REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERCITE MOULOUD MAMMERI DE TIZI OUZOU
FACULTE DE GENIE ELECTRIQUE ET INFORMATIQUE
DEPARTEMENT INFORMATIQUE

Mémoire

En vue de l'obtention du Diplôme de Master Rechercher Option Système Informatique

Thème

Conception & Réalisation d'une application J2EE Web service cas: Gestion de la scolarité du département informatique

DIRIGÉ PAR:
MM BOURKACHE

RÉALISÉ PAR :
Mlle LEHAD OMAR
Mlle SADDEDINE MOHAMED

Promotion 2011/2012

Remerciement

Nous remercions tout d'abord Dieu qui nous a donné la foi et le courage pour accomplir ce projet.

On tient aussi à exprimer nos vifs remerciements à notre promotrice madame BOURKACHE qui nous a aidés à réaliser ce travail.

Nos profondes gratitudes vont également à l'attention des membres de jury qui nous ferons honneur d'examiner notre travail, ainsi tous les enseignants qui ont contribués à notre formation.

A fin de ne pas oublier personne, nous remercions tous ceux qui ont contribués de près ou de loin à l'élaboration de ce projet.

Omar & Mohammed

Dédicaces

A mes parents
sans eux je n'aurais jamais pu arriver jusque là.
Merci pour votre amour, votre présence
et pour votre soutien continu.
J'espère que je vous rendrai de plus en plus fiers,
je vous suis extrêmement reconnaissant.

Ma chère Grand-Mère fatma que j'aime beaucoup.
Mon grand frère Boudjema, qui ma aidé beaucoup.
Mes soeurs Fatiha, Samia, Hania, Malika et Farida,
mes frères Ramdan, Iddir, Rabeh, Lyese, Smail et Marzouk que j'aime tous.
Je vous souhaite tous le bonheur que vous méritez.
Tous mes oncles et cousins.

A mon binôme Mohamed et toute sa famille.
Mes amis très proches Brahim, Amirouche, Farid,Ghiles, Lynda, Souhila, Chrifa merci pour cette Merveilleuse et inoubliable année et pour les moments bons et moins bons qu'on a traversé ensemble.

LEHAD Omar

Dédicace

A la première femme que mon regard a pu voir,
toi ma très chère mère qui a sacrifié sa vie pour l'éducation de ses enfants, quoi que je fasse je pourrai jamais t'offrir ce que tu ma procuré. Laisse moi te dédier ce modeste travail et que dieu vous garde pour nous.
A mon meilleur guide qui me montre le chemin dans les nuits obscures, à toi mon très cher père et que dieu vous garde pour nous.
A mes très chers frères Djelali, Toufik et en particulier Anis & Hacen et sœurs Djedjega, Safia et en particulier Malika, ceux qui ont sacrifiés leur vie afin que je puisse en avoir une. Merci pour avoir suscité ma vocation et permis d'achever mes études.
A toute ma famille A tous mes proches grands et petits.

A celle qui m'a fait sentir l'amour, A la femme la plus chère à mon cœur, ma future femme SOUHILA avec qui j'ai partagé les meilleurs moments de ma vie.

A tous mes amis qui étaient présent à mes
coté "Mohand", "Nacer", "Hocine F47", et surtout le groupe G8
"Lynda", "Omar", "Brahim", "Amirouche", "Farid",
"Souhila", "Cherifa"
et pleins d'autres que je ne pourrai citer faute d'espace !

A tous ceux qui m'ont aidé de prés ou de loin dans mes études.

SADDEDINE Mohamed

Table des matières

Table des matières

Introduction Générale……..……..…1

Technologie Web

Introduction……..……3

I. Les réseaux……..……..……3

I.1. Objectifs des réseaux……..……3

I.2. Classifications des réseaux…….……4

I.2.1. Réseaux local (LAN : Local Area Network)……...4

I.2.2. Réseau métropolitain (MAN : Métropolitain Area Network)…….4

I.2.3. Réseaux étendus (WAN : Wide Area Network)……..4

I.3. Architecture des réseaux……...……..4

I.3.1. Modèle de référence OSI……..……5

I.3.2. Architecture TCP/IP……7

II.Architecture Client –Serveur…….……8

II.1. Notion de base……...…….....8

II.2. Fonctionnement d'un système Client/Serveur……..……8

II.3. Objectifs de l'architecture Client/Server……..…….9

II.4. Caractéristiques de l'architecture Client/Serveur……..9

II.5. Evolution de l'architecture client-serveur……...10

II.5.1. Le Client/Serveur de première génération……...10

II.5.2. Le Client-serveur de deuxième génération……..10

II.5.3. Le Client/Serveur de troisième génération……...11

II.6. Avantages et inconvénients du modèle Client/Serveur……...12

III.INTERNET……...…….....13

III.1. Les principaux services de l'internet……...……..13

IV. Introduction au Web…….……14

IV.1. A l'origine du Web……..……..14

IV.2. Concepts du Web……..……14

IV.3. Types des pages Web……15

V. Conclusion…….…….....16

Chapitre II Technologie J2EE et Web Service

Introduction…….....17

I.Architecture logicielle……....17

I.1. Critères de qualité logicielle……..……17

II. PRESENTATION DE TECHNOLOGIE J2EE…….....……...19

II.1. Introduction à J2EE……...…….19

II.2. Fonctionnement interne…….……..19

II.3. Architecture……...…….19

II.4. Les services de JEE…….20

II.4.1. Les services d'infrastructures…..…….20

II.4.2. Les services de communication…..……21

II.5. Composants de l'architecture JEE……...21

II.5.1. Composants WEB…….....…….21

II.5.1.1. Page JSP……...……..21

II.5.1.2. Servlet……...…….....26

II.5.1.3. Conteneur de composant web…….……...27

II.5.2. Composants EJB……...…….28

II.5.2.1. Serveur EJB…….……..28

II.5.2.2. Client EJ……28

II.5.2.3. Conteneur EJB…..……31

II.5.2.4. Déploiement des EJB……...…….31

II.6. Technologie J2EE…….……...32

II.6.1. Java Name and Directory Interface…….…….32

II.6.2. JDBC…….…….....34

II.6.3. Architecture RMI…….....……...34

III.PRESENTATION DE TECHNOLOGIE WEB SERVICE……...……...39

III.1. Introduction……...….....39

III.2. Définition……...……...…….....39

III.3. Les caractéristiques d'un service Web……..……..40
III.4. Architecture d'un service Web……...……40
III.5. Fonctionnement des services Web…….42
III.5.1. Service provider service……...…42
III.5.2. Service requester programme client……...42
III.5.3. Annuaire service registry…….....43
III.6. Le protocole de communication SOAP…….43
III.6.1. Structure d'un message SOAP……44
III.6.2. Exemple de message SOAP……45
III.8. L'annuaire des services UDDI……...47
III.8.1. Consultation de l'annuaire……...48
III.8.2. Structures de données UDDI……..49
III.9. Le langage XML……....49
Conclusion……..50

Chapitre III Système LMD

Introduction…….....51
I.Historique de la réforme LMD ……...…….51
II. Objectifs de la réforme……...……52
III.L'unité d'enseignement……...……..53
IV. évaluation et progressions……..……..54
IV.1. La progression dans les études……..55
IV.2. la progression dans les études de licence……...55
IV.3. la progression dans les études de master……...55
V. l'algorithme général pour les deux systèmes de progression…….56
V.1. premier système de progression……..56
V.2. deuxième système de progression……...57
VI. Forme présentielle des études et forme tutoriel des études……..57
VI.1. Forme présentielle des études……57
VI.2. forme tutoriel des études……..57
Conclusion……..58

Chapitre IV Analyse et Conception

VI.1 Introduction
……59

VI.2 Présentation des diagrammes UML……...59

VI.2.1. Les diagrammes de structure…….59

VI.2.2. Les diagrammes de comportement : qui sont au nombre de sept……...60

VI.3. Une démarche pour l'analyse et la conception du projet……..63

VI.3.1. Description générale de la démarche……..63

VI.3.2. Phase d'analyse……..…….65

VI.3.3. Phase de conception……65

VI.3.4. Phase de réalisation……65

VI.4. Cahier de charge de notre application. …….66

VI.5. Analyse et conception de notre système……67

VI.5.1 Analyse……67

VI.5.1.1. Spécification des besoins……..67

VI.5.1.2. Cas d'utilisations……...67

VI.5.1.3. Spécification des tâches……68

VI.1.4. Diagramme des cas d'utilisation globale……70

VI.5.2. Conception…...……..74

VI.5.2.1. Elaboration des Diagrammes de séquences ……..74

VI.5.2.2. Diagramme d'activité…….83

VI.5.2.3. Diagrammes de classe…….90

VI.5.2.4. Diagramme de classe global……99

VI.5.2.5. Implémentation de la base de données…….100

Conclusion……105

Chapitre V Réalisation

Introduction……...106

Description des outils de développement…….106

I.1. Langage de programmation (Java) ……106

I.2. IDE (NetBeans) ……..106

I.3. Le serveur d'application (Glassfish) ……..107

I.4. Le SGBD (MYSQL) ……108

II. présentation de l'application……109

II.1. JDBC…….110

II.2. JPA ……...110

II.3. DAO……..110

II.4. Web Services. ……..110

III. Implémentation des différentes parties de l'application…….110

III.1. Création de la base de données ……...110

III.2. Partie Traitement……..112

III.2.1. le projet Gestion de la scolarité……112

III .2.2 Le déploiement du web service……...122

III.3. Partie Présentation……..…….126

III.3.1. Représentation des interfaces de l'application client « Agent de Scolarité » …….127

III.3.2. Représentation des interfaces de l'application client « Administrateur » ……...136

III.3.3. Représentation des interfaces de l'application client « Enseignant »…… ……..138

Conclusion...……..……...139

Conclusion Générale……..……...140

Annexe A……..……141

Annexe B……..146

Références bibliographiques……...……...154

Table des figures

Table des figures

Figure I.1 : Les différentes couches de modèle OSI…….....5
Figure I.2: Fonctionnement de modèle Client/Server…….8
Figure I.3 : Client/Serveur de première génération……..……...10
Figure I.4: Client/Serveur de deuxième génération……..11
Figure I.5 :Client/Serveur de troisième génération……..……...11
Figure II.1: Architecture et les composants de J2EE……20
Figure II.2: l'architecture générale d'un EJB……28
Figure II.3 : les technologies J2EE en fonction de type de contene……32
Figure II.4 : l'architecture de la technologie JNDI. ……33
Figure II.5: Présentation de l'architecture RMI……35
Figure II.6: Etapes d'un appel de méthode distante……37
Figure II.7: Fonctionnement d'un service web……42
Figure II.8: le protocole de communication SOAP……43
Figure II.9: Structure de message SOAP……44
II.10. Le langage de description WSDL……45
Figure II.11: Structure de document WSDL……46
Figure II.12: schéma général de l'annuaire…….……...48
Figure VI.1 : Classification des diagrammes UML (V. 2.0)……62
Figure VI.2 : Phase du Processus d'analyse, de conception et de réalisation…….……...63
Figure VI.3 : Démarche d'analyse et de conception du SAI avec les diagrammes d'UML……64
Figure VI.4 Diagramme de cas d'utilisation Agent de la scolarité……..…71
Figure VI.5 Diagramme de cas d'utilisation Administrateur……..……72
Figure VI.6 Diagramme de cas d'utilisation Enseignant…….……..73
Figure VI.7 Diagramme de cas d'utilisation Etudiant……..74
Figure VI.8 Diagramme de séquence de cas d'utilisation « Authentification »…….....75
Figure VI.9 Diagramme de séquence de cas d'utilisation « Gérer les notes » …….….76
Figure VI.10 *Diagramme de séquence de cas d'utilisation « Edition de relevé de note »…..…77*
Figure VI.11 Diagramme de séquence de cas d'utilisation « Changer le mot de passe »……..…78
Figure VI.12 Diagramme de séquence de cas d'utilisation « Création de compte» ……79
Figure VI.13 Diagramme de séquence de cas d'utilisation « Affectation des Enseignants »..80

Figure VI.14 Diagramme de séquence de cas d'utilisation « Etablir les sections et les groupe » ……81
Figure VI.15 Diagramme de séquence de cas d'utilisation « Voir les notes »…….82
Figure VI.16 Diagramme d'activité de cas d'utilisation « S'authentifier »……..83
Figure VI.17 Diagramme d'activité de cas d'utilisation «*Gérer les notes* »……....84
Figure VI.18 Diagramme d'activité de cas d'utilisation «*Edition des relevés des notes*»……85
Figure VI.19 Diagramme d'activité de cas d'utilisation «*Changer le mot de passe*»……..86
Figure VI.20 Diagramme d'activité de cas d'utilisation «*Création de Compte*»…….87
Figure VI.21 Diagramme d'activité de cas d'utilisation «*Affectation des Enseignants*»…….88
Figure VI.22 Diagramme d'activité de cas d'utilisation «*Etablir les Sections et les Groupes*»……...89
Figure VI.23 Diagramme d'activité de cas d'utilisation «*Voir les Notes »*……...*90*
Figure VI.24 Diagramme de classe détaillé du cas d'utilisation « authentification »…….....91
Figure VI.25 Diagramme de classe de cas d'utilisation « Gestion des Notes »……92
Figure VI.26 Diagramme de classe de cas d'utilisation « Editer les relevé Notes »……93
Figure VI.27 Diagramme de classe de cas d'utilisation « Changer le Mot de Passe »…….94
Figure VI.28 Diagramme de classe de cas d'utilisation « Création de Compte »……95
Figure VI.29 Diagramme de classe détaillé du cas d'utilisation « Affectation des Enseignants»…….96
Figure VI.30 Diagramme de classe de cas d'utilisation « Etablir les sections et les groupes »……...97
Figure VI.31 Diagramme de classe de cas d'utilisation « Voir les Notes »…….....98
Figure VI.32 Diagramme de classe Global……...99
Figure V.1 Interface de l'IDE Netbeans……107
Figure V.2 Interface de serveur de déploiement GlassFish……108
Figure V.3 Interface de MySQLWorkbench……..109
Figure V.4 Architecture J2EE en couche……109
Figure V.5 La base de données « bdd-Scolarité » de l'application……111
Figure V.6. Script de création de la table Etudiant……111
Figure V.7 Les tables de la BDD « bdd-scolarite »……112
Figure V.8. Application J2EE Gestion Scolarité……113
Figure V.8. Le code source du fichier de configuration [sun-resources.xml]…….114
Figure V.9. Le code source du fichier de configuration [persistence.xml]……...114

Figure V.10. Le code source de l'entité Compte……....117
Figure V.11. Le code source EJB session qui manipule l'entité Compte……...118
Figure V.12. EJB session qui implémente toutes les méthodes……..119
Figure V.13. Une Portion du code de la classe WsImplMtethodes……120
Figure V.14. Le module WsImplMethodes……...121
Figure V.15.Vue partielle des Web méthodes…….122
Figure V.16. Console administration du serveur Glassfish……123
Figure V.17. Vue Partielle de la page test du web service……..124
Figure V.18. Teste de la méthode AjouterCompte()……..124
Figure V.19. La demande SOAP de la méthode AjoutCompte…….125
Figure V.20. La Réponse SOAP pour la méthode AjoutCompte……...…125
Figure V.21. Vue partielle du fichier WSDL……126
Figure V.22. Fenêtre d'authentification Agent de Scolarité……...127
Figure V.23. Fenêtre d'accueil de l'agent de scolarité……...128
Figure V.24. Fenêtre de l'ajout d'un étudiant. ……..129
Figure V.25. Fenêtre formulaire édition de la scolarité……..130
Figure 4.26. Fenêtre édition de la scolarité…….131
Figure 4.27.Edition de relevé de note…….132
Figure V.28. Fenêtre Edition de la liste des étudiants..…….……133
Figure V.39. Fenêtre Liste des étudiants sous forme PDF……..134
Figure V.30. Fenêtre Affectation des enseignants……..135
Figure V.31. Fenêtre Ajouter compte enseignant……...136
Figure V.32. Fenêtre Modification/Suppression compte enseignant…….137
Figure V.33. Fenêtre Ajout compte étudiant……..137
Figure V.34 Fenêtre Authentification enseignant……...138
Figure V.35 Fenêtre Gestion des notes……...138

Introduction Générale

Introduction Général

Pour de nombreux développeurs, J2EE est souvent synonyme de « Entreprise JavaBeans ». En fait, J2EE est beaucoup plus que cela. En simplifiant, nous pouvons dire que J2EE est une collection de composants, de conteneurs et de services permettant de créer et de déployer des applications distribuées au sein d'une architecture standardisée.

J2EE fournit un ensemble de composants standardisés facilitant le déploiement des applications, des interfaces définissant la façon dont les modules logiciels peuvent être interconnectés, et les services standards, avec leur protocole associé, grâce auxquels ces modules peuvent communiquer.

Le Web est de plus en plus le support privilégié des applications d'entreprise. Les Web services constituent le développement ultime dans ce domaine. Un service Web est un système logiciel identifié par une URI, dont l'interface publique et les liens sont définis à l'aide d'XML. Ce système peut être découvert ou localisé par d'autres systèmes logiciels. Ces systèmes peuvent interagir avec le service d'une manière prescrite par sa définition, en utilisant des messages XML transmis à l'aide des protocoles d'Internet.

La réforme de l’enseignement supérieur est devenue, aujourd’hui plus qu’une nécessité pour remédier à tous les dysfonctionnements existants et rendre l’université algérienne plus performante, complétive et attractive et lui permettre de répondre aux grands défis de la modernisation et de l’évolution rapide des sciences et de la technologie.
Vus les aléas que le nouveau système LMD pose au niveau de la gestion des enseignements de département d’informatique de notre faculté, notre projet consiste à la mise en place d’une application en basant sur l’architecture J2EE et web service pour l’automatisation des activités de gestion de la scolarité de la faculté génie électrique.

Le mémoire est organisé en cinq chapitres :
Dans le premier chapitre, nous présenterons la technologie web en décrivant ces principaux acteurs. Le second chapitre introduira les concepts fondamentaux de l’architecture J2EE et web services qui est la solution la plus utilisée dans le domaine de gestion des entreprises.
Une présentation générale sur le système LMD et ses caractéristiques qui y sont définies fera l’objet du troisième chapitre.

Le chapitre quatre sera consacré à la conception de l'application décrite par des diagrammes UML. Dans le dernier chapitre nous décrirons le logiciel réalisé ainsi que son utilisation. Enfin nous conclurons tout en donnant quelques perspectives.

Chapitre I

Technologies Web

Introduction

Les nouvelles technologies de l'information et de la communication, représentent l'ensemble des technologies informatique qui contribuent à une véritable révolution socioculturelle, surtout leurs applications dans le domaine économique.

Les NTIC sont un ensemble de technologies utilisées pour traiter, modifier et échanger de l'information, plus spécifiquement des données numérisées. La naissance des NTIC est due notamment à la convergence de l'information, des télécommunications et de l'audiovisuel. Cette convergence génère une multitude de nouvelles possibilités. C'est en quelque sorte notre rapport à l'information, au temps et à la distance qui est changé. Toutefois, nous décrivons un modèle architectural approuvé à savoir le modèle Client-Serveur.

I. Les réseaux

Les réseaux sont nés d'un besoin d'échange des informations de manière simple et rapide entre des machines. Lorsque l'on travaillait sur une même machine, toutes les informations nécessaires au travail étaient centralisées sur cette machine. Pour des raisons de coûts ou de performances, on est venu à multiplier le nombre de machines. Les informations devaient alors être dupliquées sur les différentes machines de même site. Cette duplication était plus au moins facile et ne permettait pas toujours d'avoir des informations cohérentes. On est donc arrivé à relier d'abord ces machines entre elles. Ce fût l'apparition des réseaux locaux. Ces réseaux étaient souvent des réseaux propriétaires. Plus tard on a éprouvé le besoin d'échange des informations entre des sites distants. Les réseaux moyenne et longue distance commencèrent à voir le jour. Aujourd'hui, les réseaux se retrouvent à l'échelle planétaire. Le besoin d'échange de l'information est en pleine évolution. **[1]**

I.1. Objectifs des réseaux

Les réseaux permettent :

- Le partage de fichiers.
- Le transfert de fichiers.
- Le partage d'application : compilateur, système de gestion de base de données(SGBD).
- Le partage de périphérique.
- L'interaction avec les utilisateurs connectés : messagerie électronique, vidéo conférence, Talk…
- Le transfert de données en général (réseaux informatique)
- Le transfert de la parole, de la vidéo et des données (réseaux à intégration de services ou multimédia).

I.2. Classifications des réseaux [2]

La principale classification des réseaux est faite selon leur taille en : réseau local (LAN), Réseaux métropolitain (MAN) et réseaux étendu (WAN).

I.2.1. Réseaux local (LAN : Local Area Network)

Il est formé d'un ensemble de stations situées dans une même zone géographique limitée (par exemple : un ensemble de bureau, un immeuble).

I.2.2. Réseau métropolitain (MAN : Métropolitain Area Network)

C'est un réseau situé dans une zone géographique plus petite que les réseaux étendus. Les réseaux métropolitains permettent des communications entre différents grands comptes ou à partir d'ordinateurs personnels.

I.2.3. Réseaux étendus (WAN : Wide Area Network)

Relie plusieurs réseaux locaux entre eux. Les réseaux étendus couvrent une zone géographique importante et les liaisons entre les réseaux locaux se font à travers des lignes de communication internationales et des routeurs. Ces réseaux étendus relient des machines hétérogènes qui utilisent des systèmes d'exploitation différents.

I.3. Architecture des réseaux :

Pour assurer une connexion d'une machine, il faut réunir les supports physiques. Mais pour s'assurer du bon transfert de l'information avec une qualité de service suffisante, il faut prévoir une architecture logicielle.

Une normalisation de l'architecture logicielle s'impose. Dans cette section nous allons décrire deux architectures réseaux, la première provient de l'ISO et s'appelle OSI (open system interconnexion), la deuxième est l'architecture TCP/IP.

I.3.1. Modèle de référence OSI [3]

Le model de référence OSI se fonde sur une proposition élaborée par l'organisation internationale de normalisation(OSI) ; il est appelé model de référence OSI (open system interconnexion) parce qu'il traite de la connexion entre les systèmes ouverts en communication avec d'autre systèmes. Les principes de base ayant conduit à l'élaboration des sept couches sont les suivantes :

- Diviser les problèmes de communication sur les réseaux en problèmes plus simples et plus faciles à gérer.
- Chaque couche exerce des fonctions bien définies.
- Le nombre de couche doit être suffisamment grand pour éviter la cohabitation dans une couche de fonctions très différentes et suffisamment petit pour éviter que l'architecture devienne difficiles à maîtriser.

D'un point de vue conceptuel, chaque couche interagit avec son homologue située sur un ordinateur distant. En pratique, chaque couche communique avec la couche au dessus et en dessous d'elle.

Fonctionnement : chaque couche (n) offre un certain nombre de services à la couche (n+1) en déroulant un protocole uniquement défini à partir des services fournis par la couche (n-1).

L'architecture OSI est schématisée comme suit :

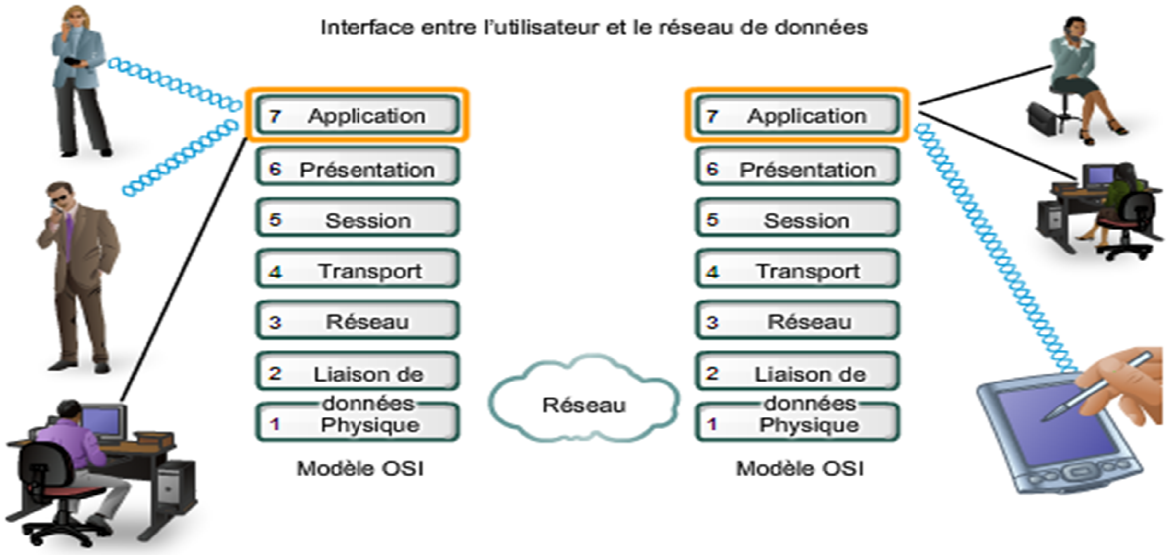


Figure I.1 : Les différentes couches de modèle OSI.

Couche Application (7)

- Sert d'interface entre les applications à chaque extrémité du réseau.
- Permet d'changer des données entre les programmes s'exécutant sur les hôtes source et de destination
- Il existe de nombreux protocoles de couche application et de nouveaux protocoles sont constamment en cours de développement.

Couche Présentation (6)

- Codage et conversion des données de la couche application afin que les données issues du périphérique source puissent être bien interprétées sur le périphérique de destination ;
- Compression des données de sorte que celles-ci puissent être décompressées par le périphérique de destination ;
- Chiffrement des données en vue de leur transmission et déchiffrement des données reçues par le périphérique de destination.

Couche Session (5)

- Permet d'initier un dialogue entre les applications source et de destination.
- Initier et maintenir un dialogue
- Redémarrer les sessions interrompues ou inactives pendant une longue période.

Couche Transport (4)

- Permet l'acheminement de bout en bout sans se soucier des relais intermédiaires.

- Fragmentation du message en unités plus petites dites paquets
- Multiplexage

Couche réseau (3)

- Permet l'acheminement de bout en bout en tenant compte des nœuds intermédiaires
- Routage et ordonnancement des paquets

Couche liaison de données (2)

- Structuration des données en trames
- Masquer les caractéristiques physiques
- Contrôle d'erreur à l'émission et à la réception

Couche physique (1)

- Assurer la transmission de bits entre les entités physiques
- Spécifie la nature du support de communication
- Le mode de connexion et le brochage le cas échéant
- La technique de codage des bits en signaux électriques
- Les tensions et les fréquences utilisées

I.3.2. Architecture TCP/IP [4]

La défense américaine devant le fonctionnement des machines utilisant des protocoles de communication différents et incompatibles à décider de définir sa propre architecture. Ces protocoles représentantes aussi, comme l'architecture **OSI**, une architecture en couches.

L'architecture **TCP/IP** se schématise comme suit :

IP : « internet protocol » protocole de niveau réseau assurant un service sans connexion.

TCP : « transmission control protocol » niveau transport (niveau 4) qui fournit un service fiable en mode connecté.

UDP : « user data protocol » niveau transport en mode non connecté.

FTP : « file transfert protocol » niveau de fichiers.

SMTP : « simple mail transfert protocol » pour le transfert de courrier électronique.

TelNet : protocol de gestion de terminal virtuel (permet d'obtenir les logiciels d'un autre ordinateur grâce au réseau).

II. Architecture Client –Serveur [5]

Le Client/Serveur est un modèle de communication entre deux processus. Le premier appelé client, demande des services au deuxième processus qui est le serveur, ce dernier exécute la requête du client et envoie des résultats en retours. Client et serveur sont en pratique deux logiciels différents communiquant au moyen d'un protocole, à travers un réseau local ou bien à travers un réseau étendu.

II.1. Notion de base :

- ✓ **Requête :** message transmis par un client à un serveur décrivant l'opération à exécuter pour le compte du client.
- ✓ **Réponse :** requête transmis par le serveur à un client suite à l'exécution d'une opération contenant les paramètres de retour de l'application.
- ✓ **Client :** processus demandant l'exécution d'une opération à un autre processus par envoi d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un message en retour. Ce client émet cette requête vers le serveur grâce à son adresse **IP** et son **port**, qui désigne un service particulier du serveur.
- ✓ **Serveur :** processus accomplissant une opération sur demande d'un client et transmettant la réponse à ce client à l'aide de l'adresse de sa machine. Il est par conséquent initialement passif car il est toujours en attente prêt à traiter les requêtes envoyées par les clients.

II.2. Fonctionnement d'un système Client/Serveur :

Un système client/serveur fonctionne selon le schéma suivant :

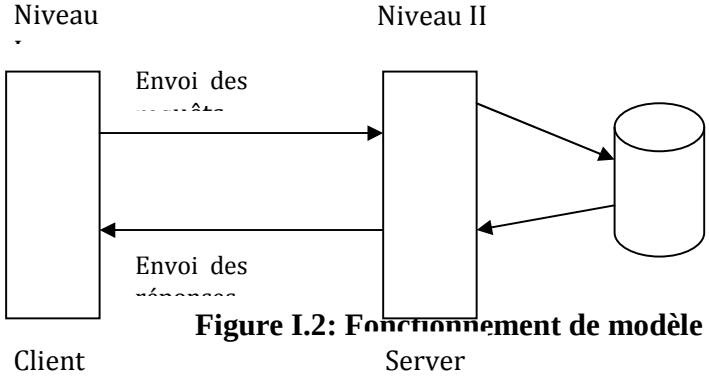


Figure I.2: Fonctionnement de modèle Client/Server.

Le client émet une requête vers le serveur grâce à son adresse et le port, qui désigne un service particulier du serveur. Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine et son port.

II.3. Objectifs de cette architecture :

- Une grande flexibilité/souplesse.
- Une plus grande sécurité (la sécurité peut être définie pour chaque service).
- De meilleures performances (les tâches sont partagées).

II.4. Caractéristiques de l'architecture Client/Serveur [5]

- **Service :** le modèle Client/Serveur est essentiellement une relation entre des processus. Le processus serveur est fournisseur de service et le client en est le consommateur, le modèle Client/Serveur établit ainsi une opération claire de rôles à partir de la notion service.
- **Partage de ressources :** un serveur est censé savoir traiter plusieurs demandes clients à la fois et leurs accès aux ressources.
- **Asymétrie des protocoles :** la relation entre le client et le serveur est de type plusieurs vers un, toutefois, le client est le déclencheur de dialogue en demandant un service alors que le serveur attend passivement les requêtes.
- **Assemblage multy-vendeur** : le client-serveur est indépendant de la plateforme matérielle ou du système d'exploitation. On doit toujours pouvoir mélanger et apparier les plates formes Client/Serveur
- **Echange de messages** : le client et le serveur sont des systèmes à liaison épisodique qui interagissent au moyen de message. Le message est un mécanisme d'émission de demandes de services et de réponses à celles-ci.
- **Encapsulation des services** : le serveur est un spécialiste, un message lui indique quel service est requis et c'est à lui de décider comment rendre ce service. Les serveurs peuvent être mis à niveau sans effet sur les clients tant que l'interface des messages reste la même.
- **Intégrité** : le code et les données du serveur sont gérés de façon centralisée, ce qui garantit un moindre coût de maintenance et une meilleure intégrité des données tandis que les clients restent individuels et indépendants.
- **Souplesse et adaptabilité :** on peut modifier le module serveur sans toucher au module client et vice versa, si une station est remplacée par u modèle plus récent, on modifie le module client sans modifier le module serveur.

II.5. Evolution de l'architecture client-serveur [5]

Le modèle Client/Serveur constitue une évolution majeure de l'informatique. Le principe de base est de décomposer un processus informatique en au mois deux taches moins complexes (le client et le serveur) associés à un mécanisme de communication leur permettant de coopérer.

II.5.1. Le Client/Serveur de première génération

Né à la fin des années 80 et basé sur des outils clients des SGBD relationnels. Le développement s'effectue sur le serveur pour la base de données et sur le client pour l'application. L'inconvénient de cette architecture est que tout le code applicatif est exécuté sur le client.

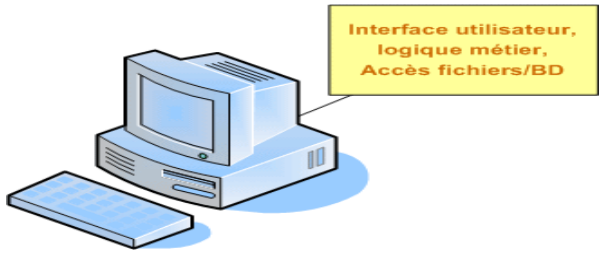


Figure I.3 : Client/Serveur de première génération.

II.5.2. Le Client-serveur de deuxième génération [6]

Né au milieu des années 90, cette génération est caractérisé par :

- La possibilité de développer des traitements applicatifs au sein du serveur de données sous forme de procédure déclenchées par l'application ou lors d'événement sur la base de données.
- L'utilisation de l'approche orientée objet.
- La facilité de déploiement des applicatifs avec partitionnement du code applicatif entre le client et le serveur.

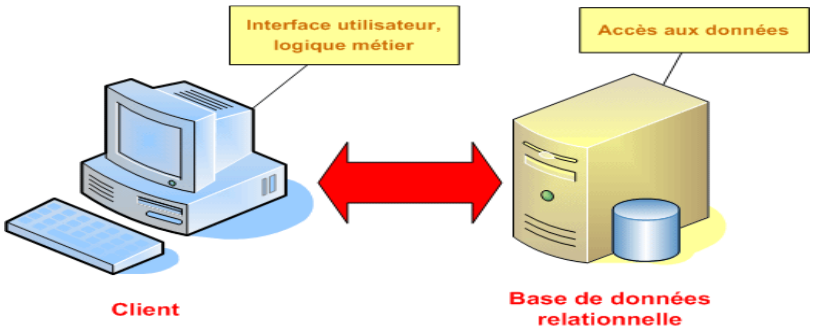


Figure I.4: Client/Serveur de deuxième génération.

II.5.3. Le Client/Serveur de troisième génération : [6]

Avec l'apparition d'internet et le web, les architectures Client-serveur sont évoluées vers des architectures à trois niveaux.

- ✓ Le client est responsable de la présentation. Il utilise pour cela des navigateurs web, comme internet explorer.
- ✓ Le serveur d'application exécute le code applicatif.
- ✓ Le serveur de données supporte le SGBD qui gère éventuellement des types de données très riches, comme les données multimédias.
- ✓ Le serveur d'application et le serveur de données peuvent être regroupés sur la même machine.
- ✓ L'architecture à trois-tiers, appelée encore Client/Serveur Web, est aujourd'hui bien adaptée aux systèmes répartis autour d'un réseau local et/ou Internet.
- ✓ Elle permet à de multiples postes ou stations de travail distribuées sur la planète de partager les mêmes données.

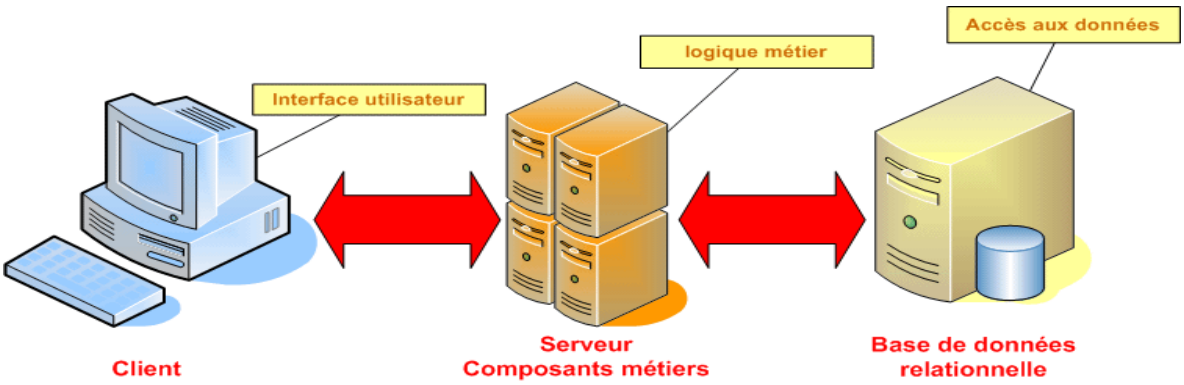


Figure I.5 :Client/Serveur de troisième génération

II.6. Avantages et inconvénients du modèle Client/Serveur [4]

A. Les avantages :

Les principaux avantages de ce modèle sont :

- ✓ Des ressources centralisées : Etant donné que le serveur est au centre du réseau, il peut gérer des ressource communes a tous les utilisateurs, comme par exemple une base de données centralisée afin d'éviter les problèmes de redondance et d'incohérence.
- ✓ Une meilleure sécurité : car le nombre de points d'accès aux données est moins important.
- ✓ Une administration au niveau serveur : Les clients ont moins d'importance dans ce modèle. Ils ont besoin d'être administrés.
- ✓ Un réseau évolutif : grâce à cette architecture, on peut supprimer ou ajouter des clients sans perturber le fonctionnement du réseau et sans modifications majeures.

B. Inconvénients :

L'architecture Client/Serveur a tout de même quelques inconvénients parmi les quels :

- ✓ Un coût élevé dû à la technicité du serveur (bien plus puissant qu'une machine de bureau).
- ✓ 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.

III. INTERNET [1]

Il est presque impossible de cerner une telle question, car c'est une technologie en constante évolution. Pour cela, on dira que internet est un gigantesque réseau mondial d'ordinateurs qui englobe plusieurs autre réseaux de moindre taille. Ces derniers permettent eux-mêmes d'interconnecter les ordinateurs individuels.

III.1. Les principaux services de l'internet

Les services d'internet sont généralement liés à la consommation :

- ✓ **E-MAIL (la messagerie électronique) :** C'est d'utiliser internet comme on utilise la poste. Il est possible de déposer un message dans la boîte aux lettres de son correspondant, qu'il soit ou non devant sa machine. Ce dernier sera capable à sa prochaine connexion de consulter sa boîte aux lettres pour lire et envoyer des messages a ses correspondant.
- ✓ **IRC (internet relay chat) :** C'est une forme communication interactive entre un individu et un autre par envoi de textes écrits.
- ✓ **NEWS :** A l'inverse du mail ou la discussion est réalisée de l'émetteur vers le récepteur, les forums de discussion sont des moyens de discussion entre plusieurs personnes.
- ✓ **FTP :** C'est le service d'échange de fichiers, il permet de déposer des fichiers sur une machine distante, mais aussi de télécharger des fichiers sur sa machine.
- ✓ **WAIS et GOPHER :** les services Wais (Wibe Area Information Servers) et Gopher permettent de retrouver des documents de divers types localisées sur des serveurs distants. Les sont très variées on y trouve par exemple des images, des sons, des banques de données.

- ✓ **WWW ou le Web :** C'est le service le plus connu de l'internet que nous allons détailler après.
- ✓ **DNS (Domain Name System) :** DNS est un service qui convertit l'URL d'une page WEB en son adresse IP (adresse binaire sur 32 bits définissant sans ambiguïté chaque ordinateur ou routeur du réseau internet). Les adresses IP sont obtenues par interrogation d'un serveur de noms. Celui-ci gère un tableau contenant les noms de domaine et les adresses IP correspondantes.
- ✓ **Multimédia :** avec le web le multimédia est entré dans l'air du grand public, boutique de modes, agences de voyages, publicités, bibliothèque virtuelles, musique en ligne, commerce électronique, toutes ces application mixent du multimédia. Dans les base de données le terme «multimédia» signifie en conséquence le mélange de différents types de données incluant texte libre, géométrie, image, son, vidéo, etc.

IV. Introduction au Web

IV.1. A l'origine du Web [7]

Le WWW (World Wide Web), appelé plus simplement le « Web » et en français la « Toile d'araignée mondiale » est un système hypertexte public fonctionnant sur Internet et permettant de consulter via un navigateur spécifique, des pages Web accessibles en ligne. Rappelons à ce titre que le Web été inventé par *Tim Berner-Lee1*, plusieurs années après Internet et qu'il n'en est qu'une de ses applications au même titre que le courrier électronique, la messagerie instantanée, Telnet,…etc. A l' origine, le Web comprenait des pages statiques reliées entre elles par des liens hypertextes rarement mises à jour, c'est ce qui est appeler le « Web 1.0 ». Ainsi les sociétés fonctionnaient selon un modèle économique estimant que celui-ci était fait de publications, non de participations, et que les annonceurs étaient les véritables acteurs moteurs.

Au milieu des années '90, s'est ensuite développé ce qu'on a appelé les « dotcom » : une structure de pages fondée cette fois sur un Web dynamique où des systèmes de gestions de contenus servaient des pages Web créées à partir d'une base de données en constante évolution. C'est l'avènement des langages de script et du *DHTML (Dynamic HTML)* 2, celui d'un Web parfois appelé également « Web 1.5 ».

IV.2. Concepts du Web L'essentiel des concepts du Web se résume à :

a. Page Web

C'est la page visualisée lorsqu'on charge sur le navigateur un fichier *HTML (HyperText Markup Language)* contenant des éléments multimédias (texte, images, son, vidéo)

b. Site Web Un site Web est un ensemble de pages Web crées par une personne ou un organisme, pour diffuser des informations sur le Web à l'aide de plusieurs pages situées sur le même serveur. Il est généralement structuré autour d'une page centrale, appelée page d'accueil, qui propose des liens vers d'autres pages hébergées sur le même serveur ou sur un autre serveur.

c. Serveur Web Un serveur Web est un programme (résident) qui tourne sur un ordinateur dans le but de répondre à des requêtes de logiciels clients tournant sur d'autres ordinateurs. Ces requêtes peuvent être le transfert d'un fichier ou plus récemment le résultat de l'exécution d'un programme indépendant du logiciel.

d. Navigateur Un navigateur est un logiciel possédant une interface graphique composée de boutons de navigation, d'une barre d'adresse, d'une barre d'état et dont la majeure partie de la surface sert à afficher des pages Web. Il permet à l'internaute de surfer entre les différentes pages de différant sites.

e. Moteur de recherche Un moteur de recherche est un serveur qui indexe un certain nombre de sites Web, et permet généralement de rechercher les informations qui intéressent l'internaute à l'aide de mots clé. C'est en quelque sorte les annuaires du Web.

g. Hypertexte et Hypermédia Un document hypertexte est un fichier texte contenant dans son texte des liens hypertexte ou hyperlien, soit vers d'autres parties de ce document, soit vers d'autres documents. Ces documents peuvent être localisés sur le même ordinateur mais aussi sur un autre ordinateur distant sur le réseau. Un document hypermédia est un document hypertexte avec la différence que les liens peuvent référencer également des fichiers son, images ou images animées (vidéo).

IV.3. Types des pages Web

- **Les pages Web statiques :**

Une page Web statique est un document qui peut être lu de haut en bas sans quitter le document, elle présente ainsi la même information à tous les usagers.
Les pages statiques font appel au langage HTML qui est un langage de description de données, on peut reconnaître une page statique par son adresse URL :
http ://www.monsite.be/accueil.html : affiche de la page accueil.html, stockée telle est sur le serveur.

- **Les pages Web dynamiques :**

Une page web dynamique (interactive) est un document élaboré qui utilise des programmes externes pour réaliser des fonctions spécifiques. Ces pages interactives permettent aux usagers de soumettre des formulaires, d'interroger des bases de données, de formater des résultats, de structurer l'affichage et d'avoir des accès à des parties du site protégées exemple saisie d'un mot de passe pour une session client. Les pages dynamiques sont mises en œuvre grâce à un langage de programmation. Grâce à lui, on pourra disposer d'instructions conditionnelles, de boucles et de fonctions de traitement complexes. Le langage de programmation variera en fonction de la technologie retenue (PHP, java, etc.). On peut reconnaître une page dynamique par son adresse URL :http://www.monsite.be/accueil.jsp?langage=fr : affiche la page accueil.jsp en demandant au serveur d'afficher le contenu de cette page en français.

V. Conclusion :

Dans ce chapitre, un vaste tour d'horizon de différents concepts a permis de cerner plusieurs notions et découvrir autant d'autres. Les réseaux ont été le fer de lance de la recherche vue leur importance sur Internet. L'accent a été mis sur Internet et ses différents services, en particulier le Web.

Nous allons présenter dans le chapitre suivant une étude détaillée sur les deus architecture : l’architecture J2EE et les services web.

Chapitre II

J2EE et Web Service

INTRODUCTION

De nos jours, le génie logiciel nous offre des nouvelles technologies qui ont les potentialités de révolutionner le monde des Conceptions des systèmes informatiques.

