Filière Francophone Informatique

AGENCE UNIVERSITAIRE DE LA FRANCOPHONIE

UNIVERSITÉ TECHNIQUE DE MOLDOVA

GESTIUNEA PROIECTELOR INFORMATICE CU AJUTORUL TICHETELOR

GESTION DES PROJETS INFROMATIQUES À L'AIDE DES TICKETS

PROJECT MANAGEMENT IN INFORMATICS BY MEANS OF TICKETS

Etudiant: **PLOAIA Vladislav**

Tuteur: **Conf. univ. MORARU Victor**

Chişinău 2017

Ministère de l'Education de la République de Moldova

Université Technique de Moldova

Faculté Ordinateurs, Informatique et Microélectronique

Département Génie logiciel et Automatique

Filière Francophone Informatique

Admis pour la soutenance

par le Chef de Département : Dr., Conf. univ. Ciorbă Dumitru

_____

„ ___" _____ 2017

GESTION DES PROJETS INFROMATIQUES À L'AIDE DES TICKETS

Projet de licence

Etudiant: _____ (**Ploaia Vladislav**)

Tuteur du projet: _____ (**Moraru Victor**)

Consultants: _____ (**Carcea Liviu**)

_____ (**Melnic Radu**)

_____ (**Covdii Galina**)

Chişinău 2017

Université Technique de Moldova

Faculté Ordinateurs, Informatique et Microélectronique

Departement Génie logiciel et Automatique

Spécialité Technologies Informationnelles

Approuve par

Chef de Département Dr., conf. univ. Ciorbă Dumitru

_____

„__”_____

2016

CAHIER DES TÂCHES

pour la thèse de licence de l'étudiant

Ploaia Vladislav

(nom et prénom d'étudiant)

1. Thème du projet de licence Gestion des projets informatiques à l'aide des tickets **confirmé par l'ordre nr. 12 de 24. 10.2016**

2. Date limite de présentation du projet 31 mai 2017

3. Données initiales pour l'élaboration du projet Etude du langage Java. Développement d'un site en ligne pour gestionner des projets informatiques à l'aide des tickets.

4. Contenu du mémoire explicatif Généralités. Description du système. Description des techniques utilisées pour concevoir le projet. Description du système en utilisant le langage UML. Description des fonctionnalités. Argumentation économique. Conclusions.

5. Contenu de la partie graphique du projet Diagrammes de cas d'utilisation, de SSD, de patterns GRASP, de composants, d'activité, d'état, de deploiement, de l'architecture logique. Images, tableaux qui decrivent le système, l'architecture de celui-ci, la mise en œuvre et justification économique.

6. Liste des consultants:

Consultant	Chapitre	Confirmation de la réalisation d'activité	
		Signature du consultant (date)	Signature d'étudiant (date)
Carcea L.	Standarts		
Covdii G.	Economie		
Melnic Radu	AMSI		

7. Date de délivrance du cahier des charges 6 mars 2017

Tuteur du projet _____
signature

La tâche a été prise pour l'exécution par l'étudiant _____
signature, date

Planification des activités

No. crt.	Noms des étapes d'élaboration	Durée de la réalisation étapes	Note
1	Etablissement des specifications sur le projet	06.03.17 – 09.03.17	
2	Planification du temps	10.03.17 – 13.03.17	
3	Conception fonctionnelle	14.03.17 – 21.03.17	
4	Conception détaillée	22.03.17 – 28.03.17	
5	Installations et configurations des outils nécessaires	28.03.17 – 06.04.17	
6	Développement de l'application	07.04.17 – 24.04.17	
7	Vérification de l'application	25.04.17 – 27.04.17	
8	Mise en œuvre	28.04.17 – 22.05.17	
9	Documentation	23.05.17 – 31.05.17	

Etudiant _____
signature

Tuteur du projet _____
signature

Adnotare

Teza dată vizează analiza tehnologiilor Web existente de tip ,,open source'' și dezvoltarea unui sistem pentru gestiunea proiectelor informatice prin intermediul ticketelor.

Această teză este compusă din introducere, cinci capitole, fiecare capitol conținând mai multe subcapitole, concluzie și bibliografie.

Capitolul unu prezintă descrierea tehnologiilor utilizate pentru planificarea, proiectarea și realizarea aplicației.

Capitolul doi este dedicat modelării sistemului utilizând limbajul unificat de modelare UML, care reprezintă un instrument util, folosit pentru analiza și modelarea aplicației.

Diagramele UML sunt elementele de bază, care descriu sistemul din diverse perspective, fiecare diagramă reprezentând o parte sau un tot întreg al aplicației. Printre acestea se numără: cazuri de utilizare, de activitate, de stare, de secvență și de componente. Adăugător, pentru fiecare caz de utilizare prezentat, s-a făcut o descriere detaliată, astfel, aducând o înțelegere mai amplă al sistemului ca o componentă integră sau o parte din acesta. De asemenea, similar s-a procedat și în cazul diagramelor de secvență SSD, furnizând o descriere detaliată a acestora. Șabloanele de proiectare GRASP se regăsesc printre aceste diagrame, și utilitatea lor este explicată prin aceea că, se alocă responsabilitățile pentru clase de obiecte în cadrul aplicației curente orientate obiect. Fiecare din ele, conține o serie de explicații care justifică utilizarea acestora în diverse cazuri. Printre diagrame UML se mai regăsesc și cele de deploiment, și de arhitectură logică.

Capitolul trei este dedicat conceperii bazei de date, mai exact, pașii care trebuie urmați pentru a obține o bază de date care v-a stoca toate înregistrările efectuate de actorii sistemului (cei principali și/sau cei de suport).

Capitolul patru este alcătuit din porțiuni de cod, scris în limbajul Java, o detaliere referitoare la conținutul fiecărei porțiuni și rezultatele prezentate prin intermediul imaginilor captate din aplicația propriu-zisă.

Capitolul cinci vine să accentueze partea economică, reprezentând o imagine generală cu privire la indicatorii economici și calculele necesare pentru proiectul dat. Partea economică se finisează cu prezentarea rezultatelor economice și o concluzie despre etapele realizate.

La nivel de limbaj de programare, de implementare a tezei, s-a decis utilizarea limbajului Java, datorită unei bune integrări cu tehnologiile pentru construirea paginilor Web. În plus, au fost folosite limbaje ca: HTML (pentru pagini web cu design static), CSS (pentru o aplicație user-friendly), JavaScript, jQuery (implementarea diverselor funcționalități), MySQL (baza de date relațională) și un tip de server Tomcat pe care rulează aplicația.

Annotation

La thèse aborde l'analyse des technologies Web existantes de type ,,open source'' et le développement d'un système pour la gestion des projets informatiques à l'aide des tickets.

Cette thèse est composée par l'introduction, cinq chapitres, chaque chapitre contenant plusieurs sous chapitres, la conclusion et la bibliographie.

Le premier chapitre présente la description des technologies utilisées pour la planification, projection et la réalisation de l'application.

Le deuxième chapitre est dédié à la modelisation du système en exploitant le langage unifié pour la modelisation UML, qui représente un instrument util, appliqué pour l'analyse et la modelisation de l'application.

Les diagrammes UML sont les éléments de base, qui decrivent le système à partir de différentes perspectives, chaque diagramme représentant une partie ou un entier de l'application. Parmi celles-ci, on trouve: les cas d'utilisation, d'activité, d'état, de séquence et des componantes. En plus, pour chaque cas d'utilisation présenté, on a été réalisé une description détaillé de la manière qu'on a apporté une large vision sur le système comme un composant ou comme une partie de celui-ci. De même, on a procédé dans le cas des diagrammes de séquence SSD, en fournissant une description détaillé. Les modèles de conception GRASP sont trouvés parmi ces diagrammes, et leurs utilité est justifié par le fait, qu'on alloue les responsabilités pour chaque classe d'objets en cadre de l'application courante orienté objet. Chacun d'elles contient une série des explications qui justifient leurs utilisation dans différents cas. Parmi les diagrammes UML, on trouve aussi ces de deployment et de l'architecture logique.

Le troisième chapitre est consacré à la conception de la base des données, notamment, aux pas qu'il faut suivre pour obtenir une base des données qui gardera les enregistrements effectués par les acteurs du système (ceux principaux et/ou ceux du support).

Le quatrième chapitre est composé de plusieures portions du code, écrit en Java, en décrivant le contenu du chaque portion de code, et en présentant les résultats par les images de capture d'écran de l'application.

Le chapitre cinq accentue la partie économique, en représentant une image générale concernant les indicateurs économiques et les calculs nécessaires pour ce projet. La partie économique se finalise avec la présentation des résultats économiques et une conclusion sur les étapes réalisées.

Au niveau de langage du programmation, on a utilisé Java, grâce à une bonne integration avec les technologies Web. En plus, on a utilisé les langages comme: HTML (pour les pages Web), CSS (pour une application user-friendly), JavaScript, jQuery (l'implementation des différentes fonctionnalités), MySQL (base des données) et serveur Tomcat.

Annotation

The thesis aims to analyze existing Web technologies ,,open source'' and developing a system for management of projects in informatics by means of tickets.

The thesis is composed of introduction, five chapters, each chapter having several subchapters, conclusion and bibliography.

The first chapter presents a description of technologies used for planification, design and implementation of application.

The second chapter is dedicated for system modeling, using the unified UML modeling language, which is a useful tool used to analyze and model the application.

UML charts are the basics that describe the system from different perspectives, each chart representing a part or a whole of the application. These include: use cases, activity, sequence and components. Additionally, for each use case presented, a detailed description has been made, thus providing a wider understanding of the system as an integral component or a part thereof. Similarly, SSD sequence diagrams have also been provided, with a detailed description of these. GRASP design templates are among these charts, and their usefulness is explained by the fact that responsabilities for object classes are allocated within the current object-oriented application. Each of them contains a number of explanations that justify their use in various cases. Among the UML charts are deployements and logic architecture.

Chapter three is devoted to designing the database, more precisely, the steps you need to follow to get a database that stores all of the records, made by actors of the system (main and/or support).

The fourth chapter consists of Java-coded portions of code, a breakdown of the content of each portion and the results presented through screenshots captured in the application itself.

Chapter five comes to emphasize the economic side, representing a general view of the economic indicators and the calculations required for the given project. The economic side finishes with the presentation of the economic results and a conclusion about the achieved stages.

At the level of programming language, it was decided to use the Java language, due a good integration with the Web technologies for building Web pages. In addition, are used languages such as HTML (for static design web pages), CSS (for a user-friendly application), JavaScript, jQuery (implementation of various functionalities), MySQL (relational database) and a server type Tomcat on which the app is running.

Table des matières

INTRODUCTION..... 10
1. TECHNOLOGIES UTILISÉES POUR CONCEVOIR LE PROJET.... 11
1.1 UML..... 11
1.2 HTML..... 13
1.3 CSS..... 13
1.4 JavaScript..... 14
1.5 PhpMyAdmin..... 14
1.6 Java..... 15
1.6.1 JSP..... 16
1.6.2 Servlet..... 16
1.6.3 Hibernate..... 16
1.6.4 HQL 17
1.6.5 Hibernate Search..... 17
1.6.6 Spring Framework..... 18
1.6.7 RESTful Web Service..... 18
1.6.8 Apache Tomcat..... 19
1.7 MySQL..... 20
1.8 JQuery..... 20
2. STRUCTURE DU PROJET..... 21
2.1 Diagrammes de cas d'utilisation..... 21
2.1.1 Caracteristiques des cas d'utilisation..... 24
2.2 Diagrammes SSD..... 35
2.3 Utilisation des patterns GRASP..... 40
2.4 Diagrammes de composants..... 42
2.5 Diagrammes d'activité..... 43
2.6 Diagrammes d'état..... 46
2.7 Diagramme de déploiement..... 47
2.8 Conception de l'architecture logique..... 47
2.9 Identification d'autres besoins..... 48
3. CONCEPTION DE LA BASE DES DONNÉES..... 49
3.1 Modèles de la base des données..... 49
4. PRÉSENTATION DU CODE ET DES RÉSULTATS..... 52

5. ARGUMENTATION ÉCONOMIQUE DU PROJET..... 65
5.1 Description du projet..... 65
5.2 L'analyse SWOT..... 66
5.3 Plan de calendrier..... 67
5.4 Argumentation économique..... 68
5.4.1 Les actifs matériels et non matériels à long terme..... 68
5.4.2 Rémunération du travail..... 69
5.4.3 Les dépenses pour la rémunération du travail..... 69
5.4.4 La somme annuel de l'impôt sur le revenu d'un développeur..... 70
5.4.5 Les dépenses indirectes..... 71
5.5 Calcul des fonds d'amortissement d'actifs matériels directe et non directe..... 71
5.6 Prix du coût..... 72
5.7 Les résultats financiers..... 73
5.8 Conclusion sur la partie économique..... 73
CONCLUSION..... 74
BIBLIOGRAPHIE..... 75
ANNEXE A..... 76

INTRODUCTION

Des nos jours, un tel ,,Gestion des projets informatiques à l'aide des tickets'' est indispensable dans le processus de gestion des projets, notamment dans le processus de la création et du suivi de la réalisation des tâches, qui aide les développeurs à se diriger par les éxigences imposées sur les projets, qui, en effet, vont livrer un produit compétitive sur le marché actuel. On trouve le système très actuel puisqu'il peut être utilisé en cadre de plusieures entreprises, pour faciliter le développement des logiciels et des solutions spécifiques pour les clients.

Ce logiciel va permettre la gestion des projets et des tickets (des tâches) à réaliser, pour accomplir les éxigences des clients. Comme personnes qui interagiront avec le système on aura des administrateurs et des développeurs qui auront des différents droits d'accès aux certaines fonctionnalités disponibles.

Parmi les fonctionnalités qui pouront être utilisées par les administrateurs seront : la création des projéts, la modification des projets, l'enlèvement des projets, la création des tickets, la modification des tickets, l'enlèvement des tickets, la création et l'envoye des message vers les utilisateurs (développeurs), l'ajout d'un utilisateur (développeur) au projet, l'enlèvement d'un utilisateur (développeur) du projet, l'enlèvement des utilisateurs (développeurs), le changement du statut d'un utilisateur (développeur) dans l'administrateur, la visualisation des utilisateurs (développeurs) du projet, la visualisation des projets, la visualisation des tickets, la visualisation des utilisateurs.

Comme fonctionnalités disponibles aux utilisateurs (développeurs) seront : la modification des tickets, la création et l'envoye des messages vers les utilisateurs (développeurs) et les administrateurs, la visualisation des tickets, la visualisation des projets, la visualisation des utilisateurs (développeurs) qui travaillent sur le projet, la visualisation des administrateurs du projet.

1. TECHNOLOGIES UTILISÉES POUR CONCEVOIR LE PROJET

Dans cette portion on a décrit les technologies utilisées dans ce projet. En ce cas-là, UML est utilisé pour la modélisation du projet. Les normes HTML, JavaScript, CSS, JSP sont utilisées pour développer l'interface utilisateur pour le projet. Java est utilisée comme un langage de programmation, pour la création du modèle des données et logique d'interaction avec le programme. MySQL est utilisé comme base des données pour garder l'information du site web et Hibernate joue le rôle d'un intermédiaire d'échange des données entre modèle des données Java et les données de base des données.

Parmi les technologies qu'on va les utiliser pour concevoir le projet, seront les suivantes:

1.1 UML

UML (en anglais Unified Modeling Language), ou Langage de Modélisation Unifié, est un langage de modélisation graphique à base de diagrammes. Il est souvent utilisé en développement logiciel, et en cadre de la conception orientée objet. UML est un résultat de la fusion de précédents langages de modélisation objet : OMT, Booch, OOSE. UML est défini par une organisation nommée Object Management Group (OMG).

Utilité d'UML

UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet. UML offre un standard de modélisation, pour représenter l'architecture logicielle.

Les différents éléments représentables sont :

- Schéma de base de données ;
- Acteurs ;
- Processus ;
- Activité d'un objet/logiciel ;
- Composants logiciels ;
- Réutilisation de composants.

Grâce aux outils de modélisation UML comme Rational Rose ou Enterprise Architect, il est également possible de générer automatiquement une partie de code, dans une variété des langages de programmation, par exemple en langage Java, à partir des divers documents réalisés.

Les diagrammes

Les 14 diagrammes UML sont dépendants hiérarchiquement et se complètent, ayant comme but, de permettre la modélisation d'un projet tout au long de son cycle de vie.

Diagrammes comportementaux

Les diagrammes comportementaux (Behavior Diagram) rassemblent :

• Diagramme des cas d'utilisation (Use Case Diagram ou use-cases) : il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système ;

• Diagramme d'activité (Activity Diagram) : permet de décrire sous forme de flux ou d'enchaînement d'activités le comportement du système ou de ses composants ;

• Diagramme états-transitions (State Machine Diagram) : permet de décrire sous forme de machine à états finis le comportement du système ou de ses composants.

Diagrammes structurels

Les diagrammes structurels ou statiques (Structure Diagram) rassemblent :

• Diagramme d'objets (Object diagram) : il sert à représenter les instances de classes (objets) utilisées dans le système ;

• Diagramme de classes (Class diagram) : il représente les classes intervenant dans le système ;

• Diagramme de composants (Component diagram) : il permet de montrer les composants du système d'un point de vue physique, tels qu'ils sont mis en oeuvre (bibliothèques, fichiers, bases de données…) ;

• Diagramme de déploiement (Deployment diagram) : il sert à représenter les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage…) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux ;

• Diagramme des paquetages (Package diagram) : un paquetage étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML, le diagramme de paquetage sert à représenter les dépendances entre paquetages, c’est-à-dire les dépendances entre ensembles de définitions ;

• Diagramme de profils (Profile diagram (en)) : depuis UML 2.2, permet de spécialiser, de personnaliser pour un domaine particulier un meta-modèle de référence d'UML ;

• Diagramme de structure composite (Composite Structure Diagram) : depuis UML 2.x, permet de décrire sous forme de boîte blanche les relations entre composants d'une classe.

Diagrammes d'interaction ou dynamiques

Les diagrammes d'interaction ou dynamiques (Interaction Diagram) rassemblent :

• Diagramme de communication (Communication Diagram) : depuis UML 2.x, représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets ;

• Diagramme de séquence (Sequence Diagram) : représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et/ou de ses acteurs ;

• Diagramme de temps (Timing Diagram) : depuis UML 2.3, permet de décrire les variations d'une donnée au cours du temps ;

• Diagramme global d'interaction (Interaction Overview Diagram) : depuis UML 2.x, permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences (variante du diagramme d'activité).

1.2 HTML

L'HyperText Markup Language, généralement abrégé comme HTML, est le format de données utilisé pour représenter les pages Web. HTML un langage de balisage, permettant d'écrire de l'hypertexte, d'où son nom. HTML permet de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec le langage de programmation JavaScript et des feuilles de style en cascade (CSS). HTML est initialement dérivé du Standard Generalized Markup Language (SGML).

Le langage de balisage HTML permet de structurer un document et de le mettre en forme. Cependant, ce langage utilisé intensément par les sites internet tend à devenir exclusivement un langage de structuration en se rapprochant du format XML, laissant la mise en forme aux feuilles de styles en cascade (CSS), et les animations au JavaScript.

L'un de ses inconvénients est que l'affichage est dépendante du terminal, notamment, de la taille de l'écran, du navigateur, du niveau de zoom, des polices de caractères disponibles, etc. Ceci nécessite donc de tester les pages avec différents terminaux. Pour éviter les mauvaises surprises, il faut respecter les standards du Web.

1.3 CSS

Les CSS, Cascading Style Sheets (feuilles de styles en cascade), servent à mettre en forme des documents web, type page HTML ou XML. Par l'intermédiaire de propriétés d'apparence (couleurs, bordures, polices, etc.) et de placement (largeur, hauteur, côte à côte, dessus-dessous, etc.), le rendu d'une page web peut être intégralement modifié sans aucun code supplémentaire dans la page web. Les feuilles de styles ont d'ailleurs pour objectif principal de dissocier le contenu de la page de son apparence visuelle. Ceci permet :

• de ne pas répéter dans chaque page le même code de mise en forme ;

• d'utiliser des styles génériques, avec des noms explicites (par exemple un style encadré pour du texte ou des images) ;

• de pouvoir changer l'apparence d'un site web complet en ne modifiant qu'un seul fichier ;

• de faciliter la lecture du code de la page.

La puissance et de l'intérêt des CSS peut être démontrée en modifiant radicalement l'apparence d'une page, sans changer son code HTML. Les CSS permettent de gagner en productivité et en maintenabilité des sites web, tout en offrant des possibilités graphiques incontestables.

1.4 JavaScript

JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs2 avec l'utilisation (par exemple) de Node.JS. JavaScript a été créé en 1995 par Brendan Eich. Il a été standardisé sous le nom d'ECMAScript en juin 1997 par Ecma International dans le standard ECMA-262. Le standard ECMA-262 en est actuellement à sa 7e édition. JavaScript n'est depuis qu'une implémentation d'ECMAScript, celle mise en œuvre par la fondation Mozilla.

Aujourd'hui, presque toutes les pages Web contiennent du code JavaScript, un langage de programmation de scripts exécuté par le navigateur Web du visiteur. Il apporte aux pages Web des fonctionnalités correspondant à des besoins spécifiques, et s'il est désactivé pour une quelconque raison, le contenu ou l'utilisation de la page Web peuvent s'en trouver restreints ou indisponibles.

La particularité du JavaScript consiste à créer des petits scripts sur une page HTML dans le but d'ajouter une petite animation ou un effet particulier sur la page. Cela permet en général d'améliorer l'ergonomie ou l'interface utilisateur, mais certains scripts sont peu utile et servent surtout à ajouter un effet esthétique à la page.

1.5 PhpMyAdmin

PhpMyAdmin [5] est un outil logiciel gratuit écrit en PHP, destiné à gérer l'administration de MySQL sur le Web. PhpMyAdmin prend en charge une large gamme d'opérations sur MySQL et MariaDB. Les opérations fréquemment utilisées (gestion des bases de données, des tableaux, des colonnes, des relations, des index, des utilisateurs, des autorisations, etc.) peuvent être effectuées via l'interface utilisateur, alors que vous avez toujours la possibilité d'exécuter directement une instruction SQL.

PhpMyAdmin peut gérer l’ensemble d’un serveur MySQL (cela nécessite un compte super-utilisateur) aussi bien qu’une seule base de données. Pour ce faire, il est nécessaire d’avoir un droit de lecture/écriture sur la base de données concernée.

Actuellement phpMyAdmin peut :

• parcourir et supprimer bases de données, tables, vues, champs et index ;

• afficher plusieurs ensembles de résultats au moyen de procédures stockées ou de requêtes ;

• créer, copier, supprimer, renommer et modifier les bases de données, tables, vues, champs et index ;

• maintenir serveur, bases de données et tables en proposant une configuration serveur ;

• exécuter, modifier et placer en signets n'importe-quelle instruction SQL, incluant les traitements par lots ;

• charger des tables à partir du contenu de fichiers texte ;

• créer et lire des fichiers d'exportation (dumps) de tables;

• exporter des données dans divers formats : CSV, XML, PDF, ISO/IEC 26300 - texte et feuille de calcul OpenDocument, Microsoft Word 2000 et LATEX ;

• importer des données et des structure MySQL à partir de feuilles de tableur OpenDocument, ainsi que de fichiers SQL, CSV et SQL

• administrer plusieurs serveurs ;

• gérer les utilisateurs et les privilèges MySQL.

1.6 Java

Le langage Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton, employés de Sun Microsystems, avec le soutien de Bill Joy (cofondateur de Sun Microsystems en 1982), présenté officiellement le 23 mai 1995 au SunWorld.

