outil de design iReport.

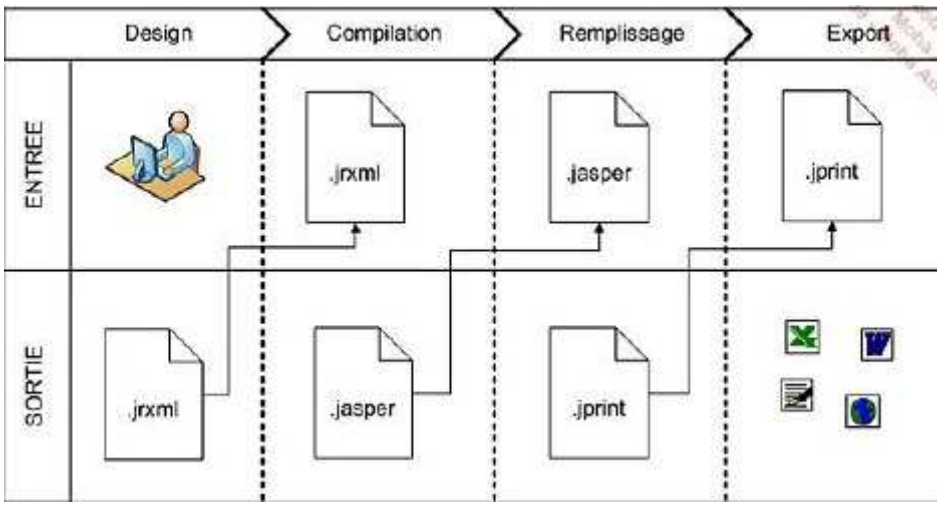


Figure 5.8 : Etapes de génération de rapports avec JsperReports et iReport

Comme le montre la figure 5.8, on crée d'abord le rapport à l'aide de iReport qui va générer un fichier jrxml. Le fichier est compilé avec iReport. Le fichier résultant de la compilation, de type jasper, sera alimenté de données provenant, dans notre cas, de la base de données. Un fichier du format que l'on choisit, ici PDF, est enfin généré et peut être affiché à l'écran pour être enregistré ou imprimé.

5.4 Application « Greffon »

Les interfaces graphiques de l'application sont très importantes, car elles permettent de faciliter le dialogue entre l'homme et la machine ainsi que d'améliorer les performances de l'application. Dans la conception des interfaces de notre application nous avons respecté un ensemble des choix ergonomiques comme la lisibilité, la compréhension, etc.

Dans ce qui suit une présentation des captures écrans des plus importantes tâches de l'application.

5.4.1 Authentification

Cette première capture dans la figure 5.9 présente l'interface d'authentification dans laquelle le néphrologue doit saisir son nom d'utilisateur et son mot de passe pour commencer à utiliser notre application. Cette interface constitue la fenêtre d'accueil de notre application. Cette étape met en valeur l'aspect sécurité : nous vérifions la disponibilité du compte utilisateur et nous lui attribuons les droits et privilèges nécessaires.

Figure 5.9 : Fenêtre d'authentification

Si le nom d'utilisateur introduit, ou le mot de passe, n'est pas valide, alors l'application renvoi un message d'erreur.

5.4.2 Menu Principal

Une fois le néphrologue authentifié, le menu principal s'affiche à l'écran.

Figure 5.10 : Fenêtre de menu principal

Comme montré sur la figure 5.10, le menu contient six fonctionnalités : ajouter receveur, liste receveurs, rechercher receveur, ajouter donneur, liste donneurs et rechercher donneur.

5.4.3 Liste Receveurs

Le néphrologue peut consulter la liste des receveurs qui existent dans le système.

Figure 5.11 : Liste des receveurs

La figure 5.11 représente la liste des receveurs. A travers cette liste, le néphrologue peut afficher le dossier d'un receveur, supprimer un receveur ou affecter un donneur à ce receveur.

