*REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE*
*Ministère de l'Enseignement Supérieur et de la Recherche Scientifique*
*Université Mouloud MAMMERI de Tizi-Ouzou*
*Faculté de Génie Electrique et d'informatique*
*Département d'informatique*

# Mémoire de fin d'études

En vue de l'obtention du diplôme de Master II en informatique

**Option** : Système d'informatique.

**Thème :**

# Conception et réalisation d'une application Web avec J2EE pour la gestion de scolarité

# Cas : « 2IntPartners »

**Proposé et encadré par :**

**Mr DIB.**

**Réalisé par :**

***M*lle AOUDIA THANINA.**

***M*lle BACHIR SIHAM.**

**Promotion : 2013/2014**

# Remerciements

Nous tenons à témoigner notre reconnaissance à DIEU tout puissant, qui nous a aidé et bénis par sa volonté durant toute cette période.

Notre profonde gratitude et sincères remerciements vont à notre promoteur Mr DIB pour sa précieuse assistance, sa disponibilité et l'intérêt qu'il a manifesté pour notre travail.

Nos plus vifs remerciements vont à tout le personnel de l'école 2IntParteners  Qui nous ont généreusement aidées durant notre stage.

Tous nos remerciement  aux membres de jury qui nous font l'honneur de juger se travail.

Nos remerciements vont également à tous ceux qui ont répondu aux questions sur des problèmes rencontrés.

Nos remerciements vont aussi à tous ceux qui ont contribué de près ou de loin à la réalisation de notre travail.

# Dédicaces

## Je dédie ce travail À :

- *Mes très chers parents, qui m'ont soutenu tout au long de mes études.*
- *La mémoire de ma belle mère, A mon beau père et sa femme.*
- *Mon Fiancé qui m'a été la source d'encouragement.*
- *Mes Frères : Ahmed et son épouse et leur petite Kenza, Amar & Khaled.*
- *Mes beaux frères : Karim, Saïd & Yacine.*
- *Tous mes cousins & cousines.*
- *Mon binôme Thanina & surtout sa mère Ainsi que tout sa famille.*
- *Toutes mes amis :Kamilia, Kamilia, Rabia…*
- *Mes camarades de MasterII (2013/2014).*
- *Tous ceux qui m'aiment.*

*…SIHAM…*

# Dédicaces

## Je dédie ce travail :

- *À mes chers parents que j'aime énormément.*
- *À mes chers frères :Ouahmed, Massinissa, Kouceila.*
- *À mon adorable sœur que j'aime trop :Dyhia et À ces copines(G130).*
- *À mes meilleures proches: soraya, samira , Dymia qui ont été a mes cotés durant ce travail, Sans oublié boussad.*
- *À mon binôme et toute sa famille.*
- *À tout mes amis(es), Surtout : À Celia qui ma trop aidé et encourager.*
- *À toute personne qui m'aime.*

*…THANINA…*

# Sommaire

# Sommaire



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



**Chapitre III : J2EE & Framework:**

# Sommaire



**Chapitre IV: Analyse & Conception:**

# Sommaire

**Chapitre V: Réalisation & mise en œuvre:**

# Liste des tableaux

# Figures

# Figures

Tout a long de l'histoire de l'informatique, les technologies ont dessiné et fait évoluer l'organisation de nos sociétés. Aujourd'hui les nouvelles technologies, et notamment celles de l'information et de la communication(NTIC) ouvrent la voie a des modèles organisationnels d'écoles plus adaptés.

Ceci laisse dire que quelle que soit la taille d'une école de formation grande ou petite, l'inclusion des nouvelles technologies informatiques est devenue indispensable et s'avère nécessaire pour son bon fonctionnement surtout dans le domaine de la gestion de l'information. Elles permettent d'améliorer, d'assurer une gestion plus fiable, plus rigoureuse, de minimiser le risque d'erreurs et d'assurer la disponibilité de l'information à toute éventuelle demande.

Notre travail consiste à concrétiser les avantages que les nouvelles technologies offres en réalisant une application Web avec J2EE pour la gestion de scolarité pour l'école 2IntPartners.

Pour la bonne organisation de notre travail, nous avons adopté la structure chapitrale suivante :

- ✓ Le premier chapitre, « Introduction aux Technologies Web » donne un aperçu général sur les nouvelles technologies de l'information et de la communication (NTIC).
- ✓ Le second chapitre, « Présentation de l'organisme d'accueil » présente l'organisme d'accueil, ainsi que la problématique.
- ✓ Le troisième chapitre, « J2EE & Framework » consiste à présenter la plate forme J2EE, et donne un aperçu général sur les Frameworks, ainsi la justification de choix d'utilisé le framwork Hibernate dans le développement de notre application.
- ✓ Le quatrième chapitre, « Analyse & Conception » est consacré à la conception de l'application en utilisant le langage de modélisation UML.
- ✓ Le cinquième chapitre, « Réalisation & mise en œuvre **»** en détaillant principalement le fonctionnement de l'application et les outils utilisés pour son développement.

## I.1. Introduction :

L'architecture des applications modernes d'entreprise est le résultat d'un processus de sélection naturelle rapide où les critères de compétitivité s'appellent modularité, maintenance, portabilité, efficacité, évolutivité... Sous la pression de ces critères et du continuel renouvellement des technologies, l'architecture des applications a progressivement évolué du système Mainframe à l'application Web en passant par le client / serveur et les systèmes distribués. Et Le terme générique « réseau » définit un ensemble d'entités (objets, personnes, etc.) interconnectées les unes avec les autres.

Un réseau permet ainsi de faire circuler des éléments matériels ou immatériels entre chacune de ces entités selon des règles bien définies.

## I.2. Les réseaux informatiques :

### I.2.1. Définition :

Un réseau est un ensemble de matériels et de logiciels permettant de connecter deux ou plusieurs ordinateurs géographiquement dispersés afin d'échanger des données. Les données circulent dans le réseau sous forme de flux de bits.ces bits sont transférés sous forme de paquets de données qui se déplacent rapidement et qui traversent plusieurs réseaux pour arriver a la destination. Quelque soit la durée de déplacement, l'information doit circuler d'une manière fiable et atteindre sans erreurs la destination voulue.

### I.2.2. Caractéristiques d'un réseau :

Les réseaux locaux sont des infrastructures complexes et pas seulement des câbles entre stations de travail. Si l'on énumère la liste des composants d'un réseau local, on sera surpris d'en trouvé une quantité plus grand que prévue :

- ✓ Le câblage constitue l'infrastructure physique, avec le choix entre paires téléphoniques, câble coaxial ou fibre optique. Il détermine le type de concentrateurs (Switch, HUB, point d'accès Wifi,…etc.)utilisé.ces équipement constituent les nœuds dans le cas de réseaux en étoile.
- ✓ La méthode d'accès décrit la façon dont le réseau arbitre les communications des différentes stations sur le câble : ordre, temps de parole, organisation des messages. Elle dépend étroitement de la topologie et donc de l'organisation spatiale des stations les unes par rapport aux autres. La méthode d'accès est essentiellement matérialisée dans les cartes d'interfaces, qui connectent les stations au câble.
- ✓ Les protocoles de réseaux sont des logiciels qui « tournent »à la fois sur les différentes stations et leurs cartes d'interfaces réseaux. c'est le langage de communication. pour que deux structures connectées sur le réseau, ils doivent « parler » le même protocole.

- ✓ Le Système d'exploitation du serveur réseau, souvent nommé gestionnaire du réseau, est installé sur le ou les serveurs. Il gère les partages, droits d'accès, …etc. Pour Microsoft on trouve Windows NT serveur, Windows 2000 serveur, Windows 2003, 2008. Ce sont des versions spécifiques. Linux est utilisé sous différentes versions serveurs. Novell Netware est un système dédié principalement efficace comme serveur de fichier.
- ✓ Le système de sauvegarde est un élément indispensable qui fonctionne de diverses manières soit en recopiant systématiquement tous les fichiers du ou des serveurs, soit en faisant des sauvegardes régulières, éventuellement automatisées.
- ✓ Un pont, Un routeur ou passerelle constituent les moyens de communication qui permettent à un de ses utilisateurs de "sortir" du réseau local pour atteindre d'autres réseaux locaux ou des serveurs distants, Internet ou autres.
- ✓ Le système de gestion et d'administration du réseau envoie les alarmes en cas d'incidents, comptabilise le trafic, mémorise l'activité de réseau et aide le superviseur à prévoir l'évolution de son réseau. Cette partie est typiquement software.

### I.2.3. Le modèle OSI :

Le modèle OSI (Open System Interconnection Model) défini en 1977 régit la communication entre 2 systèmes informatiques selon 7 niveaux. A chaque niveau, Les deux systèmes doivent communiquer « compatibles ».En matériel réseau, nous n'utilisant que les couches inférieures, jusqu'à niveau 3. Ces niveaux sont également appelés couches.

L'OSI est un modèle de base normalisé par l'International Standard Organisation(ISO).

Ce tableau montre les différentes couches de modèle OSI :

| | | | | | | |
|---|---|---|---|---|---|---|
| Application | | Couche Application | 7 | Couche Application | | |
| | | Couche Présentation | 6 | Couche Présentation | | |
| | | Couche Session | 5 | Couche Session | | |
| Transport des données | | Couche transport | 4 | Couche Transport | | |
| | | Couche Réseau (Network) | 3 | Couche Réseau (Network) | | Paquet |
| | | Couche liaison de données (Data Link) | 2 | Couche liaison de données(Data Link) | | Trames |
| | | Physique (Physical) | 1 | Couche Physique (Physical) | | BIT |
| | | Support de communication | | | | |

**Tableau (I.1) : Les couches de modèle OSI.**

- ✓ **Niveau 7 (application) :** gère le format des données entre logiciels.
- ✓ **Niveau 6 (Présentation) :** met les données en forme, éventuellement de l'encryptage de la compression, par exemple mise en forme des textes, images et vidéo.
- ✓ **Niveau 5 (session) :** gère l'établissement, la gestion et coordination des communications.
- ✓ **Niveau 4(Transport) :** s'occupe de la gestion des erreurs, notamment avec les protocoles UDP et TCP/IP.
- ✓ **Niveau 3(réseau) :** sélectionne les routes de transport (routage) et s'occupe du traitement et du transfert des messages : gère par exemple les protocoles IP (adresse et le masque de sous-réseau) et ICMP. Utilisé par les routeurs et les switchs manageables.
- ✓ **Niveau 2 (liaison de données) :** Utilise les adresses MAC. Le message Ethernet à ce stade est la trame, il est constitué d'un en-tête et des informations. L'en-tête reprend l'adresse MAC de départ, celle d'arrivée+une indication du protocole supérieur.
- ✓ **Niveau 1 (Physique) :** gère les connections matérielles et la transmission, définit la façon dont les données sont converties en signaux numériques : ca peut-être un câble coaxial, paires sur RJ45, onde radio, fibre optique,…etc.

### I.2.4. Le modèle TCP/IP:

Le modèle TCP/IP s'inspire du modèle OSI auquel il reprend l'approche modulaire mais réduit le nombre à quatre. Les trois couches supérieures du modèle OSI sont souvent utilisées par une même application. Ce n'est pas le cas du modèle TCP/IP. C'est actuellement le modèle théorique le plus utilisé.

Le tableau suivant montre les deux modèles OSI et TCP/IP :

<table>
<tr><th>Protocoles utilisés</th><th>Modèle TCP/IP</th><th>Correspondance en OSI</th></tr>
<tr><td rowspan="3"></td><td rowspan="3">Couche Application</td><td>Application</td></tr>
<tr><td>Présentation</td></tr>
<tr><td>Session</td></tr>
<tr><td>TCP/UDO, gestion des erreurs</td><td>Couche Transport</td><td>Transport</td></tr>
<tr><td>IP/ARP et RARP/ICMP/IGMP</td><td>Couche Internet</td><td>Réseau</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">Couche Accès réseau</td><td>Liaison de données</td></tr>
<tr><td>Physique</td></tr>
</table>

**Tableau(I.2) : Les modèles OSI et TCP/IP.**

De nouveau, on ajoute à chaque niveau un entête, les dénominations des paquets de données changent chaque fois :

- ✓ Le paquet de données est appelé message au niveau de la couche application
- ✓ Le message est ensuite encapsulé sous forme de segment dans la couche transport. Le message est donc découpé en morceau avant envoi pour respecter une taille maximum suivant le MTU.
- ✓ Le segment une fois encapsulé dans la couche Internet prend le nom de datagramme
- ✓ Enfin, on parle de trame envoyée sur le réseau au niveau de la couche accès réseau.

Les couches du modèle TCP/IP sont plus générales que celles du modèle OSI.

**A. Couche application :**

Elle reprend les applications standards en réseau informatique et internet :

- ✓ SMTP : « Simple Mail Transport Protocol »gère le transfert de mails entre serveurs
- ✓ POP : gère le transfert des mails entre un serveur de messagerie et un ordinateur client
- ✓ TELNET : connexion sur une machine distante(serveur) en tant qu'utilisateur
- ✓ FTP (File Transfert Protocol), transfert des fichiers via Internet et beaucoup d'autres.

**B. Couche transport:**

La Couche transport permet le transfert des données et les contrôles qui permettent de vérifier l'état de la transmission.

Les protocoles des couches suivantes permettent d'envoyer des données issues de la couche application. On ne définit pas réellement les logiciels qui communiquent, mais des numéros de ports associés au type d'application (numéro variant de 0 à 65535,$2^{16}$). Par exemple, la navigation Internet utilise le port TCP 80, l'http, le 443, le FTP utilise le 21, …etc.

La couche transport gère 2 protocoles de transport des informations, indépendamment du type de réseau utilisé :

- ✓ TCP est orienté connexion (il vérifie la bonne transmission de données par des signaux d'accusées de réception –a cknowledge – du destinataire), il assure ainsi le contrôle des données.

- ✓ UDP, archaïque et non orienté connexion, n'assure aucun contrôle de transmission des données, par exemple utilisé en streaming.

Ces 2 types (orienté connexion ou pas) sont une notion utilisé pour les firewalls. Si vous fermez un port en TCP, l'envoi d'un message ne renvoie pas de signal de retour (a cknowledge), faisant croire que l'adresse IP est libre, non utilisé. En UDP au contraire, un port fermé ne renvois pas d'informations, faisant croire à une adresse IP utilisé. Le protocole UDP renvoie uniquement un message si le port est en erreur (ne répond pas).

### C. Couche Internet:

La couche Internet est chargée de fournir le paquet des données. Elle définit les datagrammes et gère la décomposition/recomposition des segments.

