الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Nº Réf :……

Centre Universitaire
Abdelhafid Boussouf Mila

Institut des Sciences et Technologie Département de Mathématiques et Informatique

Mémoire préparé en vue de l'obtention du diplôme de Master

En : Informatique

Spécialité : Sciences et Technologies de l'Information et de la Communication (STIC)

Développement d'une application web pour la gestion des stages au niveau du centre universitaire de Mila

Préparé par

BOUCHELGHOUM Hanifa
MAICHE Norelhouda

Devant le jury

Encadré par	**BOUBAKIR Mohamed**	**MAA**	**C.U. Mila**
Président	**LALOUSSI Ali**	**MAA**	**C.U. Mila**
Examinateur	**MERABET Adil**	**MAA**	**C.U. Mila**

Année Universitaire : 2016/2017

REMERCIEMENT

Nous remercions d'abord et avant tout ***Allah*** *qui nous a donné le courage et la patience pour réaliser ce travail.*

Un remerciement particulier à notre encadreur Monsieur ***Boubakir Mohamed*** *pour sa présence, ses précieux conseils et son aide durant toute la période du travail.*

Nous adressons également nos vifs remerciements aux membres du jury qui ont bien voulu et accepté d'examiner ce modeste travail.

Sans oublier tous les enseignants qui ont contribué à notre formation durant notre vie scolaire particulièrement les enseignants de notre institut.

Finalement, nous remercions très sincèrement tous ***nos famille*** *pour leur encouragement sans limite.*

Hanifa, Norelhouda

DÉDICACE

Je dédie ce modeste travail :

À mes parents ***Mounir et Zoubida****, pour leurs encouragements*

À mes frères : ***Adem et Said****.*

À toute ma famille.

À mes très chères amies.

À notre encadreur ***Boubakir Mohamed*** *et à nos enseignants.*

À tous ceux qui nous ont soutenu au cours de notre travail notamment ***T.Said*** *et* ***Aissam****.*

Noor

DÉDICACE

Je dédie ce travail :

À mes parents ***Abdelkrim et Wahiba****, pour leurs encouragements*

À mes frères : ***Houssem, Omar et Abdallah****.*

*À mes soeurs :****Mouna et Abir***

À toute ma famille.

À mes très chères amies.

À notre encadreur ***Boubakir Mohamed*** *et à nos enseignants.*

À tous ceux qui nous ont soutenu au cours de notre travail notamment **Aissam**.

Hanifa

RÉSUMÉ

Notre projet consiste en la conception et la réalisation d'une application web JEE pour la gestion des stages à l'étranger au niveau du centre universitaire de MILA. Cette application vise à améliorer le traitement des données et l'échange d'informations et de documents entre les différents acteurs concernés par l'opération de gestion de stages. Pour la réalisation de ce projet, nous avons utilisé UML comme langage de modélisation en suivant le processus simplifié. Pour l'implémentation de l'application, nous avons utilisé les technologies web Java (Jsp, Servelet, etc.) sous l'IDE Eclipse en respectant l'architecture MVC. Enfin, nous avons utilisé MySQL comme système de gestion de bases de données. Notre application permettrait aux enseignants et aux doctorants du centre universitaire de MILA de présenter et de suivre leurs candidatures à des stages tout au long des différentes étapes de leur traitement. Le présent mémoire décrit les différentes étapes de réalisation de ce projet.

Mots-clés : 2TUP, UML, MySQL, JEE, IDE, API, stage, candidature, JSP, Servlet

ABSTRACT

Our project consists in design and realization of a JEE web application for the management of the internships at the university center of Mila. This application aims to improve the processing of data and the exchange of information and documentation between the actors involved by the operation of internship management. For the realization of this project, we used UML as a modeling language following the simplified process. For the implementation of the application, we used Java web technologies (Jsp, Servelet, etc.) under the Eclipse IDE and with respecting the MVC architecture. Finally, we used MySQL as database management system. Our application would allow teachers and PhD students from the MILA university center to present and follow their applications for internships throughout the different stages of their treatment. The present mémorie describes the different stages of this project.

Keywords : 2TUP, UML, MySQL, JEE, IDE, API, stage, candidature, JSP, Servlet.

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION GÉNÉRALE

Afin de permettre aux enseignants et aux étudiants d'améliorer leurs compétences, le centre universitaire de Mila leur propose chaque année des stages qui peuvent être à court terme ou à long terme et des conférences. Actuellement, les candidats à ces stages présentent leurs demandes sous forme de dossiers papiers à un service spécialisé au niveau de chaque institut. En plus, une gestion centralisée des stages est assurée par un autre service se trouve au niveau de l'administration du centre. Malheureusement, à l'heure actuelle, la gestion des dossiers des candidats s'effectue manuellement, ce qui provoque beaucoup de problèmes et rend la gestion difficile et moins efficace, surtout lorsque le nombre de dossiers à traiter est considérable. De plus, le risque d'erreur et de perte d'informations est élevé.

Dans ce contexte, nous proposons d'automatiser l'opération de gestion des stages au niveau du centre universitaire de Mila en développant une application web. Cette dernière vise à faciliter l'opération de gestion des stages et la rendre plus efficace et faciliter la communication entre tous les acteurs qui y sont impliqués. Par exemple, elle permet aux candidats de présenter et de suivre l'état de leurs candidatures en ligne sans la nécessité de se déplacer au centre. L'application offre aussi à l'administration et au conseil scientifique la possibilité de travailler sur une seule copie de chaque dossier, ce qui permet d'éviter les problèmes liés au déplacement et au stockage des dossiers papiers tels que le risque

d'erreur et de perte d'informations.

Pour la réalisation de notre travail, nous avons suivi un processus spécialisé au développement des applications web appelé le processus simplifié. Ce dernier s'est inspiré du processus 2TUP (2 Track Unified Process) et s'appuie sur le langage de modélisation UML (Unified Modeling Language). Pour l'implémentation, nous avons utilisé les deux technologies Jsp et Servlet proposées par le langage java pour le développement web au sein de la plateforme JEE. Ces deux technologies accompagnées des Java Beans nous ont permis de développer notre projet en respectant l'architecture MVC (Model Vue Controller) qui a prouvé sans efficacité dans le monde de l'ingénierie de logiciels. Pour le stockage des données, nous avons opté pour le système de gestion de bases de données relationnelles MYSQL.

Nous avons organisé le présent mémoire en quatre chapitres. Dans le premier chapitre, nous présentons quelques généralités sur les applications web, sur le langage de modélisation et le processus de développement utilisé pour l'accomplissement de notre projet, ainsi que sur l'architecture selon laquelle notre projet a été construit. Le deuxième chapitre est consacré à la spécification des besoins de notre système basé sur les cas d'utilisation. Dans le troisième chapitre qui est consacré à l'analyse et la conception, nous appliquons le processus simplifié pour analyser et concevoir les principaux objectifs attendus du futur système et qui ont été décrits par les diagrammes des cas d'utilisation. Le dernier chapitre est consacré à la réalisation. Dans lequel, nous présentons un certain nombre de choix technique qui nous ont servi pour le développement de l'application. Nous présentons aussi quelques tables issues de notre base de données, ainsi que quelques interfaces de l'application. Enfin, nous terminons notre mémoire avec une conclusion générale.

CHAPITRE 1

GÉNÉRALITÉS

1.1 Introduction

Dans ce chapitre, nous présentons quelques généralités sur les méthodes, les concepts et les techniques utilisées pour la réalisation de notre projet. Nous commençons par le processus de développement UP et le processus 2TUP qui l'implémente ainsi que la méthodologie suivie pour la réalisation de notre application qui a été inspirée du modèle UP. Ensuite, nous présentons l'architecture des applications web et les technologies permettant de l'implémenter. Puisque notre application a été développée en utilisant la plateforme JEE en respectant le modèle MVC, nous complétons ce chapitre par une présentation de cette plateforme et de ce modèle.

1.2 Le processus de développement logiciel

1.2.1 Définition

Un processus définit une séquence d'étapes, en partie ordonnées, qui concourent à l'obtention d'un système logiciel ou à l'évolution d'un système existant. L'objet d'un processus de développement est de produire des logiciels de qualité qui répondent aux besoins de leurs utilisateurs dans des temps et des coûts prévisibles [1].

1.2.2 Le processus unifié

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 [1].

1.2.2.1 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.

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

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

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

La gestion d'un tel processus est organisée d'après les 4 phases suivantes :

a) Pré-étude : c'est ici qu'on évalue la valeur ajoutée du développement et la capacité technique à le réaliser (étude de faisabilité).

b) Elaboration : sert à confirmer l'adéquation du système aux besoins des utilisateurs et à livrer l'architecture de base.

c) Construction : sert à livrer progressivement toutes les fonctions du système.

d) Transition : déployer le système sur des sites opérationnels.

Chaque phase est elle-même décomposée séquentiellement en itérations limitées dans le temps. Le résultat de chacune d'elles est un système testé, intégré et exécutable. L'approche itérative est fondée sur la croissance et l'affinement successifs d'un système par le biais d'itérations multiples. Le système croît avec le temps de façon incrémentale, itération par itération, et c'est pourquoi cette méthode porte également le nom de développement itératif et incrémental. Il s'agit là du principe le plus important du Processus Unifié. Ces activités de développement sont définies par 6 disciplines fondamentales qui décrivent la capture des besoins, la modélisation métier, l'analyse et la conception, l'implémentation, le test et le déploiement. Notons que ces différentes étapes (ou disciplines) peuvent se dérouler à travers plusieurs phases. Le processus unifié doit donc être compris comme une trame commune des meilleures pratiques de développement [1].

1.2.3 Le processus 2TUP

La méthode UP est considérée comme générique, c'est-à-dire, elle définit un certain nombre de critères de développement, que chaque société peut par la suite personnaliser afin de créer son propre processus plus adapté à ses besoins. 2TUP signifie « 2 Track Unified Process». C'est un processus qui répond aux caractéristiques du Processus Unifié. Le processus 2TUP apporte une réponse aux contraintes de changement continuel imposées aux systèmes d'information de l'entreprise. En ce sens, il renforce le contrôle sur les capacités d'évolution et de correction de tels systèmes. Comme le montre la figure 1.1, « 2 Track» signifie littéralement que le processus suit deux chemins. Il s'agit des « chemins fonctionnels » et « d'architecture technique », qui correspondent aux deux axes de changement imposés au système d'information.

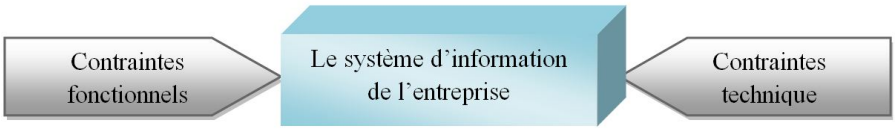

FIGURE 1.1 – Le système d'information soumis à deux types de contraintes.

1.2.3.1 La branche gauche (fonctionnelle) : capitalise la connaissance du métier de l'entreprise. Elle constitue généralement un investissement pour le moyen et le long terme. Les fonctions du système d'information sont en effet indépendantes des technologies utilisées. Cette branche comporte les étapes suivantes :

- La capture des besoins fonctionnels.
- L'analyse.

1.2.3.2 La branche droite (architecture technique) : capitalise un savoir-faire technique. Elle constitue un investissement pour le court et moyen terme. Les techniques développées pour le système peuvent l'être en effet indépendamment des fonctions à réaliser. Cette branche comporte les étapes suivantes :

- La capture des besoins techniques.
- La conception générique.

1.2.3.3 La branche du milieu : à l'issue des évolutions du modèle fonctionnel et de l'architecture technique, la réalisation du système consiste à fusionner les résultats des 2 branches. Cette fusion conduit à l'obtention d'un processus en forme de Y comme montre dans la figure 1.2. Cette branche comporte les étapes suivantes :

- La conception préliminaire.
- La conception détaillée.
- Le codage.
- L'intégration

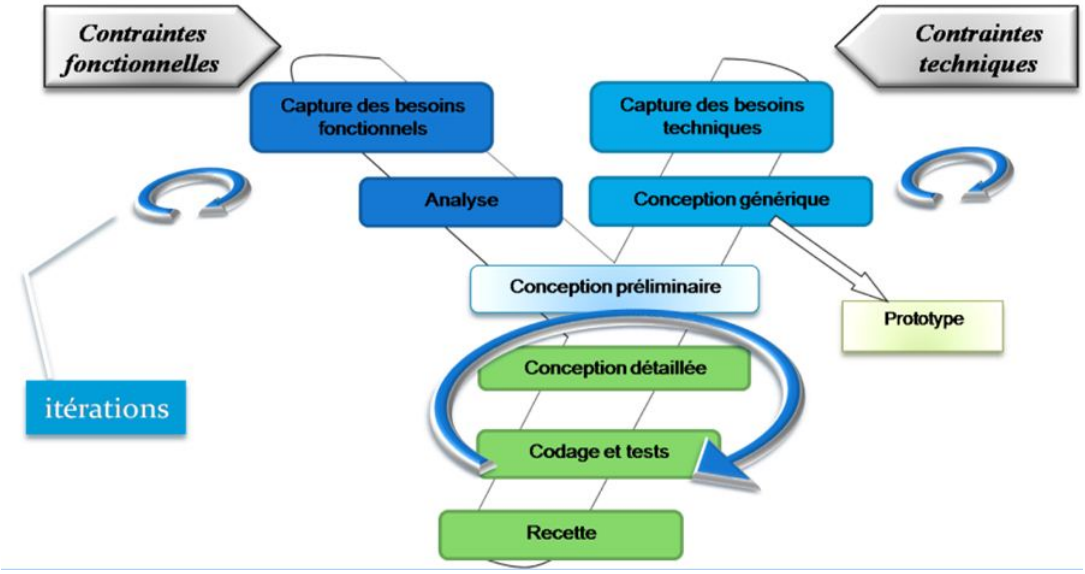

FIGURE 1.2 – Le processus de développement en Y.

1.2.4 Un processus de modélisation avec UML

Le processus 2TUP s'appuie sur UML « Unified Modeling Language » : tout au long du cycle de développement, car les différents diagrammes de ce dernier permettent de part leur facilité et clarté, de bien modéliser le système à chaque étape. UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier, concevoir des solutions et communiquer des points de vue [2]. UML unifie à la fois les notations et les concepts orientés objet. Il ne s'agit pas d'une simple

notation, mais les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage, c'est pour ça qu'UML est présenté parfois comme une méthode alors qu'il ne l'est absolument pas. UML unifie également les notations nécessaires aux différentes activités d'un processus de développement et offre, par ce biais, le moyen d'établir le suivi des décisions prises, depuis la définition des besoins jusqu'au codage [3]. Voici une présentation rapide des différents diagrammes.

1.2.4.1 Le diagramme des cas d'utilisation : représente la structure des fonctionnalités nécessaires aux utilisateurs du système. Il est normalement utilisé lors des étapes de capture des besoins fonctionnels et techniques.

1.2.4.2 Le diagramme d'activité : représente les règles d'enchaînement des activités et actions dans le système. Il peut être assimilé comme un algorithme mais schématisé.

1.2.4.3 Le diagramme de package : présent depuis UML 2.0, ce diagramme modélise des catégories cohérentes entre elles, pour un souci de partage des rôles. Correspond à l'étape de modélisation des différents scénarios d'un cas d'utilisation.

1.2.4.4 Le diagramme de classes : sûrement l'un des diagrammes les plus importants dans un développement orienté objet. Sur la branche fonctionnelle, ce diagramme est prévu pour développer la structure des entités manipulées par les utilisateurs. En conception, le diagramme de classes représente la structure d'un code orienté objet.

1.2.4.5 Le diagramme de séquence : représente les échanges de messages entre objets, dans le cadre d'un fonctionnement particulier du système.

1.2.4.6 Le diagramme d'état : représente le cycle de vie d'un objet. Il spécifie les états possibles d'une classe et leur enchainement. Ce diagramme est utilisé lors des étapes d'analyse et de conception.

1.2.4.7 Le diagramme de composant : représente les concepts connus de l'exploitant pour installer et dépanner le système. Il s'agit dans ce cas de déterminer la structure des composants d'exploitation que sont les librairies dynamiques, les instances de bases de données, les applications, les progiciels, les objets distribués, les exécutables, etc.

1.2.4.8 Le diagramme de déploiement : correspond à la fois à la structure du réseau informatique qui prend en charge le système logiciel, et la façon dont les composants d'exploitation y sont installés.

1.3 Un processus simplifié pour le développement des applications web

Nous allons présenter ici une démarche de modélisation (pour plus de détails consulter le [3]) permettant un développement efficace des applications web. Cette démarche s'inspire du processus unifié et des méthodes agiles telles que XP (eXtreme Programming) [3]. Elle s'appuie sur le langage UML dans ces différentes étapes de développement.

1.3.1 Problématique

Dans un processus de développement logiciel, le problème fondamental qu'on cherche à résoudre est généralement simple, il s'agit de répondre à la question suivante : comment passer des besoins des utilisateurs au code de l'application ? Cette problématique est illustrée par la figure 1.3.

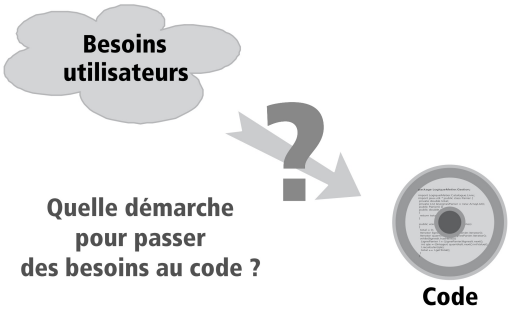

FIGURE 1.3 – Comment passer des besoins au code ?

1.3.2 Identification des besoins et spécification des fonctionnalités

1.3.2.1 Spécification des exigences d'après les cas d'utilisation

L'expression préliminaire des besoins des utilisateurs donne lieu à une modélisation par les cas d'utilisations et à une maquette d'interface homme-machine (IHM). Comme indiqué sur la figure 1.4.

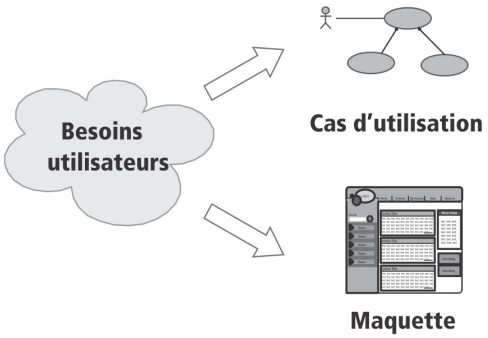

FIGURE 1.4 – Démarche des besoins qui conduisent à des cas d'utilisation et à une maquette.

a) Les maquettes : La réalisation d'une maquette graphique est une activité courante mettant en jeu des outils de dessin. Elle montre rapidement l'aspect visuel (le « look ») du site web.

b) 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. Les différentes étapes de la démarche adoptée afin d'aboutir au modèle des cas d'utilisation (voir synoptique de la figure 1.5 ci-dessous) :

- identifier les acteurs.
- identifier les cas d'utilisation.
- structurer les cas d'utilisation en packages.
- ajouter les relations entre cas d'utilisation.

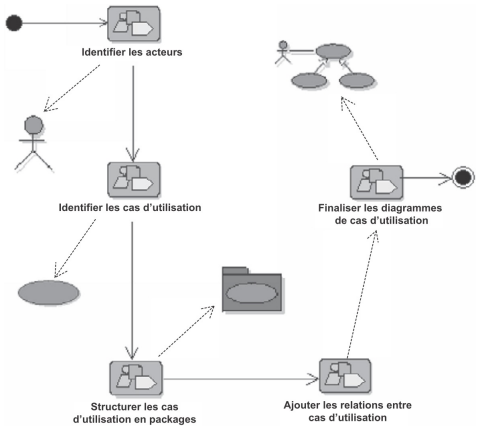

FIGURE 1.5 – Synoptique de la démarche de construction du modèle des cas d'utilisation.

1.3.2.2 Spécification détaillée des exigences

Nous décrivons de façon détaillée les cas d'utilisation par une description textuelle. On complète cette description textuelle par une représentation graphique UML très utile : le diagramme de séquence système.

