**R**épublique **A**lgérienne **D**émocratique et **P**opulaire
**M**inistère de l'**E**nseignement **S**upérieur et de la **R**echerche **S**cientifique

**U**NIVERSITE **M**OULOUD **M**AMMERI DE **T**IZI-**O**UZOU

**F**ACULTE DU **G**ENIE **E**LECTRIQUE ET D'INFORMATIQUE
**DEPARTEMENT D'INFORMATIQUE**

# Mémoire de Fin d'Etudes de MASTER ACADEMIQUE

Domaine : **Mathématiques et Informatique**
Filière : **Informatique**
Spécialité : **Système Informatique**

*Présenté par*
**nora ZEMOUCHE**
**nawel HAMMADI**

**Thème**

# Analyse conception et réalisation d'une application Java EE
# Cas : Gestion de payement fournisseur et recouvrement de l'ENIEM

*Mémoire soutenu publiquement le ……/……/ 20….. devant le jury composé de :*

**Président : M^R A DIB**

**Encadreur : M^R KERBICHE**

**Co-Encadreur : M /**

**Examinateur : M^R CHEBOUBA**

**Examinateur : M^ELLE YESLI**

# *Remerciement*

***Nous tenons a remercie d'abord le bon DIEU qui nous a aidé et qui nous a donné le courage et la volonté pour réaliser ce modeste travail.***

***Notre profonde gratitude et sincère remerciements vont à notre promoteur Mr Kerbiche .M pour nous avoir encadrés durant cette année, ainsi que pour ses conseils judicieux.***

***Nos plus vifs remerciements vont à notre co-prompteur à l'Eniem Mr Kabi et Mr Medani qui nous ont généreusement aidées durant notre stage.***

***Nos remerciements vont aussi aux membres de jury pour avoir accepté de juger ce modeste travail.***

***Nos remerciements les plus chaleureux s'adressent à nos familles et surtout nos parents qui sont la source de cette réussite et qui nous ont soutenu et encouragé pour aller au bout de ce travail.***

# *Dédicaces*

*Je dédie ce modeste travail :*

- *A mes chers parents qui m'ont beaucoup aidé et soutenu durant ma vie et surtout dans mes études et leur compréhension et leur soutien .Je pris dieu le tout puissant de les protéger du mal et les récompenser de toutes les peines et sacrifices données aux quels je ne rendrai jamais assez.*

- *A mes frères « Nadir,Yacinev».*
- *A mes sœurs.*
- *A mon cher mari « Mohamed ».*
- *A ma binôme « Nora »,*
- *A tous mes amis (es),*
- *A toute personne qui me connaît,*

***Nawel.***

# *Dédicaces*

*Je dédie ce modeste travail :*

- *A mes chers parents qui m'ont beaucoup aidé et soutenu durant ma vie et surtout dans mes études et leur compréhension et leur soutien .Je pris dieu le tout puissant de les protéger du mal et les récompenser de toutes les peines et sacrifices données aux quels je ne rendrai jamais assez.*

- *A mes frères.*
- *A mes sœurs.*
- *A ma binôme « Nawel »,*
- *A tous mes amis (es),*
- *A toute personne qui me connaît,*

***Nora.***

# Tableau de figures


| | Numéro de figure | Nom de la figure | La page |
|---|---|---|---|
| Chapitre I | Figure I.1 | Le dialogue client/serveur | P 3 |
| | Figure I.2 | Architecture à 2 niveaux | P 4 |
| | Figure I.3 | Architecture à 3 niveaux | P 4 |
| | Figure I.4 | Architecture à multi niveaux | P 5 |
| | Figure I.5 | Architecture et les composants de j2EE | P 8 |
| | Figure I.6 | L'architecture des conteneurs j2EE | P 9 |
| | Figure I.7 | Les composants WEB | P 11 |
| | Figure I.8 | Les communications d'un serveur J2EE | P 12 |
| Chapitre: II | Figure II .1 | Présentation du l'organigramme générale de ENIEM | P 17 |
| | Figure II.2 | Organigramme général de l'unité prestations techniques | P 21 |
| | Figure II.3 | Organigramme du champ d'étude | P 22 |
| | Figure II.4 | Diagramme des flux | P 24 |
| Chapitre III | Figure III.1 | Diagramme de contexte relatif a l'application de gestion de payement fournisseur et recouvrement | P 30 |
| | Figure III.2 | Le diagramme de cas d'utilisation global | P 38 |
| | Figure III.3 | Diagramme de séquence associé au scénario «Modifier mot de passe » | P 40 |
| | Figure III.4 | Diagramme de séquence associé au scénario « Ajouter compte utilisateur » | P 41 |
| | Figure III.5 | Diagramme de classe pour l'itération 1 | P 42 |
| | Figure III.6 | Diagramme de séquence associé au scénario« Ajouter facture fournisseur» | P 42 |

| | | | |
|---|---|---|---|
| | Figure III.7 | Diagramme de séquence associé au scénario « Modifier facture fournisseur » | P 43 |
| | Figure III.8 | Le diagramme de classe pour l'itération2 | P 44 |
| | Figure III.9 | Diagramme de séquence associé au scénario « Consulter recouvrement » | P 45 |
| | Figure III.10 | Diagramme de séquence associé au scénario «Modifier facture client» | P 46 |
| | Figure III.11 | Le diagramme de classe pour l'itération 3 | P 47 |
| Chapitre IV | Figure IV.1 | L'architecture Hibernate | P 50 |
| | Figure IV.2 | Le fichier hibernate.cfg.xml | P 51 |
| | Figure IV.3 | Les fichiers xml de mapping | P 52 |
| | Figure IV.4 | JavaBean | P 53 |
| | Figure IV.5 | La classe HibernateUtil | P 54 |
| | Figure IV.6 | L'interface du fichier de configuration «faces-config.xml » | P 55 |
| | Figure IV.7 | Le modèle MVC type 1 | P 56 |
| | Figure IV.8 | Le modèle MVC type 2 | P 57 |
| | Figure IV.9 | L'architecture de notre application suivant le modèle MVC | P 58 |
| | Figure IV.10 | Interface de l'IDE NetBeans | P 60 |
| | Figure IV.11 | L'interface Oracle Express Edition | P 61 |
| | Figure IV.12 | L'interface d'authentification | P 67 |
| | Figure IV.13 | L'interface d'accueil pour l'administrateur | P 67 |
| | Figure IV.14 | L'interface d'accueil pour le directeur | P 68 |
| | Figure IV.15 | L'interface d'accueil pour le financier | P 68 |
| | Figure IV.16 | L'interface de création d'un nouvel utilisateur | P 69 |
| | Figure IV.17 | L'interface de gestion des factures fournisseur | P 69 |

| | | | |
|---|---|---|---|
| | Figure IV.18 | L’interface de modification de la facture fournisseur | P 70 |
| | Figure IV.19 | L’interface de règlement de la facture client | P 70 |
| | Figure IV.20 | L’interface de consultation de la dette | P 71 |

# *Sommaire*

# Sommaire :

Introduction générale ..... 1

**Chapitre I : Technologie Java Entreprise Edition**

I.1 Introduction ..... 2

I.2 Architecture Client / Serveur ..... 2

I.2.1 Définition ..... 2

I.2.2 Notion de base ..... 2

I.2.3 Classification des architectures client/serveur ..... 4

I.2.3.1 Architecture à 2 niveaux ..... 4

I.2.3.2 Architecture à 3 niveaux ..... 4

I.2.3.3 Comparaison des deux types d'architecture ..... 5

I.2.3.4 Architecture multi niveaux ..... 5

I.2.4 Avantages et inconvénients de l'architecture client/serveur ..... 6

I.2.4.1 Avantages de l'architecture client/serveur ..... 6

I.2.4.1 Inconvénients de l'architecture client/serveur ..... 6

I.3 Java Entreprise Edition ..... 6

I.3.1 Présentation de Java Entreprise Edition ..... 6

I.3.2 Fonctionnement interne ..... 7

I.3.3 Topologie d'une application Java Entreprise Edition ..... 7

I.3.3.1 Application multi tiers ..... 7

I.3.3.2 Application web ..... 7

I.3.4 architecture Java Entreprise Edition ..... 8

I.3.5 Composantes de l'architecture Java Entreprise Edition ..... 9

I.3.5.1 Conteneur de composants Java Entreprise Edition ..... 9

I.3.5.2 les Clients Java Entreprise Edition ..... 10

I.3.5.3 Composants web ..... 10

I.3.5.4 Composants Entreprise JavaBeans [EJB] ..... 11

I.3.5.5 Serveur EJB ..... 12

I.3.6 Les services de Java Entreprise Edition ..... 12

I.3.6.1 Les services d'infrastructures ..... 12

I.3.6.2 Les services de communication ..... 13

I.3.7 Technologies Java Entreprise Edition ..... 13

I.3.8 L'environnement d'exécution des applications Java Entreprise Edition ..... 14

I.3.9 L'assemblage et le déploiement d'application Java EE ..... 14

I.3.10 Les acteurs d'une application Java EE ..... 14

I.4 Conclusion ..... 15

## Chapitre II : Organisme d'accueil

II.1 Introduction ..... 16

II.2 Historique ..... 16

II.3 Présentation de l'organigramme général de l'ENIEM ..... 17

II.4 Missions et objectifs ..... 17

II.5 Mode d'organisation ..... 18

II.5.1 Les directions ..... 18

II.5.2 Les unités ..... 19

II.6 Présentation de champ d'étude ..... 20

II.6.1 Description de l'organigramme ..... 22

II.6.2 Organigramme du champ d'étude ..... 22

II.7 la situation informatique du champ d'étude ..... 23

II.8 Etude de l'existant ..... 23

II.9 Diagramme de flux ..... 23

II.9.1 Concepts utilisés ..... 23

II.9.2 Diagramme des flux d'information ..... 24

II.9.3 Liste des flux d'information ..... 25

II.10 Critiques de l'existant et suggestions ..... 25
II.10.1 Niveau organisationnel ..... 25
II.10.2 Niveau informationnel ..... 26
II.10.3 Niveau technique ..... 26
II.11 Conclusion ..... 27

## Chapitre III : Analyse et conception

III.1 Introduction (RUP) ..... 28
III.2 Phase d'initialisation ..... 28
III.3 Phase d'élaboration ..... 29
III.3.1 Identification des acteurs ..... 29
III.3.2 Spécification des tâches ..... 29
III.3.3 Diagramme de contexte ..... 30
III.3.4 Spécification de quelque cas d'utilisation avec scénarios ..... 30
III.3.5 Représentation des diagrammes de cas d'utilisation ..... 37
III.4 Construction ..... 39
III.4.1 Réalisation de l'itération 1 ..... 39
III.4.1.1 Diagramme de séquence pour itération 1 ..... 39
III.4.1.2 Diagrammes de classe pour itération 1 ..... 42
III.4.2 Réalisation de l'itération2 ..... 42
III.4.2.1 Diagramme de séquence pour itération 2 ..... 42
III.4.2.2 Diagramme de classe pour l'itération 2 ..... 44
III.4.3 Réalisation de l'itération 3 ..... 45
III.4.3.1 Diagramme de séquence pour itération 3 ..... 45
III.4.3.2 Diagramme de classe pour l'itération 3 ..... 47
III.5 Conclusion ..... 47

## Chapitre III : Réalisation et mise en œuvre

IV.1 Introduction ..... 48

IV.2 Environnement de développement ..... 48
IV.2.1 La plate forme Java Entreprise Edition ..... 48
IV.2.2 Qu'est-ce que le Framework Hibernate ..... 49
IV.2.2.1 Mise en œuvre du Framework Hibernate ..... 49
IV.2.3 Le Framework JSF/prime faces ..... 54
IV.2.4 Architecture de notre application selon le modèle MVC ..... 56
IV.2.4.1 Le modèle MVC type1 ..... 56
IV.2.4.2 Le modèle MVC type2 ..... 57
IV.3 Outils de développement ..... 59
IV.3.1 Logiciels ..... 59
IV.3.1.1 Environnement de développement intégré « NetBeans » ..... 59
IV.3.1.2 Le serveur d'application ..... 60
IV.4 Langage de programmation ..... 66
IV.4.1 Le langage coté serveur ..... 66
IV.4.2 Le langage coté client ..... 66
IV.5 Présentation des graphiques de l'application ..... 66
IV.6 Conclusion ..... 71
Conclusion générale ..... 72

# *Introduction générale*

# Introduction générale

A l'heure où l'information n'est plus seulement considérée comme une ressource opérationnelle mais aussi comme une ressource stratégique pour l'entreprise, son système d'information devient un facteur de différenciation par rapport à ses concurrents. C'est par sa culture et son système d'information performant que l'entreprise pourra s'adapter à son environnement concurrentiel.

Nous avons constaté, en effet, pendant nos recherches que l'entreprise possède un grand nombre de bien, il devient délicat de suivre toutes les opérations faites sur ces biens. Donc la meilleure solution pour pallier aux problèmes de gestion de ces opérations est l'informatisation afin d'assurer l'accès instantané aux données et sécuriser ces dernières, ce qui simplifie le travail de l'entreprise.

Et Comme toute organisation, l'Entreprise National des Industries de l'Electroménager (ENIEM) travaille sans relâche pour mettre en œuvre cette nouvelle technologie de traitement, pour assurer une gestion plus fiable, plus rigoureuse, moins fastidieuse et pour minimiser le risque d'erreurs et assurer la disponibilité de l'information à toute éventuelle demande.

Notre travail consiste à concrétiser les avantages que les nouvelles technologies offres en réalisant une application avec Java Entreprise Edition pour la gestion de payement fournisseur et recouvrement à ENIEM de Tizi Ouzou.

Pour mener à bien notre travail, nous avons partagé le travail comme suit :

- ✓ Chapitre I, « **Présentation de Java Entreprise Edition** » consiste à présenter les généralités de Java 2 Entreprise Edition.
- ✓ Chapitre II, s'intitule «**Présentation de l'organisme d'accueil** » consacré a l'organisme d'accueil et a la présentation du champs d'étude.
- ✓ Chapitre III, nous avons présenté « **Analyse et conception** » en suivant les étapes de la démarche RUP en utilisant les diagrammes d'UML.
- ✓ Chapitre IV, « **Réalisation et mise en œuvre** » est consacré a l'analyse et a la présentation de l'environnement dans lequel notre application a été réalisée, les outils utilisés et quelque interfaces de notre application.

L'objectif de notre travail est de permettre au cadre comptable qui est le financier une meilleure gestion de différentes taches et difficultés auquel il affrontés quotidiennement.

# *Chapitre I :*
# *Présentation de Java Entreprise Edition*

## I.1 Introduction

Les années 90 ont été marquées par l'arrivée de l'informatique Client/Serveur. Cette dernière est arrivée essentiellement pour combler les lacunes des systèmes centraux, en proposant un environnement ouvert, souple. Il permet d'améliorer la flexibilité des systèmes, et procurer certains avantages à l'entreprise telle que la souplesse et la capacité d'adaptation face au changement.

Fin des années 90, début 2000 arrive enfin la plateforme J2EE, elle occupe aujourd'hui une place prépondérante dans l'évolution des systèmes informatiques. Elle permet de prendre en charge et d'améliorer les systèmes complexes existant ainsi que de simplifier leur exploitation et leur maintenance.

## I.2 Architecture Client / Serveur:

### I.2.1 Définition : [1]

L'architecture client/serveur est un modèle de fonctionnement logiciel qui peut réaliser tout type d'architecture matérielle (petites ou grosses machines) à partir du moment ou ces architectures peuvent être interconnectées.

On parle de fonctionnement logiciel dans la mesure où cette architecture est basée sur l'utilisation de deux types de logiciels : à savoir un logiciel serveur et un logiciel client s'exécutent normalement sur deux machines différentes. L'élément important dans cette architecture est l'utilisation de mécanismes de communication entre les deux applications.

Le dialogue entre les applications peut se résumer par :

- ✓ Client demande un service au serveur.
- ✓ Le serveur réalise ce service et renvoie le résultat au client.

L'un des principes fondamental est que le serveur réalise un traitement pour le client.

## I.2.2 Notion de base :

- ✓ **Le client :** processus demandant l'exécution d'une opération à un autre processus Serveur par l'envoi d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un message en retour.
- ✓ **Le serveur :** processus accomplissant une opération sur demande d'un client et transmettant la réponse à ce client.
    - **Serveur web :**

**Serveur Apache :** le serveur Apache est un serveur web qui s'occupe du traitement des requêtes des clients, le transfert des pages HTML au browser et l'exécution des programmes sur la machine serveur. C'est le serveur le plus utilisé, présentant un niveau de performances élevé pour des exigences matérielles modestes.

- **Serveur de base de données :**

**Serveur MySQL :** MySQL est un serveur de base de données relationnelle, basé sur le langage de requête SQL, il est implémenté sur un mode client/serveur avec le coté serveur : le serveur MySQL, et du coté client : les différents programmes et librairies. MySQL se caractérise par sa rapidité et sa facilité d'utilisation.

- ✓ **Requête :** message transmis par un client a un serveur décrivant l'opération à Exécuter pour le compte du client.

- ✓ **La réponse :** message transmis par un serveur a un client suite a l'exécution d'une opération contenant le paramètre de retour de l'opération.

- ✓ **Middleware :** c'est le logiciel qui situe au milieu, assure les dialogues entre les clients et les serveurs, il est souvent hétérogène ; en d'autres il constitue l'ensemble des services logiciels construits au dessus de protocole de transport afin de permettre l'échange Requêtes Réponse de manière transparente en cachant l'hétérogène des composants mis en jeu (SGBD, réseau,..).

Notons au passage que le client est toujours le premier à engager la conversation en sa qualité de demandée et fournit la réponse au client.

De plus, la plupart du temps, un client ne dialogue qu'avec un serveur à la fois alors qu'un serveur est appeler gérer plusieurs connexions (clients) simultanément.

En résumer, la figure suivante illustre ces notions :

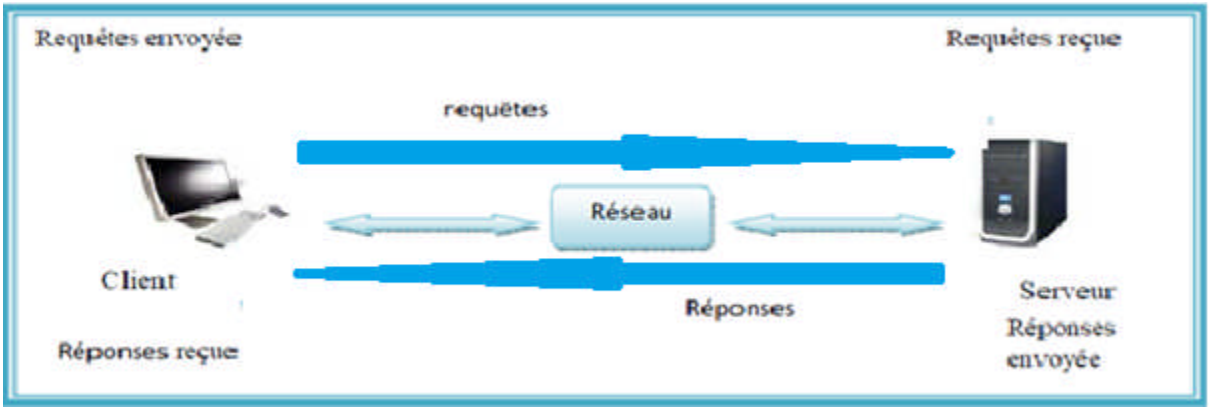



**Figure I.1 : le dialogue client/serveur.**

## I.2.3 Classification des architectures client/serveur :

### I.2.3.1 Architecture à 2 niveaux :

L'architecture à deux niveaux (aussi appelée *architecture 2-tier*,( *tier* signifiant *rangée* en anglais) caractérise les systèmes clients/serveurs pour lesquels le client demande une ressource et le serveur la lui fournit directement, en utilisant ses propres ressources. Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir une partie du service.

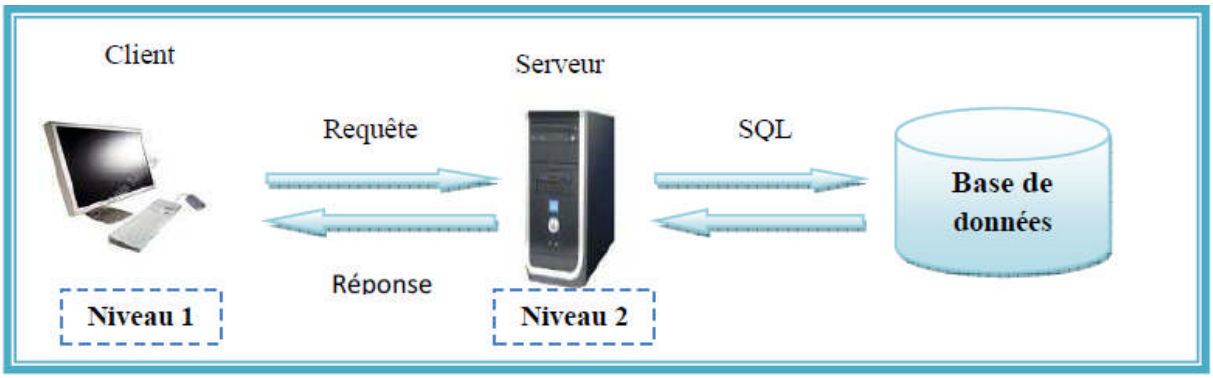



**Figure I.2 : Architecture à 2 niveaux**

### I.2.3.2 Architecture à 3 niveaux :

Dans l'architecture à 3 niveaux (appelée *architecture 3-tier*), il existe un niveau intermédiaire, c'est-à-dire que l'on appel généralement une architecture partagée entre :

1. Un client, c'est-à-dire l'ordinateur demandeur de ressources, équipée d'une interface utilisateur (généralement un navigateur web) chargée de la présentation ;
2. Le serveur d'application (appelé également **middleware**), chargé de fournir la ressource mais faisant appel à un autre serveur
3. Le serveur de données, fournissant au serveur d'application les données dont il a besoin.

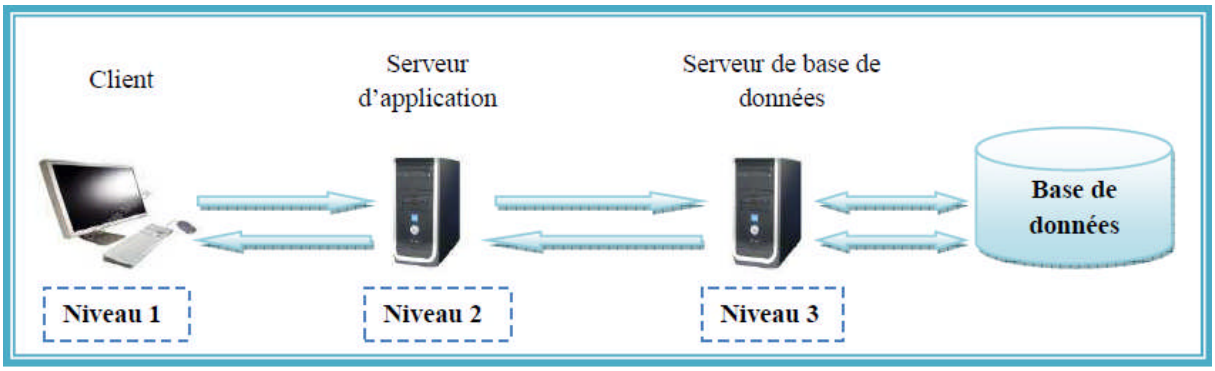



**Figure I.3 : Architecture à 3 niveaux**

Etant donné l'emploi massif du terme d'architecture à 3 niveaux, celui-ci peut parfois désigner aussi les architectures suivantes :

- ✓ Partage d'application entre client, serveur intermédiaire, et serveur d'entreprise ;
- ✓ Partage d'application entre client, serveur d'application, et serveur de base de données d'entreprise.

### I.2.3.3 Comparaison des deux types d'architecture :

L'architecture à deux niveaux est donc une architecture client/serveur dans laquelle le serveur est polyvalent, c'est-à-dire qu'il est capable de fournir directement l'ensemble des ressources demandées par le client.

Dans l'architecture à trois niveaux par contre, les applications au niveau serveur sont délocalisées, c'est-à-dire que chaque serveur est spécialisé dans une tâche (serveur web/serveur de base de données par exemple). L'architecture à trois niveaux permet :

- ✓ Une plus grande flexibilité/souplesse ;
- ✓ Une sécurité accrue car la sécurité peut être définie indépendamment pour chaque service, et à chaque niveau ;
- ✓ De meilleures performances, étant donné le partage des tâches entre les différents serveurs.

### I.2.3.4 Architecture multi niveaux :

Dans l'architecture à 3 niveaux, chaque serveur (niveaux 2 et 3) effectue une tâche (un service) spécialisée. Un serveur peut donc utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service. Par conséquent, l'architecture à trois niveaux est potentiellement une architecture à N niveaux...

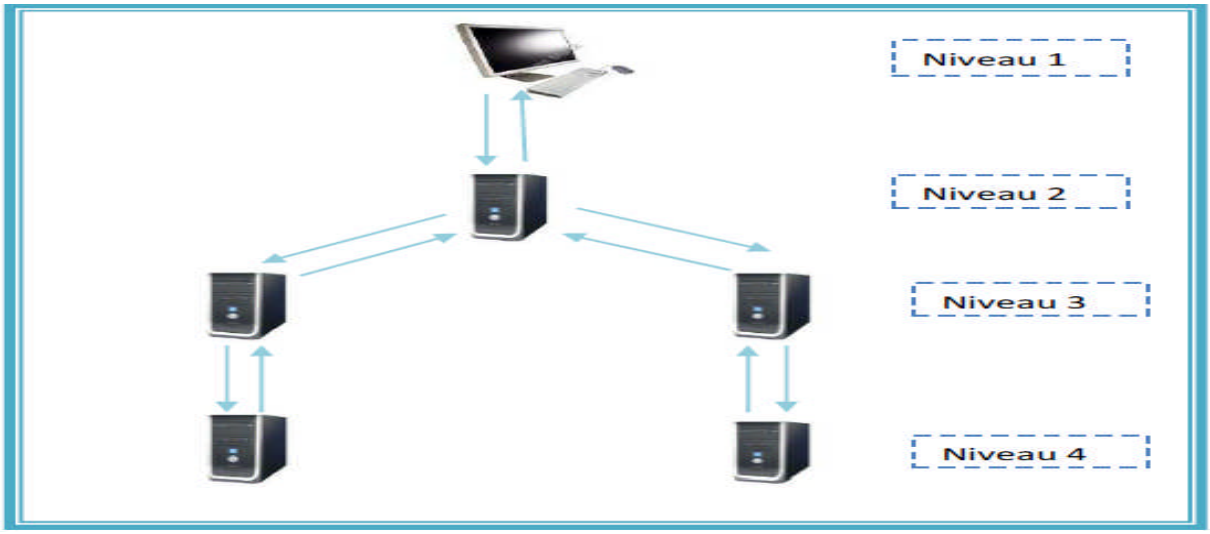



**Figure I.4 : Architecture à multi niveaux**

## I.2.4 Avantages et inconvénients de l'architecture client/serveur :

### I.2.4.1 Avantages de l'architecture client/serveur :

Le modèle client/serveur est particulièrement recommandé pour des réseaux nécessitant un grand niveau de fiabilité, ses principaux atouts sont :

- ✓ **des ressources centralisées** : étant donné que le serveur est au centre du réseau, il peut gérer des ressources communes à tous les utilisateurs, comme par exemple une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.

