Projet de Fin d'Etudes

Licence Sciences et Techniques Génie Informatique

Mise en place d'une application de gestion du parc automobiles

ⵜⴰⴳⵍⴷⵉⵜ ⵏ ⵍⵎⵖⵔⵉⴱ
ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵙⴰⵡⵍⵜ

المملكة المغربية
وزارة الثقافة

Royaume du Maroc
Ministère de la Culture

Lieu du stage : Ministère de la Culture de Rabat

Réalisé par :

Choukri Abderrahmane

Encadré par :

Pr.Aicha Majda

Mr. M.Laghmouchi

Soutenu le samedi 14 juin 2014 devant le jury composé de :

Pr.Aicha Majda

Pr.Arsalan Zarghili

Pr.Ahlam Begdouri

Année Universitaire 2013-2014

REMERCIEMENT

Au nom d'Allah le tout miséricordieux, le très miséricordieux.

Ce travail, ainsi accompli, n'aurait pas pu arriver à terme sans l'aide et le soutien d'Allah, louange au tout miséricordieux, le seigneur de l'univers. Je remercie en premier lieu mes parents qui ne préservent aucun effort pour me voir escalader à pas surs la montagne du savoir, et dépasser tous les obstacles vers l'amélioration. Tout mot dit, je ne les remercierai jamais assez.

Ensuite, je tiens à exprimer ma profond gratitude à tout le corps professionnel qui m'a amené jusqu'à ce point ci. Je remercie mes chers professeurs , en particulier qui font partie de mon jury Mme A.Begdouri et Mr A.Zarghili, je remercie infiniment mon encadrante pédagogique Mme Aicha Majda pour m'avoir accordé sa confiance, pour le temps qu'elle m'a consacré durant toute ma période de stage, pour ses conseils, son soutien, et son encadrement. Après, je tiens à adresser mes sincères remerciement ainsi que mes sentiments les plus respectueux à mon encadrant externe **Mr Laghmouchi Mouhamed** ingénieur développeur au ministère de la culture de Rabat qui a mis à ma disposition toutes les ressources nécessaires pour le bon déroulement de ce stage, et qui n'a jamais hésité un instant à m'orienter avec ces précieuses directives et ces judicieux conseils.

LISTE DES ABREVIATIONS

Abréviation	Signification
API	**Application Programming Interface**
EJB	**Enterprise JavaBeans**
IHM	**Interface home machine**
HTML	**HyperText Markup Language**
http	**HyperText Transfer Protocol**
IDE	**Integrated Environnement Development**
JAVA EE	**Java Entreprise Edition**
JDBC	**Java DataBase Connectivity**
JDK	**Java Development Kit**
JEE	**Java 2 Enterprise Edition**
JPA	**Java Persistence API**
JSF	**Java Server Faces**
JSP	**Java Server Page**
JTA	**Java Transaction API**
MVC	**Model View Controller**
FST	**Faculté des sciences et techniques**
SGBD	**Système de Gestion de Base de Données**
SQL	**Structured Query Language**
UML	**Unified Modeling language**
URL	**Uniform Resource Locator**
XHTML	**eXtensible HyperText Markup Language**
XML	**eXtensible Markup Language**

TABLE DE MÉTIERS

Remerciement 1

Liste des abréviations 2

Introduction général 5

chapitre 1 : contexte general du projet 6

1.1) Problématique et objectifs du projets 6

1.2) Presentation de l'organisme d'accueil 7

1.3) Cahier des charges de l'applcation 8

1.4) Planning du projet 9

Chapitre 2 : analyse des besoins et conception 10

2.1) Cycle de développement processus en y 10

2.2) Spécification et analyse des besoins techniques 12

2.2.1) Le langage de modélisation unifie : uml 2.0 13

2.2.2) Le modele mvc 2 15

2.2.3) Java persistence API 17

2.2.4) Java entreprse edition (JEE) 18

2.2.5) Glassfish 4.0 20

2.2.6) Mysql 20

2.2.7) Le framework jsf 21

2.2.7) Spring security 3.0.6 24

2.2.8) Ireport 5.5.1 24

2.3) Spécification et analyse des besoins fonctionnels 25

2.3.1) Capture des besoins fonctionnels 26

2.3.2) Acteurs du systéme 26

2.3.3) Les fonctionnalité principales 27

2.3.4) Contraintes du projet 29

2.3.5) ArchItecture fonctionnelle du projet 30

2.3.6) Analyse et spécification 30
2.4) Analyse et conception 36
2.4.1) Identification des cas d'utilisation 37
2.4.2) Diagrammes de sequences 41
2.4.3) Diagramme de classes 45
Chapitre 3 : Mise en œuvre du projet 47
3.1) Architecture de l'application 47
3.2) Desciption des ihm 48
Conclusion générale 55
Webographie : 56
Bibliographie : 56

INTRODUCTION GENERAL

Dans le cadre de la formation licence sciences et techniques filière : informatique de la faculté des sciences et technique FES, les étudiants de la troisième année de licence sortent en stage dans le semestre 6 pour mettre en pratique leurs connaissances pendant la formation. Ce stage qui dure environ 50 jours, assure dans le futur aux étudiants une rapidité et une facilité d'intégration dans le milieu professionnel et fera l'objectif d'une soutenance publique. C'est dans ce cadre que j'ai été accueilli par le ministère de la culture de Rabat et mon sujet était l'informatisation de la gestion du parc automobiles.

L'informatisation de la gestion du parc automobiles sera nécessaire puisque la gestion manuelle connait plusieurs difficultés dues au nombre important des informations à gérer.

Dans ce rapport on va présenter l'organisme d'accueil et le contexte général dans un premier chapitre. Dans le deuxième chapitre nous présentons l'analyse des besoins et la conception en UML. Le dernier chapitre est consacré à la présentation des IHM de l'application et sa mise en place.

CHAPITRE 1 : CONTEXTE GENERAL DU PROJET

Le sujet sur lequel repose mon projet de fin d'études est l'informatisation de la gestion du parc automobile du ministère de la culture. Avant d'entamer le vif du sujet, il est nécessaire de décrire en premier lieu ce que la gestion d'un parc automobile.

1.1) PROBLEMATIQUE ET OBJECTIFS DU PROJETS

La gestion manuelle du parc automobile devient de plus en plus difficile compte tenu de la diversité des tâches à accomplir et du nombre important des employés qui ont besoin de trouver les véhicules disponibles pour leurs missions.

Parmi les tâches fastidieuses nous avons aussi la gestion des dotations du carburant qui nécessite beaucoup d'attention , et de temps pour une vérification manuelle et régulière des nombre d'acquisition de chaque dotation et pour chaque mois .Aussi on ne peut pas s'en passer de la gestion des vélomoteurs ,ou chaque vélomoteur est associé à un fonctionnaire et on doit garder trace de chaque vélomoteur en fonction de son possesseur ainsi que la dotation donnée à ce dernier pendant chaque mois , ce qui rend la gestion manuelle très difficile et non fiable pour les données statistiques .

En outre, le non informatisation de la gestion de parc automobile rend la circulation des informations très lente. En plus, l'absence d'une base de données et le non archivage des documents papiers utilisés rendent les différentes tâches quasiment impossibles l'établissement des statistiques fiables.

Par ailleurs, les responsables du parc sont souvent en déplacement, ce qui retarde les mises à jour du tableau de planning.

Mon projet de fin d'étude s'oriente dans ce cadre, qui consiste à concevoir un système qui assure la rapidité, la fiabilité et la facilité des traitements. La solution que nous avons suggérée est une application qui sera hébergé par la suite dans le réseau local du service. Cette application va traiter au début la gestion des véhicules, la gestion des missions, la gestion des dotations, la gestion des réparations, ainsi que la gestion des profils des utilisateurs.

1.2) PRESENTATION DE L'ORGANISME D'ACCUEIL

1.2.1) Historique :

En 1968, les affaires culturelles étaient gérées par une Direction nommée le ministère de l'éducation Nationale, Tourisme, artisanat, Jeunesse et sport. De 1972 à 1974, cette division a été détachée de l'éducation et de la formation des cadres, puis relié aux affaires Islamiques et Habous jusqu'à ce qu'elle est devenue indépendante, et elle portait le nom du ministère d'état, chargée de la culture.

1.2.2) L'organigramme du ministère

- Le ministre ;
- Le secrétariat général ;
- L'inspection générale ;
- Le Cabinet ;
- Le bureau d'ordre ;
- La direction du patrimoine culturel ;
- La direction des arts ;
- La direction du livre, des bibliothèques et des archives ;
- La direction des affaires administratives et financières ;
- La division des systèmes d'information ;
- Le service du service des études de la documentation et des programmes Informatique.

Le service de développement informatique :

Notre stage de fin d'études a eu lieu au sein la Division des Systèmes d'Information (D.S.I). Cette Division a pour but :

- la gestion du parc informatique.
- Le développement d'applications utilisées au sein du ministère.
- L'étude et la documentation.

1.3) CAHIER DES CHARGES DE L'APPLCATION

L'application a pour but :

La gestion des véhicules (Ajout, Modification, Suppression et Consultation)

La gestion des chauffeurs (Ajout, Modification, Suppression et Consultation)

La gestion des vélomoteurs (Ajout, Modification, Suppression et Consultation)

La gestion des dotations de carburant pour les véhicules (des dotations qu'on donne au chauffeur d'un véhicule mensuellement réparties sur plusieurs acquisitions)

La gestion des dotations pour les vélomoteurs (des dotations qu'on donne au chauffeur d'un vélomoteur mensuellement)

La gestion des missions pour les chauffeurs

La gestion des réparations

La gestion des vidanges

Possibilités de consulter des situations concernant les éléments gérés.

N.B. La dotation mensuelle d'un véhicule ou vélomoteur est plafonnée ; donc le but ici est de suivre et contrôler les dotations données à chaque véhicule et vélomoteur.

Le plus important dans la partie des dotations de carburant est que les montants saisis doivent être mis en parallèle dans une autre situation que l'on peut consulter ou imprimer par date ou par véhicule ou état global.

Description des rôles des personnes qui vont accéder à l'application « Gestion du parc auto » :

- Fonctionnaire: peut écrire un message, effectué une recherche ou lister les véhicules disponibles.
- Chauffeur : lister ces missions, ces dotations.
- Administrateur : gestion des utilisateurs et administration.

1.4) PLANNING DU PROJET

Pour mieux débuter un projet, il faut définir un planning qui guidera à la réalisation de ce projet. Le planning permet alors de définir une décomposition du projet en plusieurs phases (illustrée ci-après) sous forme arborescente. Le planning que nous avons adopté pour la réalisation du projet est défini ci-dessous :

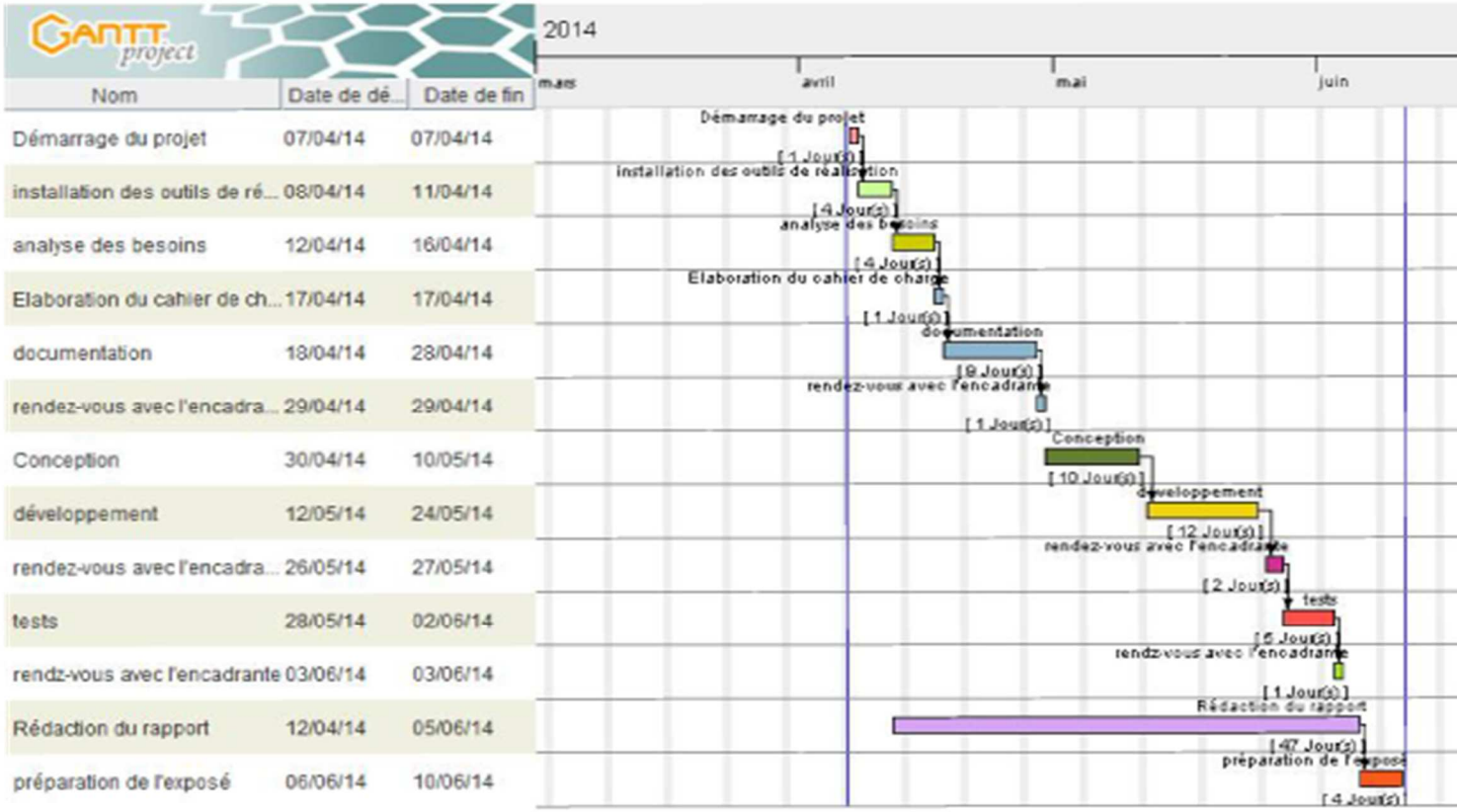

FIGURE 1: DIAGRAMME DE GANTT

CHAPITRE 2 : ANALYSE DES BESOINS ET CONCEPTION

2.1) CYCLE DE DEVELOPPEMENT PROCESSUS EN Y

Pour distinguer les besoins techniques et fonctionnels de l'application nous avons adopté le processus en Y.