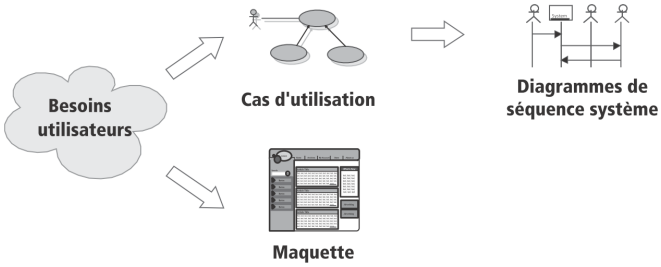

FIGURE 1.6 – Les cas d'utilisation et leurs prolongements dans la démarche.

1.3.3 Phases d'analyse

1.3.3.1 Réalisation des cas d'utilisation

Dans cette étape, on effectue la réalisation des cas d'utilisation (classes d'analyse) qui consiste à élaborer les diagrammes de classes participantes en se basant sur les cas d'utilisation et les maquettes (voire la figure 1.7), Le procédé de cette réalisation débutera par une identification des concepts du domaine, Par la suite, on ajoute des attributs à ces

entités et nous les relions entre eux par des associations. Enfin, on réalise les diagrammes de classes d'analyse participantes du système.

a) Identification des concepts du domaine

L'étape typiquement orientée objet de l'analyse est la décomposition d'un domaine d'intérêt en classes conceptuelles représentant les entités significatives de ce domaine. Il s'agit simplement de créer une représentation visuelle des objets du monde réel dans un domaine donné.

b) Ajout des associations et des attributs

Une fois que l'on a identifié les concepts fondamentaux, il est utile d'ajouter :

- associations nécessaires pour prendre en compte les relations qu'il est fondamental de mémoriser.
- attributs nécessaires pour répondre aux besoins d'information. Reprenons donc les cas d'utilisation majeurs.

c) Typologie des classes d'analyse

Pour élargir cette première identification des concepts du domaine, nous allons utiliser une catégorisation des classes d'analyse qui a été proposée par I. Jacobson et popularisée ensuite par le RUP (Rational Unified Process). Les classes d'analyse qu'ils préconisent se répartissent en trois catégories :

- qui permettent les interactions entre le système et ses utilisateurs sont qualifiées de dialogues. Il s'agit typiquement des écrans proposés à l'utilisateur : les formulaires de saisie, les résultats de recherche, etc.
- Les classes qui contiennent la cinématique de l'application sont appelées contrôles. Elles font la transition entre les dialogues et les concepts du domaine, en permettant aux écrans de manipuler des informations détenues par des objets métier. Elles contiennent les règles applicatives et les isolent à la fois des objets d'interface et des données persistantes.
- Celles qui représentent les concepts métier sont qualifiées d'entités. Elles sont très souvent persistantes, c'est-à-dire qu'elles vont survivre à l'exécution d'un cas d'uti-

lisation particulier et qu'elles permettront à des données et des relations d'être stockées dans des fichiers ou des bases de données.

d) Diagramme de classe participant :

Il s'agit de diagrammes de classes participantes (DCP) UML qui décrivent, cas d'utilisation par cas d'utilisation, les trois principales classes d'analyse et leurs relations.

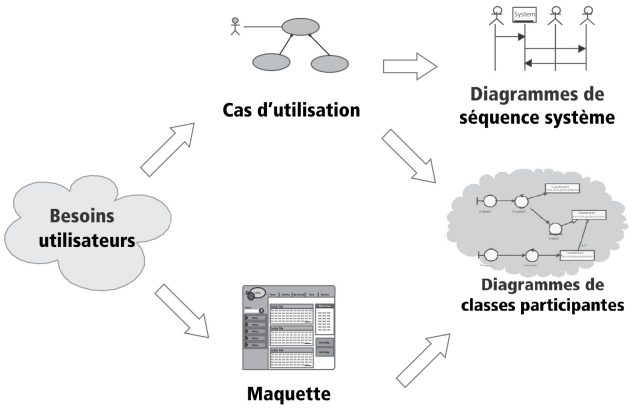

FIGURE 1.7 – DCP font la jonction entre les cas d'utilisation et la maquette.

Pour compléter ce travail d'identification, on ajoute des attributs et des opérations dans les classes d'analyse, ainsi que des associations entre elles.

- entités vont seulement posséder des attributs. Ces attributs représentent en général des informations persistantes de l'application.
- Les contrôles vont seulement posséder des opérations. Ces opérations montrent la logique de l'application, les règles transverses à plusieurs entités.
- Les dialogues vont posséder des attributs et des opérations. Les attributs représenteront des champs de saisie ou des résultats. Les résultats seront distingués en utilisant la notation de l'attribut dérivé. Les opérations représenteront des actions de l'utilisateur sur l'IHM.

Nous allons également ajouter des associations entre les classes d'analyse, mais en respectant des règles assez strictes :

- dialogues ne peuvent être reliés qu'aux contrôles ou à d'autres dialogues, mais pas directement aux entités.

– Les entités ne peuvent être reliées qu'aux contrôles ou à d'autres entités.

– Les contrôles ont accès à tous les types de classes, y compris d'autres contrôles.

Enfin, nous ajouterons les acteurs sur les diagrammes en respectant la règle suivante : un acteur ne peut être lié qu'à un dialogue.

1.3.3.2 Diagramme de navigation :

Les IHM modernes facilitent la communication entre l'application et l'utilisateur en offrant toute une gamme de moyens d'action et de visualisation comme des menus déroulants ou contextuels, des palettes d'outils, des boîtes de dialogues, des fenêtres de visualisation, etc. Cette combinaison possible d'options d'affichage, d'interaction et de navigation aboutit aujourd'hui à des interfaces de plus en plus riches et puissantes. UML offre la possibilité de représenter graphiquement cette activité de navigation dans l'interface en utilisant les digrammes d'état. Cela permet de produire des diagrammes dynamiques appelés diagrammes de navigation. Pour réaliser ces derniers de navigation nous nous basons sur les maquettes et DCP. Comme montrer sur la figure 1.8.

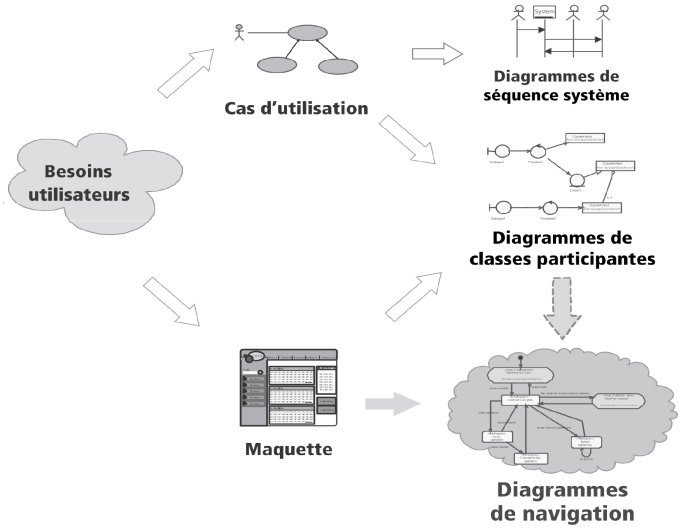

FIGURE 1.8 – Démarche de la maquette et DCP qui conduisent à un diagramme de navigation.

La figure 1.9 montre les conventions graphiques utilisées pour la modélisation des diagrammes de navigation.

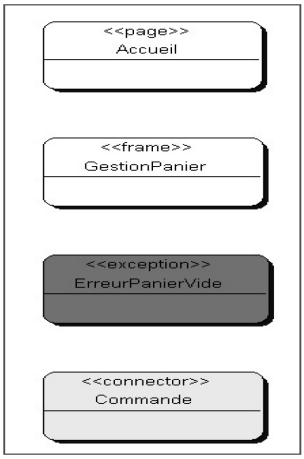

FIGURE 1.9 – Conventions graphiques spécifiques.

- une page complète de l'application («page»).
- un frame particulier à l'intérieur d'une page («frame»).
- une erreur ou un comportement inattendu du système («exception » avec un niveau de gris intermédiaire).
- une liaison vers un autre diagramme d'activité, pour des raisons de structuration et de lisibilité («connector» avec un niveau de gris soutenu).

1.3.4 Phase de conception

1.3.4.1 Diagramme d'interaction

Maintenant, il faut attribuer précisément les responsabilités de comportement, dégagées par le diagramme de séquence système dans la section 1.3.2.2, aux classes d'analyse du diagramme de classes participantes élaboré dans la section 1.3.3.1, Les résultats de cette réflexion sont présentés sous la forme de « diagrammes d'interaction » UML (voir la figure 1.10).

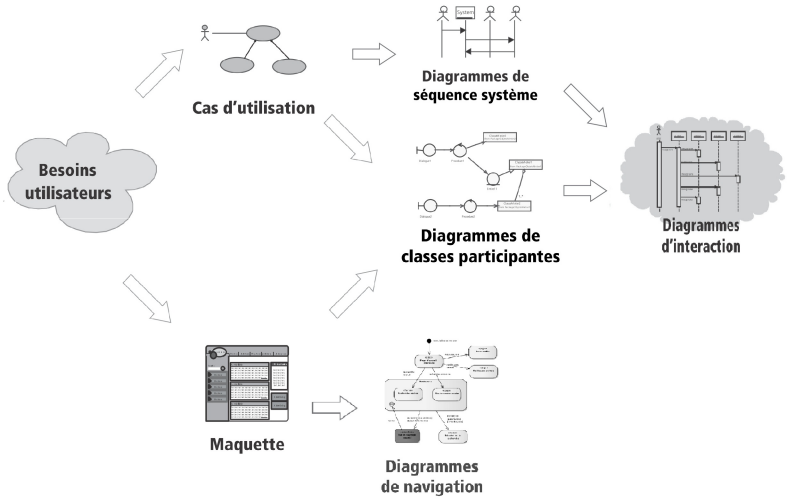

FIGURE 1.10 – Démarche de réalisation de diagrammes d'interaction.

Par rapport aux diagrammes de séquence système, le système vu comme une boîte noire est remplacé par un ensemble d'objets en interaction. Pour cela, les trois types de classes d'analyse (les dialogues, les contrôles et les entités) sont encore utilisés tout en respectant les règles qui ont été fixées sur les relations entre classes d'analyse (Section 1.3.3.1), et en s'intéressant cette fois-ci aux interactions dynamiques entre objets :

- Les acteurs ne peuvent interagir (envoyer des messages) qu'avec les dialogues.
- Les dialogues peuvent interagir avec les contrôles.
- Les contrôles peuvent interagir avec les dialogues, les entités, ou d'autres contrôles.
- Les entités ne peuvent interagir qu'entre elles.

Le changement de niveau d'abstraction par rapport au diagramme de séquence système peut ainsi être représenté sur la figure 1.11.

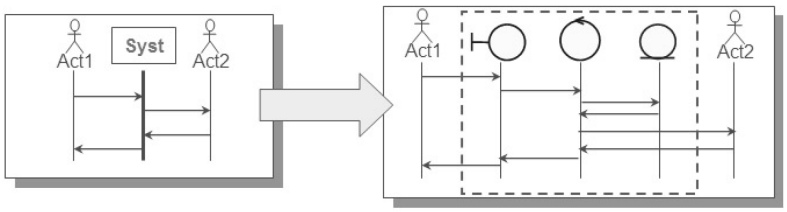

FIGURE 1.11 – Passage de diagramme de séquence au diagramme d'interaction.

1.3.4.2 Conception objet détaillée (diagramme de classes de conception) :

L'objectif de cette étape est de produire le diagramme de classes qui servira pour l'implémentation (voir la figure 1.12). Une première ébauche du diagramme de classes de conception a déjà été élaborée en parallèle du diagramme d'interaction. Il faut maintenant le compléter en précisant les opérations privées des différentes classes. Il faut prendre en comptes les choix techniques, comme le choix du langage de programmation, le choix des différentes bibliothèques utilisées (notamment pour l'implémentation de l'interface graphique), etc.

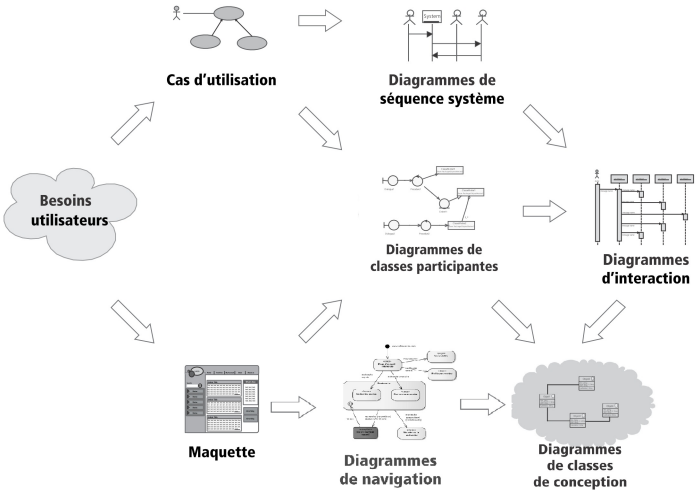

FIGURE 1.12 – Démarche de réalisation de diagrammes de classes de conception.

1.4 Les applications web

1.4.1 Définition

Une application web est un logiciel hébergé sur un serveur, accessible depuis n'importe quel navigateur web via un réseau informatique (internet, etc.) dont l'objectif est de faciliter l'accomplissement d'une tâche précise sur le Web [4].

1.4.2 Architecture d'une application web

Le web est un ensemble de machines en réseau communiquant à l'aide d'un langage commun. Comme présenté sur la figure 1.13, le web fonctionne en mode client/serveur c'est-à-dire qu'il y a des machines dites serveurs qui proposent des ressources et des machines appelées clients qui utilisent ces ressources. Les ressources sont par exemple des pages HTML (Hyper Text Markup Language), des images, des fichiers XML(eXtensible Markup Language) ou encore des programmes (PHP, Java, ASP.NET, Python, Perl, etc) chargés de les générer à la demande. Le client accède aux ressources à l'aide du protocole de communication HTTP (Hypertexte Transfer Protocol) : Ce dernier permettant de formuler des requêtes qui sont interprétées côté serveur par un programme spécifique : le serveur web. En plus du serveur web, nous pouvons avoir un serveur de données qui va héberger le Système de Gestion de Base de Données (SGBD). Et pour y accéder, on utilise le langage universel d'interrogation des bases de données : SQL. Ci-dessous l'architecture d'une application web.

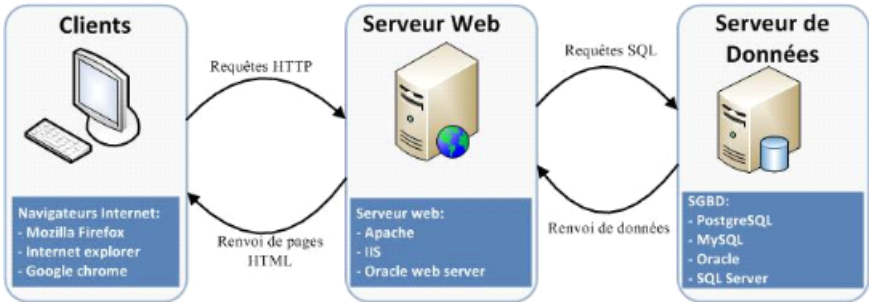

FIGURE 1.13 – Architecture d'une application web.

1.4.3 Les technologies utilisées dans une application web

1.4.3.1 Les technologies coté client

Les technologies côté client permettent de gérer l'interface utilisateur de chaque page parmi ces technologies, nous citons :

– **HTML (Hyper Text Markup Language) :** est un langage permettant de décrire

la mise en page et la forme du contenu d'un document web et d'y inclure des hyperliens.

- **CSS (Cascading Style Sheet) :** est un langage de mise en forme qui permet de décrire la présentation d'un document (positionnement des éléments, l'alignement, les polices de caractères, les couleurs, les marges et espacements, les bordures, les images de fond, etc.) écrit en HTML ou XML indépendamment de sa structure.
- **JAVAScript :** c'est un langage de script dérivé de java. JavaScript permet de dynamiser la présentation du contenu (animations, textes défilants...) ou de contrôler les données saisies dans des formulaires html.

1.4.3.2 Les technologies coté serveur

Les technologies coté serveur permettent de générer des pages en langage client parmi les technologies coté serveur, nous citons :

- **PHP :** est un langage de script qui est lu et exécuté sur le serveur où se trouve la page HTML, avant que celle-ci ne soit envoyée au navigateur (le client) qui en demande l'affichage.
- **ASP (Active Server Pages) :** est une technologie Web permettant d'exécuter des scripts côté serveur et développer des applications Web dynamiques en fournissant un accès simple à des bases de données.

  N/B il existe autre technologie tel que JSP et Servlet que nous présenterons plus tard.

1.4.3.3 Protocole HTTP

Hypertexte Transfer Protocol est un protocole de communication informatique client-serveur développé pour le World Wide Web. Il est utilisé pour transférer les documents (document HTML, image, feuille de style, etc.) entre le serveur HTTP et le navigateur Web.

1.4.3.4 Système gestion de base de données

Un Système de Gestion de Base de Données (SGBD) est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la

base de données. Parmi les logiciels les plus connus il est possible de citer : MySQL, PostgreSQL, SQLite, Oracle Database, Microsoft SQL Server, Firebird ou Ingres [12].

1.4.4 Pourquoi créer une application Web ?

Les applications web offrent de nombreux avantages aux utilisateurs :

- accés universel depuis n'importe quel type de poste : PC, portables, téléphone mobile, tablette,etc.
- il n'est plus nécessaire d'installer un logiciel sur chaque poste, ce qui diminue en grande partie les frais de maintenance et élimine certaines incompatibilités.
- ces applications sont capables de gérer des données multimédia (textes, photos, vidéos).
- certaines parties de ces applications sont ouvertes aux clients et fournisseurs et facilitent ainsi les échanges d'informations entre les partenaires.
- pour mettre à jour l'application, il suffit de modifier l'application sur le serveur et tous les postes accèdent instantanément à la nouvelle version.
- Diminuer les coûts de développement et de support technique.
- La mise en ?uvre et le déploiement sont plus rapides.

1.5 Java EE

1.5.1 Définition

Le terme « Java EE » signifie Java Enterprise Edition, et était anciennement raccourci en « J2EE ». Il fait quant à lui référence à une extension de la plate-forme standard. Autrement dit, la plate-forme Java EE est construite sur le langage Java et la plate-forme Java SE, et elle y ajoute un grand nombre de bibliothèques remplissant tout un tas de fonctionnalités que la plate-forme standard ne remplit pas d'origine. L'objectif majeur de Java EE est de faciliter le développement d'applications web robustes et distribuées,

déployées et exécutées sur un serveur d'applications [16].

1.5.2 la plate forme JEE

Java EE est une norme spécifiant à la fois l'infrastructure de gestion des applications et les APIs (Application Program Interfaces) des services utilisés pour concevoir ces applications. Les spécifications du serveur d'applications, c'est-à-dire de l'environnement d'exécution java EE définit finement les rôles et les interfaces pour les applications ainsi que l'environnement dans lequel elles seront exécutées. Ces recommandations permettent ainsi à des entreprises tierces de développer des serveurs d'applications conformes aux spécifications ainsi définies, sans avoir à redévelopper les principaux services [7]. Ces services sont des extensions JAVA indépendantes, permettant d'offrir en standard un certain nombre de fonctionnalités. Oracle fournit une implémentation minimale de ces API appelée java EE SDK (J2EE Software Development Kit). Les différents services et/ou fonctionnalités associées à J2EE sont classés comme suit :

- composants Web.
- Les composants métiers.
- Les services d'infrastructure et de communication.

La figure suivante présente les différentes technologies utilisées par la plate forme J2EE.