- ✓ **une meilleure sécurité** : car le nombre de points d'entrée permettant l'accès aux données est moins important.

- ✓ **une administration au niveau serveur** : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.

- ✓ **un réseau évolutif** : grâce à cette architecture il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeure.

### I.2.4.1 Inconvénients de l'architecture client/serveur :

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles :

- ✓ **un coût élevé** dû à la technicité du serveur
- ✓ **un maillon faible** : le serveur est le seul maillon faible du réseau client/serveur, étant donné que tout le réseau est architecturé autour de lui ! Heureusement, le serveur a une grande tolérance aux pannes.

## I.3 Java Entreprise Edition :

## I.3.1 Présentation de Java Entreprise Edition :

Java Entreprise Edition (Java EE) est une technologie 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 Entreprise Edition» pour désigner l'ensemble constitué des services (API) offerts et de l'infrastructure d'exécution. Java Entreprise Edition comprend notamment :

- ✓ Les spécifications du serveur d'application, c'est-à-dire de l'environnement d'exécution : Java Entreprise Edition 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 Entreprise Edition SDK (Java Entreprise Edition Software Development Kit)

L'utilisation de Java Entreprise Edition pour développer et exécuter une application propose plusieurs avantages :

- ➢ Une architecture d'application basée sur les composants qui permet un découpage de l'application et donc une séparation des rôles lors du développement.
- ➢ La possibilité de s'interfacer avec le système d'information existant grâce à de nombreuses API : JDBC, JNDI, JMS, JCA…
- ➢ La possibilité de choisir les outils de développement et le ou les serveurs d'applications utilisés qu'ils soient commerciaux ou libres.

## I.3.2 Fonctionnement interne :

Le langage Java, sur lequel les librairies Java Entreprise Edition sont utilisées, met à disposition un compilateur et une machine virtuelle (JVM – Java Virtual Machine) qui se charge de créer un environnement standard pour le lancement de l'application sur tout type de système opérationnel. Le compilateur compile le code source est produit le bytecode, soit un code intermédiaire qui sera en suite lit par la machine virtuelle Java. Chaque système opérationnel majeur possède une JVM expressément codée.

## I.3.3 Topologie d'une application Java Entreprise Edition :

### I.3.3.1 Application multi tiers :

Son architecture logicielle sépare les fonctions en plusieurs étages (states) de traitements.il existe trois modèles de tiers :

- ✓ Modèle 2-tier : client/serveur : 2niveaux=client et serveur.
- ✓ Modèle 3-tier : client (utilisateur), serveur d'application, serveur de données
- ✓ Modèle n-tier avec n 3 : client (utilisateur), divers niveaux de liaisons et de traitements de règles (serveur Web, serveurs d'application, brokers d'objet distants, moniteurs transactionnel, Web service), serveur de données.

### I.3.3.2 Application web :

En informatique, une application web (aussi appelée Web application) est un logiciel applicative, elle est placé sur un serveur et se manipule en actionnant des widgets à l'aide d'une interface graphique web, via un réseau informatique (Internet, Intranet, réseau local).

Exemple d'application web :

- ✓ Les messages web, les wiki, les blogs sont des applications Web.
- ✓ Les moteurs de recherche, les logiciels de commerce électronique, les jeux en ligne, les logiciels de forum peuvent être sous forme d'application Web.
- ✓ Des appareils réseau tels par exemple les routeurs sont parfois équipés d'une application Web dans leurs micros logiciels.

## I.3.4 architecture Java Entreprise Edition : [2]

Java Entreprise Edition ajout nombreuses couches de niveau entreprise au-dessus de la plate-forme Java SE - Java Standard Edition. Chaque couche est conçue pour supporter une différente technologie de développement.

• **Technologie web application**: technologies liées à la production des interfaces web dynamiques, par exemple JSP (Java Server Pages) et servlet.

• **Technologie entreprise application**: technologies plus directement liées à la logique de business : EJB (Enterprise Java Bean), JavaMail, JMS (Java Message Service), JTA (Java Transaction), etc.

• **Technologie web services**: technologies utiles au développement des applications adhérentes au paradigme SOA (Service Oriented Architecture) : web services, JAX-WS (java API for XML-based web services), JAX-RPC (Java API for XML-Based RPC).

• **Technologie management and security**: technologies liées à la gestion de la technologie enterprise afin de réaliser l'accès et l'échange d'information entre machines et services distribués : JAAS (Java Authentication and Authorization Service), JCA (Java Connector Architecture)

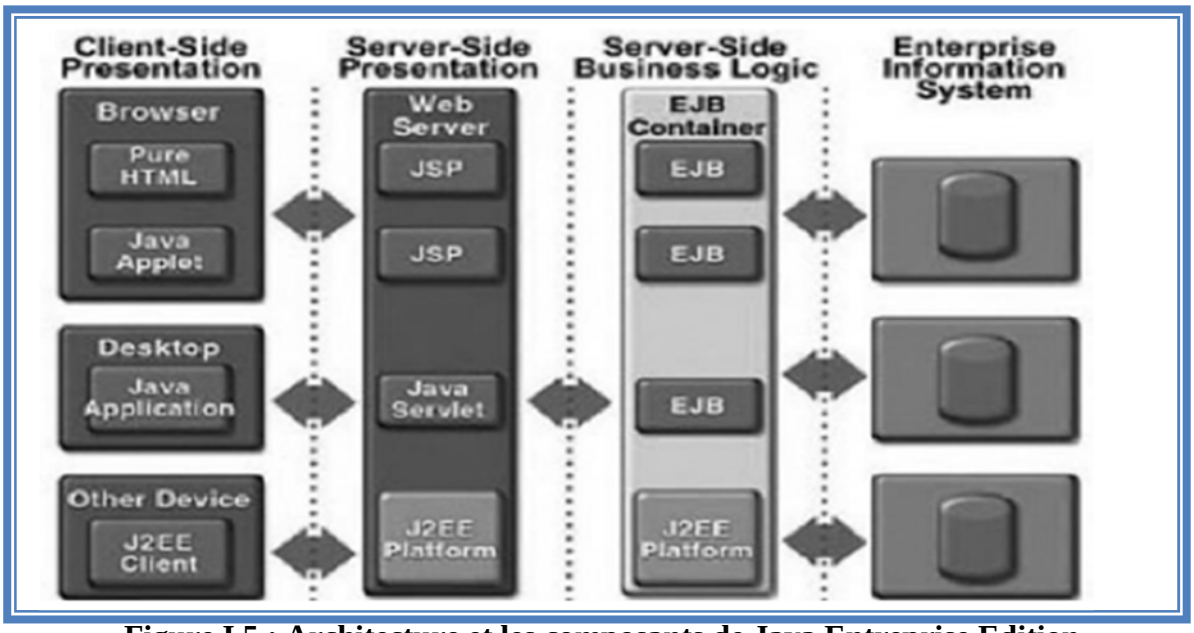



**Figure I.5 : Architecture et les composants de Java Entreprise Edition**

Pour expliquer l'utilisation de ces technologies on peut imaginer que les technologies Enterprise sont utilisées pour gérer l'accès aux données (généralement un ou plus database), les technologies web application sont utilisées pour montrer les données aux utilisateurs génériques. Dans un contexte Business to Business, les technologies web service seront utilisées pour échanger les informations avec les partenaires commerciales et les technologies de gestion gèrent tous les processus informationnels assurant la sécurité des transactions.

## I.3.5 Composantes de l'architecture Java Entreprise Edition :

- Conteneur de composants
- Client Java Entreprise Edition
- Composants web
- Composants Entreprise JavaBeans [EJB]
- Serveur EJB

### I.3.5.1 Conteneur de composants Java Entreprise Edition:

Les conteneurs assurent la gestion du cycle de vie des composants qui s'exécutent en eux. Les conteneurs fournissent des services qui peuvent être utilisés par les applications lors de leur exécution.

Il existe plusieurs conteneurs définit par Java Entreprise Edition:

- ✓ conteneur web : pour exécuter les servlets et les JSP
- ✓ conteneur d'EJB : pour exécuter les EJB
- ✓ conteneur client : pour exécuter des applications standalone sur les postes qui utilisent des composants Java Entreprise Edition

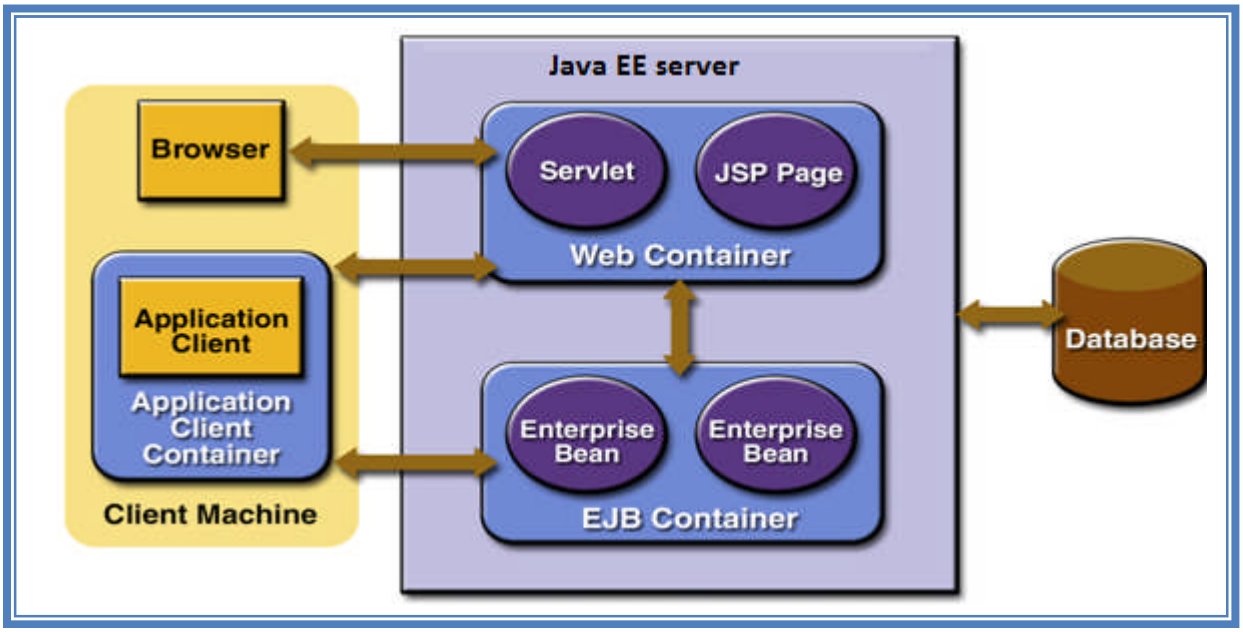



**Figure I.6 : l'architecture des conteneurs Java Entreprise Edition**

Les serveurs d'applications peuvent fournir un conteneur web uniquement (exemple: Tomcat) ou un conteneur d'EJB uniquement (exemple : JBoss, Jonas, ...) ou les deux (exemple : Websphere, Weblogic, ...).

Pour déployer une application dans un conteneur, il faut lui fournir deux éléments :

- ✓ l'application avec tous les composants (classes compilées, ressources ...) regroupés dans une archive ou module. Chaque conteneur possède son propre format d'archive.

- ✓ un fichier descripteur de déploiement contenu dans le module qui précise au conteneur des options pour exécuter l'application.

### I.3.5.2 les Clients Java Entreprise Edition :

- ➢ Les applications Java Entreprise Edition supportent des clients de différentes technologies.
- ➢ Selon la nature de ce client, la méthode de communication avec les composants Java Entreprise Edition varie :

**Un client léger ou riche (HTML côté navigateur):** il sera en communication avec le Web container via le protocole http sera interprété par les Servlet.

**Un client lourd ou une application Java** (autre qu'un navigateur, interface graphique) alors la communication sera directement avec le container d'EJB en utilisant une méthode de communication entre objets distants: RMI, IIOP, DCOM.

### I.3.5.3 Composants web :

Un composant web est une entité logicielle qui fournit une réponse à une requête. Les composants web génèrent habituellement l'interface utilisateur d'une application web. La plateforme Java Entreprise Edition définit deux types de composants web : les servlets et les Java Server Pages (JSP). La section suivante donne un aperçu de ces composants qui sont détaillés ultérieurement :

- ➢ **Servlets :**

Une servlet est un composant qui étend les fonctionnalités d'un serveur web de manière portable et efficace.

Un serveur web héberge des classes Java servlets qui sont exécutées à l'intérieur du container web. Le serveur web associe une ou plusieurs URLs à chaque servlet et lorsque ces URLs sont appelées via une requête HTTP de l'utilisateur la servlet est déclenchée. Quand la servlet reçoit une requête du client, elle génère une réponse, éventuellement en utilisant le logique métier contenu dans des EJBs ou en interrogeant directement une base de données. Elle retourne alors une réponse HTML ou XML au demandeur.

Un développeur de servlet utilise l'API servlet pour :

- ✓ Initialiser et finaliser la servlet
- ✓ Accéder à l'environnement de la servlet
- ✓ Recevoir ou rediriger les requêtes et envoyer les réponses Interagir avec d'autres serlvets ou composants
- ✓ Maintenir les informations de sessions du client
- ✓ Filtrer avant ou après traitement les requêtes et les réponses
- ✓ Implémenter la sécurité sur le tier web

- ➢ **Pages JSPs : [3]**

La technologie Java Server Pages(JSP) fournit un moyen simple et extensible pour générer du contenu dynamique pour le client web.

Une page JSP est un document texte qui décrit comment traiter la requête d'un client et comment créer une réponse.

Une page JSP contient :

Des informations de formatage (modèle) du document web, habituellement en HTML ou XML. Les concepteurs web peuvent modifier cette partie de la page sans affecter les parties dynamiques. Cette approche permet de séparer la présentation du contenu dynamique.
Des éléments JSP et de script pour générer le contenu dynamique du document Web. La plupart des pages JSP utilisent aussi des JavaBeans et/ou des Entreprise JavaBeans pour réaliser les opérations complexes de l'application. Les JSP permettent en standard d'instancier des beans, de modifier ou lire leurs attributs et de télécharger des applets. La technologie JSP est extensible en utilisant des balises personnalisées qui peuvent être encapsuler dans des bibliothèques de balises personnalisés (taglibs).

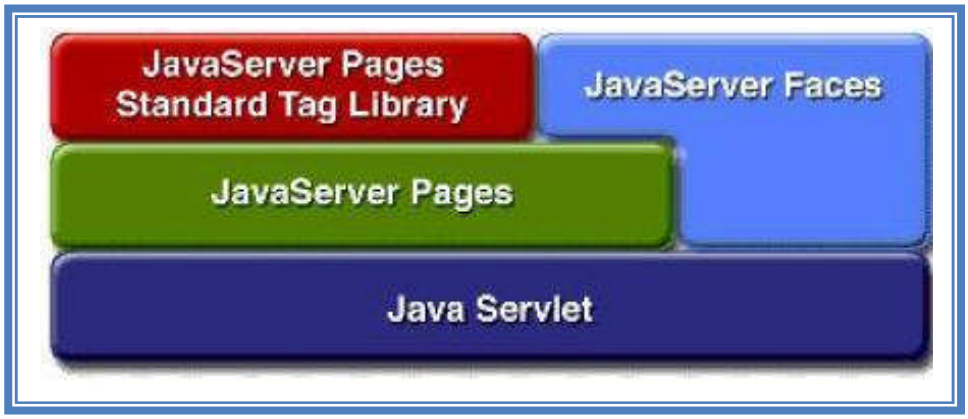



**Figure I.7 : les composants WEB**

### I.3.5.4 Composants Entreprise JavaBeans [EJB] :

L'architecture Entreprise JavaBeans est une technologie côté serveur pour développer et déployer des composants contenant le logique métier d'une application d'entreprise. Les composants Entreprise JavaBeans, aussi appelé Entreprise Beans, sont scalables, transactionnel et supporte l'accès concurrent.

Il y a 3 types d'entreprise beans : les beans de sessions, d'entité et de messages.

- ✓ **Les beans entité** modélisent des concepts métier pouvant être désignés par des noms. Par exemple un bean entité peut représenter un client, un objet d'un stock. Autrement dit, les bean entité correspondent à des objets réel. Ces objets sont dans la plupart des cas des enregistrements persistants stockés dans une base de données.

- ✓ **Les beans session** sont une extension de l'application client et ils sont responsable de La gestion des processus ou des taches.

- ✓ **Les beans orientés message** sont responsables de la coordination des tâches impliquant d'autres bean session et entité. La différence entre les beans orientés message et les beans entité réside dans la façon de d'y accéder. Un Bean session fournit une interface distante afin de définir les méthodes que l'on peut invoquer, alors que les Beans orientés message attend des messages asynchrones spécifiques auxquels il répond.

### I.3.5.5 Serveur EJB : [4]

C'est une machine de l'environnement EJB qui gère les conteneurs d'EJB .Les système d'entreprise distribués requirent un grand nombre de services communs .Cela inclut :

- ✓ La gestion de l'état.
- ✓ L'accès aux données partagées.
- ✓ La participation a des transactions.
- ✓ Le service d'un grand nombre de clients.
- ✓ La fourniture de services d'accès distant aux données et le contrôle de l'accès a ces données.

Le tier centrale de l'architecture Java Entreprise Edition qu'est le serveur d'EJB tente de réunir tous les services.

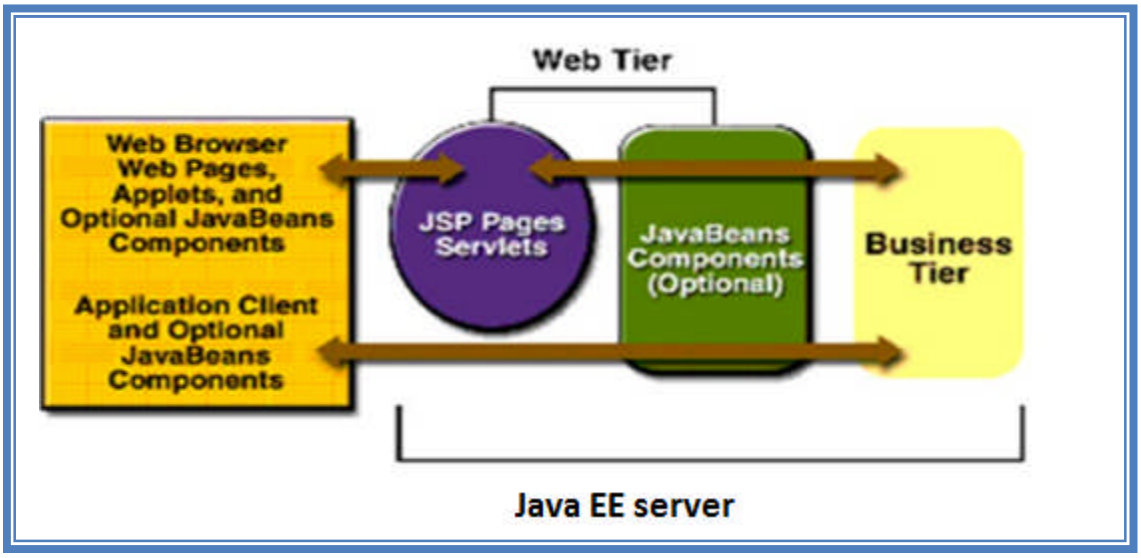



**Figure I.8 : Les communications d'un serveur Java Entreprise Edition**

- **Client EJB :**

Le client d'EJB fournit généralement la logique d'interface utilisateur sur la machine cliente. Il passe les appels aux composants EJB et interagir avec les composants EJB. Un composant EJB peut faire office de client EJB en appelant les méthodes d'un autre composant EJB.

## I.3.6 Les services de Java Entreprise Edition : [6]

Les services, pouvant être classés par catégories:

### I.3.6.1 Les services d'infrastructures :

Il en existe un grand nombre, définis ci-dessous :

**JDBC** (Java DataBase Connectivity) est une API d'accès aux bases de données Relationnelles.

**JNDI** (Java Naming and Directory Interface) est une API d'accès aux services de nommage et aux annuaires d'entreprises tels que DNS, NIS, LDAP, etc.

**JCA** (Java Entreprise Edition Connector Architecture) est une API de connexion au système d'information de l'entreprise, notamment aux systèmes dits «Legacy» tels que les ERP.

**JMX** (Java Management Extension) fournit des extensions permettant de développer des Applications.

### I.3.6.2 Les services de communication :

**JAAS** (Java Authentication and Authorization Service) est une API de gestion de l'authentification et des droits d'accès.

**JavaMail** est une API permettant l'envoi de courrier électronique.

**JMS** (Java Message Service) fournit des fonctionnalités de communication asynchrone (appelées MOM pour Middleware Object Message) entre applications.

**RMI-IIOP** est une API permettant la communication synchrone entre objets.
L'architecture Java Entreprise Edition 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.

## I.3.7 Technologies Java Entreprise Edition :

La plateforme Java Entreprise Edition comme la plateforme Java Standard Edition incluent un grand nombre de bibliothèques de code (API) prédéfinies pour les fonctions de base d'une application. Les technologies mises en œuvre sont les suivantes :

L'architecture Java Entreprise Edition Connector est l'infrastructure pour interagir avec une grande variété de systèmes d'information d'entreprise tels que des ERPs, des CRM, et autres progiciels. L'API JDBC est utilisée pour accéder à des données relationnelles à partir de programmes Java La Java Transaction API (JTA) est utilisée pour gérer et coordonner les transactions entre un ensemble hétérogène de systèmes d'information d'entreprise.

L'API Java Naming and Directory Interface est utilisée pour accéder aux services de nommage et d'annuaire de l'entreprise. L'API Java Message Service (JMS) est utilisée pour émettre et recevoir des messages via les systèmes de messagerie d'entreprise comme IBM MQ Series ou TIBCO Rendez-vous. Dans l'architecture Java Entreprise Edition les Message Driven Beans fournissent une approche à base de composant pour encapsuler les fonctionnalités de messagerie. La JavaMail API est utilisée pour émettre et recevoir des emails. Java IDL est utilisé pour appeler des services CORBA L'API Java pour XML (JAXP) est utilisée pour l'intégration avec les systèmes et applications existants et pour implémenter les webservices dans la plateforme Java Entreprise Edition.

### I.3.8 L'environnement d'exécution des applications Java Entreprise Edition : [7]

Java Entreprise Edition propose des spécifications pour une infrastructure dans laquelle s'exécutent les composants. Ces spécifications décrivent les rôles de chaque élément et précisent un ensemble d'interfaces pour permettre à chacun de ces éléments de communiquer.

Ceci permet de séparer les applications et l'environnement dans lequel elles s'exécutent. Les spécifications précisent à l'aide des API un certain nombre de fonctionnalités que doit implémenter l'environnement d'exécution. Ces fonctionnalités sont de bas niveau ce qui permet aux développeurs de se concentrer sur la logique métier.

Pour exécuter ces composants de natures différentes, Java Entreprise Edition définit des conteneurs pour chacun d'eux. Il définit pour chaque composant des interfaces qui leur permettront de dialoguer avec les composants lors de leur exécution. Les conteneurs permettent aux applications d'accéder aux ressources et aux services en utilisant les API.

Les appels aux composants se font par des clients en passant par les conteneurs. Les clients n'accèdent pas directement aux composants mais sollicitent le conteneur pour les utiliser.

### I.3.9 L'assemblage et le déploiement d'application Java Entreprise Edition : [7]

Java Entreprise Edition propose une spécification pour décrire le mode d'assemblage et de déploiement d'une application Java Entreprise Edition.

Une application Java Entreprise Edition peut regrouper différents modules : modules web, modules EJB ... Chacun de ces modules possède son propre mode de packaging. Java Entreprise Edition propose de regrouper ces différents modules dans un module unique sous la forme d'un fichier EAR (Entreprise ARchive).

- ✓ Le format de cette archive est très semblable à celui des autres archives :
- ✓ un contenu : les différents modules qui composent l'application (module web, EJB, fichier RAR, ...)
- ✓ un fichier descripteur de déploiement

Les serveurs d'applications extraient chaque module du fichier EAR et les déploient séparément un par un.

### I.3.10 Les acteurs d'une application Java Entreprise Edition :

La réalisation d'une application basée sur l'architecture Java Entreprise Edition fait appel à différents types de compétences que l'on trouve rarement chez une même personne car cela va de la conception jusqu'à la supervision de l'application en passant par le développement et le

Déploiement. Afin de pouvoir maîtriser ce processus Java Entreprise Edition adopte l'approche des partages des responsabilités.
Plus spécifiquement pour les EJB, cette approche définit plusieurs niveaux de responsabilité :

- ✓ Le fournisseur des EJB : c'est l'acteur qui fournit des composants métiers réutilisables soit par l'achat à un fournisseur de composants, soit par développement interne ;
- ✓ L'assembleur d'applications : l'acteur qui est capable de construire une application à partir d'EJB existants ;
- ✓ Le déployeur : l'acteur qui récupère l'application et s'occupe de son déploiement dans un serveur d'applications ;
- ✓ L'administrateur : l'acteur qui contrôle le fonctionnement du serveur d'application et assure la supervision des applications ;
- ✓ Le fournisseur de conteneur : l'éditeur qui commercialise un conteneur web ou un conteneur EJB ; cet éditeur commercialise souvent un serveur d'application incluant ces conteneurs ;
- ✓ Le fournisseur de serveur : c'est l'éditeur qui commercialise un serveur d'application (BEA, IBM etc...) ;

## I.4 Conclusion :

Au cours de ce chapitre, nous avons présenté l'architecture Client/serveur, son fonctionnement et ses classifications, nous avons parlé aussi de l'architecture Java Entreprise Edition, de ces composants, certaines d'entre eux seront utilisées pour le développement de notre application. Nous allons présenter dans le chapitre suivant l'organisme d'accueil ENIEM (entreprise national des industries de l'électroménager) où se déroulé notre stage pratique.

# *Chapitre II : Présentation de l'organisme d'accueil ENIEM*

## II.1Introduction :

Dans ce chapitre nous mettons notre travail dans son contexteon présente une vue globale sur l'organisme d'accueil, puis on explore en détaille notre champ d'étude afin de prendre des connaissances approfondis de ses fonctionnalités.

## II.2Historique :

L'entreprise ENIEM (entreprise national des industries de l'électroménager) se trouve au sein de la zone industrielle AISSAT –IDIR OUED-AISSI à 10km de TIZI-OUZOU, elle s'étale sur une surface totale de 55 Hectares sa direction générale se trouve à la proximité de la gare ferroviaire de Tizi-Ouzou (sud-ouest).