Le processus en Y ou Two Track Unified Process (2TUP) est une variante de l'Unified Process qui s'articule plus sur les aspects techniques. En effet ce processus permet de gérer la complexité technologique, en lui réservant toute une branche de son cycle, dissociée de l'aspect fonctionnel. Il permet donc en plus, de réduire le risque technologique. Y est de nature itérative et incrémentale et permet, de ce fait, de faire des itérations dans ses différentes phases. Il faut signaler aussi que 2TUP offre un cadre de gestion des risques à travers la déclaration et le suivi d'une liste des risques identifiés

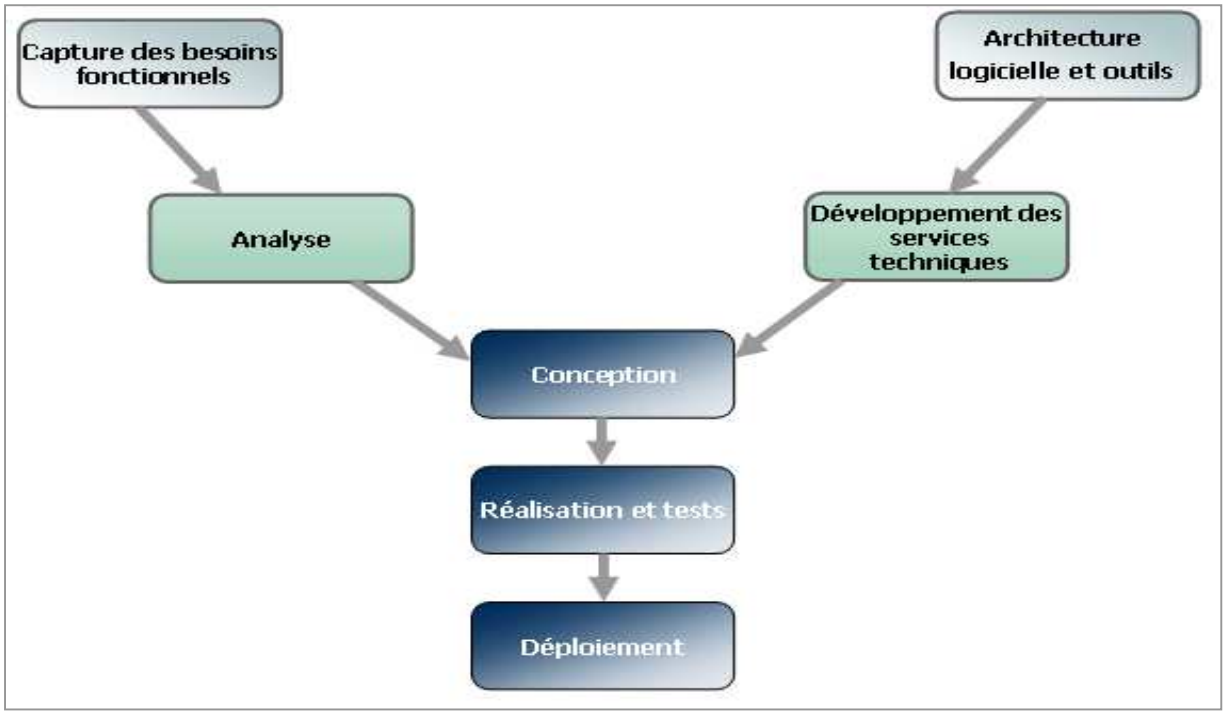

FIGURE 2 : CYCLE DE DEVELOPPEMENT EN Y

Le processus se compose de deux branches, une fonctionnelle et l'autre technique. Ces deux branches se rencontrent dans la partie de la réalisation d'où son appellation de cycle

en Y. la branche fonctionnelle est composée de deux phases : capture des besoins fonctionnels et analyse. L'objectif de la phase de capture des besoins, est de dégager et modéliser les besoins fonctionnels du projet. Cette activité repose sur le modèle des cas d'utilisation du système et utilise les éléments de modélisation acteurs et cas d'utilisation. Il s'agit donc, de construire les diagrammes des cas d'utilisation, de séquence et la description textuelle des cas d'utilisation. La construction d'une maquette du système sera de forte utilité par la suite pour représenter les besoins fonctionnels.

La phase d'analyse précise et structure les besoins pour mieux les comprendre et concourir à un modèle plus stable, au moyen de paquetages et des classes d'analyse. En effet cette phase, fournit les classes d'analyse, les réalisations de cas d'utilisation (par des collaborations entre objets d'analyse), les paquetages d'analyse et le modèle du système de point de vue analyse. Afin de dégager les classes du modèle, les analystes étudient les livrables de la phase précédente et déduisent une partie du modèle.

Ensuite ils raffinent le modèle ainsi construit par l'élaboration des diagrammes de séquence ou de collaboration des objets qui réalisent les cas d'utilisation.

En ce qui concerne la branche technique, les objectifs sont :

- Rassembler les besoins techniques : sécurité, montée en charge, intégration à L'existant et autres.
- Elaborer une architecture logicielle et applicative qui répond aux contraintes Dégagées.
- Identifier les besoins en Framework techniques afin de pallier aux manques de la technologie. Exemple : gestion de la touche Back des navigateurs, formulaires de saisie interactifs, personnalisation de l'interface graphique, moteur de persistance Objet / Relationnel avec expressions SQL / Objet.
- Proposer des règles de développement afin d'industrialiser l'implémentation (gestion des exceptions, règles de nommage, règles de codage, ...).

Après les branches fonctionnelle et technique, le processus propose la phase de conception.

La phase de conception consiste à reprendre le modèle d'analyse et le refaire selon les décisions prises dans la branche techniques. Il s'agit donc d'adapter son modèle d'analyse à l'architecture adoptée et aux Framework techniques choisis ou développés. C'est aussi dans cette phase que se fait la conception des modèles d'implémentation à travers le choix des designs patterns appropriés ou la construction de nouveaux modèles.

Après la phase de conception, les développeurs abordent la phase de codage et tests. Il s'agit là de coder les classes obtenues dans la phase précédente. Ensuite, ces classes sont testées unitairement en vue d'être intégrées dans le système tout entier. Une fois construit, le système est prêt pour l'activité de test. Cette activité a pour but de tester le système fabriqué en implémentation. Trois artefacts sont produits et utilisés :

- *Cas de test : ce qu'il faut tester dans le système. Ils s'apparentent aux cas d'utilisation ;*
- *Procédures de test : la démarche qui permet de dérouler le test ;*
- *Composants de test : l'environnement nécessaire pour pourvoir effectivement exécuter les cas de test.*

Enfin, la phase de déploiement permet la mise en marche du logiciel construit et la formation des gens qui vont l'utiliser par la suite.

2.2) SPECIFICATION ET ANALYSE DES BESOINS TECHNIQUES

Avant de rentrer dans le vif du sujet, il est nécessaire de posséder quelques connaissances sur le Framework et les outils utilisé au cours de ce stage.

Dans cette partie on va citer les différents outils utilisés pour la réalisation du projet.

FIGURE 3 : OUTILS POUR LA REALISATION

2.2.1) LE LANGAGE DE MODELISATION UNIFIE : UML 2.0

La plateforme du développement du projet est basée sur JEE c'est pourquoi UML a été opté comme langage de modélisation car il est plus approprié pour les projets orienté objet.

Dans le cadre de la conception orientée objet, un langage unifié pour la modélisation a été développé : UML (« Unifed Modeling Language »). Il s'agit d'un langage graphique de modélisation objet permettant de spécifier, de construire, de visualiser et de décrire en détail un système logiciel. Il est issu de la fusion de plusieurs méthodes dont «Booch» et

«OMT», et est adapté à la modélisation de tout type de système. La modélisation d'un système s'effectue indépendamment de toute méthode ou de tout langage de programmation.

UML est un langage, il comprend un vocabulaire et un ensemble de règles centrées sur la représentation conceptuelle et physique d'un système logiciel. Ses domaines d'utilisation sont :

- Visualisation d'un système ;
- Spécification d'un système ;
- Construction d'un système ;
- Documentation d'un système.

Ce langage de modélisation unifié favorise :

➢Une meilleure communication entre les intervenants dans un projet : il offre des moyens de capture des connaissances sur un sujet à travers divers points de vue (ces points de vue sont fournis par ses différents diagrammes).

➢Une bonne compréhension du problème : le système à étudier sera traité suivant différents angles et suivant les différents cas d'utilisation de ce système.

Le modèle conceptuel d'UML définit trois sortes de « briques » de base :

- Les éléments, qui sont les abstractions essentielles à un modèle ;
- Les relations, qui constituent des liens entre ces éléments ;
- Les diagrammes, qui regroupent des éléments et des liens au sein de divers ensembles.

UML 2.0 définit également treize types de diagrammes dépendants hiérarchiquement et se complètent, de façon à permettre la modélisation d'un projet tout au long de son cycle de vie, et ils sont divisés en trois catégories :

1. diagrammes statiques (appelés aussi diagrammes structurels) : diagrammes de classes, d'objets, de composants, de déploiement, de paquetage et de structure composite.

2. diagrammes dynamiques (appelés aussi diagrammes d'interaction) : diagrammes de séquence, de communication, de vue d'ensemble des interactions ou global d'interactions et diagramme de temps.
3. Diagrammes comportementaux : diagrammes de cas d'utilisation, d'activité, et d'états.

Durant l'étude de notre système, il a été utilisé quatre diagrammes d'UML, il s'agit de diagrammes de :

- Communication
- Cas d'utilisation
- séquence
- Classes

Pour la mise en œuvre des diagrammes, j'ai opté pour :

Enterprise Architect

Enterprise Architect est un logiciel de modélisation et de conception UML, édité par la société australienne Sparx Systems. Couvrant, par ses fonctionnalités, l'ensemble des étapes du cycle de conception d'application, il est l'un des logiciels de conception et de modélisation les plus reconnus [1].

2.2.2) LE MODELE MVC 2

Le design pattern Modèle-Vue-Contrôleur (MVC) est un pattern architectural qui sépare les données (le modèle), l'interface homme-machine (la vue) et la logique de contrôle (le contrôleur). Ce modèle de conception impose donc une séparation en 3 couches :

- Le modèle : Il représente les données de l'application. Il définit aussi l'interaction avec la base de données et le traitement de ces données.
- La vue : Elle représente l'interface utilisateur, ce avec quoi il interagit. Elle n'effectue aucun traitement, elle se contente simplement d'afficher les données

que lui fournit le modèle. Il peut tout à fait y avoir plusieurs vues qui présentent les données d'un même modèle.

- Le contrôleur : Il gère l'interface entre le modèle et le client. Il va interpréter la requête de ce dernier pour lui envoyer la vue correspondante. Il effectue la synchronisation entre le modèle et les vues.

La synchronisation entre la vue et le modèle se passe avec le pattern Observer.

Il permet de générer des événements lors d'une modification du modèle et d'indiquer à la vue qu'il faut se mettre à jour.

Voici un schéma des interactions entre les différentes couches :

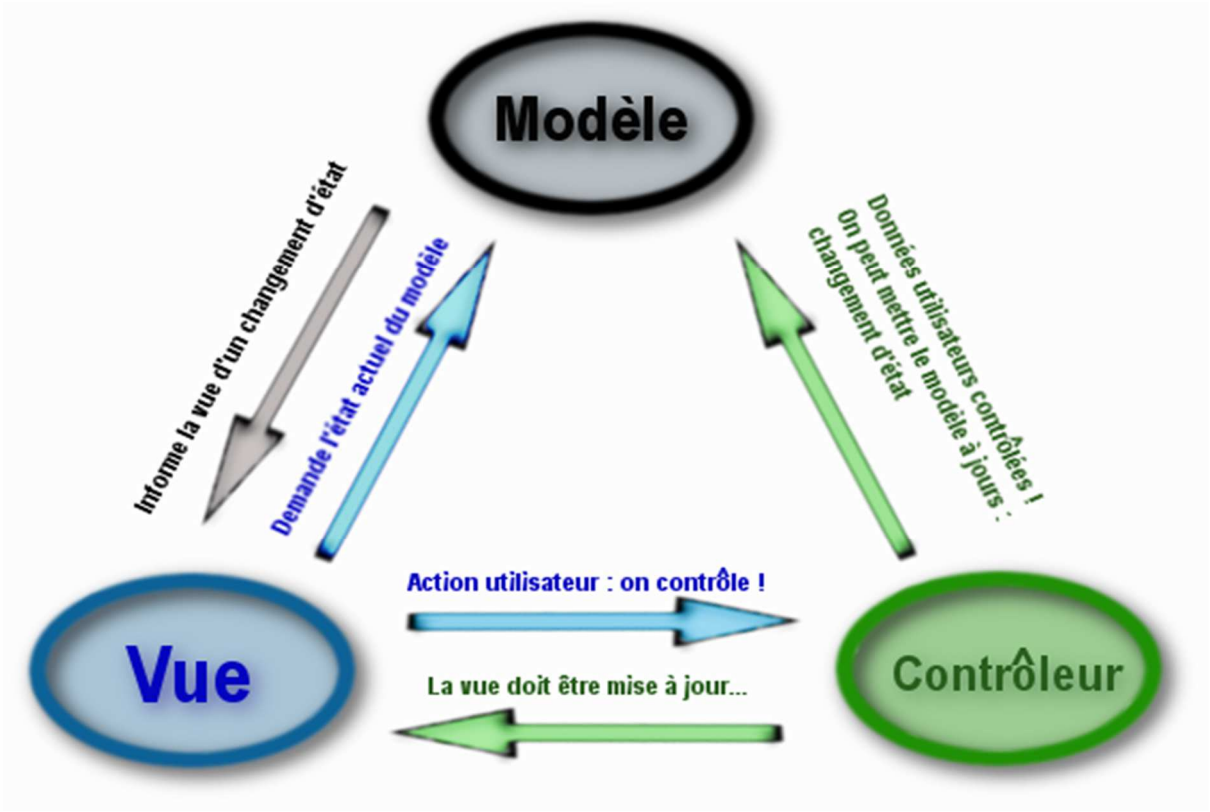

FIGURE 4 : MODELE MVC

Ce modèle de conception permet principalement 2 choses :

-Le changement d'une couche sans altérer les autres. C'est-à-dire que comme toutes les couches sont clairement séparées, on doit pouvoir en changer une pour, par exemple,

remplacer Swing par SWT sans porter atteinte aux autres couches. On pourrait aussi donc changer le modèle sans toucher à la vue et au contrôleur. Cela rend les modifications plus simples.

