Institut **A**fricain d'**I**nformatique
Représentation du Niger
(I.A.I-Niger)
Tel: (227) 20 72 56 72
BP: 12 078
www.iai-niger.org

SYNETCOM
Empowered by innovation

MEMOIRE DE FIN DE CYCLE

POUR L'OBTENTION DU DIPLOME D'INGENIEUR DES TRAVAUX INFORMATIQUES

THEME

Conception d'une plateforme web de gestion de don de sang

Présenté par :
M. Ali Gambo MAMANE LAOUALI
M. Amadou Moussa MAHAMADOU
M. Boubacar Boureima MOHAMED

Maître de stage :
M. Chaibou ABDOU
Administrateur Général

Année Académique : 2018 - 2019

Dédicaces

Nous dédions ce présent mémoire

A

Nos parents

Pour les sacrifices déployés à nos égards ; pour leur patience leur amour et leur confiance en nous. Ils ont tout fait pour notre bonheur et notre réussite. Qu'ils trouvent dans ce modeste travail, le témoignage de notre Profonde affection et de notre attachement indéfectible. Nulle dédicace ne puisse exprimer ce que nous leur devons.

Que Dieu leur réserve la bonne santé et une longue vie.

A

Nos amis

En témoignage de nos sincères reconnaissances pour les efforts Qu'ils ont consentis pour nous soutenir au cours de nos études. Que Dieu nous garde toujours unis.

A

Toutes les personnes qui ont contribué de près ou de loin à la réalisation de présent projet.

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui nous voudrons témoigner toute notre reconnaissance.

En préambule à ce mémoire nous remercions ALLAH qui nous aide et nous donne la santé, la patience ainsi que le courage durant ces longues années d'étude.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce mémoire ainsi qu'à la réussite de cette formidable année universitaire.

Ces remerciements vont également à l'égard du corps professoral et administratif de notre école IAI-Niger, pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Nous tenons à remercier sincèrement Monsieur Chaibou Abdou, qui, en tant que Directeur de mémoire, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi pour l'inspiration, l'aide et le temps qu'il a bien voulu nous consacrer et sans qui, ce mémoire n'aurait jamais vu le jour.

On n'oublie pas nos très chers parents qui ont toujours été là pour nous, pour leur contribution, leur soutien et leur patience. Vous nous avez donné un magnifique modèle de labeur et de persévérance. Nous sommes redevables d'une éducation dont on est fier.

Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours encouragé.

Merci à tous et à toutes.

Sigles et Abréviations

CTS : Centre de Transfusion Sanguine

TIC : Technologies d'Information et la Communication

NTIC : Nouvelles Technologies d'Information et de la Communication

SMS : Short Message Service (service de messages succincts)

UML : Unified Modeling Language

UP: Unified Process (Processus Unifié)

T2UP: Two Tracks Unified Process

PHP: Hypertext Preprocessor

HTML : Hyper Text Markup Language

CSS : Cascading Style Sheets (feuilles de style en cascade)

JS : JavaScript

POO : Programmation Orienté Objet

RSM : Responsable Sélection Médical

AS : Assistant Médical

CTS : Centre de Transfusion Sanguine

CNTS : Centre National de Transfusion Sanguine

PSL : Produit Sanguin Labile

Table des tableaux

Table des figures

Table des matières

Introduction

La révolution des technologies de l'information et de la communication (TIC), a engendré au cours de ces dernières années une progression notable de la création et de l'usage des applications de gestion dans la vie courante et professionnelle. Dans ce contexte de transformation numérique, et afin d'aider les centres de transfusion sanguine (CTS) à mettre en place un système de gestion automatique, SYNETCOM a initié un projet de développement d'une plateforme web standard qui répondra aux attentes des gestionnaires des centres de transfusion sanguine objet de la présente étude.

La présente étude qui nous a été soumise dans le cadre de notre projet de fin d'étude pour l'obtention du diplôme d'ingénieurs des travaux informatiques, vient pour répondre aux besoins d'un secteur de base intéressant et très sensible, à savoir la santé.

En effet, la santé fait partie des domaines qui sont à la traine dans l'adoption des outils TIC alors qu'il en a plus besoin compte tenu du volume d'information géré tant au niveau des dons de sang qu'au niveau des donneurs mais aussi au niveau des différents acteurs impliqués. C'est dans cette optique qu'il nous a été proposé comme thème de projet de fin d'études « **Conception d'une plateforme web de gestion de don de sang** ».

Le présent rapport reflète et décrie la démarche suivie pour mettre en place ce système informatisé demandé :

-La première partie décrie de façon globale la présentation du centre d'accueil ainsi que du projet ;

-La deuxième partie présente de manière globale l'analyse et la conception du projet ;

-La troisième partie est consacrée à la réalisation où nous présentons l'environnement de développement et quelques captures d'écrans.

Partie I : Etude Préalable

Chapitre I : Présentation du centre d'accueil

I. Présentation de SYNETCOM

SYNETCOM est une jeune entreprise spécialisée dans les domaines des nouvelles technologies de l'information et de la communication.

Vecteur d'innovation, levier d'efficacité, accélérateur de développement, grâce à son expérience et à son anticipation permanente sur les besoins des clients, SYNETCOM agit en véritable partenaire technologique pour lever les défis et les difficultés auxquels heurtent ses partenaires et optimiser les performances de leur système de gestion.

Sa vocation est d'apporter les ressources nécessaires pour faire bénéficier au mieux à ses partenaires les atouts des nouvelles technologies.

SYNETCOM réfléchit aux problèmes du marché et conçoit les solutions novatrices et les services d'accompagnement adaptés aux besoins des clients.

II. Organigramme

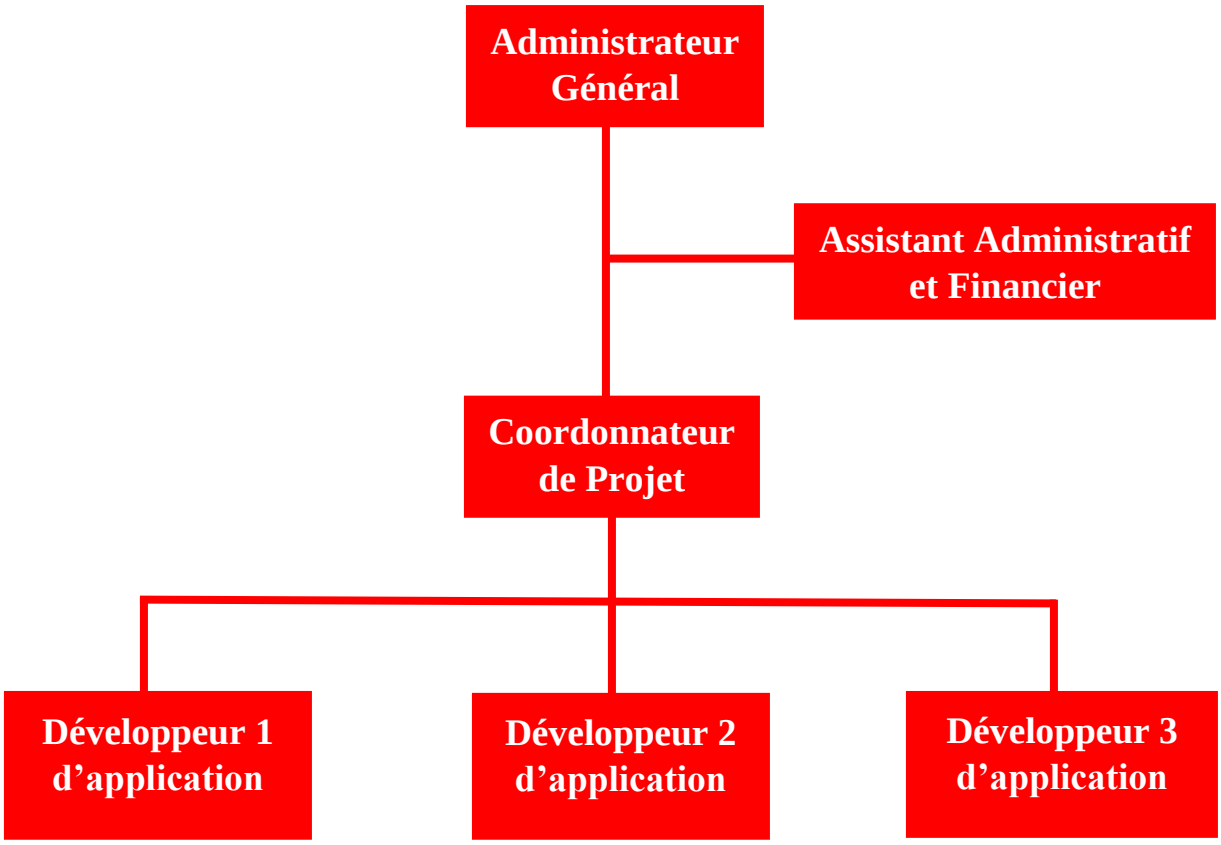


Figure 1:Organigramme de SYNETCOM

Chapitre II : Présentation du projet

I. Contexte du projet

La complexité croissante des systèmes informatiques a conduit les concepteurs à s'intéresser aux méthodes de développement. Ainsi, l'utilisation de l'outil informatique est une réalité de plus en plus présente dans les structures.
Le domaine de la santé n'a cessé ces dernières années d'évoluer et implicitement de se complexer, il devient d'autant plus difficile de pouvoir répondre aux différentes problématiques rencontrées sans avoir recours aux méthodes de gestion et d'aide à la décision ; en particulier, l'approche de modélisation et simulation a prouvé son efficacité.

Cette étude se focalise sur la composante de la transfusion sanguine de manière générale et a pour objectif de contribuer plus particulièrement à une meilleure compréhension du fonctionnement organisationnel d'un centre de transfusion sanguine. Ce document ne concerne que ces aspects.

II. Travail attendu

Ce projet a pour objectif principal de proposer une solution à un problème concret, et ceci en partant d'une définition des besoins. Le principal objectif du système est d'assurer les dons de sang, ainsi que le suivi de la traçabilité des donneurs entre les différents dons effectués et de conserver ces informations pour consultation ultérieure pour une prise en charge continue, coordonnée et sécurisée des documents.

Le système permettra globalement de :

- Gérer les donneurs,
- Gérer les prélèvements,
- Gérer la catégorisation des dons,
- Gérer le stock des poches de sang,
- Gérer le suivi de la traçabilité,
- Planifier les points de collectes (fixes et mobiles),
- Réception des notifications par les donneurs via Mail et/ou SMS sur des mises à jour d'informations (Résultats d'analyse, Urgences, Remerciements…),
- Donner les droits d'accès aux fonctionnalités par profils d'utilisateurs.

Partie II : Travaux d'Analyse et Conception

Chapitre III : Analyse

A. Méthode d'analyse

Une méthodologie de développement est un cadre utilisé pour structurer, planifier et contrôler le développement d'une application. En effets, nous disposons de nos jours d'un ensemble de méthodologie chacune respectant un certain nombre de principe pour aboutir à la réalisation d'un projet. Ainsi nous allons parler principalement de quatre types de méthodes de conception.

I. Pourquoi utiliser une méthode d'analyse

On appelle « gestion de projet » éventuellement « conduite de projet » l'organisation méthodologique mise en œuvre pour faire en sorte que l'ouvrage réalisé par le maitre d'œuvre réponds aux attentes du maitre d'ouvrage et qu'il soit livré dans les conditions de coût et de délais prévus initialement.

En effet**,** les projets sont gérés avec la méthode dite « classique » qui se caractérise pour recueillir les besoins, définir le produit, le développer et le tester avant de le livrer. On parle alors ici d'une approche prédictive. Comme son nom l'indique, il s'agit ici de prévoir des phases séquentielles où il faut valider l'étape précédente pour passer à la suivante.

Le chef de projet doit alors s'engager sur un planning précis de réalisation du projet en prévoyant des jalons de débuts et fins de phases ainsi que les tâches à effectuer. Il faut tout faire bien du premier coup car elle ne peut pas permettre de retours en arrière. Une décision ou un problème rencontré dans une phase peuvent remettre en cause partiellement ou totalement les phases précédentes validées. Dans un cycle « en cascade » les risques sont détectés tardivement puisqu'il faut attendre la fin du développement pour effectuer la phase de test. Plus le projet avance, plus l'impact des risques augmente : il sera toujours plus difficile et coûteux de revenir en arrière lorsqu'on découvre une anomalie tardivement. Par conséquent, comment peut-on augmenter la satisfaction du client en facilitant la gestion de projet et améliorant la qualité de développement ? Pour se faire, nous allons voir d'autres méthodes modernes plus adéquates.

II. Classification des méthodologies

II.1. Méthodes fonctionnelles

Les méthodes fonctionnelles ou cartésiennes sont des démarches consistant à rechercher et à caractériser les fonctions offertes par un produit pour faire les besoins de son utilisateur. Il est aussi défini comme étant une méthode permettant à décomposer hiérarchiquement une application en un ensemble de sous applications. Parmi les fonctions de chacune de celles-ci sont affinées alternativement en sous fonctions simples à coder dans un langage de programmation donné.

L'une de ces méthodes, on peut noter FAST (Fonction Analyse System Technique), SADT (Structurel-Analyse-Design-Technique) Warnier …

Forces de ces méthodes :

- Simplicité du processus de conception préconisée ;
- Adéquations à capturer les besoins de l'utilisateur ;
- Capacités de produire des solutions à plusieurs niveaux d'abstraction.

Faiblesses de ces méthodes

- Effort d'analyse concentré sur les fonctions (négligeant la cohérence des données) ;
- Règles de décomposition non explicites ;
- Difficulté à tenir compte des interactions non hiérarchiques dans les systèmes complexes.

II.2. Méthodes classiques

Souvent spécialisées pour la conception d'un certain type de systèmes, les méthodes systémiques définissent différents niveaux de préoccupation ou d'abstraction et proposent de nombreux modèles complémentaires. Autrement dit ces méthodes proposent une double démarche de modélisation, celles des données et des traitements. Elles sont influencées par les systèmes de gestion de bases de données SGBD. On a comme exemple MERISE, AXIAL RACINE...

Forces de ces méthodes :

- Les concepts sont peu nombreux et simples
- Elle est assez indépendante vis-à-vis de la technologie,
- Bonne adaptation à la modélisation des données et à la conception des bases de données,

Faiblesses de ces méthodes

- Double démarche de conception celles des données et des traitements,
- Impossible de fusionner les deux aspects (données et traitements).

II.3. Méthodes unifiées :

Les méthodes unifiées sont des méthodes de développement pour les logiciels orientés objets. Plus exactement, ce sont les meilleures pratiques du développement objet suivies pour la réalisation d'un système. Donc le Processus Unifié (UP) est un processus de développement logiciel « Itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques » :

- **Itératif et incrémental** : le projet est découpé en itérations de courte durée (environ 1 mois) Qui aident à mieux suivre l'avancement global. A la fin de chaque itération, une partie exécutable du système final est produite, de façon Incrémentale.

- **Centré sur l'architecture** : tout système complexe doit être décomposé en parties Modulaires afin de garantir une maintenance et une évolution facilitées. Cette architecture (Fonctionnelle, logique, matérielle, etc.) doit être modélisée en UML et pas seulement Documentée en texte.
- **Piloté par les risques** : les risques majeurs du projet doivent être identifiés au plus tôt, mais surtout levés le plus rapidement possible. Les mesures à prendre dans ce cadre déterminent l'ordre des itérations.
- **Conduit par les cas d'utilisation** : le projet est mené en tenant compte des besoins et des exigences des utilisateurs. Les cas d'utilisation du futur système sont identifiés, décrits avec précision et priorisés.

II.3.1. Méthodes UP :

Le Processus Unifié (PU ou UP en anglais pour Unified Process) est une méthode de développement logiciel construite sur UML ; elle est itérative et incrémentale, centrée sur l'architecture, conduite par les cas d'utilisation et pilotée par les risques.

