RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ MOULOUD MAMMERI DE TIZI-OUZOU
FACULTÉ DE GÉNIE ELECTRIQUE ET INFORMATIQUE
DÉPARTEMENT INFORMATIQUE

Mémoire de Fin d'études

En vue de l'obtention du diplôme de Master en Informatique
Option : Système d'informatique
Thème :

Conception et réalisation d'un outil à intégrer dans un Système d'Information pour la Gestion des activités dans les zones adressables ou non adressables

Dirigé par :
Mr : *Dib. A*

Proposé par :
La Société Marine Soft.

Réalisé par :

Melle : *KHIAR faiza.*
Melle : *KHADER hayat.*

Promotion 2013-2014

REMERCIEMENTS

Nous remercions d'abord le bon dieu pour nous avoir donné le courage, la Patience, la santé et la volonté d'arriver à la fin de ce travail

Nous tenons à exprimer ici notre profonde gratitude et sincères remerciements à notre promoteur Mr Ahmed Dib, Mr Ladraa Karim, Mr Si ramdane Noureddine. , Mlle Nesnas radia qui nous ont honoré en acceptant de nous encadrer et pour leurs précieux conseils et orientations, ainsi que la confiance qu'ils ont placée en nous tout au long de la préparation de ce projet.

Que les membres du jury trouvent ici nos remerciements les plus vifs pour

Avoir accepté d'honorer par leur jugement notre travail.

Que tous ceux qui, de près ou de loin ont contribué, par leurs conseils, leurs Encouragements ou leur amitié à l'aboutissement de ce travail, trouvent ici L'expression de notre profonde reconnaissance.

Pour leur encouragement, leur soutien moral et la patience qu'ils nous ont manifestée Durant toute l'année, nous remercions fortement tous les membres de nos familles.

Enfin remercier nos parents serait se répéter, citer leur affection serait pléonasme, Parfois pour exprimer plus que ce qu'on a envie de dire on a recours au silence.

DÉDICACE

Je dédie ce modeste travail

A la mémoire de mes grands parents, que dieu bénisse leurs âmes.

A ceux qui ont éclairé ma vie, a la mémoire de mon cher père que dieux le garde dans son vaste Paradies, ma très chère mère, et son soutenu tout au long de mes études, et qui a fait de moi ce que je suis aujourd'hui et j'espère qu'un jour je serai capable de la donner au moins le minimum car quoiqu'on face on arrivera jamais à rendre tout.

A mes très chères sœurs Aicha et son mari Mohammed, Fatima, Rachida.

A mes adorables frères Arezki, Ali, et Yacine.

A mon binôme Hayat et à toute sa famille.

A tous mes ami(e)s, et particulièrement Zakia, Houda, Ratiba, Karima, Smail , Yacine, Idir.

.

Sans oublier mes camarades de la promotion 2013-2014

KHIAR Faiza .

DÉDICACE

Je dédie ce modeste travail

A la mémoire de mes grands parents, que dieu bénisse leurs âmes.

A ceux qui ont éclairé ma vie, ma très chère mère et mon cher père pour leur aide et leur soutien tout au long de mes études, et qui ont fait de moi ce que je suis aujourd'hui et j'espère qu'un jour je serai capable de leur donner au moins le minimum car quoiqu'on face on arrivera jamais à leurs rendre tout.

A ma très chère sœur Dahbia.

A Mes deux adorables frères, Rézki et Mohammed.

A mon binôme Faiza et à toute sa famille.

A tout mes cousins et cousines.

A la petite Aya qui j'adore énormément.

A tous mes ami(e)s, et particulièrement : Zakia, Nacéra, Nouara, Kahina, Nabila, Zakia, Hakim, Smail , Yacine, Idir.

Sans oublier mes camarades de la promotion 2013-2014.

KHADER Hayat

SOMMAIRE

Sommaire

Partie III : Réalisation

Chapitre V : Architecture et Outils de développement.

Liste des acronymes

ZSD : Zone Sous Douane.

D1 : Document N^0 1(Manifeste Douanier).

OT : Ordre de Transfère.

J2EE : Java Entreprise Edition.

JSF : Java Server Faces

UML : Unifified Modilinge Language.

OO : Orienté Objet.

PE : Processus Elémentaire.

SP : Sous Processus.

EE : Elément d'Entrée.

ES : Elément d'Sortie.

BAE : Bon A Enlever

EAR : Entreprise Archive.

WAR: Web Archive.

JAR : Java Archive.

EJB : Entreprise Java Bean.

DAO : Data Access Object.

JAP :

CSS: Cascading Style Sheets.

HTML: Hypertext Markup Language.

JPQL : Java Persistance Query Language.

Liste des figures

Liste des tableaux

Résumé

Dans le cadre de stage de fin d'étude en vue de l'obtention du diplôme de MASTER en informatique, la Sarl Marine Soft, société de service et d'ingénierie en informatique, nous 'a proposé un projet visant à concevoir et à réaliser un système d'information pour la gestion des zones adressables et non adressables. Ce système est un module a intégré non seulement à un ERP dédié à la logistique du transport de marchandise mais dans l'import qu'il logiciel de gestion de stock.

Il s'agit donc d'une application 3tiers pour la gestion des aires de stockage destiner aux l'import quelle domaine qui traite la gestion de stock, suivant tous les mouvements de la marchandise effectuer dans leur zone de stockage.

Le présent projet se résume donc, à concevoir et réaliser un système qui permettra de répondre aux besoins de la gestion des zones adressables et non adressables. Celui-ci devra, entre autre, prendre en charge l'entrée et la sortie des marchandises ainsi que les éventuels mouvements de celle-ci à l'intérieur des zones.

Mots clés :

Marine soft, SI, 3-Tiers, N-tiers, Approche processus, POO, Eclipse, Design patterns, J2EE, Java Script, CSS, Riche Faces, BIRT, JPQL, Jboss.

****Introduction générale ****

De l'âge de la pierre à nos jours, l'esprit perfectionniste de l'homme n'a cessé de lui permettre d'améliorer sa vie quotidienne. Le passage de la mécanique aux domaines d'informatique, d'électronique, et d'automatique a révolutionné la vie journalière de l'être humain. Les nouvelles technologies de l'information et de communication illustrent ce phénomène.

Aujourd'hui, vu l'intérêt croissant de vouloir gagner en temps, de conserver les données, de limiter le nombre d'employés et pas mal d'autres raisons, ont poussé petites, moyennes et grandes entreprises à chercher des solutions informatiques capables de répondre à leurs besoins.

Dans le cadre de la présentation du mémoire de fin d'étude pour l'obtention du diplôme Master en informatique (système d'informatique), la direction recherche & développement de l'entreprise « Marine Soft » nous a chargées de réaliser le projet intitulé « la conception et la réalisation d'un outil a intégré dans un system d'Information SI pour la **Gestion des activités dans les zones adressables et non adressable »** qui est une partie importante de noyau de l'ERP.

L'objectif de notre projet est d'enrichir les produits de l'entreprise par une nouvelle solution standard destiner a un importe quel métier afin de couvrir l'ensemble des fonctions de gestion d'activités de marchandise dés sa réception jusque a son expédition passant par le processus d'entreposage, pour assurer l'organisation interne optimale de l'activité liée à cette marchandise.

Pour mener à terme notre travail, nous le répartissons de la manière suivante :

- 1er chapitre : **Présentation générale**

Dans ce chapitre nous présentons les objectifs de notre stage de fin d'étude, ainsi l'organisme d'accueil à travers son organigramme et ses missions puis nous proposons une démarche de réalisation de projet.

- 2eme chapitre : **Analyse et étude de l'existant**

Le seconde chapitre s'agit d'une prise de connaissance de l'existant pour savoir de ce que doit être capable de faire et de quoi va servir notre futur application en d'autres termes il s'agit d'une analyse et étude de l'existant.

- 3me chapitre : **Spécifications Fonctionnelles Détaillées.**

Ce chapitre va être comme reflex de solution recherche, ainsi est le canal qui relier l'analyse et la conception.

- 4eme chapitre : **Conception.**

Le quatrième chapitre sera consacré à la conception de l'application, il s'agit d'une phase de modélisation théorique de l'application.

- 5me chapitre : **Réalisation.**

Ce chapitre englobe un premier point qui porte sur L'architecture et outils de développements et un deuxième qui porte sur l'environnement de développement de l'application ainsi que quelques interfaces de celle-ci.

Partie II

Contexte Du Mémoire

Chapitre I : Présentation générale

- *Objectifs de stage de fin d'étude « MASTER ».*
- *Présentation de l'organisme d'accueil « Marine Soft.*
- *Présentation de sujet.*

Introduction

Le stage pratique sanctionnant la dernière année en informatique est une véritable mine d'or en termes d'apprentissage professionnel. Cette première expérience dans le vrai monde du travail représente pour nous une première approche auprès des professionnels qui opèrent dans notre domaine de spécialisation. La mise en pratique, sur une période de six mois, des connaissances acquises durant notre cursus et la découverte des rouages de ce domaine d'activité nous permettra de mieux cerner les taches qui nous seront assignées, et par conséquent, consolidera les bases de notre métier et donc la réussite de notre carrière professionnelle future.

MARINE SOFT, société de service et d'ingénierie en informatique est implantée à Alger. Elle dispose d'une atmosphère de travail idéale pour l'élaboration du mémoire de fin d'étude. La motivation essentielle à travers ce projet de fin d'étude réside principalement dans le fait que le produit sur lequel nous travaillons sera, une fois parachevé au terme du stage, commercialisé.

1-Présentation de l'organisme d'accueil « Marine Soft »

1.1- Historique

MARINE SOFT Sarl, EDITEUR ET INTEGRATEUR DE SOLUTIONS LOGICIELLES, est une Société de services et d'ingénierie en informatique. Elle est issue de la société **AMS (A**lgérienne **M**aritime **S**ervice) crée en 1995 en tant que représentant des armateurs allemands Sloman Neptun et Cargo Levant. Créée en Septembre 1998 comme étant une filiale d'AMS, Marine Soft évolue en tant qu'entité autonome de puis le 01/04/2000. Marine Soft est entièrement spécialisée dans la conception, la réalisation, la mise en route et l'assistance technique de divers logiciels relatifs à des activités liées au transport.

Le conseil et l'audit en info - logistique confèrent à MARINE SOFT le statut de partenaire viable pour le développement aux yeux des professionnels de la chaîne de transport. Elle relie les entreprises privées, leurs organisations, les administrations et les institutions publiques pour former une chaîne logistique optimisée en adoptant des processus communs. Ses applications et ses solutions sont spécifiquement conçues et sont entièrement dédiées aux besoins des professionnels de la logistique du transport.

La force de ses produits vient de leur haute qualité opérationnelle qui est le résultat d'un travail rigoureux, conduit dans des conditions opérationnelles pointilleuses tout au long des différentes étapes de réalisation : depuis la conception jusqu'à la mise en exploitation du produit.

1.2- Les missions et offres de Marine Soft

Marine Soft Sarl a pour mission :

La conception, la réalisation, la mise en route et l'assistance technique de divers logiciels, adaptés à l'environnement institutionnel et économique algérien et international.

L'offre de Marine Soft s'articule principalement autour de progiciels sous WINDOWS, Modulaires et homogènes, complétée par un ensemble de prestation de services et d'ingénierie

Les offres de Marine Soft sont :

- **Assistance à la maitrise d'ouvrage :**
  - -Participer aux cadrages des projets.
  - -Conduire les études préalables aux projets.
  - -Rédiger les cahiers des charges.
  - -Mission d'expertise fonctionnelle.
- **Maitrise de l'œuvre :**
  - -Conduire l'étude détaillée des projets.
  - - Réaliser des maquettes.
  - - Réaliser et mettre en œuvre les applications.
  - -Conduire des tests applicatifs.
- **Respect du « triangle de la performance » :**
  - Respect des couts, délais et de la qualité du produit final à livrer.
- **Veilles technologique.**
- **Participation au groupe qualité du service informatique.**
- **Encadrement d'équipes multidisciplinaire dans le développement des logiciels**
- **Assistance aux utilisateurs.**
- **Assurer le service après vente**
- **Diagnostic et repérage d'éléments défectueux**

1.3- L'organigramme de Marine Soft

L'organigramme présenté ci-dessous offre une vue globale de la structure fonctionnelle de Marine Soft. Mis en évidence, le département Recherche & Développement, le cœur de l'entreprise, est chargé, entre autre, de l'encadrement et du suivie des stagiaires.

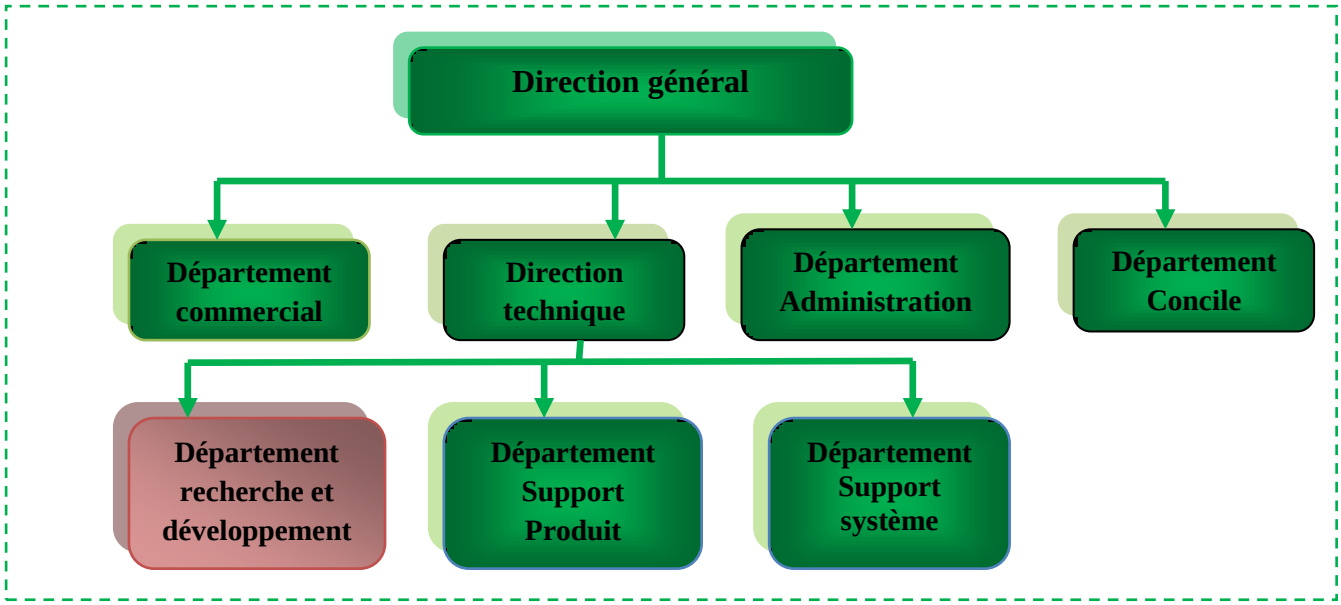


Figure I.1 : Organigramme de Marine Soft

1.4-Description de l'organigramme de Marine Soft

Le tableau ci-dessous offre une vue organisationnelle des missions des différentes départements de Marine Soft.

Département	Missions
Département Commercial	✓ Communication. ✓ Marketing. ✓ Ventes directe & indirecte ✓ Animation d'un réseau de partenaires (SSII/Intégrateur) Algérien et étrangers
Département Administration	✓ Comptabilité/paie. ✓ Facturation/Recouvrement. ✓ Administration des ventes (Contrats &Conventions de Formation).
Département Conseil	✓ Conseil (architecture des systèmes informatiques, études comparatives, étude de migration, optimisation).
Département Recherche & développement	✓ Développement de la gamme de progiciels de MARINE SOFT. ✓ Maintenance progiciels

	✓ Assistance technique
Département Support produit	✓ Formation des utilisateurs. ✓ Installation et paramétrage des logiciels. ✓ Assistance téléphonique. ✓ Télémaintenance. ✓ Service help (outils d'aide au support).
Département Support système	✓ Gestion des parcs informatiques. ✓ Assistance téléphonique et télémaintenance. ✓ Maintenance (hardware, administration, serveur). ✓ Configuration de messagerie. ✓ Suivi des achats du matériel

Tableau I.1 : Les différant départements de Marine Soft.

1.5-La clientèle de Marine Soft

Marine Soft développe ses produits pour des organismes qui ont des activités liées au domaine maritime comme :

- ✓ EPAL [Entreprise Portuaire d'Alger].
- ✓ AMS [Algerian Maritime Services] (EUKOR, Sloman Neptun, Cargo Levant).
- ✓ SBC [Shipping & Brokerage Company] (GEFCO, NAVIS, Louis Dreyfus LINES).
- ✓ CMA-CGM Algérie (Groupe CMA-CGM).
- ✓ Arkas Algérie (Groupe ARKAS).
- ✓ Mondial Shipping Company (Marfret, UASC).
- ✓ TIBA (Evergreen Marine Corp).
- ✓ Cargo Levant.
- ✓ Sloman Neptun.
- ✓ Groupe d'Alessandro Tunisie (Groupe Grimaldi).
- ✓ Tunisamar Tunisie [Groupe Ismar] (Hanjin Shipping).
- ✓ MILAHA SERVICES (Brointermed Lines Limited).

- ✓ Rail Transit (Groupe Société Nationale des Transports Ferroviaires).
- ✓ ALTERCO [ALgerian TeRminal COntainer] (Groupe CMA-CGM).
- ✓ ATERCO (Groupe ARKAS).
- ✓ ACS [Algerian Containers Services] (filiale Entreprise Portuaire d'Alger).
- ✓ Sudcargos Algérie (filiale Sudcargos).
- ✓ Satrans [Services annexes aux transports maritimes].
- ✓ Walship (Worms Services Maritimes).

1.6-Les produits de Marine soft

1.6.1- Les produits développés

Le tableau présenté ci-dessous comporte la liste des produits conçus et réalisés par Marine Soft. Cette gamme de produit est opérationnelle et fonctionne convenablement. Certains sont utilisés depuis une dizaine d'années et ont atteints un niveau de maturité exemplaire. Leurs destinations sont des œuvrant dans un ou plusieurs maillons de la chaine de transport.

Désignation	Date de création	Composants	Architecture	Utilisation
GAM **Global Agency Management**	1998	-Le suivi des Opérations Navire. -La documentation. -Déclaration douanière. -Suivi des situations BL. -Gestion des Conteneurs. -Etablissement des comptes escale et des comptes armateur. -Facturation.	Client/serveur **2tiers**	Agences consignataires
ZSD **Zone Sous Douane**	1999	-Suivi de la documentation. **-La gestion de la zone**. -Facturation.	Client/serveur **2tiers**	Le port d'Alger Les ports secs.

MSGMAN Gestion de la manutention	2002	-Suivi des Opérations Navire. -Gestion des moyens humains et matériels. -Gestion de l'exploitation. -Suivi de l'état de débarquement & embarquement. -Statistiques. -Facturations.	Client/serveur **2tiers**	Le port d'Alger Les ports secs
MS TRANSIT	2001	-Suivi des dossiers. -Déclaration. -La gestion du crédit. -Gestion des Conteneurs. -Les réimprimé. -Suivi de la tarification. -Établissement des comptes clients. -Facturation.	Client/serveur **2tiers**	Agences transitaires
MS Transport	2002	-La gestion des transferts. -La gestion des transports. -Facturation.	Client/serveur **2tiers**	Agences transitaire.
GPP Gestion des Parcs Pleins	2005	-Gestion des Conteneurs. -Suivi des différents déplacements du conteneur. -Facturation.	Client/serveur **2tiers**	Parc de conteneurs pleins
GPV Gestion des Parcs Vides	2006	-Gestion des Conteneurs. -Suivi des différents déplacements du conteneur. -Facturation.	Client/serveur **2tiers**	Parc de conteneurs vides
GRC		-Le suivi des mouvements conteneurs (entrée /sotie).		Parc qui s'occupe de la

Gestion de la Réparation Conteneurs	2006	-L'établissement des états conteneur -Facturation	Client/serveur **2tiers**	réparation des conteneurs endommagés.

Tableau I.2 : Les produits développés de Marine Soft.

1.6.2- Le produit en cours de développement

Le tableau suivante complément du précédant, comporte la liste des produit en cour de réalisation

La désignation	Date de Création	Composant	Langage de programmation	Architecture	Utilisateur
GMAO Gestion Maintenance Assistée par Ordinateur	2012	-Gestion patrimoine. -Gestion stock. -Gestion d'intervention. -Gestion d'achat. -Gestion de ressources humaines.	JAVA	Client/serveur 3tiers	Port d'alger
GLS Global logistic Softwar	2013	-Facturation -Consignation navire et marchandise -Manutention portuaire de la marchandise -Zone d'entreposage de marchandise sous douane -Transit -Transport terrestre	JAVA	Client /serveur 3-tiers	Tout les maillon de la chaine de transport Agents consignatairs transitaire ports ports secs Entropots

Tableau I.3 : Les produits en cour de développement de Marine Soft.

Remarque

Il est à noter que dans ce présent tableau, la notion **« 2-Tiers »**laisse place a la notion **« 3-Tiers »,** cette transition n'est pas le fruit du hasard, mais une démarche réfléchie par l'entreprise et qui fait suite au premier diagnostic établi par les responsables de Marine Soft

1.7- Diagnostic

Comme tout éditeur et intégrateur de solution logiciel, Marine Soft doit suivre l'évolution technologique, les tendances et les normes, il en va de sa survie. La normalisation, l'intégration, la réutilisation, l'autonomie…etc. Des notions révolutionnaires qui ne laissent pas l'organisme indifférent. Une remise en question s'imposait. L'organisme s'interroge constamment: ***Sommes-nous à jour ?,*** et si c'est le cas, ***pour combien de temps ?***

Des critiques soulevées, on retiendra les points les plus importants :

- « l'architecteur client-serveur 2 tiers ».
- mono système d'exploitation.
-exploité sur un seul type d'équipement.

Très limité et dépassée depuis des années, la maintenance des produit conçues sur cette base est complexe notamment à cause du codage en bloc et sans parler de l'intégration qui est généralement impossible ou hors de prix. Marine soft est une entreprise commerciale, par conséquent elle dépend du marché, elle doit donc s'y adapter, mot d'ordre à long terme : adopter le changement. Un projet de rénovation est lancé.

1.8- Perspectives de développement

Intitulé « Projet trois tiers » l'innovation a été mise en place avec la collaboration et l'assistance d'un expert sollicité par l'entreprise, dont l'objectif était de redonner un nouveau visage à l'entreprise en lui offrant la possibilité de s'adapter au changement et cela en adoptant des standards et des normes nouveaux. L'entreprise a opté pour la « la plate-forme J2EE », proposée par la société Sun et portée par un consortium de sociétés internationales. visant à définir un standard de développement d'application d'entreprise multi-niveaux, basées sur des composants.

Cités précédemment dans le tableau des produits en cours de développement, GAMAO a été une première mise en pratique de la nouvelle technologie. Une expérience prometteuse consolidée, ensuite, par des interventions régulières de l'expert, en JEE, qui intègre, au sein de l'entreprise, de nouveaux concepts technologiques qui rehaussent la plateforme à un niveau de performance exemplaire. Le projet GLS, figurant aussi dans le tableau des produits en cours de

développement, est un ERP, destinée à la logistique de transport, qui héritera de dix années d'expérience dans le métier et sera implanté suivant les technologies les plus récentes dans le monde.

2- Présentation de sujet.

2.1 - Position de problème.

La gestion des activités (Entrée, Sortie, mouvement) de la marchandise au sein d'un espace de stockage représente une partie puissante dans la gestion des zones, c'est un module commun à toutes les activités d'entreposage de marchandise et également le régime qui assure l'emmagasinage de cette marchandise. Les aires de stockage envahissent généralement des étendues importantes, et elles nécessitent beaucoup de moyen matériels et humains et exige un réel savoir faire quant à leur gestion.

Une multitude de solutions liées à cette activité s'ouvrent aux professionnels, ont fait l'objet de critiques, qui ont été citées, précédemment, dans le cadre du diagnostic effectué par la direction de Marine Soft concernant sa position par rapport à l'évolution technologique, qui est à l'origine du grand pas technologique dont est témoins l'entreprise. Ces solutions ne sont destinés qu'à un nombre réduit d'entreprises implantés sur le territoire national, nous en déduirons, donc, que leur exploitation à l'étranger n'est pas possible car le niveau de paramétrage ne permet pas de le faire et cela n'est pas conforme au nouveaux objectifs de l'entreprise qui veut gagner des parts du marché international, qui se veut très exigeant, sur tous les niveaux.

Parmi les produits proposés par cette entreprise pour géré les aires de stockage on citera le logicielle ZSD, ce dernier il est destiné seulement à la gestion des espaces de stockage sous douane (les conteneurs), il a pu répondre a certains exigences des clients, mais il faut signaler que les clients ont toujours des besoins plus récent et l'entreprise à l'objectif de satisfaire ses clients pour gagné leurs fidélité et standardisera ses logicielles.

Marine soft ambitionne à enrichir ses produits par une nouvelle solution qui couvre l'ensemble des fonctions de gestion d'activités de marchandise dés sa réception jusque a son expédition passant par le processus d'entreposage, afin d'assurer l'organisation interne optimale de l'activité liée à cette marchandise.

2.2- Les objectifs attendus du projet.

Nous mentionnons que les objectifs attendu de ce projet vont être conforme aux besoins des clients d'un coté et les besoin de l'entreprise d'un autre coté, donc notre travaille va allez un peut plus loin. Il sera dédié non seulement aux terminaux à conteneurs, port secs, zones sous douanes mais il doit aussi répondre à des entrepôts de différents métiers, ca veut dire quelque soit la sorte da la marchandise.

Pour atteindre les objectifs tracés par Marine Soft le nouveau système doit respecter les contraintes architecturales suivantes :

- La standardisation.
- Faciliter la maintenance.
- Paramétrage.
- Commercialisation.
- L'intégration dans un ERP.

2.3-présentation de L'ERP GLS

Entreprise Ressource Planning ***(ERP),*** *en français progiciel de Gestion Intégré* ***(PGI)*** est une notion nouvelle et fort prometteuse qui a fait son entrée au sein de l'entreprise et a suscitée beaucoup d'intérêt .Elle se décline simplement en un progiciel qui intègre les principales composantes fonctionnelles d'une entreprise. Etant un système unifié, les utilisateurs de différents métiers travaillent dans un environnement applicatif identique. Ce modèle permet d'assurer l'intégrité des données, la non-redondance de l'information, ainsi que la réduction des temps de traitement.

NOTE : L'entreprise Marine Soft nous a dédie un projet qui présente un module dans un ERP.

La représentation graphique, présentée ci-dessous, offre une vue globale de l'ERP, du point de vue métier.

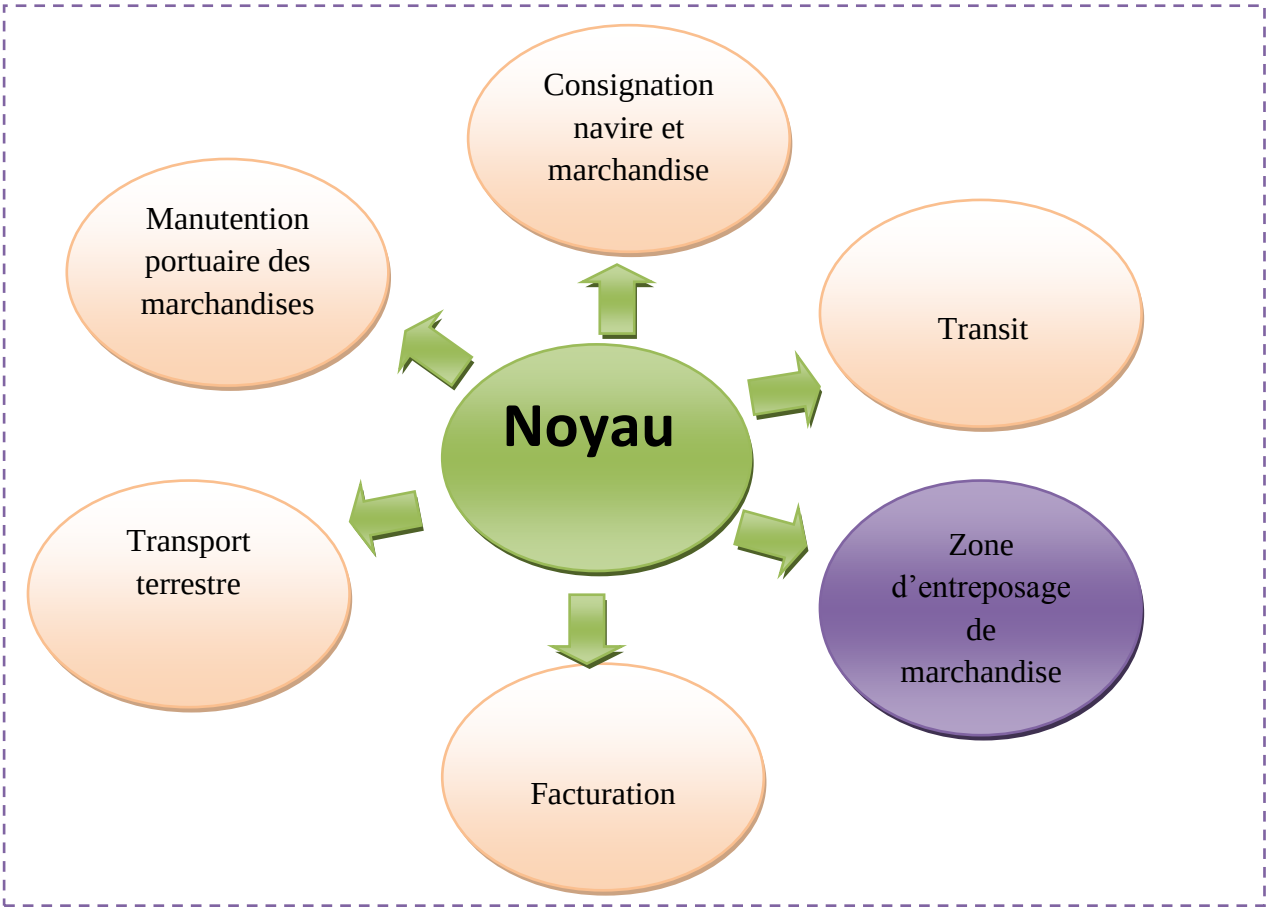


Figure I.2 : Architecture de l'ERP GLS

❖ **Conclusion**

Il a été question, au cours de ce chapitre, de mettre en évidence et de détailler les points importants qui caractérisent ce projet. Le premier étant les objectifs du stage en master, le seconde étant la présentation de l'organisme d'accueil, suivi par la présentation du sujet, et en fin conclure par la méthodologie regroupant le choix du langage de modalisation et la démarche de réalisation du projet

Le chapitre qui suit, sera consacré à l'étude de l'existant qui prendra en charge les etapes1, 2,3 et 4 de la démarche de réalisation de ce projet.

Partie II

Analyse

Chapitre II : Etude de l'existant

- *Choix de l'approche.*
- *Présentation de ZSD 2013 V1 (cas d'étude).*
- *Analyse et modélisation des processus métiers.*
- *Diagnostique des imperfections de système.*
- *Proposition de la solution informatique.*

Chapitre III : Spécifications Fonctionnelles détaillées

- *Spécifications Fonctionnelles détaillées. V 1.0*

❖ **Introduction**

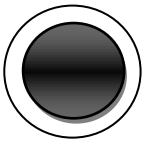
L'étude de l'existant est une étape décisive dans le processus d'analyse et modélisation de système d'information. Le choix d'une méthode d'analyse devait être conforme, d'un coté, à la tendance de normalisation de l'entreprise, et de l'autre, aux formalités liés au contenue de mémoire de fin d'étude. Nous aurons utilise comme une méthode d'analyse « l'approche processus » au quelle est consacré la premier section de ce chapitre. La deuxième section sera consacrée à la présentation d'une application commercialisée servira comme un échantillon d'étude. La troisième section mettra en évidence, via les cartographies des processus la délimitation progressive du champ d'étude pour au final présenter une solution informatique, synthétiser les résultats de cette étude.

1- Cycle de vie d'une marchandise

Apres l'entrée de la marchandise à un aire de stockage, cette dernière passe d'un état à un autre selon le cheminement des événements. L'état de la marchandise donne, non seulement, une information sur son état actuelle mais aussi sur les futures états possible de celle-ci.

Le diagramme d'état de transition est l'outil que propose UML pour la modélisation des différents états d'un objet ou d'un composant ainsi que les événements qui cause les éventuelles transitions de celui-ci d'un état à un autre. Il utilise un formalisme et une annotation simple dont les concepts sont exposés dans le tableau suivant :

Concept	Notation	Description
Etat		Un état est une situation durable dans le quelle peut se trouver un objet, à la quelle est associe des règles de gestion et des activités particulières.
Transition		Une transition est une relation entre états signifiant que le passage de l'un à l'autre est possible. Cette transition se

		fait en réponse à un ou plusieurs évènements définit
Etat initial		L'état initial est l'état d'un objet avant toutes transitions. Un seul état initial est autorisé sur un diagramme
Etat final		L'état final représente un état à partir du quel il n'est plus possible de transiter

Tableaux II.1 : formalisme du diagramme d'état de transition

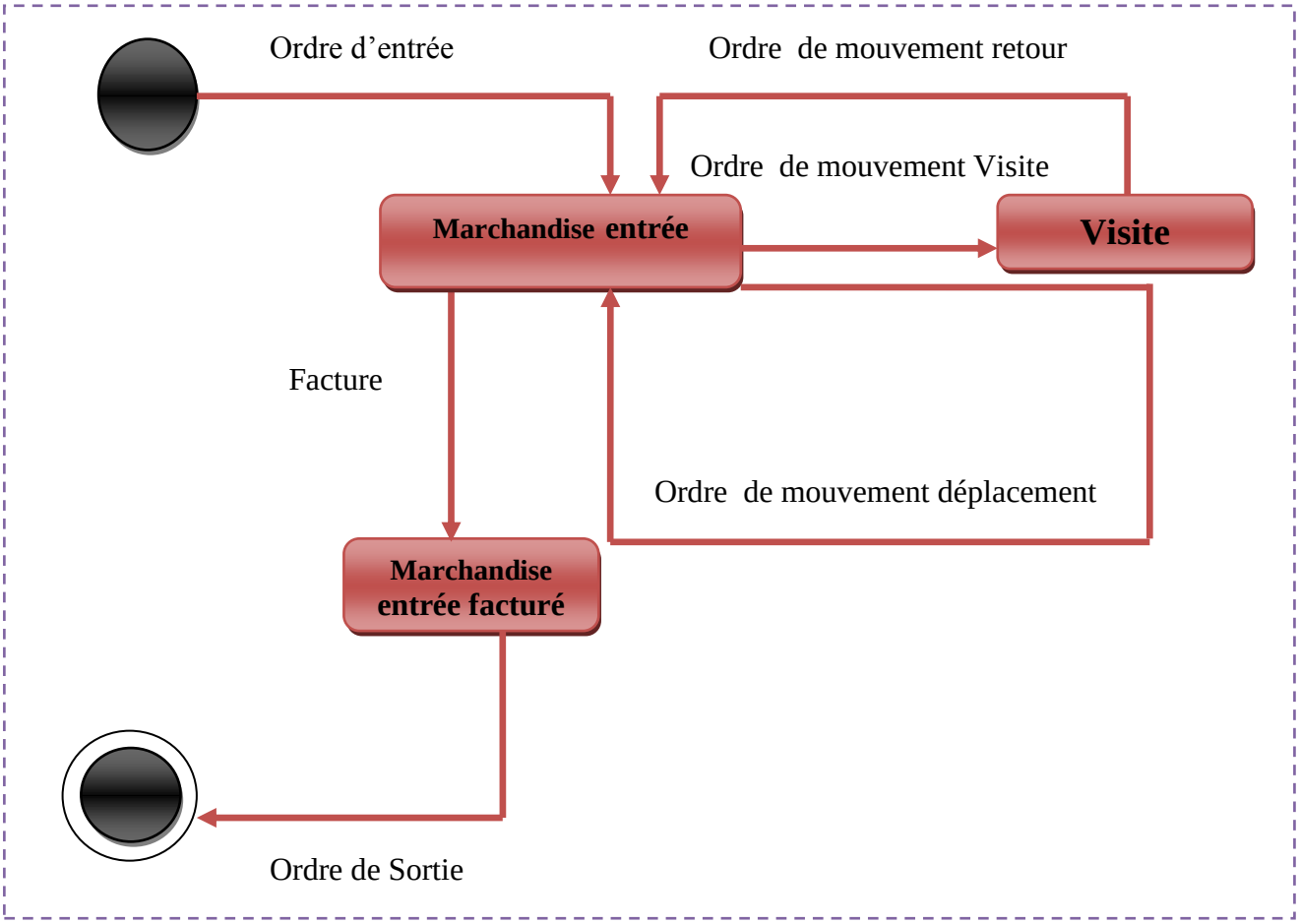


Figure II.1 : Diagramme d'état de transition d' une marchandise

2-Choix de l'approche

La norme ISO 9001:2000 s'intéresse aux processus de l'entreprise qui sont regroupés entre eux et nécessaires pour aboutir à la satisfaction des clients tout en respectant leurs exigences. L'évolution majeure de la version 2000 de la série des normes ISO 9000 concerne le management de l'organisme avec l'approche processus. Pour aller vers la satisfaction du client, au lieu de l'assurance qualité, c'est l'amélioration continue qui est avant tout visée. En prenant en compte les retours d'expérience du personnel, en intégrant les évolutions du marché et de tout l'environnement, ce concept permet aux entreprises d'améliorer la qualité de la prestation mais aussi de l'ensemble du système de management. Les conséquences de cette nouvelle approche sont importantes : le système qualité de l'entreprise est remis en question sur le fond et aussi sur la forme. Une mise à jour des connaissances et une sensibilisation à la norme sont obligatoires et cinq catégories d'exigences désormais sont inscrites :

- Responsabilités de la direction.
- Management des ressources.
- Réalisation du produit et/ou du service.
- Mesure, analyse et amélioration.
- Organisation du système de management de la qualité.

Cette norme encourage l'adoption de l'approche processus lors du développement, de la mise en œuvre et de l'amélioration de l'efficacité d'un système de management de la qualité, afin d'accroitre la satisfaction des clients par le respect de leurs exigences. D'une façon synthétique, l'approche processus a pour principaux objectifs :

- de mettre le client (interne ou externe) au centre de l'approche,
- de remodeler l'organisation avec une approche processus et non plus service,
- de permettre de mesurer l'activité des différents processus cartographiés,
- de piloter l'amélioration des processus suivis.
- de mettre en place les points de contrôle nécessaire au suivi.
- de mettre en place et de faire vivre un contrat d'objectif.
- de travailler à l'amélioration des relations entre les services. **[2]**

Le choix de cette approche réside aussi dans le retour d'expérience aux différents projets de l'entreprise Marine Soft basent sur l'approche processus.

