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

N° Réf :……

Centre Universitaire
Abd elhafid boussouf Mila

Institut des sciences et de la technologie Département de Mathématiques et Informatique

Mémoire préparé En vue de l'obtention du diplôme de Master en :
Filière informatique

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

Thème
Développement d'une application web intégrant une base de données répartie en utilisant la plateforme Java EE

Préparé par : Amireche Iman
Amireche Amina

Soutenue devant le jury :

Président	**:**	**M^eur Benchikhelhocin Madjed**	**(MA)**
Examinateur	**:**	**M^eur MERABET Adil**	**(MA)**
Encadreur	**:**	**M^eur Boukhechem Nadhir**	

Année universitaire : 2014/2015

Remerciements

Nous tenons tout d'abord à remercier le bon Dieu qui nous a donné la santé et le courage d'accomplir ce travail.

Nous ne pouvons pas oublier de présenter notre gratitude à nos parents Amireche Mohammed et Boguelout Nadia pour leur patience et les efforts inlassables qu'ils ne cessent de déployer pour nous.

Nous remercions aussi notre frère Oussama, nos sueurs Manel et Roumaissa, et nos amis qui nous ont encouragé, soutenu durant tout notre cursus.

Nos vifs remerciements vont à M^eur Boukhechem Nadhir , notre encadreur pour tousses conseils très précieux, ses encouragements, et ses orientations qui nous ont été très bénéfiques tout au long de ce travail.

Nous remercions également les membres de jury qui Nous font honneur en acceptant d'examiner et de juger notre travail.

Enfin, que tous ceux qui ont contribués de prés ou de loin à l'aboutissement de ce travail trouvent ici l'expression de notre sincère gratitude et nos remerciements les plus sincères.

Amireche Iman & Amina

Dédicace

Nous dédions ce travail en tout premiers lieu à notre dieu ALLAH le tout puissant et miséricordieux qui nous a donné la force, la volonté et le courage pour accomplir ce modeste projet.

À nos très chers parents, qui ont le droit de recevoir nos chaleureux remerciements pour le courage et le sacrifice qu'ils ont consentis pendant la durée de nos études, Nous espérons qu'ils trouveront dans ce travail toute l'expression de notre reconnaissance.

À nos frères et sœur.

À toute la famille.

À notre encadreur.

À tous nos chers amis et nos collègues de l'Université, et à tous ce qui nous ont enseigné tout au long de notre parcoure.

Amireche Iman & Amina

Résumé

Une base de données répartie est une collection de données logiquement reliées et physiquement réparties sur plusieurs machines interconnectées par un réseau de communication. L'objectif de ce travail est d'utiliser les technologies actuelles (Java EE) pour réaliser une application de commercialisation et vente de voitures qui profitent des avantages de la répartition des bases de données et plus précisément l'augmentation des performances et tolérance aux pannes.

Ce mémoire comprend quatre chapitres ; les deux premiers chapitres sont purement théoriques comprenant une présentation des applications web, et des bases de données réparties. Les deux derniers chapitres sont consacrées à la conception et la mise en œuvre de la base de données qui est répartie entre une agence centrale et un ensemble de représentants locaux distants, et à la réalisation de l'application Java EE exploitant cette base de données.

Mots clés : Applications web,Bases de données réparties,Java EE, Jboss,RMI,EJB,Vente de voiture.

Abstract

A distributed database is a collection of logically related data and physically distributed across several machines interconnected by a communication network. The objective of this work is to use current technologies (Java EE) for creating an application of marketing and sale of cars that enjoy the benefits of the division of databases and more specifically increased performance and fault tolerance .

This memory consists of four chapters ; The first two chapters are theoretical presentation including web applications and distributed databases . The last two chapters are devoted to the design and implementation of the database which is divided between a central agency and a set of remote local representatives, and the realization of the Java EE application exploiting this database.

ملخص

قواعد البيانات الموزعة هي عبارة عن مجموعة من البيانات المتعلقة منطقيا والموزعة جغرافيا عبر عدة آلات مترابطة عن طريق شبكة الاتصالات. و الهدف من هذا العمل هو استخدام التكنولوجيا الحالية مثل (Java EE) لإنشاء تطبيق لتوزيع وبيع السيارات ,هذه التكنولوجيا تسمح بتحسين الأداء و تجنب تعطل النظام.

تتكون هذه المذكرة من أربعة فصول ؛الفصلين الأولين يتعلقان بالعرض النظري بما في ذلك التطبيقات على شبكة الإنترنت و قواعد البيانات الموزعة. أما الفصلين الأخيرين فهما مكرسان لتصميم و تنفيذ قاعدة البيانات التي تنقسم بين وكالة مركزية و مجموعة من الممثلين المحليين عن بعد، وتحقيق تطبيق Java EE باستغلال قاعدة البيانات هذه.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Le monde de l'informatique évolue très rapidement, alors que son but initial, était d'offrir des services satisfaisants, du point de vue vitesse d'exécution des tâches . Actuellement, de nouveaux besoins sont apparus notamment pour les grandes entreprises, celles-ci souhaitent stocker et échanger des informations qui sont géographiquement éloignées. La collecte et le traitement d'une grande quantité d'informations dispersées est une tâche très délicate, La solution qui s'impose est de distribuer les données et les organiser dans des bases de données sur différents sites de stockage. L'ensemble de ces sites constitue un système de bases de données réparties offrant la possibilité aux utilisateurs de manipuler les différentes bases via un réseau de manière transparente, comme dans une base de données unique.

Notre projet consiste à développer en utilisant la plate-forme Java EE une application web de commercialisation et suivie des voitures. Les données de l'application sont réparties entre un site central (entreprise mère) et un ensemble de représentant locaux distants. L'application doit donc gérer ces données reparties de la manière la plus efficace possible et éviter l'arrêt ou le blocage du système et par conséquent éviter l'arrêt des ventes.

Notre mémoire est structuré en 4 chapitres comme suit :

- **Dans le premier chapitre**, nous présentons un état de l'art sur les applications web, les technologies utilisées dans ce genre des applications et plus précisément la technologie JAVA EE.

- **Le second chapitre**, aborde les différentes techniques de conception et de gestion des bases de données réparties ainsi que les principes de la réplication.

- **Le troisième chapitre**, présente les différentes étapes d'analyse et de conception du système en utilisant le langage de modélisation UML.

- **Le dernier chapitre**, Présente les outils de développement utilisé et expose quelques interfaces du système.

Chapitre 1

Les application web

- Introduction
- L'histoire des applications web.
- Définition d'une application web.
- Les avantages des applications web.
- Les technologies utilisées dans les applications web.
- La technologie JAVA EE.
- Conclusion.

Introduction

L'apparition d'Internet « Interconnected Networks » a apporté de profonds changements dans la société, elle a offertes de nouvelles opportunités aux entreprises inexistantes auparavant. Internet est un vaste réseau de communication reliant des millions d'ordinateurs entre eux, ces ordinateurs ont la possibilité d'échanger des informations à travers le monde .

Le « World Wide Web » habituellement appelé le Web est un système hypertexte public fonctionnant sur Internet et qui permet de consulter, avec un navigateur, des pages mises en ligne dans des sites. Le Web est une des applications d'Internet, comme le courrier électronique, et la messagerie instantanée.

l'Internet est donc le support physique de l'information (machines, câbles et éléments réseau) tandis que le web constitue une partie seulement du contenu accessible sur Internet.

Le développement d'applications liées au Web a connu ces dernières années une très forte croissance. Les applications Web ont été finalement intégrées dans des domaines de plus en plus diversifiés (commerce électronique, éducation, gestion des entreprises, etc). Elles présentent plusieurs caractéristiques propres tout en étant de taille et de complexité de plus en plus élevées.

Dans ce chapitre nous allons présenter les applications web, ainsi que les technologies utilisées pour leurs développements et plus précisément la technologie java EE.

1.1 L'histoire des applications Web

Le concept d'application Web n'est pas récent. En effet, un des premiers langages de programmation pour le développement d'applications Web était le « Perl ». Il fut inventé par Larry Wall en 1987 avant qu'Internet ne devienne accessible au grand public. Mais c'est en 1995, lorsque le programmeur Rasmus Lerdorf rendit le langage PHP disponible à tous que le développement d'applications Web prit vraiment son démarrage.

Quelques mois plus tard, Netscape, ancien et populaire navigateur Web, annonçait une nouvelle technologie, le JavaScript, permettant aux programmeurs de modifier dynamiquement le contenu d'une page Web. Cette technologie permit une toute nouvelle approche dans le développement des applications Web.

L'année suivante, en 1996, deux développeurs, Sabeer Bhatia et Jack Smith, lançaient Hotmail, un service de messagerie en ligne qui permettait (pour la toute première fois) au grand public, d'accéder et de consulter ses courriels partout où les utilisateurs pouvaient se trouver sur la planète et donc loin de leur poste de travail (ordinateur).

En 1998, la compagnie Google mit en ligne son premier moteur de recherche qui, par sa nouvelle façon d'indexer les pages Web, facilita grandement la recherche d'informations sur Internet.

Au début de l'année 2001, Wikipédia lancé une encyclopédie traditionnelle en ligne. Pour développer sa plateforme, Wikipédia utilisa un type d'application Web nommée « wiki » permettant à tout internaute d'ajouter du contenu.

En 2005, You Tube fut officiellement lancé, permettant aux internautes de partager des vidéos en ligne.

Twitter, quant à lui, fut lancé en 2006. Avec les années, la popularité de Twitter n'a fait qu'augmenter, passant de 1,6 million de « tweets » en 2007 à l'impressionnant chiffre de 340 millions par jour en mars 2012. [1]

1.2 Définition d'une application web

Une application web (aussi appelée web application, en anglais) désigne un logiciel applicatif hébergé sur un serveur et accessible via un navigateur web (Google Chrome, Mozilla, FireFox, ...), A l'intermédiaire d'un réseau informatique (Internet, intranet, etc.).

Contrairement à une application de bureau qui nécessite l'installation d'un logiciel dans chaque environnement dans lequel il est utilisé. L'utilisateur d'une application web n'a pas besoin de l'installer sur son ordinateur. Il lui suffit de se connecter à l'application à l'aide de son navigateur favori.

1.3 Les avantages des applications web

Nous pouvons résumer les principaux avantages d'une application web de la manière suivante :

- Il n'est plus nécessaire d'installer un logiciel sur chaque poste, ce qui diminue en grande partie les frais de maintenance et élimine certaines incompatibilités .
- Une application bien conçue peut être utilisée par différents types de terminaux (ordinateurs, laptops, tablettes, smartphones) .
- Des centaines de personnes réparties dans des lieux différents travaillent simultanément sur ce type d'applications car elles sont conçues pour supporter une grande charge qui est souvent répartie sur plusieurs serveurs .
- Ces applications sont capables de gérer des données multimédia (textes, photos, vidéos).
- Pour mettre à jour l'application, il suffit de modifier l'application sur le serveur et tous les postes. accèdent instantanément à la nouvelle version.

1.4 Les technologies utilisées dans le développement des applications web

Dans la technologie client-serveur, utilisée pour le World Wide Web, le navigateur Web (client web) envoie au serveur des requêtes relatives à des pages Web. Le serveur répond aux demandes en envoyant les pages au navigateur Web. Le navigateur affiche alors les pages à l'utilisateur.

Les applications Web utilisent cette technique pour mettre en œuvre leur interface graphique. Celle-ci est composée de pages créées de toutes pièces par le logiciel lors de chaque requête. Chaque hyperlien contenu dans la page provoque l'envoi d'une nouvelle requête.

Le protocole HTTP permet d'assurer la communication entre le navigateur et le serveur. Et les langages : HTML, CSS et JavaScript sont utilisés pour mettre en forme les données et les afficher à l'utilisateur, Ils sont tous interprétés par le navigateur, directement sur la machine client.

A la différence du couple HTML et CSS qui est un standard incontournable pour la mise en forme des pages web, il existe plusieurs technologies capables de traiter les informations sur le serveur. Java EE est l'une d'entre elles, mais il en existe d'autres : PHP, .NET... etc.

1.4.1 HTML

Le HTML (ou Hyper Text Markup Language) est un langage universel et hypertexte à balise (ou marqueur) utilisé pour communiquer sur le Web. Cela veut dire que l'on va gérer la façon dont un texte va s'afficher au sein du navigateur.

Le langage HTML n'est pas un langage de programmation au sens strict du terme, c'est un langage de description de page web. Il décrit la façon de mettre en forme le contenu de la page au sein d'un navigateur web.

Ce langage a été standardisé et cela le rende portable sur l'ensemble des systèmes le prenant en compte. [2]

+ Avantages du HTML

Le langage HTML permet au créateur de pages web divers possibilités :

- mise en page par la présence de : Séparateur, Diviseur et Paragraphes.
- De mise en forme par l'existence de : Titres, Caractères en gras, en italique, souligné.
- D'afficher des images et de les rendre interactives : Images statique, cliquables et liens.
- De présentation sous forme de listes de tableaux : Tableaux pour la présentation de résultats et différents type de listes.
- D'interaction avec les visiteurs par le biais de formulaires : Les différents types de champs utiles dans un cadre. Ajout d'interaction avec l'utilisation par le biais de champ de saisie.

1.4.2 CSS

Le CSS « Cascading Style Sheets » ou les feuilles de styles en français est un langage qui permet de gérer la présentation d'une page Web, il permet de définir des règles appliquées à un ou plusieurs documents HTML. Ces règles portent sur le positionnement des éléments, l'alignement, les polices de caractères, les couleurs , les marges et espacements, les bordures, les images de fond, etc.

Le but de CSS est séparer la structure d'un document HTML et sa présentation. En effet, avec HTML, on peut définir à la fois la structure (le contenu et la hiérarchie entre les différentes parties d'un document) et la présentation. Mais cela pose quelques problèmes. Avec le couple HTML/CSS, on peut créer des pages web où la structure du document se trouve dans le fichier HTML tandis que la présentation se situe dans un fichier CSS.[3]

+ Avantages de CSS :

CSS permet de :

- De définir un ensemble de règles stylistiques communes à toutes les pages d'un site Internet. Cela facilite ainsi la modification de la présentation d'un site entier.
- De définir aussi des règles différentes pour chaque support d'affichage.
- De séparer la structure et la présentation et ainsi fournir d'énormes bénéfices en termes d'accessibilité.

1.4.3 Javascript

Le JavaScript est le langage client de scripts inventé en 1995 pour le navigateur Netscape, et qui permet d'agir sur la page de manière dynamique. Contrairement aux langages serveurs telles que PHP, qui s'appliquent à générer un code différent selon la requête reçue, JavaScript ne fonctionne que sur le navigateur, et sert à modifier le code reçu. Ce langage permet d'effectuer des contrôles au moment de la saisie.

+ Avantages de JavaScripts

Les avantages du JavaScript sont nombreux :

- Vitesse : Les fonctions du JavaScript ne doivent pas attendre pour des réponses de leurs serveurs pour agir, ce qui accélère l'ouverture des sites web.
- Simplicité : le JavaScript est relativement simple et facile à apprendre.

1.4.4 PHP

PHP est un langage de script HTML exécuté du côté du serveur. Il veut dire « PHP : Hypertext Preprocessor ». Sa syntaxe est largement inspirée du langage C, de Java et de Perl, avec des améliorations spécifiques. Le but du langage est d'écrire rapidement des pages HTML dynamiques.

PHP est un langage côté serveur. Lors du chargement d'une page PHP, c'est le serveur qui va lire, interpréter et exécuter le code. Puis il renvoie le résultat, généralement sous la forme de code HTML au navigateur. Ainsi le navigateur et l'utilisateur ne voient jamais le véritable code PHP exécuté. De plus le résultat étant une page web classique en HTML, pas besoin d'installer sur le client des composants spécifiques.

1.4.5 .NET

Le .NET Framework est un environnement d'exécution qui fournit divers services à ses applications actives. Il comporte deux composants principaux : le Common Langage Runtime (CLR), qui est le moteur d'exécution gérant les applications actives, et la bibliothèque de classes .NET Framework, qui fournit une bibliothèque de code testé réutilisable que les développeurs peuvent appeler à partir de leurs propres applications.

DotNET fournit un ensemble de langages et composants nécessaires pour la réalisation des applications Web .

1.4.6 Java EE

Le terme « Java » fait bien évidemment référence à un langage, mais également à une plate-forme : son nom complet est « Java SE» pour Java Standard Edition, et était anciennement raccourci « J2SE ». Celle-ci est constituée de nombreuses bibliothèques, qui contiennent un nombre conséquent de classes et de méthodes prêtes à l'emploi pour effectuer toutes sortes de tâches.

Tandis que le terme «Java EE» signifie Java Enterprise Edition depuis 2006, et était anciennement raccourci en « J2EE ». Un ensemble de spécifications proposées par la société Sun (Oracle) et portées par un consortium de sociétés internationales dédiées au développement, au déploiement et à la gestion d'applications à base de composants centrées sur le serveur . Il fait quant à lui référence à une extension de la plate-forme standard. Autrement dit, la plate-forme Java EE est construite sur le langage Java et la plate-forme Java SE, et elle y ajoute un grand nombre de bibliothèques remplissant tout un tas de fonctionnalités que la plate-forme standard ne remplit pas d'origine. L'objectif majeur de Java EE est de faciliter le développement d'applications web robustes et distribuées, déployées et exécutées sur un serveur d'applications.

1.5 La technologie Java EE

1.5.1 L'Architecture Java EE

Deux types d'architectures sont utilisées dans JEE, java EE web profile et java EE full profile.

» L'architecture java EE web profile

Dans l'architecture java EE web profile le serveur web (comme tomcat) n'inclut qu'un seul conteneur qui est le conteneur web, comme indiqué dans la figure suivante :

FIGURE 1.1 – L'Architecture java EE WEB profile

»L'architecture java EE full profile

Dans cette architecture le serveur d'application (jboss par exemple) inclut deux type de conteneurs (conteneur WEB et conteneur EJB), comme représente la figure suivante :

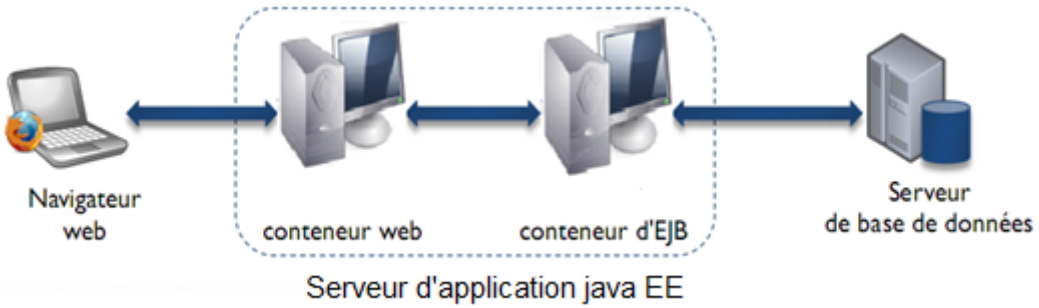

FIGURE 1.2 – L'Architecture java EE full profile

1.5.2 Le serveur d'application java EE (architecture java EE full profile)

La figure suivante permet d'inclure les différentes composantes d'un serveur d'application java EE :

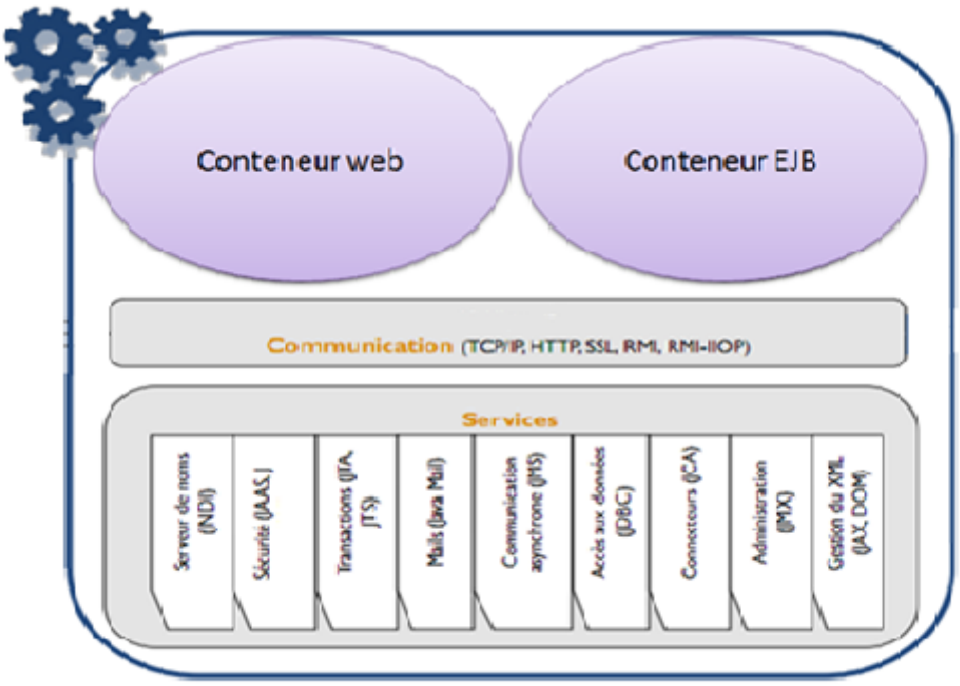

FIGURE 1.3 – Le serveur d'application java EE

A/ Partie web du serveur Java EE (Conteneur web)

Le développement en entreprise implique :

- Que l'on puisse être amené à travailler à plusieurs contributeurs sur un même projet ou une même application (travail en équipe).
- Que l'on puisse être amené à maintenir et corriger une application que l'on n'a pas créée soi-même.
- Que l'on puisse être amené à faire évoluer une application que l'on n'a pas créée soi-même.

Pour toutes ces raisons, il est nécessaire d'adopter une architecture plus ou moins standard, que tout développeur peut reconnaître, c'est-à-dire dans laquelle tous développeur sait se repérer.

Un conteneur web avec un modèle MVC dont le principe est de séparer les responsabilités permet de répondre à ces besoins.

»Le modèle MVC

Le modèle MVC (Modèle-Vue-Contrôleur) découpe littéralement l'application en couches distinctes, et de ce fait impacte très fortement l'organisation du code comme indiqué dans la figure suivante :

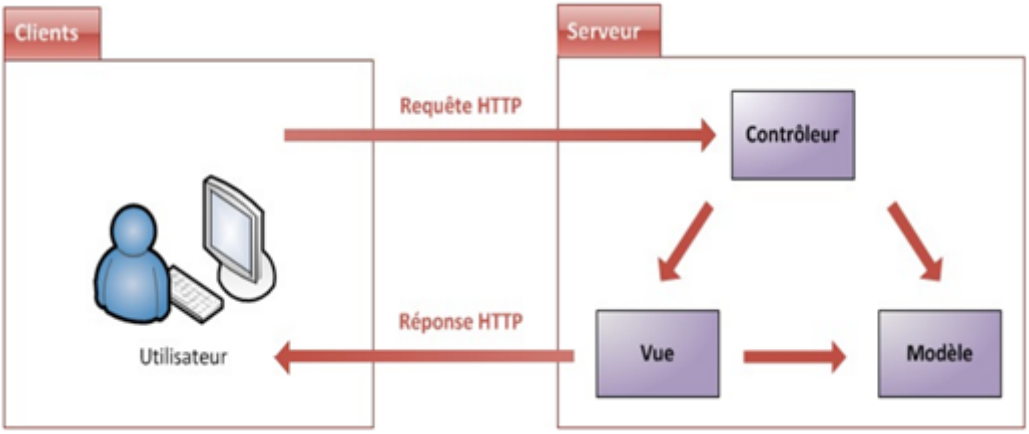

FIGURE 1.4 – Le Modèle MVC

•Le modèle

Contient à la fois les données et les traitements à appliquer à ces données. Ce bloc contient donc des objets Java d'une part, qui peuvent contenir des attributs (données) et des méthodes (traitements) qui leur sont propres, et un système capable de stocker des données d'autre part. la complexité du code dépendra bien évidemment de la complexité des traitements à effectuer par l'application. [3]

•La Vue

Des pages JSP (java server page). Une page JSP est destinée à la vue, Elle est exécutée côté serveur et permet l'écriture de gabarits (pages en langage "client" comme HTML, CSS, JavaScript, XML, etc.). Elle permet au concepteur de la page d'appeler de manière transparente des portions de code Java, via des balises et expressions ressemblant fortement aux balises de présentation HTML. [3]

•Le Contrôleur (Servlet)

Une Servlet est un objet qui permet d'intercepter les requêtes faites par un client, et qui peut personnaliser une réponse en conséquence. Il fournit pour cela des méthodes permettant de scruter les requêtes HTTP. Cet objet n'agit jamais directement sur les données, il faut le voir comme un simple intermédiaire : il intercepte une requête issue d'un client, appelle éventuellement des traitements effectués par le modèle, et ordonne en retour à la vue d'afficher le résultat au client. [3]

- Ce paterne (le modèle MVC) est bien respecté dans un serveur WEB pour une architecture java EE web profile ou le conteneur EJB n'existe pas. Mai pour une application java EE full profile, le serveur d'application inclut les deux conteneurs (WEB et EJB) et nécessite l'utilisation des EJB entité qui remplace la couche modèle du modèle MVC.

B/ Partie EJB du serveur Java EE (Conteneur EJB)

Les conteneurs d'EJB fournissent un environnement d'exécution pour les EJB.

»Les EJB(Entreprise Java Bean)

Les EJB sont des composants logiciels résidant sur le serveur, ils permettent de développer des applications distribuées. Trois types d'EJB sont utilisées, les EJB session, les EJB entité et EJB orienté messages :

•Les EJB session

-Sont des composants logiciels accessibles à distance via RMI (Remote Method Invocation) et IIOP (Internet Inter-Orb Protocole) qui est une extension de RMI pour faire communiquer des objets CORBA. Ils ont une durée de vie correspond à un échange avec un client. Ils contiennent les règles métiers de l'application.

Il existe trois types d'EJB session :

- **Les EJB session sans état « stateless»**
  * Une instance pour plusieurs connexions clientes.
  * Ne conserve aucune donnée dans son état.
- **Les EJB session avec état « stateful »**
  * Création d'une instance pour chaque connexion cliente.
  * Conserve des données entre les échanges avec le client.
- **Les EJB session instance unique « Singleton»**
  * Création d'une instance unique quelque soit le nombre de connexion.

•Les EJB Entité

Sont des objets dans lesquels des annotations définissent des correspondances entre les objets et leurs attributs, et les tables relationnelles de la base de données. Ces EJB permettent de représenter et de gérer des données enregistrées dans une base de données qu'ils implémentent.