5.4.4 Dossier receveur

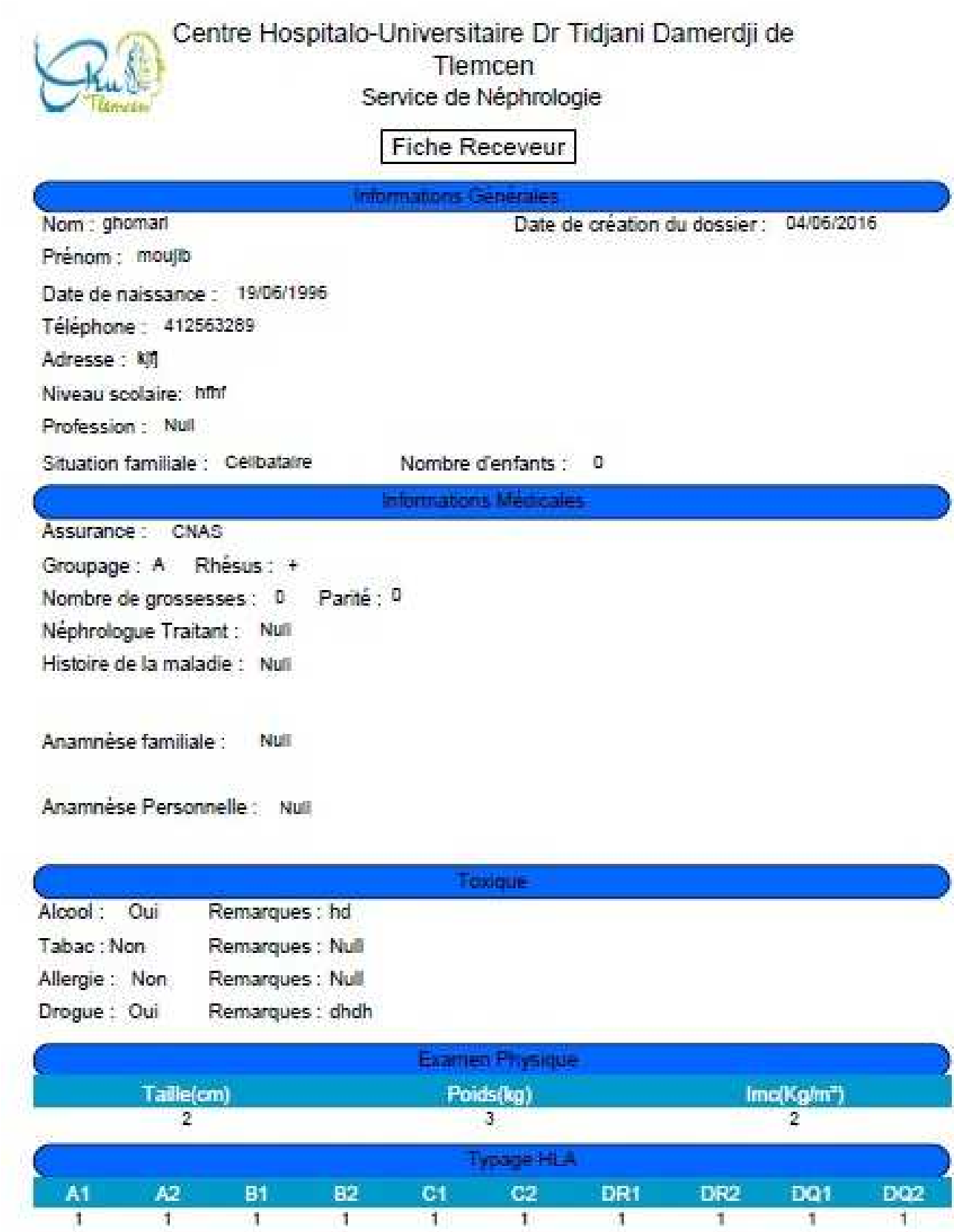
La figure 5.12 représente un examen morphologique. Lorsque le néphrologue clique sur « afficher dossier », la page suivante s'affiche et lui permet de se déplacer dans le menu.

Il peut consulter les dossiers, ajouter examen, modifier, supprimer, ou imprimer.

Figure 5.12 : Examen morphologique

5.4.5 Interface des états

Notre application offre la possibilité d'imprimer le dossier médical des patients.