Itérative et incrémentale : la méthode est itérative dans le sens où elle propose de faire des itérations lors de ses différentes phases, ceci garantit que le modèle construit à chaque phase ou étape soit

- Affiné et amélioré. Chaque itération peut servir aussi à ajouter de nouveaux incréments.

- Conduite par les cas d'utilisation : elle est orientée « utilisateur » pour répondre aux besoins de celui-ci.

- Centrée sur l'architecture : les modèles définit tout au long du processus de développement vont contribuer à établir une architecture cohérente et solide.

Pilotée par les risques : en définissant des priorités pour chaque fonctionnalité, on peut minimiser les risques d'échec du projet.

II.3.2. Méthode 2TUP :

2TUP est un processus de développement logiciel qui implémente le Processus Unifié. Le **2TUP** propose un cycle de développement en Y, qui dissocie les aspects techniques des aspects fonctionnels. Il commence par une étude préliminaire qui consiste essentiellement à :

- ✓ Identifier les acteurs qui vont interagir avec le système à construire ;
- ✓ Identifier les messages qu'échangent les acteurs et le système ;
- ✓ Produire le cahier des charges ;

- ✓ Modéliser le contexte (le système est une boîte noire, les acteurs l'entourent et sont reliés à lui, sur l'axe qui lie un acteur au système on met les messages que les deux s'échangent avec le sens).

Le processus s'articule ensuite autour de 3 phases essentielles :

Figure 2 : Le système d'information soumis à deux types de contraintes

La branche gauche (fonctionnelle) comporte :

- La capture des besoins fonctionnels, qui produit un modèle des besoins focalisé sur le métier des utilisateurs. Elle qualifie au plus tôt le risque de produire un système inadapté aux utilisateurs. De son côté, la maîtrise d'œuvre consolide les spécifications et en vérifie la cohérence et l'exhaustivité.
- L'analyse, qui consiste à étudier précisément la spécification fonctionnelle de manière à obtenir une idée de ce que va réaliser le système en termes de métier. Les résultats de l'analyse ne dépendent d'aucune technologie particulière.

La branche droite (architecture technique} comporte :

- La capture des besoins techniques, qui recense toutes les contraintes et les choix dimensionnant la conception du système. Les outils et les matériels sélectionnés ainsi que la prise en compte de contraintes d'intégration avec l'existant conditionne généralement des prés requis d'architecture technique.
- La conception générique, qui définit ensuite les composants nécessaires à la construction de l'architecture technique. Cette conception est complètement indépendante des aspects fonctionnels. Elle a pour objectif d'uniformiser et de réutiliser les mêmes mécanismes pour tout un système. L'architecture technique

Elle construit le squelette du système informatique et écarte la plupart des risques de niveau technique. L'importance de sa réussite est telle qu'il est conseillé de réaliser un prototype pour assurer sa validité.

La branche du milieu comporte :

- **La conception préliminaire**, qui représente une étape délicate, car elle intègre le modèle d'analyse dans l'architecture technique de manière à tracer la cartographie des composants du système à développer, la conception détaillée, qui étudie ensuite comment réaliser

chaque composant ; l'étape de codage, qui produit ces composants et teste au fur et à mesure les unités de code réalisées

- **La conception détaillée**, qui étudie ensuite comment réaliser chaque composant ;
- **Le codage et tests**, qui produisent ces composants et teste au fur et à mesure les unités de code réalisées ;
- **L'étape de recette**, qui consiste enfin à valider les fonctions du système développé.

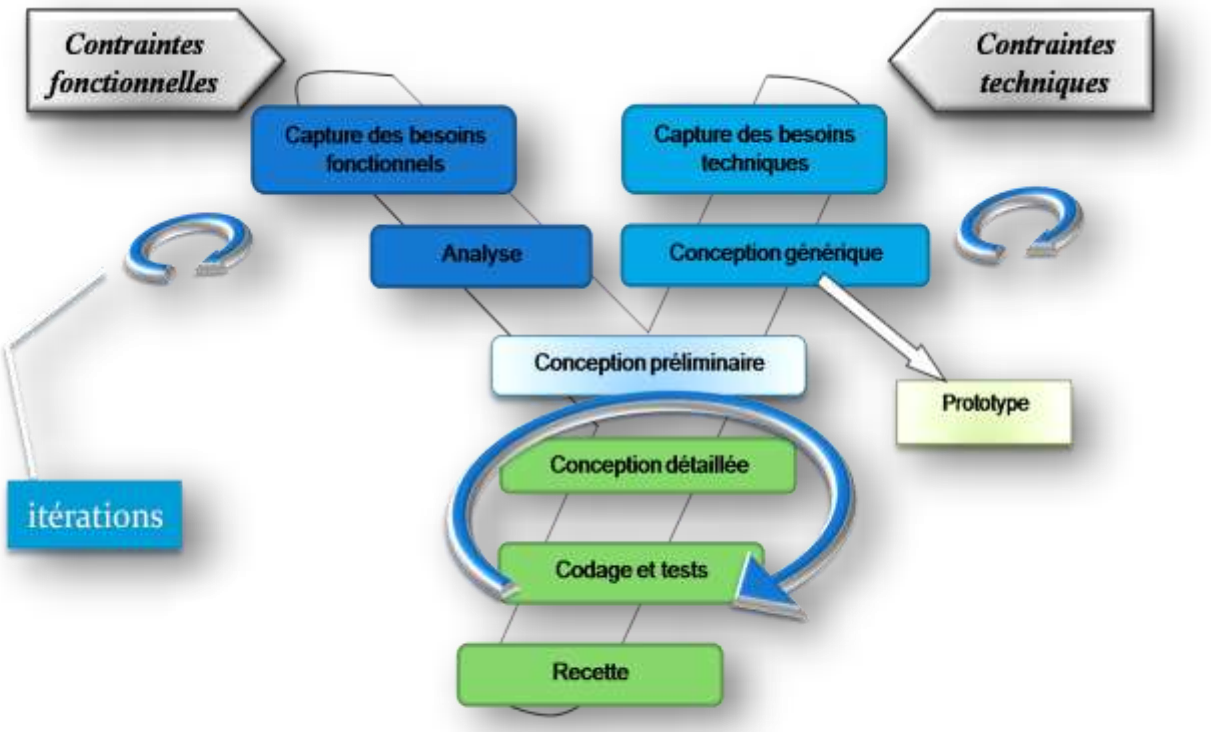


Figure 3 : Le processus de développement en Y

II.4. Méthodes agiles :

Ces méthodes constituent un gain en productivité ainsi qu'un avantage compétitif tant du côté client que du côté du fournisseur. En effet, **ces** méthodes de développement dites « méthodes agiles » visent à réduire le cycle de vie du logiciel (donc accélérer son développement) en développant une version minimale, puis en intégrant les fonctionnalités par un processus itératif basé sur une écoute client et des tests tout au long du cycle de développement.

Parmi ces méthodes nous pouvons citer : Scrum, XP (eXtreme Programming), RAD (Rapid Application Development), etc.…

II.4.1. XP :

Extrême Programming, ou XP, est une méthode agile de gestion de projet particulièrement bien adaptée aux projets de développement informatique. Elle a été conçue par Kent Beck pour accélérer les développements alors qu'il travaillait pour la société Chrysler. L'idée lui est venue alors qu'il devrait intervenir sur un logiciel de paie écrit en langage « Small talk » ayant accumulé une dette technique considérable, le rendant particulièrement complexe à maintenir et à faire évoluer.

Le principe fondamental de la méthode XP est de faire collaborer étroitement tous les acteurs du projet et d'opter pour des itérations de développement très courtes. La planification des tâches reste très souple et l'estimation des charges simplifiée par des projections à très court terme. Ainsi la correspondance entre ce qu'attendent le client et les réalisations est garantie. Les fonctionnalités sont livrées régulièrement, afin d'être testées et validée au travers de prototypes opérationnels. L'Extrême Programming préconise également le travail en binôme des développeurs, facilitant ainsi la production d'un code simple, facilement lisible et maintenable.

II.4.2. Scrum :

Scrum est considéré comme un cadre ou « Framework » de gestion de projet. Ce cadre est constitué d'une définition des rôles, de réunions et d'artefacts. Le mot Scrum est un terme emprunté au rugby qui signifie « mêlée » encore appelée « Morning ». Dans cette méthode, on focalise l'équipe de façon itérative sur un ensemble de fonctionnalités. En effet, Scrum **définit 3 rôles :**

- Le propriétaire du produit « Product Owner » qui porte la vision du produit à réaliser (représentant généralement le client) ;
- Le directeur du produit « Scrum Master » garant de l'application de la méthodologie Scrum ;
- L'équipe de développement qui réalise le produit.

La vie d'un projet Scrum est rythmée par un ensemble de réunions clairement définies et strictement limitées dans le temps (timeboxing):Planification du Sprint (Sprint = itération) : au cours de cette réunion, l'équipe de développement sélectionne les éléments prioritaires du « Product Backlog » (liste ordonnancée des exigences fonctionnelles et non fonctionnelles du projet) qu'elle pense pouvoir réaliser au cours du sprint (en accord avec le « Product Owner »).Revue de Sprint : au cours de cette réunion qui a lieu à la fin du sprint, l'équipe de développement présente les fonctionnalités terminées au cours du sprint et recueille les feedbacks du Product Owner et des utilisateurs finaux. C'est également le moment d'anticiper le périmètre des prochains sprints et d'ajuster au besoin la planification de release (nombre de sprints

restants).
Rétrospective de Sprint : la rétrospective qui a généralement lieu après la revue de sprint est l'occasion de s'améliorer (productivité, qualité, efficacité, conditions de travail, etc.) à la lueur du "vécu" sur le sprint écoulé (principe d'amélioration continue).Mêlée quotidienne : il s'agit d'une réunion de synchronisation de l'équipe de développement qui se fait debout (elle est aussi appelée "stand up meeting") en 15 minutes maximum au cours de laquelle chacun répond principalement à 3 questions :« Qu'est-ce que j'ai terminé depuis la dernière mêlée ? Qu'est-ce que j'aurai terminé d'ici la prochaine mêlée ? Quels obstacles me retardent ? »

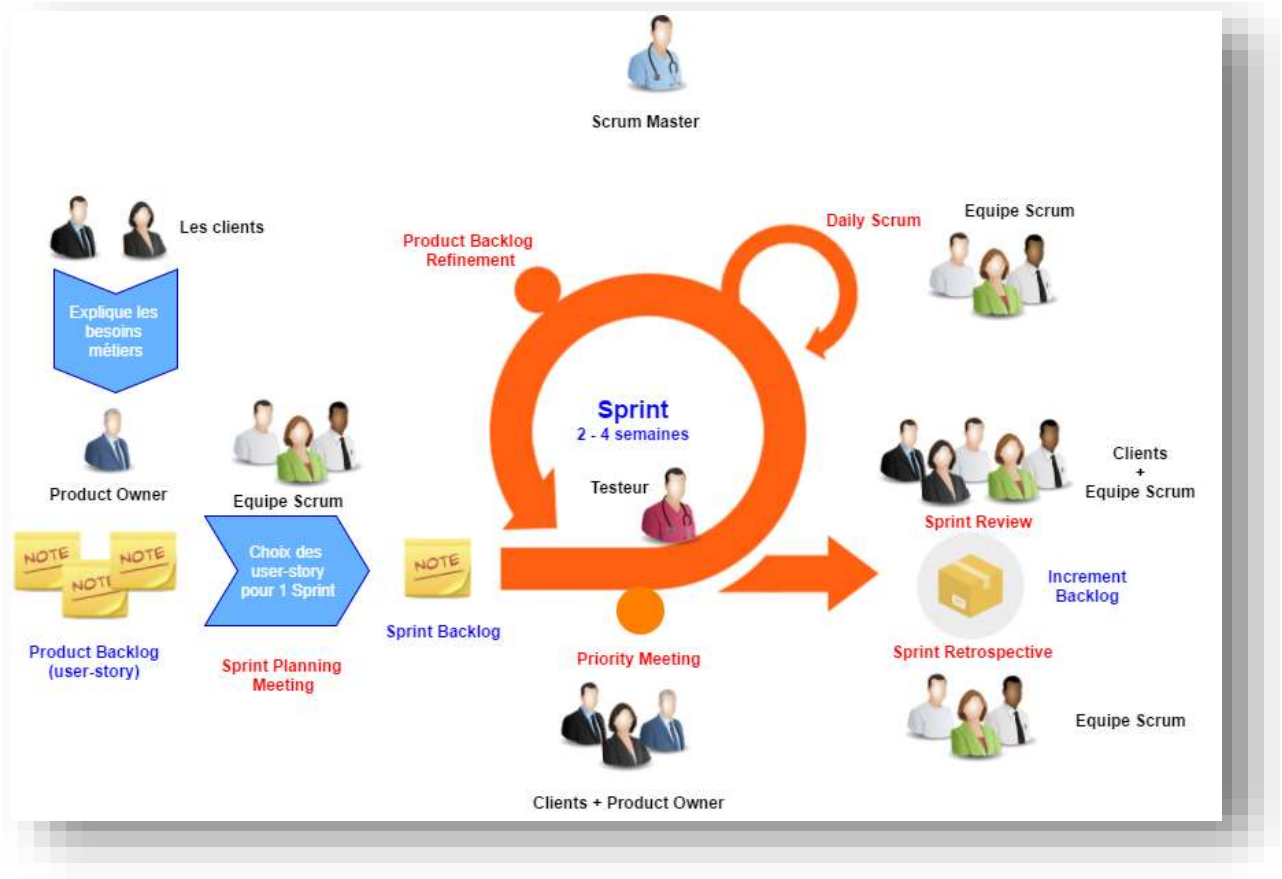


Figure 4 : La méthode scrum

III. Choix de la méthodologie :

III.1. Méthodologie de conception :

Après l'étude des différentes méthodologies, nous avons opté pour Scrum et UP pour les raisons qui suivent :

- Scrum convient aux équipes ayant un nombre de développeurs réduits. Ceci est le cas de notre projet ;

- La progression des tâches s'effectue pendant une durée de développement courte ;
- UP s'applique à tous les types de projet informatique pour les suggestions et remarques ;
- UP est une méthode itérative et incrémental ;
- UP est essentiellement basée sur la satisfaction des besoin set des exigences du clients et des utilisateurs ;
- UP se base sur le diagramme des cas d'utilisation qui illustre au mieux les besoins fonctionnels et opérationnels auxquels doivent répondre le produit final.

III.2. Présentation du langage UML :

Les auteurs des méthodes (OOSE de Jacobson, OMT de Rumbaugh, BOOCH de Grady Booch) se sont fixé des objectifs dont entre autres la représentation des systèmes entiers par des concepts objet ; l'établissement d'un couplage explicite entre les concepts et les interfaces exécutables et la création d'un langage de modélisation utilisable à la fois par les humains et les machines et adaptés aux systèmes simples et complexes. D'où la naissance d'UML qui est donc une norme du langage de modélisation objet qui a été publiée, dans sa première version en septembre 1997 par l'OMG (Object Management Group), instance de normalisation internationale du domaine de l'objet. UML est un langage de modélisation graphique et textuel destiné à comprendre et à décrire des besoins, spécifier et documenter des systèmes, esquisser architectures logicielles, concevoir des solutions et communiquer des points des vues. De plus UML comble une lacune importante des technologies objet, il permet d'exprimer, d'élaborer et de modéliser au sens de la théorie des langages. De ce fait il contient les éléments constitutifs de ces derniers : concepts, une syntaxe et une sémantique.

III.2.1. Les diagramme UML :

UML définit 13 types de diagrammes divisés en deux catégories : les diagrammes structurels ou statiques rassemblent :

- Diagramme de classe (Class Diagram) ;
- Diagramme d'Objet (Object Diagram) ;
- Diagramme de composant (Composent Diagram) ;
- Diagramme de déploiement (Deployment Diagram) ;
- Diagramme de paquetage (Package Diagram)
- Diagramme de structure composite (Composite Structure Diagram).