2.1-L'approche processus

L'approche processus est une méthode d'analyse ou de modélisation de l'entreprise qui permet de mieux maîtriser la qualité de ses produits et la satisfaction de ses clients. L'approche processus s'est généralisée comme outil de management puissant et fiable depuis la fin des années 80. Elle permet de décrire de façon méthodique une organisation ou une activité pour détecter les points faibles puis initier et suivre des actions d'amélioration. La version 2000 de la norme ISO 9001 impose l'approche processus aux entreprises souhaitant obtenir une certification qualité mais ne donne aucune directive sur comment le faire. **[2]**

2.2-Concepts de base

2.2.1-Processus

Toujours selon la norme ISO 9001, Un processus est un ensemble de **ressources** et d'**activités** liées qui transforment des éléments entrant en éléments sortant. **[3]**

2.2.2-Sous-processus

Un sous-processus est une étape du processus. Il s'agit d'une suite chronologique de tâches réalisées par des personnes ou des applications informatiques différentes à des moments et en des lieux différents. Cet ensemble de tâche a un début et une fin pour obtenir un résultat intermédiaire. **[3]**

2.2.3-Processus élémentaires

Ensemble de sous processus liés, qui dépendent directement ou indirectement des mêmes éléments d'entrée et produisant directement ou indirectement le même élément de sortie. **[3]**

2.3-Les familles de processus :

Il existe trois familles de processus :

2.3.1-Les processus de réalisation (opérationnels) :

Ensemble de processus allant du client au client. Ils permettent la réalisation du produit ou du service fourni par l'entreprise à ses clients et correspondent ainsi à l'activité «Métier» de l'organisation. Ces processus couvrent le cycle de vie du produit (service) et ont évidement un impact direct sur la satisfaction du client.

2.3.2-Les processus de support (soutien) :

Ensemble des processus donnant les ressources aux autres processus. Ils représentent une activité interne, généralement transversale, permettant d'assurer le bon fonctionnement de l'entreprise. Ils contribuent au succès des processus de réalisation, en leur fournissant les

moyens (ressources humaines, infrastructures, environnement de travail et information) relatifs à un bon déroulement.

2.3.3-Les processus de management (pilotage) :

Correspondent à la détermination d'une politique et d'une stratégie pour l'organisation et au pilotage des actions mises en œuvre pour atteindre ses objectifs. Ces processus sont sous la responsabilité de l'équipe dirigeante et agissent directement sur le fonctionnement de l'organisme et sur sa dynamique d'amélioration. Ils assurent la cohérence des processus de réalisation et support **[2]**

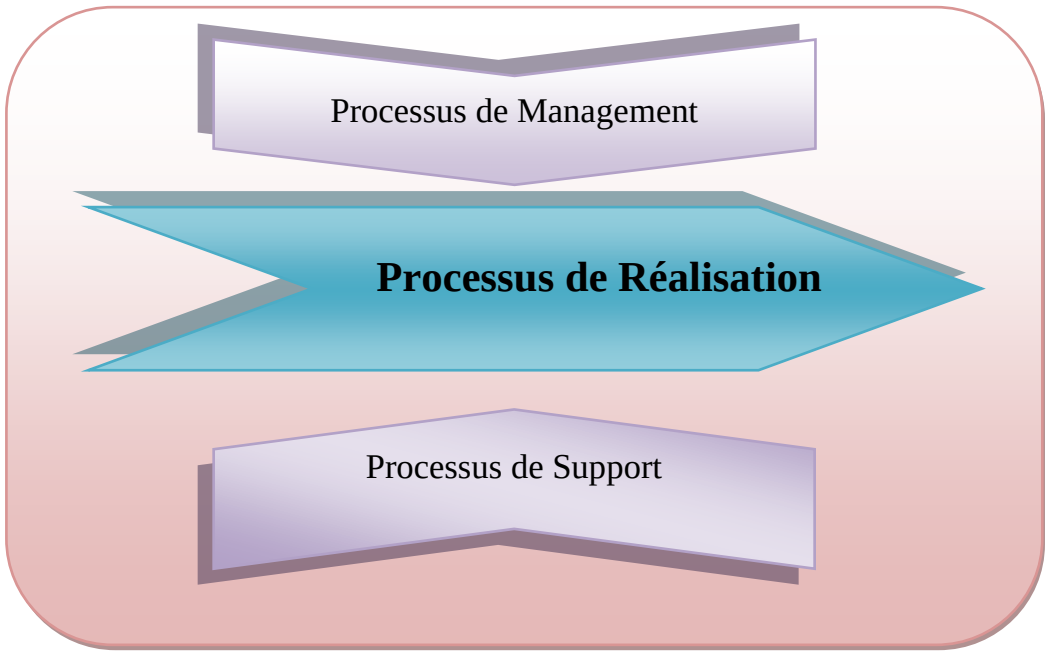


Figure II.2 : Disposition des processus dans un contexte de management qualité.

Remarque:

Contrairement au processus de réalisation, les processus de support et de management sortent du cadre de cette étude. C'est pour cette raison qu'il ne sera fait aucune référence à ces deux type de processus, au cours du reste de ce chapitre

2.4-Représentation normalisé d'un processus :

Un processus est caractérisé par :

- ✓ un nom.
- ✓ Un ou plusieurs (éléments d'entrée **EE**) : ressources nécessaires pour démarrer un processus. Ils proviennent d'un ou plusieurs fournisseurs externes ou internes (autres processus).
- ✓ Un ou plusieurs (éléments de sortie **ES**) : résultat de la transformation des éléments d'entrée par le processus. Ils sont destinés à un ou plusieurs clients externes ou internes (autres processus).
- ✓ Suite d'activité (processus élémentaires **PE**) qui transforme les éléments d'entrée en éléments de sortie en apportant une valeur ajoutée (VA). **[4]**

Sa représentation graphique peut se réaliser de la façon suivante :

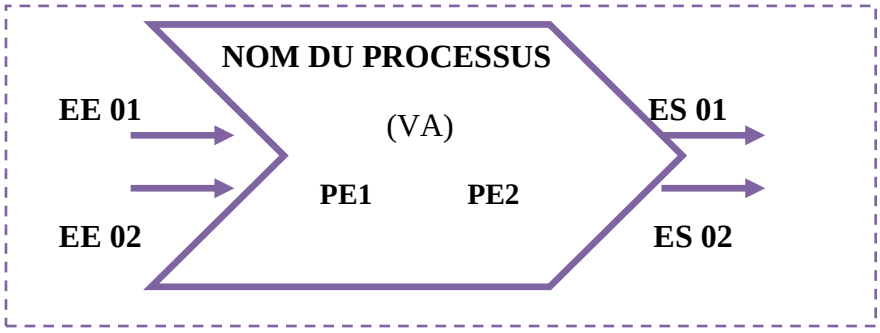


Figure II.3: Représentation normalisé d'un processus.

2.5-Représentation normalisé d'un ensemble de processus:

La phase d'identification des processus doit avoir un résultat concret, clair, facilement partageable, compris par un grand nombre d'acteurs, donc, qu'il est possible de soumettre à validation. Une simple liste commentée peut, théoriquement, suffire à exprimer le résultat de cette approche, mais une représentation schématique, simplificatrice et mémorisable convient mieux. On parle alors d'élaborer une cartographie des processus. **[2]**

La figure suivante est un exemple de ce que doit être en principe, une cartographie de processus de réalisations.

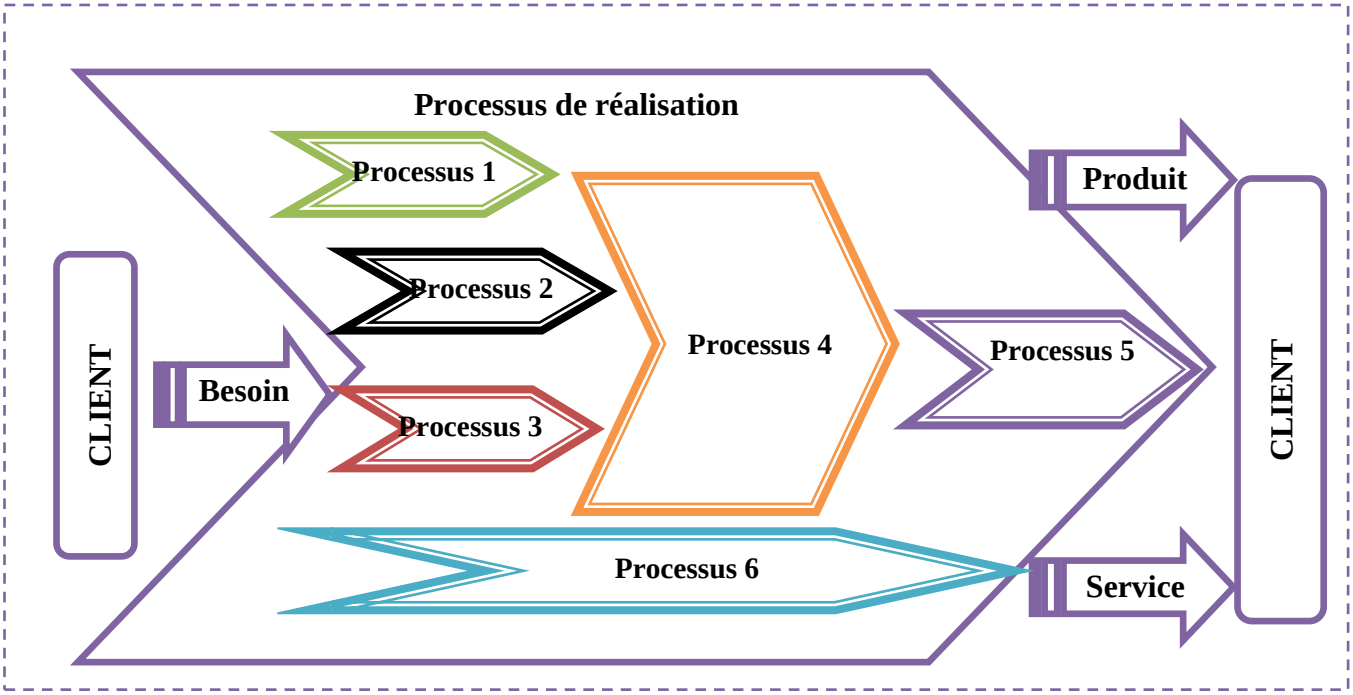


Figure II.4 : Composition d'une cartographie de processus de réalisation.

2.6-Méthodologie de l'approche processus.

L'approche processus est une approche systémique. Cela veut dire, entre autres, qu'il y aura plusieurs niveaux d'analyse. Ce qui est considéré comme ***processus*** à un niveau d'analyse « N » va devenir ***Sous-processus*** au niveau d'analyse « N+1 » et vice-versa.

La phase d'identification des processus a un caractère itératif. L'ajout ou la suppression d'un détail (élément) sur la cartographie de niveau « N », risque d'affecter celles de niveau « N+1 » et/ou « N-1 ». Ceci assure la cohérence entre les différents niveaux d'analyses. Toute fois il est possible de voir apparaitre sur la cartographie de niveau « N-1 » des éléments d'entrées ou de sorties (externe) qui n'apparaissent pas nécessairement sur celle de niveau « N » car tout simplement le niveau d'analyse ne permet pas de le voir.

Pour arriver à élaborer une cartographie de processus de n'importe quel niveau, il suffit de savoir comment faire celle du niveau « N-1 ». La méthode est simple et propose de procéder en quatre étapes.

Etape 1: Décrire l'organisme entier comme un macro-processus.

Il convient d'identifier, en premier lieu, le processus global au niveau le plus élevé. La cartographie qui en sortira porte le nom de Macro- processus et servira de référence pour les

étapes suivantes. Les caractéristiques du processus doivent toutes être mentionnées, notamment, les processus élémentaires (PE) qui correspondent aux activités de l'organisme. Pour plus de clarté, il est conseillé de regrouper les éléments d'entrées et sortie par provenance (P) et destination (D). La figure suivante est un récapitulatif de ce que doit être en principe la représentation d'un organisme sous forme de macro-processus.

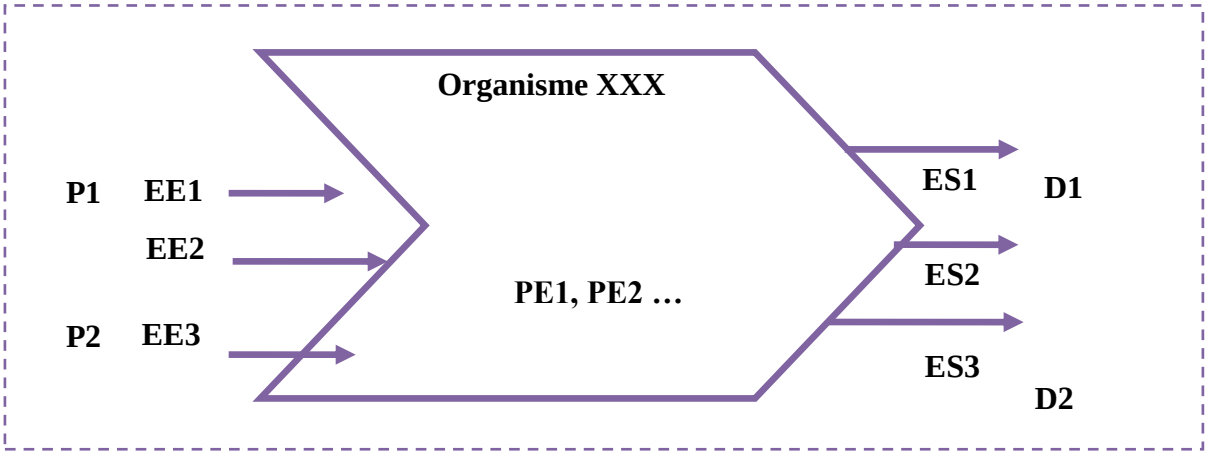


Figure II.5 : Représentation d'un organisme sous forme de macro-processus.

Etape 2: Décrire les processus qui prennent en charge les entrées du macro-processus.

Dans cette étape il s'agit d'identifier une partie des processus élémentaires, entre autre, ceux qui prennent en charge les éléments d'entrées du macro-processus. Pour cela, il est conseillé d'aller « sur le terrain » pour suivre, concrètement, auprès des acteurs concernés, qui prend en charge une entrée, quel traitement il effectue, quel est le résultat de ce traitement et où va le résultat de ce traitement. La figure suivante est le résultat de ***l'étape1*** appliquée sur l'exemple de la figure précédant.

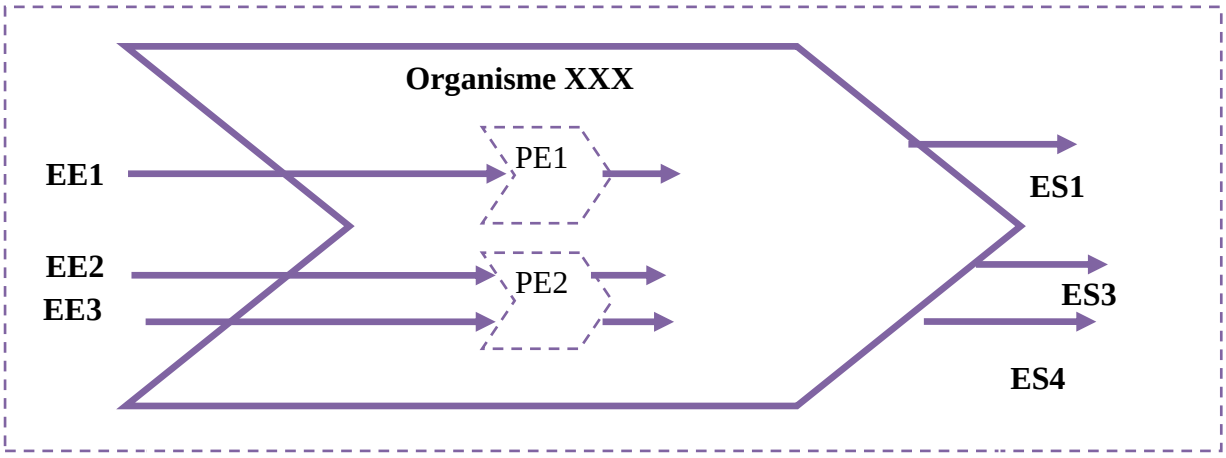


Figure II.6: Exemple de résultat de *l'étape 2*.

Etape 3: Décrire les processus qui génèrent les sorties du macro-processus.

À l'inverse de *l'étape 2*, il s'agit dans celle-ci d'identifier l'autre partie des processus élémentaires, ceux qui génèrent les éléments de sortie du macro-processus. Pour cela, il est conseillé d'aller «sur le terrain» pour remonter, concrètement, auprès des acteurs concernés, qui génère cette sortie, quels traitements il effectue, quelles entrées il prend en charge et quelles sont les origines de ces entrées. La *figure II.6* est le résultat de *l'étape2* appliquée sur l'exemple de la *figure II.6.*

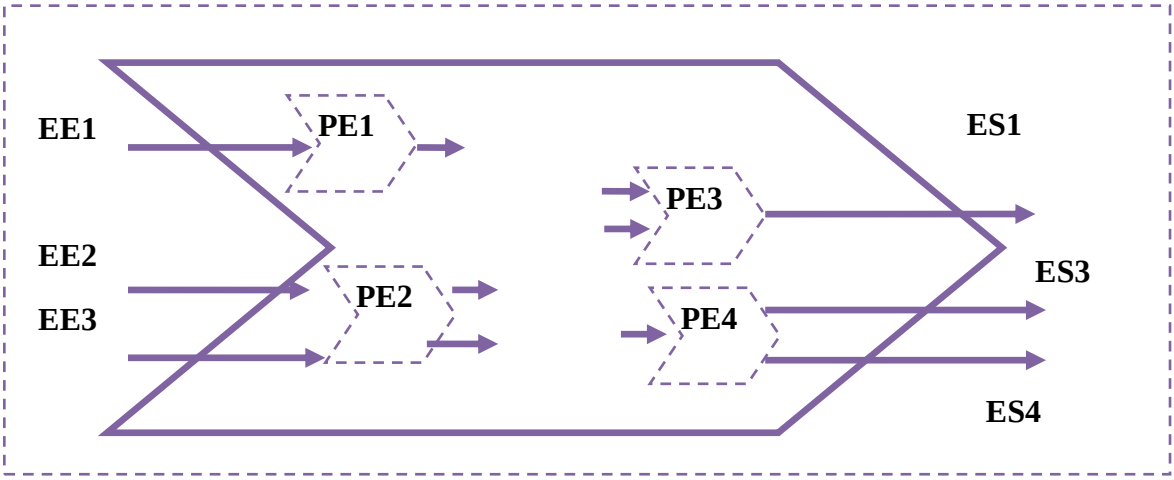


Figure II.7 : Exemple de résultat de *l'étape 3*

Etape 4 : Décrire les processus élémentaires qui manquent dans la chaîne.

Une fois le résultat de *l'étape 2* et celui de *l'étape 3* validé, des éléments de sortie des processus élémentaires issus de *l'étape 2*, ainsi que les éléments d'entrée des processus élémentaires issus de *l'étape 3*, auront était identifiés. Cette étape consiste, soit à prendre chaque élément de sortie interne et à décrire les processus qui le prennent en charge, ou bien, à prendre chaque élément d'entrée interne et à décrire les processus qui le génèrent. Il est fréquent de voir apparaitre des processus particuliers au cours de cette étape. Il s'agit de processus qui ne sont pas liés directement au processus issus des *étapes 1* et *2*, mais qui ont leurs importances.

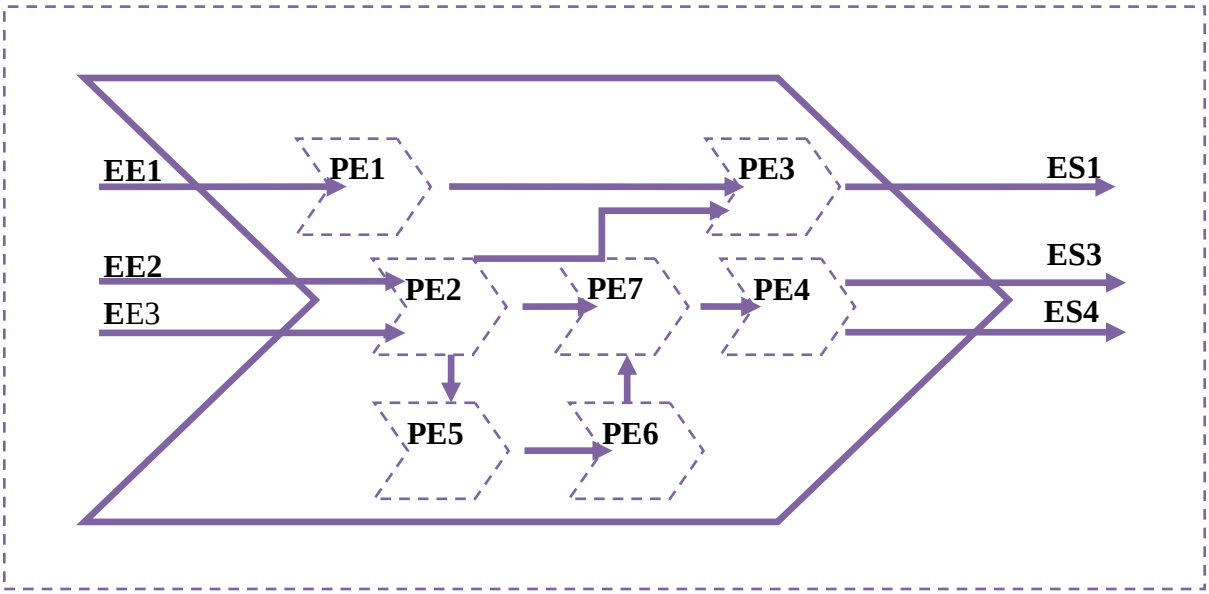


Figure II.8: Exemple de résultat de *l'étape 4*.

Au terme de ces quatre étapes, on aura élaboré la cartographie des processus de réalisation de niveau « N-1 ». Systématiquement, pour réaliser celle de niveau « N-2 » il suffit de prendre chaque processus élémentaire (PE1, PE2, …) à part et de réitérer les *étapes 2,3 et 4* en le considérant comme étant le macro-processus de *l'étape 1*.

Au final, on aura autant de cartographie, qu'il y a de processus élémentaires, qu'il suffira de regrouper dans un même document pour obtenir la cartographie de processus de réalisation de niveau « N-2». Il faut tout de même, prendre le soin de remettre chaque cartographie à la place du processus élémentaires qu'elle est supposée décrire.

Dans un contexte de management, trouver le bon niveau de détail au quel il convient de s'arrêter est délicat et exige une certaine maîtrise de l'approche. En revanche, dans le contexte d'une simple analyse et modélisation de processus, le bon niveau d'analyse serait celui jugé suffisamment détaillé pour répondre aux besoins de la modélisation.

Note :

Il est recommandé de veiller, constamment, à ce que le résultat, de chaque étape, soit validé par les acteurs concerné avant de passer à une autre. Demander avis aux acteurs, jugés susceptible d'apporter un plus à l'analyse ne ferra qu'enrichir son contenu.

3- Diagnostique du système informatique existant.

3.1-Présentation de ZSD 2013 V1.0.

ZSD®2013 V1.0 est la dernière version d'une gamme de produit spécialement conçu pour la gestion d'une zone d'entreposage de marchandise sous douane. Celui-ci repose sur une architecture fonctionnelle modulaire dont on distingue six modules prenant en charge le métier concerné, et quatre modules support.

Le schéma, présenté ci dessous, offre une vue d'ensemble des dix modules le constituant.

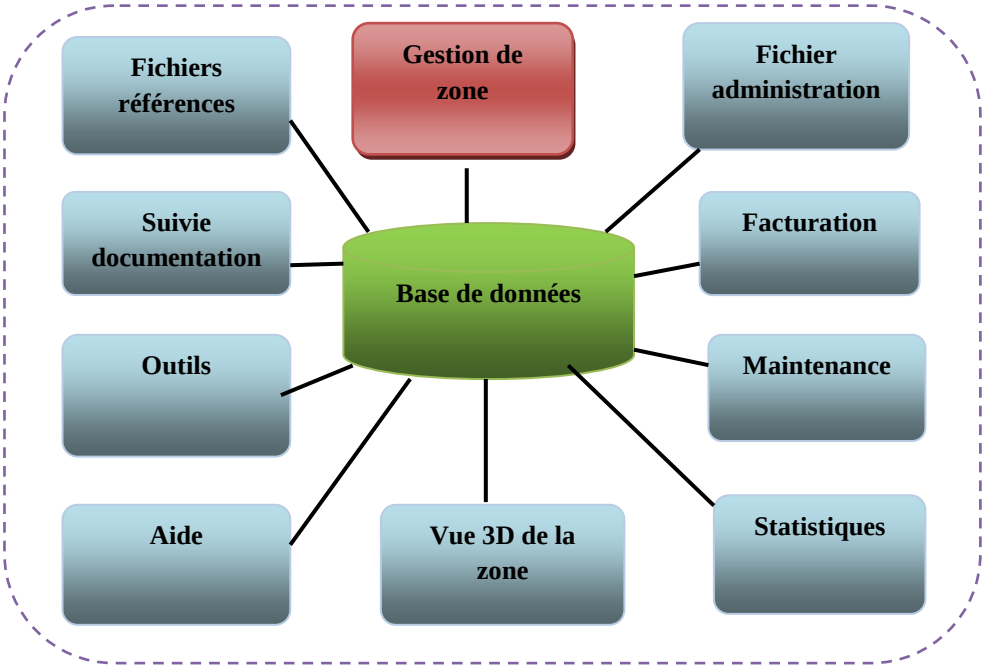


Figure II.9: Modules du progiciel ZSD®2013 V1.0

Ces modules sont accessibles par authentification via le menu principal dont est l'objet la figure présentée ci-dessous.

Figure II.10 : Menu principale du progiciel ZSD®2013 V1.0

3.2-Tableau descriptif de ZSD®2013 V1.0

Le tableau, présenté ci-dessous, illustre les différentes fonctions et fonctionnalités prise en charge par le système.

Module	Description
	✓ Accéder aux fichiers d'Administration. (Conventions, Rubrique Facturation) ✓ Affecter, enlever ou modifier une convention de payement ou de facturation ✓ Ajouter, supprimer, modifier une rubrique de facturation Editer une partie ou totalité des fichiers, sur écran, imprimé ou sur fichier.
	✓ Accéder aux fichiers de référence (Armateur, Agent consignataire, Navire, moyen de transport, conteneur, pays, port, réceptionnaire, rubrique de fabrication) ✓ Ajouter, supprimer, m-a-j un fichier de référence. Editer une partie ou totalité des fichiers, sur écran, imprimé ou sur fichier
	✓ Saisir ou modifier un manifeste.

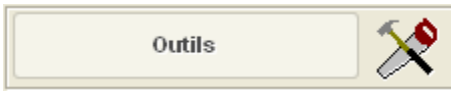
	✓ Ajouter, m-a-j, visualiser les escales et les articles ✓ Visualiser les conteneurs (par numéro de gros ou par année)
	✓ Vue globale du parc ✓ Gestion des entrées /sorties des conteneurs et leurs mouvements (saisie au clavier ou automatiquement en utilisant la sourie) ✓ Edition et réédition des bons de sortie et des ordres de mouvements. Recherche multi critères
	✓ Facturation (armateurs et réceptionnaires) ✓ Gestion des conventions Suivie de règlement
	✓ Visualiser la situation de la Zone avec une vue en trois dimension (en temps réelle).
	✓ Fonctions de maintenance et de sauvegarde des données
	✓ Fonctions de transfère et de consolidation des données
	✓ Statistiques concernant le Parc.
	✓ L'aide contextuel concernant le Progiciel **ZSD® 2013 V1.0**

Tableau II.2 : Tableau descriptif du progiciel ZSD® 2013 V1.0

3.3-Type d'architecture et la répartition du code du logiciel ZSD:

L'architecture du logiciel ZSD s'appuie sur le mode client/serveur (2tiers). Son code est réparti comme nous allons l'illustrer dans le schéma qui suit :

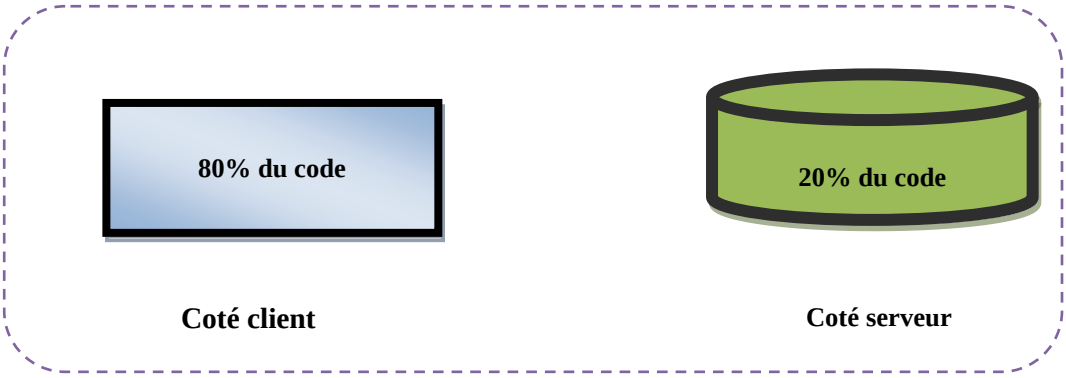


Figure II.11 : L'architecture et la répartition du code dans le logiciel ZSD

3.4-Analyse et modélisation des processus métiers.

Dans cette section, l'accent sera mis en œuvre de l'approche processus dans un contexte d'identification et modélisation de processus métier. Le but étant de décortiquer suffisamment le métier du point de vue fonctionnel et, parallèlement, récolter les données.

L'étude d'un cas particulier (entreprise) aurait figé notre conception. Pour des raisons de standardisation et d'ouverture et conformément aux recommandations de l'entreprise ,il a était convenu de procéder l'analyse de logiciel **ZSD® 2013 V1.0** comme échantillon d'étude pour voir en détail les activités assuré dans une zone d'entreposage par ce logiciel , et suivre concrètement le cheminement des activités relatives aux différents processus qu'il faudra identifier et décrire.

3.4.1-Macro-processus de réalisation d'une zone d'entreposage de marchandises sous douane :

Le processus d'entreposage d'une marchandise débute a l'arrivé de l'un des documents qui comporte toutes les informations relative à l'identité de la marchandise allant de son fournisseur jusque à ses réceptionnaires. Ce document atteste que cette marchandise est destinée à être entreposé dans cette zone. Ce processus touche à sa fin lors que le gestionnaire de la zone délivre un document attestant que cette marchandise est autorisée à être enlevée de la zone

La figure suivante englobe l'activité d'entreposage de marchandise sous douane sous forme de Macro-processus de réalisation.

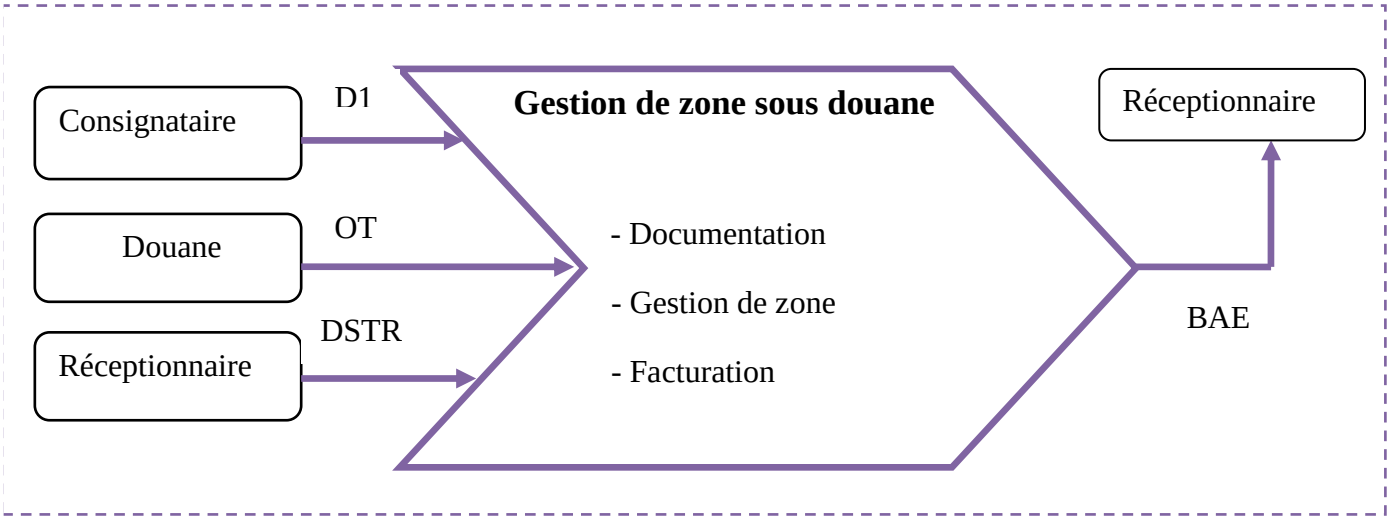


Figure II.12-Macro-processus de réalisation d' une zone d'entreposage de marchandise sous douane

Les éléments d'entrée du macro-processus sont des documents qui servent à la même chose mais qui ont des prévenances, des destinations, et des régimes différents. Le tableau suivant met en évidence ces différences :

Document	Prévenance	Destination	Régime
DI	Agence consignataire	Port sec	Liste des marchandises, ayant été transportées sur le même navire, ayant débarquées au même port et dont la totalité est destinée à être transférée, sous la demande de l'agence consignataire, du port vers un port sec.
OT	Douane	Port sec Entrepôt Privé Entrepôt public	Liste des marchandises, ayant étés transportées sur le même navire, ayant débarquées au même port et dont la totalité ou une partie, seulement est destinée à être transférées sous l'ordre d'un bureau de douane, du port vers un

			port sec ou un entrepôt.
DSTR	Réceptionnaire	Entrepôt Privé Entrepôt public	Liste des marchandises du même réceptionnaire destinées à être transférées, sous la demande du réceptionnaire, du port ou du port sec vers un entrepôt. Dans ce cas c'est le réceptionnaire qui S'occupe du transport terrestre et des formalités douanières et autres.

Tableau II.3 : Eléments entrées du processus élémentaire « Documentation»

Etape 1 :

La cartographie, de niveau « N» (figure II.11) correspond au résultat de l'étape « 1 » (figure II.4) de la méthode de la modalisation des processus de réalisation. Les étapes 2,3 et 4 (figure II.5, II.6 et II.7) ont permettent de passer au niveau « N-1 ».

3.4.2-Cartographie des processus de réalisation «N-1 »

Au cours de l'élaboration de cette cartographie (figure II.12) les étapes 2,3et 4 (figure II.5, II.6 et II.7) ont permis chacune d'identifier un processus élémentaire.

Etape 2 :

Les éléments d'entrée du macro-processus sont prises en charge par le processus « Documentation».Il consiste en la saisie de toutes les informations qui contiennent ces documents (DI, OT, DSTR). Les éléments de sortie de ce processus sont les mêmes informations réorganisées et reprise sur un support numérique de sorte qu'elles puissent être accessibles par un système informatique.

Etape 3 :

L'élément de sortie du macro-processus est généré par le processus « Gestion de zone ».Il se traduit en l'entrée et sortie de la marchandise ainsi que les éventuels mouvements de celle-ci à l'intérieure de la zone. Celui-ci débute à l'arrivée de la marchandise et se termine une fois que cette marchandise ait l'autorisation(BAE) de sortie .Il prend en entrée les éléments de sortie (information marchandise) du processus « Documentation ».ainsi que ceux (facture) du processus « Facturation ».

Etape 4 :

Le processus qui manque dans la chaine est « Facturation». Celui-ci se résume à la taxation des différentes prestations fournies par la zone sous forme de factures ainsi que le règlement de celle-ci. Ce processus prend en entrée les éléments de sortie des deux autres processus et généré un élément de sortie (facture) qui fait partie des éléments d'entrée du processus« Gestion de zone ».

Le résultat de ces étapes de modélisation de processus de réalisation est synthétisé sur la cartographie suivante :

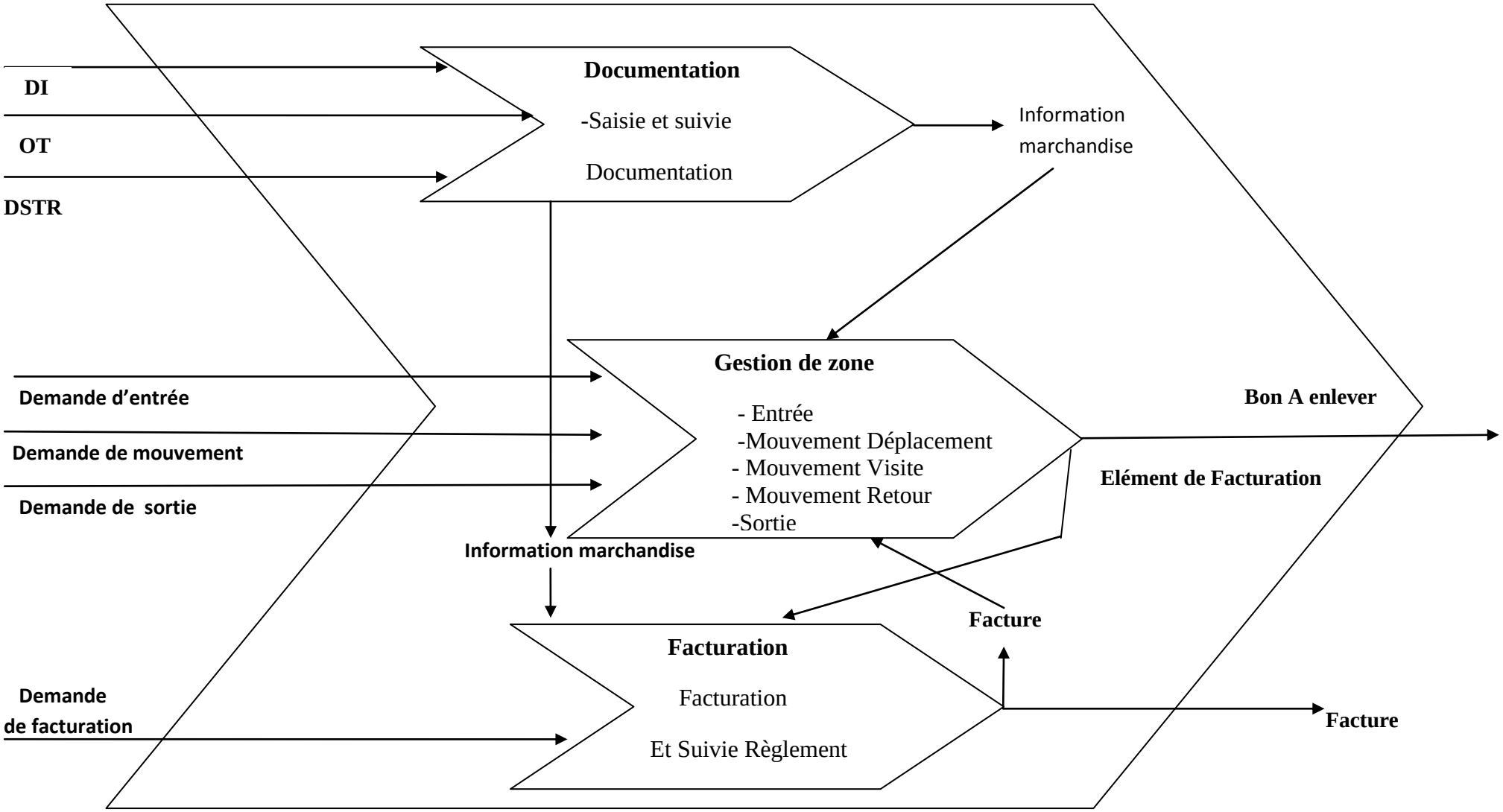


Figure II.13 : Cartographie des processus de réalisation lié à la gestion de zone de niveau « N-1 ».

Remarque

Il est à noté que dans la cartographie précédente (figure II.12) la présence d'éléments d'entrées (Demande d'entrée, Demande de mouvement, Demande de facturation, Demande de sortie) n'apparaissent pas sur le Macro-processus. Ceux-ci ont la particularité d'être, dans ce cas des éléments déclencheurs et séquentiels.