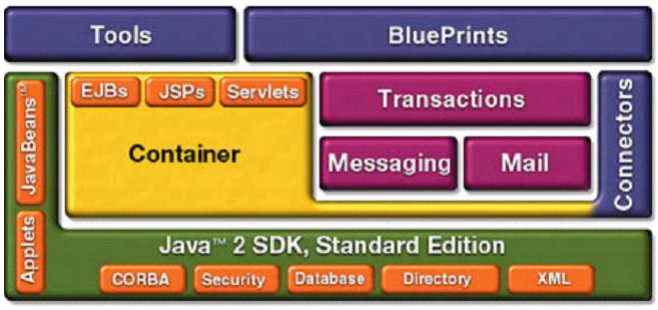

FIGURE 1.14 – Outils J2EE.

1.5.2.1 Les composants Web

Le serveur web d'une application java EE rend disponible la logique d'une application sur le web, donc c'est le serveur web qui gère la communication avec les clients web et qui répond à leurs requêtes.

- **Les servlets**

  Une servlet est en réalité une simple classe Java, qui a la particularité de permettre le traitement de requêtes et la personnalisation de réponses. Pour faire simple, dans la très grande majorité des cas une servlet n'est rien d'autre qu'une classe capable de recevoir une requête HTTP envoyée depuis le navigateur de l'utilisateur, et de lui renvoyer une réponse HTTP [6].

- **JSP**

  Les pages JSP (Java Server Page) sont une des technologies de la plate-forme Java EE les plus puissantes. Cette technologie dont l'objectif est le développement web se caractérise par sa simplicité d'utilisation et de mettre en place. Elles se présentent sous la forme d'un simple fichier au format texte, contenant des balises respectant une syntaxe à part entière. Le langage JSP combine à la fois les technologies HTML, XML, servlet et JavaBeans [6].

1.5.2.2 Les composants métiers

Les parties liées au domaine d'application, qui s'occupent de traiter la logique métier.

- **Les Java Beans**

  Les JavaBeans sont des composants écrits en Java (classe java) qui peuvent être employés dans tous les tiers d'une application. Ils sont souvent considérés comme un complément des servlets ou comme un composant des interfaces utilisateur. La spécification JavaBeans définit les composants du type JavaBeans comme des composants logiciels réutilisables manipulables visuellement dans un outil de conception.

- **Les EJB**

  Les EJB (Entreprise Java Beans) sont des composants spéciaux, fonctionnant sur un serveur et utilisés pour construire la logique applicative et permet d'accéder aux données des applications java EE. Ils possèdent certaines caractéristiques comme la

réutilisabilité, la possibilité de s'assembler pour construire une application, etc.

1.5.2.3 Les services d'infrastructures et de communication

Les services principaux sont :

a) Services d'infrastructure :

- **JDBC :** (Java Data Base Connectivity), permet l'accès aux bases de données. JDBC est une API d'accès aux systèmes de gestion de bases de données relationnelles qui permet d'exécuter des requêtes au sein d'un programme Java et de récupérer les résultats, ce qui représente une alternative aux solutions propriétaires. C'est de plus une tentative de standardiser l'accès aux bases de données car L'API est indépendant du SGBD choisi.
- **JTA/JTS :** (Java Transaction API/ Java Transaction), est un API définissant des interfaces standard avec un gestionnaire de transactions.
- **JCA :** (java EE Connector Architecture), est un API de connexion au système d'information de l'entreprise.
- **JMX :**(Java Management Extension), fournit des extensions permettant de développer des applications web de supervision d'application.
- (DNS, LDAP, etc.).

b) Services de communication

- **JAAS :** (Java Authentification Autorisation Service), service de gestion des droits d'accès.
- **RMI :** (Remote Méthode Invocation), qui permet la communication entre objets distants.
- **JavaMail :** permettant l'envoi de courrier électronique.
- **JMS :**(Java Message Service), fournit des fonctionnalités de communication asynchrone (appelées MOM pour Middleware Object Message) entre applications.

1.6 Le modèle MVC

1.6.1 Définition

L'architecture Modèle Vue Contrôleur (MVC) est une méthode de conception pour le développement d'applications logicielles qui sépare le modèle de données, l'interface utilisateur et la logique de contrôle. Ce modèle d'architecture impose la séparation entre les données, les traitements et la présentation, ce qui donne trois parties fondamentales dans l'application finale : le modèle, la vue et le contrôleur [8].

1.6.2 Les composants de modèle MVC

1.6.2.1 Le modèle : représente le comportement de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application et définit les méthodes d'accès.

1.6.2.2 La vue : correspond à l'interface avec laquelle l'utilisateur interagit. Les résultats renvoyés par le modèle sont dénués de toute présentation mais sont présentés par les vues. Plusieurs vues peuvent afficher les informations d'un même modèle. Elle peut être conçue en HTML, ou tout autre « langage » de présentation. La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle, et de permettre à l'utilisateur d'interagir avec elles.

1.6.2.3 Le contrôleur : prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle. Il n'effectue aucun traitement, ne modifie aucune donnée, il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande. En résumé, lorsqu'un client envoie une requête à l'application, celle-ci est analysée par le contrôleur, qui demande au modèle approprié d'effectuer les traitements, puis renvoie la vue adaptée au navigateur, si le modèle ne l'a pas déjà fait.

1.6.3 Interactions entre les composants

La figure 1.15, résume les relations entre les composants d'une architecture MVC :

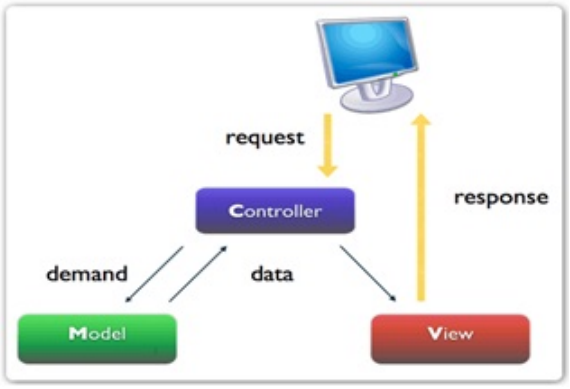

FIGURE 1.15 – L'interaction entre les composants.

La demande de l'utilisateur (exemple : requête HTTP) est reçue et interprétée par le contrôleur. Celui-ci utilise les services du modèle afin de préparer les données à afficher. Ensuite, le contrôleur fournit ces données à la vue, qui les présente à l'utilisateur (par exemple sous la forme d'une page HTML) [9].

1.7 Conclusion

Dans ce chapitre, nous avons introduit le processus 2TUP, nous avons expliqué la méthodologie de développement que nous avons adoptée pour la réalisation de notre projet, nous avons rappelé quelques notions sur les applications web, et enfin nous avons présenté le modèle MVC.

CHAPITRE 2

SPÉCIFICATION DES BESOINS

2.1 Introduction

Ce chapitre est consacré à la spécification des exigences qui est considérée comme une étape essentielle dans chaque processus de développement logiciel. Nous présentons en premier temps d'une manière générale notre futur système, ses fonctionnalités prévues ainsi que les solutions proposées pour résoudre les problèmes du système actuel. Ensuite, nous spécifions les exigences de notre système en utilisant les cas d'utilisation UML, et nous présentons les maquettes qui montrent l'aspect visuel de l'application Web. Enfin, nous détaillons la spécification des exigences en utilisant les diagrammes de séquences UML, ainsi qu'une description textuelle détaillée de cas d'utilisation.

2.2 Description du projet

2.2.1 Présentation de l'existant

Comme tout les établissements de l'enseignement supérieur, le centre universitaire de Mila offre aux enseignants, aux étudiants la possibilité d'effectuer des stages à l'étranger afin de les aider à améliorer leurs compétences. Les stages peuvent être en séjour scientifique de haut niveau de courte ou de longue durée,ou de conférences.

Actuellement, les demandes de stages sont reçues sous forme de dossiers papiers, dont chacun contient un ensemble de documents comme par exemple un CV, une demande manuscrite, le diplôme, etc. Le traitement et la sélection de ces dossiers se fait manuellement, ce qui rend l'opération lente et compliquée, et ne permet pas de mieux exploiter par la suite les informations issue des dossiers.

2.2.2 Problématique

Vu la quantité importante des demandes reçues ainsi que la difficulté de stocker et de traiter les dossiers papiers, l'opération entière de gestion des stages (réception et traite-

ment des dossiers, sélection et suivie des dossiers acceptés, etc) devienne une tâche très compliquée. En plus, le risque de perte des données est très élevé à cause de la quantité importante de documents circulant d'une personne à une autre, et qu'il faut imprimer, signer, agrafer, mettre sous pli, etc. D'une manière plus précise, la liste des problèmes suivants est dégagée :

- difficulté de consultation des demandes.
- Aucune cohérence entre les données.
- La redondance de l'information.
- Archivage papier et espace de stockage très important.
- Une grande perte de temps lors de la recherche d'une information.
- L'absence des statistiques.
- Perte de temps au niveau de l'échange des documents.
- Risque de perte de document.

2.2.3 Solution proposée

Le dégagement des problèmes nous a menés à la définition des missions suivantes :

- Etablir un formulaire pour les demandes de stage contenant toutes les informations nécessaires : l'état civil du demandeur, sa formation, le type, dates de début et fin du stage, ainsi que la possibilité de joindre le curriculum vitae pour être exploitées par la suite.
- Consulter, rechercher, éditer et accéder facilement aux informations relatives aux demandes et stages effectués telles que : stagiaire, date, rapport...
- Garder trace des demandes refusées/acceptées pour pouvoir générer par la suite des statistiques qui permettent d'avoir une vue globale sur le déroulement des stages au sein du service.
- Générer des comptes stagiaires pour pouvoir mettre à jour leurs informations et télécharger leurs rapports de stage.
- Faciliter le contact entre l'administrateur et les stagiaires en réalisant une messagerie interne.

– Avoir à tout moment la liste des stages en cours en mentionnant le temps restant pour leur fin et une liste des demandes non traitées dont la date de début du stage est très proche.

2.3 Identification des besoins et spécification des fonctionnalités

2.3.1 Spécification des exigences d'après les cas d'utilisation

Nous présenterons dans cette section les modèles de cas d'utilisation dont la réalisation a été faite selon le modèle présenté dans le chapitre précédent et illustrée dans la figure 1.5. Nous commençons par l'identification des acteurs et des cas d'utilisation. Ensuite, nous ajoutons des relations, et enfin nous raffinons nos diagrammes. Afin de bien présenter ces diagrammes, nous les structurons dans des packages. Certains cas d'utilisation sont caractérisés secondaires et ils sont regroupés ensemble dans un seule package appelé de second rang.

2.3.1.1 Identification des acteurs

Les acteurs de notre système sont les suivants :

a) Demandeur de stage : la personne qui utilise l'application pour effectuer en ligne, les opérations de consultation des annonces, présentation et suivie des candidatures grâce à son espace personnel. Le demandeur de stage peut en plus gérer les informations de son compte. Il existe deux types de demandeurs de stages : les enseignants et les étudiants en doctorat (les doctorants).

b) Administrateur : c'est l'acteur responsable de la gestion administrative des stages. Nous pouvons distinguer deux types d'administrateurs, un administrateur central qui gère les stages au niveau de tout le centre universitaire, et des administrateurs locaux, dont chacun s'occupe de la gestion des stages au niveau d'un institut.

c) Conseil scientifique : cet acteur est chargé de réaliser les taches suivantes : établissement des critères scientifiques pour la sélection des candidatures, étude des candidatures afin de les noter, classement des candidatures, et étude des recours.

2.3.1.2 Identification des cas d'utilisation

Pour chaque acteur précédemment identifié, il convient de rechercher les différentes intentions « métier » selon lesquelles il utilise le système.

2.3.2.1.1 Les cas d'utilisation de l'acteur demandeur de stage

a) Gestion des dossiers : Ce cas d'utilisation comporte trois cas :

- **Créer dossier :** permet à chaque demandeur de créer un dossier contenant des informations pour pouvoir présenter des candidatures à des stages.
- **Consulter dossier :** permet aux demandeurs de consulter leurs dossiers.
- **Modifier dossier :** permet aux demandeurs de modifier certaines informations de leurs dossiers.

b) Ajouter candidature : Le demandeur de stage présente des candidatures afin de bénéficier des stages offerts par le centre universitaire. Le nombre de candidatures est limité (actuellement, deux par an).

c) Consulter la liste initiale globale : : Après, l'étude des candidatures, ce cas permet aux demandeurs de consulter la liste des candidatures acceptées.

d) Compléter dossier : Après l'acceptation de sa candidature, le demandeur doit compléter certaines informations comme par exemple la date de départ et celle d'arrivée, ainsi que l'aéroport de départ et l'aéroport d'arrivée.

e) Faire un recours : lorsque le demandeur de stage consulte la liste des candidatures acceptées, il peut faire des recours s'il n'est pas satisfait des résultats.

2.3.2.1.2 Les cas d'utilisation de l'acteur administrateur

a) Effectuer des statistiques : l'administrateur peut consulter des statistiques sur l'opération de gestion des stages, il peut par exemple, voir les candidatures acceptées et

refusées pour une année donné, le nombre de candidatures acceptées par institut, etc.

b) Annoncer les stages : l'administrateur du centre permet d'informer les demandeurs dès l'ouverture de la session de candidatures.

c) Affecter les stages : l'administrateur du centre affecte les stages à chaque institut.

d) Vérifier les dossiers : l'administrateur de chaque institut vérifie les dossiers créés par les candidats (vérifie la conformité des informations saisie comme par exemple, le nom, le prénom, le diplôme, la spécialité, etc).

e) Créer la liste initiale : L'administrateur de chaque institut crée la liste des candidatures acceptées au niveau de cet institut en se basant sur classement des candidatures effectué par le conseil scientifique.

f) Vérifier les listes : l'administrateur du centre vérifie la liste initiale de chaque institut, et la liste globale avant et après les rectifications.

g) Créer la liste initiale globale : l'administrateur du centre vérifie la liste initiale des candidatures acceptées de chaque institut, puis crée la liste globale des candidatures acceptées au niveau du centre.

h) Mettre à jour la liste initiale globale : après l'étude des recours par le conseil scientifique de l'institut, l'administrateur de chaque institut rectifie la liste initiale globale.

i) Afficher la liste : l'administrateur du centre affiche la liste initiale globale avant et après les rectifications.

2.3.2.1.3 Les cas d'utilisation de l'acteur conseil scientifique :

a) Etablir les critères de classement : le conseil scientifique du centre établit les critères du classement.

b) Noter les candidatures : Après la clôture de la session des candidatures, le conseil scientifique de l'institut étudie, puis note chaque candidature selon les critères spécifiés par le conseil scientifique du centre.

c) Classer les candidatures : Après avoir noter les candidatures, le conseil scienti-

fique va les classer d'après leurs notes

d) Etudier les recours : Le conseil scientifique étudie les recours que fait le demandeur après l'affichage de la liste initiale globale.

Le cas d'utilisation nommé *s'authentifier* est utilisé afin de permettre aux utilisateurs d'application d'exécuter ses propres cas d'utilisation majeurs. Nous qualifierons donc ce petit cas d'utilisation de *fragment* : il ne représente pas un objectif à part entière des utilisateurs, mais plutôt un objectif de niveau intermédiaire. En plus, il y a de cas d'utilisation secondaires (notés par le stéréotype secondaire) qui ne sont pas des cas d'utilisation majeurs, mais ils donnent lieu à des développements importants au niveau du projet (ex : demander un compte, gérer sa messagerie, gérer son compte).

2.3.3 Structuration en packages

Nous organisons les cas d'utilisation et les regroupons en ensembles fonctionnels cohérents. Pour ce faire, nous utilisons le concept du package UML. Les acteurs ont également été regroupés dans un package. Le cas d'utilisation *s'authentifier* sera structuré dans un package à part intitulé *use case* de second rang. La figure 1 illustre l'organisation des cas d'utilisation et des acteurs en packages.

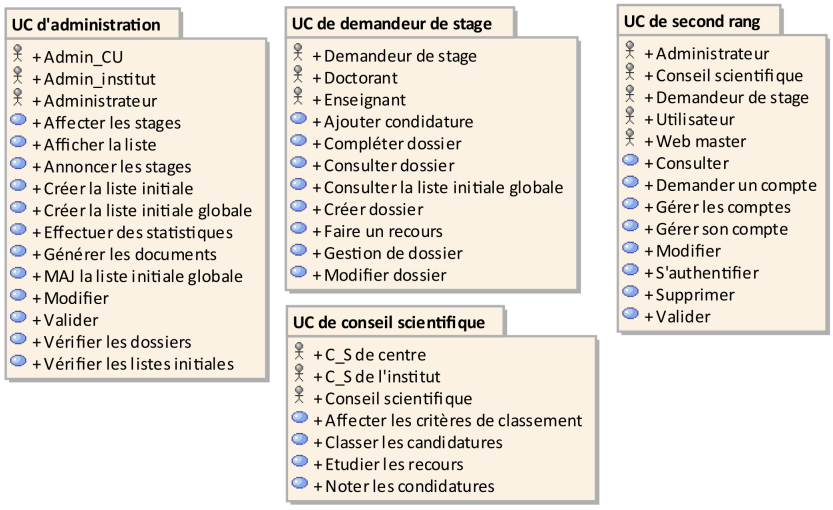

FIGURE 2.1 – Organisation des cas d'utilisation et des acteurs en package.

2.3.4 Les relations entre les cas d'utilisation

Les quatre figures qui vont suivre représentant les diagrammes de cas d'utilisation, dont chacun montre les relations entre les cas d'utilisation et les acteurs.

2.3.4.1 Diagramme des cas d'utilisation du demandeur de stage

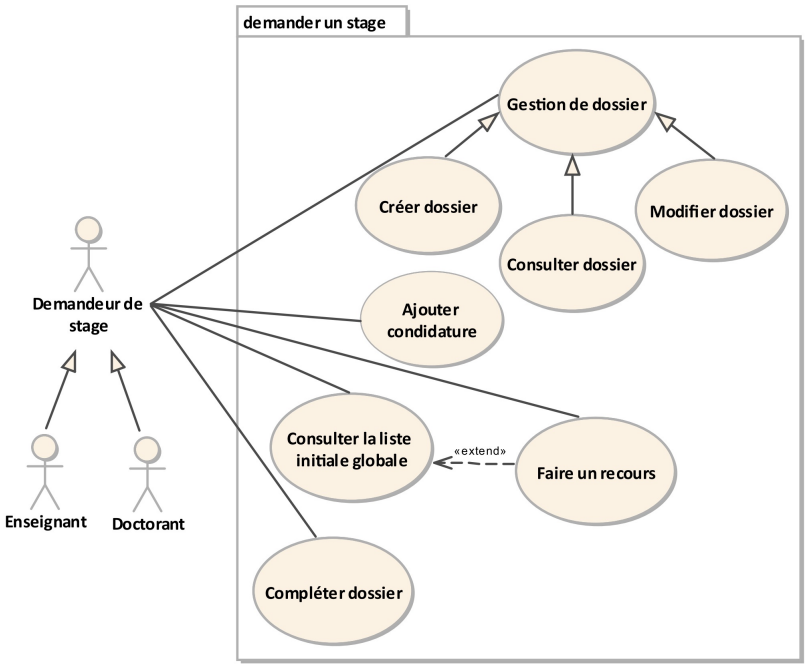

FIGURE 2.2 – Diagramme des cas d'utilisation « demander un stage ».