Centre Hospitalo-Universitaire Dr Tidjani Damerdji de Tlemcen

Service de Néphrologie

Fiche Receveur

Informations Générales

Nom : ghomari — Date de création du dossier : 04/06/2016

Prénom : moujib

Date de naissance : 19/06/1996

Téléphone : 412563289

Adresse : kjfj

Niveau scolaire: hfhf

Profession : Null

Situation familiale : Celibataire — Nombre d'enfants : 0

Informations Médicales

Assurance : CNAS

Groupage : A — Rhésus : +

Nombre de grossesses : 0 — Parité : 0

Néphrologue Traitant : Null

Histoire de la maladie : Null

Anamnèse familiale : Null

Anamnèse Personnelle : Null

Toxique

Alcool :	Oui	Remarques :	hd
Tabac :	Non	Remarques :	Null
Allergie :	Non	Remarques :	Null
Drogue :	Oui	Remarques :	dhdh

Examen Physique

Taille(cm)	Poids(kg)	Imc(Kg/m²)
2	3	2

Typage HLA

A1	A2	B1	B2	C1	C2	DR1	DR2	DQ1	DQ2
1	1	1	1	1	1	1	1	1	1

Figure 5.13 : Rapport de la fiche receveur

La figure 5.13 représente le rapport du receveur qui est généré. L'ensemble des informations personnelles et des examens sont affichées sur ce rapport. Il est exporté au format PDF et peut ensuite être imprimé ou enregistré.

5.5 Contraintes de développement

Comme pour tout développement d'un projet informatique, nous avons été confrontés à des contraintes techniques.

L'utilisation de la version 6 M1 de JBoss nous a causé des problèmes au déploiement du projet. Cette version est défaillante et a été mise a jour. Nous avons donc été contraints à changer de serveur et sommes passés à la version 6.0 Final.

Le choix du JDK est aussi important car les versions antérieures à la version 6 ne permettent pas de créer un projet Java EE.

Pour ce qui est des librairies Struts, nous nous sommes d'abord tournés vers la version la plus récente puis, ayant empêché le bon déploiement du projet, l'avons en fin de compte changé par la version 2.1.6.

Pour que les opérations de modifications et suppressions soient appliquées sur les données des tables en relation avec les données de la table modifiée ou supprimée, il faut définir des annotations dans les entités beans correspondantes. Ces annotations permettent de faire le mapping[4] des relations, qui peuvent être unidirectionnelles ou bidirectionnelles, entre les tables. On doit ajouter l'attribut cascade à cette annotation. Cet attribut permet de spécifier le niveau de propagation souhaité sur les entités liées à une entité dite cible. C'est donc une persistance en cascade.

Une réunion en fin de développement a été organisée avec un néphrologue pour la validation du produit. Malgré l'étude minutieuse des besoins, il peut toujours y avoir une erreur de compréhension ou une mal formulation des besoins. Quelques modifications ont été demandées par le néphrologue d'où l'importance de cette réunion.

[4] Mise en cohérence entre deux types d'informations distincts

Conclusion

Le développement avec les EJB de notre système « Greffon » qui est une application web exploite les fonctionnalités d'un container EJB pour :

- mettre en place une couche de persistance basée sur les **Entity beans**, le langage **JPQL** et la **Java Persistence API**.
- créer des objets métiers à l'aide des **Session beans**.
- faire de la programmation par aspect grâce aux **Interceptors**.

Nous avons mis en place une architecture Java EE qui répond aux besoins classiques d'une application web d'entreprise : interaction avec une base de données, sécurisation des accès, reporting, gestion des utilisateurs, traitements différés, développement en couches, adoption de design patterns…

Ce système sera, dans un premier temps, intégré à l'ordinateur de l'unité de greffe rénale au service de néphrologie du CHU de Tlemcen. Il sera plus tard intégré à plusieurs ordinateurs afin de permettre aux médecins de travailler en parallèle.

Nous allons donner également une formation sur l'utilisation du nouveau système.

6. CONCLUSION GENERALE

Le développement d’une application web dans le domaine médical nous a permis de découvrir non seulement un nombre d’outils informatiques, mais nous a aussi appris l’importance de la sécurité des données, car une erreur informatique peut être fatale.

La multitude de données nous a contraints à utiliser les designs patterns qui aident énormément dans la conception du système et qui donnent une modularité et une flexibilité pour des possibilités d’ajouts de nouveaux changements et d’améliorations futures.