La couche Internet utilise 5 protocoles, seuls les 3 premiers sont importants :

1. Le protocole IP : gère les destinations des messages, adresse du destinataire.
2. Le protocole ARP (Adresse Resolution Protocole) : gère les adresses des cartes réseaux et la correspondance avec l'adresse IP. Chaque carte a sa propre MAC d'identification codée sur 48 bits.
3. Le protocole ICMP (Internet Control Message Protocol) gère les informations relatives aux erreurs de transmission. ICMP ne les corrige pas, il signale uniquement que le message contient des erreurs, utilisé par exemple par la commande DOS Ping.
4. Le protocole RARP(Reverse Addresse Resolution Protocol) gère l'adresse IP pour les équipements réseaux qui ne peuvent en récupérer une automatiquement par lecture d'information dans un fichier de configuration ou via un serveur DHCP. Lorsqu'un équipement réseau démarre, le gestionnaire réseau lit l'adresse IP à utiliser, ce qui est impossible pour certains équipements qui ne possède pas de disque durs (principalement les terminaux).
5. Le protocole IGMP (Internet Group Management Protocol) permet d'envoyer le même message à des ordinateurs qui font partie d'un groupe. Il permet aussi à ces machines de s'abonner et se désabonner d'un groupe. La principale application HARDWARE de l'IGMP se trouve dans les SWITCH manageables. Ce protocole permet de regrouper des stations.

### D. Couche Accès réseau :

La couche accès réseau spécifie la forme sous laquelle les données doivent etre transmises. Elle prend en charge les notions suivantes :

- ✓ Type de réseau (Ethernet, Token Ring, …), y compris les cartes réseau

- ✓ Transfert des données
- ✓ Synchronisation de la transmission de données
- ✓ Mise en forme (format) des données
- ✓ Conversion analogique/numérique pour les modems téléphoniques
- ✓ Contrôle des erreurs.

### I.2.5. Sécurité et administration :

Un des aspects important d'un réseau informatique local est la centralisation de l'administration des données. Ceci permet de sauvegarder et sécuriser les données sur une seule machine. La sécurité reprend un ensemble de mesures contre les instructions et virus, la gestion des privilèges et droits d'accès, la sauvegarde quotidienne des données, des équipements redondants en cas de panne, …

Il n'y a pas de solution idéales pour la sécurité des réseaux (et pour la sécurité informatique en générale). Trois solutions sont envisageables : les solutions matérielles que nous verrons, des solutions basées sur Linux et des solutions basées sur Windows ou des programmes rajoutés sur ces stations sont d'ailleurs complémentaires. Sur un gros réseau « sensible », mettre un VPN hardware n'est pas suffisant. Une sécurité logicielle complémentaire incluant des contrôles d'accès au niveau administration serveur (serveur, dossier, droits d'accès) et logiciels de sécurités vérifiant le trafic sur le réseau interne n'est pas superflu.

- ✓ Les routeurs peuvent être remplacés par une solution basée sur un serveur 2003 ou 2008, par un logiciel proxy comme Wingate ou par un ordinateur configuré spécifiquement en Linux.
- ✓ Un serveur proxy est parfois intégé dans les routeurs (mais généralement sous Windows ou Linux)
- ✓ Les firwalls sont intégrés dans certains routeurs mais des logiciels assurent (presque) des fonctions équivalentes, souvent intégrés dans l'anti-virus (ex : Symantec, ZoneAlarm, Mcafee au niveau des stations)
- ✓ Les réseaux privés intégrés (VPN) sont intégrés dans certains systèmes d'exploitation serveurs mais peuvent également etre des équipements spécifiques.
- ✓ Les anti-virus sont le plus souvent des logiciels, mais peuvent etre implimantés dans des routeurs qui vérifient tout le trafic extérieur.

## I.3. Architecture client/serveur :

### I.3.1. Définition : [1]

De nombreuses applications fonctionnent selon un environnement Client/serveur, cela signifie que des machines clientes (des machines faisant partie du réseau) contactent un serveur, une machine généralement très puissante en terme de capacités d'entrée-sortie, qui leur fournit des services. Ces services sont des programmes fournissant des données telles que l'heure, des fichiers, une connexion, etc.

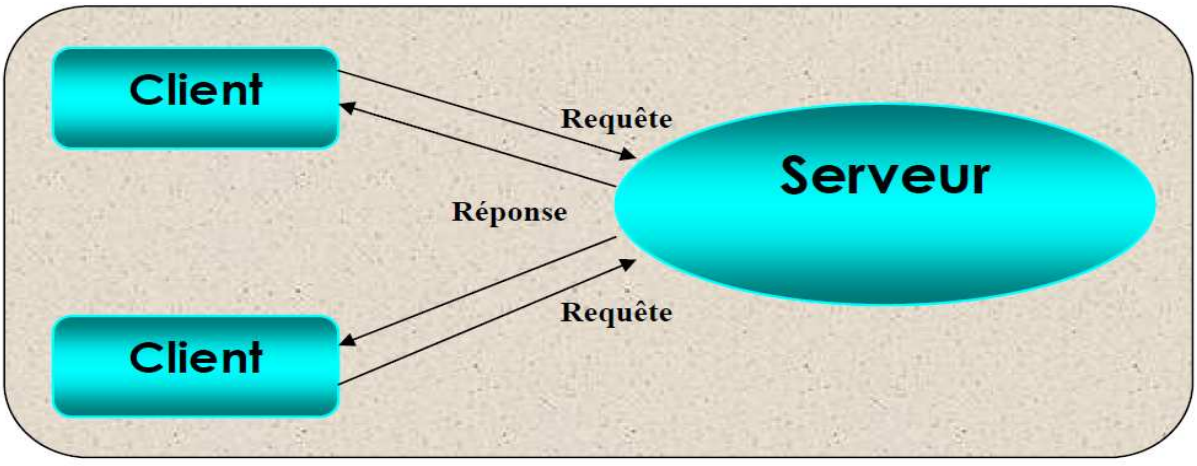
Les services sont exploités par des programmes, appelés programmes clients, s'exécutant sur les machines clientes. On parle ainsi de client lorsque l'on désigne un programme tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès d'un serveur.

### I.3.2. Notions de base :

- ✓ **Client :** c'est le 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.
- ✓ **Serveur :** c'est un processus accomplissant une opération sur demande d'un client.
- ✓ **Requête :** c'est un message transmit par un client à un serveur décrivant l'opération à exécuter pour le compte d'un client.
- ✓ **Réponse :** C'est un message transmis par un serveur à un client suite à l'exécution d'une opération contenant les paramètres de retour de l'opération.
- ✓ **Middleware :** C'est le logiciel qui est au milieu assure les dialogues entre les clients et les serveurs souvent hétérogènes.

### I.3.3. Fonctionnement d'un système client/serveur : [2]

Un système client/serveur fonctionne selon le schéma suivant :




**Figure(I.1) : principe de fonctionnement du client/serveur.**

- ✓ Le client émet une requête vers le serveur grâce à son adresse IP 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 client et son port.

### I.3.4. Les types des serveurs pour le système client/serveur :

a) **Serveur de base de données :**

Dans le cas de serveur de base de données, le client émet des requêtes SQL sous forme de message en direction du serveur. Le résultat de chaque requête est envoyé au client. Les données ainsi que le code qui traite les requêtes, réside sur la même machine (serveur). Le serveur utilise sa propre capacité de traitement pour rechercher les données demandées au lieu de transmettre tous les articles au client et de laisser en faire la sélection. Ainsi la puissance répartie est utilisée de façon beaucoup plus efficace.

b) **Serveur de transactions :**

Dans ce modèle, les clients invoquent des procédures distantes résidant sur le serveur qui comporte un moteur de base de données SQL. Ces procédures distantes exécutent un ensemble d'instructions SQL. L'échange sur le réseau consiste en un seul message de Requête/Réponse (une réponse pour un bloc de requête SQL). Pour ce type de serveur l'application Client/Serveur nécessite du code source au niveau du serveur.

c) **Serveur de groupware :**

Le groupware s'intéresse à la gestion d'informations semi structurées telles que le texte, l'image, le courrier, la messagerie et l'ordonnancement des tâches. Ces systèmes Client/Serveur mettent les utilisateurs en contact direct les uns avec les autres. Microsoft Exchange est un exemple de ce type.

d) **Serveur d'application objet :**

Dans ce type de serveur, l'application Client/Serveur est écrite sous forme d'un jeu d'objets communicants. Les objets clients communiquent avec les objets serveurs au moyen d'un courtier d'objet ou ORB (Object Request Broker). Le client invoque une méthode sur un objet disant, l'ORB localise une instance de la classe, appelle la méthode demandée et envoie les résultats l'objet client.

e) **Serveur d'application Web :**

L'internet est la plus grande application Client/Serveur dite intergalactique. Ce nouveau modèle consiste en des clients légers et portables qui communiquent avec des très gros serveurs. Le serveur Web par exemple, renvoie des documents lorsque le client les demande par leurs noms. Client et serveur communiquent via un protocole de type RPC appelé HTTP (Hyper Text Transmission Protocol).

f) **Serveur de fichiers :**

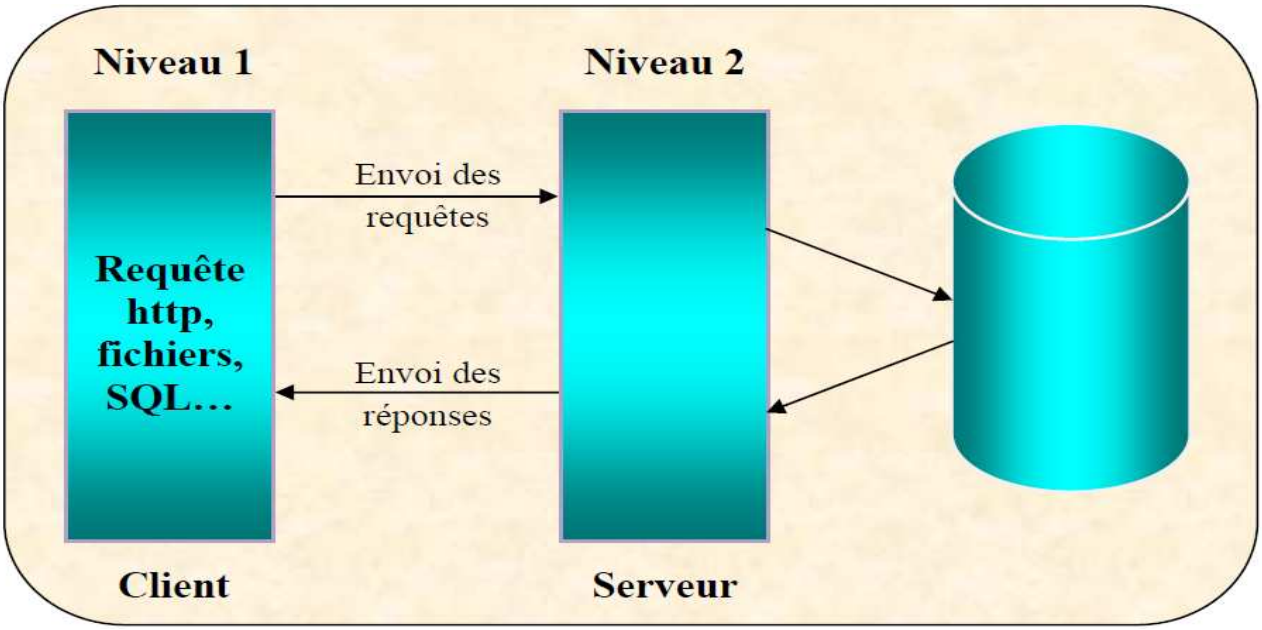
Les serveurs de fichiers sont utilisés pour partager des fichiers sur un réseau et ils sont indispensables pour créer des banques de documents, d'images…etc. Sa faiblesse réside dans l'obtention de l'information qui nécessite de nombreux échanges sur le réseau.

### I.3.5. Classification des architectures client/serveur :

#### I.3.5.1. Architecture à deux niveaux : [3]

L'architecture à deux niveaux (aussi appelée architecture 2-tier) caractérise des systèmes client/serveur pour lequel le client demande une ressource et le serveur la lui fournit directement, en utilisant ses propres ressources.

Cela signifie que le serveur ne fait pas appel à une autre application afin de fournir une partie du service.

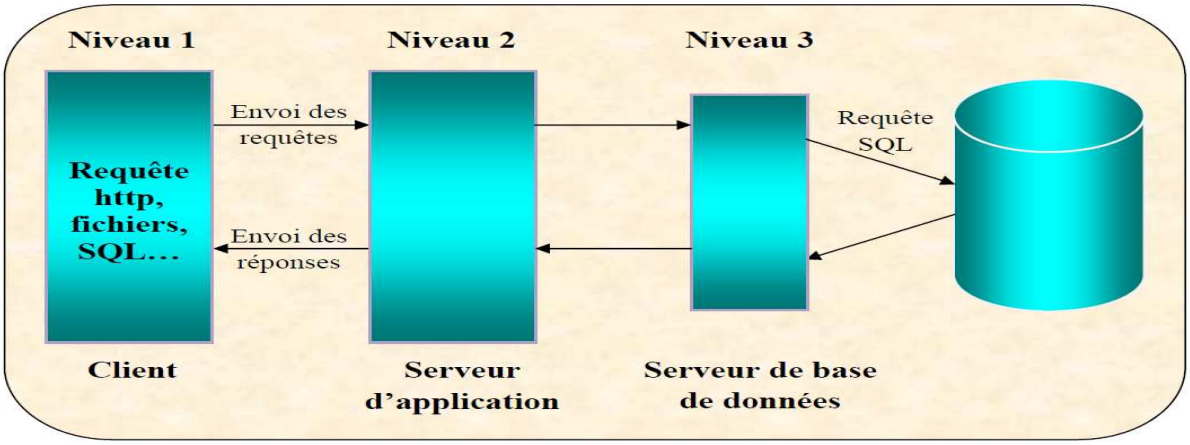



**Figure(I.2) : L'architecture client/serveur à deux niveaux.**

#### I.3.5.2. Architecture à trois niveaux : [4]

Dans l'architecture à trois niveaux, il existe un niveau intermédiaire, c'est-à-dire que nous avons généralement une architecture partagée entre :

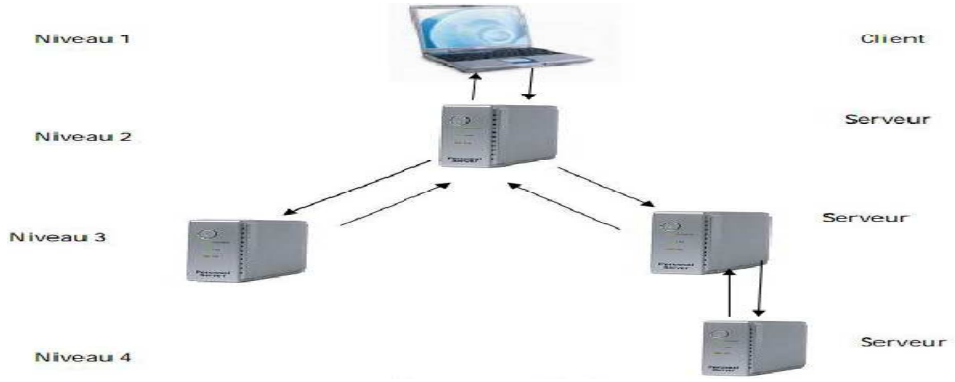
- ✓ Un client : C'est-à-dire l'ordinateur demande de ressources, équipée d'une interface utilisateur (généralement un navigateur web) chargée de la présentation.
- ✓ Le serveur d'application (appelé également middleware) : le serveur chargé de fournir la ressource mais faisant appel à un autre serveur.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




**Figure(I.3) : L'architecture client/serveur à trois niveaux.**

### I.3.5.3. Architecture multi niveaux :

L'architecture à multi niveaux a été pensée pour pallier aux limitations des architectures 3 tiers et concevoir des applications puissantes et simples à maintenir.