3.4.3- Délimitation du champ d'étude :

L'entreprise marine soft travaille sur un ERP GLS, et des équipes ont bossé sur les différents modules de ce dernier, chacun d'eux à procéder à la modélisation des processus de réalisation, selon leur métier. Le résultat a procédé vient conformer la démarche de séparation des fonctions propres au métier de celles standards ou communes. En effet l'entreprise a mis en évidence un processus standard (facturation) qui est identique sur l'ensemble des cartographies et un processus dont la valeur ajouté est commune (saisie de la marchandise) mais qui prend des noms différents sur chaque cartographie (**«**documentation**»** dans les zones sous douane).

Sur la deuxième cartographie, on distingue clairement trois processus élémentaires, seul le processus **«**gestion de zone**»** est propre au métier de la gestion d'activités dans les zones adressables et non adressables. Cette séparation est conforme au objectifs de ce projet est limite le champ d'étude au périmètre du processus métier «gestion de zone»

3.4.4-Cartographie des processus de réalisation de niveau «N-2»

Conformément au périmètre du champ d'étude, cette cartographie se limitera à nous décrire que les sous processus lié à la gestion de zone.

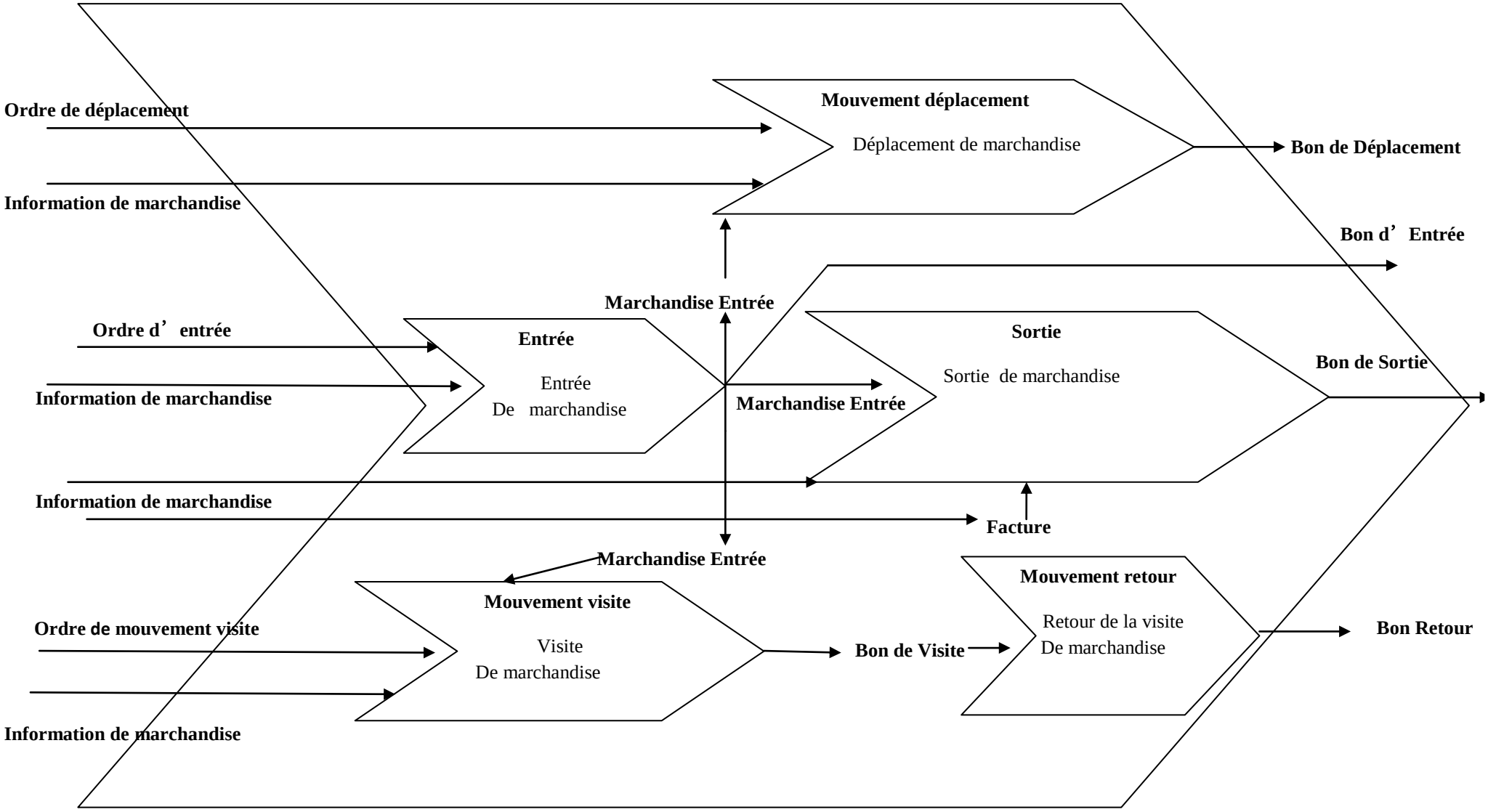


Figure II.14 : Cartographie des processus de réalisation lié à la gestion de zone de niveau « N-2 ».

3.4.4.1-Sous processus « Entrée ».

Elément d'entrée	Origine	Description
Ordre d'entrée (déclencheur)	Réceptionnaire Transporteur	Demande d'entrée d'une marchandise
Information de marchandise	PE : Documentation	Nécessaire à l'identification et contrôle de la marchandise

Tableau II.4 : Elément d'entrée du sous processus « Mouvement Entrée »

Elément de sortie	destination	Description
Bon d'entrée	**SP :** Mouvement Déplacement **SP :** Mouvement Visite **SP :** Mouvement Sortie	Un bon d'entrée confirme que la marchandise est à l'état « Entrée »

Tableau II.5 : Elément de sortie du sous processus « Mouvement Entrée »

3.4.4.2- Sous processus «Mouvement Visite»

Elément d'entrée	Origine	Description
Ordre de mouvement visite	Réceptionnaire	Demande visite d'une marchandise
Information marchandise	PE : documentation	Pour l'identification de marchandise
Marchandise entrée (Condition)	SP : Entrée	Pour pouvoir effectuer le mouvement visite d'une marchandise il faut que celle-ci soit dans l'état « Entrée »

Tableau II.6 : Elément d'entrée du sous processus « Mouvement Visite»

Elément de sortie	destination	Description
BV	**SP :** Mouvement retour	Le bon de visite est un document contenant les informations suivantes : Identification de la

		marchandise, Désignation, Identification de la zone de visite, date de mouvement visite, adresse d'emplacement de marchandise.

Tableau II.7: Elément de sortie du sous processus « Mouvement Visite »

3.4.4.3- Sous processus «Mouvement Retour»

Elément d'entrée	Origine	Description
BV (condition)	SP : Mouvement visite	Pour pouvoir effectuer le mouvement retour de visite d'une marchandise il faut que celle-ci soit dans l'état « visite »

Tableau II.8 : Elément d'entrée du sous processus « Mouvement Retour »

Elément de sortie	destination	Description
BR	Réceptionnaire	Le bon de mouvement retour est un document contenant les informations suivantes : Identification de la marchandise, Désignation, Identification de la zone de visite, date de mouvement retour.

Tableau II.9: Elément de sortie du sous processus « Mouvement Retour »

3.4.4.4- Sous processus «Sortie»

Elément d'entrée	Origine	Description
Information marchandise	PE : Documentation	Pour l'identification de la marchandise
Marchandise entrée	SP : Entrée	Pour pouvoir effectuer la

(Condition)		sortie d'une marchandise il faut que celle-ci soit dans l'état « Entrée »
Facteur (Condition)	Réceptionnaire	Atteste que la marchandise a était facturé et que le réceptionnaire est en mesure de la faire sortir

Tableau II.10 : Elément d'entrée du sous processus « Mouvement Sortie »

Elément de sortie	destination	Description
BAE	Réceptionnaire	Atteste qu'une marchandise a l'autorisation de sortie de la zone

Tableau II.11: Elément de sortie du sous processus « Mouvement Sortie »

3.4.4.5- Sous processus «Mouvement déplacement »

Elément d'entrée	Origine	Description
Ordre de déplacement	SP : Réceptionnaire	Demande de déplacement d'une marchandise d'un emplacement à un autre
Information marchandise	PE : documentation	Pour l'identification de marchandise
Marchandise entrée	SP : Entrée.	Pour pouvoir effectuer le mouvement déplacement d'une marchandise il faut que celle-ci soit dans l'état « Entrée »

Tableau II.12: Elément d'entrée du sous processus «Mouvement déplacement»

Elément de sortie	destination	Description
BD	Réceptionnaire	Le bon de déplacement document contenant les informations suivantes : Identification de la marchandise, Désignation, date de mouvement, emplacement initiale de marchandise, emplacement récent de la marchandise.

Tableau II.13: Elément de sortie du sous processus « Mouvement déplacement »

4- Les imperfections du système ZSD :

A l'état actuel, le système ZSD souffre de deux principales imperfections, à savoir :

- **Critique1 : Contraintes de l'architecture 2 tiers.**

L'architecture client-serveur (2-tiers) est le type d'architecture sur le quel se base l'application ZSD ainsi la pluparts des produits logiciels présenté par Marine Soft. Mais cette architecture présente certaines limites et on trouve à l' origine de ces limites la localisation de la logique applicative sur le poste client qui complique la maintenance et la réutilisation de code, en effet à chaque mise à jour de ZSD il faudra passer par tous les postes clients afin de les mettre à jour.

- **Critique2 : Absence de certaines fonctionnalités.**

- ZDS n'assure pas les gestions des grilles(les entrées, les sorties).
- ZDS ne gère pas les déplacements inter zones, ni les déplacements par groupes.
- La gestion d'affectation des moyens dans les zones n'est pas assurée par ZSD.

5-Présentation de la solution informatique.

Comme nous l'avons cité dans le chapitre I (Présentation générale), Marine Soft doit suivre l'évolution technologique, les tendances et les normes, il en va sa survie. La normalisation, l'intégration, la réutilisation, l'autonomie…etc. qui va pallier au problème de l'ensemble du système informatique. Nous citerons les suggestions suivantes:

- **Suggestion 1 : Passer à une architecture 3 tiers de type J2EE :**

Pour pallier aux limitations des architectures 2 tiers l'entreprise veux faire une transition vers la notion « **3Tiers** ». Afin de rendre plus opérationnelle et plus performante l'architecture logicielle des applications.

- **Suggestion 2 : Réalisation d'un outil récent:**

Le but de ce travail est de réaliser un nouvel outil qui gère les activités liées à la marchandise dans les aires de stockage. Ainsi il permettra de pallier les manques de fonctionnalités détectés dans l'application ***ZSD*** pour satisfaire les besoins du client d'une part et les besoins de l'entreprise d'une autre part.

L'IHM de notre application doit être entièrement paramétrable pour s'adapter à chaque activité de n'importe quelle application métier. Cette application doit également prendre en charge les fonctions suivantes :

1. Paramétrer un parc.
2. Importer des données à partir d'un fichier pour paramétrer le parc
3. Visualiser un parc.
4. Zoomer le parc.
5. Accorder des couleurs aux emplacements.
6. lister les emplacements vides et occupés.
7. Rechercher les coordonnées d'une position.
8. Rechercher le contenu d'une position dans le parc.
9. Effectuer un mouvement entré, sortie, déplacement (dans la zone, inter zone ou inter parc).
10. Effectuer un mouvement entré, sortie, déplacement (dans la zone, inter zone ou inter parc) par groupe.
11. Recherche multicritère.
12. Les états d'éditions.

Note :

L'analyse de l'application ZSD nous a mené à percevoir les anomalies présentes par ce dernier, et la solution proposé aura une grande portée, elle sera destiné à n'importe quel logiciel de gestion de zone quelque soit le métier.

❖ **Conclusion**

L'objectif de cette étape était de décortiquer, progressivement et de manière métrisée, le métier de la gestion d'activités de marchandise (entrée, sortie, déplacement) dan les aires de stockages (zone adressable et non adressable), en utilisant l'approche processus comme méthode d'analyse. Le choix de cette approche s'est révéler être le bon. La mis en ouvre de l'approche à effectivement, permit d'analyser convenablement les processus métiers et comprendre les interactions et les activités. A ce stade le périmètre fonctionnel est identifié. La prochaine étape, rédaction des spécifications fonctionnelles détaillées, fera office de point de transition de la phase d'analyse à la phase de conception.

Partie III

Conception

Chapitre IV : Conception

- *Identification des acteurs.*
- *Spécifications des cas d'utilisation*
- *Spécifications des diagrammes de séquences*
- *Diagramme de classe.*
- *Conception de la base de données.*

Introduction

Après avoir analysé le système actuelle (ZSD) de l'entreprise Marine Soft, et faire sortir les anomalies citées précédemment, ce présent chapitre va être comme reflex de solution recherche, ainsi que le canal qui relie l'analyse et la conception.

La spécification fonctionnelle détaillée nous permettra de tirer les services qui vont être offerts par le nouveau système.

1. Paramétrer et constituer un parc.

Description	La fonction de paramétrage et constitution d'un parc permet de créer ou de faire la mis à jour du parc, ainsi que les zones qui le constitue.
Entrées	Pour paramétrer le parc on doit avoir les informations liées au : -**Parc** : nom de la société, code du parc, nom du parc, l'adresse du parc. -**Zone** : code du parc, code de la zone, type de la zone (adressable, non adressable ou grille), dimension de la zone, désignation de la zone, les adresses d'emplacements X_i de zone adressables (telle que i représente les différents propriétés de l'adresse X qui varie de 1 à N), les espaces entre ces adresses.
Traitement	-Contrôler la syntaxe : code de parc, code de zone. ✍ Les codes doivent être un alphanumérique. -Contrôler l'existence : Si le code du parc, ne fait pas partie de la liste des codes des parcs, Déjà crées. ✍ Enregistrer les informations liées au parc ainsi que les zones, que le constituent pour le créer. Sinon ✍ Enregistrer les informations liées au parc ainsi que les zones que le constituent pour faire une mise à jour du parc.
Sortie	La liste des parcs, ainsi les zones que le constituent.

2. Importer des données à partir d'un fichier pour paramétrer le parc

Description	Cette fonction permet la mise en service des informations liées au parc sous la forme d'un fichier afin de le paramétrer.
Entrées	En général, le fichier de référence contient les informations suivantes : -**Parc** : nom de la société, code du parc, nom du parc, l'adresse du parc. -**Zone** : code du parc, code de la zone, type de la zone (adressable, non adressable ou grille), désignation de la zone, les adresses d'emplacements X_i de zone adressables (telle que i représente les différents propriétés de l'adresse X qui varie entre 1 à N), les espaces entre ces adresses.
Traitement	-Vérifier la compatibilité de l'extension du fichier. -Récupérer le contenu de ce fichier. -Contrôler la syntaxe : les codes (parc, zone) ✍ Les codes doivent êtres alphanumériques. -Contrôler l'existence: Si le code du parc, ne fait pas partie de la liste des codes des parcs déjà créer. ✍ Enregistres les informations liées au parc ainsi les zones, que le constituent pour le créer. Sinon ✍ Enregistres les informations liées au parc ainsi les zones, que le constitue pour faire une mise à jour du parc.
Sortie	La liste des parcs, ainsi les zones qu'ils constituent

3 -Visualiser un parc.

Description	Cette fonction permet donner une vue sur l'état de parc (zones et grilles).
Entrées	Liste des adresses des emplacements dans le parc.
Traitement	✓ Récupérer le nombre d'emplacements dans les zones adressables, le nombre des zones non adressables, le nombre des grilles via les tables appropries. ✓ Récupérer les états d'emplacements (vide ou occupé) via la table approprie. ✓ Représenter les données récupérer par des formes géométriques (des rectangles et des éclipses). ✓ Afficher les formes a l'écran à l'aide d'un système de coordonnées qu'utilise le pixel comme unité de mesure.
Sortie	Aperçu de parc

4. Zoomer le parc

Description	Cette fonction permet la prise de vue durant laquelle l'opérateur fait varie la focale de l'objectif.
Entrées	Une vue de parc
Traitement	-redimensionner les pixels de la vue de parc.
Sortie	Une vue de parc agrandie ou réduite.

5. Accorder des couleurs aux emplacements.

Description	Cette fonction permet d'attribuer les couleurs aux emplacements d'entreposages selon leurs états (occupé, vide).
Entrées	-Code de parc. -Code la zone
Traitement	-Récupérer les adresses d'emplacements X_i de la zone adressable à partir d'un accès à la table approprié. -Contrôler la disponibilité d'emplacements : si l'adresse d'emplacement X_i est occupée par une marchandise alors l'adresse va avoir une couleur sinon l'adresse d'emplacement X_i va avoir une autre couleur.
Sortie	Les adresses d'emplacements d'entreposages dans la zone adressable coloris selon leurs états (occupé ou vide)

6. Lister les emplacements vides ou occupés.

Description	Cette fonction permet de récupérer la liste ainsi le nombre des adresses d'emplacements disponibles ou occupés dans une zone adressable.
Entrées	- Code de parc. -Code de la zone
Traitement	-Contrôler la syntaxe : code de la zone. ✍ Le code doit être un alphanumérique. -Récupérer la liste de toutes les adresses d'emplacements par zones adressable à partir de la table approprié. -Contrôler la disponibilité des emplacements : si l'adresse d'emplacement X_i est vide alors : ✍ On utilise un compteur V qui est initialement à 0 pour recenser le

	nombre d'emplacement vide existé dans une zone adressable, donc V=V+1. ✍ L'adresse X_i est ajoutée à la liste des emplacements vide. Sinon ✍ On utilise un compteur P qui est initialement à 0 pour recenser le nombre d'emplacement occupés existé dans une zone adressable, donc P=P+1. ✍ L'adresse X_i est ajoutée à la liste des emplacements occupés.
Sortie	La liste et le nombre de tous les adresses d'emplacements vides ou occupés existés dans une zone adressable

7. Recherche les coordonnées d'une position

Description	Cette fonction sert à déterminer l'adresse d'une position dans la zone adressable.
Entrées	- Code d'entrée marchandise.
Traitement	-Contrôler la syntaxe : le code d'entrée de marchandise. ✍ Le code d'entrée de la marchandise doit être alphanumérique. -Contrôler l''existance : le numéro de la marchandise saisie doit être existé dans la table approprié. -Récupérer les coordonnés de position de la marchandise à l'aide de son numéro.
Sortie	L'adresse de la position choisit.

8. Déterminer le contenu d'une position

Description	Cette fonction permet de préciser ce qu'existe au sein d'une position sélectionner
Entrées	

	Code d'entrée de la marchandise.
Traitement	-Contrôler la syntaxe : Code d'entrée de la marchandise. ✍ Le code doit être alphanumérique. -Contrôler l'existence : le code d'entrée de la marchandise doit être présent dans la table appropriée. -Récupérer le contenu à l'aide d'une passerelle de communication avec le métier, c'est une Méthode qui prendre en paramètre le code d'entré de marchandise et retourne le contenu.
Sortie	Le contenu de l'emplacement choisit.

9. Effectuer un mouvement entré, sortie, déplacement (dans la zone, inter zone ou inter parc).

Description	Cette fonction permet de procéder aux mouvements entrée, sortie ou déplacement (dans la zone inter zone ou inter parc) d'une marchandise**.**
Entrées	-Numéro d'entrée de la marchandise. -Position futur d'emplacement X_i, ou l'adresse de la grille. -Engin.
Traitement	-Contrôler la syntaxe : le numéro de la marchandise. ✍ Le numéro doit être un Numérique. -Contrôler l'autorisation de mouvement : ✍ **Si le mouvement est « Entres»** : le Numéro de la marchandise doit faire partie d'une liste des marchandises qui sont pas encore entrés. ✍ **Si le mouvement est « Sortie» ou « déplacement»** : le Numéro de la marchandise doit faire partie d'une liste des marchandises qui sont entrés, et non sorti. -Contrôler l'existence : les Numéro de la marchandise doit être présent soit à la grille soit à une adresse X_i, selon le type de mouvement, c'est-à-dire : ✍ si la marchandise est présentée dans la grille on effectue un mouvement **« Entres»**

	✍ sinon si la marchandise est présentée dans l'emplacement X_i on effectue un mouvement **« Sortie»** ou **« déplacement»**. -Contrôler la disponibilité de la Position futur d'emplacement X_i, (position de destination). -Enregistrer le mouvement selon le type de mouvement effectué. -attribuer les couleurs aux adresses d'emplacements X_i selon leurs nouveaux états (Accorder des couleurs aux emplacements Fonction définit avant). -Editer l'ordre de mouvement selon le type de mouvements effectué« Entrées» « Sortie» ou « déplacement ». (Les états d'éditions Fonction définit après).
Sortie	Ordre de mouvement entrée, sortie ou déplacement (dans la zone, inter zone ou inter parc) d'une marchandise.

10. Effectuer un mouvement entré, sortie, déplacement (dans la zone, inter zone ou inter parc) par groupe

Description	Cette fonction permet de procéder au mouvement entrée, sortie, ou déplacement (dans la zone inter zone ou inter parc) d'un groupe de marchandise**.**
Entrées	-Les Numéros d'entrées des marchandises. -Les positions futurs dans la zone, $X_{i,j}$ (j représente le nombre de marchandise sélectionnée) ou adresse de la grille. -Engin.
Traitement	- Contrôler la syntaxe : Numéros d'entrées des marchandises. ✍ Chaque numéro doit être un Numérique. -Contrôler l'autorisation de mouvement ✍ **Si le mouvement est « Entrées»** : les Numéros des marchandises doivent faire partie d'une liste des marchandises qui ne sont pas encore entrées. ✍ **Si le mouvement est « Sortie» ou « déplacement»** : les Numéros des marchandises doivent faire partie d'une liste des marchandises qui sont

	entrées, et non sorti. -Contrôler la possibilité du mouvement : ✍ **Si le mouvement est « Sortie» ou « déplacement»** : le groupe de marchandise sélectionné doivent être positionnés dans des adresses d'emplacements adjacents -Contrôler l'existence : les Numéros d'entrées des marchandises doivent être présents soit à la grille soit à des adresses d'emplacements adjacents selon le type de mouvement, c'est-à-dire : ✍ si le groupe de marchandises est présentée dans la grille on effectue un mouvement **« Entrées»** ✍ sinon si le groupe de marchandises est présentée dans des adresses d'emplacements adjacents on effectue un mouvement **« Sortie»** ou **« déplacement»**. -Contrôler la disponibilité des positions d'emplacements de destination c'est-à-dire : les adresses d'emplacements futurs doivent être vides et adjacents. -Enregistrer le mouvement selon le type de mouvement effectué. -Attribuer les couleurs aux adresses d'emplacements X_i selon leurs nouveaux états (Accorder des couleurs aux emplacements Fonction définit avant). -Editer l'ordre de mouvement selon le type de mouvement effectué« Entrées» « Sortie» ou « déplacement ». (Les états d'éditions Fonction définit après).
Sortie	Ordre de mouvement entrée, sortie ou déplacement (dans la zone, inter zone ou inter parc) de la marchandise par groupe.

11. Recherche multicritère.

Description	Elle permet de rechercher soit des informations sur les emplacements des marchandises, soit de rechercher des informations sur les différents mouvements de cette dernière, selon différents critères de recherche.
Entrées	-Numéro d'entrées de la marchandise -Liste des adresses d'emplacements vides ou occupés. -Le contenue d'une adresse d'emplacement -Date de mouvement -Type de mouvement (entrée, sortie, déplacement (dans la zone, inter zone, inter parc), déplacement par groupe (dans la zone, inter zone, inter parc)
Traitement	-Contrôler la syntaxe : numéro de marchandise ✍ Numéro de marchandise doit être numérique. -Contrôler l'existence : Vérifier l'existence de numéro de marchandise dans la table approprié -La date de mouvement doit être égale ou inférieure à la date de jour. -Récupérer les informations depuis les tables appropries
Sortie	Informations sur la recherche effectuée.

12. Les états d'éditions

Description	Elle permet l'édition des ordres des mouvements, on retrouve dans la plupart des fonctions, l'édition.au lieu de la détailler dans chaque partie, on a préféré la considérer comme étant une fonction, qu'on détaillera à part
Entrées	Les informations d'entres sont diverses, elles peuvent être soit : -Les informations de l'ordre du mouvement (entré ou sortie) d'une marchandise. -Les informations de l'ordre du positionnement de marchandise. -Les informations de l'ordre du mouvement (entré ou sortie) d'un groupe de

	marchandise. -Les informations de l'ordre du mouvement déplacé (dans la zone, inter zone, inter parc) d'une marchandise -Les informations de l'ordre du mouvement déplacé (dans la zone, inter zone, inter parc) d'un groupe de marchandise.
Traitement	-Récupérer les éléments à imprimé. -Générer l'état de sortie choisi (écran, imprimante, dans un fichier).
Sortie	Ordre de mouvement (entré, sortie, déplacer (dans la zone, inter zone ou inter parc)) d'une marchandise ou d'un groupe de marchandise.

Conclusion :

L'objectif de ce chapitre est de tracer le futur système par la présentation des fonctions qui seront utiles. Ce stade de détaille facilite la transition de la phase d'analyse à la phase de conception.

Partie IV

Réalisation

Chapitre V : Architecteur et Outils de développements

- *Architecteurs.*
- *Outils de développement.*
- *Serveur de base donnée*
- *Serveur d'application.*

Chapitre V I: Implémentation

- *Schéma applicatif de l'application.*
- *Présentation de quelque interface graphique.*

Introduction

La conception de toute solution logicielle doit être traitée avec précision et détail, précédée d'une analyse profonde et bien réfléchie, car elle est le reflet du futur système avant même de sa concrétisation. La conception est une phase qui repose sur un certain nombres de principes et de règles qui sont dictées par les orientations de la ou les méthodes utilisées. La principale avancée de ces dernières années, en termes des méthodes de conception, réside principalement dans celles orientées objet (POO). Elle consiste à définir des rubriques logicielles appelées objets, ainsi que de leurs interactions.

Dans le but d'avoir une meilleure analyse et de rendre la conception de notre application plus complète, nous avons adopté le langage UML qui permet de bien représenter l'aspect statique et dynamique d'une application par une série de diagrammes qu'il offre.

1- Présentation de l'UML [3]

UML *(Unified Modeling Language)*, que l'on peut traduire par « langage de modélisation unifié », est une notation permettant de modéliser un problème de façon standard. Ce langage est né de la fusion de trois de méthodes orientées objet :

- OOD de ***GRADY BOOCH.***
- OMT de ***JAMES RUMBAUGH.***
- OOSE de ***IVAR JACOBSON.***

En 1994, on recensait plus de 50 méthodologies orientées objets. C'est dans le but de remédier à cette dispersion que les « poids-lourds » de la méthodologie orientée objets ont entrepris de se regrouper autour d'un standard. Ils réussissent à mettre en place non pas une méthode de conception, mais un langage de modélisation standard sous la référence « UML »qui est normalisé par l'OMG en 1997.

Parmi les avantages d'UML :

- ✓ Etre indépendant des langages de programmation et être adapté à toutes les phases du développement.
- ✓ Il est conçu pour la visualisation, la spécification et la construction des systèmes logiciels.
- ✓ Langage utilisant une notation graphique, avec une syntaxe très riche, tout en restant intuitive.

C'est pour ces raison, et tant d'autre, que Marine Soft utilise ce langage et propose que ses stagiaire en fasse l'usage. En effet l'utilisation du même langage de modélisation permet une bonne communication.

2-Identification des acteurs

Un acteur représente un rôle que peut jouer l'utilisateur dans le système. Un acteur est schématiquement symbolisé par un personnage stylisé sur les diagrammes UML. Les acteurs de notre système sont :

- **Administrateur :** personne possédant les droits de gérer l'outil.
- **Utilisateur** : personne possédant les droits d'accès à toutes les fonctionnalités du système.

Note : un administrateur peut jouer le rôle d'un utilisateur.

3-Modélisation de l'aspect dynamique de la solution proposée

Cette procédure nous donnera la possibilité d'élaborer les diagrammes de (cas d'utilisation, de séquences et de classes) dans le but d'avoir les moyens d'exposer les services et les traitements de l'application.

Après la distinction des différents acteurs, ainsi que de différentes fonctions du système à concevoir durant le chapitre précédant, nous mettons en évidence :

- Les diagrammes de cas d'utilisation mis en œuvre par les différents acteurs du système.
- Les diagrammes de séquence, qui représente les scénarios de chaque cas d'utilisation graphiquement.
- Les diagrammes de classes, qui décrive les classes.

3.1-Représentation des diagrammes de cas d'utilisation

3.1.1- Définition d'un cas d'utilisation [3]

Un cas d'utilisation précise le comportement d'un système ou d'une partie d'un système et décrit un ensemble de séquences d'actions. Les cas d'utilisation servent à saisir le comportement attendu d'un système en cours de développement, sans avoir à préciser la façon dont ce comportement est réalisé. En UML, tous les comportements sont modélisés sous la forme de cas d'utilisation via un diagramme de cas d'utilisation

3.1.2- Identification des cas d'utilisation.

Acteur	Cas d'utilisation	Sous cas d'utilisation
Administrateur	S'authentifier	
	Paramétrer un parc	✓ Paramétrer le parc manuellement. ▪ Gestion de parc (ajouter, modifier, supprimer). ▪ Gestion de la zone (ajouter, modifier, supprimer). ▪ Gestion des emplacements de la zone adressable (ajouter, modifier, supprimer). ✓ Importer des données à partir d'un fichier pour paramétrer le parc.
Utilisateur	S'authentifier	
	Gestion des mouvements	✓ Mouvement simple. ▪ Editer l'ordre de mouvement simple. ▪ Visualiser le parc. ✓ Mouvement par groupe. ▪ Visualiser le parc. ▪ Editer l'ordre de mouvement par groupe.
	Visualiser le parc	✓ Voir le contenu d'une position. ✓ Voir les coordonnées d'une position. ✓ Zoomer le parc.
	Recherche multicritères	✓ Mouvement. ▪ Rééditer un ordre de mouvement. ▪ Editer la liste des mouvements. ✓ la marchandise. ▪ Editer la liste des marchandises. ✓ Les emplacements. ▪ Editer la liste des emplacements.

Tableau IV.1 : Représentation des cas d'utilisation

3.1.3-Définition d'un diagramme de cas d'utilisation [3]

Les diagrammes de cas d'utilisation permettent de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système. Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur (acteur) le voit. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système.

❖ **Les relations entre cas d'utilisation**

- **L'inclusion** « include » quand le cas source comprend le cas destination.
- **L'extension** « extends » quand le cas source ajoute son comportement au cas destination

3.1.4-Diagramme de cas d'utilisation de l'administrateur.

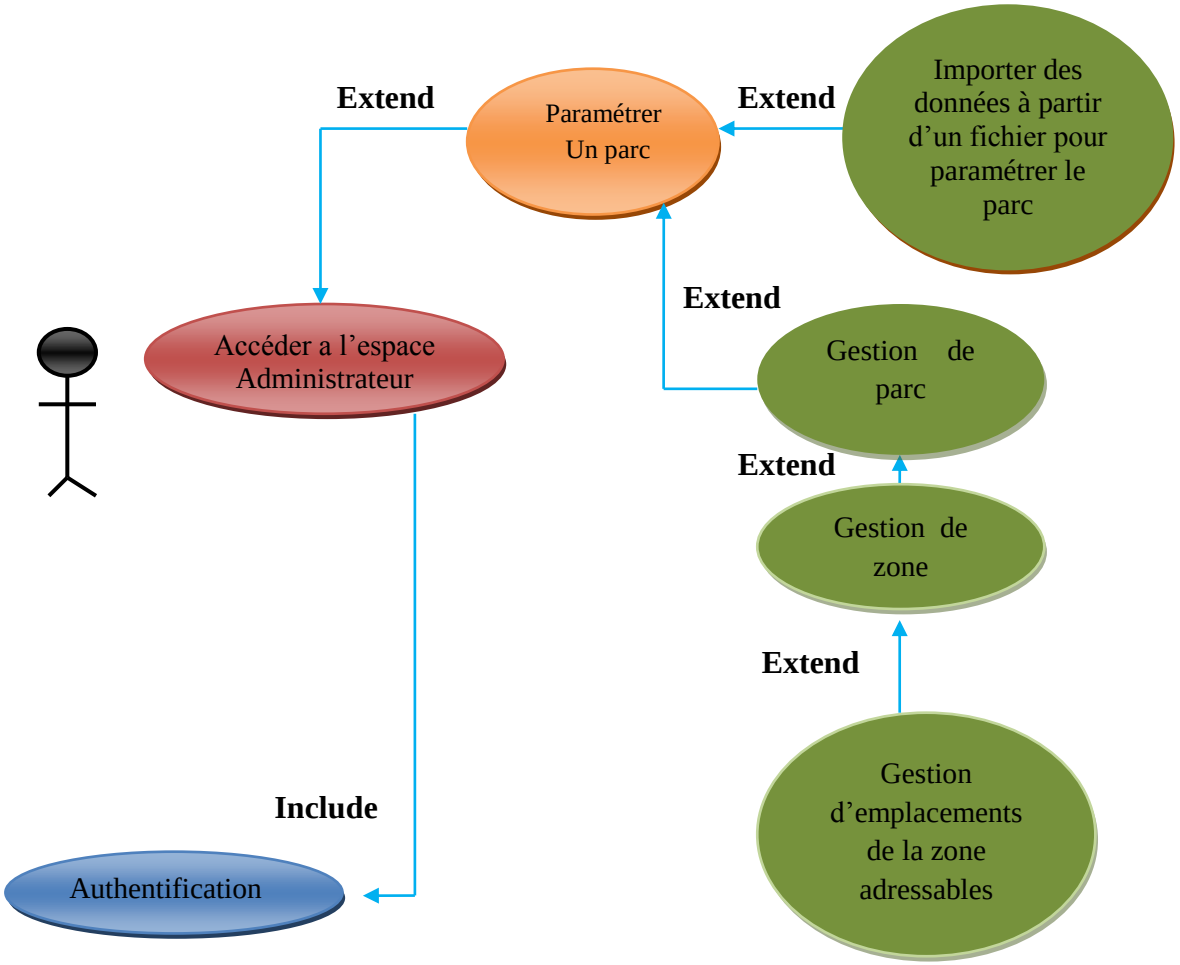
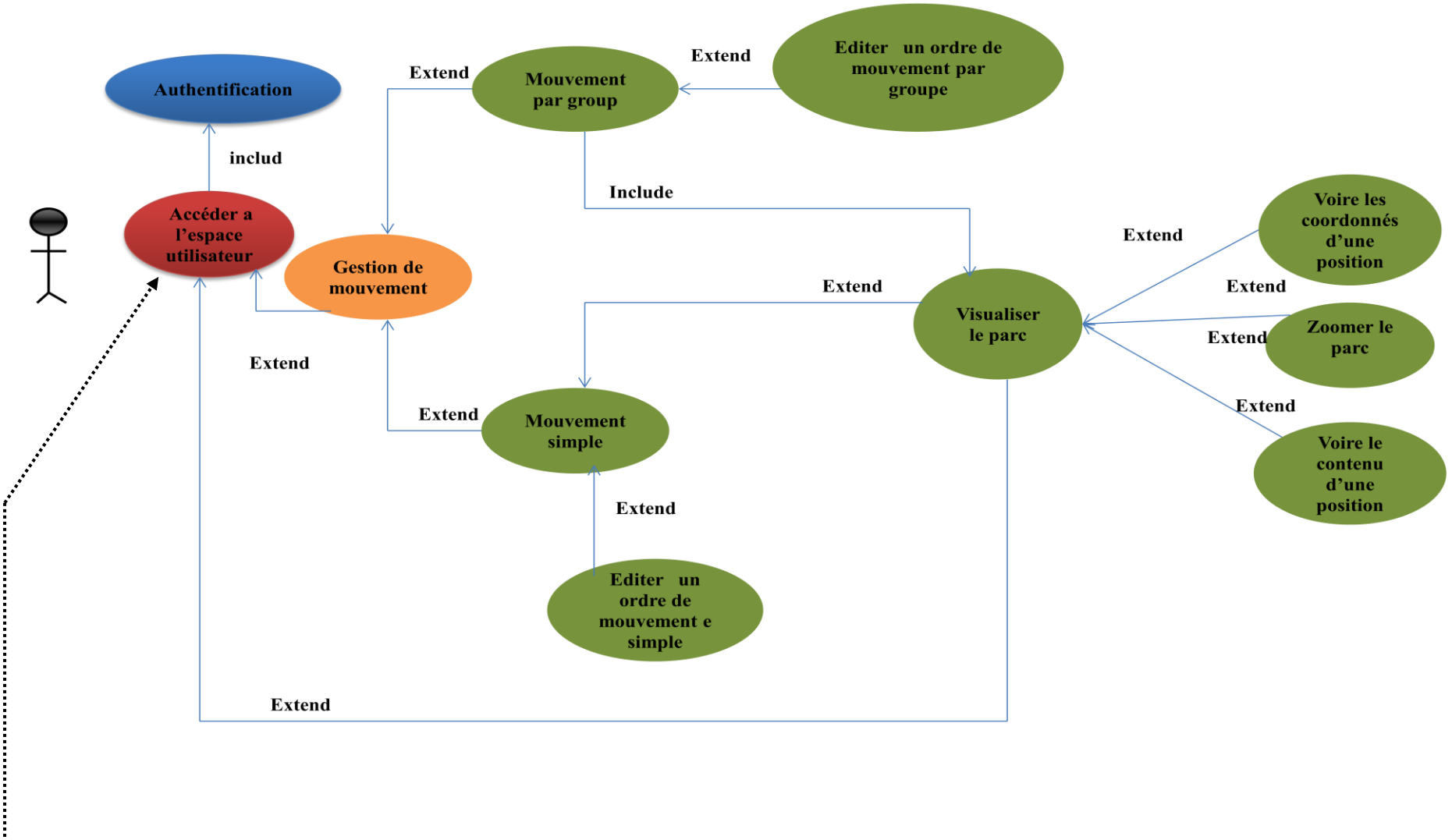


Figure IV.1 : Diagramme de cas d'utilisation de l'administrateur.