Un bon logiciel ne verra jamais le jour sans utiliser une bonne architecture logicielle qui respecte les critères de fiabilité, sécurité, portabilité...

Dans ce chapitre on va étudier deux approches de l'architecture logicielle (J2EE et Service Web), qui sont nécessaires ou bien indispensables pour la réalisation des applications des entreprises.

I. ARCHITECTURE LOGICIELLE [8]

L'architecture logicielle décrit d'une manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions. Contrairement aux spécifications produites par l'analyse fonctionnelle, le modèle d'architecture, produit lors de la phase de conception, ne décrit pas ce que doit réaliser un système informatique mais plutôt comment il doit être conçu de manière à répondre aux spécifications. L'analyse décrit le « quoi faire » alors que l'architecture décrit le « comment le faire ».

I.1. CRITÈRES DE QUALITÉ LOGICIELLE [8]

- ✓ **L'interopérabilité extrinsèque** exprime la capacité du logiciel à communiquer et à utiliser les ressources d'autres logiciels comme, par exemple, les documents créés par une certaine application.
- ✓ **L'interopérabilité intrinsèque** exprime le degré de cohérence entre le fonctionnement des commandes et des modules à l'intérieur d'un système ou d'un logiciel.
- ✓ **La portabilité** exprime la possibilité de compiler le code source et/ou d'exécuter le logiciel sur des plates-formes (machines, systèmes d'exploitation, environnements) différents.
- ✓ **La compatibilité** exprime la possibilité, pour un logiciel, de fonctionner correctement dans un environnement ancien (compatibilité descendante) ou plus récent (compatibilité ascendante).
- ✓ **La validité** exprime la conformité des fonctionnalités du logiciel avec celles décrites dans le cahier des charges.
- ✓ **La vérifiabilité** exprime la simplicité de vérification de la validité.
- ✓ **L'intégrité** exprime la faculté du logiciel à protéger ses fonctions et ses données d'accès non autorisés.

- ✓ **La fiabilité** exprime la faculté du logiciel à gérer correctement ses propres erreurs de fonctionnement en cours d'exécution.
- ✓ **La maintenabilité** exprime la simplicité de correction et de modification du logiciel, et même, parfois, la possibilité de modification du logiciel en cours d'exécution.
- ✓ **La réutilisabilité** exprime la capacité de concevoir le logiciel avec des composants déjà conçus tout en permettant la réutilisation simple de ses propres composants pour le développement d'autres logiciels.
- ✓ **L'extensibilité** exprime la possibilité d'étendre simplement les fonctionnalités d'un logiciel sans compromettre son intégrité et sa fiabilité.
- ✓ **L'efficacité** exprime la capacité du logiciel à exploiter au mieux les ressources offertes par la ou les machines où le logiciel sera implanté.
- ✓ **L'autonomie** exprime la capacité de contrôle de son exécution, de ses données et de ses communications.
- ✓ **La transparence** exprime la capacité pour un logiciel de masquer à l'utilisateur (humain ou machine) les détails inutiles à l'utilisation de ses fonctionnalités.
- ✓ **La composabilité** exprime la capacité pour un logiciel de combiner des informations provenant de sources différentes.
- ✓ **La convivialité** décrit la facilité d'apprentissage et d'utilisation du logiciel par les usagers.

II. PRESENTATION DE TECHNOLOGIE J2EE

II.1. INTRODUCTION À J2EE

J2EE (aujourd'hui appelé JEE – Java Enterprise Edition) est une spécification pour le langage de programmation Java de Sun destinée aux applications d'entreprise. La première version des librairies J2EE a été mise à disposition des développeurs en 1999. J2EE offre une plate-forme de développement et déploiement en langage Java pour les applications distribuées à plusieurs niveaux.

II.2. FONCTIONNEMENT INTERNE [9]

Le langage Java, sur lequel les librairies J2EE 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.

II.3. ARCHITECTURE [10]

J2EE ajout nombreuses couches de niveau entreprise au-dessus de la plate-forme J2SE (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 Servlet Pages) et servlet.
- **Technologie Enterprise 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 entreprise afin de réaliser l'accès et l'échange d'infirmation entre machines et services distribués : JAAS (Java Authentication and Authorization Service), JCA (Java Connector Architecture)

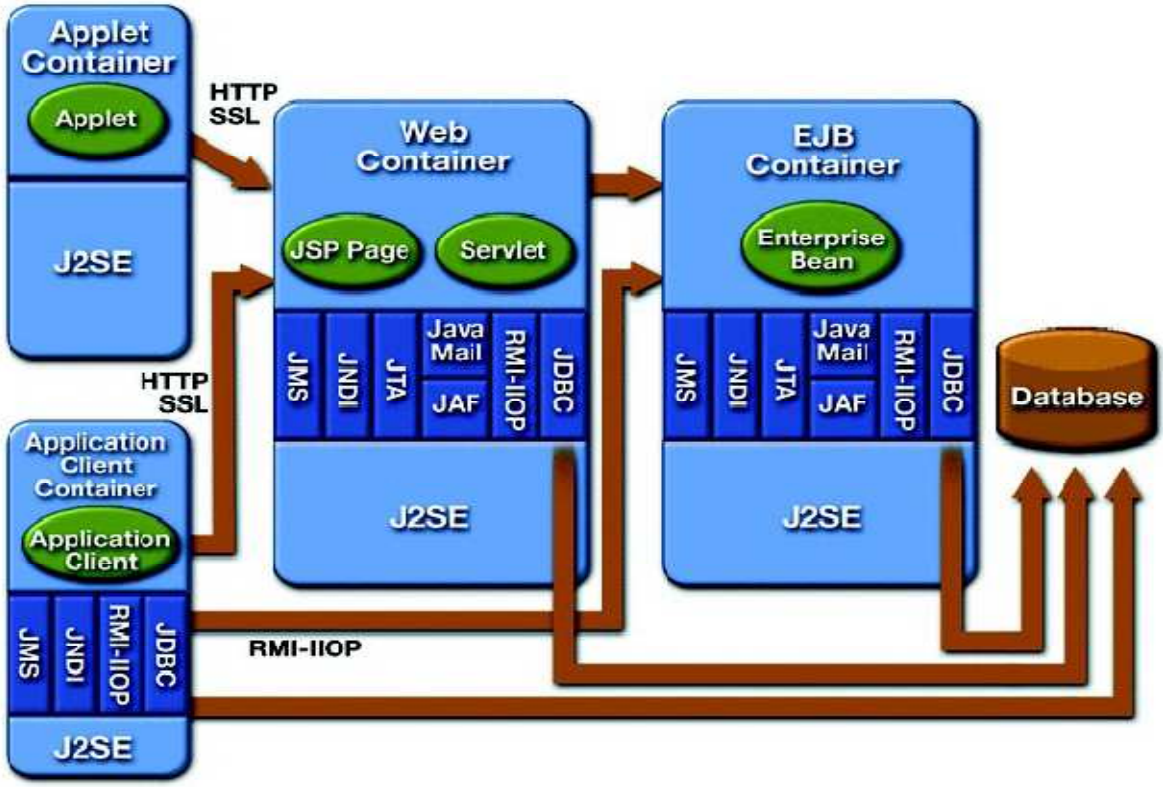


Figure II.1: Architecture et les composants de J2EE.

Pour expliquer l'utilisation de ces technologies on peut imaginer que les technologies entreprise sont utilisées pour gérer l'accès aux données (généralement une ou plusieurs 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.

II.4. LES SERVICES DE JEE [11]

II.4.1. LES SERVICES D'INFRASTRUCTURES :

- **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.
- **JTA**/**JTS** (*Java Transaction API*/*Java Transaction Services*) est un API définissant des interfaces standard avec un gestionnaire de transactions.
- **JCA** (*J2EE Connector Architecture*) est une API de connexion au système d'information de l'entreprise, tels que les ERP.

- **JMX** (*Java Management Extension*) fournit des extensions permettant de développer des applications web de supervision d'applications.

II.4.2. 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 entre objets distants.

II.5. COMPOSANTS DE L'ARCHITECTURE JEE

II.5.1. COMPOSANTS WEB

II.5.1.1. Pages JSPs [11]

Java Server Page est une technologie Java qui permet de générer dynamiquement, aussi bien cotée client que serveur, du code HTML, XML ou autre, donc des pages dont le contenu variera en fonction du contexte.

Page JSP : programme destinée à produire une page HTML.

Page JSP (seconde interprétation) : page HTML produite par l'interprétation d'une page JSP (première interprétations).

Les JSP comme les SERVLET sont considérées comme des composants car leur comportement peut être paramétré via un conteneur.

Une page JSP est conceptuellement substituable à une SERVLET. (Une page JSP est potentiellement compilée en une SERVLET).

Une page JSP se présente comme un fichier contenant de source HTML entrecoupée de code Java intégrée via différentes sortes de balises.

Il existe trois types de tags :

- **tags de directives :** ils permettent de contrôler la structure de la servlet générée.
- **tags de scripting:** ils permettent d'insérer du code Java dans la servlet.
- **tags d'actions:** ils facilitent l'utilisation des composants.

1. Les tags de directives <%@ ... %>

Les directives permettent de préciser des informations globales sur la page JSP. Les spécifications des JSP définissent trois directives :

- **page :** permet de définir des options de configuration.
- **include :** permet d'inclure des fichiers statiques dans la JSP avant la génération de la servlet.
- **taglib** : permet de définir des tags personnalisés.

Leur syntaxe est la suivante :

```
<%@ directive attribut="valeur" ... %>
```

2. LES TAGS DE SCRIPTING

Ces tags permettent d'insérer du code Java qui sera inclus dans la servlet générée à partir de la JSP. Il existe trois tags pour insérer du code Java :

- **le tag de déclaration :** le code Java est inclus dans le corps de la servlet générée. Ce code peut être la déclaration de variables d'instances ou de classes ou la déclaration de méthodes. Sa syntaxe est **<%! declarations %>**
- **le tag d'expression :** évalue une expression et insère le résultat sous forme de chaîne de caractères dans la page web générée. Sa syntaxe est **<%= expression %>**
- **le tag de scriptlets :** par défaut, le code Java est inclus dans la méthode service() de la servlet. Sa syntaxe est la suivante : **<% code Java %>**

3. LES TAGS D'ACTIONS

Les tags d'actions permettent de réaliser des traitements couramment utilisés.

LE TAG <JSP:USEBEAN>

LE TAG <JSP:USEBEAN> PERMET DE LOCALISER UNE INSTANCE OU D'INSTANCIER UN BEAN POUR L'UTILISER DANS LA JSP.

L'utilisation d'un bean dans une JSP est très pratique car il peut encapsuler des traitements complexes et être réutilisable par d'autre JSP ou composants. Le bean peut par exemple assurer l'accès à une base de données. L'utilisation des beans permet de simplifier les traitements inclus dans la JSP.

Lors de l'instanciation d'un bean, on précise la portée du bean. Si le bean demandé est déjà instancié pour la portée précisée alors il n'y pas de nouvelle instance du bean qui est créée mais sa référence est simplement renvoyée : le tag <jsp:useBean> n'instancie donc pas obligatoirement un objet.

Ce tag ne permet pas de traiter directement des EJB.

La syntaxe est la suivante :

```
<jsp:useBean
id="beanInstanceName"
scope="page|request|session|application"
{ class="package.class" |
type="package.class" |
class="package.class" type="package.class" |
beanName="{package.class | <%= expression %>}" type="package.class"
}
{ /> |
> ...
</jsp:useBean>
}
```

LE TAG <JSP:SETPROPERTY >

Le tag <jsp:setProperty> permet de mettre à jour la valeur d'un ou plusieurs attributs d'un Bean. Le tag utilise le setter (méthode setXXX() ou XXX est le nom de la propriété avec la première lettre en majuscule) pour mettre à jour la valeur. Le bean doit exister grâce à un appel au tag <jsp:useBean>.

La syntaxe est la suivante :

```
<jsp:setProperty name="beanInstanceName"
{ property="*" |
property="propertyName" [ param=" parameterName" ] |
property="propertyName" value="{string | <%= expression%>}"
}
/>
```

LE TAG <JSP:GETPROPERTY>

Le tag permet d'obtenir la valeur d'un attribut d'un Bean. Le tag utilise le getter (méthode getXXX() ou XXX est le nom de la propriété avec la première lettre en majuscule) pour obtenir la valeur et l'insérer dans la page HTML généré. Le bean doit exister grâce à un appel au tag <jsp:useBean>.

La syntaxe est la suivante :

```
<jsp:getProperty name="beanInstanceName" property=" propertyName" />
```

LE TAG DE REDIRECTION <JSP:FORWARD>

Ce tag permet de rediriger la requête vers une autre URL pointant vers un fichier HTML, JSP ou une servlet. Dès que le moteur de JSP rencontre ce tag, il redirige la requête vers l'URL précisée et ignore le reste de la JSP courante. Tout ce qui a été généré par la JSP est perdu.

La syntaxe est la suivante :

```
<jsp:forward page="{relativeURL | <%= expression %>}" />
ou
<jsp:forward page="{relativeURL | <%= expression %>}" >
<jsp:param name="parameterName" value="{ parameterValue | <%= expression %>}" /> +
</jsp:forward>
```

LE TAG <JSP:INCLUDE>

Ce tag permet d'inclure le contenu généré par une JSP ou une servlet dynamiquement au moment où la JSP est exécutée. C'est la différence avec la directive include avec laquelle le fichier est inséré dans la JSP avant la génération de la servlet.

La syntaxe est la suivante :

```
<jsp:include page="relativeURL" flush="true" />
```

LE TAG <JSP:PLUGIN>

Ce tag permet la génération du code HTML nécessaire à l'exécution d'une applet en fonction du navigateur : un tag HTML <Object> ou <Embed> est généré en fonction de l'attribut User-Agent de la requête.

Variables prédéfinies JSP

– Request (HttpServletRequest)

– response (HttpServletResponse)

– out (PrintWriter) utilisée pour écrire dans la page réponse.

– session (HttpSession) la session (si elle existe) associée à la requête

– application (ServletContext) identique `a getServletConfig().getContext().

– config (ServletConfig)

– page (this)

– pageContext. Acc`es aux classes JSP spéecifiques (javax.servlet.jsp)

Exemple JSP

```
<%@ page import="java.util.Date"%>
<html>
<body>
<%! Date dateDuJour; %>
<% dateDuJour = new Date();%>
Date du jour : <%= dateDuJour %><BR>
</body>
</html>
```

II.5.1.2. LES SERVLETS [12]

Une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont le plus généralement présentées au format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs web. Les servlets utilisent l'API Java Servlet (package `javax.servlet`).

Une servlet s'exécute dynamiquement sur le serveur web et permet l'extension des fonctions de ce dernier, typiquement : accès à des bases de données, transactions d'e-commerce, etc. Une servlet peut être chargée automatiquement lors du démarrage du serveur web ou lors de la première requête du client. Une fois chargées, les servlets restent actives dans l'attente d'autres requêtes du client.

L'utilisation de servlets se fait par le biais d'un conteneur de servlets (framework) côté serveur. Celui-ci constitue l'environnement d'exécution de la servlet et lui permet de persister entre les requêtes des clients. L'API définit les relations entre le conteneur et la servlet. Le conteneur reçoit la requête du client, et sélectionne la servlet qui aura à la traiter. Le conteneur fournit également tout un ensemble de services standards pour simplifier la gestion des requêtes et des sessions.

1. Le fonctionnement d'une servlet (cas d'utilisation de http)

Un serveur d'application permet de charger et d'exécuter les servlets dans une JVM. C'est une extension du serveur web. Ce serveur d'application contient entre autre un moteur de servlets qui se charge de manager les servlets qu'il contient.

Pour exécuter une servlet, il suffit de saisir une URL qui désigne la servlet dans un navigateur.

1. Le serveur reçoit la requête http qui nécessite une servlet de la part du navigateur si c'est la première sollicitation de la servlet, le serveur l'instancie. Les servlets sont stockées (sous forme de fichier .class) dans un répertoire particulier du serveur. Ce répertoire dépend du serveur d'application utilisé. La servlet reste en mémoire jusqu'à l'arrêt du serveur. Certains serveurs d'application permettent aussi d'instancier des servlets dès le lancement du serveur.

2. La servlet en mémoire, peut être appelée par plusieurs threads lancés par le serveur pour chaque requête. Ce principe de fonctionnement évite d'instancier un objet de type servlet à chaque requête et permet de maintenir un ensemble de ressources actives tel qu'une connexion à une base de données.

2. L'API servlet

Les servlets sont conçues pour agir selon un modèle de requête/réponse. Tous les protocoles utilisant ce modèle peuvent être utilisé tel que http, ftp, etc ...

L'API servlets est une extension du jdk de base, et en tant que tel elle est regroupée dans des packages préfixés par javax.

L'API servlet regroupe un ensemble de classes dans deux packages :

- *javax.servlet :* contient les classes pour développer des serlvets génériques indépendantes d'un protocole.
- *javax.servlet.http :* contient les classes pour développer des servlets qui reposent sur le protocole http utilisé par les serveurs web.

3. Exemple d'une servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWWW extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.println("<HTML>");
```

```
out.println("<HEAD><TITLE>Hello WWW</TITLE></HEAD>");
out.println("<BODY>");
out.println("<H1>Hello WWW</H1>"); // Pas de séparation logique de l'application / présentation
out.println("</BODY></HTML>");
}
}
```

II.5.1.3. CONTENEUR DE COMPOSANTS WEB

Les composants web sont hébergés dans des conteneurs de servlets, conteneurs de JSP et conteneurs web.

En sus des fonctionnalités normales d'un conteneur de composants, un conteneur de servlets (servlets container) fournit les services réseaux par lesquels les requêtes et réponses sont émises. Il décode également les requêtes et formate les réponses dans le format approprié.

Tous les conteneurs de servlets doivent supporter le protocole HTTP et peuvent aussi supporter le protocole HTTPS.

Un conteneur de JSP (JSP container) fournit les mêmes services qu'un conteneur de servlets.

Ces conteneurs sont généralement appelés conteneurs web (Web containers).

II.5.2. LES COMPOSANTS EJB [12]

La technologie EJB définit un modèle pour le développement et le déploiement de composants de serveur Java réutilisables également appelés **composants EJB**.

Un composant EJB est un composant de serveur non visuel dont les méthodes fournissent généralement une logique de gestion dans des applications distribuées. Un client distant, appelé client EJB, peut appeler ces méthodes qui sont généralement à l'origine de la mise à jour de la base de données.

L'architecture EJB se présente de la façon suivante :

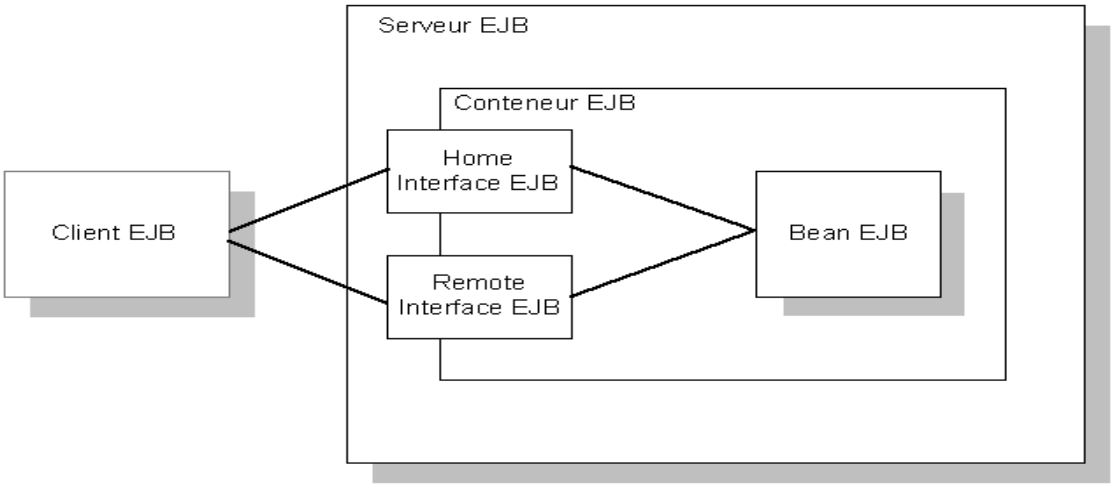


FIGURE II.2: L'ARCHITECTURE GÉNÉRALE D'UN EJB.

II.5.2.1. SERVEUR EJB [13]

EJB Server intègre le conteneur EJB assurant les services requis par le composant EJB.

II.5.2.2. CLIENT EJB

Le client EJB fournit généralement la logique de l'interface utilisateur sur la machine cliente. Il passe les appels aux composants EJB distants hébergés sur un serveur et doit savoir comment trouver le serveur 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.

Un client EJB ne communique pas directement avec un composant EJB.

Le conteneur fournit les objets proxy mettant en œuvre les Home Interface et Remote Interface des composants. Le **Remote Interface** des composants définit les méthodes de gestion pouvant être appelées par le client. Ce dernier appelle les méthodes de l'**Home Interface** pour créer et détruire les proxys pour le Remote Interface.

L'INTERFACE REMOTE

L'interface remote permet de définir les méthodes qui contiendront les traitements proposés par le bean. Cette interface doit étendre l'interface javax.ejb.EJBObject.

```
package com.jmdoudoux.ejb;

import java.rmi.RemoteException;
import javax.ejb.EJBObject;

public interface MonPremierEJB extends EJBObject {
  public String message() throws RemoteException;
}
```

L'INTERFACE HOME

L'INTERFACE HOME PERMET DE DÉFINIR DES MÉTHODES QUI VONT GÉRER LE CYCLE DE VIE DU BEAN. CETTE INTERFACE DOIT ÉTENDRE L'INTERFACE EJBHOME.

```
package com.jmdoudoux.ejb;

import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

public interface MonPremierEJBHome extends EJBHome {
  public MonPremierEJB create() throws CreateException, RemoteException;
}
```

LES DIFFÉRENTS TYPES D'EJB [13]

- **Les beans sessions :** Les beans sessions comme leur nom l'indique, ont une durée de vie correspondant à celle de la "conversation" ou "session" entre l'application cliente et le composant. Ces beans servent à fournir à l'application cliente divers services conçus par le développeur. Selon le choix de celui-ci, un bean session peut maintenir un état pendant toute la durée de la session (c'est-à-dire conserver l'état des attributs internes aux objets de façon à maintenir la conversation avec le client) ou au contraire sans état (stateless), ce qui signifie qu'il fournit un accès à des méthodes implémentant la logique applicative (comme RMI), mais ne conserve aucun résultat auquel le client pourrait faire référence ultérieurement.
- **Les beans entités :** Les beans entités représentent des objets métier du domaine de l'application tels que clients, factures, produits, etc. Ces objets persistent de façon à pouvoir être réutilisés. Le conteneur de l'architecture J2EE s'occupe de tous les détails de cette tâche de persistance. Rappelons que la persistance correspond à l'utilisation d'une base de données qui stocke la valeur des attributs de chacun de ces beans entités. Avec les beans entités, il n'est pas du tout nécessaire de maîtriser le langage SQL ainsi que la connectivité JDBC. De fait, la base de données de type relationnelle devient une base de données de type objet. Ce type de bean est vraiment intéressant puisque sans cette technologie, le développeur passe généralement beaucoup de temps à la gestion de la base de données. Avec un bean entité, le développeur ne voit pas du tout la base de données et donc ne s'en occupe pas ; il peut alors passer tout son temps sur l'application elle-même.
- **Les beans contrôlés par messages :** Le troisième type d'EJB, le bean contrôlé par message, fournit un modèle de composant permettant d'écouter un service de messages. La plateforme J2EE définit une queue de message, qui est une sorte de file d'attente dans laquelle les applications peuvent placer des messages. Elles peuvent également s'abonner à une queue de messages. L'avantage de cette architecture est que les composants qui utilisent les messages n'ont pas besoin

de savoir qui les a envoyés. Il leur suffit d'identifier la queue contenant les messages. C'est une sorte de flot d'informations que les composants ont en commun sans que ces dernières soient dédiées à un composant en particulier. Un système de gestion de portefeuilles d'actions est un exemple d'utilisation d'une telle queue. Les cours des actions sont envoyés à une queue sous forme de message, qui sont consommés par les composants qui ont besoin de connaître ces cours. Avec les beans contrôlés par messages, il est possible de créer des composants répondant aux messages concernant les cours en prenant automatiquement certaines décisions en fonction de leurs variations.

EXEMPLE EJB

```
package com.jmdoudoux.ejb;

import java.rmi.RemoteException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
public class MonPremierEJBBean implements SessionBean {

  public String message() {
    return "Bonjour";
  }

  public void ejbActivate() {
  }

  public void ejbPassivate() {
  }

  public void ejbRemove() {
  }

  public void setSessionContext(SessionContext arg0) throws EJBException,
RemoteException {
  }

  public void ejbCreate() {
  }
}
```

II.5.2.3. CONTENEUR EJB [13]

La spécification EJB définit un conteneur comme étant l'environnement dans lequel un ou plusieurs composants EJB s'exécutent. Le conteneur fournit l'infrastructure requise pour exécuter les composants distribués, permettant ainsi aux développeurs de clients et de composants de se concentrer sur la programmation de la logique de gestion et non sur le code niveau système. Sous EJB Server, le conteneur encapsule :

- ✓ la version d'exécution cliente et les classes de stubs générées ce qui permet aux clients d'exécuter des composants sur un serveur distant comme s'il s'agissait d'objets locaux.
- ✓ le service de nommage ce qui permet aux clients d'instancier les composants par leur nom et aux composants d'obtenir des ressources telles que les connexions aux bases de données par nom.
- ✓ le distributeur de composants EJB Server qui exécute la classe d'implémentation des composants et fournit des services tels que la gestion des transactions, le pool de connexions aux bases de données et la gestion du cycle de vie des instances.

II.5.2.4. LE DÉPLOIEMENT DES EJB

Un EJB doit être déployé sous forme d'une archive jar qui doit contenir un fichier qui est le descripteur de déploiement et toutes les classes qui composent chaque EJB (interfaces home et remote, les classes qui implémentent ces interfaces et toutes les autres classes nécessaires aux EJB).
Une archive ne doit contenir qu'un seul descripteur de déploiement pour tous les EJB de l'archive. Ce fichier au format XML doit obligatoirement être nommé ejb-jar.xml.
L'archive doit contenir un répertoire META-INF (attention au respect de la casse) qui contiendra lui même le descripteur de déploiement.
Le reste de l'archive doit contenir les fichiers .class avec toute l'arborescence des répertoires des packages. Le jar des EJB peut être inclus dans un fichier de type EAR.

II.6. TECHNOLOGIE J2EE

Le tableau suivant résume les APIs accessibles en fonction du type de conteneur.

API	Applet	Application Client	Web	EJB
JDBC 2.0 Extension	N	Y	Y	Y
JTA 1.0	N	N	Y	Y
JNDI 1.2	N	Y	Y	Y
Servlet 2.3	N	N	Y	N
JSP 1.2	N	N	Y	N
EJB 2.0	N	Y1	Y2	Y
RMI-IIOP 1.0	N	Y	Y	Y
JMS 1.0	N	Y	Y	Y
JavaMail 1.2	N	N	Y	Y
JAF 1.0	N	N	Y	Y
JAXP 1.1	N	Y	Y	Y
JAAS 1.0	N	Y	Y	Y
Connector 1.0	N	N	Y	Y

FIGURE II.3 : LES TECHNOLOGIES J2EE EN FONCTION DE TYPE DE CONTENEUR.

II.6.1. JAVA NAME AND DIRECTORY INTERFACE

JNDI est l'acronyme de Java Naming and Directory Interface. Cet API fournit une interface unique pour utiliser les différents services de nommages ou d'annuaires:

- LDAP (Lightweigth Directory Access Protocol)
- DNS (Domain Naming Service)
- NIS (Network Information Service) de SUN
- Service de nommage CORBA
- Service de nommage RMI

Un service de nommage permet d'associer un nom unique à un objet et faciliter ainsi l'obtention de cet objet. Un annuaire est un service de nommage qui possède en plus une représentation hiérarchique des objets qu'il contient et un mécanisme de recherche.

Pour pouvoir utiliser autant de services différents possédant des protocoles d'accès différent, JNDI utilise des pilotes SPI (Service Provider). Trois de ces pilotes sont fournis en standard :

- ✓ LDAP (Lightweigth Directory Access Protocol)
- ✓ service de nommage CORBA (COS)
- ✓ service de nommage RMI

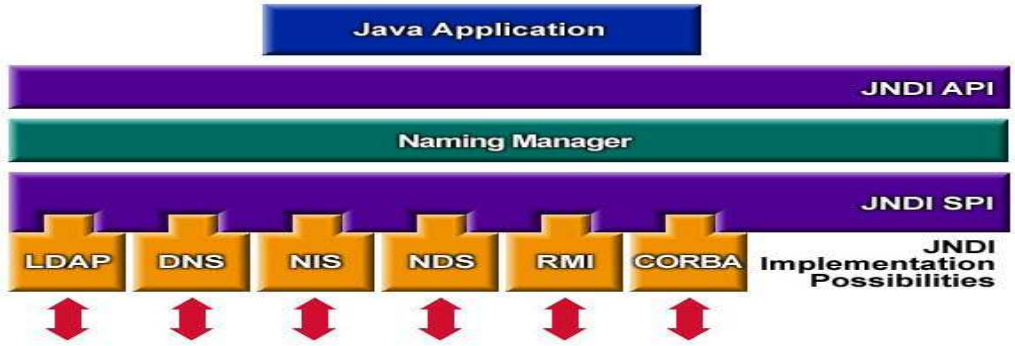


Figure II.4 : l'architecture de la technologie JNDI.

La technologie JNDI est utilisée dans J2EE pour localiser les objets sur un serveur et accéder aux objets externes à partir des composants J2EE.

Chaque conteneur stocke une référence aux objets qu'il peut créer et instancie ces objets à la demande des clients ou des applications qui fonctionnent sur le serveur.

Le conteneur met aussi à la disposition des composants un jeu de ressources JNDI initial, issu de la configuration du serveur et/ou des applications web (via les descripteurs de déploiement).

Un objet InitialContext est crée par le conteneur lorsqu'une application web est déployée. Cet objet est accessible par les composants, en lecture seulement.

L'autre rôle de JNDI dans une application J2EE est la localisation des interfaces distantes des beans.

Exemple d'utilisation de JNDI

Exemple d'accès à une ressource JDBC par JNDI :

```
// Obtain our environment naming context
Context initCtx = new InitialContext();
Context envCtx = (Context) initCtx.lookup("java:comp/env");
// Look up our data source

DataSource ds = (DataSource)

  envCtx.lookup("jdbc/EmployeeDB");

// Allocate and use a connection from the pool

Connection conn = ds.getConnection();

... use this connection to access the database ...

conn.close();
```

II.6.2. JDBC

JDBC est une API Java (ensemble de classes et d'interfaces défini par SUN et les acteurs du domaine des BD) permettant d'accéder aux bases de données à l'aide du langage Java via des requêtes SQL. Cette API permet d'atteindre de manière quasi-transparente des bases Sybase, Oracle, Informix, ... avec le même programme Java JDBC.

En fait cette API est une spécification de ce que doit implanter un constructeur de BD pour que celle ci soit interrogeable par JDBC. De ce fait dans la programmation JDBC on utilise essentiellement des références d'interface (Connection, Statement, ResultSet, ...).

Structure d'un programme JDBC

Un code JDBC est de la forme :

- recherche et chargement du driver approprié à la BD.
- établissement de la connexion à la base de données.
- construction de la requête SQL
- envoi de cette requête et récupération des réponses.
-parcours des réponses

Syntaxe

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection conX = DriverManager.getConnection(...);
```

```
Statement stmt = conX.createStatement();
ResultSet rs = stmt.executeQuery("SELECT a, b, c ... FROM ...
WHERE ...");
while (rs.next()) {
...
// traitement
}
```

II.6.3. Architecture RMI [14]

Le but de RMI (Remote Method Invocation) est de permettre l'appel, l'exécution et le renvoi du résultat d'une méthode exécuté dans une machine virtuelle différente de celle de l'objet l'appelant. Cette machine virtuelle peut être sur une machine différente pourvu qu'elle soit accessible par le réseau.

La machine sur laquelle s'exécute la méthode distante est appelée serveur.

L'appel coté client d'une telle méthode est un peu plus compliqué que l'appel d'une méthode d'un objet local mais il reste simple. Il consiste à obtenir une référence sur l'objet distant puis à simplement appeler la méthode à partir de cette référence.

La technologie RMI se charge de rendre transparente la localisation de l'objet distant, son appel et le renvoi du résultat. En fait, elle utilise deux classes particulières, le **stub** et le **skeleton**, qui doivent être générées avec l'outil **rmic** fourni avec le JDK.

Le **stub** est une classe qui se situe côté client et le **skeleton** est son homologue coté serveur. Ces deux classes se chargent d'assurer tous les mécanismes d'appel, de communication, d'exécution, de renvoie et de réception du résultat.

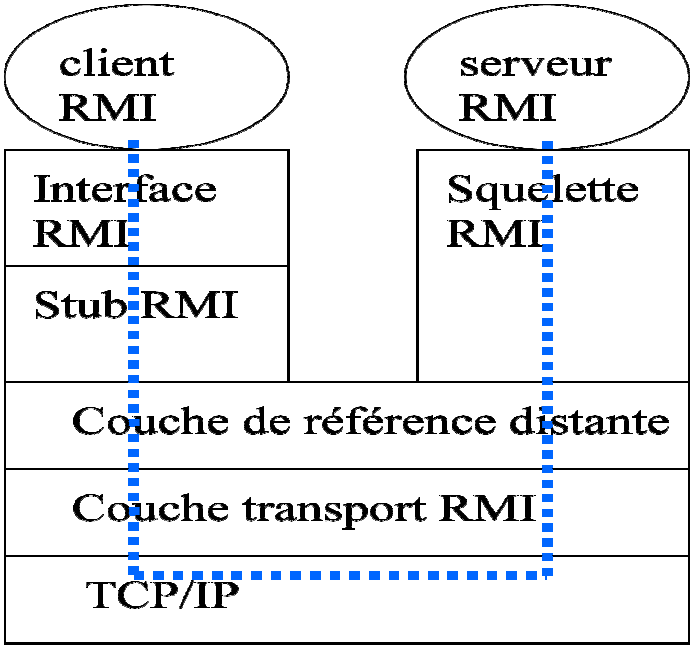


Figure II.5: Présentation de l'architecture RMI.

Les amorces (Stub/Skeleton)

- ✓ Elles assurent le rôle d'adaptateurs pour le transport des appels distants.
- ✓ Elles réalisent les appels sur la couche réseau.
- ✓ Elles réalisent l'assemblage et le désassemblage des paramètres (*marshalling, unmarshalling*).
- ✓ Une référence d'objets distribués correspond à une référence d'amorce.
- ✓ Les amorces sont créées par le générateur **rmic.**

Les Stubs

- ✓ Représentants locaux de l'objet distribué .
- ✓ Initient une connexion avec la JVM distante en transmettant l'invocation distante à la couche des références d'objets.
- ✓ Assemblent les paramètres pour leur transfert à la JVM distante.
- ✓ Attendent les résultats de l'invocation distante.
- ✓ Désassemblent la valeur ou l'exception renvoyée.
- ✓ Renvoient la valeur à l'appelant.
- ✓ S'appuient sur la sérialisation.

Les squelettes

- ✓ Désassemblent les paramètres pour la méthode distante.
- ✓ Font appel à la méthode demandée.
- ✓ Assemblage du résultat (valeur renvoyée ou exception) à destination de l'appelant.

La couche des références d'objets Remote Reference Layer

- ✓ Permet d'obtenir une référence d'objet distribué à partir de la référence locale au stub.
- ✓ Cette fonction est assurée grâce à un service de noms **rmiregister** (qui possède une table de hachage dont les clés sont des noms et les valeurs sont des objets distants).
- ✓ Un unique **rmiregister** par JVM.
- ✓ **rmiregister** s'exécute sur chaque machine hébergeant des objets distants.
- ✓ **rmiregister** accepte des demandes de service sur le port 1099.

La couche transport

- ✓ réalise les connexions réseau basées sur les flux entre les JVM
- ✓ emploie un protocole de communication propriétaire (**JRMP**: Java Remote Method Invocation) basé sur TCP/IP
- ✓ Le protocole JRMP a été modifié afin de supprimer la nécessité des squelettes car depuis la version 1.2 de Java, une même classe skeleton générique est partagée par tous les objets distants.

Les différentes étapes pour créer un objet distant et l'appeler avec RMI

Le développement coté serveur se compose de :

- ✓ La définition d'une interface qui contient les méthodes qui peuvent être appelées à distance.
- ✓ L'écriture d'une classe qui implémente cette interface.
- ✓ L'écriture d'une classe qui instanciera l'objet et l'enregistrera en lui affectant un nom dans le registre de nom RMI (RMI Registry).

Le développement côté client se compose de :

- ✓ L'obtention d'une référence sur l'objet distant à partir de son nom.
- ✓ L'appel à la méthode à partir de cette référence.

Enfin, il faut générer les classes stub et skeleton en exécutant le programme rmic avec le fichier source de l'objet distant.

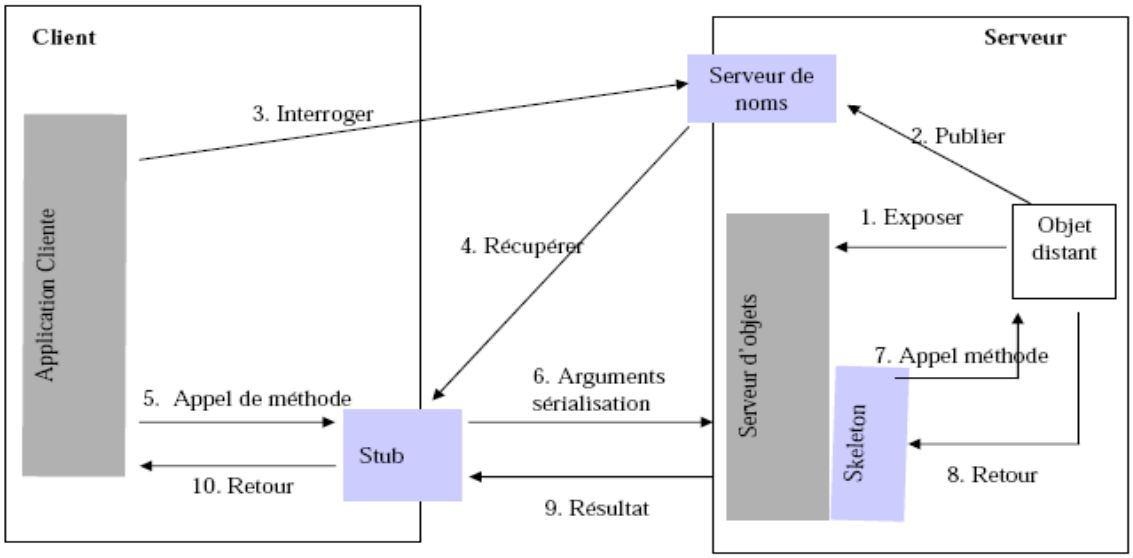


Figure II.6: Etapes d'un appel de méthode distante.

Exemple d'utilisation de RMI

- **Le développement coté client**
  - ✓ La définition d'une interface qui contient les méthodes de l'objet distant

```
package test_rmi;
import java.rmi.*;
public interface Information extends Remote {
public String getInformation() throws RemoteException;
}
```

  - ✓ L'écriture d'une classe qui implémente cette interface

```
package test_rmi;
import java.rmi.*;
import java.rmi.server.*;
public class TestRMIServer extends UnicastRemoteObject implements Information {
protected TestRMIServer() throws RemoteException {
super();
}
public String getInformation()throws RemoteException {
return "bonjour";
}
}
```

  - ✓ L'écriture d'une classe pour instancier l'objet et l'enregistrer dans le registre

```
public static void main(String[] args) {
try {
System.out.println("Mise en place du Security Manager ...");
System.setSecurityManager(new java.rmi.RMISecurityManager());
TestRMIServer testRMIServer = new TestRMIServer();
```

```
} catch (Exception e) {
System.out.println("Exception capturée: " + e.getMessage());
}
}
```

- ✓ L'enregistrement dans le registre de nom RMI en lui donnant un nom

```
public static void main(String[] args) {
try {
System.out.println("Mise en place du Security Manager ...");
System.setSecurityManager(new java.rmi.RMISecurityManager());
TestRMIServer testRMIServer = new TestRMIServer();
System.out.println("Enregistrement du serveur");
Naming.rebind("rmi://"+java.net.InetAddress.getLocalHost()+
"/TestRMI",testRMIServer);
// Naming.rebind(";rmi://localhost/TestRMI", testRMIServer);
System.out.println("Serveur lancé");
} catch (Exception e) {
System.out.println("Exception capturée: " + e.getMessage());
}
}
```

- **Le développement coté client**
  - ✓ L'appel à la méthode à partir de la référence sur l'objet distant

```
public static void main(String[] args) {
System.setSecurityManager(new RMISecurityManager());
try {
Remote r = Naming.lookup("rmi://vaio/127.0.0.1/TestRMI");
if (r instanceof Information) {
String s = ((Information) r).getInformation();
System.out.println("chaine renvoyée = " + s);
}
} catch (Exception e) {
}
}
```

III. PRESENTATION DE TECHNOLOGIE WEB SERVICE

III.1. Introduction

Le Web actuel est un immense espace d'informations, dans lequel l'individu aura certainement besoin des machines pour l'aider à les retrouver ; or la structure actuelle du Web ne

facilite pas cette tâche, étant donné qu'elle favorise plus la compréhension humaine par rapport à celui des machines.

Le Web contient des vastes bases de données avec plusieurs buts et de multiples sources non homogènes. Ceci prouve qu'il est nécessaire d'améliorer l'accessibilité à cette importante masse d'informations, et de disposer d'outils plus sophistiqués pour une meilleure recherche et organisation au sein du Web.

Les services Web fournissent une nouvelle manière de développer des applications conformes aux besoins de l'Internet, et ils semblent être la solution la plus adaptée pour assurer l'interopérabilité, qui permet de transmettre les données entre les différentes applications d'une organisation comme l'entreprise, la société ou l'individu ; ainsi la technologie des services Web permette de réaliser le traitement de ces données, et gérer les liaisons entre les différentes applications.

L'approche ultime de cette vision consiste donc à créer des applications constituées uniquement de services Web qui interagissent entre eux. Dans ce cas de figure, peu importe où est déployé le service Web, ce qui importe est que le service remplisse un rôle bien précis.

III.2. Définition [15]

Les **services web** (en anglais *web services*) représentent un mécanisme de communication entre applications distantes à travers le réseau internet indépendant de tout langage de programmation et de toute plate-forme d'exécution :

- utilisant le protocole HTTP comme moyen de transport. Ainsi, les communications s'effectuent sur un support universel, maîtrisé et généralement non filtré par les pare-feux ;
- employant une syntaxe basée sur la notation XML pour décrire les appels de fonctions distantes et les données échangées ;
- organisant les mécanismes d'appel et de réponse.

Grâce aux services web, les applications peuvent être vues comme un ensemble de services métiers, structurés et correctement décrits, dialoguant selon un standard international plutôt qu'un ensemble d'objets et de méthodes entremêlés.

Le premier bénéfice de ce découpage est la facilité de maintenance de l'application, ainsi que l'interopérabilité permettant de modifier facilement un composant (un service) pour le remplacer par un autre, éventuellement développé par un tiers. Qui plus est, les services web permettent de réduire la complexité d'une application car le développeur peut se focaliser sur un service, indépendamment du reste de l'application.

Les services web facilitent non seulement les échanges entre les applications de l'entreprise mais surtout permettent une ouverture vers les autres entreprises. Les premiers fournisseurs de services

web sont ainsi les fournisseurs de services en ligne (météo, bourse, planification d'itinéraire, pages jaunes, etc.), mettant à disposition des développeurs des API (Application Programmable Interface) payantes ou non, permettant d'intégrer leur service au sein d'applications tierces.

III.3. Les caractéristiques d'un service Web [16]

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web. Un service Web possède les caractéristiques suivantes :

- il est accessible via le réseau.
- il dispose d'une interface publique (ensemble d'opérations) décrite en XML.
- ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire.
- il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...).
- l'intégration d'application en implémentant des services Web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur. Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

III.4. ARCHITECTURE D'UN SERVICE WEB

Les services Web reprennent la plupart des idées et des principes du Web (HTTP, XML), et les appliquent à des interactions entre machines. Comme pour le **World Wide Web**, les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont HTTP, WSDL, REST, XML-RPC, SOAP et UDDI.