**Figure(I.4) : L'architecture client/serveur à multi niveaux.**

### I.3.6. Caractéristiques du Clients/Serveur :

- ✓ **Le service :** c'est le travail fourni par le serveur suite à la requête du client. Le serveur est un fournisseur de services et le client consommateur de ces services. Le modèle Client/serveur établit ainsi une séparation claire des rôles à partir de la notion du service.
- ✓ **Les ressources partagées :** le serveur est capable de servir de nombreux clients simultanément et réguler leur accès.
- ✓ **Les protocoles asymétriques :** la relation entre clients et serveur est de type « plusieurs vers un », toutefois le client est le déclencheur du dialogue en demandant un service alors que le serveur attend passivement les requêtes.
- ✓ **Assemblage multi-vendeur :** Le client/serveur est indépendant de la plate forme matérielle ou de système d'exploitation. On doit toujours pouvoir mélanger et apparier les plates formes client/serveur.

- ✓ **L'échange de messages :** Le client et le serveur sont des systèmes à liaison épisodique qui interagissent au moyen de messages. Le message est un mécanisme d'émission de demandes de services et de réponses à celles-ci.
- ✓ **L'encapsulation des services :** le serveur choisit la manière dont il réalise le service demandé ; le client se borne à définir ce qu'il désire obtenir. Ainsi l'implémentation du serveur peut être changée sans préoccuper le client.
- ✓ **Intégrité :** le code et les données du serveur sont gérés de façon centralisée, ce qui garantie un moindre cout de maintenance et une meilleure intégrité des données tandis que les clients restent individuels et indépendant.
- ✓ **Souplesse et adaptabilité :** On peut modifier le module serveur sans toucher au module client et vice versa. Si une station est remplacée par un modèle plus récent, on modifie le module client sans modifier le module serveur.

### I.3.7. Les Avantages et Les Inconvénients de l'architecture client/serveur :

#### I.3.7.1. Avantages de l'architecture client/serveur :

Les principaux avantages sont :

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

#### I.3.7.2. Inconvénients de l'architecture client/serveur :

L'architecture Client/serveur a tout de même quelques inconvénients parmi lesquelles :

- ✓ **Un coût élevé :** Dû à la technicité du serveur.
- ✓ **Un maillon faible :** Le serveur est le seul maillon faible du réseau, étant donné que tout le réseau est construit autour de lui.

## I.4. Internet et web :

### I.4.1. Internet :

#### I.4.1.1. Définition :

Internet est un **réseau informatique** qui relie des ordinateurs du monde entier entre eux et qui leur permet d'échanger des informations. Les données sont transmises par l'intermédiaire de lignes téléphoniques, de câbles ou de satellites.

Pour communiquer entre eux, les ordinateurs connectés à Internet utilisent un protocole de transmission et de communication constituant un langage commun. Ce langage s'appelle la Transmission Control Protocol / Internet Protocol (**TCP-IP).**

#### I.4.1.2. Les différents protocoles d'internet :

Un protocole est un ensemble de convention et de règles permettant la gestion de la manière avec laquelle s'effectue la communication entre plusieurs machines.

**A) les protocoles TCP/IP (Transmission Control Protocole/Internet Protocole) :**

Pour faire fonctionner un vaste réseau, Internet met en œuvre des protocoles de communication. Les deux protocoles de base sont TCP (Transmission Control Protocol) et IP (Internet Protocol). Ils sont référencés sous le vocable TCP/IP.

TCP récupère les informations à transmettre, IP les transforme en paquets et les délivre à la couche chargée du transport (composant électronique).

Comme ces protocoles sont faciles à mettre en place et permettent d'interconnecter tous les types de réseaux, ils se sont rapidement développés, et la technologie TCP/IP a permis l'existence de l'internet mais aussi de nombreux inter-réseaux.

- ✓ **Le protocole IP :** Il est mis en œuvre au niveau de la couche réseaux, sa tâche principale est de convertir le flux d'information transmis par TCP en paquets de données IP et les envoie vers une destination décrite par une adresse IP attribuer à chaque ordinateur.
    - **L'adresse IP :** Une adresse IP est un nombre binaire codé sur 32bits et attribué à un hôte, elle est utilisée pour toute communication avec cet hôte, elle est toujours représentée dans une notation décimale constituée de quatre nombres, chacun compris entre [0-255] et séparés par un point.

Les adresses IP sont reparties en fonction du nombre d'octets utilisés pour désigner le réseau.

- ✓ **Le protocole TCP :** TCP est le protocole de transport le plus important de la suite de TCP/IP, il offre aux programmes d'application un service de transport fiable de flots de données bidirectionnel avec contrôle de flux, Après avoir établi une connexion, les programmes d'application se servent de cette dernière pour s'échanger des données, et TCP garantit que ces données arrivent dans l'ordre et sans duplication.

**B) HTTP (HyperText transport Protocol)** :

C'est un protocole léger et rapide, utilisé pour délivrer des fichiers multimédia et HyperText appelés plus généralement ressources, en utilisant internet, il gère la totalité des échanges réalisés entre le client et le serveur .pour chaque demande d'accès à l'URL, une requête http est émise indépendamment par le client.

**C) DNS (Domaine Name service)** :

C'est un ensemble de protocole permettant aux utilisateurs d'un réseau TCP/IP d'accéder aux hôtes à l'aide des noms conviviaux hiérarchisés, le serveur DNS assure la transformation des noms des domaines en adresse IP et inversement, car la structure des adresses IP est complexe à manipuler pour cela on utilise le DNS, sans celui-ci les utilisateurs devraient mémoriser les adresse IP.

### I.4.1.3. Les services internet :

L'internet offre plusieurs services qui sont relatifs aux différents protocoles de communication parmi ces services nous citons :

**A) Messagerie électronique E-mail (électronique mail) :**

C'est un service qui offre la possibilité à ces utilisateurs de communiquer et de recevoir des messages texte, des images, des programmes, des sons,… par des messages électroniques.

**B) Le transfert de fichier : FTP (File Transfer Protocol) :**

En se connectant sur Le serveur FTP qui nous donne l'accès à des fichiers, les copies, et les déplacer entre les différents réseaux connectés à l'internet, et le protocole FTP a été mis au point avant l'introduction du web.

**C) Le service IRC : (Internet Relay Chat) :**

« CHAT »est un terme en anglais qui signifie en français « bavarder »ainsi IRC permet le bavardage en directe sur Internet. Il permet l'échange de messages textuels en temps réel.

**D) Les groupes de new (new group) :**

C'est une messagerie électronique qui permet à toute personne de chercher de l'aide, faire la recherche sur un sujet bien défini ou d'envoyer un courrier électronique vers un espace ouvert à tout le monde.

**E) La visioconférence :**

La visioconférence permet aux utilisateurs de communiquer en direct sur internet, avec échange de son et d'images, il est utilisé pour ramener l'école, travaille… a la maison sans se déplacer.

**F) Le World Wide Web :** en français : Toile d'araignée mondiale :

C'est le service le plus connu, le plus récent et maintenant le plus utilisé par le réseau Internet et naviguer entre des documents reliés par des liens hypertexte.

### I.4.1.4. Intranet-extranet : [5]

**a) Intranet**

L'intranet est un réseau informatique utilisé à l'intérieur d'une entreprise ou de toute autre entité organisationnelle utilisant les techniques de communication d'internet (IP, serveurs http).dans les grandes entreprises, l'intranet fait l'objet d'une gouvernance particulière en raison de sa pénétration dans l'ensemble des rouages des organisations.

Un intranet repose généralement sur une architecture à trois niveaux composée :de clients (navigateur internet généralement), d'un ou plusieurs serveurs d'applications (middleware) :un serveur web permettant d'interpréter des scripts PHP, ASP, ou autre, et les traduire en requêtes SQL afin d'interroger une base de données et d'un serveur de bases de données. De cette façon, les machines clientes gèrent l'interface graphique, tandis que les différents serveurs manipulent les données. Le réseau permet de véhiculer les requêtes et les réponses entre clients et serveurs.

**b) Extranet**

Un extranet est une extension de l'entreprise à des partenaires situés au-delà du réseau et ce de manière sécurisée (authentification par un mot de passe). De cette façon un extranet n'est ni un intranet ni un site web internet, il s'agit d'un système supplémentaire offrant par exemple aux clients, à ses partenaires ou à des filiales un accès privilégié à certaines ressources informatiques de l'entreprise par l'intermédiaire d'une interface web.

### I.4.2. Le Web :

#### I.4.2.1. Définition :

Le web est un système qui fonctionne en mode client/serveur sur internet, il permet de mettre des informations sous forme de documents hypertextes.

Pour accéder au web il est nécessaire de disposer d'un logiciel appelé Navigateur web. L'accès à un document est conditionné par la connaissance de sa localisation qui est exprimée sous forme d'URL (Uniform ressource Locator).

**Hypertexte**: partie du texte ou image sur lesquelles il est possible de cliquer avec la sourie de manière à se déplacer vers d'autres documents

#### I.4.2.2. Les caractéristiques du web :

Le web est le service le plus répondu fonctionnant sur internet, il est caractérisé par :

- ✓ **Un système hypermédia :** En www on parle de documents hypermédia car les liens peuvent référencer des fichiers, son, image ou des séquences vidéo avec un pointeur vers un autre document traversé lors d'un clic.
- ✓ **Un système distribué :** Le web est un système hypertexte distribué car les documents qui y sont mémorisés sont distribués sur tous les réseaux, l'internaute à l'impression que les pages qui sont mémorisés sur l'écran sont localisées dans le même endroit alors qu'il est possible qu'elles soient distribuées sur plusieurs mémoires d'un peu partout dans le monde.

### I.4.2.3. Les principaux termes du web :

- ✓ **Page web :** Une page web est une page sur Internet, c'est un fichier informatique unique écrit en HTML, ainsi il peut contenir des images, du son et/ou de la vidéo. Une des particularités des pages web est qu'elles n'ont pas de fin et peuvent contenir une infinité d'informations, leurs seules limites est le poids de la page et le temps qu'elle va mettre pour l'affichage.
- ✓ **Site web :** est un ensemble de pages web t d'éventuelles autre ressources, liées dans une structure cohérente publiées par un propriétaire (une entreprise, une administration, une association, un particulier, etc.) et hébergées sur un ou plusieurs serveurs web.
- ✓ **URL :** Une URL (*Uniform Ressource Locator*) est l'adresse d'accueil qui contient à lafois le nom d'une machine, le nom du service demandé, le nom d'un document …etc. C'est une chaîne de caractères permettant d'indiquer un protocole de communication et un emplacement pour toute ressource du Web. Il s'agit en quelque sorte d'un système de fichier universel sur Internet, elle se présente de la façon suivante : **http://www:numero_port/répertoire/nom_fichier.**
- ✓ **HTML ou XHTML (HyperText Mark-up Language):** toute page web comprend une base de langage HTML ou XHTML. Il s'agit donc d'un langage de balisage qui définit la structure de la page web.C'est un langage qui permet des hyperliens, à savoir des liens d'un document à un autre ou d'un endroit du document à un autre.
- ✓ **HyperText :** l'HyperText est une manière d'organiser et de présenter l'information dans laquelle certains éléments du texte, appelés liens, permettent de se déplacer vers d'autres zones du texte ou vers une autre page.
- ✓ **Navigateur web :**Un navigateur web est un logiciel client HTTP conçu pour accéder aux ressources du Web, on l'appelle aussi browser et les plus connus aujourd'hui sont Microsoft Internet Explorer (MSIE), sa fonction de base est de permettre la consultation des documents HTML disponibles sur les serveurs HTTP. Le support d'autres types de ressource et d'autres protocoles de communication dépendent du navigateur considéré.

### I.4.2.4. Les Sites web :

Un site web ou site Internet est un ensemble de pages web (fichiers HTML) dont on trouve en premier la page d'accueil, reliées entre elles par des points communs. Elles représentent souvent une entreprise et ses produits, ou bien un organisme et les services qu'il offre.

La communication entre sites est possible par le biais des liens sur leurs pages web.

### I.4.2.5. Classification des sites web :

La façon dont les types des sites web sont définis n'est qu'un classement choisi parmi plusieurs qui en existe, ainsi il comporte deux types :

- ✓ **Site web statique :**

C'est un site web qui est constitué de pages HTML prédéfinies, créées une fois pour toutes à l'aide d'un éditeur HTML. En effet l'administrateur du site compose avec un éditeur HTML des pages web stockées sur le serveur web, celui-ci renvoie ces pages à la demande aux visiteurs, par la suite ces pages ne pourront être modifiées que via un éditeur HTML, par l'administrateur. Le contenu de ces pages est fixe comme un fichier Word, et n'est pas modifié par le serveur. Le site est donc dit "statique" car son contenue change que par une intervention humaine et non pas par des fonctions automatiques opérées par le serveur.

- ✓ **Site web dynamique :**

C'est un site web dont les pages HTML se construisent lors de sa consultation par un internaute.

En effet l'administrateur du site et le visiteur utilisent le même outil : le navigateur web, mais les pages avec lesquelles travaille l'administrateur ne sont pas les mêmes que celles que le visiteur voit : il encode dans des pages d'administration (aussi appelées "formulaires"), qui nourrissent la base de données. De son côté, le visiteur visualise des pages qui font appel au contenu de la base de données.

C'est le serveur web qui s'occupe de la récupération du contenu des formulaires que remplit l'administrateur, et de renvoyer cette information dans les pages vues par le visiteur.

### I.4.2.6. Les technologies de programmation web :

**a) Coté client :**

- ✓ **HTML (HyperText Markup Language):** C'est le format de données conçu pour représenter les pages web. C'est un langage de balisage qui permet d'écrire de l'hypertext, d'où son nom. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires

de saisie, et des éléments programmables tels que des applets les principales fonctionnalités du langage HTML sont :

- Le support des tableaux.
- Le support des formulaires de saisie.
- La gestion des images maps (image cliquables).
- L'insertion de scripts (java script, VB script, CGI script, ……).

✓ **XML (Extension Markup Language) :** tous comme HTML, le langage XML représente un langage de balises décrivant la structure et la nature des données d'un document. Mais la différence du langage HTML, statique, XML est davantage orienté traitement des données et publication des documents complexes.

**b) Coté serveur :**

✓ **JSP :** technologie développée par un Sun Microsoft dans le but de créer des pages dynamiques avec le langage java, le JSP permettent d'ajouter du code java dans la page HTML qui sera interprétée par le serveur.

✓ **ASP :** technologie développée par Microsoft à partir de 1996 dans le but de créer des pages dynamiques. L'ASP permet d'ajouter de code dans les pages HTML qui sera interprété par le serveur. La partie ADO (ActiveX data Object) de l'ASP permet de se connecter à une base de données.