L'entreprise ENIEM possède des capacités de production et une expérience de 35 ans dans la fabrication et le développement dans les différentes branches de l'électroménager, notamment :

- ✓ Les appareils ménagers domestiques.
- ✓ Les lampes d'éclairage.
- ✓ Les produits sanitaires.

## II.3 Présentation de l'organigramme général de l'ENIEM :

Comme le montre le schème cité ci-après :

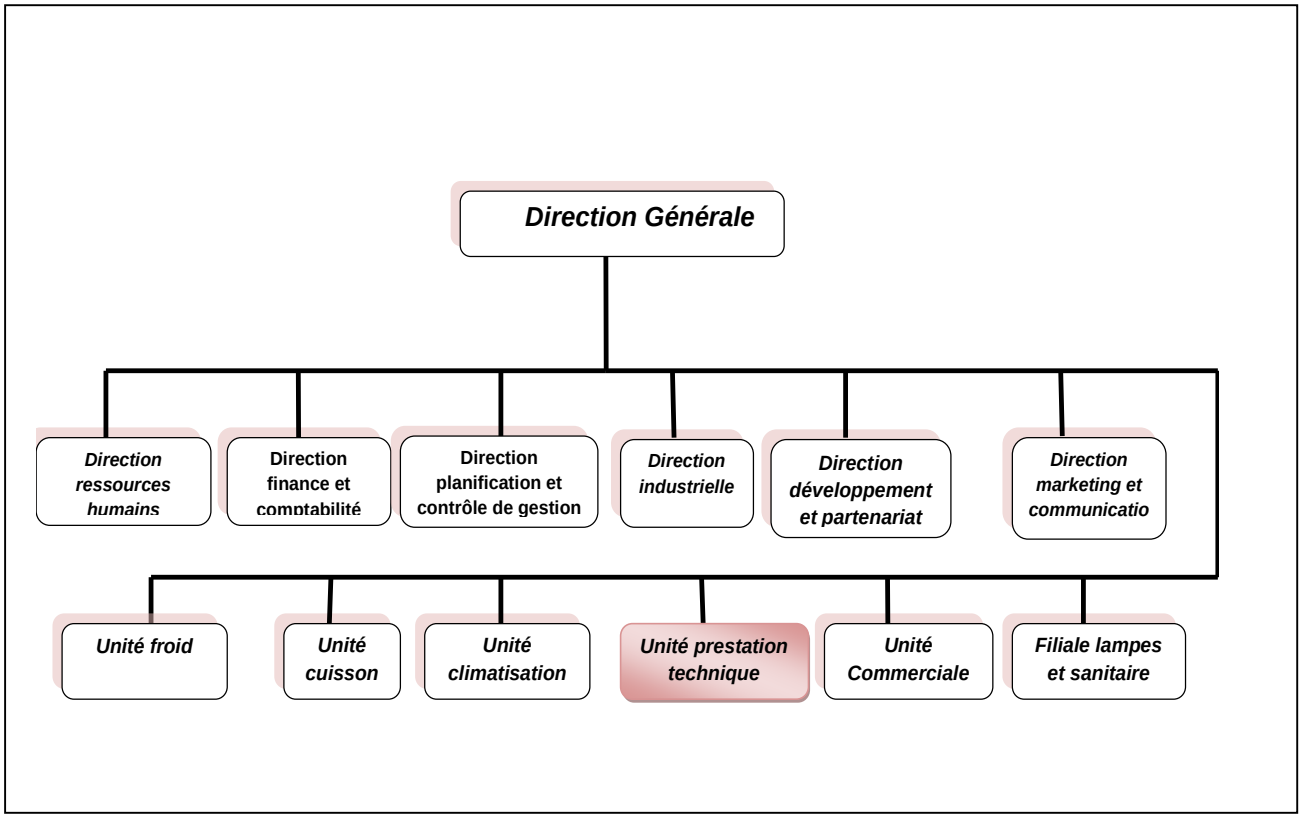



**Figure II.1: Présentation du l'organigramme générale de ENIEM**

té où se trouve notre champ d'étude.

## II.4 Missions et objectifs :

### a. MISSIONS :

La mission de l'entreprise c'est d'assurer la production et la recherche dans les différentes branches de l'électroménager notamment :

- ✓ Les appareils de réfrigération et de congélation par l'unité Froid.
- ✓ Les appareils de cuisson par l'unité cuisson.
- ✓ Les appareils de climatisation par l'unité climatisation.

**b. ACTIVITES :**

L'activité principale de l'ENIEM se concentre sur la fabrication de réfrigérateurs, cuisinières, climatiseurs.

**c. OBJECTIFS :**

L'ENIEM vise les objectifs suivants :

- ✓ L'amélioration de la qualité des produits.
- ✓ La maitrise des coûts de production.
- ✓ L'augmentation des capacités, d'étude et de développement.
- ✓ L'amélioration de la maintenance de l'outil de production.
- ✓ La valorisation des ressources humaines.
- ✓ L'augmentation des taux d'intégration (interne/externe).
- ✓ Augmentation du volume de production.
- ✓ Le renforcement de la sécurité du patrimoine et des installations.
- ✓ La réduction du cout de non qualité.

## II.5 Mode d'organisation :

### II.5.1 Les directions :

ENIEM est dotée d'une direction généraleet six directions centrales :

**a)Direction générale** :

La direction générale, l'unique entité qui est responsable de la stratégie et du développement de l'Entreprise. Elle exerce son autorité hiérarchique et fonctionnelle sur l'ensemble des directions et des unités.

**b)Direction industrielle** :

Elle est chargée de développer et de mettre en place les moyens et l'organisation industrielle nécessaire à la réalisation de la production en agissant sur les approvisionnements, les moyens et les techniques de production.

**c)Direction des ressources humaines :**

En cohérence avec la politique qualité de l'entreprise, la fonction Ressources Humaines accroit la mobilisation et la valorisation du personnel dans ses actions au service du client.

Elle pilote le recrutement, l'accueil, l'information et gère le plan des carrières du personnel. Elle conçoit le plan de formation à partir du recueil des besoins collectifs et individuels et assure son exécution.

Elle supervise la gestion administrative et légale pour le personnel et les pouvoir publique en respectant les objectifs de conformité fiabilité, délais.

**d)Direction finances et comptabilités :**

Garant des obligations légales, des règles comptables et des procédures de l'entreprise, dont elle vérifie l'application par la mise en œuvre d'un contrôle interne.

**e)Direction développement et partenariat :**

Responsable des études et du développement du produit finis ainsi que des actions de partenariat et de sous –traitante ainsi elle :

- ✓ Suit avec la direction industrielle les actions de développement des processus de fabrication et de modernisation de l'outil de production .En vue de l'amélioration de la rentabilité et des conditions de travail.
- ✓ Développe d'autres créneaux pour l'utilisation maximale et des capacités technologiques de l'entreprise.

**f) Direction planification et contrôle de gestion :**

Cette direction est responsable de la gestion, de l'audit finances ainsi que du budget de l'entreprise. Cependant elle :

- ✓ Exploite et analyse l'information relative aux agrégats afin de préconiser les actions correctives nécessaires avec toute l'anticipation attendue.
- ✓ Exploite les résultats de l'audit finances, les interprète et fait les recommandations nécessaires.
- ✓ Prépare, établit et suit le budget de l'entreprise.

**g) Direction de marketing et de la communication :**

La direction du marketing et de la communication décide en collaboration avec le Président Directeur Générale politiques commerciales et de communication qu'elle met en œuvre par l'élaboration des méthodes et outils de gestion nécessaires.

**II.5.2 Les unités :**

L'entreprise s'est organisée par centres d'activités stratégiques qui se composent de cinq (04)unités de production, d'une unité commerciale et d'une unité de prestations techniques :

**a) Unité froid :**

La mission globale de cette unité est la production et le développement des produits de froid tél que, les petits réfrigérateurs et les congélateurs verticale et horizontale.

**b) Unité cuisson** :

Cette unité à pour mission, la production et le développement des produits de cuisson, gaz électrique ou mixte et tous produits de technologie similaire .Elle produit des cuisinières 4 et 5 feux et constituée de (04) ateliers de fabrication.

**c) Unité climatisation :**

Cette unité est spécialisée dans la production et le développement des produits de climatisations, de chauffage, elle est composée de (04) ateliers de fabrication.

**d) Unité prestations techniques :**

Cette unité est chargée principalement de gérer et exploiter les moyens communs (produits d'énergie et utilités) utilisés Dans le processus de production des autres unités ainsi que la gestion des infrastructures communes.

**e) Unité commerciale :**

Cette unité est chargée de la commercialisation des produits fabriqués, de la promotion des exportations et de la gestion du réseau SAV.

**f) Unité produits sanitaires et filampes :**

L'unité lampes et produits sanitairesde Mohammedia est spécialisée dans la fabrication de lampes d'éclairage domestique ainsi que des lampes de réfrigérateurs et de développer des produits sanitaires(baignoires, lavabos…). Elle est devenue filiale à 100% ENIEM le 01 janvier1997.

## II.6 Présentation de champ d'étude :

Notre travail concerne l'unité prestations techniquesdont l'organigramme est présenté ci-dessous :

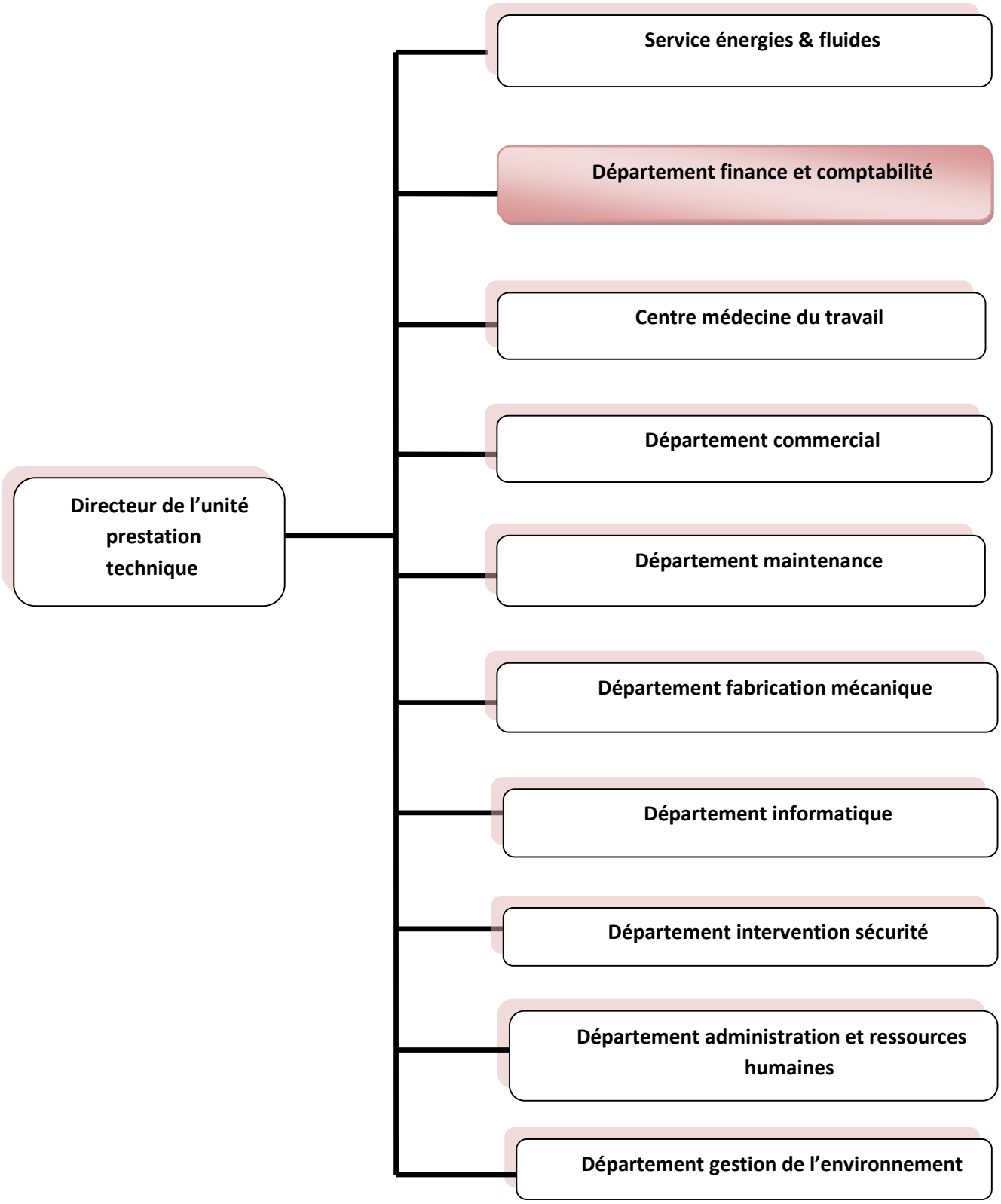



**Figure II.2 : Organigramme général de l'unité prestations techniques**

### II.6.1 Description de l'organigramme :

L'unité prestations techniques est subdivisée en plusieurs départements dont le département finance et comptabilité qui est à son tour subdivisée en plusieurs services dont le service finance.

### II.6.2 Organigramme du champ d'étude :

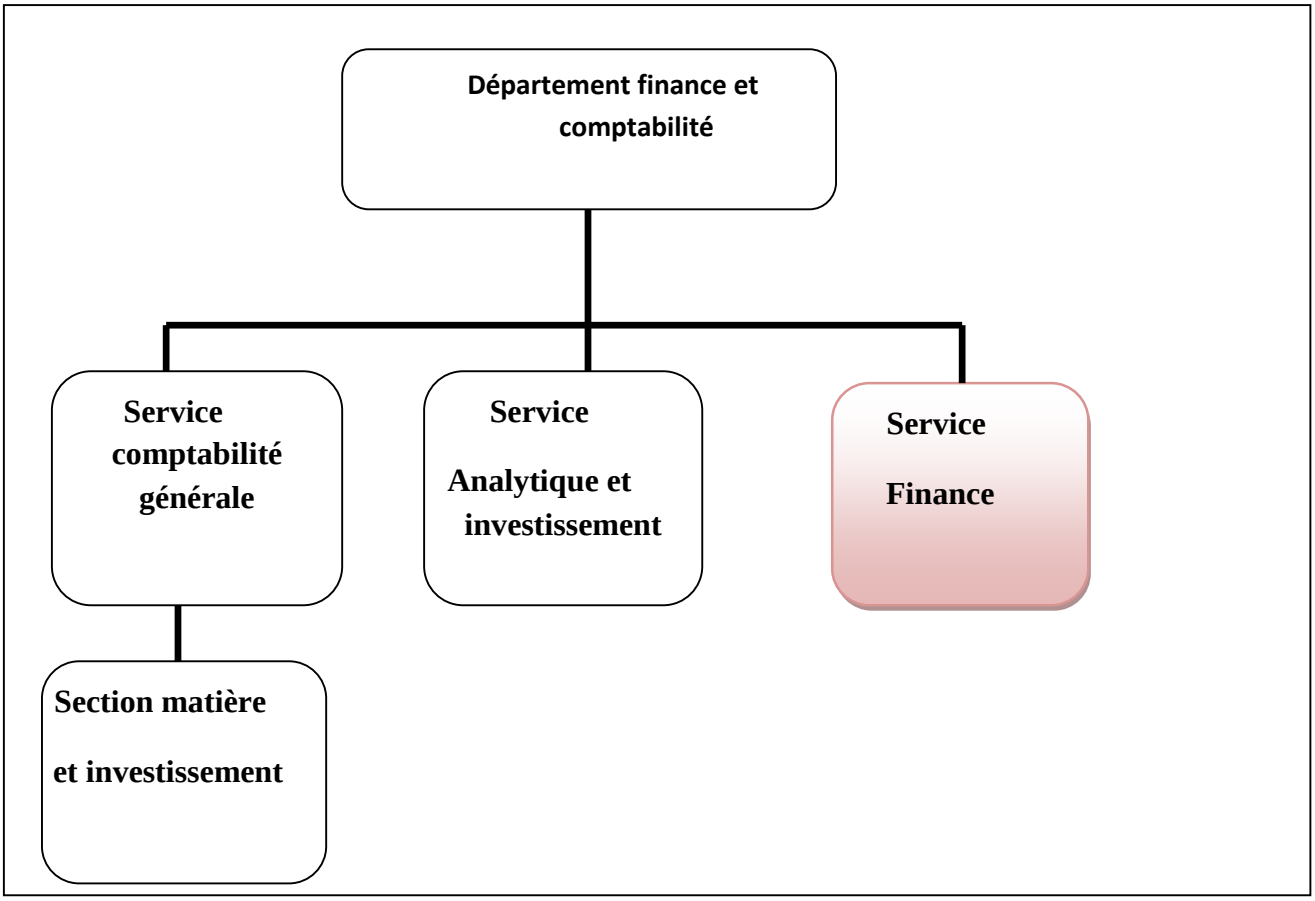



**Figure II.3 : Organigramme du champ d'étude**

### II.7 la situation informatique du champ d'étude :

Le service finance est équipé de 2 Micro ordinateur et une imprimante suivant :

| Matériels | Description | Logiciel utilisé |
|---|---|---|
| **micro-ordinateur 1** | **Processeur** : Intel Pentium 3<br>**Ram** : 256 MO<br>**Disque** : 40 GO | Système d'exploitation (millenium)<br>Office<br>-anti virus AVAST<br>-DBASE |
| **micro-ordinateur 2** | **Processeur** : Intel Pentium 2<br>**Ram** : 128 MO<br>**Disque** : 40 GO | Système d'exploitation :<br>Windows XP Professionnel)<br>-office<br>-Réflexion |
| **impriment** | **Marque** : LQ2090 | |

**Tableau 1 : la situation informatique du champ d'étude**

### II.8Etude de l'existant:

L'étude de l'existant est la première phase du processus unifié. Elle a pour but ultime de la clarification du champ de notre investigation. On doit prendre en considération :

- ✓ La capture des besoins et l'analyse
- ✓ Etudier profondément les traitements pour lesquels on souhaite automatiser le fonctionnement.
- ✓ L'ensemble des objectifs fixé.

## II.9 Diagramme de flux :

Le diagramme des flux permet de :

- ✓ mieux identifier les acteurs interne et externe par rapport au domaine d'étude
- ✓ identifier les flux échangés entre les acteurs.
- ✓ Délimiter le domaine d'étude

### II.9.1Concepts utilisés :

Le domaine d'étude est défini par la personne qui analyse la circulation de l'information au sein d'un système d'information. Il délimite l'étude à une ou plusieurs activités précises au sein d'une organisation donnée.

**Domaine d'étude :**

- ✓ **Acteur :** Un acteur est un émetteur ou un récepteur d'un flux d'information lié à une activité au sein du système d'information d'une organisation. Selon le cas, il peut s'agir d'une catégorie de personne, d'un service ou du système d'information d'une autre organisation.

On distingue :

- les acteurs internes qui font partie du domaine d'étude.
- les acteurs externes qui ne font pas partie du domaine mais qui ont des échanges avec les acteurs internes dans le cadre de l'activité étudiée.

- ✓ **Flux :**Un flux part d'un acteur source pour aboutir à un acteur but (représenté par une flèche), le numéro du flux et inscrit sur la flèche tracée.

- ✓ **Domaine connexe :** Est un composant du système d'information interagissant avec le domaine d'étude. C'est un acteur interne à l'entreprise, mais externe au domaine d'étude.

### II.9.2 Diagramme des flux d'information :

Le schéma suivant représente le diagramme des flux d'informations :

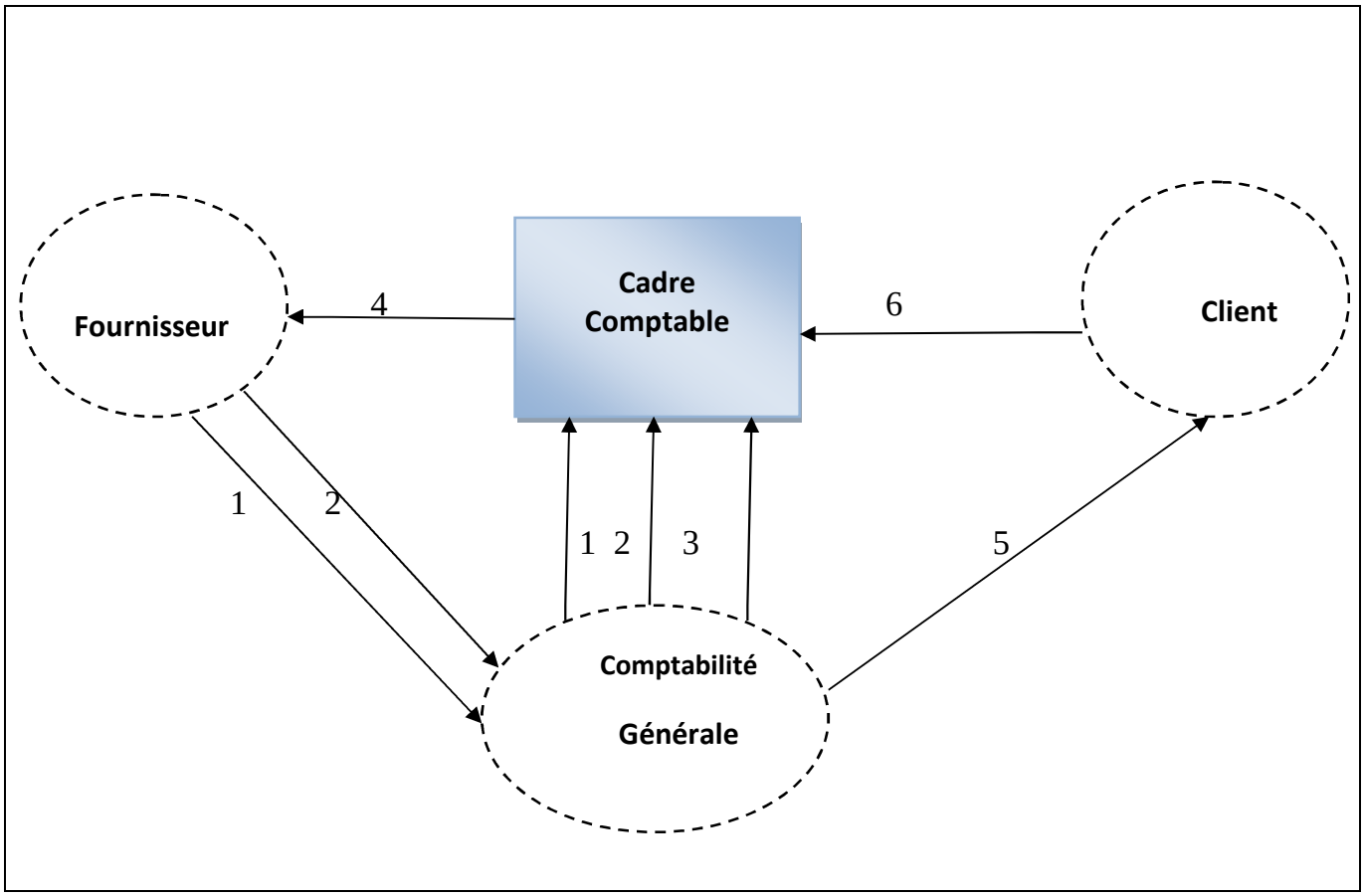



**Figure II.4 : Diagramme des flux**

### II.9.3 Liste des flux d'information :

| Numéro | description |
|---|---|
| **1** | Facture fournisseur |
| **2** | Bon de livraison |
| **3** | Bon réception |
| **4** | Règlement facture fournisseur |
| **5** | Facture client |
| **6** | Règlement facture client |

## II.10 Critiques de l'existant et suggestions:

L'étude de l'existant nous a conduits à noter quelques insuffisances dans le système actuel de l'ENIEM causés principalement par le manque de communication et de circulation de l'information entre les différents acteurs.

En effet, nous les avons classés en trois catégories:

- **Insuffisances d'ordre organisationnel** : elles concernent généralement les postes de travail, classement et archivage de documents.
- **Insuffisances d'ordre informationnel:** elles concernent généralement les rubriques des documents utilisés par les services du département Finance et comptabilité de L'ENIEM.
- **Insuffisances d'ordre technique:** elles concernent en général les outils et le matériel informatique utilisé par les services du département Finance et comptabilité de L'ENIEM.

### II.10.1 Niveau organisationnel :

- **Problème 1 :** Le volume élevé des archives et sa non indexation ne permet pas une recherche optimale des documents.
- **Cause** : Absence d'une bonne structure d'archivage.
- **Suggestion :** Sauvegarder les informations sur des supports numériques pour permettre une recherche rapide des informations.

- **Problème 2 :** Le financier à chaque fois change sa propre méthode de travail.
- **Exemple :** Le classement des documents dans le dossier de finance.
- **Cause** : Le financier n'est pas soumis à une méthode fixe de travail.
- **Suggestion :** Etablissement d'un plan unique et optimal pour spécifier la méthode de classement des documents, et pour toute méthode de travail.

### II.10.2 Niveau informationnel:

- **Problème 1 :** Redondance de certaines informations sur le même document
- **Cause :** Manque de généralisation des rubriques des documents.
- **Suggestion :** Généraliser la rubrique et supprimer les redondances

- **Problème 2 :** Non remplissage de certaines rubriques des documents
- **Exemple :**
  - N° de bon de réception sur la facture fournisseur.
- **Cause** :
  - Mal conception des documents
  - Absence de mise à jour de la structure du document en réponse au changement de procédure.
  - Utilisation d'un même document pour plusieurs fonctions (facture client, facture fournisseur).
- **Suggestion :** Remplir les rubriques utiles et éliminer les autres jugés inutiles.

### II.10.3 Niveau technique :

Apres l'étude de l'existant nous avons pu déceler quelques anomalies au niveau de la solution informatique utilisée.

- **Problème 1 :** Lenteur dans le traitement de certaines informations effectuées manuellement.
- **Suggestion :** Automatiser les tâches concernées.

- **Problème 2 :** L'obsolète application utilisée ne permet pas une prise en main facile de l'outil, son manque d'ergonomie : utilisation uniquement au clavier cause des difficultés de manipulation qui ont un impact direct sur la productivité du financier.
- **Suggestion :** Refonte totale de l'application avec les technologies modernes afin d'offrir une interface plus conviviale et une manipulation plus souple.

## II.11 Conclusion :

Dans ce chapitre nous avons pris connaissance de l'organisme d'accueil : ses moyens, ses missions et ses objectifs.

Une attention particulière a été prêtée pour l'unité prestation technique à travers une brève présentation, du fait que cette unité inclut le domaine de notre étude à savoir la gestion financière. Dans le chapitre suivant nous allons présenter l'analyse et la conception en utilisant la démarche RUP (Rational Unified Process).

# *Chapitre III :*

# *Analyse et conception*

## III.1 Introduction :

Il est important d'utiliser une méthode rigoureuse, pour développer des applications informatiques, en mettant en évidence tous les objectifs traces pour la bonne élaboration du projet souhaité.

Dans le cadre de notre projet nous avons opté pour une démarche de conception orientée objet RUP (Relational Unified Process) qui utilise le langage de modélisation unifiée (UML). Pour cela nous suivant les phases de développement suivantes :

**Initialisation :** définition du problème (vision imprécise de la finalité du projet, cas d'entreprise, périmètre, estimation globale).