L'avantage d'utiliser un tel type est que certains services sont pris en charge par le conteneur assurent la persistance des données.

•Les EJB orienté messages

Représentent une file de messages postés par le client et traités par le serveur (ou vice-versa).

C/ La partie communication de java EE

La communication entre les différentes machines est assurée par différents protocole (TCP/IP, HTTP,RMI, ...etc).

»TCP/IP (Transmission Control Protocol/Internet Protocol)

Comme les flux Java sont transformés en format TCP/IP, il est possible de communiquer avec l'ensemble des ordinateurs qui utilisent ce même protocole.

» HTTP(Hyper Text Transfer Protocol)

Permet d'assurer la communication entre le navigateur et le serveur.

»RMI (Remote Methode Invocation)

RMI est une technologie développée et fournie par Sun à partir du JDK 1.1, dont le but est de créer un modèle objet distribué java.

Avec RMI les méthodes de certains objets (appelés objets distants) peuvent être invoquées depuis des JVM différentes (espaces d'adressages distincts). En effet RMI assure la communication entre les différents systèmes via TCP/IP et ce de manière transparent pour le développeur (gère tous les détails automatiquement).

La technologie RMI se charge de rendre transparente la localisation de l'objet distant, son appel et le renvoi du résultat. Elle permet aussi de faciliter la mise en œuvre et l'utilisation d'objets distants Java.
La figure suivante permet de représenter l'architecture de RMI :

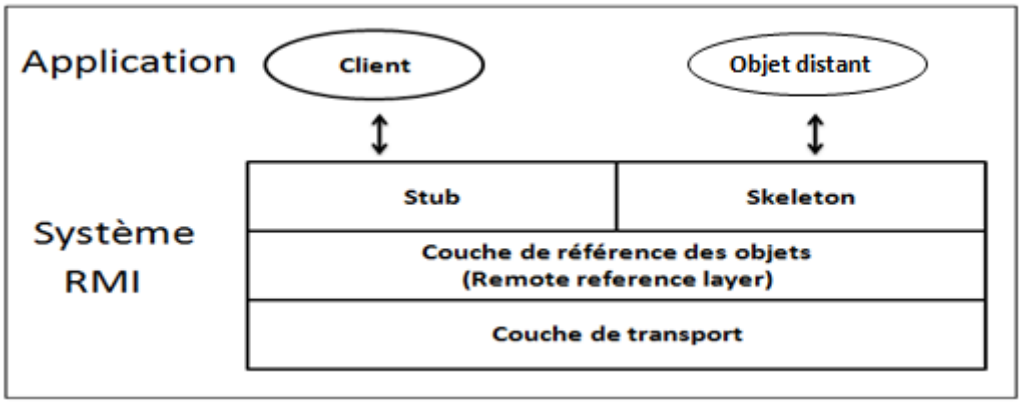

FIGURE 1.5 – L'achitecture de RMI

» **L'amorce (stub) :**
- Représentant local de l'objet distant (OD) qui implémente ses méthodes.
- Transmet l'invocation distante à la couche inférieure Remote Reference Layer.
- réalise le pliage "marshalling" des arguments des méthodes distantes.
- dans l'autre sens, il réalise le dépliage "demarshalling" des valeurs de retour. Il utilise pour cela la sérialisation des objets.

» **L'amorce (skeleton)**
- Réalise le dépliage des arguments reçus par le flux de pliage.
- Fait un appel à la méthode de l'objet distant.
- Réalise le pliage de la valeur de retour.

» **La couche des références des objets**
- Permet l'obtention d'une référence d'objet distant à partir de la référence locale au Stub.
- Ce service est assuré par le lancement du programme « rmiregister ».

» **La couche de transport**
- Connecte les 2 espaces d'adressage (JVM).
- Suit les connexions en cours.
- Ecoute et répond aux invocations.
- Construit une table des OD disponibles.

D/ La partie service dans le serveur java EE

Permet d'assurer un ensemble de services :

» JNDI

Est l'acronyme de Java Naming and Directory Interface. Cette API fournit une interface unique pour utiliser différents services de nommages ou d'annuaires et définit une API standard pour permettre l'accès à ces services.

Un service de nommage permet d'associer un nom unique à un objet et de faciliter ainsi l'obtention de cet objet. Il existe plusieurs types de services de nommage comme par exemple DNS (Domain Name System) qui est utilisé sur internet pour permettre la correspondance entre un nom de domaine et une adresse IP.

Un annuaire est un service de nommage qui possède en plus une représentation hiérarchique des objets qu'il contient et un mécanisme de recherche.

JNDI propose donc une abstraction pour permettre l'accès à ces différents services de manière standard. Ceci est possible grâce à l'implémentation de pilotes qui mettent en oeuvre la partie SPI (Service Provider Interface) de l'API JNDI. Cette implémentation se charge d'assurer le dialogue entre l'API et le service utilisé.[4]

» JAAS

Est l'acronyme de Java Authentication and Authorization Service. Cette API offre les services d'authentification et d'autorisations des utilisateurs ou systèmes externes interagissant avec l'application. [4]

– Authentification : s'assurer que le client est bien celui qu'il prétend être.

– Autorisation : s'assurer qu'il possède les droits de faire ce qu'il demande.

»JTA

Est l'acronyme de Java Transaction API, Le conteneur Java EE propose un support des transactions grâce à l'API JTA ; c'est la façon standard de gérer les transactions par le conteneur. [4]

Une transaction est un bloc d'instructions LMD (Langage de Manipulation des Données) faisant passer la base de données d'un état cohérent à un autre état cohérent. Si un problème logiciel ou matériel survient au cours d'une transaction, aucune des instructions contenues dans la transaction n'est effectuée, quel que soit l'endroit de la transaction où est intervenue l'erreur. [5]

Cette interface propose plusieurs méthodes :

- void begin() : Débuter la transaction.
- void commit() :Valider la transaction.
- void roolback() : Annuler la transaction.
- boolea isActive() :Déterminer si la transaction est active . [4]

» JavaMail

Est une API qui permet d'utiliser le courrier électronique (e-mail) dans une application écrite en Java (servlet, EJB, ...). Son but est d'être facile à utiliser, de fournir une souplesse qui permette de la faire évoluer et de rester le plus indépendant possible des protocoles utilisés.

Le courrier électronique repose sur le concept du client/serveur. Ainsi, l'utilisation d'e-mails requiert deux composants :

- Un client de mails (Mail User Agent : MUA) tel que Outlook, Messenger,...
- Un serveur de mails (Mail Transport Agent : MTA) tel que SendMail.

Les clients de mails s'appuient sur un serveur de mails pour obtenir et envoyer des messages. Les échanges entre clients et serveurs sont normalisés par des protocoles particuliers. [4]

» **JMS**

Est l'acronyme de Java Messaging Service. Cette API fournie par Sun pour permettre un dialogue standard entre des applications ou des composants via des brokers de messages ou MOM (Middleware Oriented Messages). Elle permet donc d'utiliser des services de messaging dans des applications Java comme le fait l'API JDBC pour les bases de données.

JMS définit plusieurs entités :

- Un provider JMS : outil qui implémente l'API JMS pour échanger les messages, ce sont les brokers de messages MOM.
- Un client JMS : composant écrit en java qui utilise JMS pour émettre et/ou recevoir des messages.
- Un message : données échangées entre les composants.

Différents objets utilisés via JMS sont généralement stockés dans l'annuaire JNDI du serveur d'applications ou du provider du MOM .

Les MOM ou brokers de messages permettent d'assurer l'échange de messages entre deux composants nommés client. Les deux clients n'échangent pas directement des messages : un client envoie un message et le client destinataire doit demander la réception du message. Le transfert du message et est assuré par le broker. [4]

» **JDBC**

Est l'acronyme de Java Data Base Connectivity, il est composée d'un ensemble de classes permettant le dialogue entre une application Java et une source de données compatibles SQL (tables relationnelles en général).

Pour pouvoir utiliser JDBC, il faut un pilote (driver) JDBC ; Un pilote est une couche logicielle chargée d'assurer la liaison entre l'application Java (cliente) et le SGBD (serveur). On distingue quatre types de drivers JDBC :

- Les pilotes de type 1 (JDBC-ODBC Bridge) : Utilisent la couche logicielle de Microsoft appelée ODBC (Open Data Base Connectivity). Dans ce type le pilote JDBC convertit les appels Java en appels ODBC avant de les exécuter.
- Les pilotes de type 2 (Native-API Partly-Java Driver) : Utilisent un pilote fourni par le constructeur de la base de données (natif), le pilote n'étant pas développé en Java. Cette approche convient pour les applications qui manipulent des sources de données uniques (tout Oracle par exemple).
- Les pilotes de type 3 (Net Protocol All-Java Driver) : Utilisent un pilote générique natif écrit en Java. les appels JDBC sont transformés par un protocole indépendant du SGBD. Cette approche convient pour des sources de données hétérogènes.
- Les pilotes de type 4 (Native Protocol All-Java Driver) : Sont écrits en Java, Les appels JDBC sont traduits en sockets exploités par le SGBD. Cette approche est la plus simple , elle convient pour tous types d'architectures. [5]

» JCA

Le but de l'API JCA (Java Cryptography Architecture) est de fournir des fonctionnalités cryptographiques de base à la plate-forme Java. [4]

» JMX

Est l'acronyme de Java Management eXtensions. Elle propose une API standard qui permet de gérer, de contrôler et de surveiller des applications, des composants, des services et même la JVM ou des périphériques dans la plate-forme Java. Donc son but est de proposer un standard pour faciliter le développement de systèmes de contrôle, d'administration et de supervision des applications et des ressources. Pour cella la plupart des principaux serveurs d'applications Java EE utilisent JMX pour la surveillance et la gestion de leurs composants. [4]

» Gestion du XML (DOM)

DOM est l'acronyme de Document Object Model. C'est une API qui permet de modéliser, de parcourir et de manipuler un document XML. Le principal rôle de DOM est de fournir une représentation mémoire d'un document XML sous la forme d'un arbre d'objets et d'en permettre la manipulation (parcours, recherche et mise à jour).

A partir de cette représentation (modèle), DOM propose de parcourir le document mais aussi de pouvoir le modifier. Ce dernier aspect est l'un des aspects les plus intéressants de DOM. [4]

1.5.3 Java EE et les bases de données

Les applications web traitent des données dans des volumes plus ou moins importants. Que ce volume devient assez important, les données sont stockées dans une base de données qui se trouve dans un serveur de base de données.

Dans les applications web java EE Le mapping Objet/Relationnel (mapping O/R) consiste à réaliser la correspondance entre le modèle de données relationnel et le modèle objets de la façon la plus facile possible via le JPA.

» JPA (Java Persistence API)

L'API Java Persistence repose sur des entités qui sont de simples POJOs (Plain Old Java Object) annotés et sur un gestionnaire de ces entités (EntityManager) qui propose des fonctionnalités pour les manipuler (ajout, modification, suppression, recherche). Ce gestionnaire est responsable de la gestion de l'état des entités et de leur persistance dans la base de données.

L'API propose un langage d'interrogation similaire à SQL mais utilisant des objets plutôt que des entités relationnelles de la base de données.

1.5.4 Les avantages des applications java EE

- Le java EE agrandit la portée de Java SE en fournissant des nouvelles API, et un environnement d'exécution pour la conception et l'exécution de logiciels d'entreprise distribuées.
- La sécurité des applications Java EE est assurée par la sécurité du conteneur Web et du conteneur d'EJB :
  - Si les règles de sécurité sont définies pour une ressource Web, le conteneur Web effectue un contrôle des accès lorsque la ressource est demandée par un client Web. Le conteneur Web interroge le client sur ses données d'authentification, et détermine si l'utilisateur est authentifié. Si oui l'utilisateur peut accéder aux ressources web demandées (Les servlets et les fichiers JSP).
  -Le conteneur d'EJB applique le contrôle d'accès lors de l'appel de ces méthodes EJB.

1.6 Conclusion

Nous avons présenté dans ce chapitre les technologies de base de développement des applications web. Nous avons détaillé la technologie java EE qui représente une bonne solution pour la distribution des applications d'entreprise tout en respectons le modèle MVC qui permet de réaliser un découpage très clair et robuste de l'application. Dans le chapitre suivant nous allons parler des bases de données distribuées qui jouent un rôle très important dans la distribution des applications Java EE.

Chapitre 2

Les bases de données réparties

- Introduction
- Qu'est ce qu'une base de données ?
- Qu'est ce qu'une base de données réparties ?
- Objectifs de répartition d'une base de données.
- Les avantages des bases de données réparties.
- Les inconvénients des bases de données réparties.
- Schéma globale et schéma locale.
- Conception des bases de données réparties.
- Techniques utilisés pour la distribution des bases de données .
- La réplication.
- Conclusion.

Introduction

L'évolution des techniques informatiques a permis d'adapter les outils informatiques à l'organisation des entreprises. Vu, le grand volume de données manipulées par ces dernières, la puissance des micro-ordinateurs, les performances des réseaux et la baisse considérable des coûts du matériel informatique ont permis l'apparition d'une nouvelle approche afin de remédier aux désagréments causés par la centralisation des données.

Les bases de données réparties permettent de répondre aux besoins de décentralisation des entreprises géographiquement réparties.

Ce chapitre sert à présenter les bases de données réparties , ses techniques de conception et de gestion qui sont un moyen très efficace pour pallier aux problèmes engendrés par l'approche centralisée.

2.1 Qu'est ce qu'une base de données ?

Une base de données peut être définit comme un ensemble structuré de données enregistrées sur des supports accessibles et exploitables qui modélisent un univers réel au moyen d'un ensemble de programmes. La gestion d'une base de données est assurée par un SGBD (Système de Gestion de Base de Données).

»Un SGBD

Est un ensemble de programmes qui permet la création et l'exploitation des bases de données. Le SGBD assure aussi l'interface entre la base de données physiques et les programmes d'application qui réalisent des accès, des traitements et des mises à jour.

•Le SGBD permet :

- L'accès aux données de façon simple.
- L'autorisation un accès aux informations à de multiples utilisateurs.
- La manipulation les données présentes dans la base de données (insertion, suppression, modification).

2.2 Qu'est ce qu'une base de données répartie ?

Une base de données centralisée est gérée par un seul SGBD, est stockée dans sa totalité à un emplacement physique unique et ses divers traitements sont confiés à une seule et même unité de traitement. Tandis que une base de données répartie est un ensemble de bases de données logiquement reliées et réparties entre plusieurs sites, et perçues par l'utilisateur comme une base unique, cette répartition affecte les données, les traitements, et les contrôles, ce qui permet a chaque site d'exécuter des transactions locales et participer à l'exécution de transactions globales.

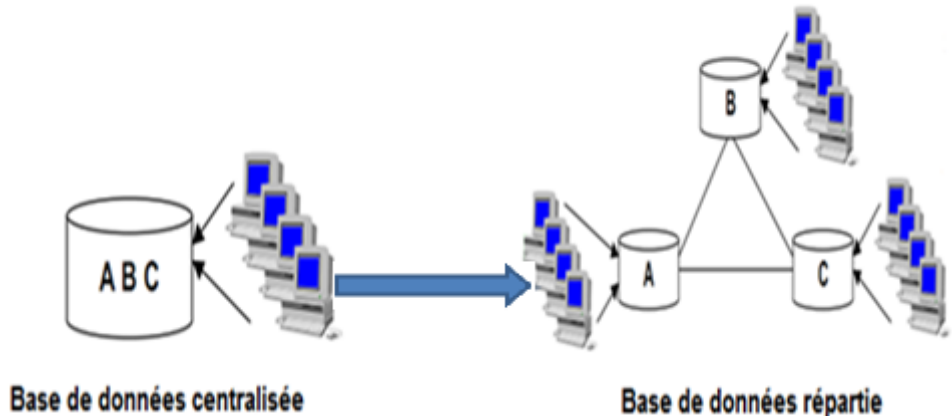

FIGURE 2.1 – De la base de données centralisée à la base de données répartie

Chaque base de données répartie sur un site est gérée par un SGBD. La totalité de la base de données répartie est gérée par un SGBDR.

»Le SGBDR (Système de Gestion de Base de Données Réparties) :

Est un système gérant une collection de BD logiquement reliées, réparties sur différents sites en fournissant un moyen d'accès rendant la distribution transparente à l'utilisateur.[6]

Le SGBDR permet la création, l'accès et la manipulation de données inter-reliées, localisées sur différents sites d'un réseau informatique.

•Les objectifs du SGBDR sont :

Un SGBDR dispose d'un dictionnaire de données et permet :

- Le traitement des requêtes réparties.
- La communication des données inter sites.
- La gestion de la cohérence et de la sécurité.

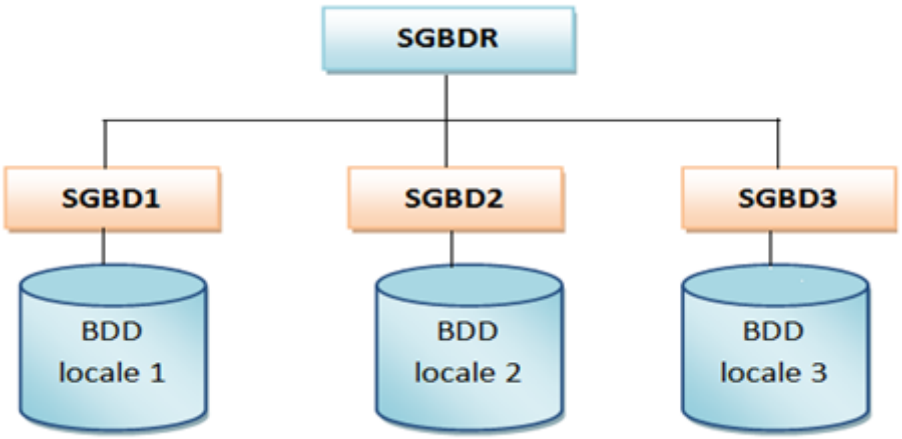

FIGURE 2.2 – Le SGBDR

2.3 Objectifs de répartition d'une base de données

Les objectifs de répartition d'une base de données peuvent être résumés ainsi :

- **Autonomie locale :**
  -La base de données locale est complète et autonome (intégrité, sécurité, gestion), elle peut évoluer indépendamment des autres.
- **Egalité entre sites :**
  -Un site en panne ne doit pas empêcher le fonctionnement des autres sites.
- **Fonctionnement continu :**
  -La distribution permet la résistance aux fautes et aux pannes.
- **Localisation transparente :**
  -Accès uniforme aux données quel que soit leur site de stockage (fragmentation transparente).
  -Les données répliquées doivent être maintenues en cohérence.
- **Requêtes distribuées :**
  -L'exécution d'une requête peut être répartie (automatiquement) entre plusieurs sites (si les données sont réparties).
- **Transactions réparties :**
  -Le mécanisme de transactions peut être réparti entre plusieurs sites.
- **Indépendance vis-à-vis du matériel :**
  -Le SGBD fonctionne sur les différentes plateformes utilisées.
- **Indépendance vis-à-vis du SE :**
  -Le SGBD fonctionne sur les différents systèmes d'exploitation.

- **Indépendance vis-à-vis du réseau :**
  -Le SGBD est accessible à travers les différents types de réseau utilisés.
- **Indépendance vis-à-vis du SGBD :**
  -La base peut être distribuée sur des SGBD hétérogènes.

2.4 Les avantages des bases de données réparties

Les objectifs de la répartition de données sont multiples :

- **Plus de fiabilité :** Les bases de données réparties ont souvent des données répliquées. La panne d'un site n'est pas très importante pour l'utilisateur, qui s'adressera à un autre site.
- **Meilleures performances :** Réduire le trafic sur le réseau est une possibilité d'accroître les performances. Le but de la répartition des données est de les rapprocher de l'endroit où elles sont accédées. Répartir une base de données sur plusieurs sites permet de répartir la charge sur les processeurs et sur les entrées/ sorties.
- **Faciliter l'accroissement :** Il est plus facile d'améliorer les performances du système de gestion de la base de données, et cela fait par l'ajout de machines sur le réseau. [7]

2.5 Les inconvénients des bases de données réparties

- Complexité des SGBDs.
- Risque d'erreurs plus important.
- Administration complexe.

2.6 Schéma globale et schéma locale

2.6.1 Schéma global

Le schéma global permet de définir l'ensemble des types de données de la base. Il ignore les concepts d'implémentation, chaque base locale en implémente une partie.

2.6.2 Schéma local

Une base de données locale comporte un schéma géré par le SGBD local. Dans une BD répartie, chaque base locale rend visible toute ou une partie de la base aux sites clients. [8]

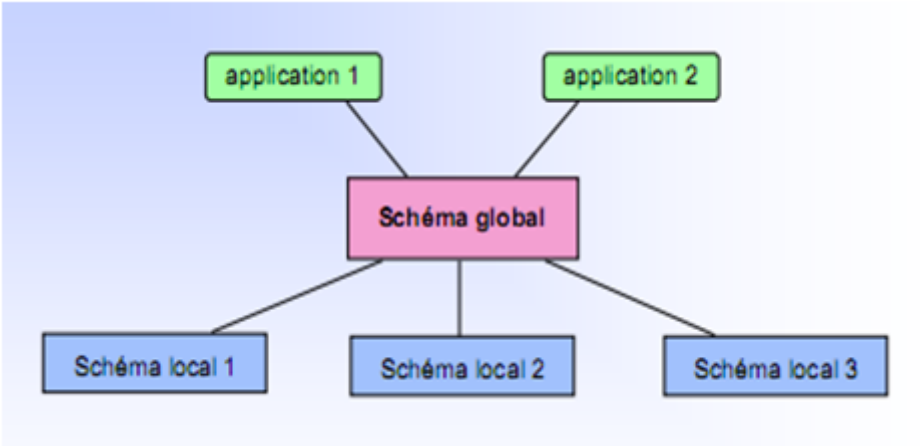

FIGURE 2.3 – Schéma locale et globale pour une BDR

2.7 Conceptions d'une base de données répartie

La conception d'une base de données répartie peut être le résultat de deux approches totalement distinctes, soit le regroupement d'une multitude de bases de données déjà existantes (approche ascendante) soit construite du zéro (approche descendante).

Pour la conception des bases de données réparties il faut prendre des décisions en fonction de critères techniques et organisationnels pour l'objectif de minimiser le nombre de transferts entre sites, les temps de transfert, le volume de données transférées, les temps moyens de traitement des requêtes... etc.

2.7.1 Approche descendante (top down design)

On commence par définir un schéma conceptuel global de la base de données répartie, puis on distribue sur les différents sites en des schémas conceptuels locaux.

La répartition se fait donc en deux étapes, en première étape la fragmentation, et en deuxième étape l'allocation de ces fragments aux sites.

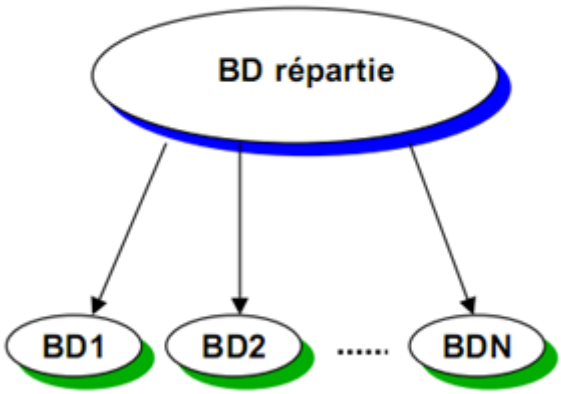

FIGURE 2.4 – Conception descendante d'une BDR

2.7.2 Approche ascendante (bottom up design)

L'approche se base sur le fait que la répartition est déjà faite, mais il faut réussir à intégrer les différentes bases de données existantes en une seule BD globale (base de données fédérée). En d'autres termes, les schémas conceptuels locaux existent et il faut réussir à les unifier dans un schéma conceptuel global. L'approche bottom up est utilisée lorsque les bases de données existent déjà.[7]

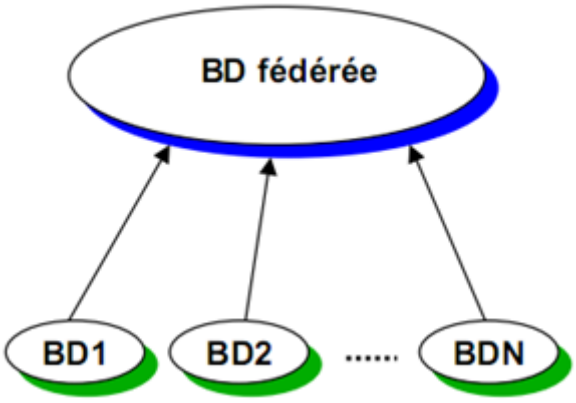

FIGURE 2.5 – Conception ascendante d'une BDR

2.8 Techniques utilisés pour la distribution des bases de données

La distribution de la base de données se fait en deux étapes, en première étape la fragmentation, et en deuxième étape l'allocation de ces fragments aux sites avec ou sans réplication. La distribution de la base de données est représentée dans la figure suivante.

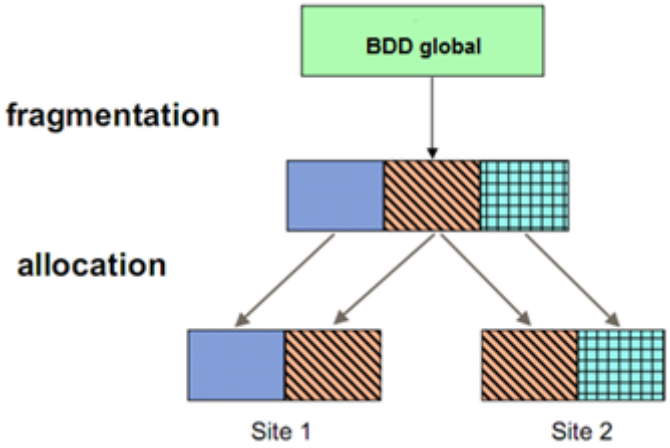

FIGURE 2.6 – Décomposition d'une BDD global

2.8.1 Fragmentation des données

La fragmentation est le processus de décomposition d'une base de données en un ensemble de sous bases de données. Cette décomposition doit être sans perte d'information.

L'utilisation de petits fragments permet de faire tourner plus de processus simultanément, ce qui entraîne une meilleure utilisation des capacités du réseau d'ordinateurs.

Elle se base sur les règles suivantes :

- **La complétude :** Pour toute donnée d'une relation R, il existe un fragment Ri de la relation R qui possède cette donnée.
- **La reconstruction :** Pour toute relation décomposée en un ensemble de fragments Ri, il existe une opération de reconstruction. [7]

Il existe généralement 3 types de fragmentation : horizontale, verticale, et mixte la figure suivante illustre ces trois types.