✓ **Les servlets :** le nom vient d'une analogie possible à faire avec les applets. Il s'agit donc de programmes crées en java et tournant sur le serveur web. L'exécution du programme génère les pages web renvoyées au client.

## I.5. Conclusion :

Au début de ce chapitre nous avons présenté la notion de base des réseaux informatiques qui sont un moyen pour minimiser les couts de transport des informations et d'augmenter les performances des systèmes, nous avons parlé des modèles OSI et TCP/IP, En suit on a parlé sur l'architecture client/serveur. En fin on a présenté certains concepts des technologies internet et web.

Ces notions seront utilisées pour le développement de notre application.

## II.1. Introduction :

Parmi les branches les plus considérable a l'obtention des bonnes formations et des bonnes capacités en étude, on trouvera l'école de formation qui est un service très utile pour l'amélioration et le développement des informations, elle occupe de charger le maximum des formations, pour avoir un nombre important des diplômes et surtout des certifications. Et Pour bien cerner notre projet, il est essentiel de bien comprendre le contexte dans lequel il est posé et d'avoir une vue générale sur l'organisme d'accueil, ce qui permet de préciser les frontières du domaine d'étude.

Pour ce faire nous avons suivi les points suivants :

- ✓ Historique de 2intparteners
- ✓ Situation géométrique
- ✓ Organisme générale de l'école
- ✓ Architecture de l'école
- ✓ Etude de personnel
- ✓ La mission de l'école

## II.2. Historique :

2IntPartner à été crée en 2011, au début elle avait contenu quelques formations qui sont : Microsoft, Cisco, java, piratage, ensuite en 2012 elle a eu un changement et une amélioration par des autres formations comme : initiation en informatique, agent de saisie, internet, développement web et les langues (français, anglais, allemand), finalement En 2013 ils ont ajouté des autres formations.

## II.3. Situation Géographique :

C'est une école qui ce trouve dans le centre commerciale "Le DRUGSTORE" place du stade 1er novembre Tizi-Ouzou. A sa 1er création elle n'été pas situées a cette place, mais après le temps elle a eu un changement de local en 2013.

## II.4. Architecture de l'école :

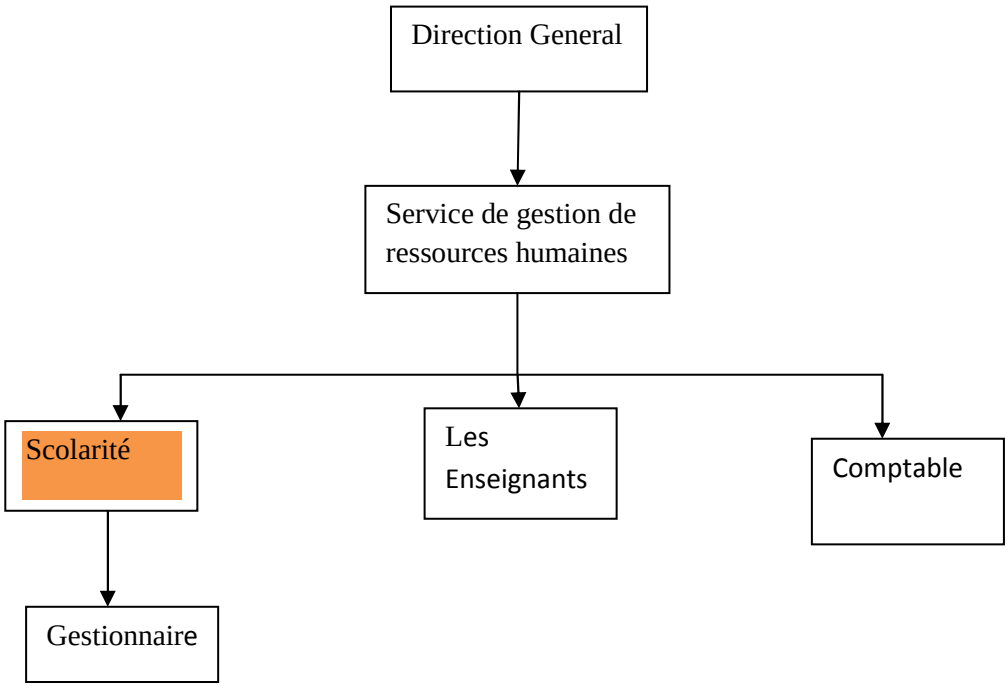
2intparteners est une école qui a été crée avec 7 salle au début, chaque salle est précisé comme suit :

- ✓ Une salle pour le master
- ✓ Une salle pour la maintenance
- ✓ Une autre pour la réception
- ✓ Et finalement, la dernière salle c'est pour le directeur des études.

Et en 2014 ils ont fait un élargissement pour l'école avec l'ouverture d'un nouveau bloc qui contient 3 salle de plus avec l'utilisation de 2 salle seulement comme suit :

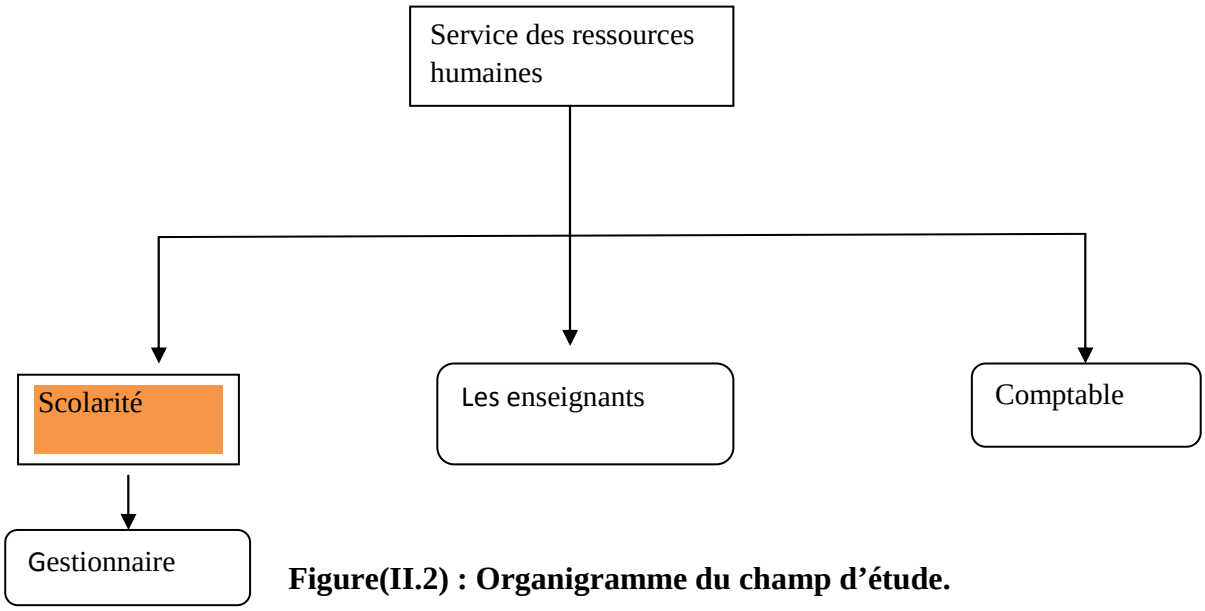
- ✓ Une salle pour les langues
- ✓ Une autre pour agent de saisie

## II.5. Organigramme générale de l'école :




**Figure(II.1) : Organigramme générale de l'école.**

### II.5.1. Organigramme du champ d'étude :




**Figure(II.2) : Organigramme du champ d'étude.**

### II.5.2. Les Rôles des différentes structures :

- ✓ Directeur : Il représente l'organisateur et l'ordonnateur de la direction de l'école, son rôle est de contrôler toutes les différentes activités de l'école, et encore d'enseigner au début a cause de manque d'enseignants, après ils ont ajouté deux profs pour l'aide, un est charger d'enseignait la formation Cisco et l'autre c'est un électronicien charger d'enseigné deux formations (maintenance et réseaux).
- ✓ Service des ressources humaines : C'est le service le plus important de l'école, il gère le recrutement du personnel de cette dernière, au début le recrutement d'une informaticienne chargée d'enseigner quelques formations (agent de saisie, développement web, Photoshop).

Récemment : le recrutement d'autres enseignants : en mangement, en langue (Allmen, français, anglais et chaque langue avec un prof de cette dernière),enseignant en java, un 2eme prof de Cisco, un enseignant pour oracle, un autre pour infographie, une enseignante en agent de saisie(c'est la 2eme) et enfaite le recrutement d'une nouvelle gestionnaire.

- ✓ La Scolarité : sous l'autorité du directeur, Il contribue au choix des spécialités et des étudiants, veille sur le respect des méthodes d'enseignement, des règles d'hygiène, de sécurité et de discipline.

Il propose les emplois du temps, participe aux actions d'information et d'inscription.

- ✓ Les enseignants : Un enseignant c'est la personne qui est chargé d'enseigner et d'informer les étudiants et de les aides à comprendre toute difficulté d'un module précis (bien sur on tenu compte avec chaque module pour un prof spécialisé).
- ✓ Comptable : Il est chargé de tout règlement des factures, les entrés et les sorties des formations c'est-à-dire (entrés : nombre de personne inscrit a chaque formation. et les sorties : c'est le nombre des gens qui sont finis chaque formation).
- ✓ Gestionnaire : C'est la personne la plus essentielle de l'école, elle est chargée de faire tout : de suivre tout changement d'information de l'école, elle examine par exemple toute les inscriptions a des formations quelque soit, elle prépare les bons de payement pour chaque étudiant…..etc.

### II.6. La mission de l'école :

La mission de l'école des nouvelles technologies 2intparteners est consiste à:

- ✓ Délivrer un nombre important du diplôme.

### II.7. La problématique :

Au cours de notre passage par différents services de l'école, 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…

Suite à ces problèmes, l'objectif principal de notre site est d'assurer les services suivants :

- ✓ Faciliter la gestion du nombre considérable du personnel de l'école.
- ✓ Eliminer les erreurs dues à la gestion aléatoire et manuelle des heures d'enseignements pour chaque formation.
- ✓ Assurer une bonne disposition des statistiques.

### II.8. Conclusion :

Nous avons présenté dans ce chapitre l'organisme d'accueil, la problématique posée, pour bien tirer les anomalies et les insuffisances rencontrées et remédier aux lacunes afin de nous préparer aux analyses concrètes de notre application développées au fur et à mesure dans les prochains chapitres de ce présent mémoire.

## III.1. Introduction :

La technologie java est aujourd'hui incontournable, tant pour le développement d'applications autonomes que pour l'utilisation des utiles orientés web. La technologie Java est ici employée afin de générer des pages HTML dynamiques (DHTML). Elles permettent l'intégration de l'ensemble des fonctionnalités de Java Entreprise Edition (XML, bases de données, services web, systèmes distribués CORBA, …etc.), contrairement à d'autres langages web (tel que PHP par exemple).

Dans ce chapitre nous allons présenter l'essentiel des connaissances permettant de développer des applications web en s'appuyant sur J2EE.

## III.2. Architecture J2EE :

### III.2.1. Définition :

Java 2 entreprise Edition (J2EE) est une spécification pour le langage de programmation Java de Sun destinée aux applications d'entreprise. J2EE offre une plate-forme de développement et déploiement en langage Java pour les applications distribuées à plusieurs niveaux. On parle généralement de « plate-forme J2EE » pour désigner l'ensemble constitué des services API (Application Programmable Interface) offerts et de l'infrastructure d'exécution.

Dans la mesure où J2EE s'appuie entièrement sur Java, il bénéficie des avantages de ce langage, en particulier une bonne portabilité et une maintenance du code. De plus, l'architecture J2EE repose sur des composants distincts, interchangeables et distribués, ce qui signifie notamment qu'un système reposant sur J2EE peut posséder des mécanismes de haute-disponibilité, afin de garantir une bonne qualité de service et que la maintenance des applications est facilitée.

### III.2.2. Fonctionnement interne :

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 et 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.

### III.2.3. L'Architecture de J2EE : [6]

J2EE ajoute de nombreuses couches de niveau entreprise 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 dynamique, par exemple JSP (Java server Pages) et Servlet.
- ✓ **Technologie entreprise application :** technologies plus directement liées à la logique de business : EJB (Entreprise 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 :** technologie liées à la gestion de la technologie entreprise afin de réaliser l'accès et l'échange d'information entre machines et services distribués : JAAS (Java Authentification and Autorisation Service), JCA (Java Connector Architecture).

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

L'architecture J2EE permet ainsi de séparer la couche présentation, correspondant à l'interface homme-machine (IHM), la couche métier contenant l'essentiel des traitements de données en se basant dans la mesure du possible sur des API existantes, et enfin la couche de données correspondant aux informations de l'entreprise stockées dans des fichiers, dans des bases de données relationnelles ou XML, dans des annuaires d'entreprise ou encore dans des systèmes d'informations complexes.

### III.2.4. Le serveur d'application : [6]

Les API J2EE ont était projetés pour travailler avec un particulier type de serveur appelé J2EE Application Server. Un serveur d'application est une couche software résident sur une machine serveur qui fournit les services que la technologie J2EE nécessite. Il y a plusieurs applications serveurs. Parmi les produits commerciaux on rappel Bea WebLogic, IBM Web sphère, Sun Application Serveur, etc. parmi les produits libres le plus connu est JBOSS. Les différences principales entre les différents serveurs d'applications sont relatives aux opérations de deploy, clustring, etc …. Par contre toutes les fonctionnalités qui concernent strictement le fonctionnement des applications J2EE adhérent aux spécifications proposées par la Sun.

Un serveur d'application s'installe et se lance comme un normal serveur web (de fait il met à disposition des services typiques d'un serveur web). En plus il possède un panneau d'administration accessible par un poste distant à travers lequel on peut définir les différents services. On peut considérer le serveur d'application comme une application Java standalone exécutée par une J2SE qui dessert les requêtes provenant des différents clients.

Les serveurs d'application se sont développés depuis la création de J2EE. On peut distinguer principalement 2 grandes catégories de serveurs :

- ✓ **Open source :** évolue grâce à la communauté par exemples Tomcat : Apache, Jonas : ObjectWeb, JBoss : JBoss …
- ✓ **Propriétaire :** évolue selon l'éditeur par exemples WebSphere : IBM, WebLogic : BEA, WebObject : Apple, Oracle Application Server : Oravle …

### III.2.5. Outils de programmation :

Plusieurs Environnements de développement intégrés (IDE) sont disponibles pour la plate-forme J2EE. Parmi les logiciels libres on rappel NetBeans de la Sun qu'a était créé exclusivement pour faciliter le développement d'applications J2EE. Encore on retrouve Eclipse de IBM, qui met à disposition, grâce au système des plug-ins, un riche environnement de développement, c'est l'outil que nous avons choisi pour le développement de notre application.