REST

REST (*Representational State Transfer*) est une architecture de services Web. Élaborée en l'an 2000 par *Roy Fiedling*, l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres

travaux fondamentaux, REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

XML-RPC

XML-RPC EST UN PROTOCOLE SIMPLE UTILISANT XML POUR EFFECTUER DES MESSAGES RPC. LES REQUÊTES SONT ÉCRITES EN XML ET ENVOYÉES VIA HTTP POST. LES REQUÊTES SONT INTÉGRÉES DANS LE CORPS DE LA RÉPONSE HTTP. XML-RPC EST INDÉPENDANT DE LA PLATE-FORME, CE QUI LUI PERMET DE COMMUNIQUER AVEC DIVERSES APPLICATIONS. PAR EXEMPLE, UN CLIENT JAVA PEUT PARLER DE XML-RPC À UN PERLSERVER !

SOAP

SOAP (*Simple object Access Protocol*) est un protocole standard de communication. C'est l'épine dorsale du système d'interopérabilité. SOAP est un protocole décrit en XML et standardisé par le W3C. Il se présente comme une enveloppe pouvant être signée et pouvant contenir des données ou des pièces jointes. Il circule sur le protocole HTTP et permet d'effectuer des appels de méthodes à distance.

WSDL

WSDL (*Web Services Description Language*) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées.

UDDI

UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION) EST UN ANNUAIRE DE SERVICES. IL FOURNIT L'INFRASTRUCTURE DE BASE POUR LA PUBLICATION ET LA DÉCOUVERTE DES SERVICES WEB. UDDI PERMET AUX FOURNISSEURS DE PRÉSENTER LEURS SERVICES WEB AUX CLIENTS.
LES INFORMATIONS QU'IL CONTIENT PEUVENT ÊTRE SÉPARÉES EN TROIS TYPES :

- les pages blanches qui incluent l'adresse, le contact et les identifiants relatifs au service Web.
- les pages jaunes qui identifient les secteurs d'affaires relatifs au service Web.
- les pages vertes qui donnent les informations techniques.

III.5. Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant :

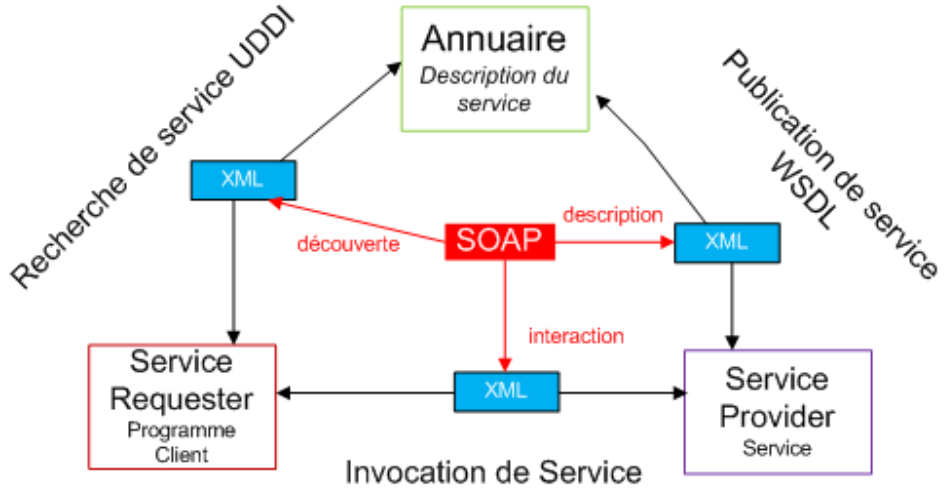


FIGURE II.7: FONCTIONNEMENT D'UN SERVICE WEB.

III.5.1. SERVICE PROVIDER SERVICE

Le fournisseur de service met en application le service Web et le rend disponible sur Internet.

III.5.2. SERVICE REQUESTER PROGRAMME CLIENT

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP).

III.5.3. ANNUAIRE SERVICE REGISTRY

Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver. Les interactions entre ces trois acteurs suivent plusieurs étapes :

- **La publication du service** : le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
- **La recherche du service** : le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.

- **L'invocation du service** : une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

III.6. LE PROTOCOLE DE COMMUNICATION SOAP

Nous avons proposé une définition de trois architectures : SOAP, son ancêtre XML-RPC et REST. Nous verrons celle de SOAP plus en détail, car elle est de nos jours la plus implémentée. SOAP est un protocole d'invocation de méthodes sur des services distants. **Basé sur XML**, SOAP a pour principal objectif **d'assurer la communication entre machines**. Le protocole permet d'appeler une méthode RPC et d'envoyer des messages aux machines distantes via HTTP. Ce protocole est très bien adapté à l'utilisation des services Web, car il permet de fournir au client une grande quantité d'informations récupérées sur un réseau de serveurs tiers, voyez :

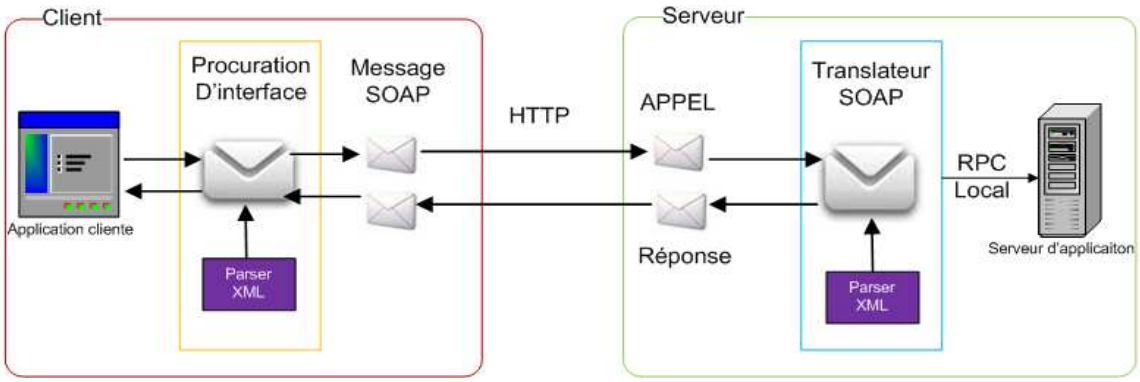


Figure II.8: le protocole de communication SOAP.

SOAP est bien plus populaire et utilisé que XML-RPC. C'est une recommandation du W3C. D'après cette recommandation, SOAP est destiné à être un protocole léger dont le but est d'échanger des informations structurées dans un environnement décentralisé et distribué. Une des volontés du W3C vis-à-vis de SOAP est de ne pas réinventer une nouvelle technologie. SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies existantes.

III.6.1. Structure d'un message SOAP

La grammaire de SOAP est assez simple à comprendre. Elle procure un moyen d'accès aux objets par appel de méthodes à distance. Les deux plus fortes fonctionnalités de SOAP sont sa simplicité et le fait que tout le monde a accepté de l'utiliser. Un message SOAP est composé de deux parties obligatoires : l'enveloppe SOAP et le corps SOAP ; et une partie optionnelle : l'en-tête SOAP.

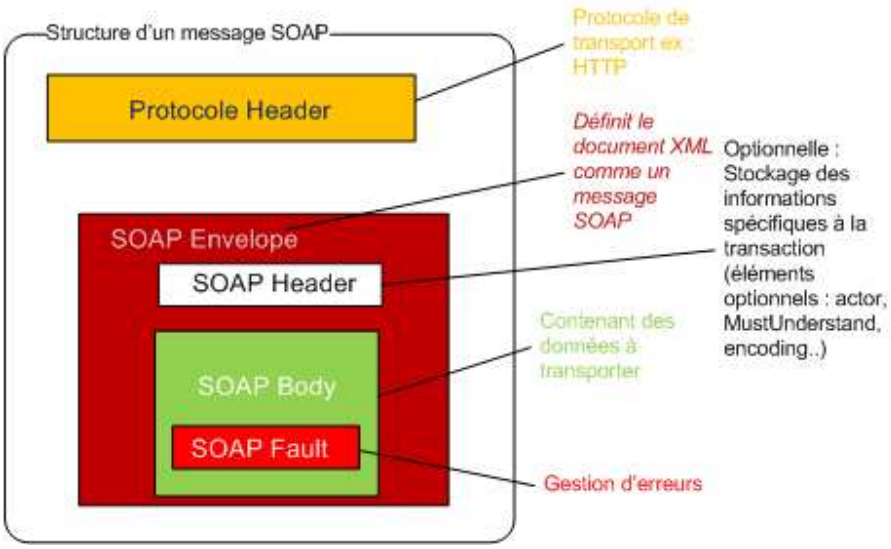


Figure II.9: Structure de message SOAP.

- **SOAP envelope** (enveloppe) est l'élément de base du message SOAP. L'enveloppe contient la spécification des espaces de désignation (namespace) et du codage de données.
- **SOAP header** (entête) est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent. C'est ici qu'il est indiqué si le message est mandataire ou optionnel. L'entête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.
- **SOAP body** (corps) est un *container* pour les informations mandataires à l'intention du récepteur du message, il contient les méthodes et les paramètres qui seront exécutés par le destinataire final.
- **SOAP fault** (erreur) est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs.

III.6.2. Exemple de message SOAP

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
soap:encodingStyle="http://schemas.xmlsoap.org/
soap/
encoding/">
<soap:Body>
<soap:Fault>
<faultcode>soap:MustUnderstand</faultcode>
<faultstring>Mandatory Header error.</faultstring>
<faultactor>http://www.wrox.com/heroes/endpoint.asp</
faultactor>
<detail>
<w:source xmlns:w="http://www.wrox.com/">
```

```
<module>endpoint.asp</module>
<line>203</line>
</w:source>
</detail>
</soap:Fault>
</soap:Body>

</soap:Envelope
```

II.10. Le langage de description WSDL

Un document WSDL se compose d'un ensemble d'éléments décrivant les types de données utilisés par le service, les messages que le service peut recevoir, ainsi que les liaisons SOAP associées à chaque message. Le schéma suivant illustre la structure du langage WSDL qui est un document XML, en décrivant les relations entre les sections constituant un document WSDL.

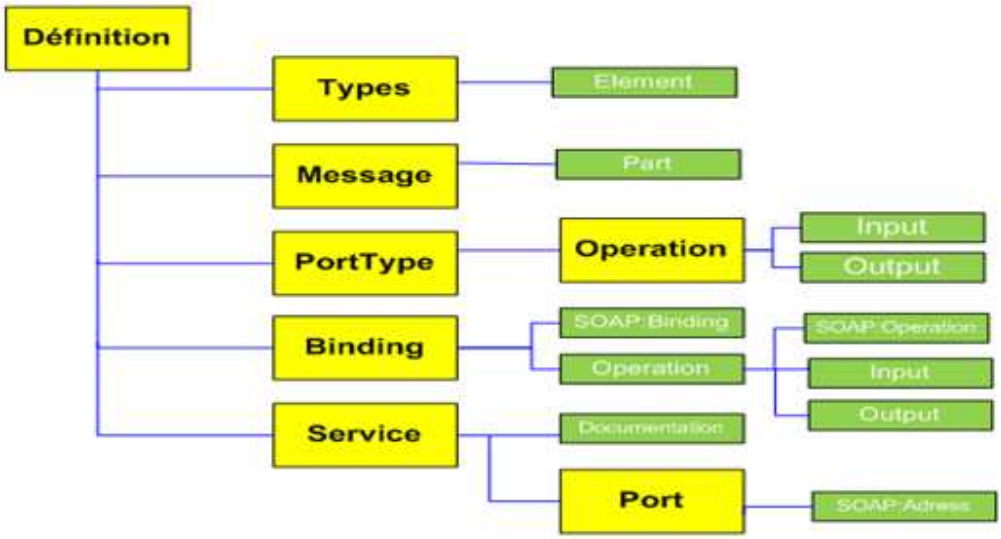


Figure II.11: Structure de document WSDL

Un fichier WSDL contient donc sept éléments.

- **Types** : fournit la définition de types de données utilisés pour décrire les messages échangés.
- **Messages** : représente une définition abstraire (noms et types) des données en cours de transmission.
- **PortTypes** : décrit un ensemble d'opérations. Chaque opération a zéro ou un message en entrée, zéro ou plusieurs messages de sortie ou d'erreurs.
- **Binding** : spécifie une liaison entre un <portType> et un protocole concret (SOAP, HTTP...).
- **Service** : indique les adresses de port de chaque liaison.

- **Port** : représente un point d'accès de services défini par une adresse réseau et une liaison.
- **Opération** : c'est la description d'une action exposée dans le port.

Le document WSDL peut être divisé en deux parties. Une partie pour les **définitions abstraites**, tandis que la deuxième contient les **descriptions concrètes.**
La description concrète est composée des éléments qui sont orientés vers le client pour le service physique. Les trois éléments concrets XML présents dans un WSDL sont :

- <wsdl:service>
- <wsdl:port>
- <wsdl:binding>

La description abstraite est composée des éléments qui sont orientés vers la description des capacités du service Web. Ses éléments abstraits définissent les messages SOAP de façon totalement indépendante de la plate-forme et de la langue. Cela facilite la définition d'un ensemble de services pouvant être implémentés par différents sites Web. Les quatre éléments abstraits XML qui peuvent être définis dans un WSDL sont :

- <wsdl:types>
- <wsdl:message>
- <wsdl:operation>
- <wsdl:portType>

Exemple

```
<types>
<schema targetNamespace=http://advocatemedia.com/GetStockQuote.xsd
xmlns="http://www.w3.org/2000/10/XMLSchema">
<element name="StockQuoteRequest">
<complexType>
<all>
<element name="symbol" type="string"/>
</all>
</complexType>
</element>
<element name="StockQuoteResponse">
<complexType>
<all>
<element name="price" type="float"/>
</all>
</complexType>
</element>
</schema>
```

```
</types>
```

III.8. L'annuaire des services UDDI

L'annuaire des services UDDI est un standard pour la publication et la découverte des informations sur les services Web. La spécification UDDI est une initiative lancée par *ARIBA*, *Microsoft* et *IBM*. Cette spécification n'est pas gérée par le W3C mais par le groupe OASIS. La spécification UDDI vise à créer une plate-forme indépendante, un espace de travail (framework) ouvert pour la description, la découverte et l'intégration des services des entreprises.

III.8.1. CONSULTATION DE L'ANNUAIRE

L'annuaire UDDI se concentre sur le processus de découverte de l'architecture orientée services (SOA), et utilise des technologies standards telles que XML, SOAP et WSDL qui permettent de simplifier la collaboration entre partenaires dans le cadre des échanges commerciaux. L'accès au référentiel s'effectue de différentes manières.

- Les pages blanches comprennent la liste des entreprises ainsi que des informations associées à ces dernières (coordonnées, description de l'entreprise, identifiants...).
- Les pages jaunes recensent les services Web de chacune des entreprises sous le standard WSDL.
- Les pages vertes fournissent des informations techniques précises sur les services fournis.

Les entreprises publient les descriptions de leurs services Web en UDDI, sous la forme de fichiers WSDL. Ainsi, les clients peuvent plus facilement rechercher les services Web dont ils ont besoin en interrogeant le registre UDDI.
Lorsqu'un client trouve une description de service Web qui lui convient, il télécharge son fichier WSDL depuis le registre UDDI. Ensuite, à partir des informations inscrites dans le fichier WSDL, notamment la référence vers le service Web, le client peut invoquer le service Web et lui demande d'exécuter certaines de ses fonctionnalités.
Le scénario classique d'utilisation d'UDDI est illustré ci-dessous. L'entreprise **B** a publié le service Web **S**, et l'entreprise **A** est client de ce service :

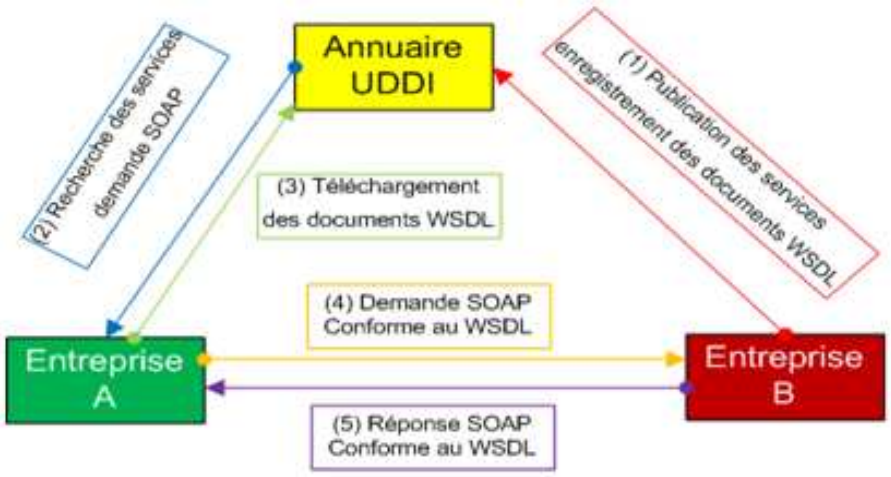


Figure II.12: schéma général de l'annuaire

III.8.2. STRUCTURES DE DONNÉES UDDI

UN REGISTRE UDDI SE COMPOSE DE QUATRE TYPES DE STRUCTURES DE DONNÉES, LE **BUSINESSENTITY**, LE **BUSINESSSERVICE**, LE **BINDINGTEMPLATE**ET LE T**MODEL**. CETTE RÉPARTITION PAR TYPE FOURNIT DES PARTITIONS SIMPLES POUR FACILITER LA LOCALISATION RAPIDE ET LA COMPRÉHENSION DES DIFFÉRENTES INFORMATIONS QUI CONSTITUENT UN ENREGISTREMENT.

EXEMPLE

```
<element name="tModel" type="uddi:tModel" />
<complexType name="tModel">
<sequence>
<element ref="uddi:name" />
<element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
<element ref="uddi:overviewDoc" minOccurs="0" />
<element ref="uddi:identifierBag" minOccurs="0" />
<element ref="uddi:categoryBag" minOccurs="0" />
</sequence>
<attribute name="tModelKey" type="uddi:tModelKey" use="required" />
<attribute name="operator" type="string" use="optional" />
<attribute name="authorizedName" type="string" use="optional" />