-La synchronisation des vues. Avec ce design pattern, toutes les vues qui montrent la même chose sont synchronisées. Il faut tout de même garder en mémoire, que la mise en œuvre de MVC dans une application n'est pas des plus simples. En effet, ce modèle de conception introduit tout de même un niveau de complexité assez élevé. De plus, implémenter MVC dans votre application nécessite une bonne conception dès le départ. Ce qui peut prendre du temps. Ce pattern n'est donc à conseiller que pour les moyennes et grandes applications [3].

« JSF et la plupart des frameworks web encouragent la séparation des problèmes en utilisant des variantes du modèle MVC. Ce dernier est un modèle d'architecture permettant d'isoler la logique métier de l'interface utilisateur car la première ne se mélange pas bien avec la seconde : leur mélange produit des applications plus difficiles à maintenir et qui supportent moins bien la montée en charge. Dans la section "JavaServer Pages" du chapitre précèdent, nous avons vu une page JSP qui contenait à la fois du code Java et des instructions SQL : bien que ce soit techniquement correct, imaginez la difficulté de maintenir une telle page... Elle mélange deux types de développement différents (celui de concepteur graphique et celui de programmeur métier) et pourrait finir par utiliser bien plus d'API encore (accès aux bases de données, appels d'EJB, etc.), par gérer les exceptions ou par effectuer des traitements métiers complexes. Avec MVC, l'application utilise un couplage faible, ce qui facilite la modification de son aspect visuel ou des règles métiers sous-jacentes sans pour autant affecter l'autre composante [10].

2.2.3) JAVA PERSISTENCE API

La Java Persistence API (abrégée en JPA), est une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java.

La Java Persistence API est à l'origine issue du travail du groupe d'experts JSR 220.

La persistance dans ce contexte recouvre 3 zones :

- l'API elle-même, définie dans le paquetage javax.persistence
- le langage Java Persistence Query (JPQL)
- l'objet/les métadonnées relationnelles

La Java Persistence API repose essentiellement sur l'utilisation des annotations, introduites dans Java 5. Elles permettent de définir très facilement, et précisément des objets métier, qui pourront servir d'interface entre la base de données et l'application [5].

EclipseLink

EclipseLink est un framework open source de mapping objet-relationnel pour les développeurs Java. Il fournit une plateforme puissante et flexible permettant de stocker des objets Java dans une base de données relationnelle et/ou de les convertir en documents XML.

EclipseLink est dérivé du projet TopLink de la société Oracle. Il supporte un certain nombre d'API relatives à la persistance des données et notamment la JPA[1].

2.2.4) JAVA ENTREPRSE EDITION (JEE)

La plateforme Java entreprise (Java EE) est un ensemble de spécifications coordonnées et pratiques qui permettent ensemble des solutions pour le développement, le déploiement, et de la gestion des applications multi tiers centralisées sur un serveur. Construit sur la plateforme de Java 2 édition standard (Java SE), la plateforme Java EE

ajoute les possibilités nécessaires pour fournir une plateforme complète, stable, sécurisée, et rapide de Java au niveau entreprise.

La plateforme entreprise fournit un ensemble de services permettant aux composants de dialoguer entre eux:

- HTTP et HTTPS
- Java Transaction API (JTA)
- Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP)
- Java Interface Definition Language (Java IDL)
- Java DataBase Connectivity (JDBC)
- Java Message Service (JMS)
- Java Naming and Directory Interface (JNDI)
- API JavaMail et JAF (JavaBeans Activation Framework)
- Java API for XML Processing (JAXP)
- Java EE Connector Architecture
- Gestionnaires de ressources
- Entreprise Java Beans (EJB)
- Java Server Pages (JSP)
- Servlet
- Java API for XML Web Services (JAX-WS, anciennement JAX-RPC)
- SOAP with Attachments API for Java (SAAJ)
- Java API for XML Registries (JAXR)[6].

Les EJB 3:

La spécification EJB 3 est le résultat d'un ensemble d'évolutions et d'innovations qui se sont installées sur le marché sans pour autant faire partie intégrante de la

spécification Java EE 5. De nombreux Frameworks et technologies ont approuvé certaines principes et modèles de développement. La spécification a su prendre parti des avantages de chacune de ces innovations afin d'élaborer la meilleure structure possible pour les EJB[8].

2.2.5) GLASSFISH 4.0

GlassFish est un serveur d'applications compatible Java EE créé par Anthony Gaspard.

GlassFish est le nom du serveur d'applications Open Source Java EE 5 et désormais Java EE 6 avec la version 3 qui sert de socle au produit Oracle GlassFish Server1 (anciennement Sun Java System Application Server2 de Sun Microsystems). Sa partie Toplink persistence3 provient .d'Oracle. C'est la réponse aux développeurs Java désireux d'accéder aux sources et de contribuer au développement des serveurs d'applications de nouvelle génération [1].

2.2.6) MYSQL

MySQL est un système de gestion de base de données relationnelle (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde1, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, Informix et Microsoft SQL Server.Son nom vient du prénom de la fille du cocréateur Michael Widenius, My. SQL fait allusion au Structured Query Language, le langage de requête utilisé [1].

phpMyAmin 4.2.2

phpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée en PHP et distribuée sous licence GNU GPL.

Il s'agit de l'une des plus célèbres interfaces pour gérer une base de données MySQL sur un serveur PHP. De nombreux hébergeurs, qu'ils soient gratuits ou payants, le proposent ce qui permet à l'utilisateur de ne pas avoir à l'installer.

Cette interface pratique permet d'exécuter, très facilement et sans grandes connaissances dans le domaine des bases de données, de nombreuses requêtes comme les créations de table de données, les insertions, les mises à jour, les suppressions, les modifications de structure de la base de données. Ce système est très pratique pour sauvegarder une base de données sous forme de fichier .sql et ainsi transférer facilement ses données. De plus celui-ci accepte la formulation de requêtes SQL directement en langage SQL, cela permet de tester ses requêtes par exemple lors de la création d'un site et ainsi de gagner un temps précieux [11].

2.2.7) LE FRAMEWORK JSF

Java Server Faces (JSF) est une technologie dont le but est de proposer un Framework qui facilite et standardise le développement d'applications web avec Java. Son développement a tenu compte des différentes expériences acquises lors de l'utilisation des technologies standards pour le développement d'applications web.

Le grand intérêt de JSF est de proposer un Framework qui puisse être mis en œuvre par des outils pour permettre un développement de type RAD (Rapid Application

Développent) pour les applications web et ainsi faciliter le développement des applications de ce type. (…) Ainsi de par sa complexité et sa puissance, JSF s'adapte

Parfaitement au développement d'applications web complexes en facilitant leur écriture. (…)JSF est une technologie utilisée côté serveur dont le but est de faciliter le développement de l'interface utilisateur en séparant clairement la partie « interface » de la partie « métier » d'autant que la partie interface n'est souvent pas la plus compliquée mais la plus fastidieuse à réaliser [9].

PrimeFaces

Primefaces est un jar facile à ajouter dans les projets jsf pour obtenir un jeu de composants additionnel beaucoup avancé qui prend en compte les nouveautés des technologies du web surtout Ajax.

Il existe des alternatives à Primefaces dont les plus connues et les plus anciennes sont RichFaces de la communauté JBoss et IceFaces qui s'est suicidé à partir de sa version 3 en copiant en totalité le code de primefaces[4] .

jQuery

jQuery est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant Ajax) et HTML, et a pour but de simplifier des commandes communes de JavaScript. La première version date de janvier 2006.

La bibliothèque contient notamment les fonctionnalités suivantes :

Parcours et modification du DOM (y compris le support des sélecteurs CSS 1 à 3 et un support basique de XPath) ;

- Événements ;
- Effets visuels et animations ;

- Manipulations des feuilles de style en cascade (ajout/suppression des classes, d'attributs…) ;
- Ajax ;
- Plugins ;
- Utilitaires (version du navigateur web…)[1].

Ajax

L'architecture informatique Ajax (acronyme d'Asynchronous JavaScript and XML) permet de construire des applications Web et des sites web dynamiques interactifs sur leposte client en se servant de différentes technologies ajoutées aux navigateurs web entre 1995 et 2005.

Ajax combine JavaScript, les CSS, XML, le DOM et le XMLHttpRequest afin d'améliorer maniabilité et confort d'utilisation des Applications Internet Riches (abr. RIA)1,2 :

-DOM et JavaScript permettent de modifier l'information présentée dans le navigateur en respectant sa structure ;

-L'objet XMLHttpRequest sert au dialogue asynchrone avec le serveur Web ;

XML structure les informations transmises entre serveur Web et navigateur [1].

Netbeans IDE 7.4 :

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Development Kit JDKest requis pour les développements en Java.

NetBeans constitue par ailleurs une plateforme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE NetBeans s'appuie sur cette plateforme.

L'IDE Netbeans s'enrichit à l'aide de greffons[1].

2.2.7) SPRING SECURITY 3.0.6

Spring Security, est un contrôleur d'authentification flexible et puissant pour assurer une application Web Java basé sur Spring…[2].

SPRING est effectivement un conteneur dit " léger ", c'est-à-dire une infrastructure similaire à un serveur d'applications JEE. 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 JEE et des EJBs). C'est en ce sens que SPRING est qualifié de conteneur " léger "[1].

2.2.8) IREPORT 5.5.1

iReport est un logiciel de création WYSIWYG (What You See Is What You Get) de modèles de documents pour JasperReports.

Il permet donc de produire de manière assez intuitive des fichiers .jrxml (fichiers XML) exploitables par JasperReports pour générer des rapports au sein d'une application Java. Le format de rapport généré dépend ensuite de JasperReports et du code utilisé (html, pdf, csv...).

C'est une application Java pure qui nécessite l'installation d'une JVM pour s'exécuter [1].

JasperReports 5.5.1 :

JasperReports est un outil de Reporting Open Source, offert sous forme d'une bibliothèque qui peut être embarquée dans tous types d'applications Java.

jasperReports se base sur des fichiers XML (dont l'extension est en général .jrxml) pour la présentation des états. Il peut être couplé à iReport (outil WYSIWYG) ou JasperStudio (plugin Eclipse équivalent) pour faciliter sa mise en œuvre dans une application Java, classique ou orientée web.

La version complète de l'application se nomme JasperReports Server (JRS) depuis la V4 (anciennement JasperServer) et propose un serveur d'application et la création de rapports web [1].

2.3) SPECIFICATION ET ANALYSE DES BESOINS FONCTIONNELS

Dans ce paragraphe on va décrire les besoins fonctionnels du cahier des charges. Ils ont été identifiés lors des réunions tenus avec le chef des projets, et le responsable de parc au ministère de culture.

Nous avons déjà définir le contexte générale du projet dans le chapitre précèdent, ainsi ce dernier nous a permis de comprendre les différents difficultés rencontrées dans la gestion manuelle du parc automobile.

Apres avoir analysé le processus de fonctionnement de la gestion manuelle du parc automobile, on peut identifier les points de dysfonctionnement et répertorier les

contraintes à prendre en compte. Il s'agira ici d'évaluer et de critiquer la situation actuelle de la gestion du parc automobile en termes d'organisation et de méthodes de travail.

2.3.1) CAPTURE DES BESOINS FONCTIONNELS

Cette phase de l'application a pour objectif de définir la frontière fonctionnelle entre le système et son environnement.

Afin de raffiner les objectifs définis dans e contexte générale du projet, les deux questions suivantes doivent-être répondues : quels sont les utilisateurs du système ? Et qu'attendent-ils de ce système ?

Pour trouver la réponse, nous avons opté pour la démarche suivante :

1. Définir les acteurs du système.
2. Lister ce que doit faire le système de point de vue de son utilisateur

2.3.2) ACTEURS DU SYSTEME

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

Un acteur peut consulter et/ou modifier directement l'état du système, en émettant et/ou en recevant des messages susceptibles d'être porteurs de données [7].

L'étude de la topologie des acteurs de ce système a conduit à la classification suivante :

Acteur	Type	Description
Administrateur	Humain	C'est un privilège unique, il n'y a qu'un seul administrateur du système, il se charge de la gestion des acteurs.
Chauffeur	Humain	Aura accès après authentification à des fonctionnalités restreint de système, ce dernier il ne peut que lister ces missions, ces dotations de carburant, et les véhicules.
Fonctionnaire	Humain	Aura accès après authentification à la page qui affiche la liste des véhicules disponibles.

Tableau 1: acteurs du système

2.3.3) LES FONCTIONNALITE PRINCIPALES

Notre système doit résoudre les problèmes rencontrés dans la gestion manuelle des ressources et prendre en compte les perspectives d'évolution et les besoins des utilisateurs. Pour y arriver, notre boulot consistera à mettre en place un système dont les fonctionnalités permettent :

- Une meilleure réparation des véhicules entre les différents chercheurs pour leurs missions.
- Une bonne gestion des chauffeurs du parc.
- Une gestion efficiente des documents.
- Un meilleur suivi des acquisitions de chaque dotation de carburant.
- Un suivi facile des vélomoteurs du parc.
- Une gestion efficace des réparations et des vidanges de chaque véhicule du parc.