L’utilisation de l’environnement de développement Eclipse nous a permis d’acquérir de nouvelles connaissances en java et de découvrir des plugins, notamment Dali qui permet la génération des entités à partir des tables en base de données. Pour ces dernières, nous avons eu recours à la base de données Oracle 10g Express pour sa capacité de stockage adéquate au projet développé ainsi que pour son aptitude à stocker les images sous bonne qualité.

D’une part, Struts est encore récemment utilisé, ajoutons à cela son intégration avec les EJB qui n’est pas courante, nous avons eu une documentation limitée. Mais d’une autre part, Struts2 a été d’un grand recours et ce grâce à son système de validation de formulaires et d'entrées, simple à mettre en œuvre et à la gestion évoluée du routage ou navigation. De plus, il se base sur l’architecture Modèle-Vue-Contrôleur (MVC) qui permet une séparation et une distribution des tâches entre développeurs, et rend l'application plus flexible (ajouter une vue sans changer le modèle). MVC donne une bonne structure du code et facilite ainsi la maintenance et l'extensibilité : bonne structure aide à diminuer la complexité du code.

L’application «Greffon» a été codée de manière ouverte de telle sorte que l’ajout de nouveaux utilisateurs de types différents et de nouveaux services est possible. Il est aussi possible d’internationaliser « Greffon » et ce, grâce à l'internationalisation pour le développement de sites multilingues pris en charge par STRUTS2.

Références bibliographiques

[1] Dr Peter Mareen. (2012) Medipedia.

[2] Wikipédia,.

[3] Pascal ROQUES, *UML 2 par la pratique étude de cas et exercices corrigés.*, 2006.

[4] Laboratoire SUPINFO, *EJB 3 Des concepts à l'écriture du code.*, 2006.

[5] Laurent DEBRAUWER, *Design Patterns en Java.*, 2013.

[6] Eric PAPET, "Présentation succincte du serveur d'application JBOSS," 2002.

[7] Celinio FERNANDES, *Les EJB 3 (avec Struts 2, JSF 2, JasperReports 3, Flex 3).*, 2010.

Liste des figures

Résumé

Les néphrologues de l'unité de greffe rénale du CHU de Tlemcen connaissent beaucoup de difficultés dans leurs tâches quotidiennes. Une application distribuée, basée sur le principe du client/serveur, est une solution qui permet d'automatiser leur travail. C'est en introduisant le framework STRUTS basé sur le MVC que l'on arrive à distinguer chaque couche de l'architecture 3-tiers. Il offre, de plus, une parfaite abstraction des données en base en adaptant facilement les EJB à sa structure, ce qui permet une bonne modularité et la possibilité d'opérer des changements et maintenances sans conséquents sur le reste du système.

Mots-clés : client/serveur, architecture 3-tiers, Struts2, MVC, EJB.

ملخص

يعاني أطباء وحدة زرع الكلى بمستشفى جامعة تلمسان من صعوبات كثيرة في مهامهم اليومية .و لتنظيم نظام عملهم إعتمدنا على التطبيقات التي تقوم على مبدأ العميل / الخادم المكونة من ثلاث طبقات(العميل, الخادم, قاعدة البيانات (و منه إستعملنا أطر وأدوات التطوير المختلفة التي تمكننا من تمييز هذه الطبقات .الإطار Struts2، الذي يعتمد على نموذج MVCيلبي العروض و احتياجات الزبون المتوقعة، بالإضافة إلى ذلك، يقدم صورة واضحة للمعلومات في قاعدة البيانات و هذا بتبني EJB التي تتكيف بسهولة مع هيكله الذي يتيح تقسيم جيد للعمل ، وبالتالي القدرة على إجراء الصيانة والتغييرات دون التأثير على بقية النظام.

الكلمات المفتاحية : EJB, MVC, Struts2, architecture 3-tiers, Client/serveur.

Abstract

Nephrologists of renal transplant unit of the University Hospital of Tlemcen encounter many difficulties in their daily tasks. A distributed application, based on the principle of client / server, is a solution that automates their work. It is introducing the Struts framework based on MVC that allows distinguishing each layer of the 3-tier architecture. It offers, in addition, a perfect base abstraction of data by easily adapting EJB to its structure, which allows a good modularity and the possibility to make changes and maintenance without affect the rest of the system.

Key words : client/server, 3-tier architecture, Struts2, MVC, EJB.