</COMPLEXTYPE>
```

III.9. Le langage XML [17]

Le XML est une famille de technologies développées au sein du W3C. XML est né de la tentative de mettre SGML sur le Web. La première spécification de XML est apparue en février 1998 et se concentre sur les données, contrairement à HTML qui concentre sur la présentation. XML

permet donc de transformer l'Internet d'un univers d'information et de présentation de sites Web statiques à un univers Web programmable et dynamique, centré sur les données. XML est largement utilisé par les entreprises et supporté par les manufacturiers informatiques. Il est indépendant des plateformes informatiques. Il est lisible par l'humain mais est destiné à être lu par la machine. XML permet aux données d'êtres universellement navigables. Cependant, XML a aussi la particularité d'être adapté aux utilisateurs, ce qui signifie que la composition de ses balises (tag) peut grandement différée.

Les objectifs du langage XML : Quand XML était développé, les objectifs ont été clairement déclarés dans la spécification XML 1.0, ils sont :

- ✓ Le XML doit être facilement utilisable sur le Web.
- ✓ Le XML doit supporter une grande variété d'applications.
- ✓ Le XML doit être compatible avec SGML.
- ✓ Il doit être facile d'écrire des programmes qui traitent des documents XML.
- ✓ Le nombre d'options doit être réduit au minimum, idéalement à zéro.
- ✓ Les documents XML doivent être lisibles et raisonnablement clairs.
- ✓ La conception de XML doit être menée rapidement.
- ✓ La description de XML doit être formelle et concise.
- ✓ Les documents XML doivent être faciles à créer.
- ✓ La concision du balisage XML est d'une importance minime.

Conclusion

L'architecture J2EE et les services Web sont le résultat de la collaboration exceptionnelle des technologies de l'information, qui se sont entendus sur un certain nombre de protocoles et d'approches qui favoriseront l'interopérabilité entre les plateformes, les systèmes d'exploitation et les langages de programmation.

Ces deux dernières technologies sont suffisamment développés pour que les développeurs les utilisent maintenant dans tous les domaines de l'informatique, afin de récolter les divers bénéfices de la technologie, Ils représentent aujourd'hui la technologie la plus adaptée pour le développement des systèmes d'information distribués sur l'Internet.

Nous allons présenter dans le chapitre suivant une étude détaillée sur le système LMD

Chapitre III

Système LMD

Introduction

La réforme de l'enseignement supérieur est devenue, aujourd'hui, une nécessité pour remédier à tous les dysfonctionnements existants et rendre l'université algérienne plus performante, compétitive et attractive et il lui permet de répondre aux grands défis de la mondialisation et de l'évolution rapide des sciences et de la technologie, ainsi qu'aux grandes mutations que connaît notre société.

La nouvelle architecture retenue pour l'enseignement supérieur est articulée selon trois paliers correspondant chacun à un diplôme :

- Le niveau licence correspondant à un cycle de formation de trois années après le BAC.
- Le niveau master correspondant à deux années supplémentaires après le niveau licence.
- Le niveau doctorat correspondant à trois années supplémentaires après le niveau master.

I.Historique de la réforme LMD [19]

La réforme de LMD pour « licence Master Doctorat » désigne un ensemble de modifiant le système d'enseignement supérieur pour l'adapter aux standards européens. Elle met en place principalement une architecture basée sur trois grades : Licence, Master, Doctorat, une organisation des enseignements en semestres et unités d'enseignement, la mise en œuvre des crédits européens et par la délivrance d'une annexe descriptive au diplôme. Les textes fondateurs de cette réforme sont parus en 2002, mais celle-ci s'est étalée sur plusieurs années, et en 2010 certaines formations, notamment celles de santé n'ont pas été modifiées.

Le 25 mai 1998, les ministres en charge de l'enseignement supérieur de la France, de l'Allemagne, du Royaume-Unis et de l'Italie se sont réunis à la Sorbonne et ont fait une déclaration commune en vue d'harmoniser l'architecture du système européen d'enseignement supérieur.

Le 19 juin 1999 à Bologne, les ministres de l'éducation de 29 pays européens poursuivent la réflexion sur la base de la déclaration de la Sorbonne et se fixent une série d'objectifs dont la réforme actuelle est l'aboutissement.

Le 30 mars 2001 à Salamanque, plus de 300 institutions européens se réunissent afin de rappeler les principes d'harmonisation du système européen d'enseignement supérieur et de préparer la conférence de Prague.

Enfin, le 19 mai 2001 à Prague, la déclaration commune des ministres européens de l'éducation réaffirme la volonté de continuer les efforts sur les six principaux de la déclaration de Bologne.

1. Adoption d'un système de reconnaissance rendant les diplômes universitaires plus transparent et lisibles.

2. Mise en place de cursus universitaires fondés notamment sur un premier cycle de trois ans.
3. Introduction d'un système de crédit.
4. Promotion de la mobilité des étudiants, des chercheurs ainsi que du personnel administratif.
5. Développement d'instruments communs permettant d'évaluer la qualité des enseignements.
6. Accroissement de la dimension européenne du contenu des cursus universitaires.

La réforme LMD est entrée en vigueur à partir de la rentrée universitaire 2004-2005 en Algérie, et elle a touchée dans un premier temps 10 établissements de l'enseignement supérieur sur les 58 existants. Introduit de manière graduelle, le système LMD commence à se généraliser à l'université Mouloud MAMMERI de Tizi-Ouzou (UMMTO).

II. Objectifs de la réforme [19]

La correction des différents dysfonctionnements soulignés, aussi bien au niveau de la gestion qu'au niveau des performances et de l'efficacité de l'Université algérienne, passe nécessairement par la mise en œuvre d'une réforme globale et profonde touchant ces différents aspects. Cette réforme doit réaffirmer les principes essentiels qui sous-tendent la vision des missions dévolues à l'Université algérienne, à savoir :

- ✓ Assurer une formation de qualité, en prenant en charge la satisfaction de la demande sociale, légitime, en matière d'accès à l'enseignement supérieur.
- ✓ Réaliser une véritable osmose avec l'environnement socio-économique en développant toutes les interactions possibles entre l'université et le monde qui l'entoure.
- ✓ Développer les mécanismes d'adaptation continue aux évolutions des métiers.
- ✓ Consolider sa mission culturelle par la promotion des valeurs universelles qu'exprime l'esprit universitaire, notamment celles de la tolérance et du respect de l'autre.
- ✓ Etre plus ouverte sur l'évolution mondiale, particulièrement celles des sciences et des technologies.
- ✓ Encourager et diversifier la coopération internationale selon les formes les plus appropriées.
- ✓ Asseoir les bases d'une bonne gouvernance fondée sur la participation et la concertation.

La réforme LMD, articulée sur les trois niveaux de formation : Licence – Master – Doctorat, est venue pour répondre à ces objectifs. Il s'agit d'un processus qui se veut promoteur du développement des capacités des établissements à adapter et à renouveler leurs offres de formation, en tenant compte des évolutions scientifiques et technologiques d'une part, et du marché de l'emploi d'autre part. Dans cette démarche, il est préconisé d'offrir une plus grande liberté à l'étudiant pour construire son parcours universitaire avec comme finalité son insertion dans la vie active.

Ce nouveau système vise à rendre plus lisibles les offres de formation de chaque établissement en adoptant des niveaux et des appellations universelles pour les diplômes. Il permet d'accroître ainsi la

fiabilité et la transférabilité des diplômes délivrés par l'Université algérienne, facilitant ainsi la mobilité de nos étudiants.

III. L'unité d'enseignement

C'est ensemble de matières soumises à une organisation pédagogique cohérente. Ces derniers sont dispensées sous toute forme d'enseignement (Cours, TD, TP ; conférences, séminaires, projets stages….) les unités d'enseignement sont dispensées semestriellement et se distinguent en 3 catégories :

1. Unité fondamentale (UF) : elle groupe les matières fondamentales pour une spécialité donnée.
2. Unité méthodologique : elle regroupe les matières d'enseignement d'outils méthodologiques nécessaire à la réalisation du parcours de formation.
3. Unité de découverte : les matières qu'elle regroupe ne correspondent pas forcement à la spécialité de l'étudiant mais elles servent beaucoup plus à enrichir ses connaissances et sa culture générale.

Chaque UE et chacune des ses matières constitutives sont affectées d'une valeur en crédits.

La valeur totale de ses crédits est fixée à 30 par semestre.

IV. évaluation et progressions

Les aptitudes et l'acquisition des connaissances, concernant chaque unité d'enseignement, sont appréciées semestriellement soit par un contrôle continu (TD, TP, Travail personnel, etc…) soit par un examen final soit par les deux modes de contrôle.

- ✓ L'unité d'enseignement est définitivement acquise pour tout étudiant ayant obtenus une note supérieur ou égale à 10/20 dans toute les matières qui la constituent, ou par compensation si la moyenne de l'ensemble des notes obtenues pondérées de leurs coefficient respectifs est égale ou supérieur à 10/20.
- ✓ L'exclusion d'un étudiant dans une matière composant une unité d'enseignement ne permet pas l'acquisition de cette dernière par le calcul de la moyenne des notes obtenues dans les autres matières qui la constituent.
- ✓ En cas d'échec à la première session, l'étudiant se présente à la session de rattrapage aux épreuves relatives aux unités d'enseignement non acquises. Dans ce cas, l'étudiant garde le bénéfice des matières acquises.
- ✓ Dans le cas d'une unité d'enseignement acquise par la compensation, l'étudiant peut être autorisé à se présenter, en session de rattrapage, aux matières non acquises de ladite unité.
- ✓ Lors de la session de rattrapage, la note, pour chacune des matières concernées, est alors déterminé sur la base de la note obtenue à l'épreuve de rattrapage selon les modalités de contrôle des connaissances.

- ✓ La note finale retenue pour la matière sera la meilleure des moyennes entre la première session et la session de rattrapage.
- ✓ Dans le cas ou une unité d'enseignement n'est pas acquise, les crédits affectés aux matières acquise qui la composent sont capitalisables.
- ✓ Le principe de compensation s'applique :
  - **A l'unité d'enseignement :** il permet l'acquisition de l'unité d'enseignement par le calcul de la moyenne des notes des matières constitutives affectées de leur coefficient respectif.
  - **Au semestre :** il permet l'acquisition du semestre par le calcul de la moyenne des notes des unités d'enseignement qui le composent, affectées de leur coefficient respectif.
  - **Au niveau (L1, L2, L3) :** il permet l'acquisition du nouveau L1, L2 ou L3, le calcul de la moyenne des notes des unités d'enseignement qui le composent, affectées de leur coefficient respectif.

IV.1. La progression dans les études

Le passage du premier au second semestre d'une même année universitaire dans un même parcours de formation est de droit pour tout étudiant régulièrement inscrit.

IV.2. la progression dans les études de licence

- L'acquisition d'une année pédagogique dans le cycle licence est conditionnée par l'acquisition des semestres qui la compose.
- Cependant, le passage peut aussi se faire comme suit :
  - ✓ De la première à la deuxième année de licence peut être autorisé pour tout étudiant ayant acquis au minimum 30 crédits dont 1/3 au moins dans un semestre.
  - ✓ De la deuxième à la troisième année peut être autorisé à tout étudiant ayant validé au minimum 90 crédit et acquis les unités d'enseignements fondamentales requises à la poursuit des études en spécialité.
- L'étudiant, non admis à progresser en deuxième année ou en troisième année d'un parcours de formation, est, selon le cas, autorisé à se réinscrire dans le même parcours ou orienté, par l'équipe de formation, vers un autre parcours de formation.
- La procédure d'orientation fait autant que possible l'objet d'une application prioritaire pour les étudiants en situation d'échec dans leur parcours de formation initial. Elle doit conduite, par les biais de passerelles, à la construction d'un parcours individualisé plus conforme aux aptitudes de l'étudiant et devrait lui permettre une meilleure progression dans son cursus d'étude.
- En aucun cas, l'étudiant inscrit en licence ne peut y séjourner plus de cinq années au maximum, même dans le cas d'une réorientation.

IV.3. la progression dans les études de master

Le passage de la première à la deuxième année est de droit si l'étudiant a acquis les deux premiers semestres du cursus de formation.

Cependant, le passage de la première à la deuxième année peut être autorisé pour tout étudiant ayant validé au minimum 45 crédits et acquis les unités d'enseignements requises à la poursuit des études en spécialité.

Au aucun cas, l'étudiant inscrit en master ne peut y séjourner plus de 3 années maximum, même dans le cas d'une réorientation.

V. l'algorithme général pour les deux systèmes de progression

On distingue deux systèmes de progression.

V.1. premier système de progression

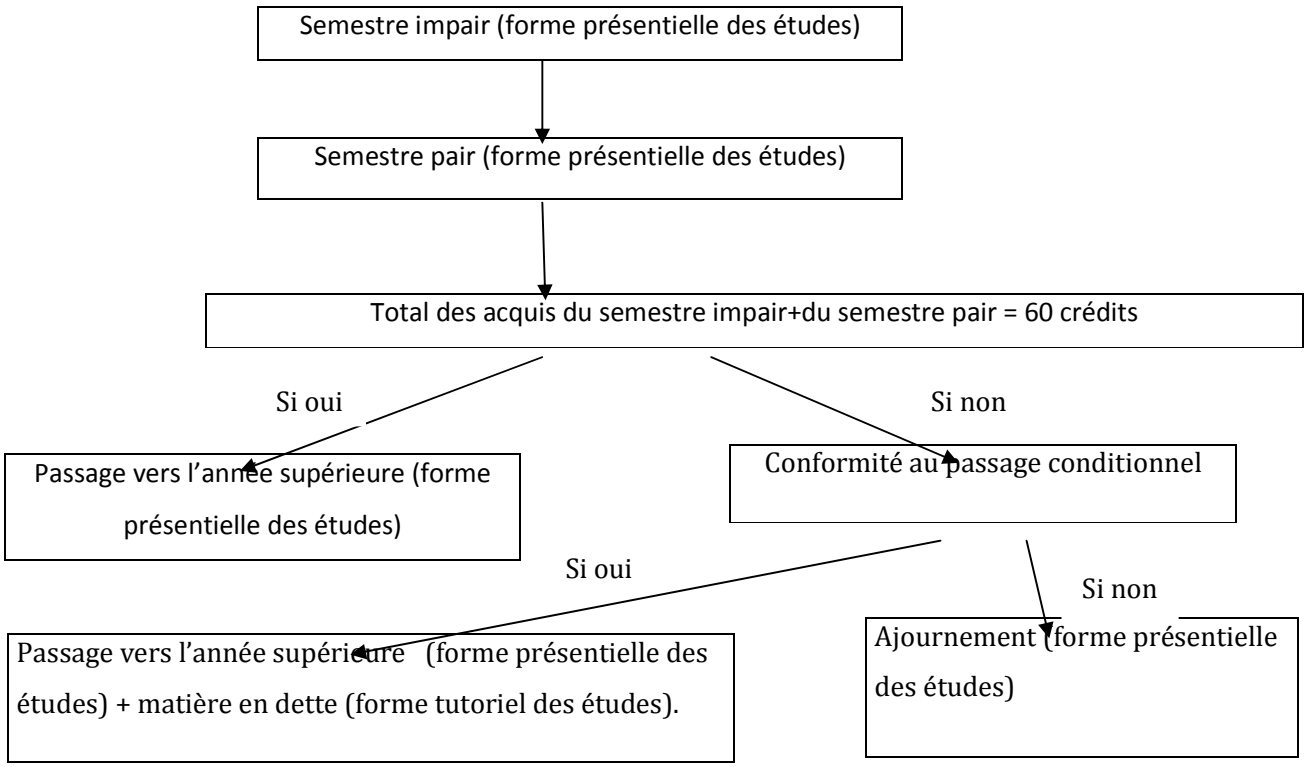


Figure III.1 : L'algorithme du premier système de progression

V.2. deuxième système de progression

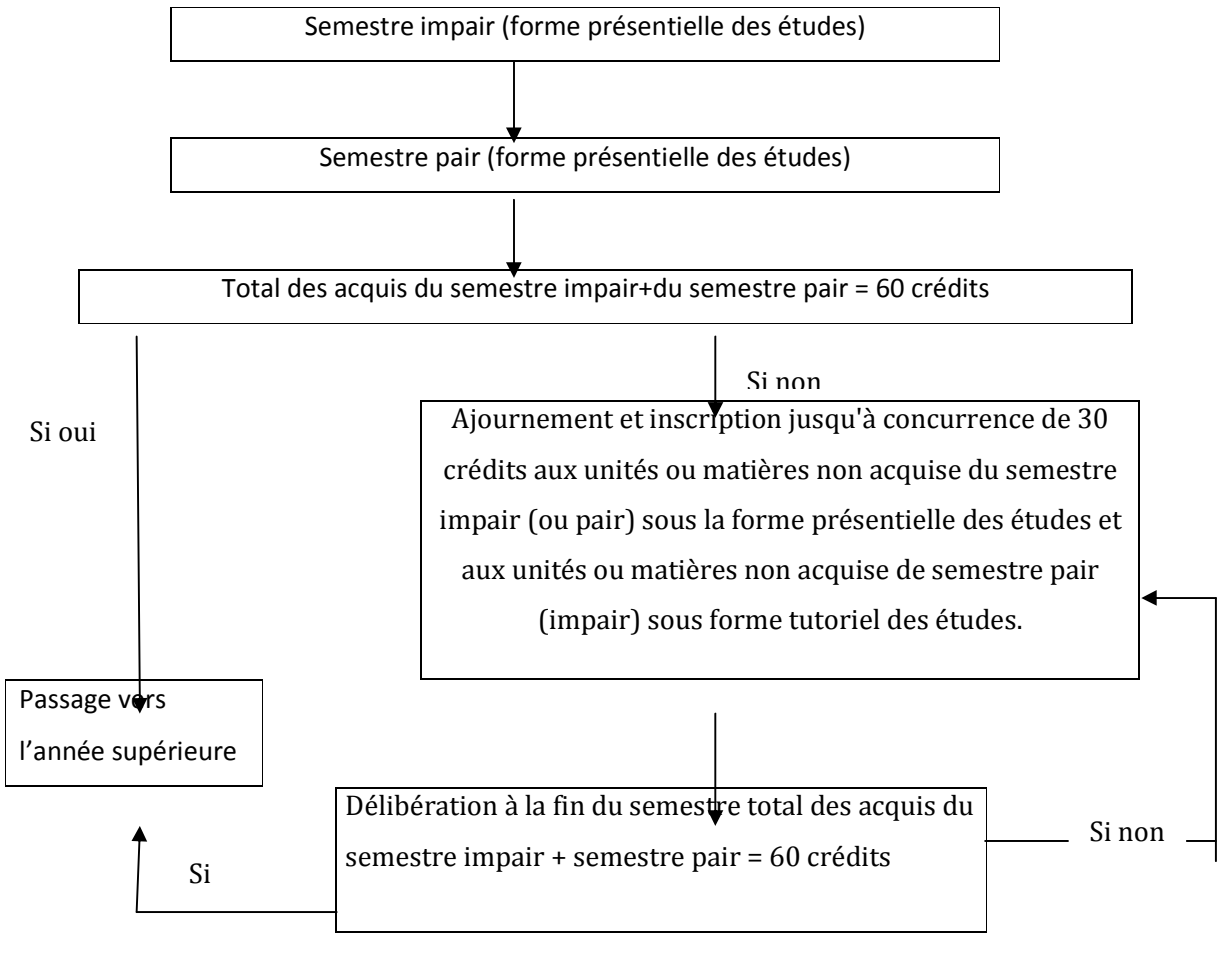


Figure III.2 : L'algorithme du deuxième système de progression

VI. Forme présentielle des études et forme tutoriel des études

VI.1. Forme présentielle des études

Forme classique des études (cours, TD avec présence obligatoire)

VI.2. forme tutoriel des études

- Prise en charge par un enseignant appelé tuteur principal
- Peut être secondé par des tuteurs secondaires dans le cas de sureffectif étudiant
- Le tuteur principal a pour tâches :
  - ✓ Afficher de programme détaillé.
  - ✓ Annoncer le ou les documents de référence.
  - ✓ Organiser une ou des séances de consultation hebdomadaires.
  - ✓ Répondre aux sollicitations des étudiants par courrier électronique.
  - ✓ Rédiger, organiser et corriger le ou les examens.

- ✓ Remettre le PV des résultats à l'administration.

Conclusion

Toutefois les règles de progression sont compliquées et justifient l'utilisation de l'outil informatique permettant d'automatiser le calcul de moyenne et de passage.

Nous allons présenter dans le chapitre suivant les détails de la conception.

Chapitre IV

Analyse et Conception

VI.1 INTRODUCTION :

La conception de toute solution logicielle doit être traitée avec précision et détail, précédée d'une analyse profonde et bien réfléchie, car elle est le reflet du futur système avant même sa concrétisation. Dans le but d'avoir une meilleure analyse et de rendre la conception de notre application plus complète, nous avons adopté le langage **UML** (Unified Modeling Language) qui permet de bien représenter l'aspect statique et dynamique d'une application par une série de diagrammes qu'il offre.

VI.2 Présentation des diagrammes UML. [18]

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 2.0 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 diagrammes UML peuvent être classés sous deux grandes catégories :

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

VI.2.1. Les diagrammes de structure ou statique *:* qui sont au nombre de six :

1. 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.

2. Les diagrammes d'objet : 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.

3. 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, librairie, exécutables,…) pour montrer la mise en œuvre physique des modèles statiques avec l'environnement de développement.

4. 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,… etc.

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

5. 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.

6. 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.

VI.2.2. Les diagrammes de comportement : qui sont au nombre de sept :

1. Les diagrammes d'activités : décrivent le comportement d'une méthode, le déroulement d'un cas d'utilisation, les enchaînements 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).

2. Les diagrammes de cas d'utilisation : Les cas d'utilisations (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.

3. 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.

4. Les diagrammes de séquence : permettent de représenter les interactions entre objets selon un point de vue temporel. L'accent est mis sur la chronologie des envois de messages.

5. 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.

6. Les diagrammes de vue d'ensemble des interactions : donnent une 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.

7. Les diagrammes de timing : permettent de définir des contraintes temporelles pour un ou plusieurs objets.

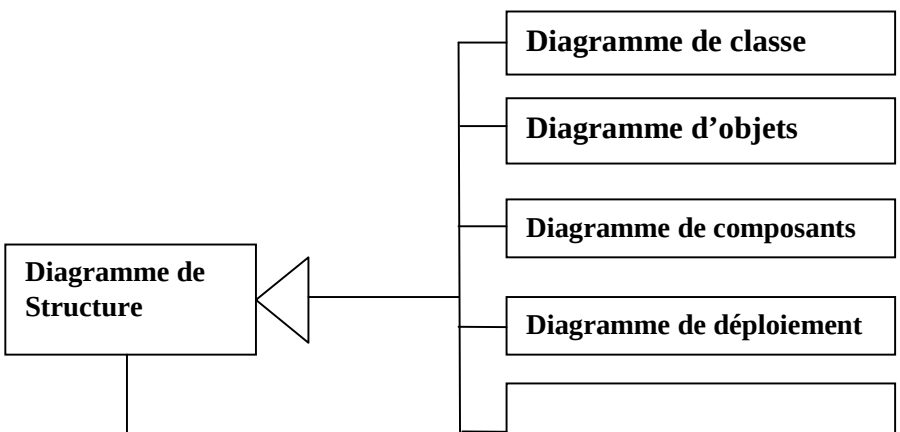


Figure VI.1 : Classification des diagrammes UML (V. 2.0)

Nous remarquons dans la figure VI.1 extraite de (Fowler, 2004), que les diagrammes de comportement sont eux-mêmes classés en deux types de diagrammes. On distingue les diagrammes d'interaction représentés par : les diagrammes de séquence, de communication, de vue d'ensemble des interactions et timing des autres diagrammes d'activité, de cas d'utilisation, et de machine d'état.

Nous pouvons considérer que les diagrammes d'activité et de cas d'utilisation décrivent plus le fonctionnement du système alors qu'un diagramme de machine d'état décrit la dynamique d'un objet en termes de changements d'états.

VI.3. Une démarche pour l'analyse et la conception du projet.

VI.3.1. Description générale de la démarche :

Afin de construire notre application, nous proposons la démarche suivante dont l'objectif est d'aboutir au schéma de la base de données de l'application ainsi que les interfaces logicielles nécessaires à son exploitation. Rappelons d'abord qu'une démarche de génie logiciel est sensée définir les différentes étapes par lesquelles doit passer le développement d'un produit logiciel comme le montre le schéma suivant

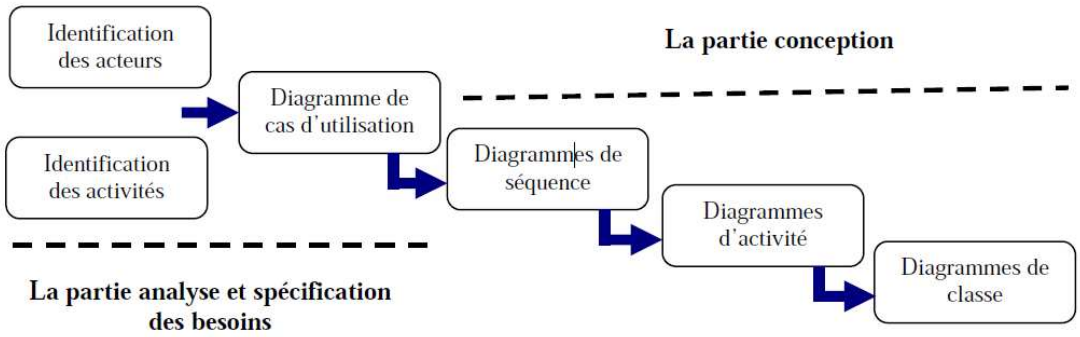
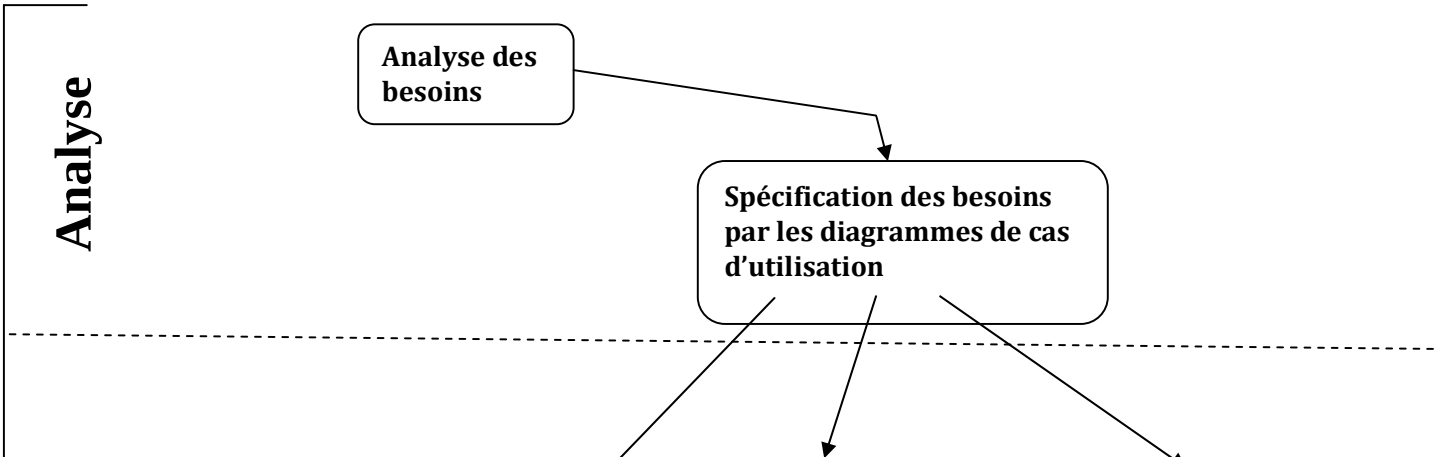


Figure VI.2 : Phase du Processus d'analyse, de conception et de réalisation de notre application.

Conformément à cette démarche, nous proposons l'usage des diagrammes d'UML suivants :



– – – –

Figure VI.3 : Démarche d'analyse et de conception du SAI avec les diagrammes d'UML

En principe, les diagrammes d'UML peuvent être utilisés librement à tous les niveaux d'un processus d'élaboration d'un produit logiciel. Cependant, la pratique montre bien que certains diagrammes s'apprêtent bien mieux à certaines phases qu'à d'autre.

VI.3.2. Phase d'analyse :

- ***Analyse des besoins :*** cette phase consiste à étudier l'existant en recensant toutes les informations nécessaire sur le champ d'étude.

- ***Diagrammes de cas d'utilisation :*** Les cas d'utilisation permettront de décrire les fonctionnalités du système et leurs interactions avec le ou les utilisateurs. La description des interfaces pourrait très bien se faire à partir de ce point.

VI.3.3. Phase de conception :

- ***Diagrammes de séquence* :** Ces schémas permettront de détailler tous les scénarios possibles d'un cas d'utilisation en précisant les objets du système et les acteurs impliqués dans le scénario. L'affinage des interfaces peut se faire à ce niveau ou bien au niveau des diagrammes de collaboration.
- ***Diagrammes d'activité :*** Permettra de modéliser les aspects dynamiques du système en donnant une représentation des différents processus faisant intervenir les acteurs concernés et manipulant les ressources nécessaires
- ***Diagrammes de classes :*** Permettra d'établir un schéma conceptuel de la partie statique du système d'information en décrivant les classes et les associations entre classes d'information.

VI.3.4. Phase de réalisation :

C'est à partir du diagramme de classe que l'on pourra dériver le modèle relationnel de la base de données à implémenter. Les diagrammes d'activité permettront de mettre on œuvre les interfaces logicielles nécessaires à l'exploitation de la base de données.

VI.4. Cahier de charge de notre application :

Notre projet porte sur la mise en place d'une application en basant sur l'architecture J2EE et l'architecture Web Service pour l'automatisation des activités de gestion de la scolarité de notre département, vu les calculs nuisibles, tout en garantissant plus de sûreté et de rapidité.

L'application assurera un environnement interactif afin de rendre les tâches de gestion accessibles aux différents acteurs administratifs du département informatique et ce via un réseau local ou externe dans le but de maîtriser les différentes fonctions de base du système qui sont :

- ***Pour l'agent de la scolarité***

- S’authentifier.
- Accéder à son espace privé.
- Introduction des emplois du temps.
- Introduction des plannings des examens.
- Etablissent des listes globales des étudiants et leurs répartitions par sections et par groupes (affectation manuelle ou dynamique des étudiants)
- Affectation des enseignants responsable selon les modules et les sections qu’ils enseignent.
- Préparation des procès verbaux globaux.
- Edition des certificats de scolarité.
- Edition des relevés de notes.
- Changement de mot de passe.

✓ ***Pour l’administrateur :***

- Authentification pour accéder à son espace personnel.
- Création de comptes.
- Suppression de comptes.
- Modification de compte.
- Changer son mot de passe.
- Mise à jour des programmes.

✓ ***Pour l’enseignant :***

- Consultation des programmes.
- Consultation des emplois du temps.
- Consultation des plannings des examens.
- Gestion des notes des étudiants après l’authentification.
- Consultation des groupes.
- Changement de mot de passe.

✓ **Côté étudiant :**

- Consultation des programmes.
- Consultation des emplois du temps.
- Consultation des plannings des examens.
- Consultation des notes.

VI.5. Analyse et conception de notre système