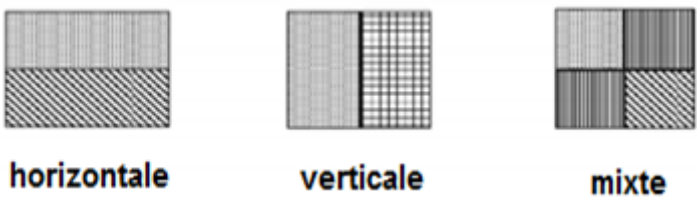

FIGURE 2.7 – Types de fragmentation

» Fragmentation horizontale

Dans ce type de fragmentation les fragments sont définis par sélection (Les tuples sont répartis).

$$*L'opérateur \quad de \quad partitionnement \quad est \quad la \quad sélection(\theta)$$

$$*L'opérateur \quad de \quad recomposition \quad est \quad l'union(\cup)$$

L'exemple suivant permet d'expliquer cette approche :

Les tuples de la relation Client seront fragmenter en deux fragments (Client1 et Client2).

Client

nclient	nom	Ville
C1	Naim	Mila
C2	Oussama	Annaba
C3	Amir	Mila
C4	Ramzi	Setif

$Client1 = \theta[where\ ville = "Mila"]Client$

Client1

nclient	nom	Ville
C1	Naim	Mila
C3	Amir	Mila

$Client2 = \theta[where\ ville \neq "Mila"]Client$

Client2

nclient	nom	Ville
C2	Oussama	Annaba
C4	Ramzi	Setif

$$Pour \quad la \quad Reconstruction : \quad Client = \quad Client1 \cup Client2$$

» Fragmentation verticale

La fragmentation verticale consiste à découper et fragmenter les tuples d'une relation. Elle nécessite une colonne commune dupliquée (la clé doit apparaître dans tous les groupes).

$$*L'opérateur \quad de \quad partitionnement \quad est \quad la \quad projection(\pi)$$

$$*L'opérateur \quad de \quad recomposition \quad est \quad la \quad jointure(*)$$

Les tuples de la relation Client seront découper et fragmenter en deux fragments (Client1 et Client2)

Client

nclient	nom	Ville
C1	Naim	Mila
C2	Oussama	Annaba
C3	Amir	Mila
C4	Ramzi	Setif

$$Client1 \quad = \quad \pi \quad [nclient, nom] \quad Client$$

Client1

nclient	nom
C1	Naim
C2	Oussama
C3	Amir
C4	Ramzi

$$Client2 \quad = \quad \pi \quad [nclient, ville] \quad Client$$

Client2

nclient	Ville
C1	Mila
C2	Annaba
C3	Mila
C4	Setif

$$Pour \quad la \quad Reconstruction : \quad Client = Client1 * Client2$$

» **Fragmentation mixte**

C'est la combinaison des deux fragmentations précédentes, horizontale et verticale.

- L'opération de partitionnement est une combinaison de projections et de sélections.
- L'opération de recomposition est une combinaison de jointures et d'unions.

2.8.2 Allocation des fragments aux sites

L'affectation des fragments sur les sites est décidée en fonction des requêtes qui ont servi à la fragmentation. Le but est de placer les fragments sur les sites où ils sont le plus utilisés, et ce pour minimiser les transferts de données entre les sites.

L'allocation peut se faire avec réplication ou sans réplication. Sachant que la réplication favorise les performances des requêtes et la disponibilité des données, mais est coûteuse en considérant les mises à jour des fragments répliquas. [7]

2.9 La réplication

2.9.1 Pourquoi la réplication ?

Confrontées à des risques constants menaçant la continuité et la rapidité d'accès à leurs données, les entreprises mettent en place des systèmes assurant la sauvegarde et la duplication de leurs bases à travers le monde.

2.9.2 Définition

La réplication est une techniques implémentée par plusieurs système de gestion de bases de données, afin d'augmenter la disponibilité des données et d'améliorer la fiabilité des systèmes.

La réplication est définie comme étant le processus de duplication et d'entretien d'objets de la base de données. Plusieurs copies de la même base de données sont hébergées sur des sites distincts, tout en garantissant la consistance et la cohérence des copies. Ceci dans le but d'augmenter la fiabilité du système, d'améliorer la tolérance aux pannes et d'assurer

une haute disponibilité des données, qui permet en conséquent d'améliorer l'accessibilité et les performances.

La réplication est une technologie spécifique des bases de données réparties dont l'objectif est de partager les données parmi des sites multiples. Dans le même cadre, on identifier aussi la technique de fragmentation (appelé aussi partitionnement). Néanmoins, les bases de données répliquées et les bases de données fragmentées représentent deux notions distinctes. En fait, dans une base de données fragmentée, les données sont disponibles dans plusieurs emplacements, mais une relation, voire un tuple, particulier et hébergé à un emplacement bien précis. Tandis que la réplication signifie que les mêmes données sont disponibles dans plusieurs sites.

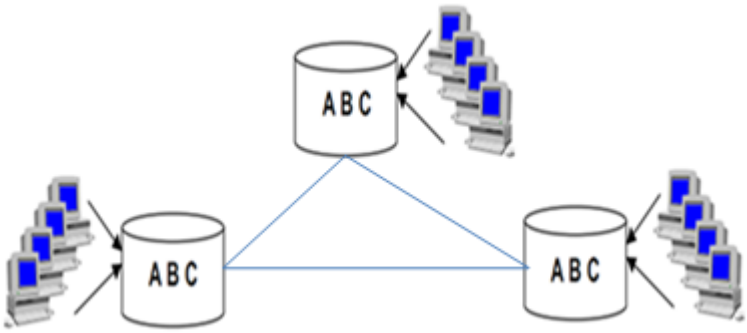

FIGURE 2.8 – Réplication de données

2.9.3 Les Types de réplication

A.La réplication synchrone

Aussi appelée « Réplication en temps réel » La synchronisation est effectuée en temps réel puisque chaque requête est déployée sur l'ensemble des bases de données avant la validation (commit) de la requête sur le serveur où la requête est exécutée. Ce type de réplication assure un haut degré d'intégrité des données mais requiert une disponibilité permanente des serveurs et de la bande passante. Ce type de réplication, fortement dépendant des pannes des systèmes, nécessite de gérer des transactions multi sites coûteuses en ressources. La réplication synchrone est utilisée dans le cas des compagnies aériennes, des banques. . . etc .

Deux type de réplication synchrone : La réplication synchrone asymétrique et la réplication synchrone symétrique.

A.a La réplication synchrone asymétrique

Elle utilise un site primaire qui propage les mises à jour en temps réel vers un ou plusieurs sites secondaires. La table répliquée est immédiatement mise à jour pour chaque modification par utilisation de trigger sur la table maître.

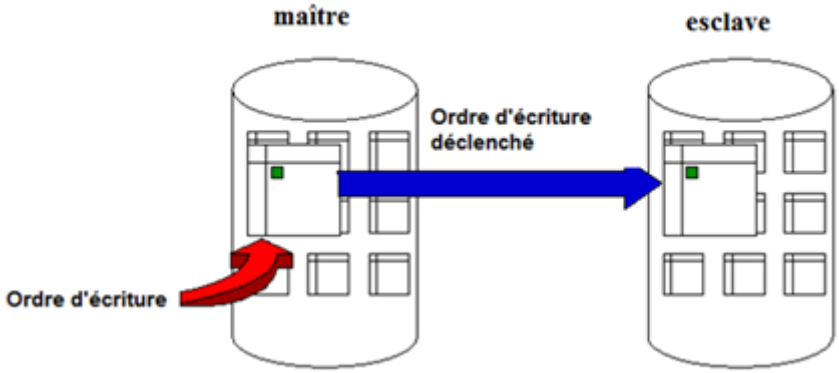

FIGURE 2.9 – Réplication synchrone asymétrique

A.b La réplication synchrone symétrique

Dans ce cas, il n'y a pas de table maître, mais chaque table possède ses triggers, déclenchés lors d'une modification. Il est alors nécessaire de définir des priorités et de gérer les blocages des tables en attendant qu'une modification soit entièrement propagée. D'autre part, les triggers doivent différencier les mises à jour issues d'une réplication des mises à jour de requête directes.

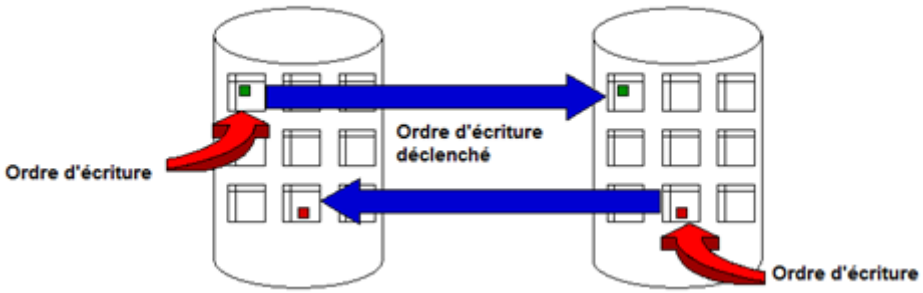

FIGURE 2.10 – Réplication synchrone symétrique

B.La réplication asynchrone

La réplication asynchrone stocke les opérations intervenues sur une base de données dans une fille d'attente pour les propager plus tard à l'aide d'un processus de synchronisation. Ce type de réplication est plus flexible que la réplication synchrone. Il permet en effet de définir les intervalles de synchronisation, ce qui permet à un site de fonctionner même si un site de réplication n'est pas accessible. Si le site distant est victime d'une panne, l'absence de synchronisation n'empêche pas la consistance de la base maître. La réplication asynchrone est appropriée pour une équipe de développeurs travaillant à distance sur une application.

Deux type de réplication asynchrone : La réplication asynchrone asymétrique et la réplication asynchrone symétrique.

B.a La réplication asynchrone asymétrique

Elle propage les mises à jour en temps différé via une file persistante. Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, l'horloge par exemple.

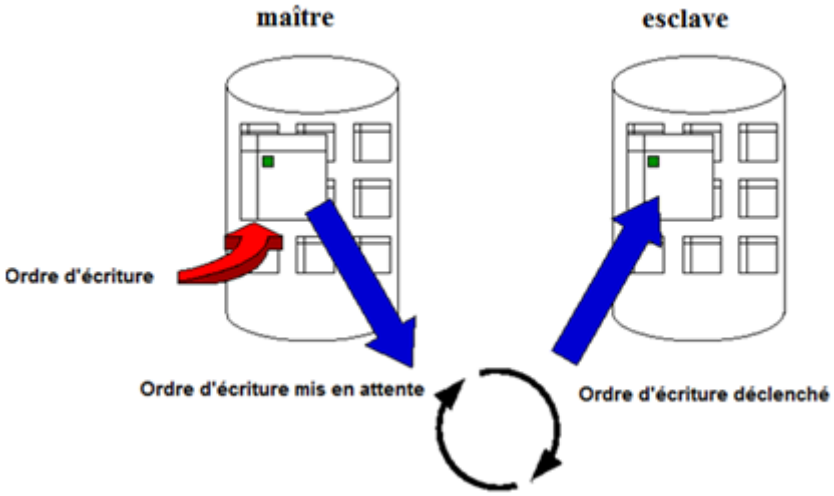

FIGURE 2.11 – Réplication asynchrone asymétrique

B.b La réplication asynchrone symétrique

Dans ce cas, la mise à jour des tables répliquées est différée. Cette technique risque de provoquer des incohérences de données.

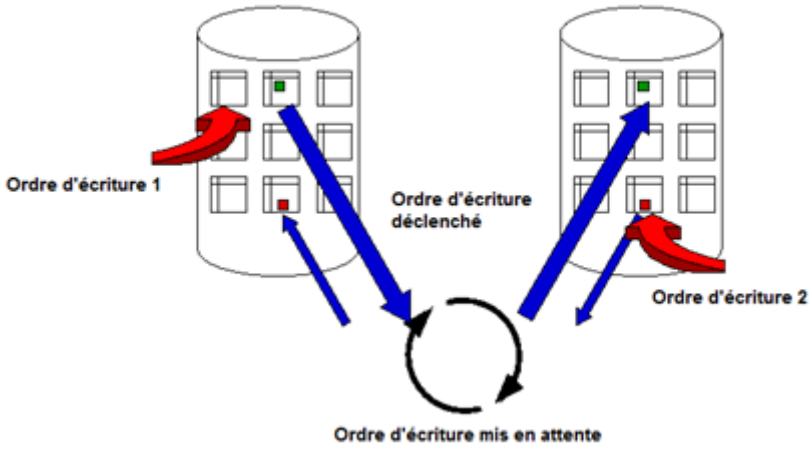

FIGURE 2.12 – Réplication asynchrone symétrique

-Il est bien évident que la réplication synchrone nécessite des moyens beaucoup plus coûteux que l'asynchrone, en raison principalement de la bande passante nécessaire pour assurer un échange parfait entre les différents serveurs d'un réseau. Les solutions asynchrones sont donc privilégiées par les entreprises dont l'exigence en "temps réel" et en continuité de service est réduite.

2.9.4 Les avantages de la réplication

- **La disponibilité :** Les utilisateurs et les applications ont une multitude de chemins d'accès à une même donnée. Ainsi, si un site devient inaccessible pour une raison ou pour une autre. Le système peut continuer à répondre aux requêtes des utilisateurs de manière transparente.
- **La fiabilité :** La défaillance d'un site n'implique pas l'arrêt du système, les utilisateurs connectés au site défaillant sont immédiatement servis par un autre en marche.
- **Les performances :** La réplication accélère l'exécution des requêtes utilisateurs, Certains utilisateurs peuvent accéder à plus d'un serveur, Tandis que d'autres utilisateurs accèdent à des serveurs différents.
- **Vues de données selon utilisation :** La réplication permet de créer plusieurs copiés hébergées sur des sites distincts , ou chaque site sera complément dédié à rependre aux requêtes bien précis.

2.10 Conclusion

Les bases de données réparties représentent un domaine important pour la gestion de grand volume de données pour les entreprises réparties géographiquement.

Dans ce chapitre, nous avons vu c'est quoi une base de données répartie ces techniques de conception et le gain que nous pouvons obtenir lorsque nous répartissons une base de données. Nous avons vu également, l'intérêt de la réplication dans ce genre de base de données. Dans le chapitre suivant nous allons passer à la phase de représentation de notre projet ainsi que la modélisation avec UML.

Chapitre 3

Modélisation avec UML

- **Introduction.**

- **Phase 01 :Identification des besoins.**

- **Phase 02 :Analyse et conception.**

Introduction

Ce chapitre constitue l'essentiel de notre travail.Pour réaliser notre projet Nous allons suivre une méthode simple et générique qui comporte deux phases essentielles ; la première phase est l'identification des besoins, la deuxième phase est l'analyse et la conception de notre application. Nous utilisons également les diagrammes UML pour modéliser les différents composants et fonctionnalités de notre application.

Phase 01 : Identification des besoins.

Introduction

Dans cette phase nous allons identifier les besoins du système à réaliser. Au début nous allons définir le cahier de charges qui contient toutes les objectives de ce projet. Puis nous modélisons le système avec le diagramme de cas d'utilisation. Ensuite nous décrivons la description textuelle des cas d'utilisation.

3.1 Élaboration du cahier de charge

3.1.1 Présentation du projet

Notre projet consiste à réaliser une application web intégrant une base de données répartie pour la gestion des ventes et le suivie des voitures entre une agence centrale et un ensemble de représentants.

Principalement les échanges d'informations est entre l'agence centrale et ses représentants, ils concernent les commandes de voitures, l'affectation des commandes et la gestion des pannes des voiture. Des autres échanges peuvent se déroulé entre les représentants eux mêmes, ils concernent le dépannage des différents clients.

»L'agence centrale :

Cette agence se charge de la distribution des voitures pour un ensemble de représentants d'une côté, et d'autre côté elle gère les pannes des voitures vendues par ses représentants :

- Elle reçoit les commandes envoyées par ses représentants.
- Elle affecte les voitures demandées aux représentants.
- Elle reçoit les statistiques des pannes calculées par ses représentants.

»Le représentant :

Le représentant s'occupe de la demande et de la vente des voitures et le dépannage des clients :

- Le représentant demande à l'agence centrale un certain nombre de voitures.
- Le représentant vend les voitures aux clients.

- Le représentant est responsable du dépannage de ses clients, et aussi les clients des autres représentants.
- Les représentants échanges les informations de dépannages entre eux.

3.1.2 Recueil des besoins fonctionnels

- **Au niveau de l'agence central**

» **La gestion des commandes :**

- L'affectation d'une commande : Permet à l'administrateur central d'affecter un certain nombre de voitures à un représentant après avoir vérifié la disponibilité des voitures qu'il a commandées.
- La recherche sur une commande : Permet à l'administrateur central de rechercher et consulter une commande.

» **La gestion des représentants :**

- L'ajout d'un nouveau représentant : Permet à l'administrateur central d'ajouter un nouveau représentant.
- La modification d'un représentant : Permet à l'administrateur central de modifier les informations d'un représentant.
- L'envoi des représentants : Permet à l'administrateur central d'envoyer a chaque représentant la liste de coopérants.

» **Le calcul des statistiques des pannes :**

- Permet à l'administrateur central de calculer les statistiques des pannes des voitures vendues par ses représentants pour améliorer la qualité de ses produits.

- **Au niveau du représentant local**

» **Gestion des commandes :**

- Commander voitures : Permet à l'administrateur local de créer une commande caractérisée par un modèle et une quantité et d'envoyer cette commande à l'agence centrale.
- Annulation d'une commande envoyée : Permet à l'administrateur local d'annuler une commande déjà envoyée à l'agence centrale.
- Vérifier satisfaction d'une commande envoyée : Permet à l'administrateur local de vérifier si une commande a été satisfaite ou non.

» **Vendre voiture :**

- Vérifier la disponibilité d'une voiture : Permet à l'administrateur local de vérifier si la voiture demandée par le client est disponible ou non.
- Vendre voiture : Permet à l'administrateur local de vendre une voiture à un client.

» **Gérer panne :**

- Dépannage avec garantie : Permet à l'administrateur local de dépanner une voiture qui possède une garantie après avoir vérifié la validité de ses informations.
- Dépannage sans garantie : Permet à l'administrateur local de dépanner une voiture.

3.1.3 Recueil des besoins opérationnels

- **Sécurité :** Chaque utilisateur de l'application doit s'authentifier par un nom d'utilisateur et un mot de passe, pour qu'il puisse utiliser le système et accéder à l'application.
- **Temps de réponse :** Doit être acceptable.
- **Interface graphique :** Doit être simple à utiliser et convivial.

3.1.4 Identification des acteurs

Les acteurs sont des entités externes qui interagissent avec le système, comme une personne humaine, un autre système ou un robot.Dans notre

application on distingue 2 catégories d'acteurs :

»L'administrateur d'agence centrale :

Utilisateur au niveau de l'agence centrale. son rôle se résume à :
- La gestion des commandes.
- La gestion des représentants.
- Le calcul des statistiques des pannes.

»L'administrateur du représentant local :

Utilisateur au niveau du représentant local. son rôle se résume à :
- La gestion des commandes.
- La vente des voitures.
- La gestion des pannes.

3.2 Diagramme de cas d'utilisation

3.2.1 Définition

Un diagramme de cas d'utilisation permet de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Il permet aussi de capturer le comportement d'un système tel qu'un utilisateur extérieur le voit. [9]

3.2.2 Identification des cas d'utilisation

Un cas d'utilisation « use case » permet de décrire ce que le futur système devra faire, sans spécifier comment il le réalisera. Il permet d'exprimer le besoin des utilisateurs d'un système, il est donc une vision orientée utilisateur.

Maintenant nous allons représenter notre diagramme de cas d'utilisation.

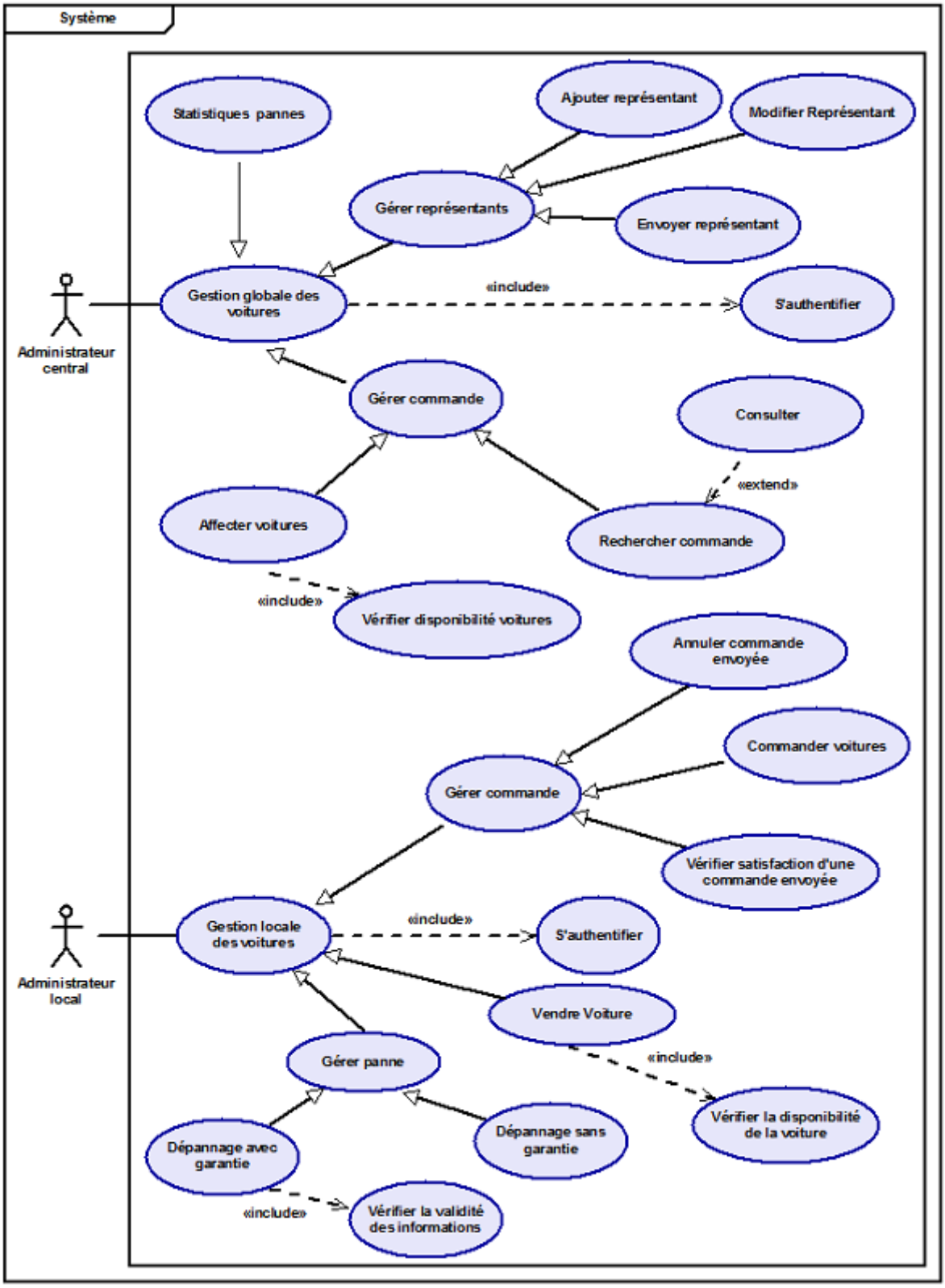

FIGURE 3.1 – Diagramme de cas d'utilisation

3.2.3 Description textuelle des cas d'utilisation

Maintenant nous allons détailler chaque cas d'utilisation par une description textuelle qui représente une forme souple pour l'illustration des différentes situations.

A. Au niveau du représentant :

Nous représentons ici la description textuelle des cas d'utilisations au niveau du système du représentant.

A.1 **S'authentifier**

– Fiche descriptive

Titre	S'authentifier.
But	Vérifier l'autorisation d'accéder au système.
Résumé	Permet à l'administrateur d'accéder au système.
Acteur	Administrateur local/central.

TABLE 3.1 – Sommaire d'identification(S'authentifier)

Pré conditions	**-** L'administrateur possède les informations d'accès.
Post condition	**-** L'administrateur est authentifié par le système.
Scénario nominal	**-** Ce cas d'utilisation commence lorsque l'administrateur veut accéder au système et choisit pour cela l'option « S'authentifier » : **.** Le système demande à l'utilisateur de saisir le login et le mot de passe. **.** L'administrateur saisit le login et le mot de passe et valide. **.** Le système lit le login et le mot de passe et les vérifie. **.** Le système donne à l'administrateur l'autorisation d'accès.
Scénario alternatif	**-** Si le login et le mot de passe sont erronés : **.** Le système affiche un message d'erreur. **.** Le système propose à l'utilisateur de saisir une nouvelle fois le login et le mot de passe.

TABLE 3.2 – Description détaillée(S'authentifier)

A.2 Commander voitures

– Fiche descriptive

Titre	Commander voitures.
But	Créer et envoyer une nouvelle commande.
Résumé	Permet à l'administrateur d'ajouter une nouvelle commande dans la liste des commandes du système local ainsi que d'envoyer cette commande au système central.
Acteur	Administrateur local.

Table 3.3 – Sommaire d'identification(Commander voitures)

Pré conditions	**-**L'administrateur est authentifié.
Post condition	**-**La commande est ajoutée à la liste des commandes du système local et central.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur veut commander des voitures et choisit pour cela l'option «Commander voitures » : **.**Le système affiche un formulaire de création **.**L'administrateur remplie le formulaire de commande et valide les informations saisies. **.** Le système lit les informations saisies et les vérifie. **.** Le système affiche un message « l'opération est terminée avec succès».
Scénario alternatif	**-**Si l'un des champs n'est pas rempli : **.** le système affiche un message «il faut remplir tous les champs ».
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à commander les voitures : **.** le système affiche un message d'erreur« il y a un problème soit au niveau du serveur d'application local /distant,soit au niveau du réseau de communication,soit au niveau du serveur de la BDD local /distant»

Table 3.4 – Description détaillée(Commander voitures)

A.3 Annuler commande envoyée

– Fiche descriptive

Titre	Annuler commande envoyée.
But	Annuler une commande déjà envoyé.
Résumé	Permet à l'administrateur d'annulée une commande déjà envoyé a la BDD centrale.
Acteur	Administrateur local.