Les diagrammes comportementaux qui rassemblent :

- Diagramme des cas d'utilisation (Use Case Diagram).
- Diagramme d'état-transitions (State Machine Diagram) ;
- Diagramme d'activité (Activity Diagram) ;
- Diagramme de séquence (Sequence Diagram) ;

- Diagramme de communication (Communication Diagram) ;
- Diagramme global d'interaction (Interaction Overview Diagram) ;
- Diagramme de temps (Timing Diagram) ;

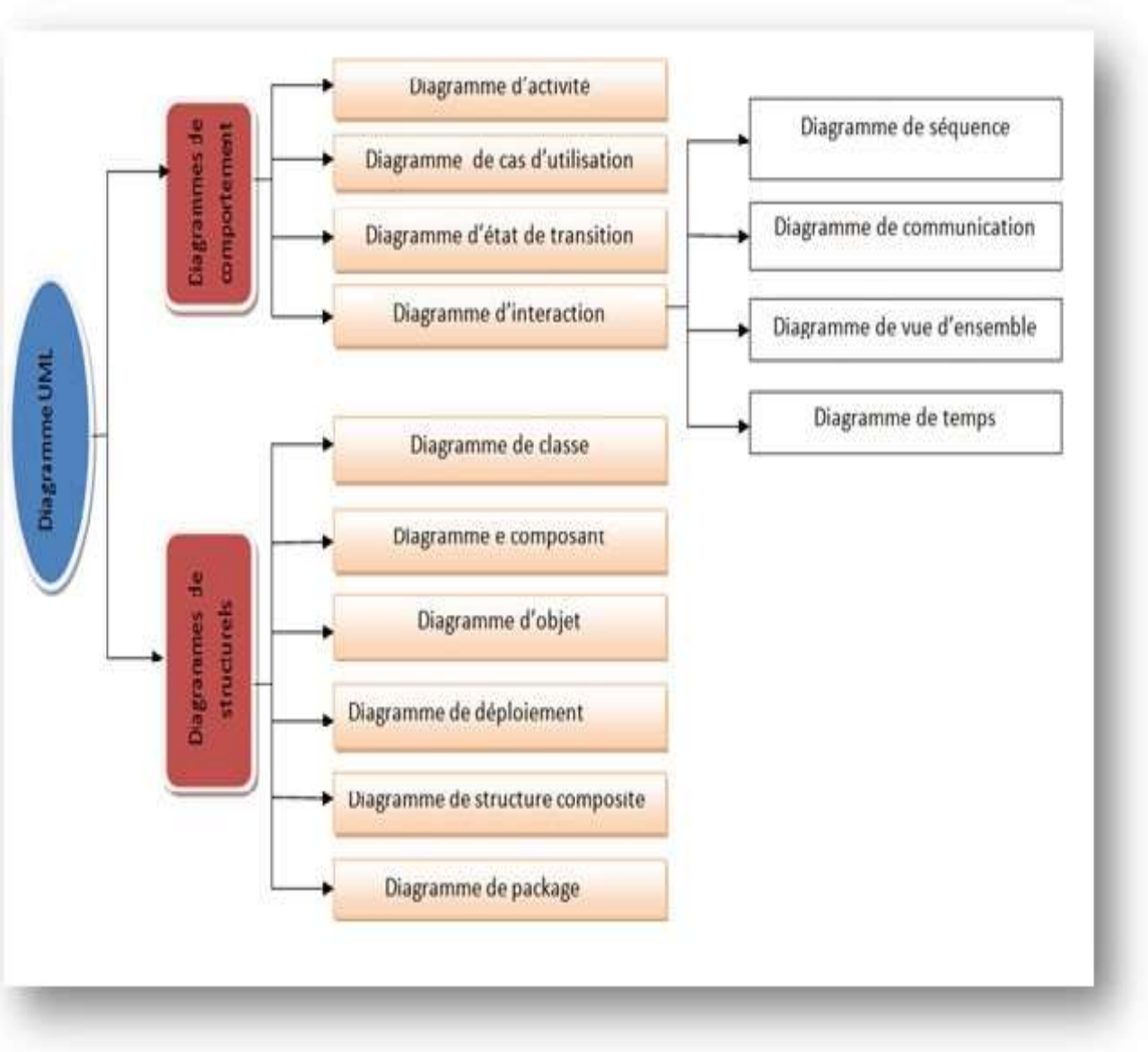


Figure 5 : Les diagrammes UML

IV. Planning prévisionnel

La clé principale de la réussite d'un projet est un bon planning. En effet, le planning aide à bien subdiviser le travail et séparer les taches à réaliser, il offre une meilleure estimation et gestion de temps nécessaire pour chaque tâche. De plus, il donne assez de visibilité permettant d'estimer approximativement la date d'achèvement de chaque tâche.

Nous avons choisi comme outil pour la planification de notre projet Gantt Project qui est un logiciel de gestion de projet.

Gantt Project permet de planifier les projets et les ressources, et d'assurer le suivi des projets durant leur réalisation. Il permet de réaliser une représentation graphique du déroulement d'un projet et de rendre compte de son avancement.

Tâches	Début	Durée	Fin
Etude du cahier de charge	Lundi 8 juillet 2019	10 jours	Vendredi 19 juillet 2019
Apprentissage du Framework	Lundi 1 juillet 2019	20 jours	Vendredi 26 juillet 2019
Conception du système	Lundi 29 juillet 2019	25 jours	Vendredi 30 août 2019
Développement module 1 (Gestion des planifications des collectes	Lundi 2 septembre 2019	3 jours	Mercredi 4 septembre 2019
Développement module 2 (Gestion des donneurs, Gestion des dossiers médicales)	Jeudi 5 septembre 2019	10 jours	Mercredi 18 septembre 2019
Développement module 3 (Gestion de stock des poches de sang)	Jeudi 19 septembre 2019	8 jours	Lundi 30 septembre 2019
Rédaction du rapport	Lundi 8 juillet 2019	63 jours	Mercredi 2 octobre 2019

Figure 6 : Planning du projet

B. Recueil des besoins fonctionnels

Cette phase correspond à une recherche sur le terrain pour bien définir le cadre de notre système. Nous avons effectué plusieurs recherches pour identifier au mieux les besoins auxquels l'application doit répondre, et ceci afin de répondre aux attentes des potentiels utilisateurs.

A cet effet nous sommes allés chercher les informations à travers des interviews menées avec le Directeur Général de SYNETCOM et un responsable du centre national de transfusion sanguine (CNTS) lors de notre visite au CNTS afin d'effectuer un don de sang.

Tout ceci dans le but de comprendre le fonctionnement de don de sang, et avoir une idée de comment s'effectue la gestion d'un centre de transfusion sanguine :

Les centres de transfusion sanguine sont responsables d'assurer la collecte, le traitement et la distribution des produits sanguins labiles.

Le CNTS est situé au sein de la capitale où nous avons constaté une demande très importante en produits sanguins labiles (PSL).
Les donneurs qui se présentent au centre de transfusion sanguine (CTS) passent d'abord par l'accueil pour s'enregistrer et ensuite passent à l'entretien médical qui détermine si le donneur est apte à donner son sang, dans ce cas-là la prochaine étape et la salle de prélèvement ou l'opération de don se déroule.

À la fin le donneur profite d'une collation avant de quitter le centre pendant que le sang collecté est transféré pour être analysé et séparé en produits sanguins labiles (PSL) qui va être stocké en attendant d'être livré aux malades.

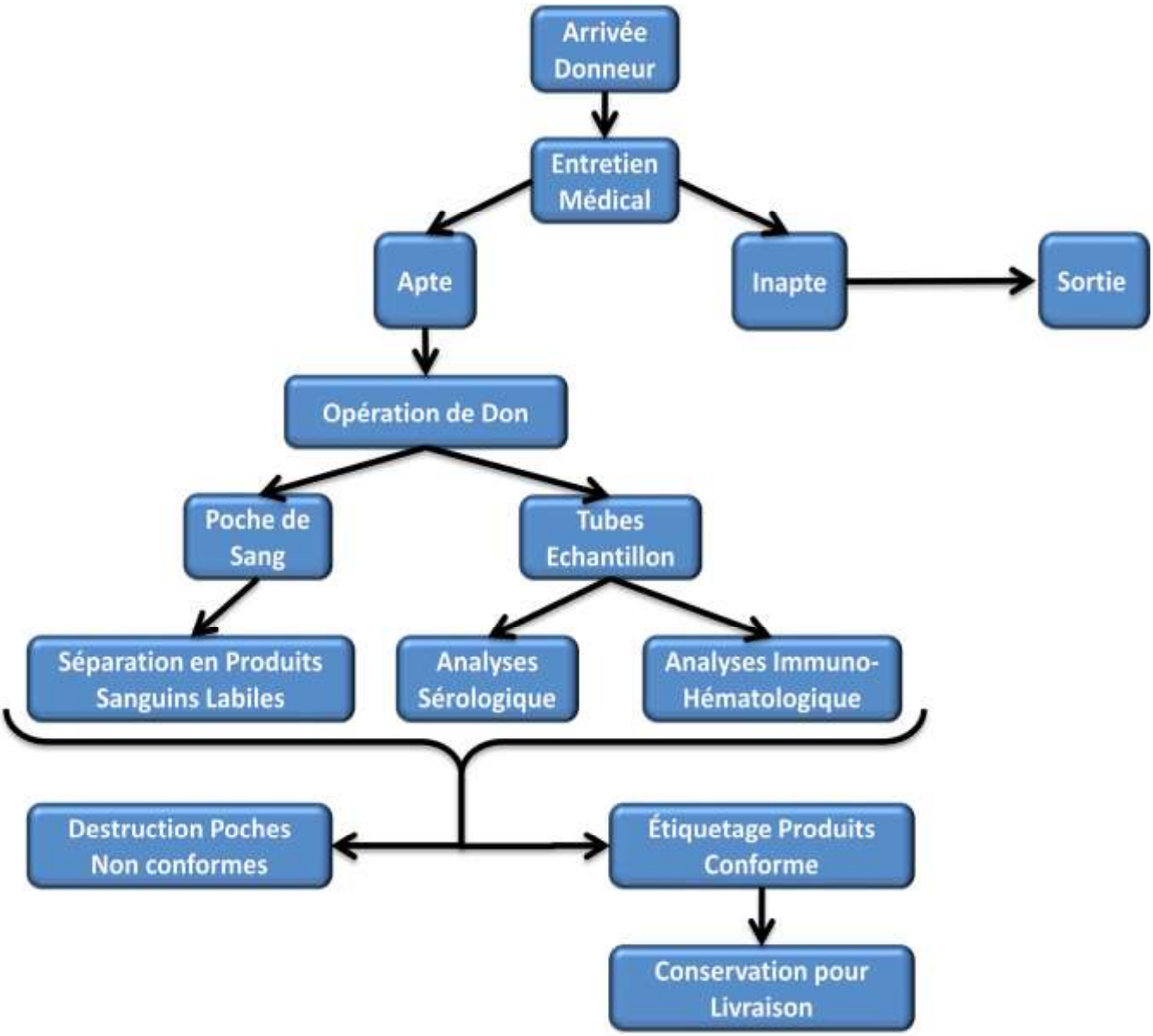


Figure 7 : Différents Processus d'un CTS

C. Analyse des besoins

I. Les acteurs du système

I.1. Définition

Un acteur représente l'abstraction d'un rôle joué par des entités externes (utilisateur, dispositif matériel ou autre système) qui interagissent directement avec le système étudié. Un acteur peut consulter et/ou modifier directement l'état du système, en remettant et/ou en recevant des messages éventuellement porteurs de données.

I.2. Identification des acteurs

Les acteurs qui interagissent avec le système dans un premier temps sont :

- Le préleveur :
- Le laborantin :
- L'assistant médical :
- Le responsable de la sélection médical :
- L'administrateur : l'administrateur a accès à toutes les fonctionnalités de bases et avancées de l'application. Il effectue les paramétrages de base. Il crée les profils utilisateurs et attribue les droits d'accès.

II. Le diagramme de contexte

Le diagramme de contexte statique délimite le domaine d'étude en précisant ce qui est à la charge du système et en identifiant l'environnement extérieur au système étudié avec lequel ce dernier communique. Ses composants sont :

- Les acteurs externes. Un acteur externe est une entité externe au système étudié qui interagit avec le système.
- Un processus unique symbolisant le Système Information étudié :
- Echange entre le système étudié et son environnement

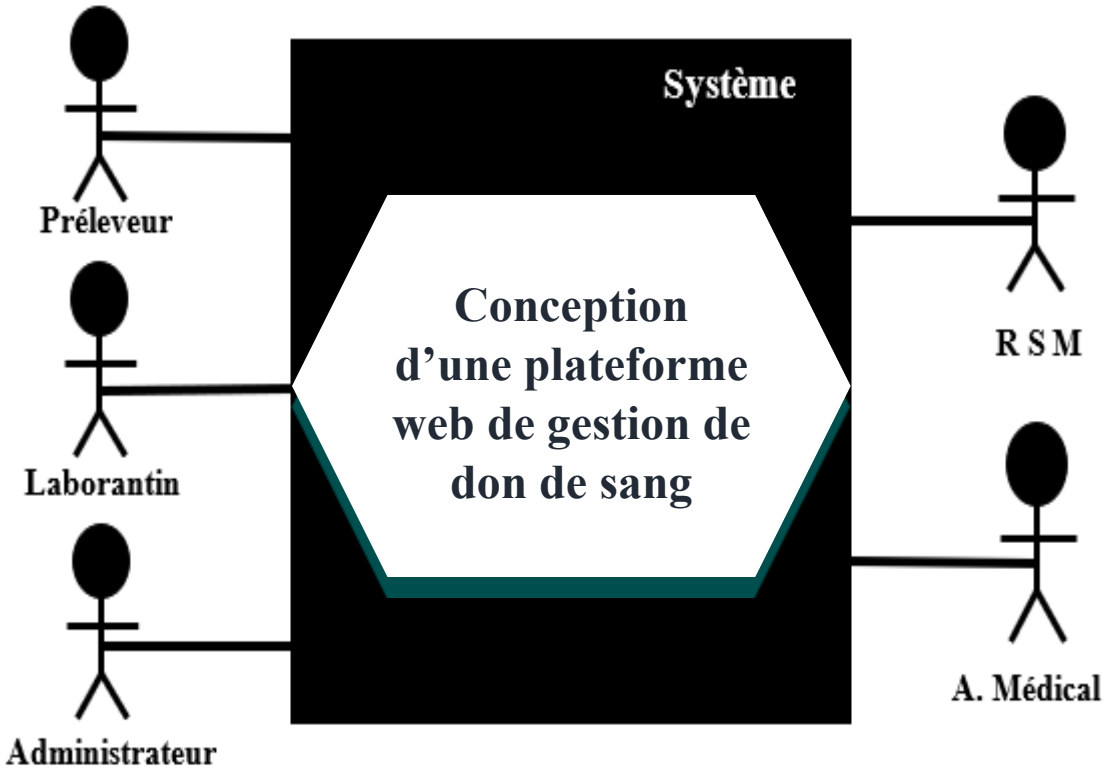


Figure 8 : Diagramme de contexte statique

II.1. Identification des messages

Définition : un message représente la spécification d'une communication unidirectionnelle entre les objets qui transporte de l'information avec l'intention de déclencher une activité chez le récepteur.

Les messages répertoriés entre le système et ses acteurs sont :

Message émis par le système :

- La liste des donneurs ajouté.
- La liste des donneurs ayant été prélevé.
- Les prélèvements effectués.
- Les dossiers médicaux des donneurs.
- La liste des dons acceptés et rejetés à la fin d'un prélèvement.
- La liste des donneurs apte au don.
- La liste des donneurs inapte au don.
- Les notifications sur les mises à jour d'informations (remerciement, résultat, urgence).