La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que UNIX, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.

La première caractéristique, le caractère orienté objet (« OO ») et familier, fait référence à une méthode de programmation et de conception du langage et le fait qu'un programme écrit en Java ressemble assez fort à un programme écrit en C++.

Dans la version 1.5 du langage ont été rajoutés les génériques, un mécanisme de polymorphisme semblable (mais différent) aux templates du langage C++ ou aux foncteurs d'OCaml. Les génériques permettent d'exprimer d'une façon plus simple et plus sûre les propriétés d'objets comme des conteneurs (listes, arbres…) : le type liste est alors considéré génériquement par rapport au type d'objet contenu dans la liste.

1.6.1 JSP

Le JavaServer Pages ou JSP est une technique basée sur Java permettant aux développeurs de créer dynamiquement du code HTML, XML ou tout autre type de page web. Cette technique permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique. Depuis la version 2.0 des spécifications, la syntaxe JSP est complètement conforme au standard XML.

La syntaxe du JSP ajoute des balises XML, appelées actions JSP, qui peuvent être utilisées pour appeler des fonctions. De plus, cette technique permet la création de bibliothèques de balises JSP (taglib) qui agissent comme des extensions au HTML ou au XML. Les bibliothèques de balises offrent une méthode indépendante de la plate-forme pour étendre les fonctionnalités d'un serveur HTTP. Il existe aussi un langage de script particulier, appelé Expression Language (EL) destiné à réduire l'injection de code java au sein des pages JSP ainsi qu'à étendre les possibilités des taglibs, tel que la JSTL.

Les JSP sont compilées par un compilateur JSP pour devenir des servlets Java. Un compilateur JSP peut créer une servlet Java en code source Java qui peut à son tour être compilé par le compilateur Java, ou peut créer le pseudo-code Java interprétable directement. Dans les deux cas, il est bon de comprendre comment le compilateur JSP transforme la page en servlet Java.

1.6.2 Servlet

La technologie Servlet est utilisé pour créer des applications Web (réside à côté serveur et génère la page Web dynamique). La technologie Servlet est robuste et évolutive en raison du langage java. Avant Servlet, CGI (Common Gateway Interface) langage de script est populaire comme un langage de programmation côté serveur. Mais il y avait de nombreux inconvénients de cette technologie.

Il existe de nombreuses interfaces et classes de l'API de servlet tels que Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

1.6.3 Hibernate

Hibernate est une solution de type ORM (Object Relational Mapping) qui facilite le développement de la couche persistance d'une application. Hibernate permet de représenter une base de données en objets Java et vice versa.

Il facilite la persistance et la recherche de données dans une base de données en réalisant lui-même la création des objets et les traitements de remplissage de ceux-ci en accédant à la base de données. La quantité de code ainsi épargnée est très importante d'autant que ce code est généralement fastidieux et redondant.

Hibernate est très populaire notamment à cause de ses bonnes performances et de son ouverture à de nombreuses bases de données.

Les bases de données supportées sont les principales du marché : DB2, Oracle, MySQL, PostgreSQL, Sybase, SQL Server, Sap DB, Interbase,etc.

Hibernate a besoin de plusieurs éléments pour fonctionner :

• un fichier de configuration qui assure la correspondance entre la classe et la table (mapping) ;

• des propriétés de configuration notamment des informations concernant la connexion à la base de données .

Hibernate permet aussi de générer la base de données en spécifiant dans un fichier de configuration, une option pour la génération de la base de données.

1.6.4 HQL

Hibernate Query Language (HQL) est un langage de requête orienté objet, similaire à SQL, mais au lieu de fonctionner sur des tables et des colonnes, HQL fonctionne avec des objets persistants et leurs propriétés. Les requêtes HQL sont traduites par Hibernate en requêtes SQL classiques qui, à leur tour, effectuent une action sur la base de données. On peut utiliser des instructions SQL directement avec Hibernate à l'aide de Native SQL, mais je on recommande d'utiliser HQL chaque fois que possible pour éviter les tracas de portabilité de la base de données et de profiter des stratégies de génération et de mise en cache SQL de Hibernate.

Les mots-clés tels que SELECT, FROM et WHERE etc. ne sont pas sensibles à la casse, mais les propriétés comme les noms de tableaux et de colonnes sont sensibles à la casse dans HQL.

1.6.5 Hibernate Search

Hibernate Search indexe de manière transparente des objets et offre une recherche rapide et régulière de texte intégral et de géolocalisation. La facilité d'utilisation et le regroupement facile sont essentiels.

Recherche de texte intégral pour les entités

Offre un support de recherche de texte intégral pour les objets stockés par Hibernate ORM, Infinispan et d'autres sources. Pensez-y comme Google (tm) pour les entités:

• Mots de recherche avec du texte ;

• Ordonne les résultats par pertinence ;

• Trouve par approximation (recherche floue).

Cluster-friendly

La clusterisation de l’index n'est pas trivial. Hibernate Search offre plusieurs stratégies de clustering faciles à configurer:

• Maître / esclaves ;

• Réplication JMS / JGroups ;

• Index distribué Infinispan .

Facettes et géolocalisation

Les entités géolocalisées sont aussi faciles que @Spatial. Filtrer les résultats autour d'un certain emplacement, comme la position de l'utilisateur. Hibernate Search offre deux algorithmes: un léger ou un plus évolutif. Faceting classe les résultats par des propriétés comme la gamme de prix ou la marque.

Facile à utiliser

Conçu pour être facile à utiliser dès le début. Gère l'indexation, la synchronisation de banque de données, le clustering et l'infrastructure de manière transparente pendant que vous vous concentrez sur le côté commercial de vos requêtes.

1.6.6 Spring Framework

Spring est un framework libre [1] pour construire et définir l'infrastructure d'une application java, dont il facilite le développement et les tests.

Comme on peut voir dans la référence [2], Spring est considéré comme un conteneur dit « léger ». La raison de ce nommage est expliquée par Erik Gollot dans l’introduction du document Introduction au framework Spring. Spring est effectivement un conteneur dit « léger », c’est-à-dire une infrastructure similaire à un serveur d'applications J2EE. Il prend donc en charge la création d’objets et la mise en relation d’objets par l’intermédiaire d’un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets. Le gros avantage par rapport aux serveurs d’application est qu’avec Spring, les classes n’ont pas besoin d’implémenter une quelconque interface pour être prises en charge par le framework (au contraire des serveur d'applications J2EE et des EJBs). C’est en ce sens que Spring est qualifié de conteneur « léger ».

Ce framework, grâce à sa couche d’abstraction, ne concurrence pas d’autres frameworks dans une couche spécifique d’un modèle architectural Modèle-Vue-Contrôleur mais s’avère un framework multi-couches pouvant s’insérer au niveau de toutes les couches ; modèle, vue et contrôleur. Ainsi il permet d’intégrer Hibernate ou iBATIS pour la couche de persistance ou encore Struts et JavaServer Faces pour la couche présentation.

1.6.7 RESTful Web Service

REST (en anglais ,,Representational State Transfer’’, en français ,,Transfert d'État représentatif ’’) a été introduit et défini en 2000 par Roy Fielding dans sa thèse de doctorat.

REST est un style architectural pour la conception de systèmes distribués. Ce n'est pas une norme, mais un ensemble de contraintes, telles que l'apatride, la relation client / serveur et une interface uniforme. REST n'est pas strictement lié à HTTP, mais il est le plus souvent associé à celui-ci.

Principes de REST

• Les ressources exposent les URI de structure de répertoire facilement compréhensibles. Les représentations transfèrent JSON ou XML pour représenter les objets et les attributs de données ;

• Les messages utilisent explicitement les méthodes HTTP (par exemple GET, POST, PUT et DELETE) ;

• Les interactions sans état ne stockent aucun contexte de client sur le serveur entre les requêtes. Les dépendances d'état limitent et limitent l'évolutivité. Le client détient l'état de session.

Types de supports

Les en-têtes HTTP Accept et Content-Type peuvent être utilisés pour décrire le contenu envoyé ou demandé dans une requête HTTP. Le client peut configurer Accepter à application / json s'il demande une réponse dans JSON. À l'inverse, lors de l'envoi de données, le réglage de Content-Type sur application / xml indique au client que les données envoyées dans la requête sont XML.

1.6.8 Apache Tomcat

Le logiciel Apache Tomcat est une application open source de Java Servlet, JavaServer Pages, Java Expression Language et Java WebSocket. Les langages Java Servlet, JavaServer Pages, Java Expression Language et Java WebSocket sont développés sous Java Process Community.

Le logiciel Apache Tomcat est développé dans un environnement ouvert et participatif et publié sous la version Apache License 2. Le projet Apache Tomcat est destiné à être une collaboration entre les meilleurs développeurs du monde entier.

Le logiciel Apache Tomcat gère de nombreuses applications Web à grande échelle et stratégiques dans une large gamme d'industries et d'organisations. Certains de ces utilisateurs et leurs histoires sont répertoriés sur la page wiki PoweredBy.

Apache Tomcat, Tomcat, Apache, la plume d'Apache et le logo du projet Apache Tomcat sont des marques déposées de Apache Software Foundation.

Tomcat 7.x implémente les spécifications Servlet 3.0 et JSP 2.2. Il nécessite la version 1.6 de Java, bien que les versions précédentes soient exécutées sur Java 1.1 à 1.5. Les versions 5 à 6 ont vu des améliorations dans la collecte des ordures, l'analyse par JSP, les performances et

l'évolutivité. Les wrappers natifs, appelés «Tomcat Native», sont disponibles pour Microsoft Windows et Unix pour l'intégration de plates-formes.

1.7 MySQL

MySQL est le système de gestion de base de données, le plus populaire Open Source relationnelle SQL. C'est l'un des meilleurs SGBDR utilisé pour développer des applications logicielles basées sur le Web.

Il est rapide, SGBDR facile à utiliser étant utilisé pour de nombreuses petites et grandes entreprises. Il est devenu ainsi populaire en raison de nombreuses bonnes raisons:

• Est un programme très puissant dans son propre droit. Il gère une grande partie des fonctionnalités des packages de base de données les plus chères et les plus puissants ;

• Est publié sous une licence open-source. On n'a donc rien à payer pour l'utiliser ;

• Fonctionne sur de nombreux systèmes d'exploitation et avec de nombreuses langues, y compris PHP, PERL, C, C ++, JAVA, etc ;

• Est très pratique pour PHP, le langage le plus apprécié pour le développement web ;

• Supporte de grandes bases de données, jusqu'à 50 millions de lignes ou plus dans une table. La limite de taille de fichier par défaut pour une table est de 4 Go, mais vous pouvez augmenter cette (si votre système d'exploitation peut gérer) à une limite théorique de 8 millions de téraoctets .

1.8 JQuery

JQuery est une bibliothèque JavaScript multiplate-forme conçue pour simplifier les scripts côté client de HTML. C'est un logiciel open source gratuit utilisant la licence MIT permissive. L'analyse web indique que c'est la bibliothèque JavaScript le plus largement déployée par une large marge.

La syntaxe de jQuery est conçue pour faciliter la navigation dans un document, sélectionner les éléments DOM, créer des animations, gérer des événements et développer des applications Ajax. JQuery fournit également des fonctionnalités aux développeurs pour créer des plug-ins au-dessus de la bibliothèque JavaScript. Cela permet aux développeurs de créer des abstractions pour une interaction et une animation de bas niveau, des effets avancés et des widgets à haut niveau et à thème. L'approche modulaire de la bibliothèque jQuery permet la création de puissantes pages Web et applications Web dynamiques.

L'ensemble des fonctionnalités de base jQuery-sélections d'éléments DOM, de traversée et de manipulation activés par son moteur de sélection (appelé "Sizzle" de v1.3) a créé un nouveau "style de programmation", des algorithmes de fusion et des structures de données DOM. Ce style a influencé l'architecture d'autres frameworks JavaScript comme YUI v3 et Dojo, stimulant plus tard la création de l'API de sélecteurs standard.

Microsoft et Nokia regroupent jQuery sur leurs plates-formes. Microsoft l'inclut avec Visual Studio pour être utilisé dans les frameworks ASP.NET AJAX et ASP.NET MVC de Microsoft, tandis que Nokia l'a intégré dans la plate-forme de développement de widgets Web Run-Time.

2. STRUCTURE DU PROJET

En cette partie, on va montrer à l'aide des diagrammes UML, la structure du projet, qui va decrire l'architecture du système et les interactions entre les composantes logiques du celle-ci. Chaque type de diagramme complète de plus en plus, la vision de differents points de vue, notamment par les differents types des diagrammes comme celles: comportamentales, structureles et dynamiques.

2.1 Diagrammes de cas d'utilisation

Premièrement, pour accomplir chaque fonctionnalité du système, on aura besoin de diagrammes de cas d'utilisation qui vont permettre d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système.

Pour une meilleure compréhension des fonctionnalités qui doit fournir le système, on a décidé de créer une diagramme générale des paquets contenant des paquets dans lesquels on va placer les diagrammes de use cases. On a pris cette décision pour structurer d'une telle manière car, sans l'utilisation des paquets on va obtenir un diagramme de cas d'utilisation de grande taille, mais à l'aide des paquets, on peut le structurer en sous-ensambles gérables et lisibles.

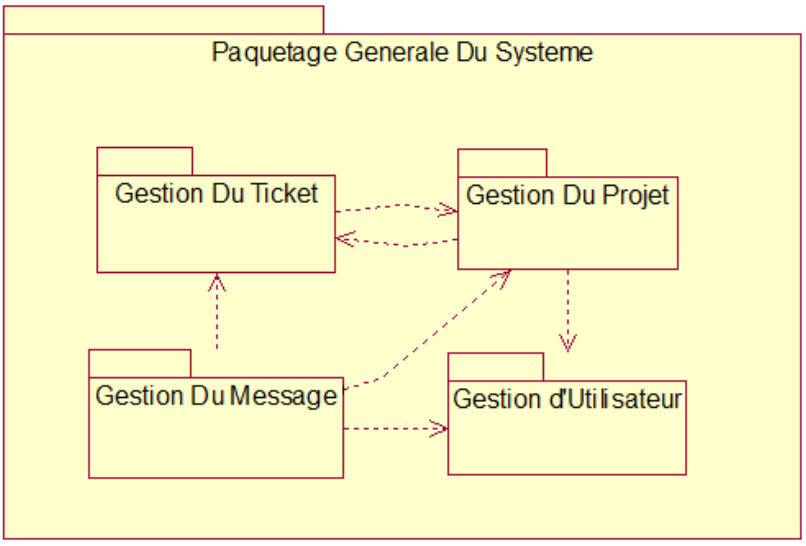

Figure 2.1.1
Diagramme générale de paquetages pour le système

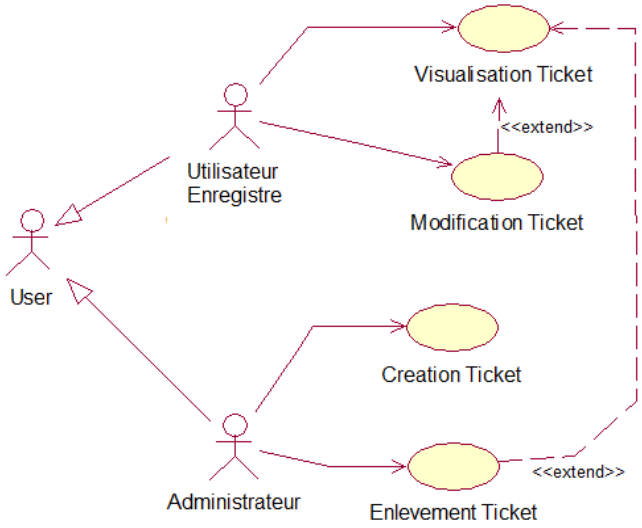

Figure 2.1.2
Diagramme de cas d'utilisation pour le système, du paquet Gestion du Ticket

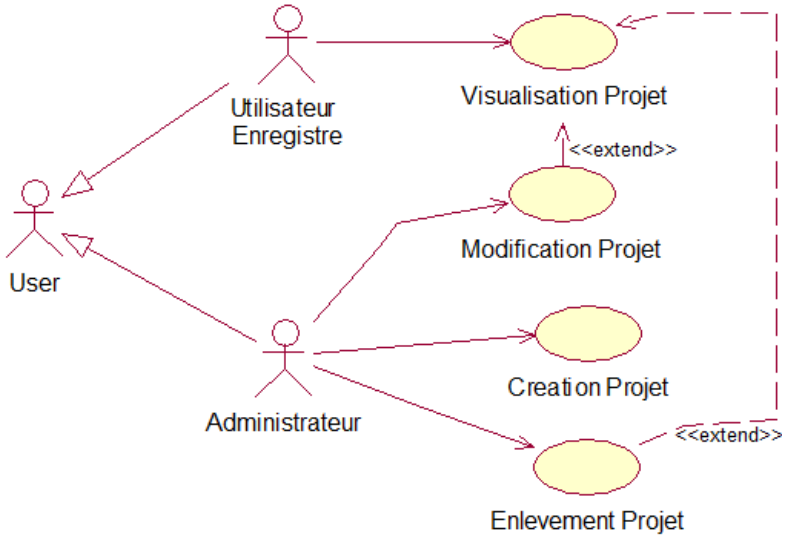

Figure 2.1.3
Diagramme de cas d'utilisation pour le système, du paquet Gestion du Projet

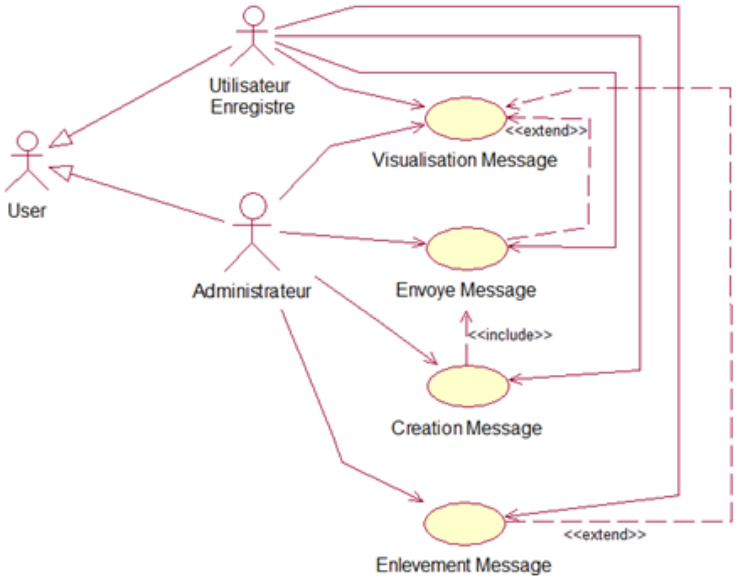

Figure 2.1.4
Diagramme de cas d'utilisation pour le système, du paquet Gestion du Message

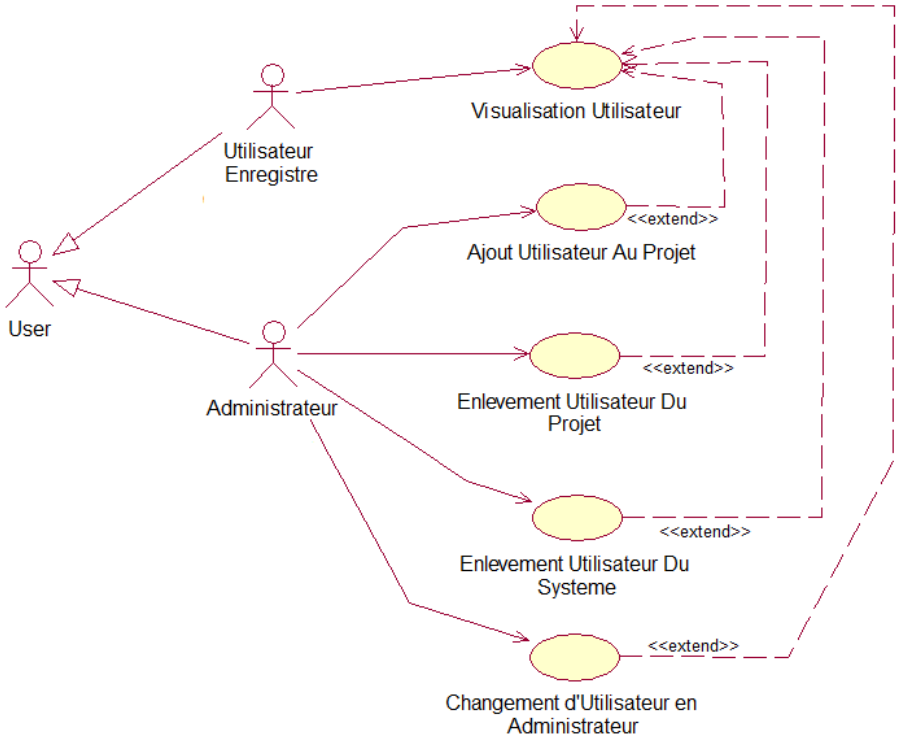

Figure 2.1.5
Diagramme de cas d'utilisation pour le système, du paquet Gestion Utilisateur

2.1.1 Caracteristiques des cas d'utilisation

Parmi les cas d'utilisation, on a les suivants :

Visualisation Ticket, Modification Ticket, Creation Ticket, Enlèvement Ticket, Visualisation Projet, Modification Projet, Creation Projet, Enlèvement Projet, Visualisation Message, Envoye Message, Creation Message, Enlèvement Message, Visualisation Utilisateur, Ajout Utilisateur Au Projet, Enlèvement Utilisateur Du Projet, Enlèvement Utilisateur Du Systeme, Changement d'Utilisateur En Administrateur.

Tableau 2.1.1.1 – Cas d'utilisation Visualisation Ticket

1. Nom de cas d'utilisation	**Visualisation Ticket**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	Administrateur: 1.D'obtenir accès aux informations des tickets; 2.D'obtenir les tickets disponibles du projet dans le système; 3.D'assurer la sécurité des données sur le ticket. Utilisateur: 1.D'obtenir accès aux informations des tickets; 2.D'obtenir les tickets disponibles du projet dans le système.
6. Preconditions	Utilisateur:1.D'avoir déjà créé le ticket; 2.D'avoir déjà créé le projet; 3.D'être ajouté au projet; 4.D'être authentifié dans le système. Administrateur:1.D'avoir déjà créé le ticket; 2.D'avoir déjà créé le projet; 3.D'être administrateur du projet; 4.D'être authentifié dans le système.
7. Postconditions	1.On a obtenu des informations détaillés sur les tickets; 2.On a obtenu le statut du chaque ticket (ouvert/en progress/en code review/finit); 3.On a vu quels utilisateurs travaillent sur le ticket; 4.On a vu qui est l'administrateur du ticket.
8. Scénario de base	1.L'utilisateur demande le système de donner les tickets; 2.L'utilisateur introduit les données du projet; 3.L'utilisateur accepte les informations de recherche introduites; 4.L'utilisateur reçoit les tickets demandés
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas ; 4.Les administrateurs résolvent la panne; 5.L'utilisateur et l'administrateur continuent le processus de la visualisation du ticket.
10. Exigences non fonctionnelles	1.Le temps du réponse avec l'affichage du résultat de la recherche < 2 secondes
11.Technologies et types de données	1.Moteur de recherche
12.Fréquence d'utilisation	Très souvent
13.Autres commentaires/exigences	Les critères d'affichage : par l'identificateur/ projet / administrateur.