2.3.4.2 Diagramme des cas d'utilisation de l'Administrateur

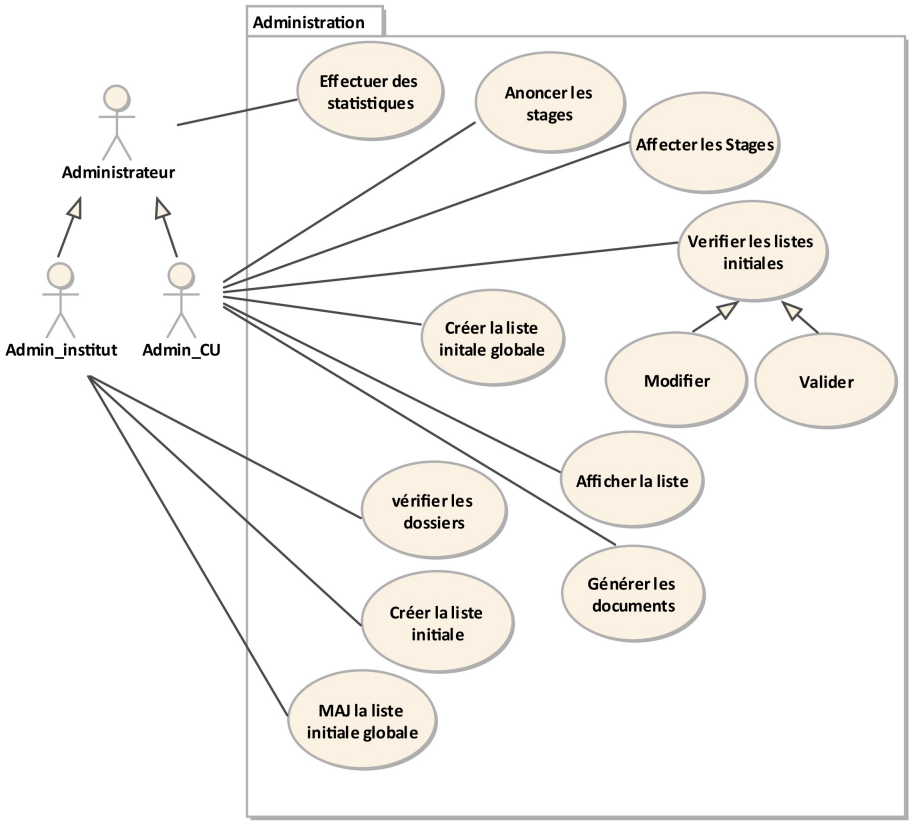

FIGURE 2.3 – Diagramme des cas d'utilisation « Administration ».

2.3.4.3 Diagramme des cas d'utilisation de conseil scientifique

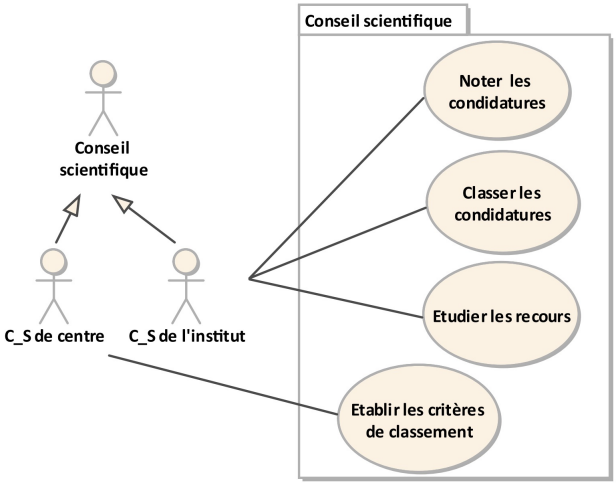

FIGURE 2.4 – Diagramme des cas d'utilisation « conseil scientifique ».

2.3.4.4 Diagramme des cas d'utilisation second rang

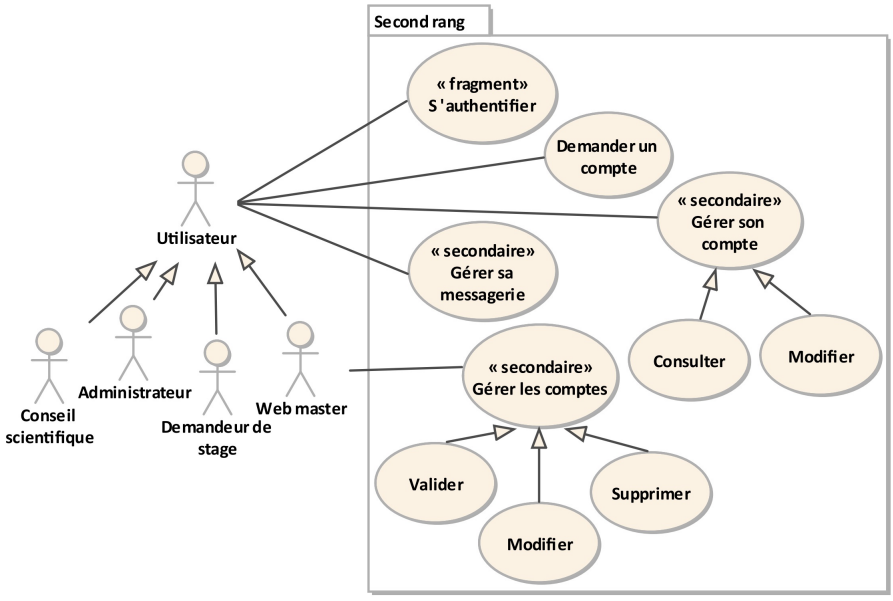

FIGURE 2.5 – Diagramme des cas d'utilisation « second rang ».

2.3.2 Les Maquettes

Nous présentons ci-dessus les maquettes de quelques cas d'utilisation.

2.4.1 La page d'accueil

FIGURE 2.6 – La page d'accueil.

2.4.2 La page d'accueil d'un demandeur de stage

A partir de cette page le demandeur de stage peut consulter et gérer son espace (voir la figure 2.7).

FIGURE 2.7 – La page d'accueil de demandeur de stage.

2.4.3 La page d'authentification

L'écran d'authentification illustré par la figure 2.8, doit apparaitre à chaque fois qu'un utilisateur veut accéder son espace personnel. Afin de réaliser l'accès, l'utilisateur s'identifie par un nom d'utilisateur et un mot de passe.

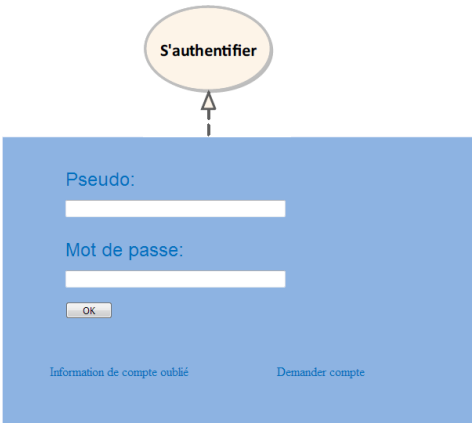

FIGURE 2.8 – La page d'authentification.

2.4.4 La page créer dossier

A partir de cette page le demandeur peut créer un dossier. Pour le faire, il doit remplir toutes les informations générales comme l'indique la figure 2.9.

FIGURE 2.9 – La page créer dossier.

2.4.5 La page ajouter candidature

A partir de cette page le demandeur peut ajouter une candidature. Pour cela, il doit remplir le formulaire de la figure 2.10 qui contient un ensemble d'informations sur le stage.

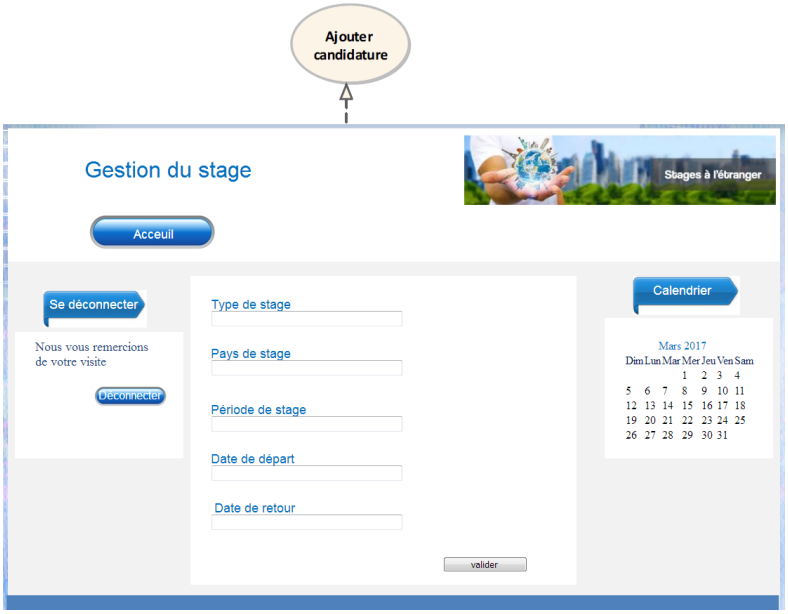

FIGURE 2.10 – La page ajouter candidature.

2.4.6 La page demander un compte

A partir de cette page le demandeur peut demander un compte. Pour cela, il doit remplir le formulaire de la figure 2.11.

FIGURE 2.11 – La page demander un compte.

2.4.7 La page faire un recours

A partir de cette page le demandeur peut faire un recours.

FIGURE 2.12 – La page faire un recours.

2.4.8 La page gérer sa messagerie

A partir de cette page l'utilisateur peut gérer sa messagerie.

FIGURE 2.13 – La page gérer sa messagerie.

2.3.3 Spécification détaillée des exigences

Cette étape consiste détailler les cas d'utilisation à l'aide des descriptions textuelles (fiches types) et les diagrammes de séquence système. Nous détaillons par la suite les cas d'utilisation les plus importants de chaque acteur : Cette étape consiste à faire une description textuelle et à élaborer les diagrammes de séquence système des cas d'utilisation les plus importants de chaque acteur :

2.3.3.1 Les cas d'utilisation de l'acteur « demandeur de stage »

2.3.3.1.1 Cas d'utilisation « gestion de dossier»

a) Cas d'utilisation « Créer dossier » : Nous allons décrire ce cas d'utilisation dans la fiche de description suivante :

Acteur :

Demandeur de stage.

Objectif :

Création de dossier pour accéder au stage.

Pré condition

1. Le demandeur s'est authentifié au site.
2. Le dossier n'a pas encore été crée.

Post condition

Le dossier est crée.

Scénario nominal

1. Le système affiche le formulaire de création.
2. Le demandeur rempli les informations puis valider.
3. Le système vérifie les informations.
4. Le système enregistre les informations.
5. Le système confirme la création de dossier.

Scénario alternatif :

3a. Le Système détecte que certaines informations sont incomplètes ou erronées.

1. Le Système propose au demandeur de modifier ou compléter les informations, reprendre le scénario nominal au point 2.

FIGURE 2.14 – Fiche type « créer dossier ».

– **Diagramme de séquence « créer dossier » :** Le diagramme de la figure suivante montre les interactions entre l'acteur demandeur de stage et le système au moment de la création d'un dossier.

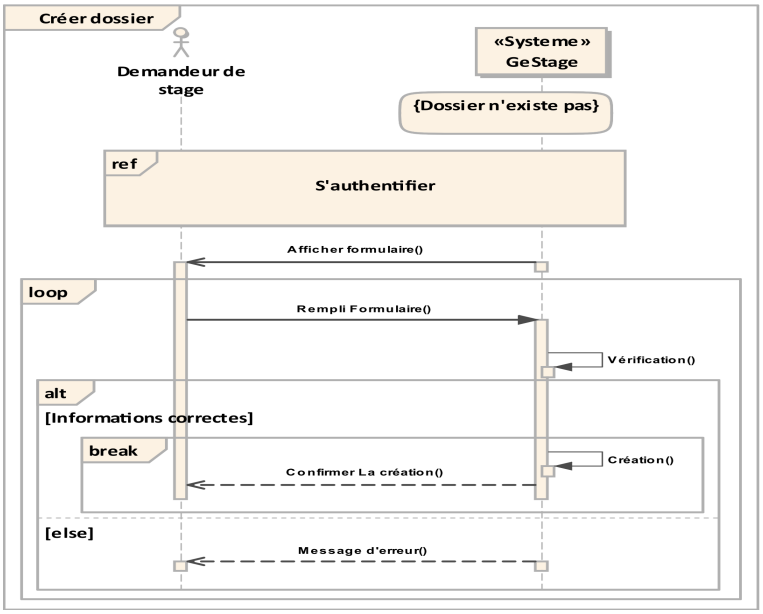

FIGURE 2.15 – Diagramme de séquence « créer dossier ».

b) Cas d'utilisation « modifier dossier » : Nous allons décrire ce cas d'utilisation dans la fiche de description suivante :

Acteur :
Demandeur de stage.
Objectif :
Modifier le dossier.
Pré condition
Le demandeur s'est authentifié et le dossier existe.
Post condition
Le dossier est modifié.
Scénario nominal
1. Le système affiche le formulaire de modification.
2. Le demandeurmodifiele formulaire.
3. Le système vérifie les informations.
4. Le système enregistre les informations.
5. Le système confirme la modification de dossier.

Scénario alternatif :
3a. Le Système détecte que certaines informations sont incomplètes ou erronées.
1. Le Système propose au demandeur de modifier ou compléter les informations, reprendre le scénario nominal au point 2.

FIGURE 2.16 – Fiche type « modifier dossier ».

– **Diagramme de séquence « modifier dossier » :** Le diagramme de la figure suivante montre les interactions entre l'acteur demandeur de stage et le système au

moment de la modification d'un dossier.

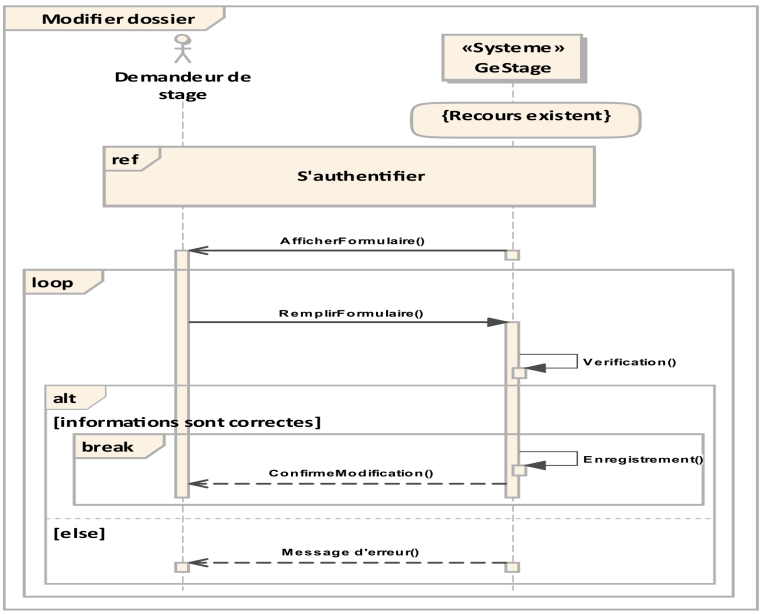

FIGURE 2.17 – Diagramme de séquence « modifier dossier ».

2.3.3.1.2 Cas d'utilisation « Ajouter candidature » : Nous allons décrire ce cas d'utilisation dans la fiche de description suivante :

Acteur :

Demandeur de stage.

Objectif :

Ajouter des candidatures pour demander un stage.

Pré condition

1. Le demandeur s'est authentifié au site.
2. Le dossier existe.

Post condition

La candidature est ajoutée.

Scénario nominal

1. Le système affiche le formulaire d'ajout.
2. Le demandeur rempli les informations puis valider.
3. Le système vérifie les informations.
4. Le système enregistre les informations.
5. Le système confirme l'ajout de candidature.

Scénario alternatif :

3a. Le Système détecte que certaines informations sont incomplètes ou erronées.

1. Le Système propose au demandeur de modifier ou compléter les

FIGURE 2.18 – Fiche type « Ajouter candidature ».

– **Diagramme de séquence « Ajouter candidature » :** Le diagramme de la figure suivante montre les interactions entre l'acteur demandeur de stage et le système au moment de l'ajout d'un dossier.

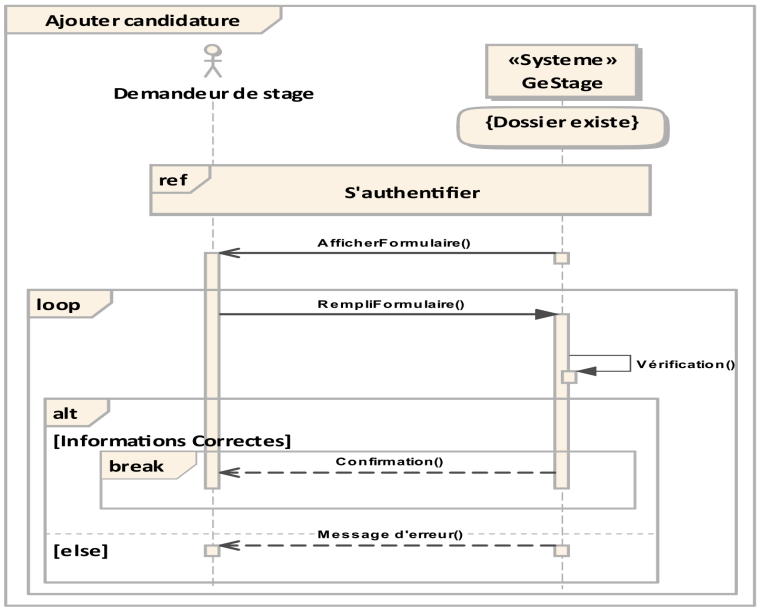

FIGURE 2.19 – Diagramme de séquence « Ajouter candidature ».

2.3.3.1.3 Cas d'utilisation « consulter la liste initiale globale » : Nous allons décrire ce cas d'utilisation dans la fiche de description suivante :

Acteur :
Demandeur de stage.
Objectif :
Consulter la liste initiale globale des candidatures acceptées.
Pré condition
1. Le demandeur de stage s'est authentifié.
2. La liste des candidatures acceptées existe.

Post condition
/
Scénario nominal
1. Le système affiche la liste initiale globale des candidatures acceptées.
2. Le demandeur de stage vérifie la liste.

FIGURE 2.20 – Fiche type « consulter la liste initiale globale ».

– **Diagramme de séquence « consulter la liste initiale globale » :** Le diagramme de la figure suivante montre les interactions entre l'acteur demandeur de stage et le système au moment de la consultation de la liste des candidatures acceptées.

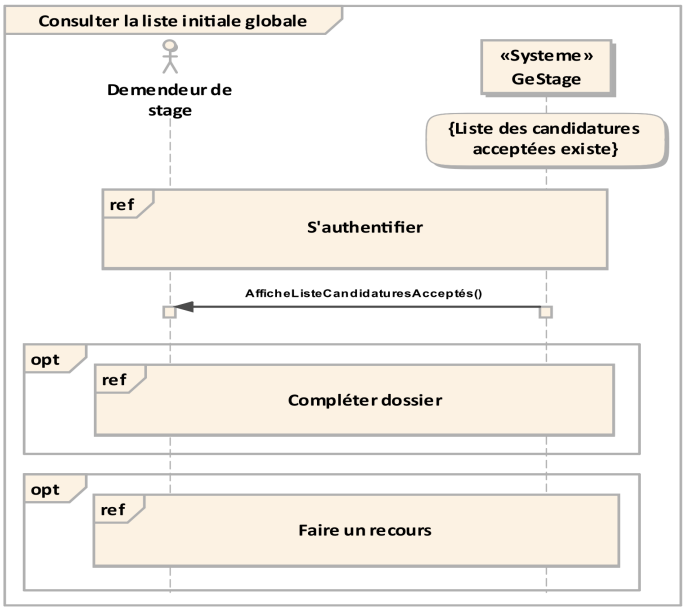

FIGURE 2.21 – Diagramme de séquence « consulter la liste initiale globale ».

2.3.3.2 Les cas d'utilisation de l'acteur « administration »

2.3.3.2.1 Cas d'utilisation « Créer la liste initiale » : la fiche de description suivante présente la description de ce cas d'utilisation.

Acteur :
Admin_ institut.

Objectif :
Créer la liste initiale.

Pré condition
Les candidatures sont classées par le conseil scientifique.

Post condition
La liste des candidatures acceptés est crée.

Scénario nominal

1. Le système affiche le formulaire.
2. L'administrateur de l'institut saisi les informations puis valide.
3. Le système crée la liste.
4. Le système affiche message de confirmation.

FIGURE 2.22 – Fiche type « créer la liste initiale ».