**Elaboration :** vision plus élaborée, implémentation itérative de l'architecture noyau, identification de la plupart des besoins et du périmètre réel, estimation plus réaliste et résolution des risques élèves.

**Construction :** implémentation itérative des éléments qui présente des risque et une complexité moindres et préparation du déploiement.

**Transition :** recettage et déploiement.

Les phases de développement sont les grandes étapes du développement du logiciel, le projet commence en phase d'initialisation et termine en phase de transition.

**III.2 Phase d'initialisation :** cette phase permet l'analyse approximative de notre projet permet de desséver les points suivantes :

- ✓ Notre application touche à la gestion payement fournisseur et recouvrement,
- ✓ Notre application web est de type Client/serveur.
- ✓ Notre application est sécurise par la mis en œuvre d'un système de sécurisation (authentification).

## III.3 Phase d'élaboration :

Notre travail consiste à élaboré une application WEB sous la plateforme Java Entreprise Edition (Java EE) pour la gestion de payement fournisseur et recouvrement de l'entreprise ENIEM pour la gestion des différentes procédures de finances, et qui permet l'édition des documents administratif comme :

La facture client : établit par l'ENIEM.

Le bon de livraison : établit par le fournisseur.

Le bon de réception : document établit après la réception d'un bon de livraison.

La facture fournisseur : établit par le fournisseur.

La réalisation d'une base de donne plus adéquate pour l'archivage des documents et mieux garder l'historique des pannes survenus, elle nous permet aussi l'automatisation des activités de la gestion payement fournisseur et recouvrement.

### III.3.1 Identification des acteurs :

- ✓ **Acteur :** un acteur et une entité externe interagissant avec le système.

  **Administrateur :** personne chargé de la gestion des comptes utilisateurs.

  **Financier :** personne chargé de gestion des factures, gestion des dettes et recouvrement.

  **Directeur :** Garant des obligations légales, des règles comptables et des procédures de l'entreprise, dont il valide les factures… .

### III.3.2 Spécification des tâches :

- ✓ **Tache :** ensemble des fonctions qu'un acteur bien spécifié peut effectuer.

Le tableau suivant résume les taches que nous acteurs assure :

| **Acteurs** | **Taches** |
|---|---|
| **Financier** | T01 : S'authentifier.<br>T02 : Gérer les clients.<br>T03 : Gérer les fournisseurs.<br>T04 : Gérer les factures clients.<br>T05 : Gérer les factures fournisseurs.<br>T06 : Gérer les bons de livraisons.<br>T07 : Gérer les bons de réceptions.<br>T08 : Gérer les dettes.<br>T09 : Gérer les recouvrements.<br>T10 : Gérer les payements pas échéance.<br>T11 : Gérer le solde unité.<br>T12 : Consulter les rapports fournisseurs.<br>T13 : Consulter les rapports clients.<br>T14 : Se déconnecter. |
| **Administrateur** | T01 : S'authentifier.<br>T02 : Gérer les comptes utilisateurs.<br>T03 : Se déconnecter. |
| **Directeurs** | T01 : S'authentifier.<br>T02 : Consulter les rapports fournisseurs.<br>T03 : Consulter les rapports clients.<br>T04 : Se déconnecter. |

### III.3.3 Diagramme de contexte :

Le diagramme du contexte est un modèle conceptuelle qui permet d'avoir une vision globale des interactions entre les systèmes et les liens avec l'environnement exterieur.il permet aussi de délimiter le champ d'étude.

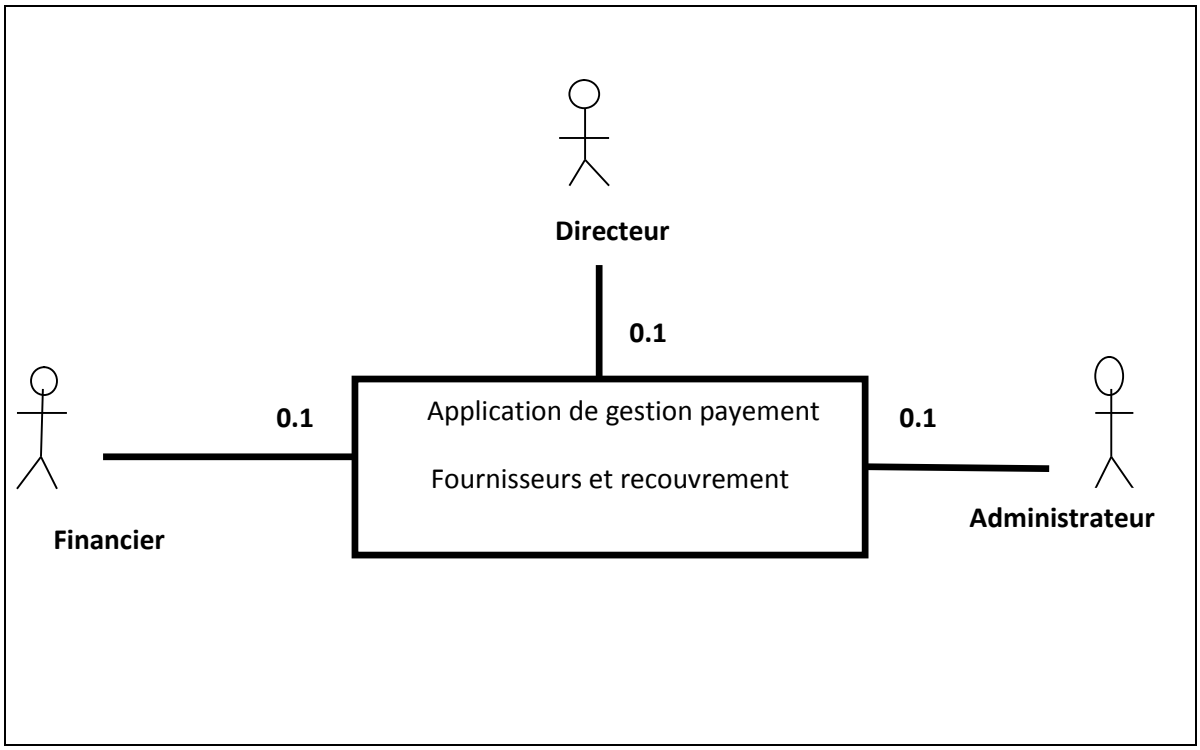



**Figure III.1 : Diagramme de contexte relatif a l'application de gestion de payement fournisseur et recouvrement**

### III.3.4 Spécification de quelque cas d'utilisation avec scénarios :

- **Un cas d'utilisation :** représente la structure des grandes fonctionnalités nécessaires Aux utilisateurs du système. C'est le premier diagramme du modèle UML, celui ou s'assure la relation entre l'utilisateur et les objets mets en œuvre par le système.
- **Un scénario :** est une succession particulière d'action, s'exécutant du début jusqu'à la fin du cas d'utilisation. Les scenarios doivent être définis par les utilisateurs eux même, chaqu'une des taches effectuées par un ou plusieurs acteurs sera décrite par un ensemble des scenarios.

**Remarque :** On va décrire seulement quelques-uns des cas d'utilisation vu leurs nombres important.

Les figures suivantes représentent une description de certains cas d'utilisation de notre système :

- **Cas d'utilisation N°1: «** identification d'utilisateur **»**

**Acteur :** Utilisateur (tous les utilisateurs du système).

**Pré-condition :** L'utilisateur doit se connecter

**Post-condition :** Accès au système

**Scenario nominal :**

1. Demande d'ouverture de session.
2. Le système affiche quatre champs de session (login, password, profile).
3. L'utilisateur introduit le login, password, profile, profil
4. Le système vérifie l'existence de l'utilisateur dans le système.
5. Le système affiche l'interface correspondant à l'utilisateur

**Enchainement alternatif :**

A1 :l'identification est provisoirement erronée.

L'enchainement A1 démarre au point 4 du scénario nominal.

5. Le système indique a l'utilisateur que le code est erroné pour la 1 ère ou 2 eme fois, le scénario nominal reprend au point 2.

**Enchainement d'exception :**

E1 : code d'identification définitivement erroné.

5. Le système indique à l'utilisateur que le code est erroné pour la 3 eme fois
6. Le cas d'utilisation se termine.

- **Cas d'utilisation N°2: «** Modifier le mot de passe **»**

**Acteur :** Utilisateur (tous les utilisateurs du système).

**Pré-condition :** L'utilisateur doit se connecter (password, login, profile).

**Post-condition :** mot de passe modifier.

**Scenario nominal :**

1. L'utilisateur atteint son interface puis ses pages et clique sur le bouton « changer le mot de passe ».
2. Le système lui renvoi un formulaire contenant les champs : ancienne password, nouveau password, confirme le mot de passe.

3. L'utilisateur remplit le formulaire et soumit la requête.
4. Le système vérifie le mot de passe.
5. Le système lui renvoi un message de confirmation.

**Enchainement alternatif :**

A1 : le mot de passe est provisoirement erroné.

L'enchainement A1 démarre au point 4 du scénario nominal.

5. Le système indique à l'utilisateur que le code est erroné pour la 1 ère ou 2 eme fois, le scénario nominal reprend au point 2.

**Enchainement d'exception :**

E1 : le mot de passe définitivement erroné.

5. Le système indique à l'utilisateur que le code est erroné pour la 3 eme fois
6. Le cas d'utilisation se termine.

- **Cas d'utilisation N°3: «** Ajouter compte utilisateur **»**

**Acteur :** Administrateur.

**Pré-condition :** L'administrateur doit être connecté (password, login, profile).

**Post-condition :** compte créé.

**Scenario nominal :**

1. l'administrateur atteint la page compte.
2. Le système retourne le formulaire du compte.
3. L'administrateur saisit et sélectionne les informations.
4. L'administrateur clique sur le bouton « Ajouter ».
5. le système vérifier la validité des données saisies.
6. Le système affiche un message de confirmation.

**Scénario alternatif :**

L'enchainement commence au point 5 du scenario nominal.

6. le système affiche un message d'erreur indiquant la donnée manquante.

Le système nominal reprend au point 3.

✓ **cas d'utilisation N°4: «**Gérer les Factures **»**

- **Ajouter Facture**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** facture créé.

**Scenario nominal :**

1. financier atteint la page facture.
2. Le système retourne le formulaire de la facture.
3. Le financier saisit et sélectionne les informations.
4. Le financier clique sur le bouton « Ajouter ».
5. le système vérifier la validité des données saisies.
6. Le système affiche un message de confirmation.

**Scénario alternatif :**

L'enchainement commence au point 5 du scenario nominal.

6. le système affiche un message d'erreur indiquant la donnée manquante.

Le système nominal reprend au point 3.

- **Modifier la facture :**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** facture modifie.

**Scenario nominal :**

1. financier atteint la page facture.
2. Le système retourne le formulaire de la facture.
3. Le financier saisit et sélectionne les champs du formulaire.
4. Le financier clique sur le bouton « rechercher ».
5. Le système affiche la liste des factures trouvé.
6. Le financier choisit une facture à modifier.
7. Le financier effectue la modification de facture et la sauvegarde.
8. Le système affiche un message de confirmation.

- **Supprimer la facture :**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** facture supprimé.

**Scenario nominal :**

1. financier atteint la page facture.
2. Le système retourne le formulaire de la facture.
3. Le financier saisit et sélectionne les champs du formulaire.
4. Le financier clique sur le bouton « rechercher ».
5. Le système affiche la liste des factures trouvé.
6. Le financier choisit une facture à supprimer.
7. Le système affiche un message de confirmation.

- **Consulter facture :**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** afficher la liste des factures.

**Scenario nominal :**

1. financier atteint la page facture.
2. Le système retourne le formulaire de la facture.
3. Le financier saisit et sélectionne les champs du formulaire.
4. Le financier clique sur le bouton « rechercher ».
5. Le système affiche la liste des factures à consulter.
6. Le système affiche les informations de la facture choisit.

✓ **Cas d'utilisation N°5: «**Gérer les Dettes **»**

- **Ajouter Dette**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** Dette créé.

**Scenario nominal :**

1. Financier atteint la page dette.
2. Le système retourne le formulaire de la dette.
3. Le financier saisit et sélectionne les informations.
4. Le financier clique sur le bouton « Ajouter ».
5. le système vérifier la validité des données saisies.
6. Le système affiche un message de confirmation.

**Scénario alternatif :**

L'enchainement commence au point 5 du scenario nominal.

6. le système affiche un message d'erreur indiquant la donnée manquante.

Le système nominal reprend au point 3.

- **Modifier la dette :**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecter (password , login, profile).

**Post-condition :** dette modifie.

**Scenario nominal :**

1. financier atteint la page dette.
2. Le système retourne le formulaire de la dette.
3. Le financier saisit et sélectionne les champs du formulaire.
4. Le financier clique sur le bouton « rechercher ».
5. Le système affiche la liste des dettes trouvé.
6. Le financier choisit une dette à modifier.
7. Le financier effectue la modification de dette et la sauvegarde.
8. Le système affiche un message de confirmation.

- **Supprimer la dette :**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** dette supprimé.

**Scenario nominal :**

1. financier atteint la page dette.
2. Le système retourne le formulaire de la dette.
3. Le financier saisit et sélectionne les champs du formulaire.
4. Le financier clique sur le bouton « rechercher ».
5. Le système affiche la liste des dettes trouvé.
6. Le financier choisit une dette à supprimer.
7. Le système affiche un message de confirmation.

- **Consulter dette:**

**Acteur :** Utilisateur (Financier).

**Pré-condition :** Le Financier doit être connecté (password, login, profile).

**Post-condition :** afficher la liste des dettes.

**Scenario nominal :**

1. financier atteint la page dette.
2. Le système retourne le formulaire de la dette.
3. Le financier saisit et sélectionne les champs du formulaire.
4. Le financier clique sur le bouton « rechercher ».
5. Le système affiche la liste des dettes à consulter.
6. Le système affiche les informations de la dette choisit.

### III.3.5 Représentation des diagrammes de cas d'utilisation :

- ✓ **Diagramme de cas d'utilisation :**

Les diagrammes de cas d'utilisations sont les premiers diagrammes du modèle UML, s'assurent la relation entre l'utilisateur et les objets mets en œuvre par le système. Représente la structure des grandes fonctionnalités nécessaires Aux utilisateurs du système.

- ✓ **Définition des itérations :**

Le critère de définition des itérations se base sur les deux points suivants :

- Ressemblance entre cas d'utilisation.
- Dépendance entre cas d'utilisation.

✓ **Représentation du diagramme de cas d'utilisation global :**

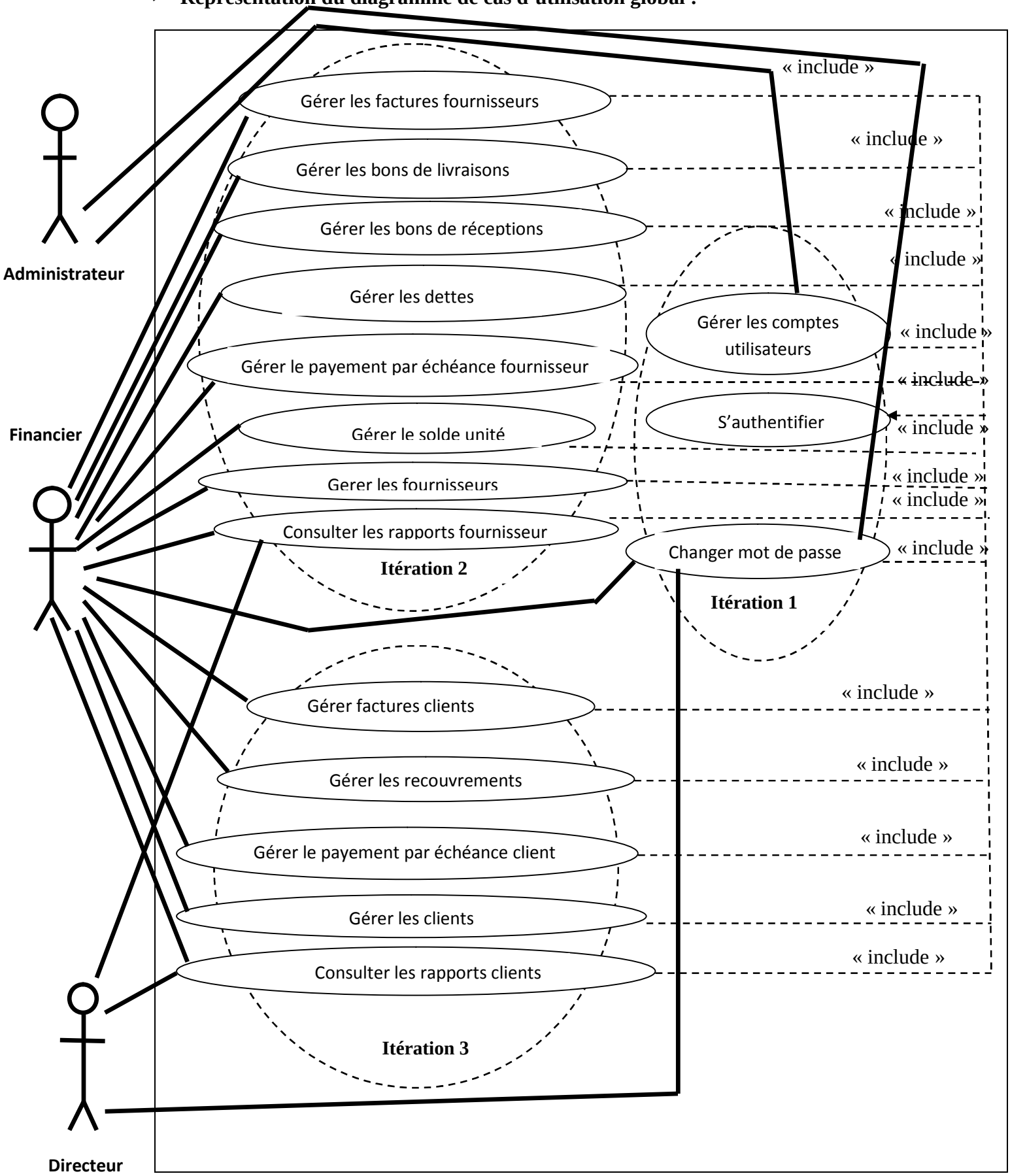
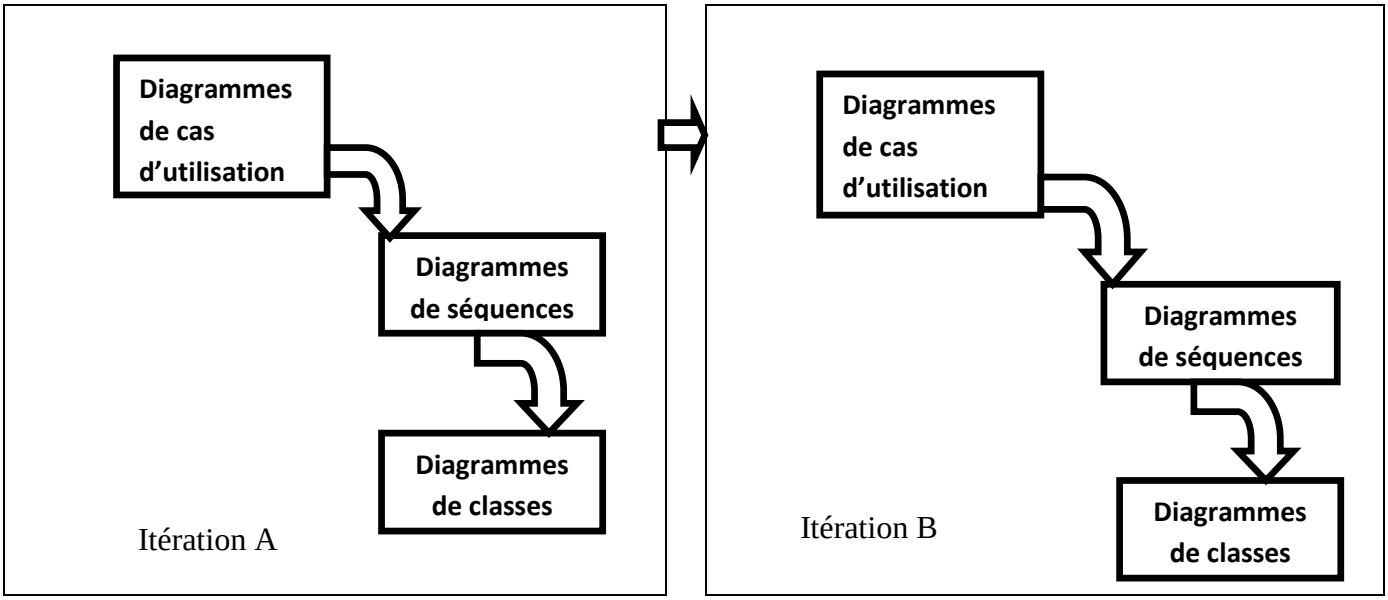



**Figure III. 2: Le diagramme de cas d'utilisation global.**

## III.4 Construction :

Pour chaque itération réalisée précédemment, nous allons suivre un cycle de vie en cascade pour aboutir à un diagramme de classe en passant par un diagramme de séquence selon la figure suivante :




### III.4.1 Réalisation de l'itération 1 :

#### III.4.1.1 Diagramme de séquence pour itération 1 :

Le diagramme de séquences permet de montrer les interactions d'objets dans le cadre d'un scénario d'un Diagramme des cas d'utilisation. Le but étant de décrire comment se déroulent les actions entre les acteurs ou objets.

Apres avoir déterminé nos besoins et réaliser le diagramme de cas d'utilisation nous allons élaborer quelques diagrammes de séquences de l'itération.

- **Modifier mot de passe :**

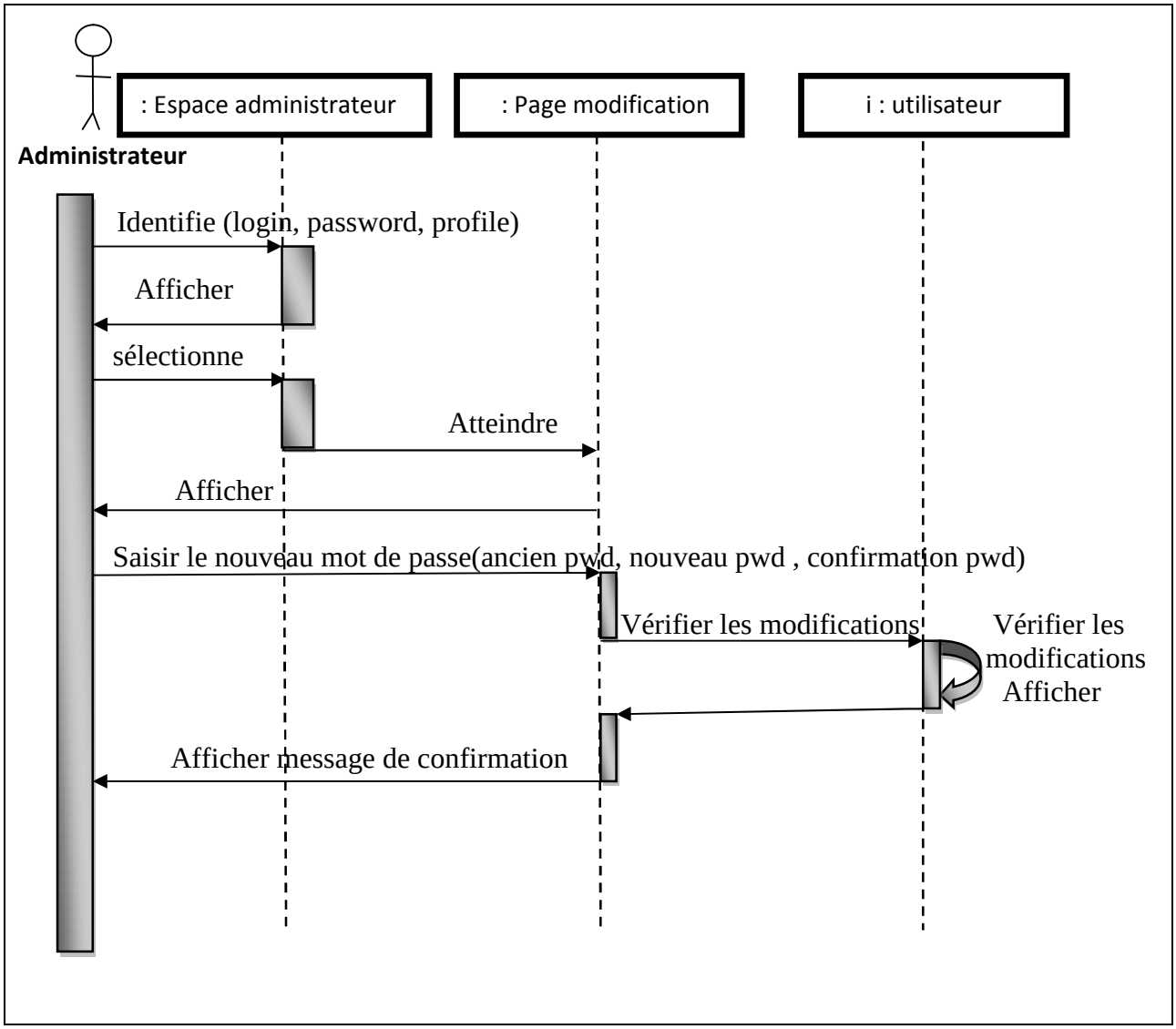



**Figure III.3: diagramme de séquence associé au scénario**

**«Modifier mot de passe »**

- **Ajouter compte utilisateur:**

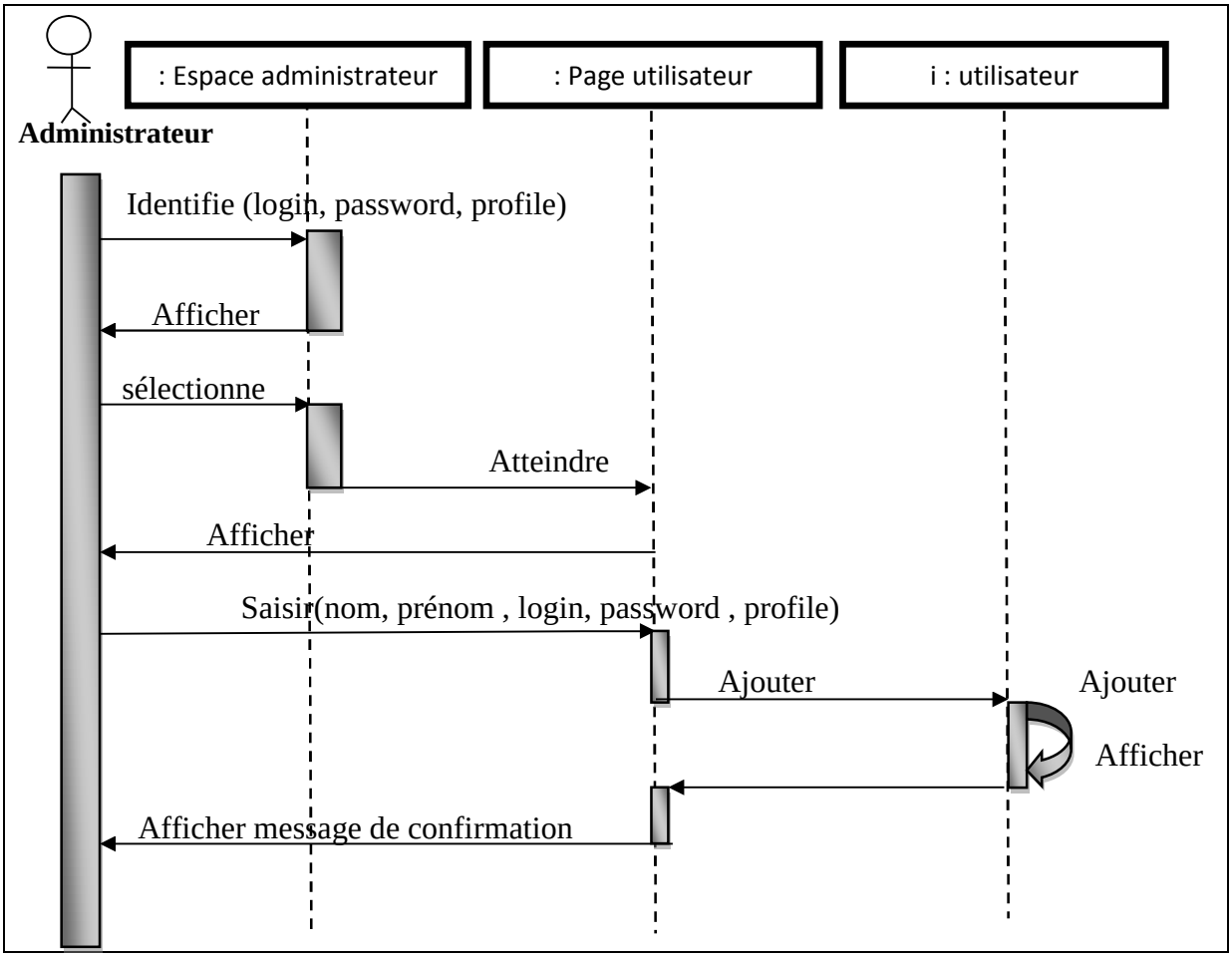



**Figure III.4: diagramme de séquence associé au scénario « Ajouter compte utilisateur »**

### III.4.1.2 Diagrammes de classe pour itération 1 :

Les diagrammes de classes sont les plus courants dans la modélisation des systèmes orientés objet. Ils représentent un ensemble de classes, d'interfaces et de collaboration ainsi que leurs relations. Ces diagrammes sont utilisés pour modéliser la vu de conception statique.