- Un accès et une circulation aux informations en temps réel.
- La rapidité, la facilité et la fiabilité des traitements.
- L'archivage, la sécurité et la confidentialité des données.

a)La gestion administrative du parc :

- Cette application gère l'administration complète d'un ensemble de modules tels que : la gestion des véhicules, la gestion des vélomoteurs, la gestion des dotations de carburant, la gestion des missions, la gestion des réparations, la gestion des réservations, la gestion des utilisateurs.
- Elle doit permettre à l'administrateur du système de modifier les informations concernant les missions (date_Mission), (observation) aussi elle doit permettre la modification des informations des véhicules (changer la situation d'une véhicule), des vélomoteurs …
- Elle doit permettre à l'administrateur d'enregistrer/supprimer les informations d'un véhicule, traiter les réservations, mettre à jour une réparation, afficher la liste les notifications de vidange …
- Elle doit être dotée aussi d'une fonction de recherche multicritère ainsi que le tri des données.
- L'administrateur peut aussi supprimer une mission, un véhicule…
- Elle doit faciliter le remplissage des champs des formulaires.
- Elle doit permettre à l'administrateur de gérer les profils des autres utilisateurs.
- Elle permettre l'affichage des statistiques de la consommation de carburant …
- Elle permettre l'impression des informations des missions, des véhicules…

b) Gestion des consultations (chauffeurs du parc et fonctionnaires) :

Il s'agit d'un ensemble des gestions de consultation intermédiaire entre l'administrateur et déclenchées par l'utilisateur (dans ce cas l'utilisateur est le chauffeur) :

L'application doit permettre au chauffeur d'afficher la liste de ces missions, la liste des acquisitions de carburant qui lui concerne.

L'application doit permettre au fonctionnaire d'afficher la liste des véhicules disponibles

2.3.4) CONTRAINTES DU PROJET

L'objectif ici est d'exposer les différentes catégories de contraintes auxquelles le sous-système de gestion d'administration doit répondre, à savoir les contraintes de sécurité, les contraintes techniques, et les contraintes architecturales.

Contraintes de sécurité :

L'utilisateur du système de gestion peut accéder à cette application. Ceci est garanti via son compte personnel qui possède déjà au sein de son organisation, donc il lui suffit d'introduire son login et son mot de passe, ce qui permet d'éviter toute intrusion non désirable.

Une fois authentifiés, l'utilisateur peut profiter des différents services offerts par l'application.

Contraintes technique :

- L'interface web doit être conviviale et facile à utiliser.
- La vérification de l'authentification se fait au niveau d'un SGBD.

Contraintes architecturales :

- L'application est développée sur la base du Framework JEE.
- L'application communique avec Mysql comme SGBD d'administration.

2.3.5) ARCHITECTURE FONCTIONNELLE DU PROJET

Fonctionnellement l'application peut être découpée en trois parties :

- Administration : Cette partie implémente toutes les tâches administratives : elle permet de gérer les différentes ressources du système.
- Consultation : Cette partie peut être considérée comme un moyen d'interfaçage et de communication entre le système et les utilisateurs qui (Chauffeurs) qui peuvent afficher les acquisitions des dotations, et leurs missions.

  Ainsi, les fonctionnaires peuvent consulter la liste des véhicules disponibles dans le parc.

2.3.6) ANALYSE ET SPECIFICATION

Apres avoir définit les principales gestions qui interviennent dans la gestion du parc automobiles, il est nécessaire d'établir une analyse détaillée pour chaque gestion afin de mieux cerner ses fonctionnalités et pourtant faciliter la conception.

a)La gestion des véhicules

La solution proposé doit permettre d'identifier l'ensemble des véhicules existants et de gérer les informations générales et techniques correspondantes comme :

- Immatriculation
- Carburant
- Type d'immatriculation
- Marque
- Date de mise en circulation

- Service
- Type de véhicule
- Puissance
- Ville
- Poids
- Situation
- Nombre de places

Seulement l'utilisateur qui a le droit de la consultation complète de ce module, il peut :

- Ajouter un véhicule
- Modifier un véhicule
- Supprimer un véhicule
- Afficher la liste des véhicules
- Rechercher un véhicule avec plusieurs critères de recherches

b) La gestion des dotations

Le système doit également gérer les dotations de carburant, chaque dotation est décomposé en plusieurs acquisition, voici les caractéristiques des dotations et des acquisitions :

Dotation :

- **Numéro de dotation**
- **Montant totale**
- **Montant restant**
- **Bénéficiaire**

Acquisition :

- **Numéro d'acquisition**
- **Montant d'acquisition**

Chauffeur :

Le chauffeur peut consulter la liste de ces missions, et la liste des acquisitions, pour ce faire il peut :

- Cliquer sur l'icône des dotations qui se trouve dans le menu, et automatiquement une liste des chaque acquisitions s'affiche avec les informations de chaque acquisition (date, montant) ;
- Trouver les acquisitions avec un critère de recherche.

L'administrateur :

L'administrateur est le responsable de toutes les opérations de mises à jour et de validation des dotations il peut :

- Afficher la liste des dotations.
- Supprimer une dotation.
- Modifier une dotation.
- Ajouter une acquisition.
- Modifier une acquisition.
- Supprimer une acquisition.
- Trouver une dotation avec plusieurs critères de recherche.

c)La gestion des missions

Pour les missions l'administrateur enregistre pour chaque mission le chauffeur, l'observation, le véhicule, le nombre de kilomètre …

- **Chauffeur :**

Il consulte la liste de ces missions, il peut :

- En un simple clic sur le menu (l'icône des missions), afficher la liste des missions avec plusieurs critères de recherche et de trie.
- Trouver une mission.

- **L'administrateur :**

Il gère le processus de la mission, il peut :

- Enregistrer la mission (nombre de kilomètres, véhicule, chauffeur…)
- Imprimer la fiche de sortie.
- Supprimer une mission.
- Modifier une mission.

d) La gestion des vélomoteurs

Le système doit permettre de définir l'ensemble des véhicules existants et de gérer les informations générales et techniques correspondantes comme :

- Id_vélomoteur ;
- Id_marque ;
- Id_carburant ;
- Kilométrage ;
- Immatriculation ;

- Modèle ;
- Puissance ;
- Id_fonctionnaire.

L'administrateur est la seule qui a le droit de consulter ce module, il peut :

- Ajouter un vélomoteur
- Modifier un vélomoteur
- Supprimer un vélomoteur
- Afficher la liste des vélomoteurs
- Rechercher un vélomoteur

e)La gestion des réparations

Par entretien de l'automobile, on entend la vérification de l'état des différents sous-systèmes d'un véhicule (le moteur, la direction, la transmission, la suspension, le freinage, le refroidissement, l'échappement) et le remplacement éventuel de pièces ou de liquides.

Pour une automobile, les interventions principales sont des opérations nécessaires au maintien d'un bon niveau de performance et de sécurité. Un certain nombre de vérifications nécessaires et accessibles à tous doivent être faites régulièrement [1].

Notre système doit permettre à l'administrateur de gérer les réparations et les entretiens, qui sont représentés dans une seule table et qui sont caractérisé par :

- Id_Réparation ;
- Fournisseur ;
- Date_Réparation ;
- Nature_Réparation ;
- Num_Facture ;
- Num_Bon ;
- Num_Immatriculation.

- **L'administrateur :**

Est le seule qui peut consulter et gérer les réparations, il peut :

- Ajouter une réparation ;
- Modifier une réparation ;
- Supprimer une réparation ;
- Consultation de l'historique des réparations.

f) La gestion des fonctionnaires

Il s'agit de gérer les données relatives au fonctionnaire en relation avec les unités de fonction auxquelles ils sont affectés.

- **L'administrateur :**

Est le seule qui a le droit gérer cette fonctionnalité. Il peut :

- Ajouté un nouveau fonctionnaire ;
- Supprimer un fonctionnaire ;
- Modifier les informations d'un fonctionnaire ;
- Afficher les informations d'un fonctionnaire ;
- Chercher un fonctionnaire selon un critère donnée.

g) La gestion des profils utilisateurs

Le système doit assurer :

- La sécurité d'accès des utilisateurs aux données ainsi qu'aux différents modules du système d'information doit être paramétrée par un profile utilisateur.
- La gestion des droits d'accès selon les niveaux hiérarchiques.

L'administrateur :

Il peut :

- Ajouter un nouvel utilisateur ;
- Supprimer un utilisateur ;
- Afficher la liste des utilisateurs ;
- Modifier un utilisateur.

Conclusion :

Dans cette partie, nous avons présentés les différentes fonctionnalités auxquelles doit répondre notre application, cela nous donne une idée sur les classes métiers, les différents cas d'utilisation et les acteurs qui vont interagir avec notre système.

2.4) ANALYSE ET CONCEPTION

Pour représenter le contexte dynamique, il a été nécessaire d'utiliser le diagramme de communication. Ce dernier permet de mettre en évidence les interactions entre les différents objets du système étudié. Il fait également apparaitre les interactions entre des objets et les messages qu'ils s'échangent car il insiste plus particulièrement sur la notion organisationnelle :

- Le système étudié est représenté par un participant central.
- Ce participant central est entouré par d'autres participants symbolisant les différents acteurs.
- Des liens relient le système à chacun des acteurs.
- Sur chaque lien sont montrés les messages en entrée et en sortie du système.

La figure ci-dessous montre les différentes interactions entre le système et les acteurs :

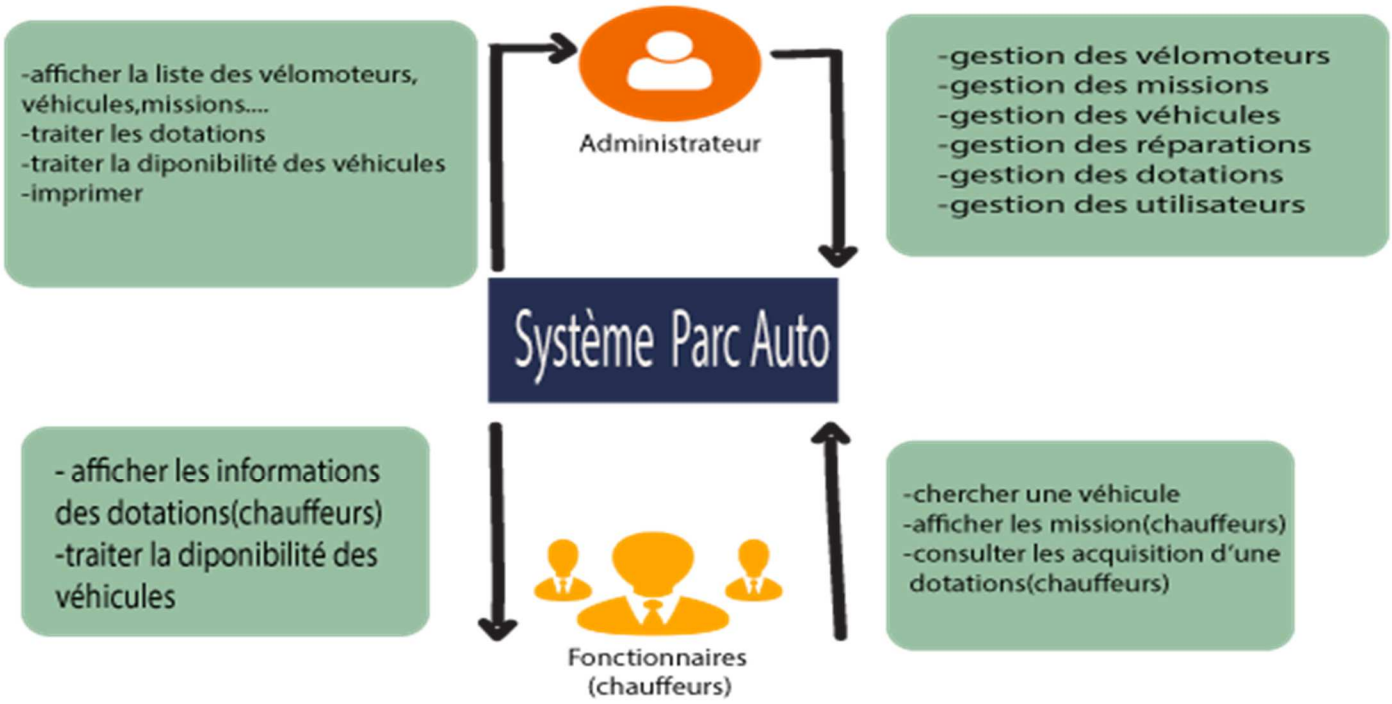

FIGURE 5 : DIAGRAMME DE COMMUNICATION

2.4.1) IDENTIFICATION DES CAS D'UTILISATION

Nous synthétisons dans ce paragraphe tout ce qui a été dit dans cette phase d'analyse.

Nous présentons les diagrammes des cas d'utilisation des différents modules de l'application et introduisons les cas d'utilisation qui les composent.

Ces diagrammes schématisent :

- Le regroupement des fonctionnalités sous forme de cas d'utilisation ;
- Les interactions entre les acteurs et le système ;
- La structuration des cas d'utilisation et leurs interdépendances.

a)Diagramme de cas d'utilisation : Administrateur du parc

Ce sont les fonctionnalités principales que le système doit assurer.