3.1.5-Diagramme de cas d'utilisation de l'utilisateur


Authentification
Extend
Mouvement par group
Extend
Editer un ordre de mouvement par groupe
includ
Accéder a l'espace utilisateur
Include
Gestion de mouvement
Extend
Visualiser le parc
Extend
Voire les coordonnés d'une position
Extend
Zoomer le parc
Extend
Voire le contenu d'une position
Extend
Extend
Mouvement simple
Extend
Editer un ordre de mouvement e simple
Extend

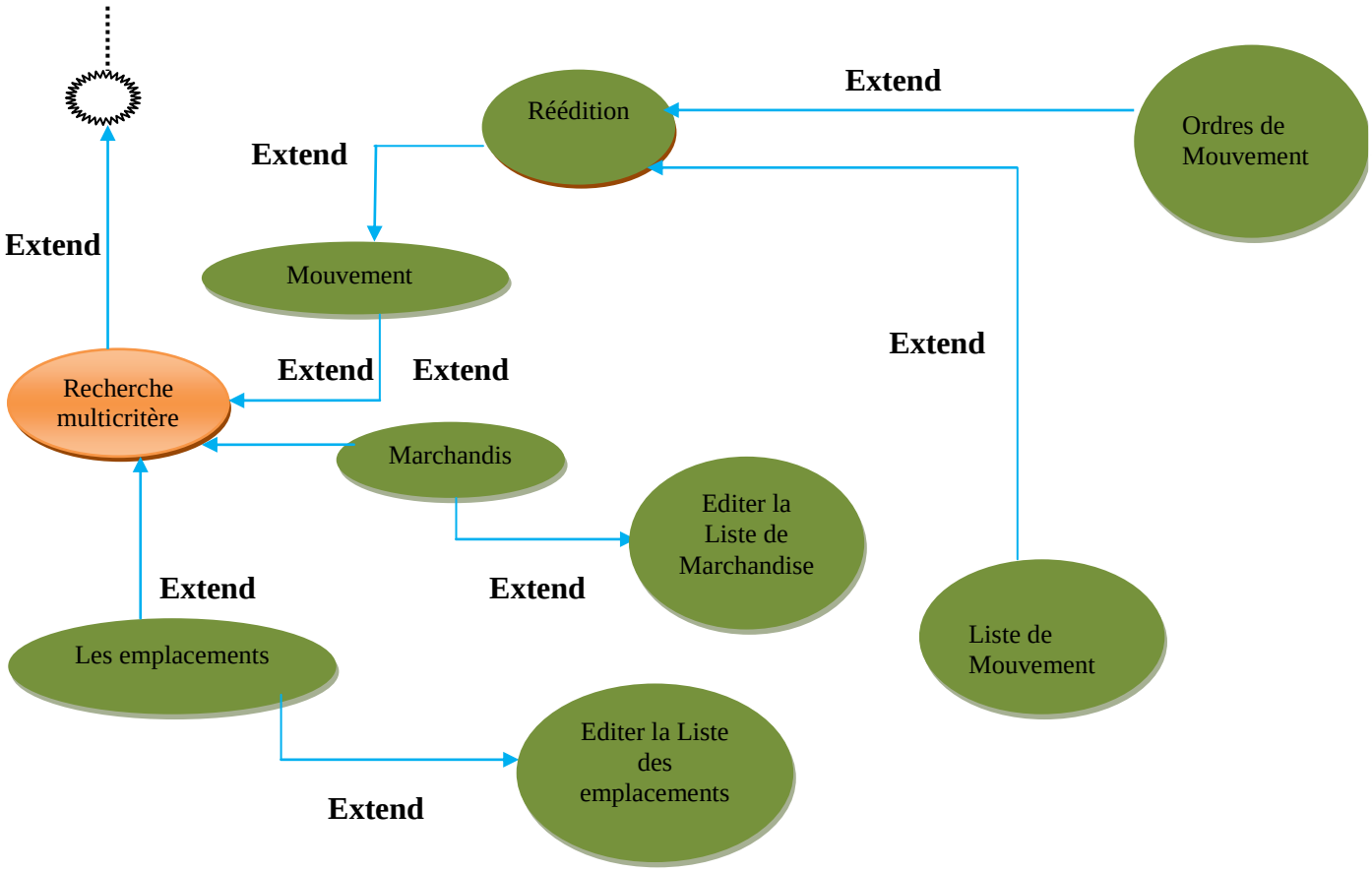


Figure IV.2: Diagramme de cas d'utilisation « Utilisateur ».

3.2- Représentation des diagrammes de séquence

3.2.1- Définition du diagramme de séquence

Les principales informations contenues dans les diagrammes de séquences sont les messages échangés entre les lignes de vie. Un diagramme de séquences met toujours l'accent sur l'ordre chronologique des messages.

Ces diagrammes peuvent être utilisés pour modéliser les responsabilités et les collaborations sans prendre en compte les mécanismes définis par l'architecture du système. [4]

3.2.2-Spécifications des diagrammes de séquence

Nous allons présenter dans ce qui suit les diagrammes de séquence pour quelques cas d'utilisation.

Quelques définitions :

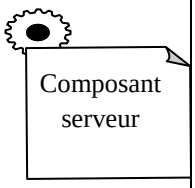
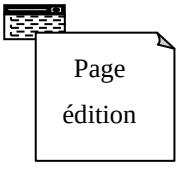
Composant	Description	Icône
Composant serveur	Un composant serveur représente une classe qui possède des scripts exécutés par le serveur. Ces scripts interagissent avec des ressources serveur tels que les bases de données ou systèmes externes. Les opérations de l'objet représentent des fonctions dans les scripts et ses attributs représentent les variables qui sont visibles dans la portée de la page.	
Page client	Une instance d'une page client est une page Web formatée en langage Web. Les pages clients qui sont restituées par des navigateurs client, peuvent contenir des scripts interprétés par les navigateurs. Les fonctions d'une page client correspondent aux fonctions des scripts de la page Web.	
Formulaire client	Une classe formulaire est un ensemble de champs de saisie faisant partie d'une page client. Les attributs de cette classe correspondent aux éléments de saisie d'un formulaire Web (zone de saisie, zone de texte, boutons d'option à cocher et éléments cachés).	

Tableau IV.2 : Présentation des éléments de diagramme de séquence

- **Diagrammes de séquence du cas d'utilisation « Configurer schéma de parc ».**

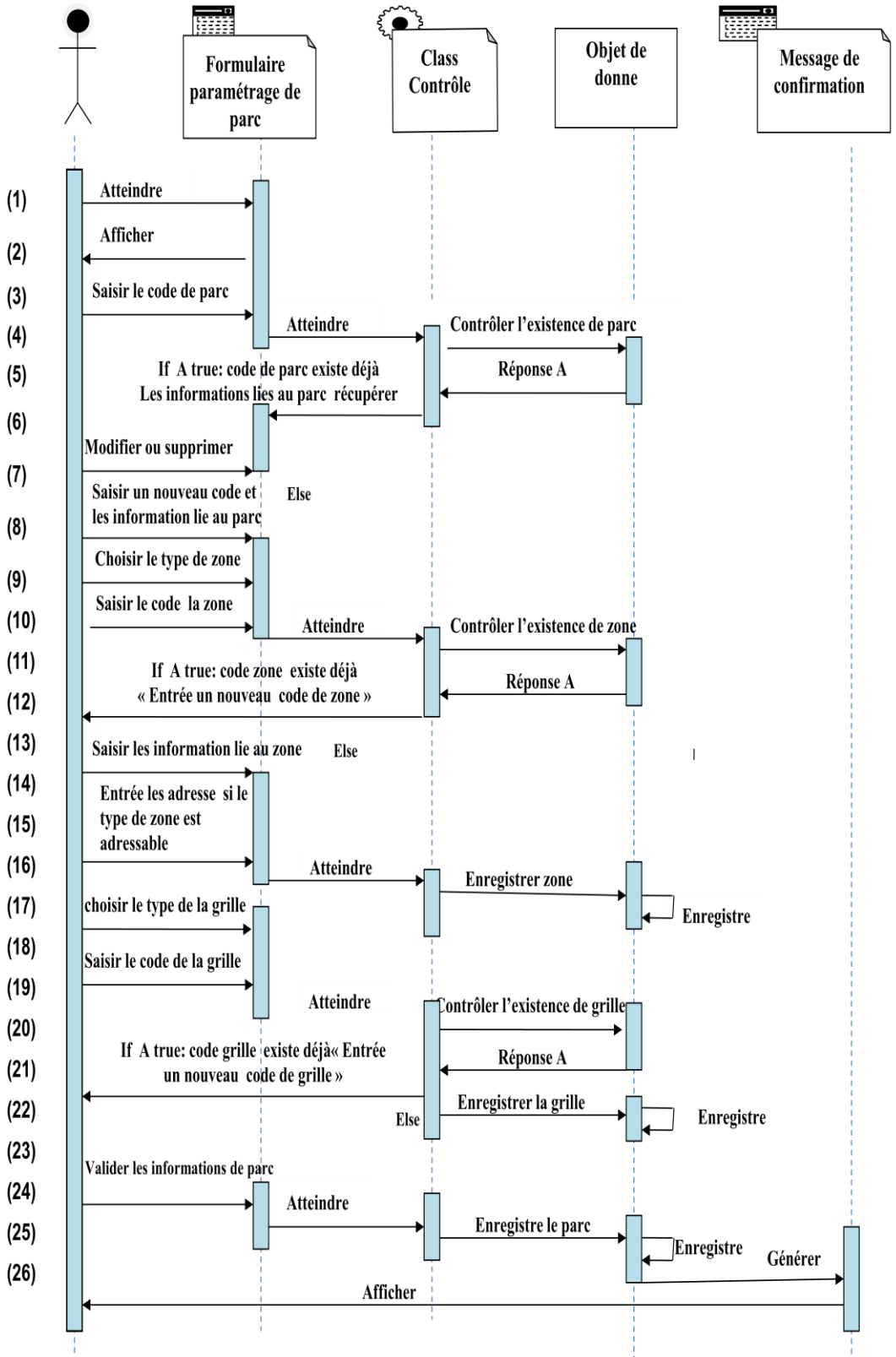


Figure IV.3: Diagramme de séquence «paramétrer un parc manuellement ».

1) L'administrateur atteint le formulaire paramétrage de parc.
2) Le système affiche le formulaire «paramétrage de parc».
3) L'administrateur saisit le code de parc.
4) Le système attient la classe de contrôle d'existence.
5) Le système contrôle l'existence de parc dans la table approprie.
6) Le système retourne une réponse A.
7) Si la réponse est « true » le système récupère les informations liées au parc (zones et grilles).
8) L'administrateur a le choix de modifier les informations liées au parc (zones et grilles) ou bien de les supprimer en suite validé les modifications apportées.
9) Sinon l'administrateur saisir un autre code de par et les informations liées au parc.
10) L'administrateur choisit le type de zone.
11) L'administrateur entre le code de la zone
12) Le système attient la classe de contrôle d'existence.
13) Le système contrôle l'existence de la zone dans la table approprie.
14) Le système retourne une réponse A.
15) Si la réponse est « true »la zone existe déjà, l'administrateur doit saisi un autre code de zone.
16) Sinon l'utilisateur saisit le nombre d'emplacement si la zone est adressable.
17) Le système enregistre la zone dans la table approprie.
18) L'administrateur choisit le type de la grille.
19) L'administrateur saisit le code de la grille.
20) Le système attient la classe de contrôle d'existence.
21) Le système contrôle l'existence de la grille dans la table approprie.
22) Le système retourne une réponse A.
23) Si la réponse est « true » l'administrateur doit saisi un autre code de grille.
24) Sinon le système enregistre la grille dans la table approprie.
25) Le système valide (enregistre) les informations liées au parc dans la table approprie.
26) Le système construit et affiche le message de confirmation à l'écran.

- **Diagrammes de séquence du cas d'utilisation « Mouvement entrée simple avec schéma ».**

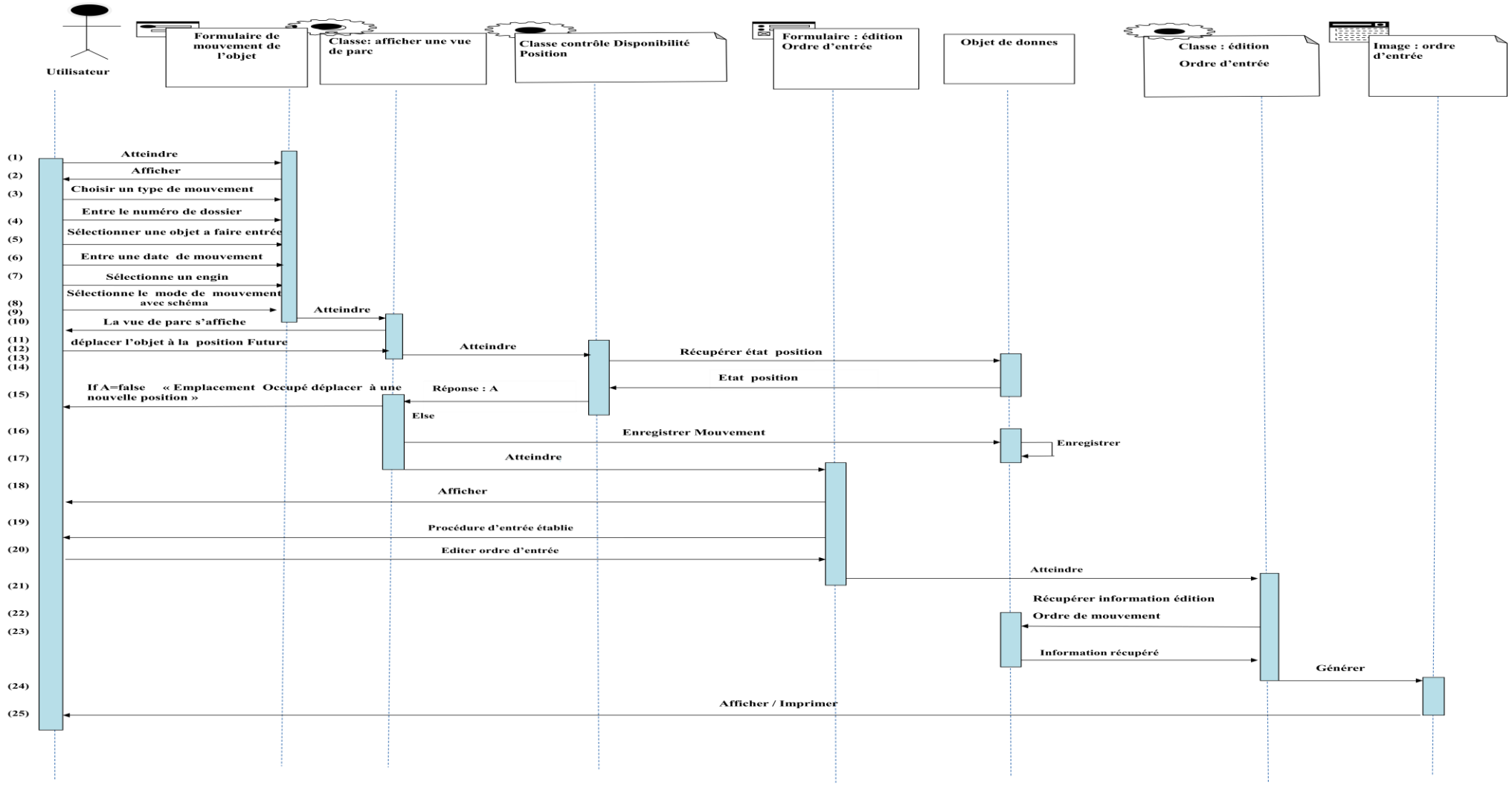


Figure IV.4: Diagramme de séquence «Mouvement Entrée simple avec schéma ».

1) L'utilisateur atteint le formulaire de mouvement simple.
2) Le système affiche le formulaire de mouvement simple.
3) L'utilisateur choisit un type de mouvement simple « Entrée ».
4) L'utilisateur saisir le numéro de dossier.
5) L'utilisateur sélectionne l'objet à faire entrée.
6) L'utilisateur entre une date de mouvement.
7) L'utilisateur sélectionne un engin.
8) L'utilisateur sélectionne le mode de mouvement simple avec schéma.
9) Le système atteint la classe « afficher la vue de parc »
10) Le système affiche une vue de parc.
11) L'utilisateur déplacer l'objet à la position dans la quelle l'objet sera posé.
12) Le système atteint la classe de disponibilité de position.
13) Le système contrôle l'état de la position.
14) Le système récupéré l'état de la position via la table appropriée
15) Le système retourne la réponse sur l'état de position
16) Si la réponse est « false » i.e. emplacement occupé, l'utilisateur doit saisir une autre Position d'emplacement.
17) Sinon le système enregistre le « mouvement Entrée simple » dan la table appropriée.
18) Le système atteindre le formulaire d'édition d'ordre d'entrée.
19) Le système affiche le formulaire d'édition d'ordre d'entrée.
20) Le système établit la procédure d'entrée de la marchandise.
21) L'utilisateur lance l'édition d'ordre d'entrée.
22) Le système atteint la classe d'édition d'ordre d'entrée.
23) Le système récupère les informations d'édition d'ordre d'entrée via la table.
24) Le système génère l'image d'ordre d'entrée.
25) Le système affiche ou d'imprime l'ordre d'entrée selon le choix d'utilisateur.

- **Diagrammes de séquence du cas d'utilisation « Mouvement déplacement simple sans schéma».**

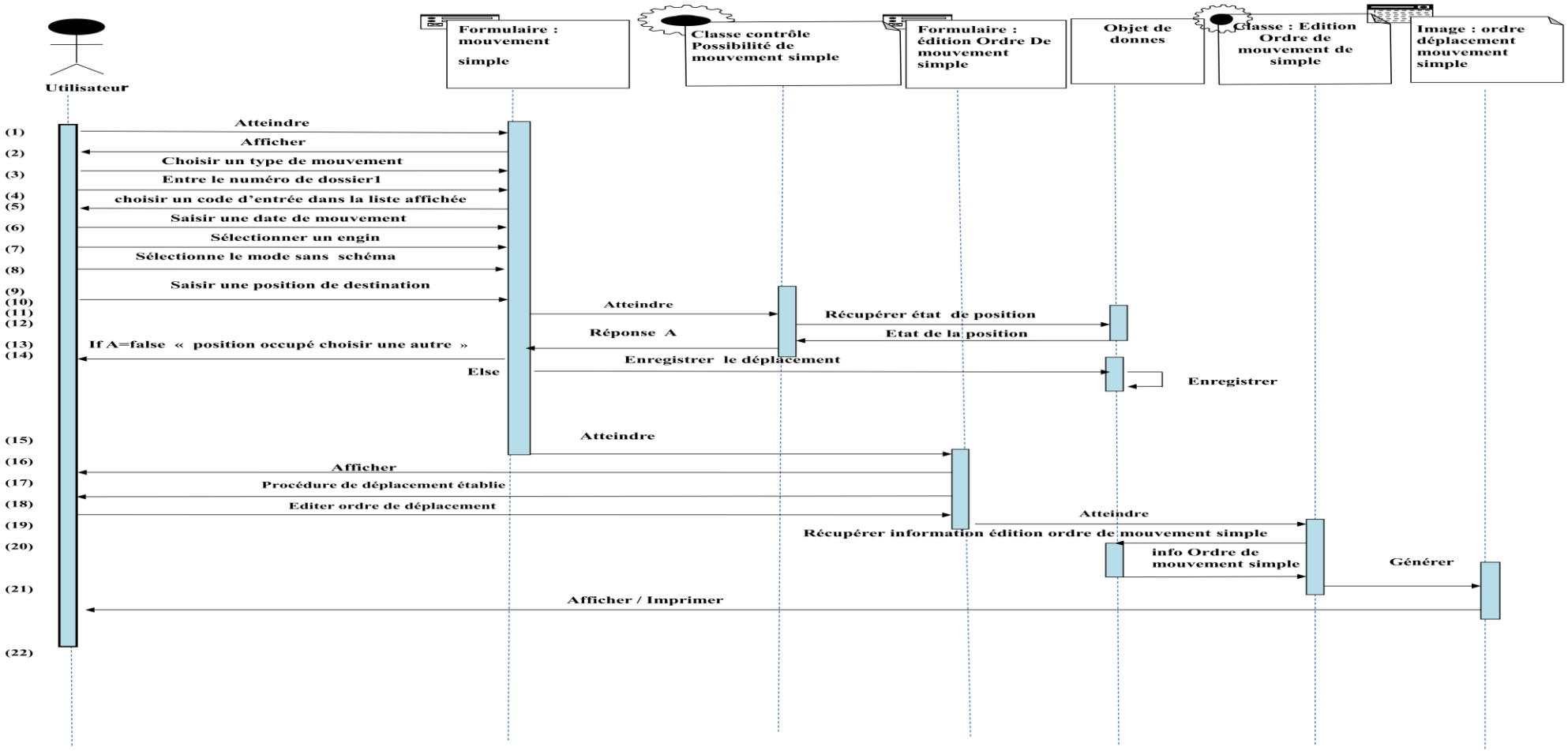


Figure IV.5: Diagramme de séquence «Mouvement déplacement simple».

»

1) L'utilisateur atteint le formulaire mouvement simple.
2) Le système affiche le formulaire de mouvement simple.
3) L'utilisateur choisit un type de mouvement simple « déplacement ».
4) L'utilisateur entre un numéro de dossier1.
5) L'utilisateur choisit un code d'entrée dans la liste affichée.
6) L'utilisateur entre une date de mouvement
7) L'utilisateur sélectionne un engin.
8) L'utilisateur sélectionne le mode sans schéma pour le mouvement simple « déplacement ».
9) L'utilisateur entre une position de destination pour le déplacement et clique sur « déplacer »
10) Le système atteindre classe possibilité de mouvement simple.
11) Le système récupère l'état de la position via la table approprie.
12) Le système retourne la réponse sur l'état de la position.
13) Si la réponse est « false »c'est-à-dire l'emplacement est occupé, l'utilisateur doit saisi une autre position de destination
14) Sinon le système enregistre le mouvement simple.
15) Le système attient le formulaire édition d'ordre de mouvement simple.
16) Le système affiche le formulaire d'édition à l'utilisateur.
17) Le système établit la procédure de déplacement simple de l'objet
18) L'utilisateur lance l'édition d'ordre de déplacement simple.
19) Le système attient la classe d'édition d'ordre de déplacement simple.
20) Le système récupère les informations d'édition d'ordre de déplacement simple via la table.
21) Le système génère l'image d'ordre de mouvement de déplacement simple
22) Le système affiche ou imprime l'ordre de mouvement de déplacement simple selon le choix de l'utilisateur.

- **Diagrammes de séquence du cas d'utilisation « Mouvement déplacement par groupe».**

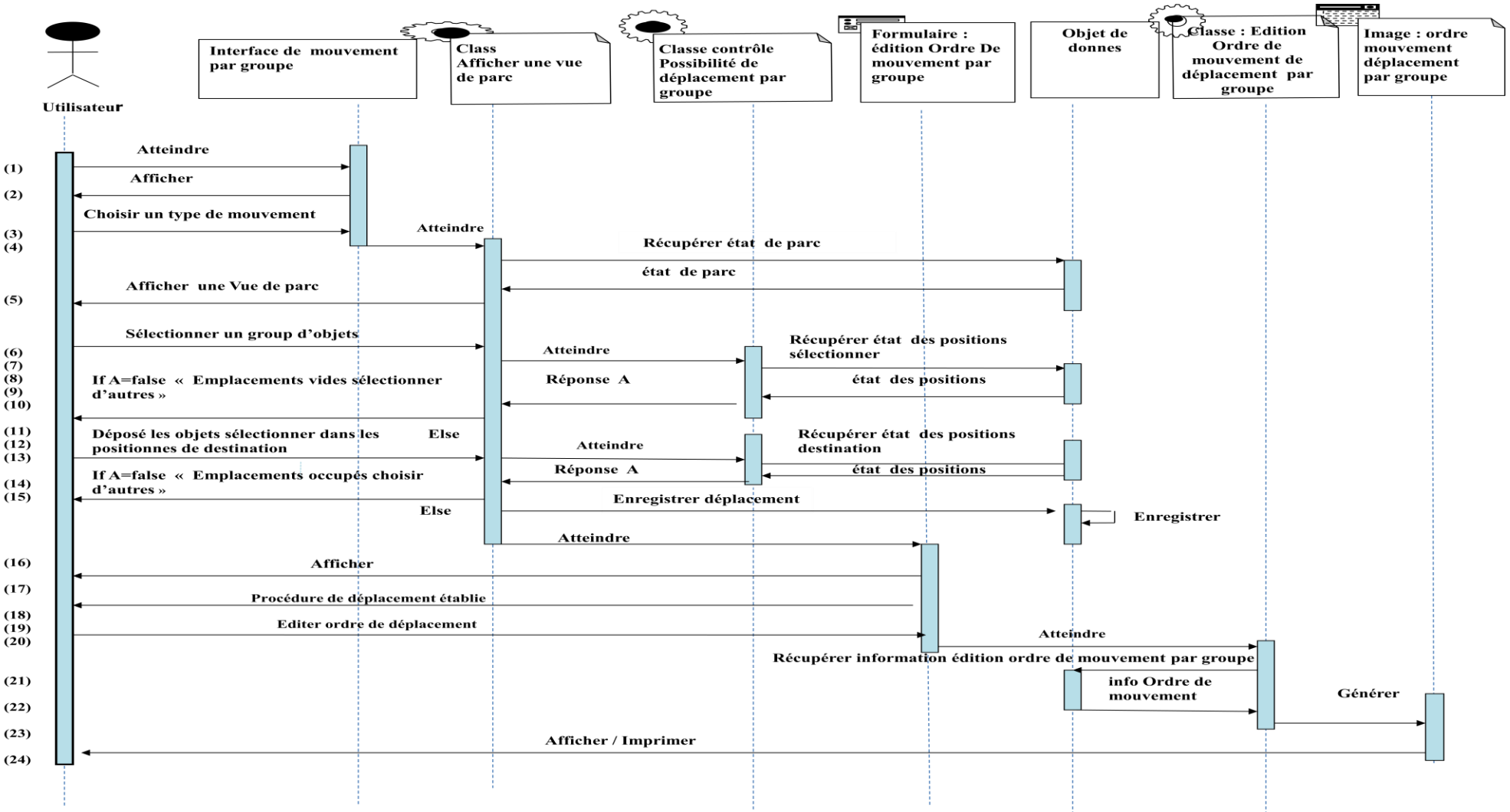


Figure IV.6: Diagramme de séquence «Mouvement déplacement par groupe graphiquement ».

»

1) L'utilisateur atteint Interface de mouvement par groupe.
2) Le système affiche l'interface de mouvement par groupe.
3) L'utilisateur choisit un type de mouvement par group « déplacement ».
4) Le système atteint la classe « afficher un vue de parc »
5) Le système récupère l'état de parc via la table approprie.
6) Le système affiche une vue de parc.
7) L'utilisateur sélectionne le groupe d'objet à faire déplacer
8) Le système atteint la classe contrôle de possibilité de mouvement déplacement par groupe
9) Le système récupère les états des positions sélectionnes.
10) le système retourne la réponse les états des positions sélectionnées
11) Si la réponse est « false» c'est-a-dire les emplacements sélectionnées sont vides l'utilisateur doit sélectionnée d'autre emplacements.
12) Sinon le système atteint la classe contrôle de possibilité de mouvement déplacement par groupe
13) Le système récupère les états des positions destinations.
14) le système retourne la réponse A.
15) Si la réponse est « false » c'est-a-dire au moins l'un des emplacements destinations est occupé l'utilisateur doit choisi d'autre emplacements.
16) Sinon le système enregistre le « mouvement déplacement par groupe ».
17) Le système attient le formulaire édition d'ordre de déplacement par groupe.
18) Le système affiche le « formulaire d'édition » à l'utilisateur.
19) Le système établit la procédure de déplacement par groupe.
20) L'utilisateur lance l'édition d'ordre de déplacement par groupe.
21) Le système attient la classe d'édition d'ordre de déplacement par groupe.
22) Le système récupère les informations d'édition d'ordre de déplacement par groupe via la table.
23) Le système génère l'image d'ordre de mouvement de déplacement par groupe.
24) Le système affiche ou imprime l'ordre de mouvement de déplacement par groupe selon le choix de l'utilisateur.

- **Diagrammes de séquence du cas d'utilisation « Mouvement sortie sans schéma ».**

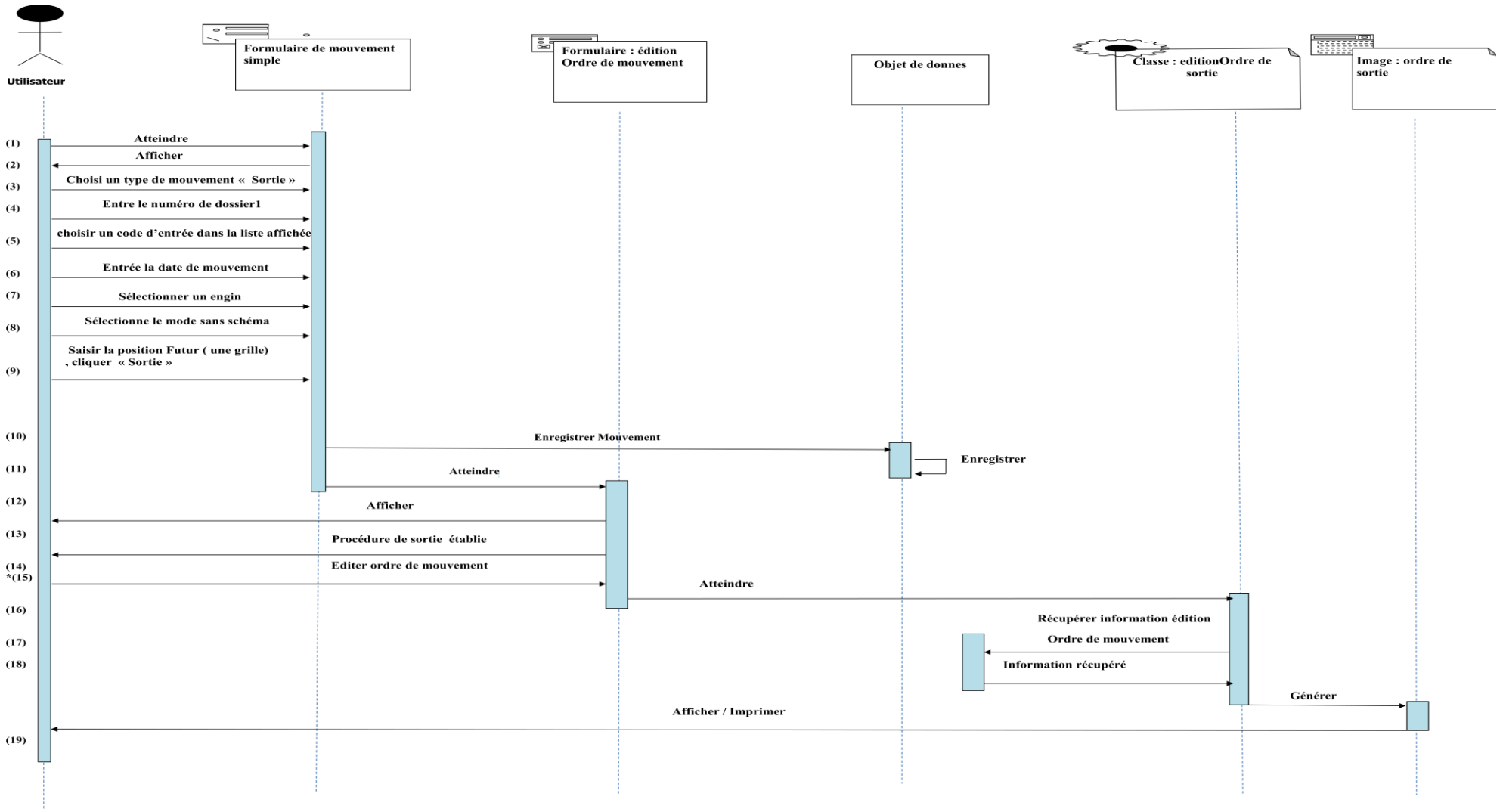


Figure IV.7: Diagramme de séquence «Mouvement Sortie ».

1) L'utilisateur atteint le formulaire de mouvement simple.
2) Le système affiche le formulaire de mouvement simple.
3) L'utilisateur choisit un type de mouvement simple « sortie».
4) L'utilisateur saisit un numéro de dossier1.
5) L'utilisateur choisit un code d'entrée dans la liste affichée
6) L'utilisateur entre une date de mouvement.
7) L'utilisateur sélectionne un engin.
8) L'utilisateur sélectionne le mode sans schéma.
9) L'utilisateur saisit une adresse de grille par la quelle l'objet doit sortie. et cliquer « sortie ».
10) le système enregistré le mouvement simple « sortie» dans la table approprie.
11) Le système atteindre le formulaire d'édition d'ordre mouvement simple.
12) Le système affiche le formulaire d'édition
13) Le système établit la procédure de sortie.
14) L'utilisateur lance l'édition d'ordre de mouvement sortie.
15) Le système atteint la classe d'édition d'ordre de sortie.
16) Le système récupère les informations d'édition d'ordre de sortie via la table
17) Le système génère image d'ordre de sortie
18) Le système affiche ou d'imprime l'ordre de sortie selon le choix d'utilisateur

3.3- Le diagrammes de classes [5]

3.3.1-Définition :

Le diagramme des classes identifie la structure des classes d'un système, y compris les propriétés et les méthodes de chaque classe. Les diverses relations, telles que la relation d'héritage par exemple, qui peuvent exister entre les classes sont également représentées.

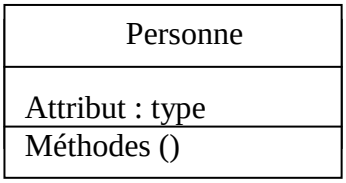
Il permet aussi de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

3.3.2-Définition et représentation des éléments du diagramme de classes

- **Classes**

Les classes sont les modules de base de la programmation orientée objet. Une classe est représentée en utilisant un rectangle divisé en trois sections. La section supérieure est le nom de la classe. La section centrale définit les propriétés de la classe. La section du bas énumère les méthodes de la classe.

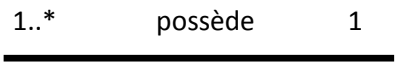
Exemple :



- **Association**

Une association est une relation générique entre deux classes. Elle est modélisée par une ligne reliant les deux classes. Cette ligne peut être qualifiée avec le type de relation, et peut également comporter des règles de multiplicité (par exemple un à un, un à plusieurs, plusieurs à plusieurs) pour la relation

Exemple :



- **Composition**

Si une classe ne peut pas exister par elle-même, mais doit être un membre d'une autre classe, alors elle possède une relation de composition avec la classe contenant.

Exemple :

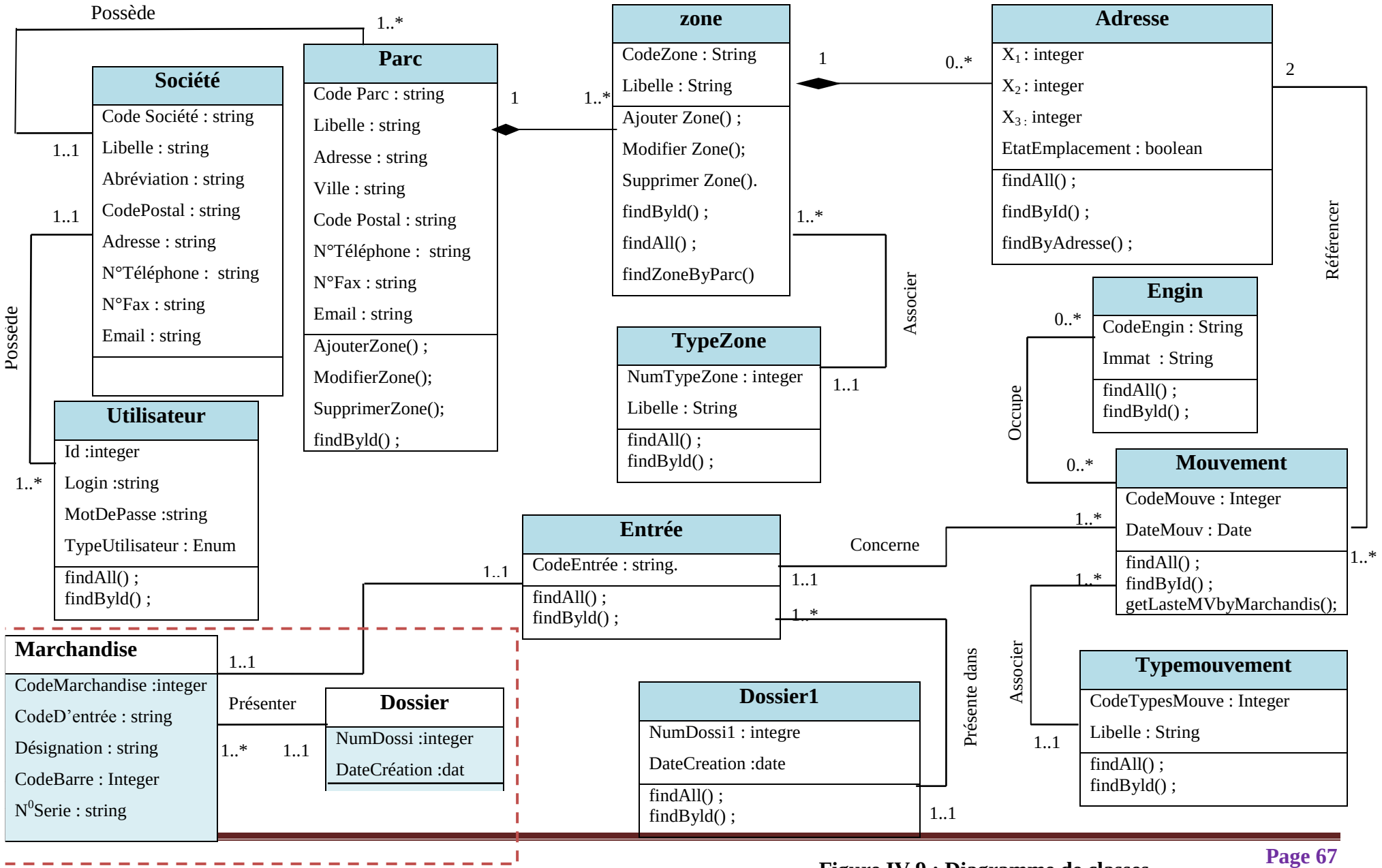


Figure IV.9 : Diagramme de classes

4-Modélisation de l'aspect statique de la solution proposée

Le processus de modélisation de l'aspect statique se résume par les étapes suivantes:

- Présenter les règles de gestion.
- Construire le modèle conceptuel de données (MCD).
- Transformer le modèle conceptuel de données (MCD) en logique de données (MLD).

4.1- Les règles de gestion

- ✓ **RG1 :** une société possède un ou plusieurs parcs.
- ✓ **RG2 :** une société possède un ou plusieurs utilisateurs.
- ✓ **RG3 :** un parc est composé d'une ou plusieurs zones (adressables ou non adressables).
- ✓ **RG4 :** une zone adressable possède une ou plusieurs adresses.
- ✓ **RG5 :** un type de zone est associé à une ou plusieurs zones.
- ✓ **RG6 :** un mouvement concerne une ou plusieurs marchandises entrées.
- ✓ **RG7 :** un mouvement utilise zéro ou plusieurs engins.
- ✓ **RG8 :** un mouvement fait référence à deux adresses.
- ✓ **RG9:** un type de mouvement fait référence à un ou plusieurs mouvements.
- ✓ **RG10 :** un dossier contienne une ou plusieurs entrées.

4.2- Elaboration du modèle conceptuel de données MCD

4.2.1-Définition :

Le Modèle Conceptuel de Données (MCD) est une représentation statique du système d'information de l'entreprise. Il a pour objectif de constituer une représentation claire et cohérente des données manipulées dans l'entreprise en décrivant leur sémantique (le sens attaché à ces données) et les rapports qui existent entre elles. Les règles de construction du MCD permettent d'aboutir à une représentation graphique standard qui élimine les redondances et les ambiguïtés.

4.2.2-Concepts de base :

❖ **Objet**

Un objet est une entité dotée d'une existence propre et est décrit par un identifiant et une liste de propriétés qui lui sont spécifiques.

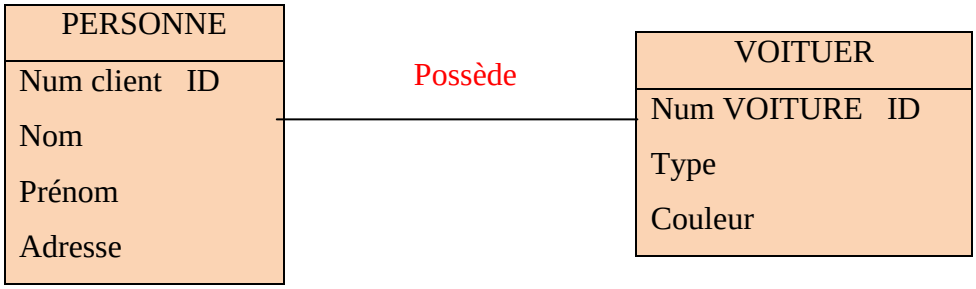
Exemple :

CLIENT
Num client ID
Nom
Prénom

❖ **Association**

Une association (ou relation) décrit le lien existant entre deux objets ou plus. Elle peut être porteuse de propriétés ou non.