– **Diagramme de séquence « créer la liste initiale » :** : Le diagramme de la figure suivante montre les interactions entre l'acteur administrateur de l'institut et le système au moment de la création de la liste initiale des candidatures acceptées (voire la figure 2.23) :

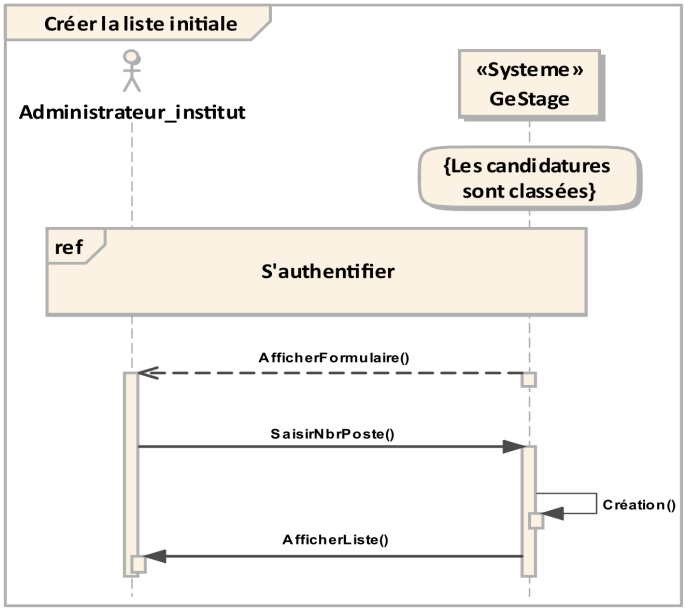

FIGURE 2.23 – Diagramme de séquence « Créer la liste initiale ».

2.3.3.2.2 Cas d'utilisation « mettre à jour la liste initiale globale » : la description textuelle de ce cas d'utilisation sera présentée dans la fiche de description suivante :

Acteur :
Admin_ institut.
Objectif :
Modifier la liste initiale globale selon le résultat de l'étude des recours.
Pré condition :
Les recours sont étudiés par le conseil scientifique de l'institut.
Post condition :
Nouvelle liste.
Scénario nominal :
1. L'administrateur de l'institut demande le résultat de l'étude des recours et la liste initiale globale.
2. Le système affiche le résultat et la liste.
3. L'administrateur de l'institut modifie la liste selon le résultat de l'étude des recours puis confirme.
4. Le système vérifie les modifications puis enregistrer.
5. Le système affiche la réussite de modification.
Scénario alternatif :
3a. Le système détecte que les modifications sont incomplètes ou erronées.
1. Le Système propose à l'administrateur de modifier ou compléter les informations, reprendre le scénario nominal au point 3.

FIGURE 2.24 – Fiche type « Mettre a jour liste initial globale ».

– **Diagramme de séquence « Mettre a jour liste initial globale » :**Le diagramme de la figure suivante montre les interactions entre l'acteur administrateur de l'institut et le système au moment de la mise à jour de la liste initiale globale des candidatures acceptées (voir la figure 2.25) :

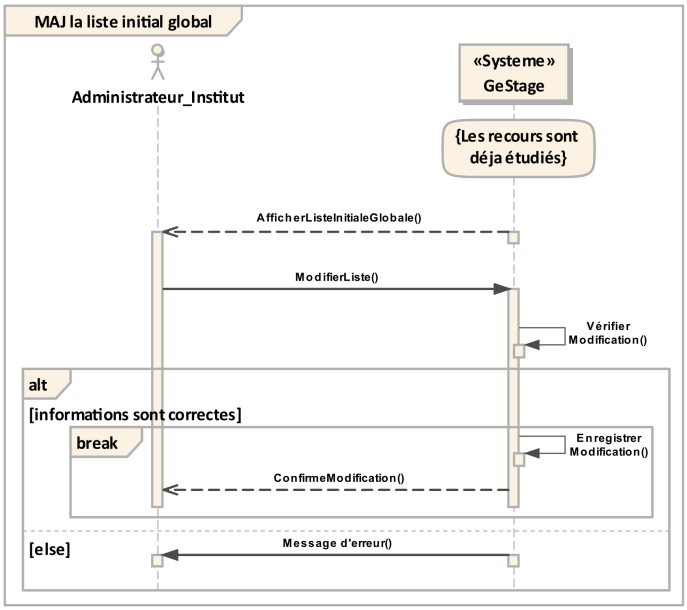

FIGURE 2.25 – Diagramme de séquence « mettre à jour la liste initiale globale ».

2.3.3.3 Les cas d'utilisation de l'acteur « conseil scientifique »

2.3.3.3.1 Cas d'utilisation « Etudier les recours » : la fiche de description suivante présente la description de cas d'utilisation étudier les recours :

Acteur :
C_S de l'institut.
Objectif :
Consulter et répondre aux recours.
Pré condition

1. Liste des recours non vide.
2. Le conseil scientifique de l'institut s'est authentifié

Post condition
Les recours sont étudiés.
Scénario nominal

1. Le système affiche les recours.
2. Le conseil scientifique sélectionne un recours.
3. Le système affiche une description du recours.
4. Le conseil scientifique répond au recours.

FIGURE 2.26 – Fiche type « Etudier les recours ».

– **Diagramme de séquence « Etudier les recours » :** Le diagramme de la figure suivante montre les interactions entre l'acteur conseil scientifique de l'institut et le système au moment de l'étude des recours :

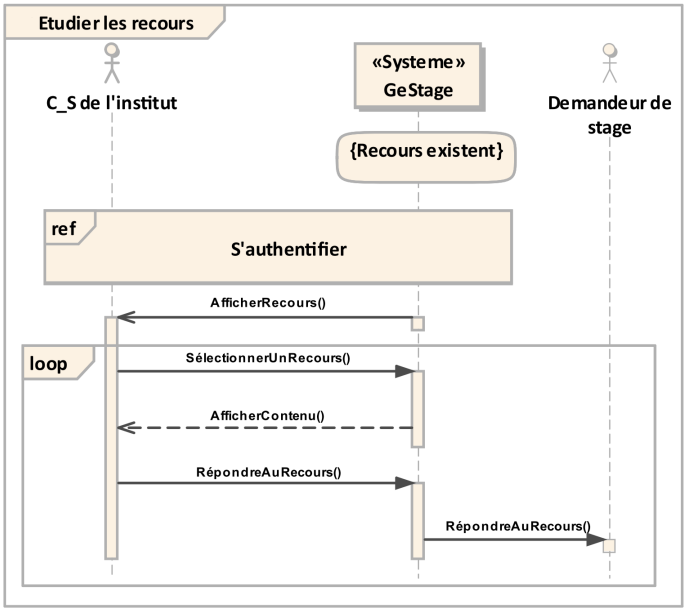

FIGURE 2.27 – Diagramme de séquence « Etudier les recours ».

2.3.3.3.2 Cas d'utilisation « noter les candidatures » : Le diagramme de la figure suivante montre les interactions entre l'acteur conseil scientifique de l'institut et le système au moment de l'étude et la notation de candidature.

Acteur :

C_S de l'institut.

Objectif :

Noter les candidatures des demandeurs.

Pré condition

Conseil scientifique de l'institut s'est authentifié et les candidatures déjà existent.

Scénario nominal

1. Le système affiche les candidatures.
2. Le conseil scientifique sélectionne une candidature.
3. Le système affiche le contenu de dossier.
4. Le conseil scientifique étudie le contenu du dossier selon les critères d'acceptation.
5. Le conseil scientifique donne une note à la candidature.

FIGURE 2.28 – Fiche type « noter les candidatures ».

– **Diagramme de séquence « noter les candidatures » :** : Ce diagramme (voire

la figure 2.29) montre l'interaction entre le conseil scientifique et le système dans l'étude et la notation des candidatures.

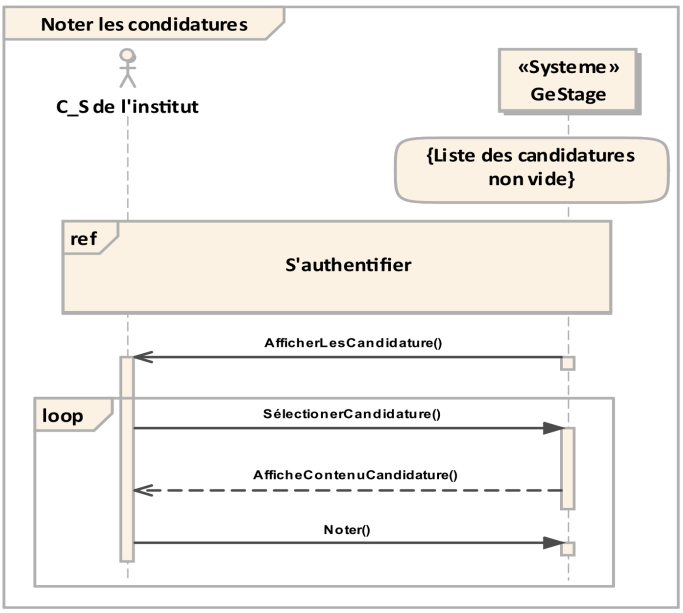

FIGURE 2.29 – Diagramme de séquence « noter les candidatures ».

2.3.3.4 Les cas d'utilisation de l'acteur « Utilisateur »

2.3.3.4.1 Cas d'utilisation « s'authentifier » : Le diagramme de la figure suivante montre les interactions entre l'acteur utilisateur et le système au moment de l'authentification.

Acteur :
Utilisateur.
Objectif :
Vérifier l'autorisation d'accès aux comptes.
Pré condition :
Compte existe.
Post condition :
L'utilisateur s'est authentifié.
Scénario nominal :
1. Le système affiche les formulaires de nom et de mot de passe.
2. L'utilisateur saisit le nom et le mot de passe.
3. Le système vérifie la validité du nom et du mot de passe
4. Le système affiche le compte.

Scénario alternatif :
4a. Le système détecte que le nom ou le mot de passe est erroné.
1. Le Système propose à l'utilisateur une nouvelle fois de saisir le nom et le mot de passe, reprendre le scénario nominal au point 2.

FIGURE 2.30 – Fiche type « s'authentifier ».

– **Diagramme de séquence « s'authentifier » :** Ce diagramme montre l'interaction de l'authentification entre l'utilisateur et le système.

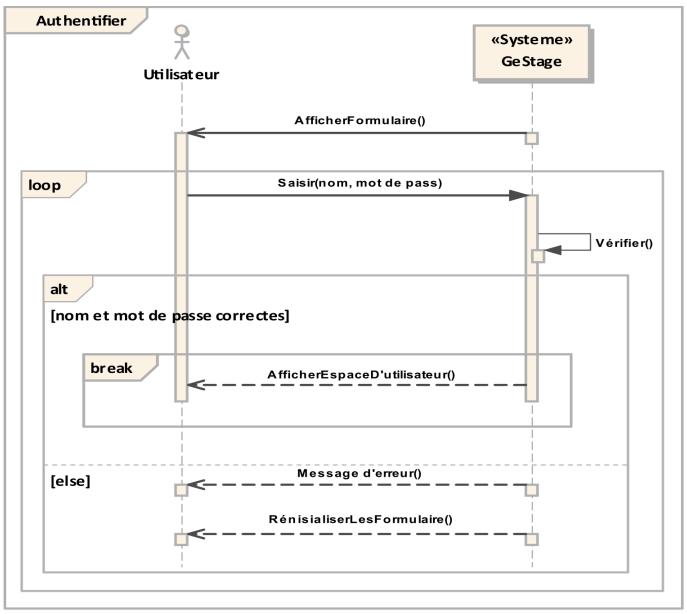

FIGURE 2.31 – Diagramme de séquence « s'authentifier ».

2.3.3.4.2 Cas d'utilisation « demander un compte » : : la description textuelle de ce cas d'utilisation sera présentée dans la fiche de description suivante

Acteur :
Utilisateur.
Objectif :
Demander un compte.
Pré condition :
Compte n'existe pas.
Post condition :
Le compte est crée.
Scénario nominal :
1. Le système affiche le formulaire nécessaire.
2. L'utilisateur saisi les informations puis envoyer.
3. Le système vérifie les informations.
4. Le système confirme la demande d'un compte.

Scénario alternatif :
3a. Le Système détecte que les informations sont incomplètes ou erronées.
1. Le système demande au web master de modifier ou compléter les informations, reprendre le scénario nominal au point 2.

FIGURE 2.32 – Fiche type « demander un compte ».

– **Diagramme de séquence « demander un compte » :** Le diagramme de la figure suivante montre les interactions entre l'acteur utilisateur et le système au moment de la demande de compte.

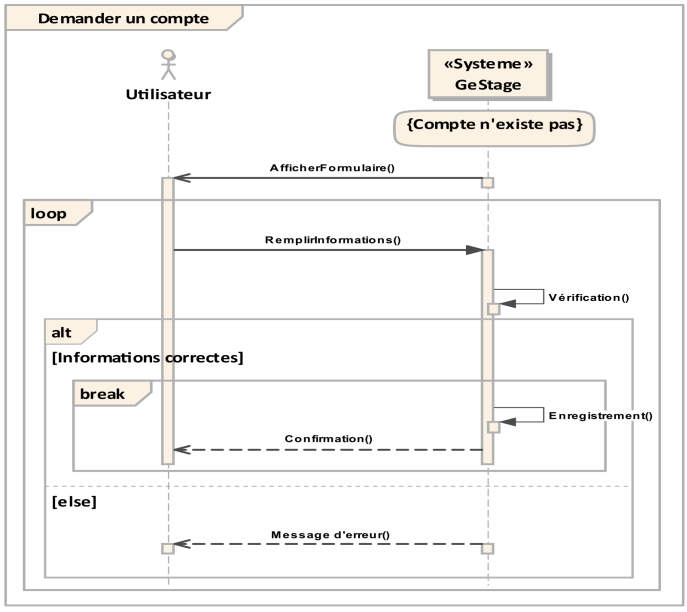

FIGURE 2.33 – Diagramme de séquence « Demander un compte ».

2.4 Conclusion

Ce chapitre a été consacré à l'étude préliminaire de notre système qui a été réalisée suivant la première étape du processus simplifié présenté dans le chapitre précédent. Cette étude nous a permis de spécifier les besoins utilisateur via les diagrammes de cas utilisation et élaborer les IHMs qui montrent l'aspect visuel de notre application. Ces besoins ont été détaillés à l'aide des fiches types et des diagrammes de séquence.

CHAPITRE 3

ANALYSE ET CONCEPTION

3.1 Introduction

Ce chapitre est consacré à la phase d'analyse et celle de conception. Nous commençons par l'analyse qui comprend la réalisation des cas d'utilisation, la présentation des diagrammes d'état de certaines classes, et la modélisation de la navigation. Par la suite, nous passons à la phase de conception qui comprend les diagrammes d'interaction et le diagramme de classe de conception.

3.2 Phase d'analyse

Nous débuterons cette phase par la réalisation d'un digramme de classe participante. Ensuite un diagramme de navigation et enfin un diagramme d'état qui offre une vision complète de l'ensemble des comportements d'un seul élément.

3.2.1 Réalisation des cas d'utilisation

La réalisation de nos cas d'utilisation s'effectue selon les étapes suivantes : 1) identification des concepts du domaine, 2) ajout des attributs et des associations, 3) construction des diagrammes de classes participantes.

3.2.1.1 Identification des concepts du domaine

Nous allons prendre les cas d'utilisation les plus importants, puis identifie les concepts métier

a) Pour le cas d'utilisation « ajouter candidature », nous identifions les concepts fondamentaux suivants :

- Candidature.
- Demandeur de stage.
- Stage.
- Liste des candidatures.

b) Pour le cas d'utilisation « classer les candidatures », nous identifions :

– Liste de candidatures.

– Candidature.

– Demandeur de stage.

c) Pour le cas d'utilisation « étudier recours », nous identifions :

– Recours.

– Demandeur de stage.

d) Pour le cas d'utilisation « demander compte », nous identifions :

– Utilisateur.

– Demandeur de stage.

– Administrateur.

– Conseil scientifique.

3.2.1.2. Ajout des associations et des attributs

Après l'identification des concepts métiers des cas d'utilisation, nous ajoutons des attributs et des associations, et nous les illustrons par une présentation graphique.

a) Ajouter candidature :

Un demandeur présente sa candidature pour les stages désirés.

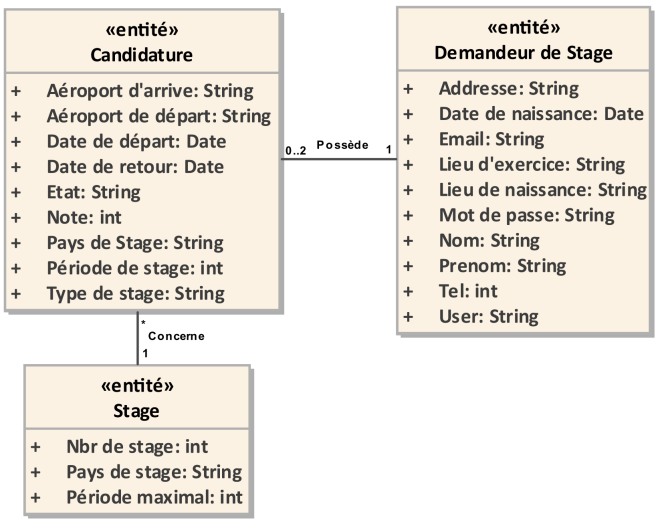

FIGURE 3.1 – Concepts liés à l'ajout de candidature.

b) Classer les candidatures : Le conseil scientifique classe dans une liste les candidatures présentées par les demandeurs.

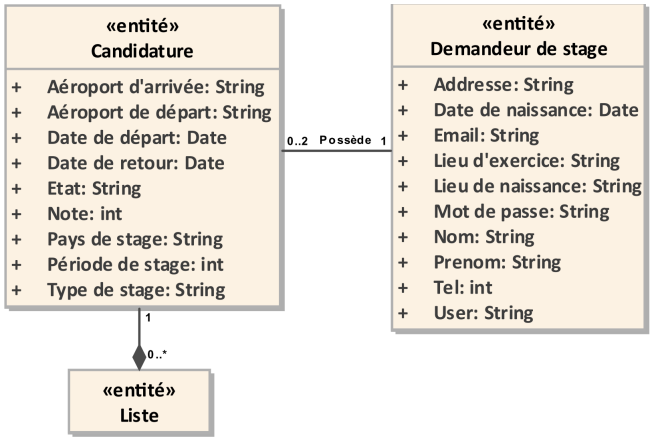

FIGURE 3.2 – Concepts liés au classement des candidatures.

c) Etudier les recours : Le conseil scientifique de l'institut étudie les recours faits par les demandeurs.

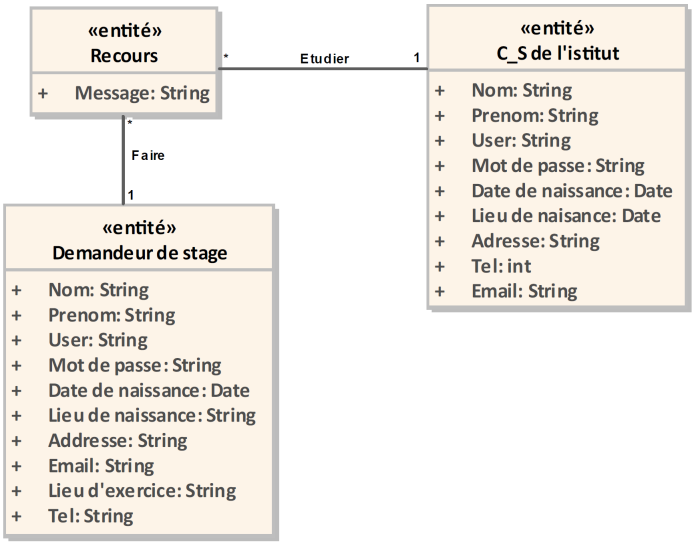

FIGURE 3.3 – Concepts liés à l'étude des recours.

d) Demander compte : Si l'utilisateur n'a pas de compte, il demander à ouvrir un.