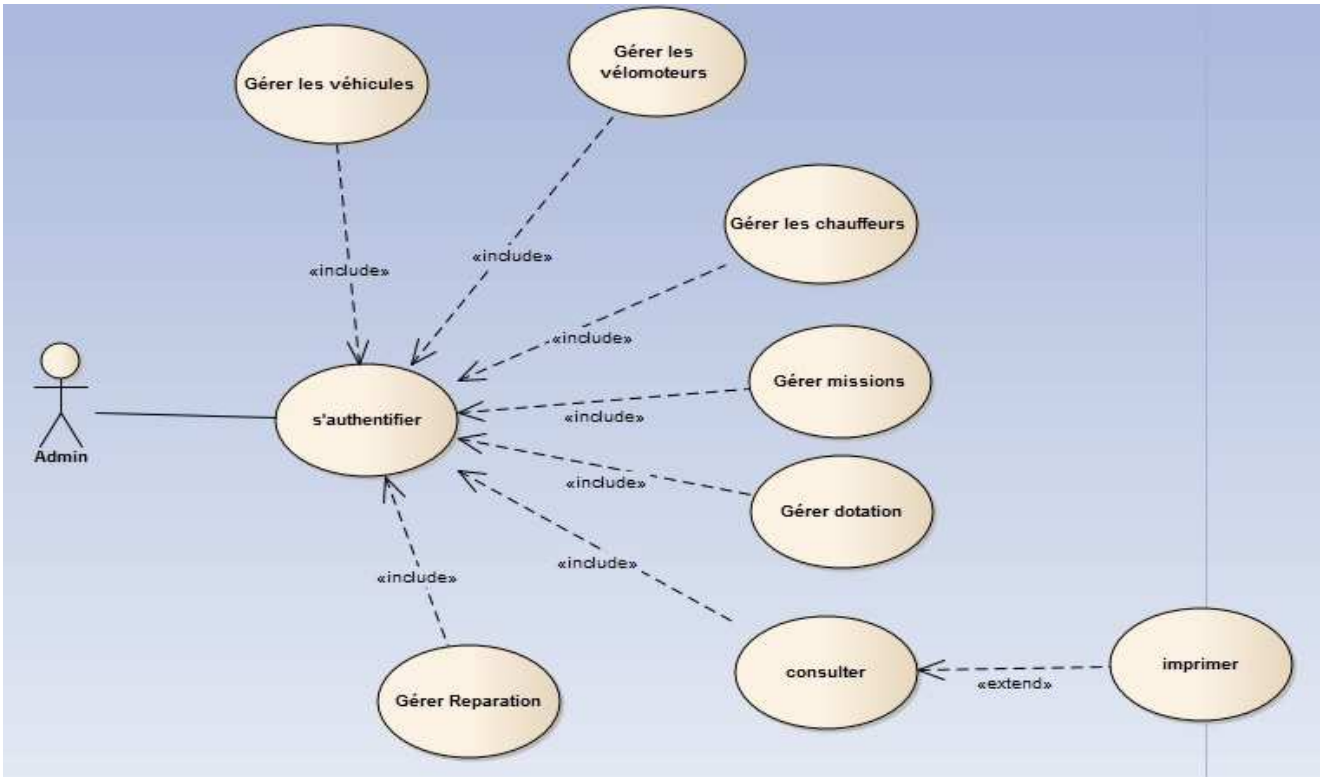

FIGURE 6: DIAGRAMME DE CAS D'UTILISATION ADMINISTRATEUR

Description du cas d'utilisation : Administration du parc	
Acteur	administrateur du parc
Description	Ce cas d'utilisation engendre presque tous les cas d'utilisation de notre système. Il contient plusieurs cas d'utilisation :gestion des :messions vélomoteurs….
Préconditions	L'administrateur doit être authentifié
Poste-condition	Après l'authentification les fonctionnalités proposées par notre application devient tous accessible

TABLEAU 2 : DESCRIPTION DU CAS D'UTILISATION ADMINISTRATEUR

b) Diagramme de cas d'utilisation : Fonctionnaire

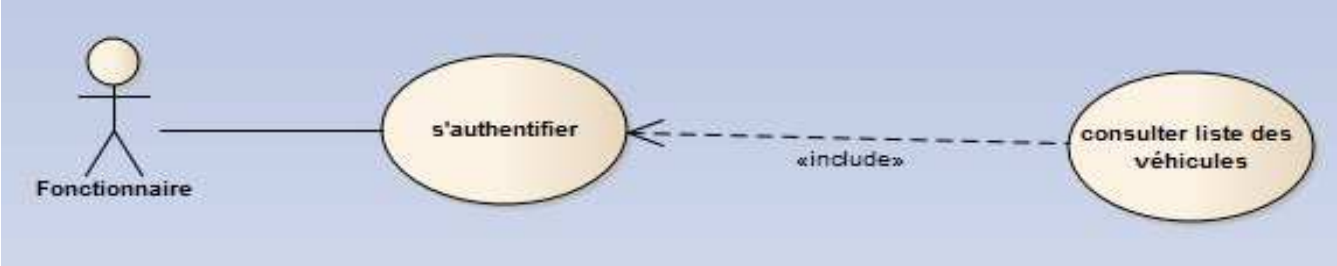

FIGURE 7 : CAS D'UTILISATION 'FONCTIONNAIRE'

Description du cas d'utilisation : fonctionnaire	
Acteur	Fonctionnaire
Description	Ce cas d'utilisation montre ce qu'il peut faire un fonctionnaire. En fait, il ne peut que consulter la liste des véhicules disponible
Préconditions	le fonctionnaire doit être authentifié
Poste-condition	Affichage de la liste des véhicules disponibles dans le parc

TABLEAU 3 : DESCRIPTION DU CAS D'UTILISATION FONCTIONNAIRE

c)Diagramme du cas d'utilisation : chauffeur

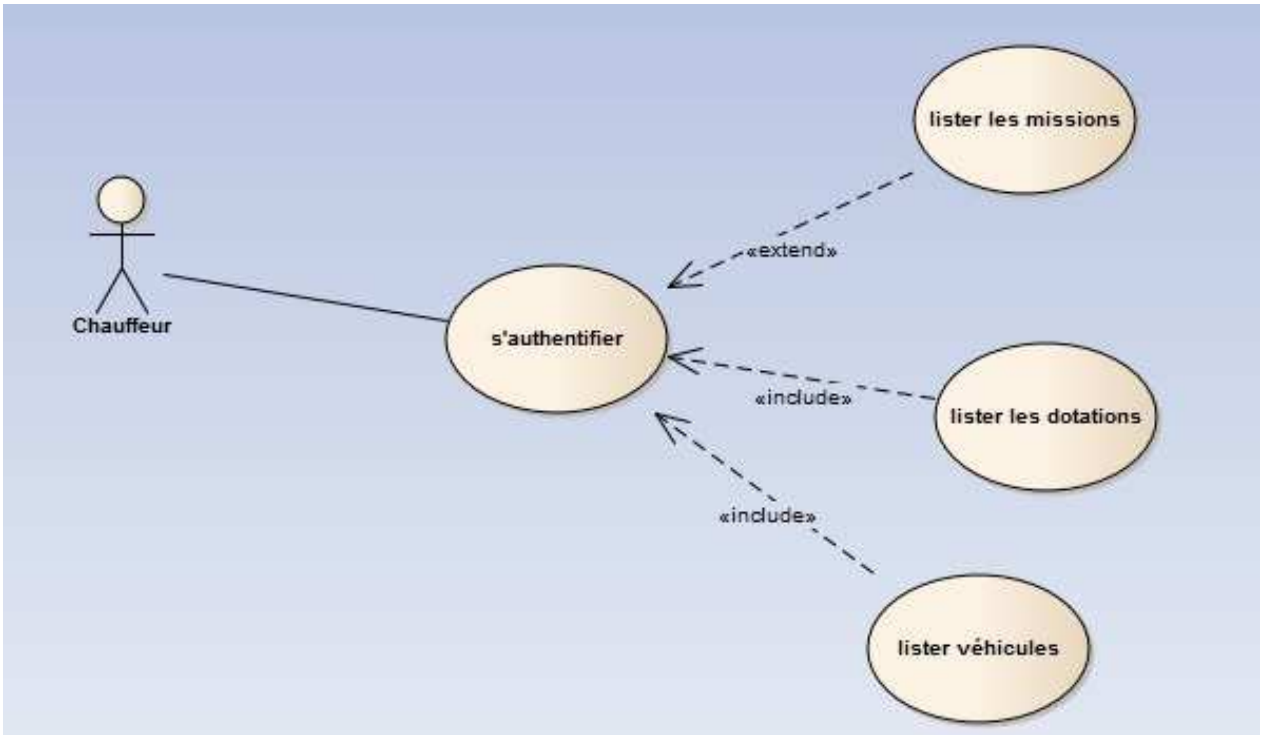

FIGURE 8 : DIAGRAMME DU CAS D'UTILISATION CHAUFFEUR

Description du cas d'utilisation : chauffeur	
Acteur	Chauffeur
Description	Ce cas d'utilisation illustre ce que propose l'application pour Les chauffeurs. Un chauffeur peut : -lister ces missions -lister ces dotations -lister les véhicules
Préconditions	le chauffeur doit être authentifié
Poste-condition	Possibilité de listage

TABLEAU 4 : DESCRIPTION DU CAS D'UTILISATION CHAUFFEUR

2.4.2) DIAGRAMMES DE SEQUENCES

Les acteurs interagissent avec le système par l'intérimaire de messages. C'est pour cette raison, pour chaque cas d'utilisation il a été nécessaire d'élaborer un diagramme de séquence pour illustrer les interactions par échanges de messages des futurs utilisateurs avec le système.

Les diagrammes de séquence présentés ci-dessous dérivent tous les scénarios les plus pertinents.

a)Diagramme de séquence : Authentification

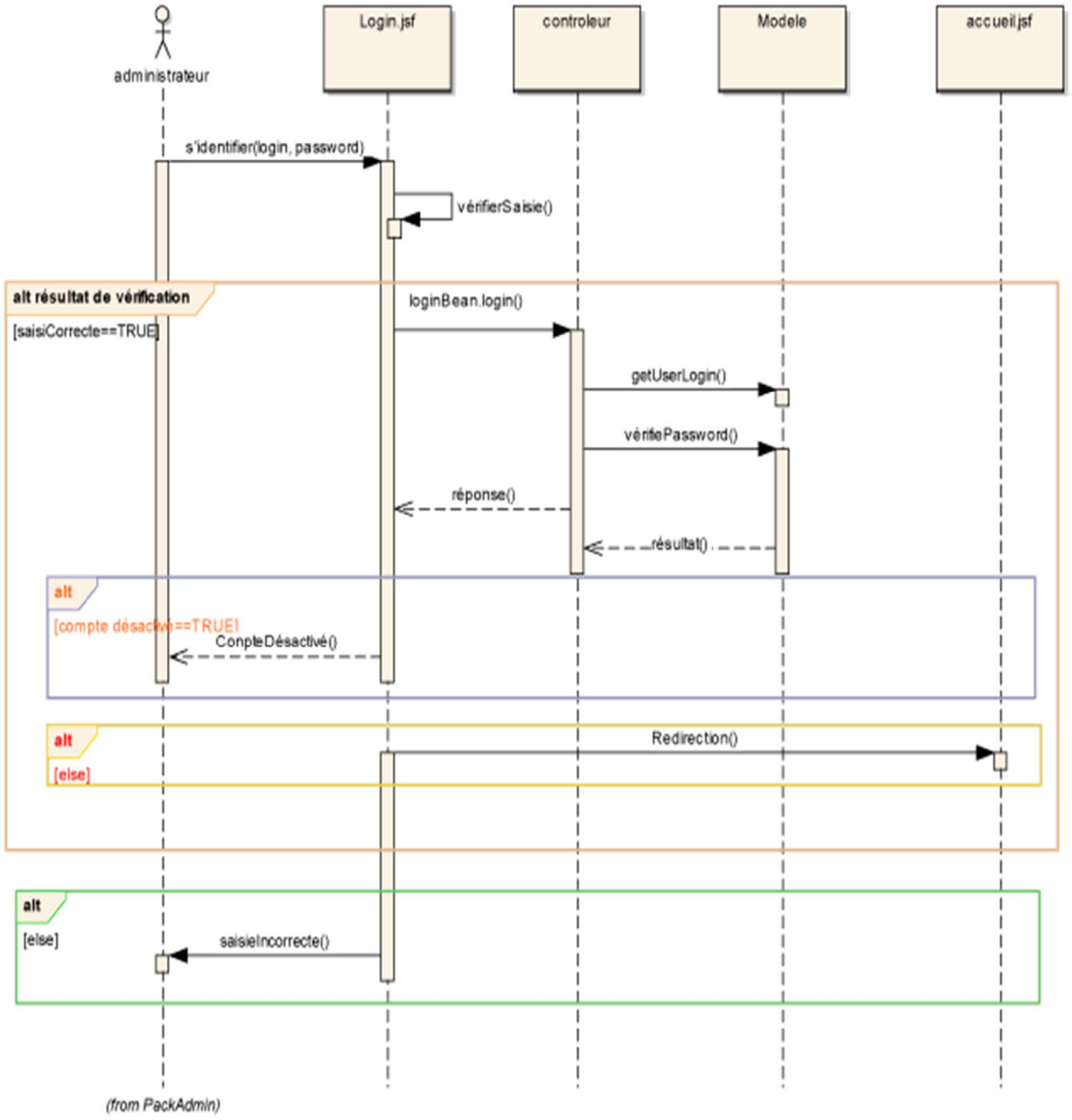

FIGURE 9 : DIAGRAMME DE SEQUENCE 'AUTHENTIFICATION'

En générale, le cas d'utilisation authentification permet à un utilisateur de se connecter à l'application en introduisant son login et son mot de passe dans la page login.jsf.

Lorsque l'utilisateur clique sur le bouton *s'identifier* la page login vérifier les champs saisies et notifie l'utilisateur en cas d'erreur.

La vérification du login et de mot de passe se fait au niveau du contrôleur qui s'en charge de faire la vérification auprès d'une base de données locale(Modèle).