Tableau 2.1.1.2 – Cas d'utilisation Modification Ticket

1. Nom de cas d'utilisation	**Modification Ticket**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur et Administrateur

5. State holder et interests	Utilisateur:1. De pouvoir sauvegarder les travaux effectués sur le ticket; 2.De pouvoir effectuer des travaux sur le ticket dans un temps raisonnable; 3.D'avoir la possibilité à changer le statut du ticket. Administrateur: 1.De pouvoir changer les informations sur le ticket; 2.De pouvoir changer le statut du ticket; 3.De pouvoir sauvegarder les modifications sur le ticket .
6. Preconditions	1.D'avoir déjà créé le ticket; 2.D'avoir déjà créé le projet; 3.D'être ajouté au projet; 4.D'avoir acces au ticket demandé; 5.D'être authentifié dans le système
7. Postconditions	1.On a sauvegardé les modifications sur le ticket; 2.On a changé le statut du ticket; 3.On a finit à réaliser le ticket.
8. Scénario de base	Utilisateur: 1.Il demande le système de modifier un/plusieurs ticket/s; 2.Il choisit le ticket; 3.Il applique les travaux effectués sur le ticket; 4.Il change le statut du ticket; 5.Il accepte les modifications; 6.Il sauvegarde le/les tickets ; 7.Le système enregistre les modifications sur le/les ticket/s ; 8.Le système actualise les tickets. Administrateur: Les pas 1 et 2 du cas d'utilisateur ; 3.Il change des informations concernant le/les ticket/s; Les pas de 5 à 8 du cas d'utilisateur.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant dans la base de données; 2.Le système sauvegarde l'état courant sur la machine qui a effectué les modifications; 3.Le système annonce les administrateurs, pour retirer la panne; 4.Le système annonce l'utilisateur que quelque chose ne marche pas ; 5.Les administrateurs résolvent la panne; 6.L'administrateur et l'utilisateur continuent le processus de la modification du projet.
10. Exigences non fonctionnelles	1. Le temps du processus de sauvegarder les modifications < 2 secondes; 2.La transmission de données d'une manière cryptée.
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Régulièrement
13.Autres commentaires/exigences	------

Tableau 2.1.1.3 – Cas d'utilisation Creation Ticket

1. Nom de cas d'utilisation	**Creation Ticket**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur: 1.De pouvoir réaliser des travaux sur le ticket; 2.D'avoir acces aux informations des tickets; 3.D'assurer une bonne qualité des travaux sur le ticket; 4.De satisfaire les exigences du développement.
6. Preconditions	1.D'avoir déjà créé le projet; 2.D'avoir le statut d'administrateur; 3.D'être authentifié dans le système.
7. Postconditions	1.On a créé le ticket; 2.Les utilisateurs ont acces au ticket.
8. Scénario de base	Administrateur: 1.Il demande le système de créer un ticket; 2.Il choisit le projet ou il va créer le ticket; 3.Il introduit les informations sur le ticket; 4.Il accepte les informations introduites; 5.Il voit le ticket créé.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant dans la base de données; 2.Le système sauvegarde l'état courant sur la machine qui a effectué la création;

	3.Le système annonce les administrateurs, pour retirer la panne; 4.Le système annonce l'utilisateur que quelque chose ne marche pas; 5.Les administrateurs résolvent la panne; 6.L'administrateur continue le processus de la création du ticket .
10. Exigences non fonctionnelles	1. Le temps du processus de sauvegarder le ticket créé < 3 secondes; 2.La transmission de données d'une manière cryptée.
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Régulièrement
13.Autres commentaires/exigences	------

Tableau 2.1.1.4 – Cas d'utilisation Enlèvement Ticket

1. Nom de cas d'utilisation	**Enlèvement Ticket**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur: 1.D'avoir moins de tickets à réaliser; 2.De pouvoir s'orienter parmi les tickets (le nombre réduit de tickets).
6. Preconditions	1.D'avoir déjà créé le ticket; 2.D'avoir déjà fini le ticket; 3.D'être authentifié dans le système; 4.D'être en rôle d'administrateur.
7. Postconditions	1.On a enlevé le ticket; 2.On ne voit pas le ticket enlevé.
8. Scénario de base	Administrateur: 1.Il demande le système d'enlever un ticket; 2.Il choisit le ticket à enlever; 3.Il confirme l'enlèvement du ticket choisit.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant dans la base de données; 2.Le système sauvegarde l'état courant sur la machine qui a effectué l'enlèvement; 3.Le système annonce les administrateurs, pour retirer la panne; 4.Le système annonce l'utilisateur que quelque chose ne marche pas; 5.Les administrateurs résolvent la panne; 6.L'administrateur continue le processus de l'enlèvement.
10. Exigences non fonctionnelles	1. Le temps du réponse pour l'enlèvement du ticket < 2 secondes; 2.La transmission de données d'une manière cryptée.
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	------

Tableau 2.1.1.5 – Cas d'utilisation Visualisation Projet

1. Nom de cas d'utilisation	**Visualisation Projet**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	Administrateur: 1.D'obtenir accès aux informations des projets; 2.D'assurer la sécurité des données sur le projet; 3.D'obtenir les projets disponibles du système. Utilisateur: 1.D'obtenir accès aux informations des projets.
6. Preconditions	Utilisateur: 1.D'avoir déjà créé le projet; 2.D'être ajouté au projet; 4.D'être

	authentifié dans le système. Administrateur: 1.D'avoir déjà créé le projet; 2.D'être authentifié dans le système; 3.D' être administrateur du projet
7. Postconditions	1.On a obtenu des informations détaillés sur le/les projet/s; 2.On a obtenu le statut du chaque projet (ouvert/en progress/en code review/finit); 3.On a vu quels utilisateurs travaillent sur le projet; 4.On a vu qui est l'administrateur du projet.
8. Scénario de base	Utilisateur: 1.Il demande le système de donner les projets; 2.Il introduit les données du projet; 3.L'utilisateur accepte les informations de recherche introduites; 4.L'utilisateur reçoit le/les projet/s demandés.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas; 4.Les administrateurs résolvent la panne; 5.Les utilisateurs et les administrateurs continuent le processus de la visualisation.
10. Exigences non fonctionnelles	1.Le temps du réponse avec l'affichage du résultat de la recherche < 2 secondes
11.Technologies et types de données	1.Moteur de recherche
12.Fréquence d'utilisation	Très souvent
13.Autres commentaires/exigences	Les critères d'affichage : par l'identificateur/ nom / administrateur.

Tableau 2.1.1.6 – Cas d'utilisation Modification Projet

1. Nom de cas d'utilisation	**Modification Projet**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur:1. D'avoir plus de détailles sur le projet; 2.De savoir l'administrateur du projet; 3.De réaliser le projet en temps raisonnable.
6. Preconditions	1.D'avoir déjà créé le projet; 2.D'être administrateur du projet; 3.D'être authentifié dans le système.
7. Postconditions	1.On a sauvegardé les modifications sur le projet; 2.On a changé le statut du projet.
8. Scénario de base	Administrateur: 1.Il demande le système de modifier les projets; 2.Il choisit le projet; 3.Il applique les travaux effectués sur le projet; 4.Il change le statut du projet; 5.Il accepte les modifications; 6.Il sauvegarde le projet; 7.Le système enregistre les modifications sur le projet ; 8.Le système actualise les projets.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant dans la base de données; 2.Le système sauvegarde l'état courant sur la machine qui a effectué les modifications; 3.Le système annonce les administrateurs, pour retirer la panne; 4.Le système annonce l'utilisateur que quelque chose ne marche pas ; 5.Les administrateurs résolvent la panne; 6.L'administrateur continue le processus de la modification du projet.
10. Exigences non fonctionnelles	1. Le temps du processus de sauvegarder les modifications < 2 secondes; 2.La transmission de données d'une manière cryptée.
11.Technologies et types de données	1.Données cryptées

12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	------

Tableau 2.1.1.7 – Cas d'utilisation Creation Projet

1. Nom de cas d'utilisation	**Creation Projet**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur: 1.De pouvoir réaliser des travaux sur le projet; 2.D'avoir acces aux informations du projet; 3.D'assurer une bonne qualité des travaux sur le projet; 4.De satisfaire les exigences du développement.
6. Preconditions	1.D'avoir le statut d'administrateur; 2.D'être authentifié dans le système.
7. Postconditions	1.On a créé le projet; 2.Les utilisateurs ont acces au projet.
8. Scénario de base	Administrateur: 1.Il demande le système de créer un projet; 2.Il introduit les informations sur le projet; 3.Il accepte les informations introduites; 5.Il voit le projet créé.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant dans la base de données; 2.Le système sauvegarde l'état courant sur la machine qui a effectué la création; 3.Le système annonce les administrateurs, pour retirer la panne; 4.Le système annonce l'utilisateur que quelque chose ne marche pas; 5.Les administrateurs résolvent la panne; 6.L'administrateur continue le processus de la création du projet .
10. Exigences non fonctionnelles	1. Le temps du processus de sauvegarder le projet < 2 secondes; 2.La transmission de données d'une manière cryptée.
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	------

Tableau 2.1.1.8 – Cas d'utilisation Enlèvement Projet

1. Nom de cas d'utilisation	**Enlèvement Projet**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur: 1.D'avoir moins de projets à réaliser; 2.De pouvoir s'orienter parmi les projets (le nombre réduit de projets).
6. Preconditions	1.D'avoir déjà créé le projet; 2.D'avoir déjà fini tous le tickets du projet; 3.D'être authentifié dans le système; 4.D'être en rôle d'administrateur.
7. Postconditions	1.On a enlevé le projet; 2.On ne voit pas le projet enlevé.
8. Scénario de base	Administrateur: 1.Il demande le système d'enlever un projet; 2.Il choisit le projet à enlever; 3.Il confirme l'enlèvement du projet choisit.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant dans la base de données; 2.Le système sauvegarde l'état courant sur la machine qui a effectué l'enlèvement; 3.Le système annonce les administrateurs, pour retirer la panne; 4.Le système annonce l'utilisateur que quelque chose ne marche pas;

	5.Les administrateurs résolvent la panne; 6.L'administrateur continue le processus de l'enlèvement.
10. Exigences non fonctionnelles	1. Le temps du réponse pour l'enlèvement du projet < 2 secondes; 2.La transmission de données d'une manière cryptée.
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Pas souvent
13.Autres commentaires/exigences	------

Tableau 2.1.1.9 – Cas d'utilisation Visualisation Message

1. Nom de cas d'utilisation	**Visualisation Message**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	Administrateur: 1.D'obtenir accès aux messages; 2.D'obtenir accès aux opinions, difficultés que les utilisateurs puissent avoir sur les projets ou sur le tickets; 3.D' avoir la possibilité de communiquer avec les utilisateurs et les autres administrateurs. Utilisateur: 1.D'obtenir accès aux messages; 2.D'obtenir accès aux détailles sur le projet et/ou le/les ticket/s; 3.D'avoir la possibilité de communiquer avec les administrateurs et les autres utilisateurs.
6. Preconditions	Utilisateur:1.D'avoir déjà créé le/les message/s; 2.D'avoir déjà créé le projet; 3.D'être ajouté au projet; 4.D'être authentifié dans le système. Administrateur:1.D'avoir déjà créé le/les message/s; 2.D'avoir déjà créé le projet; 3.D'être authentifié dans le système.
7. Postconditions	1.On a vu le/les message/s.
8. Scénario de base	Utilisateur: 1.Il demande le système pour donner à visualiser les messages; 2.Il choisit le message à voir; 3.Il voit le message demandé. Administrateur: 1.Il demande le système pour donner à visualiser les messages; 2.Il choisit le message à voir; 3.Il voit le message demandé.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas ; 4.Les administrateurs résolvent la panne; 5.L'utilisateur et l'administrateur continuent le processus de la visualisation des messages.
10. Exigences non fonctionnelles	1.Le temps du réponse avec l'affichage du message < 2 secondes
11.Technologies et types de données	Serveur de couriel électronique; Client de couriel électronique
12.Fréquence d'utilisation	Très souvent
13.Autres commentaires/exigences	Les critères d'affichage : par la date de la reception/ par l'utilisateur(administrateur).

Tableau 2.1.1.10 – Cas d'utilisation Envoye Message

1. Nom de cas d'utilisation	**Envoye Message**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	Administrateur: 1.D'informer les utilisateurs sur quelque chose qui tient de projet ou de ticket; 2.De communiquer avec les utilisateurs. Utilisateur: 1.D'informer l'administrateur ou les autres utilisateurs concernant le projet ou le/les ticket/s; 2. De communiquer avec les administrateurs ou avec les autres utilisateurs.
6. Preconditions	1.D'avoir déjà créé le message; 2.D'avoir déjà choisit l'utilisateur et/ou l'administrateur; 3.D'être authentifié dans le système.
7. Postconditions	1.L'autre utilisateur et/ou administrateur a reçu le message.
8. Scénario de base	Utilisateur et Administrateur: 1.Il demande le système pour envoyer le message; 2.Il accepte la transmission du message; 3.Il voit le réponse du système que le message a été transmit.
9. Scénarios alternatifs	On a perdu la liaison avec le serveur du couriel électronique: 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas ; 4.Les administrateurs résolvent la panne; 5.L'utilisateur et l'administrateur continuent le processus de l'envoye du message.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat de la transmission < 2 secondes
11.Technologies et types de données	Serveur de couriel électronique; Client de couriel électronique
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	Utiliser le protocole SMTP

Tableau 2.1.1.11 – Cas d'utilisation Creation Message

1. Nom de cas d'utilisation	**Creation Message**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	Administrateur: 1.D'informer les utilisateurs sur quelque chose qui tient de projet ou de ticket; 2.De communiquer avec les utilisateurs. Utilisateur: 1.D'informer l'administrateur ou les autres utilisateurs concernant le projet ou le/les ticket/s; 2. De communiquer avec les administrateurs ou avec les autres utilisateurs.
6. Preconditions	1.D'être authentifié dans le système.
7. Postconditions	1.Le message est prêt d'être envoyé.
8. Scénario de base	Utilisateur et Administrateur: 1.Il demande le système pour créer un message; 2.Il introduit le destinataire, le thème, le sujet et, s'il est nécessaire, une pièce jointe; 3.Il accepte les informations introduites. accepte la transmission du message; 4.Il voit le réponse du système que le message a été sauvegardé.
9. Scénarios alternatifs	On a perdu la liaison avec le serveur du couriel électronique: 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas ; 4.Les administrateurs résolvent la panne;

	5.L'utilisateur et l'administrateur continuent le processus de la création du message.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat de la création < 2 secondes
11.Technologies et types de données	Serveur de couriel électronique; Client de couriel électronique
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	----------

Tableau 2.1.1.12 – Cas d'utilisation Enlèvement Message

1. Nom de cas d'utilisation	**Enlèvement Message**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	1.De libérer de l'éspace totale pour les nouvelles sauvegardes.
6. Preconditions	1.D'être authentifié dans le système; 2.D'avoir le message à enlever.
7. Postconditions	1.Le message est enlevé; 2. On a plus de l'éspace totale pour les nouvelles sauvegardes.
8. Scénario de base	Utilisateur et Administrateur: 1.Il demande le système pour donner les messages; 2.Il reçoit les messages qui lui appartient; 3.Il choisit le message à enlever; 4.Il accepte l'enlèvement du message; 5.Il reçoit la notification qui lui de l'enlèvement du message choisit.
9. Scénarios alternatifs	On a perdu la liaison avec le serveur du couriel électronique: 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas ; 4.Les administrateurs résolvent la panne; 5.L'utilisateur et l'administrateur continuent le processus de l'enlèvement du message.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat de l'enlèvement < 2 secondes
11.Technologies et types de données	Serveur de couriel électronique; Client de couriel électronique
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	----------

Tableau 2.1.1.13 – Cas d'utilisation Visualisation Utilisateur

1. Nom de cas d'utilisation	**Visualisation Utilisateur**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Utilisateur, Administrateur
5. State holder et interests	Administrateur et Utilisateur: 1.D'obtenir accès aux informations des utilisateurs.
6. Preconditions	Utilisateur: 1.D'avoir déjà créé le projet; 2.D'être ajouté au projet; 3.D'être authentifié dans le système. Administrateur: 1.D'avoir déjà créé le projet; 2.D'être authentifié dans le système; 3.D'être administrateur du projet
7. Postconditions	1.On a obtenu des informations détaillés concernant l'utilisateur; 2.On a vu

	sur quel projet travail l'utilisateur; 3. En cas d'utilisateur, on a obtenu les informations concernant du fait qui est administrateur du projet.
8. Scénario de base	Utilisateur et Administrateur: 1.Il demande le système de donner les utilisateurs du projet; 2.Il choisit l'utilisateur à visualiser; 3.Il voit l'utilisateur choisit.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas; 4.Les administrateurs résolvent la panne; 5.Les utilisateurs et les administrateurs continuent le processus de la visualisation des utilisateurs.
10. Exigences non fonctionnelles	1.Le temps du réponse avec l'affichage du résultat de la recherche < 2 secondes
11.Technologies et types de données	---------
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	---------

Tableau 2.1.1.14 – Cas d'utilisation Ajout Utilisateur Au Projet

1. Nom de cas d'utilisation	**Ajout Utilisateur Au Projet**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur: 1.D'avoir accès aux tickets; 2.De pouvoir réaliser les tickets.
6. Preconditions	1.D'avoir déjà créé le projet; 2.D'avoir des utilisateurs dans le système; 3.D'être authentifié dans le système; 4.D'être administrateur du projet.
7. Postconditions	1.L'utilisateur a obtenu accès au projet; 2.L'utilisateur peut réaliser les tickets du projet; 3.L'utilisateur peut voir les tickets du projet.
8. Scénario de base	Administrateur : 1.Il demande le système de donner à visualiser les utilisateurs; 2.Il choisit l'utilisateur; 3.Il ajout l'utilisateur au projet; 4.Il accepte les modifications faites ; 5.L'utilisateur reçoit une notification, qu'il a été ajouté au projet respectif.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas; 4.Les administrateurs résolvent la panne; 5.L'administrateur continuent le processus de l'ajout d'utilisateur au projet.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat de l'ajout d'utilisateur au projet < 2 secondes
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	---------

Tableau 2.1.1.15 – Cas d'utilisation Enlèvement Utilisateur Du Projet

1. Nom de cas d'utilisation	**Enlèvement Utilisateur Du Projet**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Utilisateur: 1.D'avoir moins de projet à réaliser; 2.De pouvoir s'orienter parmi les projets (le nombre réduit de projets).
6. Preconditions	1.D'avoir déjà créé le projet; 2.D'avoir des utilisateurs dans le système; 3.D'avoir ajouté l'utilisateur au projet; 4.D'être authentifié dans le système; 5.D'être administrateur du projet.
7. Postconditions	1.L'utilisateur a perdu la possibilité d'avoir accès au projet; 2.L'utilisateur ne voit pas voir les tickets; 3.L'utilisateur ne peut pas réaliser les tickets du projet dont il a été enlevé.
8. Scénario de base	Administrateur : 1.Il demande le système de donner à visualiser les projets; 2.Il choisit le projet; 3.Il demande le système de lui donner les utilisateurs du projet; 4.Il choisit l'utilisateur du projet; 5.Il enlève l'utilisateur du projet; 6.Il accepte l'enlevement de l'utilisateur.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas; 4.Les administrateurs résolvent la panne; 5.L'administrateur continuent le processus de l'enlevement d'utilisateur du projet.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat de l'enlevement d'utilisateur du projet < 2 secondes
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	---------

Tableau 2.1.1.16 – Cas d'utilisation Enlèvement Utilisateur Du Systeme

1. Nom de cas d'utilisation	**Enlèvement Utilisateur Du Systeme**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Administrateur: 1.D'assurer la sécurité du système
6. Preconditions	1.D'avoir l'utilisateur dans le système; 2.D'avoir enlevé l'utilisateur du projet.
7. Postconditions	1.L'utilisateur n'a pas accès dans le système.
8. Scénario de base	Administrateur : 1.Il demande le système de donner à visualiser les projets; 2.Il choisit le projet; 3.Il demande le système de lui donner les utilisateurs du projet; 4.Il choisit l'utilisateur du projet; 5.Il enlève l'utilisateur du projet; 6.Il accepte l'enlevement de l'utilisateur du projet ; 7.Il demande le système d'enlever les utilisateurs; 8.Il choisit l'utilisateur a enlever; 9.Il confirme l'enlevation de l'utilisateur. 10.Le système actualise les utilisateurs.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas; 4.Les administrateurs résolvent la panne;

	5.L'administrateur continuent le processus de l'enlevement d'utilisateur du système.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat de l'enlevement d'utilisateur du système < 2 secondes
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	---------

Tableau 2.1.1.17 – Cas d'utilisation Changement d'Utilisateur En Administrateur

1. Nom de cas d'utilisation	**Changement d'Utilisateur En Administrateur**
2. Scope	Gestion desprojets à l'aide de tickets
3. Niveau	Role utilisateur
4. Acteur Principal	Administrateur
5. State holder et interests	Administrateur: 1.D'assurer la sécurité du système; 2.De donner plein de droits sur les projets et les tickets; 3.D'assurer une meilleure qualité des projets et la réalisation des tickets en temps réduit.
6. Preconditions	1.D'avoir l'utilisateur dans le système.
7. Postconditions	1.L'utilisateur est changé dans un administrateur.
8. Scénario de base	Administrateur : 1.Il demande le système de visualiser les utilisateurs; 2.Il choisit un utilisateur; 3.Il change le rôle d'utilisateur en administrateur; 4.Il confirme le changement; 5.Le système actualise les utilisateurs.
9. Scénarios alternatifs	On a perdu la liaison avec la base de données : 1.Le système sauvegarde l'état courant; 2.Le système annonce les administrateurs, pour retirer la panne; 3.Le système annonce l'utilisateur que quelque chose ne marche pas; 4.Les administrateurs résolvent la panne; 5.L'administrateur continuent le processus de l'enlevement d'utilisateur du système.
10. Exigences non fonctionnelles	1.Le temps du réponse avec le résultat du changement de l'utilisateur en administrateur < 2 secondes
11.Technologies et types de données	1.Données cryptées
12.Fréquence d'utilisation	Moyen
13.Autres commentaires/exigences	---------

2.2 Diagrammes SSD

En cadre de ce type de diagrammes, on aura les acteurs et leur intéraction avec le système.

1. Creation Projet

Le diagramme suivant va représenter le SSD pour le cas d'utilisation Creation Projet

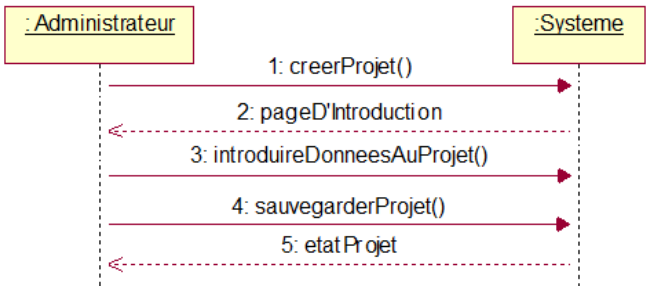

Figure 2.2.1.1
Diagramme SSD pour le cas d'utilisation Creation Projet

Tableau 2.2.1.1 – Opération creerProjet()

Nom de l'opération	creerProjet()
Renvoi (cross-référence)	Creation Projet
Pré-conditions	---
Post-conditions	1.A été créé un projet 2. Projet a été instancié avec : idProjet, nomProjet, domain, dateCreation 3. Projet a été associé avec Base De Donnees

Tableau 2.2.1.2 – Opération introduireDonneesAuProjet()

Nom de l'opération	introduireDonneesAuProjet()
Renvoi (cross-référence)	Creation Projet
Pré-conditions	1. Il faut qu'il existe le projet en cours de création
Post-conditions	1.A été créé l'instance de CreationProjet 2.CreationProjet a été associé avec AdministrateurD et avec FichierRegistre.