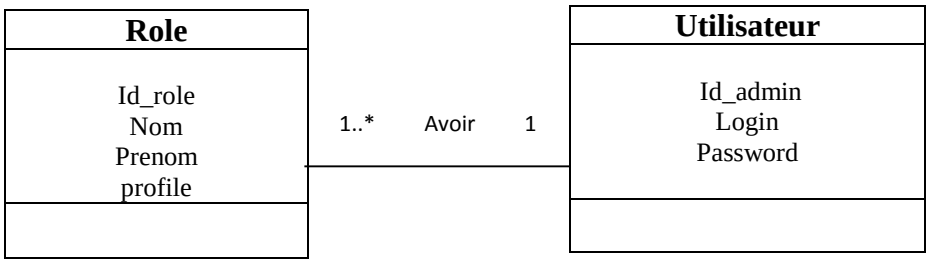



**Figure III.5: Diagramme de classe pour l'itération 1**

### III.4.2 Réalisation d l'itération 2 :

**III.4.2.1 Diagramme de séquence pour itération 2 :** Voici quelque diagramme de séquence de l'itération 2.

- **Ajouter facture fournisseur :**

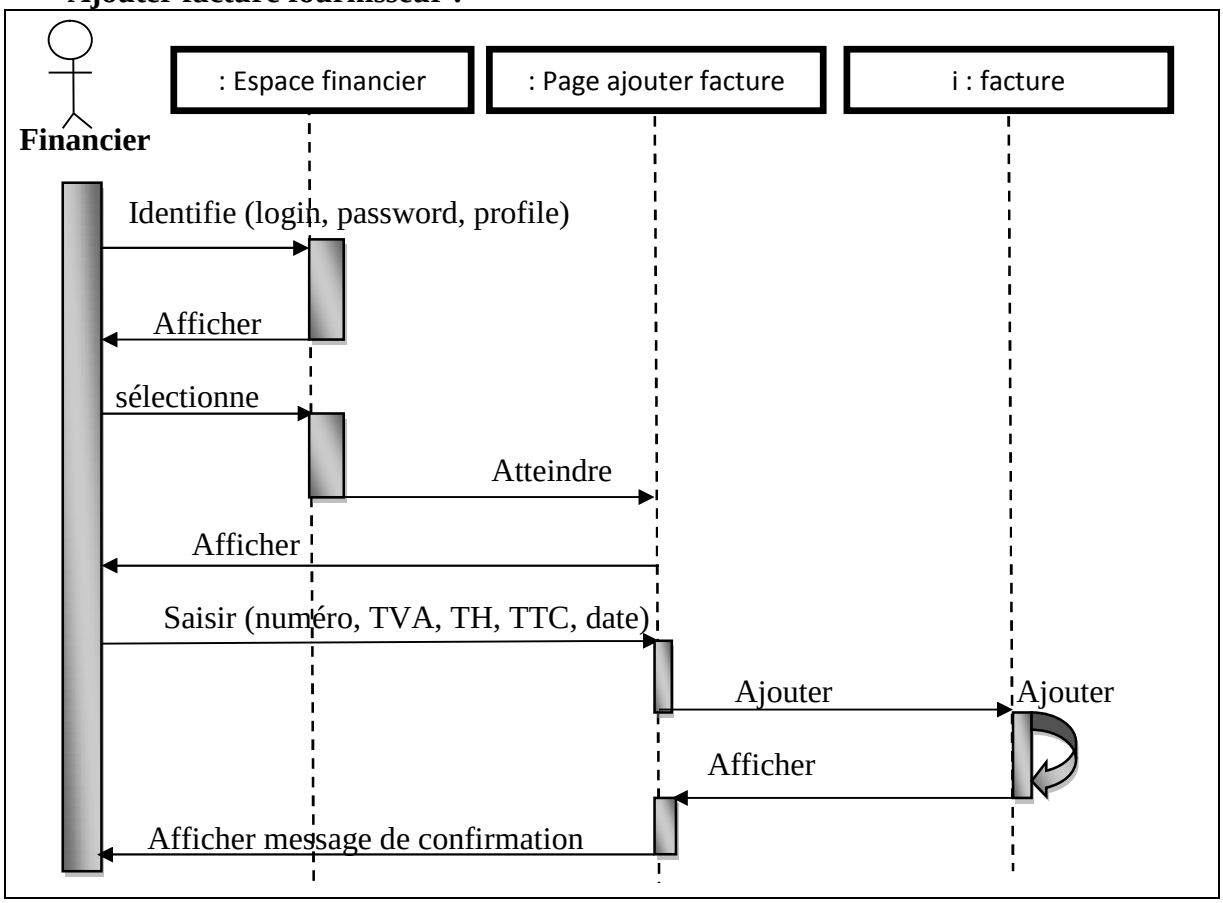



**Figure III.6 : diagramme de séquence associé au scénario« Ajouter facture fournisseur »**

- **Modifier facture fournisseur :**

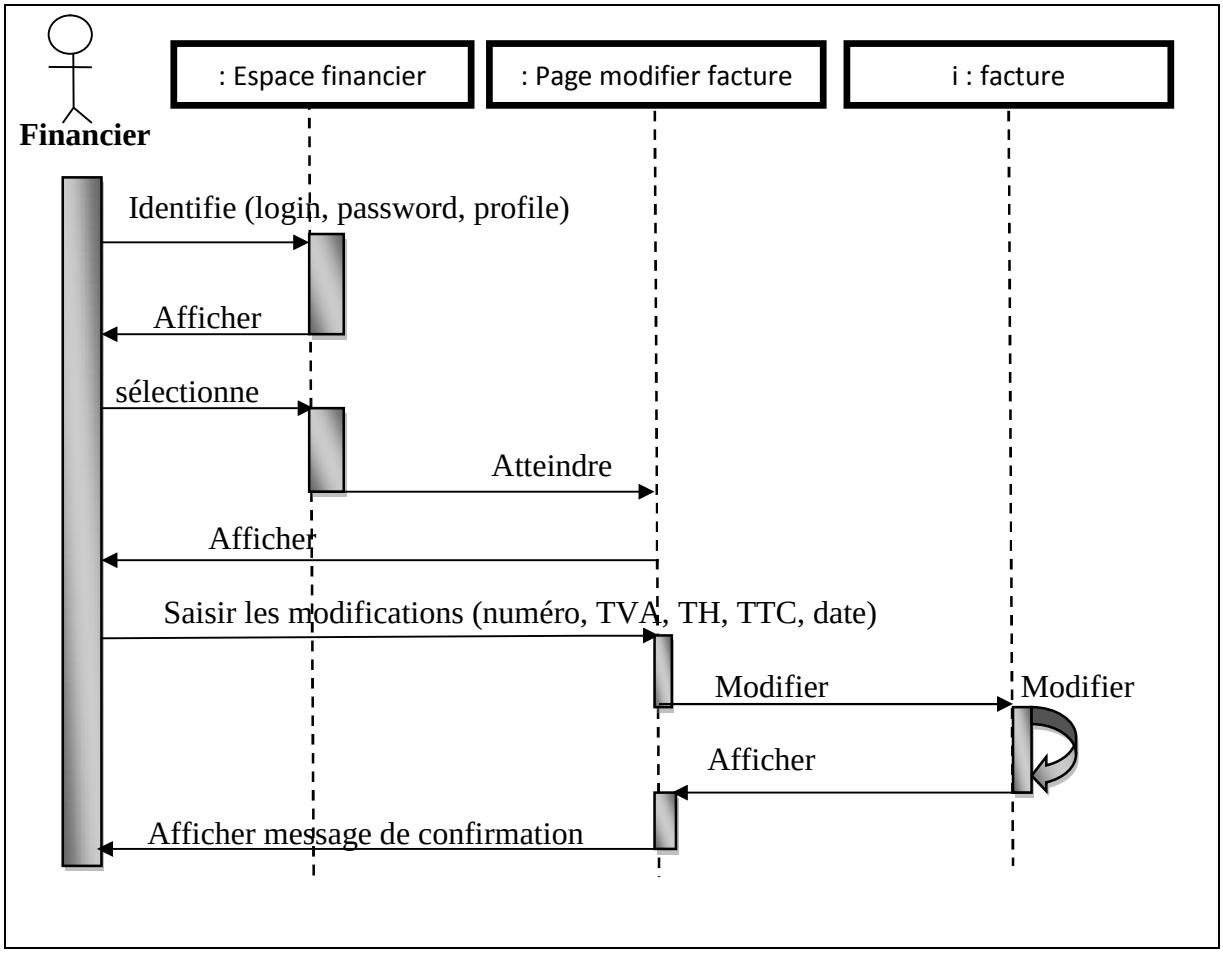



**Figure III.7 : diagramme de séquence associé au scénario**

**« Modifier facture fournisseur »**

### III.4.2.2 Diagramme de classe pour l'itération 2 :

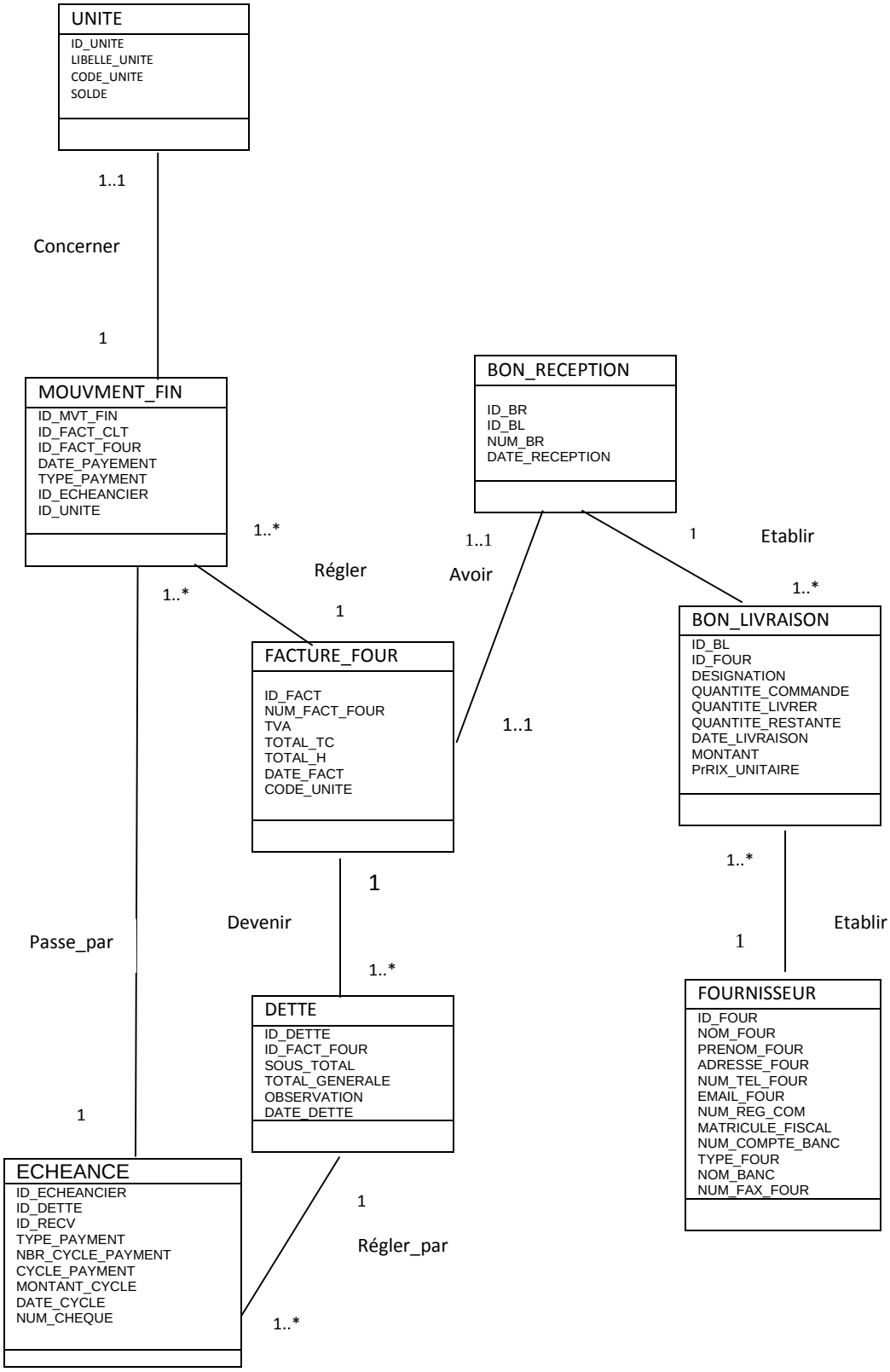



**Figure III.8 : le diagramme de classe pour l'itération2**

### III.4.3 Réalisation d l'itération 3 :

**III.4.3.1 Diagramme de séquence pour itération 3 :** Voici quelque diagramme de séquence de l'itération 3.

- **Consulter recouvrement :**

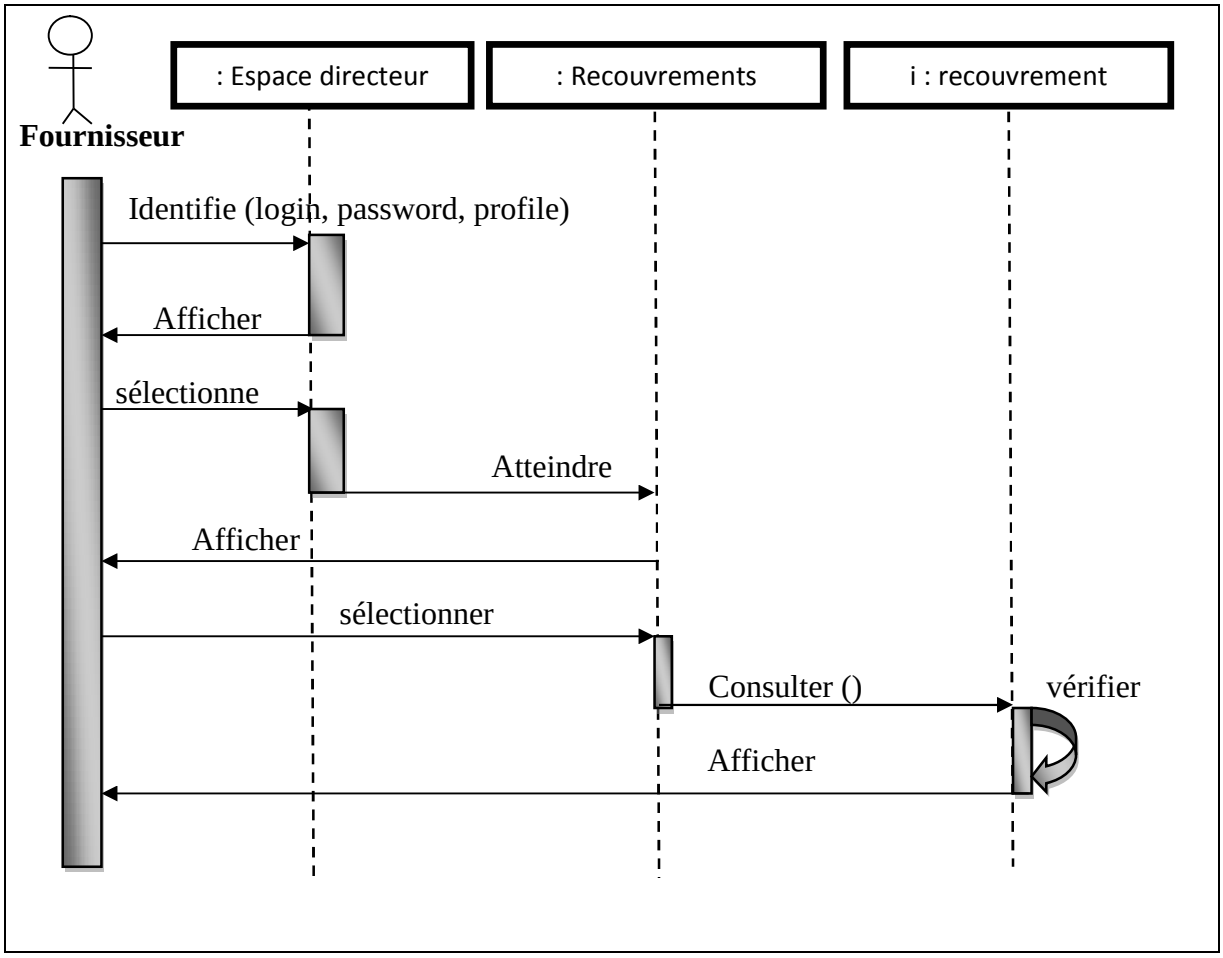



**Figure III.9 : diagramme de séquence associé au scénario**

**« Consulter recouvrement »**

- **Modifier facture client :**

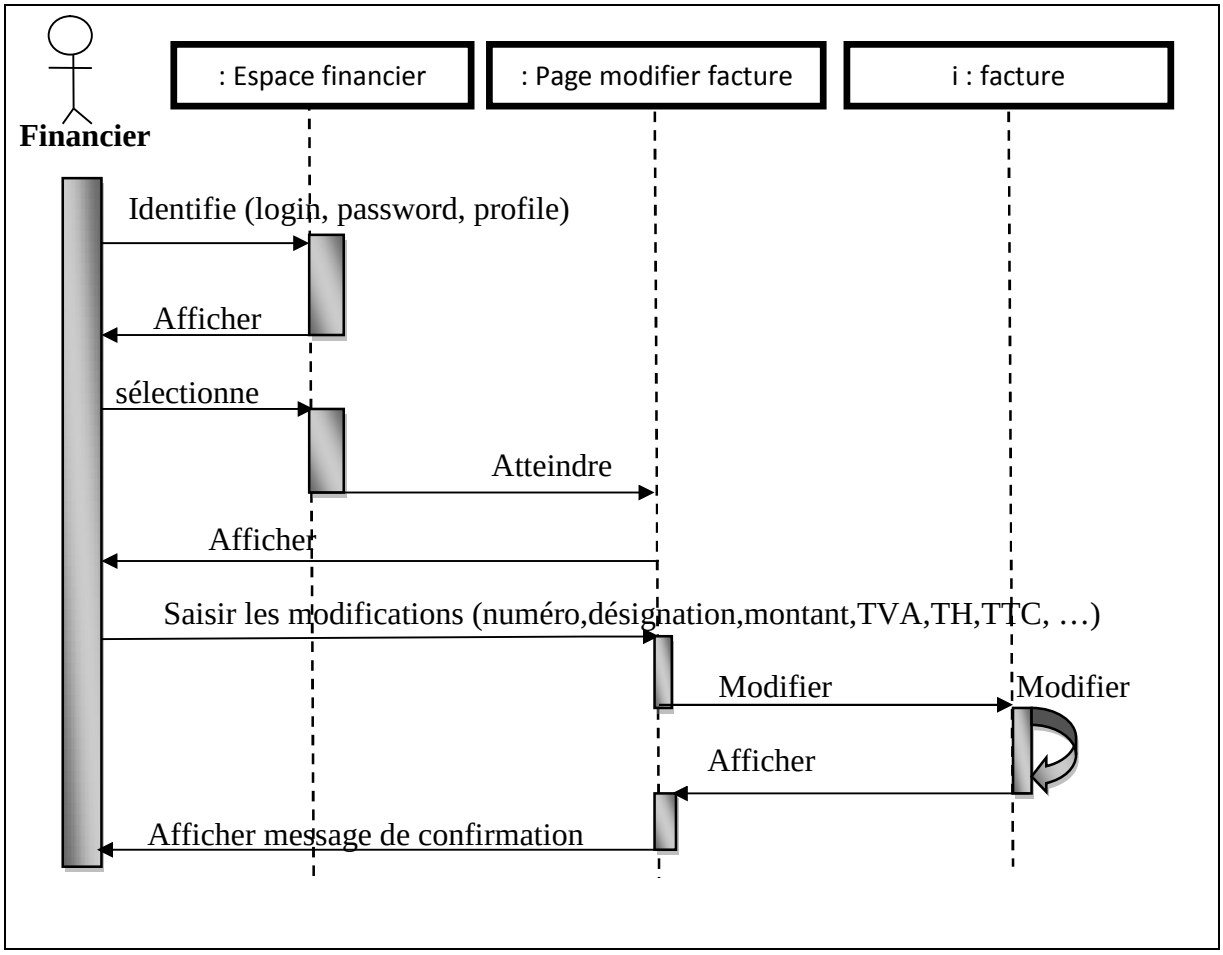



**Figure III.10 : diagramme de séquence associé au scénario**

**« Modifier facture client»**

### III.4.3.2 Diagramme de classe pour l'itération 3 :

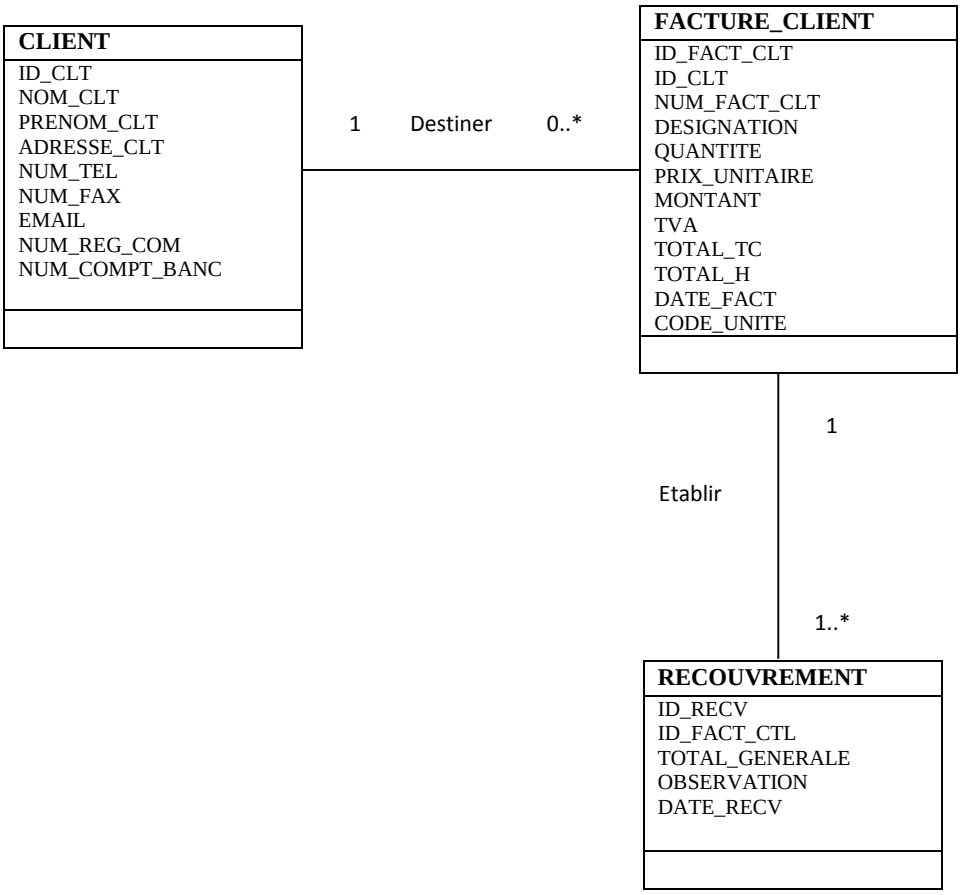



**Figure III.11 : Le diagramme de classe pour l'itération 3**

## Remarque :

La classe Recouvrement est reliée avec la classe Echéancier de l'itération 2 par les cardinalités (1.*….1).
La classe Echéance est reliée avec la classe Mouvement financier de l'itération 2 par les cardinalités (1….1.*).
La classe Mouvement financier est reliée avec la classe unité de l'itération 2 par les cardinalités (1….1.1).

## III.5 Conclusion :

Dans ce chapitre nous avons utilisé la démarche unifiée RUP pour l'analyse et conception de notre application en utilisant le langage de modélisation (UML). Dans le chapitre suivant, nous essayons de présenter l'environnement et les outils de développement utilisées, et nous présenterons quelques interfaces de notre application.

# *Chapitre IV :*

# *Mise en œuvre et Réalisation*

## IV .1 Introduction :

Apres avoir présenté dans le chapitre précédent les différentes étapes d'analyse et conception, nous allons présenter dans ce dernier chapitre, consacré a la réalisation et la mise en œuvre de notre application de gestion payement fournisseur et recouvrement, les outils de développement adopté dans notre application ainsi nous représenteront certains interfaces développées pour illustrer notre application.