### III.2.6. Les avantages d'utiliser J2EE :

L'utilisation de J2EE pour développer et exécuter une application représente plusieurs avantages :

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

## III.3. Le modèle MVC :

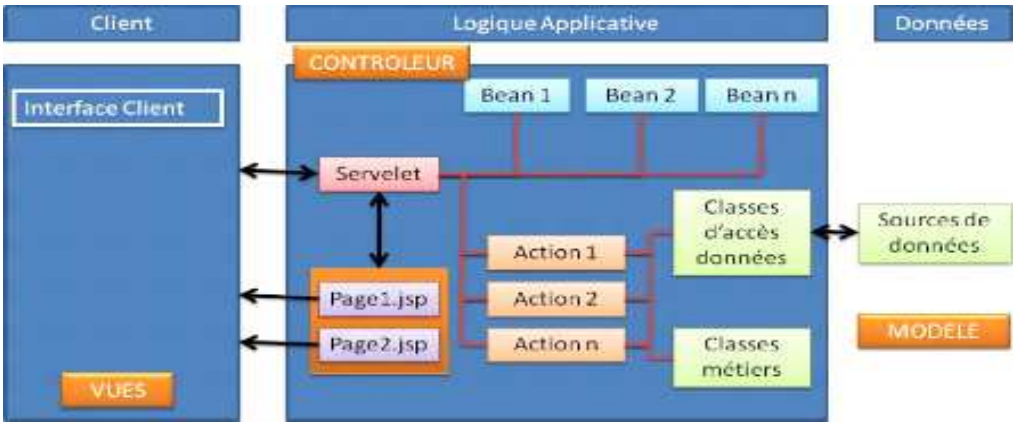
Le modèle MVC (Model View Controler) a été initialement développé pour le langage Smalltalk dans le but de mieux structurer une application avec une interface graphique.

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

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

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

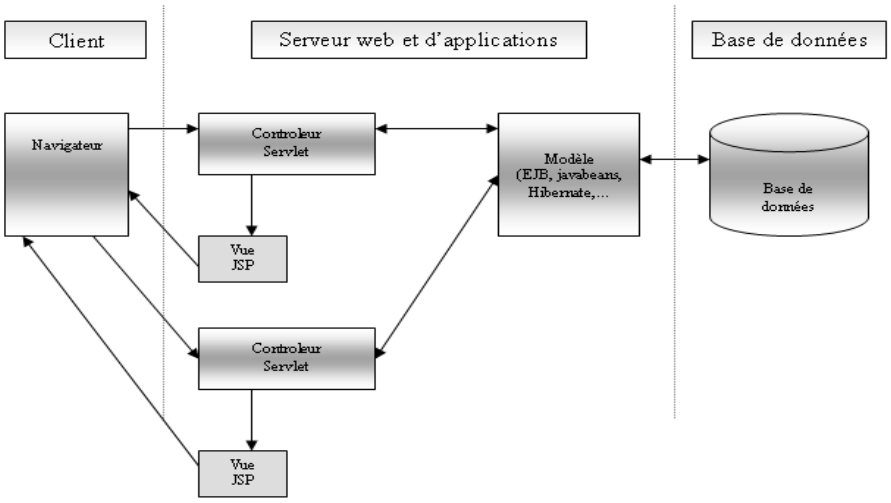
L'utilisation du modèle MVC rend un peu plus compliqué le développement de l'application qui le met en œuvre mais il permet une meilleure structuration de celle-ci.




**Figure(III.1) : architecture du modèle MVC.**

### III.3.1. Le modèle MVC type1 :

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



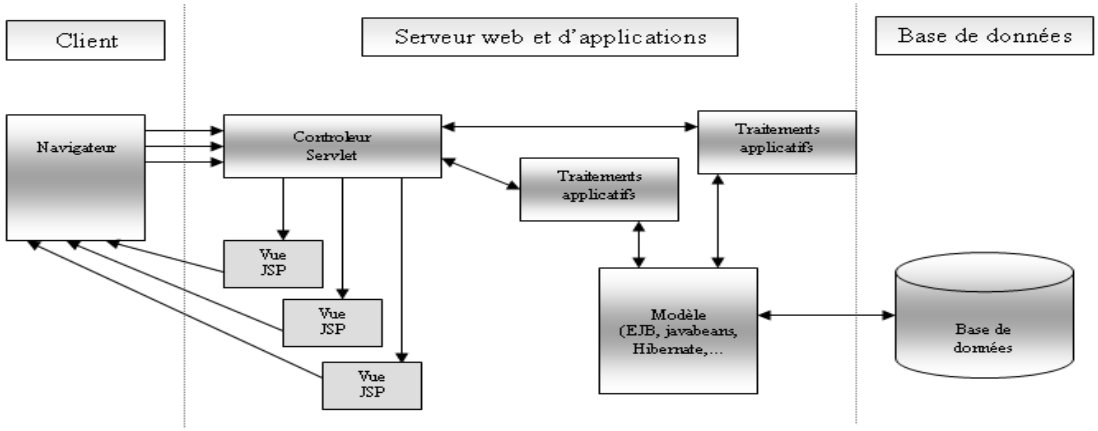

**Figure(III.2) : modèle MVC de type1.**

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

### III.3.2. Le modèle MVC de type2 :

Le principal défaut du modèle MVC est le nombre de servlets à développer pour une application. Pour simplifier les choses, le modèle MVC model 2 ou MVC2 de Sun propose de n'utiliser qu'une seule et unique servlet comme contrôleur. Cette servlet se charge d'assurer le workflow des traitements en fonction des requêtes http reçues.

Le modèle MVC 2 est donc une évolution du modèle 1 : une unique servlet fait office de FFF la forme d'un fichier au format XML.




**Figure(III.3) : le modèle MVC de type2.**

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

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

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

1. Le client envoie une requête à l'application, requête est prise en charge par la servlet faisant office de contrôleur.
2. La servlet analyse la requête et appelle la classe dédiée contenant les traitements

3. Cette classe exécute les traitements nécessaires en fonction de la requête, notamment, en faisant appel aux objets métiers.
4. En fonction du résultat, la servlet redirige la requête vers la page JSP
5. La JSP génère la réponse qui est renvoyée au client.

## III.4. Framework :

### III.4.1. Définition : [7]

Le framework désigne le cadre dans lequel va s'insérer une application. En programmation orientée objet, il désigne l'infrastructure logicielle qui facilite la conception des applications par l'utilisation de bibliothèques de classes ou de générateurs de programmes.

C'est une bibliothèque de classes fournissant une ossature générale pour le développement d'une application dans un domaine particulier. Ces composants sont organisés afin d'être utilisés en interaction les uns avec les autres et sont spécifiques généralement à un type d'application.

Les frameworks facilitent ainsi le travail du développement en fournissant un squelette d'application qu'il suffit de remplir pour l'adapter à ses besoins. La contrepartie est qu'un framework représente un sur ensemble de tous les besoins génériques, ce qui conduit à supporter un grand nombre de choses même si souvent, une toute petite partie est utile pour le cas à réaliser. Ceci complique également la tâche d'apprentissage et d'assimilation de l'environnement par le développeur.

Un framework est un ensemble de classes abstraites qui, dans le but de faciliter la création d'une partie d'un système logiciel, collaborent entre elles. Il partage le domaine visé en classes abstraites et permet de définir les responsabilités de chacune par rapport aux autres et les rapports entre elles.

### III.4.2.Objectif d'un framework :

L'objectif premier d'un framework est d'améliorer la productivité des développeurs qui l'utilisent. Plutôt sympa, non ? Souvent organisé en différents composants, un framework offre la possibilité au développeur final d'utiliser tel ou tel composant pour lui faciliter le développement, et lui permet ainsi de se concentrer sur le plus important. Prenons un exemple concret. Il existe dans Symfony2 un composant qui gère les formulaires HTML : leur affichage, leur validation, etc.

Le développeur qui l'utilise se concentre sur l'essentiel dans son application : chaque formulaire effectue une action, et c'est cette action qui est importante, pas les formulaires. Étendez ce principe à toute une application ou tout un site internet, et vous comprenez l'intérêt d'un framework ! Autrement dit, le framework s'occupe de la forme et permet au développeur de se concentrer sur le fond.

### III.4.3. Différents types de framework :

**III.4.3.1. Framework d'infrastructure système :** pour développer des systèmes d'exploitation, des interfaces graphiques, des outils de communication. (Exemple : Framework .Net, Eclipse, NetBeans, Struts).

**III.4.3.2. Framework d'intégration intergicielle :** pour fédérer des applications hétérogènes. Pour mettre à dispositions différentes technologies sous la forme d'une interface unique. (Exemple : Ampoliros avec ses interfaces RPC, SOAP, XML).

**III.4.3.3. Framework d'entreprise :** pour développer des applications spécifiques au secteur d'activité de l'entreprise.

**III.4.3.4. Framework orienté Système de gestion de contenu :** Les principaux avantages de ces frameworks sont la réutilisation de leur code, la standardisation du cycle de vie du logiciel (Spécification, développement, maintenance, évolution), ils permettent de formaliser une architecture adaptée au besoin de l'entreprise. Ils tirent parti de l'expérience des développements antérieurs.

Ces frameworks sont en quelque sorte des progiciels extrêmement souples et évolutifs.

### III.4.4. Composants d'un framework :

#### III.4.4.1. Bibliothèque :

Un *Framework* peut être ou peut contenir une ou des bibliothèques. Une bibliothèque, en informatique, est un ensemble d'algorithmes qui peut être répétitif. Donc les bibliothèques sont utilisées pour que nous n'ayons pas à les réécrire à chaque nouveau projet ou plusieurs fois dans le même projet. Une bibliothèque peut aussi être un ensemble d'algorithmes offert par des développeurs pour d'autres développeurs pour leur simplifier la vie. Par exemple, la bibliothèque *Math* qui contient une grande quantité de formules mathématiques.

#### III.4.4.2. Structure :

En plus de contenir ou non des bibliothèques, un Framework peut avoir une structure prédéfinie pour concevoir des programmes. Par exemple, la fenêtre de l'application et ses composantes déjà conçues et pouvant être modifiées selon les goûts et les besoins des programmeurs. La structure peut aussi être une architecture du code permettant une utilisation facile de conception complexe.

#### III.4.4.3. Autres :

Les *Framework* peuvent avoir d'autres éléments intégrés pour faciliter les programmeurs selon le but ou le domaine de celui-ci. Le *Framework* XNA de Microsoft offre un Content Pipeline qui est un conteneur d'éléments extérieurs au programme, comme des images, des vidéos, des fichiers audio, des styles de textes, des effets, etc.

### III.4.5. Avantages et inconvénients d'un framework :

#### III.4.5.1. Avantages :

L'avantage premier est le gain en productivité. Mais il en existe bien d'autres. On peut les classer en plusieurs catégories : le code, le travail et la communauté.

Tout d'abord, un framework aide à réaliser un « bon code ». Par « bon code », on entend qu'il incite, de par sa propre architecture, à bien organiser le code. Et un code bien organisé est un code facilement maintenable et évolutif. De plus, un framework offre des briques prêtes à être utilisées (le composant *Formulaire* de Symfony2 par exemple), ce qui évite de réinventer la roue, et surtout qui permet d'utiliser des briques puissantes et éprouvées. En effet, ces briques sont développées par des équipes de développeurs chevronnés, elles sont donc très flexibles et très robustes. On économise ainsi des heures de développement.

Ensuite, un framework améliore la façon dont on travaille. En effet, dans le cas d'un site internet, on travaille souvent avec d'autres développeurs PHP et un designer. Un framework v aide doublement dans ce travail en équipe. D'une part, un framework utilise presque toujours l'architecture MVC, c'est donc une façon d'organiser son code qui sépare le code PHP du code HTML. Ainsi, le designer peut travailler sur des fichiers différents de ceux du développeur, fini les problèmes d'édition simultanée d'un même fichier. D'autre part, un

framework a une structure et des conventions de code connues. Ainsi, on peut facilement travailler avec un autre développeur : s'il connaît déjà le framework en question, il s'intégrera très rapidement au projet.

Enfin, le dernier avantage est la communauté soutenant chaque framework. C'est elle qui fournit les tutoriaux ou les cours (de l'aide sur les forums, et bien sûr les mises à jour du framework) Ces mises à jour sont très importantes

Un framework, c'est aussi :

- ✓ Une communauté active qui utilise le framework et qui contribue en retour ;
- ✓ Une documentation de qualité et régulièrement mise à jour ;
- ✓ Un code source maintenu par des développeurs attitrés ;
- ✓ Un code qui respecte les standards de programmation ;
- ✓ Un support à long terme garanti et des mises à jour qui ne cassent pas la compatibilité ;…

**III.4.5.2. Inconvénients :**

La courbe d'apprentissage qui est plus élevée. En effet, pour maîtriser un framework, il faut un temps d'apprentissage non négligeable. Chaque brique qui compose un framework a sa complexité propre qu'il vous faudra appréhender.

Notez également que pour les frameworks les plus récents, tels que Symfony2 justement, il faut également être au courant des dernières nouveautés de PHP. Je pense notamment à la programmation orientée objet et aux namespaces. De plus, connaître certaines bonnes pratiques telles que l'architecture MVC est un plus.

### III.4.6. Framework Hibernate:

#### III.4.6.1. Définition Hibernate:

Hibernate est une solution open source de type ORM (Object Relational Mapping) qui permet de faciliter le développement de la couche persistance d'une application. Hibernate permet donc de représenter une base de données en objets Java et vice versa.

Hibernate facilite la persistence et la recherche de données dans une base de données en réalisant lui-même la création des objets et les traitements de remplissage de ceux-ci en accédant à la base de données. La quantité de code ainsi épargnée est très importante d'autant que ce code est généralement fastidieux et redondant.

Hibernate est très populaire notamment à cause de ses bonnes performances et de son ouverture à de nombreuses bases de données.

Les bases de données supportées sont les principales du marché : DB2, Oracle, MySQL, PostgreSQL, Sybase, SQL Server, Sap DB, Interbase, ...