- Les informations du donneur liées au don.
- Les statistiques relatives aux informations enregistrées.
- Le nombre de poche de sang en stock par groupage.
- La liste des demandes de poche de sang.
- Les plannings des collectes.
- La liste des planifications des collectes.

Message reçus par le système :

- Les créations, modifications, suppressions de profils utilisateurs.
- Les créations, modifications de dossiers médicales des donneurs.
- Les paramétrages des envois des notifications.
- Les éditions des planifications des collectes.
- Les éditions des donneurs inapte au don.
- Les ajouts, modifications, des dons des donneurs.
- Les ajouts, modifications, suppressions des donneurs.

II.2. Modélisation du contexte

A partir des informations obtenues lors des deux précédentes étapes, nous allons modéliser le contexte de notre application. Ceci va nous permettre dans un premier temps, de définir les fonctionnalités spécifiques de chaque acteur dans le système :

Utilisateurs finaux	Description des besoins fonctionnels
Administrateur	Cet utilisateur a accès à toutes les fonctionnalités de bases et avancées de l'application. L'application doit permettre à l'administrateur de : ▪ S'authentifier ▪ Créer les profils utilisateurs ▪ Donner des droits d'accès.
Responsable Sélection Médical (RSM)	L'application doit permettre au RSM de : ▪ S'authentifier ▪ Affecter des numéros de don aux donneurs ▪ Consulter et examiner les donneurs ▪ Consulter son profil ▪ Créer/modifier/Supprimer les dossiers médicaux des donneurs
Assistant Médical	L'application doit permettre à l'assistant médical de : ▪ S'authentifier ▪ Consulter son profil

	▪ Créer/modifier/Supprimer les donneurs ▪ Gérer les planifications des collectes ▪ Créer les demandes des poches de sang
Préleveur	L'application doit permettre au préleveur de : ▪ S'authentifier ▪ Consulter les donneurs aptes au don ▪ Prélever les donneurs ▪ Gérer les prélèvements ▪ Consulter son profil
Laborantin	L'application doit permettre au laborantin de : ▪ S'authentifier ▪ Consulter les prélèvements effectués ▪ Gérer le stock des poches de sang ▪ Consulter son profil

Tableau 1 : Description des besoins fonctionnels

Chapitre IV : Conception

I. Diagramme des cas d'utilisation

Ils permettent de décrire l'interaction entre l'acteur et le système. L'idée forte est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système ! Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant le système. Les *use case* (cas d'utilisation) sont représentés par une ellipse sous-titrée par le nom du cas d'utilisation (éventuellement le nom est placé dans l'ellipse). Un acteur « représenté par un bonhomme dont le nom est indiqué en bas » et un cas d'utilisation sont mis en relation par une association représentée par une ligne. On distingue ainsi les inclusions ou « include » qui, représentés de A vers B, signifient que le cas d'utilisation A fait toujours appel au cas d'utilisation B pour s'exécuter ; les extensions ou « extend » qui, représentés de B vers A, signifient que le cas d'utilisation A fait appel au cas d'utilisation B sous certaines conditions.

Le plus souvent, le diagramme des cas est établi par la maîtrise d'ouvrage (MOA) d'un projet lors de la rédaction du cahier des charges afin de transmettre les besoins des utilisateurs et les fonctionnalités attendues associées à la maîtrise d'œuvre (MOE).

I.1. Identification des cas d'utilisation :

Un cas d'utilisation (« use case ») représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Pour constituer les cas d'utilisation, il faut considérer l'intention fonctionnelle de l'acteur par rapport au système dans le cadre de l'émission ou de la réception de chaque message. En regroupant les intentions fonctionnelles en unités cohérentes, on obtient les cas d'utilisations.

Cas d'utilisation	Acteurs	Messages émus/reçus
Gérer les donneurs	Assistant Médical	**Emet :** - Ajout, modification, suppression d'un donneur. **Reçoit :** - Confirmation
Consulter les donneurs	Responsable de la Sélection Médical / Assistant Médical	**Reçoit :** - Information sur les donneurs - Liste des donneurs

Gérer les dossiers Médicaux	Responsable de la Sélection Médical	**Emet :** - Enregistrement du dossier medical d'un donneur examiné, - Mises à jour des dossiers médicaux des donneurs **Reçoit :** - Confirmation, - Situation médicale des donneurs, - Liste des donneurs examinés
Prélever	Préleveur	**Emet :** - Mises à jour des dossiers médicaux des donneurs **Reçoit :** - Confirmation, - Situation médicale des donneurs, - Liste des donneurs apte au don
Analyser	Laborantin	**Emet :** - Envoie en stock d'une poche de sang - Rejet d'une poche de sang **Reçoit :** - Confirmation, - Situation médicale des donneurs, - Liste des donneurs prélevés
Planifier les collectes	Assistant Médical	**Emet :** - Ajout, modification, suppression d'une planification de collecte **Reçoit :** - Confirmation - Liste des planifications fixes - Liste des planifications mobiles

Gérer les poches du sang	Laborantin	**Emet :** - Enregistrement d'une poche de Sang - Retrait d'une poche de Sang **Reçoit :** - Confirmation
Gérer les profils	Administrateur	**Emet :** - Ajout, modification, suppression d'un utilisateur. **Reçoit :** - Confirmation
Gérer les rôles	Administrateur	**Emet :** - Ajout, modification, suppression d'un rôle. **Reçoit :** - Confirmation
S'authentifier	Tous les utilisateurs	Emet: - Nom d'utilisateur / Mot de pass Reçoit: - Confirmation

Tableau 2 : Description détaillée des cas d'utilisation

II. Diagramme de package

Les besoins variés des acteurs et le nombre de fonctionnalités dont le futur logiciel devra disposer nous semble assez souvent compliqués. Pour mieux s'y prendre et nous faciliter la tâche, on peut découper le futur logiciel en parties distinctes, en fonction des « familles » de fonctionnalités de façon à pouvoir les analyser séparément. Chacune de ces parties correspond à un domaine fonctionnel ou package.

II.1. Structuration des cas d'utilisations en packages :

Pour définir la stratégie de regroupement des cas d'utilisation pour un projet, trois critères de regroupements sont souvent utilisés :

- Par domaine d'expertise métier : le plus intuitif et souvent le plus efficace. Le but étant de trouver des parties qui regroupent des fonctionnalités qui ont un lien ou qui font partie d'une même « famille » et que l'on peut analyser séparément ;
- Par acteur : simple à mettre en œuvre uniquement si chaque cas d'utilisation est relié à un et un seul acteur, sinon il s'apparente souvent au critère précédent ;
- Par lot de livraison : dans le cadre d'un développement itératif et incrémental, il est intéressant de regrouper dans un même package les cas d'utilisations qui seront livrés ensemble au client.

Pour mieux s'organiser le regroupement par domaine d'expertise métier sera utilisé pour notre projet.

Cas d'utilisation	Acteurs	Package
Gérer les donneurs	Assistant Médical	**Gestion des donneurs**
Consulter les donneurs	Assistant Médical/RSM	
Gérer les dossiers médicaux	RSM	**Gestion des dossiers médicales**
Prélever	Préleveur	
Analyser	Laborantin	
Planifier les collectes	Assistant Médical	**Gestion des planifications des collectes**
Gérer les poches de sang	Laborantin	**Gestion de stock des poches de sang**

Gérer les profils	Administrateur	**Gestion des droits d'accès**
S'authentifier	Tous les acteurs	**Authentification**

Tableau 3 : Liste des cas d'utilisation par package

➢ Diagramme de package :

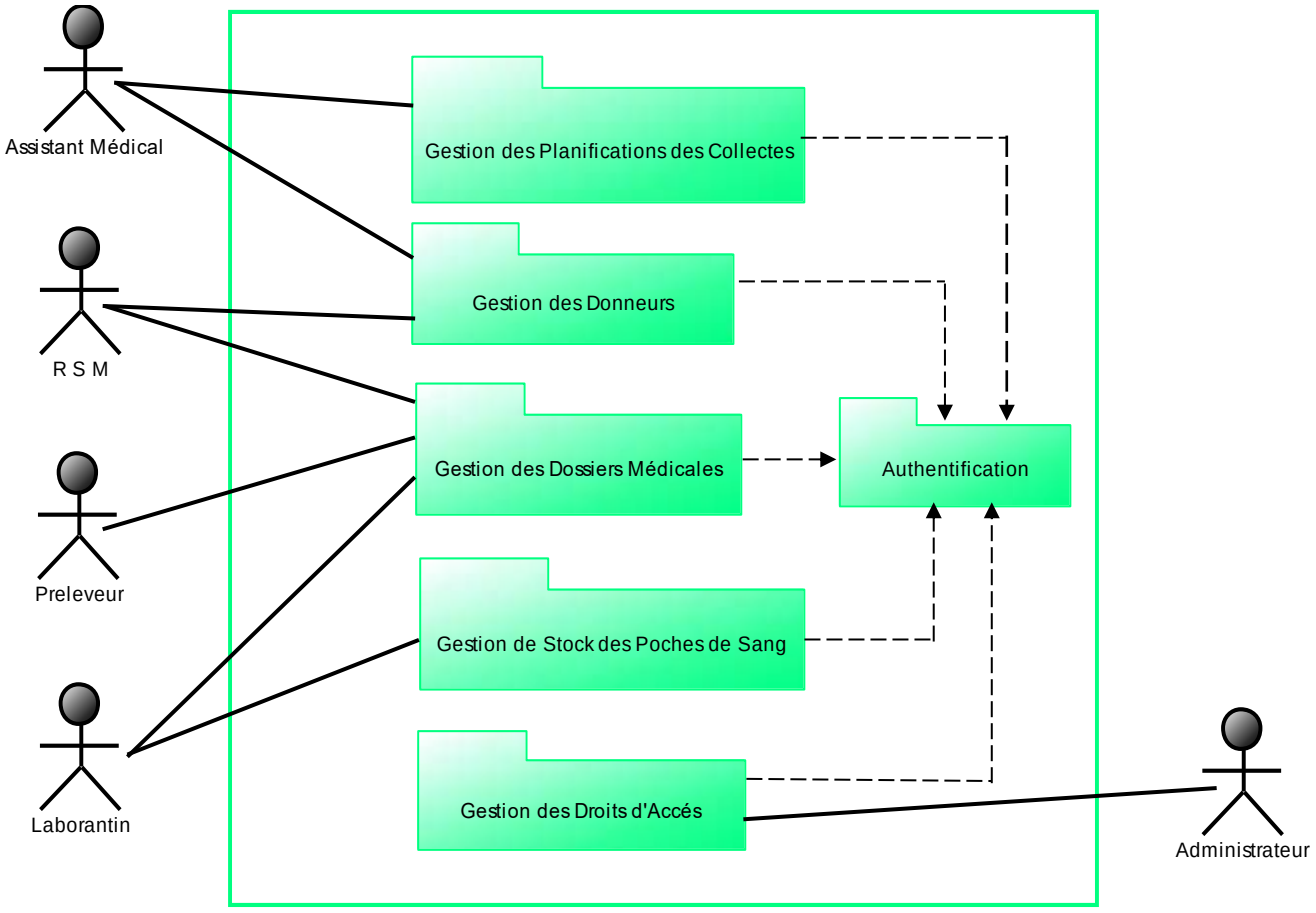


Figure 9 : Diagramme de package

➢ Diagramme de cas d'utilisation, package « gestion des donneurs » :

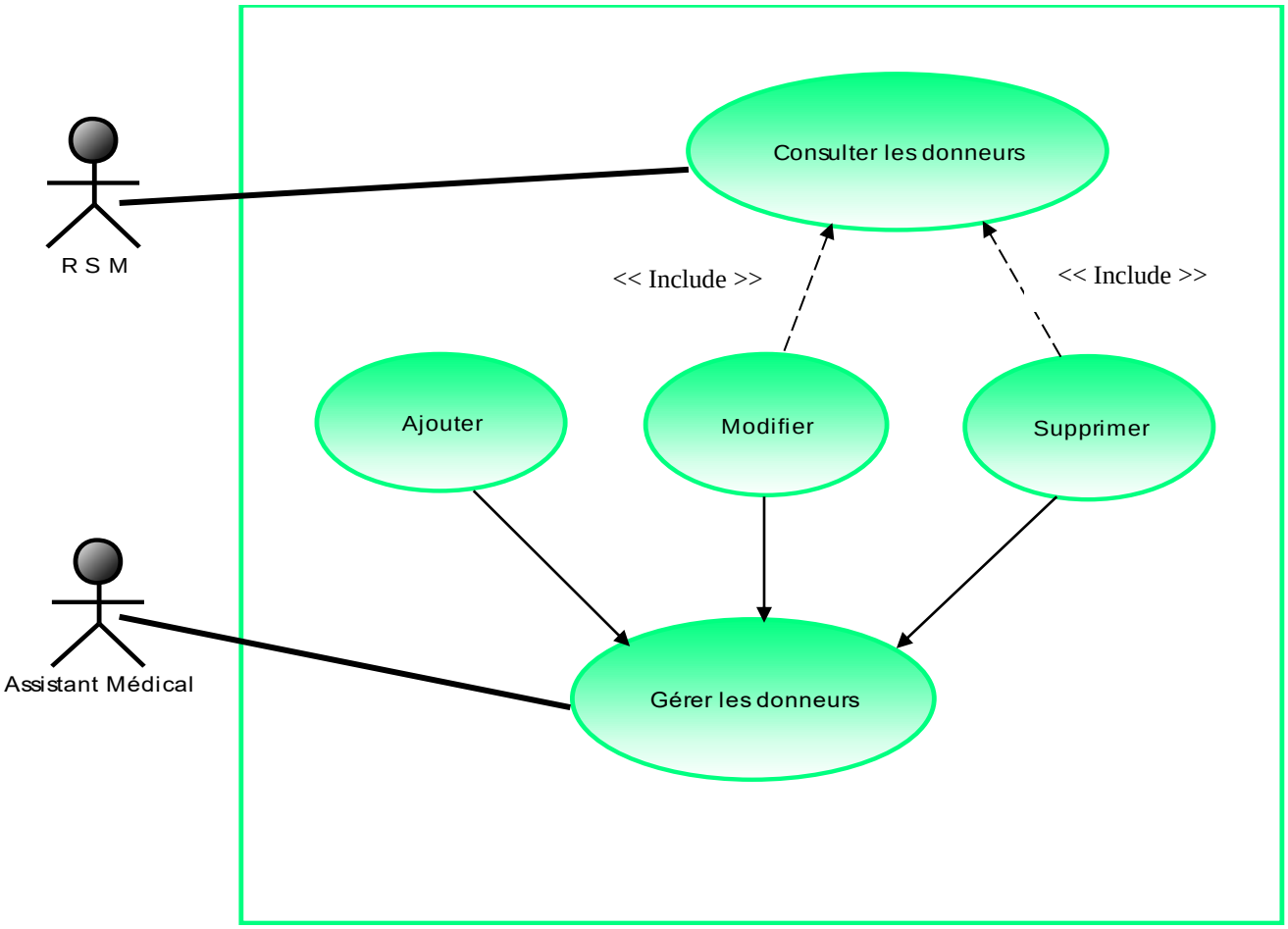


Figure 10 : Diagramme de cas d'utilisation, package « gestion des donneurs »

➢ Diagramme de cas d'utilisation, package « gestion des dossiers médicales » :