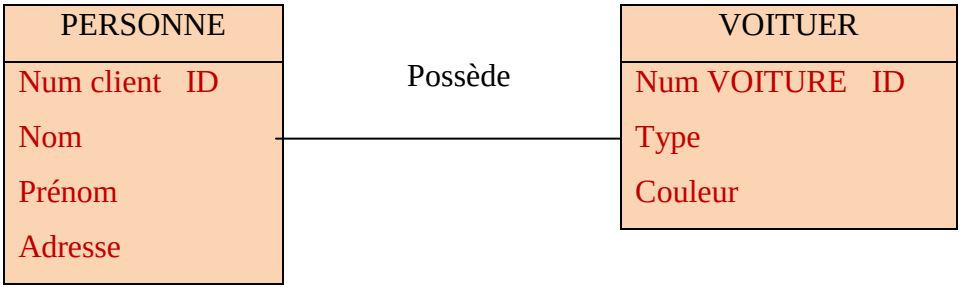
Exemple :



❖ **Propriété**

Propriété ou attribut est une donnée élémentaire d'informations ayant un sens en elle-même et appartenant à la description d'un objet ou d'une association.

Exemple :



❖ **Identifiant**

Un identifiant est un attribut d'entité ou un ensemble d'attributs d'entité, dont les valeurs identifient de manière unique chaque occurrence de l'entité. L'identifiant dans le MCD correspond à la clé primaire ou à la clé alternative dans le MPD.

Chaque entité doit comporter au moins un identifiant. Si une entité n'a qu'un seul identifiant, alors ce dernier est désigné par défaut identifiant primaire de l'entité. C'est l'identifiant principal d'une entité.

❖ **Occurrence**

Une occurrence d'une propriété est une valeur prise par cette propriété, une occurrence d'un objet est un exemplaire de cet objet caractérisé par un ensemble des valeurs de ses propriétés.

❖ **Cardinalité**

La cardinalité d'un lien entre un objet et une association indique le **minimum** et **maximum** de fois qu'un individu de l'objet peut être concerné par l'association. La cardinalité minimale est de 0 ou 1 ; la cardinalité maximale est de 1 ou N

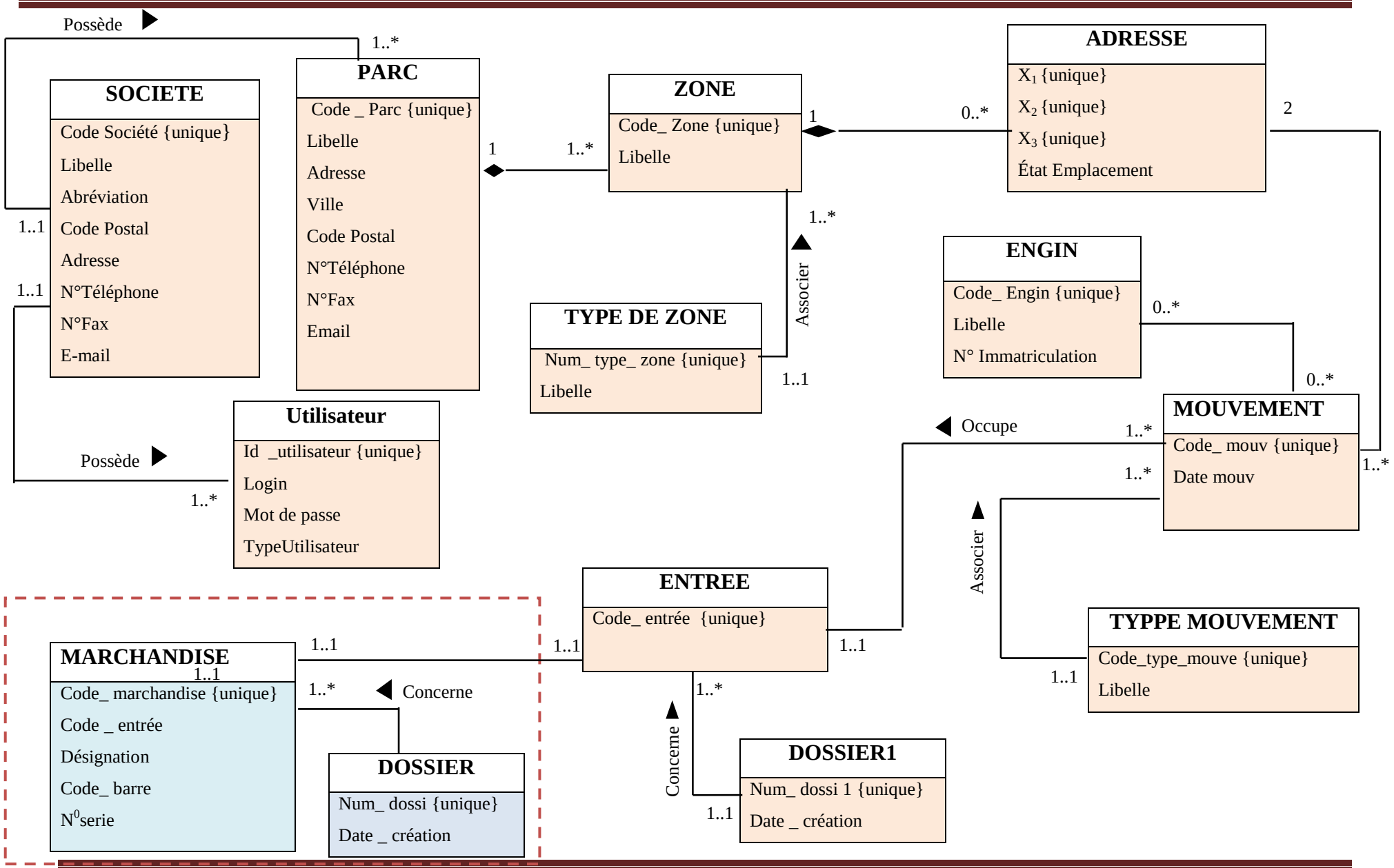


Figure IV.10 : Modèle conceptuelle de données MCD

4.2.3 -Dictionnaire de données.

Un dictionnaire des données est une collection de métadonnées ou de données de référence nécessaire à la conception les déférents tables d'une base de données.

1. Utilisateur

Attribut	Type	Contrainte	Description
Id _utilisateur {unique}	Integer	Unique	Identifiant de l'utilisateur
TypeUtilisateur	Integer		0 : administrateur 1 : utilisateur
Login	string		
Mot de passe	string		

2. Société

Attribut	Type	Contrainte	Description
Code_ société	String	Unique	Code de la société
Libelle	String		Libelle de la société
Abréviation	String		Abréviation du non de la société
Adresse	String		Adresse de la société
Code Postal	String		Code postal de la société
N°Téléphone	String		Numéros de téléphones de la société
N°Fax	String		Numéro de fax de la société
E-mail	String		Adresse e-mail de la société

3. Parc

Attribut	Type	Contrainte	Description
Code_ parc	String	Unique	Code du parc
Libelle	String		Libelle du parc
Adresse	String		Adresse du parc
Ville	String		Ville dans la quelle est situé le parc
Code Postal	String		code postal du parc
N°Téléphone	String		Numéros de téléphone du parc

N°Fax	String		Numéro de fax du parc
Email	String		Adresse e-mail du parc

4. Zone

Attribut	Type	Contrainte	Description
Code_ zone	String	Unique	Code de la zone
Libelle	String		Libelle de la zone

5. Type de zone

Attribut	Type	contrainte	Description
Num_type_zone	Enum {1,2 ,3 , 4, 5}	unique	1 : zone adressable 2 : zone non adressable 3 : grille d'entrée 4 : grille de sortie 5 : grille d'entrée/sortie
Libelle	String		Libelle de type de zone

6. Adresse

Attribut	Type	Contrainte	Description
X_1	Integer	Unique,>0	L'adresse d'emplacement x1
X_2	Integer	Unique,>0	L'adresse d'emplacement x2
X_3	Integer	Unique, >0	L'adresse d'emplacement x3
Etat d'emplacement	Integer		0 : Emplacement occupé. 1 : Emplacement vide.

7. Engin

Attribut	Type	Contrainte	Description
Code_ engin	String	Unique	Code de l'engin

Libelle	String		Libelle de l'engin
Immatriculation	String		Immatriculation de l'engin

8. Mouvement

Attribut	Type	Contrainte	Description
Code_ mouv	Integer	Unique	Code du mouvement
Date_mouv	Date	Non nul	Date et heure de mouvement

9. Type Mouvement

Attribut	Type	Contrainte	Description
Num _ type mouvement	Enum {1, 2, 3, 4, 5, 6, 7, 8, 9}	Unique	1 : entrée 2 : déplacement simplet dans la zone adressable ou non adressable 3 : déplacement simple inter zone adressable ou non adressable 4 : déplacement simple inter parc 5 : déplacement par groupe dans la zone adressable 6 : déplacement par groupe inter zone adressable 7 : déplacement par groupe d'une zone adressable a une zone non adressable 8 : déplacement par groupe inter parc 9 : sortie.

10. Entrée

Attribut	Type	Contrainte	Description
Code _ entrée	Integer	Unique	Code d'entrée

11. Dossier1

Attribut	Type	Contrainte	Description
Num_ dossi1	Integer	Unique	Numéro de dossier
Date _ de_ création	date		Date de création de dossier

- **Les tables de l'autre système.**

12. marchandise

Attribut	Type	Contrainte	Description
Code_ marchandise	Integer	Unique	Code de la marchandise
Désignation	string		Désignation de la marchandise
Code_ barre	Integer		Code barre de la marchandise
N^0 série	String		Numéro de série de marchandise

13. Dossier

Attribut	Type	Contrainte	Description
Code_ dossi	Integer	Unique	Code de dossier
Date _ de_ création	Date		Date de création de dossier

4.3- Elaboration du modèle relationnelle MLD

4.3.1-Définition :

Il consiste à donner une représentation schématique des données du système du point de vue de la base de données qui va les stocker. En d'autres termes, il s'agit d'une définition de la structure des données telles qu'elle sera employée dans la base de données. Il donne le nom des tables, les champs avec les types de chacun (chaîne de caractère, décimal, entier, etc.) et les relations de dépendance entre les tables. Le MLD est très dépendant de la base de données employée, car elle en est tout simplement son schéma.

4.3.2-Concepts de base :

❖ **Tables, lignes et colonnes**

Lorsque les données ont la même structure on peut alors les organiser en tables dans lesquelles les colonnes décrivent les champs en commun les lignes contiennent les valeurs de ces champs pour chaque enregistrement.

❖ **Clés primaires et clés étrangères**

Les lignes d'une table sont uniques ⇒ il existe au moins une colonne qui sert à identifier les lignes : il s'agit de la clé primaire de la table.

Propriétés requises :

-la valeur vide (NULL) est interdite

-la valeur de la clé primaire d'une ligne ne devrait pas changer au cours du temps

❖ **Schéma relationnel**

Les tables sont appelées relations, les liens entre les clés étrangères et leur clé primaire sont symbolisés par un connecteur.

❖ **Notations**

On dit qu'une association binaire (entre deux entités ou réflexive) est de type :

1 :1 (un à un) si aucune des 2 cardinalités maximales n'est n.

1 : n (un à plusieurs) si une des 2 cardinalités maximales Est n.

n : m (plusieurs à plusieurs) si les 2 cardinalités maximales sont n.

4.3.3-Règles de dérivation du MCD au MLD :

- Pour chaque entité de modèle conceptuelle, une table est créée dans le modèle relationnelle.
- Les attributs de l'entité deviennent les attributs de la table et son identifiant simple ou composé, deviennent la clé primaire de la table, chaque attribut formant la clé primaire porte la montions {PK}.
- En ce qui concerne une entité d'association, la table correspondante aura comme clé primaire une clé composée des attributs de tous les identifiants des entités participant à l'association et chaque attribut de la clé composée portera la mention « PK, FK ».
- Dans le cas d'une association dans la multiplicité du cotée de l'association X et de type (1..1), la table correspondante à l'association se trouve dans l'autre coté de l'association comportera les attributs constituant l'identifient de l'association X chacune des attributs portera la mention « FK » qui symbolise la clé étranger

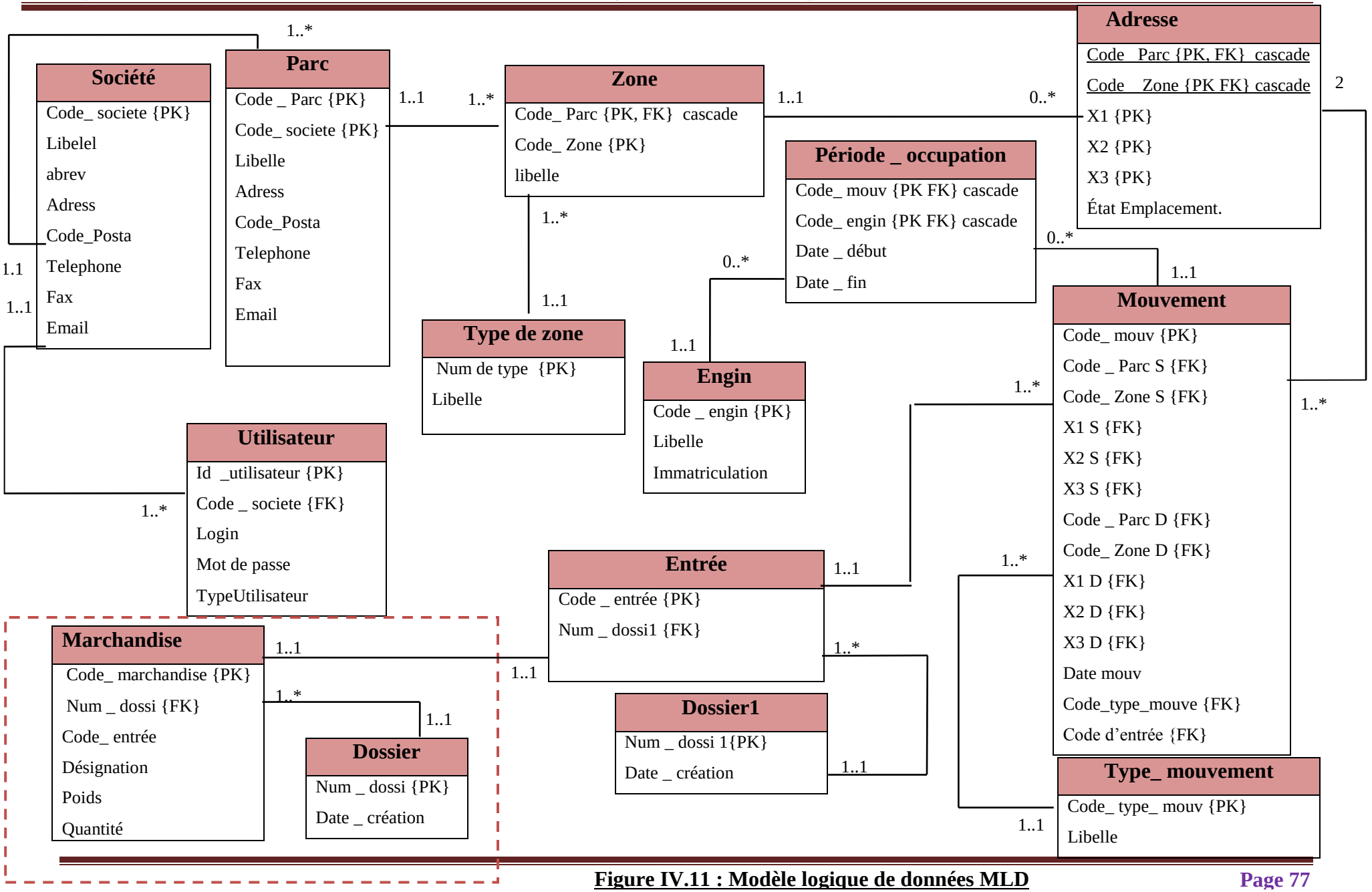


Figure IV.11 : Modèle logique de données MLD

4.3.4- Liste des tables :

1. Utilisateur

Colonne	Type	Taille	Nul	Description
Id _utilisateur {PK}	Integer	4	Non	Identifiant de l'utilisateur
Code_ societe	Varchar	10		Code de la societe
TypeUtilisateur	Integer	1		0 : administrateur 1 : utilisateur
Login	string	10		Login de l'utilisateur
Mot_de_passe	string	10		Mot de passe de l'utilisateur

2. Société

Colonne	Type	Taille	Nul	Description
Code société {PK}	Varchar	10	Non	Code de la société
Libelle	Varchar	50		Libelle de la société
Abrév	Varchar	20		Abréviation du non de la société
Adresse	Varchar	120		De la société
Code_ Postal	Varchar	10		Code postal de la société
Téléphone	Varchar	80		Numéros de téléphones de la société
Fax	Varchar	80		Numéro de fax de la société
Email	Varchar	80		Adresse e-mail de la société

3. Parc

Colonne	Type	Taille	Nul	Description
Code parc {PK}	Varchar	10	Non	Code du parc
Code société {FK}	Varchar	10	Non	Code du la société
Libelle	Varchar	80		Libelle du parc
Adress	Varchar	120		Adresse du parc
Ville	Varchar	25		Ville dans la quelle est situé le parc
Code_ Posta	Varchar	10		code postal du parc

Téléphone	Varchar	50		Numéros de téléphone du parc
Fax	Varchar	50		Numéro de fax du parc
Email	Varchar	80		Adresse e-mail du parc

4. Zone

Colonne	Type	Taille	Nul	Description
Code_ Parc {PK FK}	Varchar	10	Non	Code unique du parc dans le quelle se trouve la zone
Code_ Zone {PK}	Varchar	10	Non	Code de la zone
Libelle	Varchar	30		Libelle de la zone

5. Type zone

Colonne	Type	Taille	Nul	Description
Num_ type_ zone {PK}	Enum {1, 2, 3, 4,5}	2	non	1 : zone adressable 2 : zone non adressable 3 : grille d'entrée 4 : grille de sortie 5 : grille d'entrée/sortie

6. Adresse

Colonne	Type	Taille	Nul	Description
Code_ Parc {PK FK} cascade	Varchar	10	Non	Code du parc
Code_ Zone {PK FK} cascade	Varchar	10	Non	Code de la zone
X 1{PK}	Integer	1	Non	Numéro de X1
X 2{PK}	Integer	1	Non	Numéro de X2
X 3{PK}	Integer	1	Non	Numéro de X3
Etat d'emplacement	Integer	1	Non	0 : l'emplacement occupé 1: désactiver vide.

7. Engin

Colonne	Type	Taille	Nul	Description
Code engin {PK}	Varchar	10	Non	Code de l'engin
Libelle	Varchar	100		Libelle de l'engin
Immatriculation	Varchar	30	Non	Immatriculation de l'engin

8. Période _ occupation

Attribut	Type	Taille	Nul	Description
Code _ engin {PK FK} cascade	Varchar	10	non	Code entrée
Code_ mouv {PK FK} cascade	Varchar	2	non	Code de mouvement
Date de début	Date time		non	Date de début du période
Date de fin	Date time		non	Date de fin du période

9. Mouvement

Colonne	Type	Taille	Taille	Description
Code_ Mouve {PK}	Integer	8	Non	Numéro séquentiel
Code_ Type_ Mouve {FK}	Integer	2	Non	Code du type de mouvement
Code_ entrée {FK}	Varchar	11	Non	Numéro d'entrée de la marchandise
Date_ Heure	Date time		Non	Date et heure de mouvement
Code_ Parc S {FK}	Varchar	10	Non	Code du parc dans le quelle le mouvement s'est effectué
Code_ Zone S {FK}	Varchar	10	Non	Code de la zone dans le quelle le mouvement s'est effectué
X1 S {FK}	Int	3	Non	Numéro de X1 source
X2 S {FK}	Int	3	Non	Numéro de X2 source
X2 S {FK}	Int	3	Non	Numéro de X3 source
Code_ Parc D {FK}	Varchar	10	Non	Code du parc distinction dans le quelle le mouvement s'est effectué

Code_ Zone D {FK}	Varchar	10	Non	Code de la zone de destination
X1 D {FK}	Int	3		Numéro de X1 destination
X2 D {FK}	Int	3		Numéro de X2 destination
X3 D {FK}	Int	3		Numéro de X3 destination

10. Type Mouvement

Colonne	Type	Taille	Nul	Description
Code type mouv {PK}	Enum {1, 2, 3, 4, 5, 6, 7, 8, 9}	2	Non	1 : entrée 2 : déplacement simplet dans la zone adressable ou non adressable 3 : déplacement simple inter zone adressable ou non adressable 4 : déplacement simple inter parc 5 : déplacement par groupe dans la zone adressable 6 : déplacement par groupe inter zone adressable 7 : déplacement par groupe d'une zone adressable a une zone non adressable 8 : déplacement par groupe inter parc 9 : sortie.

11. Entrée

Colonne	Type	Taille	Nul	Description
Code entrée {PK}	Integer	2	Non	Code d'entrée
Code_dossi1 {FK}	Integer	8	Non	Code dossier1 dans le quel se trouve le code d'entreé.

11. dossier1

Colonne	Type	Taille	Nul	Description
Code_ dossi 1	Integer	8	Non	Code de dossier
Date _ de_ création	Date time		Non	Date de création de dossier

❖ **Les tables de l'autre système.**

12. marchandise

Colonne	Type	Taille	Nul	Description
Code_ marchandise {PK}	Integer	10	Non	Code de la marchandise
Code_ dossi {FK}	Integer	8		
Code _ entreé {FK}	Varchar	10		Code d'entrée
Désignation	Varchar	80		Date de création de dossier
Code_ barre	Integer	14		Code barre de la marchandise
N^0 série	Varchar	50		Numéro de série

13. Dossier

Colonne	Type	Taille	Nul	Description
Code_ dossi {PK}	Integer	8	non	Code de dossier
Date _ de_ création	Date time			Date de création de dossier

Conclusion

Tout au long de ce chapitre, nous avons traduits les besoins recensés au par avant dans le chapitre analyse auxquels le système doit répondre, par la suite en les a formalisés graphiquement à l'aide des diagrammes offerts par le langage UML.

Ce stade de développement auquel nous sommes arrivés, nous permet aisément de mettre en œuvre l'application, en utilisant des techniques de développement qui feront l'objet du chapitre V de la partie suivante.

Bibliographie

Introduction :

Pour la réalisation de notre application nous avons eu recours à plusieurs architectures et outils de développement, nous les citons dans ce qui suit.

1. Architectures

1.1. Java EE (Java Entreprise Edition)[1]

La plate-forme Java EE s'appuie entièrement sur le langage Java. Java EE est donc une norme, qui permet à des développeurs de réaliser leur propre application qui implémente en totalité ou partiellement les spécifications de SUN. En simplifiant, il est possible de représenter Java EE comme un ensemble de spécifications d'APIs (ensemble de librairies et de services), une architecture et une méthode de packaging.

Il existe actuellement beaucoup d'autres plates-formes de développement qui sont basées sur d'autres langages (C#, PHP5, .NET ...). Les principaux avantages d'utiliser Java EE (est donc Java) sont la portabilité, l'indépendance, la sécurité et la multitude de librairies proposées. Le développement d'applications d'entreprise nécessite la mise en œuvre d'une infrastructure importante. Beaucoup de fonctionnalités sont utilisées et développées, le but étant de produire des applications sûres, robustes et faciles à maintenir. Certains services sont d'ailleurs récurrents comme : l'accès aux bases de données, l'envoi de mails, les transactions, la gestion de fichiers, la gestion d'images, le téléchargement, le chargement et la supervision du système...etc.

C'est pour cela que l'architecture Java EE est intéressante car tous les éléments fondamentaux sont déjà en place. Pas besoin de concevoir une architecture, des librairies et des outils spécialement adaptés. Cela nécessite un temps et un investissement considérables.

Enfin, la plate-forme Java EE est basée sur des spécifications, ce qui signifie que les projets sont portables sur n'importe quel serveur d'application conforme (Tomcat, JBoss, WebSphere et Weblogic...) à ces spécifications. Cette implémentation est gratuite et permet de bénéficier de la totalité de l'API sans investissement. La plate-forme Java EE est la plus riche des plates-formes Java et offre un environnement standard de développement et d'exécution d'applications d'entreprise multi-tiers.

Le fait que Java EE soit standardisé, il a contribué à son adoption par de très nombreux éditeurs de logiciels/outils informatique. Ces éditeurs associés à Sun Microsystems font partis du JCP (Java community Process). Le Java Community Process regroupe les entreprises suivantes :

Sun, IBM, Oracle, Borland, Nokia, Sony, la fondation Apache, ObjectWeb... etc. (*Pour plus de détails voir l'annexe*).

1.2. Design Pattern[2]

Un Design Pattern est une solution à un problème récurrent dans la conception d'applications orientées objet. Un patron de conception décrit alors la solution éprouvée pour résoudre ce problème d'architecture de logiciel.

Le concepteur objet tente de faciliter la réutilisation et la maintenance du code. On peut donc concevoir un modèle d'application comme une forme d'organisation transposable à plusieurs applications. Ces systèmes peuvent apparaître complexes aux débutants, voir inutiles, il est pourtant très important d'en connaître plusieurs et de les appliquer.

A noter qu'en se plaçant au niveau de la conception les Design Patterns sont indépendants des langages de programmation utilisés. (*Pour plus de détails voir l'annexe*).

1.3. Expresso :

Expresso est à la fois une architecture et une bibliothèque de composants qui nous aide à construire rapidement et facilement des applications web robustes et riches en fonctionnalités.

Expresso est basé sur Java qui permet cependant de programmer et met à la disposition de l'utilisateur une interface configurable composé minimalement d'une page graphique dans laquelle on peut dessiner à l'aide de commandes de style « géométrie de la tortue » et aussi des commandes de style « géométrie analytique ». On peut aussi compléter cette interface, à l'aide des commandes élémentaires, en y ajoutant des menus, des boutons, des glissières, une zone de texte, en utilisant des fenêtres de dialogue ou en réagissant aux clics ou aux « glisser » de souris, Son but est de servir de Framework pour créer des systèmes d'application orientée WEB. C'est aussi une bibliothèque de composants au sens où nous pouvons utiliser ses composants sans utiliser Expresso comme Framework.

1.3.1. Avantages techniques d'Expresso :

- **architecture J2EE basée sur Sun:**

  Expresso est entièrement J2EE (Java 2 Enterprise Edition), il est également compatible dans les environnements non-EJB.

- **Personnalisable:**

  Source complet signifie une plus grande souplesse pour la personnalisation à nos besoins.

- **Plate-forme et système d'exploitation indépendant:**

Expresso s'appuie sur J2EE pour assurer l'indépendance des systèmes d'exploitation et plates-formes.

- **Base de données indépendant:**

Expresso peut fonctionner sur une base de données relationnelle SQL.

- **Indépendance du serveur d'applications:**

Expresso peut être déployé avec succès / configuré avec divers servlet-conteneurs dont Tomcat, Resin, JRun et applications-serveurs compatibles J2EE, y compris mais sans s'y limiter: BEA WebLogic, IBM Websphere, JBoss, et iPlanet (NAS).

- **HTTPS Compatible:**

Expresso est capable de supporter le protocole HTTPS (HTTP sécurisé).

2. Développement

2.1. JAVA [3]

C'est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut-être utilisé sur internet pour des petites applications intégrées à la page web ou encore comme langage serveur .

2.2. HTML (Hypertext Markup Language)

HTML est un langage dit de «« marquage » ou de « balisage », est le format de données conçu pour représenter les pages web. C'est un langage de balisage qui permet d'écrire de l'hypertexte, d'où son nom.

HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des éléments programmables tels que des applets. Il est souvent utilisé simultanément avec des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade).

Le langage HTML est un langage qui peut être lue par des ordinateurs de différentes marques et aussi par des navigateurs divers.

2.3. JavaScript

JavaScript est un langage de script, qui permet notamment de manipuler les éléments d'une page Web, d'interagir avec le navigateur internet et de réagir aux actions de l'utilisateur.

Contrairement à certaines idées reçues, JavaScript est un langage moderne qui offre de nombreuses fonctionnalités puissantes comme :

- ❖ Le développent orienté objet ou encore la gestion des exceptions.
- ❖ Les fonctions du JavaScript ne nécessitent pas des accès aux serveurs pour fonctionner ce qui allège la page.

Le JavaScript est relativement simple et facile à apprendre. Il ne nécessite pas un programme spécial pour l'interpréter, ni pour l'écrire. De plus, JavaScript occupe un espace mémoire négligeable.

2.4. CSS (Cascading Style Sheets)

CSS (feuilles de styles en cascade), est un langage des styles, sert à personnaliser la présentation d'un document. Les feuilles de style en cascade CSS en particulier améliore l'apparence et la structure des documents HTML. Grâce aux CSS, on peut surtout définir pour une partie du texte ou pour tout un document les polices de caractères (type, taille, style), les couleurs, la présentation des images, des tableaux, les alignements, etc.

2.5. JSF (Java Server Faces)

JSF est un Framework qui a été programmé en Java. Faisant partie du standard J2EE, il offre au développeur une bibliothèque de composants et d'outils très large. Il permet de mieux organiser le code de l'application, séparer la vue (code html, JavaScript et CSS) du contrôleur (code java pur, lié au métier de l'application web) et du modèle (données et connexion à la base de données). JSF a deux avantages principaux :

- Un développement rapide grâce à ses composants.
- Une maintenance simplifiée grâce à sa structure.

Il permet :

- Une séparation nette de la couche présentation des autres couches d'une application.
- Une mise en place du mapping HTML/OBJET.
- Une liaison simple entre les actions côté client de l'utilisateur et le code Java correspondant côté serveur.

Il existe plusieurs Framework Web Java dédiés au développement d'interfaces utilisateur mais aucun n'est devenu un standard et est aussi performent que JSF.

2.6. RichFaces

Le projet RichFaces est une librairie de composants permettant l'intégration de comportement AJAX dans les vues, et ce, de manière très simple. Ce Framework s'appuie sur les composants JSF (Java Server Face). RichFaces permet aussi de définir des thèmes pour personnaliser simplement le rendu des pages.

2.7. JPQL (Java Persistance Query Language)

JPQL est un langage de requête orienté objet indépendant de la plateforme, défini dans la spécification Java Persistance API.

JPQL sert à exécuter des requêtes sur des entités persistées en base de données mais en travaillant sur les entités java correspondantes aux tables plutôt que sur les tables elles- mêmes.

Remarque :

Le tableau ci-dessous résume, sous forme de point, les critères de choix de chaque outil de développement. L'ensemble des choix sont exigés par l'organisme d'accueil.

Outil	Critères de choix
Java	- Richesse : un des aspects important de l'environnement de JAVA est sa richesse de ses librairies des classes JAVA. - Interprète, portable et indépendant des architectures matérielles. Cette caractéristique est un avantage primordial pour Java face à des applications transmises par un réseau et exécutées sur des machines hétérogènes.
HTML	- Simplicité (il est simple à utiliser). - Indépendance (sa conception lui permet de rester indépendant vis à vis des plateformes et de pouvoir être échangé sur les réseaux). - Aucun logiciel spécialisé n'est nécessaire pour composer des pages HTML, il peut être composé sur n'importe quel système avec un simple éditeur de textes.
JavaScript	- Vitesse(Les fonctions du JavaScript ne doivent pas attendre pour des réponses de leurs serveurs pour agir). - Simplicité (le JavaScript est relativement simple et facile à apprendre). - Versatilité - Le JavaScript ne nécessite pas un programme spécial pour l'interpréter, ni pour l'écrire.

CSS	- Permettre des choses irréalisables en html. - Allègement du code-source des pages Web donc de plus petits fichiers et un chargement plus rapide. - plus facile de faire des **changements d'ensemble** (un seul fichier CSS à modifier plutôt que toutes les pages une à une)
JSF	- La rapidité (un développement rapide grâce à ses composants). La maintenabilité (une maintenance simplifiée grâce à sa structure). - Organisation du code (Il sépare la présentation des traitements).
RichFaces	- Elle est implémentée par le Framework JSF. - Permettre des choses irréalisables en html.
JPQL	- Un langage indépendant du SGBD (Système de gestion de base de données) car il se base sur l'approche orienté objet (Il permet de manipuler des objets, mais pas des tables). - Il se base sur le concept de Java EE.

Tableau V.1 : les causes de choix d'outils de développement

3. Serveur de données

SQL SERVER 2005

SQL Server 2005 est un système de gestion de base de données relationnelle développée par Microsoft . Comme une base de données, il est un logiciel dont la fonction principale est de stocker, récupérer, traiter et sécuriser des données. Il fournit des accès contrôlés et des traitements de transactions rapides pour répondre aux besoins des applications les plus gourmandes en données utilisées au sein des entreprises. Il offre également les fonctions nécessaires pour faire face à des besoins de haute disponibilité.

4. Serveur d'applications

JBOSS 5.0.1

JBoss Application Server est un serveur d'applications J2EE Libre entièrement écrit en Java, publié sous licence GNU LGPL. Puisque le logiciel est écrit en Java, JBoss Application Server peut être utilisé sur tout système d'exploitation fournissant une machine virtuelle Java.

5. Environnement de développement

Eclipse 3.4 (Ganymede)

Eclipse IDE est un environnement de développement intégré libre, extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation.

Conclusion

Dans ce chapitre nous avons présenté les architectures ainsi que les outils et l'environnement de développement utilisés pour la réalisation de notre application, et mis en évidence les composants de la base de donnés.

1 voir Annexe 1
2 voir Annexe 2
3 voir Annexe 3

Introduction

Dans ce chapitre, nous commencerons par présenter le schéma fonctionnel de l'application, qui nous aidera à comprendre les fonctionnalités principales sur lesquelles nous nous sommes basés, ainsi que le déroulement de la navigation sur l'application. Ensuite nous allons passer au schéma applicatif qui a pour but de montrer les livrables que nous avons produit en respectant le principe de Java EE. Nous enchaînerons par la présentation du principe de fonctionnement de notre application en se basant sur le mouvement simple « déplacement» et finirons par donner quelques interfaces de notre application.

1. Le schéma fonctionnel de l'application

Le schéma fonctionnel ou l'arborescence d'une application, représente les différentes pages, organisées logiquement et hiérarchiquement.

L'arborescence d'une application est représentée sous la forme d'un arbre. La première page doit être la page d'accueil (appelée « la racine »), puis les autres pages apparaissent ensuite dans un ordre logique. L'arborescence d'une application aide l'internaute à comprendre la structure, la consultation et la mémorisation des pages est ainsi plus facile, rapide et efficace.

L'arborescence d'une application doit être au préalable réalisée sur papier avant de commencer la création. Elle est la base de toute application web à sa création.

Une fois l'utilisateur identifié grâce à son login et son mot de passe, le menu apparait et donne accès à 4 rubriques en principe, suivant le modèle que nous avons conçu (comme le schéma ci-dessus le montre). Mais dans la partie pratique de notre travail, nous nous sommes uniquement intéressés à réaliser le mouvement simple « déplacement » (tous les autres : paramétré le parc, visualiser le parc et recherche multicritères sont inhibés).

Dans ce module, l'utilisateur choisit le type de mouvement (dans notre cas déplacement), puis il saisir un code de dossier et sélectionne une marchandise à faire déplacer.

L'utilisateur à le chois d'effectuer le mouvement simple avec ou sans schéma, en suite éditer l'ordre de mouvement simple.

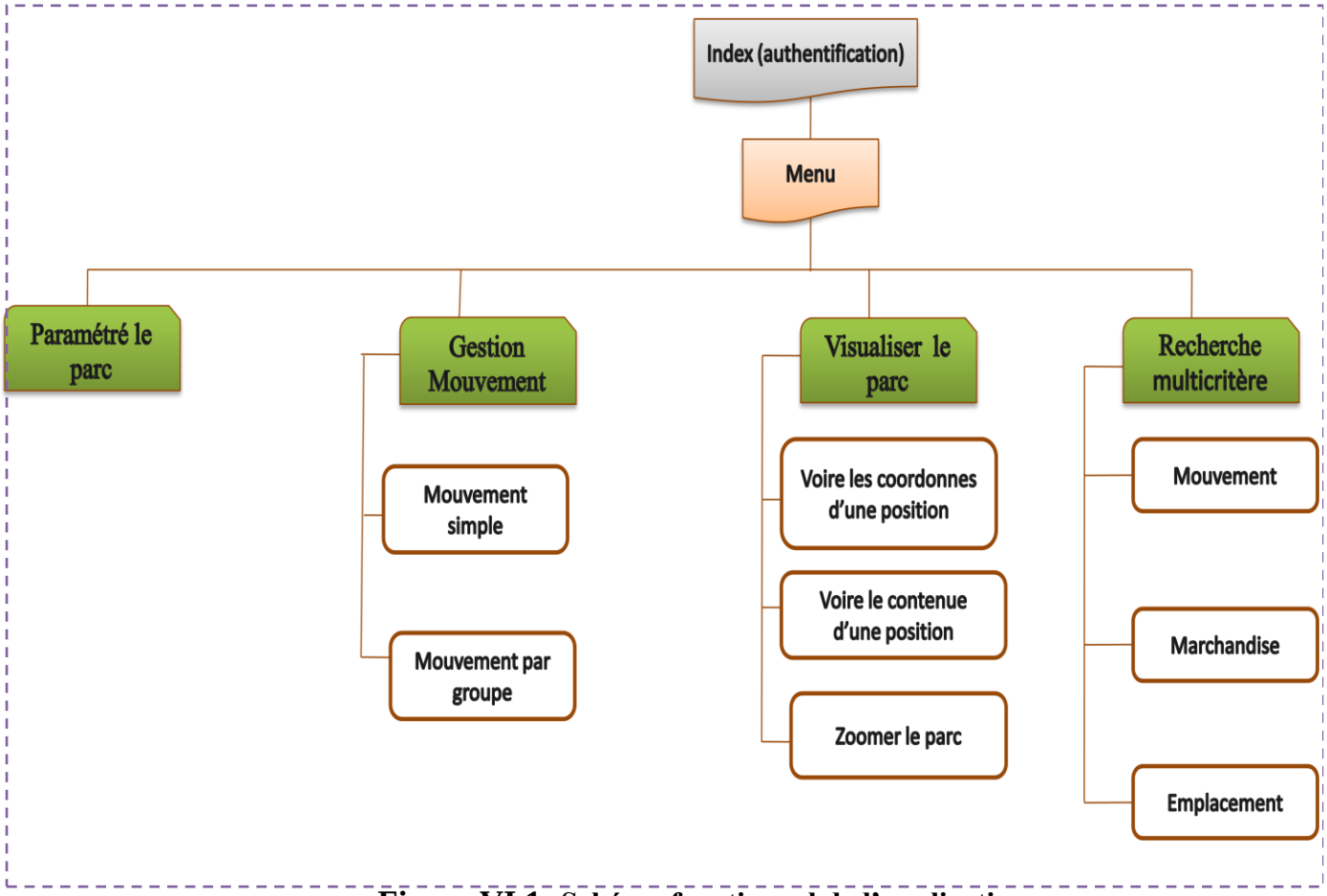


Figure VI.1: Schéma fonctionnel de l'application.

2. Le schéma applicatif de l'application

Développer une application Java EE revient à créer différents livrables, suivant la complexité des besoins de l'application.

Nous présentons ci-dessous le contenu de chaque livrable de notre application.