Si l'authentification réussi, l'utilisateur était redirigé automatiquement vers la page d'accueil .Dans le cas contraire, un l'un de ces deux messages d'erreurs serait affiché :

- Vos identifiants sont incorrects! merci de vérifier
- Votre compte est désactivé. Merci de contactez L'administrateur

b) Diagramme de séquence : afficher liste des véhicules

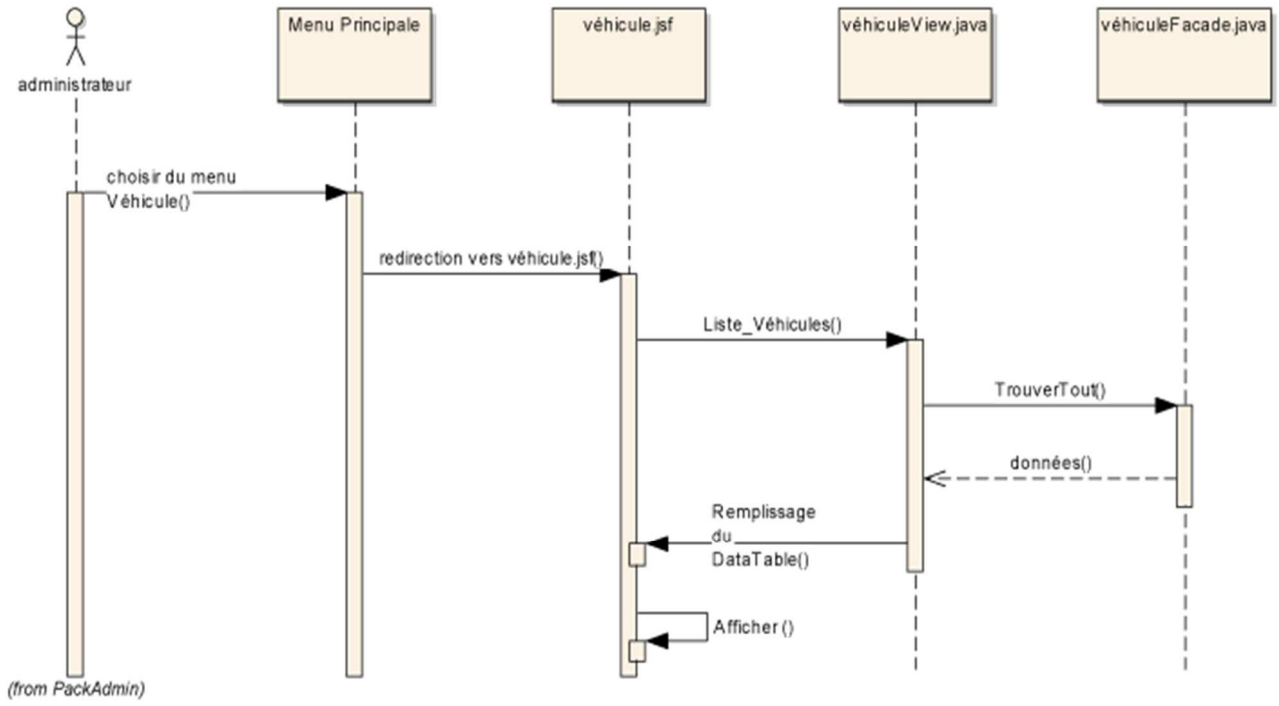

FIGURE 10 : DIAGRAMME DE SEQUENCE 'LISTE DES VEHICULES'

D'abord, l'utilisateur doit s'authentifier, en suite l'utilisateur choisi le menu véhicule Et alors la liste des véhicules sera affichée.

c)Diagramme de séquence : ajouter véhicule

Le bouton qui permet l'ajout d'un véhicule sera affiché juste à l'administrateur. Le diagramme de séquence ci-dessous illustre l'opération d'ajout d'un véhicule.

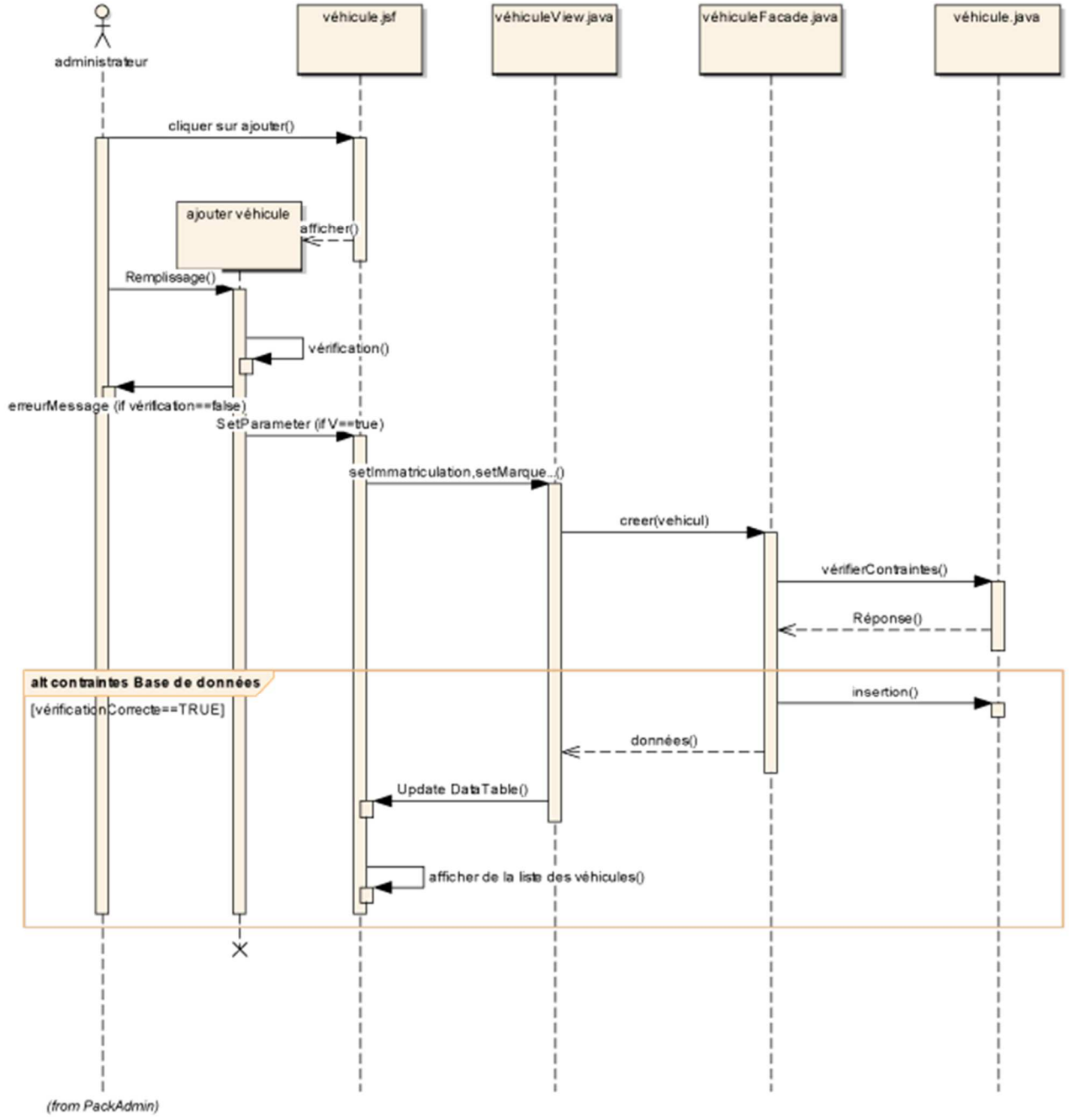

FIGURE 11 : DIAGRAMME DE SEQUENCE 'AJOUTER VEHICULE'

L'administrateur doit s'authentifier, puis il choisit le menu véhicule puis le bouton ajouter, une boite de dialogue s'affiche qui propose un formulaire à l'utilisateur , après la vérification de saisi dans la page véhcule.jsf , le système saisie dans la base de donnée la nouvelle véhicule (s'il n'existe pas déjà dans la base) .

2.4.3) DIAGRAMME DE CLASSES

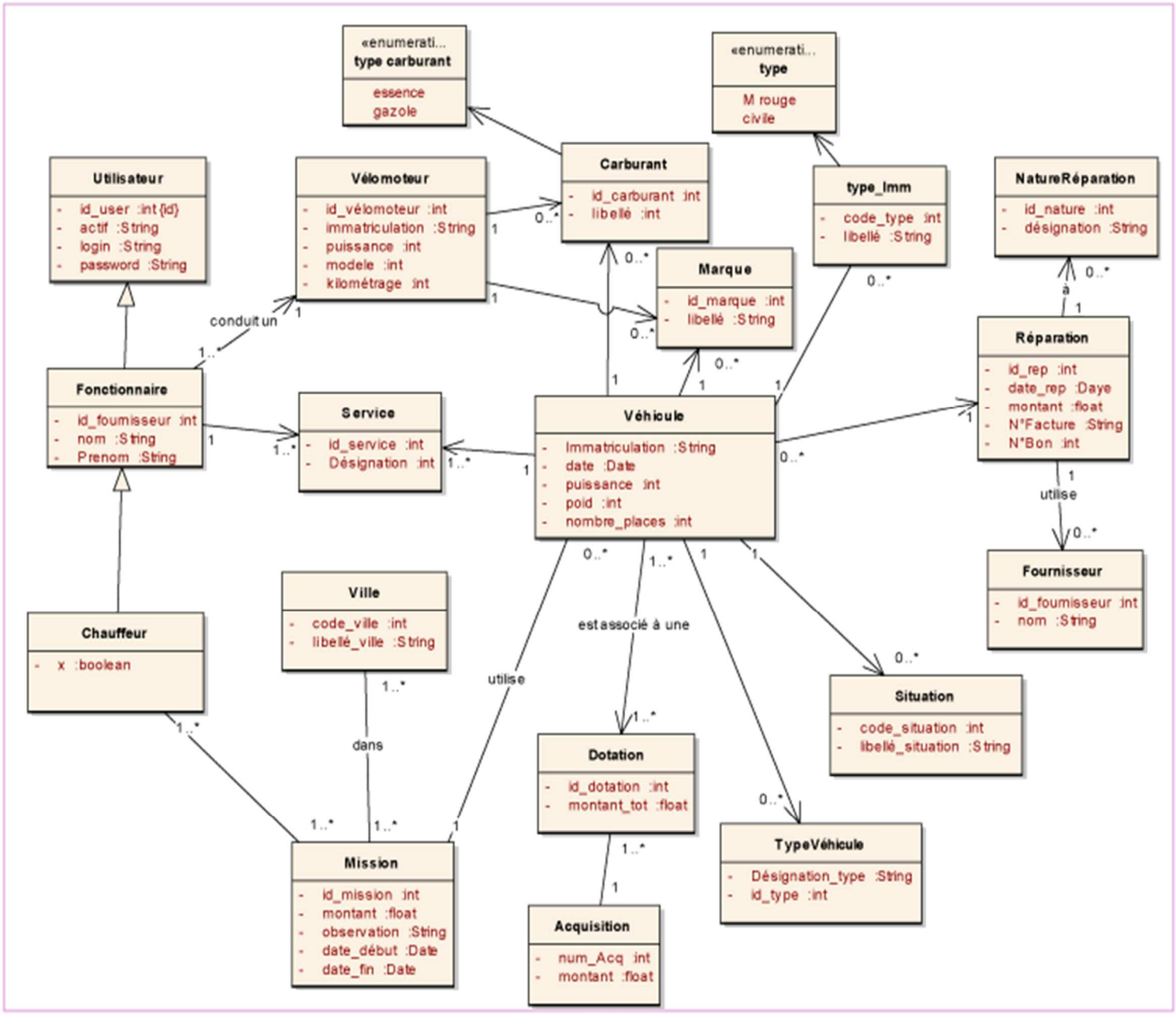

FIGURE 12 : DIAGRAMME DE CLASSES

Le diagramme de classe de la figure : 12 (ci-dessus) est constitué de plusieurs classes dans chacune représente un objet dans le système et répond à un besoin bien précis :

- Fonctionnaire : cette classe représente les informations des fonctionnaires (nom, prénom..), un fonctionnaire travail dans un service.
- Chauffeur : Cette classe est une spécialisation de la classe fonctionnaire (elle hérite de la classe fonctionnaire), un fonctionnaire peut être un chauffeur ou non.
- Utilisateur : Cette classe représente les utilisateurs de l'application, elle définit toutes les informations personnelles ainsi que les compétences qui peuvent aider pour une bonne gestion du projet. Un utilisateur est caractérisé par un login, mot de passe et un profil d'utilisateur, le profil d'utilisateur permet de filtrer les utilisateurs et alors limiter les droits d'accès. Ainsi l'utilisateur est une généralisation d'un fonctionnaire, c'est-à-dire que partir d'un utilisateur on peut garder trace à un fonctionnaire.
- Véhicule : Cette classe représente toutes les informations des véhicules du parc. Un véhicule est caractérisé par un type d'immatriculation, une marque, un type de carburant, un type, une situation et une date de mise en circulation.
- Vélomoteur : Cette classe regroupe les informations d'un vélomoteur, un vélomoteur est utilisé par un fonctionnaire, et il a un type de carburant et une marque .En plus il se caractérise par un numéro d'immatriculation, puissance, kilométrage et le modèle.
- Réparation : Cette classe représente les réparations des véhicules, elle a une nature réparation et un fournisseur, ce dernier est caractérisé par une classe qui représente les fournisseurs qui ont un accord avec le service.
- Mission : cette classe représente les missions, elle présente les informations nécessaires pour le suivi de chaque mission.
- Dotation : Cette classe représente les dotations de carburant, chaque dotation peut avoir une ou plusieurs acquisitions.

CHAPITRE 3 : MISE EN ŒUVRE DU PROJET

Après avoir définir les besoins fonctionnels et techniques du projet ainsi que les outils de développement utilisés, dans ce chapitre on va présenter le fonctionnement global de l'application et les différents interfaces du projet.