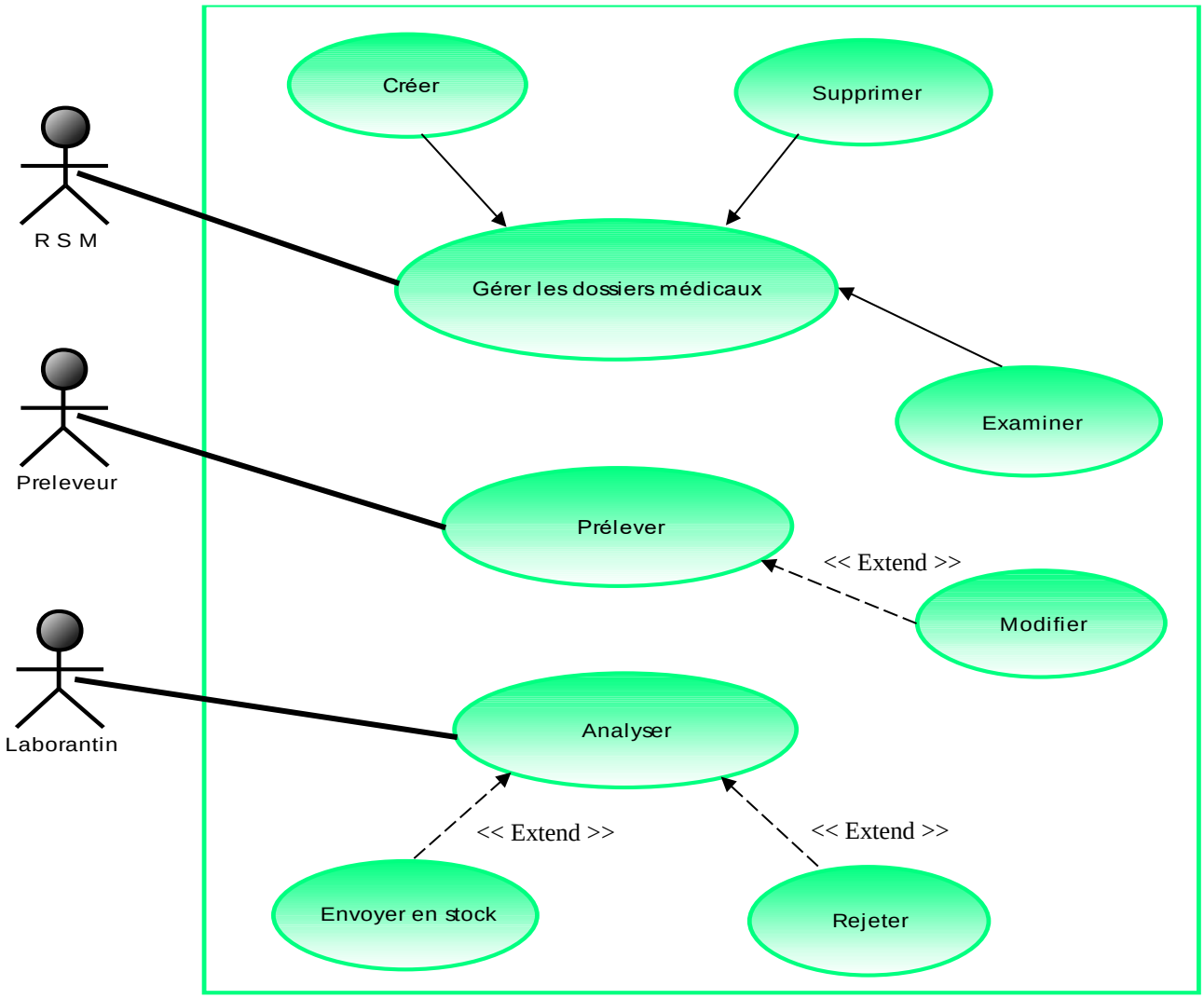


Figure 11 : Diagramme de cas d'utilisation, package « gestion des dossiers médicales »

➢ Diagramme de cas d'utilisation, package « gestion des planifications des collectes » :

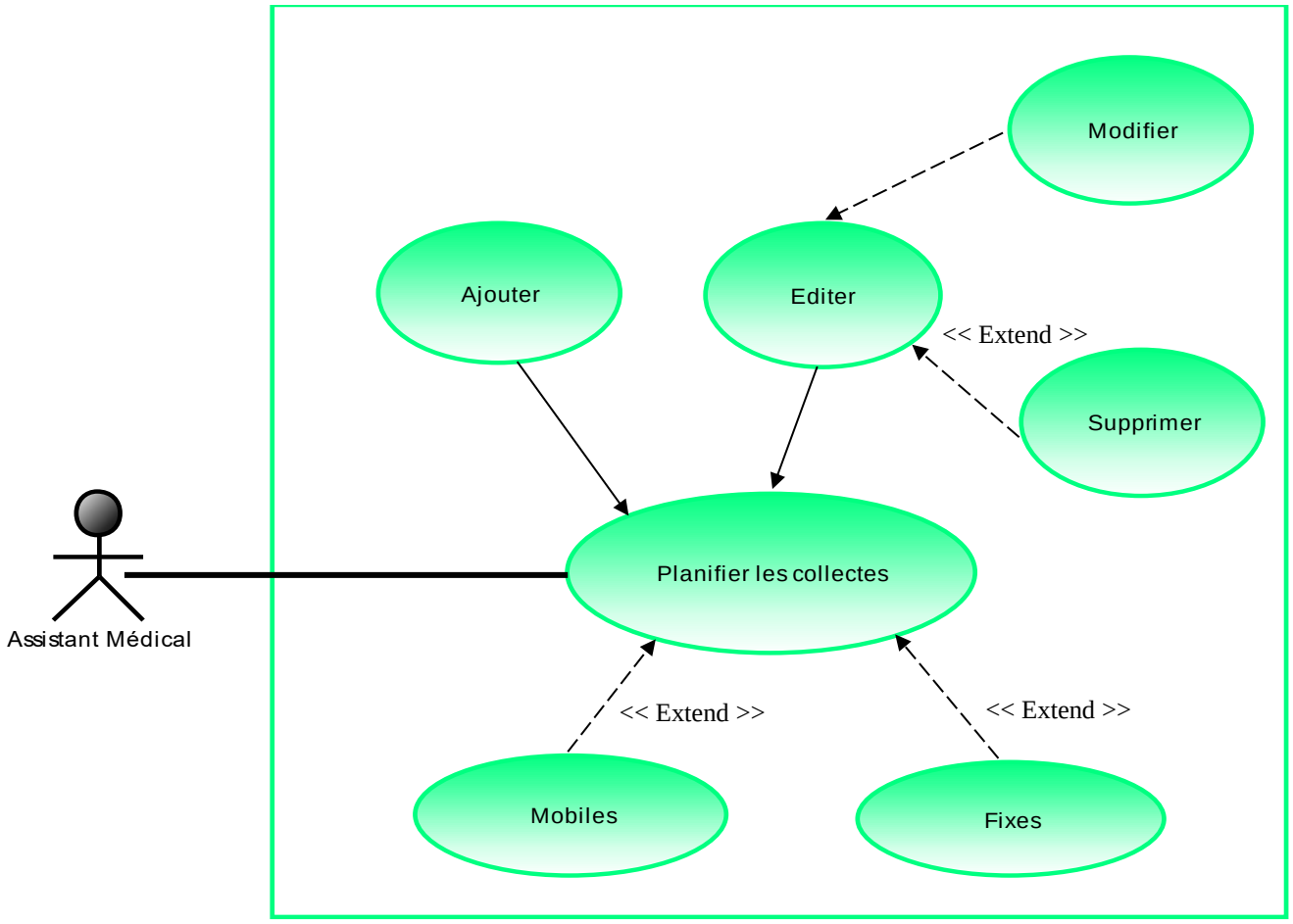


Figure 12 : Diagramme de cas d'utilisation, package « gestion des planifications des collectes »

- **Diagramme de cas d'utilisation, package « gestion de stock des poches de sang » :**

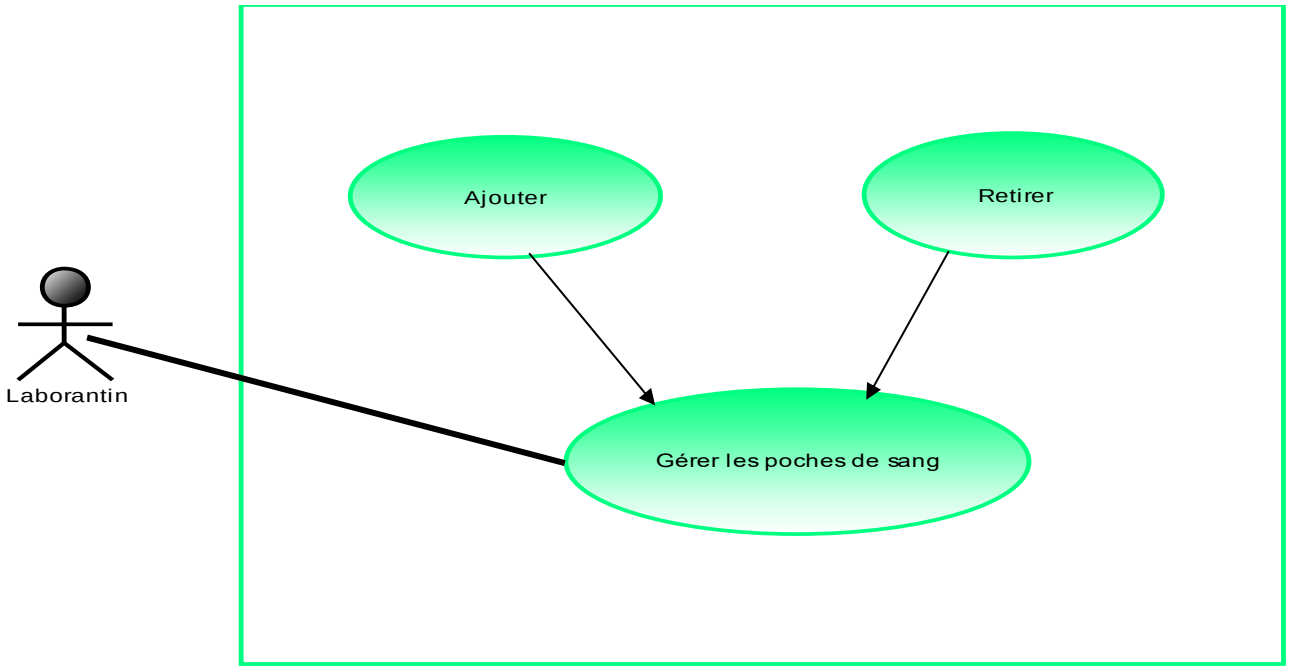


Figure 13 : Diagramme de cas d'utilisation, package « gestion de stock des poches de sang »

➢ Diagramme de cas d'utilisation, package « Gestion des droits d'accès » :

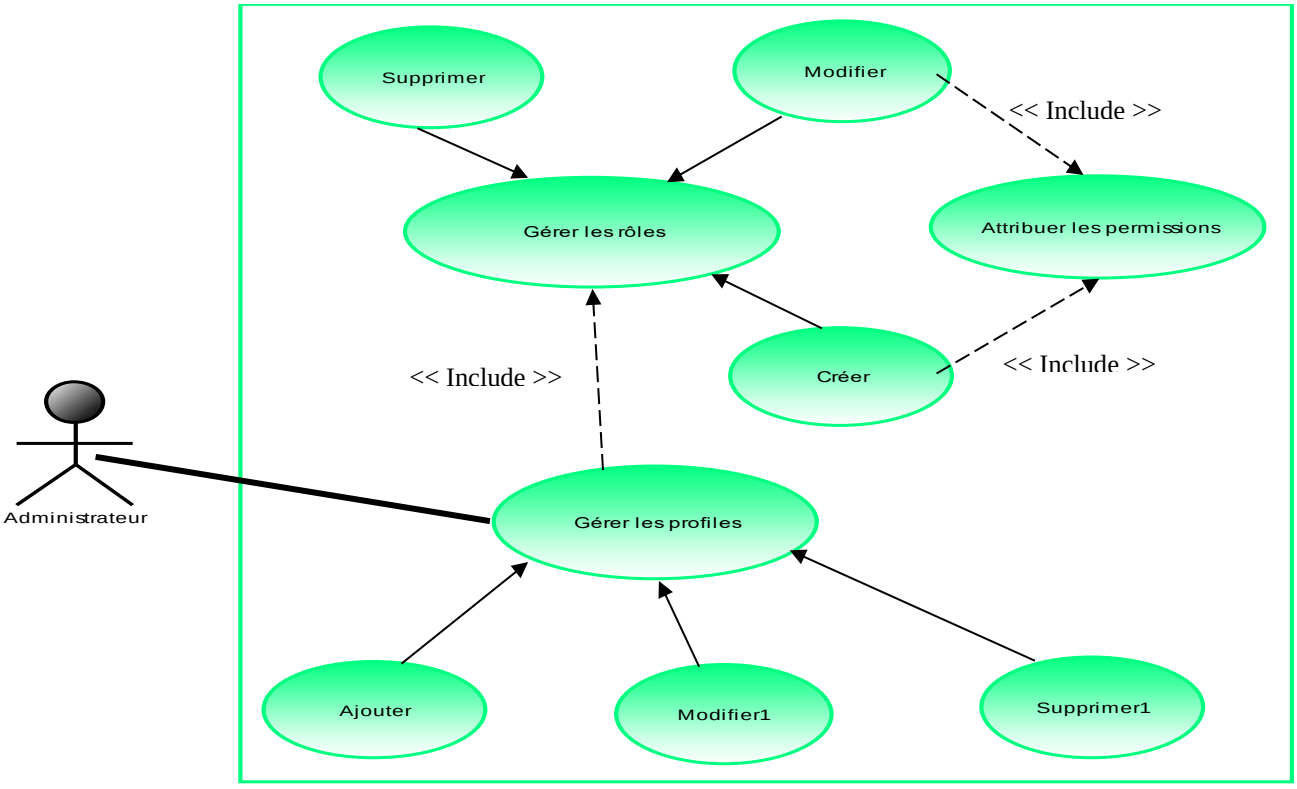


Figure 14 : Diagramme de cas d'utilisation, package « gestion des droits d'accès »

II.2. Description détaillée des cas d'utilisations :

Les diagrammes réalisés jusqu'à maintenant (diagramme de contexte, diagramme de package, diagramme de cas d'utilisation) nous ont permis de découvrir petit à petit les fonctionnalités (appelées aussi des cas d'utilisation) que l'on devrait avoir dans le futur logiciel.

Nous allons maintenant détailler les cas d'utilisations qui doivent faire l'objet d'une définition a priori qui décrit l'intention de l'acteur lorsqu'il utilise le système et les séquences d'actions principales qu'il est susceptible d'effectuer. Ces définitions servent à fixer les idées et n'ont pas pour but de spécifier un fonctionnement complet et irréversible.

Pour exprimer les cas d'utilisations de notre système, nous avons choisi le formalisme suivant :

Titre	**Le titre du cas d'utilisation**
Acteur	Acteurs participants au cas d'utilisation
Description	But du cas d'utilisation
Description des enchainements	
Préconditions	Condition qui doit être remplie avant le début du cas d'utilisation
Scenario nominal Séquence d'actions normales associées au cas d'utilisation	
Scenario alternatif Séquence d'actions alternatives. C'est le cas des étapes liées à des conditions.	

Tableau 4 : Formalisme de description d'un cas d'utilisation

- **Cas d'utilisation : Gérer les donneurs, package « Gestion des donneurs » :**

Titre	**Gérer les donneurs**
Acteur	Assistant Médical
Description	Le système doit permettre de gérer les donneurs à savoir l'ajout, la modification, la recherche et la suppression des donneurs
Description des enchainements	
Préconditions	Le donneur doit être authentifié (cas d'utilisation : s'authentifié – package : authentification) et avoir la permission d'effectuer l'opération
Scenario nominal 1. S'il choisit l'option « Gestion des donneurs » du menu général et choisit l'élément « donneurs » 2. Le système affiche une page contenant la liste des donneurs, accompagnée des options « ajouter », « modifier », « supprimer » **3. S'il choisit l'option d'ajouter**	

4. Le système affiche une page contenant un formulaire pour saisir les informations du donneur
5. L'assistant médical remplit les champs du formulaire et valide l'opération d'ajout
6. Le système ajoute les données dans la base et affiche un message de validation.
7. **S'il choisit de supprimer un donneur**
8. Le système effectue l'opération, et affiche une notification, la suppression a été effectuer avec succès.
9. **S'il choisit de modifier un donneur**
10. Le système affiche un formulaire contenant les informations du donneur
11. L'utilisateur apporte les modifications et valide l'opération
12. Le système enregistre les changements effectués et affiche un message de validation

Scenario alternatif

2.a En cas d'erreur, le système notifie l'assistant médical que la connexion a échoué ou une erreur est survenue sur la base de données lors de la sélection des données

4.a L'assistant médical décide de quitter l'ajout. Le scenario reprend au point 2 du scenario nominal.