#### III.4.6.2. Architecture d'Hibernate :

Hibernate a besoin de plusieurs éléments pour fonctionner :

- ✓ une classe de type javabean qui encapsule les données d'une occurrence d'une table
- ✓ un fichier de configuration qui assure la correspondance entre la classe et la table (mapping)
- ✓ des propriétés de configuration notamment des informations concernant la connexion à la base de données

Une fois ces éléments correctement définis, il est possible d'utiliser Hibernate dans le code des traitements à réaliser. L'architecture d'Hibernate est donc la suivante :

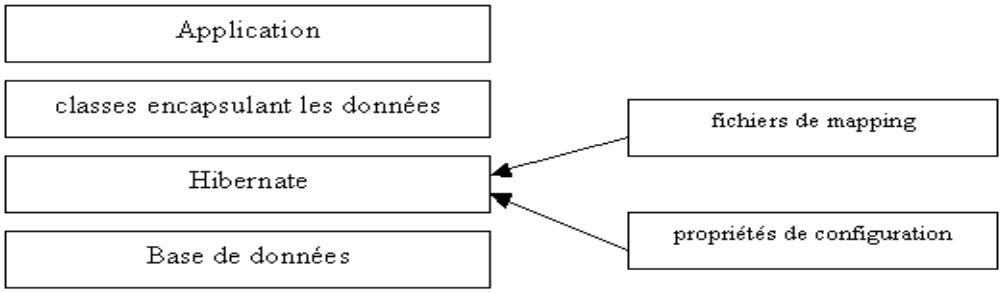



**Figure (III.4): architecture d'Hibernate.**

### III.4.6.3. Avantages et inconvénients d'Hibernete:

#### III.4.6.3.1. Les avantages :

- ✓ Hibernate génère le code SQL nécessaire, ce qui rend l'application plus portable (s'adapte à la base de données).
- ✓ La persistance est transparente. Vous pouvez faire de vos classes métiers des classes persistantes sans ajout de code.
- ✓ La récupération de données est optimisée. On peut interroger la base de données de plusieurs façons (requête SQL, langage HQL…)
- ✓ Portabilité du code en cas de changement de base de données.
- ✓ Propose une API performante et robuste
- ✓ Est mature et dispose d'un support fiable et d'une documentation abondante
- ✓ Supporte la gestion des transactions
- ✓ S'intègre facilement à Eclipse et Spring.
- ✓ Appropriation rapide.

#### III.4.6.3.2. Les inconvénients :

- ✓ Il est dure de faire des requête complexe avec HQL (ex : sous requêtes).
- ✓ Etant une technologie jeune, il reste des problèmes à résoudre (ex : les fichiers
- ✓ de mapping ne sont pas toujours bien formés)
- ✓ Ne se base pas sur les standards
- ✓ Ne se base pas sur les spécifications et standards JDO ou ODMG
- ✓ Nécessite un module pour la connectivité aux bases Oracle
- ✓ Possibilité de perte de vitesse après l'acceptation des spécifications JDO 2.0.

### III.4.6.4. comparaison entre Hibernate et deux autres frameworks:

Nous avons choisis de comparer Hibernet avec deux parmi les framework les plus utilisés Spring et Struts. Nous allons donc en premier lieu définit chacun des deux framework (Spring et Struts), décrit leurs architectures et nous allons citer également les avantages et les inconvénients de chacun d'eux. Pour finir avec un tableau comparatif où nous allons faire une comparaison entre Hibernet et ses deux framework selon certains critères tel que la date de création de chacun, la maturité, l'interprétation java…

Ce qui nous permis de bien choisir le framework adéquat pour réaliser notre application.

#### III.4.6.4.1. Le Framework Spring :

##### III.4.6.4.1.1. Définition :

Spring est un socle pour le développement d'applications, principalement d'entreprises mais pas obligatoirement. Il fournit de nombreuses fonctionnalités parfois redondantes ou qui peuvent être configurées ou utilisées de plusieurs manières. C'est ainsi un des frameworks les plus répandus dans le monde Java : sa popularité a grandie au profit de la complexité de Java EE notamment pour ses versions antérieures à la version 5 mais aussi grâce à la qualité et la richesse des fonctionnalités qu'il propose :

- ✓ son coeur reposant sur un conteneur de type IoC assure la gestion du cycle de vies des beans et l'injection des dépendances
- ✓ l'utilisation de l'AOP
- ✓ des projets pour faciliter l'intégration avec de nombreux projets open source ou API de Java EE

Spring était un framework applicatif à ses débuts mais maintenant c'est une véritable plate-forme composée du framework Spring, de projets qui couvrent de nombreux besoins et de middlewares. Il permet une grande flexibilité dans les fonctionnalités et les projets utilisés dans une application. Il est par exemple possible d'utiliser le conteneur Spring pour gérer de façon basique les beans sans utiliser l'AOP. Par contre, certains projets et certaines fonctionnalités ont des dépendances avec d'autres projets. Spring est associé à la notion de

conteneur léger (lightweight container) par opposition aux conteneurs lourds que sont les serveurs d'applications Java EE.

#### III.4.6.4.1.2. Architecture de Spring :

Spring Framework contient toutes les fonctionnalités de base pour développer des applications.

Le cœur de Spring Framework 3.0 est composé d'un ensemble d'une vingtaine de modules qui sont regroupés en plusieurs parties :

- ✓ Spring Core Container : regroupe les modules de base pour mettre en œuvre le conteneur
- ✓ AOP and Instrumentation : permet de mettre en œuvre l'AOP
- ✓ Data Accès/Intégrations : regroupe les modules d'accès aux données
- ✓ Web : regroupe les modules pour le développement d'applications web
- ✓ Test : propose des fonctionnalités pour les tests automatisés avec Spring

La partie Spring Core Container contient plusieurs modules :

- ✓ Spring Core et Spring Beans : contiennent les fonctionnalités de base notamment le conteneur et des utilitaires
- ✓ Spring Context : propose un support de la définition du context Spring (sa configuration) mais aussi des fonctionnalités de base comme le mail, l'internationalisation, JNDI, ...
- ✓ Spring Expression Langage (SpEL) : propose un langage d'expressions pour interroger et manipuler les objets gérés par le conteneur

La partie AOP and Instrumentation contient plusieurs parties :

- ✓ Spring AOP : propose un support de l'AOP
- ✓ AspectJ : propose une intégration d'AspectJ
- ✓ Instrumentation : propose une instrumentation des classes et plusieurs implémentations de classloaders utilisés par certains serveurs d'applications.

La partie Data Acces/Integration contient plusieurs modules

- ✓ Spring JDBC : propose une abstraction de l'utilisation de JDBC avec notamment une hiérarchie d'exceptions dédiées
- ✓ Spring ORM : propose un support pour des outils de type ORM (JPA, JDO, Hibernate, iBatis)
- ✓ Spring Transaction : propose un support déclaratif et par programmation de la gestion des transactions
- ✓ Spring OXM : propose une abstraction pour le mapping objet/XML avec un support de JAXB, Castor, XMLBeans, JiBX et XStream
- ✓ Spring JMS : propose des fonctionnalités pour faciliter la mise en oeuvre de JMS avec Spring

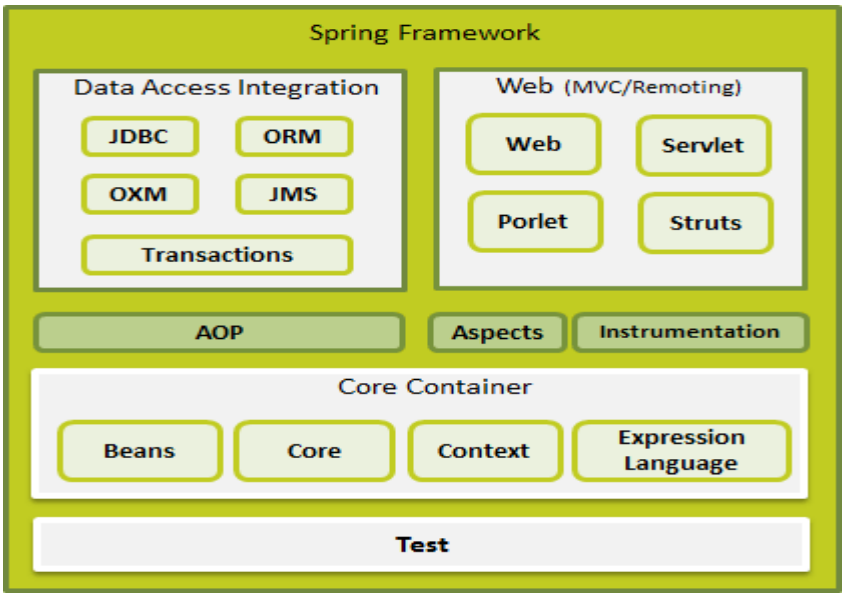
La partie Web contient plusieurs modules :

- ✓ Spring Web : propose des fonctionnalités de base pour les développements web (initialisation du conteneur, gestion des contextes, support multipart, extraction des paramètres d'une requête http, ...)
- ✓ Spring Web-Servlet : framework pour le développement d'applications qui met en oeuvre le motif de conception MVC. Ceci permet entre autres de choisir la technologie utilisée pour la vue (JSP, Velocity, Tiles, iText, ...)
- ✓ Spring Web-Struts : propose un support de Struts
- ✓ Spring Web-Portlet : propose un support pour les portlets

La partie Test contient un seul module :

- ✓ Spring Test : propose un support pour les tests automatisés avec un support de JUnit et TestNG

Ces modules sont utilisés comme base pour le développement d'applications.




**Figure(III.5) : architecture de Spring.**

### III.4.6.4.1.3. Avantages et inconvénients de Spring :

Spring est un framework open source majoritairement développé par SpringSource mais il n'est pas standardisé par le JCP. Il est très largement utilisé dans le monde Java, ce qui en fait un standard de facto et constitue une certaine garantie sur la pérennité du framework.

Spring propose une très bonne intégration avec des frameworks open source (Struts, Hibernate, ...) ou des standards de Java (Servlets, JMS, JDO, ...). Toutes les fonctionnalités de Spring peuvent s'utiliser dans un serveur Java EE et pour la plupart dans un simple conteneur web ou une application standalone. Les fonctionnalités offertes par Spring sont très nombreuses et les sujets couverts ne cessent d'augmenter au fur et mesure des nouvelles versions et des nouveaux projets ajoutés au portfolio.

La mise en oeuvre de Spring n'est pas toujours aisée car il existe généralement plusieurs solutions pour mettre en oeuvre une fonctionnalité : par exemple, généralement avec Spring 3.0, une fonctionnalité est utilisable par configuration XML, par annotations ou par API. Bien sûr cela permet de choisir mais cela impose un arbitrage selon ses besoins. Il n'est pas rare que les livrables aient une taille importante du fait des nombreuses librairies requises par Spring et ses dépendances.

### III.4.6.4.2. Le Framework Struts:

#### III.4.6.4.2.1. Définition:

Struts est un framework pour applications web développé par le projet Jakarta de la fondation Apache. C'est le plus populaire des frameworks pour le développement d'applications web avec Java. Il a été initialement développé par Craig Mc Clanahan qui l'a donné au projet Jakarta d'Apache en mai 2000. Depuis, Struts a connu un succès grandissant auprès de la communauté du libre et des développeurs à tel point qu'il sert de base à de nombreux autres framework open source et commerciaux et que la plupart des grands IDE propriétaires (Borland, IBM, BEA, ...) intègrent une partie dédiée à son utilisation. Il met en oeuvre le modèle MVC 2 basé sur une seule servlet faisant office de contrôleur et des JSP pour l'IHM. L'application de ce modèle permet une séparation en trois parties distinctes de l'interface, des traitements et des données de l'application.

Struts se concentre sur la vue et le contrôleur. L'implémentation du modèle est laissée libre aux développeurs : ils ont le choix d'utiliser des JavaBeans, un outil de mapping objet/relationnel, des EJB ou toute autre solution.

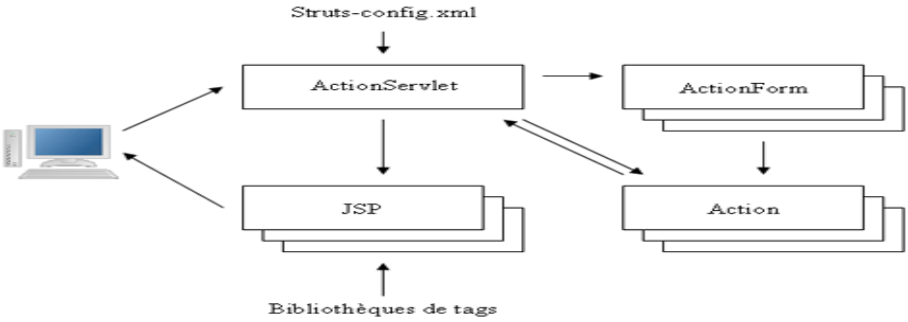
Pour le contrôleur, Struts propose une unique servlet par application qui lit la configuration de l'application dans un fichier au format XML. Cette servlet de type ActionServlet reçoit toutes les requêtes de l'utilisateur concernant l'application. En fonction du paramétrage, elle instancie un objet de type Action qui contient les traitements et renvoie une valeur particulière à la servlet. Celle-ci permet de déterminer la JSP qui affichera le résultat des traitements à l'utilisateur.

#### III.4.6.4.2.2. Architecture de Struts :

Dans cette section, nous allons vous expliquer l'architecture de Struts. Struts est connu par son architecture robuste et est utilisé pour développer des projets de petites et grandes tailles. Struts est un projet opensource utilisé pour développer des applications web JEE en utilisant le modèle d'architecture MVC. Struts utilise l'API JEE qu'il étend pour permettre aux développeurs d'adopter une architecture MVC. Le framework Struts comporte trois composantes :

- ✓ Le gestionnaire de requêtes fourni par le développeur de l'application qui permet de lier un comportement à une URL.

- ✓ Le gestionnaire de réponses qui permet de transférer le contrôle vers une autre ressource qui va s'occuper du rendu de la réponse (vers le client)
- ✓ Une librairie de « tags » permettant au développeur de créer des formulaires interactifs avec des pages web. Struts fournit toute l'architecture pour implémenter un MVC dans vos applications et ainsi permettre aux développeurs de se concentrer sur la partie « métier ».




**Figure(III.6) : architecture de struts.**

Les données issues de la requête sont encapsulées dans un objet de type ActionForm. Struts va utiliser l'introspection pour initialiser les champs de cet objet à partir des valeurs fournies dans la requête. Struts utilise un fichier de configuration au format XML (struts-config.xml) pour connaître le détail des éléments qu'il va gérer dans l'application et comment ils vont interagir lors des traitements. Pour la vue, Struts utilise par défaut des JSP avec un ensemble de plusieurs bibliothèques de tags personnalisés pour faciliter leur développement.

#### III.4.6.4.2.3. Avantages et inconvénients de Struts :

**a) Les Avantages :**

- ✓ Est quasiment un standard de fait
- ✓ Nombreuses sources d'informations
- ✓ Bibliothèques de tags HTML
- ✓ Intégration dans les IDE
- ✓ open source

**b) Les Inconvénients :**

- ✓ Son avenir
- ✓ Peu d'évolutions
- ✓ une architecture vieillissante
- ✓ beaucoup de classes et de code à produire (partiellement automatisable avec des outils dédiés).

Voici le tableau comparatif :

| | Date de création | maturité | Open source | mvc | Composant ou requête | servelet | Interprétation direct du java | Template | compilateur java vers java script | Facilité de prise en main/ facilité |
|---|---|---|---|---|---|---|---|---|---|---|
| hibernate | | oui | oui | mvc | | non | | Page Xhtml | | |
| spring | 2000 | oui | oui | Mvc2 | requête | Utilisé/ servelet centrale/compétence nécessaires | non | Page jsp/velocity,freemaker | non | Non/technologie des servelet à maitriser |
| struts | | oui | oui | Mvc2 | requête | Utilisé/servelet centrale/compétence nécessaires | non | Page jsp | non | Non/technologie des servelet à maitriser |

**Tab(III.1) : La comparaison entre Hibernate et deux autres framework.**

Apres cette comparaison on a choisir d'utiliser Hibernate dans notre application.