Tableau 2.2.1.3 – Opération sauvegarderProjet()

Nom de l'opération	sauvegarderProjet()
Renvoi (cross-référence)	Creation Projet
Pré-conditions	1. Il faut qu'il existe le projet en cours de création 2.Il faut que l'instance de CreationProjet soit associée avec Administrateur et avec FichierRegistre
Post-conditions	1.A été sauvegardé le projet 2.A été créé l'instance de ProjetD

2. Modification Ticket

Le diagramme suivant va représenter le SSD pour le cas d'utilisation Modification Ticket

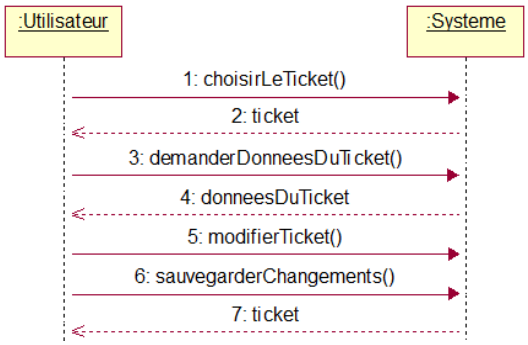

Figure 2.2.1.2
Diagramme SSD pour le cas d'utilisation Modification Ticket

Tableau 2.2.2.1 – Opérations choisirLeTicket() et demanderDonneesDuTicket()

Nom de l'opération	choisirLeTicket()	demanderDonneesDuTicket()
Renvoi (cross-référence)	Modification Ticket	Modification Ticket
Pré-conditions	1. Il existe l'instance de Ticket	1. Il existe l'instance de Ticket
Post-conditions	----------	1. On voit l'instance de Ticket

Tableau 2.2.2.2 – Opérations modifierTicket() et sauvegarderChangements()

Nom de l'opération	modifierTicket()	sauvegarderChangements()
Renvoi (cross-référence)	Modification Ticket	Modification Ticket
Pré-conditions	1. Il existe l'instance de Ticket	1. Il existe l'instance de Ticket 2. Les attributs sont remplient
Post-conditions	1. Les attributs du Ticket sont changés	1. Les attributs du Ticket sont sauvegadrés 2. On a créé une nouvelle instance de Ticket

3. Ajout Utilisateur Au Projet

Le diagramme suivant va représenter le SSD pour le cas d'utilisation Ajout Utilisateur Au Projet

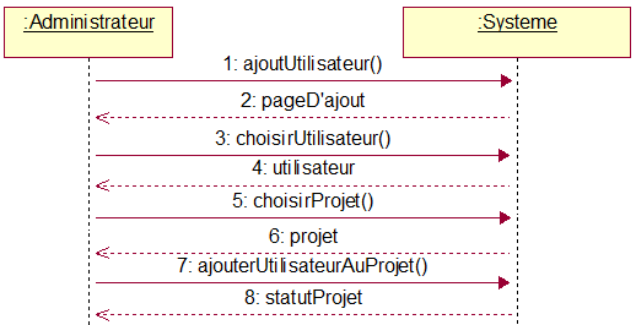

Figure 2.2.1.3
Diagramme SSD pour le cas d'utilisation Ajout Utilisateur Au Projet

Tableau 2.2.3.1 – Opérations ajoutUtilisateur() et choisirUtilisateur()

Nom de l'opération	ajoutUtilisateur()	choisirUtilisateur()
Renvoi (cross-référence)	Ajout Utilisateur Au Projet	Ajout Utilisateur Au Projet
Pré-conditions	1. Il existe l'instance d'Utilisateur	1. Il existe l'instance d'Utilisateur
Post-conditions	----------	1. On voit l'instance d'Utilisateur 2. Les attributs d'Utilisateur restent les mêmes

Tableau 2.2.3.2 – Opérations choisirProjet() et ajouterUtilisateurAuProjet()

Nom de l'opération	choisirProjet()	ajouterUtilisateurAuProjet()
Renvoi (cross-référence)	Ajout Utilisateur Au Projet	Ajout Utilisateur Au Projet
Pré-conditions	1. Il existe l'instance du Projet	1. Il existe l'instance d'Utilisateur 2. Il existe l'instance du Projet 3. Il faut avoir le Projet en cours
Post-conditions	1. On voit l'instance du Projet 2.Les attributs du Projet restent les mêmes	1.Utilisateur a été associé avec Administrateur et Projet.

4. Changement D'Utilisateur En Administrateur

Le diagramme suivant va représenter le SSD pour le cas d'utilisation Changement D'Utilisateur En Administrateur

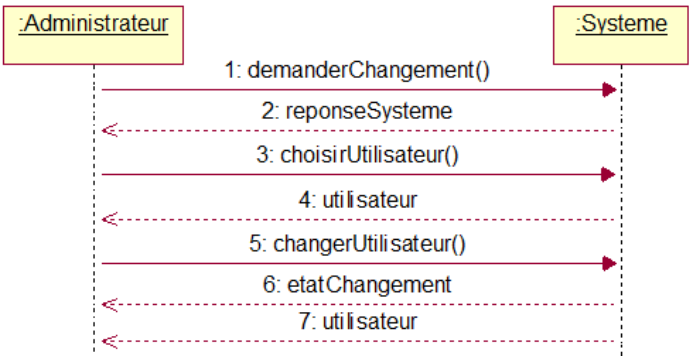

Figure 2.2.1.4
Diagramme SSD pour le cas d'utilisation Changement D'Utilisateur En Administrateur

Tableau 2.2.4.1 – Opération demaderChangement()

Nom de l'opération	demanderChangement()
Renvoi (cross-référence)	Changement D'Utilisateur En Administrateur
Pré-conditions	1. Il existe l'instance d'Utilisateur
Post-conditions	---------

Tableau 2.2.4.2 – Opération choisirUtilisateur()

Nom de l'opération	choisirUtilisateur()
Renvoi (cross-référence)	Changement D'Utilisateur En Administrateur
Pré-conditions	1. Il existe l'instance d'Utilisateur
Post-conditions	1. On voit l'instance d'Utilisateur

Tableau 2.2.4.3 – Opération changerUtilisateur()

Nom de l'opération	changerUtilisateur()
Renvoi (cross-référence)	Changement D'Utilisateur En Administrateur
Pré-conditions	1. Il existe l'instance d'Utilisateur 2. L'attribut isAdmin a le valeur False
Post-conditions	1. L'attribut isAdmin a été changé en True 2. A été créé l'instance Administrateur

5. Enlevement D'Utilisateur Du Projet

Le diagramme suivant va représenter le SSD pour le cas d'utilisation Enlevement D'Utilisateur Du Projet

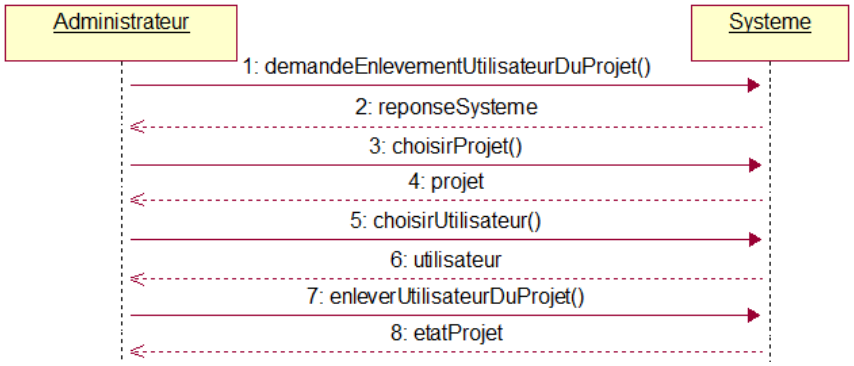

Figure 2.2.1.5
Diagramme SSD pour le cas d'utilisation Enlevement D'Utilisateur Du Projet

Tableau 2.2.5.1 – Opération demandeEnlevementUtilisateurDuProjet()

Nom de l'opération	demandeEnlevementUtilisateurDuProjet()
Renvoi (cross-référence)	Enlevement D'Utilisateur Du Projet
Pré-conditions	1. Il existe l'instance d'Utilisateur 2. Il existe l'instance du Projet
Post-conditions	---------

Tableau 2.2.5.2 – Opération choisirProjet()

Nom de l'opération	choisirProjet()
Renvoi (cross-référence)	Enlevement D'Utilisateur Du Projet
Pré-conditions	1. Il existe l'instance du Projet
Post-conditions	1. On voit l'instance du Projet

Tableau 2.2.5.3 – Opération choisirUtilisateur()

Nom de l'opération	choisirUtilisateur()
Renvoi (cross-référence)	Enlevement D'Utilisateur Du Projet
Pré-conditions	1. Il existe l'instance d'Utilisateur
Post-conditions	1. On voit l'instance d'Utilisateur

Nom de l'opération	demanderLaCreationDuMessage()
Renvoi (cross-référence)	Creation Message
Pré-conditions	----------
Post-conditions	1.A été créé un message 2. Message a été instancié avec : idMessage, sujet, domain, corps, destinataire 3. Message a été associé avec Utilisateur et CreationMessage

Tableau 2.2.5.4 – Opération enleverUtilisateurDuProjet()

Nom de l'opération	enleverUtilisateurDuProjet()
Renvoi (cross-référence)	Enlevement D'Utilisateur Du Projet
Pré-conditions	1. Il existe l'instance d'Utilisateur 2. Il exite l'instance du Projet 3. L'attribut isAdmin a le valeur False
Post-conditions	----------

6. Creation Message

Le diagramme suivant va représenter le SSD pour le cas d'utilisation Creation Message

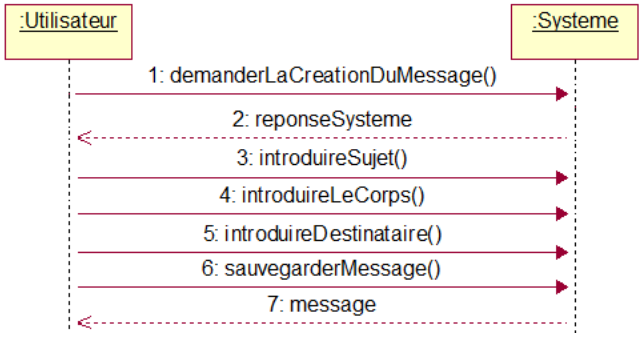

Figure 2.2.1.6
Diagramme SSD pour le cas d'utilisation Creation Message

Tableau 2.2.6.1 – Opération introduireSujet()

Nom de l'opération	introduireSujet()
Renvoi (cross-référence)	Creation Message
Pré-conditions	1. Il faut qu'il existe un message en cours de création
Post-conditions	1. Message a été instancié avec : sujet 2. Message a été associé avec Utilisateur et CreationMessage

Tableau 2.2.6.2 – Opération introduireLeCorps()

Nom de l'opération	introduireLeCorps()
Renvoi (cross-référence)	Creation Message
Pré-conditions	1. Il faut qu'il existe un message en cours de création
Post-conditions	1. Message a été instancié avec : corps 2. Message a été associé avec Utilisateur et CreationMessage

Tableau 2.2.6.3 – Opération introduireDestinataire()

Nom de l'opération	introduireDestinataire()
Renvoi (cross-référence)	Creation Message
Pré-conditions	1. Il faut qu'il existe un message en cours de création
Post-conditions	1. Message a été instancié avec : destinataire 2. Message a été associé avec Utilisateur et CreationMessage

Tableau 2.2.6.4 – Opération sauvegarderMessage()

Nom de l'opération	sauvegarderMessage()
Renvoi (cross-référence)	Creation Message
Pré-conditions	1. Il faut qu'il existe un message en cours de création 2. Il faut que Message soit instancié avec tous les attributs
Post-conditions	1. L'attribut du Message.isCompleted = true 2. Message a été associé avec Utilisateur et CreationMessage

2.3 Utilisation des patterns GRASP (General Responsibility Assignment Software Patterns)

Utilisation du pattern ,,Controller''

Le contrôleur est responsable du traitement des événements d'entrée du système, déléguer des responsabilités à leurs classes de manipulation compétentes. Dans le cas général, le contrôleur met en oeuvre une ou plusieurs options d'utilisation.

Il est connu le concept du contrôleur externe (Front Controller), qui représente l'ensemble du système (comporte une fonctionnalité générale du système entier dans la même classe).

En utilisant le contrôleur on peut séparer la logique de la présentation, augmentant ainsi la réutilisation de code.

Problème : Qui détient les responsabilités de la gestion de l'événement système lié de la création d'un projet?

Solution : Nous créons la classe CreationProjetHandler qui est chargé de traiter tous les événements systèmes contenus dans le scénario de cas d'utilisation ,,Creation Projet'.

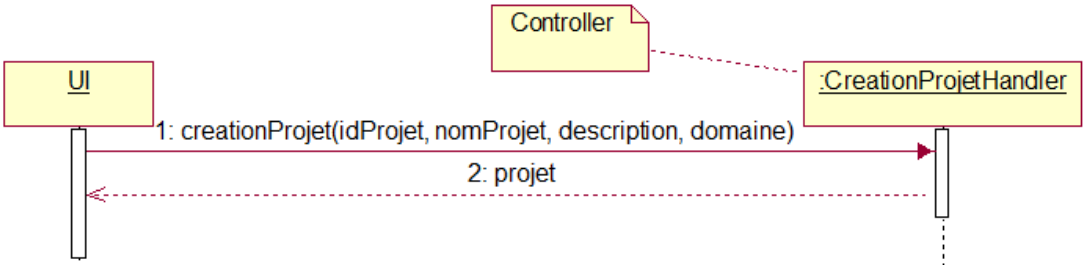

Figure 2.3.1
Diagramme qui représente l'utilité du pattern Controller du GRASP

Utilisation du pattern ,,Créateur''

Créateur ou Fabrication – il sert à la création des autres objets. sont sous la responsabilité d'un tel objet est qu'il crée d'autres objets. Immédiatement frappé par l'analogie avec les usines. Il est donc. Les usines ont aussi la responsabilité est - le Créateur.

Problème : Qui peut avoir la responsabilité de créer une nouvelle instance de la classe Projet?

Solution : Une classe A doit créer des instances de classe B si un (ou plusieurs cas) peut être appliqué dans la situation du logiciel :

1. Instances de A enregistrent des instances de B ;
2. Instances de A utilisent des instances de B ;
3. Instances de A contiennent les données(des infos) d'initialisation pour les instances B

Comme la classe CreationProjetHandler possède les informations données par l'utilisateur pour créer un projet, on va l'utiliser comme la résponsable de la création d'un objet de type Projet. Car la classe CreationProjetHandler est un controller associé à tous les événements système du cas d'utilisation ,,Creation Projet'' et possède les informations nécéssares pour instancier la classe Projet on va l'utiliser comme Createur pour celui-ci.

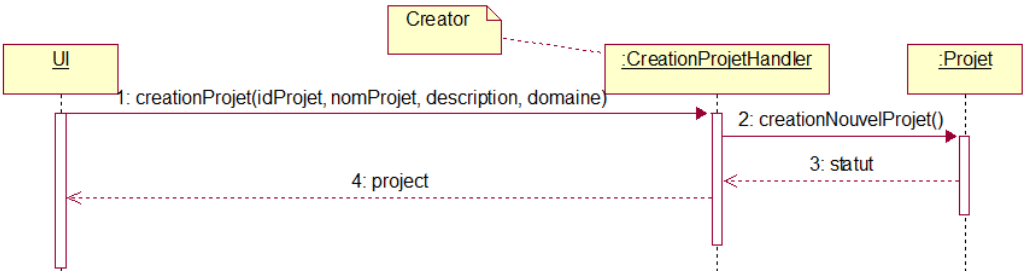

Figure 2.3.2
Diagramme qui représente l'utilité du pattern Creator du GRASP

Utilisation du pattern ,,Pure Fabrication''

Il y a un concept de modèle de programmation pour un domaine, selon lequel chaque entité du domaine correspond à une ou plusieurs classes de l'environnement logiciel. Cette approche présente l'inconvénient évident – une grande liaison des modules du système. Ce pattern sert à resoudre cette problème, par l'introduction dans le milieu du programmation, d'une autre classe en lui donnant les fonctions nécessaires.

Problème : Que faire quand les concepts du monde réel (objets du domaine) ne sont pas utilisables en respectant le Faible couplage et la Forte cohésion ? Qui va s'occuper avec le sauvegardage des données liées d'un projet en respectant le Faible couplage et la Forte cohésion?

La classe Projet n'est pas indiquée pour effectuer des opérations de sauvegardage parcequ'elle sera trop chargée et on ne respecte pas le principe du pattern High cohesion.

Solution : Créer une classe artificielle ou de commodité, qui ne représente pas un concept du domaine qui effectuera les opérations de sauvegardage.

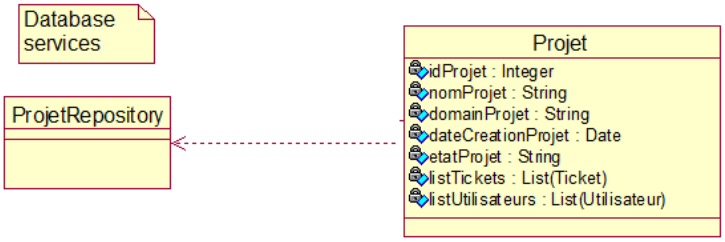

Figure 2.3.3
Diagramme qui représente l'utilité du pattern Pure Fabrication du GRASP

2.4 Diagrammes de composants

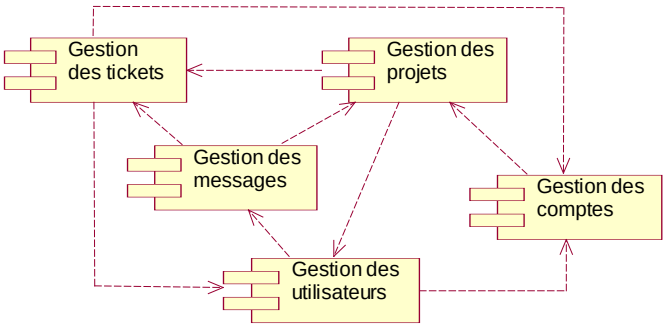

Figure 2.4.1
Diagramme de composantes pour représenter les composantes du système

2.5 Diagrammes d’activité

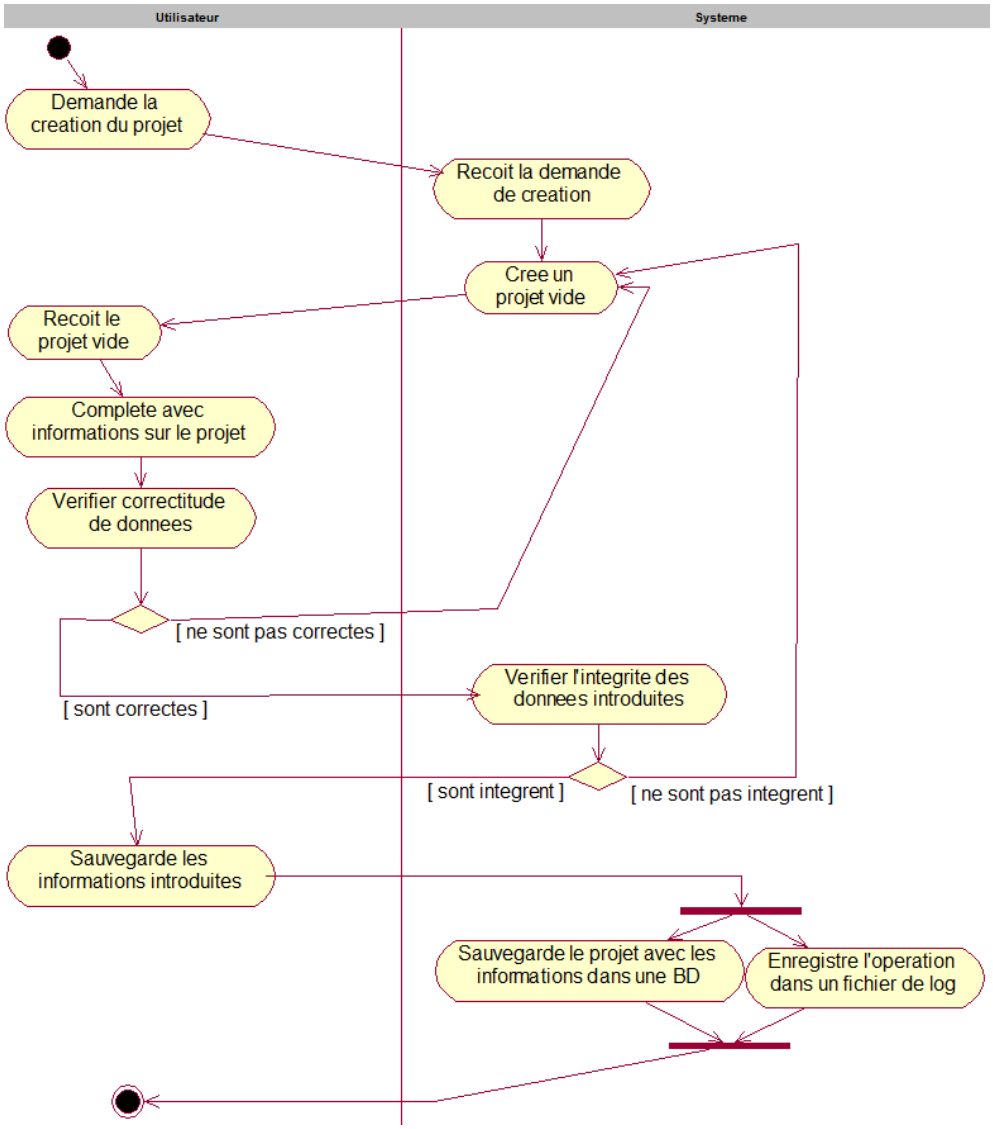

Figure 2.5.1
Diagramme d’activité pour représenter les activités pour la création d’un projet

L’Utilisateur est celui qui commence le flou : il demande la création d’un projet et offre les infomations nécéssaires pour la création d’un projet. Le système va vérifier l’integrité des données introduites par utilisateur, et en fonction du résultat il va sauvegarder les informations sur le projet et enregistrer cette opération de création dans un fichier avec logs.