TABLE 3.5 – Sommaire d'identification(Annuler commande envoyée)

Pré conditions	**-**Une commande est déjà envoyée par L'administrateur.
Post condition	**-**La commande est supprimée de la BDD central via une requête envoyée par le système local.
Scénario nominal	**-** Ce cas d'utilisation commence lorsque l'administrateur veut annuler une commande déjà envoyée et choisit pour cela l'option «annuler commande envoyée ». **.** système affiche la liste des commandes envoyées et non affectées. **.** L'administrateur sélectionne la commande qu'il veut annuler. **.** Le système demande la confirmation de l'annulation. **.** L'administrateur confirme l'action. **.** Le système affiche un message « annulation réussi».
Scénario exceptionnel	**-** Lorsque le système ne parvient pas d'annuler la commande : **.**Le système affiche un message d'erreur « il y a un problème au niveau du serveur d'application local /distant, soit au niveau du réseau de communication, soit au niveau du serveur de la BDD local /distant»

TABLE 3.6 – Description détaillée(Annuler commande envoyée)

A.4 Vérifier satisfaction d'une commande envoyée

– Fiche descriptive

Titre	Vérifier satisfaction d'une commande envoyée.
But	Vérifier satisfaction d'une commande envoyée.
Résumé	Permet à l'administrateur local de vérifier si une commande a été satisfaite par le système central ou pas.
Acteur	Administrateur local.

TABLE 3.7 – Sommaire d'identification(Vérifier satisfaction d'une commande envoyée)

Pré conditions	**-** Les commandes sont envoyées.
Post condition	/
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur local veut vérifier la satisfaction d'une commande et choisit pour cela l'option «vérifier satisfaction d'une commande envoyée » **.** Le système affiche la liste des commandes envoyées et non satisfaites. **.** L'administrateur local choisit les commandes qu'il veut vérifier. **.** Le système local vérifie la satisfaction des commandes. **.** Le système affiche le résultat concerné.
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à vérifier la satisfaction d'une commande envoyée : **.** le système affiche un message d'erreur « il y a un problème au niveau du serveur d'application local ou au niveau du serveur de BDD local ».

TABLE 3.8 – Description détaillée(Vérifier satisfaction d'une commande envoyée)

A.5 Vérifier la disponibilité de la voiture

– Fiche descriptive

Titre	Vérifier la disponibilité de la voiture.
But	Vérifier la disponibilité de la voiture pour une opération de vente.
Résumé	Permet à l'administrateur local de vérifier la disponibilité de la voiture selon les conditions du client.
Acteur	Administrateur local.

TABLE 3.9 – Sommaire d'identification(Vérifier la disponibilité de la voiture)

Pré conditions	**-** L'administrateur est authentifié.
Post condition	/
Scénario nominal	**-** Ce cas d'utilisation commence lorsque l'administrateur local veut vérifier la disponibilité d'une voiture et choisit pour cela l'option « vérifier la disponibilité de la voiture » **.** Le système affiche un formulaire pour saisir le modèle, la couleur et l'année de fabrication que l'utilisateur souhaite. **.** L'administrateur remplie le formulaire et valide les informations saisies. **.** Le système lit les informations saisies et les vérifie. **.** Le système affiche une liste de voitures ayant les caractéristiques demandées. **.** L'administrateur choisit une voiture parmi la liste des voitures affichées.
Scénario alternatif	**-** si l'administrateur local annule la vérification : **.** Le système local n'interroge pas la BDD locale
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à vérifier la disponibilité de la voiture : **.** Le système affiche un message d'erreur « il y a un problème dans le serveur d'application local, ou le serveur de la BDD local ».

TABLE 3.10 – Description détaillée(Cas d'utilisation : Vérifier la disponibilité de la voiture)

A.6 Vendre voiture

– Fiche descriptive

Titre	Vendre voiture.
But	Vendre une voiture à un client.
Résumé	Permet à l'administrateur local de vendre une voiture à un client.
Acteur	Administrateur local.

TABLE 3.11 – Sommaire d'identification(Vendre voiture)

Pré conditions	**-**L'administrateur local a vérifié la disponibilité de la voiture.
Post condition	**-** La voiture est affectée à un client.
Scénario nominal	**-** Ce cas d'utilisation commence lorsque l'administrateur local veut vendre une voiture et choisit pour cela l'option «vendre voiture» : **.** Le système affiche les informations concernant la voiture choisie et demande à l'administrateur local de saisir le type du garantie choisie ainsi que le nom et le prénom du client. **.** Si le client n'existe pas dans la BDD le système affiche un formulaire de création. **.** L'administrateur local remplie le formulaire de création et valide les informations saisies. **.** Le système lit les informations saisies et les vérifie. **.** Le système affiche un message« vente voiture réussi ». **.** Si le client existe déjà le système affiche les informations de la voiture choisit et du client, et demande à l'administrateur local de sauvegarder les informations. **.** Le système affiche un message « vente voiture réussi ».
Scénario alternatif	**-** Dans le cas d'un nouveau client si l'un des champs n'est pas remplit : **.** le système affiche dans un message «Il faut remplir tous les champs»
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à vendre une voiture a un client : **.** le système affiche un message d'erreur « il y a un problème dans le serveur d'application local, ou le serveur de la BDD local»

TABLE 3.12 – Description détaillée(Vendre voiture)

A.7 Dépannage sans garantie

– Fiche descriptive

Titre	Dépannage sans garantie.
But	Dépanner une voiture avec paiement.
Résumé	Permet à l'administrateur local de dépanner une voiture sans garantie.
Acteur	Administrateur local.

TABLE 3.13 – Sommaire d'identification(Dépannage sans garantie)

Pré conditions	**-** L'administrateur local est authentifié.
Post condition	**-** La panne est ajoutée à la base de données locale.
Scénario nominal	**-** Ce cas d'utilisation commence lorsque l'administrateur local veut ajouter une panne d'une voiture qui ne possède pas une garantie et il choisit pour cela l'option «dépannage sans garantie» **.** Le système affiche un formulaire pour saisir le numéro de chassis et le type de dépannage et la description de panne. **.** L'administrateur local remplie le formulaire et valide les informations saisies. **.** Le système lit les informations saisies et les vérifier. **.** Le système affiche un message «ajout avec succès ».
Scénario alternatif	**-**Si l'un des champs n'est pas remplit : **.** Le système affiche dans un message «Il faut remplir tous les champs ».
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à ajouter la panne : **.** Le système affiche un message d'erreur « il y a un problème soit au niveau du serveur d'application local ,soit au niveau du serveur de la BDD local».

TABLE 3.14 – Description détaillée(Dépannage sans garantie)

A.8 Vérifier la validité des informations

– Fiche descriptive

Titre	Vérifier la validité des informations.
But	Vérifier la validité des informations pour l'ajout d'une panne avec garantie.
Résumé	Permet à l'administrateur local de vérifier la validité des informations fournies par le client pour l'ajout d'une panne avec garantie.
Acteur	Administrateur local.

TABLE 3.15 – Sommaire d'identification (Vérifier la validité des informations)

Pré conditions	**-**L'administrateur local est authentifié.
Post condition	**-**Possibilité d'ajout ou non d'une panne.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur local veut vérifier la validité de la garantie d'une voiture et choisit pour cela l'option «vérifier la validité des informations» **.** Le système affiche un formulaire pour saisir le numéro de chassis et la garantie correspondante. **.** L'administrateur local remplie le formulaire et valide les informations saisies. **.** Si la personne est un client du représentant alors le système vérifie les informations saisies localement, sinon il interroge la BDD du représentant concerné. **.** Le système affiche un message« informations valides/invalides ».
Scénario alternatif	**-**Si l'administrateur local annule la vérification : **.** le système local n'interroge pas la BDD locale/distant.
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à vérifier la validité des informations : **.** le système affiche un message d'erreur « il y a un problème soit au niveau du serveur d'application local/distant, soit au niveau du réseau, soit au niveau du serveur de la BDD local/distant »

TABLE 3.16 – Description détaillée(Vérifier la validité des informations)

A.9 Dépannage avec garantie

– Fiche descriptive

Titre	Dépannage avec garantie.
But	Dépanner une voiture sans paiement.
Résumé	Permet à l'administrateur local de dépanner une voiture avec garantie.
Acteur	Administrateur local.

TABLE 3.17 – Sommaire d'identification(Dépannage avec garantie)

Pré conditions	**-**L'administrateur local a vérifié la validité des informations.
Post condition	**-**La panne est ajoutée à la liste des pannes du système local.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur veut ajouter une panne d'une voiture qui possède une garantie et choisit pour cela l'option «dépannage avec garantie». **.** Le système affiche un formulaire pour saisir le numéro de chassis et la garantie(et le nom du représentant en cas de client distant). **.** L'administrateur local remplie le formulaire et valide les informations saisies. **.** Le système lit les informations saisies et les vérifie. **.** Le système affiche un message «ajout avec succès ».
Scénario alternatif	**-** Si l'un des champs n'est pas remplit : **.** Le système affiche dans un message «Il faut remplir tous les champs ».
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à ajouter la panne : **.** Le système affiche un message d'erreur «il y a un problème soit au niveau du serveur d'application local/distant, soit au niveau du réseau, soit au niveau du serveur de la BDD local/distant»

TABLE 3.18 – Description détaillée(Dépannage avec garantie)

B. Au niveau de l'agence centrale :

Nous représentons ici la description textuelle des cas d'utilisations au niveau du système central.

B.1 Vérifier disponibilité voitures

– Fiche descriptive

Titre	Vérifier disponibilité voitures.
But	Vérifier la disponibilité des voitures commandées pour les affecter.
Résumé	Permet à l'administrateur central de vérifier la disponibilité des voitures pour les affecter à un représentant.
Acteur	Administrateur central.

TABLE 3.19 – Sommaire d'identification(Vérifier disponibilité voitures)

Pré conditions	**-**L'administrateur central est authentifié.
Post condition	**-**Permet l'affectation ou non de voitures à un représentant.
Scénario nominal	**-** Ce cas d'utilisation commence lorsque l'administrateur central veut vérifier la disponibilité des voitures demandées et choisit pour cela l'option «vérifier disponibilité voitures» : **.** Le système affiche une liste des commandes non affectées. **.** L'administrateur choisit une commande. **.** Le système vérifie la disponibilité des voitures demandées. **.** Le système affiche un message « commande disponible ».
Scénario alternatif	**-** Si l'administrateur central annule la vérification : **.** Le système central n'interroge pas la BDD centrale.
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à vérifier la disponibilité : **.** Le système affiche un message d'erreur « il y a un problème dans le serveur d'application central ou le serveur de la BDD central»

TABLE 3.20 – Description détaillée(Vérifier disponibilité voitures)

B.2 Affecter voitures

– Fiche descriptive

Titre	Affecter voitures.
But	Affecter les voitures demandées a un représentant.
Résumé	Permet à l'administrateur central d'envoyer la liste des voitures demandées au représentant concerné.
Acteur	Administrateur central.

TABLE 3.21 – Sommaire d'identification(Affecter voitures)

Pré conditions	**-**L'administrateur central vérifie la disponibilité des voitures demandées.
Post condition	**-**La liste des voitures est envoyée de la BDD centrale à la BDD locale du représentant.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur veut affecter des voitures et choisit pour cela l'option «affecter voitures» : **.** Le système affiche une liste des commandes qui peuvent être satisfaites. **.** L'administrateur central choisit une commande. **.** Le système envoie les voitures demandées à la BDD du représentant concerné. **.** Le système affiche un message « commande envoyée ».
Scénario alternatif	**-** Si l'administrateur central annule l'affectation : **.**Le système central n'invoque pas le système du représentant.
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à affecter les voitures : **.** le système affiche un message d'erreur « il y a un problème soit au niveau du serveur d'application local/distant, soit au niveau du réseau, soit au niveau du serveur de la BDD local/distant».

TABLE 3.22 – Description détaillée(Affecter voitures)

B.3 Rechercher commande

– Fiche descriptive

Titre	Rechercher commande.
But	Recherche une commande dans la base de données.
Résumé	Permet à l'administrateur de trouver une commande stockée dans la base de données pour la consulter.
Acteur	Administrateur central.

Table 3.23 – Description détaillée(Rechercher commande)

Pré conditions	**-**L'administrateur central est authentifié.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur central veut retrouver une commande et choisit pour cela l'option «rechercher commande» : **.**L'administrateur central saisie les arguments de recherche. **.**Le système effectue la recherche et retourne le résultat.
Scénario alternatif	**-**Lorsque la commande n'est pas retrouvée : **.**Le système central affiche un message « aucun résultat disponible».
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à effectuer cette recherche : **.**Le système affiche un message d'erreur « il y a un problème dans le serveur d'application central ou le serveur de la BDD central»

Table 3.24 – Description détaillée(Rechercher commande)

B.3 Consulter

– Fiche descriptive

Titre	Consulter.
But	Consulter une commande dans la base de données.
Résumé	Permet à l'administrateur central de consulter une commande stockée dans la base de données.
Acteur	Administrateur central.

Table 3.25 – Sommaire d'identification(Consulter)

Pré conditions	**-**L'administrateur central effectue une recherche sur la commande concernée.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur central veut consulter une commande et choisit pour cela l'option «Consulter» : **.**Le système affiche les informations de la commande recherchée. **.**L'administrateur central consulte les informations affichées.
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à afficher le informations de la commande désirée : **.**Le système affiche un message d'erreur « il y a un problème dans le serveur d'application central ou le serveur de la BDD central»

Table 3.26 – Description détaillée(Consulter)

B.4 Ajouter représentant

– Fiche descriptive

Titre	Ajouter représentant.
But	Créer un nouveau représentant.
Résumé	Ajouter un nouveau représentant dans la liste des représentants du système.
Acteur	Administrateur central.

TABLE 3.27 – Sommaire d'identification(Ajouter représentant)

Pré conditions	**-**L'administrateur central est authentifié.
Post conditions	**-**Le représentant est ajouté à la liste des représentants du system.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur veut faire la gestion des représentant et choisit pour cela l'option «Ajouter représentant ». **.**Le système affiche un formulaire de création. **.**L'administrateur remplie le formulaire et valide les informations saisies. **.**Le système lit les informations saisies et les vérifie. **.**Le système affiche un message « ajout réussi ».
Scénario alternatif	**-**Si le nom du représentant ajouté existe déjà : **.**Le système affiche un message « Ce représentant existe déjà ». **-**Si l'un des champs est vide : **.**Le système affiche un message « Il faut remplir tous les champs »
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à ajouter le représentant dans la base de données : **.**Le système affiche un message d'erreur « il y a un problème dans le serveur d'application central ou le serveur de la BDD central»

TABLE 3.28 – Description détaillée(Ajouter représentant)

B.5 Modifier représentant

– Fiche descriptive

Titre	Modifier représentant.
But	Modifier les informations d'un représentant.
Résumé	Modifier les informations d'un représentant.
Acteur	Administrateur central.

Table 3.29 – Sommaire d'identification(Modifier représentant)

Pré conditions	**-**L'administrateur central est authentifié.
Post conditions	**-**Les informations du représentant sont modifiées.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur veut faire la gestion des représentant et choisit pour cela l'option «Modifier représentant ». **.**Le système affiche la liste des représentants. **.**L'administrateur central choisit un représentant. **.**Le système affiche un formulaire qui contient tous les informations de représentant choisi. **.**L'administrateur central change les champs qu'il veut dans le formulaire et valide ces changements. **.**Le système lit les nouvelles informations saisies et les vérifie. **.**Le système affiche un message « modification réussi».
Scénario alternatif	**-**Si le nouveau nom du représentant saisi existe déjà : **.**Le système affiche dans un message « ce nom de représentant existe déjà ».
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à modifier le représentant dans la base des données : **.**Le système affiche un message d'erreur « il y a un problème dans le serveur d'application central ou le serveur de la BDD central»

Table 3.30 – Description détaillée(Modifier représentant)

B.6 Envoyer représentant

– Fiche descriptive

Titre	Envoyer représentant.
But	Envoyer représentant.
Résumé	Permet à l'administrateur central d'envoyer la liste des coopérants à ses représentants.
Acteur	Administrateur central.

TABLE 3.31 – Sommaire d'identification(Envoyer représentant)

Pré conditions	**-**L'administrateur central est authentifié.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur central veut envoyée la liste des coopérants à ses représentants et choisit pour cela l'option «Envoyer représentant» : **.**Le système affiche la liste des représentants. **.**L'administrateur choisit l'envoi ou pas.
Scénario exceptionnel	**-**Lorsque le système ne parvient pas à envoyer la liste des représentants : **.**Le système affiche un message d'erreur « il y a un problème soit au niveau du serveur d'application local/distant, soit au niveau du réseau, soit au niveau du serveur de la BDD local/distant».

TABLE 3.32 – Description détaillée(Envoyer représentant)

B.7 **Statistiques pannes**

– Fiche descriptive

Titre	Statistiques pannes.
But	Calculer les statistiques des pannes.
Résumé	Permet à l'administrateur central d'avoir les statistiques des pannes des voitures.
Acteur	Administrateur central.

TABLE 3.33 – Sommaire d'identification(Statistiques pannes)

Pré conditions	**-**L'administrateur central est authentifié.
Scénario nominal	**-**Ce cas d'utilisation commence lorsque l'administrateur central veut calculer les statistiques des pannes et choisit pour cela l'option «statistiques pannes» : **.**L'administrateur saisit le modele qu'il veut savoir leur statistiques. **.**Le système central interroge les systèmes des différents représentant pour calculer les statistiques et renvoyer le résultat. **.**Le système central affiche les résultats. **.**L'administrateur central consulte les résultats affichées.
Scénario exceptionnel	**-** Lorsque le système ne parvient pas à effectuer les calcules : **.**Le système affiche un message d'erreur « il y a un problème soit au niveau du serveur d'application local/distant, soit au niveau du réseau, soit au niveau du serveur de la BDD local/distant».

TABLE 3.34 – Description détaillée(Statistiques pannes)

3.3 Conclusion

Durant cette phase nous avons recueilli toutes les informations d'aspect fonctionnel afin de pouvoir fixer les principales fonctionnalités que doit disposer notre futur système. À présent, on est prêt à passer à la phase d'analyse et de conception.

Phase 02: Analyse et conception

Introduction

Dans cette phase nous allons identifier et réaliser le diagramme de classe ainsi que les diagrammes d'interaction de notre application.

3.4 Diagramme de classes

3.4.1 Définition

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation car il montre la structure interne et permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation.[9]

Les principaux éléments de cette vue statique sont les classes et leurs relations telles que association (entre deux classe défirent), agrégation, composition et plusieurs types de dépendances, tel que la réalisation.

Maintenant nous allons représenter notre diagramme de classe avec les règles suivant :

» **Au niveau de l'agence centrale :**

- Un représentant peut demander une ou plusieurs commandes, et une commande concerne un et un seul représentant.
- Une commande contient une ou plusieurs voitures, et la voiture est liée à une et une seul commande.

» **Au niveau du représentant :**

- Une commande contient une ou plusieurs voitures, et la voiture est liée à une et une seul commande.
- Une voiture peut posséder un ou plusieurs pannes ou pas, et une panne concerne une et une seule voiture.
- Un client peut acheter une ou plusieurs voitures, et une voiture est achetée par un et un seul client.

3.4.2 Diagramme de classe global

Maintenant nous représentons notre diagramme de classe global qui permet de décrire les classes de l'application comme étant un seul système avec une base de données centralisée.

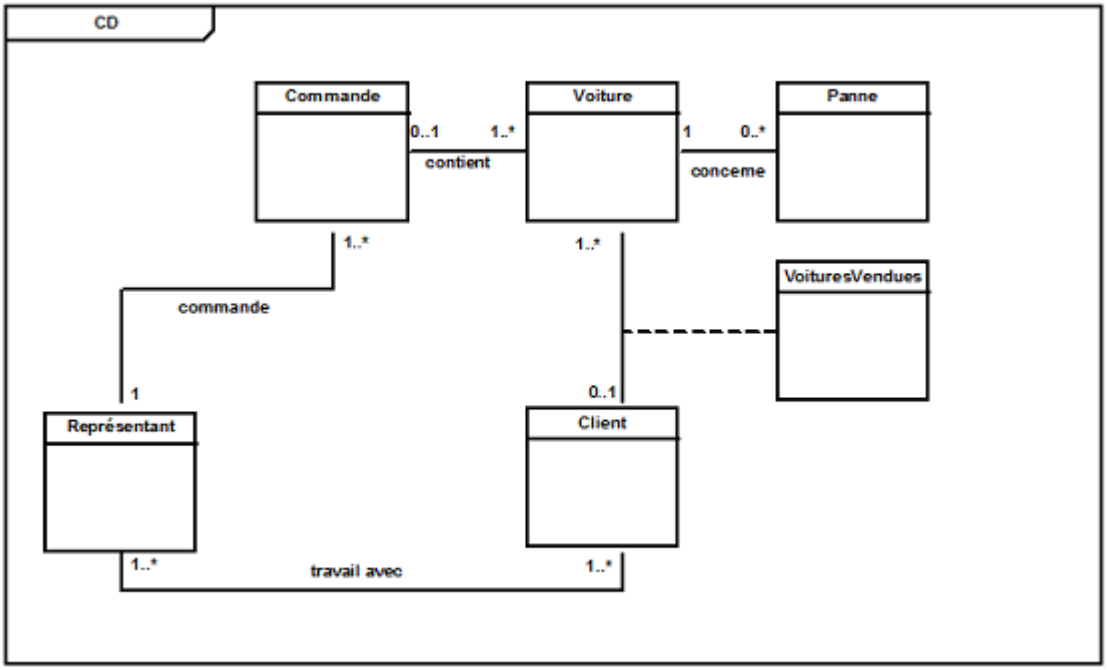

FIGURE 3.2 – Diagramme de classe global

3.4.3 Diagramme de classe détaillé

A/ Classes et attributs

– **Au niveau de l'agence central :**

Classe	Code	Description	Type
Représentant	nomRep	Nom du représentant	Chaine de caractères
	adresseRep	Adresse IP du représentant	Chaine de caractères
CommandeCentral	numCommande	Numéro commande	Entier
	nomRep	Nom du représentant	Chaine de caractères
	dateReception	Date de réception	Date
	dateAffectation	Date d'affectation	Date
	etatCommande	Etat de la commande	Chaine de caractères
	modele	Modèle de la voiture	Chaine de caractères
	quantite	Quantité de voitures	Entier
VoitureCentral	numChassis	Numéro de chassis	Entier
	modele	Modèle de la voiture	Chaine de caractères
	couleur	Couleur de la voiture	Chaine de caractères
	puissance	Puissance de la voiture	Chaine de caractères
	placesAssises	Places assises de la voiture	Entier
	garantieParKM	Garantie par kilomètre	Chaine de caractères
	garantieParAnnee	Garantie par année	Chaine de caractères
	prix	Prix de voiture	réelle
	etatVoiture	Etat de la voiture	Chaine de caractères

TABLE 3.35 – Classes et attributs(Au niveau de l'agence centrale)

– **Au niveau du représentant :**

Classe	Code	Description	Type
CommandeLocal	numCommande	Numéro commande	Entier
	dateCreation	Date de creation	Date
	dateAffectation	Date d'affectation	Date
	etatCommande	Etat de la commande	Chaine de caractères
	modele	Modèle de la voiture	Chaine de caractère
	quantite	Quantité de voitures	Entier
VoitureLocal	numChassis	Numéro de chassis	Entier
	modele	Modèle de la voiture	Chaine de caractères
	couleur	Couleur du voiture	Chaine de caractère
	puissance	Puissance de la voiture	Chaine de caractères
	placesAssises	Places assises du voiture	Entier
	garantieParKM	Garantie par kilomètre	Chaine de caractères
	garantieParAnnee	Garantie par année	Chaine de caractères
	prix	Prix de voiture	Réelle
	etatVoiture	Etat de la voiture	Chaine de caractères
PanneLocal	numPanne	Numéro de panne	Entier
	numChassis	Numéro de chassis	Entier
	typeDepannage	Type de dépannage	Chaine de caractères
	DesceiptionPanne	Description du panne	Chaine de caractères
	dateDep	Date de dépannage	Date
Client	numClient	Numéro du client	Entier
	nomClient	Nom du client	Chaine de caractères
	prenomClient	Prénom du client	Chaine de caractères
	adresse	Adresse du client	Chaine de caractères
	numTel	Numéro téléphone	Entier
	email	Email	Chaine de caractères

VoituresVendues	garantieChoisit	La garantie choisit	Chaine de caractères
	dateAchat	Date achat	Date

TABLE 3.36 – Classes et attributs(Au niveau du représentant)

Le diagramme de classe détaillé permet de spécifier la distribution de notre système (système central et système local), Nous utilisons ici une fragmentation horizontale pour fragmenter les tables commande et Voiture.
La figure décrit les deux systèmes :

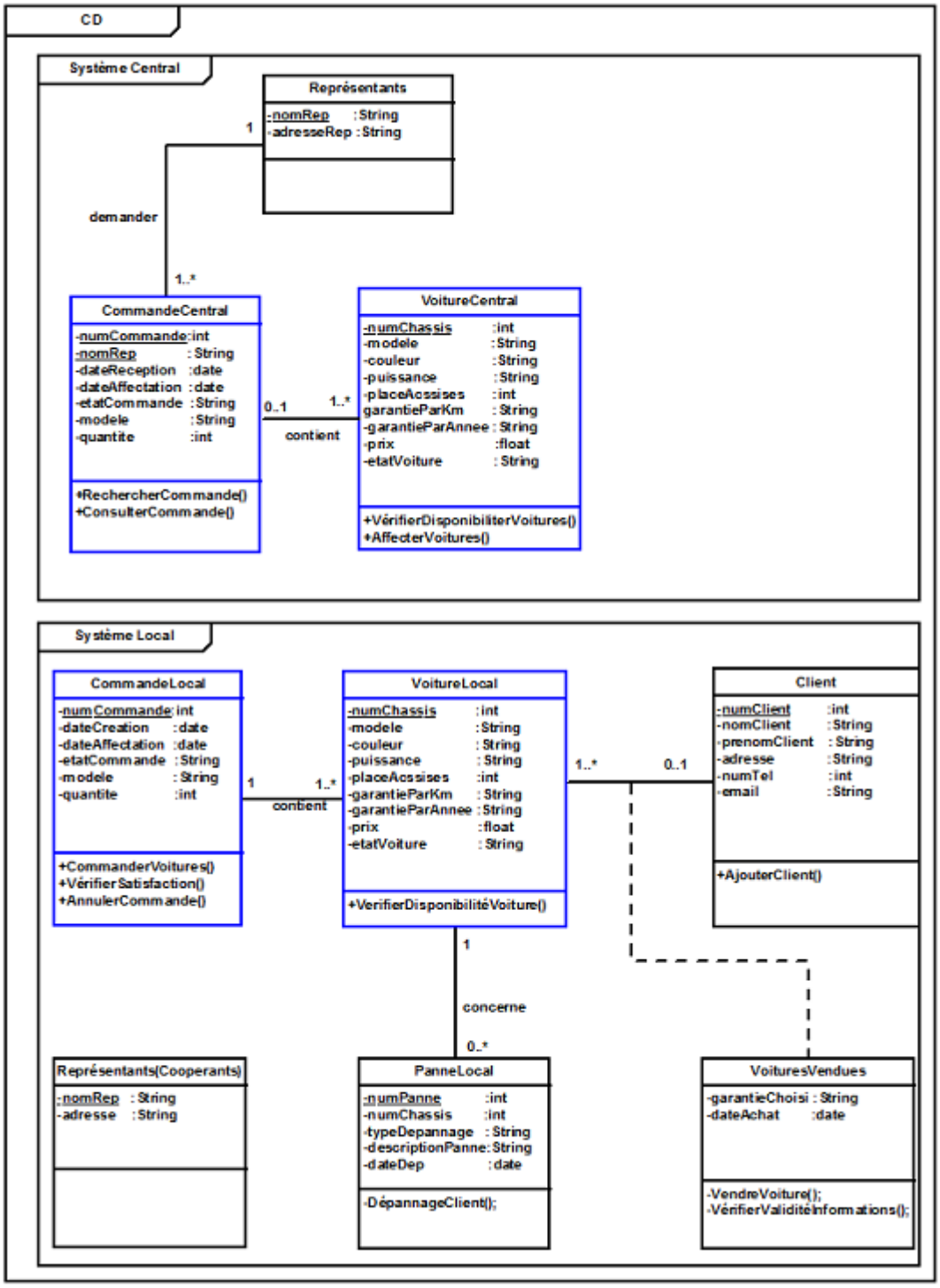

FIGURE 3.3 – Diagramme de classe détaillé

3.5 Diagramme d'interaction

Les diagrammes d'interaction permettent d'établir un lien entre les diagrammes de cas d'utilisation et les diagrammes de classes : ils montrent comment des objets communiquent pour réaliser une certaine fonctionnalité. ils apportent ainsi un aspect dynamique à la modélisation du système.