### III.5. Conclusion :

Au cours de ce chapitre, nous avons abordé les notions de base de l'architecture J2EE et le Model MVC. Ensuite on a parlé sur l'objectif d'un Framework et ces différents exemples ou on a choisir Hibernate avec une justification de notre choix.

Ces notions seront utilisées pour le développement de notre application

## IV.1. Introduction :

L'étude de l'existant nous a permet de collecter les informations qui concernent le fonctionnement du système existant, et ainsi de connaitre les besoins et les orientations des utilisateurs.

Dans ce chapitre, nous allons entamés le processus de développement de notre application. Pour se faire, nous appuyons sur le **langage UML** qu'est un langage de modélisation graphique mettant en œuvre les principes de la modélisation objet.

Pour cela nous allons optés la démarche suivante :

- ✓ A partir de la définition des besoins nous allons identifiés les différents acteurs et les tâches dans lesquelles nous déduirons les cas d'utilisation.
- ✓ Ceux-ci nous permettent d'établir un ensemble de scénario d'utilisation à l'aide des diagrammes de séquence et les activités pour modéliser les activités d'une opération spécifique.
- ✓ Ces diagrammes nous aiderons à identifier les différentes classes, cela pour but d'aboutir au diagramme de classe.

## IV.2. Présentation d'UML (Unified Modeling Language) :

### IV.2.1. Définition :

UML est un langage ou formalisme de modélisation orienté objet qui représente un moyen de spécifier et représenter les composantes d'un système informatique. Il permet notamment la spécification de toutes les décisions importantes en matière d'analyse, de conception et d'implémentation, décisions qui doivent être prises lors du développement d'un système à forte composante logiciel.

### IV.2.2. Les diagrammes UML : [8]

Le Langage UML est d'écrit par un méta-modèle, c'est-à dire une représentation simple, mais rigoureuse, de ses constituants élémentaires, avec leurs règles d'utilisation et de syntaxe. Un diagramme est défini par les acteurs d'UML comme la représentation graphique d'un ensemble sélectionné de constituants UML.

Ils ont élaboré treize diagrammes, chacun d'eux étant dédié à la représentation des concepts particuliers d'un système logiciel.

Une classification courante distingue les diagrammes qui traduisent la structure et les diagrammes qui présentent le comportement du système étudié.

Parmi ces diagrammes, on a choisit les suivants qui sont suffisant pour étudier et analyser la situation existante :

- ✓ Le diagramme de cas d'utilisation.

- ✓ Le diagramme de classe.
- ✓ Le diagramme de séquence.
- ✓ Le diagramme d'activité.

## IV.3. Phase d'analyse :

Cette phase s'intéresse à la définition des besoins ainsi qu'au domaine couvert par l'application et cela pour objectif de spécifier de manière claire et détaillé l'application de gestion de la scolarité. Pour se faire, il est nécessaire de déterminer globalement ce qui se trouve dans le champ de l'application.

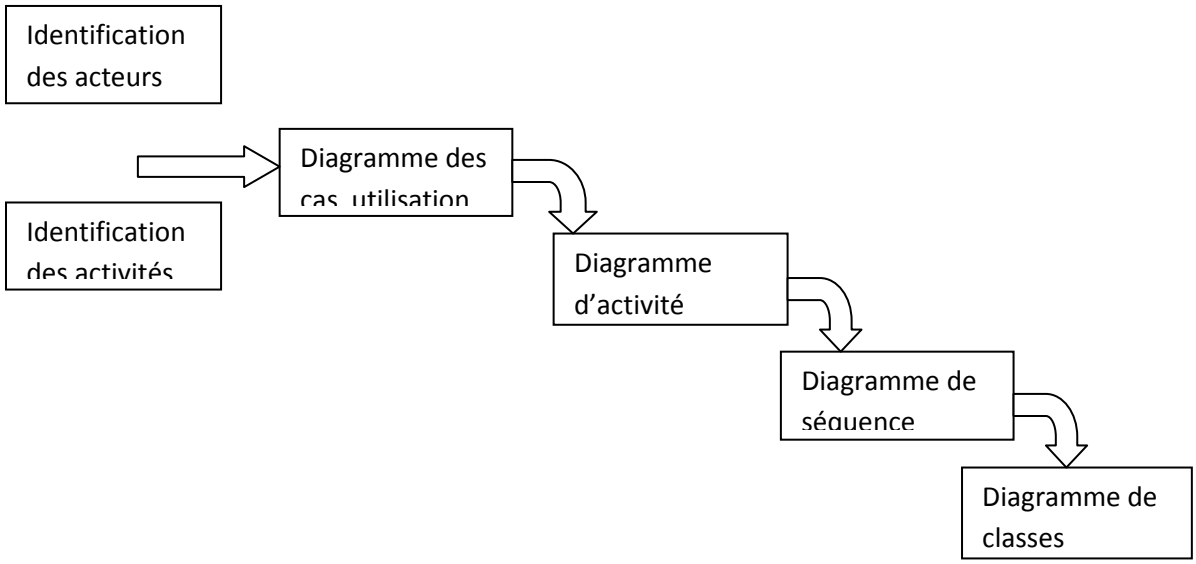
### IV.3.1. Identification des besoins :

Au cours de notre passage par différents services de l'école, 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…

Suite à ces problèmes, l'objectif principal de notre site est d'assurer les services suivants :

- ✓ Faciliter la gestion du nombre considérable du personnel de l'école.
- ✓ Eliminer les erreurs dues à la gestion aléatoire et manuelle des heures d'enseignements pour chaque formation.
- ✓ Assurer une bonne disposition des statistiques.

Dans le but d'une meilleure organisation et une bonne maitrise du travail, tout processus de développement d'application ou systèmes informatiques doit suivre une méthode ou une démarche bien définie. La figure ci-dessous montre la représentation graphique de la démarche de modélisation choisie pour concevoir notre application :




**Figure(IV.1) : la démarche de modélisation de l'application.**

### IV.3.2. Identification des acteurs de l'application :

Le stage que nous avons fait nous a permis de procéder à l'identification de ces principaux acteurs qui seront les futurs utilisateurs de notre application :

1. **Les visiteurs :** c'est le grand public qui visite le site pour voir les différentes formations disponibles…
2. **L'enseignant** : c'est un rôle qui peut être joué par tous les enseignants de l'école.
3. **L'étudiant :** c'est un rôle qui peut être joué par tous les étudiants de l'école.
4. **L'agent de la scolarité** : c'est un rôle qui peut être joué par tous les personnes chargée des services de scolarité.
5. **Administrateur (directeur)** : un rôle qui peut être joué par la personne qui est chargée de l'administration.

### IV.3.3. Identification des cas d'utilisation :

- ✓ **Définition d'un cas d'utilisation : [9]**

Un cas d'utilisation « use case » représente un ensemble d'actions réalisées par le système en réponse à une action d'un acteur.

Le tableau suivant récapitule les cas d'utilisation de chaque acteur de notre application :

| Acteur | Les taches |
|---|---|
| étudiant | T1 : se connecter au site.<br>T2 :s'authentifier.<br>T3 : consulter les formations disponibles.<br>T4 : consulter les cours.<br>T5 : changer son mot de passe. |
| Enseignant | T1,<br>T6 :s'authentifier.<br>T7 : gérer des cours (ajouter, modifier).<br>T8 : Consulter l'emploi de temps.<br>T9 : Consulter la liste des étudiants.<br>T10 : changer son mot de passe. |

| | |
|---|---|
| Agent de scolarité | T1,<br>T11 :s'authentifier.<br>T12 : éditer les listes des étudiants.<br>T13 : introduction des emplois de temps (heure, groupe, salle).<br>T14 : établir les attestations des étudiants.<br>T15 : gérer les étudiants (ajouter, modifier).<br>T16 : changer son mot de passe. |
| Administrateur<br>(directeur) | T1,<br>T17 :s'authentifier.<br>T18 : gérer les formations (ajouter, modifier).<br>T19 : gérer les enseignants (ajouter, modifier).<br>T 20 : consulter les listes des (formations, enseignants).<br>T21 : changer son mot de passe. |
| visiteur | T22 : voir les différentes formations disponibles.<br>T23 : voir des informations a propos de l'école. |

**Tableau(IV.1) : Spécification des tâches des acteurs.**

### IV.3.4. Diagramme de contexte : [9]

Le diagramme de contexte est un modèle conceptuel qui permet d'avoir une vision globale des interactions entre le système et les liens avec l'environnement extérieur. Il permet aussi de délimiter le champ d'étude.

Dans notre cas le diagramme de contexte est donné par la figure suivante :

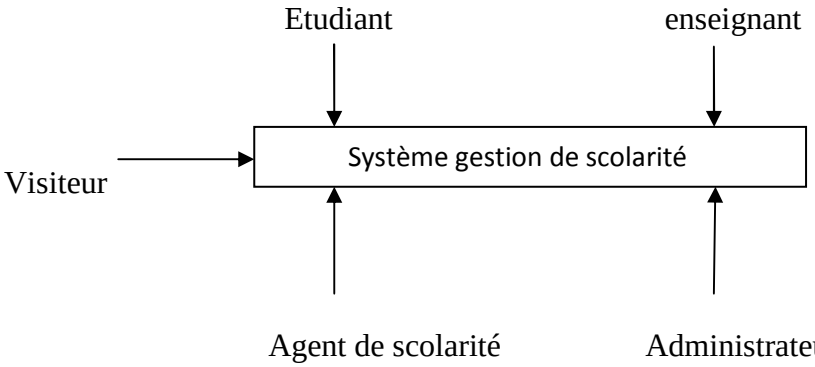



**Figure (IV.2) : diagramme de contexte de l'application.**

### IV.3.5. spécification des scénarios :

**Définition d'un scénario:** un scénario représente une succession particulière d'enchainement qui s'exécute du début a la fin du cas d'utilisation, un enchainement étant l'unité de description de séquences d'actions.

Chacun des acteurs qu'on a définis effectue des taches, chaque tache est décrite par des scenarios qu'on résume comme suit :

| acteur | Les taches | Scénarios |
|---|---|---|
| étudiant | T1 : se connecter au site. | S0 : saisir l'URL de site |
| | T2 :s'authentifier. | S0,<br>S1 : cliquer sur le lien étudiant.<br>S2 : remplir le formulaire et valider. |
| | T3 : consulter les formations disponibles. | S0, S1, S2<br>S3 : cliquer sur le lien <<formation disponible>> et consulter. |
| | T4: consulter les cours. | S0, S1, S2<br>S4 : cliquer sur le lien << cours>>pour choisir un. |
| | T5 : changer son mot de passe. | S0, S1, S2<br>S5 : cliquer sur le lien<< changement de mot de passe >><br>S6 : remplir le formulaire puis valider. |
| enseignant | T1,<br>T6 :s'authentifier. | S0<br>S7 : cliquer sur le lien enseignant.<br>S8 : remplir le formulaire puis valider. |
| | T7 : gérer les cours. | S0, S7, S8<br>S9 : cliquer sur <<gérer>> cour.<br>S10 : remplir le formulaire puis valider. |
| | T8 : Consulter l'emploi de temps. | S0, S7, S8<br>S11 : cliquer sur le lien << emploi de temps>> et consulter. |
| | T9 : Consulter la liste des étudiants. | S0, S7, S8<br>S12 : cliquer sur le lien << listes des étudiants>> et consulter. |
| | T10 : changer son mot de passe. | S0, S7, S8<br>S13 : cliquer sur le lien <<changement de mot de passe>> |

| | | |
|---|---|---|
| | | S14 : remplir le formulaire puis valider |
| Agent de scolarité | T1,<br>T11 :s'authentifier. | S0<br>S15 : cliquer sur le lien agent de scolarité.<br>S16 : remplir le formulaire puis valider. |
| | T12 : éditer les listes des étudiantes. | S0, S15, S16<br>S17 : cliquer sur le lien << liste étudiant>>.<br>S18 : remplir le formulaire puis valider. |
| | T13 : introduction des emplois de temps. | S0, S15, S16<br>S19: cliquer sur le lien << emploi de temps >>.<br>S20: remplir le formulaire puis valider. |
| | T14 : établir les attestations des étudiants. | S0, S15, S16, S17<br>S21 : cliquer sur le lien <<attestation>>.<br>S22 : remplir le formulaire puis valider. |
| | T15 : gérer les étudiants (ajouter, modifier) | S0, S13, S14, S17<br>S23 : cliquer sur le lien<<gérer étudiant>>.<br>S24 : cliquer sur le lien (ajouter ou modifier) étudiant.<br>S25 : remplir le formulaire puis valider. |
| | T16 : changer son mot de passe. | S0, S13, S14,<br>S26 : cliquer sur le lien <<changement de mot de passe>><br>S27 : remplir le formulaire puis valider |
| Administrateur (directeur) | T1,<br>T17 :s'authentifier. | S0<br>S28 : cliquer sur le lien administrateur.<br>S29 : remplir le formulaire puis valider. |
| | T18 : gérer les formations. | S0, S28, S29<br>S30 : cliquer sur le lien <<formation >>.<br>S31 : cliquer sur le lien <<ajouter ou modifier >> formation.<br>S32 : remplir le formulaire puis valider. |
| | T19 : gérer les enseignants. | S0, S28, S29<br>S33 : cliquer sur le lien <<enseignant >>.<br>S34 : cliquer sur le lien<<modifier ou ajouter>> enseignant. |

| | | |
|---|---|---|
| | | S35 : remplir le formulaire puis valider. |
| | T 20 : consulter les listes des (formations, enseignants). | S0, S28, S29<br>S36 : cliquer sur le lien<<enseignant ou formation>> pour consulter. |
| | T21 : changer son mot de passe. | S0, S28, S29<br>S37 : cliquer sur le lien <<changement de mot de passe>><br>S38 : remplir le formulaire puis valider |
| Visiteur | T1,<br>T22 : voir les différentes formations disponibles. | S0,<br>S39 : cliquer sur le lien <<visiteur>>.<br>S40 : cliquer sur le lien <<formation disponible>>pour voir les différentes formations disponibles. |
| | T23 : voir des informations a propos de l'institut. | S0, S39<br>S41 : cliquer sur le lien<< à propos de l'institut>> et consulter.<br>. |

**Tableau(IV.2) : Spécification des scenarios des acteurs.**

### IV.3.6. spécification de quelques cas d'utilisations :

**Administrateur :**

- ✓ Cas d'utilisation :<<authentifier>> :

Use case : Authentification

Role : Administrateur

Scenarios : S0, S28, S29

Description :

1:l'administrateur clique sur le lien <<administrateur>>

2 : le système affiche le formulaire d'identification

3 :l'administrateur remplit le formulaire et valider

4 : le système affiche le message d'erreur si le formulaire est mal remplit sinon

5 : le système affiche l'espace administrateur.