Consiste à déterminer ce que l’application devra faire, l’expression de son comportement et de son architecture ainsi que l’examen des cas d’utilisation et leurs scénarios.

VI.5.1 Analyse :

Cette activité commence par la mise en évidence des différents acteurs intervenants dans le système cible ainsi que leurs besoins. Ensuite, la phase conception donnera la modélisation des objectifs à atteindre en s'appuyant sur les résultats de la phase analyse.

VI.5.1.1. Spécification des besoins :

Au cours de notre passage par différents services de la scolarité LMD de l'informatique, nous avons constaté quelques problèmes tel que les traitements manuels d'où un taux d'erreur assez important, difficulté dans la recherche de l'information ainsi l'importance charges des travaux et l'exécution quotidienne des mêmes traitements.

Suite a ces problèmes, nous avons pu tracer l'objectif d'améliorer le fonctionnement et l'organisation du système actuel.

VI.5.1.2. Cas d'utilisations :

Les cas d'utilisation sont des outils formels qui permettent de consigner et d'exprimer les interactions et les dialogues entre le système et ses utilisateurs (Acteurs). L'ensemble des cas d'utilisation donne une description détaillée de comportement du système en réponse aux sollicitations de ces derniers. Un cas d'utilisation doit exprimer ce que le système doit faire sans préjuger de façon dont cela sera fait.

Avant de décrire les différents cas d'utilisation, il est nécessaire de mettre en évidence les acteurs utilisant le système ainsi que leurs tâches respectives et les scénarios qui les décrivent. En regroupant un ensemble de ces scénarios nous obtenons les différents cas d'utilisation.

Identification des acteurs :

Acteur : entité externe qui agit sur le système ; Le terme acteur ne désigne pas seulement les utilisateurs humains mais également les autres systèmes. Les acteurs sont des entités qui représentent des rôles au travers d'une certaine utilisation (cas) et non pas des personnes physiques.

Notre application fait intervenir quatre acteurs :

1- **Administrateur :** désigné par le département (informaticien ou non) et il se peut qu'il soit l'administrateur de la BDD.
2- **Agent de la scolarité :** Le responsable chargé pour la gestion de la scolarité.
3- **Enseignant :** qui enseigne au sein du département informatique.
4- **Etudiant :** qui fait ses études au sein du département informatique.

VI.5.1.3. Spécification des tâches :

Les acteurs définis précédemment effectuent un certain nombre de taches, ces taches sont résumées dans le tableau ci-dessous :

Acteurs	Tâches
Administrateur	T1 : Authentification pour accéder à son espace personnel. T2 : Création de comptes. T3 : Suppression de comptes. T4 : Modification de compte. T5 : Changer son mot de passe. T6 : Mise à jour des programmes.
Enseignant	T- S'authentifier (pour sécuriser les notes des étudiants). T- Accéder à l'interface enseignant. T- Consulter les programmes. T- Consulter les emplois du temps. T- Consulter les plannings des examens. T- Gérer les notes des étudiants. T- Voir les notes d'un groupe/section. T- Voir les groupes/section. T- Changer le mot de passe.
Etudiant	T- Se connecter à l'interface étudiant. T- Consulter les programmes. T- Consulter les emplois du temps. T- Consulter les plannings des examens. T- Voir les notes.

Agent de Scolarité	T-S'authentifier. T-Accéder a son espace privé. T-Introduction des emplois du temps. T-Introduction des plannings des examens. T-Etablissent des listes globales des étudiants. T-Affectation des enseignants. T-Préparation des procès verbaux globaux. T-Edition des certificats de scolarité. T-Edition des relevés de notes. T-Changement de mot de passe.

Tableau VI.1 Spécification des tâches.

VI.1.4. Diagramme des cas d'utilisation globale

Il décrit le comportement du système du point du vue utilisateur sous la forme d'actions et de réactions. Il existe deux concepts fondamentaux dans la modélisation par cas d'utilisation :

- Les acteurs qui utilisent le système
- Les cas d'utilisation qui représentent l'utilisation du système par les acteurs

Chaque cas d'utilisation indique une fonctionnalité du système déclenchée par un acteur externe du système. Ce genre de diagramme permet de mettre en place et de comprendre les besoins des utilisateurs.

Les cas d'utilisation peuvent être structurés. En plus de la relation de communication, qui consiste au déclenchement d'un cas d'utilisation par un acteur, nous pouvons citer deux types de liens ou relations qui sont les plus utilisés : le lien d'utilisation et le lien d'extension. [1]

- **Le lien d'utilisation :** ce lien nommé « utilise » (ou « include » en anglais) indique que le cas d'utilisation source contient le comportement décrit dans le cas d'utilisation destination.

- **Le lien d'extension :** ce lien indique que le cas d'utilisation source « étend » (en anglais « extend ») ou précise le cas d'utilisation destination.

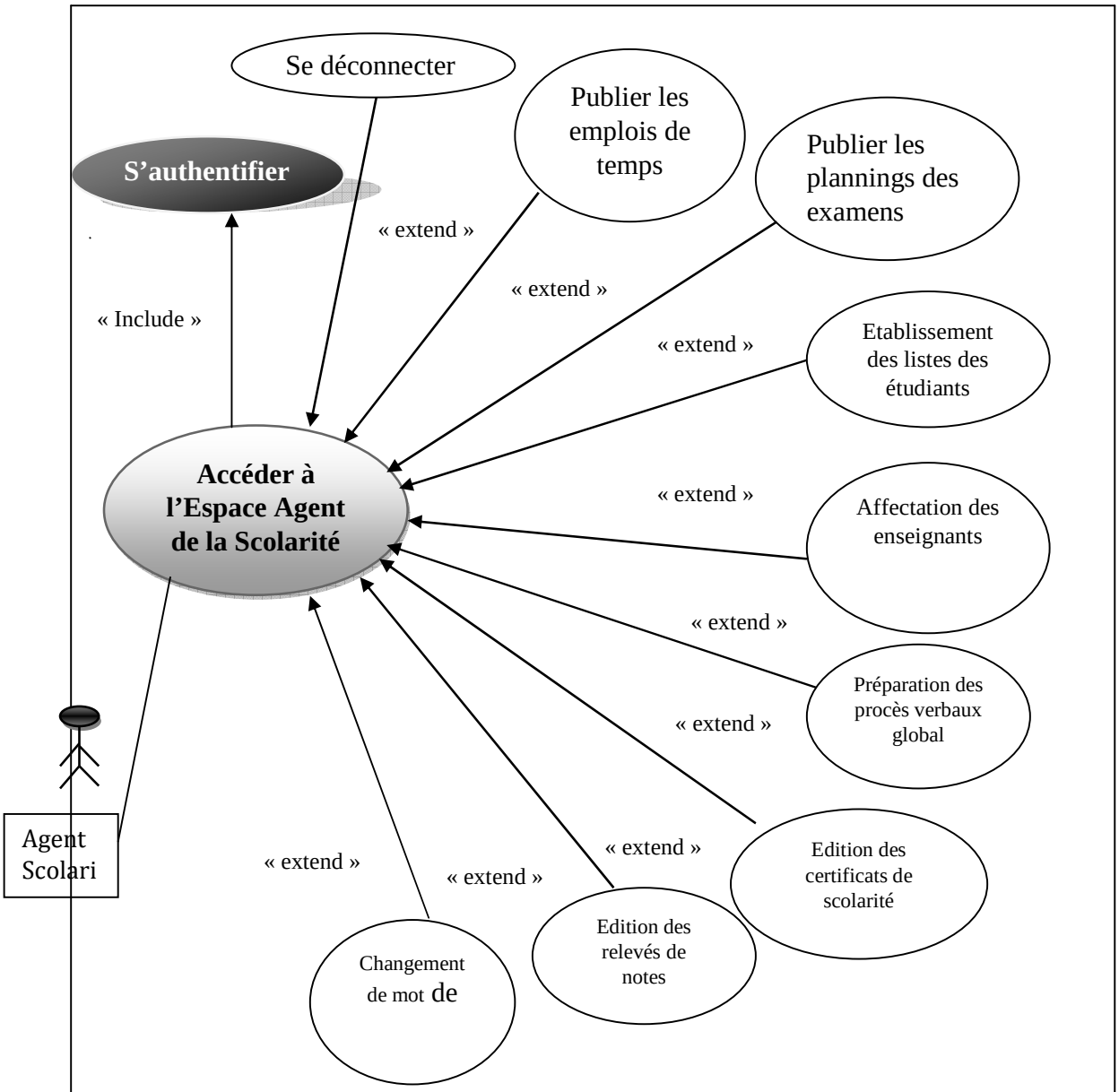


Figure VI.4 Diagramme de cas d'utilisation Agent de la scolarité.

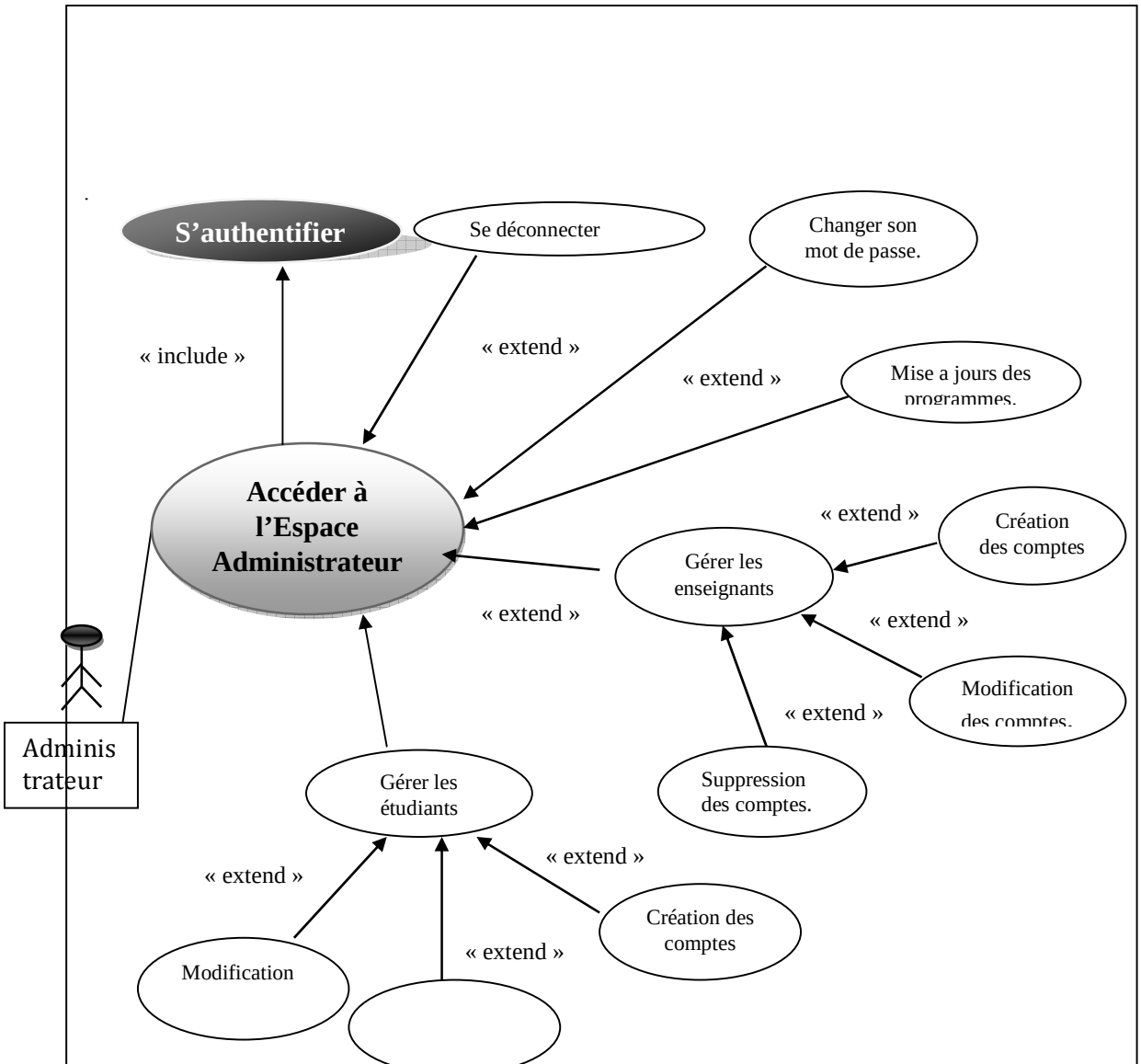


Figure VI.5 Diagramme de cas d'utilisation Administrateur.

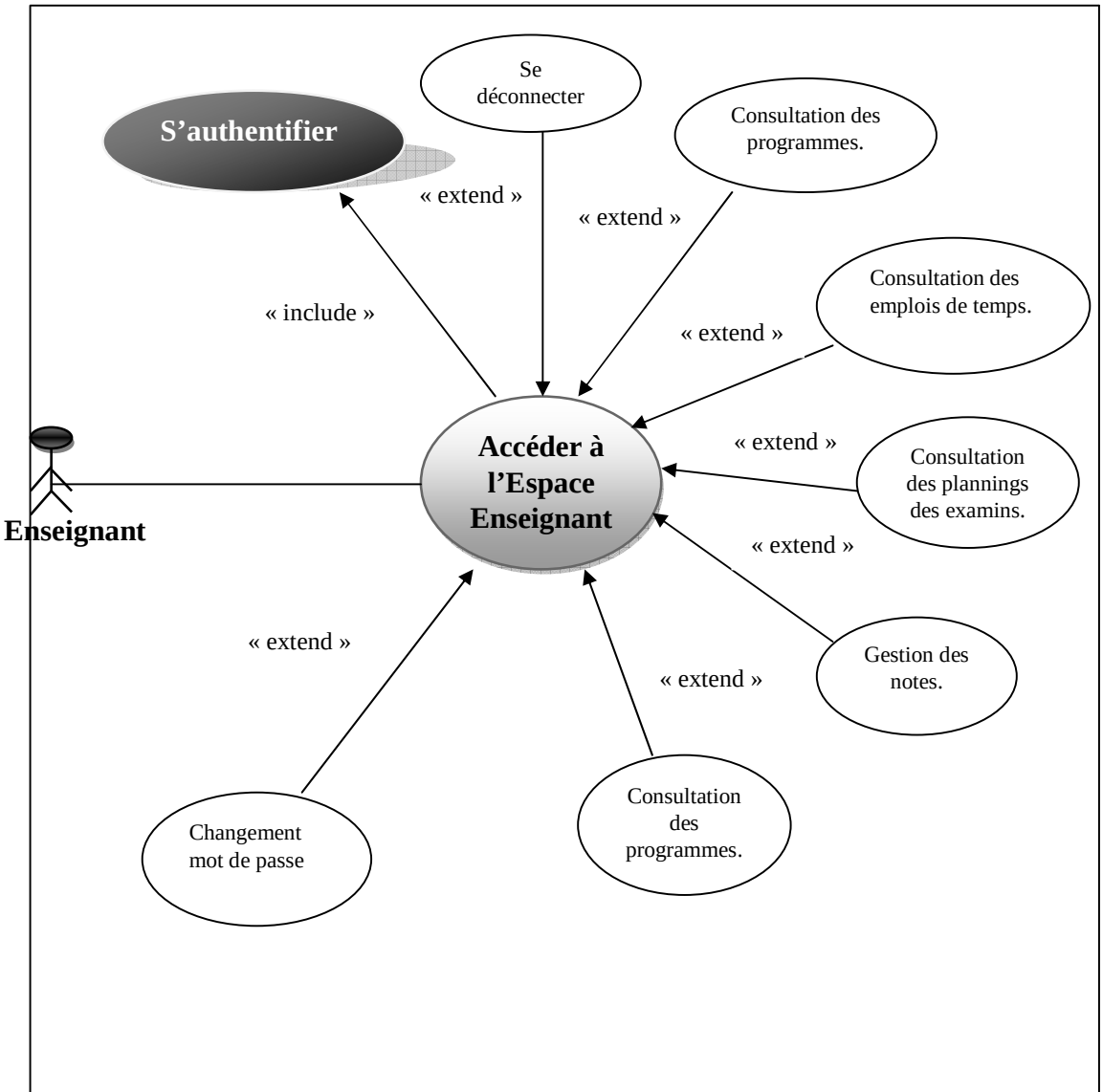


Figure VI.6 Diagramme de cas d'utilisation Enseignant.

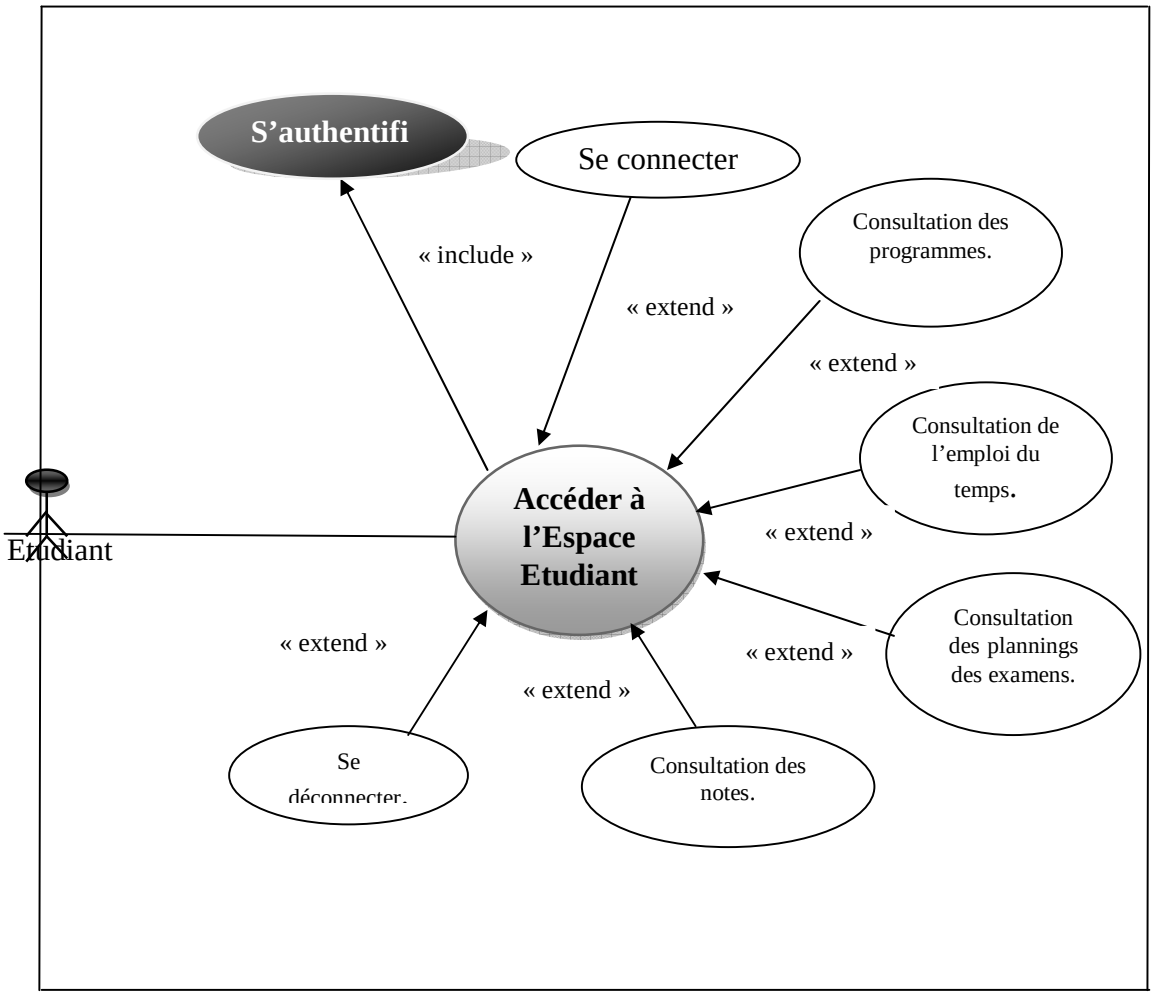


Figure VI.7 Diagramme de cas d'utilisation Etudiant.

VI.5.2. Conception :

Après avoir spécifié les besoins de l'application et déterminé les cas d'utilisations, nous allons passer à la phase de conception du projet, qui consiste à affiner les spécifications d'après leurs contenus.

La conception des applications web se distingue de celle d'autres systèmes par deux activités majeurs :

VI.5.2.1. Elaboration des Diagrammes de séquences :

Dans l'étape suivante nous avons choisi de modéliser l'interaction enter le système et les acteurs grâce au diagramme de séquence.

Vu le nombre élevé de cas d'utilisation recensés, et afin d'éviter qu'ils s'étalent sur plusieurs pages, nous avons décidé d'en étudier que quelque diagrammes :

Diagramme de séquence du cas d'utilisation « Authentification » :

<u>Les acteurs</u> : l'Administrateur, l'Enseignant et l'Agent de scolarité.

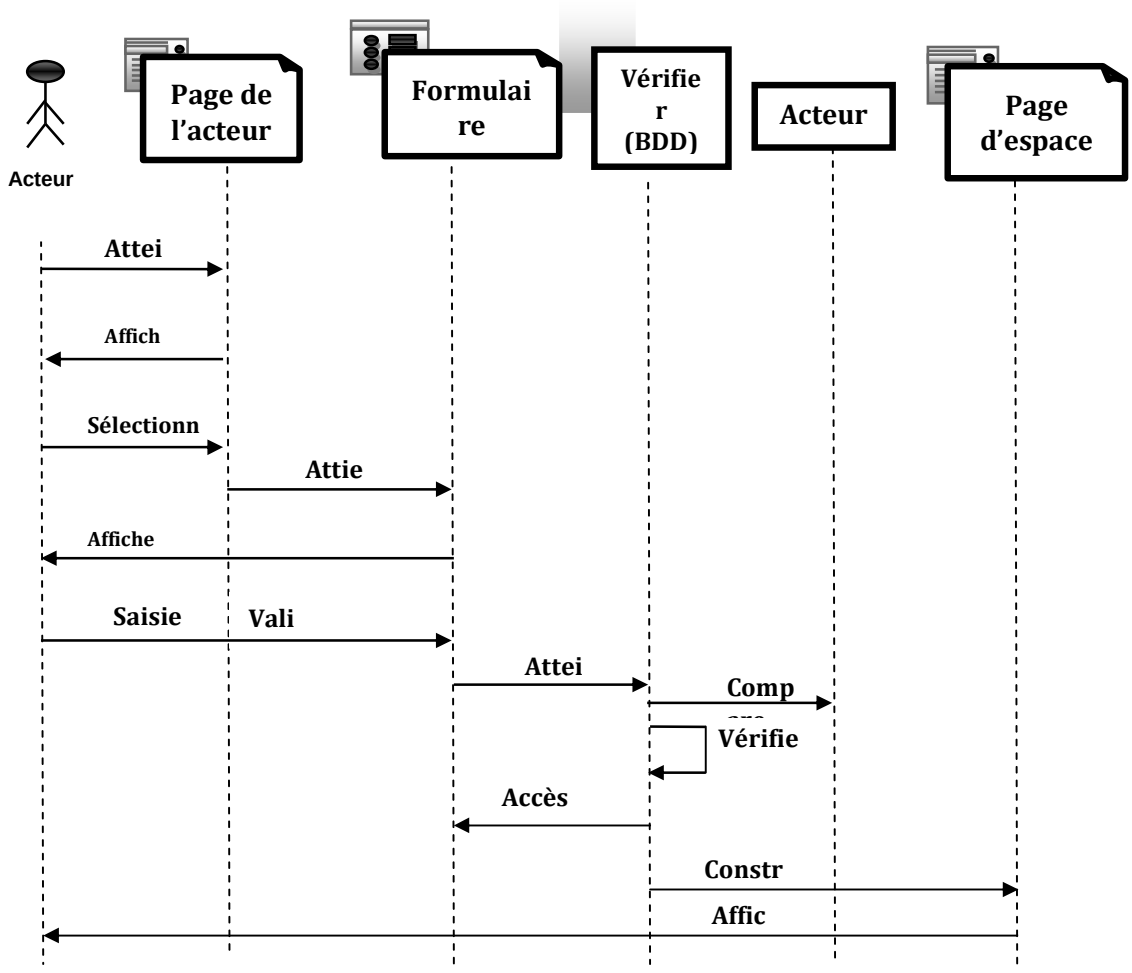


Figure VI.8 Diagramme de séquence de cas d'utilisation « Authentification ».

Diagramme de séquence de cas d'utilisation « Gérer les notes » :

<u>Acteur</u> : Enseignant.

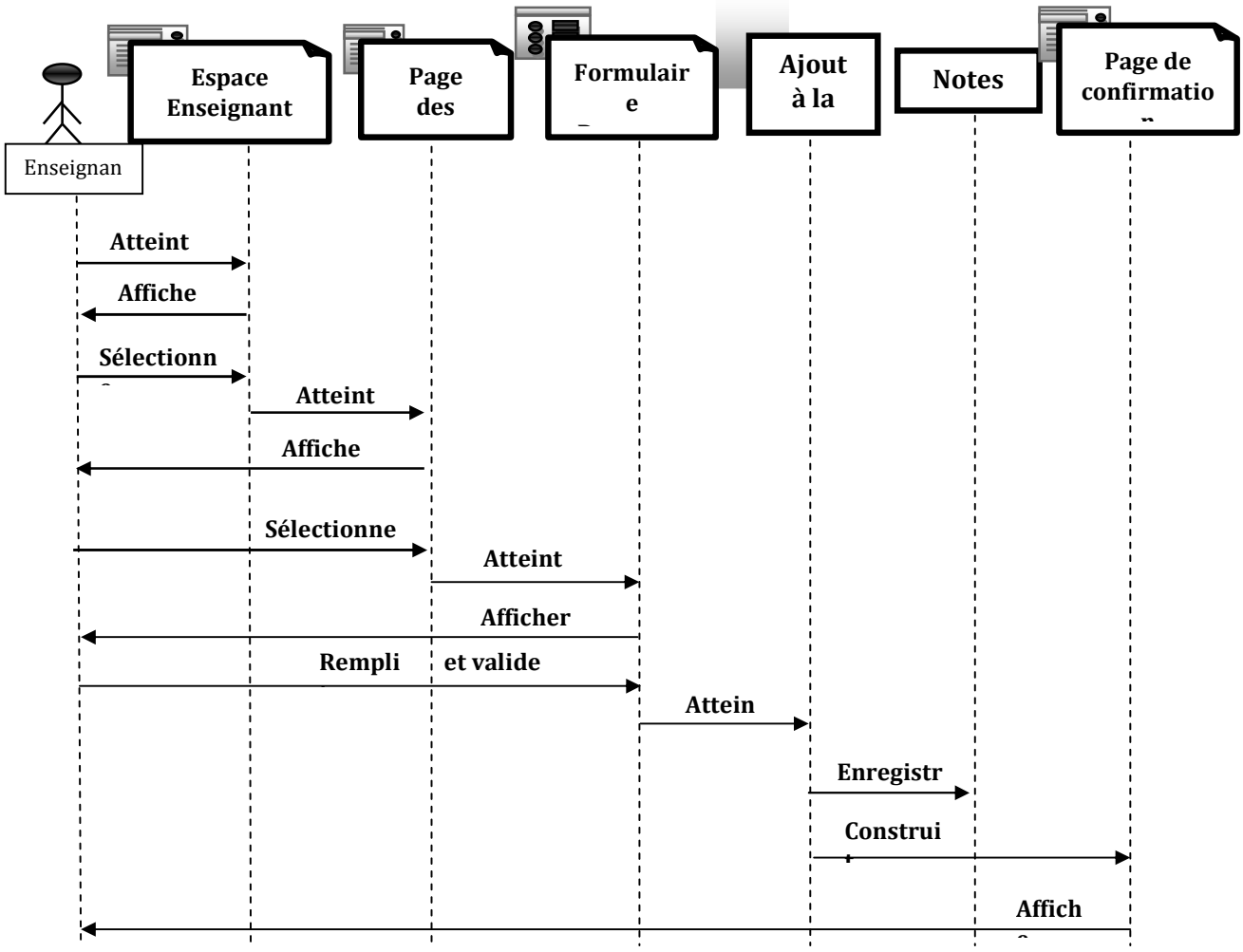


Figure VI.9 Diagramme de séquence de cas d'utilisation « Gérer les notes » :

Diagramme de séquence de cas d'utilisation « Edition des relevés de note » :

<u>Acteur</u> : Agent de scolarité.

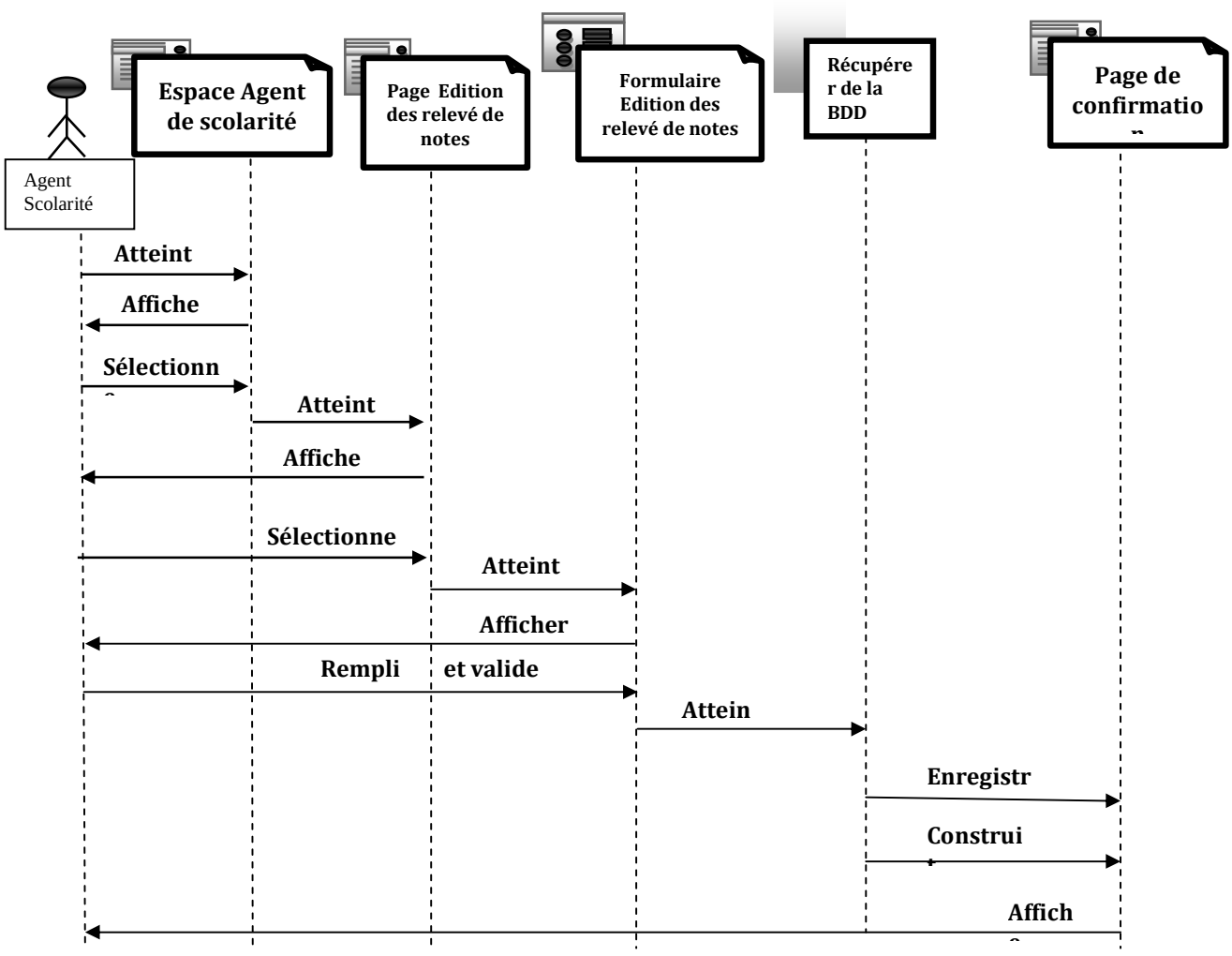


Figure VI.10 ***Diagramme de séquence de cas d'utilisation « Edition de relevé de note »***

Diagramme de séquence de cas d'utilisation « Changer le mot de passe » :

<u>Acteurs</u> : Agent de scolarité, Enseignant, Administrateur.

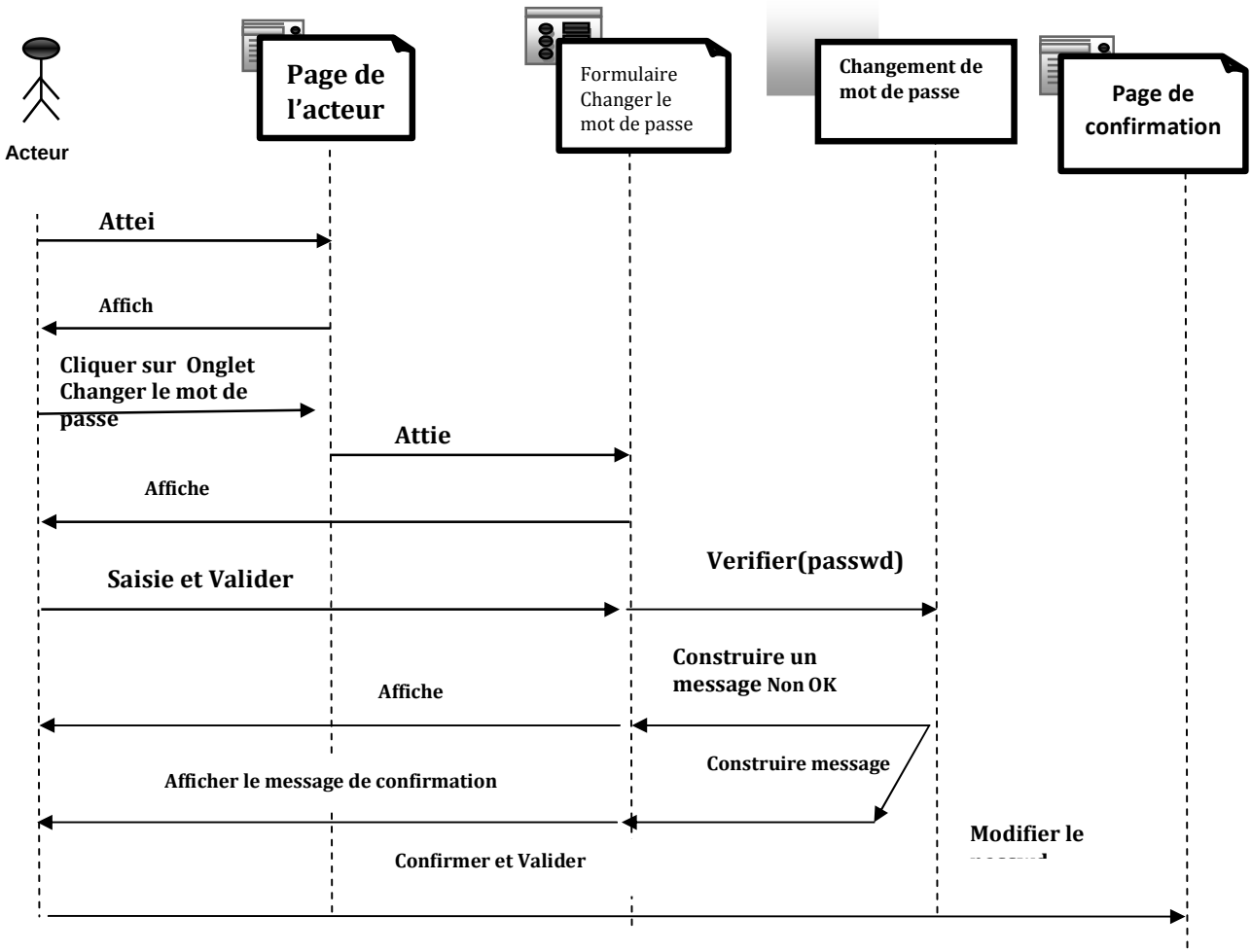


Figure VI.11 Diagramme de séquence de cas d'utilisation « Changer le mot de passe »

Diagramme de séquence de cas d'utilisation « Création de compte» :

<u>Acteur</u> : Administrateur.

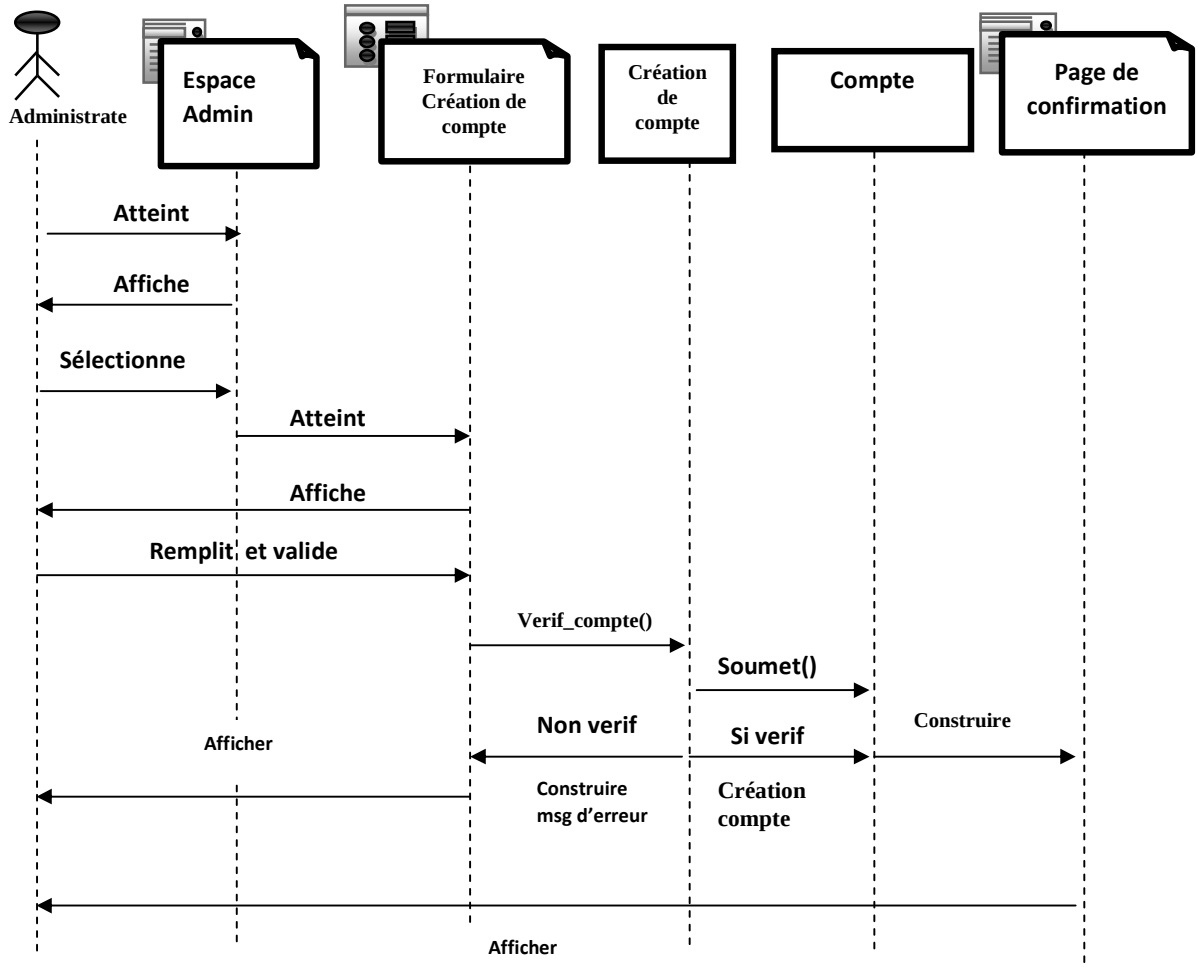


Figure VI.12 Diagramme de séquence de cas d'utilisation « Création de compte»

Diagramme de séquence de cas d'utilisation « Affectation des Enseignants » :

<u>Acteur</u> : Agent de Scolarité.

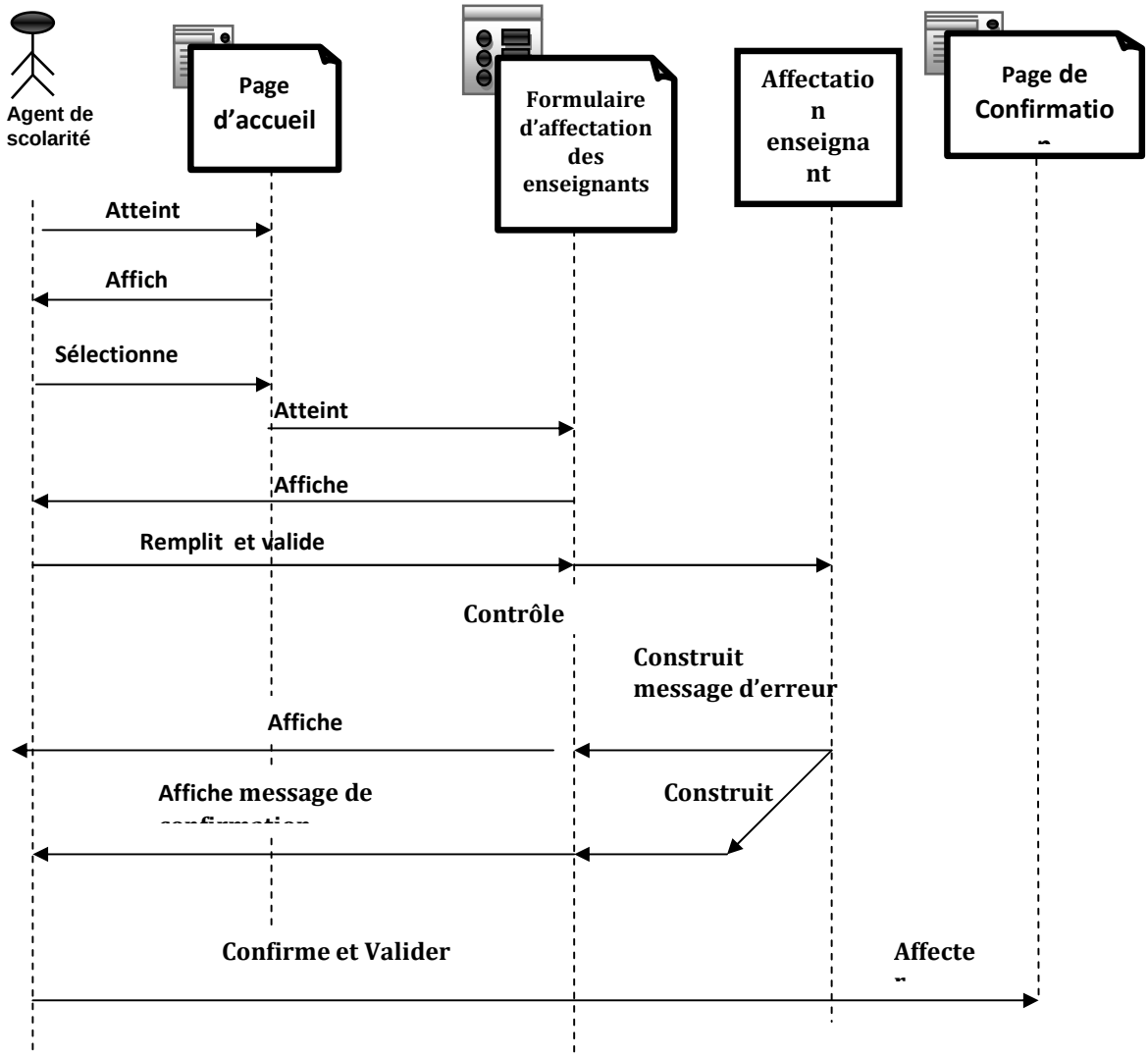


Figure VI.13 Diagramme de séquence de cas d'utilisation « Affectation des Enseignants »

Diagramme de séquence de cas d'utilisation « Etablir les sections et les groupe » :

<u>Acteur :</u> Agent de scolarité.

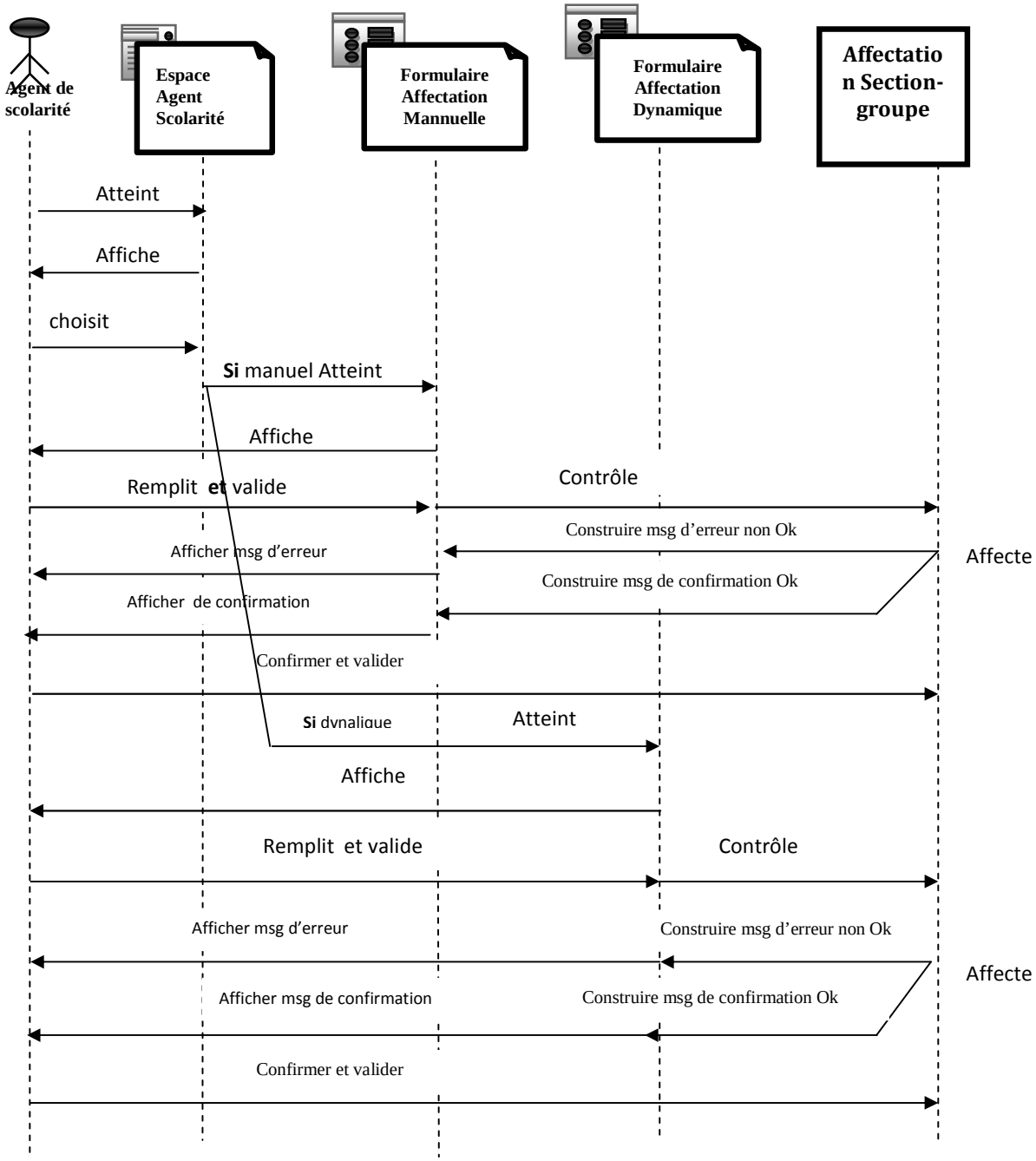


Figure VI.14 Diagramme de séquence de cas d'utilisation « Etablir les sections et les groupes»

Diagramme de séquence de cas d'utilisation « Voir les notes » :

<u>Acteur :</u> Etudiant.

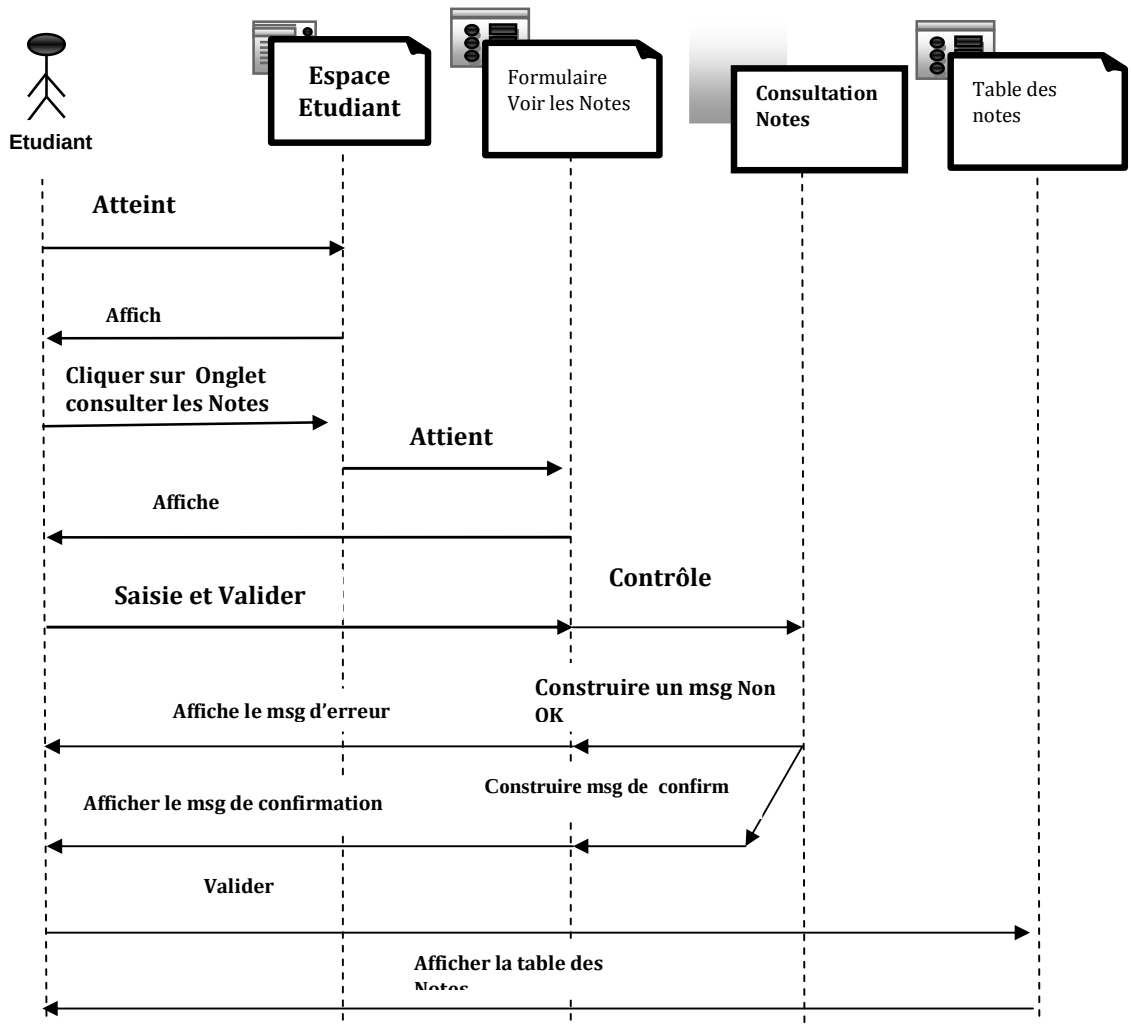


Figure VI.15 Diagramme de séquence de cas d'utilisation « Voir les notes ».

VI5.2.2. Diagramme d'activité :

Le diagramme d'activité est l'un des diagrammes d'UML qui sont conçus pour la modélisation de l'aspect dynamique du système. Un diagramme d'activité représente l'état de l'exécution d'un mécanisme, sous forme d'un déroulement d'étapes regroupées séquentiellement dans des branches parallèles de flux de contrôle.

Diagramme d'activité de cas d'utilisation «*Authentification* ».

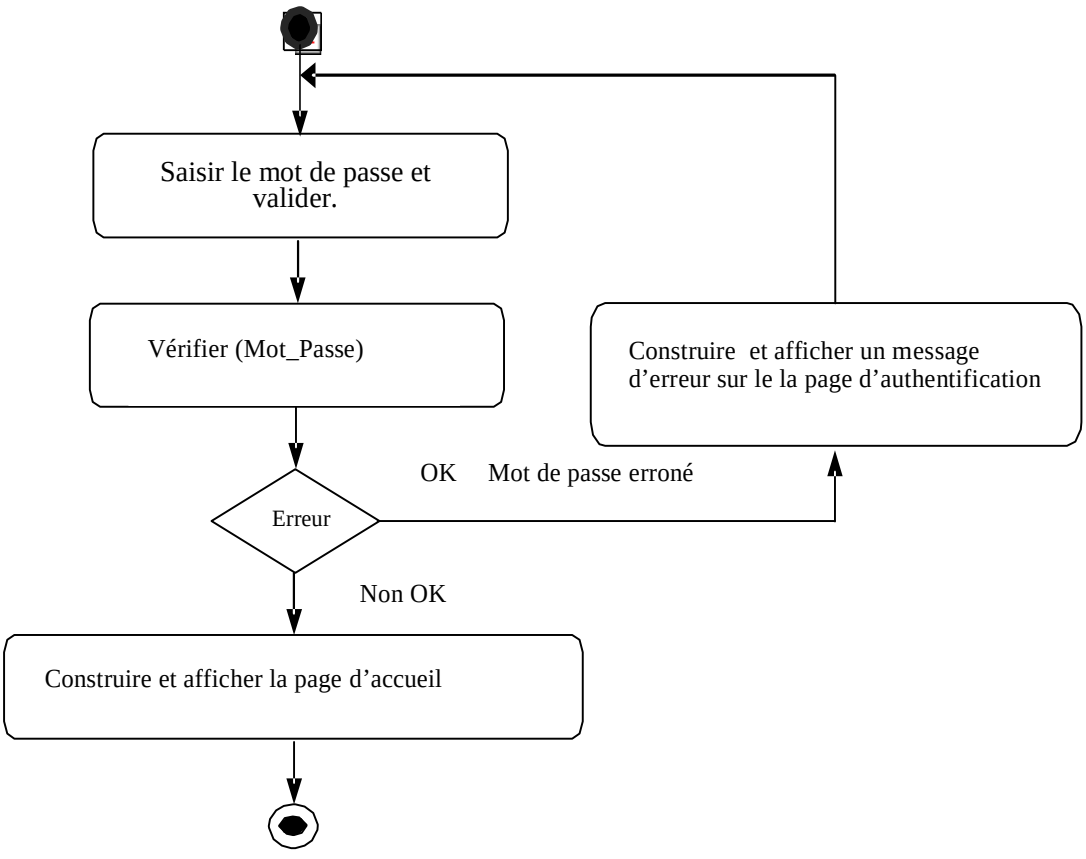


Figure VI.16 Diagramme d'activité de cas d'utilisation « S'authentifier ».

Diagramme d'activité de cas d'utilisation «*Gérer les notes*».

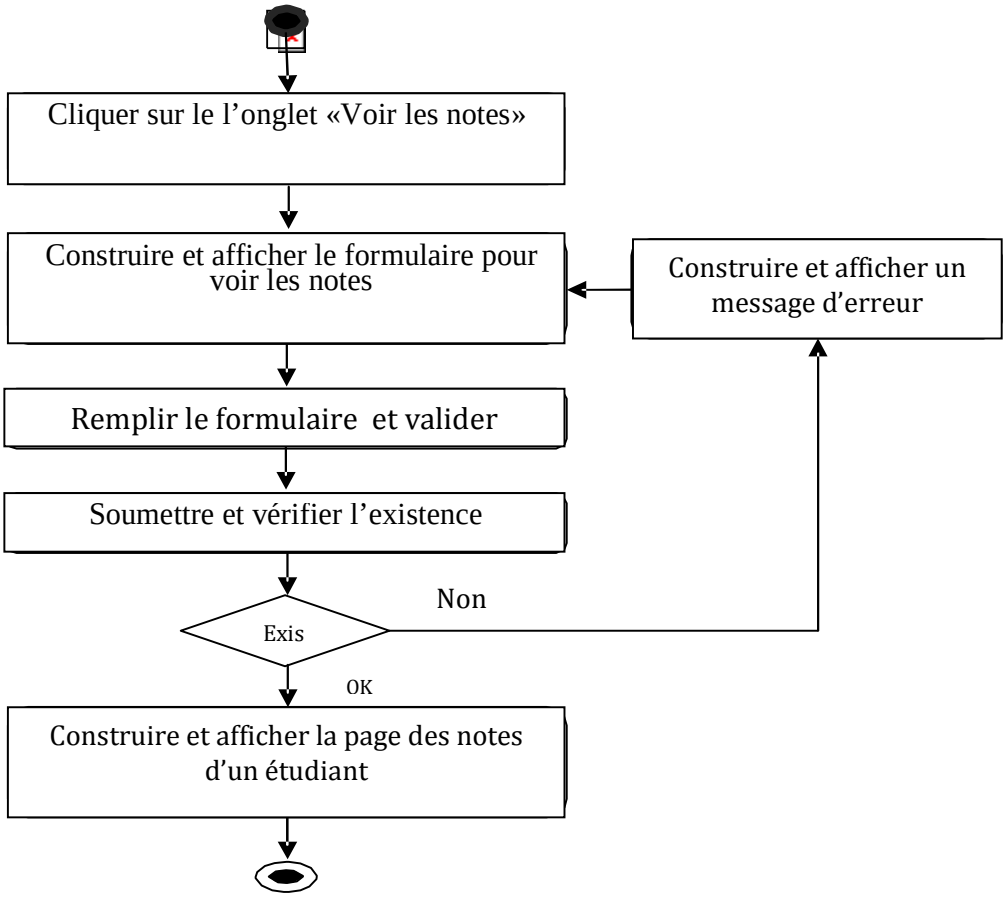


Figure VI.17 Diagramme d'activité de cas d'utilisation «*Gérer les notes* ».

Diagramme d'activité de cas d'utilisation «*Edition des relevés des notes*».

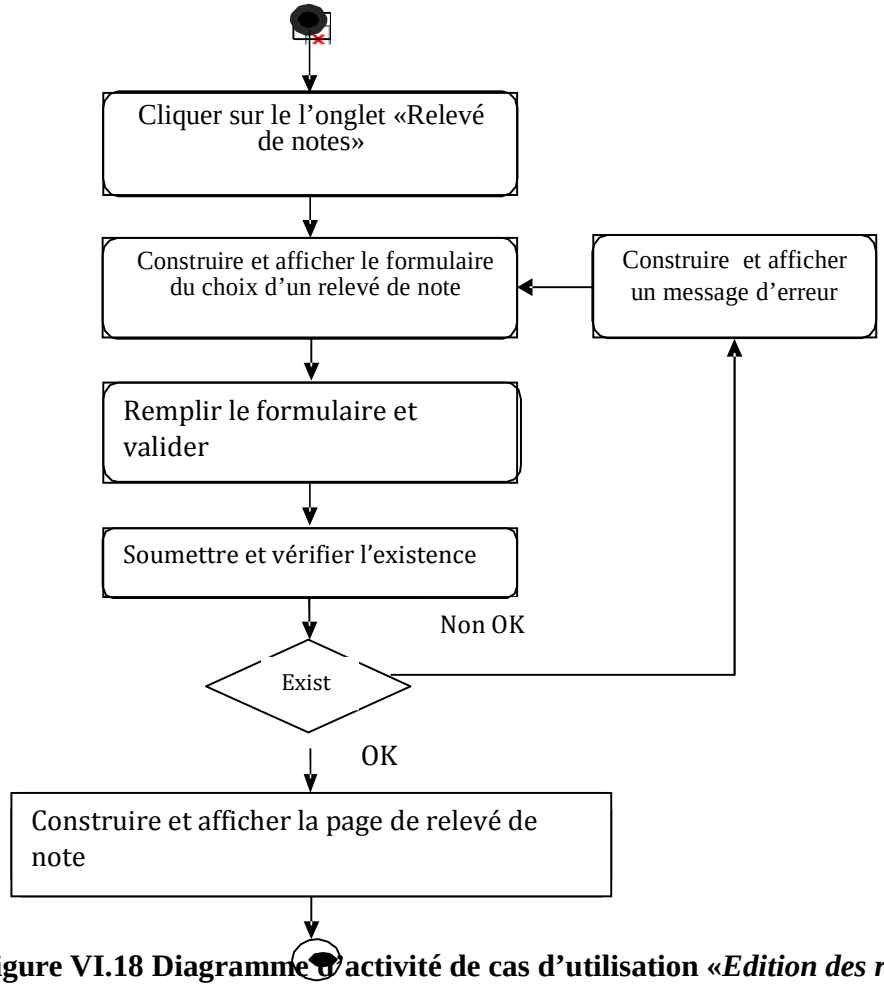


Figure VI.18 Diagramme d'activité de cas d'utilisation «*Edition des relevés des notes*».

Diagramme d'activité de cas d'utilisation «*Changer le mot de passe*».

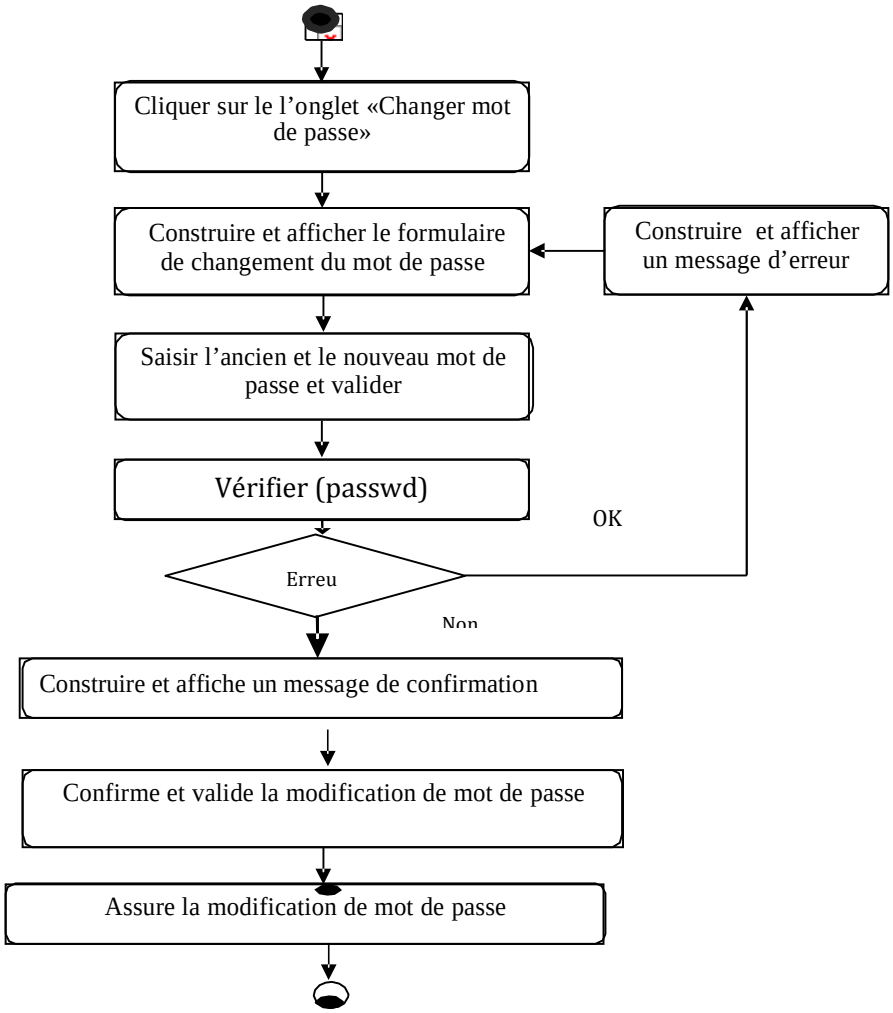


Figure VI.19 Diagramme d'activité de cas d'utilisation «*Changer le mot de passe*».

Diagramme d'activité de cas d'utilisation «*Création de Compte* ».

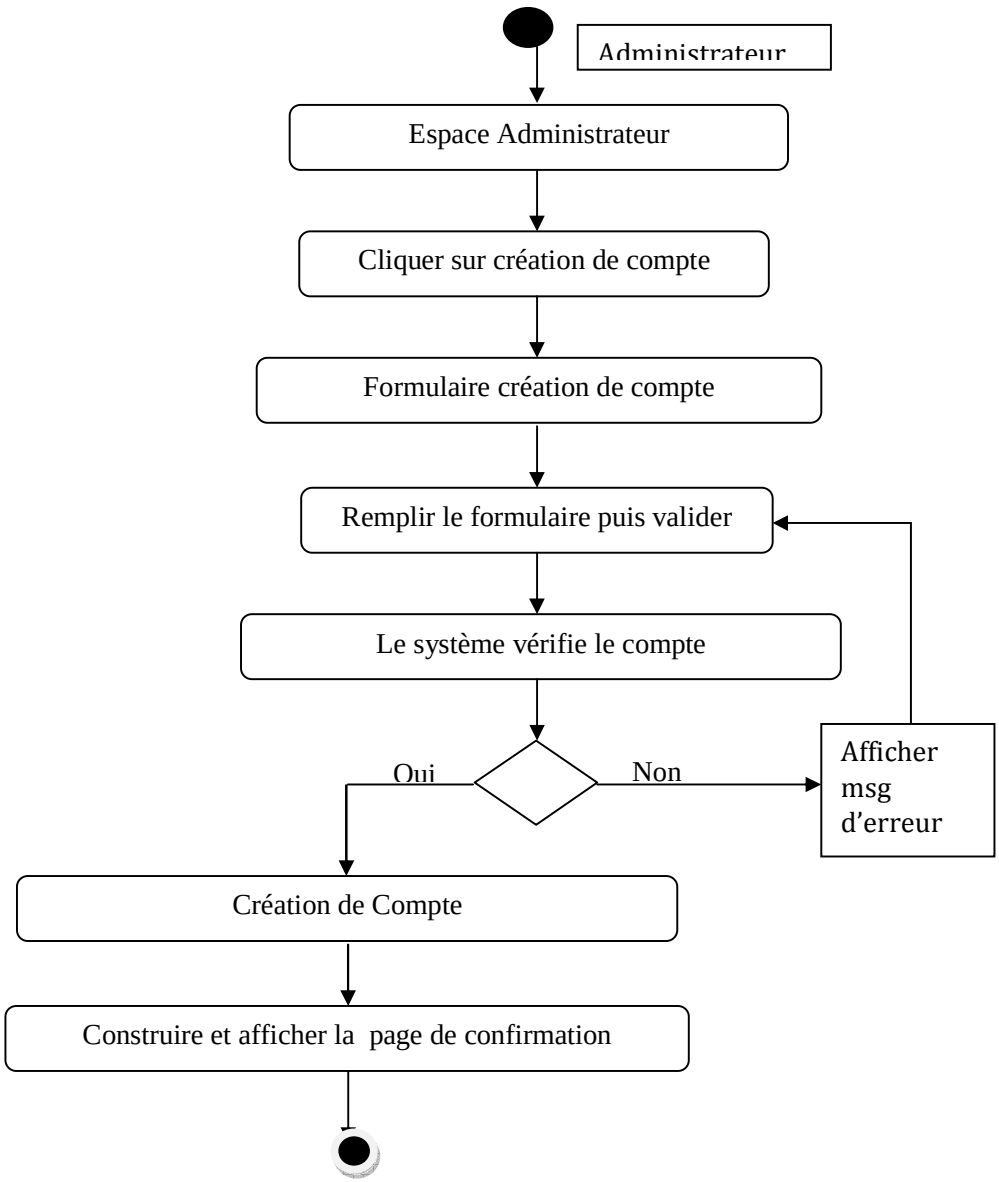


Figure VI.20 Diagramme d'activité de cas d'utilisation «*Création de Compte*».

Diagramme d'activité de cas d'utilisation «*Affectation des Enseignants*».

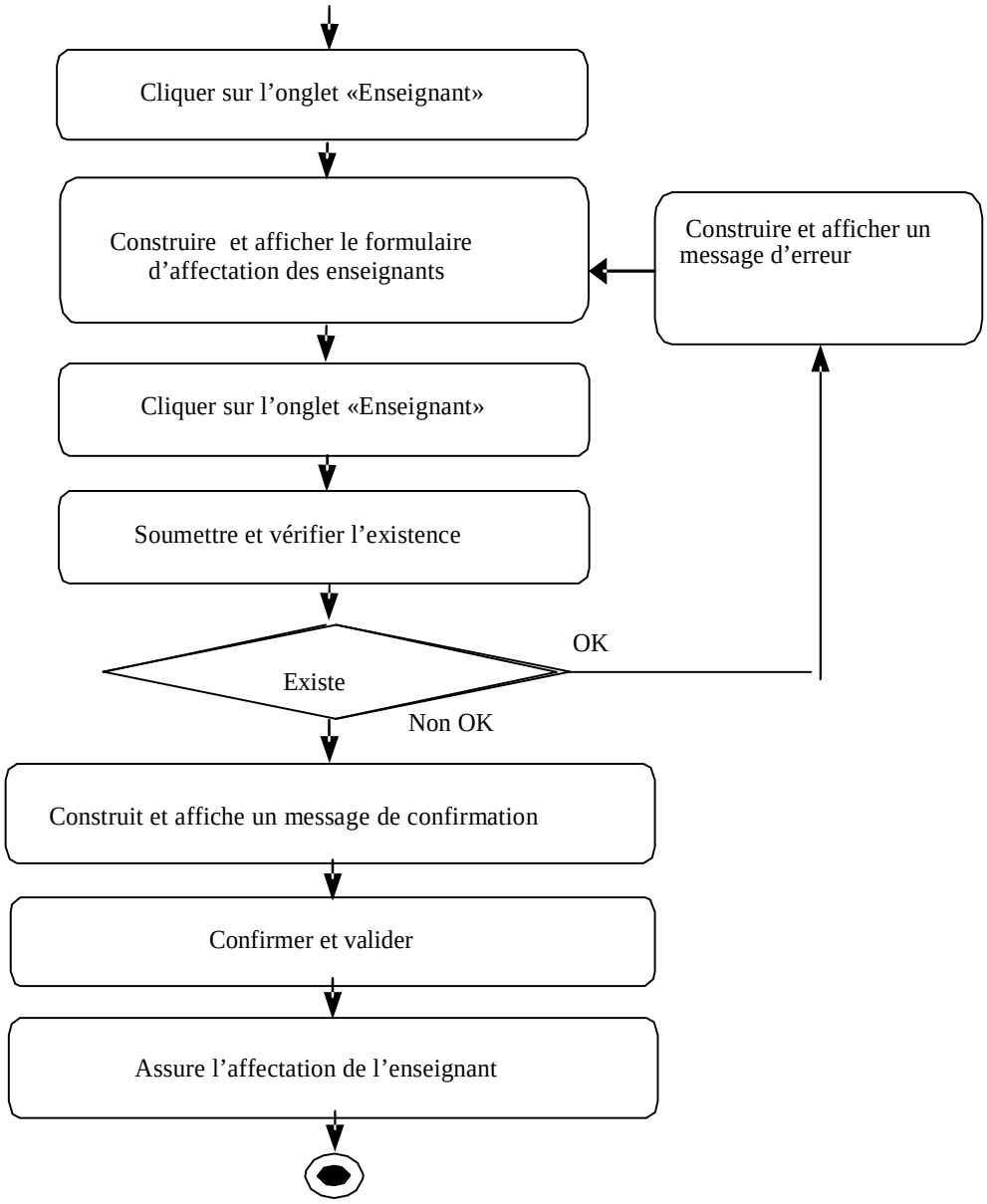


Figure VI.21 Diagramme d'activité de cas d'utilisation «*Affectation des Enseignants*».

Diagramme d'activité de cas d'utilisation «Etablir les Sections et les Groupes» :

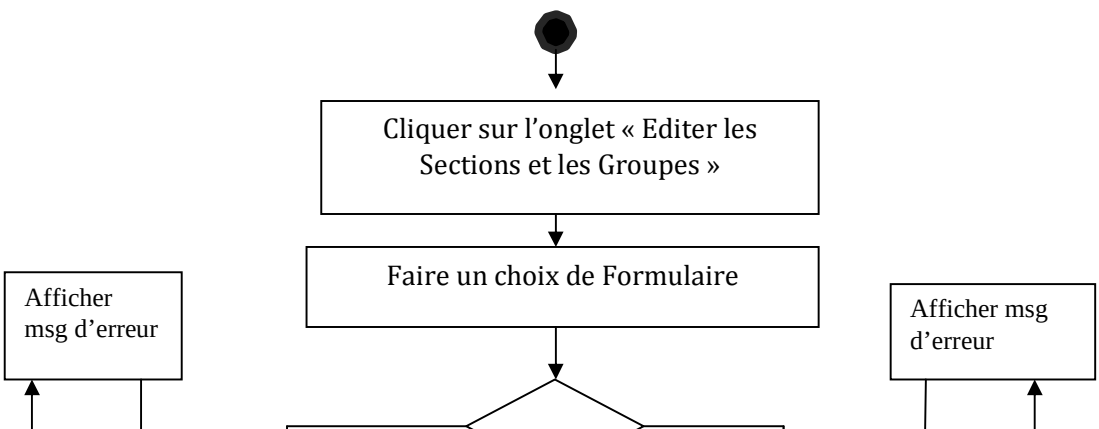


Figure VI.22 Diagramme d'activité de cas d'utilisation «*Etablir les Sections et les Groupes*».

Diagramme d'activité de cas d'utilisation «Voir les Notes» :

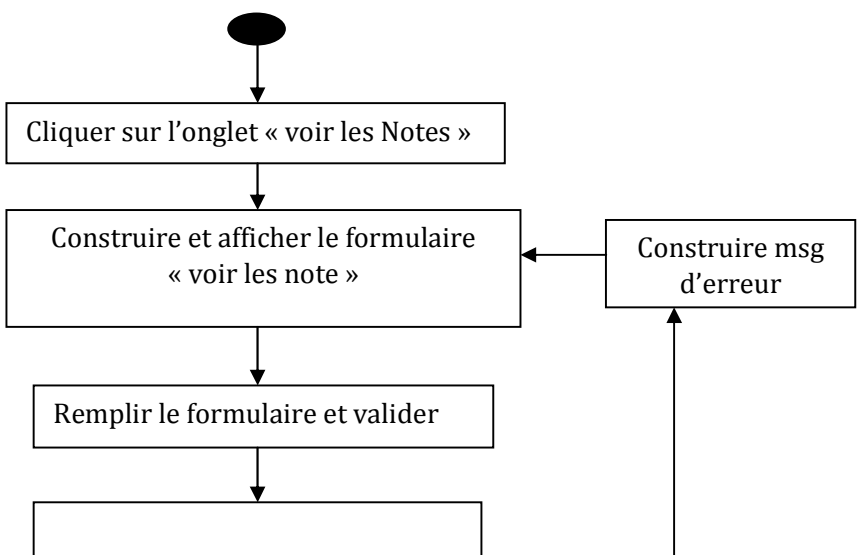


Figure VI.23 Diagramme d'activité de cas d'utilisation «*Voir les Notes* ».

VI.5.2.3. Diagrammes de classe :

Un diagramme de classe représente la vue logique du système, l'interêt majeur de ces diagrammes est de modéliser les entités d'un système.

Diagramme de classe détaillé du cas d'utilisation « authentification ».

« Methode Java »
Verifier la saisie

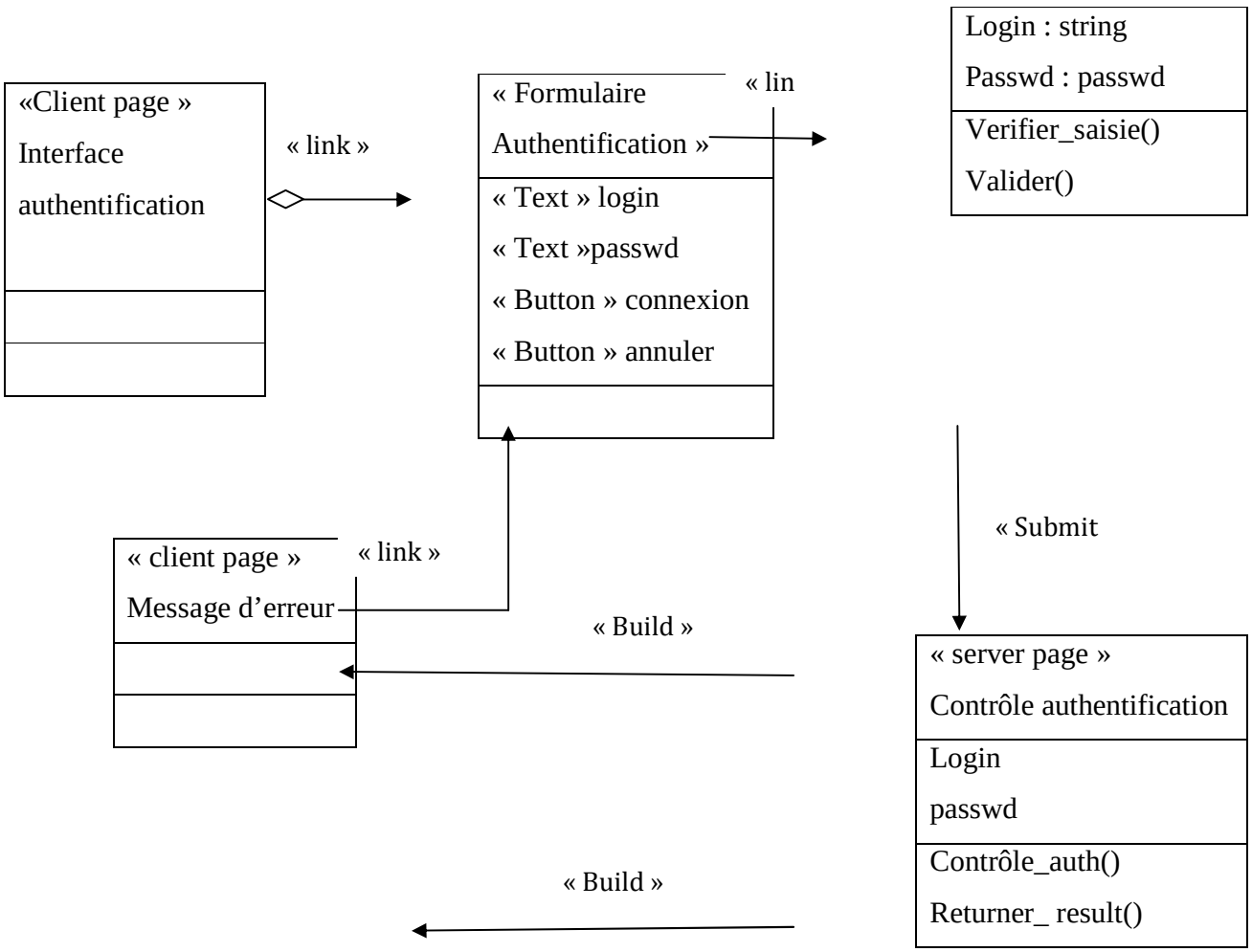


Figure **VI.24 Diagramme de classe détaillé du cas d'utilisation « authentification ».**

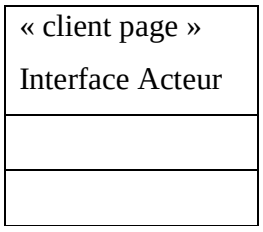


Diagramme de classe de cas d'utilisation « Gestion des Notes ».

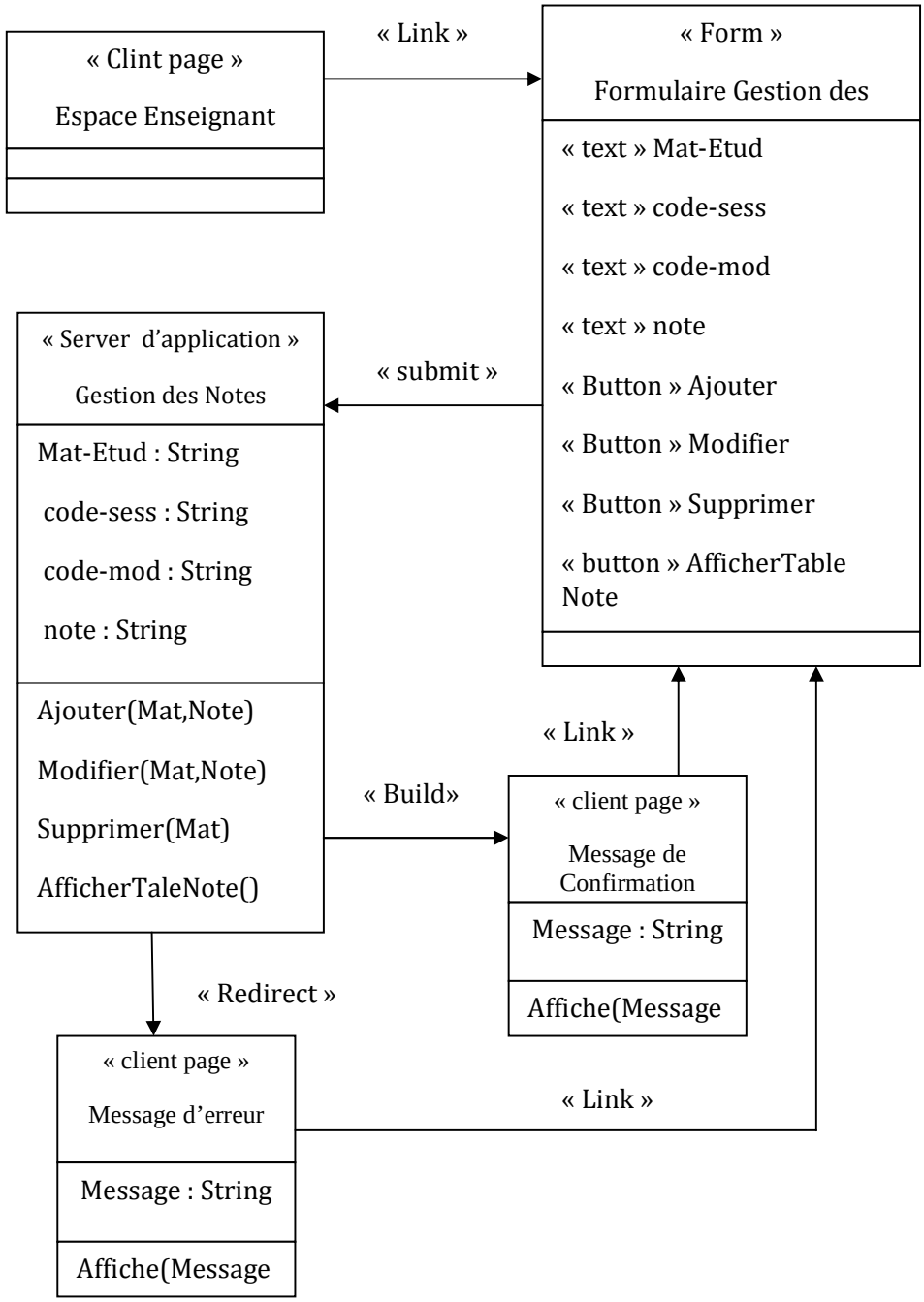


Figure VI.25 Diagramme de classe de cas d'utilisation « Gestion des Notes ».

Diagramme de classe de cas d'utilisation « Editer les relevé Notes ».

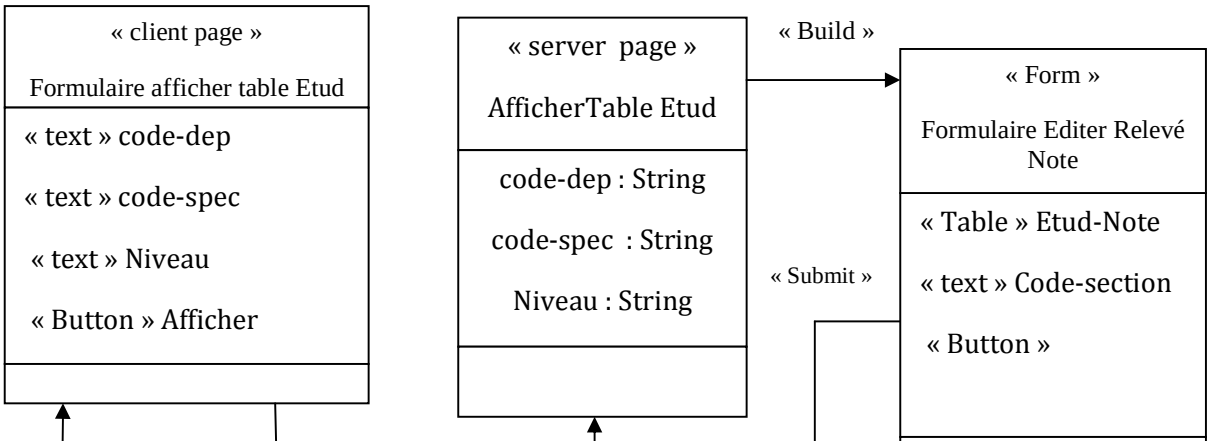


Figure VI.26 Diagramme de classe de cas d'utilisation « Editer les relevé Notes ».

Diagramme de classe de cas d'utilisation « Changer le Mot de Passe ».

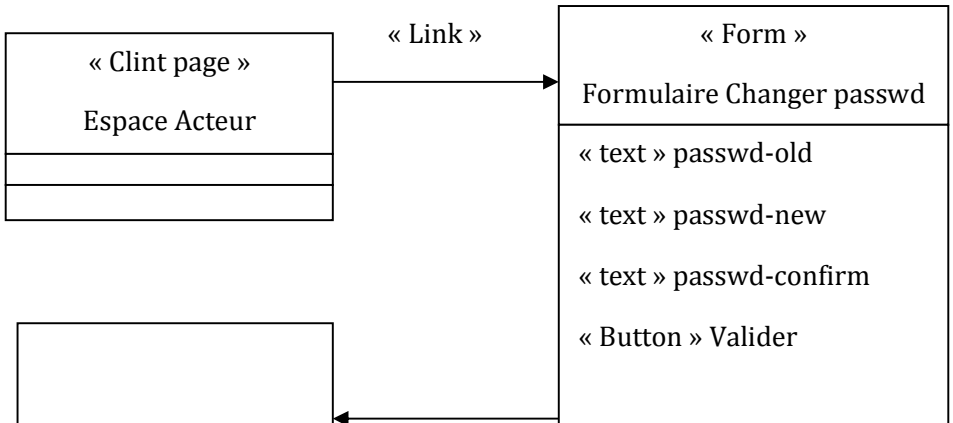


Figure VI.27 Diagramme de classe de cas d'utilisation « Changer le Mot de Passe ».

Diagramme de classe de cas d'utilisation « Création de Compte ».

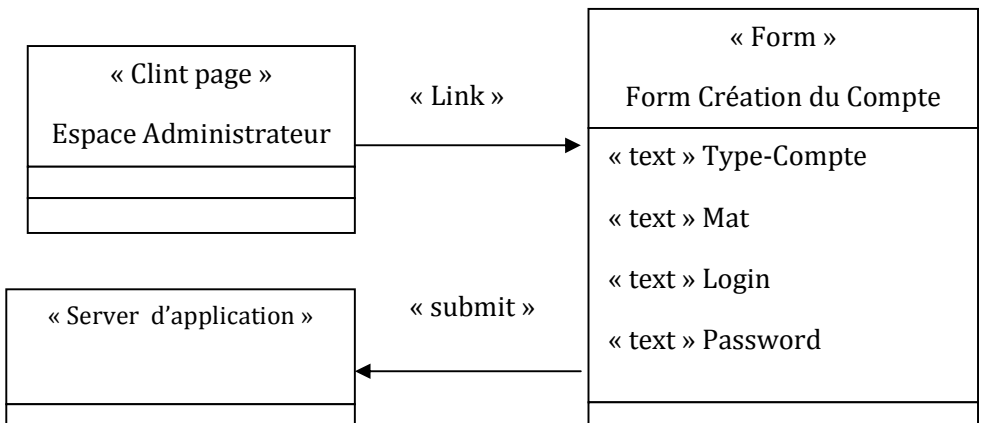


Figure VI.28 Diagramme de classe de cas d'utilisation « Création de Compte ».

Diagramme de classe détaillé du cas d'utilisation « Affectation des Enseignants».

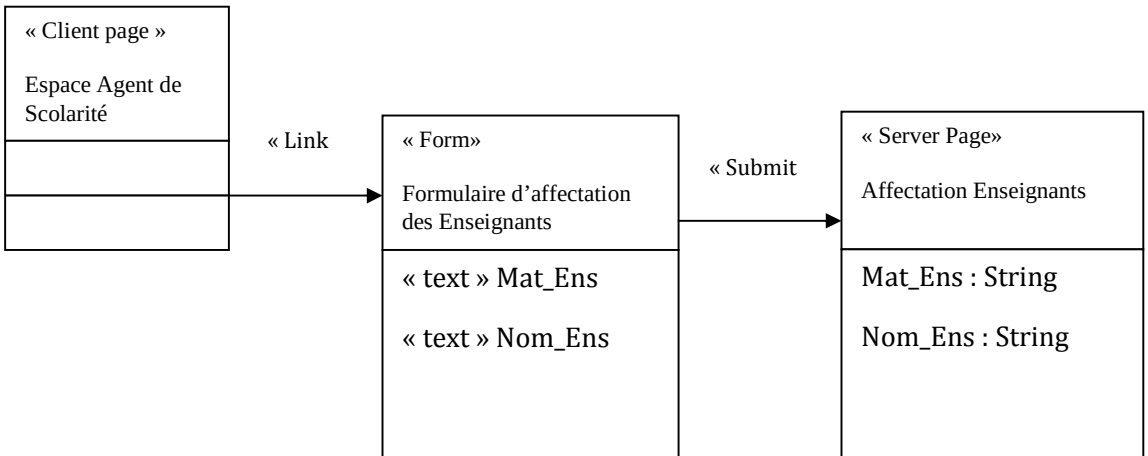


Figure VI.29 Diagramme de classe détaillé du cas d'utilisation « Affectation des Enseignants».

Diagramme de classe de cas d'utilisation « Etablir les sections et les groupes ».

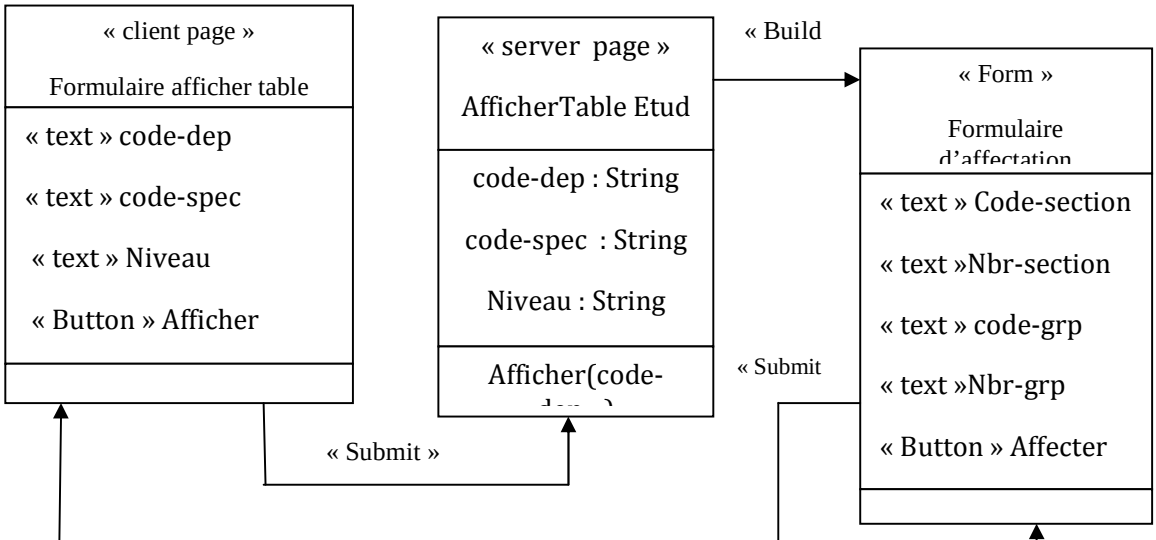


Figure VI.30 Diagramme de classe de cas d'utilisation « Etablir les sections et les groupes ».

Diagramme de classe de cas d'utilisation « Voir les Notes ».

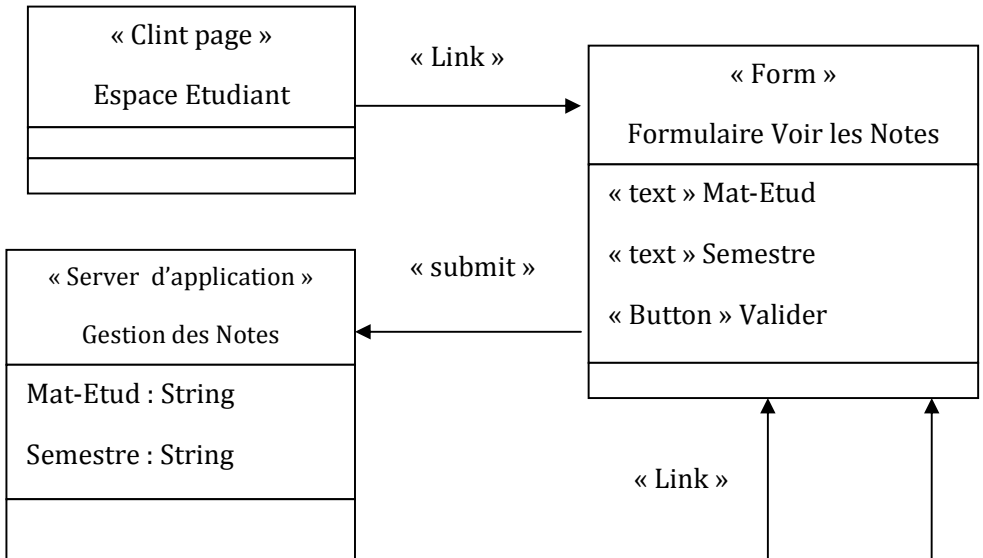


Figure VI.31 Diagramme de classe de cas d'utilisation « Voir les Notes ».

VI.5.2.4. Diagramme de classe global

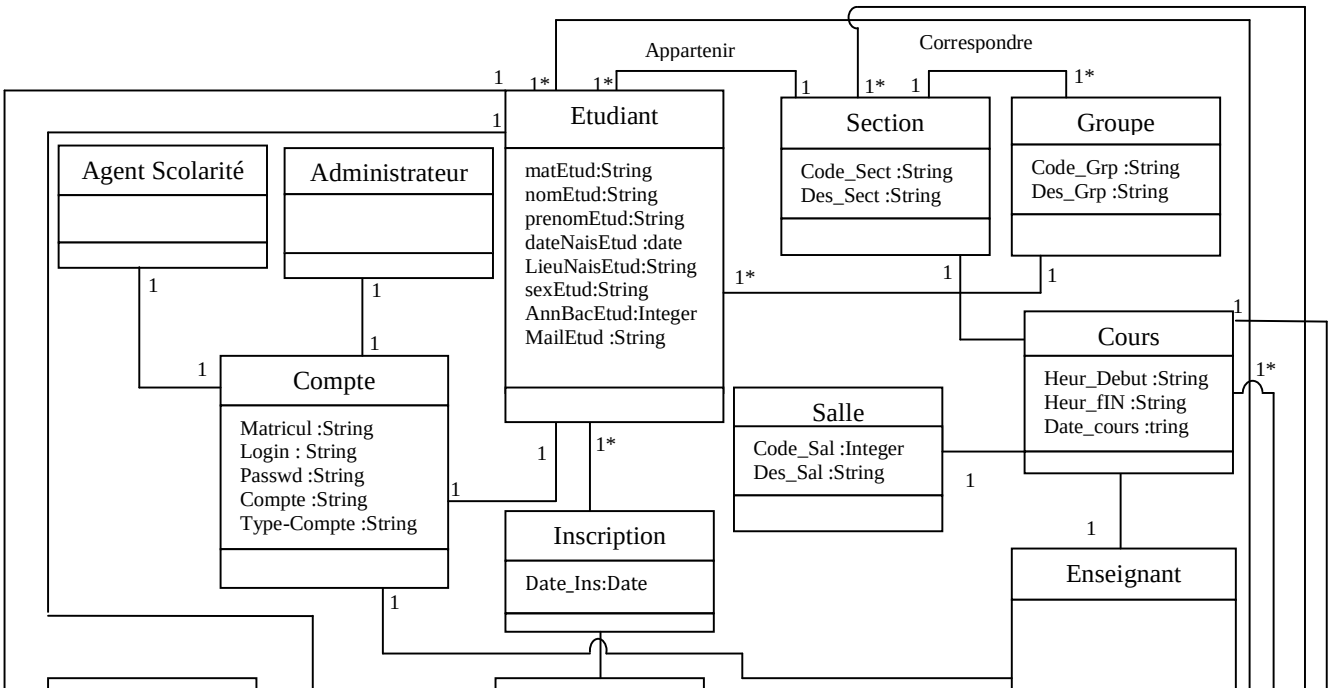


Figure VI.32 Diagramme de classe Global.

VI5.2.5. Implémentation de la base de données :

En se basant sur le diagramme de classe globale que nous avons présenté, nous aboutissons à l'élaboration du modèle physique de données composé des tables suivantes :

Table Enseignant :

Nom du champ	Description	Type	Clé(s)
matEns	Matricule de l'enseignant	Varchar (20)	Primaire
nomEns	Nom de l'enseignant	Varchar (30)	
prenomEns	Prénom de l'enseignant	Varchar (30)	
dateEns	Date de naissance	date	

LieuNaisEns	Lieu de Naissance	Varchar(30)	
AdrEns	Adresse Enseignant	Varchar(45)	
SexEns	Sexe de l'enseignant	Varchar(8)	
gradeEns	Grade de l'enseignant	Varchar (20)	
StatEns	Statut de l'enseignant	Varchar (14)	
MailEns	Adresse email de l'enseignant	Varchar (40)	

Table Etudiant :

Nom du champ	**Description**	**Type de donn**	**Clé(s)**
matEtud	Matricule de l'étudiant	Varchar (20)	Primaire
nomEtud	Nom de l'étudiant	Varchar (30)	
prenomEtud	Prénom de l'étudiant	Varchar (30)	
dateNaisEtud	Date de naissance de l'étudiant	date	
LieuNaisEtud	Lieu de naissance Etudiant	Varchar(30)	
sexEtud	Sexe de l'étudiant	Varchar (8)	
AnnBacEtud	Année obtention du BAC	Integer	
NivEtud	Niveau d'etude	Integer	
Code_Form	Code de formation	Varchar(8)	Etrangére
MailEtud	Adresse email	Varchar (40)	

Table Compte :

Nom du champ	**Description**	**Type de données**	**Clé(s)**
Matricul	Matricule de la personne qui possède le compte	Varchar(14)	Primaire
Login	Nom d'utilisateur	Varchar(25)	
passwd	Mot de passe pour le compte	Varchar(15)	

Type-Compte	Type de compte(admin, Ens, Agent de Scol, Etud)	Varchar(15)	

Table Module :

Nom du champ	Description	Type de données	Clé(s)
Code_Mod	Code du Module	Varchar(25)	primaire
Des_Mod	Description du Module	Varchar(45)	
Coef_Mod	Coefficient	Integer	
Type_Mod	Type de Module	Varchar(25)	
Code_UE	Code de l'Unité d'Enseignement	Varchar(3)	Etrangére
Code_Form	Code de formation	Varchar(10)	Etrangére

Table Unite d'Enseignement UE :

Nom du champ	Description	Type de données	Clé(s)
Code_UE	Code de l'unité d'Enseignement	Varchar(3)	Primaire
Credit_UE	Credit de l'unité d'Enseignement	Integer	
Des_UE	Description de l'unité d'Enseignement	Varchar(20)	
Code_Sem	Code de semestre	Integer	Etrangère

Table Departement :

Nom du champ	Description	Type de données	Clé(s)
Code_Dep	Code de département	Varchar(8)	Primaire
Des_Dep	Description de Département	Varchar(20)	

Table Emploi_de_temps :

Nom du champ	Description	Type de données	Clé(s)
Code_Mod	Code du module	Varchar(15)	Primaire
MatEns	Matricule Enseignant	Varchar(20)	Primaire
Code_Sect	Code de Section	Varchar(2)	Primaire
Code_Grp	Code de groupe	Varchar(2)	Primaire
Num_sem	Numéro du semestre	Int (2)	Primaire
Code_Sal	Code de la salle	Integer	Primaire

Table Note :

Nom du champ	Description	Type de données	Clé(s)
matEtud	Matricule de l'étudiant	Varchar (20)	Primaire
matEns	Matricule de l'enseignant	Varchar (20)	Primaire
codeMod	Code du module	Varchar (15)	Primaire
Code_Sess	Session de l'examen	Varchar(15)	Primaire
Note	Note de l'examen	Float	

Table Inscription :

Nom du champ	Description	Type de données	Clé(s)
Code_sem	Code de semestre	Integer	Primaire
MatEtud	Matricule Etudiant	Varchar(20)	Primaire
Date_Insc	Date d'inscription	Date	

Table Programme :

Nom du champ	Description	Type de données	Clé(s)
Intitul_Pgm	Intitulé de programme	Varcahr(20)	Primaire
Code_Mod	Code de module	Varchar(15)	Etrangére
Des_Pgm	Description du programme	Varchar(30)	

Table Session :

Nom du champ	Description	Type de données	Clé(s)
Code_Sess	Code de la session	Vatchar(15)	Primaire
Des_Sess	Description de la session	Varchar(30)	

Table Cours :

Nom du champ	Description	Type de données	Clé(s)
Code_Sect	Code de la section	Varchar (20)	Primaire
Code_Sal	Code de la salle	Varchar (20)	Primaire
MatEns	Matricule Enseignant	Varchar (20)	Primaire
Heur_Debut	Heur de début de cours	Date	
Heur_fIN	Heur fin de cours	Date	
Code_mod	Code de module	Varchar(20]	Etrangère

Table Section

Nom du champ	Description	Type de données	Clé(s)
Code_Sect	Code de Section	Vatchar(2)	Primaire
Des_Sect	Description de la Section	Varchar(30)	

Table Groupe

Nom du champ	Description	Type de données	Clé(s)
Code_Grp	Code de groupe	Vatchar(2)	Primaire
Des_Grp	Description de groupe	Varchar(30)	

Table Salle :

Nom du champ	Description	Type de données	Clé(s)
Code_Sal	Code de la salle	Integer	Primaire
Des_Sal	Description de la salle	Varchar(30)	

Table Formation :

Nom du champ	Description	Type de données	Clé(s)
Code_Form	Code de Formation	Varchar(8)	Primaire
Des_Form	Description de la Formation	Varchar(30)	
Code_Dep	Code de departement	Varchar(20)	Etrangère
Code_Ens_Res	Enseignant responsable	Varchar(20)	Etrangère

Table Niveau d'etude :

Nom du champ	Description	Type de données	Clé(s)
Niv_Etud	Niveau d'étude	Varchar(8)	Primaire
Des_Niv	Description Niveau	Varchar(30)	

Table Affectation :

Nom du champ	Description	Type de données	Clé(s)
Code_Ens	Code de l'enseignant	Varchar(10)	Primaire
Code_Dep	Code de département	Varchar(8)	Primaire
Code_For	Code de formation	Varchar(30)	Primaire
Code_Niv	Code de niveau	Varchar(20)	Primaire
Code-Mod	Code de module	Varchar(20)	Primaire
Cod_Sec	Code de section	Varchar(20)	Primaire

Conclusion

Dans ce chapitre, nous nous sommes intéressés à la conception de l'application. Nous avons d'abord présenté l'approche globale de conception que nous avons suivi pour l'élaboration de notre application. Par la suite nous avons cité les fonctionnalités qu'elle offrait aux différents acteurs. Nous nous sommes également appuyé sur les diagrammes offerts par le langage UML afin de nous approfondir dans l'analyse et la conception et ce en modélisant graphiquement certains cas d'utilisation de l'application.

Chapitre V

Réalisation

Introduction

Dans l'étude conceptuelle nous avons expliqué notre système avec ses différents services Web ; maintenant, nous allons le détailler avec l'étape de la réalisation.
En outre, la réalisation concerne aussi le déploiement de l'application dans son environnement réel. Nous allons présenter l'implémentation et la mise en œuvre de ces Service Web.
Nous présenterons en premier les langages et le SGBD utilisés dans la réalisation, puis les différentes interfaces de l'utilisateur, ainsi que chacune de leurs fonctionnalités.

I. Description des outils de développement

I.1. Langage de programmation (Java)

Java est un langage de programmation informatique orienté objet créé par James Gosling et Patrick Naughton de Sun Microsystems. Mais c'est également un environnement d'exécution.
Java peut être séparé en deux parties. D'une part, le programme écrit en langage Java et d'autre part, une machine virtuelle (JVM) qui va se charger de l'exécution du programme
Java. C'est cette plateforme qui garantit la portabilité de Java. Il suffit qu'un système ait une machine virtuelle Java pour que tout programme écrit en ce langage puisse fonctionner.

I.2. IDE (NetBeans)

C'est un environnement de développement intégré (IDE) pour Java, placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et des pages web). NetBeans est disponible sous Windows, Linux, Solaris (sur x86 et SPARC), Mac OS X et Open VMS.

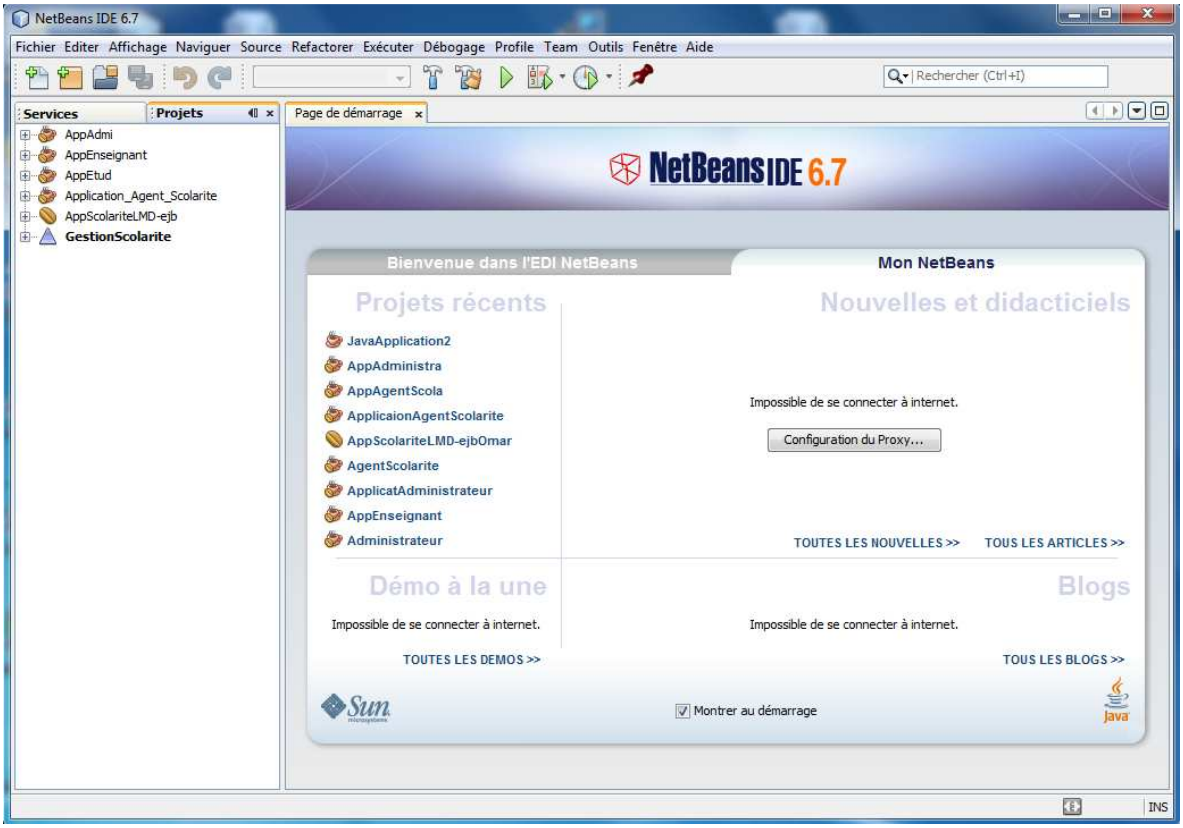


Figure V.1 Interface de l'IDE Netbeans.

I.3. Le serveur d'application (Glassfish)

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

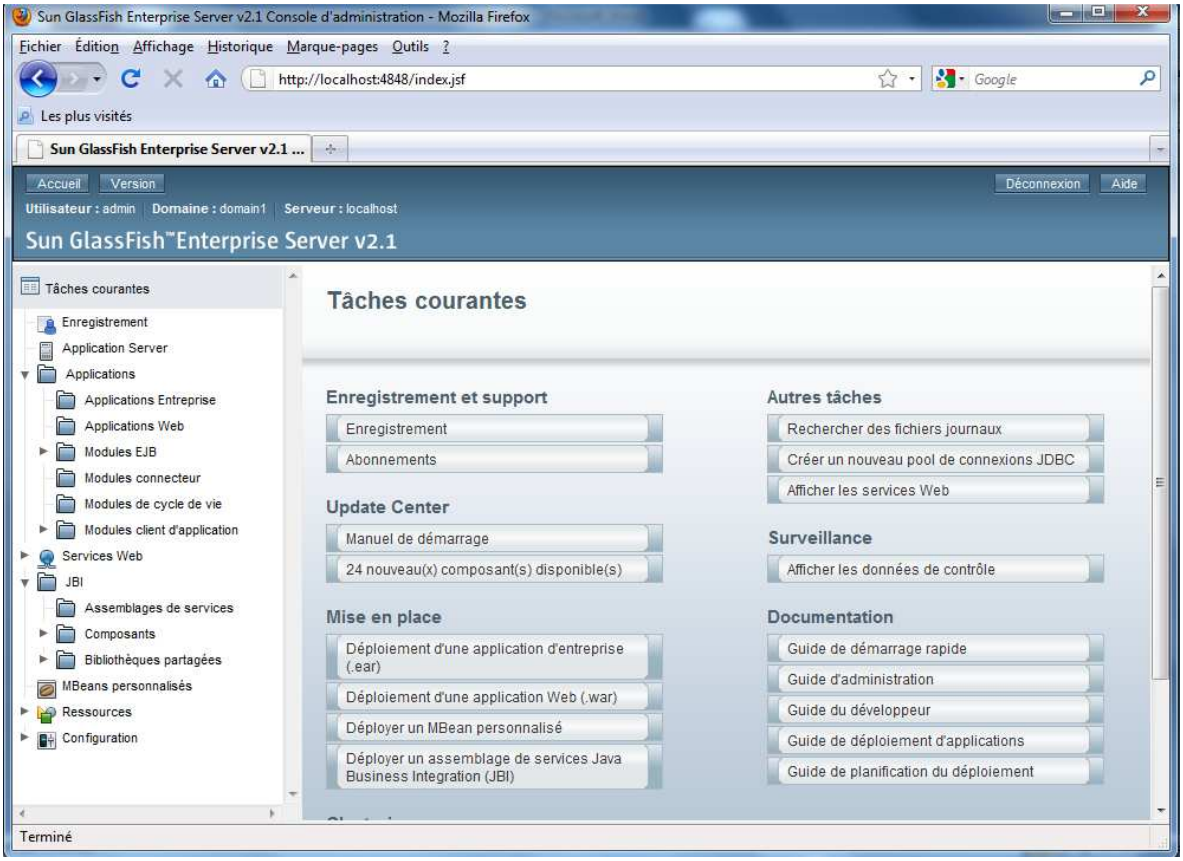


Figure V.2 Interface de serveur de déploiement GlassFish.

I.4. Le SGBD (MYSQL)

MySQL est un gestionnaire de base de données libre. Il est très utilisé dans les projets libres et dans le milieu industriel.

MySQL est un SGBD relationnel développé dans un souci de performances élevées. Il est multi threads, multiutilisateurs.

On a opté pour MySQLWorkbench car cet outil n'est pas indispensable mais très efficace et permet de manipuler MySQL de manière très simple. Cette interface graphique est en fait une couche de manipulation de MySQL.

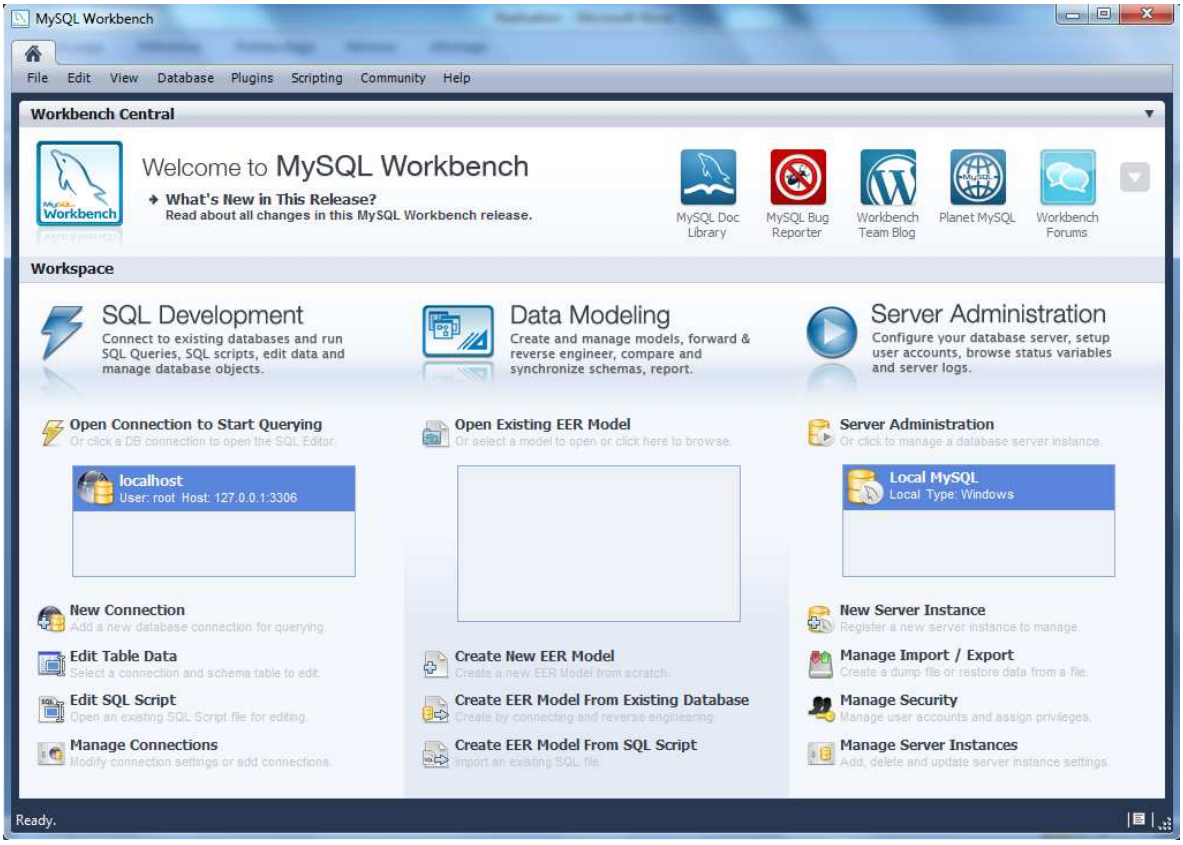


Figure V.3 Interface de MySQLWorkbench

II. présentation de l'application

Pour la réalisation proprement dite de notre système, on a opté pour la technologie des **web services JEE**. Notre application suivra donc l'architecture JEE en couche suivante :

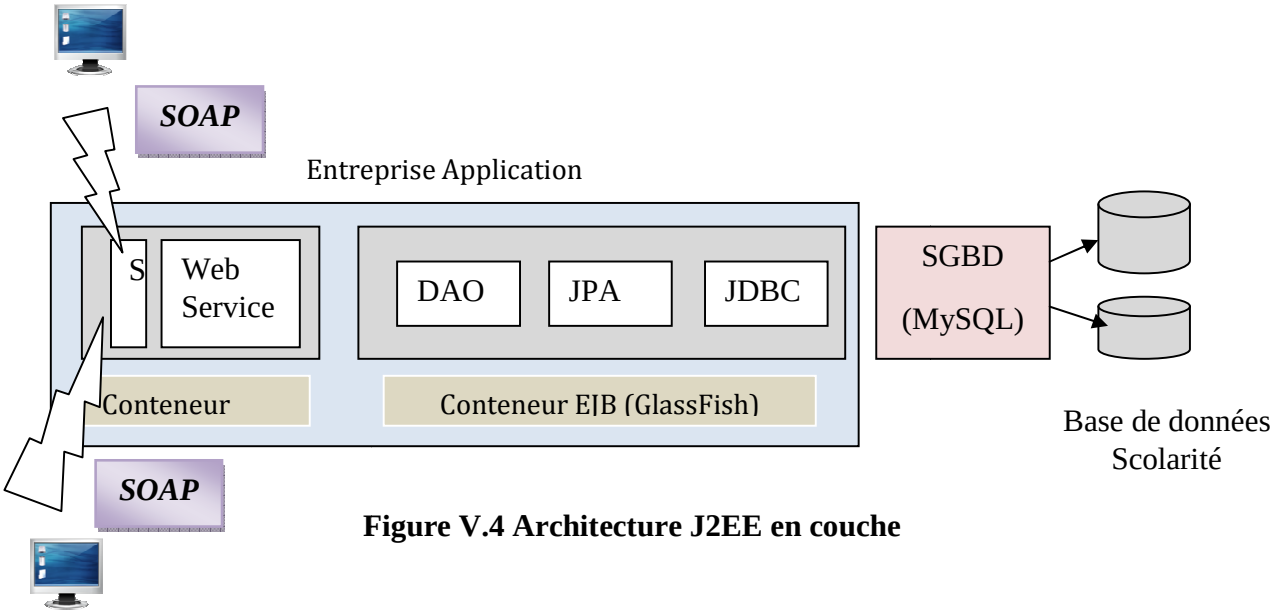


Figure V.4 Architecture J2EE en couche

II.1. JDBC : cette couche gère la connexion avec la (ou les) base(s) de données. Ici on utilisera la notion de pool de connexion. Un pool de connexion est un ensemble de connexions avec la base de données déjà instanciées. Cela permet aux requêtes de s'exécuter plus rapidement. On peut venir connecter plusieurs couches JPA sur la couche JDBC si nécessaire.

II.2. JPA : la couche JPA (Java Persistence Annotation) est une couche d'abstraction de la couche JDBC. Elle permet notamment de faire du Mapping Relationnel-Objet (ORM, Object-Relationnal Mapping en anglais) qui consiste à modéliser la base de données sous forme d'objets pour une manipulation plus simple à travers le code Java (requêtes pré-écrites, gestion des liens entre les tables,…). Généralement la couche JPA contient une classe (entité) par table, des contrôleurs (fonctions de base implémentées) et des gestionnaires d'exceptions.

II.3. DAO : Cette couche représente l'intelligence de l'application. Elle est composée d'un ensemble d'interfaces locales (local) et distantes (remote). Les DAO (Data Access Object) permettent d'accéder aux objets et proposent des méthodes de CRUD (Create, Read, Update, Delete). Un EJB (Entreprise Java Bean) sera piloté à partir d'une autre application distante ou locale (client EJB).

II.4. Web Services : Cette couche a pour but de définir des services qui pourront être appelés selon le protocole SOAP. Ainsi les informations pourront circuler entre les applications sous forme de messages XML. Cela peut servir à faire communiquer deux applications qui peuvent être codées dans deux langages différents, en local ou à distance.

III. Implémentation des différentes parties de l'application

III.1. Création de la base de données :

Pour l'implémentation de cette partie, Nous avons créé avec le SGBD MYSQL une base de données [bdd-scolarite] contenant les différentes tables de chapitre conception.

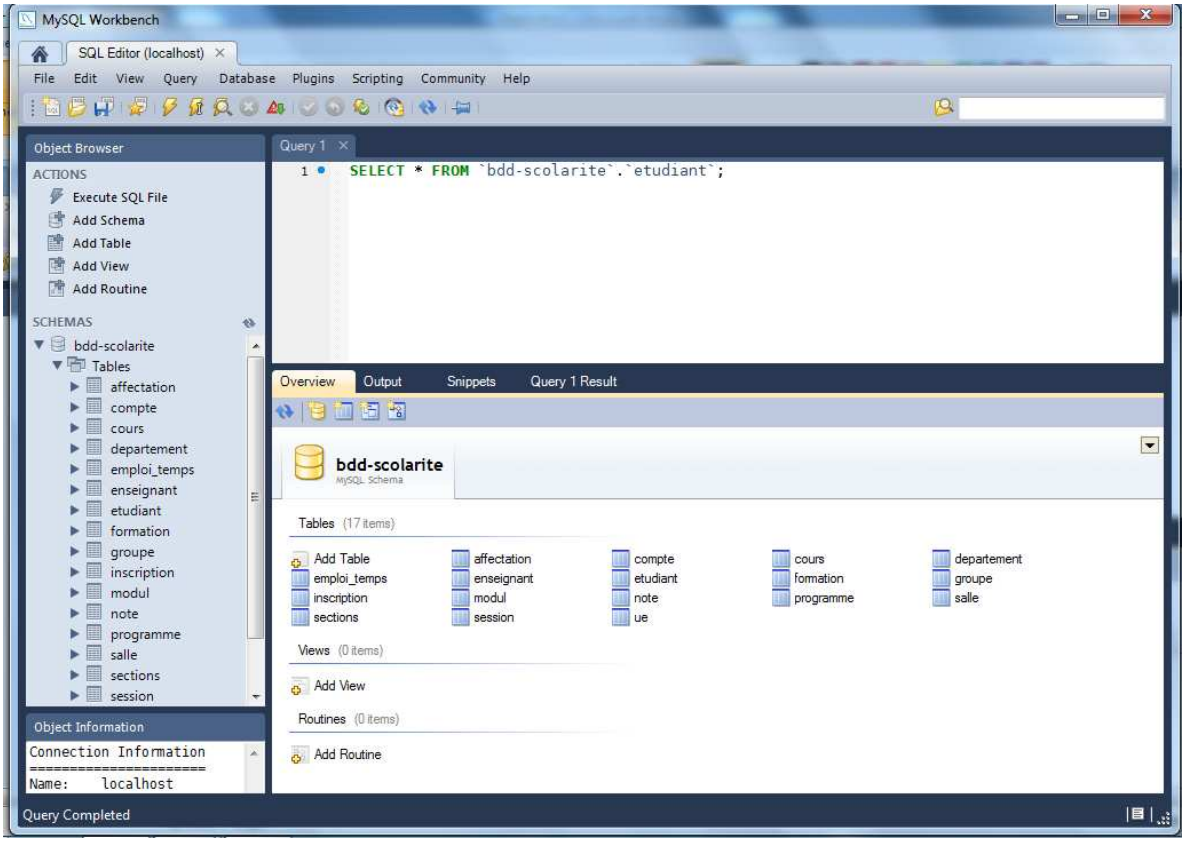


Figure V.5 La base de données « bdd-Scolarité » de l'application

Voici un exemple de script de création d'une des tables :