## IV.2 Environnement de développement :

### IV.2 .1 La plate forme Java Entreprise Edition :

Java Entreprise Edition est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées. Elle est composée de deux parties essentielles :

- ✓ un ensemble de spécifications pour une infrastructure dans laquelle s'exécutent les composants écrits en Java : un tel environnement se nomme serveur d'applications.
- ✓ un ensemble d'API qui peuvent être obtenues et utilisées séparément. Pour être utilisées, certaines nécessitent une implémentation de la part d'un fournisseur tiers.

Sun propose une implémentation minimale des spécifications de Java Entreprise Edition : le Java EE SDK. Cette implémentation permet de développer des applications respectant les spécifications mais n'est pas prévue pour être utilisée dans un environnement de production. Ces spécifications doivent être respectées par les outils développés par des éditeurs tiers.

- ✓ Java Entreprise Edition 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 Entreprise Edition. 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 standalone, 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
- ✓ L'utilisation de Java Entreprise Edition pour développer et exécuter une application offre plusieurs avantages :
- ✓ une architecture d'applications basée sur les composants qui permet un découpage de l'application et donc une séparation des rôles lors du développement
- ✓ la possibilité de s'interfacer avec le système d'information existant grâce à de nombreuses API : JDBC, JNDI,...
- ✓ la possibilité de choisir les outils de développement et le ou les serveurs d'applications utilisés qu'ils soient commerciaux ou libres

- ✓ **le Framework :**

Le terme Framework est fréquemment utilisé dans des contextes différents mais il peut être traduit par cadre de développement.

Les Framework se présentent sous diverses formes, qui peuvent inclure tout ou partie des éléments suivants :

- ✓ Un ensemble de classes généralement regroupées sous la forme de bibliothèques pour proposer des services plus ou moins sophistiqués.
- ✓ Un cadre de conception reposant sur les design patterns pour proposer tout ou partie d'un squelette d'applications.
- ✓ Des recommandations sur la mise en œuvre et des exemples d'utilisation.
- ✓ Des normes de développement
- ✓ Des outils facilitant la mise en œuvre.

Nous pouvons par exemple citer JSF, Spring, Struts, ou encore Hibernate.

### IV.2.2 Qu'est-ce que le Framework Hibernate :

Hibernate est un Framework open source, léger & ORM, ce dernier à été lancé en 2001 par Gavin King.

Hibernate est un Framework de persistance utilisé pour gérer la persistance des objets java dans une base de données, il peut être utilisé dans un développement web ou bien un développement client lourd.

Hibernate peut utiliser sql comme il peut utiliser son propre langage de requête HQL(Hibernate Query Language).

**ORM(Object relational mapping)** : c'est une méthode de programmation pour mapper les objets.

**HQL(Hibernate Query Language)** : est la version orienté objet du langage sql (on utilise le nom de la classe au lieu du nom de la table).

### IV.2.3.1 Mise en œuvre du Framework Hibernate :

Hibernate a besoin de plusieurs éléments pour fonctionner :

- ✓ Création du projet Java supportant Hibernate sous NetBeans

- ✓ L'ajout du support de Hibernate revient à rajouter un ensemble de Jar du répertoire lib de Hibernate.

- ✓ Une classe du type JavaBean qui encapsule les données d'une table de données nommée « classe de persistance ».
- ✓ Un fichier de correspondance qui configure la correspondance entre la classe et la table.
- ✓ Des propriétés de configuration, notamment des informations concernant la connexion à la base de données.

Une fois ces éléments correctement définis, il est possible d'utiliser Hibernate dans le code des traitements à réaliser.

L'architecture du Framework Hibernate est la suivante :

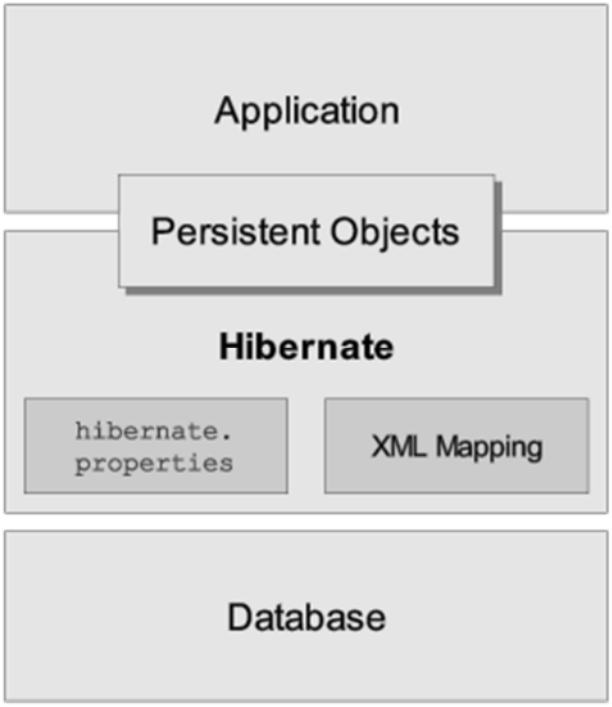



**Figure IV.1 :** L'architecture Hibernate.

## A. Création du fichier de configuration : hibernate.cfg.xml

Hibernate propose des classes qui héritent de la classe Dialect pour chaque base de données supportée. C'est le nom de la classe correspondant à la base de données utilisée qui doit être obligatoirement fourni à la propriété hibernate.dialect.

Les propriétés sont alors définies par un tag <property>. Le nom de la proprieté est fourni grâce a l'attribut « Name » et sa valeur est fournie dans le corps du tag.

Il est possible de fournir des propriétés de configuration « Hibernate » dans un fichier hibernate.propérities.

Le fichier **hibernate.cfg.xml** se présente comme suit :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dt
<hibernate-configuration>
  <session-factory>
    <property name="hibernate.dialect">org.hibernate.dialect.OracleDialect</property>
    <property name="hibernate.connection.driver_class">oracle.jdbc.OracleDriver</property>
    <property name="hibernate.connection.url">jdbc:oracle:thin:@localhost:1521:XE</property>
    <property name="hibernate.connection.username">gestion</property>
    <property name="hibernate.connection.password">gestion</property>
    <property name="hibernate.show_sql"> true </property>


    <!--<property name="hibernate.hbm2ddl.auto"> update </property> -->


    <mapping resource="gestion_pf/Fournisseur.hbm.xml"/>
    <mapping resource="gestion_pf/FactureClient.hbm.xml"/>
    <mapping resource="gestion_pf/FactureFournisseur.hbm.xml"/>
    <mapping resource="gestion_pf/Dette.hbm.xml"/>
    <mapping resource="gestion_pf/Role.hbm.xml"/>
    <mapping resource="gestion_pf/Unite.hbm.xml"/>
    <mapping resource="gestion_pf/MouvementFinancier.hbm.xml"/>
    <mapping resource="gestion_pf/BonLivraison.hbm.xml"/>
    <mapping resource="gestion_pf/BonReception.hbm.xml"/>
    <mapping resource="gestion_pf/Echeancier.hbm.xml"/>
    <mapping resource="gestion_pf/Administrateur.hbm.xml"/>
    <mapping resource="gestion_pf/Client.hbm.xml"/>
    <mapping resource="gestion_pf/Recouvrement.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

**Figure IV.2 :** le fichier hibernate.cfg.xml.

## B. Création des fichiers xml de mapping :

Pour assurer le mapping, Hibernate a besoin d'un fichier de correspondance (mapping file) au format XML qui va contenir des informations sur la correspondance entre la classe définie et la table de la base de données.

Même si cela est possible, il n'est pas recommandé de définir un fichier de mapping pour plusieurs classes. Le plus simple est de définir un fichier de mapping par classe, nommé du nom de la classe suivi par ".hbm.xml". Ce fichier doit être situé dans le même répertoire que la classe correspondante ou dans la même archive pour les applications packagées.

Différents éléments sont précisés dans ce document XML :

- la classe qui va encapsuler les données
- l'identifiant dans la base de données et son mode de génération
- le mapping entre les propriétés de classe et les champs de la base de données
- les relations
- ...

Dans cette partie on va présenter le fichier de mapping relatif à la table « Fournisseur » nommée Fournisseur.hbm.xml.

Les fichiers **xml de mapping** se présentent comme suit :

```xml
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 23 mai 2016 13:42:25 by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
    <class name="gestion_pf.Fournisseur" table="FOURNISSEUR" schema="GESTION" optimistic-lock="version">
        <id name="idFour" type="big_decimal">
            <column name="ID_FOUR" scale="0" />
            <generator class="native">
                <param name="sequence">FOURNISSEUR_SEQ</param>
            </generator>
        </id>
        <property name="nomFour" type="string">
            <column name="NOM_FOUR" length="20" />
        </property>
        <property name="prenomFour" type="string">
            <column name="PRENOM_FOUR" length="20" />
        </property>
        <property name="adresseFour" type="string">
            <column name="ADRESSE_FOUR" length="30" />
        </property>
        <property name="numTelFour" type="big_decimal">
            <column name="NUM_TEL_FOUR" scale="0" />
        </property>
        <property name="numFaxFour" type="big_decimal">
            <column name="NUM_FAX_FOUR" scale="0" />
        </property>
        <property name="emailFour" type="string">
            <column name="EMAIL_FOUR" length="30" />
        </property>
        <property name="numRegCom" type="string">
            <column name="NUM_REG_COM" length="40" />
```

**Figure IV.3** : Les fichiers **xml de mapping.**

Il est indispensable d'ajouter la référence du fichier « Fournisseur.hbm.xml » dans le fichier de configuration « hibernate.cfg.xml » au niveau de la balise <mapping/> comme suit :

**<mapping resource="gestion_pf/Fournisseur.hbm.xml"/>**

### C. Création des classes de données :

Cette classe doit respecter le standard des **Javabeans**, notamment, encapsuler les propriétés dans ses champs **private** avec des getters et setters et avoir un constructeur par défaut.

Les classes de données se présentent comme suit :

```
GESTION_PAYEMENT_FOURNISSEUR - NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Start Page | BeanAuthentification.java | ServiceAuthentification.java | Fournisseur.hbm.xml | BeanFournisseur.java
Source | History
26   */
     @ManagedBean(name="beanfournisseur")
     @SessionScoped
29   public class BeanFournisseur implements Serializable{
30
31       private Fournisseur f;
32       private ServiceFournisseur dao;
33       private List<Fournisseur>   Liste_Fournisseur ;
34
35       public Fournisseur getF() {
36           return f;
37       }
38
39       public void setF(Fournisseur f) {
40           this.f = f;
41       }
42
43       public ServiceFournisseur getDao() {
44           return dao;
45       }
46
47       public void setDao(ServiceFournisseur dao) {
48           this.dao = dao;
49       }
50
51       public List<Fournisseur> getListe_Fournisseur() {
52          this.Liste_Fournisseur=dao.Liste_Fournisseur();
53       return Liste_Fournisseur;
54       }
55
56       public void setListe_Fournisseur(List<Fournisseur> Liste_Fournisseur) {
57           this.Liste_Fournisseur = Liste_Fournisseur;
Notifications                                                    1:1   INS
```

**Figure IV.4 :** JavaBean.

## D. La classe Hibernate Util :

La classe Hibernate nommée SessionFactory permet d'établir la connexion avec la source de données à partir du fichier de configuration « hibernate.cfg.xml ».

On remarque que la classe SessionFactory serait instanciée autant de fois qu'il y a de threads, il est donc plus adapté de rendre une même instance de SessionFactory accessible par les threads. Cette classe possède une méthode appelée currentSession() qui retourne la session hibernate en cours si elle existe sinon elle se charge d'ouvrir une nouvelle session .

La classe HibernateUtil se présentente comme suit :

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package DAO;

import org.hibernate.cfg.AnnotationConfiguration;
import org.hibernate.SessionFactory;

/**
 * Hibernate Utility class with a convenient method to get Session Factory
 * object.
 *
 * @author nawel
 */
public class HibernateUtil {