Dans les diagrammes d'interaction, les objets communiquent en s'envoyant des messages qui invoquent des opérations sur les objets récepteurs. Il est ainsi possible de suivre visuellement les interactions dynamiques entre objets, et les traitements réalisés par chacun d'eux.

Par rapport aux diagrammes de séquences système, nous remplaçons ici le système, vu comme une boîte noire, par un ensemble d'objets en collaboration. Ces objets sont des instances des trois types de classes d'analyse du diagramme de classes participantes, à savoir des dialogues, des contrôles et des entités. [9]

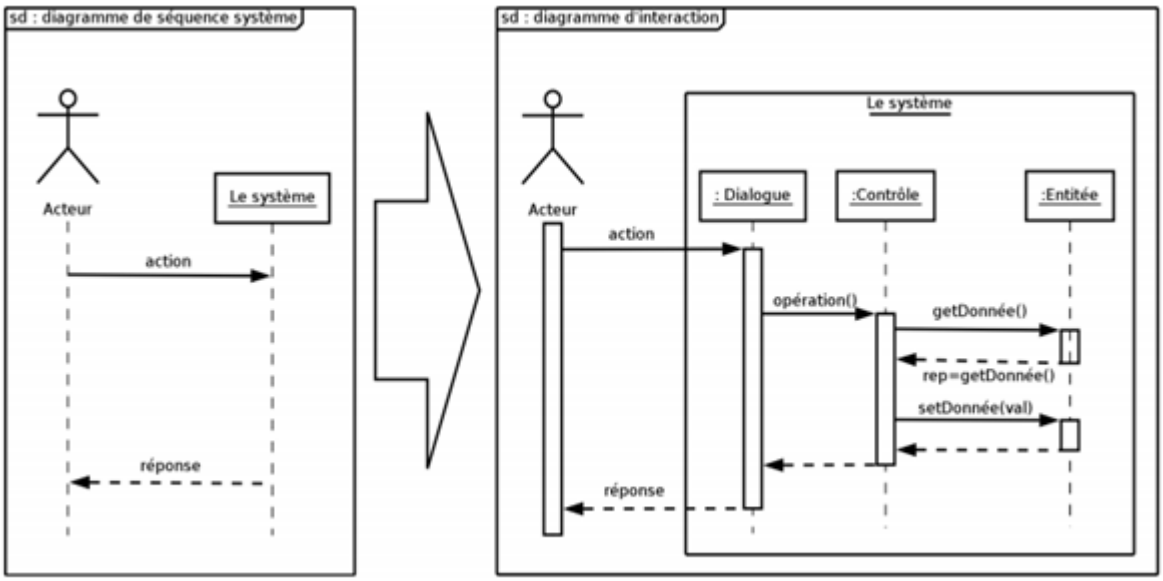

FIGURE 3.4 – Exemple de diagramme d'interaction

» Maintenant nous allons représenté nos diagrammes d'interaction pour les différentes cas d'utilisation au niveau de système de l'agence centrale et de système de représentant.

A. Au niveau du représentant

A.1 S'authentifier

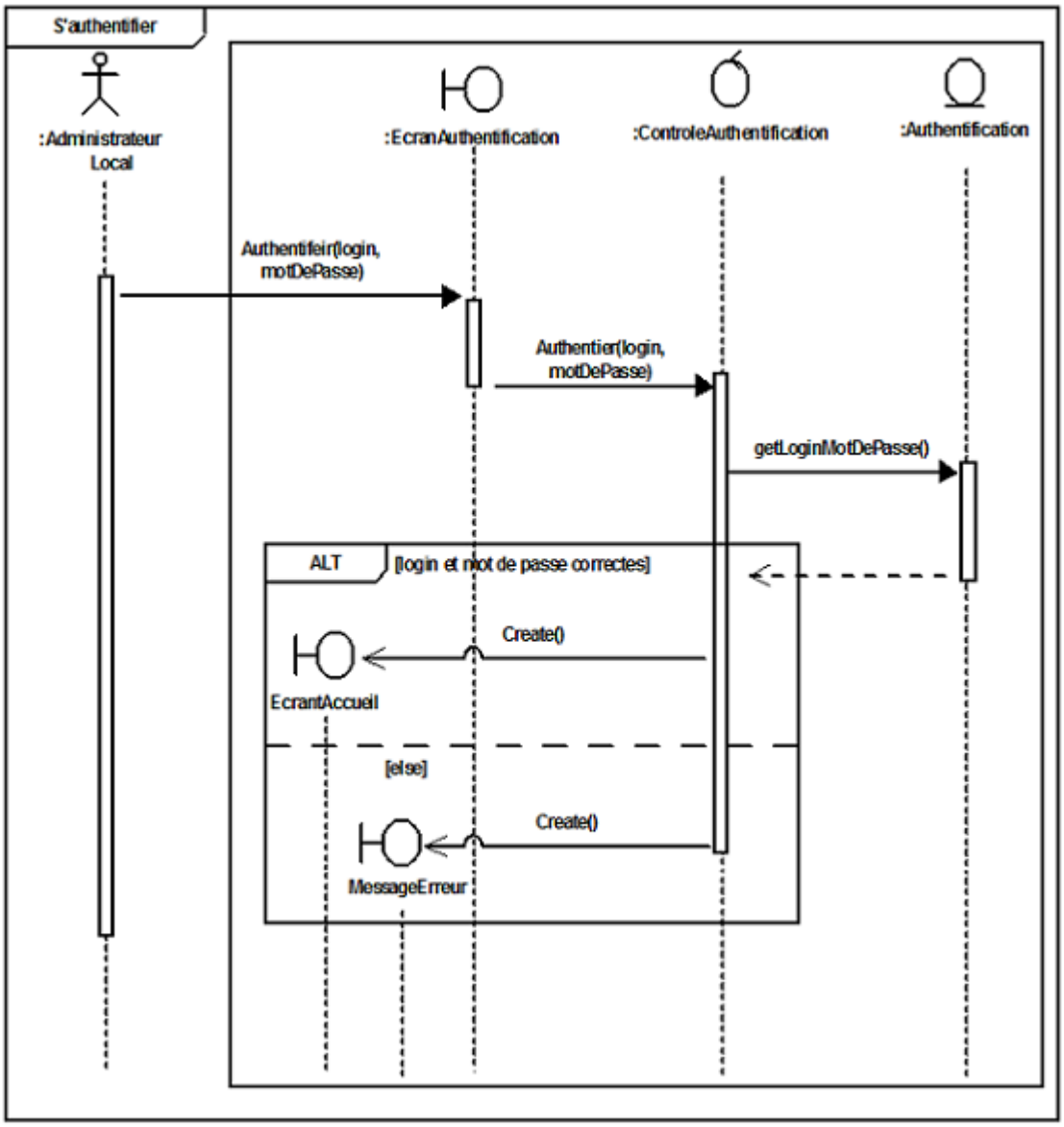

FIGURE 3.5 – S'authentifier

A.2 Commander Voitures

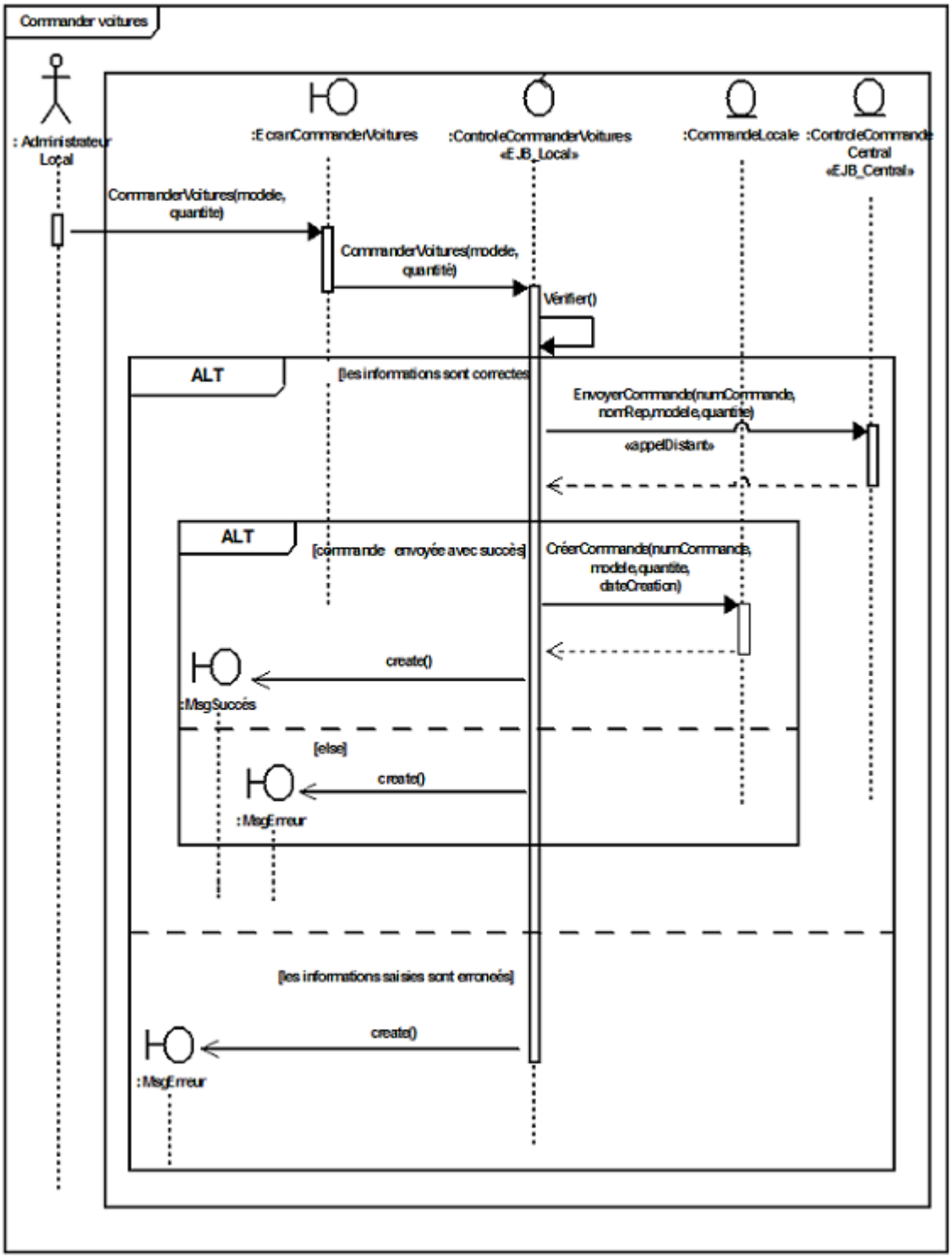

FIGURE 3.6 – Commander voitures

A.3 Annuler commande envoyée

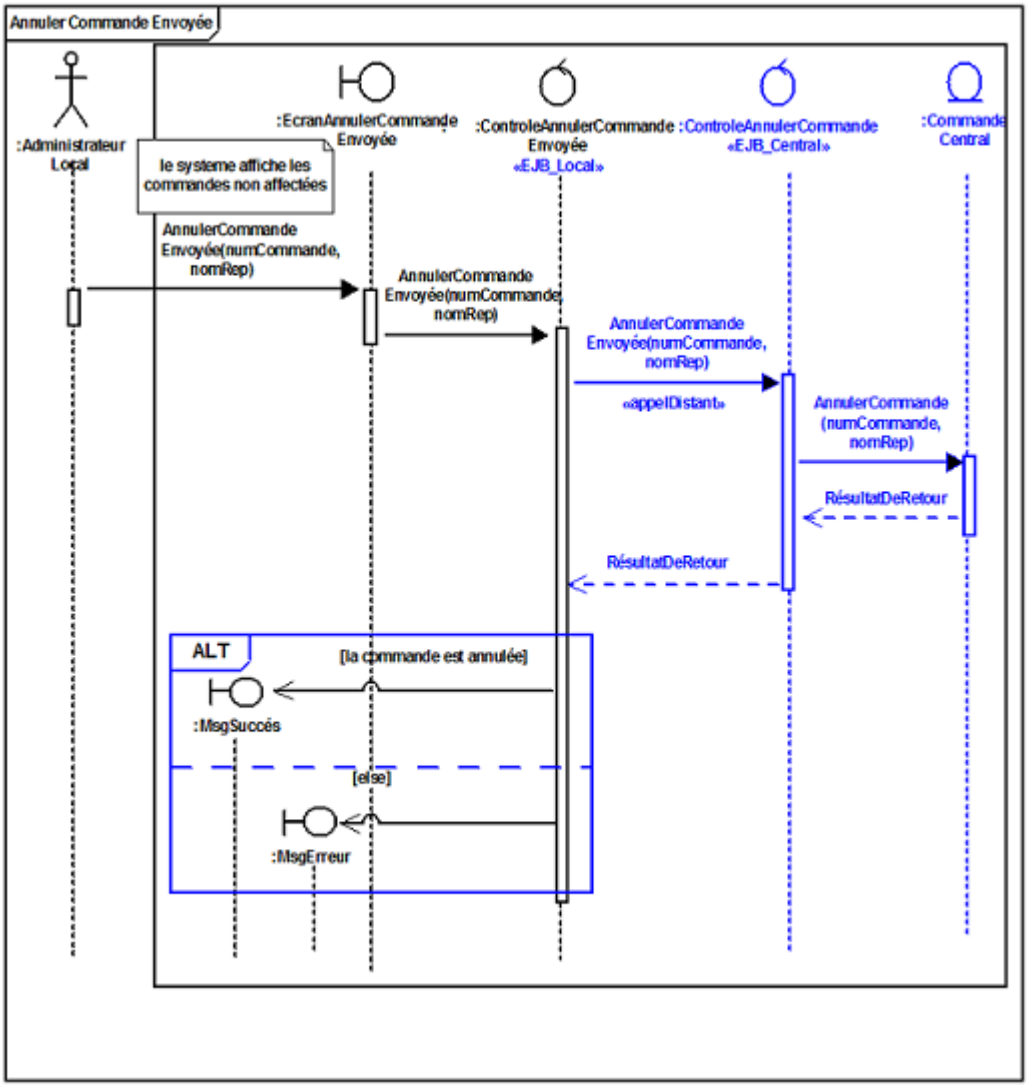

FIGURE 3.7 – Annuler commande envoyée

A.4 Vérifier satisfaction d'une commande envoyée

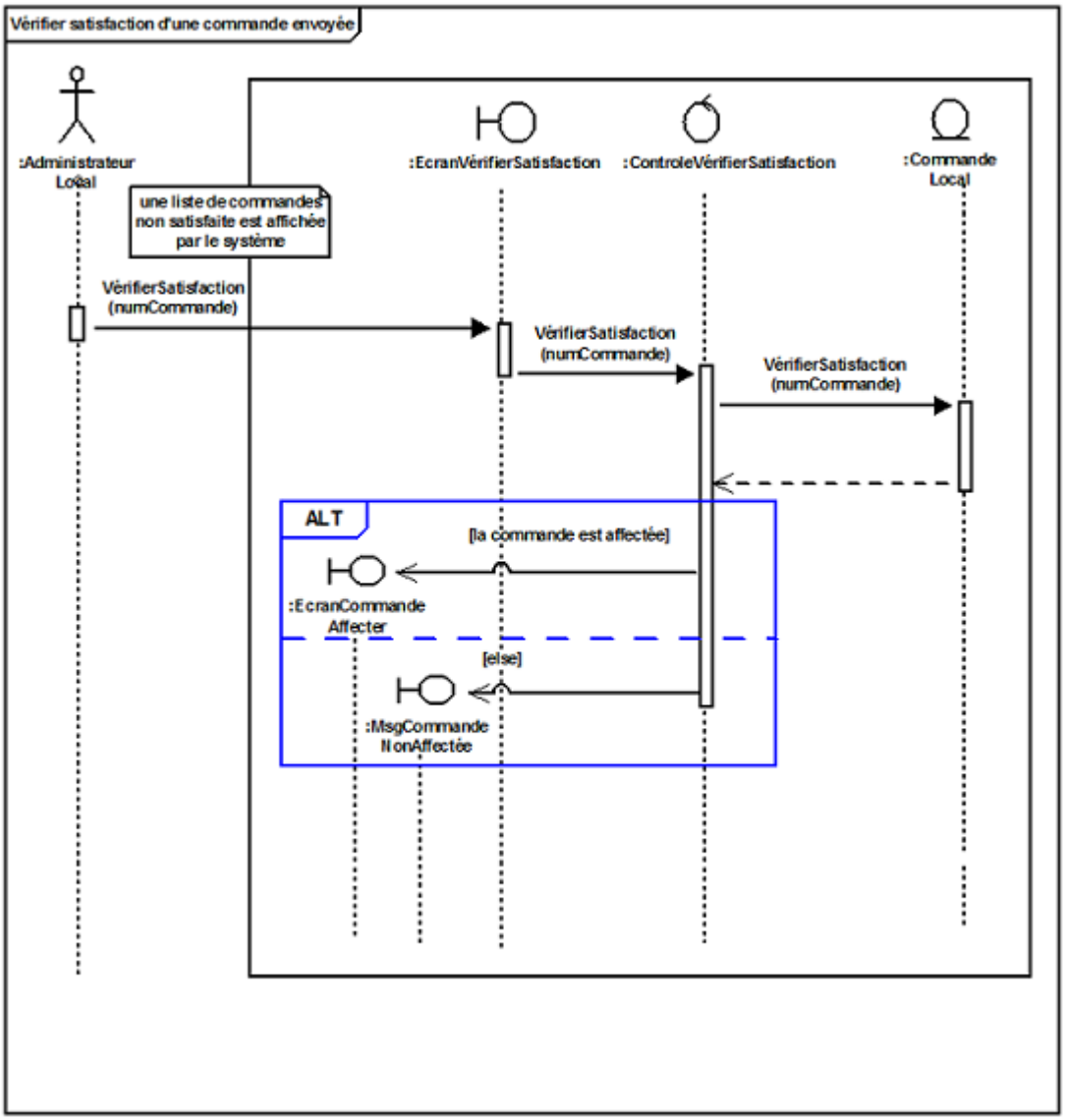

FIGURE 3.8 – Vérifier satisfaction d'une commande envoyée

A.5 Vérifier la disponibilité de la voiture

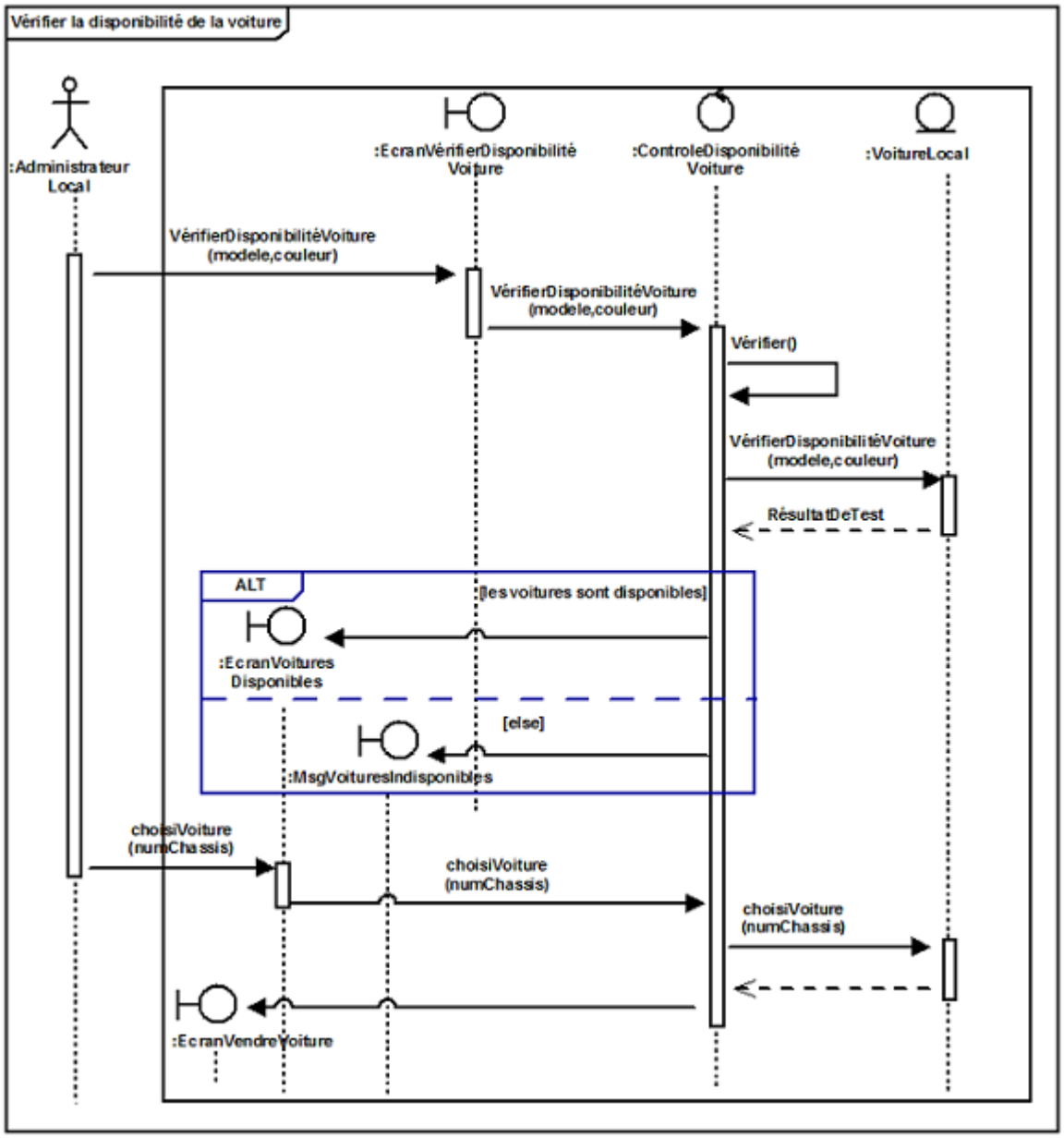

FIGURE 3.9 – Vérifier la disponibilité de la voiture

A.6 Vendre voiture

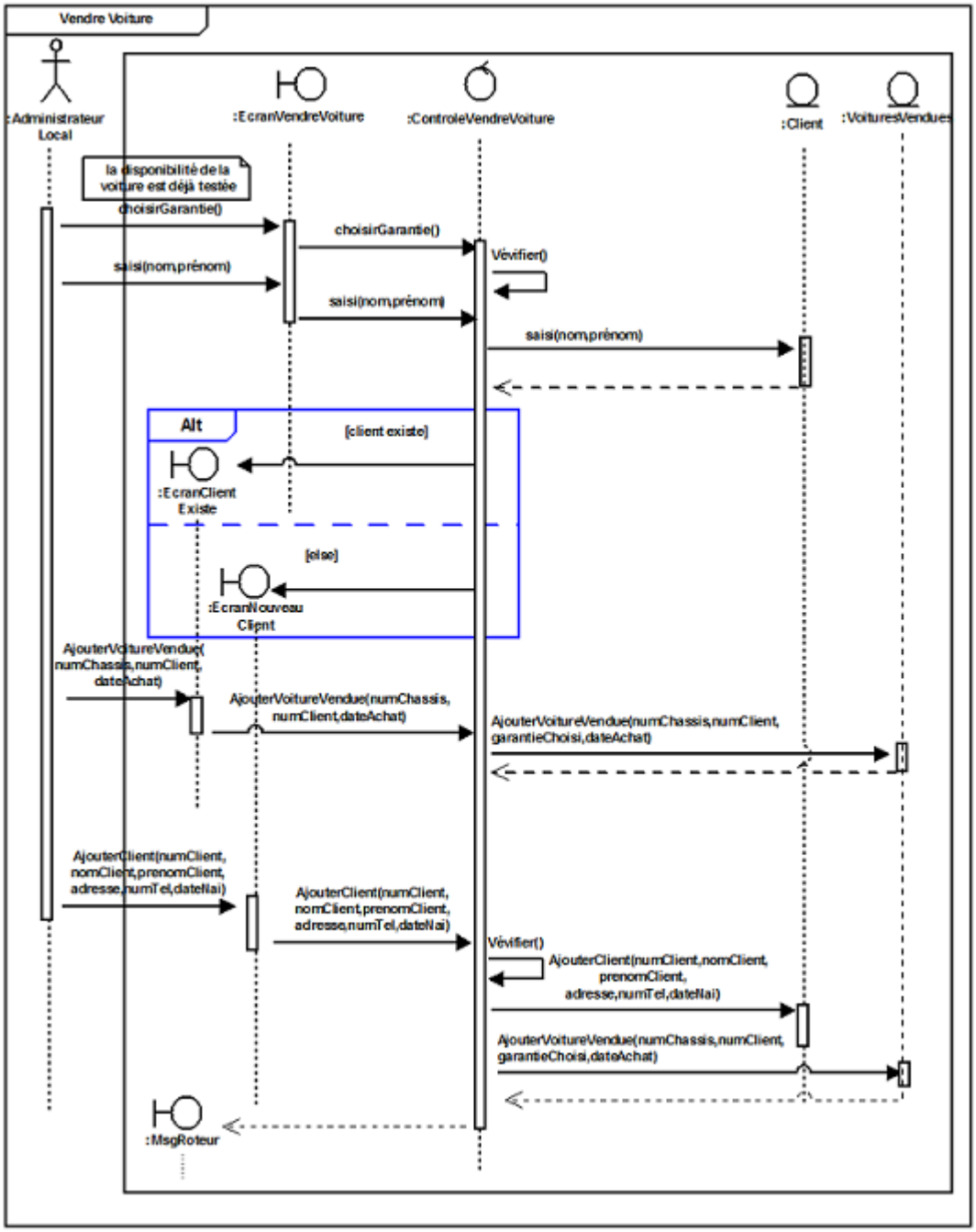

FIGURE 3.10 – Vendre voiture

A.7 Dépannage avec garantie(le cas d'un client distant)