5.a Si les champs requis ne sont pas remplis ou invalide, le système affiche une notification, veuillez bien renseigner tous les champs du formulaire. Le scenario reprend au point 4 du scenario nominal.

5.b En cas d'erreur, le système notifie l'assistant médical que la connexion a échoué ou une erreur est survenue sur la base de données lors de l'ajout des données. Le scenario reprend au point 4 du scenario nominal.

8.a Si le donneur a un dossier médical, le système affiche une notification, vous ne pouvez pas supprimer un donneur qui a un dossier médical. Le scénario reprend au point 2 du scenario nominal.

10.a L'assistant médical décide de quitter la modification. Le scenario reprend au point 2 du scenario nominal.

11.a Si les champs requis ne sont pas remplis ou invalide, le système affiche une notification, veuillez bien renseigner tous les champs du formulaire. Le scenario reprend au point 10 du scenario nominal.

11.b En cas d'erreur, le système notifie l'assistant médical que la connexion a échoué ou une erreur est survenue sur la base de données lors de la modification des données. Le scenario reprend au point 10 du scenario nominal.

Tableau 5 : Description détaillée, cas d'utilisation gérer les donneurs, package « Gestion des donneurs »

- **Cas d'utilisation : Planifier les collectes, package « Gestion des planifications des collectes » :**

Titre	**Planifier les collectes**
Acteur	Assistant Médical
Description	Le système doit permettre de planifier les collectes à savoir l'ajout, la modification, la recherche et la suppression des planifications
Description des enchainements	
Préconditions	Le donneur doit être authentifié (cas d'utilisation : s'authentifié – package : authentification) et avoir la permission d'effectuer l'opération
Scenario nominal 1. L'assistant médical choisit l'option « Planification des collectes » du menu général et choisit l'élément « Collectes Fixes » ou « Collectes Mobiles » 2. Le système affiche une page contenant la liste des planifications fixes ou mobiles, accompagnée des opérations « ajouter », « modifier », « supprimer » **3. S'il choisit l'option d'ajouter** 4. Le système affiche une page contenant un formulaire pour saisir les informations du la planification 5. L'assistant médical remplit les champs du formulaire et valide l'opération d'ajout 6. Le système ajoute les données dans la base et affiche un message de validation. **7. S'il choisit de supprimer une planification** 8. Le système effectue l'opération, et affiche une notification, la suppression a été effectuer avec succès. **9. S'il choisit de modifier une planification** 10. Le système affiche un formulaire contenant les informations de la planification 11. L'utilisateur apporte les modifications et valide l'opération 12. Le système enregistre les changements effectués et affiche un message de validation	
Scenario alternatif	

3.a En cas d'erreur, le système notifie l'assistant médical que la connexion a échoué ou une erreur est survenue sur la base de données lors de la sélection des données 6.a L'assistant médical décide de quitter l'ajout. Le scenario reprend au point 2 du scenario nominal. 7.a Si les champs requis ne sont pas remplis ou invalide, le système affiche une notification, veuillez bien renseigner tous les champs du formulaire. Le scenario reprend au point 4 du scenario nominal. 6.b En cas d'erreur, le système notifie l'assistant médical que la connexion a échoué ou une erreur est survenue sur la base de données lors de l'ajout des données. Le scenario reprend au point 4 du scenario nominal. 12.a L'assistant médical décide de quitter la modification. Le scenario reprend au point 2 du scenario nominal. 13.a Si les champs requis ne sont pas remplis ou invalide, le système affiche une notification, veuillez bien renseigner tous les champs du formulaire. Le scenario reprend au point 10 du scenario nominal. 12.b En cas d'erreur, le système notifie l'assistant médical que la connexion a échoué ou une erreur est survenue sur la base de données lors de la modification des données. Le scenario reprend au point 10 du scenario nominal.

Tableau 6 : Description détaillée, cas d'utilisation planifier les collectes, package « Gestion des planifications des collectes »

II. Diagramme de séquence

Les diagrammes de séquences sont la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation Unified Modeling Language (UML). On montre ces interactions dans le cadre d'un scénario du diagramme des cas d'utilisation. Dans le souci de simplification, on représente l'acteur principal à gauche du diagramme et les acteurs secondaires éventuels à droite du système. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets. Les périodes d'activité des classes sont symbolisées par des rectangles. Plusieurs types de messages (actions) peuvent transiter entre les acteurs et objets :

- ✓ **Message simple :** le message n'a pas de spécificité particulière d'envoi et de réception.
- ✓ **Message avec durée de vie** : l'expéditeur attend une réponse du récepteur pendant un certain temps et reprend ses activités si aucune réponse n'a lieu dans un délai prévu.
- ✓ **Message synchrone :** l'expéditeur est bloqué jusqu'au signal de prise en compte par le destinataire.
- ✓ **Message asynchrone :** le message est envoyé, l'expéditeur continue son activité que le message soit parvenu, pris en compte ou non. Les messages asynchrones sont symbolisés par des demi-flèches.
- ✓ **Message dérobant :** le message est mis en attente dans une liste d'attente de traitement chez le récepteur.

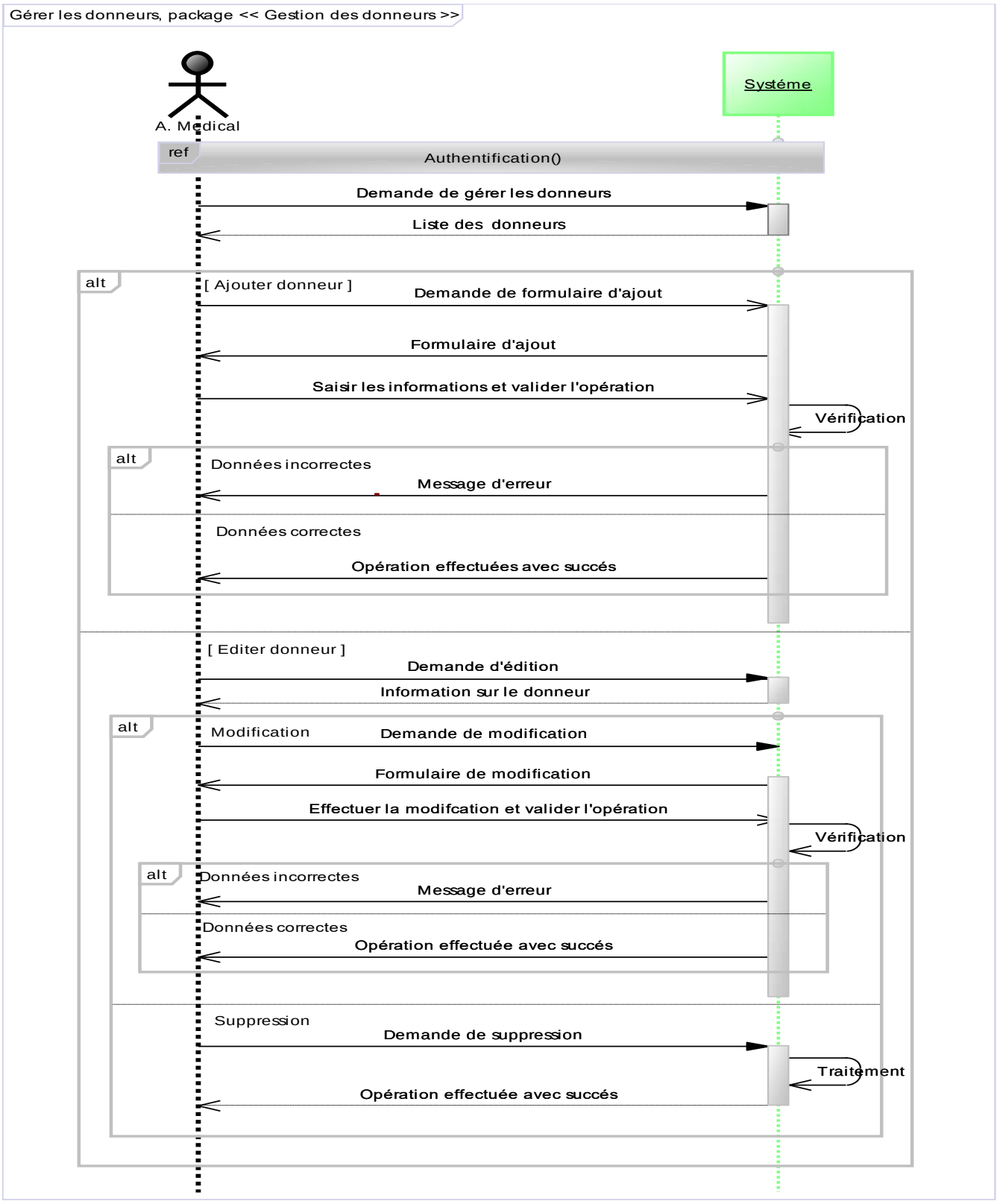


Figure 15 : Diagramme de séquence gérer les donneurs, package « Gestion des donneurs »

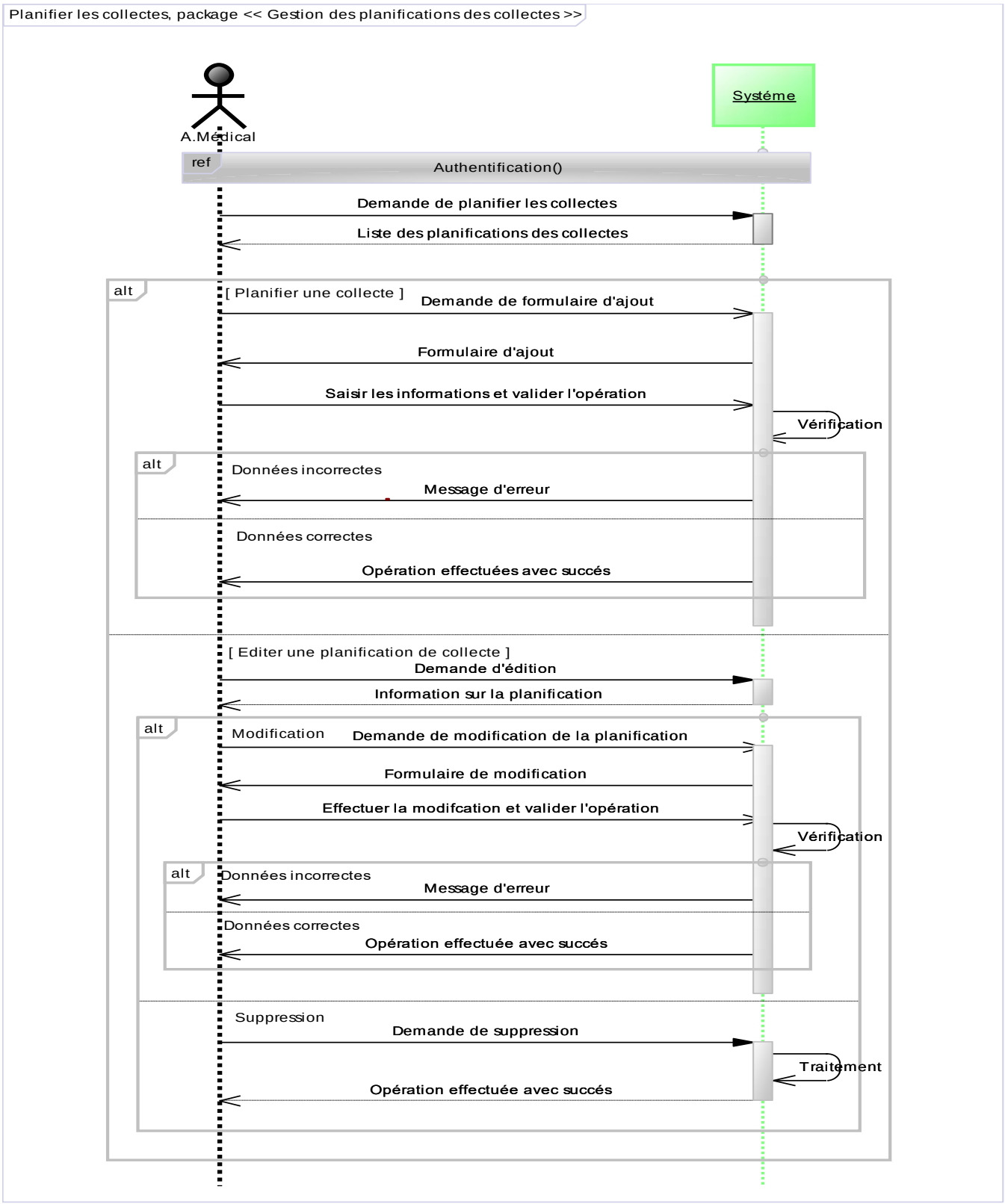


Figure 16 : Diagramme de séquence planifier les collectes, package « Gestion des planifications des collectes »

III. Diagramme d'activité

Les diagrammes d'activités décrit le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités. Une activité désigne une suite d'action.

Le passage d'une action à une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une action et provoquent le début immédiat d'une autre (elles sont automatiques).