    private static final SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from standard (hibernate.cfg.xml)
            // config file.
            sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Log the exception.
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }
```

**Figure IV.5 :** La classe HibernateUtil.

### IV.2.3 Le Framework JSF/prime faces :

JSF, alias "**Java Server Faces**" est un framework de type MVC, destiné aux applications web respectant l'architecture Java EE.

Le premier objectif de **Java Server Faces**, est de procurer un environnement de développement permettant de construire une interface de type web, sans devoir toucher au code HTML et JavaScript. Ceci est réalisé par la mise en place d'un mapping entre l'HTML et les objets concernés. JSF est donc basé sur la notion de composants, comparable à celle de Swing, ou l'état de ces composants est sauvegardé puis restauré au retour de la requête.

Le principal atout de JSF est qu'il tient compte des différentes expériences acquises non seulement avec des technologies de type standard comme les Servlets, les JSP,… mais aussi avec les technologies de type framework comme Struts. Dès lors, JSF propose un framework qui puisse être mis en œuvre par des outils facilitant le développement d'applications web. Rien n'empêche cependant que JSF puisse être utilisé par codage à la volée, néanmoins, l'utilisation d'un outil approprié est vivement recommandé afin d'exploiter rapidement les possibilités de JSF.

L'implémentation de JSF nécessite l'implémentation des deux fichiers suivants :

- **Face-config.xml :**

Le **faces-config.xml**, est le fichier de configuration, qui permet de gérer les beans, et les règles de navigation entre les pages. Voila, ci-dessous un bref, exemple de fichier de configuration.

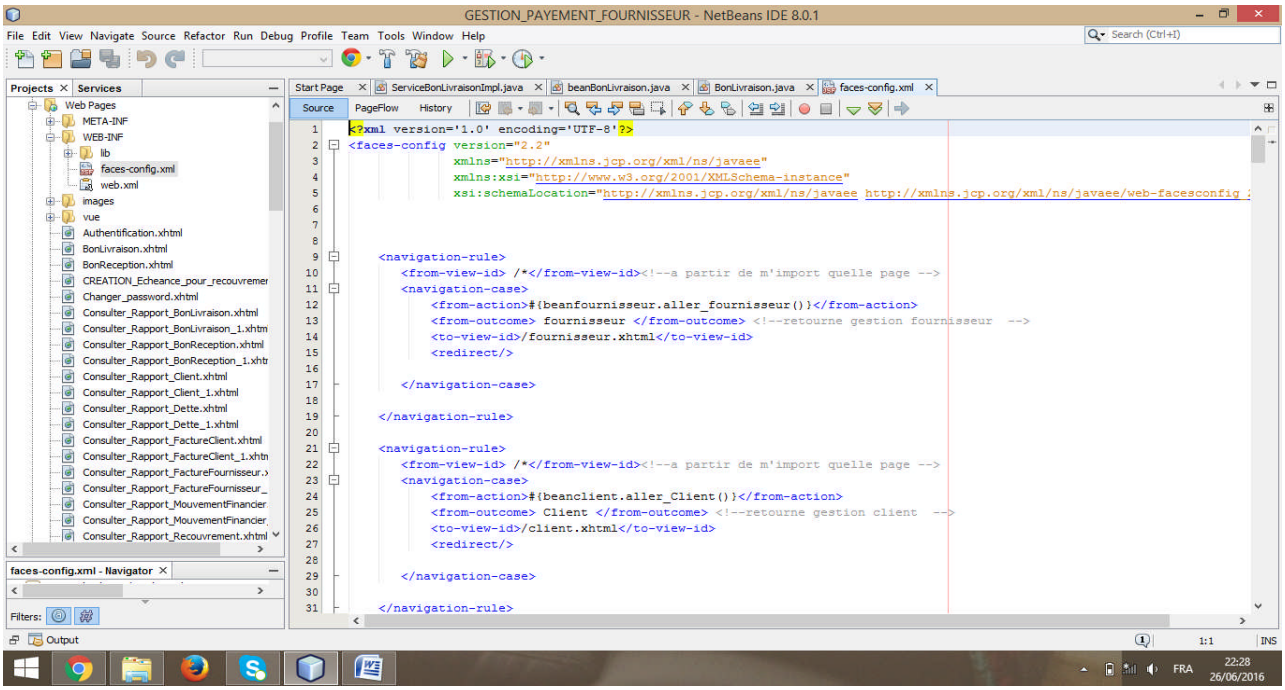



**Figure IV.6:** L'interface du fichier de configuration «faces-config.xml **».**

- **Web.xml : description de déploiement**

Le web.xml qui permet de définir l'emplacement des pages JSP, ainsi que de possible contraintes de sécurité concernant l'accès à des pages. Il établit également l'endroit ou se situe le Faces Servlet. Faces Servlet, est la classe, qui est le moteur de chaque application JSF. Chaque application JSF à sa propre Faces Servlet, qui gère toutes les informations relatives à la requête courante.

- **PrimeFaces :**

PrimeFaces est un framework open source, qui peut être assimilé à une suite de composants d'interfaces riches pour JSF. Ce dernier apporte des éléments très basiques d'UI. PrimeFaces va plus loin et propose plus de 100 composants, plus de 35 thèmes, des composants mobiles et bien plus encore. C'est actuellement de loin la suite pour JSF la plus populaire, ainsi qu'un outil graphique très populaire partout pour le développement de Java Web.

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

### IV.2.4 Architecture de notre application selon le modèle MVC :

Le modèle MVC (Model View Controler) a été initialement développé pour le langage Small talk dans le but de mieux structurer une application avec une interface graphique.

Ce modèle est un concept d'architecture qui propose une séparation en trois entités des données, des traitements et de l'interface :

- ✓ le Modèle représente les données de l'application généralement stockées dans une base de données.
- ✓ la Vue correspond à l'IHM (Interface Homme Machine).
- ✓ le Contrôleur assure les échanges entre la vue et le modèle notamment grâce à des composants métiers.

Initialement utilisé pour le développement des interfaces graphiques, ce modèle peut se transposer pour les applications web sous la forme d'une architecture dite 3-tiers : la vue est mise en oeuvre par des JSP, le contrôleur est mis en oeuvre par des servlets et des Javabeans. Différents mécanismes peuvent être utilisés pour accéder aux données.

L'utilisation du modèle MVC rend un peu plus compliqué le développement de l'application qui le met en œuvre mais il permet une meilleure structuration de celle-ci.

#### IV.2.4.1 Le modèle MVC type1 :

Dans ce modèle, chaque requête est traitée par un contrôleur sous la forme d'une servlet. Celle-ci traite la requête, fait appel aux éléments du model si nécessaire et redirige la requête vers une JSP qui se charge de créer la réponse à l'utilisateur.

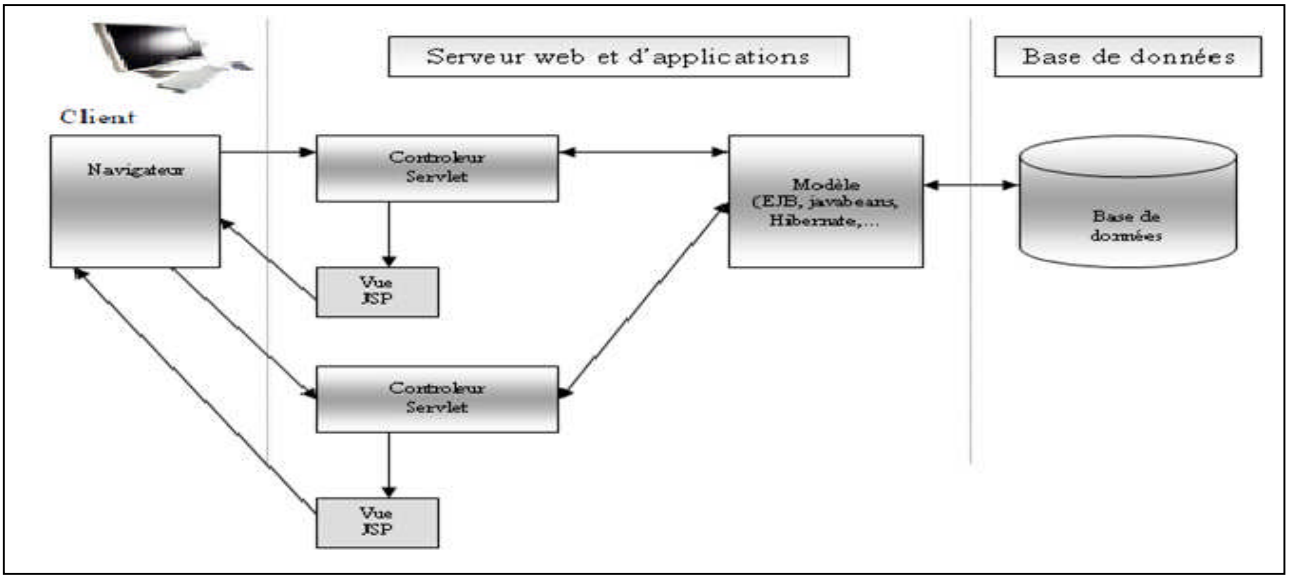



**Figure IV.6 :** Le modèle MVC type 1.

L'inconvénient est donc une multiplication du nombre de servlets nécessaires à l'application : l'implémentation de plusieurs servlets nécessite beaucoup de code à produire d'autant que chaque servlet doit être déclarée dans le fichier web.xml.

### IV.2.4.2 Le modèle MVC type2

Le principal défaut du modèle MVC est le nombre de servlets à développer pour une application.

Pour simplifier les choses, le modèle MVC model 2 ou MVC2 de Sun propose de n'utiliser qu'une seule et unique servlet comme contrôleur. Cette servlet se charge d'assurer le workflow des traitements en fonction des requêtes http reçues.

Le modèle MVC 2 est donc une évolution du modèle 1 : une unique servlet fait office de contrôleur et gère toutes les requêtes à traiter en fonction d'un paramétrage généralement sous la forme d'un fichier au format XML.

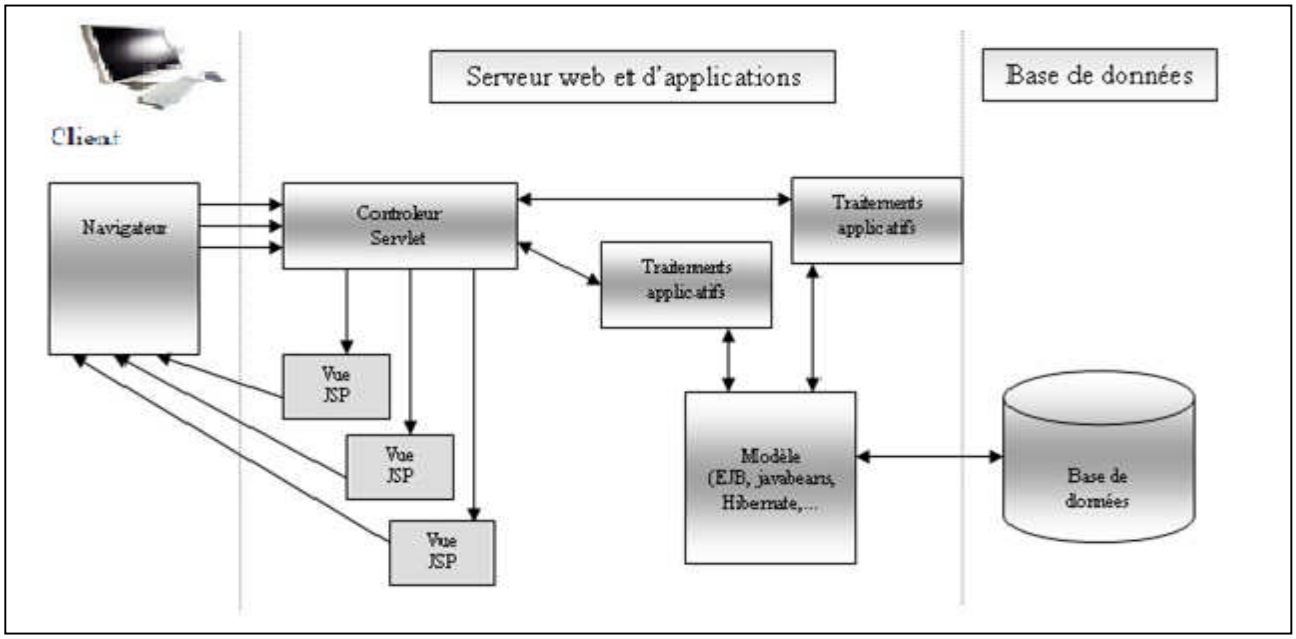



**Figure IV.8 :** Le modèle MVC type 2.

Le modèle MVC II conserve les principes du modèle MVC, mais il divise le contrôleur en deux parties en imposant un point d'entrée unique à toute l'application (première partie du contrôleur) qui déterminera à chaque requête reçue les traitements applicatifs à invoquer dynamiquement (seconde partie du contrôleur).

Une application web implémentant le modèle MVC de type II utilise une servlet comme contrôleur traitant les requêtes. En fonction de celles-ci, elle appelle les traitements dédiés généralement encapsulés dans une classe.

Dans ce modèle, le cycle de vie d'une requête est le suivant :

1. Le client envoie une requête à l'application, requête est prise en charge par la servlet faisant office de contrôleur.
2. La servlet analyse la requête et appelle la classe dédiée contenant les traitements
3. Cette classe exécute les traitements nécessaires en fonction de la requête, notamment, en faisant appel aux objets métiers.
4. En fonction du résultat, la servlet redirige la requête vers la page JSP
5. La JSP génère la réponse qui est renvoyée au client.

La figure suivante montre l'architecture de notre application suivant le modèle MVC :

# Modèle - Vue – Contrôleur

| | | |
|---|---|---|
| **JSF (java server face ) :** Implémentation : framework primfaces.<br><br>**(Vue)** | Implémentation par l'utilisation des interfaces ( **objetservice, objettImp , javabean )**<br><br>**(Contrôle** ) | **Hibernet**<br><br>« persistances des données »<br><br>**(Model**) |
| 1. Un ensemble de composant réutilisable ; qui communique avec les **javabean.**<br><br>2. Descripteur de déploiement : **(Web.XmI).**<br><br>3. Fichier de configuration : (**faces – config** ) | 1. Pour chaque segment de la réalité étudier en défini un service et son implémentation, ces derniers nous ont permet de répondre aux règles de gestion du champ d'étude.<br><br>**2. javabean :** les objets qui font le lien entre la partie métier et l'interfaces graphique. | 1. **Fichier de configuration Hibernet** : hebernet .cfg.xml(configuration générale – accès a la base de donnée )<br><br>2**. Fichier de mapping** : ensemble de fichier xml- lien entre les fichiers java et les tables dans la bdd<br><br>**3. Fichier util** : génération de session pour l'exploitation du Framework Hibernet et matérialisation des accès à la base de donnés . |

**Figure IV .9 :** L'architecture de notre application suivant le modèle MVC.

## IV.3 Outils de développement :

### IV.3.1 Logiciels :

#### IV.3.1.1 Environnement de développement intégré « NetBeans » :

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

Conçu en Java, NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java).

L'environnement de base comprend les fonctions générales suivantes:

- ✓ configuration et gestion de l'interface graphique des utilisateurs,
- ✓ support de différents langages de programmation,
- ✓ traitement du code source (édition, navigation, formatage, inspection..),
- ✓ fonctions d'import/export depuis et vers d'autres IDE, tels qu'Eclipse ou JBuilder,
- ✓ accès et gestion de bases de données, serveurs Web, ressources partagées,
- ✓ gestion de tâches (à faire, suivi ...),
- ✓ documentation intégrée.

L'éditeur intégré propose des fonctions de complétion, de contrôles syntaxiques et sémantiques, d'avertissements et de conseils, de reprise de codes ("refactoring": renommage, changement des méthodes, gestion des classes, ...), de sauvegarde et reprise.

NetBeans comprend un explorateur de bases de données qui supporte toutes les bases relationnelles pour lesquelles un connecteur JDBC existe (selon les versions des gestionnaires de bases de données): JavaDB (Derby) MySQL, PostgreSQL, Oracle, Microsoft SQL, PointBase, jTDS, IBM Redistributable DB2, ...

L'explorateur comprend un éditeur de requêtes, un gestionnaire intégré de bases de données MySQL.

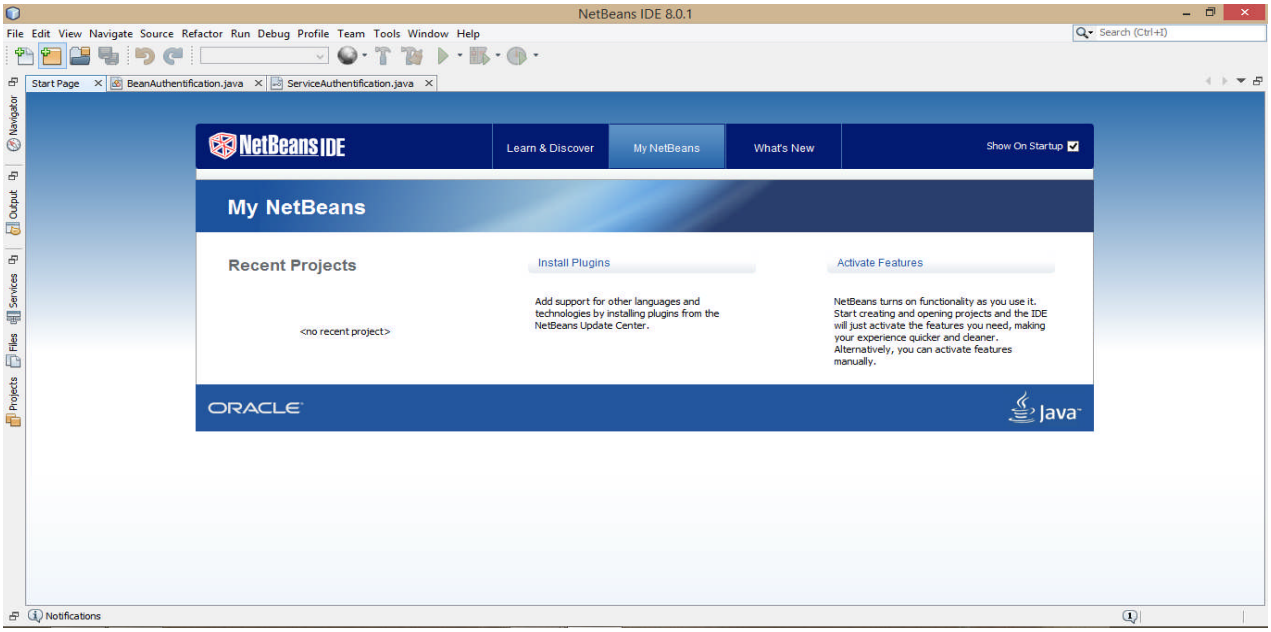



**Figure IV.10 :** Interface de l'IDE NetBeans.

### IV.3.1.2 Le serveur d'application :

- **Apache Tomcat :**

Pour faire fonctionner une application web Java Entreprise Edition, nous avons besoin de mettre en place un serveur d'application. Il en existe sur le marché, on a choisi d'utiliser Tomcat, car c'est un serveur léger, gratuit, libre, multiplateforme et assez complet pour ce que nous allons développer. On le rencontre d'ailleurs très souvent dans des projets d'entreprise, en phase de développement comme en production.

- **L'interface Oracle Express Edition 10g (Oracle EX) :**

Oracle 10g est un puissant Système de Gestion de Bases de Données Relationnelles proposant, en plus du moteur de la base, de nombreux outils à l'utilisateur, au développeur et à l'administrateur. Ces outils ont un langage commun : le SQL.

Oracle permet de gérer les données d'une application en respectant une logique, devenue standard, le modèle relationnel. Les fondements de ce modèle ont été établis au début des années 70 et restent une référence pour la gestion des données.

Parmi les versions d'oracle 10g : Oracle Express Edition, Oracle Database Standard Edition et Oracle Database Enterprise nous avons choisi oracle Expression Edition à cause de sa gratuité et sa rapidité d'installation (mois de 15 minutes).

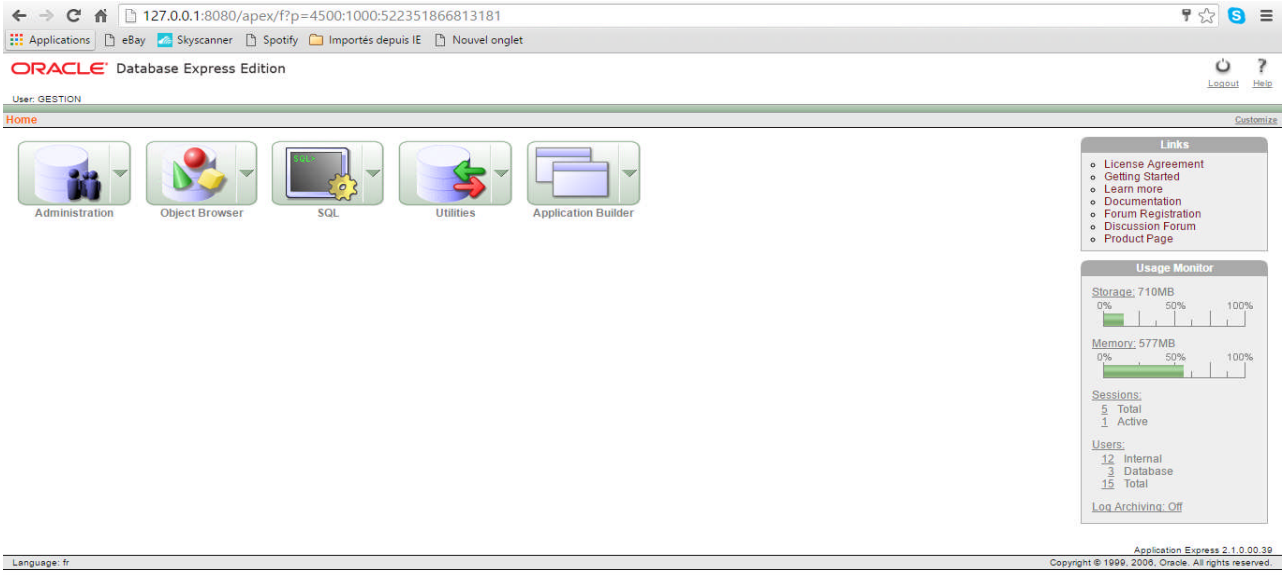



**Figure IV.11:** L'interface Oracle Express Edition.

- **Construction de la base de données selon les itérations :**

✓ **Itération 1 :** Voila son modèle relationnel :

**Administrateur (**Id_admin, Login, Password**) ;**

**Role (**id_role, Id_admin *, profile, nom, prenom**) ;**

Voila la table de **l'itération 1 :**

✓ **Table Administrateur :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_admin | Number | 10 | Identificateur de l'utilisateur |
| Login | Varchar | 20 | login de l'utilisateur |
| Password | Varchar | 20 | Mot de passe de l'utilisateur |

✓ **Table role :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_role | Number | 10 | Identificateur de rôle |
| Id_admin | Number | 10 | Identificateur administrateur |
| Profile | Varchar | 20 | Profile |
| Nom | Varchar | 20 | Nom utilisateur |
| Prenom | Varchar | 20 | Prénom utilisateur |

- ✓ **Itération 2 :** Voila son modèle relationnel :

**Fournisseur(**Id_four, Nom_four, Prenom_four, Adresse_four, Num_tel_four, Email_four, Num_reg_com, Matricule_fiscale, Num_compt_bonc, Type_four, Nom_banc **)** ;

**Bon_livraison (**Id_bl, Id_four*, Designation , Num_bl, Quantite_commander, Quantite_livrer, Quantite_restante, Date_Livraison , Prix_unitaire, Montant**)**;

**Bon_reception (**Id_br, Id_bl*,Num_br, Date_reception**)** ;

**Facture_Fournisseur(**Id_fact_four**,** Id_br*, Id_four* ,Num_fact_four, Tva, Total_tc, Total_h, Date_fact, code_unite**)**;

**Dette (**Id_dette , Id_fact_four*, Observation, Date_Dette**)**;

**Echeance(**Id_echeance, Id_dette*, Id_recv*, Type_payement, Nbr_cycle_payement, Cycle de payement, Montant_cycle, Date_cycle, Num_cheque**)**;

**Mouvment_financier (**Id_mvt_fin, Id_fact_clt*, Id_fact_four*, Id_echeance*, Id_unite*, Id_echeance*, Date_payement, Type_payement**)**;

**Unite(**id_unite, Code_unite, Libelle_unite, Solde**)**;

Voila les tables de **l'itération 2 :**

- ✓ **Table Fournisseur :**

| **Nom du champ** | **Type de données** | **Taille** | **Description** |
|---|---|---|---|
| Id_four | Number | 10 | Identificateur du fournisseur |
| Nom_four | Varchar | 20 | nom du fournisseur |
| Prenom_four | Varchar | 20 | prénom du fournisseur |
| Adresse_four | Varchar | 40 | adresse du fournisseur |
| Num_tel_four | Number | 13 | Numéro du téléphone du fournisseur |
| Email_four | Varchar | 40 | email du fournisseur |
| Num_fax_four | Number | 19 | Numéro fax d fournisseur |
| Num_reg_com | Number | 15 | Numéro du registre de commerce |
| Matricule_fiscale | Varchar | 40 | matricule fiscale du fournisseur |
| Num_compt_bonc | Number | 10 | Numéro compte bancaire |
| Type_four | Varchar | 15 | Type du fournisseur |
| Nom_banc | Varchar | 20 | Nom de la bonc |

- ✓ **Table Bon_ livraison :**

| **Nom du champ** | **Type de données** | **Taille** | **Description** |
|---|---|---|---|
| Id_bl | Number | 10 | Identificateur du bon de livraison |
| Id_four | Number | 10 | Identificateur du fournisseur |
| Designation | Varchar | 20 | Designation |
| Num_bl | Number | 19 | Numéro du bon de livraison |

| | | | |
|---|---|---|---|
| Quantite_commander | Number | 10 | Quantité commander |
| Quantite_livrer | Number | 10 | Quantité livrer |
| Quantite_restante | Number | 10 | Quantité restante |
| Date_Livraison | Datte | / | Date de livraison |
| Prix_unitaire | Number | 10 | Pris unitaire |
| Montant | Number | 10 | Montant |

- ✓ **Table Bon_réception :**

| **Nom du champ** | **Type de données** | **Taille** | **Description** |
|---|---|---|---|
| Id_br | Number | 10 | Identificateur du bon de réception |
| Id_bl | Number | 10 | Identificateur du bon de livraison |
| Num_br | Number | 19 | Numéro |
| Date_reception | Datte | / | Date de recéption |

- ✓ **Table Facture_Fournisseur :**

| **Nom du champ** | **Type de données** | **Taille** | **Description** |
|---|---|---|---|
| Id_fact_four | Number | 10 | Identificateur de la facture fournisseur |
| Id_br | Number | 10 | Identificateur du bon de réception |
| Id_four | Number | 10 | Identificateur du fournisseur |
| Num_fact_four | Number | 19 | Numéro de la facture fournisseur |
| Tva | Number | 10 | Taux de valeur ajouté |
| Total_tc | Number | 10 | Toux Total complet |
| Total_h | Number | 10 | Total hors taxe |
| Date_fact | Date | / | Date d'enregistrement de la facture du fournisseur |
| Code_unite | 10 | Number | Code de l'unité |

- ✓ **Table Dette :**

| **Nom du champ** | **Type de données** | **Taille** | **Description** |
|---|---|---|---|
| Id_dette | Number | 10 | Identificateur de la dette |
| Id_fact_four | Number | 10 | Identificateur de la facture fournisseur |
| Observation | Varchar | 40 | Observation |
| Date_Dette | Datte | / | Date d'enregistrement de la dette |

- ✓ **Table Echéance :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_echeance | Number | 10 | Identificateur de l'échéances |
| Id_dette | Number | 10 | Identificateur de la dette |
| Id_recv | Number | 10 | Identificateur du recouvrement |
| Type_payement | Varchar | 20 | Type de payement |
| Nbr_cycle_payement | Number | 40 | Nombre de Cycle de payement |
| Cycle de payement | Number | 13 | Cycle de payement |
| Montant_cycle | Number | 40 | Montant cycle |
| Date_cycle | Date | 19 | Date cycle |
| Num_cheque | Number | 15 | Numéro du chèque |

- ✓ **Table Mouvment_financier :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_mvt_fin | Number | 10 | Identificateur du mouvement financier |
| Id_fact_clt | Number | 10 | Identificateur du la facture client |
| Id_fact_four | Number | 10 | Identificateur de la facture fournisseur |
| Id_echeance | Number | 10 | Identificateur de l'échéance |
| Id_unite | Number | 10 | Identificateur du l'unité |
| Date_payement | Datte | / | Date de payement |
| Type_payement | Varchar | 15 | Type payement |