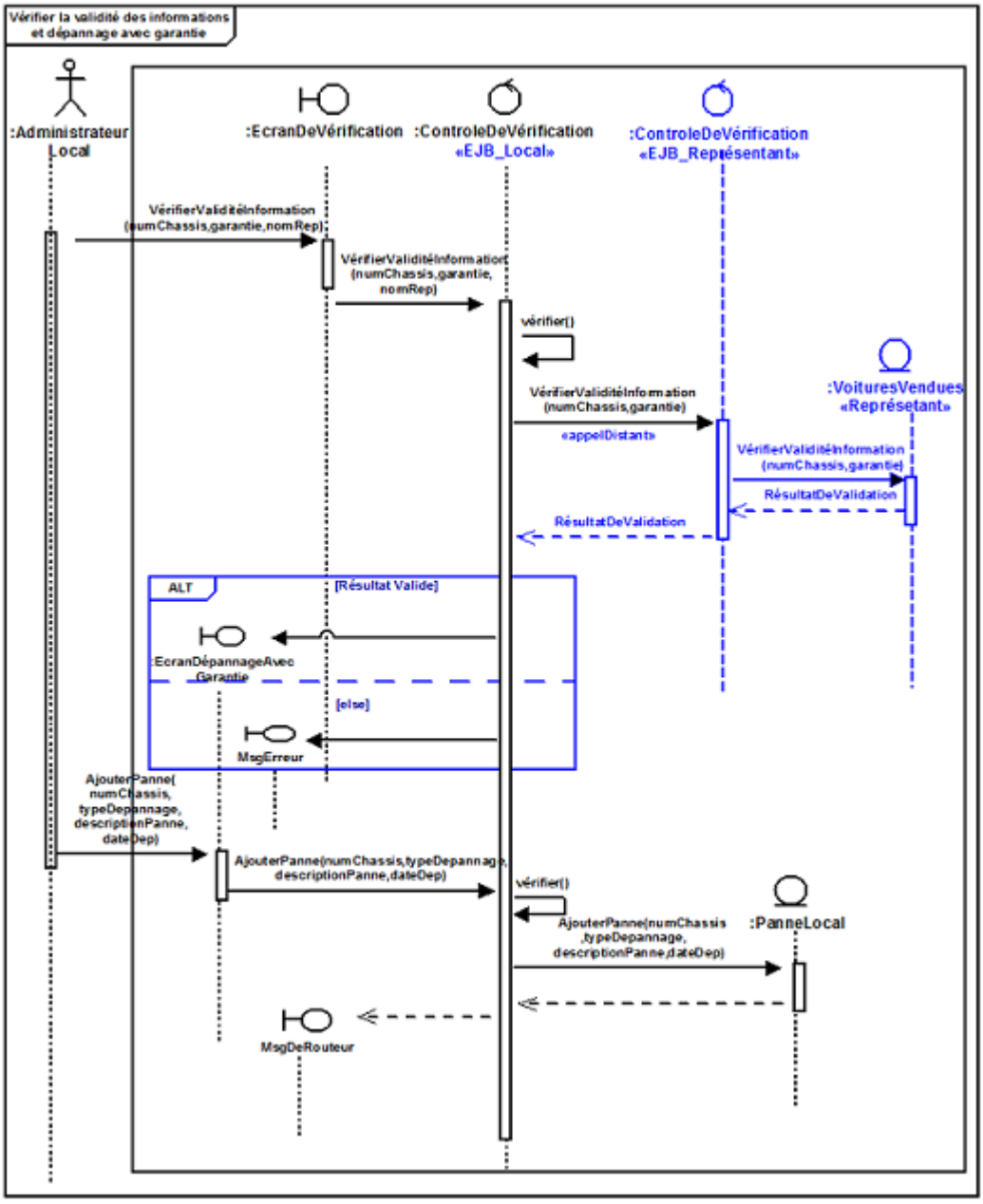

FIGURE 3.11 – Dépannage avec garantie(le cas d'un client distant)

A.8 Dépannage avec garantie(le cas d'un client local)

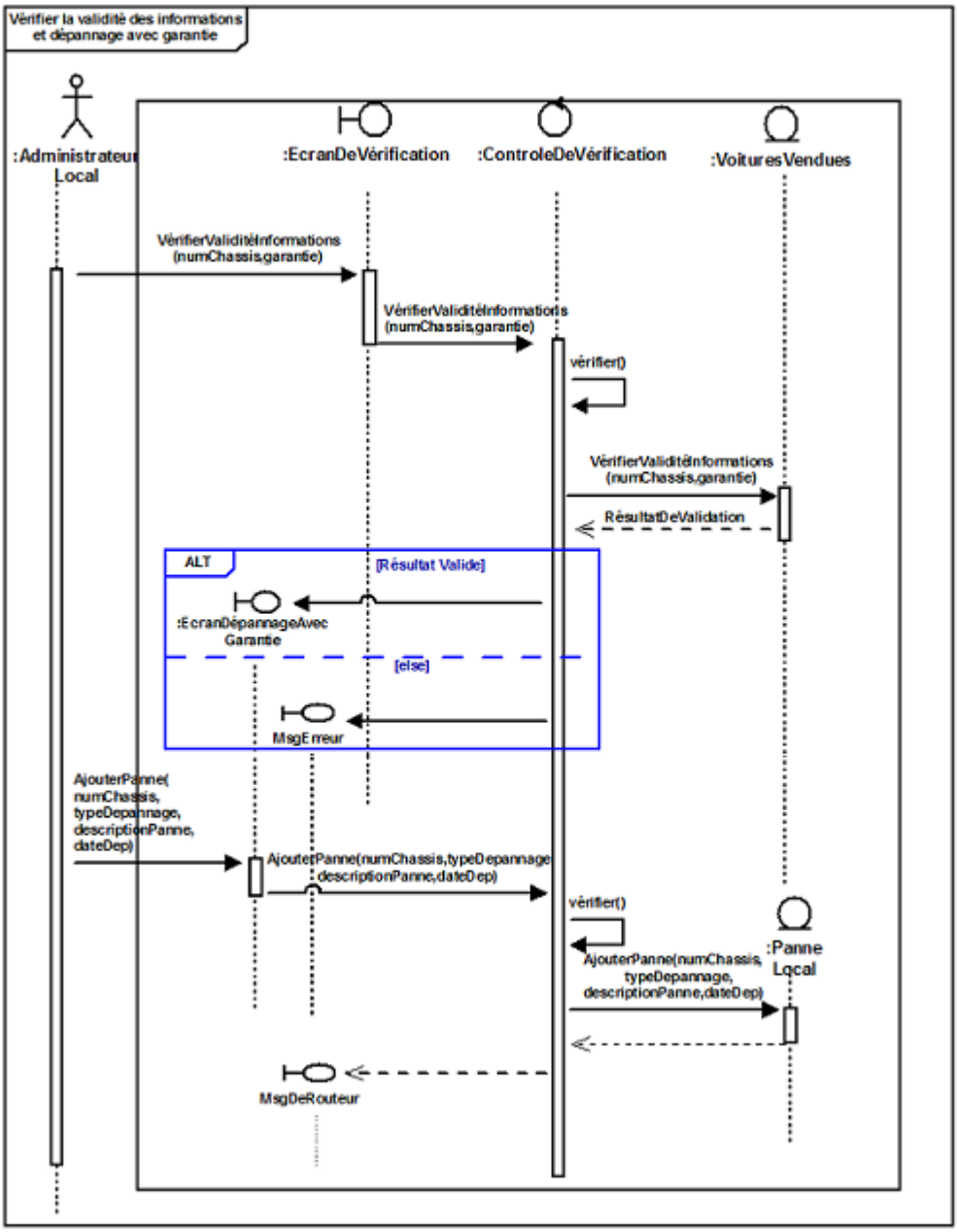

FIGURE 3.12 – Dépannage avec garantie(le cas d'un client local)