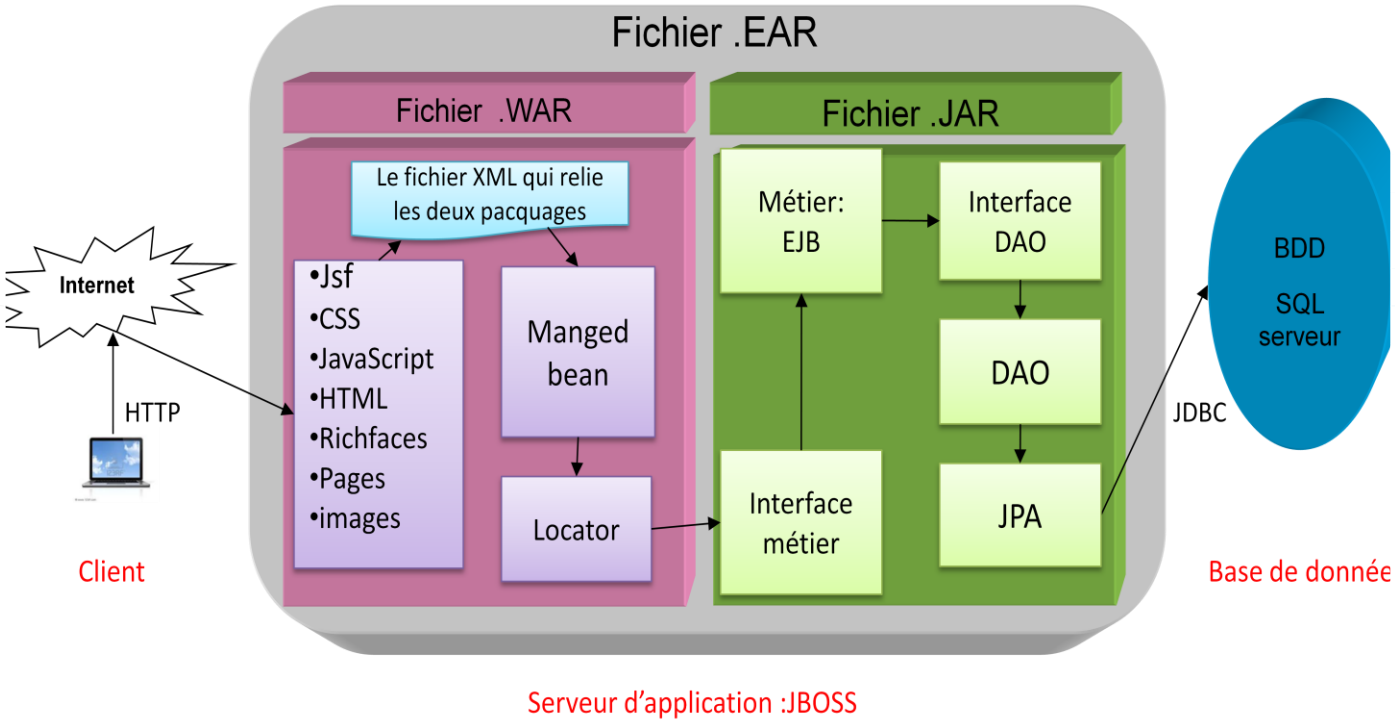


Figure VI.2: Schéma applicatif de l'application.

A. Livrable EAR (Entreprise ARchive)

Comme son nom l'indique ce format de fichier est utilisé pour emballer tous les composants d'une application Web : les livrables d'extension war et jar.

Le but de ce format d'archive est de rendre le déploiement et à son tour le processus de maintenance, plus facile. Cela aura un fichier XML nommé web.xml comme le descripteur de déploiement. Ce descripteur de déploiement est utilisé par le conteneur Web pour déployer l'application web correctement.

Ce fichier doit être déployé sur le serveur d'application J2EE (Jboss).

B. Livrable WAR (Web ARchive)

C'est un fichier utilisé pour distribuer:

- Ecrans de l'application (pages, HTML, JSF).
- Images de l'application.
- Eléments de graphisme (Feuilles de style CSS, JavaScript, richfaces).
- Classes Java métier (Managed Bean) : la fonction principale du managed bean est de sauvegarder des informations saisies dans le formulaire, ou récupérer depuis la base de données.

- Locator qui est une classe qui permet la communication entre la couche présentation et la couche métier, plus précisément entre le managed bean et couche métier.
- Fichiers de configuration permettant de configurer un service pour lequel la spécification J2EE n'est pas précise. On a utilisé deux fichiers de configuration : le fichier web.xml qui sert à configurer l'application web et le faces-config.xml qui contient tous les managed bean des formulaires.

C. Livrable JAR (Java ARchive)

Ce livrable contient :

- Les interfaces métier : Elles rendent les méthodes des classes métier visibles par les autres classes qui les utilisent (managed bean).
  Elles représentent la frontière de communication entre les classes métier et la classe locator.
- Les classes métier : ce sont les EJBs session, elles modélisent les processus métier de notre application (traitements).
- L'interface DAO : Elle permet de publier les méthodes de la classe DAO pour qu'elles soient utilisées par les classes métier.
- La classe DAO : c'est une classe qui implémente les quatre opérations de base pour la persistance des données (JAP), soient : **C**reate, **R**ead, **U**pdate, **D**elete.
- Les classes données (JPA) : ce sont les EJBs entity, elles représentent les données stockées dans la base de données.
- Un fichier de configuration Persistance.xml, permettant la connexion à la base de données.

3. Le schéma applicatif détaillé de l'application

Dans cette partie, nous allons détailler le contenu des livrables de notre application en présentant les différents packages de notre projet.

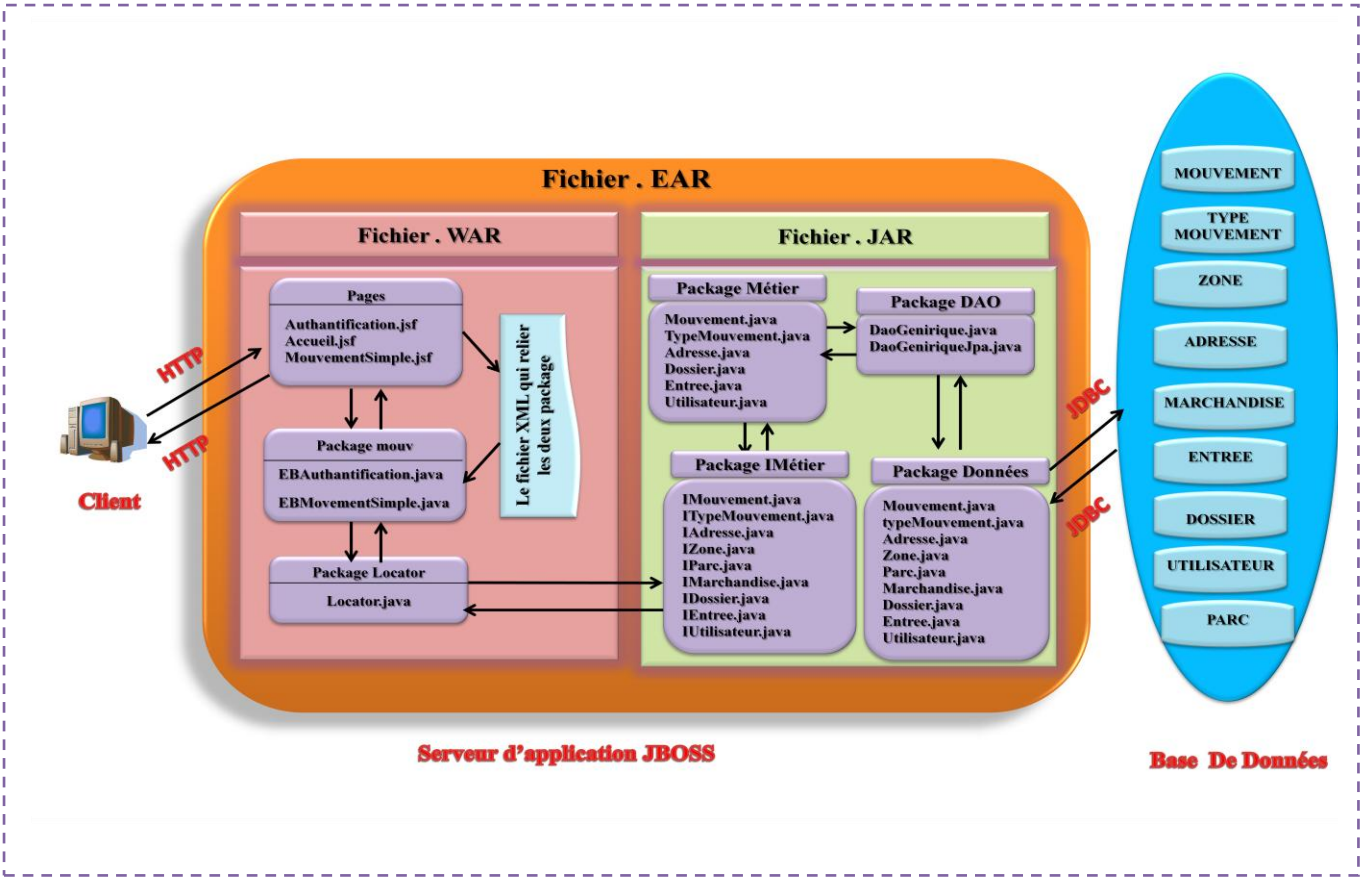


Figure VI.3: Schéma applicatif détaillé de l'application

❖ **Les différents packagent de notre projet :**

- ✓ **Un dossier de nos pages JSF** : Authantification.jsf, Menu.jsf, Mouvement.jsf, etc.
- ✓ **Package Mouv :** il contient l'EBMouvement.java, EBAuthantification.java.
- ✓ **Package Locator.java :** il contient la classe locator.java qui assure la communication entre le EBMouvement .java et l'interface IMouvement.java.
- ✓ **Package IMetier** : il contient l'interface IMouvement.java qui publie toutes les méthodes de la classe Mouvement.java.
- ✓ **Package Métier :** il contient les classes Mouvement.java, Entree.java, Adresse.java.Elles renferment toutes les méthodes de traitement.
- ✓ **Package DAO** : il contient deux classes : la classe DaoGeneriqueJpa.java qui implémente les quatre opérations de base CRUD et la classe DaoGenerique.java qui est une interface de DaoGeneriqueJpa.java.
- ✓ **Package données :** c'est une image de la base de données sous forme de classe.java.

4. Présentation de quelques interfaces

L'application que nous avons réalisé se présente sous une fenêtre principale qui permet à l'utilisateur de d'authentifier pour qu'il puisse accéder au menu principal puis aux fonctionnalités que comporte notre application.

4.1. Interface d'authentification

L'utilisateur doit saisir son nom d'utilisateur et son mot de passe. Si ces deux données sont correctes, le menu principal de notre application sera affiché sinon, un message d'erreur est affiché en rouge sur le formulaire d'authentification précisant à l'utilisateur l'erreur détectée.

L'interface d'authentification est représentée ci-dessous.

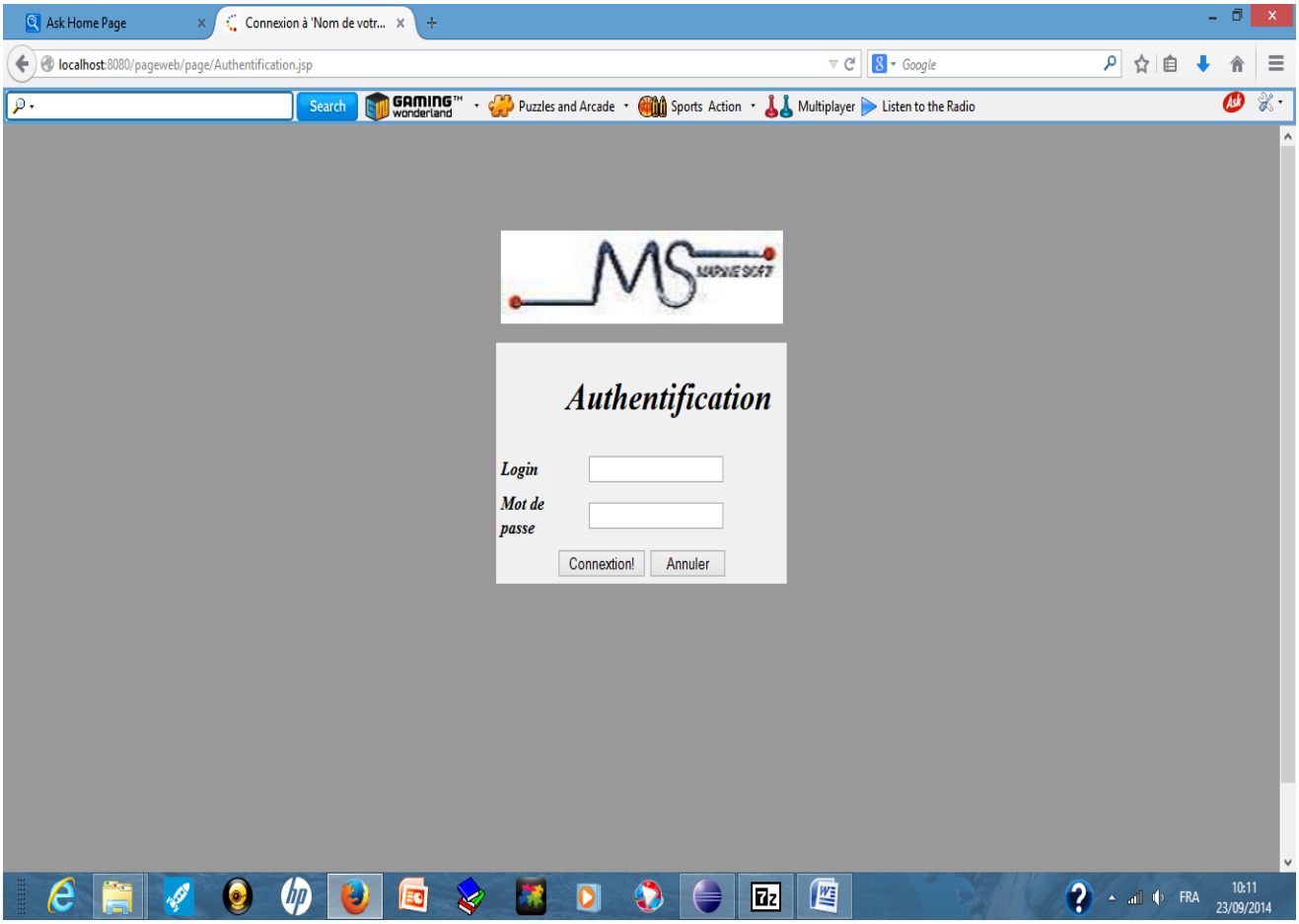


Figure VI.4:Interface d'authentification.

4.2. Interface du menu principal

Une fois l'utilisateur identifié grâce à son nom d'utilisateur et son mot de passe, le menu apparait et donne accès à 4 rubriques en principe :

1. Paramétrer le parc : destiner seulement a un utilisateur de type administrateur.
2. Gestion de mouvement :
   - Mouvement simple.
   - Mouvement par groupe.
3. Visualiser le parc :
   - Voire le contenue d'une position.
   - Voire les coordonnées d'une position
   - Zoomer le parc.
4. Recherche multi critères : c'est un outil de recherche des
   - Mouvement (par un code de mouvement, type de mouvement, date de mouvement… etc.)
   - Adresses (par adresse occupes, adresse vides …etc.).
   - Marchandise (par code d'entreé de marchandise, numéro de dossier de la marchandise).

La figure suivante illustre cette page :

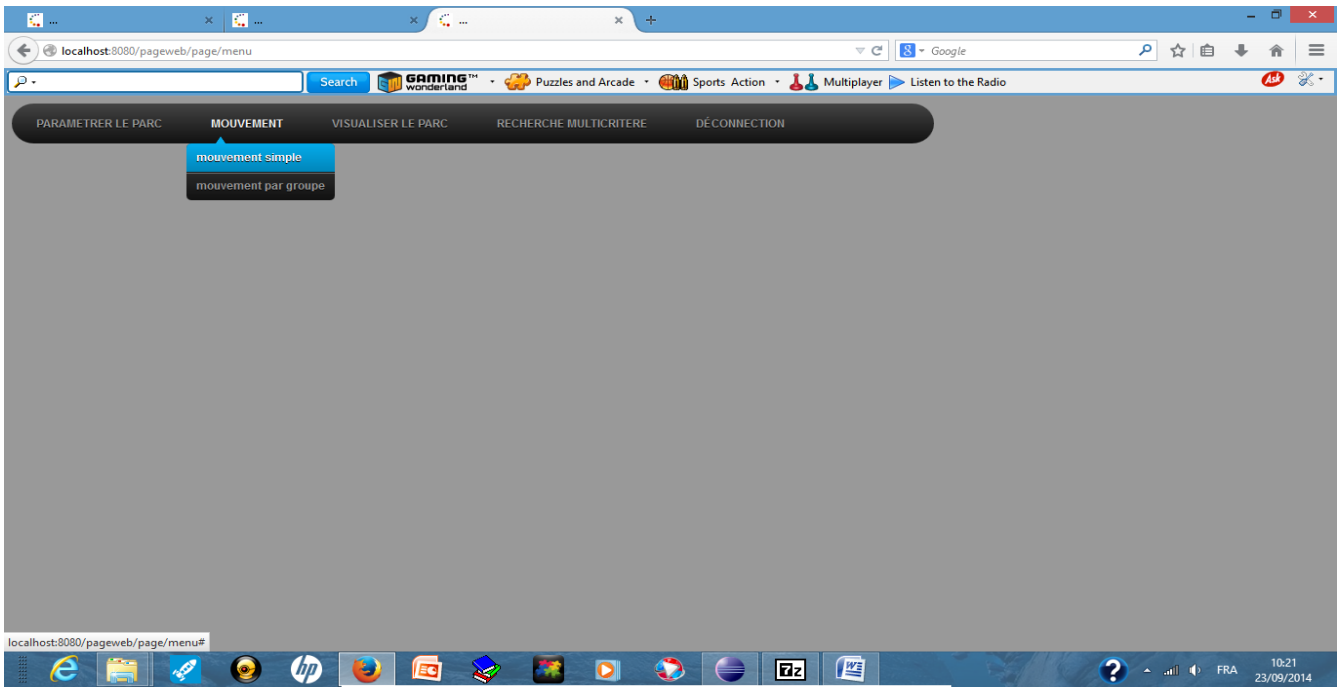


Figure VI.5:Interface du menu principal.

Voici le formulaire de mouvement simple « déplacement simple ».

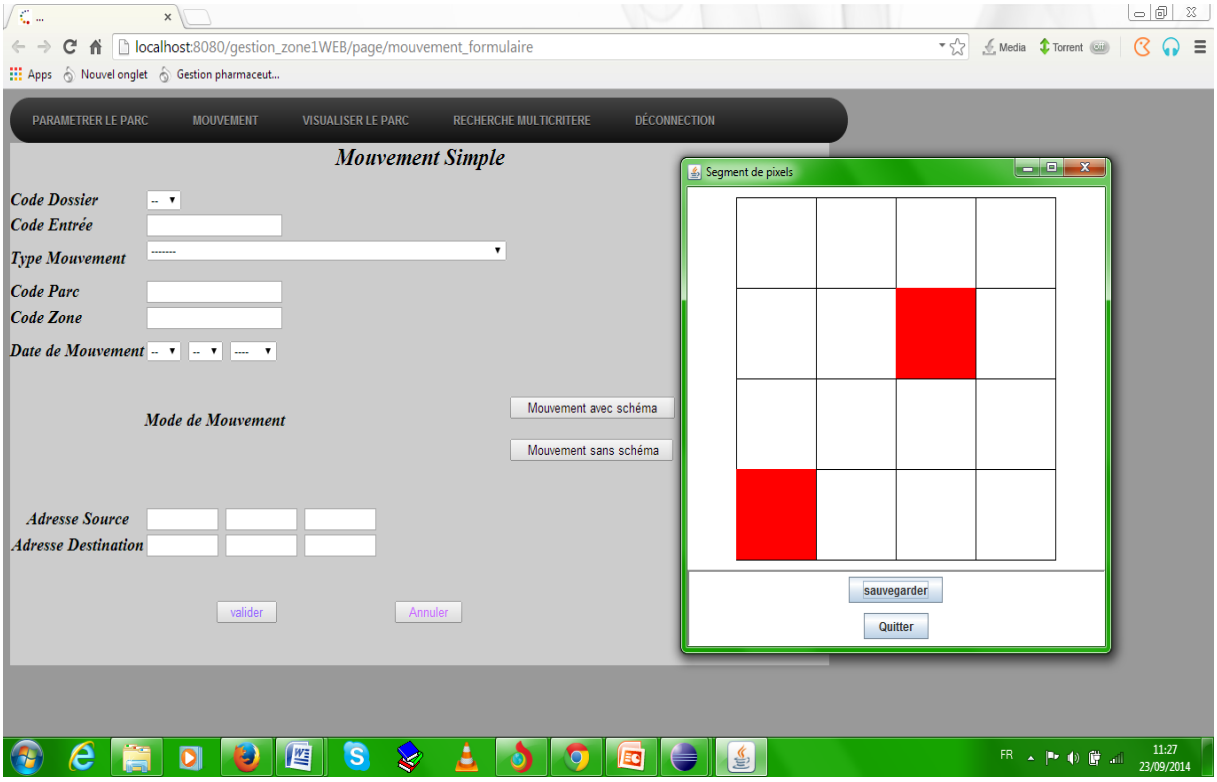


Figure VI.7:Interface de mouvement simple « entreé ».

- L'utilisateur doit saisir le numéro de dossier, pour que le système lui affiche la liste des marchandises à faire déplacer depuis la base de données.
- L'utilisateur entre les informations nécessaires et effectuer le mouvement selon le mode « mouvement avec schémas ».

Conclusion

Dans ce chapitre, nous avons abordés la réalisation de notre application d'une manière générale. Nous avons insisté sur les schémas fonctionnel et applicatif et la présentation des interfaces .Il nous a encore permis de mettre en pratique les notions théoriques de Java EE. Autrement dit, nous avons pu séparer la couche présentation, couche métier et celles de la couche base de données. s

.

****Conclusion générale ****

La conception et la réalisation d'un système d'information standard pour la gestion des d'activités dans les aires de stockages, était la mission qui nous a été confiée. Les étapes qui ont été suivies pour l'aboutissement de ce travail étaient :

- L'analyse et l'étude de l'existant, qui nous a permis de prendre connaissance de la situation de logiciel De gestion de zone (ZSD) produit par l'organisme d'accueil.
- L'identification des besoins qui nous a permis de concevoir notre application, la modéliser en utilisant le langage UML.
- La réalisation de l'application en se basant sur la plate-forme Java EE.

Ce travail nous a été bénéfique car il nous a permis d'acquérir des nouvelles connaissances et une certaine expérience au sein d'un nouveau milieu, d'appliquer nos connaissances théoriques acquises durant notre cursus universitaire, et aussi de :

- Elargir nos connaissances et découvrir une des architectures d'application distribuée à base de composants qui est Java EE.
- Se familiariser avec les outils de développement et de nouveaux Framework: Eclipse, Jboss, JSF…etc.
- S'adapter au travail de groupe.

Ce travail n'est pas une fin en soi. Il peut toutefois être complété. En effet, il ne couvre qu'une partie des fonctionnalités conçues. De ce fait, des perspectives d'amélioration suivantes sont ouvertes :

- L'implémentation des autres fonctionnalités.
- L'implémentation d'une couche de communication avec les autres modules de l'ERP.

Ce projet informatique reste une modeste contribution dans le domaine des applications web.

Nous espérons qu'il constituera une source d'inspiration bénéfique pour les futurs étudiants qui aborderont ce genre de thème.

Annexes

- ***Annexe I :*** *Architecteur Client -Serveur*
- ***Annexe II :*** *La plateforme Java EE.*
- ***Annexe III :*** *Programmation Oriente Object.*
- ***Annexe VI :*** *Design Pattenes.*

1) Introduction

L'enjeu pour les entreprises est de réaliser une intégration de l'informatique personnelle dans le système informatique de l'entreprise avec l'objectifs de permettre à tout utilisateur dans l'entreprise d'accéder a toute information utile à sa tache dès lorsque celle-ci est automatisée par les règles de confidentialité et de sécurité en vigueur. L'accès doit être instantané et doit pouvoir être à partir de n'importe quel poste de travail, en plus l'accès à l'information doit avoir lieu par une interface aussi simple que possible. Pour cela une solution est apportée dans les années 90, qui est le modèle client serveur.

2) Architecture centralisée(Mainframe) :

Dans ce type d'architectures, les composants matériels (tels que les unités de stockage, le(s) processeur(s) (CPU), les lecteurs de bande, etc.) étaient centralisés et les utilisateurs équipés de « simple » terminaux. (Les terminaux n'étaient utilisés qu'afin d'afficher les données, les calculs mais également la préparation de la présentation étaient effectués sur le Host, les terminaux étant passif)

Les utilisateurs se connectent aux applications exécutées par le serveur central (le mainframe) à l'aide de terminaux passifs se comportant en esclaves. C'est le serveur central qui prend en charge l'intégralité des traitements, y compris l'affichage qui est simplement déporté sur des terminaux passifs.

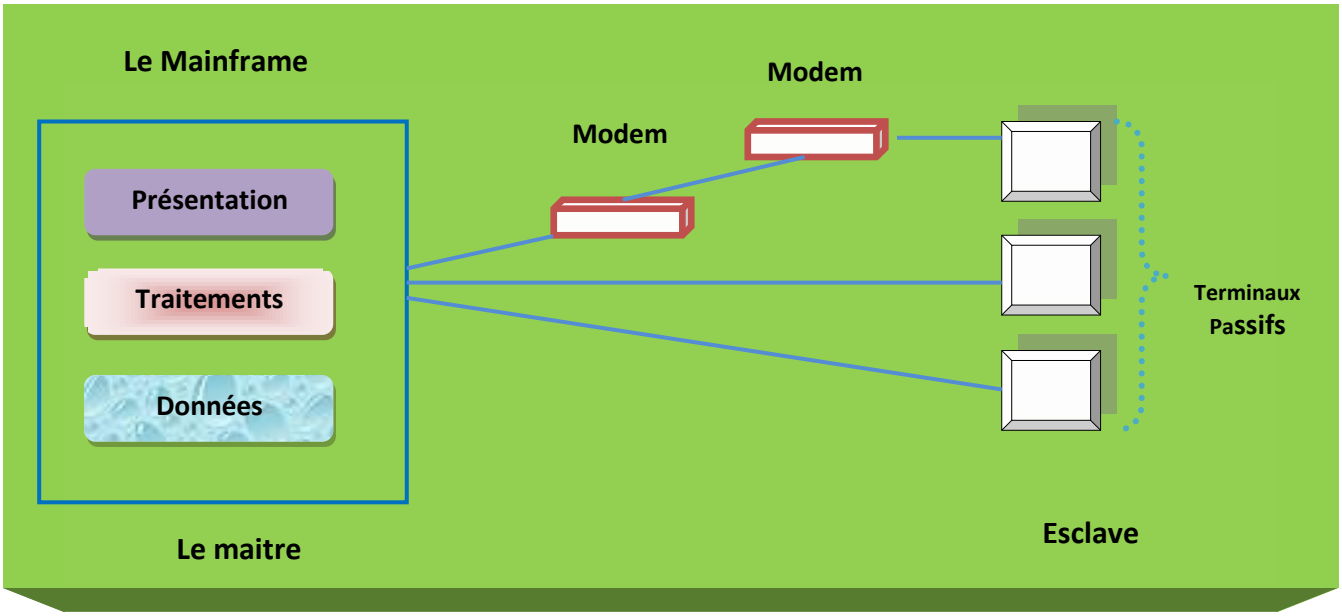


Fig.1 : schéma d'une architecture Mainframe

2.1) Avantages et limites :

- **Avantage:**

-Performance : la centralisation de la puissance sur une seule et même machine permet une utilisation optimale des ressources.

-Sécurité et fiabilité.

- Facilité d'administration.

- **Limite :**

- Les applications sur site central souffrent d'une interface utilisateur en mode caractères et la cohabitation d'applications micro exploitant des données communes n'est pas fiable au delà d'un certain nombre d'utilisateurs.

2) L'architecteur client – serveur :

2.1) Définition :

L'architecture client-serveur désigne l'ensemble des moyens matériels, logiciels de communication permettant de mettre en place une application conforme au modèle client-serveur, ce modèle est un mode de fonctionnement théorique basé sur la séparation des rôles. Il décrit le mode de fonctionnement des applications.

3) Généralités sur le modèle client-serveur :

3.1) Les technique de dialogue client /serveur *:*

Le client/serveur est avant tout un mode de dialogue entre deux processus, le premier s'appelle « client » et le second s'appelle « serveur ». Le modèle de communication client/serveur est orienté vers la fourniture de services par un processus serveur à un processus client. Un échange consiste en transmission d'une requête à un processus serveur qui exécute l'opération demandée et envoie en retour la réponse au client. Nous définissons ci-dessous plus précisément ces concepts de base. On définit trois acteurs principaux pour l'architecture client/serveur :

- ✓ **Client:**

C'est une entité (processus, programme, ordinateur…) qui demande l'exécution d'une opération à une autre entité par envoie d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un massage en retour.

✓ **Serveur:**

C'est une entité (processus, programme…) qui accomplisse une opération sur la demande d'un client et lui transmet la réponse.

✓ **Requête :**

C'est un message transmis par un client à un serveur décrivant l'opération à exécuter pour le compte du client.

✓ **Réponse:**

C'est le message de retour, transmis par le serveur suite à l'exécution de l'opération formulée dans la requête. La réponse contient les paramètres de retour de l'opération.

Un client exécute une application et demande l'exécution d'une opération à un serveur par le biais d'une requête. Il reçoit une réponse, lui indiquant que l'opération à été bien exécutée. Les appels mis en jeu sont au nombre de quatre :

- **Send request ( ) :**

Permet au client d'émettre le message décrivant la requête à une adresse correspondant au port d'écoute de serveur.

- **Receive request ( ) :**

Permet au serveur de recevoir la requête sur son port d'écoute.

- **Send reply ( ) :**

Permet au serveur d'envoyer la réponse sur le port d'écoute du client.

- **Receive reply ( ) :**

Permet au client de recevoir la réponse en provenance du serveur.

3.2) Caractéristique de modèle client/serveur [2]*:*

Bien que le Client/serveur soit une forme d'informatique distribuée, il n'est supposé d'appeler systèmes Client/serveur que ceux qui partagent les caractéristiques suivantes :

- **Services** : Le modèle client serveur est une relation entre des processus qui tournent sur des machines séparées. Le serveur est un fournisseur de services. Le client est un consommateur de services.
- **Partage des ressources :** un serveur est sensé pouvoir traiter plusieurs clients simultanément et gérer leurs accès au ressources.

➢ **Asymétrie des protocoles :** le client est toujours le déclencheur de dialogue par une demande de service .et le serveur attend passivement les requêtes de client.

➢ **Echange de messages** : Client et serveur sont des systèmes à liaison épisodique qui interagissent au moyen de messages. Le message est le mécanisme d'émission des demandes de service et de réponses à celles ci.

➢ **Encapsulation des services:** le message envoyer indique au serveur quel service est requis. C'est à lui de décider comment rendre ce service. Les services peuvent être mis à niveau sans effet sur les clients tant que l'interface des messages reste la même.

➢ **Redimensionnement:** Les systèmes client/serveur peuvent être redimensionnés horizontalement ou verticalement. Le redimensionnement horizontal correspond à l'ajout ou au retrait de stations clientes, avec un léger impact sur les performances. Le redimensionnement vertical correspond à la migration du serveur vers une machine plus puissante ou consiste à distribuer la charge de traitement sur plusieurs serveurs.

➢ **Intégrité:** Le code et les données des serveurs sont gérés de façon centraliser ce qui garantie un moindre coût de maintenance et une meilleure intégré des données partagées .De l'autre coté, les clients restent individuels et indépendants.

3.3) Les modules de l'architecteur client serveur :

Architecteur client serveur est compose de trois modules.ces module sont repartis sur le client et sur le serveur :

- interface utilisateur ou présentation.
- traitement ou de logique applicative.
- gestion de données.

3.4) Les modèles de client serveur selon (*Gartner Group):*

Le processus client demande l'exécution de services au serveur qui retourne les réponses. Le serveur doit bien sûr être en mesure de traiter les requêtes de plusieurs clients. La classification du Gartner Group dépend de la répartition de trois fonctions :

❖ **Modèle de Gartner pour les systèmes à deux niveaux (2-tiers)**

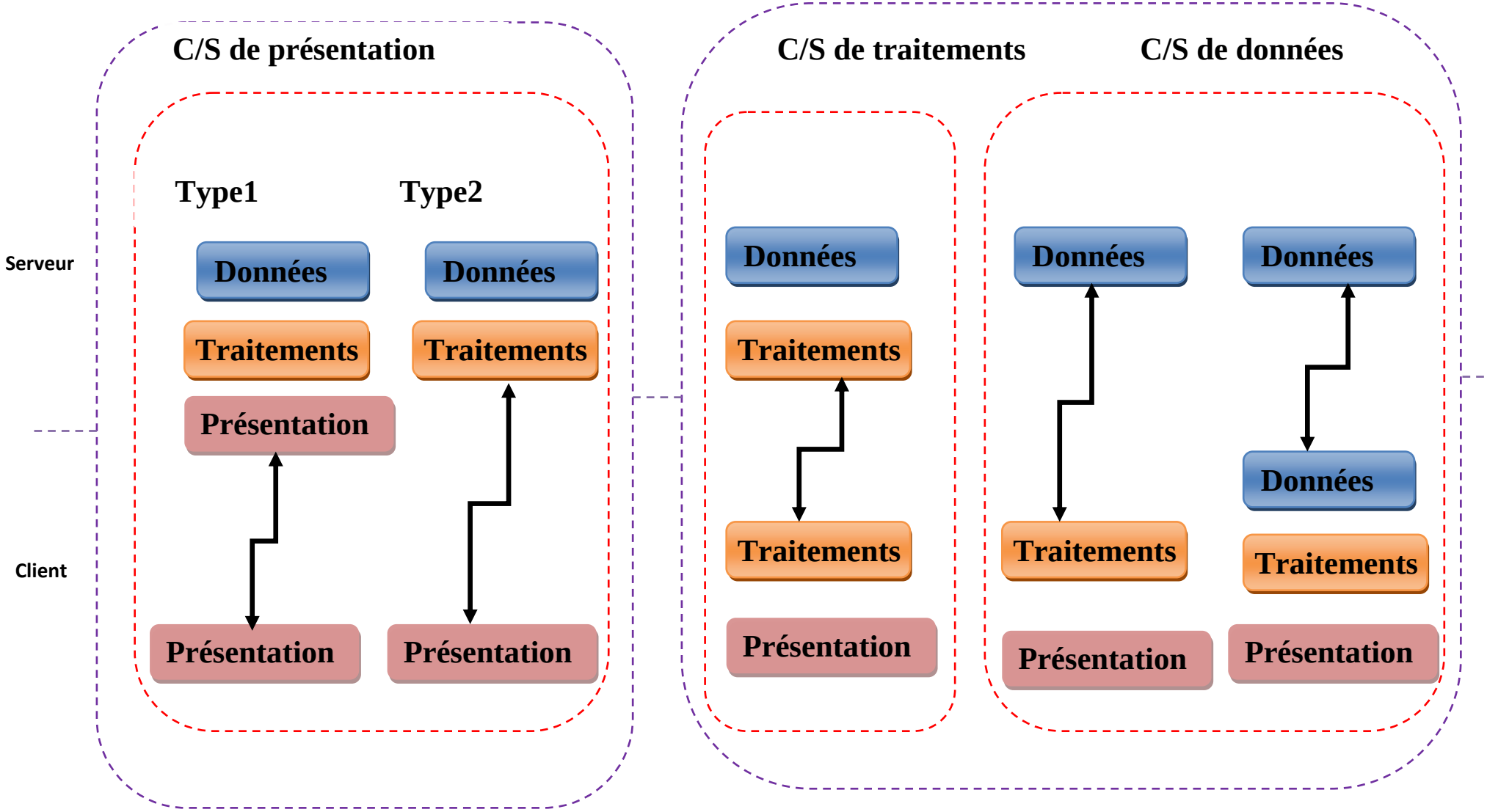


Fig.2 : les déférentes types de C/S selon classification de Gartner Group

- **Client-serveur de données** : Type de client-serveur dans lequel un programme applicatif contrôlé par une interface de présentation sur une machine cliente accède à des données sur une machine serveur par des requêtes de recherche et mis à jour, souvent exprimée avec le langage SQL.
- **Client-serveur de présentation** : Type de client dans lequel un processus exécute seulement les fonctions de dialogue avec l'utilisateur ; l'autre gérant les données et exécutant le code applicatif
- **Client-serveur de procédure** : type de client-serveur dans lequel un programme applicatif contrôlé par une interface de présentation sur une machine cliente sous-traite l'exécution de procédures applicatives à une machine serveur, ces procédures encapsulant souvent une base de données.

3.5) Type des clients :

❖ **Client léger** : Le terme « client léger », désigne une application accessible via une interface web (en HTML) consultable à l'aide d'un navigateur web, où la totalité du logique métier est traitée du côté du serveur. Le fait que l'essentiel des traitements soit réalisé du côté du serveur permet une grande souplesse de mise à jour.

❖ **Client lourd** : Le terme « client lourd », désigne une application cliente graphique exécutée sur le système d'exploitation de l'utilisateur. Un client lourd possède généralement des capacités de traitement évoluées et peut posséder une interface graphique sophistiquée. Néanmoins, ceci demande un effort de développement et tend à mêler la présentation avec les traitements

❖ **Client riche** : Un « client riche » est un compromis entre le client léger et le client lourde L'objectif du client riche est donc de proposer une interface graphique, décrite avec une grammaire de description basée sur la syntaxe XML, permettant d'obtenir des fonctionnalités similaires à celles d'un client lourd (glisser déposer, onglets, multi fenêtrage, menus déroulants). Les clients riches permettent ainsi de gérer l'essentiel des traitements du côté du serveur. Les données sont ensuite transmises dans un format d'échange standard utilisant la syntaxe XML, puis interprétées par le client riche.

3.6) Type des serveurs :

- **Serveur de base de données :**

Dans ce cas du serveur, le client émet des requêtes SQL sous forme de messages en direction du serveur. Les données ainsi que le code qui traite les requêtes résident sur la même machine serveur. Le stockage et le traitement de ces données sont réalisés par le serveur Exemple : Oracle, Microsoft Server SQL.

- **Serveur de fichiers** :

Le serveur gère la gestion des fichiers. L'écriture et lecteur des fichiers se fait en émettant des requêtes sur un réseau par le client vers le serveur et ce type de serveur Permet de partager des fichiers sur un réseau et il est indispensables pour créer des banques de documents, d'images.

- **Serveur de transaction** :

Dans ce modèle le client invoque des procédures distantes résidant sur le serveur qui comporte un moteur de base de données SQL. Ces procédures distantes exécutent un ensemble d'instructions SQL. L'échange sur le réseau consiste en un seul message requête/réponse (contrairement à l'application serveur base de données pour laquelle le message requête/réponse est émis pour chaque instruction SQL). Pour ce type de serveurs l'application client/serveur nécessite du code source au niveau du Serveur.