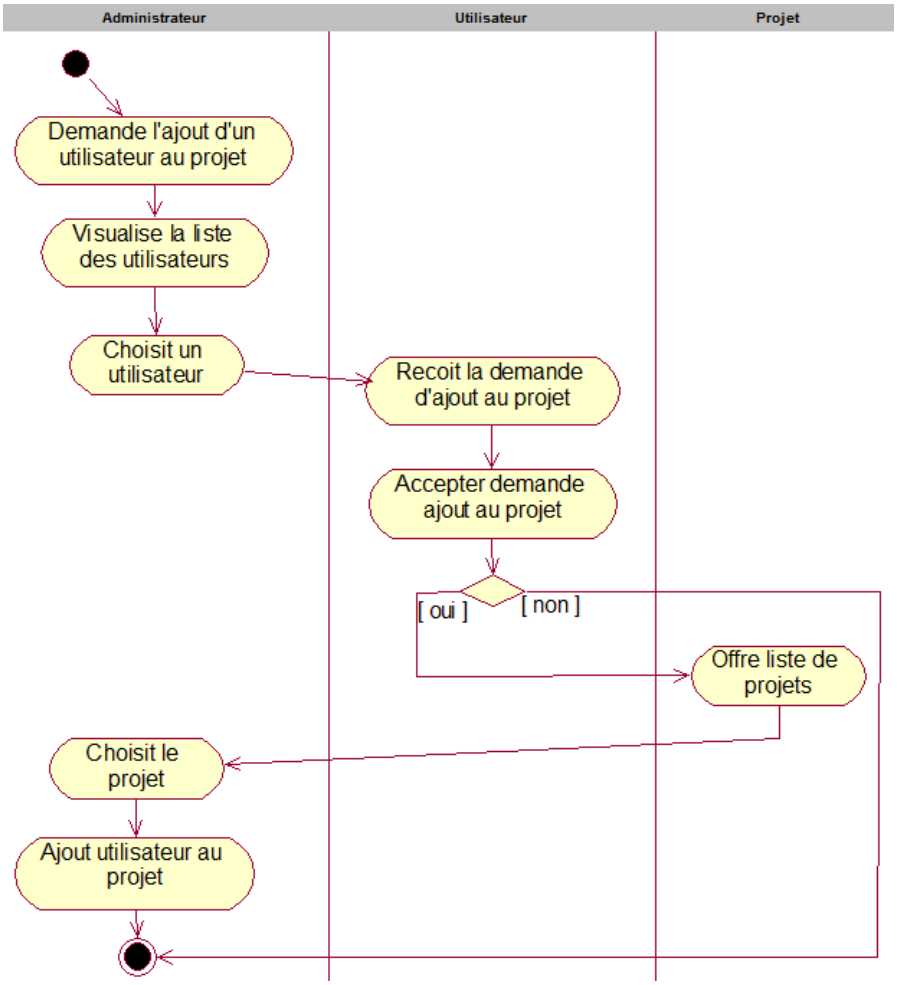

Figure 2.5.2

Diagramme d'activité pour représenter les activités pour l'ajout d'un utilisateur au projet

L'Administrateur est celui qui commence le flou : il demande l'ajout d'un utilisateur au projet en choisissant un utilisateur. L'utilisateur accepte ou pas le demande et puis, en dépendence du résultat donné par l'utilisateur, l'administrateur choisit le projet et ajout l'utilisateur à ce projet.

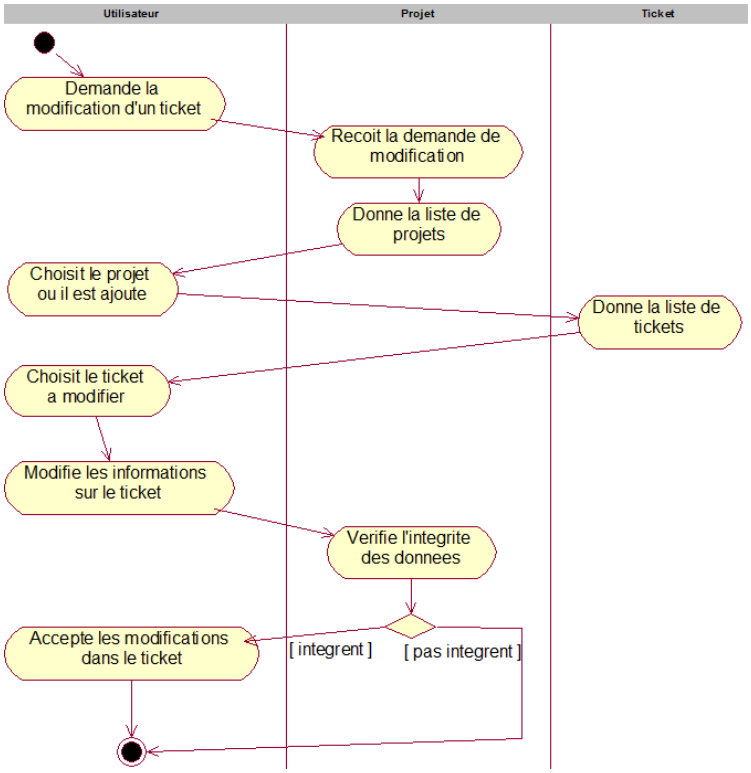

Figure 2.5.3
Diagramme d'activité pour représenter les activités pour la modification d'un ticket

L'Utilisateur est celui qui commence le flou : il demande la modification d'un ticket du projet. Le projet donne la liste à l'utilisateur, puis il choisit le projet. D'une manière pareil, le ticket donnne la liste est l'utilisateur choisit le ticket. Il change les donnees et finalement, accèpte les modifications.

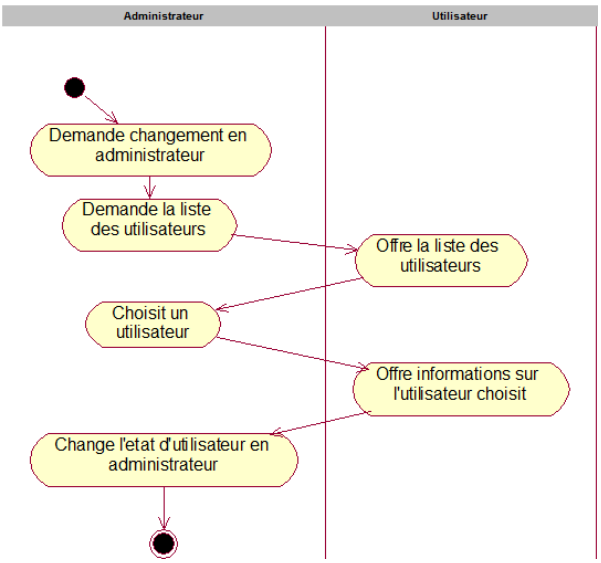

Figure 2.5.4
Diagramme d'activité pour représenter les activités pour le changement d'un utilisateur en administrateur

L'Administrateur est celui qui commence le flou : il demande le changement en administrateur d'un utilisateur. L'utilisateur donne la liste des utilisateurs, puis l'admin choisit l'utilisateur. En offrant les informations sur l'utilisateur choisit, l'administrateur change l'etat d'utilisateur en administrateur.

2.6 Diagrammes d'état

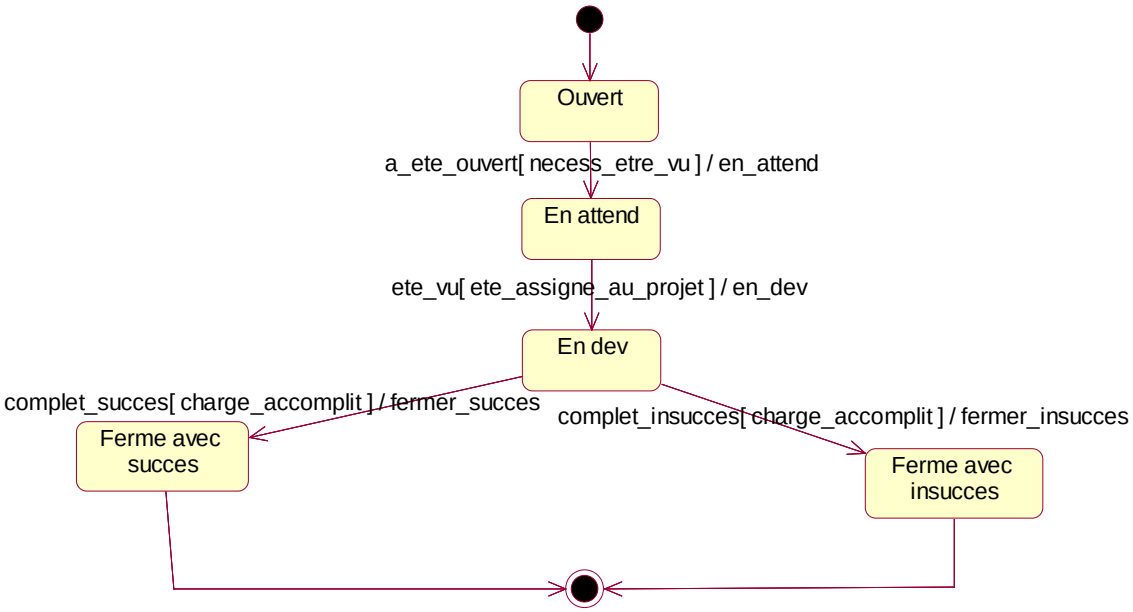

Figure 2.6.1
Diagramme d'état pour représenter les états d'un ticket en cadre du système

Dans ce diagramme d'état on a décrit les états qui peut avoir un ticket. Les états sont les suivants: état initial, Ouvert, En dev, Ferme avec succes, Ferme avec insucces, état final.

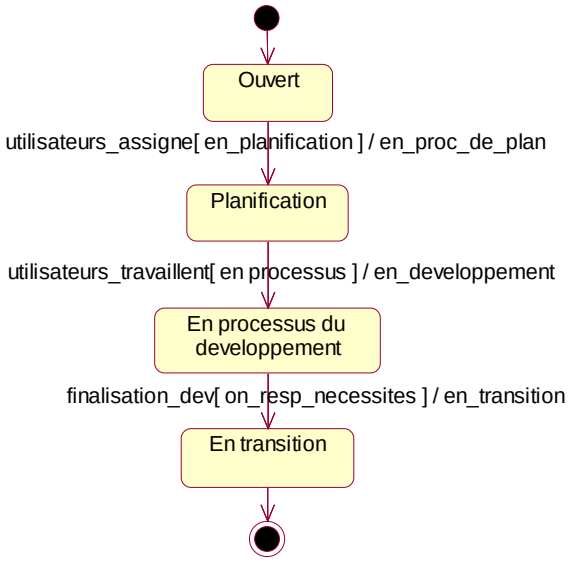

Figure 2.6.2
Diagramme d'état pour représenter les états d'un projet en cadre du système

Dans ce diagramme d'état on a décrit les états qui peut avoir un projet. Les états sont les suivants: état initial, Ouvert, Planification, En processus du developpement, En transition, état final.

2.7 Diagramme de déploiement

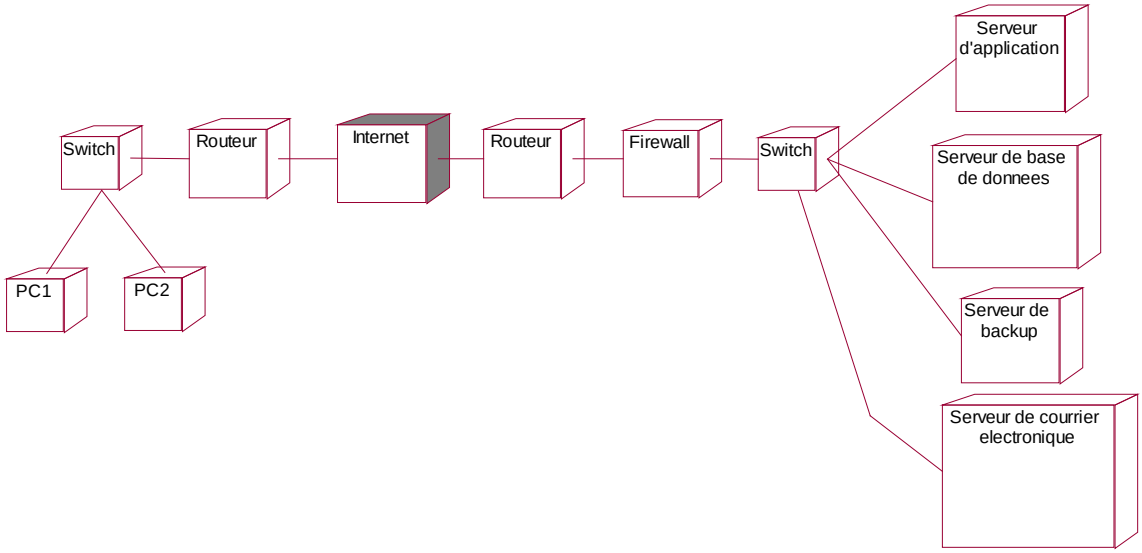

Figure 2.7.1

Diagramme de déploiement pour représenter les équipements qui assurent la fonctionnalité du système

2.8 Conception de l'architecture logique

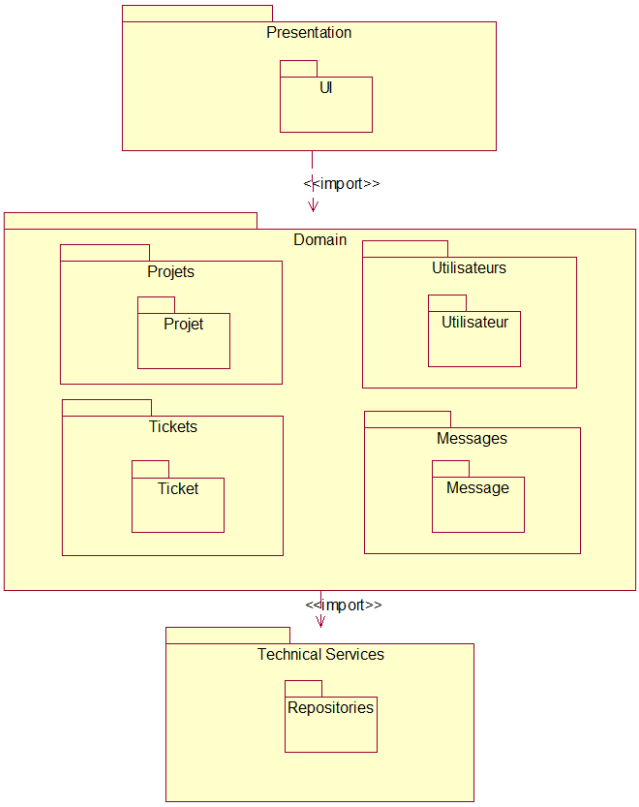

Figure 2.8.1

Diagramme de conception de l'architecture logique qui décrit le système

2.9 Identification d’autres besoins

- Fiabilité

  Le système devra être fiable, en assurant que les transactions soient traitées de façon complète, que, si un problème surgit durant un traitement, les modifications ne soient pas partielles. Il faut aussi s’assurer de l’intégrité de toutes les informations saisies.

- Disponibilité

  Le système devrait fonctionner 90% du temps au moins.

- Sécurité

  Les données transitant par internet doivent être cryptées, surtout les informations de connexion, l’identificateur d’usager et son mot de passe.

- Protection contre intrusions, modification, destruction

A l’instant des serveurs web sur le marché, il faut s’assurer que celui sur lequel fonctionne le site présente les exigences de protection courantes, tel un pare-feu, antivirus, mises à jour du système d’exploitation contre les failles critiques.

3. CONCEPTION DE LA BASE DES DONNÉES

3.1 Modèles de la base des données

Modèle conceptuel des données

Le modèle conceptuel des données (MCD) a pour but d'écrire de façon structurée les données qui seront utilisées par le système d'information. Le modèle conceptuel des données décrit la sémantique c'est à dire le sens attaché à ces données et à leurs rapports et non à l'utilisation qui peut en être faite. On établit le MCD après avoir recensé et donné un nom à l'ensemble des données du domaine étudié. Ensuite on étudie les relations existantes entre ces données (les dépendances fonctionnelles), pour aboutir au MCD. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

Modèle Conceptuel de Données (MCD) :

- permet de modéliser la sémantique des informations d'une façon compréhensible par l'utilisateur de la future base de données;
- utilise le formalisme (graphique) Entité-Relation;
- ne permet pas d'implémentation informatique de la base de données dans un SGBD donné.

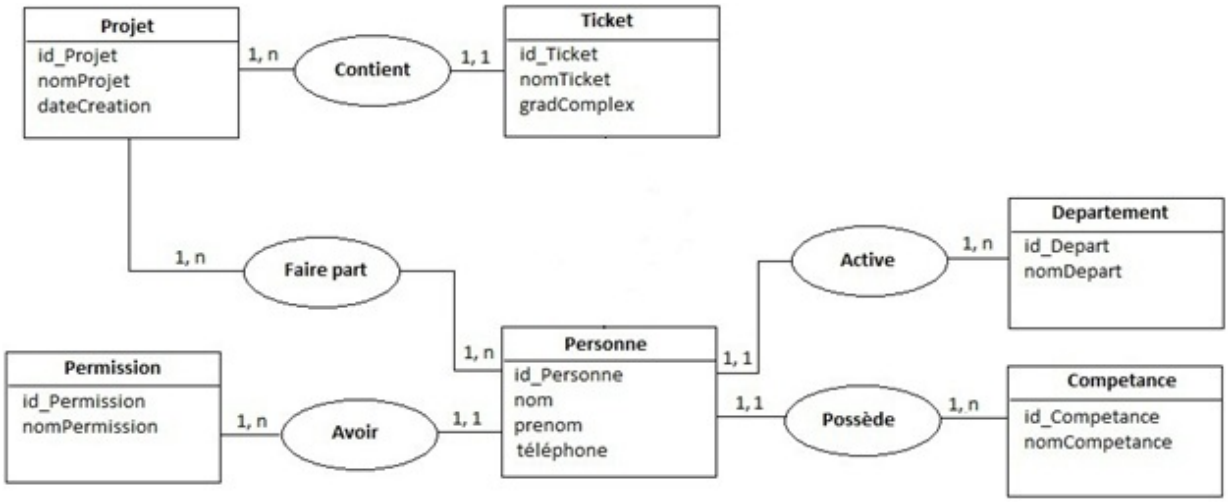

Figure 3.1.1
Représentation du modèle conceptuel des données

Modèle logique des données

La description conceptuelle a permis de représenter le plus fidèlement possible les réalités de l'univers à informatiser. Mais cette représentation ne peut pas être directement manipulée et acceptée par un système informatique. Il est donc nécessaire de passer du niveau conceptuel à second un niveau plus proche des capacités des systèmes informatiques.

Modèle Logique de Données (MLD) :

- permet de modéliser la structure selon laquelle les données seront stockées dans la future base de données ;
- est adapté à une famille de SGBD : ici les SGBD relationnels (MLD Relationnels ou MLD-R) ;
- utilise le formalisme graphique Merise ;
- permet d'implémenter la base de données dans un SGBD donné.

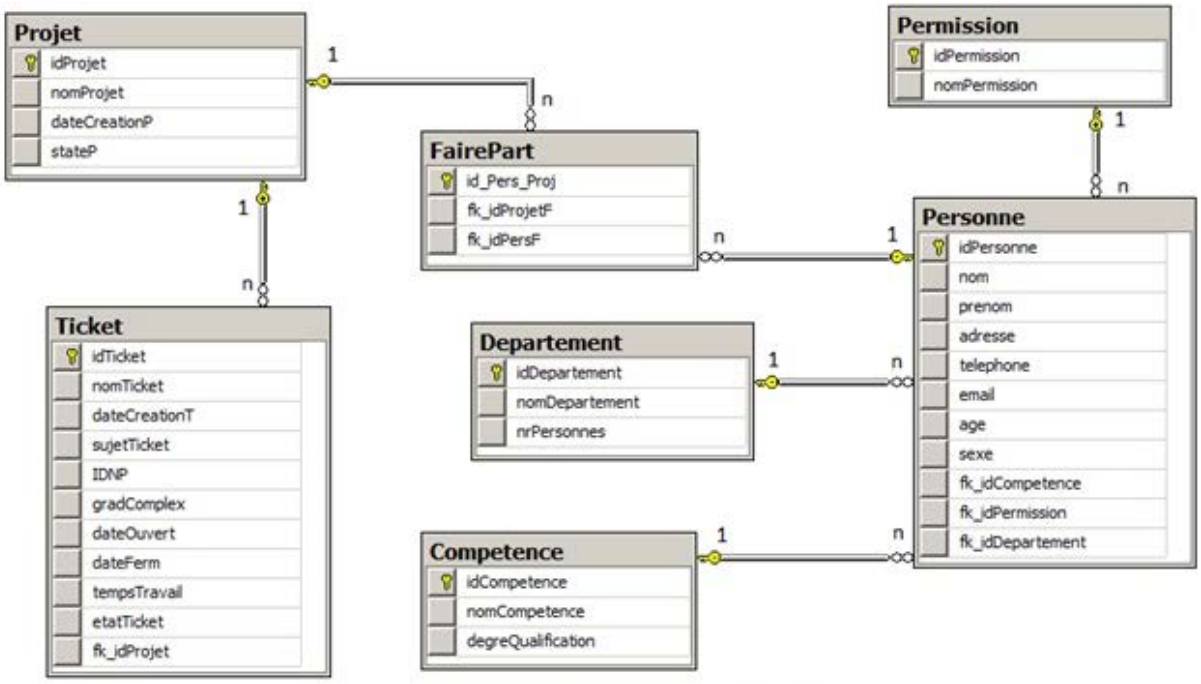

Figure 3.1.2
Représentation du modèle logique des données

La figure suivante [15] représente les démarches d'élaboration d'un MLD Relationnel, appliqué en cadre de développement du projet.

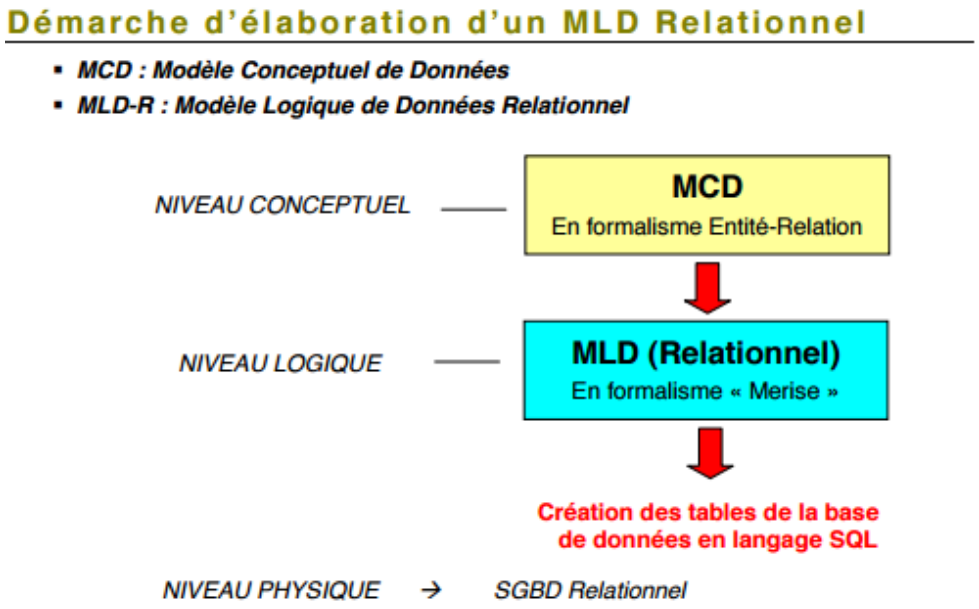

Figure 3.1.3
Représentation des demarches d'élaboration d'un MLD Relationnel

En fin, en se conduisant par les étapes pour passer d'un niveau conceptuel au niveau logique, on a obtenu une telle diagramme de la base des données.