3.1) ARCHITECTURE DE L'APPLICATION

Notre application se base sur une architecture java EE qui permet l'utilisation du jpa (java Persistance api) et les jsf. La figure :13 illustre les différents couches de l'architecture de l'application :

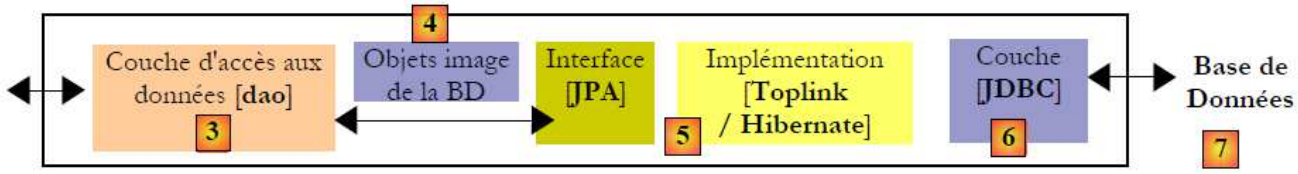

FIGURE 13 : ARCHITECTURE DE L'APPLICATION

En pratique, on a essayé de caractérisé chaque couches par un package qui regroupe Plusieurs éléments :

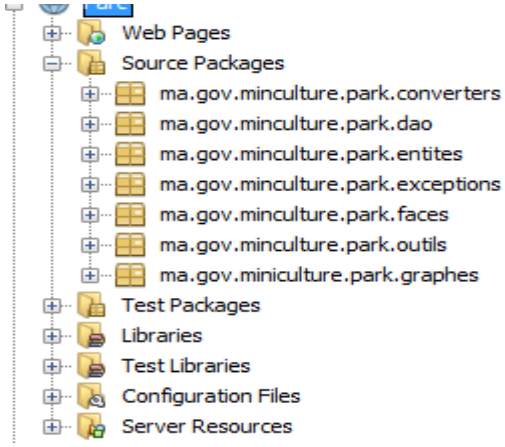

FIGURE 14 : STRUCTURE DU PROJET

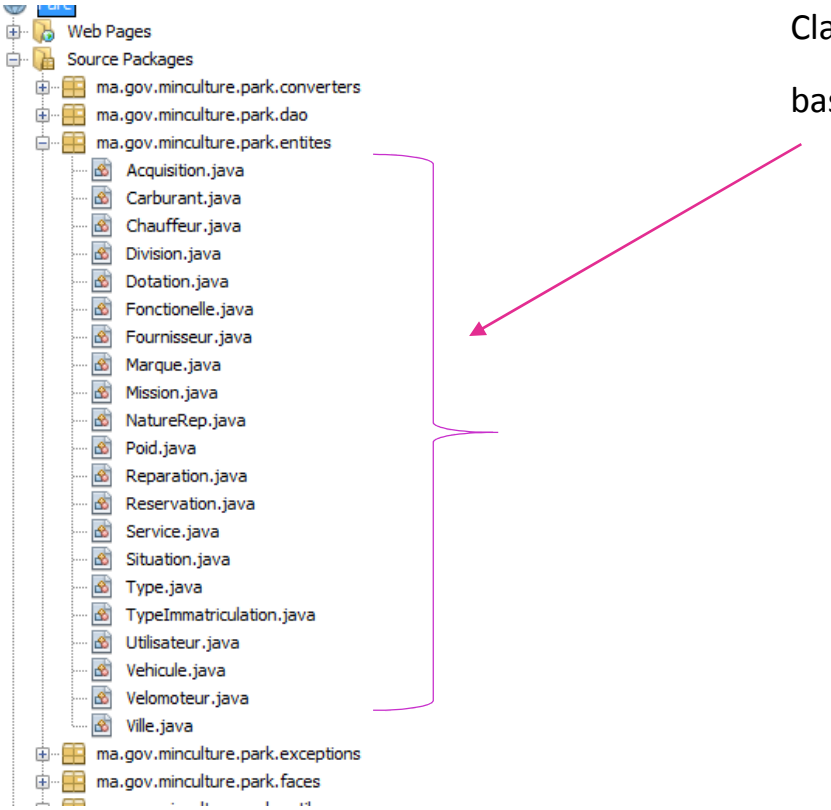

FIGURE 15 : CLASSE GENERER A PARTIR DE LA BASE DE DONNEES

3.2) DESCIPTION DES IHM

Cette partie a pour objectif de donner une présentation visuelle du système réalisé. Ainsi, quelques écrans du système seront présentés.

a)Authentification

Cette interface oblige l'utilisateur de s'authentifier avant d'utiliser l'application, ainsi que l'utilisateur sera redirigé automatiquement à cette page après la fin de la session.

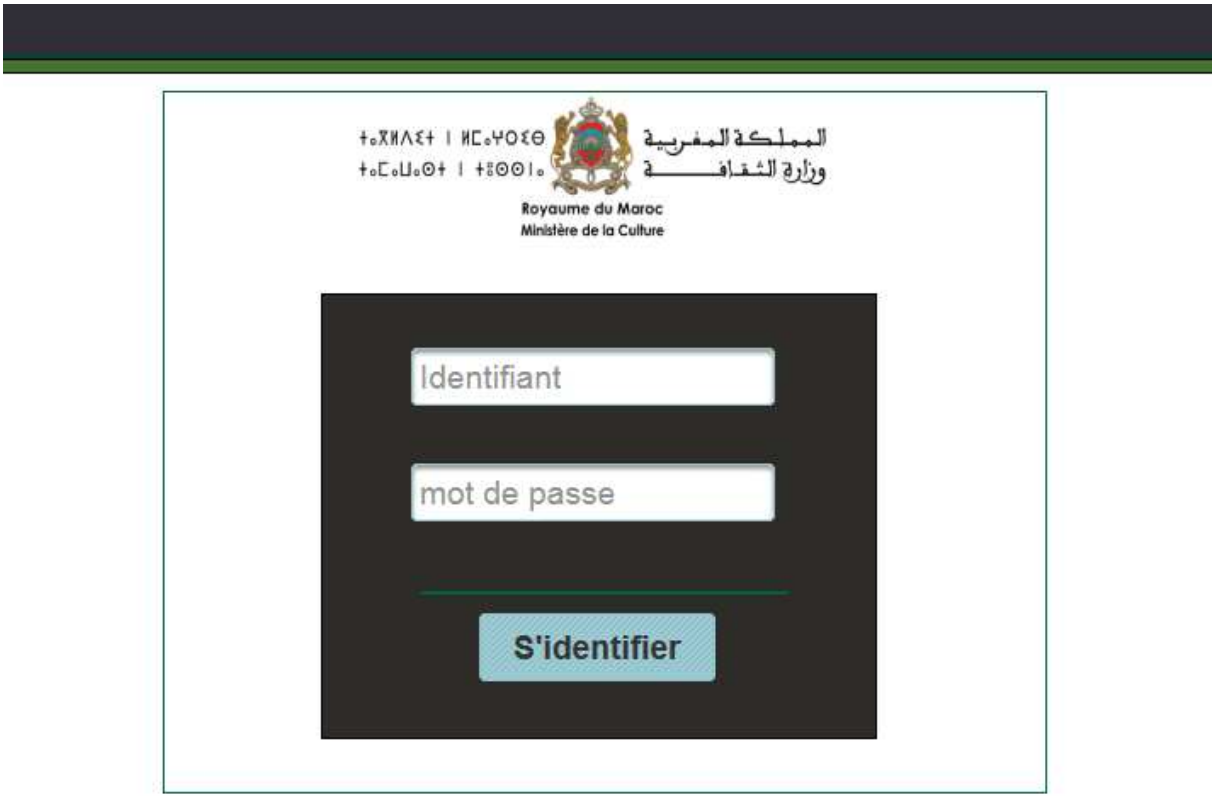

FIGURE 16 : IHM AUTHENTIFICATION

Contrôle de saisie :

Si l'utilisateur n'entre pas l'identifiant ou le mot de passe, le message suivant s'affiche :

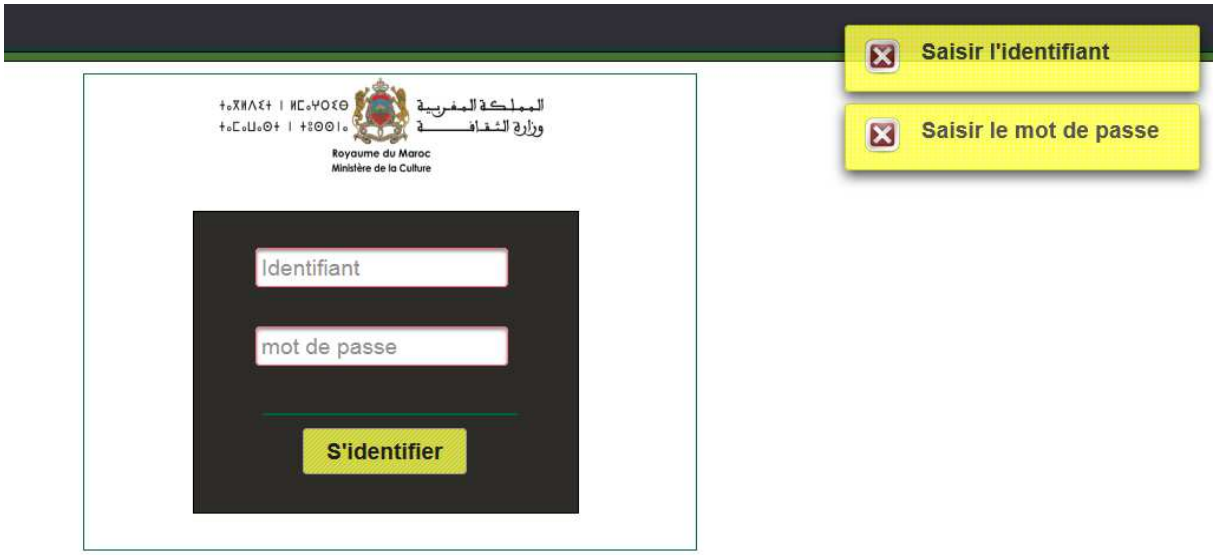

FIGURE 16 : IHM AUTHENTIFICATION ERREUR SAISIE

- Si la saisie est correcte mais les identifiants sont incorrecte :

Un message d'erreur s'affiche :

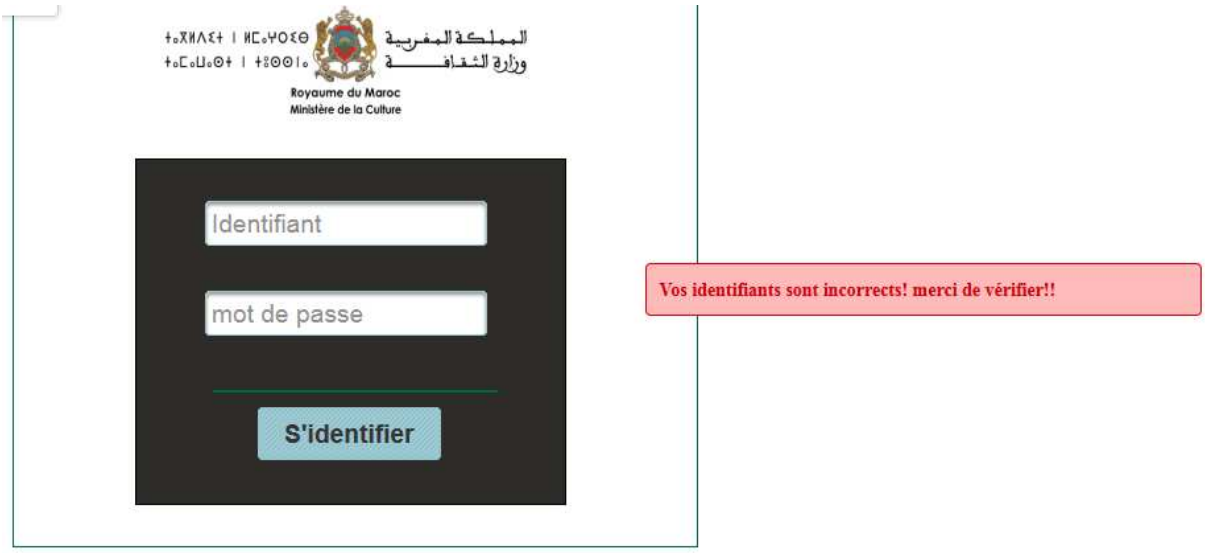

FIGURE 17 : IHM AUTHENTIFICATION IDENTIFIANTS INCORRECTE

- Si le saisie est correcte mais le compte d'utilisateur est désactivé :

Le message : '' **Votre compte est désactivé. Merci de contactez L'administrateur!!** ''S'affiche.

b) l'IHM de la page véhicule.jsf

Si l'identifiant et le mot de passe sont corrects, passage au mode de session correspondant

au type d'utilisateur et affichage de la page d'accueil (vehicule.jsf).

Cette interface représente le style général de l'application (c'est-à-dire le reste des

Interfaces suivent le même design).