A.9 Dépannage sans garantie

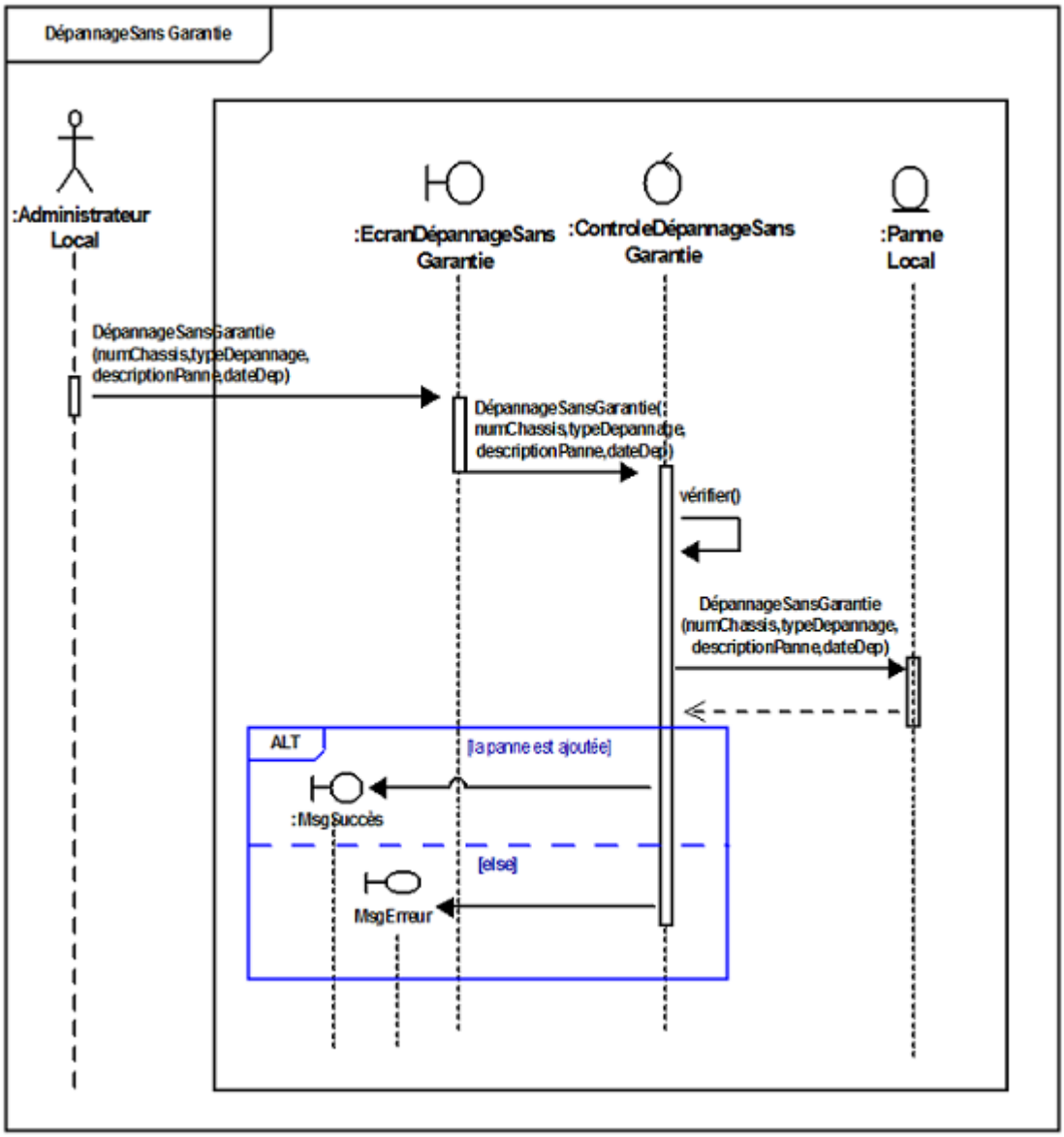

FIGURE 3.13 – Dépannage sans garantie

B. Au niveau de l'agence centrale

B.1 Vérifier disponibilité et affecter voitures

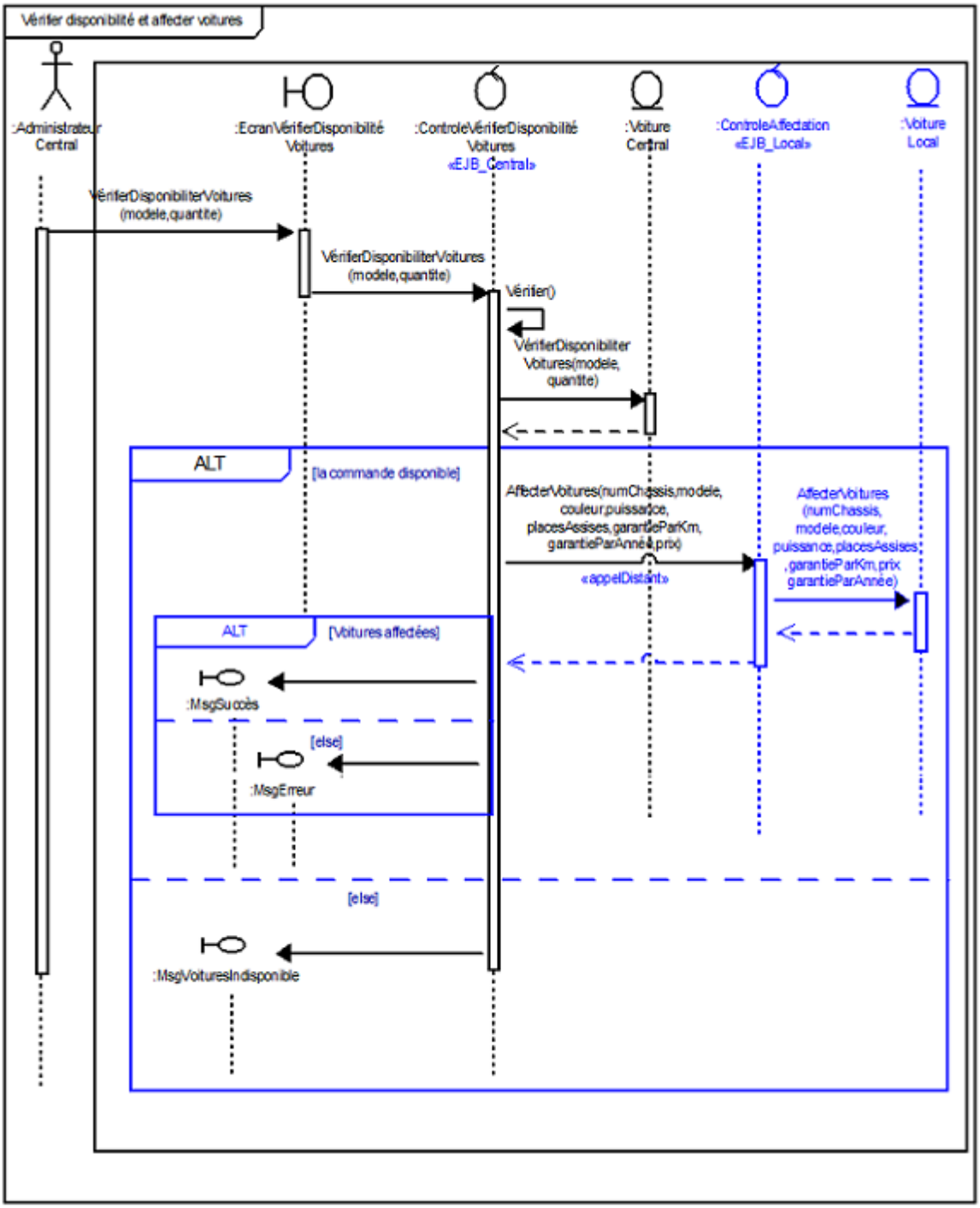

FIGURE 3.14 – Vérifier disponibilité et affecter voitures

B.2 Rechercher commande et consulter

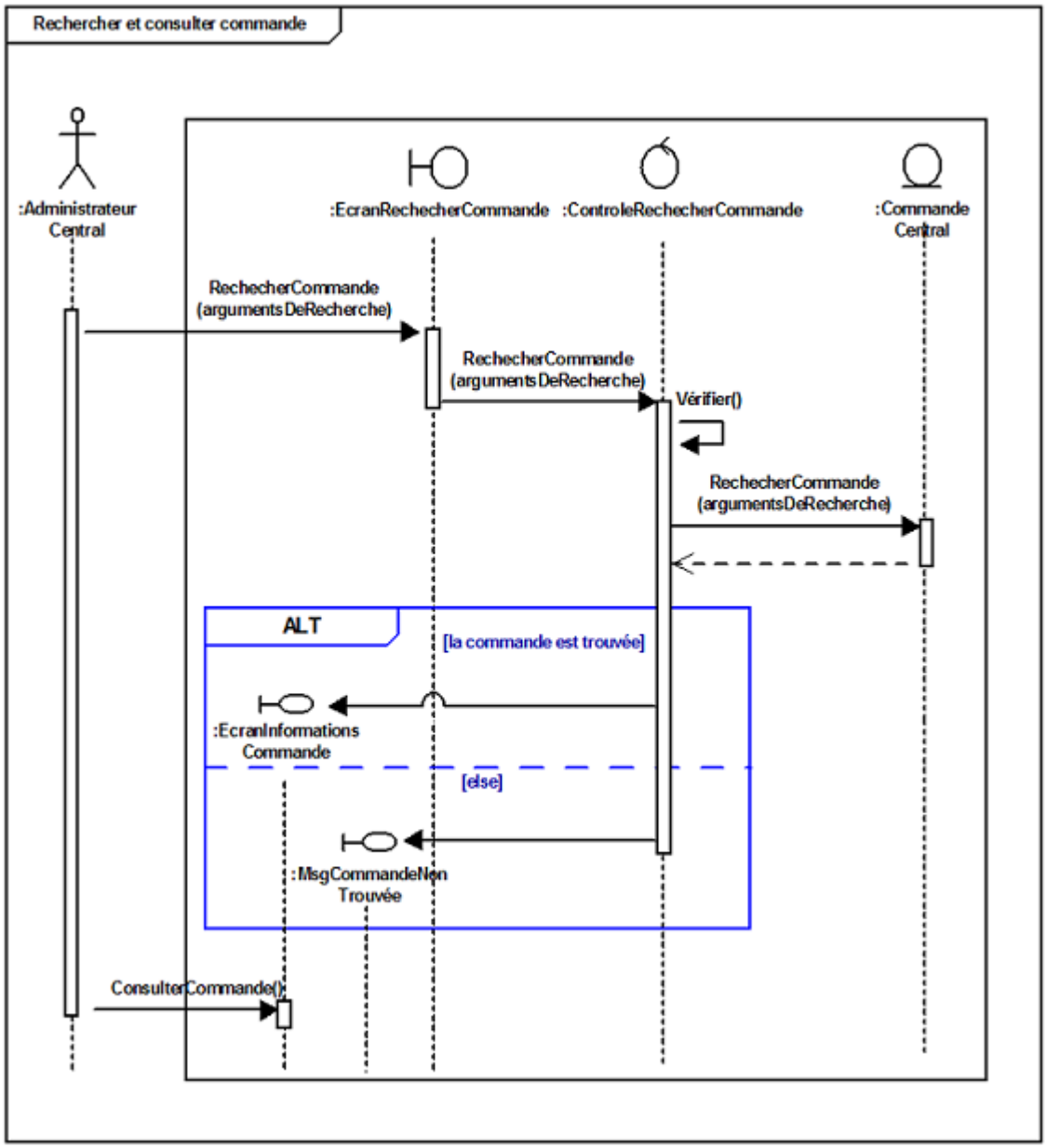

FIGURE 3.15 – Rechercher commande et consulter

B.3 Ajouter représentant

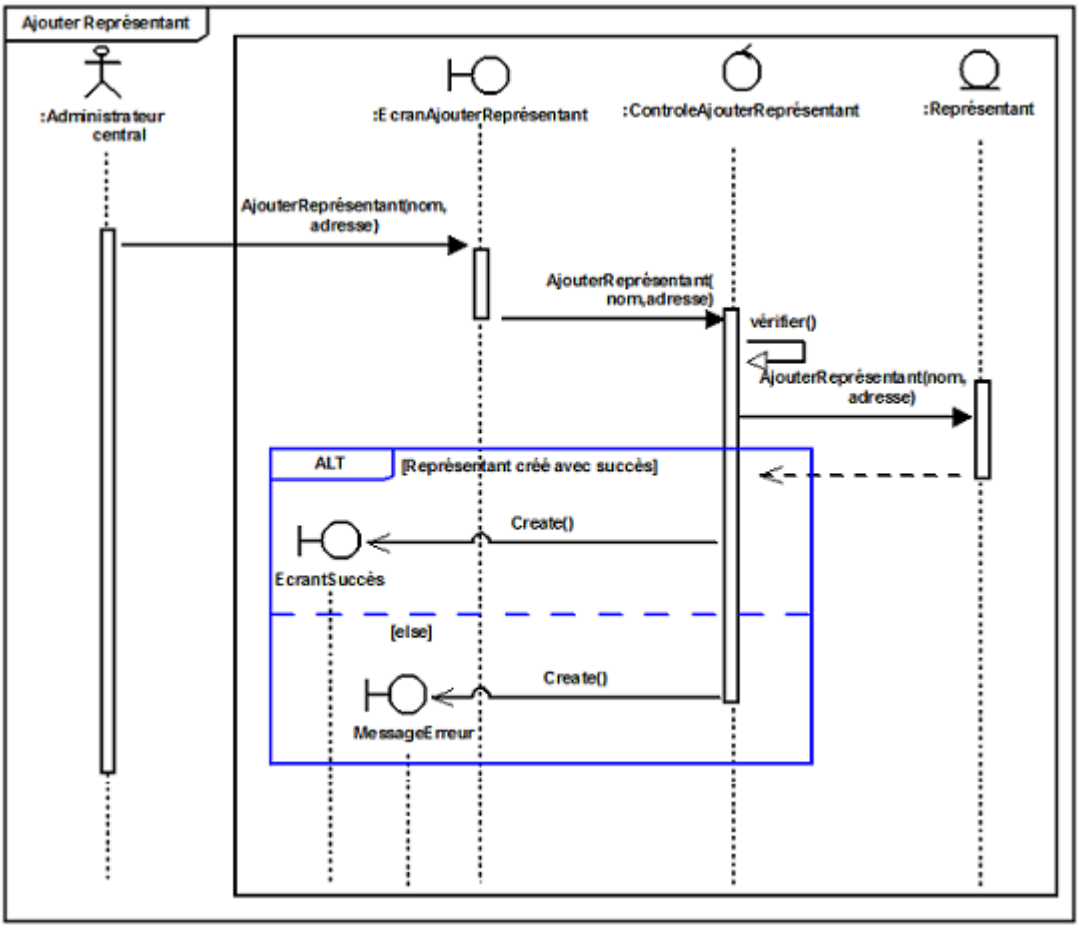

FIGURE 3.16 – Ajouter représentant

B.4 Modifier représentant

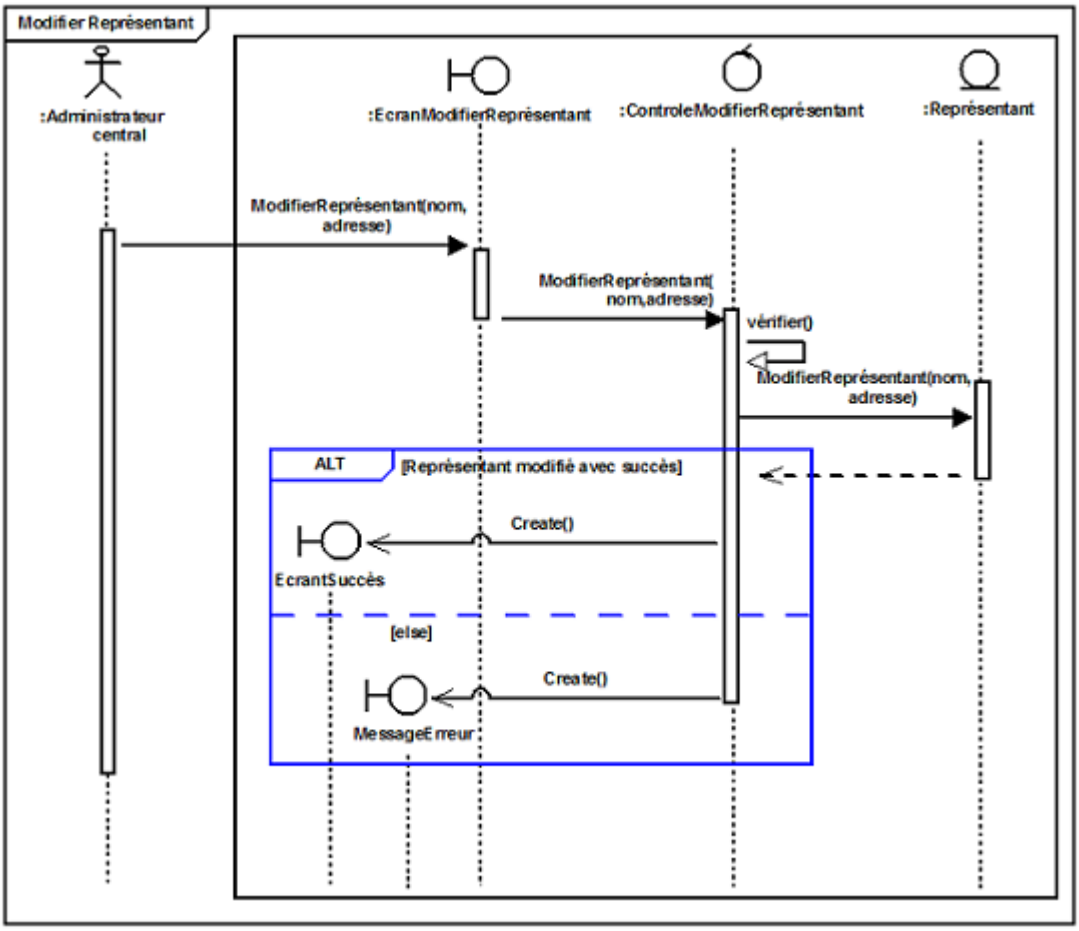

FIGURE 3.17 – Modifier représentant

B.5 Envoyer représentant

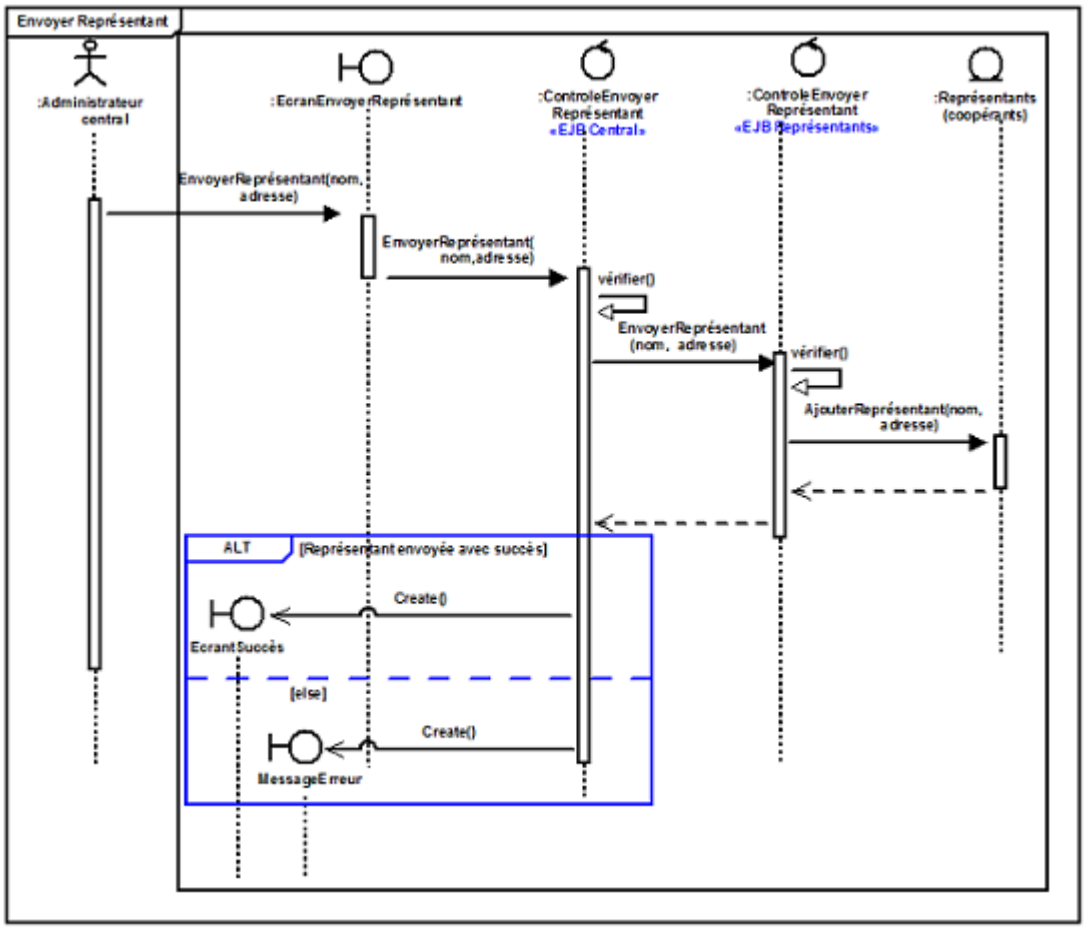

FIGURE 3.18 – Envoyer représentant

B.6 Statistiques pannes

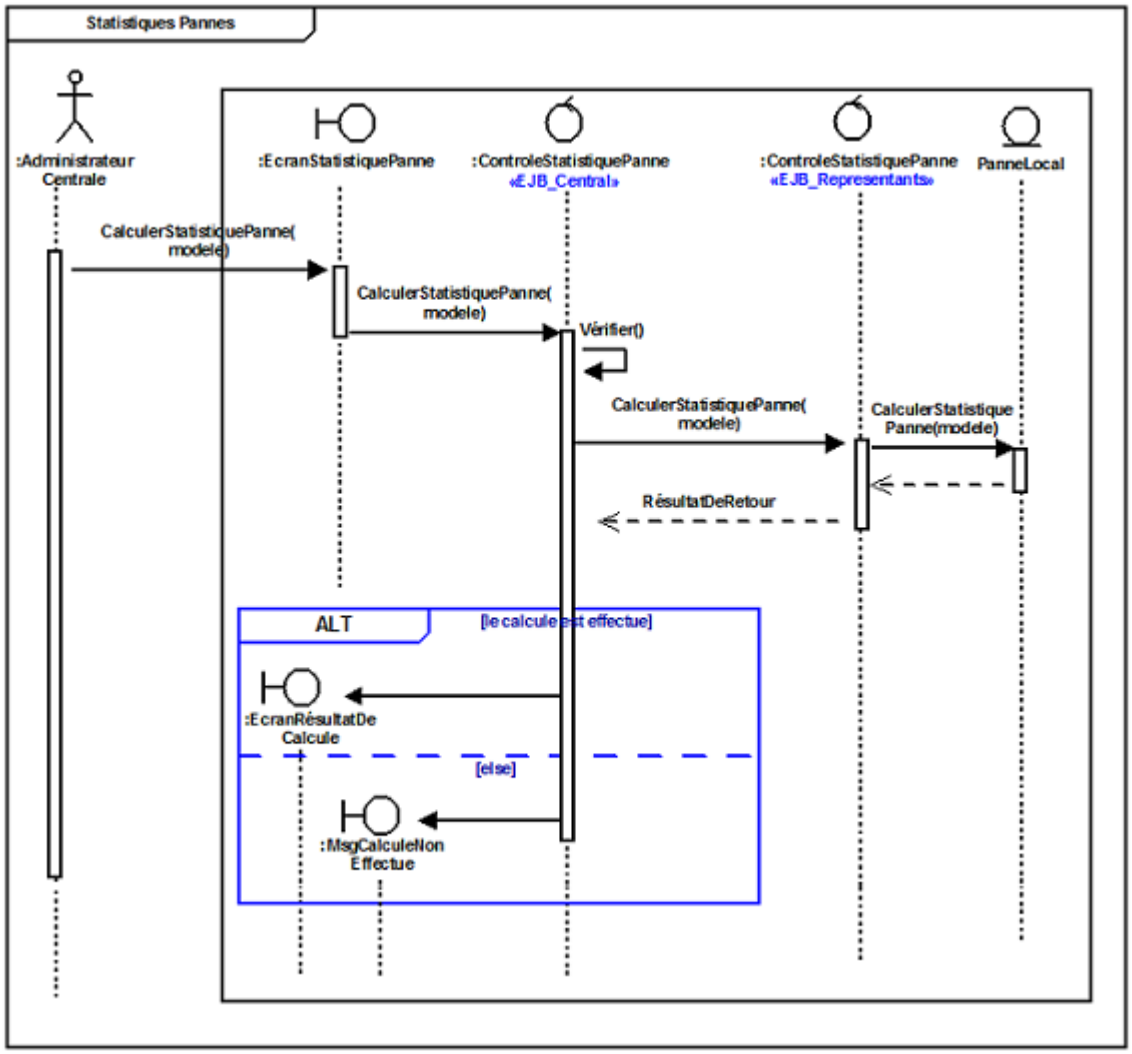

FIGURE 3.19 – Statistiques pannes

3.6 Conclusion

Dans cette phase, nous avons présenté le diagramme de classe, Ainsi que les diagrammes d'interaction qui montrent le fonctionnement du système. Il nous reste qu'une dernière étape qui est l'étape de réalisation qu'on vas la présenter dans le chapitre suivant .

Chapitre 4

Réalisation

- **Introduction.**

- **L'architecture de notre système.**

- **Choix technologique.**

- **Outils et environnements de développement.**

- **Quelques interfaces de notre système.**

- **Conclusion.**

Introduction

Dans ce chapitre nous allons modéliser et décrire l'architecture applicative de notre application, Pour l'architecture technique nous utilisons le diagramme de déploiement. Nous allons aussi présenter les différents technologies et les outils utilisés pour développer notre application.

4.1 L'architecture de notre système

4.1.1 Définition d'une architecture

L'architecture spécifie la structure d'un système. On parle d'architecture fonctionnelle pour définir les services du système, d'architecture technique pour les composants techniques utilisés et d'architecture applicative pour décrire le découpage en sous-systèmes.

4.1.2 L'architecture en trois couches

Le développement en Java EE reposent sur un découpage en couches ou tiers, nous parlons alors d'applications multi-tiers. Trois grandes couches sont représentées :

- **Couche présentation :**C'est la face visible de notre application.
- **Couche métier :** C'est l'ensemble des règles métier de l'application.
- **Couche persistance de données :**Elle permet la manipulation et la conservation des données.

Ces trois couches se conforment à l'architecture de couches fermées «Closed layer architecture» (une couche peut communiquer seulement avec la couche qui lui est adjacente).

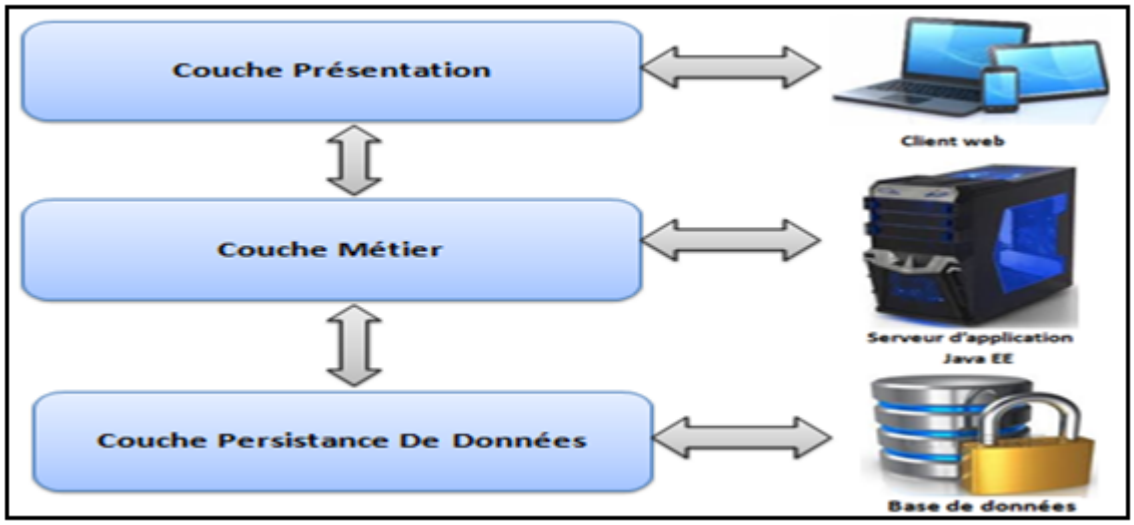

FIGURE 4.1 – L'Architecture java EE standard

4.1.3 Architecture applicative

Le modèle précédent en trois couches peut être affiné pour être plus fidèle à l'architecture finale de notre application. Ci-après un diagramme de paquetage décrivant les couches de l'application :

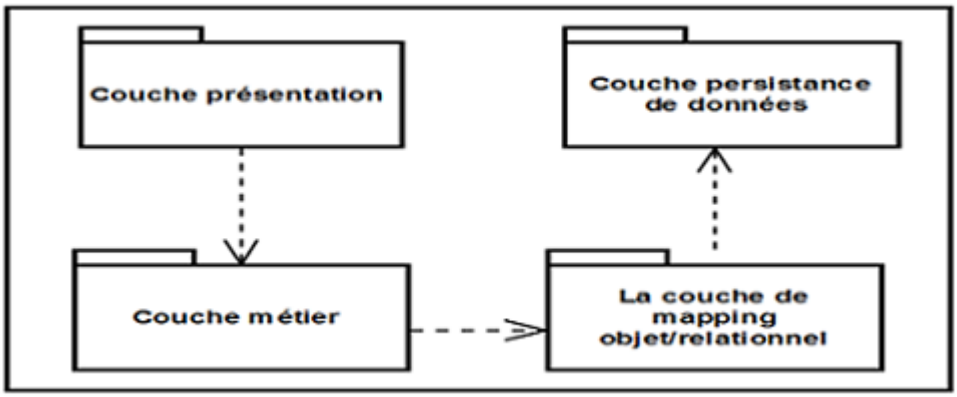

FIGURE 4.2 – Diagramme de paquetage décrivant les couches de l'application

» **Couche de présentation**

La couche de présentation est la partie visible de l'application qui permet à un utilisateur d'interagir avec le système. Elle relaie les requêtes de l'utilisateur à destination de la couche métier, et en retour lui présente les résultats renvoyés par les traitements. La couche présentation peut être exploitée sous plusieurs formes : Client lourd (application graphique), client léger (navigateur web).

Nous utilisons dans notre application un client léger (navigateur web). Pour illustrer ce niveau logique de l'architecture java EE, plusieurs technologies sont utilisées mais ce qui nous intéresse sont :

- Les Servlets : Une servlet est une classe java qui s'exécute côté serveur pour générer dynamiquement des données exploitables par un navigateur web.
- Les Java Server Pages (JSP) : C'est un mélange de données statique (HTML, JavaScript, CSS) et du code java pour générer du contenu dynamique.

» Couche métier

La couche de traitement métier correspond à la partie fonctionnelle ou métier de l'application. Elle implémente la logique et les règles de gestion permettant de répondre aux requêtes de la couche présentation.

Pour fournir ces services, elle s'appuie sur les données du système, accessibles au travers des services de la couche inférieure, c'est à-dire la couche de données. En retour, elle renvoie à la couche présentation les résultats qu'elle a calculés. En pratique, on trouve au niveau de la couche métier :

- Des **Sessions Bean** qui proposent des méthodes pour manipuler les entités.
- Des **Entités Bean** dont la persistance est assurée par la couche de mapping.
- Les **API JNDI**, pour accéder au service de nommage ou d'annuaire.

» Couche de mapping objet/relationnel

La couche de mapping objet/relationnel transforme la représentation physique des données en une représentation objet et inversement. Ce mécanisme est assuré par JPA (Java Persistence API) qui utilise le protocole JDBC pour exécuter les appels SQL. Cette couche n'est utilisée que parce que notre base de données est relationnelle. En effet, si la persistance était faite de manière native en objet, ce

mapping n'aurait pas lieu d'être.

» **Couche de persistance**

Cette couche contient les données sauvegardées physiquement sur disque, c'est-à-dire la base de données. En Java, le protocole d'accès aux données est JDBC.

4.1.4 Architecture technique

Les couches que nous venons de décrire sont avant tout logiques et servent à décrire la conception de l'application. Nous allons maintenant projeter cette architecture sur plusieurs emplacements physiques et ci par le diagramme de déploiement.

» **Diagramme de déploiement**

Ce diagramme est très important dans les environnements distribués, Il décrit les ressources matérielles et la répartition du logiciel dans ces ressources [10]. Il permet d'établir la cartographie complète de déploiement du logiciel sur le matériel, de visualiser la topologie matérielle d'un système, et d'établir la nature des connexions reliant les éléments matériels du système.

Les principaux éléments utilisés dans les diagrammes de déploiement sont :

– **Les Nœuds :**

Un nœud représente un ensemble d'éléments matériels du système sur lequel sont déployés et exécutés un certain nombre de composants logiciels du système. Un nœud peut être un processeur, un périphérique ou un serveur. Cette entité est représentée par un cube tridimensionnel.

Il est possible de représenter des nœuds spécialisés. UML propose en standard les deux types de nœuds suivants :

* Les Unités de traitement : Ce nœud est une unité physique disposant de capacité de traitement sur laquelle des artefacts peuvent être déployés. Une unité de traitement est un nœud spécialisé caractérisé par le mot-clé «device».
* Les Environnements d'exécution : Ces nœuds représentent un environnement d'exécution particulier sur lequel certains artefacts peuvent être exécutés. Un environnement d'exécution est un nœud spécialisé caractérisé par le mot-clé «executionEnvironment ».

– **Les Artefacts :**

Un artefact est la spécification d'un élément physique qui est utilisé ou produit par le processus de développement du logiciel ou par le déploiement du système. C'est donc un élément concret comme par exemple : un fichier, un exécutable ou une table d'une base de données. Un artefact peut être relié à d'autres artefacts par notamment des liens de dépendance.

– **Les Composants :**

Un composant est assimilé à un élément exécutable du système. Sur un diagramme de déploiement, les composants sont placés dans des nœuds pour identifier l'endroit de leur déploiement.[11]

– **Les lignes de communication :**

Permet de représenter le lien entre un artefact ou un composant et son nœud d'appartenance.

Notre diagramme de déploiement se base sur une implémentation d'une architecture JEE avec six nœuds, plusieurs composants sont déployés :

* Le serveur d'application jboss est considéré comme une unité de traitement, donc il est caractérisé par le mot-clé «device», Ce serveur contient un interpréteur java (JVM - Java Virtual Machin) qui va exécuter le code java installé sur le lequel est déployée l'archive «.ear » composée de l'application web «.war» où se trouvent les composant servlet, jsp, css, et d'autres composants nécessaires au fonctionnement de cette archive web comme «ejb.jar» (classes permettant l'appel aux EJB) où se trouvent les composant session bean et entité bean.
* Le serveur de base de données (MySQL) : Le SGBD MySQL, il est surtout installé pour les applications Web, et celui qui va contenir la base de données du système.
* PC client : qui contient un navigateur Internet qui va se connecter avec le serveur d'application jboss via un réseau et par des requêtes HTTP il va scruter le système.

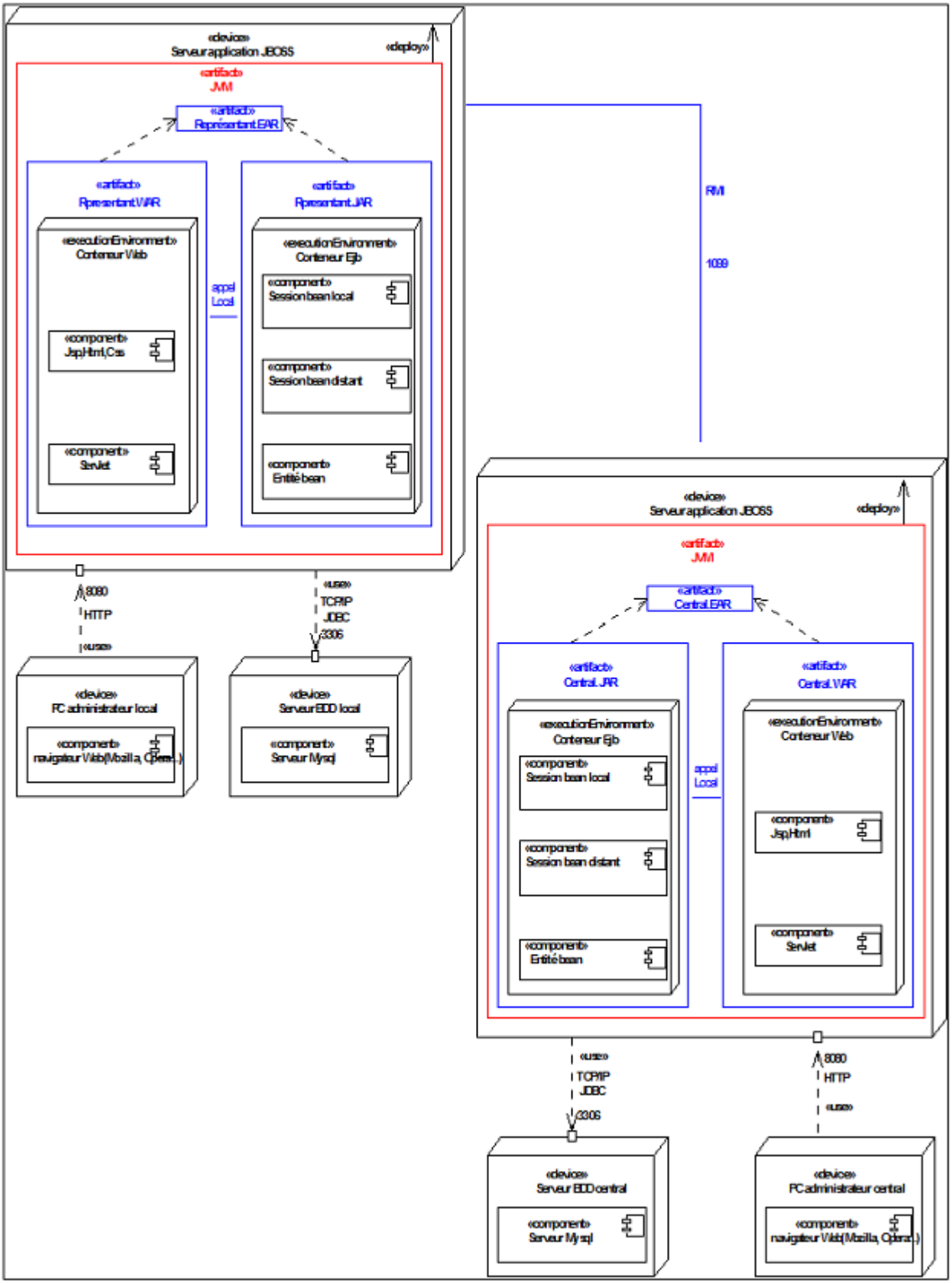

FIGURE 4.3 – Diagramme de déploiement

4.2 Choix technologiques

– **Servlet :**
Une servlet est en réalité une simple classe Java, qui reçoit une requête du client, et effectue des traitements et renvoie le résultat.

– **JSP :**
Java Server Page est une technologie Java qui permet de générer dynamiquement du code HTML. Elle mélange la puissance de Java côté serveur et la facilité de mise en page d'HTML côté client.

– **JSTL :**
JSP Standard Tag Library c'est un ensemble de balises personnalisées (Custom Tag), elle permet d'éviter l'utilisation de code Java dans les pages JSP et de réduire la quantité de code à écrire ainsi que rend le code des pages JSP plus lisible.

– **CSS :**
Cascading Style Sheets ou les feuilles de styles en français est un langage qui permet de gérer la présentation d'une page Web, il permet de définir des règles appliquées à un ou plusieurs documents HTML.

– **EJB Entité :**
Ces des EJB permettent de représenter et de gérer des données enregistrées dans une base de données.

– **EJB Session :**
Les EJB session sont des EJB de service dont la durée de vie correspond à un échange avec un client. Ils contiennent les règles métiers de l'application.

– **JNDI :**
Cette API fournit une interface unique pour utiliser différents services de nommages ou d'annuaires et définit une API standard pour permettre l'accès à ces services.

– **JPA :**
Java Persistence API permet faire la correspondance entre le modèle

de données relationnel et le modèle objets de la façon la plus facile possible.

– **RMI :**
Remote Method Invocation est un mécanisme permettant l'appel de méthodes entre objets Java s'exécutant sur des machines virtuelles différentes (espaces d'adressage distincts).

4.3 Outils et environnement de développement

- Java

Java est un langage de programmation orienté objet créé par James Gosling et Patrick Naughton de Sun Microsystems. Le langage Java a été introduit par la société SUN en 1995. Il possède de nombreuses caractéristiques on peut citer :

– Java fonctionne comme une machine virtuelle (Indépendant de toute plate forme).
– Java est simple .
– Java peut être utilisée sur Internet.
– Java est gratuite.
– C'est un langage compilé : avant d'être exécuté, il doit être traduit dans le langage de la machine sur laquelle il doit fonctionner.
– Java contient une très riche bibliothèque de classes(Packages) qui permettent de :
  – Créer des interfaces graphiques.
  – Communiquer à travers les réseaux. . . etc.

-L'IDE ECLIPSE

Notre choix s'est porté vers l'IDE Eclipse. Ce n'est pas le seul existant, c'est un outil puissant, gratuit, libre et multiplateforme, Massivement utilisé en entreprise.

l'IDE Eclipse permet l'intégration des outils nécessaires au développement et au déploiement d'une application et la génération automatique de portions de code ainsi que l'assistance à la volée lors de l'écriture du code.

-Le Serveur d' Application Jboss

Le serveur d'application jboss est un outil libre écrit en Java qui permet l'exécution de composants Java côté serveur (servlets, JSP, EJB, ...) selon les spécifications de la plate-forme J2EE/JEE.

-Le Serveur MYSQL

Le SGBD MySQL est supporté par un large éventail d'outils. MySQL est surtout installé pour les applications Web, ce SGBD est solide, libre, gratuite, et il est utilisé par de grands groupes spécialisés dans l'Internet.