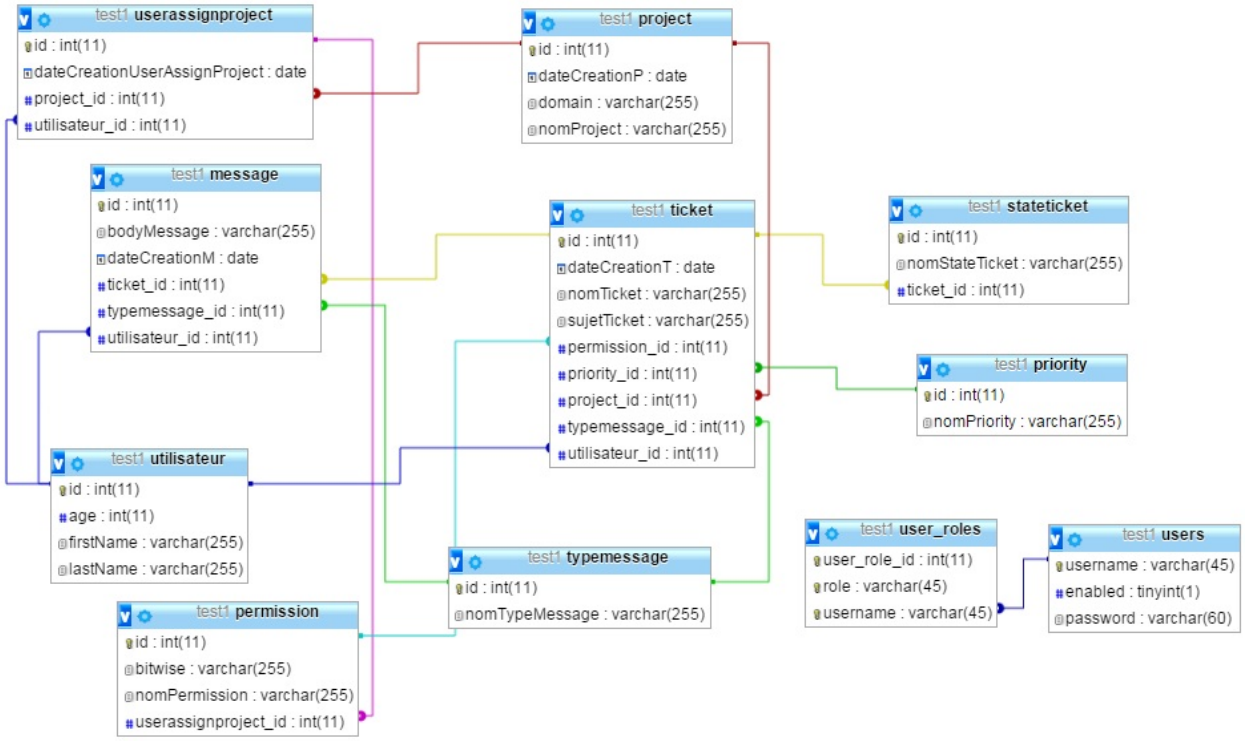

Figure 3.1.4
Diagramme de la base des données

4. PRÉSENTATION DU CODE ET DES RÉSULTATS

Avant de commenter les portions du code, on présente une image [16] qui décrit l'arhitecture du logiciel, et les componantes qui communiques entre eux, pour faciliter un bon fonctionnement.

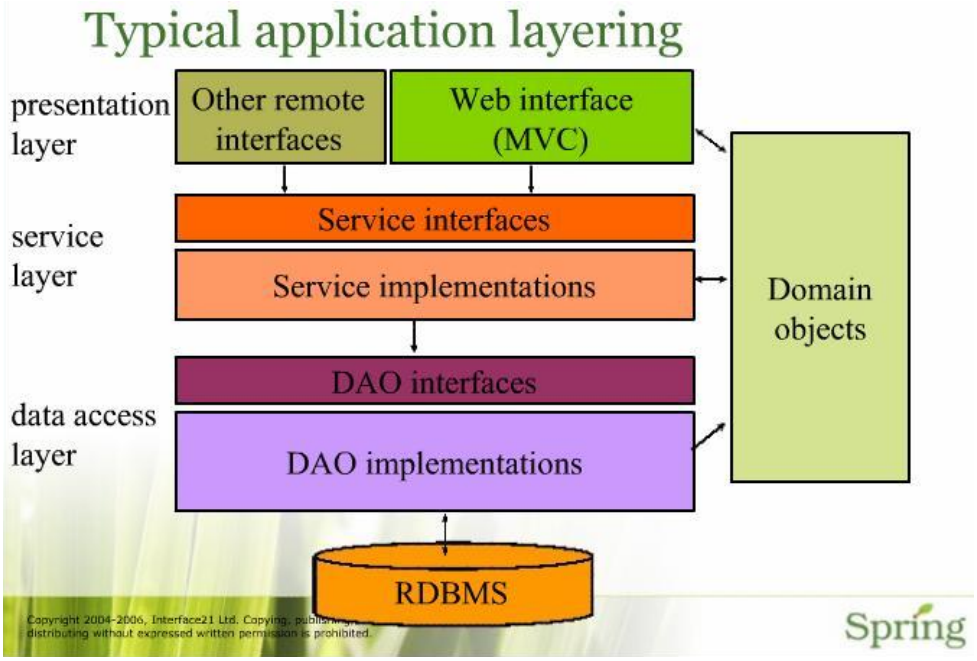

Figure 4.1
Image qui décrit les niveaux de l'application

La portion suivante du code, représente la classe Project POJO (Plain Old Java Object) :

```
@Entity
@Indexed
@Table(name="project")
public class Project extends AbstractTimestampEntity {
	@Id
	@GeneratedValue
	@JsonProperty("id")
	private Integer id;
	……
	@ManyToOne
	@JoinColumn(name = "domain_id")
	@JsonProperty("domainProject")
	@JsonManagedReference
	private DomainProject domainProject;

    @OneToMany(mappedBy="project", targetEntity = Ticket.class,
cascade=CascadeType.ALL, fetch=FetchType.EAGER)
    @JsonBackReference
    private Set<Ticket> tickets = new HashSet<Ticket>(0);
    @OneToMany(mappedBy="project", targetEntity = UserAssignProject.class,
cascade=CascadeType.ALL, fetch=FetchType.EAGER)
    @JsonBackReference
    private Set<UserAssignProject> userAssignProjects =
                                    new HashSet<UserAssignProject>(0);
 //getters et setters
}
```

Comme on obsèrve, parmi le code, on a plusieurs signes @, ce que signifie la notation pour les annotations. Dans le langage de programmation informatique Java, une annotation est une forme de métadonnées syntaxiques qui peuvent être ajoutées au code source Java. Les classes, les variables et les packages peuvent être annotés.

Contrairement au Javadoc, les annotations Java peuvent être réfléchies dans les fichiers de classe générés par le compilateur et par la machine virtuelle Java pour être récupérés au moment de l'exécution. Il est possible de créer des méta-annotations à partir de celles existantes en Java.

@Entity - spécifie que la classe est une entité. Cette annotation est appliquée à la classe entière.

@Indexed - marque le projet en tant qu'entité qui doit être indexée par Hibernate Search.

@Table(name="project") - cette annotation permet de spécifier les détails du tableau qui seront utilisés pour persister l'entité dans la base de données.

L'annotation @Table fournit quatre attributs, ce qui vous permet de remplacer le nom de la table, son catalogue et son schéma et d'imposer des contraintes uniques sur les colonnes du tableau. Pour l'instant, nous utilisons uniquement le nom du tableau qui est projet.

Chaque entité bean aura une clé primaire, que nous annotons sur la classe avec l'annotation @Id. La clé primaire peut être un champ unique ou une combinaison de plusieurs champs en fonction de la structure de votre table.

Par défaut, l'annotation @Id déterminera automatiquement la stratégie de génération de clé principale la plus appropriée, mais vous pouvez l'annuler en appliquant @GeneratedValue.

@JsonProperty - c'est une annotation de type Jackson Annotation qui est utilisé pour indiquer le nom de la propriété dans JSON.

@ManyToOne – définit une association à une seule valeur à une autre classe d'entité qui a une multiplicité multiple. Normalement, il n'est pas nécessaire de spécifier l'entité cible de façon explicite puisqu'elle peut généralement être déduite du type d'objet référencé. Si la relation est bidirectionnelle, le côté de l'entité OneToMany non propriétaire doit utiliser l'élément mappedBy pour spécifier le champ de relation ou la propriété de l'entité qui est le propriétaire de la relation.

@JoinColumn - spécifie une colonne pour rejoindre une association d'entité ou une collection d'éléments. Si l'annotation JoinColumn elle-même est en défaut, une seule colonne de jointure est supposée et les valeurs par défaut s'appliquent.

@JsonManagedReference - l'annotation utilisée pour indiquer que la propriété annotée fait partie de la liaison bidirectionnelle entre les champs; Et que son rôle est "parent" (ou "forward"). Le type de valeur (classe) de la propriété doit avoir une seule propriété compatible annotée avec JsonBackReference. Le couplage est traité de telle sorte que la propriété annotée avec cette annotation soit traitée normalement (sériellement normalement, pas de manipulation spéciale pour la désérialisation); C'est la référence arrière qui nécessite une gestion spéciale

@OneToMany - définit une association un – à – plusieurs avec une multiplicité d'un à plusieurs. Si la collection est définie à l'aide de génériques pour spécifier le type d'élément, le type d'entité cible associée ne doit pas être spécifié; Sinon la classe d'entité cible doit être spécifiée. Si la relation est bidirectionnelle, l'élément mappedBy doit être utilisé pour spécifier le champ de relation ou la propriété de l'entité qui est le propriétaire de la relation.

@JsonBackReference - utilisée pour indiquer que la propriété associée fait partie de la liaison bidirectionnelle entre les champs; Et que son rôle est «lien enfant» (ou «retour»). Le type de valeur de la propriété doit être un faisceau: il ne peut pas s'agir d'une collection, d'une carte, d'un tableau ou d'un énumération. Le couplage est géré de sorte que la propriété annotée avec cette annotation n'est pas sérialisée; Et pendant la désérialisation, sa valeur est définie sur l'instance qui a le lien "géré" (vers l'avant). Toutes les références ont un nom logique permettant de gérer plusieurs liens; Un cas typique serait celui où les noeuds ont des liens parents / enfants et frères et sœurs. Dans l'affirmative, les paires de références devraient être nommées différemment. C'est une erreur pour une classe d'avoir plusieurs références arrière avec le même nom, même si les types pointu sont différents.

La portion suivante du code, représente l'interface ProjectDAO:

```
public interface ProjectDAO {
    public void addProject(Project project);
    public void updateProject(Project project);
    public Project getProject(int id);
    public void deleteProject(int id);
    public List<Project> getProjects();
    public List<DomainProject> getDomains();
    public DomainProject getDomain(int id);
    public List<Project> getSearchProjects(String text);

}
```

Dans un logiciel informatique, un objet d'accès aux données (DAO) [7] est un objet qui fournit une interface abstraite à un type de base de données ou à un autre mécanisme de persistance. En mappant les appels d'application vers la couche de persistance, le DAO [16] fournit des opérations de données spécifiques sans exposer les détails de la base de données. Cet isolement soutient le principe de responsabilité unique. Il sépare les données auxquelles l'application a besoin, en termes d'objets spécifiques au domaine et de types de données (l'interface publique du DAO), de la manière dont ces besoins peuvent être satisfaits avec un SGBD spécifique, un schéma de base de données, etc. (la mise en œuvre de la DAO).

L'avantage d'utiliser des objets d'accès aux données est la séparation relativement simple et rigoureuse entre deux parties importantes d'une application qui ne peuvent se connaitre et qui peuvent évoluer fréquemment et indépendamment. La modification de la logique métier peut s'appuyer sur la même interface DAO, tandis que les modifications apportées à la logique de persistance n'influent pas sur les clients DAO tant que l'interface reste correctement implémentée. Tous les détails du stockage sont cachés sur le reste de l'application.

Comme on peut observer, on a les definitions des methodes, pour:

- créer des projets (addProject) ;
- modifier le projet (updateProject) ;
- trouver le projet par un identificateur (getProject) ;
- supprimer un projet (deleteProject) ;
- obtenir une liste des projets (getProjects) ;
- obtenir une liste des domains pour les projets (getDomains) ;
- trouver un projet spécifique d'après un chaine de caractères(getSearchProjects).

La portion suivante du code, représente la classe ProjectDAOImpl :

```
@Repository
public class ProjectDAOImpl implements ProjectDAO {

	@Autowired
	private SessionFactory sessionFactory;
	private Session getCurrentSession() {
		return sessionFactory.getCurrentSession();
	}
	@Override
	public void addProject(Project project) {
		getCurrentSession().save(project);
	}
	@Override
	public void updateProject(Project project) {
		Project projectToUpdate = getProject(project.getId());
		projectToUpdate.setNomProject(project.getNomProject());
		projectToUpdate.setDomainProject(project.getDomainProject());
		projectToUpdate.setDateCreationP(project.getDateCreationP());
		getCurrentSession().update(projectToUpdate);
	}
	@Override
	public Project getProject(int id) {
		Project project = (Project) getCurrentSession().get(Project.class, id);
		return project;
	}
	@Override
	public void deleteProject(int id) {
		Project project = getProject(id);
		if(project != null)
		getCurrentSession().delete(project);
	}
```

```
@Override
	public List<Project> getProjects() {
		Authentication auth =
SecurityContextHolder.getContext().getAuthentication();

		Query query = sessionFactory.getCurrentSession().createQuery(
				“FROM Project WHERE id IN “
				+ “(SELECT project FROM UserAssignProject WHERE
				    utilisateur = “
				+ “(SELECT id FROM Utilisateur WHERE id =”
				+ “(SELECT userUtilisateur FROM User WHERE username =
:username))) order by id asc”);
		String 56user = auth.getName();
		query.setParameter(“username”, 56user);
		List<Project> list = query.list();
		return list;
	  }

      ……….
}
```

Avant, on a présenté l’intérface ProjectDAO, qui a eu des declarations des méthodes sans implementation. En cadre de la classe ProjectDAOImpl, on fournisse l’implémentation des méthodes déclarées avant, en implémentant l’intérface ProjectDAO, par le mots clé implements. En plus, cette classe contient des annotations comme:

@Repository - indique qu'une classe annotée est un «référentiel», initialement défini par Domain-Driven Design (Evans, 2003) comme «un mécanisme pour encapsuler le stockage, la récupération et le comportement de recherche qui émule une collection d'objets».

@Autowired - marque un constructeur, un champ, une méthode de configuration ou une méthode de configuration pour être automatiquement câblé par les installations d'injection de dépendance de Spring. Un seul constructeur (au maximum) de toute classe de bean donnée peut porter cette annotation, ce qui indique que le constructeur doit être automatiquement utilisé comme faisceau de Spring. Un tel constructeur ne doit pas être public.

Les champs sont injectés juste après la construction d'un haricot, avant que les méthodes de configuration ne soient invoquées. Un tel champ de configuration ne doit pas être public.

Les méthodes de configuration peuvent avoir un nom arbitraire et un nombre quelconque d'arguments; Chacun de ces arguments sera automatiquement câblé avec un faisceau correspondant dans le conteneur Spring. Les méthodes de création de propriétés de Bean ne constituent en effet qu'un cas particulier d'une telle méthode de configuration générale. De telles méthodes de configuration ne doivent pas nécessairement être publiques.

SessionFactory - le contrat principal ici est la création d'instances de Session. Habituellement, une application comporte une instance unique de SessionFactory et les demandes de suivi des clients pour obtenir des instances de session de cette usine. L'état interne

d'une SessionFactory est immuable. Une fois qu'il est créé, cet état interne est défini. Cet état interne inclut toutes les métadonnées sur Object / Relational Mapping.

getCurrentSession() - la methode qui s'occupe d' obtiention de la session courante.

Query - une représentation orientée objet d'une requête Hibernate. Une instance de requête est obtenue en appelant Session.createQuery (). Cette interface expose certaines fonctionnalités supplémentaires au-delà de celles fournies par Session.iterate () et Session.find ():

- Une page particulière de résultats peut être sélectionnée en appelant setMaxResults (), setFirstResult () ;
- Les paramètres de requête nommés peuvent être utilisés ;
- Les résultats peuvent être retournés comme une instance de ScrollableResults.

En cadre de la methode createQuery(), on a formé le HQL, par lequel, on va obtenir dans la liste des projets list, les projets de la base de données. On a formé une requete des plusieurs selects, pour avoir en résultat, les projets auquels l'utilisateur est assigné. On obtient comme paramètre, l'utilisateur, par l'interface Authentication qui fournit l'utilisateur signé dans le système.

La portion suivante du code, représente l'intérface ProjectService :

```
public interface ProjectService {
    public void addProject(Project project);
    public void updateProject(Project project);
    public Project getProject(int id);
    public void deleteProject(int id);
    public List<Project> getProjects();
    public List<DomainProject> getDomains();
    public DomainProject getDomain(int id);
    public List<Project> getSearchProjects(String text);

}
```

C'est évident que cette intérface semble totalement avec l'intérface ProjectDAO. Cela est fait pour faire la liaison avec les intérfaces Web et les DAO intérfaces, pour qu'elles peuvent communiquer, afin de transmettre les données de vues, au base des données et viceversa.

La portion suivante du code, représente la classe ProjectServiceImpl :

```
@Service(value="projectDAO")
@Transactional
public class ProjectServiceImpl implements ProjectService {
	@Autowired
	private ProjectDAO projectDAO;
	@Override
	public void addProject(Project project) {
		projectDAO.addProject(project);
	}
	@Override
	public void updateProject(Project project) {
		projectDAO.updateProject(project);
	}
	@Override
	public Project getProject(int id) {
		return projectDAO.getProject(id);
	}
	@Override
	public void deleteProject(int id) {
		projectDAO.deleteProject(id);
	}
	@Override
	public List<Project> getProjects() {
		return projectDAO.getProjects();
	}
	  ……
}
```

Comme dans le cas de la classe ProjectDAOImpl, dans cette classe on écrit les fonctionnalités pour les methodes crées en cadre de l'intérface ProjectService.

@Service – c'est une spécialisation de l'annotation du composant. Elle ne fournit actuellement aucun comportement supplémentaire au cours de l'annotation @Component, mais il est judicieux d'utiliser @Service sur @Component dans les classes de couche de service car il spécifie mieux l'intention. De plus, le support d'outils et les comportements supplémentaires peuvent compter sur le futur.

@Transactional - en utilisant @Transactional, de nombreux aspects importants tels que la propagation des transactions sont traités automatiquement.

L'annotation @Transactional elle-même définit la portée d'une transaction de base de données unique. La transaction de la base de données se produit dans le cadre d'un contexte de persistance.

@Autowired - marque un constructeur, un champ, une méthode de configuration ou une méthode de configuration pour être automatiquement câblé par les installations d'injection de dépendance de Spring.

La portion suivante du code, représente la classe ProjectController :

```
@Autowired
private ProjectService projectService;

@RequestMapping(value="/add", method=RequestMethod.GET)
public ModelAndView addprojectPage(Map<String, Object> map) {
	ModelAndView modelAndView = new ModelAndView("add-project-form");
	List<DomainProject> domains = projectService.getDomains();
	map.put("project", new Project());
	modelAndView.addObject("domainsList", domains);
	modelAndView.addObject("project", new Project());
	return modelAndView;
}

@RequestMapping(value="/add", method=RequestMethod.POST)
public ModelAndView addingproject(@ModelAttribute Project project) {
	ModelAndView modelAndView = new ModelAndView("home");
	projectService.addProject(project);
	String message = "project was successfully added.";
	modelAndView.addObject("message", message);
	return modelAndView;
}

@RequestMapping(value="/list", method=RequestMethod.GET )
public ModelAndView listOfprojects() {
	ModelAndView modelAndView = new ModelAndView("list-of-projects");
	List<Project> projects = projectService.getProjects();
	modelAndView.addObject("projects", projects);
	return modelAndView;
}

@RequestMapping(value="/delete/{id}", method=RequestMethod.GET)
public ModelAndView deleteproject(@PathVariable Integer id) {
	ModelAndView modelAndView = new ModelAndView("home");
	projectService.deleteProject(id);
	String message = "project was successfully deleted.";
	modelAndView.addObject("message", message);
	return modelAndView;
  }
```

C’est une classe qui intéractionne dirrectement avec les pages jsp, étant comme une intérface entre ce qui voit l’utilisateur et l’application.

@RequestMapping - annotation pour la cartographie des requêtes Web sur des classes de gestionnaires spécifiques et / ou des méthodes de traitement. Fournit un style cohérent entre les environnements Servlet et Portlet, la sémantique s'adaptant à l'environnement concret.

ModelAndView - Titulaire pour le modèle et la vue dans le cadre MVC Web. Ceux-ci sont entièrement distincts. Cette classe supporte simplement les deux pour permettre à un contrôleur de retourner le modèle et la vue en une seule valeur de retour.

Représente un modèle et une vue retournés par un gestionnaire, à résoudre par un DispatcherServlet. La vue peut prendre la forme d'un nom de vue de chaîne qui devra être résolu par un objet ViewResolver; Alternativement, un objet View peut être spécifié directement. Le modèle est une carte, permettant l'utilisation de plusieurs objets codés par le nom.

Figure 4.2

Capture d'écran qui représente le page web avec une forme d'authentification dans le système

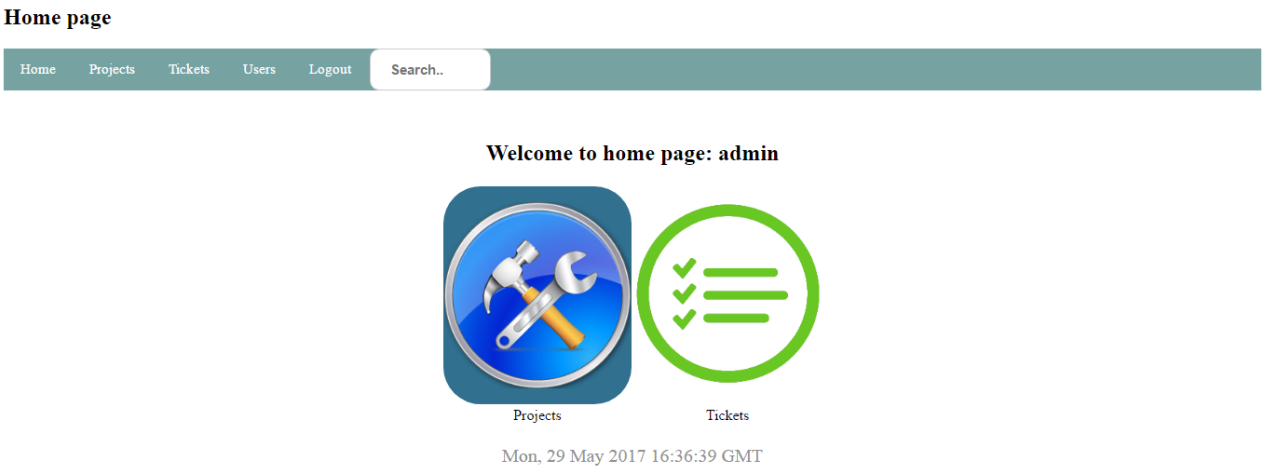

Figure 4.3

Capture d'écran qui représente le page web de l'administrateur avec deux choix, d'acceder le page des projets ou de tickets

Dans la Figure 4.3, on peut voir une barre en haut, qui offre la possibilité aux ceux qui utilisent cette application Web, de naviguer parmi les pages principales comme celles de Projets, Tickets, Users et Home. L'option Logout est pour quitter le système et de devenir un visitateur simple.