Ajout d'un véhicule

Ou Impression de la liste des véhicules

Le login de l'utilisateur

ordonné par

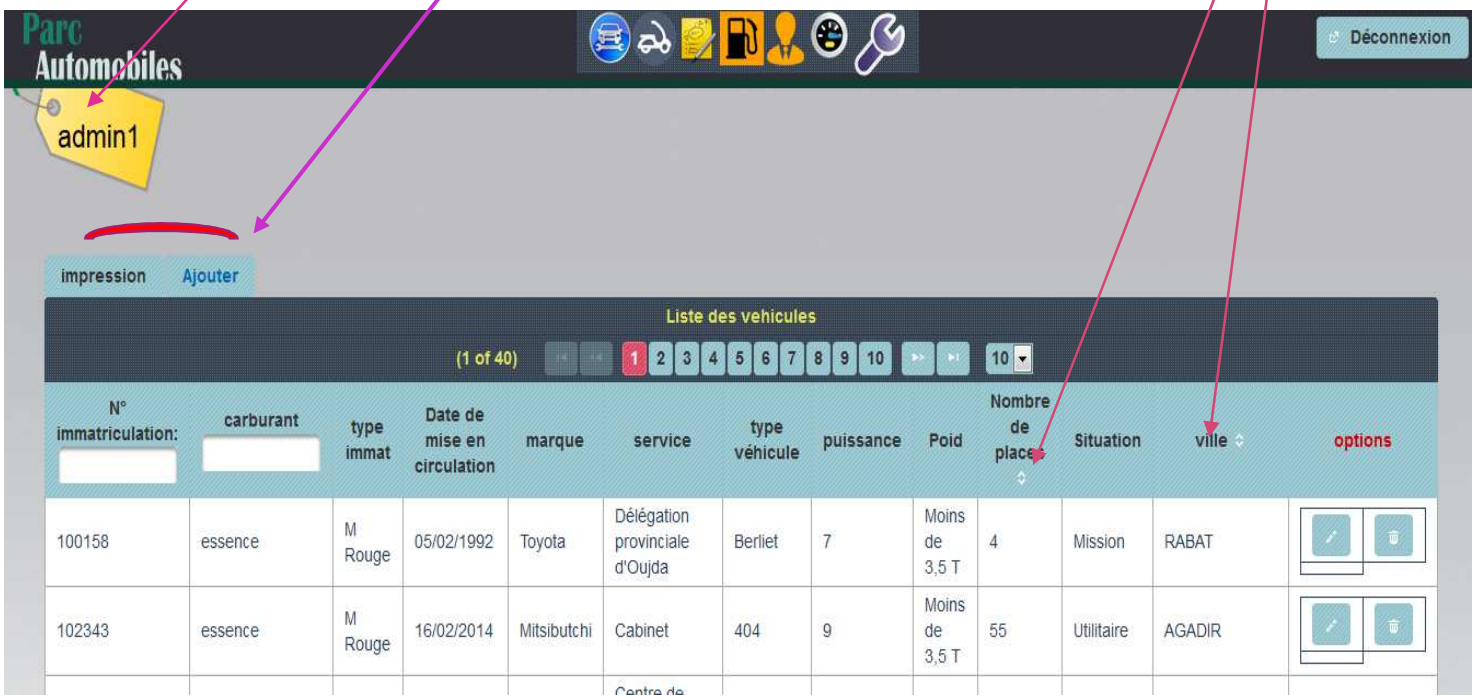

FIGURE 18 : IHM GESTION VEHICULE

c) le menu principale :

FIGURE 19 : IHM MENU PRINCIPALE

d) L'IHM d'ajout d'un vélomoteur :

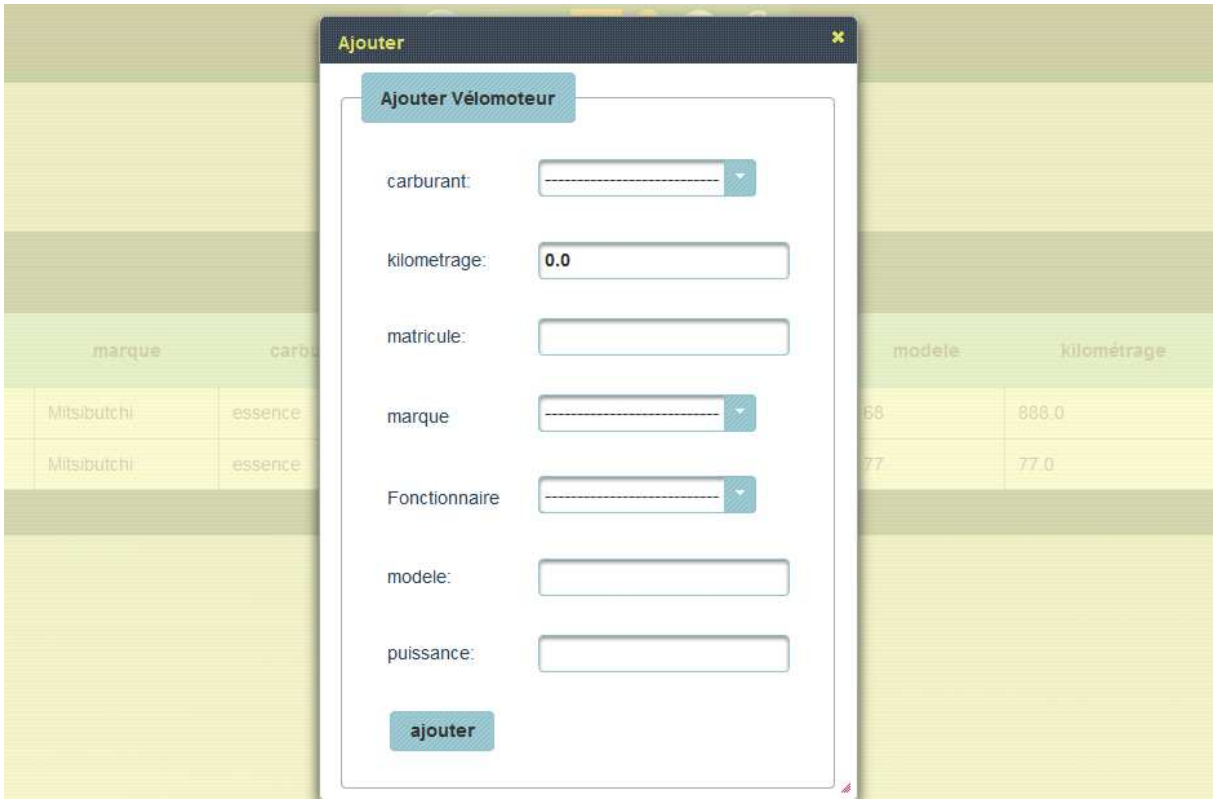

FIGURE 20 : IHM AJOUT VEHICULE

e) L'IHM supprimer un véhicule :

FIGURE 21 : IHM SUPPRIMER VEHICULE

Après la suppression on donne un feed-back immédiat à l'utilisateur en lui informant que le véhicule est bien supprimé.

f) L'IHM modifier un vélomoteur :

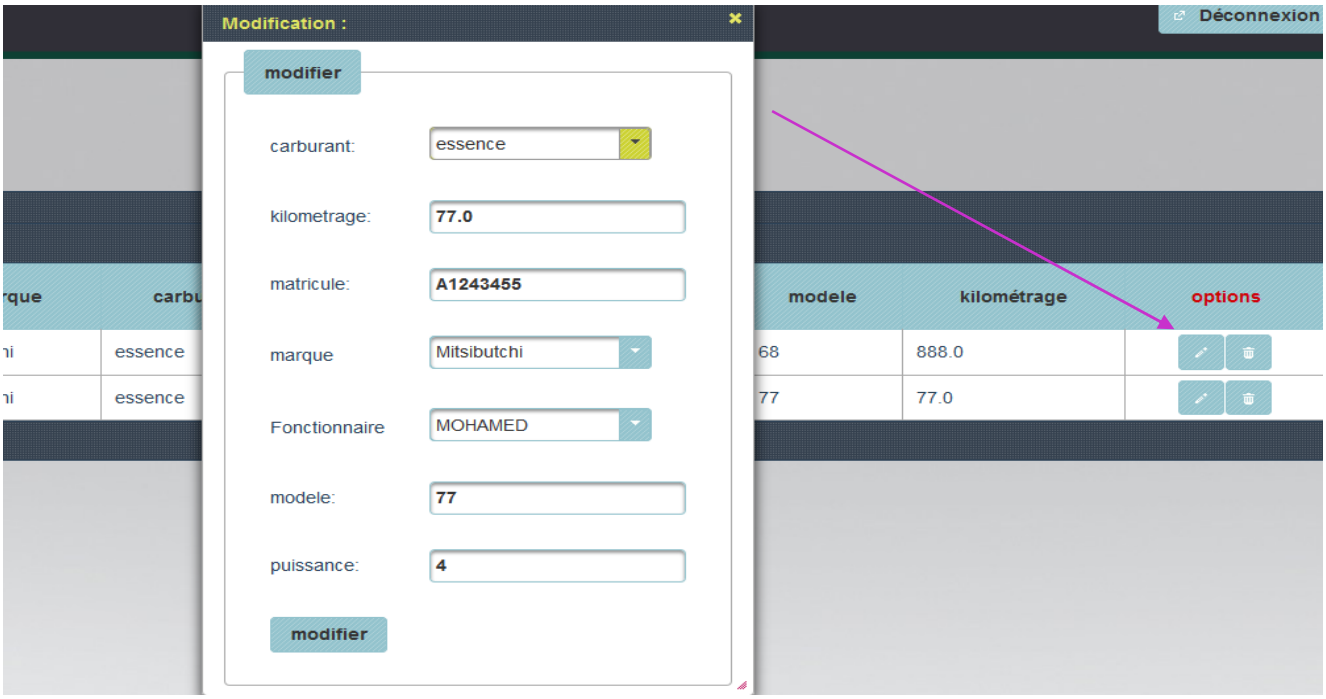

FIGURE 22 : IHM MODIFIER VELOMOTEUR

g) L'IHM de gestion des dotations de carburant :

Cette interface permet à l'administrateur de gérer les dotations de carburant, et elle lui donne une vision globale sur la différence entre le montant-totale et el montant restant, ce dernier est calculé à partir des acquisitions que l'administrateur peut les suivre en cliquant sur un bouton.

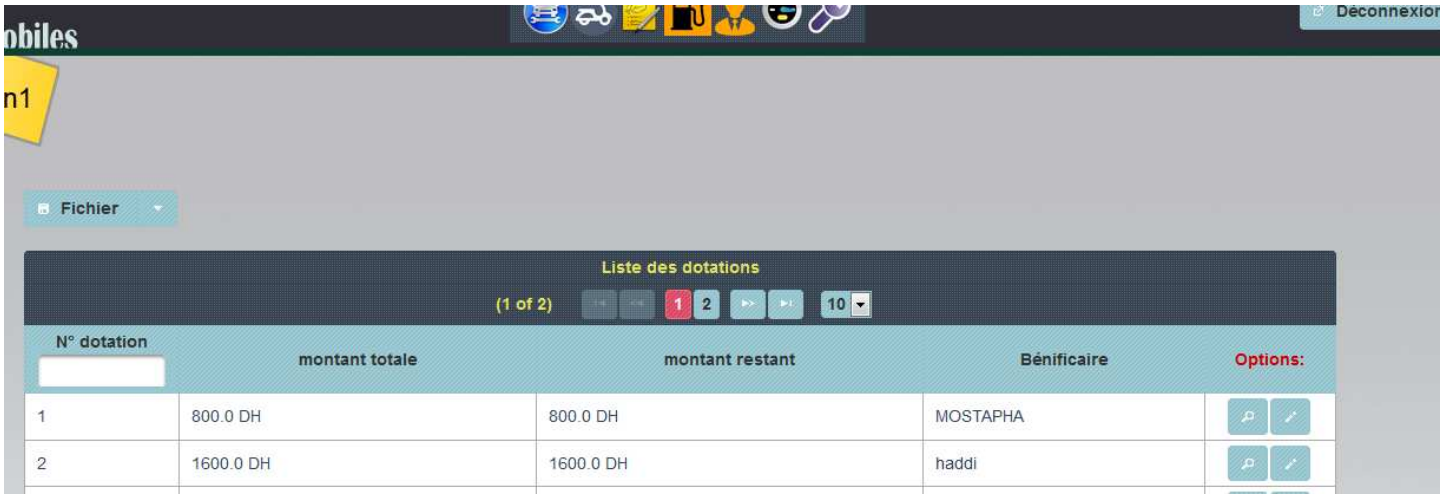

FIGURE 23 : IHM DOTATION DU CARBURANT

h) L'IHM des acquisitions:

Cette IHM représente les acquisitions d'une dotation donné.

FIGURE 24 : ACQUISITIONS D'UNE DOTATION

i) L'IHM gestion des utilisateurs:

Cette IHM permet à l'administrateur de gérer les utilisateurs.

FIGURE 25 : GESTION DES UTILISATEURS

CONCLUSION GENERALE

Notre projet de fin d'étude effectué au sein de ministère de la culture de Rabat était très bénéfique, car il m'a permis de se familiariser avec un nouveau environnement de développement que l'on pas vu pendant notre formation, et il m'a permis aussi d'exploiter ce que j'ai acquis durant mon cursus de formation. En plus, ce stage était une opportunité pour travailler dans un milieu professionnel et découvrir le monde du travail au sein de la division du système d'information qui exige une responsabilité, une assiduité et un esprit d'équipe.

L'informatisation du parc automobile va permettre donc une meilleure gestion des données, et elle va résoudre plusieurs problèmes de la gestion manuelle qui rend la circulation des informations très lente (la gestion des missions, des véhicules, des dotations...).

Jusqu'à présent notre application assure la gestion des véhicules, vélomoteurs, missions, réparations et des utilisateurs. Ce travail peut être considéré comme une première étape pour une gestion complète d'applications. En outre, notre application a besoin de gérer d'autres besoins comme les notifications pour les vidanges et les assurances, et elle doit permettre aux fonctionnaire d'envoyer des demandes de réservations des véhicules, ainsi qu'on doit ajouter dans la page d'accueil un emploi du temps éditable qui va permettre à l'administrateur du parc de planifier ces taches et cette page d'accueil doit contenir aussi des graphes représentants les statistiques …

En somme, dans ce rapport qui rassemble l'étude de l'existant et le cahier des charges, nous avons défini le futur système d'information et ses différentes contraintes. J'aimerai bien que ce travail sera apprécié par les responsables et en suite connait son achèvement.

WEBOGRAPHIE :

[1] :http://fr.wikipedia.org/wiki/

[2] : http://www.mkyong.com/tutorials/spring-security-tutorials/

[3] : http://baptiste-wicht.developpez.com/tutoriels/conception/mvc/

[4] : http://primefaces-fr.blogspot.com/2013/01/primefaces-une-introduction.html

[5]:http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[6] :http://javaweb.developpez.com/faq/javaee/?page=DEFINITIONS#DEFINITION_javaee

https://glassfish.java.net/documentation.html

http://www.mkyong.com/tutorial

BIBLIOGRAPHIE :

[7] : UML 2 par la pratique : auteur **<<Pascale Roques>>**

[8] : EJB 3 : auteur **<<Alexis Moussine-Pouchkine >>** *(disponible à la bibleothéque de FSTF)*

[9] : Développons en Java :auteur **<<J.M.Doudoux>>**

[10]:Pearson Education France-Java EE 6 et GlassFish3:auteur**<<Antonio Goncalves>>**

Développement n-tiers avec JAVA auteur **<<JEROME LAFROSSE>>**

Primefaces_users_guide_4_0