```
create table Etudiant(
Mat_etud varchar(14) not null,
Nom_etud varchar(25) not null,
Prenom_etud varchar(25) not null,
Date_nais_etud Date not null,
Lieu_nai_etud varchar(30) not null,
Sex_etud varchar(1) not null,
Ann_bac_etud Integer(4) not null,
Niv_etud varchar(15) not null,
Code_form_etud varchar(25) not  null,
Primary Key (Mat_Etud)
);
```

Figure V.6: Script de création de la table Etudiant

Après avoir créé cette base de données, on a établie une connexion NetBeans /MYSQL à l'aide de pilote JDBC.

L'image suivante est une capture d'écran montrant la connexion établie, ainsi que les différentes tables de notre BDD.

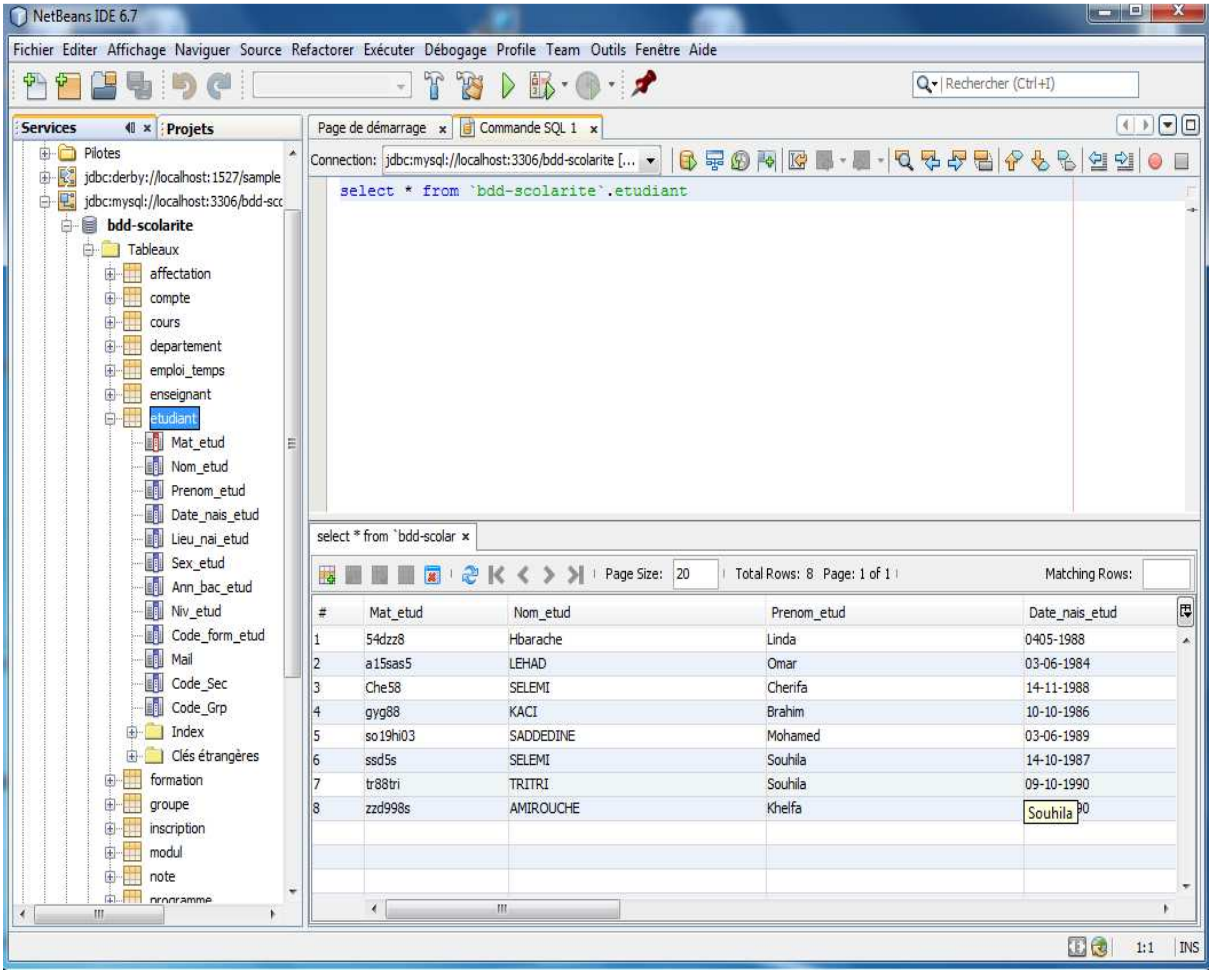


Figure V.7 Les tables de la BDD « bdd-scolarite ».

III.2. Partie Traitement

Dans cette section, nous allons faire une brève description des différents projets NetBeans implémentant cette partie, ainsi les différentes couches JEE qui la constituent.

III.2.1. le projet Gestion de la scolarité

Ce projet est un projet **JEE** de type Entreprise Application, c'est lui qui nous permet le déploiement de notre service web au sein de serveur Glassfish.

Ce dernier est composé de deux modules :

- Un module de type web application qui implémente notre web service.
- Un module EJB dont dépend notre web service.

La figure suivante est une capture d'écrans montrant ce projet sous NetBeans.

Figure V.8. Application J2EE Gestion Scolarité

III.2.1.1. le module AppScolariteLMD-EJB

C'est un projet de type EJB Module de la catégorie Java EE des projets NetBeans.
Ce module construit un composant logiciel distribué (c'est-à-dire déployé sur le serveur distant Glassfish), il implémente les couches : [JDBC], [JPA], [DAO] de notre application.

Afin que ce dernier puisse exploiter les données de notre base de données, en assurant leurs persistances, on a configuré ça sous forme des fichiers XML qui ont été crées dans ce module.

1-Ajout d'une ressource JDBC au serveur Glassfish :

On a ajouté une ressource JDBC au serveur Glassfish afin qu'il puisse accéder à la base de données. Ce qui génère le fichier de configuration [sun-resources.xml] dans le répertoire « Server Resources » du module EJB, dont le contenu est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE resources PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Resource Definitions //EN"
"http://www.sun.com/software/appserver/dtds/sun-resources_1_3.dtd">

<resources>

  <jdbc-resource enabled="true" jndi-name="jdbc/myDatasource" object-type="user" pool-name="ScolaritePool">

    <description/>

  </jdbc-resource>

</resources>
```