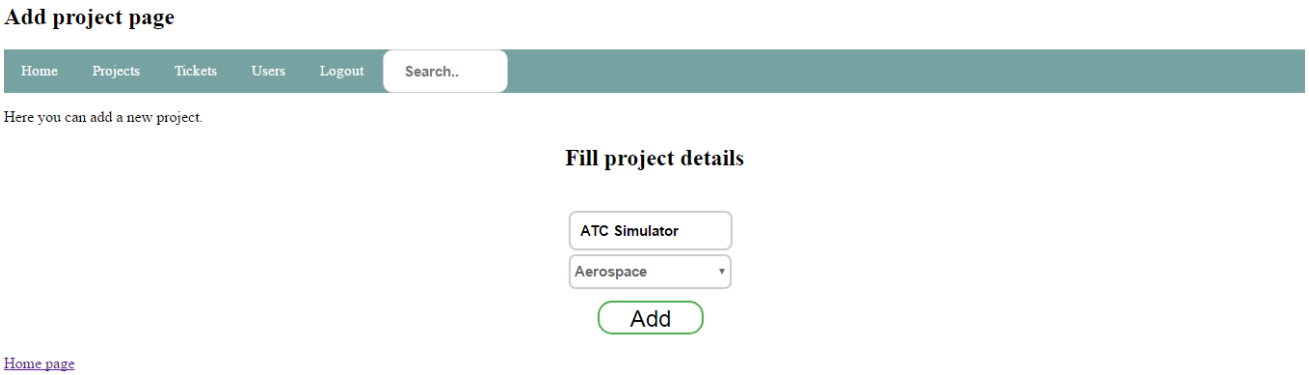

Figure 4.4
Capture d'écran qui représente le page web pour ajouter un nouvel projet

Dans la Figure 4.4, on voit deux champs qui sont remplis par l'information désirée, introduite par l'utilisateur pour ajouter un projet. Dans le deuxième champ, on choisit parmi les domains qui sont déjà présentes. Le bouton Add, confirme l'introduction du projets avec ces données.

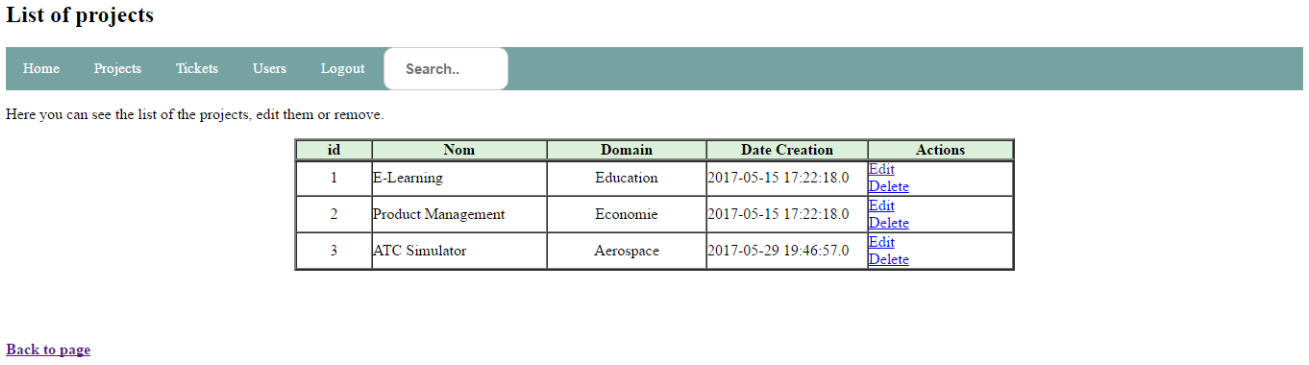

Figure 4.5
Capture d'écran qui représente le page web pour la visualisation des projets existants

L'information sur les projets est présentée sous la forme du tableau, chaque colonne ayant une signification pour le projet et les actions de modification ou pour supprimer le projet choisit. Il faut rémarquer qu'on a dans la barre en haut, une portion blanche avec la possibilité de trouver des projets par leurs nom.

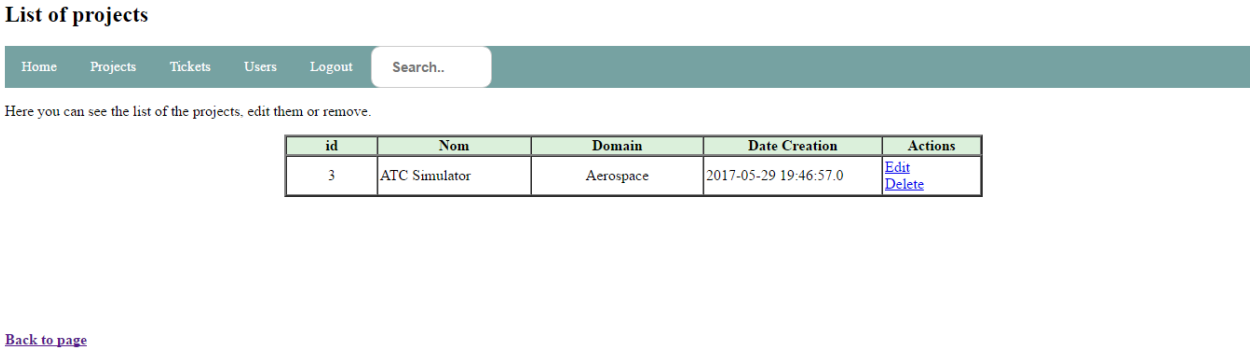

Figure 4.6
Capture d'écran qui représente le page web pour la visualisation du projet cherché

Dans la Figure 4.6., on a écrit le commencement du nom d'un projet dans Search et le projet a été trouvé.

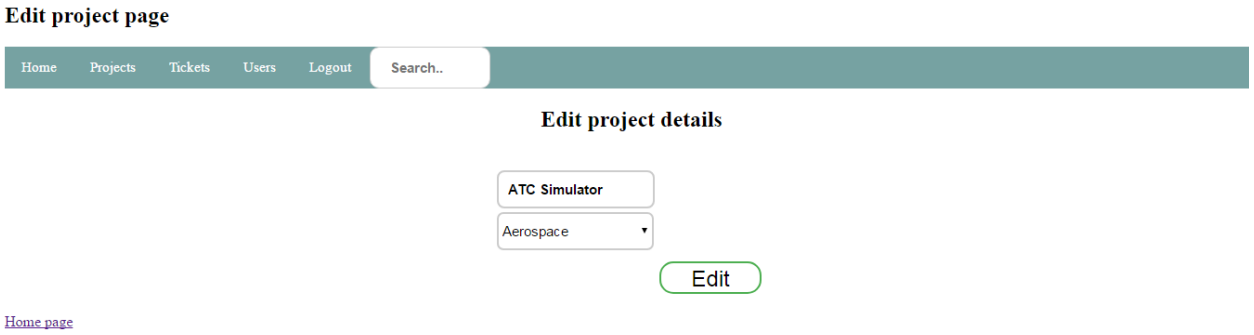

Figure 4.7
Capture d'écran qui représente le page web pour la modification d'un projet choisit de l'option Edit du tableau des projets

Dans la Figure 4.7, on voit deux champs qui peuvent être modifiés selon les besoins d'utilisateur. Le bouton Edit, confirme les changements sur le projet.

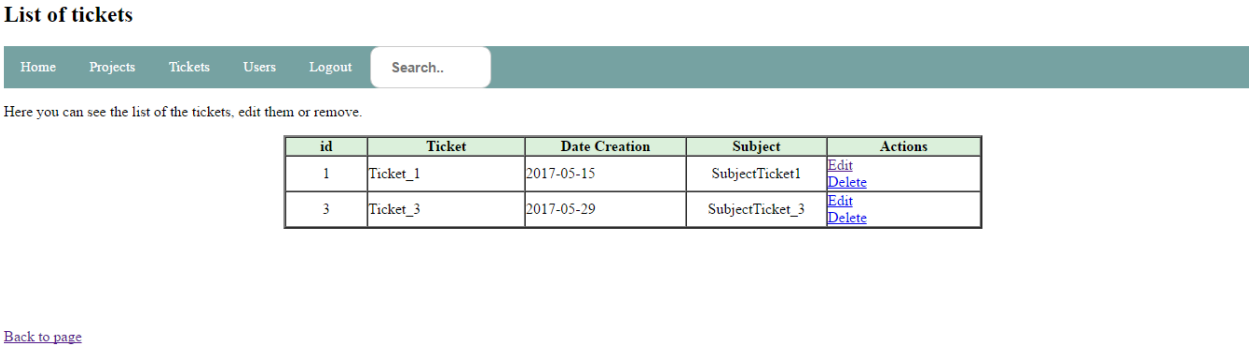

Figure 4.8
Capture d'écran qui représente le page web pour la visualisation des tickets existants

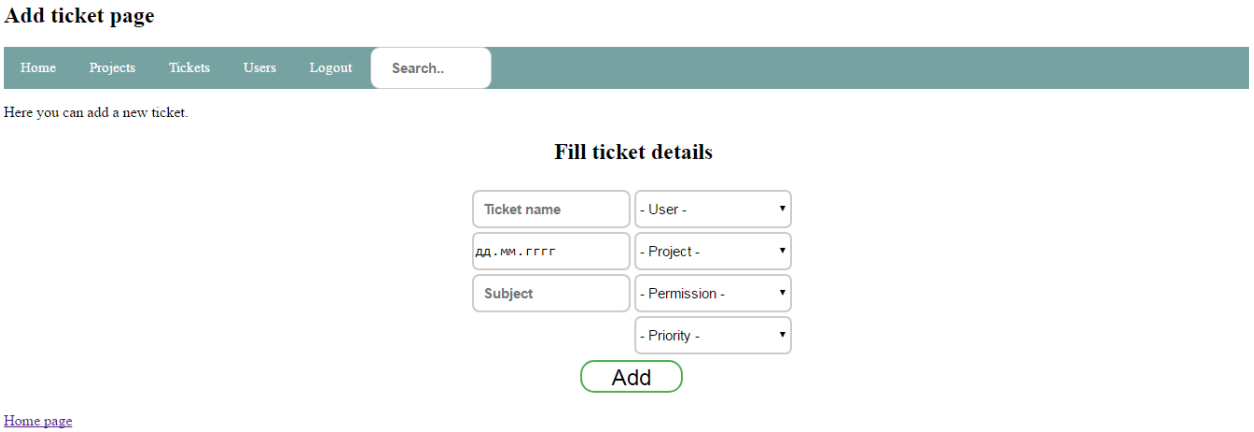

Figure 4.9
Capture d'écran qui représente le page web pour la modification du ticket choisit

En cadre de la Figure 4.9, on voit la page Web, avec les champs à accomplir et choisir par l'utilisateur, pour le ticket choisit avec l'intention de lui modifier. Le bouton Add, confirme le choix de l'utilisateur et modifie le ticket.

La portion suivante du code, représente Le résultat de la requête http://localhost:8080/Tickets/project/1 par la methode GET pour le projet avec l'identificateur id=1


```
{
    "id":1,
    "nomProject":"E-Learning",
    "domainProject":{
        "id":3,
        "nameDomain":"Education"
    },
    "dateCreationP":1494858138000,
    "beginninglife1":"339:20377:1222658 "
}
```


Dans la portion du code en haut, on voit le résultat de la requête en format JSON, qui représente les informations qu'on peut les trouver sur le projet démandé.

La portion suivante du code, représente le résultat de la requête http://localhost:8080/Tickets/project/projects par la methode GET pour tous les projets


```
[
    {
        "id":1,
        "nomProject":"E-Learning",
        "domainProject":{
            "id":3,
            "nameDomain":"Education"
        },
        "dateCreationP":1494858138000,
        "beginninglife1":"339:20386:1223213 "
    },
    {
        "id":2,
        "nomProject":"Product Management",
        "domainProject":{
            "id":4,
            "nameDomain":"Economie"
```

```
                    },
                    "dateCreationP":1494858138000,
                    "beginninglife1":"339:20386:1223213 "
                },
                {
                    "id":3,
                    "nomProject":"ATC Simulator",
                    "domainProject":{
                        "id":1,
                        "nameDomain":"Aerospace"
                    },
                    "dateCreationP":1496076417000,
                    "beginninglife1":"1:82:4934 "
                }
            ]
```


Dans la portion du code en haut, on voit le résultat de la requête en format JSON, qui représente tous les projets d'utilisateur, et leurs informations.

Il faut mentionner qu'on obtienne des résultats un peut différents par rapport aux ces présentés, car les requête se font en utilisant un browser simple. La portion suivante du code, représente le résultat (non-formaté) de la requête http://localhost:8080/Tickets/project/projects par la methode GET pour tous les projets:

[{"id":1,"nomProject":"E-Learning","domainProject":{"id":3,"nameDomain":"Education"},"dateCreationP":1494858138000,"beginninglife1":"339:20399:1223984"},{"id":2,"nomProject":"Product Management","domainProject":{"id":4,"nameDomain":"Economie"},"dateCreationP":1494858138000,"beginninglife1":"339:20399:1223984"},{"id":3,"nomProject":"ATC Simulator","domainProject":{"id":1,"nameDomain":"Aerospace"},"dateCreationP":1496076417000,"beginninglife1":"1:95:5705 "}]

Pour formater les réponses en JSON, on a utilisé un outil Web, publique, qui nous permet de faire les manipulations nécessaires. L'outil mentionné peut être trouvé en cliquant sur la référence suivante [17].

5. ARGUMENTATION ÉCONOMIQUE DU PROJET

5.1 Description du projet

Le système de gestion pour le projets est très actuel dans nos jours. Le travail des logiciels de gestion de projet est généralement d'automatiser des tâches de sauvegarde et/ou de la gestion du temps.

Par exemple, les systèmes de gestion de versions, ou les systèmes de gestion de configuration enregistrent différents états d'un projet et gardent une trace de la date de modification.

Une partie importante des logiciels de gestion de projet s'occupent de la planification des projets, c'est-à-dire de l'ordonnancement de tâches en vue de leur réalisation future. Plusieurs méthodes sont utilisées pour ordonnancer les tâches dont :

- la méthode PERT avec la détermination du chemin critique. Cette méthode n'est plus utilisée de nos jours par les éditeurs de logiciels de planification. La méthode PERT est une méthode de type AOA (Activity On Arcs) ;
- la méthode PDM (Precedence Diagram Method). Cette méthode est du type AON (Activity On Noddles). La méthode PDM est utilisée par la plupart des logiciels de planification de projets ;
- la méthode de la chaîne critique ;
- les méthodes d'ordonnancement basées sur l'emplacement (pour les travaux dont la productivité chute fortement en cas de coactivité dans une zone).

Ce projet est de type production, qui a comme but, la conception des projets informatique, visant à faciliter une gestion detaillée concernant les tâches qu’on va les accomplir pour atteindre le but principal, de développer le projet entier en distribuant des tickets aux ceux qui sont impliqués dans le processus de développement de nouveaux produits.

L’application web « Gestion desprojets à l'aide de tickets » est une application base sur web qui peut rouler sur chaque plateforme ou dispositif qui supporte HTML/CSS/JavaScript comme smartphones, des tablettes, laptop, PC, etc. C’est une application basé sur web pages qui peut être accesée par tous les personnes, et qui nécessite d’avoir une connexion au Internet.

Le but du cette application est de diriger le processus de gestion des projets informatiques, qui va impliquer plusieurs utilisateurs qui vont travailler sur les projets et accomplir les tickets qu’on obtienne pendant le cycle de vie d’une application. Pour avoir accès aux fonctionnalités du cette application, il faut avoir un compte sur ce site Web, qui est totalement gratuit, sans paiements impliqués pour la création du compte utilisateur.

Si l'utilisateur n'a pas encore un compte sur ce site, on peut facilement s'enregistrer et atteindre les administrateurs qui vont les assigner aux projets qui sont créées, ou qui sont en plein derroulement de développement.

L'interface avec utilisateur sera facile à utiliser, à naviguer sur le site Web, utilisant de tous les possibilités qui seront mise à la disposition pour faciliter l'accomplissement des charges en déroulement sur les projets existantes.

5.2 L'analyse SWOT

Tableau 5.2.1 L'analyse SWOT

Points forts : a) le site est facile à utiliser; b) il utilise les dernières technologies; c) le prix du projet est moins que les prix des marchés IT; d) le site est simple et facile de comprendre; e) le site est adapté pour tous les browsers; f) le site est adapté pour smartphones; g) le projet ne nécessite pas beaucoup de personnes impliquées pour le développement; h) grâce à un nombre réduit des personnes qui sont impliquées dans le développement, la communication est effective et efficace.	Points faibles : a) sur le projet travaillé un programmeur débutant; b) le site est bien connu ayant une sécurité faible; c) sur le projet, on a un seul testeur; d) le projet n'est pas unique sur le marché; e) la période de développement est petite.
Opportunités : a) le site est actuel sur le marché local est régionale; b) le niveau du secteur IT est élevé.	Risques : a) Le niveau réduit de connaissances du Java par le Testeur; b) les fautes faites par les responsables des tests.
+10	-7

Pour diminuer les risques on fait les actions suivantes :

- on va faire des code review pour diminuer les erreurs de code et de performance;
- on va avoir des testeurs qui possederont un niveau avancé du langage Java;
- on va sponsoriser des cours d'Anglais.

Pour diminuer les points faibles :

- le programmeur débutant va perfectionner ses connaissances ayant de cours de programmation en Java;

- application des plusieurs testes;
- on va installer des logiciels anti-virus

5.3 Plan de calendrier

Tableau 5.3.1 – Dépenses matériels et non matériels nécessaires à l'élaboration du projet

Nr.	Evènement	Nombre de jours	Personnes employées	Période
1.	L'etablissement des specifications sur le projet	4	Chef du Project Développeur Testeur	06.03.17- 09.03.17
2.	Ecrire les Tests Cases pour l'application Web	3	Testeur	10.03.17- 14.03.17
3.	Lancer les Test Cases	5	Testeur	15.03.17- 21.03.17
4.	Fixer le résultat de l'exécution (BUG)	5	Développeur Chef du Project	22.03.17- 28.03.17
5.	L'étude Servlet, Selenium, Hibernate, Spring framework	7	Testeur Développeur	29.03.17- 06.04.17
6.	L'étude du langage Java, syntaxe	12	Testeur	07.04.17- 24.04.17
7.	Réalisation logique du test	3	Testeur Chef du Project	25.04.17- 27.04.17
8.	Implémentation du site et exécution de Tests Automatisée	15	Développeur Testeur	28.04.17- 22.05.17
9.	Reporter et corriger les erreurs	7	Testeur Développeur Chef du Project	23.05.17- 31.05.17
Total inclusif		**54**		06.03.17- 31.05.17
	Testeur	50		
	Développeur	34		
	Chef du projet	15		

Les chiffres présentés dans le Tableau 5.3.1, représentent la quantité de travail dans chaque semaine, qui peut être également étendu en fonction des possibilités des personnes, et leur rithme d'accomplissement du travail démandé.

5.4 Argumentation économique

Pour chaque activité qui tient de l'élaboration d'un produit, il est toujours nécessaire et essentiel de calculer les dépenses et le profit. Il faut remarquer que, quand on a une bonne vision sur le projet, sur son coût, on a le budget du projet, les indicateurs économiques qui décrivent les performances du projets du point de vue économique.

5.4.1 Les actifs matériels et non matériels à long terme

Tableau 5.4.1.1 – Les dépenses matérielles

Nr.	Nom	Unité de mesure	Prix à l'unité, lei	Quantité	Valeur d'entrée, lei
1	Ordinateur avec 2 moniteurs	Pièce	11000	2	22000
2	Laptop	Pièce	13000	1	13000
3	Imprimante	Pièce	2000	1	2200
Total					**37200**

Les actifs matériels Cm = 37200 lei.

Tableau 5.4.1.2 – Les actifs non matériels à long terme

Nr.	Nom	Unité de mesure	Prix à l'unité, lei	Quantité	Valeur d'entrée, lei
1	OS Windows 10	Pièce	3800	3	11400
2	Tomcat server	Pièce	gratuite	3	gratuite
3	Eclipse	Pièce	gratuite	3	gratuite
4	Open Office	Pièce	gratuite	3	gratuite
5	Java jdk	Pièce	gratuite	3	gratuite
6	XAMPP	Pièce	gratuite	3	gratuite
7	Rational Rose	Pièce	2400	2	4800
8	Selenium	Pièce	gratuite	3	gratuite
Total					**16200**

Les actifs non matériels Cm = 16200 lei.

Tableau 5.4.1.3 – La consommation matériel directe

Nr.	Nom	Unité de mesure	Prix à l'unité, lei	Quantité	Somme, lei
1	Papier	paquet (500 feuilles)	60,00	1	60
2	Cahier	pièce	10,00	4	40
3	Stylo	pièce	5,00	4	20
4	Crayon	pièce	4,00	5	20
5	Règle	pièce	15,00	2	30
4	Café	pièce	70	2	140
5	Thé	pièce	30	3	90
Total					**400**

La consommation matérielle directe Cm = 400 lei.

5.4.2 Rémunération du travail

La rémunération du travail s'effectue selon une multitude des facteurs, comme l'expérience, le stage dans une entreprise, le niveau de connaissances, les aptitudes etc. Le salaire est lié de la somme qui est payé par le client, alors quand il demande un projet. Si la durée du projet est plus longue que six mois, le salaire peut être augmenté, tant que les capacités d'employeur deviennent plus vastes. Cette somme dépende de la grandeur du projet, de la compléxité du travail et des nombres de programmeurs qui vont travailler sur le projet. On calcule la somme des contributions dans le Fond Social et la somme des contributions pour l'Assurance Médicale.

En cadre du développement du projet il y en a 3 personnes :

a) Développeur

- Avoir des bonnes connaissances liées OOP et du Java pour réaliser le projet;
- Il doit avoir des connaissances sur CSS, JavaScript et SQL pour concevoir et maintenir la base de données;
- Il doit connaitre d'une manière générale comment tester les unités du programme.

b) Chef du projet

- Avoir la capacité de coordoner les personnes qui travaillent sur le projet;
- Avoir la capacité de comprendre les éxigences du projet et les charges qu'on doit les réaliser;
- Il doit avoir des aptitudes nécessaires pour se concentrer sur le travail du testeur et développeur.

c) Testeur

- Avoir de l'expérience de la gestion des bogues dans le projet;
- Avoir des connaissances du Java, Tomcat, Selenium
- Posseder des connaissances concernant le CSS, afin de trouver les différentes problèmes d'affichage dans les navigateurs Web.

5.4.3 Les dépenses pour la rémunération du travail

Tableau 5.4.3.1– Les dépenses pour la rémunération du travail

Nr.	Employé	Volume de travail, jours	Salaire contractuel par unité, lei	FSB, lei
1	Chef du projet	15	1100	16500
2	Testeur	50	450	22500
3	Développeur	34	900	30600
Total				**69600**

La valeur du Fond Social en 2017 en Republique de Moldova constitue 23%, et pour la caisse d'assurance médicale constitue 4,5%. En résultat, on a présentée dans le tableau Tableau 7.4.3.2 les prix pour toutes les personnes qui participe à l'élaboration du ce projet.