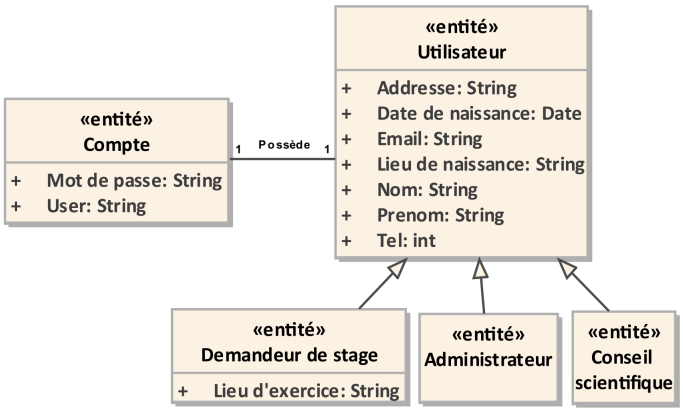

FIGURE 3.4 – Concepts liés à la demande d'un compte.

3.2.1.3 Classes d'analyse participantes des cas d'utilisation

Cette étape consiste à élaborer les diagrammes de classes participantes des cas d'utilisation les plus importants de chaque acteur :

3.2.1.3.1 les diagrammes de classes participantes de l'acteur « demandeur de stage » :

a) Créer dossier

Le demandeur veut créer un dossier : Il entre ses informations puis crée. Supposons que nous avons un seul écran, avec un seul contrôle et un dialogue (voir la figure 3.5).

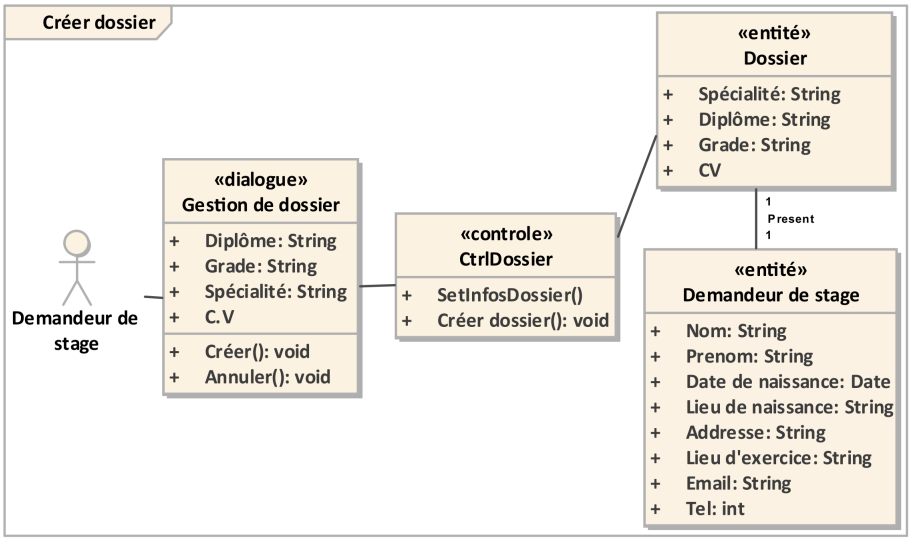

FIGURE 3.5 – DCP créer dossier.

b) Ajouter candidature Si le demandeur veut ajouter une candidature : Il entre ses informations puis valide.

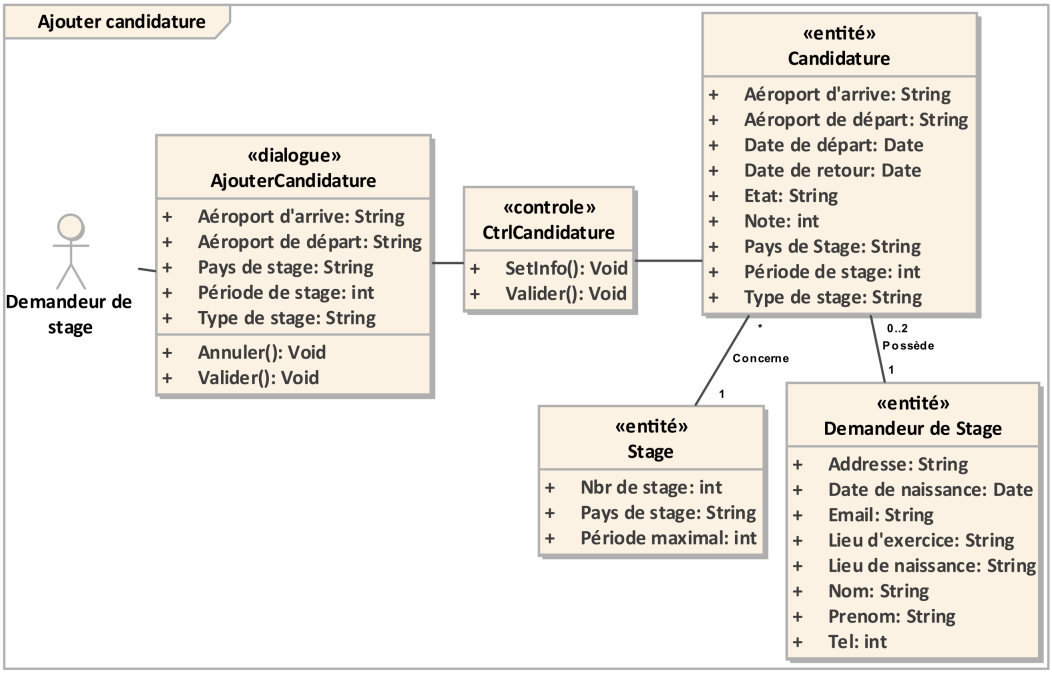

FIGURE 3.6 – DCP ajouter candidature.

3.2.1.3.2 les diagrammes de classes participantes de l'acteur « Administrateur »

a) Créer la liste initiale

L'administrateur de l'institut veut créer une liste initiale des candidatures acceptées. Supposons que la maquette nous montre deux écrans :

- Le premier où l'on saisie le nombre de poste maximal.
- Le second affiche la liste créé. (voire le figure 3.7)

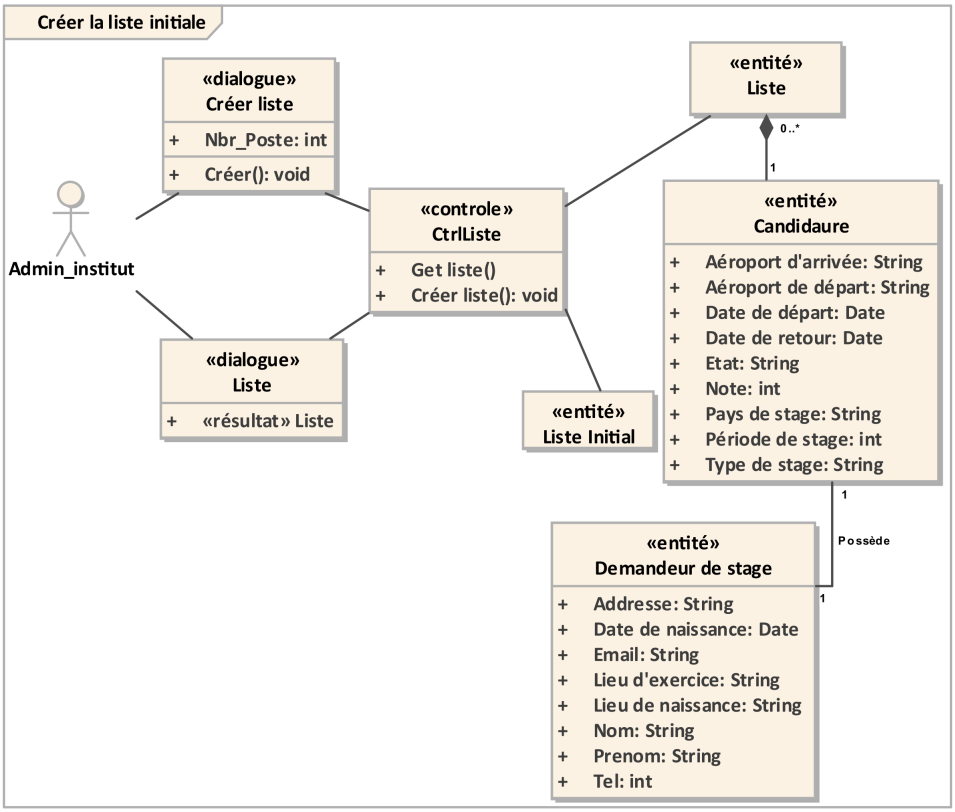

FIGURE 3.7 – DCP créer la liste initiale.

3.2.1.3.3 les diagrammes de classes participantes de l'acteur « conseil scientifique »

a) Etudier les recours

Le conseil scientifique étudie les recours des demandeurs et leur répond. Supposons que la maquette nous montrer deux écrans :

- Le premier où l'on sélectionne un recours.
- Le deuxième qui affiche le contenu de recours pour la réponse (voire la figure 3.8).

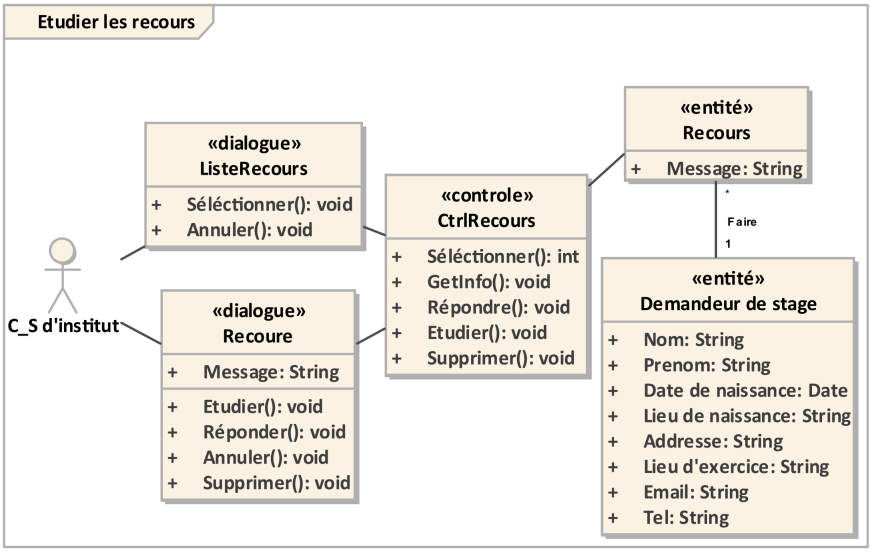

FIGURE 3.8 – DCP étudier les recours.

b) Noter les candidatures :

Le conseil scientifique de l'institut va noter les candidatures. Il les étudie d'abord puis les note. Supposons que la maquette nous montre deux écrans :

- Un écran où il sélectionne une candidature.
- Un deuxième qui affiche les informations de dossier pour la noter (voire la figure 3.9).

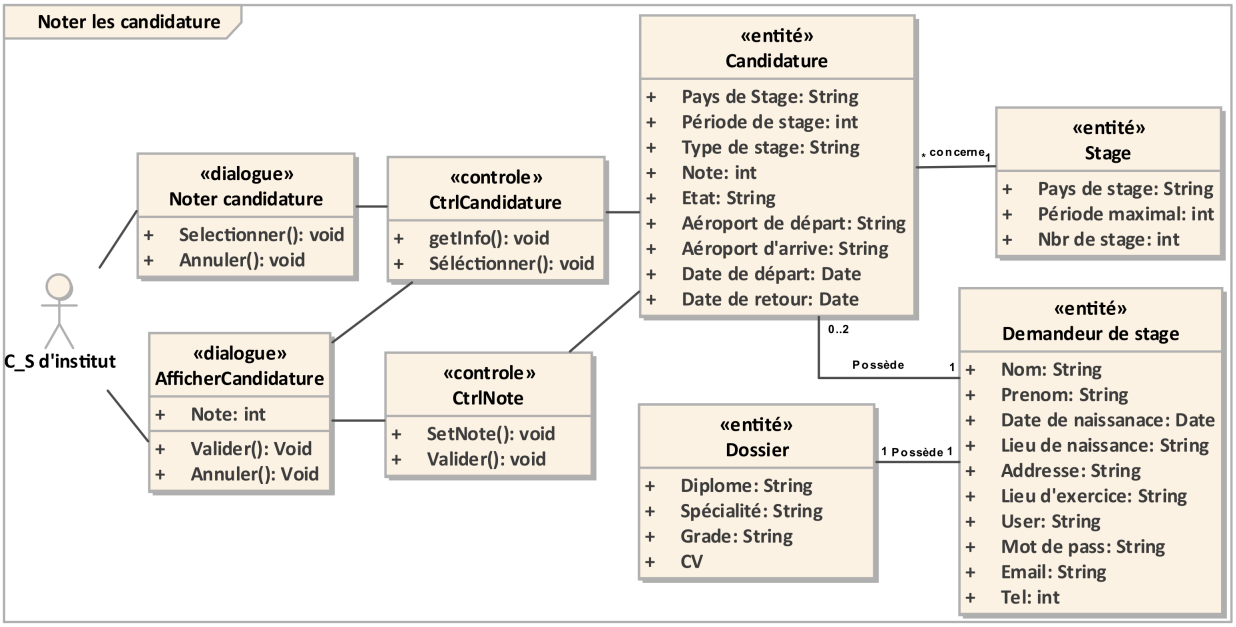

FIGURE 3.9 – DCP noter les candidatures.

3.2.1.3.4 les diagrammes de classes participantes de l'acteur «Utilisateur»

a) S'authentifier :

Un utilisateur a besoin de s'identifier pour accéder à son propre compte. Supposons que la maquette nous montre deux écrans :

- Un écran où l'on entre le nom d'utilisateur et le mot de passe.
- Un deuxième qui nous donne la page demandée (voire la figure 3.10).

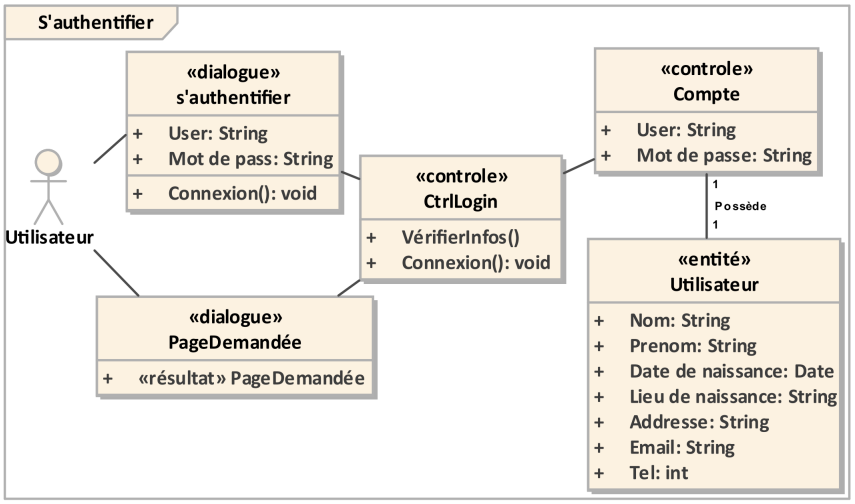

FIGURE 3.10 – DCP s'authentifier.

b) Demander un compte :

L'utilisateur demande un compte. Supposons que nous avons un seul écran, avec un seul contrôle et un dialogue (voire la figure 3.11).

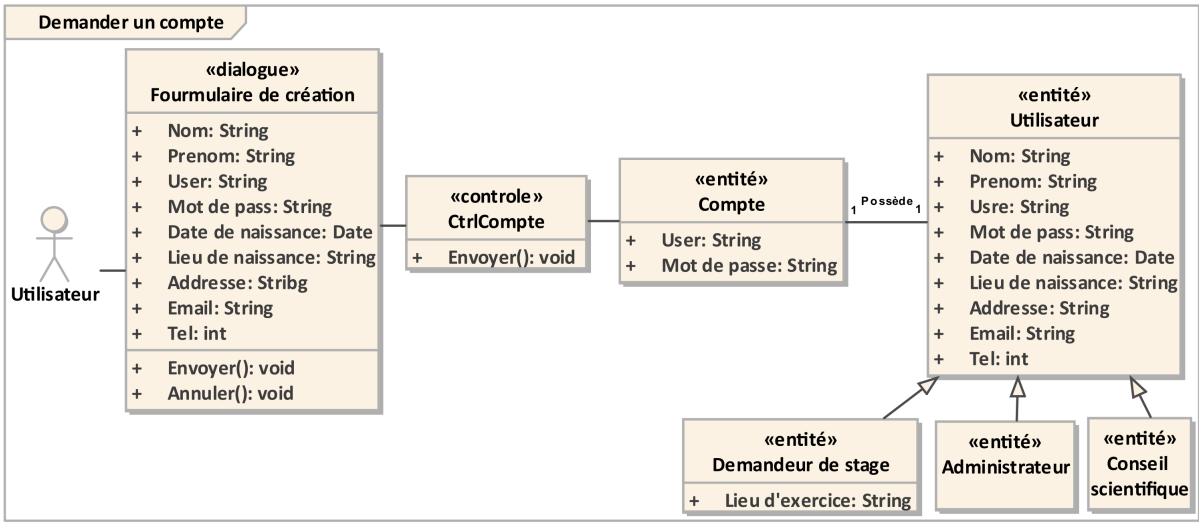

FIGURE 3.11 – DCP Demander compte.

3.2.2 Diagramme d'états

Nous nous contentons ici de présenter le diagramme d'états d'un seul élément du système qui est la classe candidature. L'ensemble des comportements d'une instance de cette classe se définit par :

- L'ajout d'une candidature mène à l'état *ajouté*.
- Lorsque la candidature est étudiée, on passe à l'état *étudié*.
- Après l'étude de celle-ci, on met la classe dans une liste, l'acceptation de cette candidature dépendra de son classement : si celui-ci est inférieure ou égale au nombre de postes offerts, on passe à l'état *accepté*, sinon, on passe à l'état *refusé*.

Le diagramme d'état est représenté sur la figure 3.12.

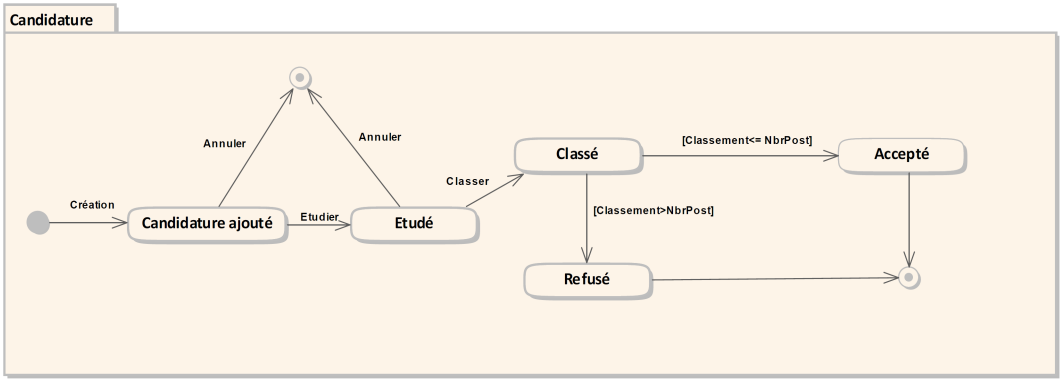

FIGURE 3.12 – Diagramme d'états de candidature.

3.2.3 Modélisation de la navigation

La navigation de l'application par les acteurs sera complètement différente entre chacun d'eux, Car elle n'accède pas du tout aux mêmes fonctionnalités, comme nous l'a confirmé l'analyse des acteurs et des cas d'utilisation effectuées au chapitre précédant. La modélisation de la navigation peut donc se structurer tout d'abord par acteur. Le diagramme de navigation de chaque acteur est représenté sur les figures suivantes.

3.2.3.1 Navigation du demandeur de stage

Nous illustrons la navigation de quelques cas d'utilisation :

3.2.3.1.1 Créer dossier

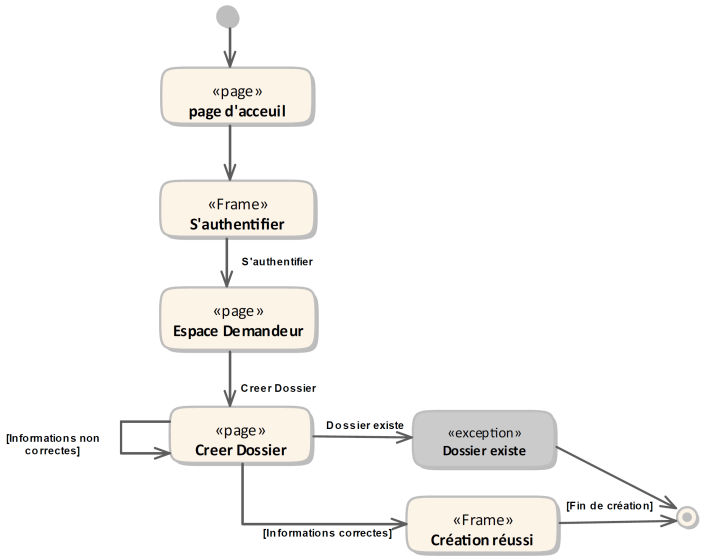

FIGURE 3.13 – Diagramme de navigation de la création du dossier.