4.4 Quelques interfaces de notre système

- Interface authentification

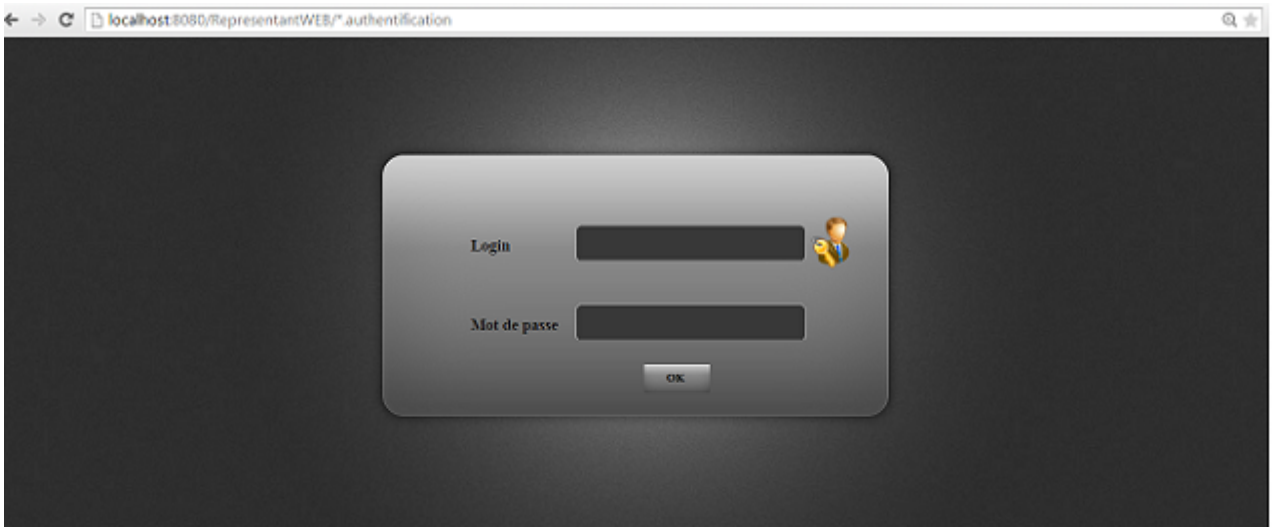

FIGURE 4.4 – Capture interface authentification

- **Interface d'accueil(Agence centrale)**

FIGURE 4.5 – Capture interface d'accueil

- **Interface d'accueil(Représentant)**

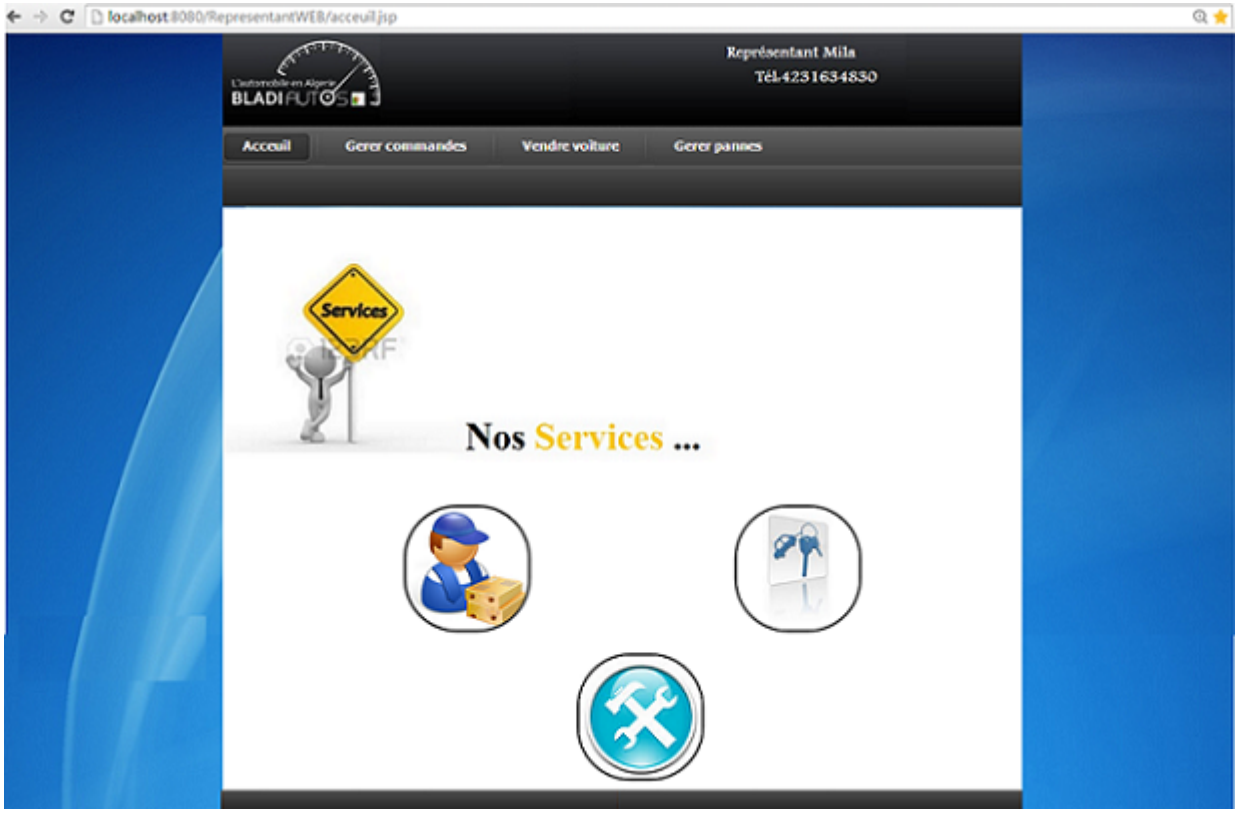

FIGURE 4.6 – Capture interface d'accueil

- **Interface commander voitures(Représentant)**

FIGURE 4.7 – Capture interface commander voitures

- **Interface affecter commande(Agence centrale)**

FIGURE 4.8 – Capture interface affecter commandes

- **Interface vérifier satisfaction des commandes(Représentant)**

FIGURE 4.9 – Capture interface vérifier satisfaction des commandes

- Interface consulter commande(Représentant)

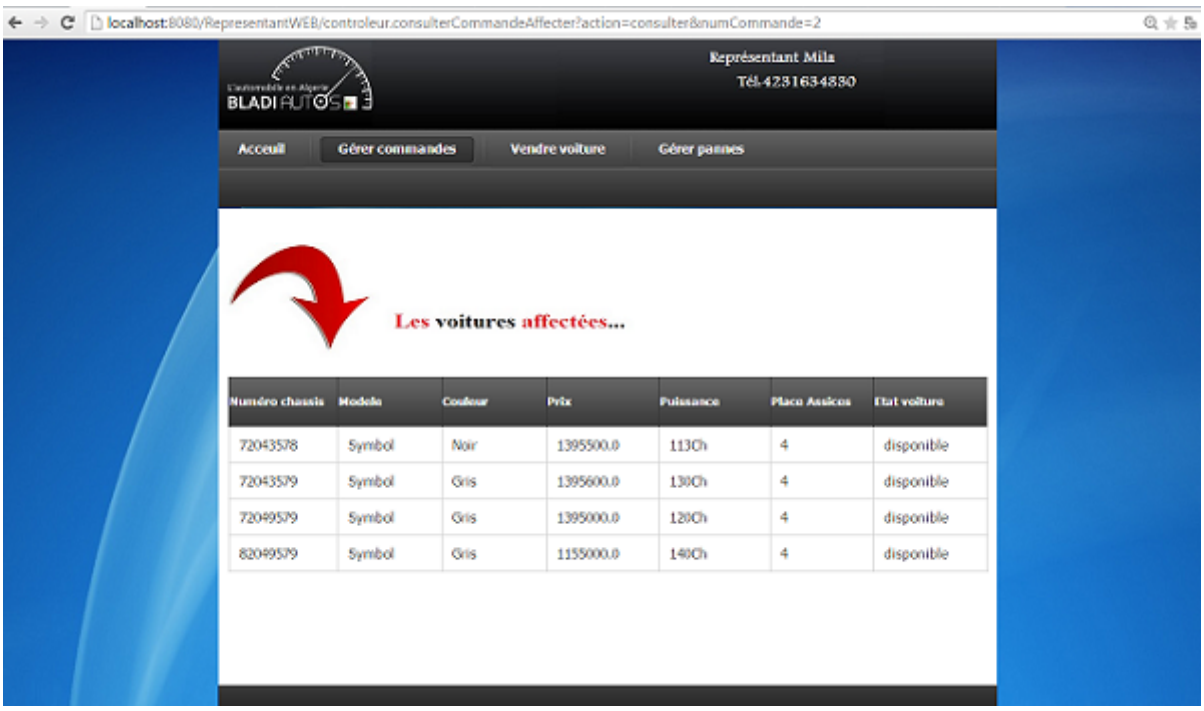

FIGURE 4.10 – Capture interface consulter commande

- Interface vérifier disponibilité de voiture(Représentant)

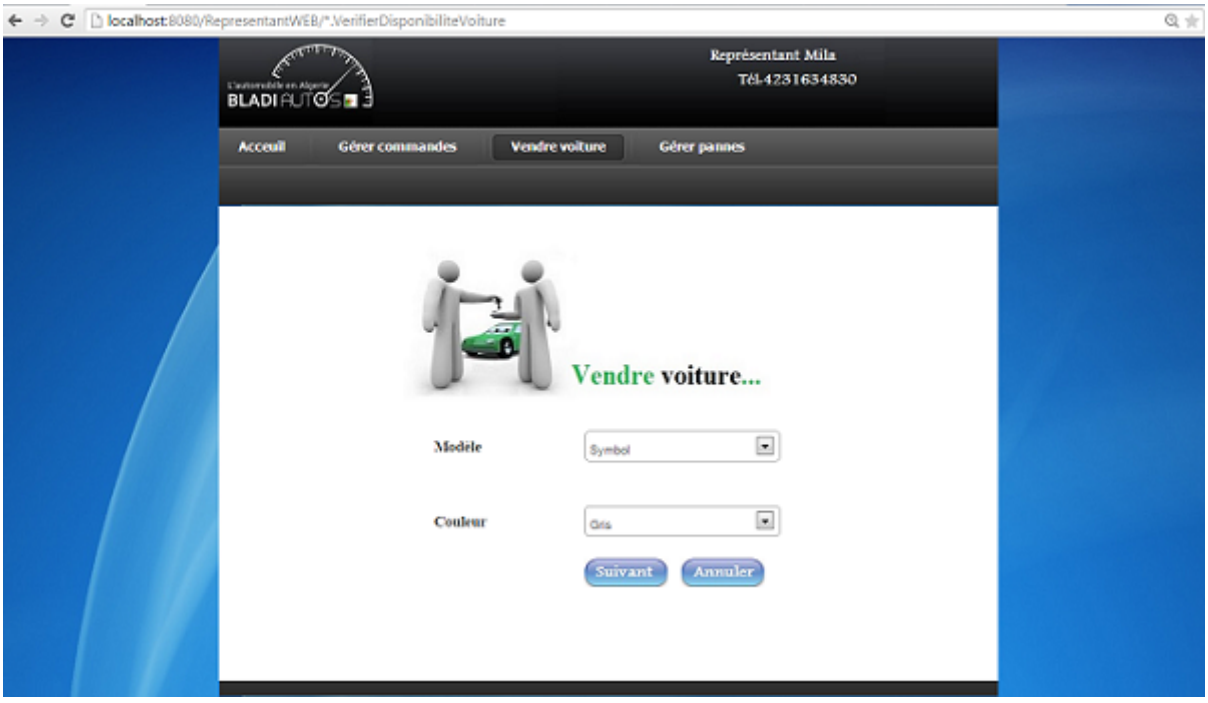

FIGURE 4.11 – Capture interface vérifier disponibilité de voiture

- Interface vendre voiture (Représentant)

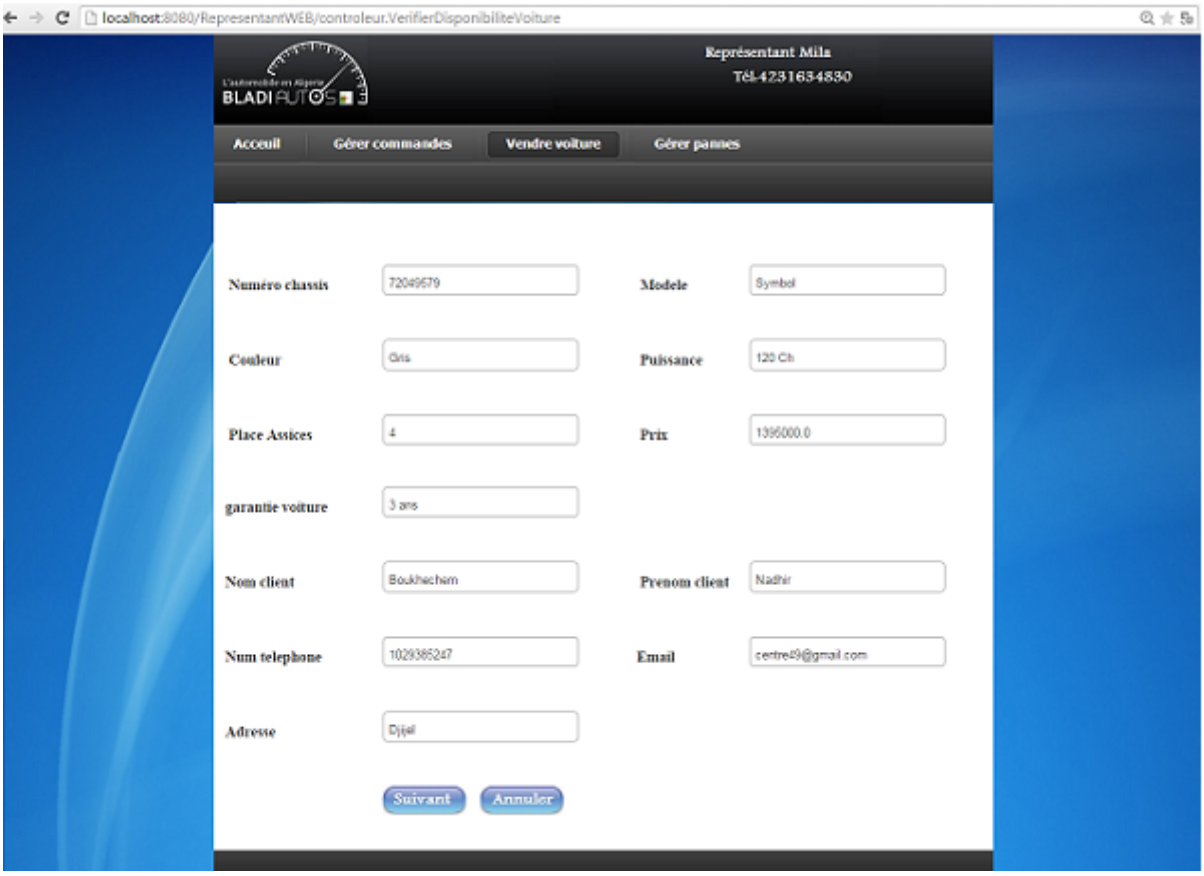

FIGURE 4.12 – Capture interface vendre voiture

- Interface vérifier garantie client-distant(Représentant)

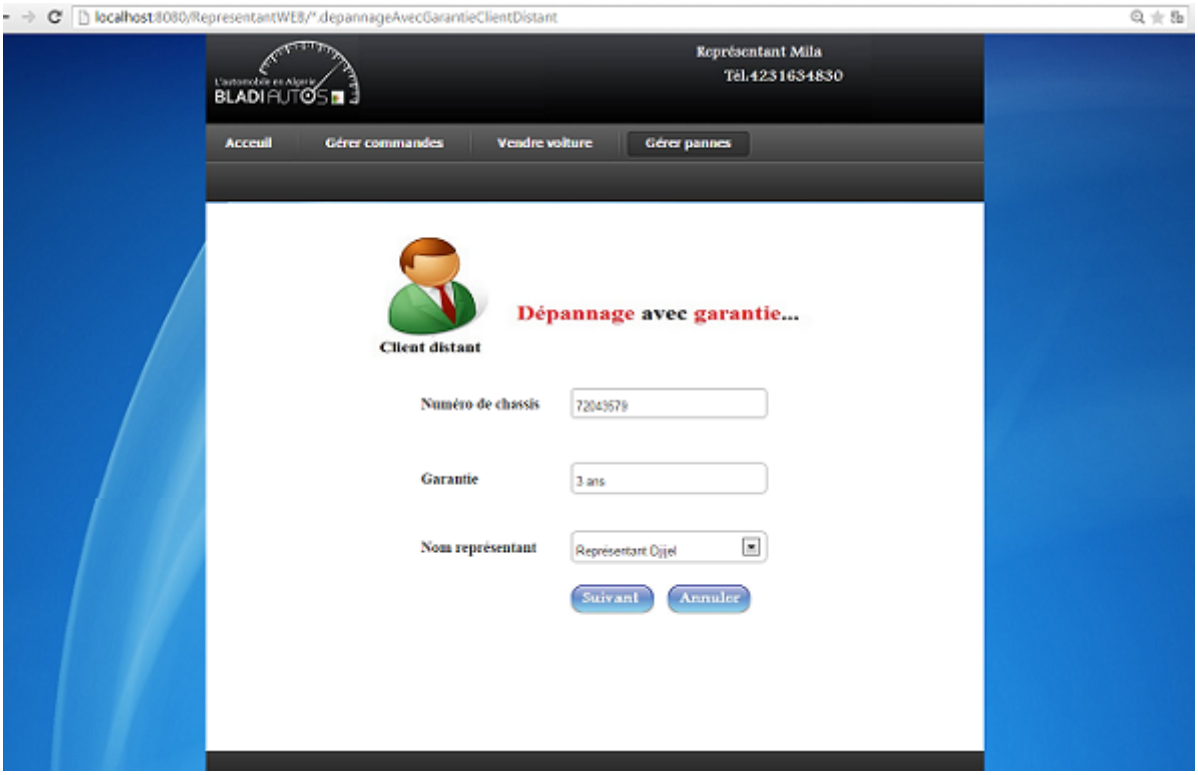

FIGURE 4.13 – Capture interface vérifier garantie client-distant

- **Interface dépannage avec garantie client-distant(Représentant)**

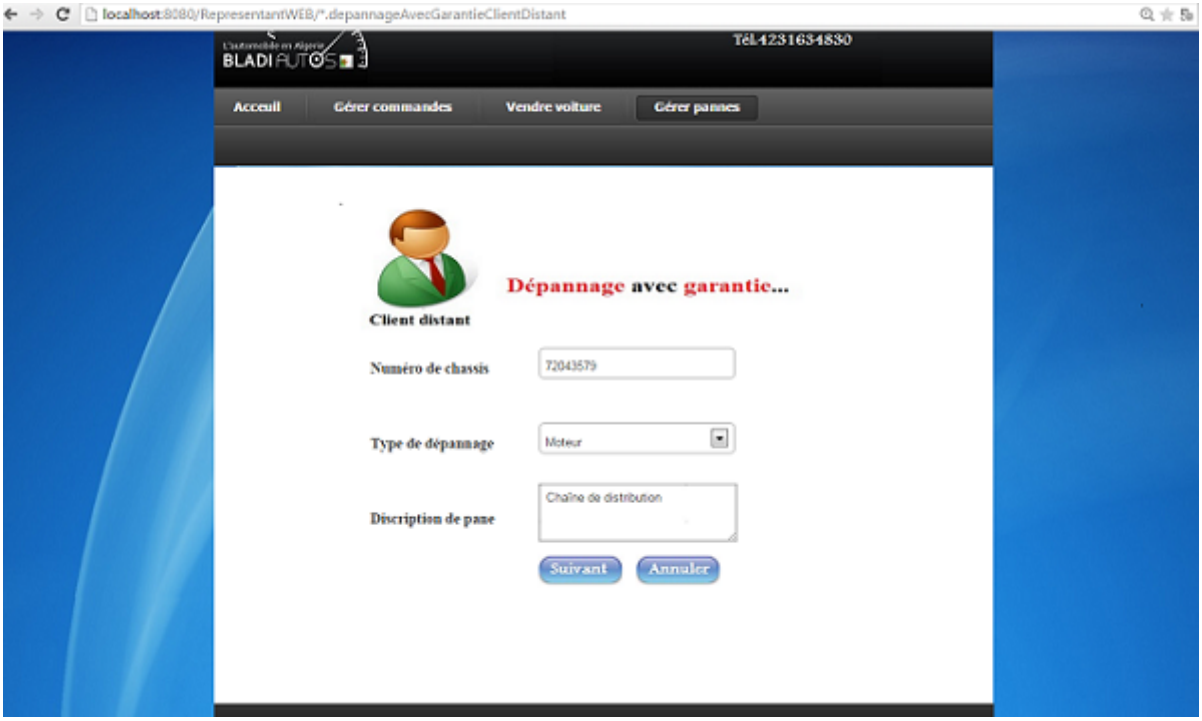

FIGURE 4.14 – Capture interface dépannage avec garantie client-distant

- **Interface calculer statistiques pannes(Agence centrale)**

FIGURE 4.15 – Capture calculer statistiques pannes

- Interface résultat statistiques pannes(Agence centrale)

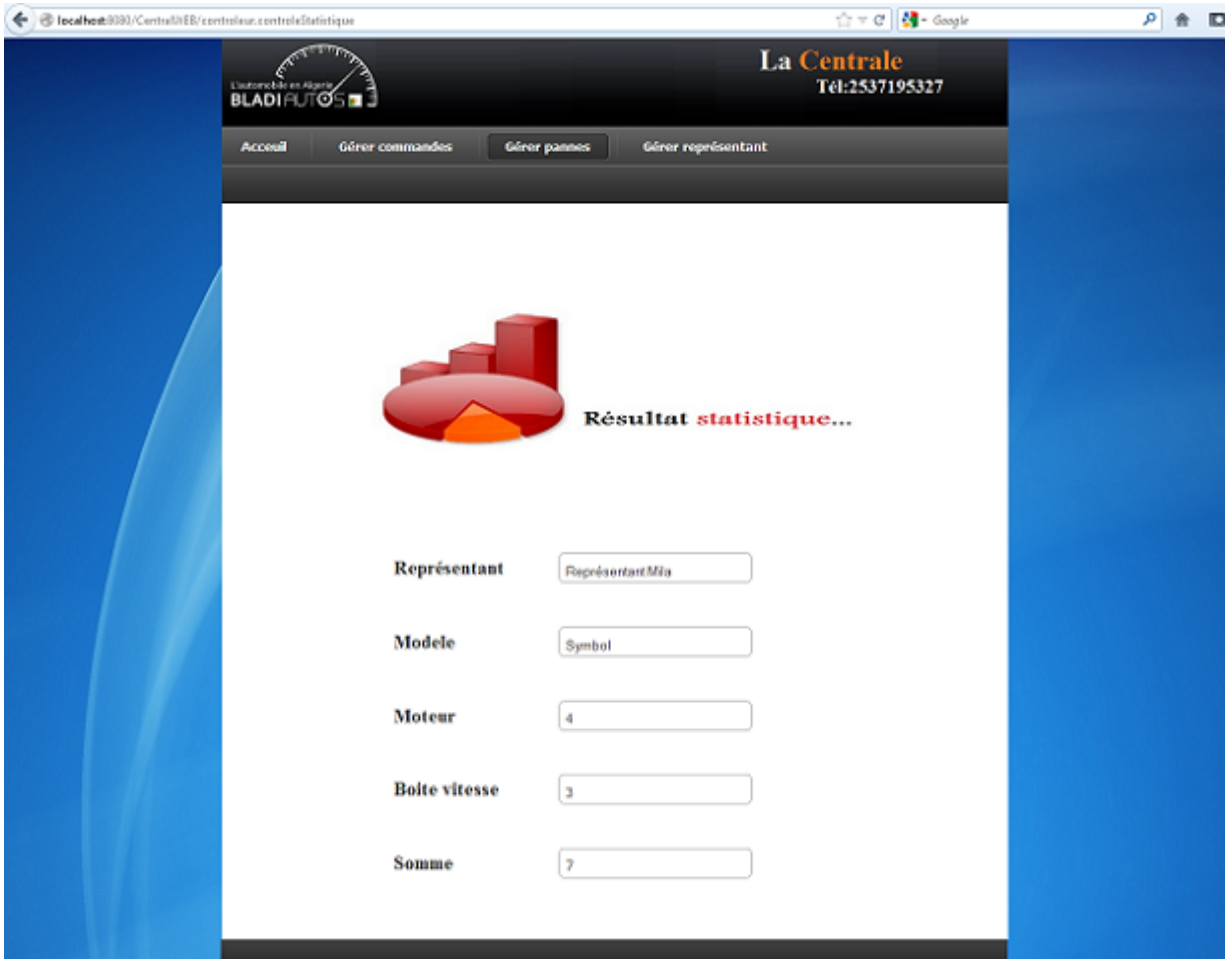

FIGURE 4.16 – Capture résultat statistiques pannes

4.5 Conclusion

Dans cette partie nous avons présenté l'architecture de notre application, Nous avons présenté aussi une brève description des différentes choix technologique, technique, et des outils de développement , finalement nous avons inclus quelques interfaces de notre système.

Conclusion générale

On a développé dans notre projet une application de vente et suivie des voitures, notre application est répartie entre différents sites, un site centrale et plusieurs sites représentants, les différents sites sont reliés par un réseau TCP/IP. L'application intègre une interface WEB très claire et facile a manipulé. Pour développer l'application on a utilisé Java EE, c'est une plate-forme très puissante, elle intègre différents outils qui facilitent le développements d'applications d'entreprises. Pour déployer l'application on a utilisé Jboss, un serveur d'application très populaire.

On peut dire qu'on a atteint les objectifs visés au début du projet, on a réussit à développer une application qui non seulement intègre une base de données répartie mais, en utilisant les EJB on a aussi distribué l'application qui gère cette base. La distribution de la base et de l'application rendent celle-ci très performante (grâce aux traitements et calculs réparties), et robuste aux pannes des liens du réseau entre les différents sites (grâce a la répartition des données).

Notre application peut être considéré comme un modèle qui peut s'appliquer pas seulement a la vente et suivie des voitures, mais aussi à n'importe quel autre produit, ce modèle sera très bénéfique aux entreprises qui veulent profiter de la puissance des nouvelle technologies en informatique pour améliorer leurs performances.

Malgré ce qui a été fait ce travail n'est pas encore terminé, il manque une partie très importante qui est la sécurité des applications et qui n'a pas encore été développé. Nous espérons pouvoir le compléter dans le future.

Bibliographie

[1] *Histoire développement d'applications Web.*Cours de Monsieur Martin Véronneau.

[2] *En pratique : HTML - définition.*Cours de Monsieur Jean-François Richard,novembre 2007.

[3] *Créez votre application web avec Java EE.*Livre, Par Coyote,Dernière mise à jour le 31/01/2013.

[4] *dévellopons en java* .Livre, Par Jean-Michel Doudoux.

[5] *SQL pour Oracle.*Livre, Par CHristian Soutou.

[6] *Gestion d'hébergement des Résidences Universitaires de Tlemcen Sous Oracle.*Mémoire de fin d'études pour l'obtention du diplôme de Master 2 en Informatique ,Université de Tlemcen 2012-2013.

[7] *Systèmes de Gestion de Bases de Données Réparties et Mécanismes de Répartition avec Oracle* .Cours,Par Rim Moussa 2005-2006.

[8] *Conception et réalisation d'une base de données repartie sous oracle.*Mémoire,Université A/Mira de Bejaia,Master II 2009.

[9] *UML 2.05.*Livre, Par Laurent Audibert.

[10] *UML, le langage de modélisation objet unifié.*Livre, Par Laurent Piechocki.

[11] *UML 2 analyse et conception.*Livre, Par Joseph Gabay et David Gabay.