- ✓ **Table Unite :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_unite | Number | 10 | Identificateur du l'unité |
| Solde | Number | 10 | Solde |
| Code_unite | Number | 5 | code de l'unité |
| Libelle_unite | varcher | 40 | libellé de l'unité |

- ✓ **Itération 3 :** Voila son modèle relationnel :

**Client (**Id_clt, Nom_clt, Prenom_clt, Adresse_clt, Num_tel, Num_fax, Email, Num_reg_com, Num_compt_bonc**)** ;

**Facture_client(**Id_fact_clt, Id_clt*, Num_fact_clt, Designation, Quantite, Prix_Unitaire, Montant, Tva, Total_tc, Total_h, Date_fact,code_unite**)** ;

**Recouvrement (**Id_recv, Id_fact_clt*, Observation, Date_recv**)**;

Voila les tables de **l'itération 3 :**

- ✓ **Table Client :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_clt | Number | 10 | Identificateur du client |
| Nom_clt | Varchar | 20 | Nom client |
| Prenom_clt | Varchar | 20 | Prénom du client |
| Adresse_clt | Varchar | 40 | Adresse du client |
| Num_tel | Numbr | 13 | Numéro de téléphone du client |
| Num_fax | Numbr | 19 | Numéro fax du client |
| Email | Varchar | 40 | Email du client |
| Num_reg_com | Number | 15 | Numéro registre de commerce |
| Num_compt_bonc | Number | 10 | Numéro compte bancaire |
| Nom_banc | Varchar | 20 | Nom de la banque |

- ✓ **Table Facture_client :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_fact_clt | Number | 10 | Identificateur du la facture client |
| Id_clt | Number | 10 | Identificateur du client |
| Num_fact_clt | Number | 19 | Numéro de la facture client |
| Désignation | Varchar | 20 | Désignation |
| Quantite | Number | 10 | Quantité |
| Prix_Unitaire | Number | 10 | Prix unitaire |
| Montant | Number | 10 | Montant |
| Tva | Number | 10 | Taux de valeur ajouté |
| Total_tc | Number | 10 | Toux Total complet |
| Total_h | Number | 10 | Total hors taxe |
| Date_fact | Date | 10 | Date d'enregistrement de la facture |
| Code_unite | Number | 10 | Code de l'unité |

- ✓ **Table Recouvrement :**

| Nom du champ | Type de données | Taille | Description |
|---|---|---|---|
| Id_recv | Number | 10 | Identificateur du recouvrement |
| Id_fact_clt | Number | 10 | Identificateur du la facture client |
| Observation | Varchar | 40 | Observation |
| Date_recv | Datte | / | Date d'enregistrement de recouvrement |

## IV.4 Langage de programmation :

Pour satisfaire les besoins de l'application a réaliser, nous avons utilisé les langages suivants :

### IV.4.1 Le langage coté serveur :

- ✓ Le langage orienté objet **JAVA** pour la création des différentes classes java.

- ✓ **HQL (Hibernate Query Language)** : Hibernate Query Langage est un langage d'interrogation des classes persistantes avec une syntaxe proche du SQL. Cependant les objets manipulés sont les classes et membres du mapping contrairement au SQL qui réalise les requêtes directement sur la base de données. Aussi, le HQL contient des fonctionnalités spécifiques au modèle objet.

### IV.4.2 Le langage coté client :

- ✓ **Le langage XHTML (Extensible Hypertext Markup Language)** : est un langage de balisage servant à écrire des pages pour le World Wide Web. Conçu à l'origine comme le successeur de HTML, XHTML se fonde sur la syntaxe définie par XML, plus récente, mais plus simple que celle définie par SGML sur laquelle repose HTML. Il s'agissait en effet à l'époque de tirer parti des bénéfices techniques attendus de la simplification offerte par XML.

- ✓ **XML (Extensible Markup Language) :** Pour la creation de fichiers mapping relationnel-objet et pour la déclaration.

**XML** (entendez eXtensible Markup Language et traduisez Langage à balises étendu, ou Langage à balises extensible) est en quelque sorte un langage HTML amélioré permettant de définir de nouvelles balises. Il s'agit effectivement d'un langage permettant de mettre en forme des documents grâce à des balises (markup).

## IV.5 Présentation des graphiques de l'application :

Dans cette partie de ce chapitre on va donner un aperçu de l'interface homme/machine de notre application

- **L'interface de l'authentification :**

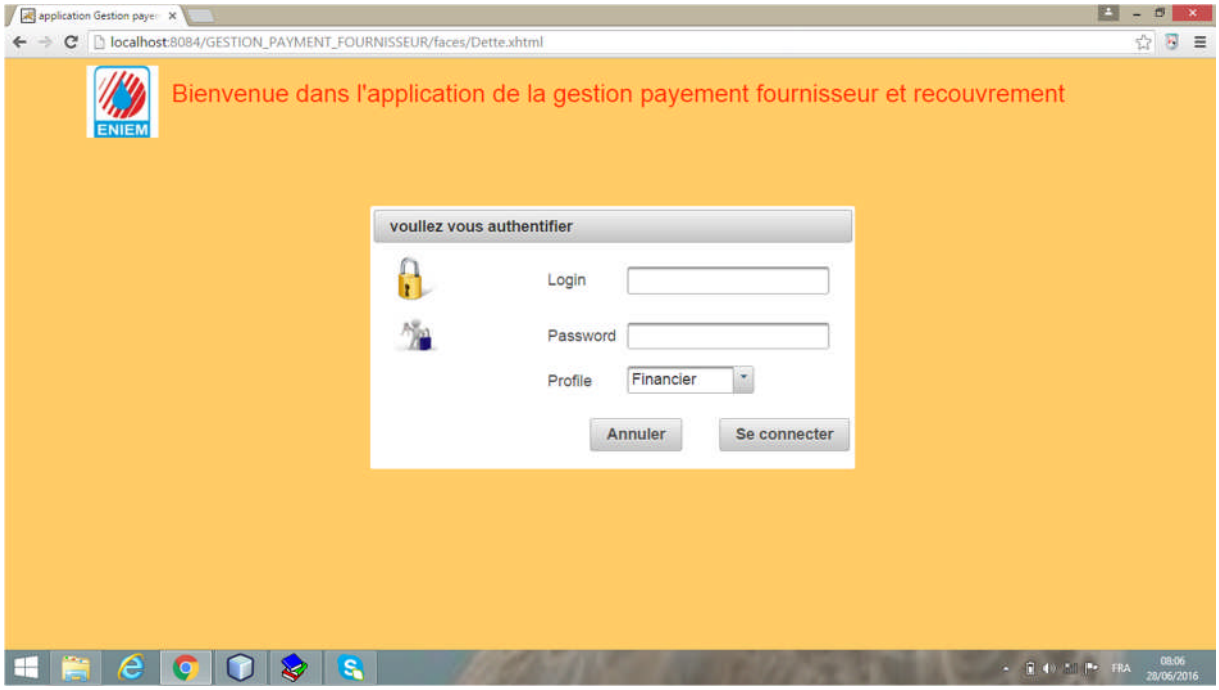



**Figure IV.12 :** L'interface d'authentification.

- **L'interface d'accueil pour l'administrateur :**

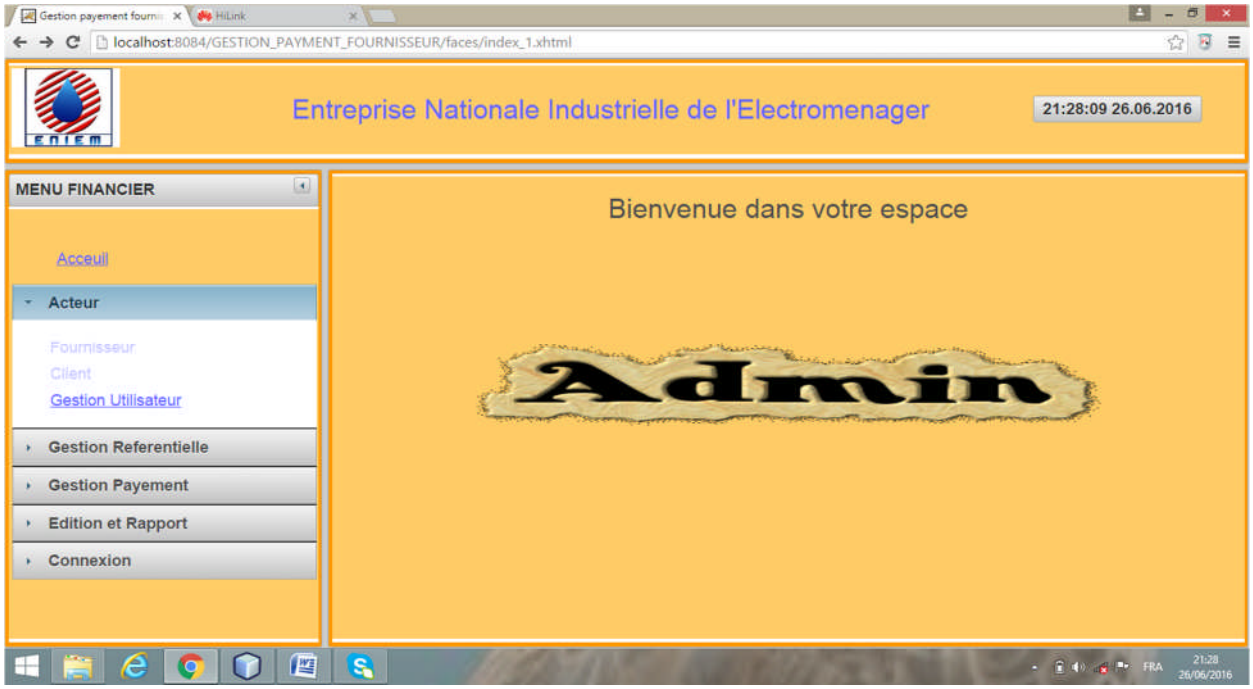



**Figure IV.13 :** L'interface d'accueil pour l'administrateur

- **L'interface d'accueil pour le directeur :**

**Figure IV.14 :** L'interface d'accueil pour le directeur

- **L'interface d'accueil pour le financier :**

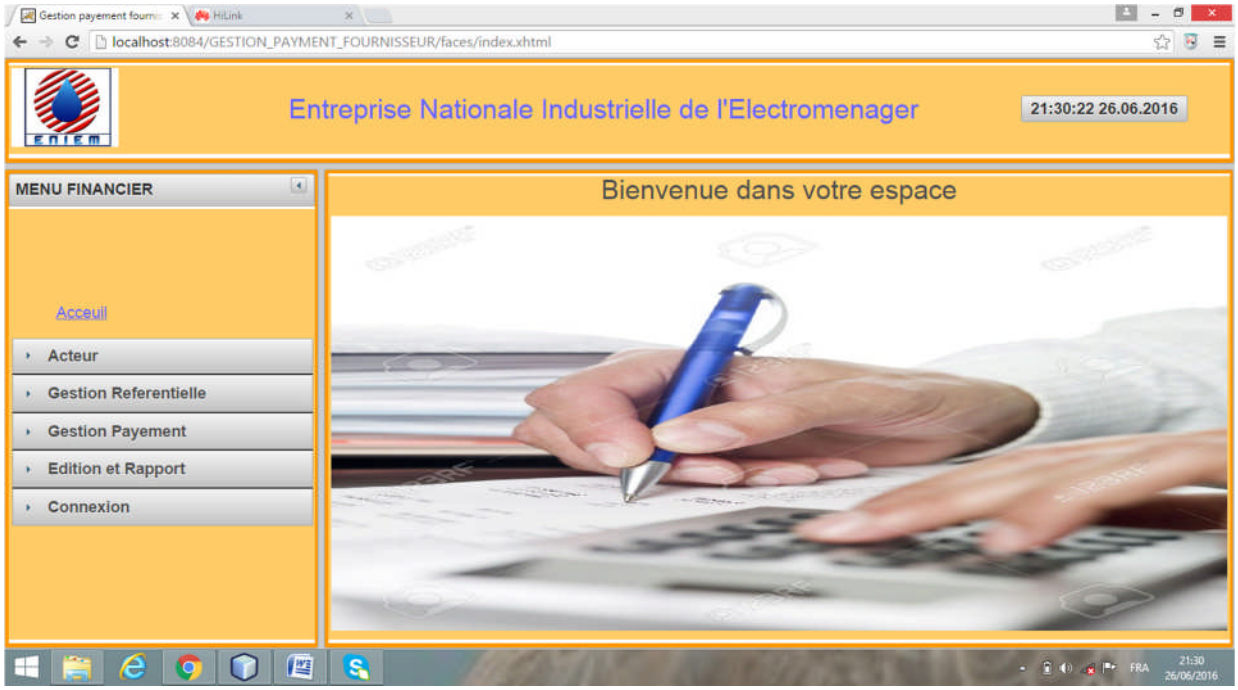



**Figure IV.15 :** L'interface d'accueil pour le financier.

- **L'interface de création d'un nouvel utilisateur :**

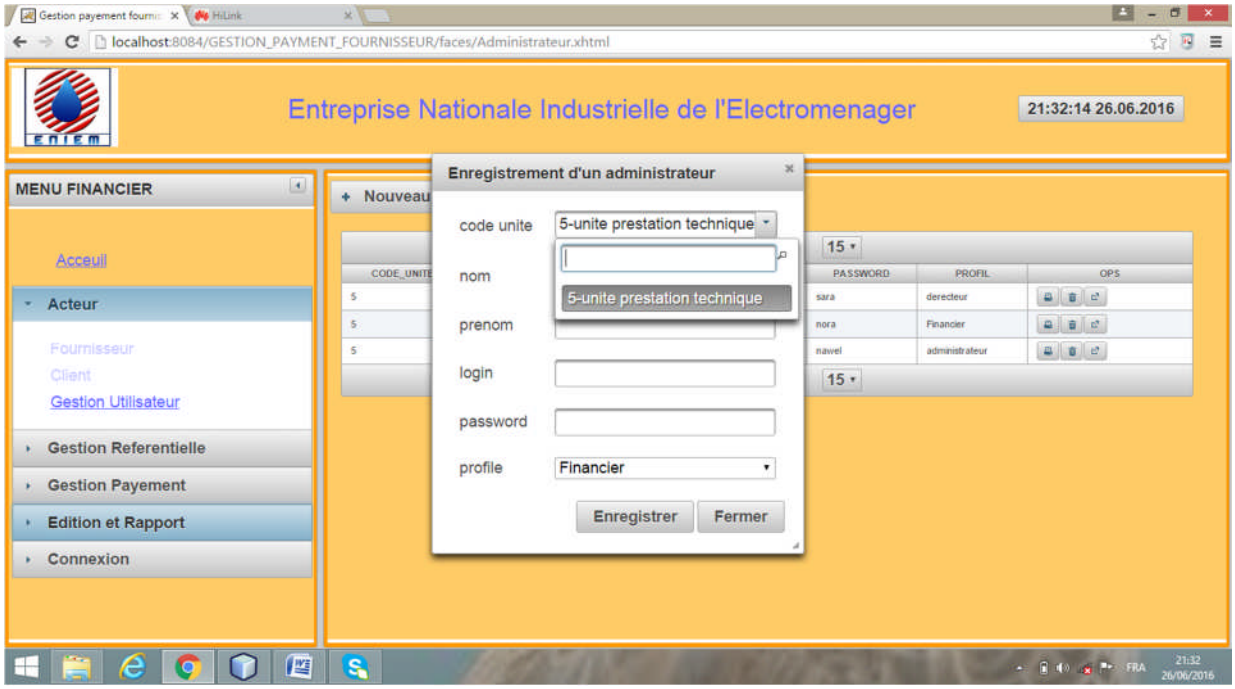



**Figure IV.16 :** L'interface de création d'un nouvel utilisateur.

- **L'interface de gestion de facture fournisseur :**

**Figure IV.17 :** L'interface de gestion des factures fournisseur.

- **L'interface de modification de la facture fournisseur :**

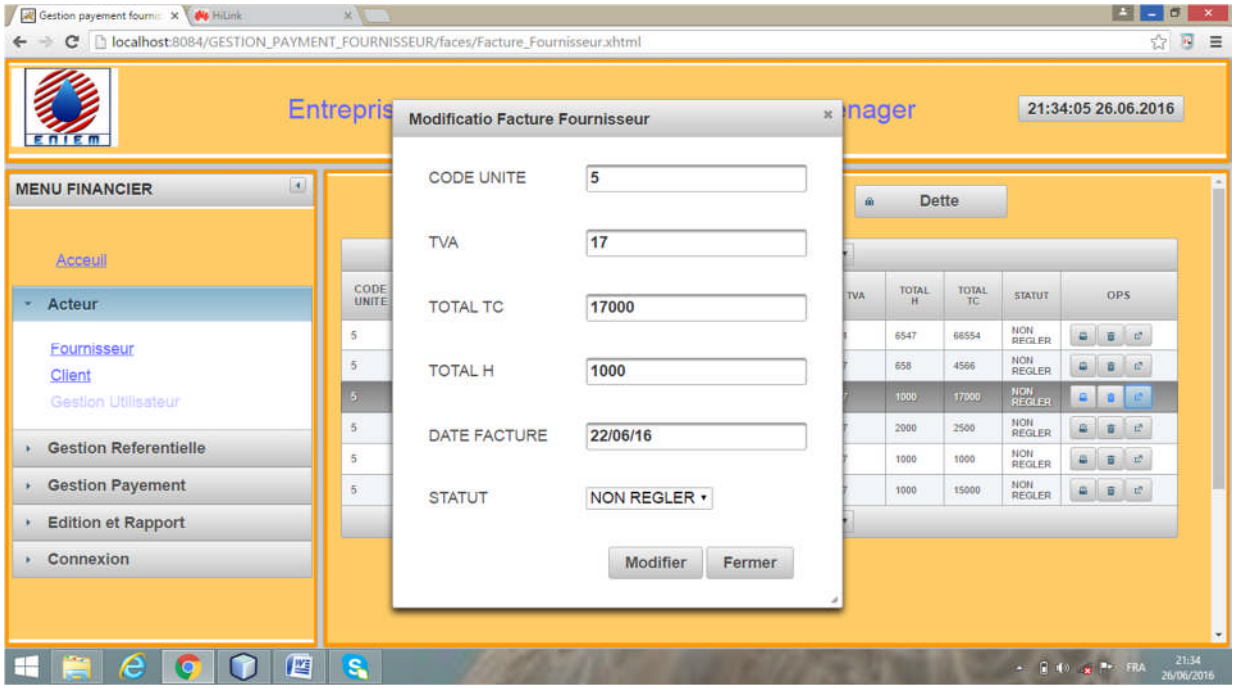



**Figure IV.18 :** L'interface de modification de la facture fournisseur.

- **L'interface de règlement de la facture client :**

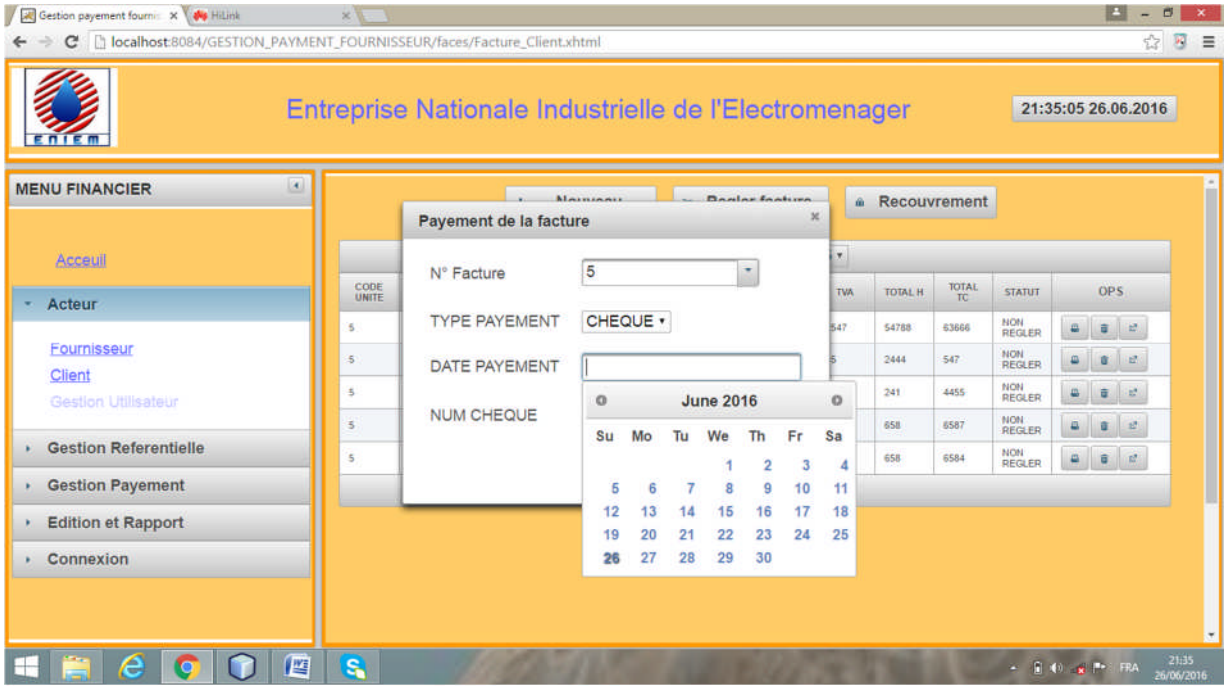



**Figure IV.19 :** L'interface de règlement de la facture client.

- **L'interface consultation de la dette :**

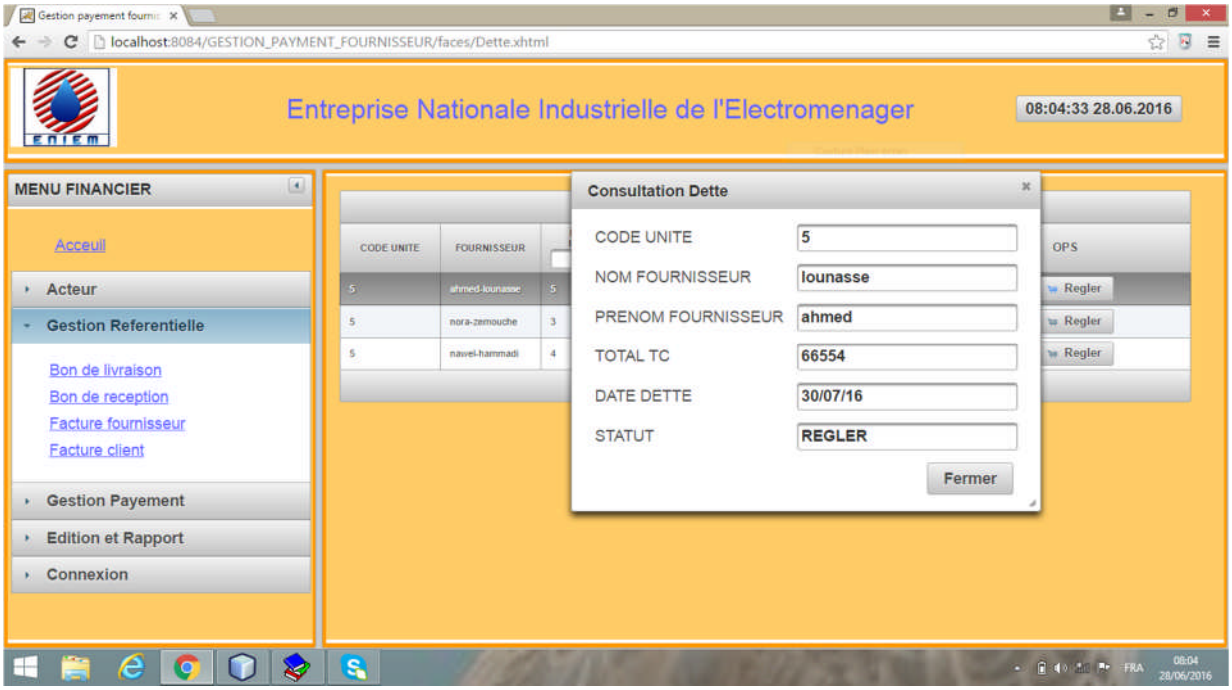



**Figure IV.20 :** L'interface de consultation de la dette.

## IV.6 Conclusion :

Dans ce dernier chapitre nous avons présenté l'étape de réalisation de notre application ainsi, nous avons pu avoir des exigences technique du système avant de présenter les différents outils de développement et les langages de programmations utilisés pour le développement dans notre application. Par la suit, nous avons donné un aperçu de l'application réaliser.

# *Conclusion générale*

Notre étude nous a permet d'analyser et de concevoir une application web pour la gestion payement fournisseur et recouvrement.

Sur le plan technique nous avons découvert le monde du développement web et nous avons aussi appris les bases de la plate forme Java Entreprise Edition, notamment l'architecture MVC.

Ce travail qu'on a fait nous a donné l'occasion d'acquérir de nouvelle connaissance et d'en approfondir d'autre sur le développement web tel que JSF (java server face) ,on cite le langage de programmation JAVA précisément sous la plate forme Java Entreprise Edition et avec le langage de modélisation UML et aussi de métriser la démarche de développement logiciel RUP.

On a appris aussi de s'adapter avec un certains nombre d'outils informatique de développement tel que NetBeans, Tomcat, Oracle XE (Express Edition).

Notre application a abouti à un logiciel fonctionnel qui répond aux critères imposé dans ce domaine. Néanmoins, quoiqu'elle présente quelques améliorations etextensions peuvent être apportées à notre application.

# *Annexe*

## Présentation des diagrammes UML :

Un modèle est une représentation simplifiée d'un problème. UML permet d'exprimer les modèles objets à travers un ensemble de diagrammes. Ces derniers sont des moyens de description des objets ainsi que des liens qui les relient.

Un diagramme est une représentation graphique qui s'intéresse à un aspect précis du modèle. UML offre 13 types de diagrammes. Chaque type de diagramme offre une vue d'un système. Combinés, les différents types de diagrammes offrent une vue complète du système.

Les digrammes UML peuvent êtres classés sous deux grandes catégories :

- Les diagrammes de structure ou statiques,
- Les diagrammes de comportement.

## II.1. Les diagrammes de structure ou statiques : qui sont au nombre de six :

- ✓ **Les diagrammes de classes** sont sans doute les diagrammes les plus utilisés d'UML. Ils décrivent les types des objets qui composent un système et les différents types de relations statiques qui existent entre eux.

Les diagrammes de classes font abstraction du comportement du système.

- ✓ **Les diagrammes d'objets :** représente un instantané des objets d'un système à un moment donné. Ces diagrammes sont souvent appelés diagrammes d'instances car ils représentent des instances et non pas des classes.

- ✓ **Les diagrammes de composants :** Décrivent l'architecture interne d'une classe. Ils sont très utilisés pour représenter l'architecture physique et statique d'une application en termes de module (fichiers, sources, librairies, exécutables,…) pour montrer la mise en œuvre physique des modèles statiques avec l'environnement de développement.

- ✓ **Les diagrammes de déploiement :** représentent l'agencement physique d'un système montrant sur quels composants matériels les différents composants logiciels s'exécutent. Les principaux éléments sont des nœuds connectés par des voies de communication. Un nœud représentera une unité qui va héberger un logiciel, un équipement matériel ou ordinateur, un environnement d'exécution, un système d'exploitation,…

Les nœuds contiennent des artefacts (implémentation d'un composant) tels que les fichiers (.exe, .dll, scripts, ….).

- ✓ **Les diagrammes de structure composite :** font parties des nouveautés ramenées par UML 2.0. Ils permettent de décomposer hiérarchiquement une classe en une structure interne.
- ✓ **Les diagrammes de package :** permettent de grouper des éléments UML dans des unités de plus haut niveau. En effet, ils peuvent servir à organiser des éléments UML

selon leurs types, leurs fonctionnalités ou leurs architectures. Ils sont plus souvent utilisés pour grouper des classes.

### II.1.1. Les diagrammes de comportement : qui sont au nombre de sept :

- ✓ **Les diagrammes d'activités :** décrivent le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchainements d'activités. Une activité désigne une suite d'actions. Le passage d'une action vers une autre est matérialisé par une transition. Les transitions sont déclenchées par la fin d'une action et provoquent le début immédiat d'une autre (elles sont automatiques).

- ✓ **Les diagrammes de cas d'utilisation (vue fonctionnelle ):** les cas d'utilisation (en anglais use cases) ont été introduits par Ivar jacobson dans sa méthode OOSE (Oriented Object Software Engeneering). Ils constituent une technique qui permet de déterminer les besoins des utilisateurs et de capturer les exigences fonctionnelles d'un système. En d'autres termes, ils décrivent le comportement d'un système du point de vue de ses utilisateurs. Ils décrivent les interactions entre les utilisateurs d'un système et le système lui-même. Et un diagramme de cas d'utilisation permet de représenter graphiquement les cas d'utilisation. Le fait qu'un acteur déclenche un cas d'utilisation est représenté par une flèche entre ces deux derniers.

- ✓ **Les diagrammes de machine d'état :** appelés diagrammes d'état- transitions dans UML 1.x, permet de décrire le comportement d'un objet durant son cycle de vie.

Ils permettent plus précisément de décrire les changements d'états d'un objet, en réponse aux interactions avec d'autres objets ou acteurs.

- ✓ **Les diagrammes de séquence (vue fonctinnelle) :** permettent de représenter les interactions entre objets selon un point de vue temporel. L'accent est mis sur la chronologie des envois de messages.
- ✓ **Les diagrammes de communication :** appelés diagrammes de collaboration dans UML 1.x, ils décrivent l'interaction en mettant l'accent sur les liaisons des données les différents participants à l'interaction.
- ✓ **Les diagrammes de vue d'ensemble des interactions :** donnent uns vue d'ensemble des interactions en combinant des diagrammes d'activités et des diagrammes de séquence. En mettant l'accent sur les liaisons des données entre les différents participants à l'interaction.
- ✓ **Les diagrammes de timing :** permettant de définir des contraintes temporelles pour un ou plusieurs objets.

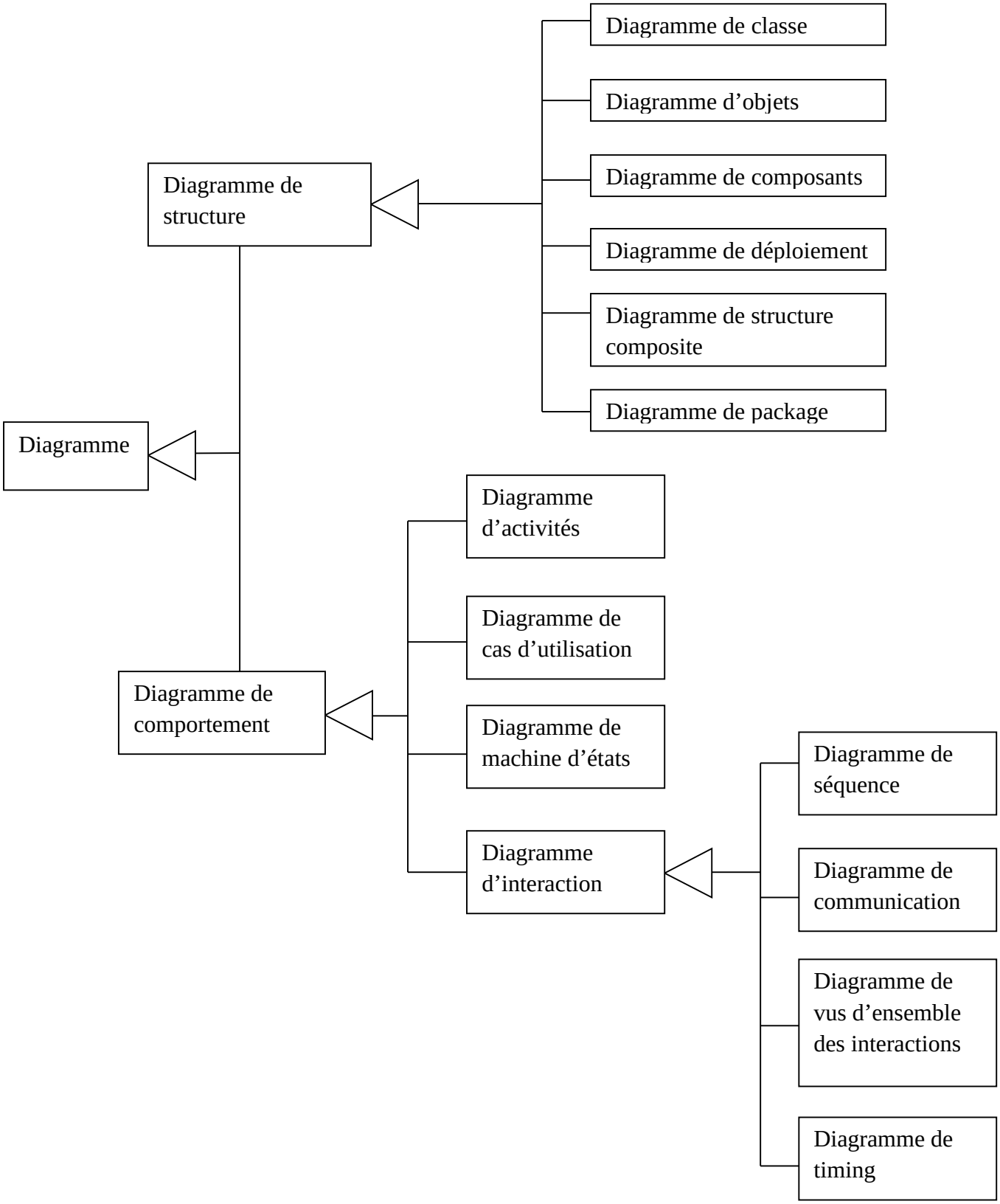



**Figure :** Classification des diagrammes UML

## I. Introduction :

**Processus unifié (UP)** est un processus de développement logiciel : il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

Caractéristiques essentielles du processus unifié :

Le processus unifié est à base de composants,

- ✓ Le processus unifié utilise le langage UML (ensemble d'outils et de diagramme).
- ✓ Le processus unifié est piloté par les cas d'utilisation.
- ✓ Centré sur l'architecture.
- ✓ Itératif et incrémental.

## II. RUP (Rationnal Unified Process) :

- ✓ Dériver de UP (UP a été crée en 1996)
- ✓ Commercialisée par IBM en 1998.

### II.1 Méthode de développement logiciel :

- ✓ Itératif.
- ✓ Incrémentale.
- ✓ Pilotée par les cas d'utilisation.
- ✓ Centré sur l'architecture et la réduction des risques.
- ✓ Produit de qualité.

### II.2 phases du RUP :

### II.2.1 phase d'initialisation :

- ✓ Définir la portée du projet
- ✓ Spécification
- ✓ Affectation des taches
- ✓ Evaluation des risques
- ✓ Cycle de vie

### II.2.2 phase d'élaboration :

- ✓ Planifier le projet,
- ✓ Spécifier les fonctionnalités
- ✓ Construire l'architecture
- ✓ Spécification du produit
- ✓ Conception de l'architecture
- ✓ Planification activités-ressources

### II.2.3 phase de construction :

- ✓ Construire le produit
- ✓ Implémentation du produit
- ✓ Tests
- ✓ Produit opérationnel

### II.2.4 phase de transition :

- ✓ Transition du produit vers les utilisateurs
- ✓ Livraison
- ✓ Formation
- ✓ Qualité

### II.3 cycle de vie RUP :

- ✓ **Exigence** détermination des besoins :

-fonctionnels (ce que l'on attend du système)

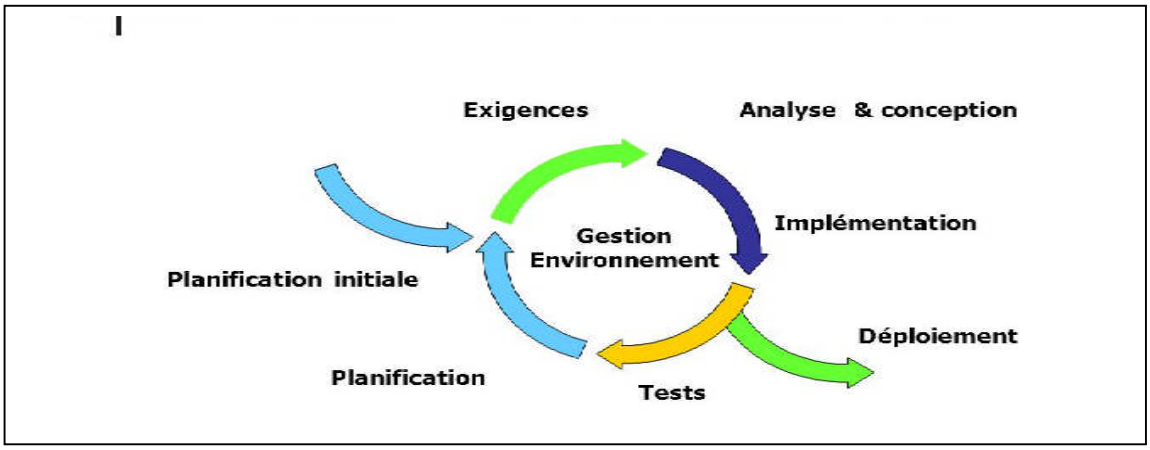
- non fonctionnels (fiabilité, temps de réponse, environnement distribué, etc…).

- ✓ **Analyse et conception :** évoluer depuis la spécification des besoins jusqu'à une solution informatique
  -analyse besoins fonctionnels
  -conception intègre aussi les besoins non fonctionnels.
- ✓ **Implémentation :**

-transcription dans un langage de programmation ou de base de données.

-utilisation de composants existants.

- ✓ **Test :** estimer si les besoins sont satisfaits. S'il ya des erreurs/défauts à corriger renforcer et stabiliser l'architecture.
- ✓ **Déploiement :** distribuer le logiciel dans sans environnement opérationnel, installation, test, formation des utilisateurs, Migration des données.




**Cycle de vie de RUP**

### II.4 Différence entre incrémental et itératif :

- **RUP est incrémental :**

Le produit final est livré en incréments

- ✓ Chaque incrément livre une fonctionnalité ayant une valeur pour l'utilisateur final
- ✓ Chaque incrément consiste en un ensemble de cas d'utilisation
- ✓ Les incréments doivent être de courte durée (quelque semaine a quelque mois)
- ✓ Les incréments sont des mini-projets dans un projet.

- **RUP est itératif :**

Une itération est une séquence d'activités qui répond à un plan et a des critères d'évaluation et qui produit une version exécutable.

- **Force de la méthode RUP :**

- ✓ Traçabilité a partir des uses Cases jusqu'au déploiement
- ✓ Approche basée sur l'architecture
- ✓ Gestion des risques dans les projets

- **Faiblesse de la méthode RUP**

- ✓ Cout de personnalisation souvent élevés
- ✓ Très axé processus

## III. Conclusion :

Cette partie nous a permis de prendre connaissance le processus RUP (Rationnel Unified Processus) est une méthode de développement pour logiciel orienté objet, c'est une méthode générique itérative et incrémentale.

# Bibliographie

**[1]** Mémoire : Conception et réalisation d'une application client/serveur pour la gestion des immobilisations selon le nouveau système comptable et Financier (SCF).

Cas : ENIEM de Tizi Ouzou.

Réalisé par : M$^{elle}$ AMRANI Katia et M$^{elle}$ TAVARES GOMES Ivandra Marise

Dirigé par : M$^{r}$ Si-Mohammed Malik.

Promotion : 2012/2013.

**[2]** Titre : Cours NFE107 Urbanisation & Architecture des Systèmes d'Information Auditeur : Leonardo AMODIO.

Année Édition **:** 2008/2009.

**[3]** Titre : J2EE vs NET, Cours de Java fait par Philippe Poulard,

Année Édition **:** 2008/2009.

**[4]** Mémoire : Conception et réalisation d'une application Java EE pour la gestion du matériels

Cas : ENIEM de Tizi Ouzou.

Réaliser par : Rassoul Idir.

Dirigé par : M$^{lle}$ MOKHBI Ouiza et M$^{lle}$ TOUBAL Samira.

Promotion : 2012 – 2013.

**[5]** Conception et réalisation d'une application N-tiers J2EE

Cas : Gestion de ressources humaines a l'ENIEM.

Dirigé par : M$^{me}$ Bourkache Ghenima.

promotion : 2011 – 2012.

## Webliographie :

**[6] http://www.commentcamarche.fr**

**[7] www.developpez.com**

**[8] http://www.jmdoudoux.fr**