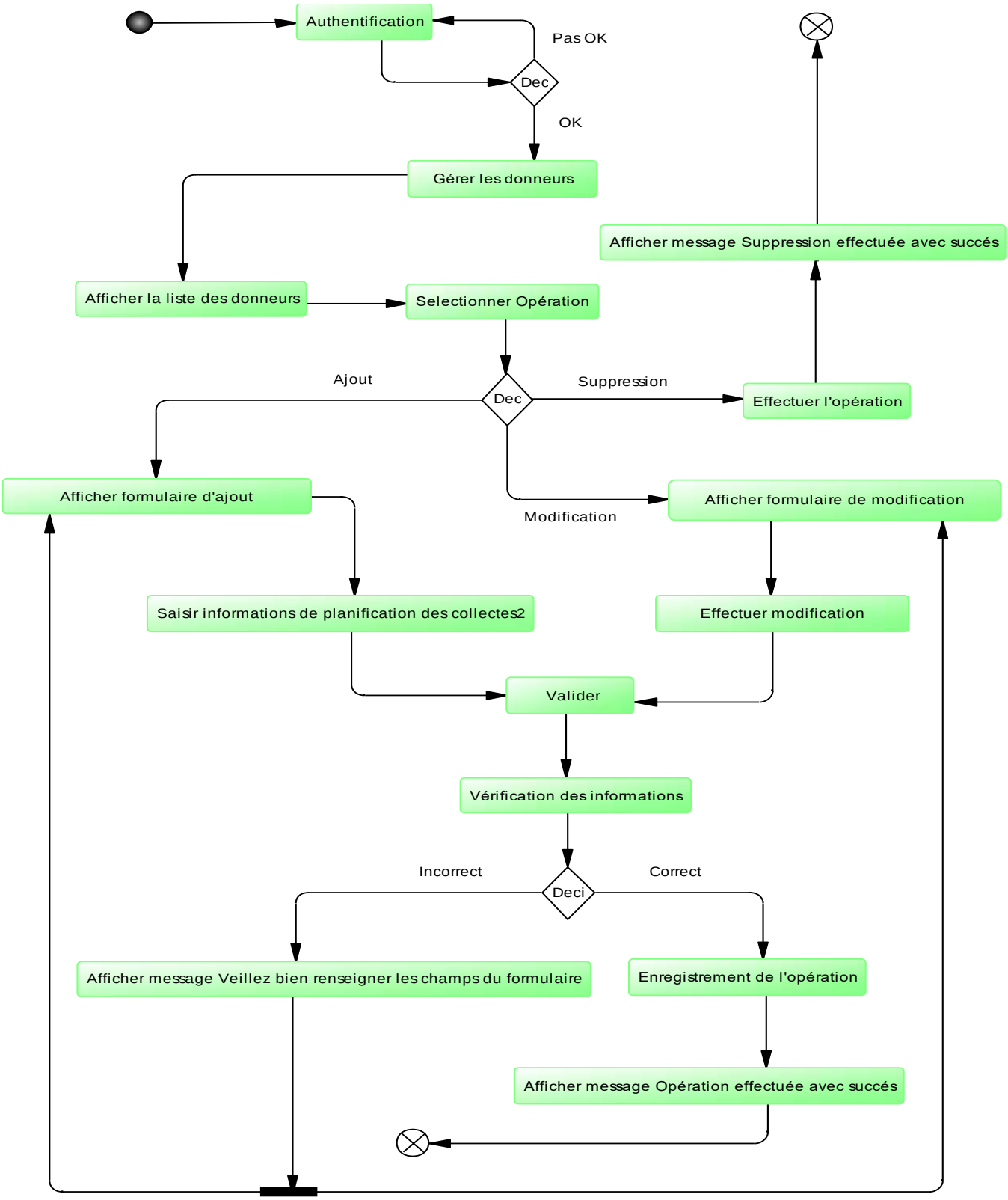


Figure 17 : Diagramme d'activité gérer les donneurs, package « Gestion des donneurs »

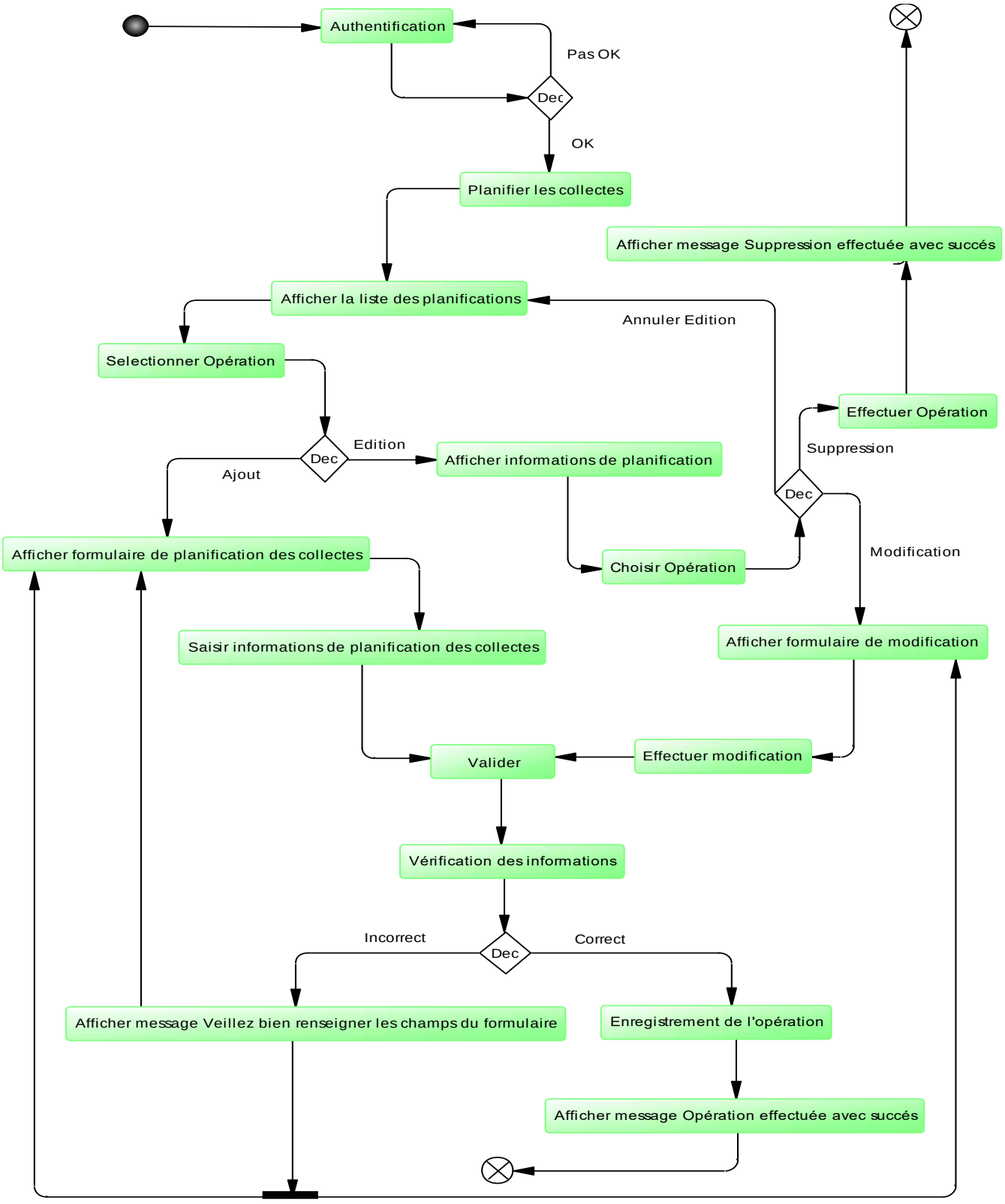


Figure 18 : Diagramme d'activité planifier les collectes, package « Gestion de planification des collectes »

IV. Diagramme de classe

Le diagramme de classes exprime la structure statique du système en termes des classes et de relations entre ces classes. L'intérêt du diagramme de classe est de modéliser les entités du système d'information. Le diagramme de classe permet de représenter l'ensemble des informations finalisées qui sont gérées par le domaine. Le diagramme met en évidence d'éventuelles relations entre ces classes. Le diagramme de classes comporte 5 concepts :

- La notion de classe : Elle définit leur structure, leur comportement et leurs relations. Les classes sont représentées par des rectangles compartimentés en trois (3) parties. Une première partie contenant le nom de la classe, une deuxième représentant les attributs de la classe et enfin la troisième partie représentant les opérations de la classe ;
- La notion d'attribut : C'est un ensemble d'informations élémentaires d'une classe. Un attribut représente la modélisation d'une information élémentaire représentée par son nom et son format. UML définit 3 niveaux de visibilité pour les attributs : les attributs publics notés (+) ; les attributs privés notés (-) et les attributs protégés notés (#).
- La notion d'identifiant : c'est un attribut particulier qui permet de repérer de façon unique chaque objet, instance de la classe. ;
- La notion d'opération : l'opération représente un élément de comportement des objets, défini de manière globale dans la classe. Une opération est une fonctionnalité assurée par une classe. La description des opérations peut préciser les paramètres d'entrée et de sortie ainsi que les actions élémentaires à exécuter. Tout comme les attributs, les opérations ont des niveaux de visibilité : la visibilité des opérations publiques est notée (+) ; la visibilité des opérations privées est notée (-) et enfin (#) pour la visibilité des opérations protégées ;
- La notion de relation : S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives. Les liens entre les objets doivent être considérer comme des instances de relation entre classes.

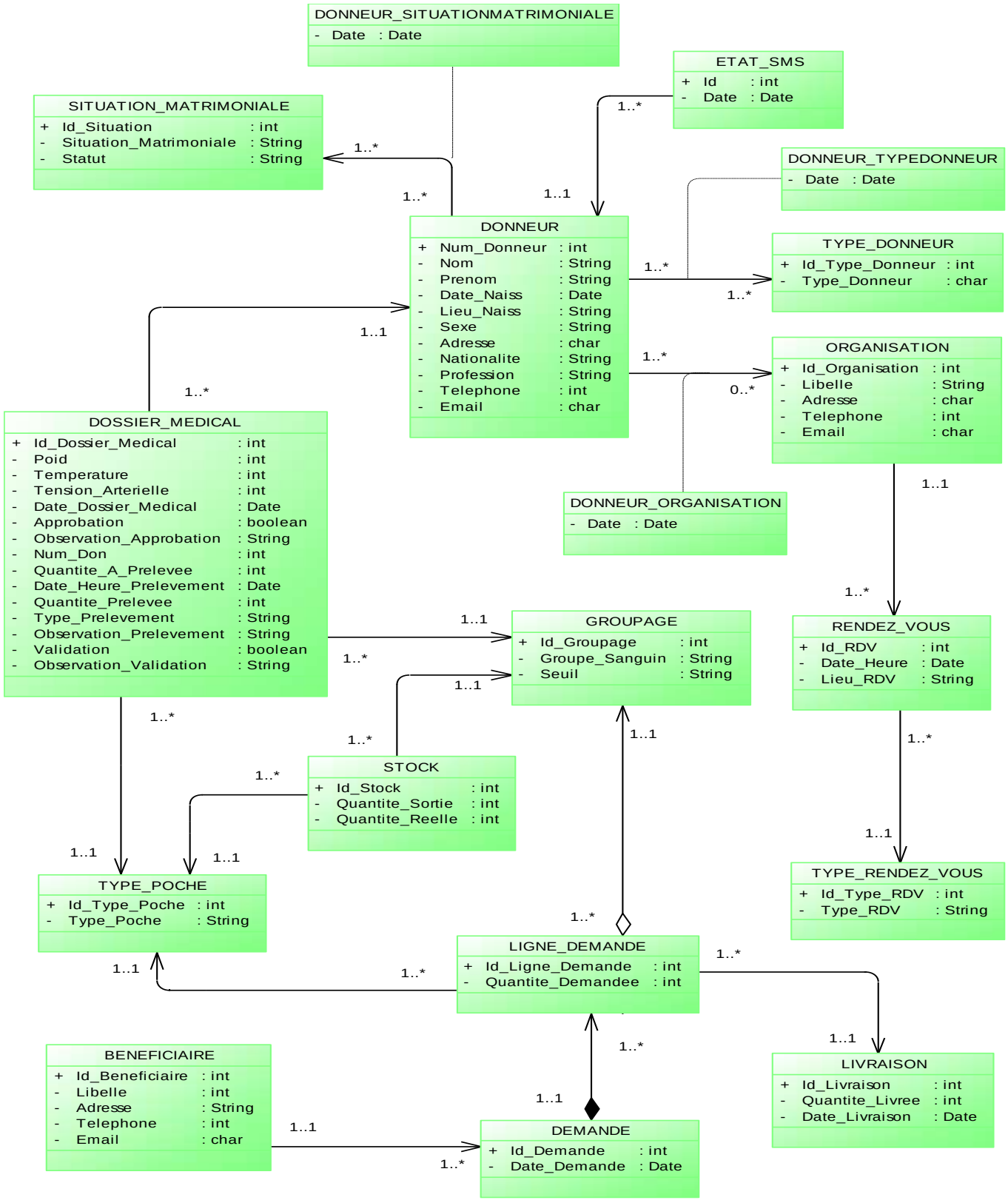


Figure 19 : Diagramme de classe global

V. Diagramme de déploiement

Il indique l'organisation matérielle de l'application à concevoir et spécifie-les Composants physiques nécessaires pour l'application.

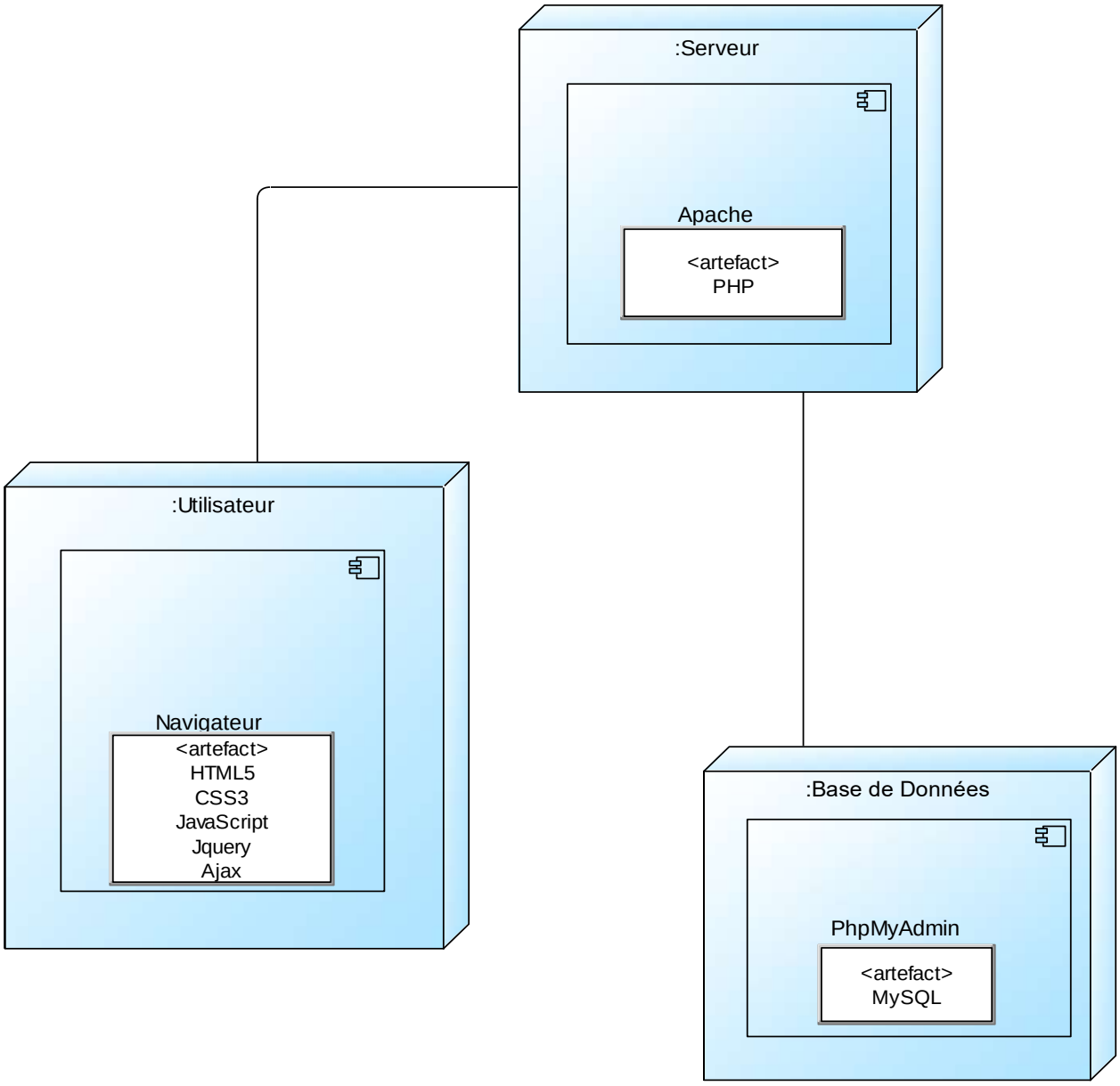


Figure 19 : Diagramme de déploiement

Partie III : Réalisation

Chapitre V : Architecture et description des outils et technologies utilisés

V.1. Architecture interne de l'application :

Une architecture est un modèle générique et conceptuel qui se rapporte à un sujet et qui représente la fonctionnalité, la structure, le positionnement, l'interrelation des différents types d'élément qui la composent.

En règle générale, une application est découpée 3 couches d'abstraction :

- La couche présentation ; c'est la partie de l'application visible par les utilisateurs (nous parlons de l'interface utilisateur). Dans notre cas cette couche est un navigateur web, qui se présente sous forme de page HTML, composé de formulaire et de bouton ;
- La couche métier : correspond à la partie fonctionnelle de l'application, celle qui implémente la logique, et qui décrit les opérations que l'application opère sur les données en fonction des requêtes d'un utilisateur effectuées à travers la couche présentation :
- La couche accès aux données : c'est la partie gérant l'accès à la base des données du système.