Tableau 5.4.3.2 – Le fond social et l'assurance médicale

Employé	**Salaire de base (lei)**	**Fond social (lei)**	**Assurance médicale (lei)**
Chef du projet	16500	3795	742.5
Testeur	22500	5175	1012.5
Développeur	30600	7038	1377
Total	**69600**	**16008**	**3132**

5.4.4 La somme annuel de l'impôt sur le revenu d'un développeur

Le revenu annuel d'un développeur impliqué dans le projet est d'environ :

900 lei/jour * 260 jours = 230 000 lei

En utilisant les taux d'impôt courant pour l'année 2017, nous calculons la revenu annuel net et l'impôt sur le revenu transférée au budget de l'Etat. On calculer les retenues au fonds social FS et les contributions d'assurance médicale (FAM) :

$$FR = 0,06 * 230\,000 = 13800 \text{ lei} \quad (1)$$
$$FAM = 0,045 * 230\,000 = 10350 \text{ lei} \quad (2)$$

Suivant le calcul de la revenue imposable:

$$RI = RB – FR – FAM – SP – SiP – SM \quad (3)$$

Où :

R_I - revenu imposable;

R_B - revenu brut;

F_R - fond de retraite (assurance sociale), valeur anuelle, 6% de la revenue;

FAM - fond d'assurance médicale, valeur anuelle , 4,5 % de la revenue;

S_P - exemption personnelle, valeur anuelle (6% de RB), 13800 lei;

$$VI = 230000 - 13800 - 10350 - 10620 = 195230 \text{ lei} \quad (3.a)$$

On calcule la somme du revenu net en appliquant les taux d'imposition en vigueur:

$$RN = VB – IV – FP – FAM \quad (4)$$

R_N – revenu net;

R_B – revenu brut;

I_P - impôt sur le profit;

F_R - fond de retraite (assurance sociale);

F_{AM} - fond d'assurance médicale;

$$I_V = V_I - I \quad (5)$$

I - Impôt sur le revenu;

- pour le revenu annuel d'un maximum de 31140 lei – taux d'imposition appliqué est de 7% ;

- pour les revenus supérieurs à 31140 lei – taux d'imposition appliqué est de 18% ;

$I_V = V_I – I = 31140*0.07 + (144924 - 31140)*0.18\% = 27685$ lei (5.a)

Le revenu net :

$$V_N = 230000 – 27685 – 13800 – 10350 = 178165 \text{ lei} \quad (4.a)$$

5.4.5 Les dépenses indirectes

Supplémentairement aux salaires, il faut payer d'autres services comme les frais Internet, l'alocation de l'espace, la consommation d'électricité, une partie du logiciel et du matériel utilisé, la sécurité. Pendant la periode d'utilisation des équipements techniques, elles pertent leurs prix initial (l'usure) et il faut calculer l'amortissement des coûts indirects. En le cas de mon projet il est nécessaire de calculer l'amortissement pendant tout le temps du travail de l'equipement, prenant en considération la situation initiale de celui-ci. Puis, on peut ici ajouter les prix d'actions de support technique pendant ce temps.

Tableau 5.4.5.1 – Dépenses indirectes

Titre de l'article	**Unité de mesure**	**Quantité**	**Prix/unité, lei**	**La valeur totale, lei**
Contrat de location	m^2	35	250	8750
Sécurité	Abonnement, mois	3	1600	4800
Electricité	KWh	180	1,92	345
Consulting services	Heures	40	100	4000
Internet	Abonnement, mois	3	150	450
Total				**18345**

5.5 Calcul des fonds d'amortissement d'actifs matériels directe et non directe

Le calcul des coûts indirects qui représentent le fonds d'amortissement est très important. On va utiliser la méthode linéaire pour calculer :

$$Uan = MFvuz / T \text{ ou}$$

$$Uan = MFvuz \times Nuz /100\% \quad (1)$$

Uan – somme de l'amortissement annuel;

MFvuz - valeur d'attrition de immobilisations

$$MFvuz = MFvi - MFvr \quad (2)$$

MFvi - valeur initiale d'immobilisations;

MFvr - valeur restant d'immobilisations;

Plus souvent on voit: MFvr = 0.

Nuz - la norme d'usure

L’entreprise a procure technique en somme de 37200 lei. La durée de fonctionnement sur ce projet c’est 3 moins. La somme qui reste à la fin du projet c’est 3500 lei.

$$Uan = (Mfvi - MFvr) / T = (37200 -3500) / 3 = 11233 \text{ lei / an} \quad (1)$$

$$Umois = Uan/12 = 936 \text{lei / mois} \quad (3)$$

$$Uac = Uan * t \quad (4)$$

Uac - usure accumulée;

Uan - usure annuelle;

t – période de fonctionnement, en mois

Tableau 5.5.1 – Fond d’amortissement

Anées	**Valeur initiale**	**Somme de l’usure / an**	**L’usure accumule**	**Le solde**
Au moment initial	37200			37200
A la fin de 1ere année	37200	11233	11233	25967
A la fin de 2eme année	37200	11233	22466	14734
A la fin de 3eme année	37200	11233	33699	3501

Pendant la durée de projet l’usure accumulée représente 936 lei/mois. La somme totale, pendant trois mois est de 2808 lei.

5.6 Prix du coût

Le coût est calculé sur une unité. Si on développe un site Web ou une application, cela sera le coût de la mise au point, mais si on a plusieures copies des produits alors, il est nécessaire de calculer le prix qui revient pour une copie.

Tableau 5.6.1 - Prix du coût

Éléments de calcul	**La valeur, lei**	**Poids,%**
Les dépenses pour la rémunération du travail	69600	63,10
La contribution sociale	16008	14,51
Assurance médicale	3132	2,84
Dépenses indirecte	18345	16,63
Consommation matérielle directe	400	0,36
Fond d’amortissement	2808	2,54
Total	**110293**	**100**

5.7 Les résultats financiers

Ici est calculé le chiffre d'affaires nette et brute. L'entreprise est de type SRL ou le taux d'imposition représente 12 %. Le produit qui a été implémente, sera en production commençant avec 01.07.2017.

Nous pouvons vendre le produit à l'autre compagnie a prix de 14000 lei. Nous avons 10 compagnies pour qui nous prestons les services IT. Ça signifie que nous pouvons obtenir la somme de 140000 lei.

$$VV_N = 140000 \text{ lei}$$

$$P_B = VV_N - C_T \ (1)$$

$$P_B = 140000 - 110293 = 29707 \text{ lei } (1)$$

$$P_N = P_B - I_V \ (2)$$

Où IP - l'imposition sur le profit (12% du profit imposable)

$$P_N = 29707 - 29707 * 0{,}12 = 26142 \text{ lei } (2)$$

$$\text{Rentabilité économique} = P_N/C_T*100\% \ (3)$$

$$\text{Rentabilité économique} = (26142 / 110293) *100\% = 23{,}7\% \ (3)$$

5.8 Conclusion sur la partie économique

En cadre de la partie économique du projet, j'ai déterminé tous les coûts nécessaires pour son développement. Initialement, j'ai établis les objectifs du projet et à partir des objectifs j'ai déterminé les activités nécessaires à effectuer. On a réalisé une estimation du temps pour chaque activité, l'étape de laquelle elle fait partie et j'ai trouvé la personne résponsable pour l'accomplissement de l'activité.

Après l'établissement du plan calendrier j'ai effectué les calculs économiques, c'est-à-dire les dépenses pour les actifs matériels, immatériels et les consommations directes. Des calculs on peut observer que les dépenses plus grandes sont effectuées à l'achat de la technique de calcul (actifs matériels). Des autres dépenses sont effectuées à la rémunération du travail.

En plus, j'ai calculé le salaire du chaque membre de l'équipe en multipliant le nombre de jours travaillés avec l'unité de salaire par jour, du salaire obtenu j'ai extrait les dépenses pour le fond sociale et l'assurance médicale. Pour savoir quel est le salaire net d'un employé j'ai calculé les retenues pour le fond de pension, l'impôt sur le revenu, la franchise personnel et j'ai fait les calculs nécessaires.

CONCLUSION

La conception d'un tel système informatique, nous permet de comprendre mieux le système en applicant les langages de la programmation et les bonnes recommendations pour écrire un code lisible et bien compris pour tous les programmeurs. Ayant comme but de réaliser les charges posées dans le projet, on a vu les étapes et le cycle de vie de l'application comme: les spécifications, la conception des diagrammes UML, le développement, les testes, la validation, la correction des bogues et ensui de suite.

Selon moi, l'analyse d'un système est bénéfique pour détailler chaque component, chaque activité qui devra être rélisée par le système crée. Dans mon cas, je voudrais remarquer qu'on a effectué l'analyse et le développement de l'application, qui m'a permis d'assimiler tant des choses en ce qui concerne les diagrammes de différents types, comment les appliquer pour décrire les particularites du projet, comment les transformer dans le code, en applicant les principes OOP de la programmation et comment atteindre une minimisation des options mais une variété des possibilités qui nous offre ce système. Je tiens à mentioner qu'à chaque étape de réalisation, on a eu des differentes difficultés, comme la logique du fonctionnement, comment implémenter la logique, et faire ça d'une manière efficace et stable. Ce qui m'a aidé est la diversification des diagrammes UML , initialement qui m'a paru un peux difficile, mais en même temps, elle m'a conduit aux idées que si nous feront qualitativement tous les diagrammes, nous pourrons réduire ou bien minimiser le coûts d'équipement, les erreurs, les pannes qui pourront apparaître dans le système.

On doit mentionner que, ce qu'on a construit, poura être utilisé presque en cadre du chaque entreprise qui s'occupe de la livraison des solutions informatiques incapsulées dans les projets de différente dimension et thème. C'est une solution particulière et authentique de logiciel pour gestionner les projets informatiques, et le but été de parcourir toutes les étapes dans le développement et voir comment utiliser les outils pour construir des diagrammes UML en Rational Rose, et implémenter dans le code écrit en Java et quelques frameworks. Cette thèse de licence vient à totaliser les connaissances acquises pendant les quatres années de l'université, et de prouver les competences dans la programmation et de projection des spécifications nécessaires, afin de faciliter un véritable résultat.

BIBLIOGRAPHIE

1. Spring Framework Reference Documentation [resource électronique].
   - Mode d'accès:
   http://docs.spring.io/spring/docs/current/spring-framework-reference/pdf/spring-framework-reference.pdf

2. Spring Framework [resource électronique]
   - Mode d'accès: https://fr.wikipedia.org/wiki/Spring_(framework)

3. Mark Grand and Jonathan Knudsen, Java Fundamental Class Reference, O'Reilly, 1997.

4. Peter Coad and Mark Mayfield, Java Design: Building Better Apps and Applets, Yourdon Press, 1996.

5. PhpMyAdmin's Documentation [resource électronique]
   - Mode d'accès : https://docs.phpmyadmin.net/en/latest/

6. Eric Jendrock, Ricardo Cervera-Navarro: The Java EE 7 Tutorial: Volume 1 (Fifth Edition), Addison-Wesley, 2014.

7. Core J2EE Patterns - Data Access Object [resource électronique]
   - Mode d'accès : http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

8. Eric Jendrock, Ricardo Cervera-Navarro: The Java EE 7 Tutorial: Volume 2 (Fifth Edition), Addison-Wesley, 2014

9. Gestion de projet- typologie des projets, Gérard Casanova - Denis Abécassis [présentation électronique]
   - Mode d'accès :
   http://ressources.aunege.fr/nuxeo/site/esupversions/3b04551a-e8ea-4bd1-ad75-75732c334a3d/res/pdf.pdf

10. Java T Point [ressource électronique]. – Mode d'accès : http://www.javatpoint.com

11. Stackoverflow [ressource électronique]. – Mode d'accès : http://stackoverflow.com/

12. Code Java [ressource électronique]. – Mode d'accès : http://www.codejava.net/

13. ,,Effective Java (2nd Edition)", Joshua Bloch, 2010

14. ,,Spring in action" , Fourth Edition, Craig Walls, 2014

15. ,,Elaboration d'un Modèle Logique de Données Relationnel (MLD-R)",
    Bernard ESPINASSE, Novembre 2012, [présentation électronique]
    – Mode d'accès : http://www.lsis.org/espinasseb/Supports/BD/ER-Relationnel-4p.pdf

16. ,,Service layer and DAO Arhitecture", [ressource électronique]
    – Mode d'accès : https://biese.wordpress.com/2007/10/08/service-layer-and-dao-architecture/

17. JSON FORMATTER & VALIDATOR, [ressource électronique]
    – Mode d'accès : https://jsonformatter.curiousconcept.com/

Annexe A

Le résultat de la requête http://localhost:8080/Tickets/ticket/1, pour le ticket avec l'id 1, en format JSON.


```
{
  "id":1,
  "nomTicket":"Ticket_1",
  "dateCreationT":"2017-05-15",
  "sujetTicket":"SubjectTicket1",
  "project":{
    "id":2,
    "nomProject":"Product Management",
    "domainProject":{
      "id":4,
      "nameDomain":"Economie"
    },
    "dateCreationP":1494858138000,
    "beginninglife1":"339:20384:1223079 "
  },
  "utilisateur":{
    "id":2,
    "firstName":"Ploaia",
    "lastName":"Vladislav",
    "fullName":" Ploaia Vladislav",
    "age":23,
    "message":[

    ]
  },
  "priority":{
    "id":2,
    "nomPriority":"Low"
  },
  "permission":{
    "id":1,
    "nomPermission":"FULL",
```

```
    "bitwise":"2",
    "userAssignProject":{
      "id":2,
      "dateCreationUserAssignProject":"2017-04-27",
      "utilisateur":{
        "id":2,
        "firstName":"Ploaia",
        "lastName":"Vladislav",
        "fullName":" Ploaia Vladislav",
        "age":23,
        "message":[

        ]
      },
      "project":{
        "id":2,
        "nomProject":"Product Management",
        "domainProject":{
          "id":4,
          "nameDomain":"Economie"
        },
        "dateCreationP":1494858138000,
        "beginninglife1":"339:20384:1223079 "
      }
    }
  },
  "typemessage":{
    "id":2,
    "nomTypeMessage":"TicketMessage"
  }
}
```

La classe SecurityConfig dans laquelle on va securiser les requêtes faites par les utilisateurs, et assurer qu'ils sont authentifiés dans le système

```
@Configuration
@Component
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

	@Autowired
	@Qualifier("userDetailsService")
	UserDetailsService userDetailsService;

	@Autowired
	public void configureGlobal(AuthenticationManagerBuilder auth) throws
Exception {
		auth.userDetailsService(userDetailsService);
	}

	@Override
	protected void configure(HttpSecurity http) throws Exception {

		http.authorizeRequests()
		.antMatchers("/admin/**", "/home/**", "/project/delete/**",
"/ticket/delete/**", "/utilisateur/delete/**",
				"/project/edit/**", "/ticket/edit/**",
"/utilisateur/edit/**", "/project/search/**"  )
			.access("hasRole('ROLE_ADMIN')").and().formLogin()
			.loginPage("/login").failureUrl("/login?error")
				.usernameParameter("username")
				.passwordParameter("password")
				.and().logout().logoutSuccessUrl("/login?logout")
				.and().csrf()
				.and().exceptionHandling().accessDeniedPage("/403");

		http.formLogin().loginPage("/login")
		.and()
		.rememberMe()
		.tokenValiditySeconds(2419200)
		.key("userKey");
	}

}
```

La classe TicketController représente le contrôleur qui est la première composante qui traite les événements de l'interface avec l'utilisateur. Cette classe décrit la logique des opérations pour l'ajout, la modification et l'élimination d'un ticket.

```
public class TicketController {

	@Autowired
	private TicketService ticketService;

	@RequestMapping(value = "/add", method = RequestMethod.GET)
	public ModelAndView addTicketPage(Map<String, Object> map) {
		ModelAndView modelAndView = new ModelAndView("add-ticket-form");
		List<Project> projects = ticketService.getProjects();
		List<Utilisateur> utilisateurs = ticketService.getUtilisateurs();
		List<Permission> permissions = ticketService.getPermissions();
		List<Priority> priorities = ticketService.getPriorities();
		map.put("ticket", new Ticket());
		modelAndView.addObject("utilisateursList", utilisateurs);
```

```
		modelAndView.addObject("projectsList", projects);
		modelAndView.addObject("permissionsList", permissions);
		modelAndView.addObject("prioritiesList", priorities);
		modelAndView.addObject("ticket", new Ticket());

		return modelAndView;
	}

	@RequestMapping(value = "/add", method = RequestMethod.POST)
	public ModelAndView addingTicket(@ModelAttribute Ticket ticket) {
		ModelAndView modelAndView = new ModelAndView("home");
		ticketService.addTicket(ticket);
		String message = "Ticket was successfully added.";
		modelAndView.addObject("message", message);

		return modelAndView;
	}


	@RequestMapping(value="/list")
	public ModelAndView listOfTickets() {
		ModelAndView modelAndView = new ModelAndView("list-of-tickets");

		List<Ticket> tickets = ticketService.getTickets();
		modelAndView.addObject("tickets", tickets);

		return modelAndView;
	}


	@RequestMapping(value = "/edit/{id}", method = RequestMethod.GET)
	public ModelAndView editTicketPage(@PathVariable Integer id, Map<String,
Object> map) {
		ModelAndView modelAndView = new ModelAndView("edit-ticket-form");
		List<Project> projects = ticketService.getProjects();
		List<Utilisateur> utilisateurs = ticketService.getUtilisateurs();
		List<Permission> permissions = ticketService.getPermissions();
		List<Priority> priorities = ticketService.getPriorities();
		map.put("ticket", new Ticket());
		Ticket ticket = ticketService.getTicket(id);
		modelAndView.addObject("utilisateursList", utilisateurs);
		modelAndView.addObject("projectsList", projects);
		modelAndView.addObject("permissionsList", permissions);
		modelAndView.addObject("prioritiesList", priorities);
		modelAndView.addObject("ticket", ticket);

		return modelAndView;

	}

	@RequestMapping(value = "/edit/{id}", method = RequestMethod.POST)
	public ModelAndView edditingTicket(@ModelAttribute Ticket ticket,
@PathVariable Integer id) {
		ModelAndView modelAndView = new ModelAndView("home");
		ticketService.updateTicket(ticket);
		String message = "Ticket was successfully edited.";
		modelAndView.addObject("message", message);

		return modelAndView;
	}
```

```
	@RequestMapping(value = "/delete/{id}", method = RequestMethod.GET)
	public ModelAndView deleteTicket(@PathVariable Integer id) {
		ModelAndView modelAndView = new ModelAndView("home");
		ticketService.deleteTicket(id);
		String message = "Ticket was successfully deleted.";
		modelAndView.addObject("message", message);

		return modelAndView;
	}


	@InitBinder
	public void initBinder(WebDataBinder binder) {
		SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
		sdf.setLenient(true);
		binder.registerCustomEditor(Date.class, new CustomDateEditor(sdf,
true));
	}

}
```

La portion suivante représente la page écrit en utilisant JSP, pour l'ajout d'un projet.

```
<%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<%@taglib prefix="sec" uri="http://www.springframework.org/security/tags"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@page session="true"%>

<?xml version="1.0" encoding="ISO-8859-1" ?>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title>Add project page</title>
<style>
     <%@ include file="MenuStyle.css"%>

</style>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.3/jquery.min.js"
type="text/javascript"></script>

</head>
<body>
<h2>Add project page</h2>
<jsp:include page="header.jsp" />

<sec:authorize access="hasRole('ROLE_ADMIN')">
<p>Here you can add a new project.</p>

<form:form method="POST" commandName="project"
action="${pageContext.request.contextPath}/project/add.html">
<h2 align="center">Fill project details</h2>
<table align="center">
<tbody>
```

```
	<tr>
		<td><form:input class="info" placeholder="Project name"
path="nomProject" required="required"/></td>
	</tr>

	<tr>
		<td>
			<form:select style="color:dark; font-size:15px; height:36px;
width:169px; border-radius:7px; border:2px solid #ccc; box-sizing: border-box; font-
weight:bolder; color:#686868; " path="domainProject.id" enctype="multipart/form-
data">
				<option value="Select" label=" - Domain - " ></option>
				<form:options items="${domainsList}" path="domain_id"
itemValue= "id" itemLabel= "nameDomain"></form:options>
			</form:select>
		</td>
	</tr>
	<tr>
	</br>
	<td><tr></tr></td>
	<td align="center"><input type="submit" value="Add" class="button button1"
/></td>
		<td></td>
		<td></td>
	</tr>
</tbody>
</table>
<input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}"/>
</form:form>
</sec:authorize>
<sec:authorize access="hasRole('ROLE_USER')">
  <h3>Only Admin can add new projects!</h3>
</sec:authorize>
<sec:authorize access="isAnonymous()">
<br/><h3 style="padding-left: 20px;">For add new projects, please login first</h3>
</sec:authorize>

<p><a href="${pageContext.request.contextPath}/index.html">Home page</a></p>
</body>
</html>
```

UNIVERSITÉ TECHNIQUE DE MOLDOVA

FACULTÉ Ordinateurs, Informatique et Microélectronique

Filière Francophone Informatique

AVIS

du projet de licence

Thème :« Gestion de projets informatiques à l'aide des tickets »

Étudiant : Ploaia Vladislav, gr. FI-131

1. Actualité du thème

Le sujet de la thèse est orienté sur la gestion des projets informatiques et vise l'implémentation d'une méthode actuelle qui facilite l'intégration et l'interopérabilité des personnes travaillant dans le cadre des projets. Cette application permet aux utilisateurs de connaître les tâches a réaliser, leur statut, elle offre plusieurs fonctionnalités pour les visualiser, les modifier, les supprimer. On y trouve aussi plusieurs fonctionnalités concernant la gestion de plusieurs projets à la fois et l'implication des utilisateurs.

2. Caractéristique de la thèse de licence

Le projet effectué par M. Ploaia Vladislav vise à étudier le domaine de gestion du processus du développement des logiciels et la mise en œuvre d'un système d'attribution et de suivi de taches dans le cadre d'un projet.

3. Analyse du prototype

Des systèmes similaires ont été analysés : DAO, Creator, Controller. Cela a permis d'établir le cahier de charge de ce projet et de spécifier ses fonctionnalités de base

4. Estimation des résultats obtenus

Le projet de licence est composé de cinq chapitres. La description de chaque chapitre est présentée dans l'annotation du document. Le processus d'élaboration de la thèse par l'étudiant a contribué à une bonne compréhension du système en appliquant des différentes technologies de programmation : Java, UML, HTML, CSS, JavaScript, Jquery, etc. Ce spectre d'applications vise le développement des logiciels de complexités différentes.

5. Exactitude du matériel exposé

Le matériel présenté dans cette thèse est original et correspond aux exigences établies dans le cahier de charge. L'étudiant a employé des techniques actuelles de conception pour son travail : JavaScript, Java, UM, etc.

6. Qualité du matériel graphique

Les diagrammes présentes sont de bonne qualité, lisibles et faciles à comprendre. Elles décrivent le fonctionnement intégral du système et permettent aussi de comprendre sa structure et le rôle de chaque composant.

7. Valeur pratique de la thèse

L'application proposée est un outil pour gérer le processus de développement des projets informatiques et elle répond aux plusieurs besoins de nos jours.

8. Observations et suggestions

Le logiciel n'est pas suffisamment documenté ce que pose de problèmes au niveau de la compréhension de son fonctionnement et de son utilisation pratique. Il faut aussi réfléchir a adapter se système aux spécificités de l'entreprise et aux processus particuliers de développement.

9. Caractéristique d'étudiant et le titre attribué

Monsieur Ploaia Vladislav, en réalisant ce projet de licence, a fait preuve d'excellentes capacités intellectuelles et professionnelles indispensables pour un ingénieur. Il a su appliquer ses connaissances en informatique et des différentes techniques de programmation pour réaliser le prototype de son système. À mon avis cette thèse de licence correspond aux exigences requises pour les thèses de licence et peut être appréciée avec la note 10 , M. Ploaia Vladislav mérite l'attribution du titre d'ingénieur en informatique.

Tuteur de la thèse de licence
Maître de conférences, dr. MORARU Victor _____

DECLARAȚIA DE ONESTITATE A STUDENTULUI

Subsemnatul (a) _____ declar pe proprie răspundere că lucrarea de faţă este rezultatul muncii mele, pe baza propriilor cercetări şi pe baza informaţiilor obţinute din surse care au fost citate şi indicate, conform normelor etice, în note şi în bibliografie. Declar că lucrarea nu a mai fost prezentată sub această formă la nici o instituţie de învăţământ superior în vederea obţinerii unui grad sau titlu ştiinţific ori didactic.

Semnătura autorului,_____