Figure V.8. Le code source du fichier de configuration [sun-resources.xml]

2-Création d'une unité de persistance :

L'unité de persistance [persistence.xml] configure la couche JPA : elle indique l'implémentation JPA utilisée, ainsi elle nous permet d'indiquer à Hibernate le type de SGBD à gérer, ce fichier est crée dans le répertoire « Configuration Files » du module EJB, son contenu est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>

<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">

  <persistence-unit name="AppScolariteLMD-ejbPU" transaction-type="JTA">

    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

    <jta-data-source>jdbc/ScolariteJndi</jta-data-source>

    <exclude-unlisted-classes>false</exclude-unlisted-classes>

    <properties/>

  </persistence-unit>

</persistence>
```

Figure V.9. Le code source du fichier de configuration [persistence.xml].

3-Création des entités JPA :

Pour donner une image objet a notre base de données, on a crée un package [EJBentity] dans le répertoire « Source Packages » du module EJB, ce package implémente la couche JPA de l'architecture précédente, il contient des entités JPA.

Chaque entité de ce package encapsule les lignes d'une table de la base de données, donc il y' a autant d'entités dans ce package que de tables dans la base de données.

Voici le code d'une de ces entités :

```
package EJBentity;


import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
@Entity
@Table(name = "compte")
@NamedQueries({@NamedQuery(name = "Compte.findAll", query = "SELECT c FROM
Compte c"), @NamedQuery(name = "Compte.findByMat", query = "SELECT c FROM
Compte c WHERE c.mat = :mat"), @NamedQuery(name = "Compte.findByLogin", query =
"SELECT c FROM Compte c WHERE c.login = :login"), @NamedQuery(name =
"Compte.findByPasswd", query = "SELECT c FROM Compte c WHERE c.passwd =
:passwd"), @NamedQuery(name = "Compte.findByTypeCompte", query = "SELECT c FROM
Compte c WHERE c.typeCompte = :typeCompte")})
public class Compte implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "Mat")
    private String mat;
    @Basic(optional = false)
    @Column(name = "Login")
    private String login;
    @Basic(optional = false)
    @Column(name = "Passwd")
```

```
    private String passwd;
    @Basic(optional = false)
    @Column(name = "TypeCompte")
    private String typeCompte;

    public Compte() {
    }

    public Compte(String mat) {
        this.mat = mat;
    }

    public Compte(String mat, String login, String passwd, String typeCompte) {
        this.mat = mat;
        this.login = login;
        this.passwd = passwd;
        this.typeCompte = typeCompte;
    }

    public String getMat() {
        return mat;
    }

    public void setMat(String mat) {
        this.mat = mat;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getPasswd() {
```

```
        return passwd;
    }

    public void setPasswd(String passwd) {
        this.passwd = passwd;
    }

    public String getTypeCompte() {
        return typeCompte;
    }

    public void setTypeCompte(String typeCompte) {
        this.typeCompte = typeCompte;
    }
     Compte other = (Compte) object;
        if ((this.mat == null && other.mat != null) || (this.mat != null &&
  !this.mat.equals(other.mat))) {
            return false;
        }
        return true;
    }
}
```

Figure V.10. Le code source de l'entité Compte.

4-Création de la couche EJB d'accès aux entités JPA :

Afin de pouvoir exploiter nos objets (donc manipuler nos données), on a crée un package [EJBSession]. Qui contient par exemple une classe CompteFacade.java:

- Une classe java [CompteFacade.java] qui implémente l'objet **EntityManager** qui gère l'accès au contexte de persistance, elle interroge nous classes objets avec des requêtes de type JPQL (Java Persistance Query Language).

Voici le code d'une requête JPQL qui nous permet de récupérer un patient par son identifiant :

```
package EJBsession;

import EJBentity.Compte;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
@Stateless
public class CompteFacade implements CompteFacadeLocal, CompteFacadeRemote {
    @PersistenceContext
    private EntityManager em;

    public void create(Compte compte) {
        em.persist(compte);
    }

    public void edit(Compte compte) {
        em.merge(compte);
    }

    public void remove(Compte compte) {
        em.remove(em.merge(compte));
    }

    public Compte find(Object id) {
        return em.find(Compte.class, id);
    }

    public List<Compte> findAll() {
        return em.createQuery("select object(o) from Compte as o").getResultList();
    }

}
```

Figure V.11. Le code source EJB session qui manipule l'entité Compte.

Une interface locale [CompteFacade Local], une interface distante [CompteFacade Remote] qui sont drivées de l'interface [CompteFacade] de l'EJB.

Ces interfaces permettent d'exposer les méthodes de l'EJB en locale et a distance, assurant ainsi la réutilisabilité de ce module par d'autre application.

5-Création d'une couche EJB session ImplMethodes:

Afin de pouvoir exploiter facilement nos objets, on a crée un autre EJB session (ImplMethodes) qui implémente tous les EJB session précédemment crées.

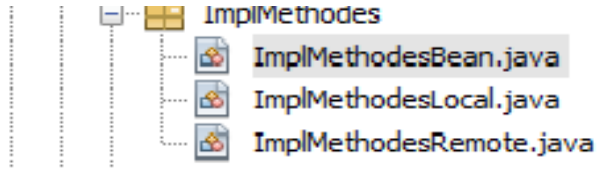


Figure V.12. EJB session qui implémente toutes les méthodes

III.2.1.2. Le module WebServiceImplMethodes

C'est un projet de type web application qui s'exécute sur le serveur Glassfish.
Le service web que nous construisons va utiliser l'EJB précédent, il a aussi besoin de référencer son archive .jar, donc on a ajouté le module EJB aux bibliothèques du ce projet.

On a implémenté le service web avec la classe [**ImplMethodes**], cette classe est la classe principale de notre système. Elle implémente l'interface [IDao] qui est définie dans l'archive de l'EJB, comme l'indique les premières lignes du code suivant:

```
@WebService()
@Stateless()
public class WsImplMethodes {
   @EJB
   private ImplMethodesLocal ejbRef;
   // Add business logic below. (Right-click in editor and choose
```

```
// "Web Service > Add Operation"


@WebMethod(operationName = "AjouterCompte")
public boolean AjouterCompte(@WebParam(name = "mat")
String mat, @WebParam(name = "login")
String login, @WebParam(name = "passwd")
String passwd, @WebParam(name = "type")
String type) {
    return ejbRef.AjouterCompte(mat, login, passwd, type);
}


@WebMethod(operationName = "UpdateCompte")
public boolean UpdateCompte(@WebParam(name = "mat")
String mat, @WebParam(name = "login")
String login, @WebParam(name = "passwd")
String passwd, @WebParam(name = "type")
String type) {
    return ejbRef.UpdateCompte(mat, login, passwd, type);
}
```

Figure V.13. Une Portion du code de la classe WsImplMtethodes.

- L'annotation **@WebService,** nous a permis de faire de la classe [WsImplMethodes] un web service.

- Toutes les méthodes du service web sont taguées avec l'annotation **@WebMethod** pour en faire des méthodes visibles aux clients distants. La création de cette classe donne naissance

au service web WsImplMethodes et les méthodes qu'il expose aux clients distants.

La figure suivante est une capture d'écran montrant ce module crée ainsi que la classe [WsImplMethodes] et le dossier Web service **WsImplMethodes** généré :

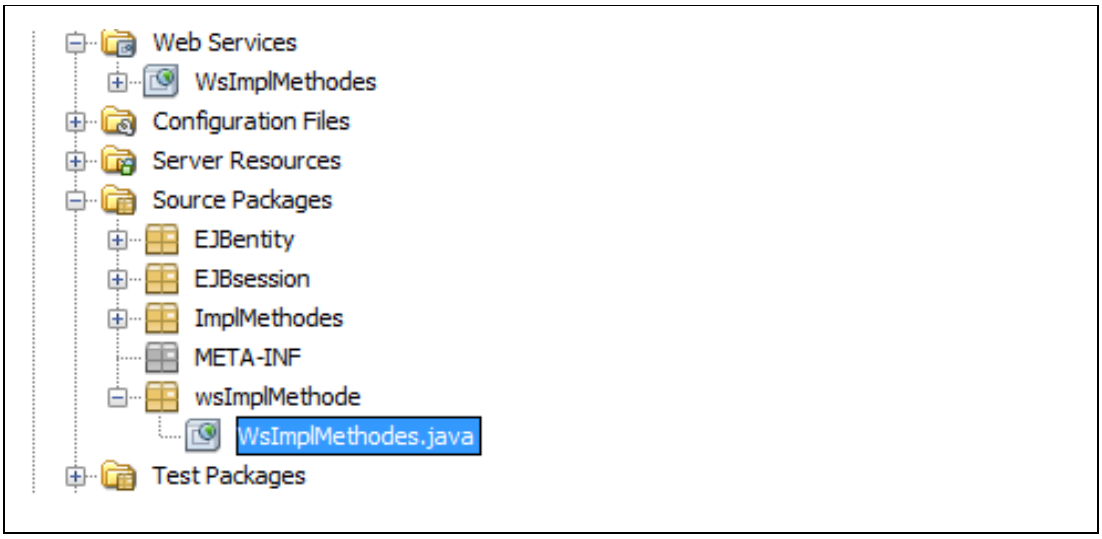


Figure V.14. Le module WsImplMethodes.

Comme on a dit déjà le package WsImplMethodes contient l'ensemble des méthodes offertes par notre service web pour les clients distants, ces méthodes seront utilisées par ces derniers pour permettre l'échange de données entre les utilisateurs finaux et notre système.

L'image suivante est une capture d'écran montrant quelques méthodes de notre web service :

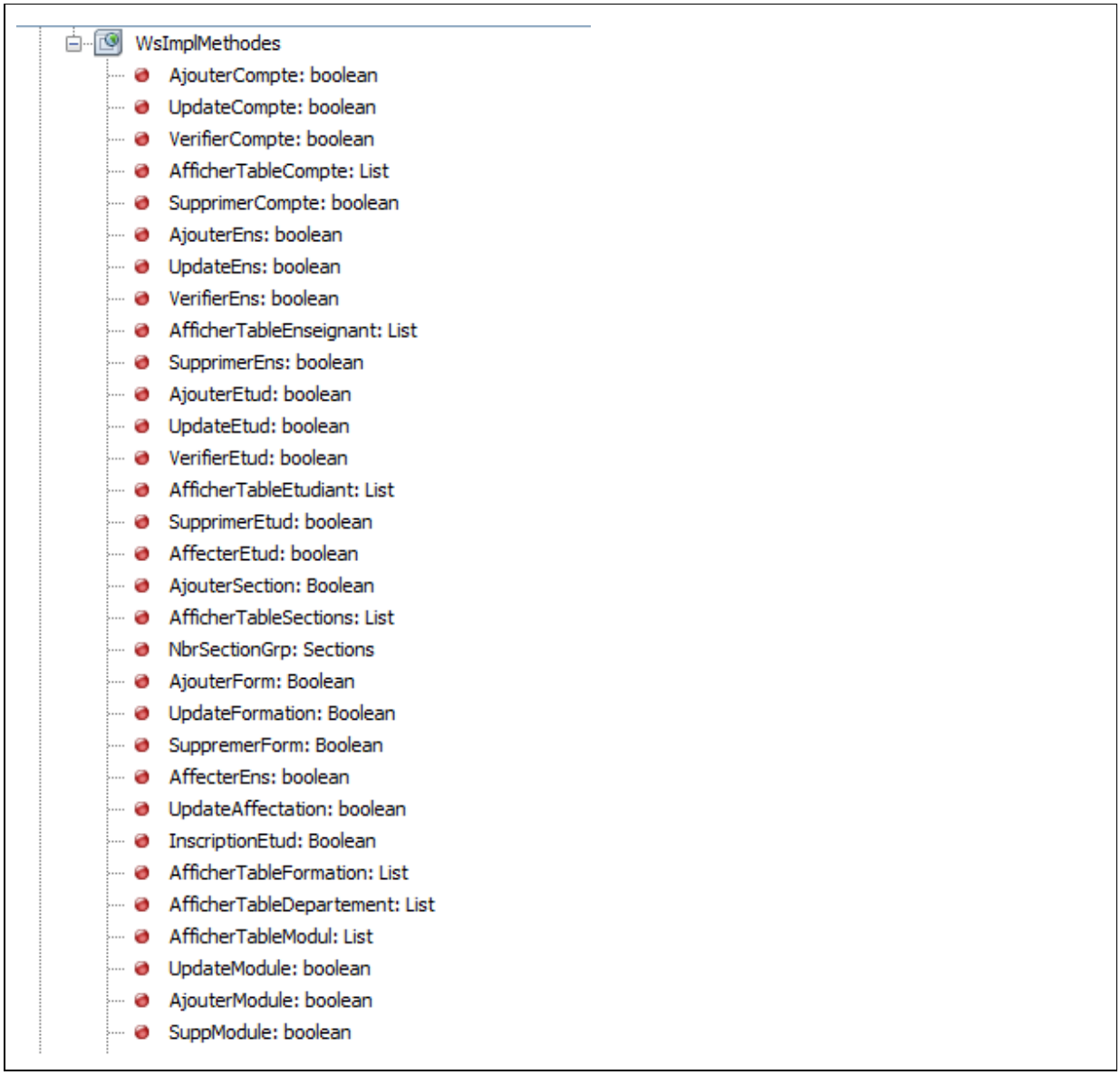


Figure V.15.Vue partielle des Web méthodes.

III .2.2 Le déploiement du web service

Une fois les deux modules précédents (**AppScolariteLMD-EJB**, **WsImplMethodes**) sont construits et rassemblées dans le projet Gestion de la scolarité, on déploie ce dernier au sein du serveur Glassfish.

Le web service [**WsImplMethodes**] apparaît dans la branche [web services] de ce serveur, ce qui nous indique que notre service web est désormais disponible sur ce serveur.

L'image suivante est une capture d'écran de la console administrateur du serveur Glassfish montrant le web service déployé, ainsi que les diverses informations qui le caractérise.

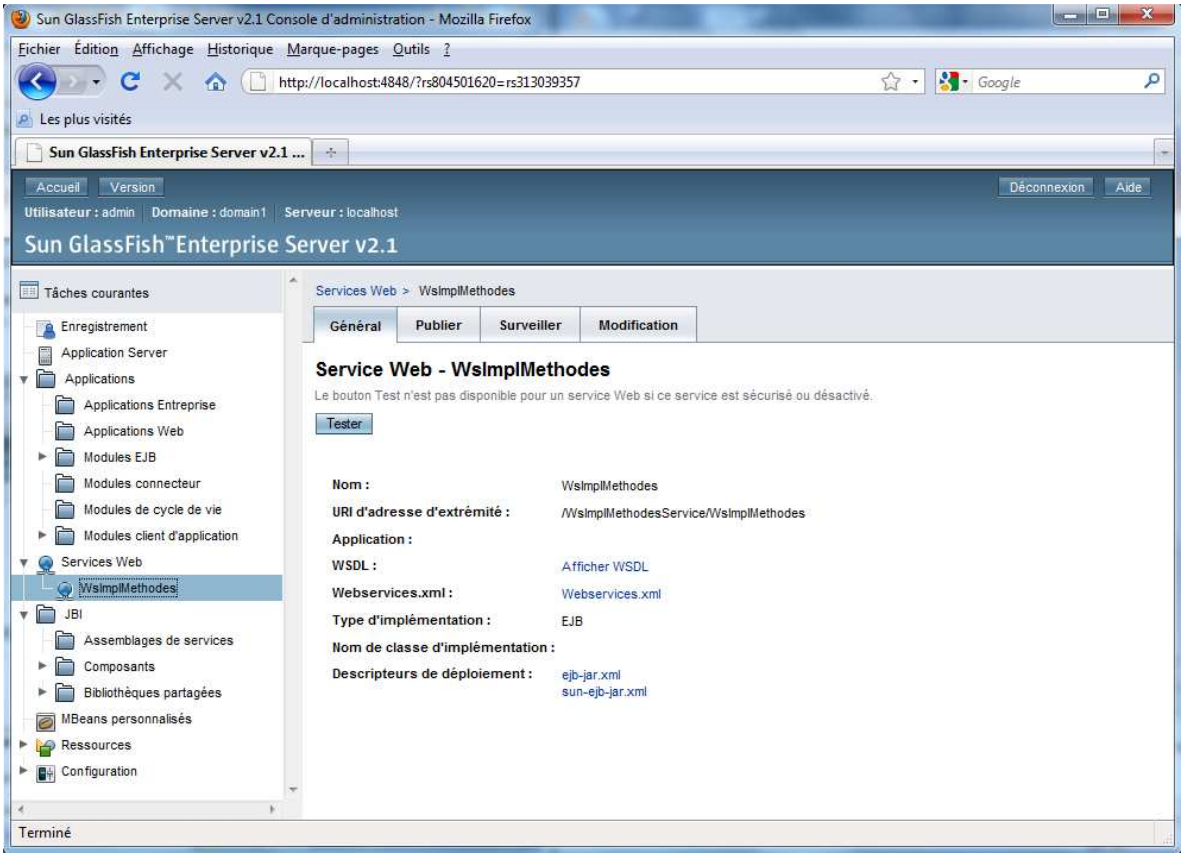


Figure V.16. Console administration du serveur Glassfish.

III .2.2.1 Tester les web méthodes

Cette console nous permet de tester notre web service en cliquant sur le bouton **tester**, la page de test s'ouvre. Toutes les méthodes (**@WebMethod**) exposées par le service web sont affichées et peuvent être testées.

La figure suivante est une capture d'écran montrant cette page de test, ainsi quelques web méthodes qu'elle expose :

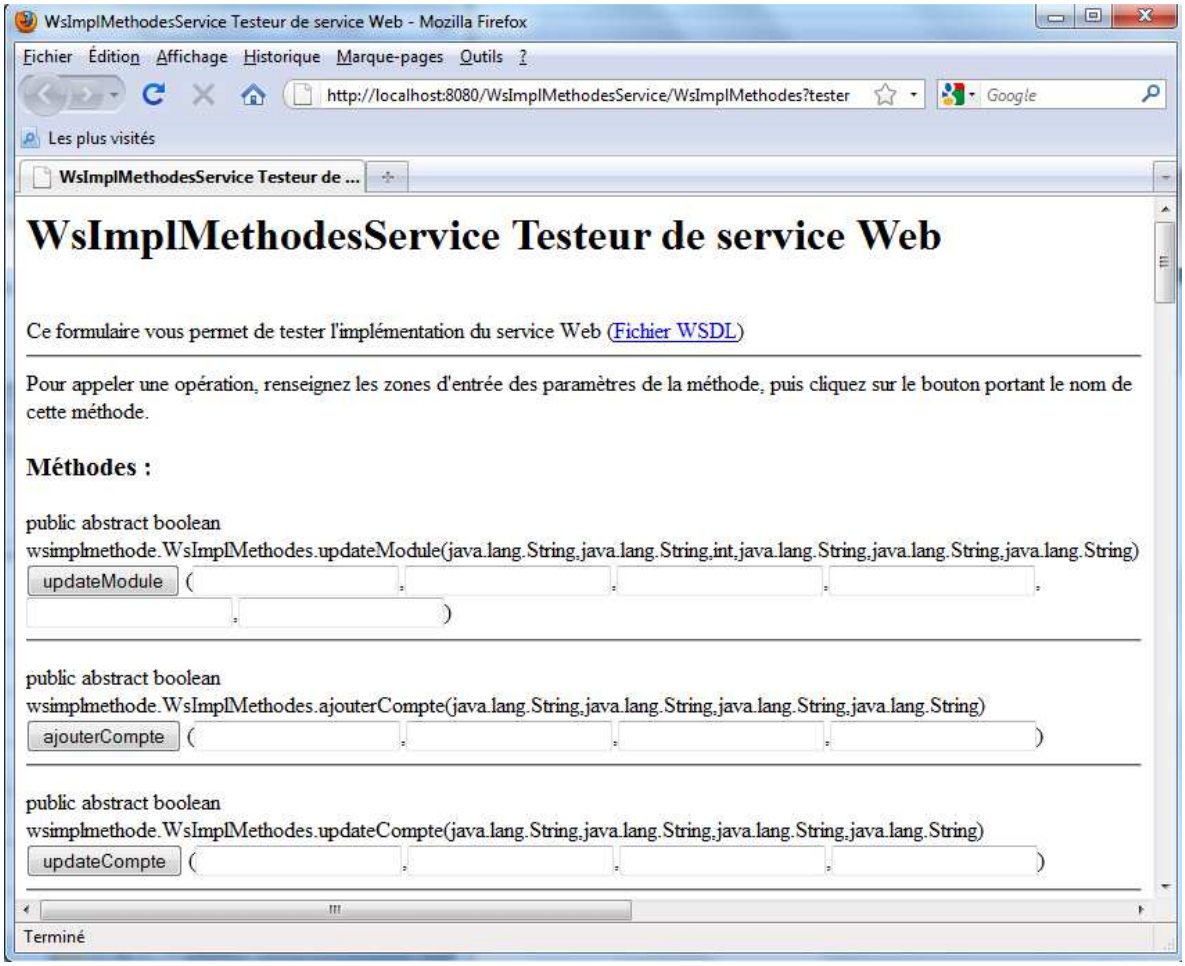


Figure V.17. Vue Partielle de la page test du web service.

Nous allons dans ce qui suit tester quelques méthodes de notre service web, pour mieux illustrer les requêtes et les réponses SOAP échangées entre notre web service les clients qui vont l'invoquer :

- Tester la méthode **AjouterCompte():**

Cette méthode récupère un patient par son identifiant, elle prend comme paramètre l'identifiant du patient comme le montre la figure suivante :

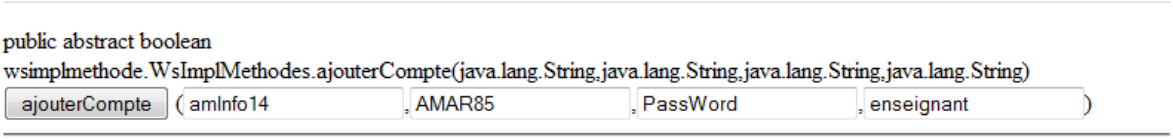


Figure V.18. Teste de la méthode AjouterCompte()

En cliquant sur le bouton **AjouterCompte()**, cette méthode sera invoquée et on aura une page qui nous indique d'un côté, la Demande SOAP transmise au serveur, comme l'illustre la figure suivante :

Demande SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:AjouterCompte xmlns:ns2="http://wsImplMethode/">
            <mat>amInfo14</mat>
            <login>AMAR85</login>
            <passwd>PassWord</passwd>
            <type>enseignant</type>
        </ns2:AjouterCompte>
    </S:Body>
</S:Envelope>
```

Figure V.19. La demande SOAP de la méthode AjoutCompte

Et de l'autre côté on aura la réponse SOAP à cette demande, comme l'illustre la figure suivante :