Il existe différentes architectures dont entre autres ;

- ✓ L'architecture 1_tiers : La conception de l'application est élaborée de manière à fonctionner sur un ordinateur unique. En fait, tous les services fournis par l'application résident sur la même machine et sont inclus dans l'application. Toutes les fonctionnalités sont donc comprises dans une seule couche logicielle.
- ✓ L'architecture 2_tiers : Appelée aussi architecture client lourd/serveur, elle est assez simple dans sa mise en œuvre. Ce type d'architecture est constitué uniquement de deux parties : le « client lourd » demandeur de service d'une part et le « serveur de données » qui fournit le service d'autre part. Nous aurons donc la base de données qui sera délocalisée sur un serveur dédié appelé le serveur de données qui fournira les données à exploiter.
- ✓ L'architecture 3_tiers : Cette architecture physique est assez semblable à l'architecture client/serveur, mais en plus des « clients » et du serveur de données évoquées plus haut, un serveur d'application intervient comme un troisième tiers. En effet, les machines clientes également appelées « clients légers » ne contiennent que l'interface de l'application de manière qu'elles soient déchargées de tout traitement. En effet, le traitement est ainsi assuré par le serveur d'application, qui sert de liaison entre l'interface applicative et les données localisées au niveau du serveur de données.

V.2. Choix de l'architecture :

Pour nous, utiliser une architecture à 3 niveaux est beaucoup plus propre. Cela permet de diviser les tâches et par conséquent d'avoir des développeurs spécialisés sur un des trois niveaux. Cependant dans notre application nous avons utilisé l'architecture 3_tiers qui est illustrée dans la figure qui suit :

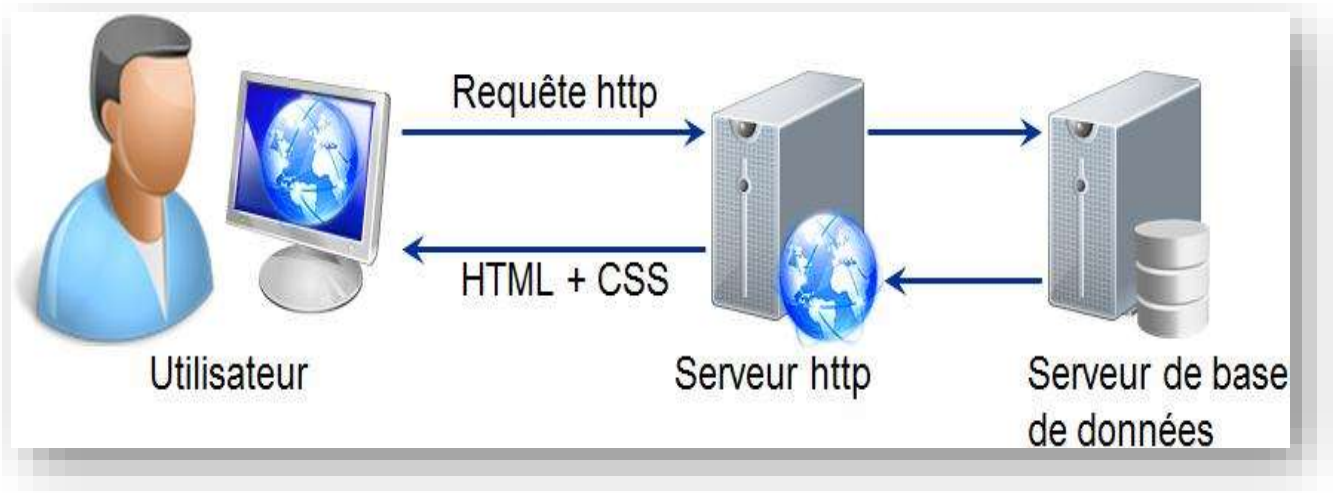


Figure 20 : Architecture 3 tiers

V.3. Outils utilisés :

- PowerAMC :

PowerAMC fait partie des AGL (Atelier de Génie Logiciel), programme qui assiste le développeur au moment de la conception de certaines tâches, les AGL sont très pratique compte tenu de leur souplesse, leur efficacité et permettent aux développeurs de gagner plus de temps dans la réalisation de certain nombre de travaux ;

StarUML :

C'est un logiciel de modélisation UML open source sous une licence modifiée de GNU GPL. L'objectif de ce projet est de se substituer à des solutions commerciales comme IBM Rational Rose ou Borland Together. StartUml gère la plupart des diagrammes spécifiés dans la norme Uml 2.0 ;

V.4. Outils de développements

- **Apache**

Le logiciel **Apache** est un **serveur** http en Open Source Utilisé principalement sur les hébergements Internet en Linux, bien qu'il soit également utilisable en Windows (concurrent d'Internet et Information Service-IIS), Unix ou OSX. C'est actuellement le plus utilisé sur le WEB ;

- **MySql**

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui de mises à jour fréquentes et fortement sécurisées. La première version de MySQL est apparue le 23 mai 1995.

- **Lampp**

LAMPP est un acronyme informatique permettant de désigner une architecture logicielle basée sur les logiciels libres suivant :

- L « Linux », le système d'exploitation ;
- A « Apache », le serveur web ;
- M « MySQL », le serveur de base de données ; (qui permet de stocker et d'organiser les données).
- P « PHP » le langage de script/moteur d'exécution de code (permet la génération de pages dynamiques et la communication avec le serveur MySQL).

« L'environnement LAMP est un véritable atout pour développer nos applications, simplement parce qu'il est **économique, robuste et facilement déployable**. De plus, c'est un ensemble de logiciels libres (en Open Source) supporté par une communauté extrêmement large de programmeurs ».

V.4. Outils de conceptions

- **Google Chrome**

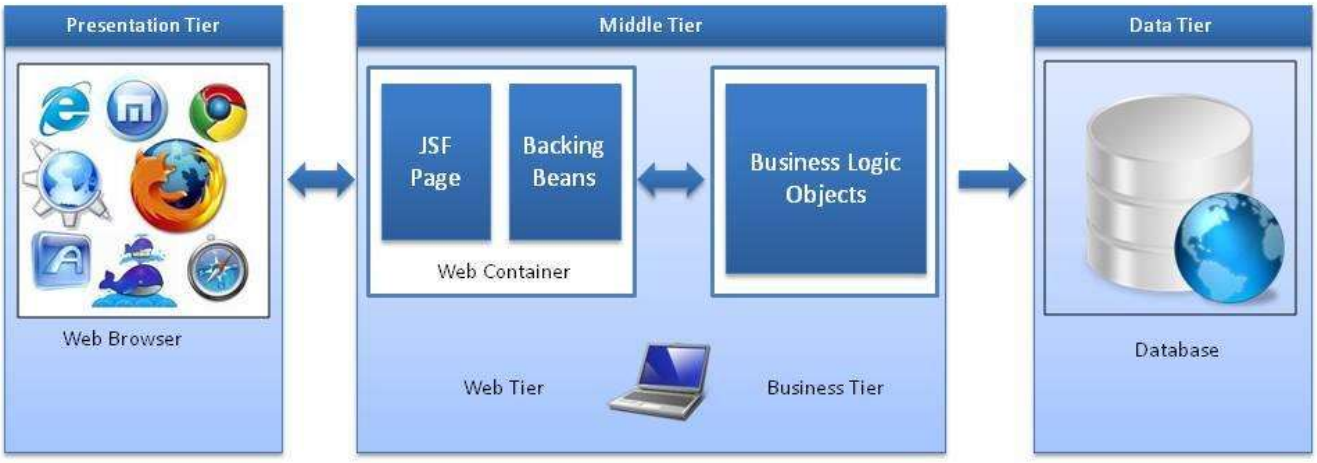
Google Chrome est un navigateur web créé par Google et basé sur le projet open source Chromium. L'objectif initial de Google en développant Chrome était de fournir aux internautes un nouveau navigateur plus rapide et proposant plusieurs innovations par rapport à Mozilla Firefox ou à Microsoft Internet Explorer. En plus des questions de performances, Google Chrome se différencie notamment de sa concurrence en dédiant un espace mémoire à l'utilisateur et un processus unique pour chaque fenêtre ou nouvel onglet, afin d'assurer une meilleure stabilité et sécurité.

- **Mozilla Firefox**

Mozilla Firefox est un navigateur Web libre et gratuit, développé et distribué par la Mozilla Foundation avec l'aide de centaines de bénévoles grâce aux méthodes de développement du logiciel libre/open source et à la liberté du code source. Firefox est à l'origine un programme dérivé du logiciel Mozilla (aujourd'hui connu sous le nom de Sea Monkey), mais

reprenant uniquement les fonctions de navigation de ce dernier. Ce logiciel multiplateforme est compatible avec diverses versions de Windows, Mac OSX et GNU/Linux ;

- **MVC**



Le paradigne MVC est un schéma de programmation qui propose de sépare une application en trois parties :

- ✓ Le modèle : qui contient la logique et l'état de l'application ;
- ✓ La vue : qui représente l'interface utilisateur ;
- ✓ Le contrôleur qui gère la synchronisation entre la vue et le modèle.

- **Visual Code Studio**

Visual Studio Code est un éditeur de code extensible développé par Microsoft pour Windows, Linux et OS X. Visual Studio Code est un nouvel outil qui combine la simplicité d'un éditeur de code avec ce dont les développeurs ont besoin pour le cycle edit-build-debug. Aller au-delà de la coloration syntaxique et de la saisie semi-automatique avec IntelliSense, qui fournit des complétions intelligentes basées sur des types de variables, des définitions de fonctions et des modules importés. Code de débogage à partir de l'éditeur. Lancez ou attachez à vos applications en cours d'exécution et déboguez avec des points d'arrêt, des piles d'appels et une console interactive.

- Twilio

Twilio est une API de service Web de téléphonie vous permettant d'utiliser vos aptitudes et langages Web existants pour générer des applications vocales et de SMS. Twilio est un service tiers, et non une fonctionnalité.

V.5. Langage de programmation utilisé

- **PHP**

PHP (abréviation de Personal Home Page Hypertext Préprocesseur) à ce jour est un langage se scripts que s'intègre aux pages Html et qui permet de réaliser des pages dynamiques. La première version date 1994 et s'appelait PHP FI. Elle n'avait pour ambition que de pouvoir insérer quelques traitements simples dans une page Html, comptage des visites. PHP s'exécute coté serveur, c'est-à-dire que le serveur interprète, analyse et exécute le script PHP avant d'en envoyer le résultat au client (navigateur). De ce fait, le client n'interagit pas directement avec PHP ce qui renforce la sécurité de l'application. De plus PHP est un produit « Open Source » c'est-à-dire que le code est accessible à tout développeur. En plus, il propose un accès facile aux bases de données. Malgré son manque de puissance face à d'autre langage de programmation (Perl, C, Java, JEE, etc …).PHP n'en demeure pas moins un langage solide et accessible à tous.

- **HTML**

HTML (HyperText Markup Language) est un langage de balisage (un langage qui utilise des balises '<>') servant à l'écriture de page du Word Wide Web (WWW). Il permet :

- De structurer sémantiquement et de faire la mise en page du contenu des pages ;
- D'inclure des ressources multimédias dont des images ;
- De réaliser des formulaires de saisie, etc.

Il est souvent utilisé conjointement avec langages de programmation comme PHP, JavaScript et les CSS. HTML5 est l'évolution la plus récente d'HTML. Cette évolution consiste en une multitude des nouvelles fonctionnalités qui ont été apportées au langage HTML ainsi qu'au JavaScript. Si vous savez faire de l'HTML « classique » vous devriez donc apprendre à manipuler ces nouvelles fonctionnalités.

- **CSS**

CSS (Cascading Style Sheets : feuille de style en cascade) est un langage informatique qui sert à décrire la présentation des pages Web CSS s'applique à une ou plusieurs pages du site. Le terme « en cascade » indique que la mise en forme d'une page peut faire appel) plusieurs feuilles de style. La nouvelle version de CSS est le CSS3 qui a apporté au langage de nombreuses nouveautés. Il s'agit par exemple d'un ensemble de nouveaux effets à appliquer sur nos éléments HTML, d'un ensemble d'un ensemble de nouveaux sélecteurs, de nouvelles manières de spécifier les couleurs, une détection des caractéristiques de l'appareil de l'utilisateur, des calculs dans les feuilles de style, des SVG en arrière-plan... Bref, le CSS3 est un généreux enrichissement des feuilles de style qui, en plus d'être profitable à l'utilisateur, l'est également pour le développeur.

- **JQuery**

JQuery est une bibliothèque (c'est-à-dire un ensemble de codes prêts à l'emploi) conçue pour simplifier l'écriture de codes JavaScript et AJAX. Créée en 2006 par John Resig, cette bibliothèque est la plus célèbre et la plus utilisée à ce jour.

JQuery est une bibliothèque (c'est-à-dire un ensemble de codes prêts à l'emploi) conçue pour simplifier l'écriture de codes JavaScript et AJAX. Créée en 2006 par John Resig, cette bibliothèque est la plus célèbre et la plus utilisée à ce jour.

- **Ajax**

AJAX (acronyme d'Asynchronous JavaScript And XML.) est apparu en 1995. Son utilisation est très intéressante, car elle permet de mettre à jour une partie (et une partie seulement) d'une page Web en demandant les données nécessaires à un serveur. Les échanges client/serveur sont donc limités et les pages Web sont affichées plus rapidement, pour le plus grand plaisir des utilisateurs.

- **JavaScript**

Le JavaScript est un langage de script incorporé dans un document HTML. Historiquement il s'agit même du premier langage de script pour le Web. Ce langage est un lange de programmation qui permet d'apporter des améliorations au langage HTML, en permettant d'exécuter des commandes du coté client, c'est-à-dire au niveau du navigateur et non du coté serveur web. Ainsi le langage JavaScript est fortement dépendant du navigateur appelant la page web dans laquelle le script est incorporé, mais en contrepartie il ne nécessite pas de compilateur, contrairement au langage Java, avec lequel il a longtemps été confondu. Il peut également être utilisé coté serveur comme PHP.

- **Bootstrap**

Bootstrap est une collection d'outils utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur ... etc. ...) de sites et d'applications web. C'est un ensemble qui contient des codes HTML et CSS, des formulaires, boutons, outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option. C'est l'un des projets les plus populaires sur la plate-forme de gestion de développement GitHub.

V.6. Framework de développement

V.6.1. Pourquoi utiliser un Framework ?

Un Framework est un ensemble d'outils et de composants logiciels organisés conformément à un plan d'architecture et des patterns, l'ensemble formant ou promouvant un « squelette » de programme. Il est souvent fourni sous la forme d'une bibliothèque logicielle, et accompagné du plan de l'architecture cible du Framework.

Les avantages des Frameworks sont nombreux. En effet, un Framework est portable, de la part de son abstraction de la base de données et de la gestion générique du cache. Un Framework permet le développement des applications sécurisées. Grâce aux systèmes d'authentification, à la gestion des injections SQL ainsi qu'à la protection CSRF (Cross-Site Request Forgery) qui est gérée par la plupart des Framework. Les Framework sont des outils communautaires et ont, par conséquent, des forums, des listes de diffusion et des canaux IRC pour les soutenir. De plus vu que les Framework sont largement déployés, la chance de trouver les correctifs des problèmes rencontrés est plus grande.

V.6.2. Laravel

Comme Framework de développement nous avons utilisé Laravel. Laravel est un Framework web open-source écrit en PHP respectant le principe modèle-vue-contrôleur et entièrement développé en programmation orientée objet. Laravel est distribué sous licence MIT, avec ses sources hébergées sur GitHub. Laravel, créé par Taylor Otwel, initie une nouvelle façon de concevoir un Framework en utilisant ce qui existe de mieux pour chaque fonctionnalité.

En Mars 2015, Laravel est considéré comme l'un des plus populaires Framework PHP.

V.7. Quelques captures d'écrans