4) Les différentes générations de client-serveur :

4.1) client-serveur 2tiers :

4.1.1) Présentation :

Dans l'architecture à 2 tiers La gestion des données est prise en charge par un SGBD (*system de gestion de base de donnes*) centralisé, s'exécutant le plus souvent sur un serveur dédié. Ce dernier est interrogé en utilisant un langage de requêtes qui le plus souvent est SQL. Le dialogue entre client et serveur se résume donc à l'envoi de requêtes et au retour des données correspondant aux requêtes. La figure suivante illustre ce type d'architecture :

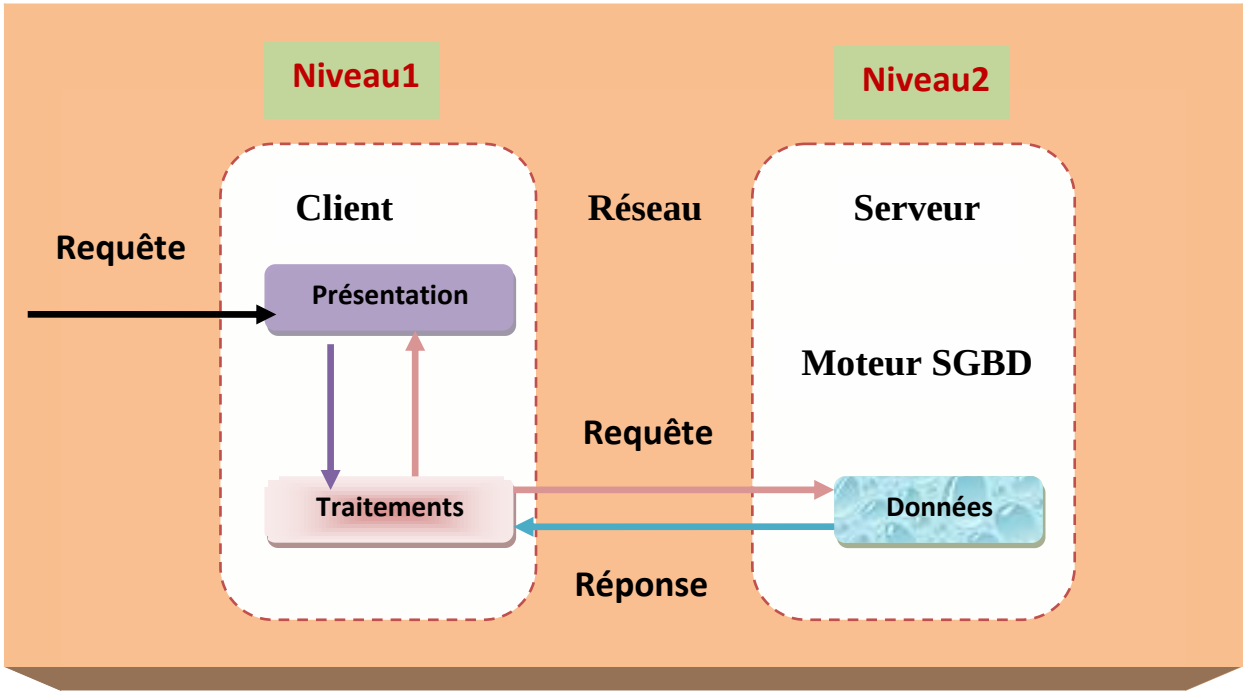


Fig.2.Architecture client-serveur 2tiers

4.1.2) Limites de client-serveur 2tiers :

La simplicité est le facteur principal qui explique la popularité de l'architecture client/serveur à deux niveaux, cette dernière offre une grande souplesse et facilite la création d'outils permettant aux utilisateurs de créer leur application. Elle est considérée comme une bonne solution informatique lorsque le nombre d'utilisateurs ne dépasse pas la centaine.

Ce modèle était une solution pour les petites applications mais avec l'arrive des nouvelles technologies l'architecteur client-serveur a 2tiers a montra ses limites.

L'origine des limites présentées par cette architecteur est la localisation de la logique applicative sur le poste client qui complique la maintenance et la réutilisation de code. En effet l'expérience à démontré qu'il était coûteux de vouloir faire porter l'ensemble des traitements applicatifs par le poste client qui devient de ce fait un client lourd. Les limites du client-serveur deux-tiers peuvent se résumer essentiellement à :

on ne peut pas soulager la charge du poste client, qui supporte la grande majorité des traitements applicatifs, le poste client est fortement sollicité, il devient de plus en plus complexe et doit être mis à jour régulièrement pour répondre aux besoins des utilisateurs, les applications se prêtent assez mal aux fortes montées en charge car il est difficile de modifier l'architecture initiale, la relation étroite qui existe entre le programme client et l'organisation de

la partie serveur complique les évolutions de cette dernière, ce type d'architecture est grandement contracté par les coûts et la complexité de sa maintenance.

Pour résoudre les limitations du client-serveur deux tiers tout en conservant ses avantages, on a cherché une architecture plus évoluée, facilitant les forts déploiements à moindre coût. La réponse est apportée par les architectures distribuées.

4.2) Client-serveur 3tiers

4.2.1) Présentation :

Dans ce type d'architecture, un niveau médian est ajouté entre les deux niveaux précédents, permettant de séparer les traitements de l'interface graphique et du serveur de base de données. Trois types de plates-formes doivent donc supporter ces composants. Autrement dit, avec le client/serveur 3-tiers, le client fournit l'interface graphique et interagit avec le serveur par des appels de services distants. La logique applicative se trouve dans le niveau médian. Les traitements deviennent des entités de première importance, gérées et déployées indépendamment de l'interface utilisateur et de la base de données. La logique applicative qui a maintenant son propre niveau séparé, peut même s'exécuter sur un ou plusieurs serveurs.

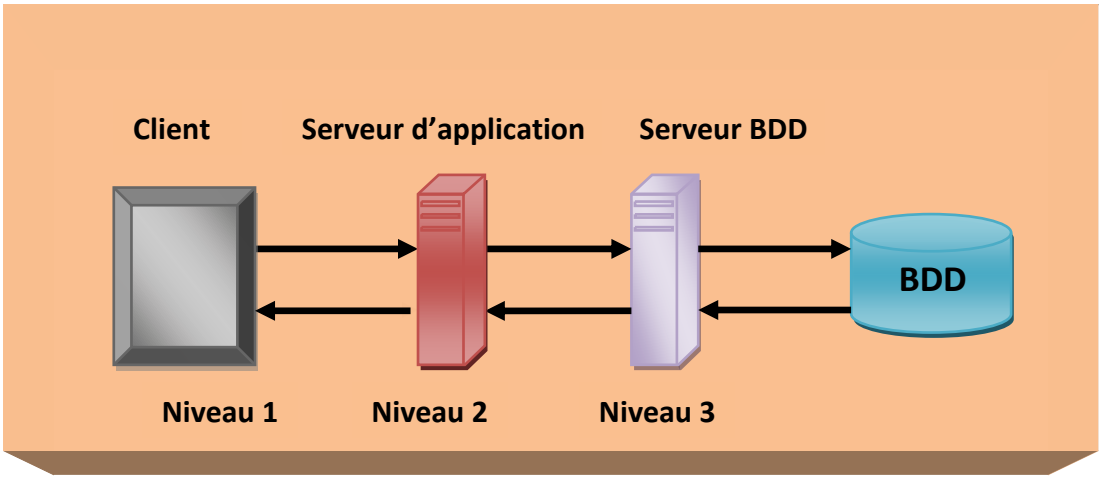


Fig.3. Architecture client/serveur à 3 tiers

4.2.2) Les avantages de l'architecteur 3tiers :

Cette architecture se développe actuellement au sein des entreprises grâce aux nombreux avantages qu'elle présente. L'avantage principal d'une architecture 3-tiers est la facilité de déploiement, L'application en elle même n'est déployée que sur la partie serveur (serveur applicatif et serveur de base de données). Le client ne nécessite qu'une installation et une configuration minime, Cette facilité de déploiement aura pour conséquence non seulement de réduire le coût de déploiement mais aussi de permettre une évolution régulière du système. Le deuxième avantage est l'amélioration de la sécurité car l'accès à la base de donne n'est effectué que par le serveur applicatif. Ce serveur est le seul à connaitre la façon de se connecter à cette base. Il en résulte que ses performances et son niveau de sécurité sont plus élevés que ceux du modèle 2-tiers. En théorie, les systèmes client/serveur à trois niveaux, plus robustes et plus souples. De plus, ils peuvent intégrer des données venant de sources multiples.

4.3) Client-serveur N tiers :

4.3.1) Présentation :

Nous avons vu ce qu'était une architecture trois-tiers, une architecture multi-tiers va plus loin dans le découpage de l'application sur différents serveurs. Le découpage de base du système reste toujours le même, une partie gestion de données, une partie gestion de la logique applicative et bien entendu le client assurant la présentation. Toutefois les deux parties développées coté serveur vont pouvoir être déployées chacune sur plusieurs serveurs.

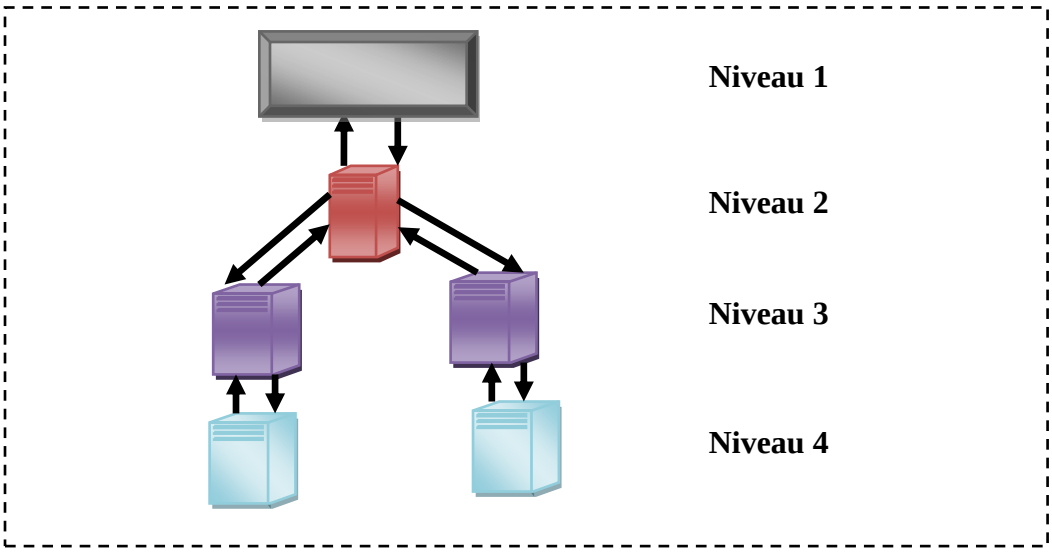


Fig.4. Architecture client/serveur multi tiers

4.3.2) Avantages de l'architecteur N tiers :

Plusieurs avantages sont offerts par ce type d'architecteur :

- **Développement de grandes applications par étapes :**

Les architectures basées sur les composants permettent de développer tant de grandes applications à missions critiques que de petits projets.

- **Réutilisation des composants :**

Les composants sont ici réutilisables sous forme de 'boites noires' binaires qui peuvent être recombinées de diverses façons en fonction des applications.

- **Accès fiable des clients aux données :**

Les clients envoient des requêtes vers des composants pour les exécutée. Les composants serveur encapsulent le détail de la logique applicative, augmentant ainsi le niveau d'abstraction.

5) Le Middleware :

Une application peut désormais être constituée de centaines de morceaux qui auront été écrits indépendamment les uns des autres. Mais pour que le niveau médian (qui renferme les composants) agisse comme s'il était un système (ou une application) unique, il à besoin de middleware qui permettra d'unifier, pour les applications, l'accès et la manipulation de l'ensemble des services réseau.

5.1) Définition :

Ensemble des services logiciels construits au-dessus d'un protocole de transport afin de permettre l'échange de requêtes et des réponses associées entre client et serveur de manière transparente.

5.2) Les services d'un middleware

Il assure les services suivants :

- ✓ **Conversion**: Service utilisé pour la communication entre machines mettant en œuvre des formats de données différents
- ✓ **Adressage** : Permet d'identifier la machine serveur sur laquelle est localisé le service demandé afin d'en déduire le chemin d'accès.
- ✓ **Sécurité** : Permet de garantir la confidentialité et la sécurité des données à l'aide de mécanismes d'authentification et de cryptage des informations.

- ✓ **Communication** : Permet la transmission des messages entre les deux systèmes sans altération. Ce service doit gérer la connexion au serveur, la préparation de l'exécution des requêtes, la récupération des résultats et la déconnexion de l'utilisateur.

Remarque

Le middleware masque la complexité des échanges inter-applications et permet ainsi d'élever le niveau des API utilisées par les programmes. Sans ce mécanisme, la programmation d'une application client-serveur serait complexe et difficilement évolutive.

5.3) Les objectifs :

- Les transports des requêtes et des réponses.
- La simplification de la session utilisateur.
- L'harmonisation des types de données.
- Les performances et la fiabilité.

D'autre part le middleware à une double mission d'interface. Avec le réseau il assure la mise en forme de la donnée en vue de sa prise en charge par la couche transport. Avec le processus client ou serveur, il permet la gestion des appels de fonctions de l'application ou la gestion du renvoi des résultats.

5.4) Les avantages :

- Il offre des services de haut niveau aux applications.
- Il rend portable les applications.
- Il prend en charge les protocoles de conversion des caractères.
- Il établit des sessions entre clients et serveurs hétérogènes.

Conclusion :

Après la présentation de l'architecture client serveur on a constate que cette dernière ni qu'une transformation des applications monolithiques vers des applications distribuées entre le client et le serveur. Grâce a la répartition des taches entre ces deux processus on arriver à distinguer les différente générations de cette architecteur, et cette repartions est pour le but d'améliore l'ouverture du système d'information d'entreprise et accélère le déploiement des architectures réparties et hétérogènes.

Le modèle 2-tiers est loin d'être obsolète. Il existe un grand nombre d'applications idéales pour les architectures 2-tiers.par exemple dans les petits projets, les applications 2-tiers sont plus faciles à développer que celles 3-tiers (ou n-tiers).

En général Avec les applications n-tiers, on dispose enfin d'une vision cohérente du système d'information. Tous les services sont représentés sous la forme d'objets interchangeables qu'il est possible d'implanter librement, en fonction des besoins.

Introduction :

De nombreuses possibilités existent pour réaliser des applications Internet depuis plusieurs années. Des langages ont été crées, des architectures et des environnements de travail ont été conçus pour répondre aux besoins et faciliter la tâche des développeurs. Sun (le concepteur de Java) a donc mis en place un ensemble de technologies pour réaliser des applications Web. Ces technologies sont regroupées sous le nom J2EE (Java 2 Entreprise Edition), désormais Java EE.

1) pourquoi utiliser une plateforme [1]

Une plateforme est une base générique qui fournit un ensemble de fonctionnalités utiles pour une majorité d'applications. Une plateforme se construit sur la base d'un ensemble de besoins génériques partagés entre plusieurs applications.

Il peut exister plusieurs types de plateformes. De la plus générique à la plus spécifique (optimisée pour un type de métier précis par exemple). Bon nombre de grandes entreprises ont déjà développé des plateformes tels que : IBM (Web Sphère) …etc.

L'avantage principal de partir d'une plateforme est que l'équipe de développement n'a pas à s'acquitter de développer certaines tâches. Ce sont des tâches que l'on retrouve très souvent dans un grand nombre de projet et qui n'ont pas d'intérêt à être recoder à chaque fois (perte de temps et d'argent). De plus, mieux vaut travailler sur une plateforme qui présente une forte stabilité.

Un autre avantage est la facilité de prise en main des API de la plateforme. En effet, celle-ci cache très souvent la complexité d'accès à telle ou telle ressource et permet donc un gain de temps énorme pour le développeur qui a donc plus de temps pour se préoccuper du fonctionnement réel de son application (pas de tâche difficiles ou générique à développer).

2) La présentation de JAVA EE [1]

J2EE (Java 2 Enterprise Edition) est une norme proposée par la société Sun, portée par un consortium de sociétés internationales, visant à définir un standard de développement d'applications d'entreprises multi-niveaux, basées sur des composants.

On parle généralement de «plate-forme JAVA EE» pour désigner l'ensemble constitué des services (API) offerts et de l'infrastructure d'exécution. JAVA EE comprend notamment :

- **Les spécifications du serveur d'application**, c'est-à-dire de l'environnement d'exécution JAVA EE. Définit finement les rôles et les interfaces pour les applications ainsi que l'environnement dans lequel elles seront exécutées. Ces recommandations permettent ainsi à des entreprises tierces de développer des serveurs d'application conformes aux spécifications ainsi définies, sans avoir à redévelopper les principaux services.

- **Des services, au travers d'API,** c'est-à-dire des extensions Java indépendantes permettant d'offrir en standard un certain nombre de fonctionnalités. Sun fournit une implémentation minimale de ces API appelée JAVA EE SDK (J2EE Software Développent Kit). Dans la mesure où JAVA EE s'appuie entièrement sur le Java, il bénéficie des avantages et inconvénients de ce langage, en particulier une bonne portabilité et une maintenabilité du code.

JAVA EE permet une grande flexibilité dans le choix de l'architecture de l'application en combinant les différents composants. Ce choix dépend des besoins auxquels doit répondre l'application mais aussi des compétences dans les différentes API de JAVA EE.

L'architecture d'une application se découpe idéalement en au moins trois tiers :

- **la partie cliente :** c'est la partie qui permet le dialogue avec l'utilisateur. Elle peut être composée d'une application web ou d'applets
- **la partie métier :** c'est la partie qui encapsule les traitements (dans des EJB ou des JavaBeans)
- **la partie donnée :** c'est la partie qui stocke les données

3) L'architecteur JAVA EE

3.1) présentation de l'architecteur

L'architecture JAVA EE permet ainsi de séparer la couche présentation, correspondant à l'interface homme-machine (IHM), la couche métier contenant l'essentiel des traitements de données en se basant dans la mesure du possible sur des API existantes, et enfin la couche de données correspondant aux informations de l'entreprise stockées dans des fichiers, dans des bases de données relationnelles ou XML, dans des annuaires d'entreprise ou encore dans des systèmes d'information complexes

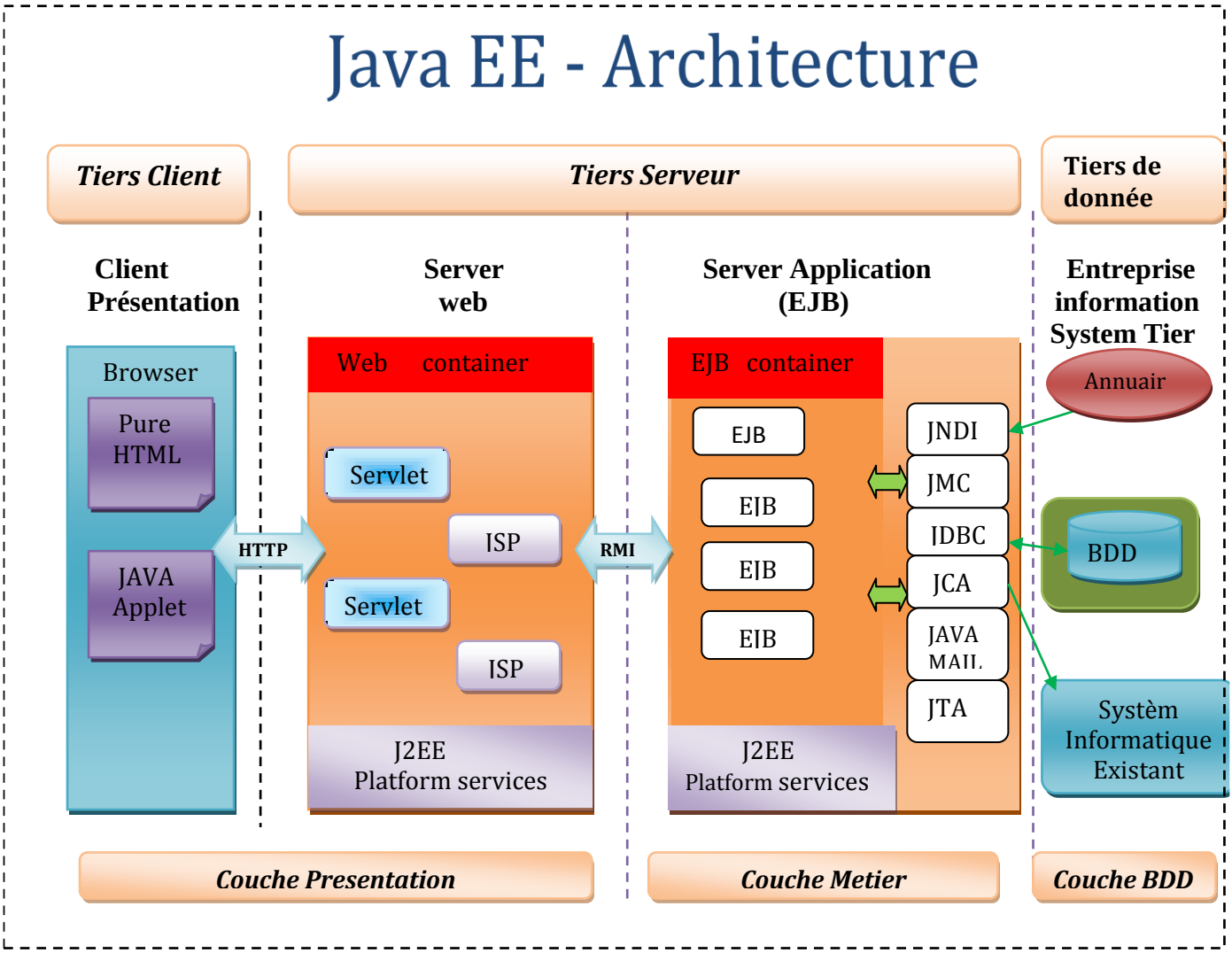


Fig. VI.1 : Architecteur j2ee en couches

L'architecture présentée dans la figure ci-dessus est un exemple type composée de plusieurs couches. Elle nécessite un travail de personnalisation en fonction du projet à développer avant de permettre la mise en œuvre d'application à mettre en place. Les couches de base qui la composent sont les suivantes :

- **La couche présentation (tiers Web) :**

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. On parle alors d'interface homme machine (IHM) et aucun traitement n'est implémenté dans cette couche.

- **La couche métier (tiers Métier ou tiers Business):**

Elle se décompose en 3 parties comme illustre cette figure:

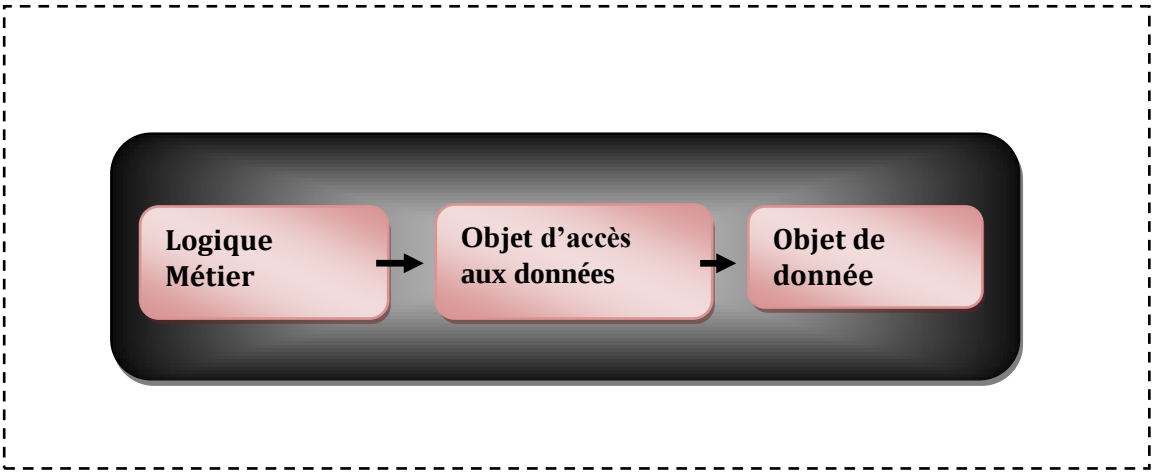


Fig. VI.2. : La couche Métier.

- **Le logique métier**

Le logique métier correspond à la partie fonctionnelle de l'application, qui est responsable de l'implémentation de la « logique », décrivant les opérations que l'application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation. Les différentes règles de gestion et de contrôle du système sont mises en œuvre dans cette couche. La couche métier offre des services applicatifs et métiers à la couche présentation. Pour fournir ces services, elle s'appuie sur les données du système, accessibles au travers des services de la couche d'accès aux données. En retour, elle renvoie à la couche présentation les résultats qu'elle à calculés.

➢ **Couche d'accès aux données**

Elle consiste en la partie gérant l'accès aux de données du système. Ces données peuvent être propres au système, ou gérées par un autre système. La couche métier n'a pas à s'adapter à ces deux cas, ils sont transparents pour elle, et elle accède aux données de manière uniforme.

➢ **Objets de données**

Elle assure la persistance des données, à chaque table de la base de données correspond une classe entité

- **La couche base de données :**

Elle inclut la base de donnes.

4) Les API de la plateforme JAVA EE

4.1) Définition d'une API

***Application Programming Interface* :** Une interface de programmation est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les autres, de manière analogue à une interface homme-machine, qui rend possible l'interaction entre un homme et une machine.

Du point de vue technique une API est un ensemble de fonctions, procédures ou classes mises à disposition par une bibliothèque logicielle, un système d'exploitation ou un service. La connaissance des API est indispensable à l'interopérabilité entre les composants logiciels.

4.2) Les trois catégories d'API

J2EE définit un certain nombre de services accessibles par le biais d'APIs normalisées. La plupart de ces services répondent à des problématiques récurrentes rencontrées dans les applications d'entreprise, notamment l'accès aux applications patrimoniales, aux annuaires, ainsi la gestion de la sécurité.

Les API peuvent être regroupées en trois grandes catégories :

❖ **Les composants web & métier**

Les composants sont des bibliothèques de code Java qui décrit les services qu'ils fournissent, les contrats qu'ils respectent (comportement transactionnel, persistance, etc.), ainsi que leurs dépendances vis-à-vis d'autres composants ou des éléments de l'environnement d'exécution, On distingue deux familles de composants :

- **Les composants web :**

Il s'agit de la partie chargée de l'interface avec l'utilisateur (on parle de logique de présentation).

a) JSP

Les JSP (Java Server Page) sont les pages servant à générer l'ensemble du code HTML de l'interface utilisateur. On y intègre aussi bien du code HTML que des scriplet Java (code java) ou encore des balises personnalisées (tag-lib). Cette technologie est donc dédiée à la génération de HTML et non au traitement de la requête de l'utilisateur. On l'appelle généralement : Vue.

b) Servlet

Une Servlet est une classe Java qui permet de traiter une requête venant d'un client. Cette technologie doit s'occuper de traiter les données envoyées par l'utilisateur et choisir la Vue à retourner à celui-ci. On appelle cette partie : Contrôleur. En général, la classe Java ne doit quasiment pas générer de code HTML (sauf dans certains cas précis)

- **les composants métier :**

EJB (Entreprise JavaBean) :

Il s'agit de composants spécifiques chargés des traitements des données propres à un secteur d'activité (on parle de logique métier ou de logique applicative) et de l'interfaçage avec les bases de données. On parle de la partie : Modèle

Les EJB forment l'une des spécifications majeures de JAVA EE et utilisent le principe des annotations Java. Il se décline en:

- **entités (*Entity Bean*) :** permettent de représenter une donnée persistante ayant une existence au niveau d'un support de stockage telle une base de données relationnelle.
- **sessions (*Session Bean*) :** sont utilisées pour représenter les données d'un client et les comportements qui leur sont associés.
- **messages *(Message Bean*)** : permettent la mise en œuvre d'applications basées sur un traitement asynchrone de messages. Pour fonctionner une application utilisant ce type de composants devra s'appuyer sur un service JMS (Java Messages Service). Ils sont apparus avec le standard EJB 2.0.

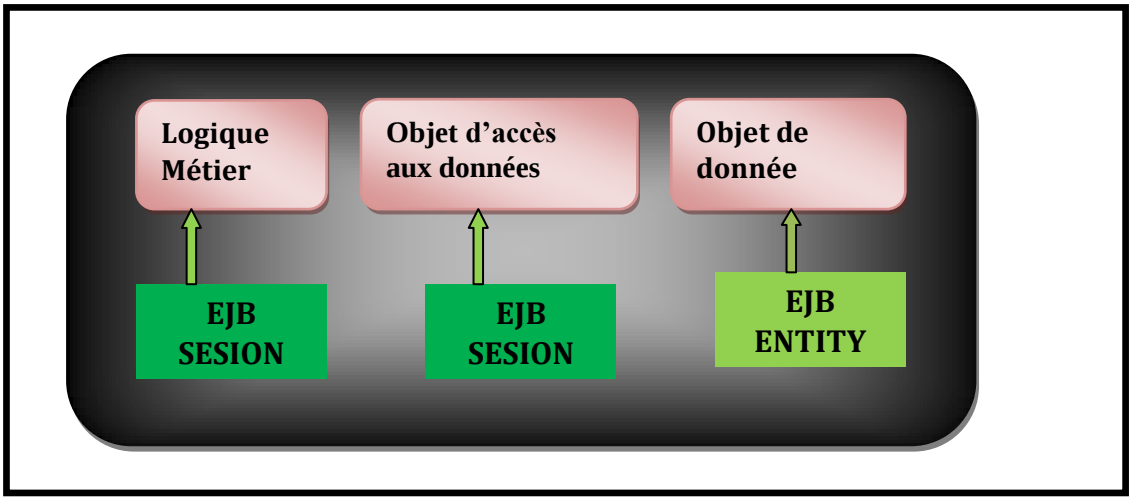


Fig. VI.3. L'emplacement des EJBs dans la couche métier.

- **Services d'infrastructures :**

Il en existe un grand nombre, en citeront **:** JDBC, JNDI, JCA (voire le tableau)

- **Les services de communication**:

Il en existe un grand nombre, en citeront **:** JAAS, Java Mail, JMS, RMI-IIOP**5**

(Voire le tableau).

4.3) Présentation de quelque API.

JAVA EE regroupe un ensemble d'API pour le développement d'applications d'entreprise.

API	**Rôle**
Entreprise Java Bean (EJB)	Composants serveurs contenant le logique métier
Remote Method Invocation (RMI) et RMI-IIOP	RMI permet l'utilisation d'objets Java distribués. RMI-IIOP est une extension de RMI pour une utilisation avec CORBA.
Java Naming and Directory Interface (JNDI)	Accès aux services de nommage et aux annuaires d'entreprises
Java Database Connectivity (JDBC)	Accès aux bases de données. J2EE intègre une extension de cette API
Java Transaction API	Support des transactions

(JTA) Java Transaction Service (JTS)	
Java Messaging service (JMS)	Support de messages via des MOM (Messages Oriented Middleware)
Servlets	Composants basés sur le concept C/S pour ajouter des fonctionnalités à un serveur. Pour le moment, principalement utilisé pour étendre un serveur web
Java Server Pages (JSP)	
Java IDL	Utilisation de CORBA
JavaMail	Envoi et réception d'e-mails
J2EE Connector Architecture (JCA)	Connecteurs pour accéder à des ressources du système d'information de l'entreprise telles que CICS, TUXEDO, SAP
Java API for XML Parsing (JAXP)	Analyse et exploitation de données au format XML
Java Authentication and Authorization Service (JAAS)	Echange sécurisé de données
JavaBeans Activation Framework	Utilisé par JavaMail : permet de déterminer le type mime

5) Le serveur d'application JAVA EE

5.1) Qu'est ce qu'un serveur d'application JAVA EE [2] :

Les serveurs d'application sont des logiciels qui aident des entreprises à développer, déployer et contrôler un grand nombre d'applications qui sont la plupart du temps distribuées. Du point de vue développement, la différence principale apportée par un serveur d'application est la séparation du logique métier de la logique présentation et de la logique base de données. Essentiellement, les serveurs d'application nous aident à démontrer des applications 3-tiers où la base de données est logiquement séparée (parfois physiquement séparé aussi) du logique métier. Il prend en charge l'ensemble des fonctionnalités qui permettent à N clients d'utiliser une même application.

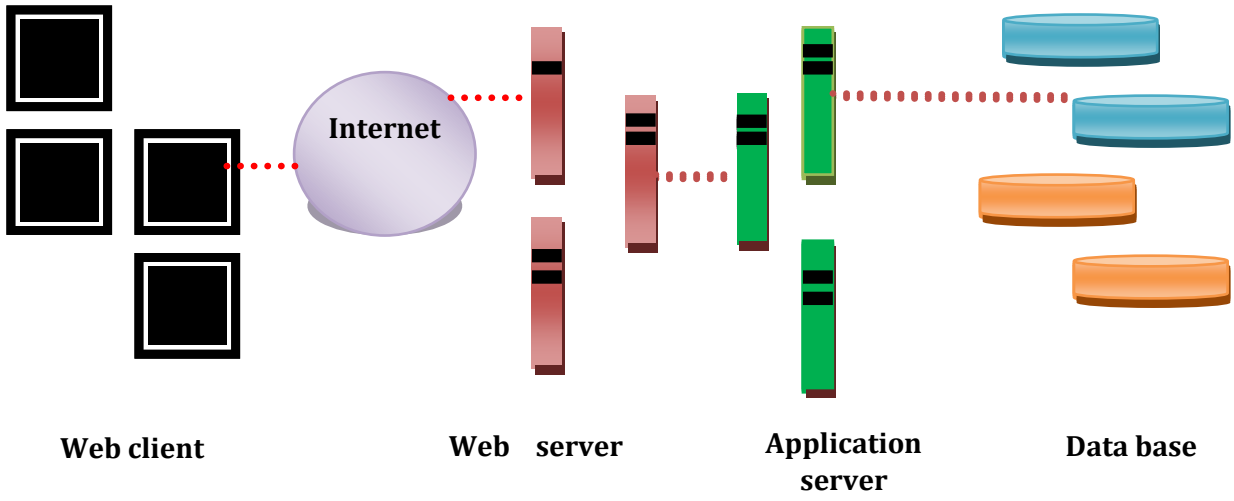


Fig. 1.4-Modèle de serveur d'application

5.2) Les avantages des serveurs d'application

Les serveurs d'application possèdent un grands nombre d'avantages :

- **Client léger :** Fournit un client beaucoup plus léger.
- **Intégrité de données et du code :** En centralisant le logique métier sur un ou sur quelque machines serveur, des mises à jour et des mises à niveau des applications pour tous les utilisateurs peuvent être garanties. Il n'y a aucun risque de vieilles versions d'accès aux données ou manouvrantes d'application d'une façon plus ancienne et incompatible.
- **Configuration centralisée :** Des changements de configuration d'application, telle qu'un mouvement des serveurs de base de données, peuvent être faits de manière centralisé.
- **Sécurité :** Un point central par lequel l'accès aux données ou parties de l'application elle-même peut être contrôlé est considéré comme un avantage de sécurité, plaçant la responsabilité de l'authentification loin de la couche potentiellement peu sûre de client sans exposer la couche de base de données.

5.3) Quelques serveurs d'application JAVA EE [2]

Glassfish, BEA Weblogic, JBoss, IBM Webspher, JAVA EE x, JOnAS, Tomcat ...etc.

5.4) Les conteneurs [1] :

Un conteneur JAVA EE est un environnement d'exécution Java chargé de gérer des composants applicatifs et de donner accès aux API JAVAEE. Outre l'identité associée à l'environnement d'exécution, J2EE ne spécifie aucune identité pour les conteneurs. Ce degré élevé de souplesse permet une grande variété de fonctionnalités au sein de l'environnement d'exécution du conteneur. Il existe deux types de conteneurs:

5.4.1) Le conteneur Web :

Est un environnement d'exécution pour les pages JSP et les Servlets qui constituent une passerelle entre l'interface utilisateur et les EJB qui implémentent le logique métier.

5.4.2) Le conteneur EJB :

Est un environnement d'exécution pour les EJB, il héberge les composants métiers et leur fournit les services nécessaires.

Remarque :

JAVA EE comporte également deux autres types de conteneurs :

- Un conteneur d'applets permettant d'exécuter des applets Java.
- Un conteneur d'application clientes permettant d'exécuter des applications Java classiques côté client.

6) Avantages de la norme J2EE [2] :

La spécification J2EE fournit une norme qui permet de protéger les investissements liés à l'acquisition ou au développement d'applications. Une série complète de tests de compatibilité indépendants garantit le respect des normes JAVA EE par les fournisseurs.

Le déploiement d'une architecture compatible JAVA EE offre les avantages suivants :

- Une architecture simplifiée basée sur des composants, des services et des clients standards. Cette architecture tire parti de la possibilité de réutiliser les éléments après leur création grâce à la technologie JAVA.

- Des services permettant l'intégration aux systèmes existants, notamment JDBC (Java DataBase Connectivity), JMS (Java Message Service), JCA (Java Connector Architecture), Java IDL (Java Interface Définition Language), l'API JavaMail et les API JTA et JTS (Java Transaction API) pour des transactions métier fiables.
- Des fonctions évolutives permettant de répondre à la demande, obtenues, par exemple, en répartissant les conteneurs sur plusieurs systèmes et en regroupant les connexions de base de données.
- Un bon outil de développement d'applications et des composants provenant de fournisseurs qui offrent des solutions standard.
- Un modèle de sécurité souple qui assure la prise en charge de la fonction de connexion unique, l'intégration avec les schémas de sécurité existants et une approche unifiée de sécurisation des composants d'application.

Conclusion :

Nous soumettons au premier lieu à la présentation de la plateforme JAVA EE .cette derniere elle offre une ensemble de technologies permettant le développement des applications multi tiers, stables, sécurisées et portables au niveau des entreprises grâce à l'utilisation des composants.

Notre travail s'est consacré ensuite à l'étude de l'architecture Java EE en couche qui ce base sur l'architecteur 3 tiers, puis nous avons aussi aborde le concept de serveur d'application qui permet la séparation du logique métier de la logique présentation et de la logique base de données, afin de fournit un client beaucoup plus léger, une configuration centralisée, et une forte Sécurité.

Nous nous sommes rendu compte qu'elle est appropriée à notre travail car elle rassemble à la fois les avantages d'une plate-forme, de Java et ceux de l'architecture trois tiers.

Introduction :

La solution vers laquelle tend l'informatique aujourd'hui est l'utilisation de la technologie objet (apparition de normalisation objet comme CORBA, ActiveX...etc.). Cette technologie bouleverse les idées reçues en plaçant au centre du développement les objets (et non pas les traitements). En l'adaptant au monde des bases de données, elle bouleverse une fois encore les méthodes de développement en focalisant l'attention sur les objets du métier c'est à dire sur le cœur de la logique d'entreprise. S'appuyer sur une technologie orientée objet pour construire une architecture orientée objet, s'est s'assurer de ne pas se retrouver en marge des standards de demain.

1) Définition POO :

L'Orienté Objet (ou OO) où on manipule uniquement des objets c'est-à-dire des ensembles groupés de variables et de méthodes associées à des entités intégrant naturellement ces variables et ces méthodes.

L'idée de base de la programmation orientée objet est de rassembler dans une même entité appelée objet les données et les traitements qui s'y appliquent.

2) Définitions des concepts de bases POO [3].

2.1) Classe :

Une classe regroupe un ensemble de données (qui peuvent être des variables primitives ou des objets) et un ensemble de méthodes de traitement de ces données et/ou de données extérieures à la classe.

Pour écrire un programme avec un langage orienté-objet, le programmeur écrit uniquement des classes correspondant aux objets de son système. Les traitements à effectuer sont programmés dans les méthodes de ces classes qui peuvent faire appel à des méthodes d'autres classes. En général, on définit une classe, dite "exécutable", dont une méthode peut être appelée pour exécuter le programme.