Réponse SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:AjouterCompteResponse xmlns:ns2="http://wsImplMethode/">
            <return>true</return>
        </ns2:AjouterCompteResponse>
    </S:Body>
</S:Envelope>
```

Figure V.20. La Réponse SOAP pour la méthode AjoutCompte.

D'après cette figure on peut voir que cette méthode a bien renvoyé la valeur « **True** » qui veut dire que le compte a été bien ajouté.

III .2.2.2 Le WSDL du web service :

Parmi les informations qui caractérise notre service web et qui sont accessibles d'après la console administrateur du serveur Glassfish, on trouve le lien **Afficher WSDL.**

Ce lien nous permet d'afficher le fichier de description du service web, comme le montre la figure suivante :

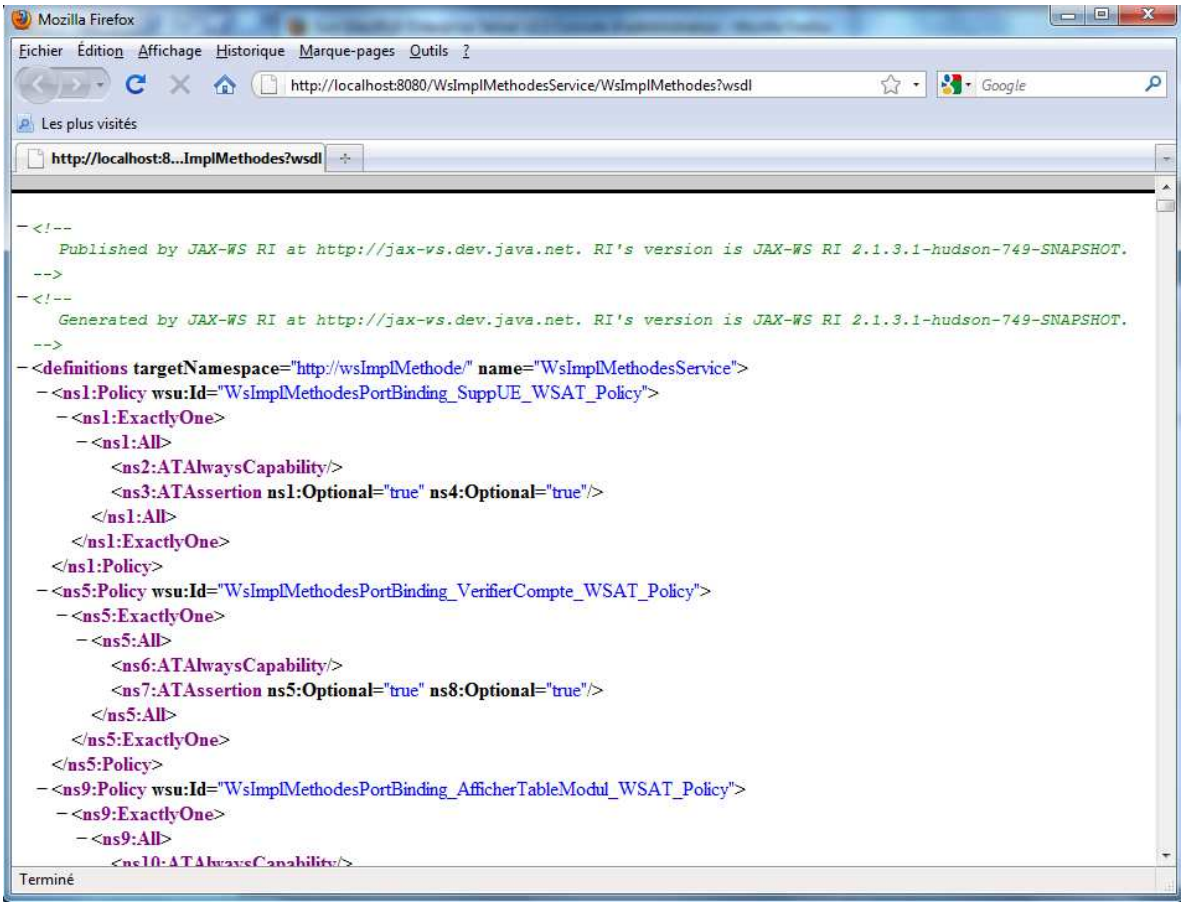


Figure V.21. Vue partielle du fichier WSDL.

L'URI (UNIFORM RESOURCE IDENTIFIER) du fichier WSDL est une information importante à connaître. Elle est nécessaire pour configurer les clients de ce web service.
On remarque que ce fichier de description est entièrement écrit en XML.

III.3. Partie Présentation

Pour illustrer l'échange de données entre le client et notre système, nous avons réalisé quatre applications client : application client pour l'agent de scolarité, application client pour l'administrateur, application client pour l'enseignant et une dernière application client pour l'étudiant. Ces applications font appel à des web méthodes que nous avons développées dans notre web service.

À travers les interfaces présentées ci-dessous, nous visons à donner une vue générale de notre application conçue.

III.3.1. Représentation des interfaces de l'application client « Agent de Scolarité »

Authentification agent de scolarité

L'interface d'authentification (figure V.22) est la première fenêtre télécharger et visualisé par l'utilisateur. Il devra taper son nom utilisateur, son mot de passe pour pouvoir accéder à l'son espace.

Figure V.22. Fenêtre d'authentification Agent de Scolarité.

Alors, si les cordonnées (Username, Password) saisies par l'utilisateur sont justes, l'interface correspondante à cet utilisateur sera téléchargée.

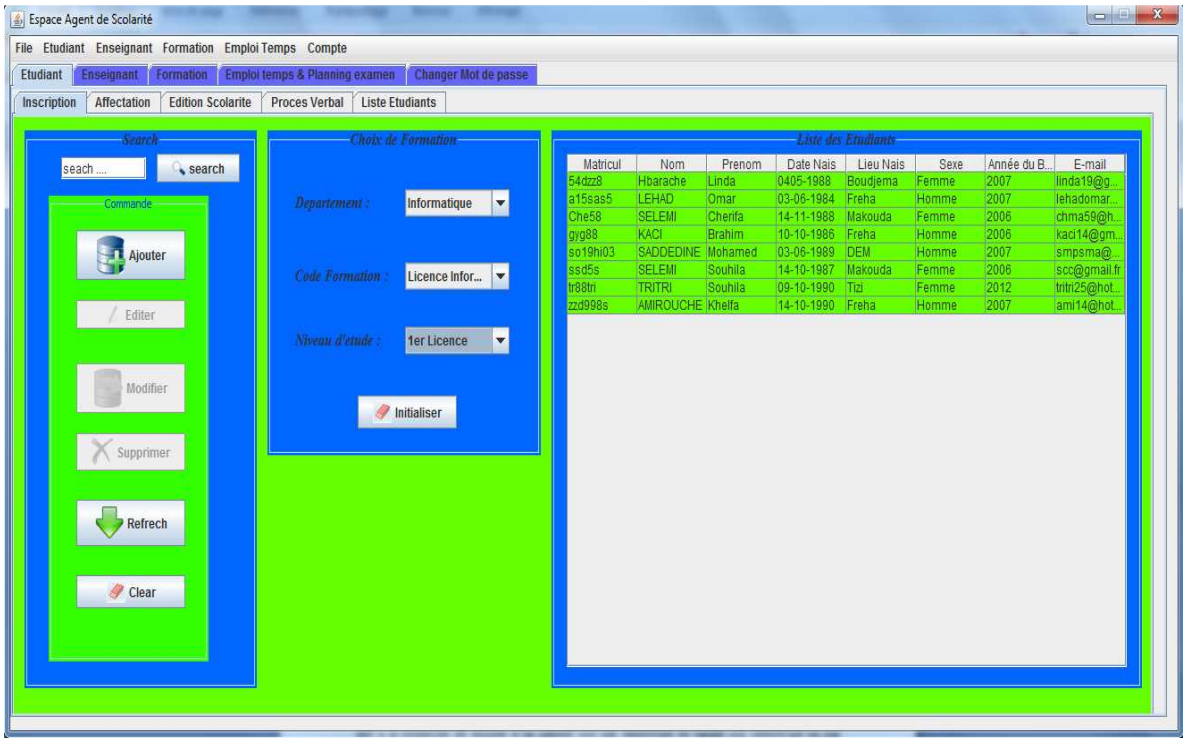


Figure V.23. Fenêtre d'accueil de l'agent de scolarité

L’espace Agent de scolarité nous montre les différentes fonctionnalités de l’agent de scolarité, et on cite quelques interfaces.

Ajout d’un Etudiant :

Pour ajouter un étudiant, l’agent de scolarité doit cliquer sur « Ajouter dans l’onglet Etudiant ->Inscription », et l’interface suivante **(figure V.24)** s’affiche. Après il remplie toutes les informations sur l’étudiant à s’inscrire.

Figure V.24. Fenêtre de l’ajout d’un étudiant.

Après le remplissage du formulaire par l’agent de scolarité, ce dernier clique sur le bouton Ajouter pour sauvegarder les informations au niveau de la base de données.

Edition de certificat de scolarité

L’édition du certificat de scolarité se fait ; après l’affichage de la liste des étudiants en choisissant un département, code formation et le niveau d’étude ; en sélectionnant un étudiant et on clique sur le bouton « Editer Scolarité ».

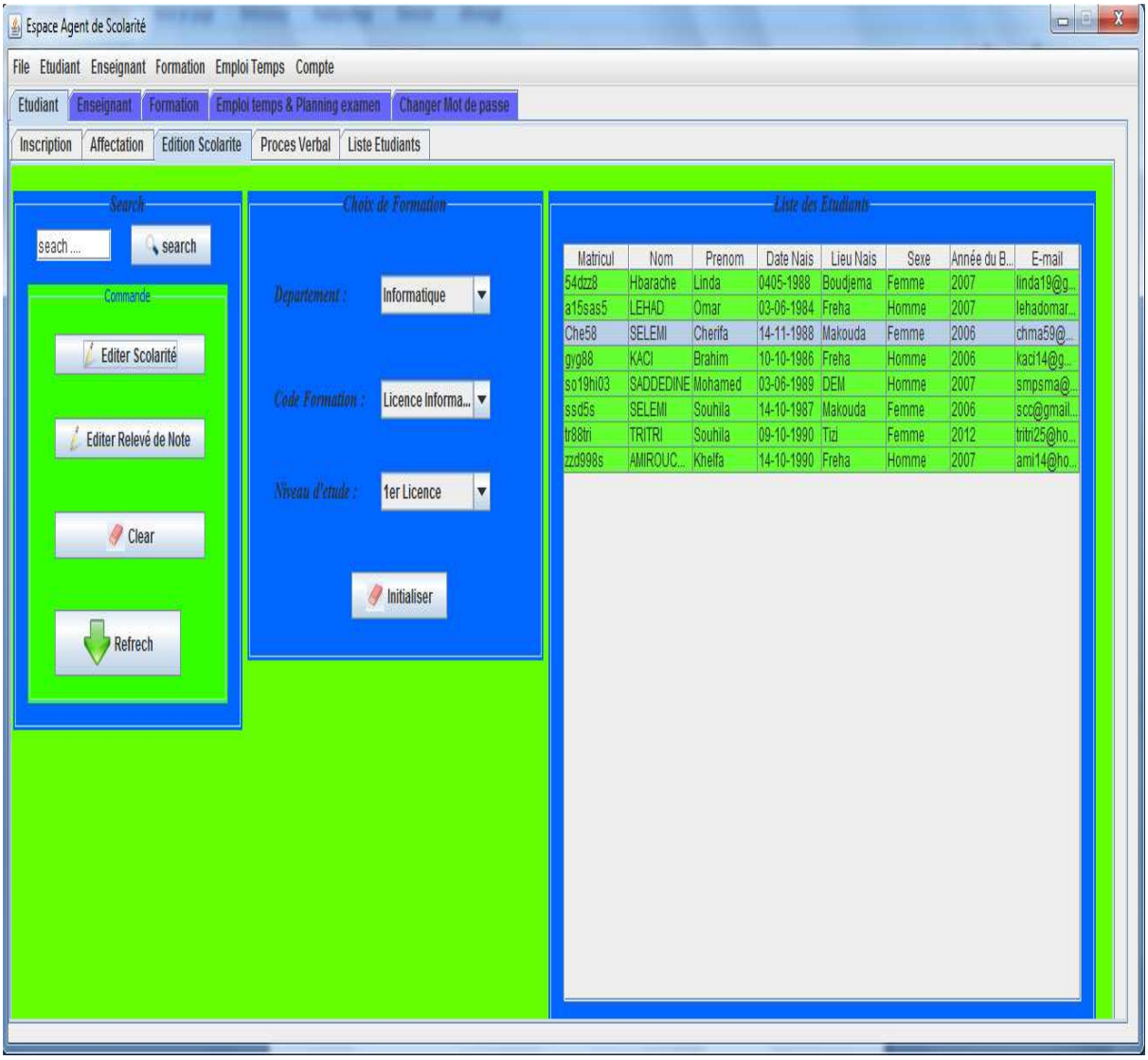


Figure V.25. Fenêtre formulaire édition de la scolarité

Après le clique sur le bouton « Editer Scolarité » une fenêtre s'ouvre, contient les informations nécessaires sur la scolarité de l'étudiant sélectionné.

Figure 4.26. Fenêtre édition de la scolarité

Ici dans cette fenêtre on a deux choix, générer la scolarité sous forme PDF, ou bien imprimer directement la page.

Edition de relevé de note

L'édition de relevé de note c'est pareil à l'édition du certificat de scolarité.

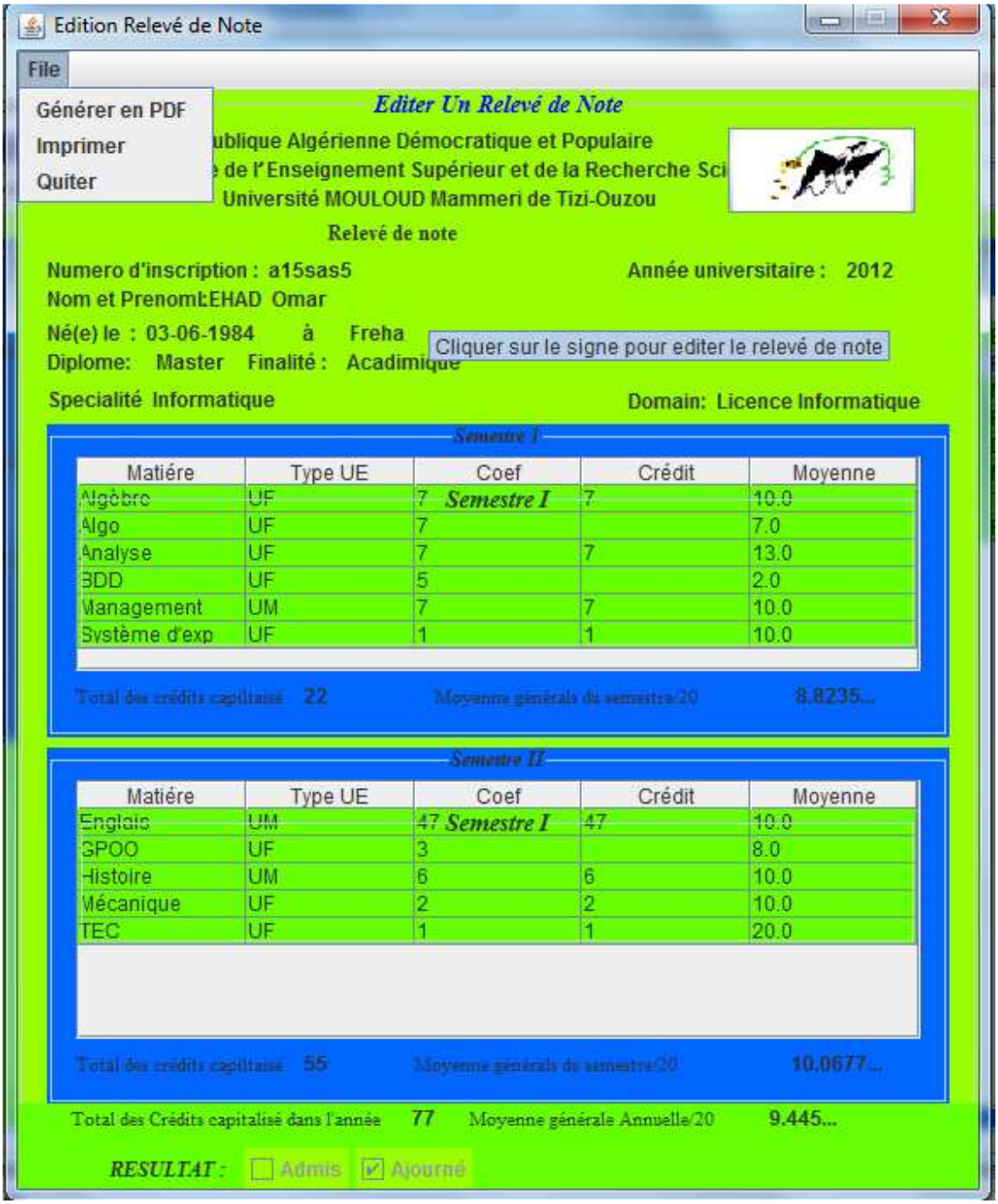


Figure 4.27.Edition de relevé de note

Ici aussi on a deux choix, générer le relevé de note sous forme PDF, ou bien imprimer directement la page.

Liste des étudiants

On affiche tout d'abord la liste des étudiants en choisissant le code la formation et le niveau d'étude.

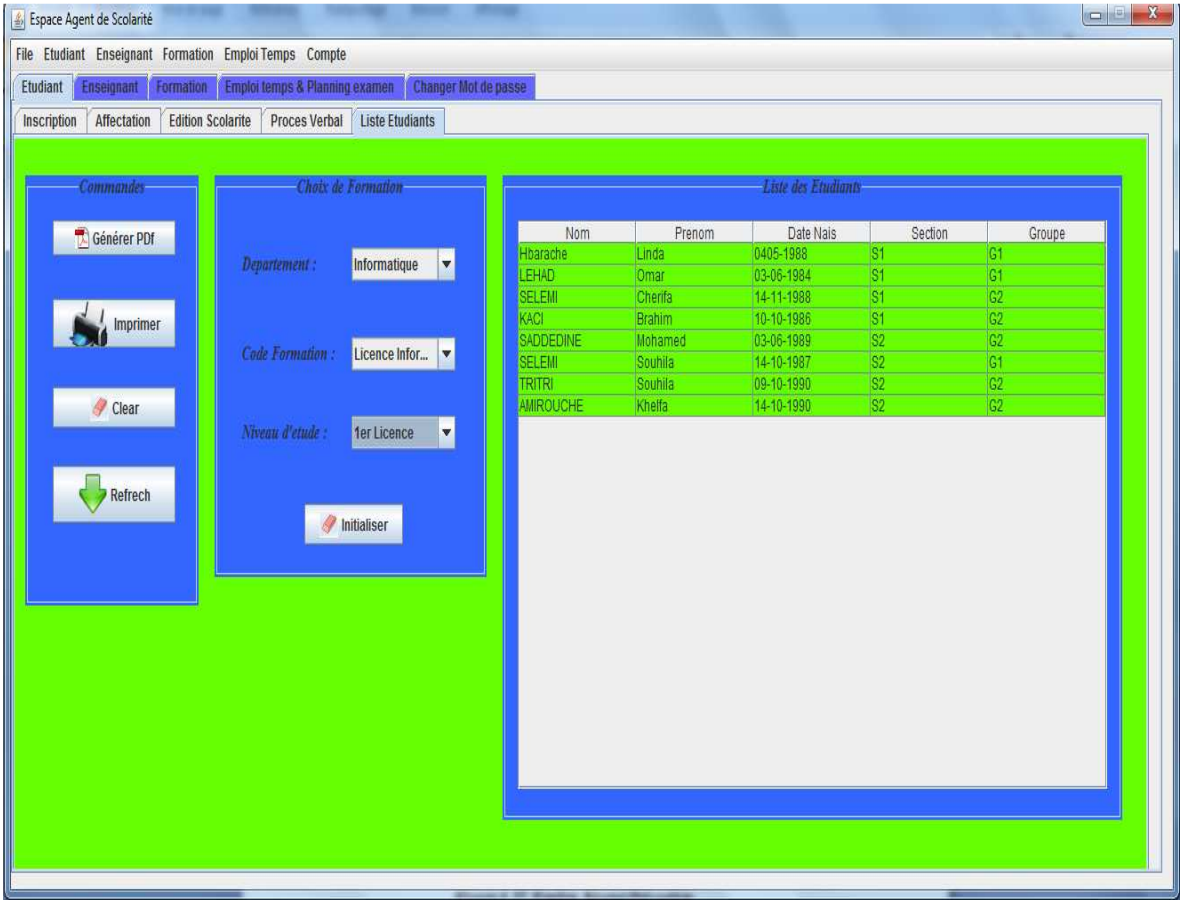
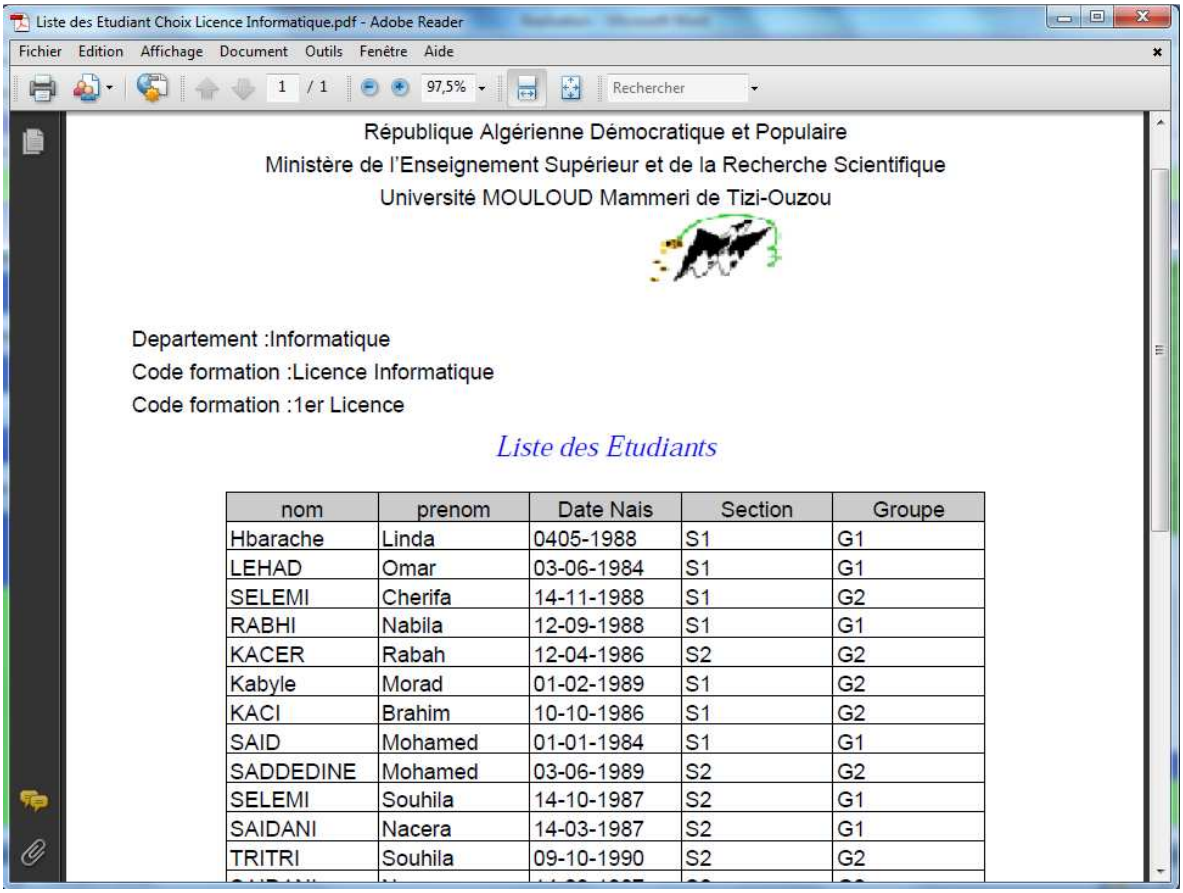


Figure V.28. Fenêtre Edition de la liste des étudiants.

Après on clique sur le bouton « Générer PDF » ou bien sur le bouton « imprimer » pour imprimer la liste des étudiants.

La figure suivante c'est la liste des étudiants sous forme PDF d'une Formation donnée et un niveau choisi.



nom	prenom	Date Nais	Section	Groupe
Hbarache	Linda	0405-1988	S1	G1
LEHAD	Omar	03-06-1984	S1	G1
SELEMI	Cherifa	14-11-1988	S1	G2
RABHI	Nabila	12-09-1988	S1	G1
KACER	Rabah	12-04-1986	S2	G2
Kabyle	Morad	01-02-1989	S1	G2
KACI	Brahim	10-10-1986	S1	G2
SAID	Mohamed	01-01-1984	S1	G1
SADDEDINE	Mohamed	03-06-1989	S2	G2
SELEMI	Souhila	14-10-1987	S2	G1
SAIDANI	Nacera	14-03-1987	S2	G1
TRITRI	Souhila	09-10-1990	S2	G2

Figure V.39. Fenêtre Liste des étudiants sous forme PDF

Affectation des enseignants

L'affectation des enseignants c'est un formulaire à remplir, après la sélection de l'enseignant à affecté.

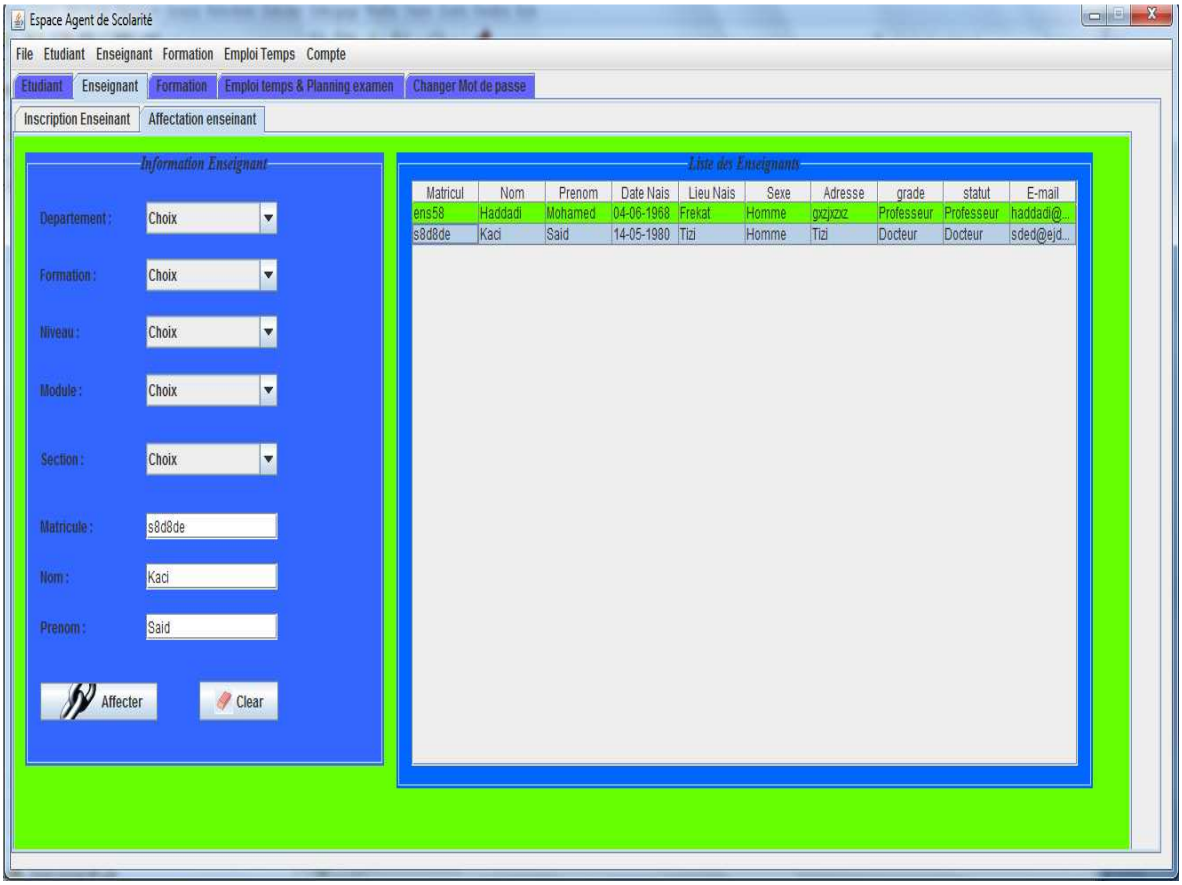


Figure V.30. Fenêtre Affectation des enseignants

Les informations de l'enseignant sélectionné seront copies automatiquement au formulaire « Affection enseignant ». L'agent de scolarité clique ensuite sur le bouton « Affecter » pour sauvegarder les informations dans la BDD.

III.3.2. Représentation des interfaces de l'application client « Administrateur »

Ajouter un compte enseignant

L'administrateur accède à son espace de travail, et c'est lui le responsable sur création, suppression et modification des comptes. Donc l'administrateur sélectionne l'enseignant à qui il affecte un login et un mot de passe. A la fin, il valide l'affection en cliquant sur le bouton ajouter.

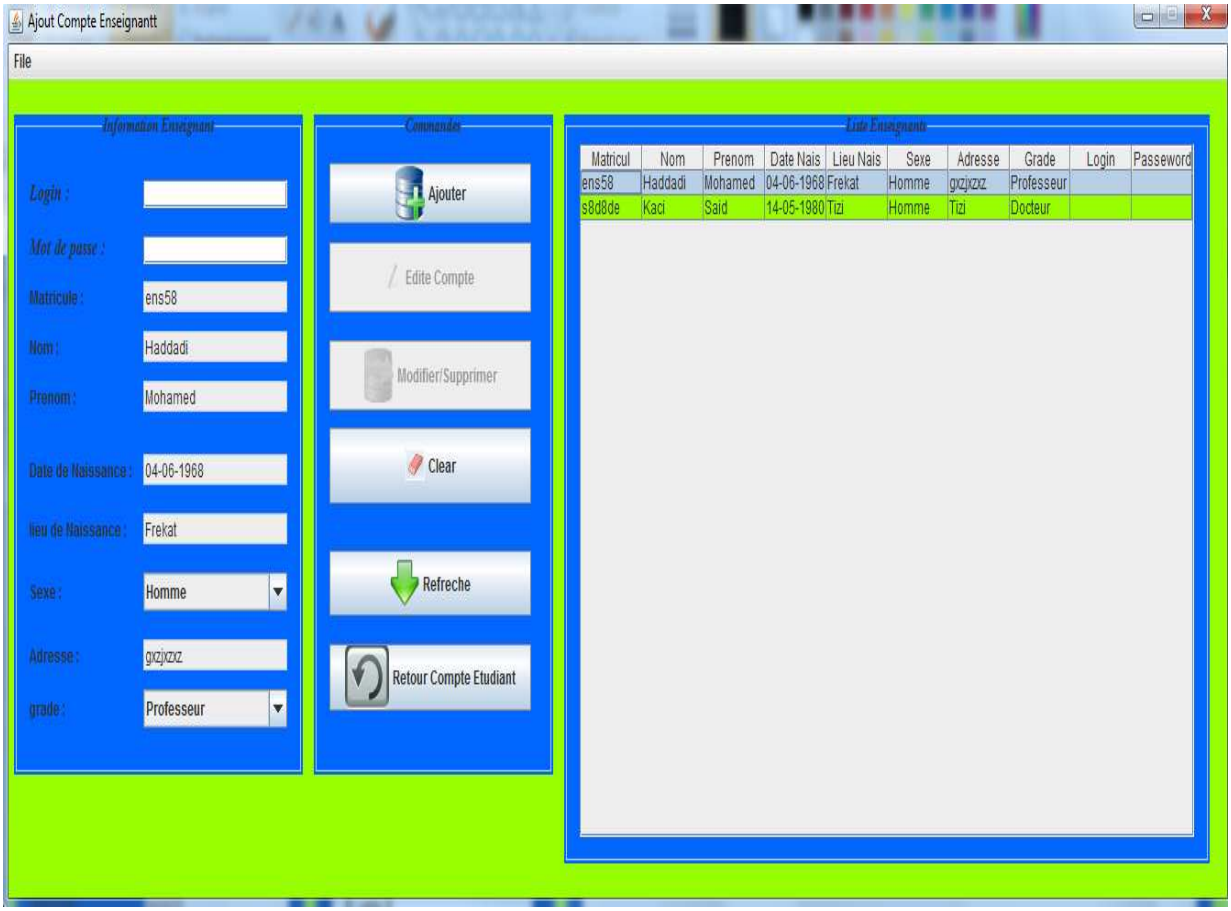


Figure V.31. Fenêtre Ajouter compte enseignant

Modification/ Suppression compte enseignant

L’administrateur sélectionne le compte à modifier ou à supprimer, ensuite il clique sur le bouton « Modifier/Supprimer », une nouvelle fenêtre s’ouvre avec les informations sur le compte. L’administrateur a le choix entre la modification ou bien la suppression du compte.

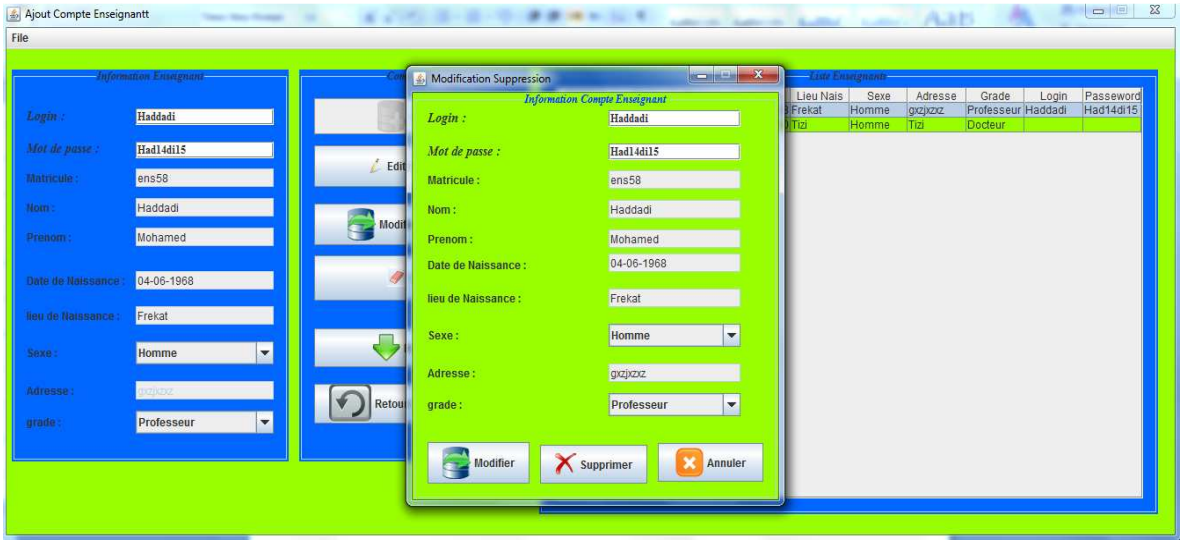


Figure V.32. Fenêtre Modification/Suppression compte enseignant

Ajouter un compte Etudiant

Ici c’est pareil à l’ajout de compte enseignant, la différence c’est qu’ici, il sélectionne la formation et le niveau d’étude.

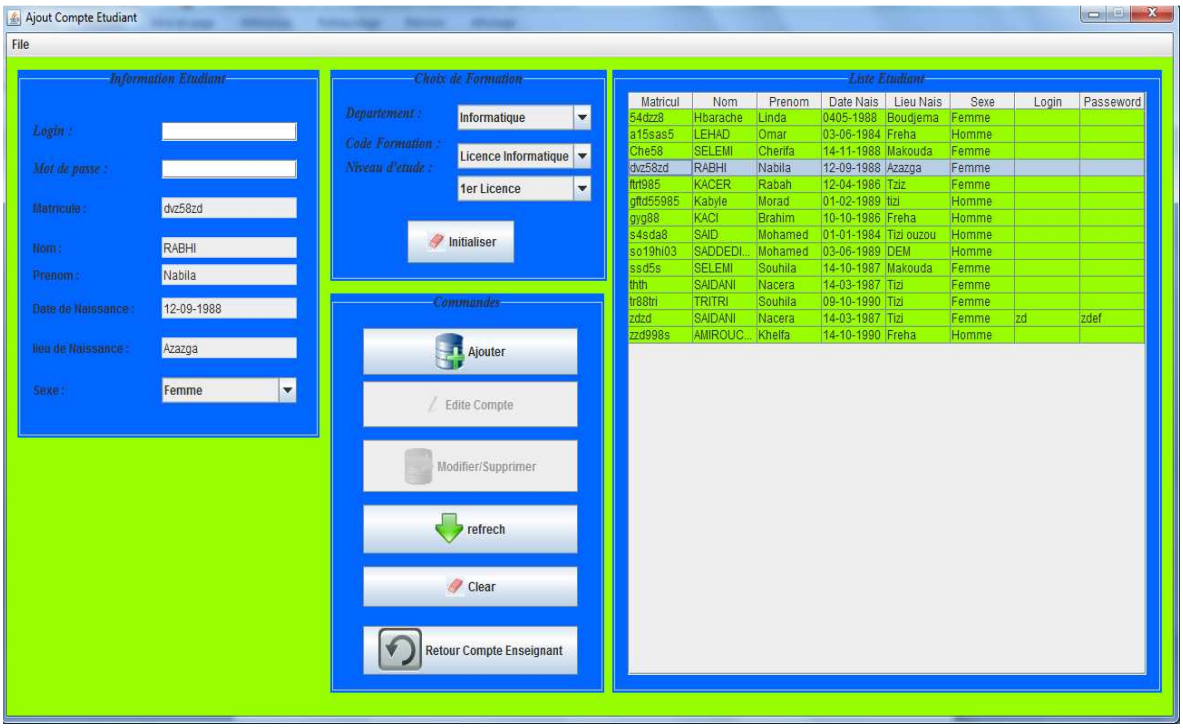


Figure V.33. Fenêtre Ajout compte étudiant.

III.3.3. Représentation des interfaces de l’application client « Enseignant »

Authentification Enseignant

L'interface d'authentification (figure V.34) est la première fenêtre télécharger et visualisé par l'enseignant. Il devra taper son nom utilisateur, son mot de passe pour pouvoir accéder à l'son espace.

Figure V.34 Fenêtre Authentification enseignant

Gestion des notes des étudiants

L'enseignant accède à son espace, ensuite il clique sur l'onglet « gestion des notes », il remplie le formulaire pour afficher la liste des étudiants afin qu'il puisse saisir les notes.

Figure V.35 Fenêtre Gestion des notes

Conclusion :

On a devisé ce chapitre en deux parties, primo c’est la façon dont on a implémenté l’application en commençant par la présentation de la base de données, puis la création de module EJB avec tous ces composants, ensuite on a passé à la création du module web avec sont composant web service, et on a fini cette partie par la création des modules clients. Secundo c’est la partie présentation de quelques interfaces de l’application.

Conclusion Générale

Conclusion Générale

Le but fixé à notre travail était de réaliser et de développer une plateforme assez complète dans son architecture, simple dans sa composition, abordable dans son utilisation, intégrant le plus de fonctionnalités, qui permettent d'assurer une gestion complet de la scolarité et offrant aux utilisateurs une panoplie d'outils, qui permettent à chacun d'accomplir son rôle par notre système.

Vu les avantages qu'offrent les deux technologies (J2EE, Web service) qui deviennent prometteuses, comme en témoignent l'interopérabilité et la réutilisation de ces fonctionnalités, donc il est indispensable pour une entreprise de faire appel à ces deux nouvelles technologies pour laquelle elles assurent la fiabilité et la sécurité de l'immense flux d'informations entre les différents acteurs de l'entreprise.

Ce projet consiste en l'implémentation d une application J2EE Web service pour rendre les tâches de l'agent de scolarité dynamiques en assurant la fiabilité et la sécurité des informations. Nous avons d'abord étudié profondément les composants de l'architecture J2EE et les concepts fondamentaux du Web service.

Ce projet nous a permis d'approfondir nos connaissances en:

- Gestion d'applications entreprises.
- Interface JDBC.
- Modélisation orientée objet…
- Maitrise de l'architecture J2EE avec tous ses composants.
- l'architecture Web service et ses protocoles de communication.
-Déploiement des modules EJB et les web services.
-programmations JAVA pour le web (EJB, Web Service, Interface graphique…).

Enfin nous espérons que notre travail sera de grands intérêts pour le département informatique de notre faculté de génie électronique et d'informatique (FGEI) et un guide efficace pour les nouvelles promotions.

Annexe A

UML Unified Modeling Language.

Annexe A
UML Unified Modeling Language.

A.1 A quoi sert UML ?

UML permet de définir et de visualiser un modèle, à l'aide de diagrammes. Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle; c'est une perspective du modèle, pas "le modèle". Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis).
Un type de diagramme UML véhicule une sémantique précise (un type de diagramme offre toujours la même vue d'un système). Combines, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.
Par extension et abus de langage, un diagramme UML est aussi un modèle (un diagramme modélise un aspect du modèle global). UML est un langage graphique et repose sur neuf types de diagrammes. Chacun de ces diagrammes utilise le même principe : les concepts sont représentés par des symboles, et les relations entre les concepts sont représentées par des lignes qui relient les symboles. Dans ce qui suit nous allons présenter les diagrammes les plus utilisés dans la modélisation et conception d'un système d'information distribuée sur l'Internet.

A.2 Les diagrammes des cas d'utilisation

Les cas d'utilisation décrivent le comportement du système du point de vue de l'utilisateur. Ils permettent de définir les limites du système et les relations entre le système et son environnement. Un cas d'utilisation est une manière spécifique d'utiliser le système. C'est l'image d'une fonctionnalité déclenchée en réponse à la stimulation d'un acteur externe. Ils permettent de centrer la construction du système sur les besoins des utilisateurs. II existe deux concepts fondamentaux dans les cas d'utilisation :

• **Les acteurs qui utilisent le système :**

- Acteurs primaires qui sont la raison de l'existence de ce système.
- Acteurs secondaires qui ont des rôles d'administration et qui lui fournissent toutes les informations nécessaires à son bon fonctionnement.

• **Les use cases qui représentent l'utilisation du système par les acteurs.**

Les diagrammes font intervenir les acteurs, les autres systèmes et les cas d'utilisation eux mêmes.

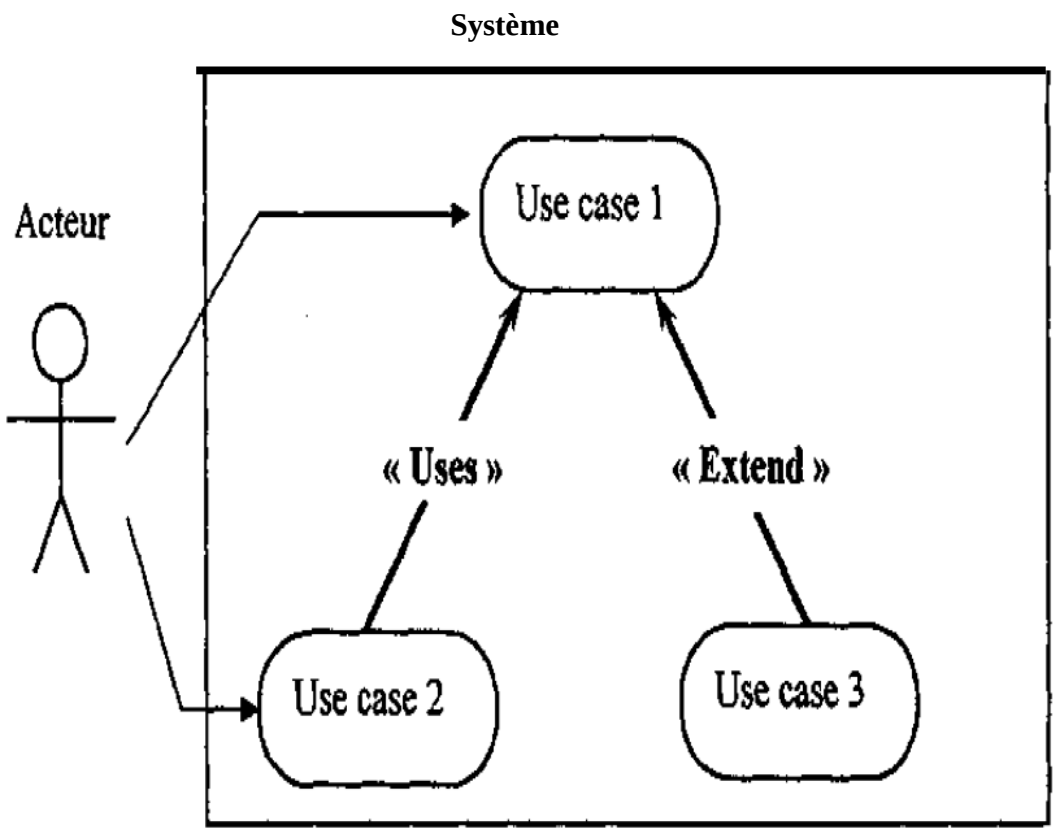


Figure A : Représentation graphique d'un cas d'utilisation [Pamu04].

II peut exister des relations entre les cas d'utilisation que l'on représente au moyen de stéréotypes :

- Le cas d'utilisation peut utiliser un autre cas d'utilisation et on le dénote par une relation de type « uses»,
- Le cas d'utilisation peut étendre un autre cas d'utilisation et on le dénote par une relation de type « Extend ».

A.3 Les diagrammes de classes

Ces diagrammes décrivent l'architecture du système. On y représente les classes et les relations entre classes, qu'elles soient d'héritage, d'agrégation,…etc.

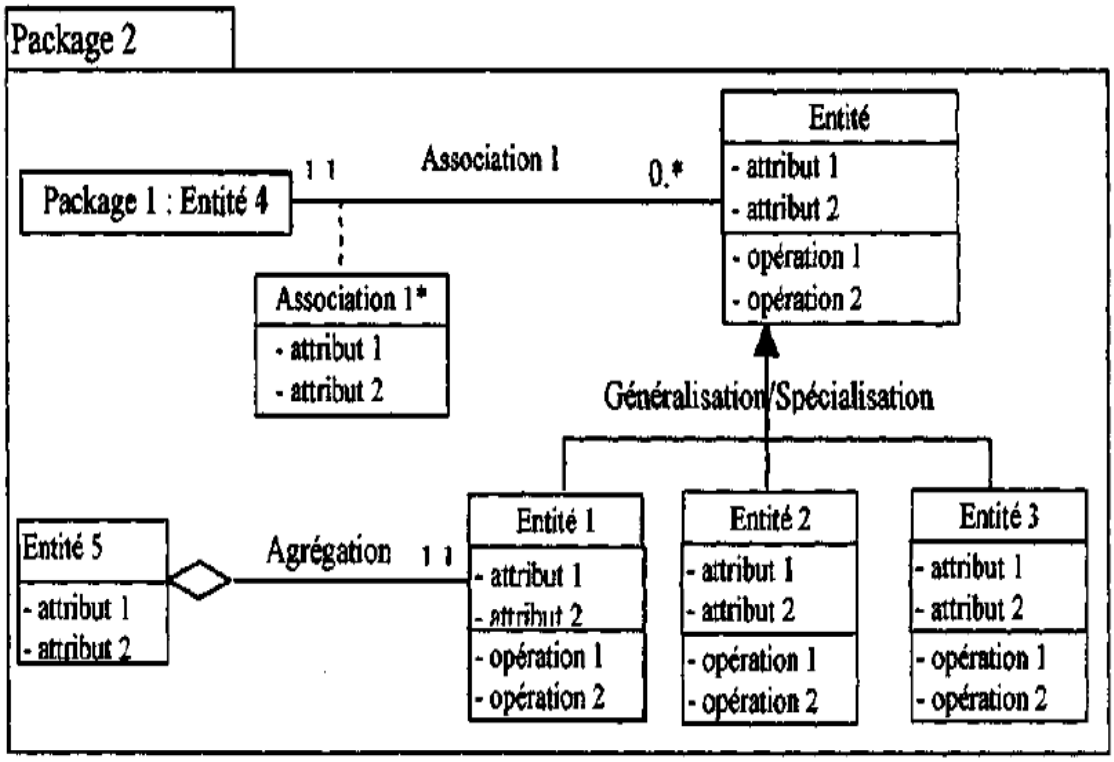


Figure B : Représentation graphique d'un diagramme de classes.

A.4 Les diagrammes d'états

C'est un automate d'états finis, compose de transitions, d'événement et d'activités. II représente la vue dynamique du système par la détermination du comportement des objets d'une classe en termes d'états et de transitions d'états. Ce diagramme est représenté par un graphe constitue de nœuds décrivant des états ainsi que des flèches représentant des transitions portant des paramètres et des noms d'événements.

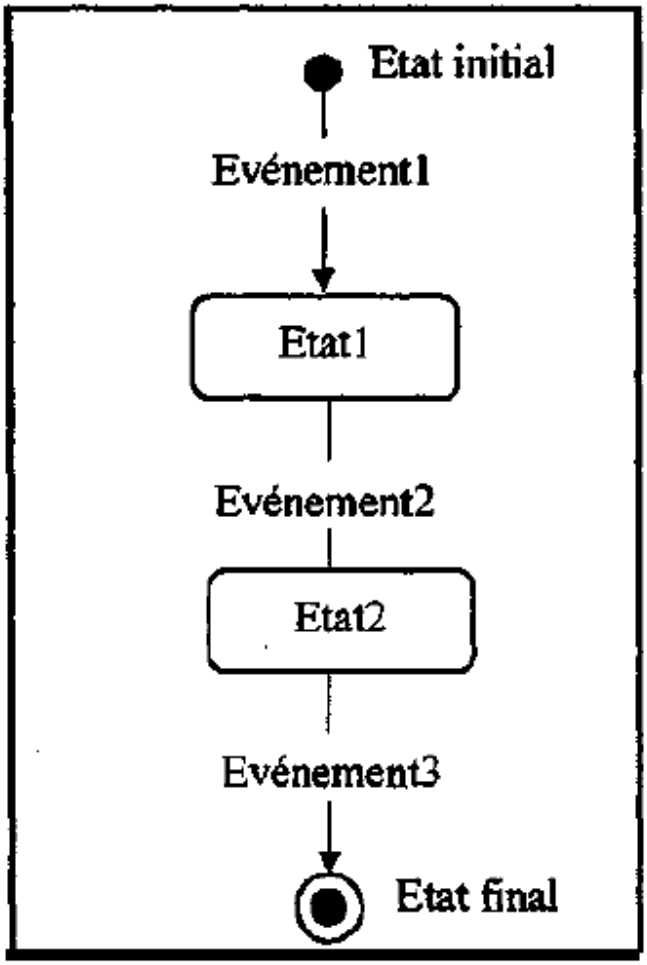


Figure C : Représentation graphique d'un diagramme d'états.

Le diagramme d'états est propre à une classe donnée, il doit être construit pour les objets de l'application ayant un comportement non trivial. Un état est une situation. Une transition est une relation entre deux états signifiant qu'un passage de l'un a l'autre est possible. En d'autres termes, c'est une relation orientée entre deux états, a laquelle est attaché un événement et qui indique qu'un objet dans le premier état passera dans le second si certaines conditions sont remplies.

A.5 Les diagrammes de séquences

Les diagrammes de séquences illustrent les entités ainsi que les messages qu'elles s'échangent, dans une procédure bien précise. Donc, il y aura autant de scénarios que de procédures. Ces diagrammes seront classés par procédures et par acteurs.

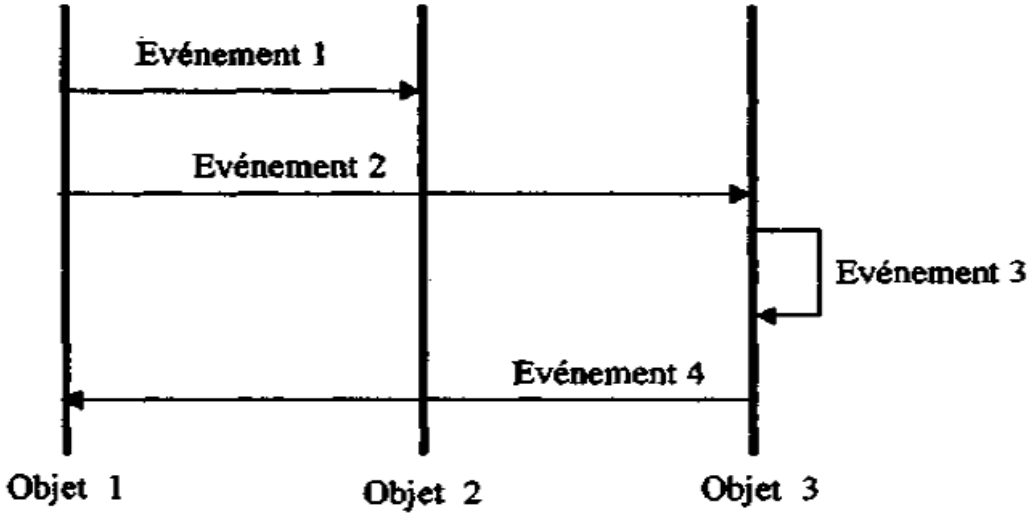


Figure D : Représentation graphique d'un diagramme de séquences.

Annexe B

Présentation de XML

Annexe B

Présentation de XML

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).

Contrairement à HTML, qui est à considérer comme un langage défini et figé (avec un nombre de balises limité), XML peut être considéré comme un métalangage permettant de définir d'autres langages, c'est-à-dire définir de nouvelles balises permettant de décrire la présentation d'un texte.

La force de XML réside dans sa capacité à pouvoir décrire n'importe quel domaine de données grâce à son extensibilité. Il va permettre de structurer, poser le vocabulaire et la syntaxe des données qu'il va contenir.

En réalité les balises XML décrivent le contenu plutôt que la présentation (contrairement À HTML). Ainsi,**XML permet de séparer le contenu de la présentation** .. ce qui permet par exemple d'afficher un même document sur des applications ou des périphériques différents sans pour autant nécessiter de créer autant de versions du document que l'on nécessite de représentations !
XML a été mis au point par le XML Working Group sous l'égide du (W3C) dès 1996. Depuis le 10 février 1998, les spécifications *XML 1.0* ont été reconnues comme recommandations par le W3C, ce qui en fait un langage reconnu.

XML est un sous ensemble de SGML (*Standard Generalized Markup Language*), défini par le standard ISO8879 en 1986, utilisé dans le milieu de la Gestion Electronique Documentaire (GED). XML reprend la majeure partie des fonctionnalités de SGML, il s'agit donc d'une simplification de SGML afin de le rendre utilisable sur le web !

1. La structure de base du XML :

Le XML est un langage de balise [Markup Langage]. Mais au contraire du HTML ou les balise sont définies, vous devez inventer vos balises. Rapplez-vous.le XML est extensible.il faut donc écrire soi-même le nom des balises utilisées.

Il y a quelque règle pour la composition des noms (mais elle ne déroule pas les habitudes de JavaScript) :

- Les noms peuvent contenir des lettres, des chiffres ou d'autre caractère.
- Les noms ne peuvent débuter par un nombre ou un signe de ponctuation.
- Les noms ne peuvent commencer par les lettre XML (ou XML…).
- Les noms ne peuvent contenir des espaces.
- La longueur des noms est libre mais on conseille de rester raisonnable.
- On évitera certains signe qui pourraient selon les logiciels, prêter confusion comme ‘’-‘’, ‘’;’’,’’,’’,’’.’’,’’<’’,’’>’’, etc....
- Les caractères spéciaux pour nous francophone comme é, à, è,….sont à priori permis mais pourraient être mal interprètes par certaine programmes.

  On profitera de cette liberté dans les noms pour les rendre le plus descriptif possible Comme par exemple :<gras_et_italiqur>.
- Les balises sont sensibles au majuscules et minuscules [case sensitive.].

  Ainsi, la balise <message>est déférente de la balise <Message>.la balise douverture et la balise de fermeture doivent donc être identiques. Ainsi par exemple : <Message>……..< /message>est incorrectes <message>……..<message/>est correcte.

  Une tendance se dégage pour n'écrire les balises qu'en minuscules, limitant ainsi les erreurs possibles.
- Toute balise ouverte doit impérativement être fermée.

  Fini les écritures bâclées du html ou l'on dans certaine cas omettre la balise de fin comme pour le paragraphe <p>ou élément liste <li>.

  Ainsi en HTML, ce qui suit est affiché correctement :

```
<p>
  <UL>
  <Li>point 1
   <Li> point 2
```

  Le XML est beaucoup plus strict. On devrait avoir :

```
<p>
```

<UL>
<Li>point 1 </Li>
<Li> point 2 </Li>
<p>

Les éventuelles balises uniques ou appeler aussi balises vides' comme
, <méta>ou <img> en HTML, doivent également comporter un signe de fermeture soit balise /.ainsi une balise <méta/>est correcte en HTML.

- Les balises doivent être correctement imbriquées :

Le XML étant très préoccupé par la structure des données, des balises mal imbriquées sont des fautes graves de sens.

Ainsi l'écriture suivante est incorrecte car les balise ne sont pas bien imbriquées :

<parent><enfant>Loïc</parent></enfant>.

L'écriture correcte avec une bonne imbrication des éléments est :

<parent><enfant>Loïc</enfant></parent>.

- Tout document XML doit comporter une racine.

En fait, la première paire de balise d'un document XML sera considéré comme la balise de racine [root].

Par exemple :

<racine>
…….suite du document XML…..
</racine>.

Si en ose faire un lien avec le HTML, votre élément racine était <body>……..</body>. Tout les autre éléments seront imbriques entre ces balises de racine.

Par exemple :

<Parents>
<Enfants>
<Petits-enfants>……. </Petits-enfants>
</Enfants>
</Parents>

- Les valeurs des attributs doivent toujours être mises entre des guillemets.

Le XML peut avoir (comme le HTML) des attributs avec des valeurs. En XML, les valeurs des attributs doivent obligatoirement être des guillemets, au contraire du HTML ou leur absence n'a plus beaucoup d'importance.

Ainsi écriture suivante est incorrecte car il manque les guillemets.

<date anniversaire=071185>.

La bonne écriture est :

<date anniversaire= ‘’071185’’>.

2. Structure d’un document XML :

En réalité un document XML est structuré en 3 parties :

La première partie, appelée *prologue* permet d'indiquer la version de la norme XML utilisée pour créer le document (cette indication est obligatoire) ainsi que le jeu de caractères (en anglais *encoding*) utilisé dans le document (attribut facultatif, ici on spécifie qu'il s'agit du jeu ISO-8859-1, jeu LATIN, pour permettre de prendre en compte les accents français). Ainsi le prologue est une ligne du type

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

Le prologue se poursuit avec des informations facultatives sur des instructions de traitement à destination d'applications particulières. Leur syntaxe est la suivante :

<?instruction de traitement?>

Le second élément est une déclaration de type de document (à l'aide d'un fichier annexe appelé DTD -*Document Type Definition*)

Et enfin **la dernière** composante d'un fichier XML est l'arbre des éléments (comme celui ci-dessus).

3. La syntaxe des éléments en XML :

L'arbre des éléments, c'est-à-dire le véritable contenu du document XML, est constitué d'une hiérarchie de balises comportant éventuellement des attributs.

a.les élément :

Toute donnée est ainsi encapsulée entre une balise ouvrante <*balise*> et une balise fermante <*/balise*>(Sachant qu'une donnée peut éventuellement être un ensemble d'éléments XML). Ainsi un élément vide est uniquement constitué d'une balise spécifique dont la syntaxe est la suivante : <*balise/*>.

D'autre part, il est interdit en XML de faire chevaucher des balises, c'est-à-dire d'avoir une succession de balises du type :

<balise1>

<balise2>

</balise1>

</balise2>

D'autre part, il est possible entre les balises (donc pas à l'intérieur d'une balise) d'ajouter :

- ✓ des espaces.
- ✓ des tabulations.
- ✓ des retours chariots.

Cela est très utile pour définir une indentation des balises (ce qui est possible puisqu'elles ne se chevauchent pas).

```
<annuaire>
  <personne class = "etudiant">
        <nom>Pillou</nom>
        <prenom>Jean-Francois</prenom>
        <telephone>555-123456</telephone>
        <email>webmaster@commentcamarche.net</email>
  </personne>
</annuaire>
```

b.les attributs :

Un attribut est une paire clé valeur écrit sous la forme *Cle="Valeur"*, ainsi une balise affectée d'un attribut aura la syntaxe suivante :

<balise cle="valeur">

Il n’y a pas vraiment de règle quant au choix entre un document utilisant uniquement des éléments sans attribut et un autre utilisant des éléments avec attributs.

Exemple1 :

```
<Liste>
    <personne>
        <prénom> anne </prénom>
        <nom> martin </nom>
        <adresse>
            <rue> rue de mandur </rue>
```

```
                <ville>rouen</ville>
            </adresse>
        </perssone>
        <personne>
            <prénom> gaelle</prénom>
            <nom> martin </nom>
            <adresse>
                <rue> rue de mandur </rue>
                <ville>rouen</ville>
            </adresse>
        </perssone>
    </liste>
```

Exemple2 :

```
    <liste>
        <personne prénom=' Anne' nom=' martin' >
            <adresse  rue= 'rue de mandur'  ville='rouen'  />
        </perssone>
        <personne prénom=' gaelle' nom=' martin' >
            <adresse  rue= 'rue de mandur'  ville='rouen'  />
        </perssone>

    </liste>
```

4. Les avantages de XML :

Voici les principaux atouts de XML :

- La lisibilité : aucune connaissance ne doit théoriquement être nécessaire pour comprendre un contenu d'un document XML
- Auto descriptif et extensible
- Une structure arborescente : permettant de modéliser la majorité des problèmes informatiques
- Universalité et portabilité : les différents jeux de caractères sont pris en compte

- ❖ Déployable : il peut être facilement distribué par n'importe quels protocoles à même de transporter du texte, comme HTTP
- ❖ Intégrabilité : un document XML est utilisable par toute application pourvue d'un parser (c'est-à-dire un logiciel permettant d'analyser un code XML)
- ❖ Extensibilité : un document XML doit pouvoir être utilisable dans tous les domaines d'applications.

Ainsi, XML est particulièrement adapté à l'échange de données et de documents.

L'intérêt de disposer d'un format commun d'échange d'information dépend du contexte professionnel dans lequel les utilisateurs interviennent. C'est pourquoi, de nombreux formats de données issus de XML apparaissent (il en existe plus d'une centaine) :

- ✓ OFX : Open Financial eXchange pour les échanges d'informations dans le monde financier
- ✓ MathML : Mathematical Markup Language permet de représenter des formules mathématique
- ✓ CML : Chemical Markup Language permet de décrire des composés chimiques
- ✓ SMIL : Synchronized Multimedia Integration Language permet de créer des présentations multimédia en synchronisant diverses sources : audio, vidéo, texte,...

Conclusion

XML est un standard pour:

- Structurer un document « primaire »
- Echanger des métadonnées

XML exprime un contenu et est indépendant de l'apparence

XML, pour être utile, doit être transformé:

- Formatage pour produire un document affichable ou imprimable
- Transformation pour réutilisation de l'information

L'utilisation d'XML suppose l'acquisition de connaissances et compétences techniques.

Bibliographie

Bibliographie

[1] : Douglas Comer, « Internet : Services et réseaux », Edition DUNOD 2003.

[2] Bertrand Petit , Architecture des réseaux, Page 5.

[3] Jean, François, Dillon, Introduction au réseau local, Edition Dunod.

[4] George et Olivier Gardarin, «Le client / serveur », Edition EYROLLES 2000.

[5] Alin Lefebvre, « L'architecture client-serveur », Edition ARMAND COLIN 2006.

[6] paolo Zanella & Yves Ligier, « Architecture et technologie des Ordinateurs », Dunod 4émé Edition 2005.

[7] D.Benabdesselam, « premier pas avec internet », Edition ORGANISATION 2002.

[8] www.wikipedia.org.

[9] http://www.dyalo.com/coursinformatique/introductionJ2EE.htm

[10] Documentations J2EE http://java.sun.com/j2ee/

[11] Développons en java : par jean Michel DOUDOUX

[12] http://www.jmdoudoux.fr/java/dej/indexavecframes.htm

[13] Jean-Michel DOUDOUX, « Développons en java ».

[14] www.pps.jussieu.fr/ejb/RMI/rmi2.html

[15] Ethan Cerami, Web Services Essentials, édition O'Reilly, février 2002.

[16] www.siteduzero.com

[17] Eric Lease Morgan « XML pour les bibliothécaires : un manuel et un atelier »

[18] GHENAIT née ABDAT Nadia et Melle MAHDAOUI Latifa édition 2007, « UML pour la pratique des systèmes d'information »

[19] www.usthb.dz site de l'Université Science Technique « Houari Boumediene »