3.2.3.1.2 Ajouter candidatures

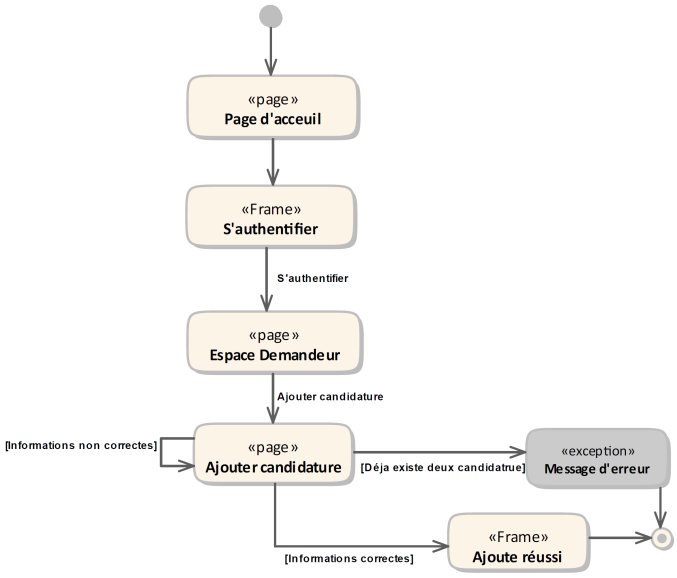

FIGURE 3.14 – Diagramme de navigation pour ajouter une candidature.

3.2.3.1.3 Consulter la liste initiale

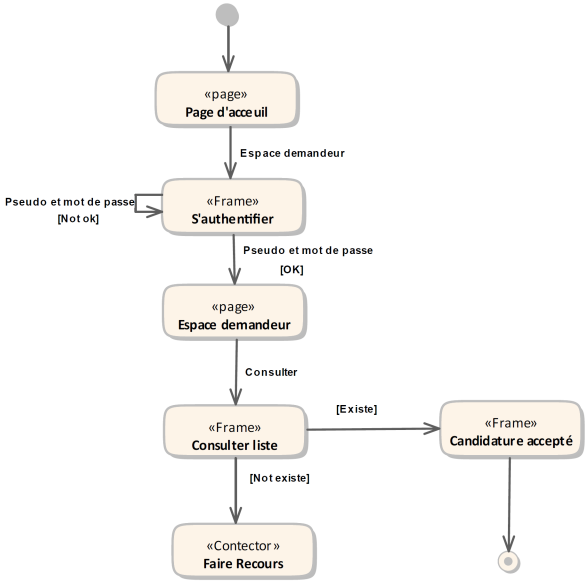

FIGURE 3.15 – Diagramme de navigation de la consultation de la liste initiale.

3.2.3.1.4 La navigation globale du demandeur

Maintenant nous réalisons un diagramme global simplifié de navigation du demandeur. Il va comprendre l'ensemble des pages, frames et actions principales des cas d'utilisation :

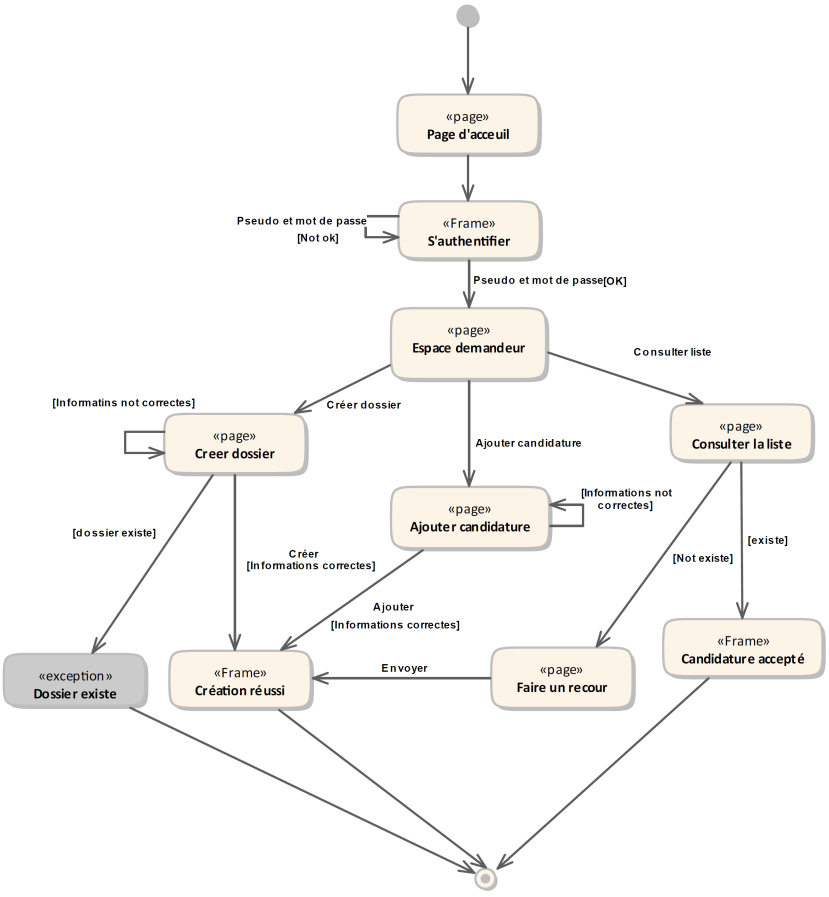

FIGURE 3.16 – Diagramme globale de la navigation du demandeur.

3.2.3.2 Navigation de l'administrateur

Nous réalisons un diagramme global simplifié de navigation du demandeur. Il va comprendre l'ensemble des pages, frames et actions principales des cas d'utilisation :

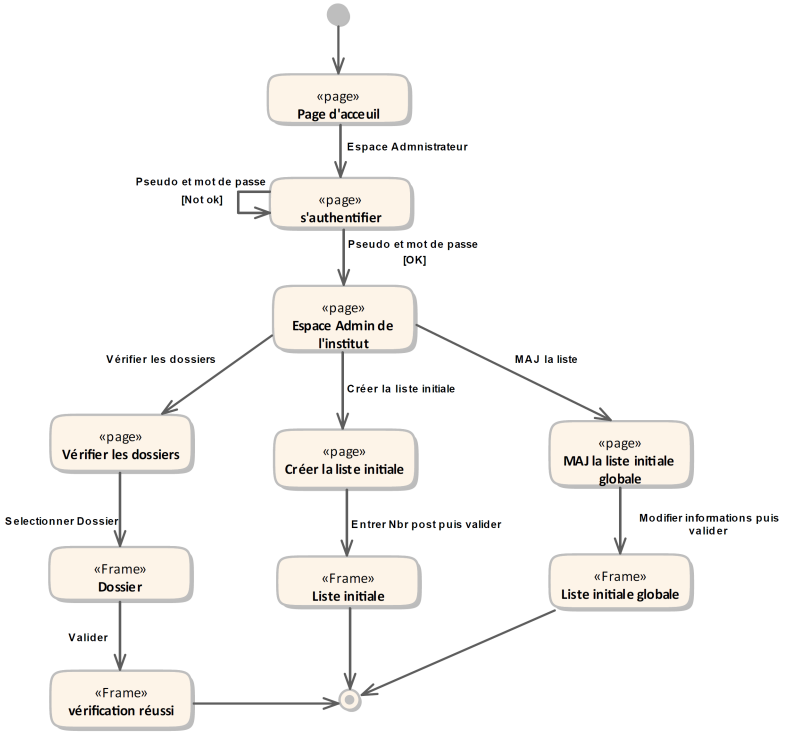

FIGURE 3.17 – Diagramme globale de la navigation de l'administrateur.

3.2.3.3 Navigation du conseil scientifique

3.2.3.3.1 Noter candidatures

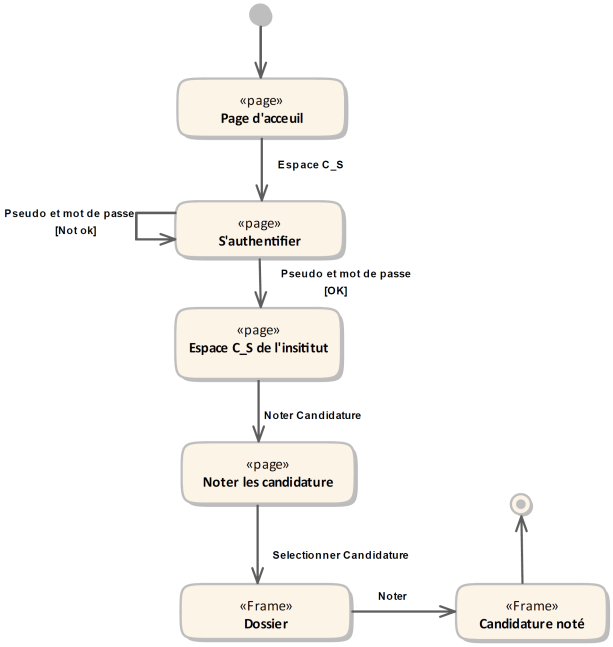

FIGURE 3.18 – Diagramme de navigation pour noter les candidatures.

3.2.3.3.2 La navigation globale du conseil scientifique

Nous réalisons un diagramme global simplifié de navigation du conseil scientifique. Il va comprendre l'ensemble des pages, frames et actions principales des cas d'utilisation :

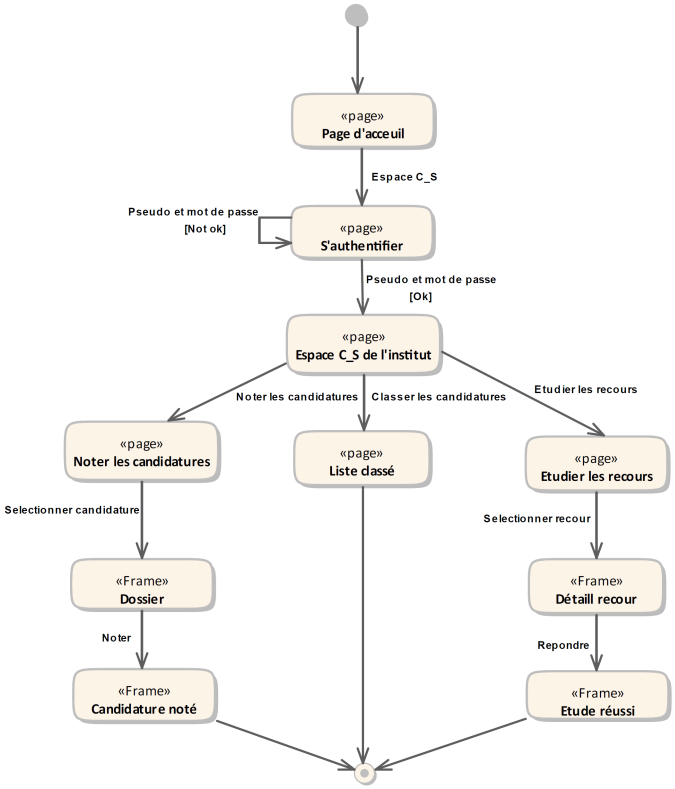

FIGURE 3.19 – Diagramme globale de la navigation du conseil scientifique.

3.3 Phase de conception

Nous débuterons cette conception objet par une réalisation d'un digramme d'interaction et enfin un diagramme de classes de conception.

3.3.1 Digramme d'interaction

Nous représentons les diagrammes d'interaction des cas d'utilisations les plus importants. Nous nous intéressons ici à l'interaction entre chaque acteur et les trois types de classes qui composent le système.

3.3.1.1 Diagramme d'interaction créer dossier

Nous présentons successivement l'interaction entre le demandeur de stage et le système qui contient un écran de création, contrôle et entité dossier.

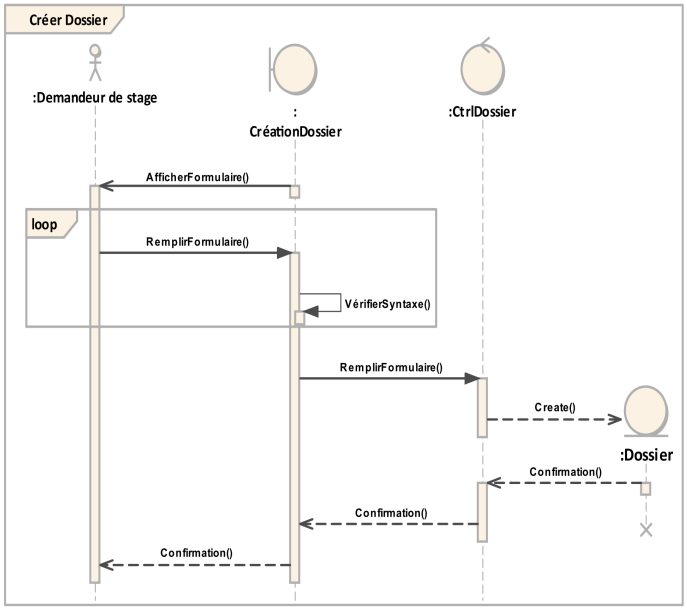

FIGURE 3.20 – Diagramme d'interaction « créer dossier ».

3.3.1.2 Diagramme d'interaction ajouter candidature

Nous présentons successivement l'interaction entre l'administrateur de l'institut et le système qui contient un écran d'ajout de candidature, contrôle candidature et deux entités candidature et liste.

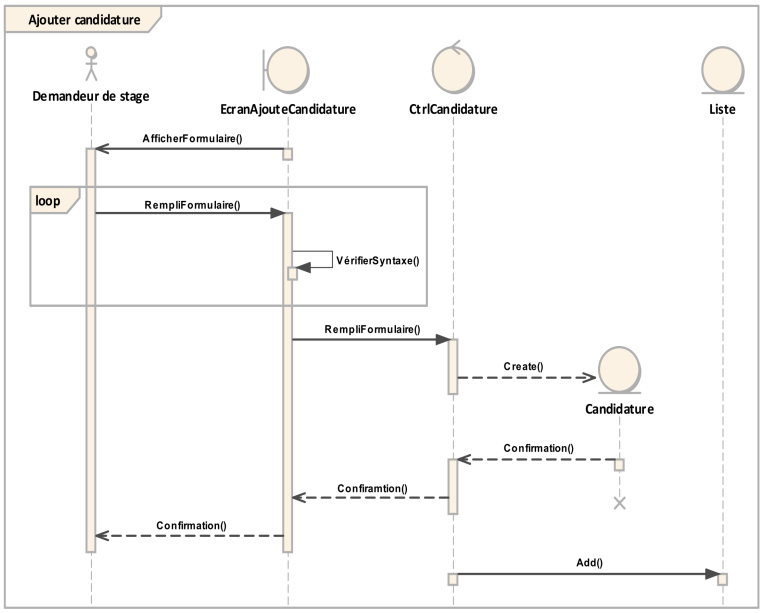

FIGURE 3.21 – Diagramme d'interaction « ajouter candidature ».

3.3.1.3 Diagramme d'interaction créer la liste initiale

Nous présentons successivement l'interaction entre l'administrateur de l'institut et le système qui contient un écran de création liste, contrôle liste et deux entités liste et liste initiale.

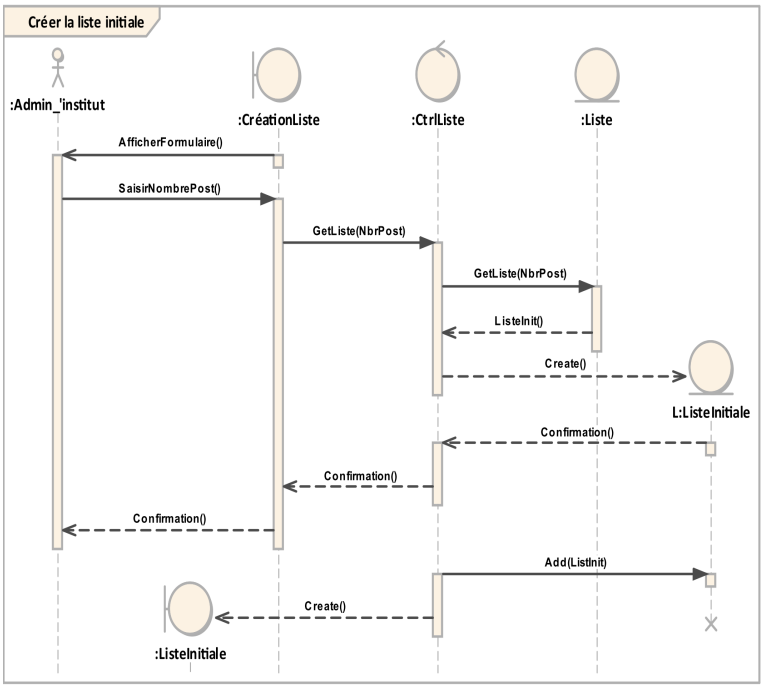

FIGURE 3.22 – Diagramme d'interaction « créer la liste initiale ».

3.3.1.4 Diagramme d'interaction noter les candidatures

Dans ce diagramme nous présentons successivement l'interaction entre le conseil scientifique de l'institut et le système.

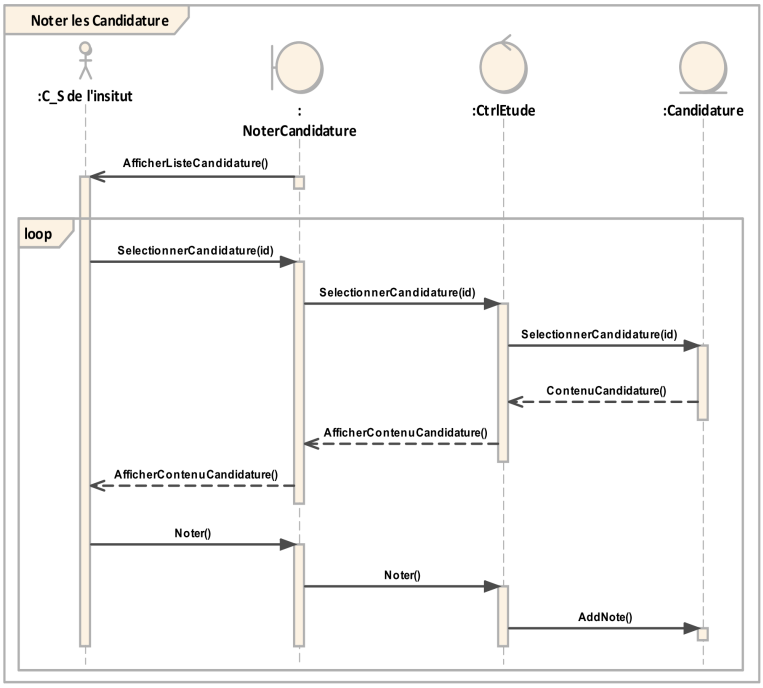

FIGURE 3.23 – Diagramme d'interaction « noter les candidatures ».

3.3.2 Diagramme de classe de conception

Le diagramme de classe est donné par la figure ci-dessous.