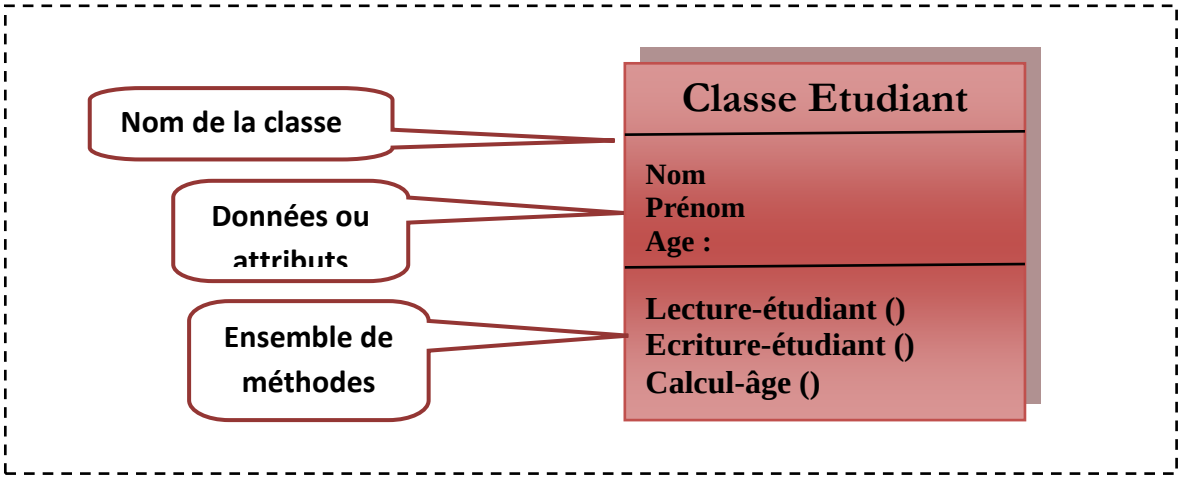


Fig.1 Représentation de la classe Etudiant

- **Les classes abstraites [1] :**

-La classe abstraite est une classe qu'on ne peut pas directement instancier car certaines de ses méthodes ne sont pas implémentées. Une classe abstraite peut donc contenir des variables, des méthodes implémentées et des signatures de méthode à implémenter.

-Une classe abstraite peut hériter d'une classe ou d'une classe abstraite. Le mot-clé abstract est utilisé devant le mot-clé class pour déclarer une classe abstraite, ainsi que pour la déclaration de signatures de méthodes à implémenter.

-Une classe est automatiquement abstraite dès qu'une de ses méthodes est déclarée abstraite. Il est possible de définir une classe abstraite sans méthodes abstraites.

- **Classes, objets, instances**

L'opération ***d'instanciation*** qui permet de créer un objet à partir d'une classe consiste précisément à fournir des valeurs particulières pour chacun des attributs d'instance c'est-à-dire chaque objet aura sa propre copie de ses données

Exemple

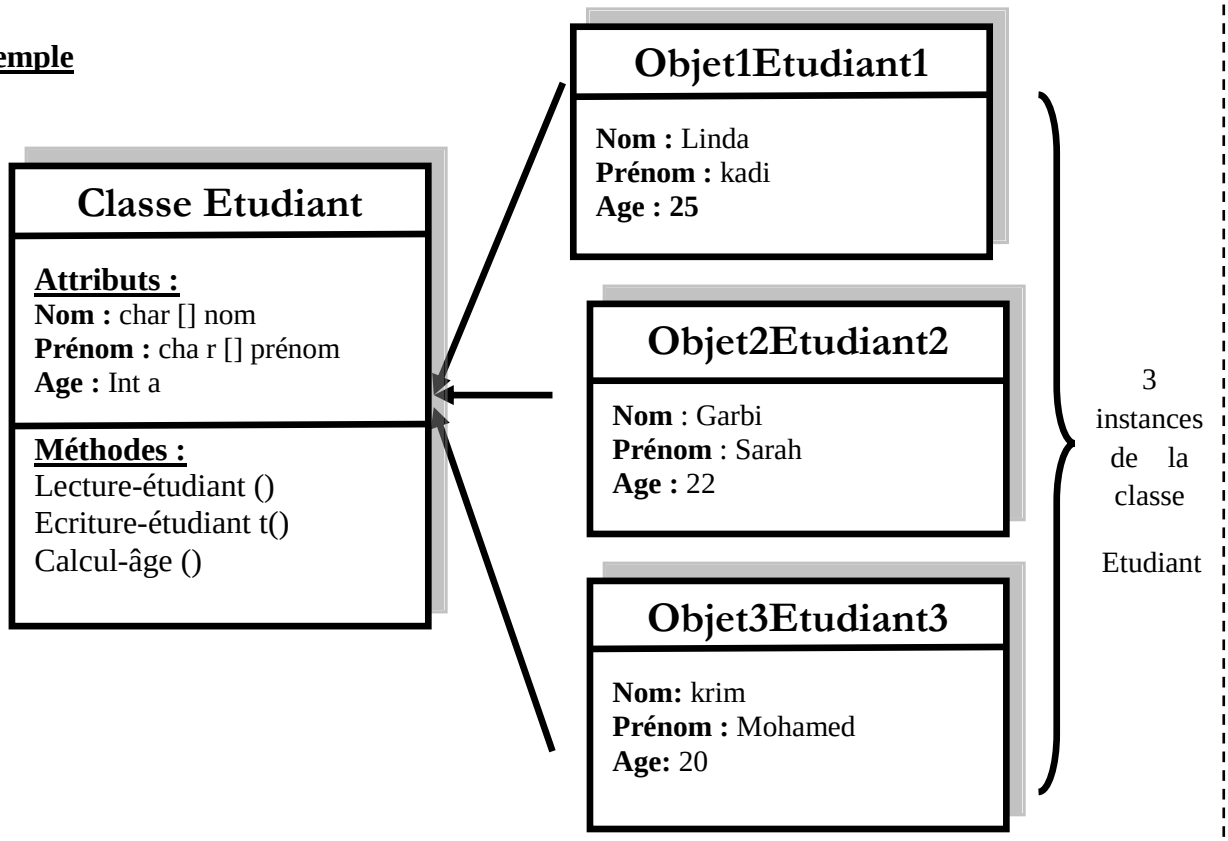


Fig.2– Instanciation d'une classe en trois objets

- **Les modificateurs de classe**

Modificateur	Rôle
Public	**La classe est accessible partout**
private	**la classe n'est accessible qu'à partir du fichier où elle est définie**
Final	**La classe ne peut pas être modifiée, sa redéfinition grâce à l'héritage est interdite. Les classes déclarées final ne peuvent donc pas avoir de classes filles**
abstract	**La classe contient une ou des méthodes abstraites, qui n'ont pas de définition explicite. Une classe déclarée abstract ne peut pas être instanciée : il faut définir une classe qui hérite de cette classe et qui implémente les méthodes nécessaires pour ne plus être abstraite.**

Remarque :

Par convention, en Java les noms des classes commencent par une lettre majuscule, si elle est composé de plusieurs mots, ces mots doivent être liés par « _ » et commencent par une lettre majuscule.

2.2) Objet

L'objet est l'unité de base de la conception orientée objet d'un système informatique. Il représente des entités réelles (personne, animal…) ou conceptuelles (cercle, point, graphique…).Les objets contiennent des attributs et des méthodes. Les attributs sont des variables ou des objets nécessaires au fonctionnement de l'objet. En java, une application est un objet. La classe est la description d'un objet. Un objet est une instance d'une seule classe. Pour chaque instance d'une classe, le code est le même, seul les données sont différentes à chaque objet. Un objet est caractérisé par :

- ✓ **Les attributs** portent des valeurs attachées à l'objet. L'ensemble de ces valeurs à un instant donné constitue l'état de l'objet à cet instant,
- ✓ **Message** est une demande d'activation d'une méthode envoyé à un objet.
- ✓ **Les méthodes** sont des fonctions ou procédures liée à un objet qui est déclenchée à la réception d'un message particulier, la méthode déclenchée correspond strictement au message reçu. La liste des méthodes définies au sein d'un objet constitue l'interface de l'objet pour l'utilisateur, ce sont les messages que l'objet peut comprendre si on les lui envoie et dont à la réception déclenche les méthodes correspondantes.
- ✓ **Une identité** qui permet de le distinguer des autres objets, de manière unique.
- ✓ **Signature** d'une méthode représente la précision de son nom, du type de ses arguments et du type retournée.

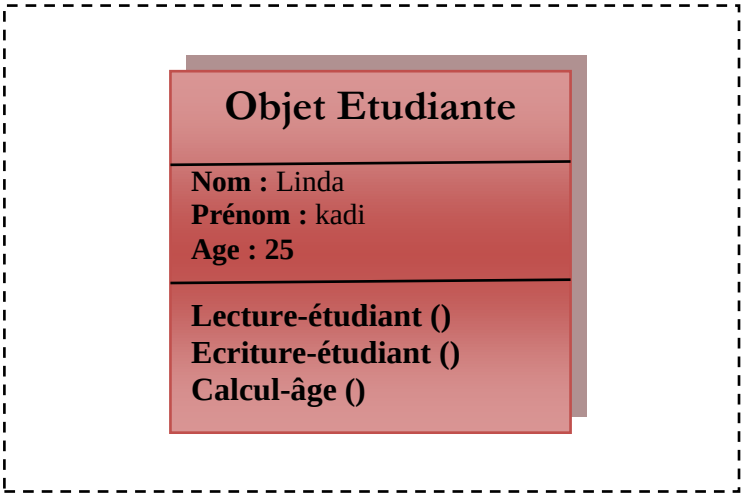


Fig3. Exemple d'objet

- **La durée de vie d'un objet**

Les objets ne sont pas des éléments statiques et leur durée de vie ne correspond pas forcément à la durée d'exécution du programme. La durée de vie d'un objet passe par trois étapes :

- ✓ la déclaration de l'objet et l'instanciation grâce à l'opérateur ***NEW*** **,** et les attributs de l'objet sont initialisés.
- ✓ l'utilisation de l'objet en appelant ces méthodes.
- ✓ la suppression de l'objet : elle est automatique en java grâce à la machine virtuelle,

  La libération de la mémoire l'objet n'est plus référencé, la place mémoire occupée est Récupérée.

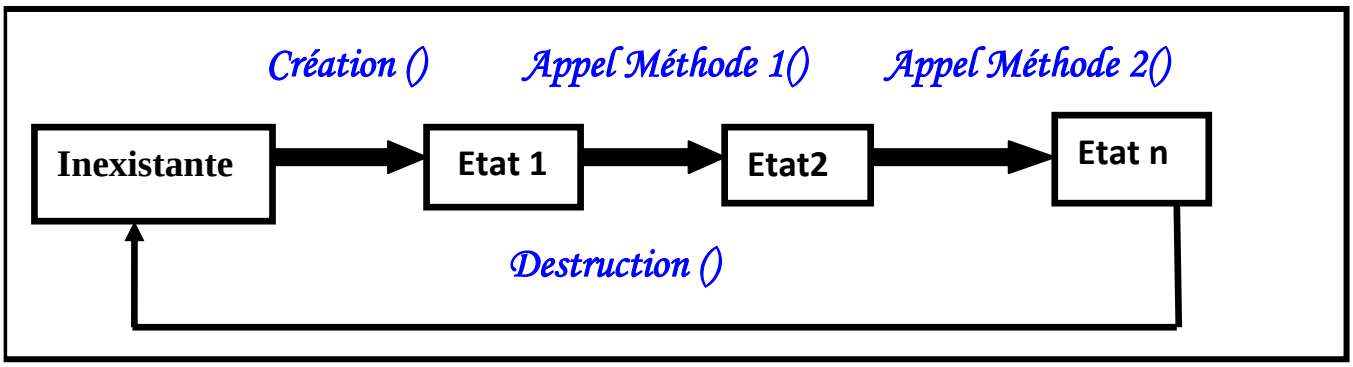


Fig. 4: cycle de vie d'un objet

2.3) L'interfaces [1] :

Une interface est un type, au même titre qu'une classe, mais abstrait et qui donc ne peut être instancié. Une interface décrit un ensemble de signatures de méthodes, sans implémentation, qui doivent être implémentées dans toutes les classes qui implémentent l'interface. L'utilité du concept d'interface réside dans le regroupement de plusieurs classes, tel que chacune implémente un ensemble commun de méthodes, sous un même type. Une interface possède les caractéristiques suivantes :

- elle contient des signatures de méthodes ;
- elle ne peut pas contenir de variables ;
- une interface peut hériter d'une autre interface (avec le mot-clé **extends**).
- une classe (abstraite ou non) peut implémenter plusieurs interfaces. La liste des interfaces implémentées doit alors figurer après le mot-clé **implements** placé dans la déclaration de classe, en séparant chaque interface par une virgule.

2.4) Encapsulation :

L'encapsulation est le faite qu'un objet renferme ses propres attributs et ses méthodes. Les détails de l'implémentation d'un objet sont masqués aux autres objets du système à objets. On dit qu'il ya encapsulation de données et du comportement des objets. On précise trois modes d'accès aux attributs d'un objet.

- **Le mode public :** avec lequel les attributs seront accessibles directement par l'objet lui-même ou par d'autres objets. Il s'agit du niveau le plus bas de protection.
- **Le mode private :** avec lequel les attributs de l'objet seront inaccessibles à partir d'autres objets : seules les méthodes de l'objet pourront y accéder. Il s'agit du niveau le plus fort de protection.
- **Le mode protected :** cette technique de protection est étroitement associée à la notion d'héritage. C'est-à-dire seules les méthodes présentes dans le même package que cette classe ou ses sous classes pourront y accéder.

2.5) L'héritage :

L'idée principale de l'héritage est d'organiser les classes de manière hiérarchique. La relation d'héritage est unidirectionnelle et, si une classe B hérite d'une classe A, on dira que B est une sous-classe de A. Cette notion de sous-classe signifie que la classe B est un cas particulier de la classe A et donc que les objets instanciant la classe B instancient également la classe A.

On parle d'héritage simple lorsqu' une classe fille ne possède qu'une classe mère. On parle d'héritage multiple lorsqu'une classe fille possède plusieurs classes filles.

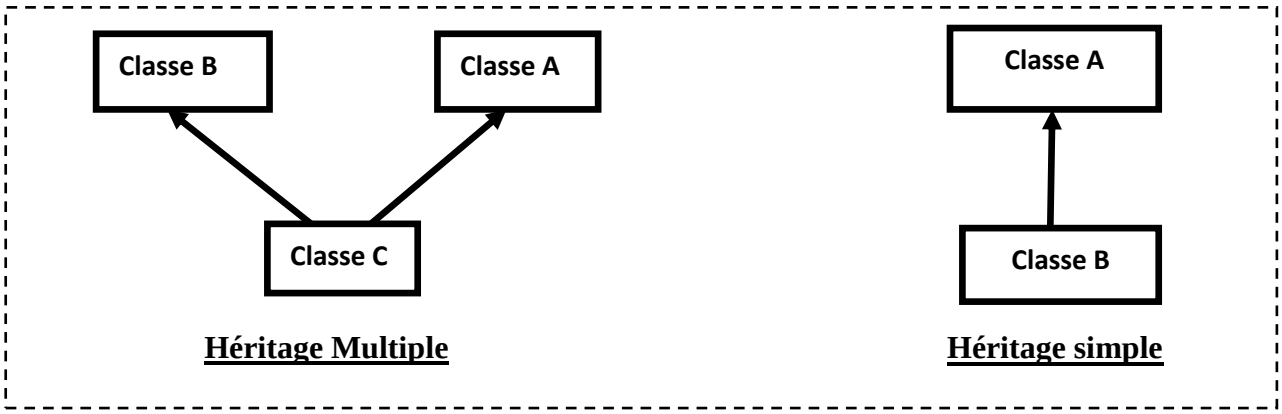


Fig.5– Exemple de relations d'héritage

- Le concept d'héritage permet à la classe dérivée de :
  1. Hériter les attributs et les méthodes de la classe de mère.
  2. Ajouter ses propres définitions des attributs et des méthodes.
  3. Redéfinir et surcharger une ou plusieurs méthodes héritées.

Remarque :

Une classe ne peut avoir qu'une seule classe mère : il n'y a pas d'héritage multiple en java.

2.6) Le polymorphisme [1] :

Le terme polymorphisme issu du ***GREC*** signifie la capacité de prendre plusieurs formes.

Le polymorphisme est un concept extrêmement puissant en POO, Comme son nom l'indique le polymorphisme permet à une méthode d'adopter plusieurs formes sur des classes différentes .en autre terme Le polymorphisme est un mécanisme qui permet à une sous classe de redéfinir une méthode dont elle a hérité tout en gardant la même signature de la méthode.

2.7) Notion de paquetage (package) [1] :

Il existe un moyen de regrouper des classe voisines ou qui couvrent un même domaine, ce sont les packages. Un package peut être considéré comme une bibliothèque des classes, il permet de regrouper un certain nombre des classes relativement liées. Les packages peuvent être organisés d'une manière hiérarchique en sous-packages et ainsi de suite.

- **Avantages**
  - ✓ Facilite le développement des bibliothèques et des modules autonomes.
  - ✓ Les classes d'un même package travaillent conjointement sur un même domaine.
  - ✓ Facilite la réutilisabilité.

3) Avantage de la programmation oriente objet [2]

L'OO élimine peu à peu le procédural dans les grands programmes car il présente d'énormes avantages:

- ✓ facilité d'organisation et la réutilisation de code, encapsulation et abstraction.
- ✓ méthode plus intuitive.
- ✓ possibilité d'héritage.
- ✓ facilité de correction.
- ✓ projets plus faciles à gérer.
- ✓ Améliorer la conception et la maintenance des grands systèmes.
- ✓ Programmation par « composants ».

L'intérêt principal de l'OO réside dans le fait que l'on ne décrit plus par le code des actions à réaliser de façon linéaire mais par des ensembles cohérents appelés objets.

- L'OO est facilement compréhensible car il décrit des entités comme il en existe dans le monde réel. "Dans un modèle à objets, toute entité du monde réel est un objet, et réciproquement, tout objet représente une entité du monde réel".
- L'OO permet en quelque sorte de factoriser le code en ensembles logiques. Du point de vue de la programmation, L'OO permet d'écrire des programmes facilement lisibles avec un minimum d'expérience, de taille minimale et à la correction aisée. Ces programmes sont, de plus, souvent très stables.
- Les informations concernant un domaine étant centralisées en objets, il est facile de sécuriser le programme en interdisant ou autorisant l'accès à ces objets aux autres parties du programme.

4) JAVA langage de programmation oriente objet

4.1) Définition [1]:

Le langage Java est un langage généraliste de programmation synthétisant les principaux langages existants lors de sa création en 1995 par Sun Microsystems. Il permet une programmation orientée-objet, et reprend une syntaxe très proche de celle du langage C.

Outre son orientation objet, le langage Java a l'avantage d'être :

- **Modulaire :** on peut écrire des portions de code génériques, c.-à-d. utilisables par plusieurs applications.
- **Rigoureux :** la plupart des erreurs se produisent à la compilation et non à l'exécution.
- **Portable :** un même programme compilé peut s'exécuter sur différents environnements).

4.2) Caractéristiques de JAVA [4]

Java possède un certain nombre de caractéristiques qui ont largement contribué à son énorme succès :

- **Java est interprétée :**

La source est compilé en pseudo code ou byte code puis exécuté par un interpréteur Java, En effet, le byte code, s'il ne contient pas de code spécifique une plate−forme particulière peut être exécuté et obtenir quasiment les même résultats sur toutes les machines disposant d'une **JVM** (Java Virtual Machine).

- **Java est portable (il est indépendant de toute plate−forme) :**

Il n'y a pas de compilation spécifique pour chaque plate forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une Java Virtual Machine. Cette indépendance est assurée au niveau du code source grâce au niveau du byte code.

- **Java est orienté objet :**

Comme la plupart des langages récents, Java est orienté objet. Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).

- **Java est simple :**

Le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs.

- **Java est fortement typée :**

Toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données. Si une telle conversion doit être réalisée, le développeur doit obligatoirement utiliser une classe ou une méthode statique fournie en standard pour la réaliser.

- **Java assure la gestion de la mémoire :**

L'allocation de la mémoire pour un objet est automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector qui restitue les zones de mémoire laissées libres suite à la destruction des objets.

- **Java est économe :**

Le pseudo code à une taille relativement petite car les bibliothèques de classes requises ne sont liées qu'à l'exécution.

4.3) La machine virtuelle Java (*JVM*)

Java est un langage ***multi-plates-formes*** qui permet, selon son concepteur ***Sun Microsystems***, d'écrire une fois pour toute des applications capables de fonctionner dans tous les environnements. Cet objectif est atteint grâce à l'utilisation d'une machine virtuelle Java (JVM) qui exécute les programmes écrits dans ce langage.

4.4) Environnement exécution JAVA [3] :

Java est un langage interprété, ce qui signifie qu'un programme compilé n'est pas directement exécutable par le système d'exploitation mais il doit être interprété par un autre programme, qu'on appelle interpréteur. La figure suivante illustre ce fonctionnement.

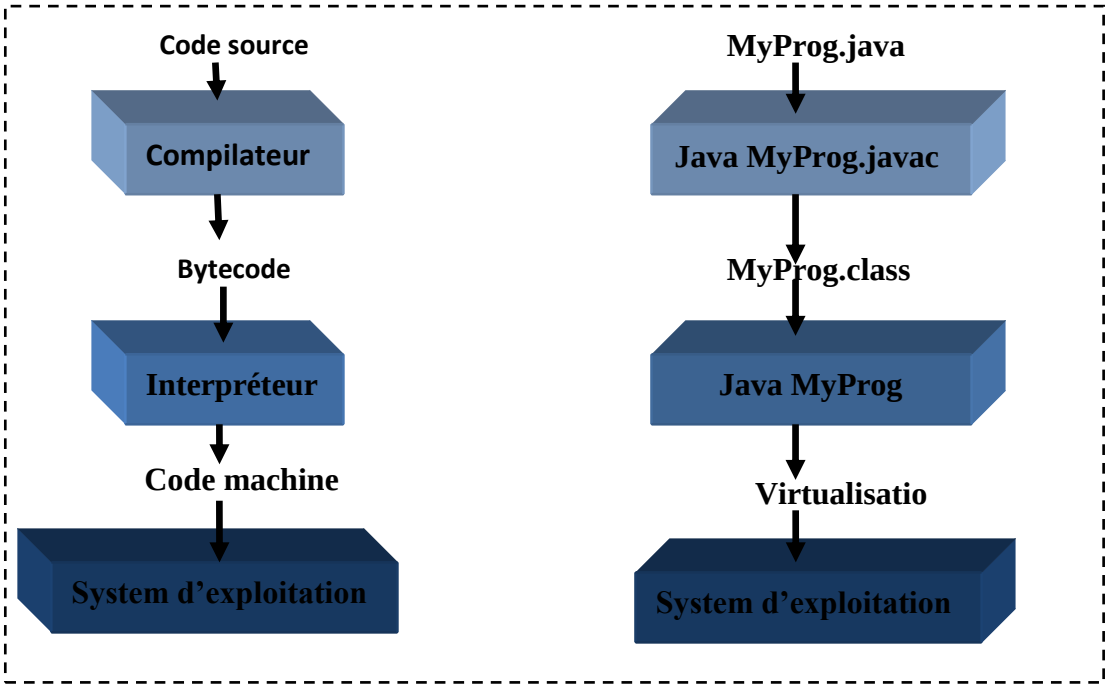


Fig.6 Interprétation du langage JAVA

Un programmeur Java écrit son code source, sous la forme de classes, dans des fichiers dont l'extension est .java. Ce code source est alors compilé par le compilateur javac en un langage appelé bytecode et enregistre le résultat dans un fichier dont l'extension est .class. Le bytecode ainsi obtenu n'est pas directement utilisable. Il doit être interprété par la machine virtuelle de Java qui transforme alors le code compilé en code machine compréhensible par le système d'exploitation. C'est la raison pour laquelle Java est un langage portable : le bytecode reste le même quelque soit l'environnement d'exécution.

4.5) Les avantages de langage JAVA [4] :

Sun définit Les objectifs de langage Java comme suit :

- Simple et dynamique.

- Orienté objet.
- Réparti.
- Interprété.
- Robuste.
- Sûr et multitâches.
- Portable.
- Performant.

5) Conclusion

Nous n'avons aborde dans ce chapitre quelques uns des concepts de la programmation orientée objet qui est en réalité beaucoup plus vaste. Les relations d'héritage, d'encapsulation et de polymorphisme sont les plus fondamentales. Citons que la plupart des langages de programmations récents se basent sur l'approche orientée objet, telle que le langage JAVA qui est simple, portable et performante.

A la suit de notre étude on à constaté que cette approche facilite l'écriture et la maintenance du code, notamment grâce à une modularité plus facile à implémenter qu'avec d'autres paradigmes de programmation. Il faut cependant remarquer que cette approche orientée objet, surtout en Java, est très servie par le monde, dans de grandes entreprises et de gros logiciels.

Introduction :

L'orienté objet a permit d'éclater les applications en composants plus simples et réutilisables. Mais cette approche se termine à un piège si le découpage s'effectue sans règles précises. Le concepteur finira par être surchargé par la complexité du codage, afin de le solutionner il a fallu exploiter un niveau élevé dans la conception objet.

L'architecte **CHRISTOPHER ALEXANDER** a constaté que la phase de conception en architecte laisse apparaître des problèmes récurrents, et pour cela il s'efforce à résoudre l'ensemble de ces problèmes et il arrive à crée un langage de 253 patterns qui protège tous les cotes de l'instruction.et grâce à cette architecte que l'idée des design patterns a apparu en 1977.

En 1995 le '**Gang of Four**' a développé un ouvrage qui porte le titre '**Design Patterns : Elements of Reusable Object-Oriented Software**'. Il propose sur 23 Design Patterns qui font aujourd'hui référence dans le monde de l'informatique. Ensuite deux livres se spécialisant à la plate-forme JAVA EE sont arrivés, « EJB Design Pattern » de Floyd Marinescu et « Core J2EE Pattern » de DeepakAlur.

Définition :

Les design patterns sont un modèle de conception décrivent des organisations pratiques de classes objets. Ces organisations proviennent souvent d'une conception habituelle. Décrivent, Déterminer des solutions testé à des problèmes spécifiques et récurrents. Un patron décrit à la fois un problème qui se produit très fréquemment dans l'environnement et l'architecture de la solution à ce problème de telle façon que l'on puisse utiliser cette solution des milliers de fois sans jamais l'adapter deux fois de la même manière

Présentation d'un Design Pattern :

Le GOF (**Gang of Four**) a choisit la description monotone et structurée. Les caractéristiques sont : Nom, Intention, Alias, Motivation, Indications d'utilisation, Structure, Participants, Collaborations, Conséquences, Implémentation, Exemple de code, Utilisations connues et Patrons apparentés.

La plus part des articles se limitaient à citer que : le nom, le problème, la solution et les conséquences d'utilisations.

Catégories de Design Patterns :

Le classement des design patterns s'est réalisé Conformément à leur rôle et leur domaine d'application. On a distingué trois catégories, cité ci-dessous :

❖ **Design patterns de Création**

Description de la manière dont un objet ou un ensemble d'objets peuvent être créés, initialisés, et configurés. Ils sont au nombre de cinq :

- **Abstarctfactory** <Fabrique abstraite>
- **Builder** <Monteur>
- **Factorymethod** <Fabrique)>
- **Prototype** <Prototype>
- **Singleton** <Instance inique>.

❖ **Design Patterns de structure :**

Abstraction de la manière dont les classes et les objets sont composés pour former des structures plus importantes. Ils sont au nombre de sept :

- **Adapter** <Adaptateur>.
- **Bridge** <Pont>.
- **Composite** <Objet composite>.
- **Decorator** <Décorateur>.
- **Façade** <Façade>.
- **Proxy** <Proxy>.
- **Flyweight** <Poids-plume>.

❖ **Design Patterns de comportement :**

Description de structures d'objets ou de classes avec leurs interactions

- **Chain of responsibility** <Chaîne de responsabilité>
- **Command** <Commande>
- **Interpreter** <Interpréteur>
- **Iterator** <Littérateur>
- **Mediator** <Médiateur>
- **Memento** <Mémento>

4) Design patterns en J2EE :

Il y avait initialement, quinze (15) patterns, identifies par Sun Java Centre, utilisés au niveau des trois couches (3tiers) .Et de nouveaux besoins laissent apparaitre de nouveaux patterns.

Ci-dessous les cinq patterns les plus utilisés :

- **Session façade :**

Les appels distants sont coûteux, pour les limiter il faut trouver un moyen. La session façade est une session bean qui a accès aux interfaces locales d'autres beans. Un appel à une méthode d'une session façade entraine généralement plusieurs appels vers un ou plusieurs autres beans.

Exemple : (chercher un espace, copier, coller, supprimer l'originale), il permet de faire qu'un pour arriver au même résultat **(déplacer).**

- **Value Object :**

Comment allons-nous lire le contenu d'un objet ?puisque nous n'avons pas de référence directe à un objet persistent, une solution très simple qui ajoute aussi de nombreuse possibilités d'amélioration, ce pattern est objet qui offre une vue figée de notre entité a un moment donné.

- **Service Locator :**

Les clients ont besoin d'un moyen de localiser les objets distants .Et dans le but de rendre notre système flexible et réutilisable, nous définissons un service Locator pour servir aux clients les références des objets distants.

- **Data Access Object :**

Afin d'exploiter un système persistant de données, il peut y avoir beaucoup de mécanismes différents et donc beaucoup de différance des API utilisées pour accéder aux données.

Data Access Object propose d'uniformiser l'accès aux données sur différents systèmes de stockage (BDD relationnelle, un annuaire LDAP, des documents XML, etc.) En n'exposant que les interfaces permettant de manipuler les données.

- **Business Delegate :**

Business Delegate est une adaptation de pattern Adapter, appliquée au modèle J2EE. Il a des buts :

-Se débarrasser de la dépendance du client pour J2EE. J2EE offre les bases nécessaires pour séparer le client du serveur.

- Fournir des méthodes particulières aux clients.
- Exécuter des sections de code nécessaires à chaque appel distant.
- Transformer les Exceptions génériques en Exceptions propres à votre application.

❖ **Annexe design patterns J2EE :**

Deux pattern œuvrent côte requêtes :

- Intercepting Filter (Filtre d'interception).
- Front Controller (Front controleur).

Ci-dessus le reste des patterns :

- **Service to Worker** (service des travailleurs).
- **View Helper** (Aide des vue).
- **Dispatcher View** (Réparation).
- **Composite View** (Vue composite).
- **Service Activator** (Actionneur).
- **Composite Entity** (Entité composite).
- **Value Liste Handler** (Manutention liste valeur).
- **Transfer Object Assembler** (Assembleur d'objet de transfert).

5) Le Model-View-Controller:

Le Model-View-Controller est un pattern crée par **XEROX PARC** pour Smalltalk-80 ce pattern répandu et fort utile (recommandé en J2EE).Ce dernier a séparé les données, les traitements et la présentation. Or l'application se retrouve fractionnée en trois éléments essentiels : le modèle, la vue et le contrôleur, comme le montre la figure suivante :

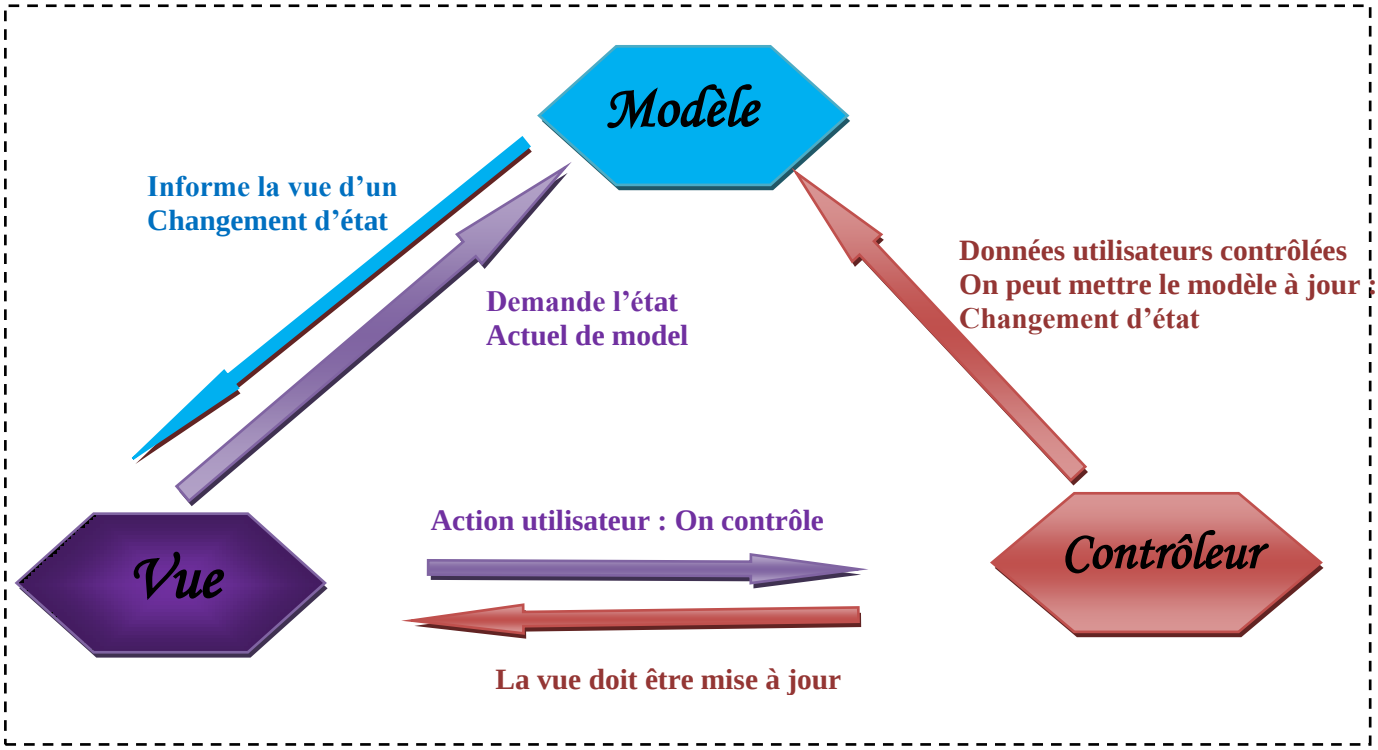


Fig1-fonctionnement de Model-View-Controller

5.1 Le modèle :

Décrit les données et les règles métiers. Les traitements liés au cœur de métier se réalisent au niveau de ce composant. Il est important de marquer que les données sont indépendantes de la présentation et ces données peuvent être liées à une base de données, des EJBs, des web service. Ces données pourront être affichées par plusieurs vue.

5.2 La vue :

Correspond à l'IHM, elle présente les donnes et interagit avec l'utilisateur. Dans les applications web, il s'agit d'une interface HTML, mais n'importe quel composant graphique peut jouer ce rôle.

5.3 Le contrôleur :

Le rôle de ce composant et d'intercepter les requêtes de l'utilisateur, d'appeler le modèle puis rediriger vers la vue conforme .Il ne fait que de l'interception et de la redirection.

6) Les patterns et la reconstruction logicielle :

La question suivante : on a mis au point (construit) un produit (logiciel), comment le reproduire efficacement et d'une manière plus rapide ? Et une discipline de génie logiciel **(la gestion de configuration de logiciel « SMC »)** ayant un objet de répondre a cette question.

On cintrera quelques un de ces patterns :

- **Bill of Moterials**
- **Self-idetifying Configuration**
- **Shared version Cache**
- **Shared Source Escalation**
- **Version-Controlled Environment**
- **Archived Environment**
- **Daily Build and Smoke Test**
- **Reproducible Build**

Remarque :

Les patterns utilisés dans le cadre de la reconstruction logiciel sont des intermédiaires entre le nouveau produit (celui qu'on veut reconstruire) et les versions précédentes de ce produit.

Conclusion :

Le tour qu'on a fait dans cette section sur les design patterns nous conduit à dire que c'est une Véritable solution afin de réduire la complexité, à promouvoir la réutilisation et à fournir un vocabulaire commun aux concepteurs. L'apparition des design patterns a rendu le développement des applications de plus en plus performant. Et même en génie logiciel occupent une place très importante.

Bibliographie

Bibliographie

Les Ouvrages:

- Marine Soft, Documentation interne, rapports d'ingénieurs.
- JEAN-PIERRE WOJTYNA, L'approche processus, mode d'emploi, Edition 2009.
- Marine Soft Direction Technique, Département Recherche & Développement Rapport sur les progiciels, Septembre 2011.
- PATRICK ITEY, Architecture J2EE, Juin 2011.
- Marine Soft Direction Technique, Département Recherche & Développement Rapport sur les outils de travail, Décembre 2011.
- ***RICHARD HODGSON***, Conception de bases de données avec UML.
- Benoit CHARROUX, Aomar OSMANI, Yann THIERRY-MIEG, UML2, pratique de la modélisation 2ème édition, 20011.
- Olivier GERBE ,The unified software développement process ,2010
- Campus Press, Programmeur Java SE 6, publié en Juillet 2010.
- Développement Web : « Zone grand débutants», Publié en 2010, dernière mise à jour le 16 janvier 2011.
- « **ATOL Conseils et Développements** 3 bd Eiffel 21600 LONGVIC » Introduction à JSF « Java Server Faces », Publié par la boite en 2009.
- Martin HELLER, RicheFaces Guide du développeur, publié le 05/05/2010.
- Loïc FRERING et Baptiste MEURANT Cours JPQL, publié le 16/12/2008.
- l'ensemble des membres du forum Business Intelligence: Bérénice MAUREL - Stefan CARACAS - Julien SAUVEBOIS- GATTINO – Charly CLAIRMONT - Erwan BODERE, FAQ BIRT, Publié le : 11/03/2009.
- Renaud DUBOURGUAIS, JBoss Application Server : exploitation et sécurisation, le 17/12/2006.
- Pierre-Arnaud MARCELOT 2.0 Laurent, AUDIBERT, Cour Plugin Eclipse, février 2007.
- Pascal ROQUES, UML par la pratique, Etudes des cas et exercices corrigés, Avril 2001.
- Erich GAMMA, Richard HELM, Ralph JOHNSON, John VLISSIDES, Addison

WESLEY, Design Pattern Elements of Reusable Object-Oriented Software, 1995.

- Jean-Michel Ormes, Comprendre les différents design patterns de construction, 2011.
- Mines Saint-Etienne, O. BOISSIER, G. PICARD, SMA/G2I/ENS, Design Patterns, Septembre 2009.
- University of Paderborn, Software Engineering Group, Design Patterns, 2009.
- Alexandre Brillant, Introduction aux Design Patterns en Java, 2011.
- Yannick PRIE, SIMA Réutilisation dans les SI : patrons de conception, 2008.
- université de Montréal, M. HERMAK, L. EL BADAOUI, Détection automatique des patrons de conception, 2008.
- Jean - Philippe RETAILLE, Refactoring des applications Java/J2EE, 2011.
- Christophe LUDET, Patterns J2EE / EJB, Mars 2005.
- University of Paderborn, Dr. Giese HOLGER, Software Engineering Group Design Pattern and Software Architecture, 2011.
- Université du Québec à Montréal, G. EL BOUSAIDI H. MILLI, LTCE Les patrons de conception: Représentation et mise en œuvre en 2008.
- Wikipédia, Patron de conception, Février 2012.
- Les designs patterns ou Les motifs de conception ou Les modèles de conception, JEE Pour Tous, Mai 2011.
- Alexis HASLER, Mise en œuvre du design pattern "Business Delegate , 2009
- MARINE SOFT Le rapport JEE FINAL en 10/09/2009.
- Jérôme LAFOSSE Java EE Guide de développement d'applications Web en Java, 2007
- istia.univ-angers.fr, Mr Serge TAHE Introduction à Java EE 5 avec Netbeans 6.8 et le serveur d'applications Glassfish V3, à juin 2010.
- ludovic.maitre@free.fr,L'architecture J2EE , 2012
- Alexandre PAUCHET, Informatique Répartie Introduction aux EJB, 2011
- Elysée FONTEP, Phuong-Anh HOANG, LES SERVEURS D'APPLICATION, 2011

Les sites web

http://laurent-piechocki.developpez.com/uml/tutoriel/lp/cours/

http://www.marinesoft.dz/

http://vogella.developpez.com/tutoriels/eclipse/prise-en-main-eclipse-ganymede/

http://www.itechno.com/fr/index.php

http://www.betecommechou.com/2009/02/j2ee-1-4-composants-applicatifs-et-conteneurs

http://jmdoudoux.fr/java/dej/chap-j2ee-javaee.htm