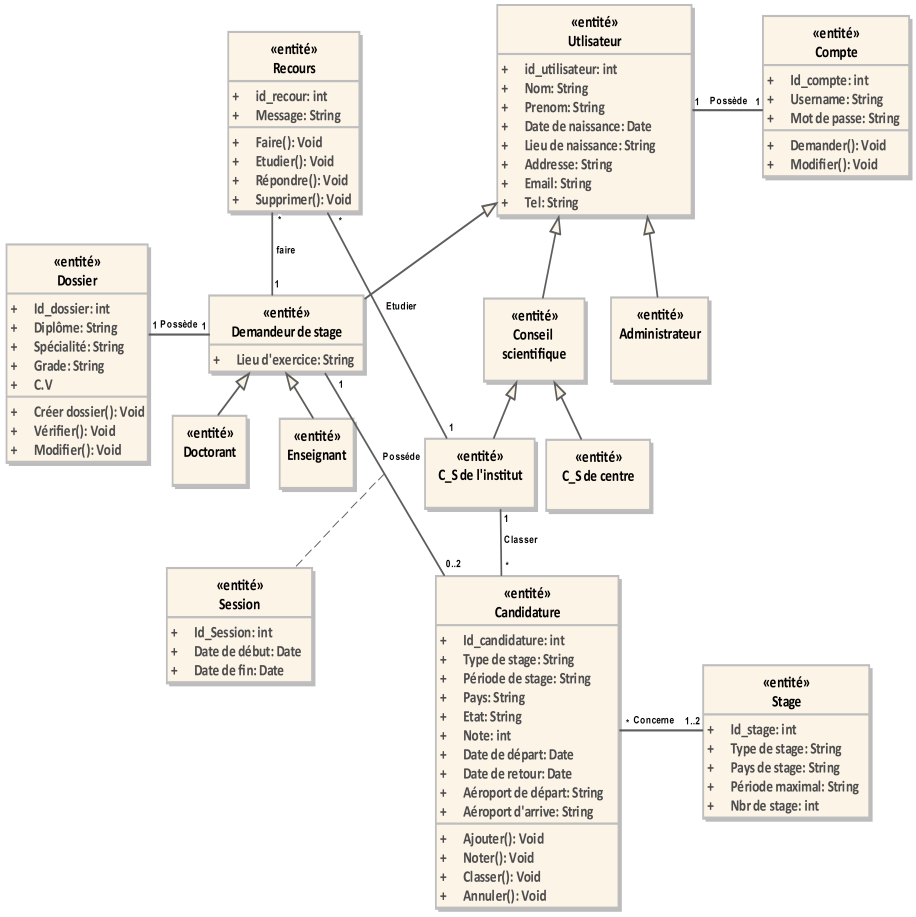

FIGURE 3.24 – Diagramme de classe de conception.

3.4 Conclusion

Dans ce chapitre, nous avons effectué la conception de notre système qui s'est traduit par le diagramme d'interaction et le diagramme de classe participante. Tout cela a été réalisé selon les différentes étapes du processus simplifié que nous avons présenté dans le premier chapitre.

CHAPITRE 4

RÉALISATION

4.1 Introduction

Ce chapitre est consacré à la réalisation de notre application qui s'appuie sur la modélisation présentée dans les chapitres précédents. Nous allons présenter les choix techniques et la base de données. Enfin nous présenterons les principales interfaces de l'application.

4.2 Choix techniques

4.2.1 JEE

Java Enterprise Edition, ou Java EE (anciennement J2EE), est une spécification pour la technique Java d'Oracle plus particulièrement destinée aux applications d'entreprise. Ces applications sont considérées dans une approche multi-niveaux. Dans ce but, toutes les implémentations de cette spécification contiennent un ensemble d'extensions au framework Java standard (JSE, Java Standard Edition) afin de faciliter notamment la création d'applications réparties.

4.2.1.1 Java

Java est un langage de programmation et une plate-forme informatique très utilisée, notamment par un grand nombre de développeurs professionnels, ce qui en fait un langage incontournable actuellement, qui a été développé par Sun Microsystems aujourd'hui racheté par Oracle. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut-être utilisé sur internet pour des petites applications intégrées à la page web ou encore comme langage serveur (jsp)[13].

4.2.1.2. Servlet

Une Servlet est un composant web conçu sous la forme d'une classe Java. Elle fonctionne dans un moteur/conteneur de servlet dans un serveur d'application. L'objectif

d'une servlet est de rendre les contenus web dynamiques [10]. Elle comporte deux méthodes fondamentales : une méthode doGet() et une méthode doPost().

4.2.1.3. JSP

Les pages web JSP sont une technologie développée par Sun basée sur Java qui simplifie le processus de développement de sites web dynamiques. L'utilisation du langage de programmation Java permet aux concepteurs de pages qui utilisent JSP d'incorporer rapidement des éléments dynamiques dans les pages web en intégrant du code Java et en utilisant quelques balises (tags) simples. JSP est un langage script côté serveur. Le code de la page est exécuté par le serveur web [11].

4.2.1.4. JSTL

La JSTL est une bibliothèque, une collection regroupant des balises implémentant des fonctionnalités à des fins générales, communes aux applications web. Citons par exemple la mise en place de boucles, de tests conditionnels, le formatage des données ou encore la manipulation de données XML. Son objectif est de permettre au développeur d'éviter l'utilisation de code Java dans les pages JSP, et ainsi de respecter au mieux le découpage en couches recommandé par le modèle MVC.

4.2.1.5. Implémentions de MVC

Le MVC est une solution de développement web coté serveur, permettant de séparer la partie logique/métier de la partie présentation dans une application web. Ceci est trés important, car cela permet au Web Develloper et au Web Designer de travailler séparément, chacun sur des fichiers ou composants, au sein d'un projet. Dans l'architecture MVC [12] :

- Le Modèle est représenté par les EJBs et/ou les JavaBeans.
- La Vue est représenté par le JSPs.
- Le Controleur est représenté par le Servlets.

Le client envoie une requête HTTP à destination d'une servlet. La servlet récupére les données transmises dans la requête HTTP et délègue les traitements avec/sur ces données à des composants JavaBean. Selon les traitements à effectuer, les composants

JavaBean peuvent accéder à des sources de données. Une fois les traitements terminés, les composants rendent la main à la Servlet en lui retournant un résultat. La Servlet stocke ce résultat dans un contexte (session, requête, etc). La Servlet transmet la suite du traitement de la requête vers une JSP. La JSP récupère les données stockées par la Servlet dans un des contextes et génère la réponse HTTP. La réponse HTTP est renvoyée au client [12].

4.2.2 Les standard web

Nous avons utilisé pour développer et organiser les pages web les deux langues web suivantes :

4.2.2.1. HTML

HTML est d'un langage de description (et non pas d'un langage de programmation) qui va nous permettre de décrire l'aspect d'un document web, d'y inclure des informations variées (textes, images, sons, animations etc.).

4.2.2.2. CSS

CSS (Cascading Style Sheets) est un type de fichier permettant de définir la présentation des documents HTML/XML.

4.2.3 Serveur d'application

On utilise tomcate comme un serveur d'application :

4.2.3.1. Le serveur Tomcate

Apache Tomcat est un conteneur web libre de servlets et JSP Java EE. Issu du projet Jakarta, c'est un des nombreux projets de l'Apache Software Foundation. Il implémente les spécifications des servlets et des JSP du Java Community Process, est paramétrable par des fichiers XML et des propriétés, et inclut des outils pour la configuration et la gestion.

4.2.4 L'IDE Eclipse

L'IDE (Integrated Development Environment) eclipse est un logiciel qui simplifie la programmation en proposant un certain nombre de raccourcis et d'aide à la programmation. Il est développé par IBM, est gratuit et disponible pour la plupart des systèmes d'exploitation. C'est un outil puissant, gratuit, libre et multiplateforme, massivement utilisé en entreprise. Il permet l'intégration des outils nécessaires au développement et au déploiement d'une application.

4.3 Base de données

Pour implémenter notre base des données, nous avons utilisé l'environnement de création de base des données PHPMyAdmin et le système de gestion de base des donnés MySQL.

4.3.1 le passage relationnel

Il est nécessaire d'utiliser des règles de transformation du schéma conceptuel obtenu à l'aide de l'extension d'UML pour les applications Web vers un schéma logique de type relationnel.

- Chaque classe du diagramme UML devient une relation.
- Il faut choisir un attribut de la classe pouvant jouer le rôle d'identifiant (PK).
- Transformation des associations (un-a-plusieurs, plusieurs-a-plusieurs, un-a-un).

Nous appliquons sur les classes les mêmes règles. Le schéma de base de données est le suivant :

Utilisateur (id.utilisateur, nom, prénom, date de naissance, lieu de naissance, adresse, email, téléphone, usernamen , mot de passe, lieu d'exercice).

Dossier (id_dossier, id_utilisateur, deplome, spécialité, grade, cv).

Candidature (id_candidature, id_utilisateur, type de stage, période de stage, pays, etat, note, aéroport de départ, aéroport d'arrivée, date de départ, date de retour).

Stage (id_stage, type de stage, pays de stage, période maximale, nbr de stage).

Recoures (id_recoure, id_utilisateur, message).

Session (id_session, id_utilisateur,id_candidature, date de début, date de fin).

4.4 Outil de développent

4.4.1 MySQL

MySQL est un serveur de base de données relationnelle SQL. Il est multi-thread et multi-utilisateurs. Il est très souvent utilisé avec le langage de création des pages Web dynamiques JSP.

4.4.2 Entreprise Architecte

Enterprise Architect a un support complet pour tous les éléments, les relations et les diagrammes UML spécifiées dans 2.5. L'UML est régi par le 'Object Management Group' (OMG) et Sparx Systems est un membre actif et contributeur au processus de gestion et l'amélioration de la langue [14].

4.4.3 Open Element

OpenElement est un logiciel gratuit de création de site Internet conçu pour offrir une interface de développement qui génère automatiquement le code nécessaire au bon fonctionnement du site, et enlève des limites d'édition imposées par d'autres outils. L'avantage est que cela donne une approche plus facile pour les débutants et plus rapide pour certaines tâches répétitives des professionnels. Le concept des "éléments" ou "packs" permet

d'utiliser une multitude d'objets très couramment utilisés dans la création de sites Web, juste par un simple glisser/déposer sur la page et sans avoir à écrire le code. Pour autant, il n'est pas limité aux éléments qu'il contient car vous pouvez créer, partager et réutiliser vos propres éléments [15].

4.5 Présentation de quelques interfaces de notre application

Nous allons présenter dans cette partie les principales pages de l'application.

4.5.1 Page d'accueil

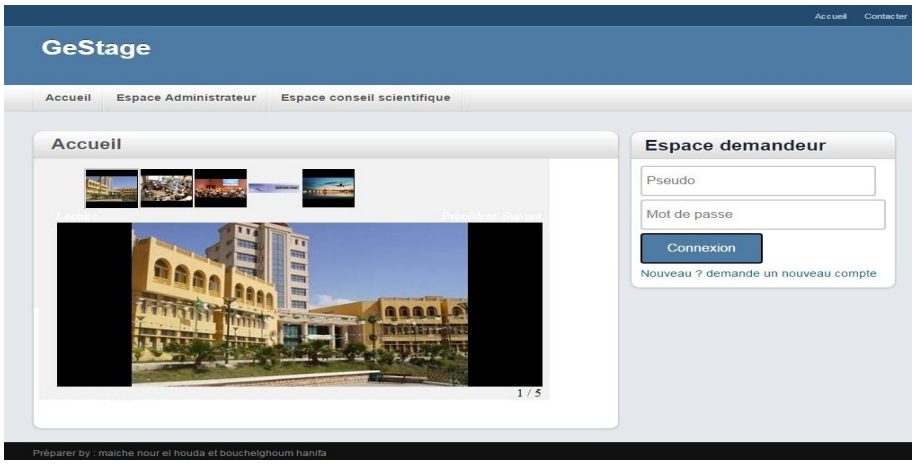

FIGURE 4.1 – L'interface page d'accueil.

4.5.2 La page créer dossier

Parmi les fonctionnalités de notre application, le demandeur de stage peut créer un dossier comme indiquer la figure 4.2.

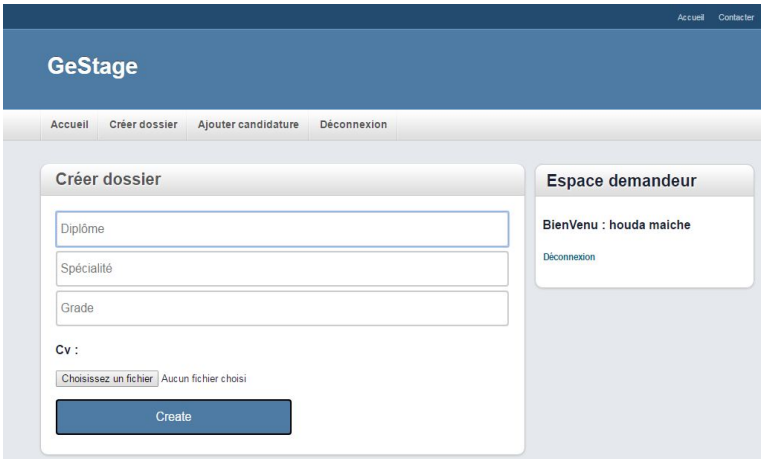

FIGURE 4.2 – L'interface créer dossier.

Remplir le formulaire comme indiquer sur la figure 4.3.

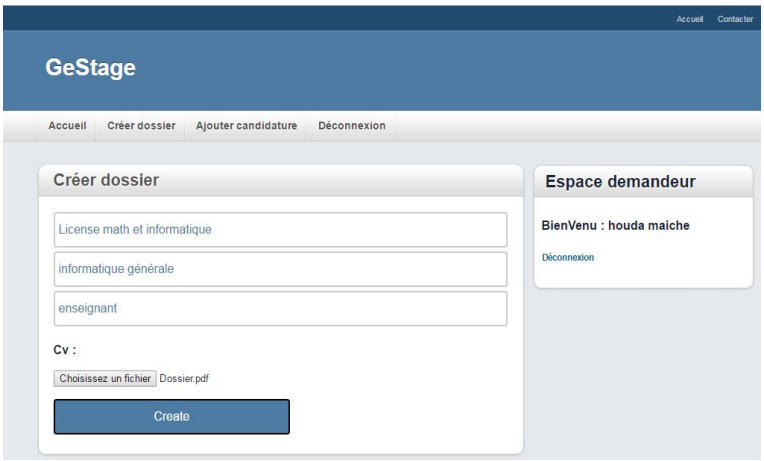

FIGURE 4.3 – Formulaire créer dossier.

La figure 4.4 affiche message de création réussie si les information correcte et le dossier n'existe pas.

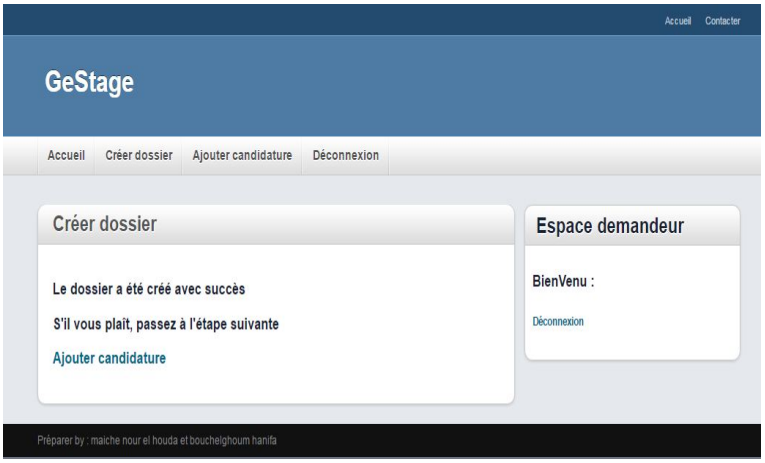

FIGURE 4.4 – Création de dossier réussie.

La figure 4.5 affiche message d'erreur si le dossier déjà existe.

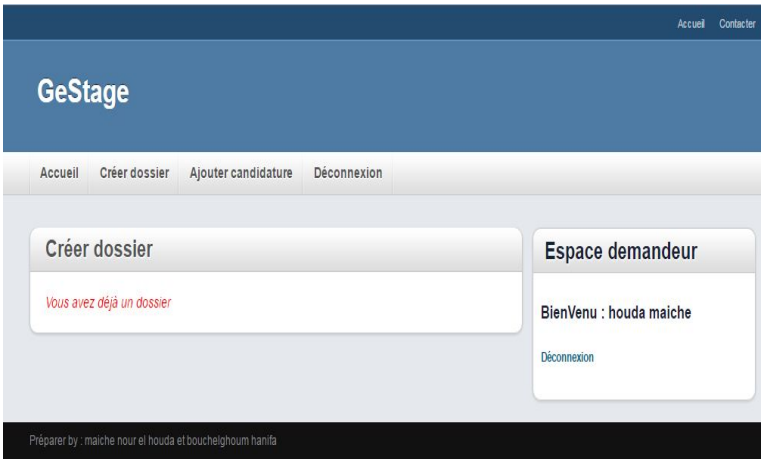

FIGURE 4.5 – Message d'erreur de création.

4.5.3 la page d'authentification

FIGURE 4.6 – L'interface authentification.

4.6 Conclusion

Dans ce chapitre a été consacré à la réalisation et la mise en ?uvre de notre application web, nous avons présenté le langage de programmation et les outils de développement adoptés, ainsi que l'environnement utilisé et enfin nous avons montré les principales interfaces et fenêtres de l'application.

CONCLUSION GÉNÉRALE

Notre travail a été réalisé dans le cadre de projets de fin d'études. Ce travail a comme objectif principal la gestion des stages à l'étranger au sein du centre universitaire de MILA. Pour cela nous avons proposé de concevoir et de réaliser une application web permettant d'assurer une gestion efficace des stages, et d'améliorer la communication et les échanges d'informations et de documentations entre les candidats stagiaires (les enseignants et les étudiants), l'administration et le conseil scientifique. L'application devrait couvrir toutes les étapes de la gestion des stages dès l'ouverture de la session et la présentation des candidatures jusqu'à la délivrance des documents nécessaires au voyage.

Pour la conception de notre application, notre choix s'est porté sur une démarche spécialisée au développement web qui s'appelle « processus simplifié ». Ce dernier s'est inspiré du processus unifié "UP" et s'appuie sur le langage de modélisation UML dans ces différentes phases. L'utilisation du processus simplifié nous a permis de bien comprendre la problématique et de modéliser les besoins fonctionnels de notre système. Il nous a aussi fourni un bon support pendant les phases d'analyse et de conception. Pour la réalisation, nous avons utilisé les technologies proposées par le langage Java pour le développement web et MySQL comme système de gestion de base de données.

Ce travail était pour nous une opportunité pour aborder le domaine de développe-

ment web et d'acquérir une expérience pratique avec une démarche de développement et le langage UML, ainsi qu'avec une panoplie de technologies et d'outils comme le langage Java, la plateforme JEE (les JSPs, les Servlets, le serveur d'application Tomcat), et l'environnement de développement Eclipse.

Malgré les efforts déployés, la réalisation de l'application n'a pas été entièrement achevée, certaines fonctionnalités ne sont pas encore implémentées.

BIBLIOGRAPHIE

[1] P. Roques et F.Vallé, *UML en action*, Livre, quatrième édition, (2007).

[2] D.Pilone et N.Pitman, *UML 2 en concentré*, livre, éditeur O'Reilly,(2006).

[3] P. Roques, *UML-Modélisation une application web* , Livre, troisième édition, (2007).

[4] https ://www.ideematic.com/dictionnaire-web/application-web.

[5] A.Pauthonnier , *Les SIG et les technologies de l'information et de la communication*, Article.

[6] M.munier, *Créer votre application web avec Java EE*, Livre,(2003).

[7] http ://slideplayer.fr/slide/181812/.

[8] http :http ://www.techno-science.net/.

[9] http ://prof.bpesquet.fr/cours/modele-mvc/.

[10] http ://cibretagne.org/news/liens/courshtm/Java/javaj2eeservlet.html.

[11] http ://www-igm.univ-mlv.fr/ dr/XPOSE2001/Jourdan/.

[12] B.Aumaille, *J2EE-Développement d'application Web*, Livre, Editeur ENI, (2002)

[13] http ://www.futura-sciences.com/tech/definitions/internet-java-485/.

[14] http ://www.sparxsystems.fr/.

[15] https ://www.openelement.fr/

[16] J.Molière, *J2EE-les cahiers du programmeur*, Livre, Editeur Eyrolles, (2005.)