**Figure (IV.3) : Cas d'utilisation :<<authentifier>>(Administrateur).**

**Etudiant :**

- ✓ Cas d'utilisation :<< consulter les formations disponible>> :

Use case : consulter et imprimer toutes les formations disponibles
Role: étudiant.
Scenarios : S0, S1, S2.
Description :
1: l'étudiant clique sur le lien <<formation disponible>>
2 : le système affiche toutes les formations disponibles Dans l'institut.
3 :l'étudiant consulte et imprime la liste s'il veut avec le Clique sur imprimer.
4 : le système imprime la liste.

**Figure (IV.5) : Cas d'utilisation :<< consulter les formations disponible>> (etudiant).**

**Enseignant :**

- ✓ Cas d'utilisation :<<changer son mot de passe>> :

Use case : changer son mot de passe
Role: Enseignant.
Scenarios : S0, S7, S8
Description :
1 : l'enseignant clique sur le lien <<changement de mot de passe>>.
2 : le système affiche le formulaire de modification.
3 : l'enseignant remplit le formulaire puis valider.
4 : le système affiche un message d'erreur s'il ya une Erreur sinon
5 : le système affiche le message de succès.

**Figure (IV.6) : Cas d'utilisation :<<changer son mot de passe>> (enseignant).**

**Agent de scolarité :**

- ✓ Cas d'utilisation :<< : éditer les listes des étudiantes >> :

| |
|---|
| Use case établir les listes des étudiants<br>Role: Agent de scolarité.<br>Scenarios : S0, S15, S16.<br>Description :<br>1 :l'agent clique sur le lien <<liste des étudiants>><br>2 : le système affiche une liste à choisir.<br>3 : l'agent clique sur la formation désiré.<br>4 : le système affiche un formulaire à remplir avec la liste des Etudiant qui convient cette formation puis valider<br>5 : le système affiche la liste demandé. |

**Figure (IV.7) : Cas d'utilisation :<< établissement de la liste globale des étudiantes >> (agent de scolarité).**

### IV.3.7. Diagrammes des cas d'utilisations :

Le diagramme de cas d'utilisation est un diagramme UML utilisé pour donner une vision globale du comportement fonctionnel d'un système logiciel.

- ✓ **Etudiant :**

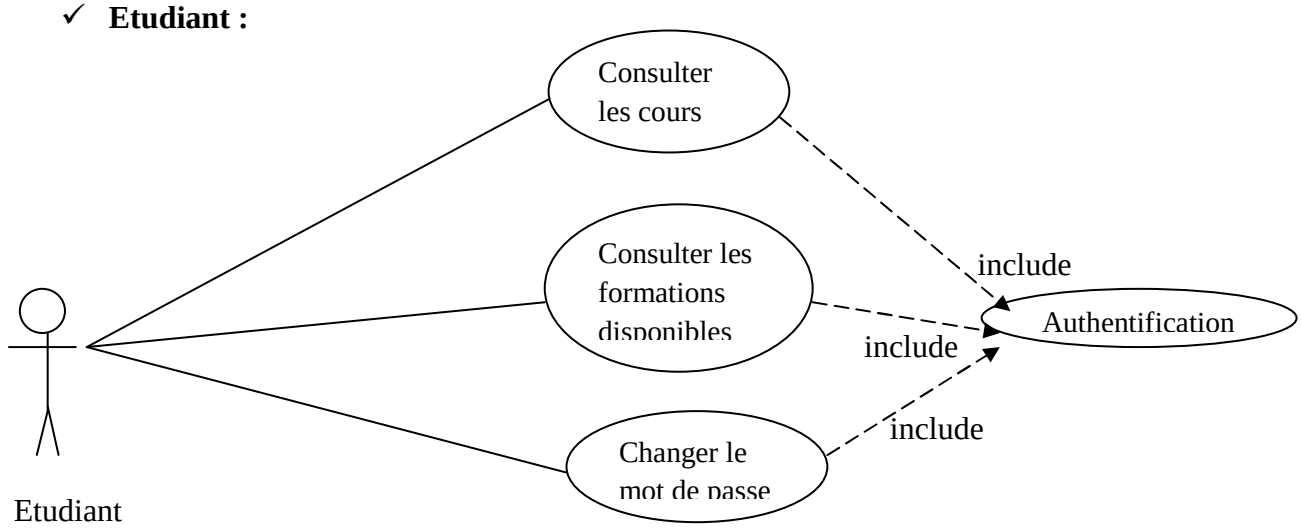



**Figure (IV.8) : Diagramme global de cas d'utilisation relatif à l'étudiant.**

✓ **Enseignant :**

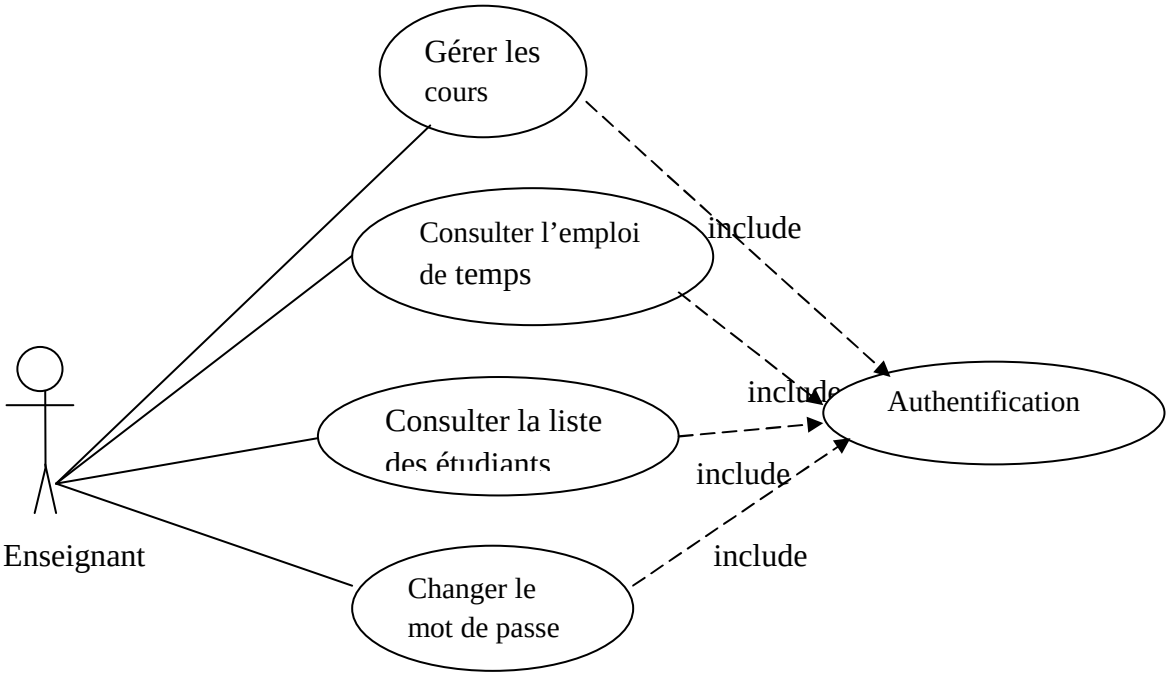



**Figure (IV.9) : Diagramme global de cas d'utilisation relatif à l'Enseignant.**

✓ **Agent de scolarité :**

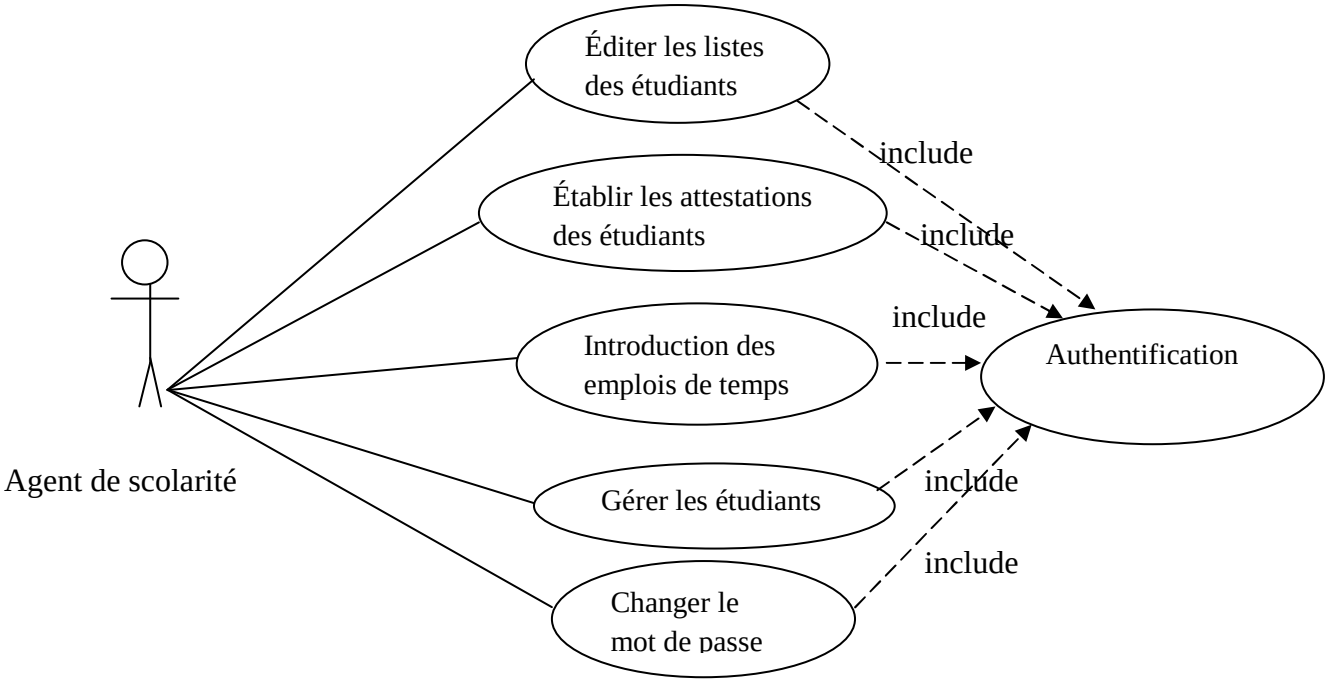



**Figure (IV.10) : Diagramme global de cas d'utilisation relatif à l'Agent de scolarité.**

✓ **Administrateur :**

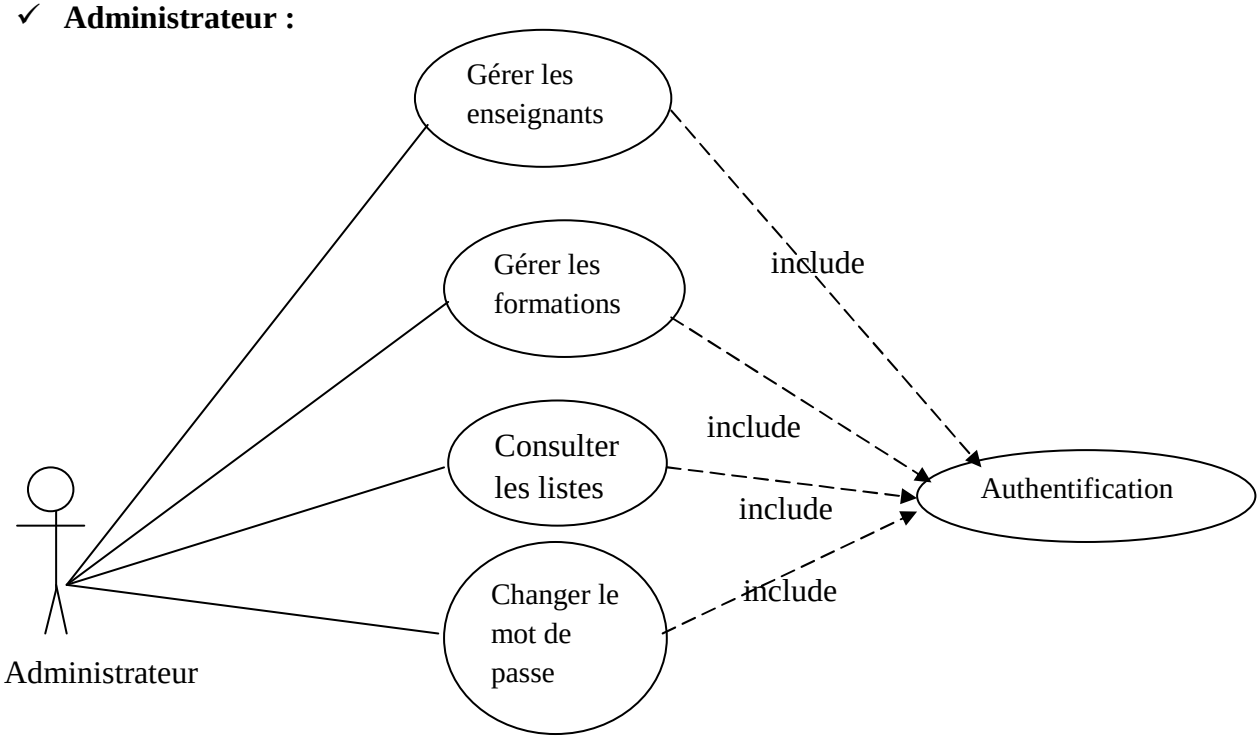



**Figure (IV.11) : Diagramme global de cas d'utilisation relatif à l'administrateur.**

✓ **Visiteur :**

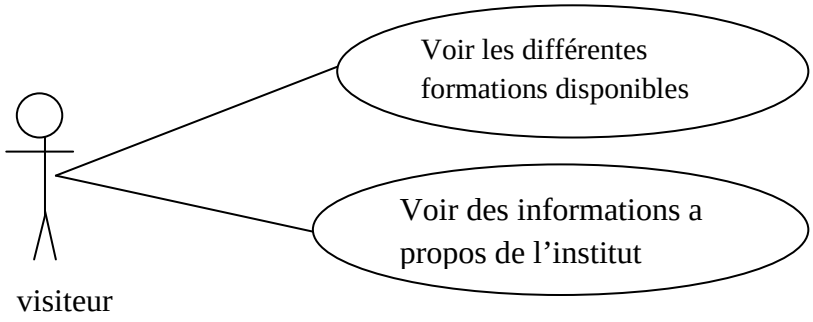



**Figure (IV.12) : Diagramme global de cas d'utilisation relatif au visiteur.**

## IV.4. La Phase de la conception :

Le but principale de la conception est de rendre le modèle d'analyse réalisable sous forme logicielle, c'est ici que les abstractions métiers viennent pour la première fois, au contact de la réalité logicielle.

Pour modéliser la conception de notre application, nous avons utilisé l'extension d'UML pour le web, en construisant les diagrammes de séquences suivis des diagrammes de classes.

### IV.4.1. Diagrammes d'activités :

Le diagramme d'activité apporte un point de vue complémentaire à l'aspect dynamique de la modélisation. Il offre un pouvoir d'expression très proche des langages de programmation objets. Il est donc bien adapté à la spécification détaillée des traitements en phase de réalisation. Un diagramme d'activités se concentre plutôt sur les activités entre les objets, c'est-à-dire, il met en évidence l'activité qui a lieu dans le temps, donc les opérations transmises entre les objets.

- ✓ **Diagramme d'activités du cas d'utilisation <<S'authentifier>> pour l'administrateur :**

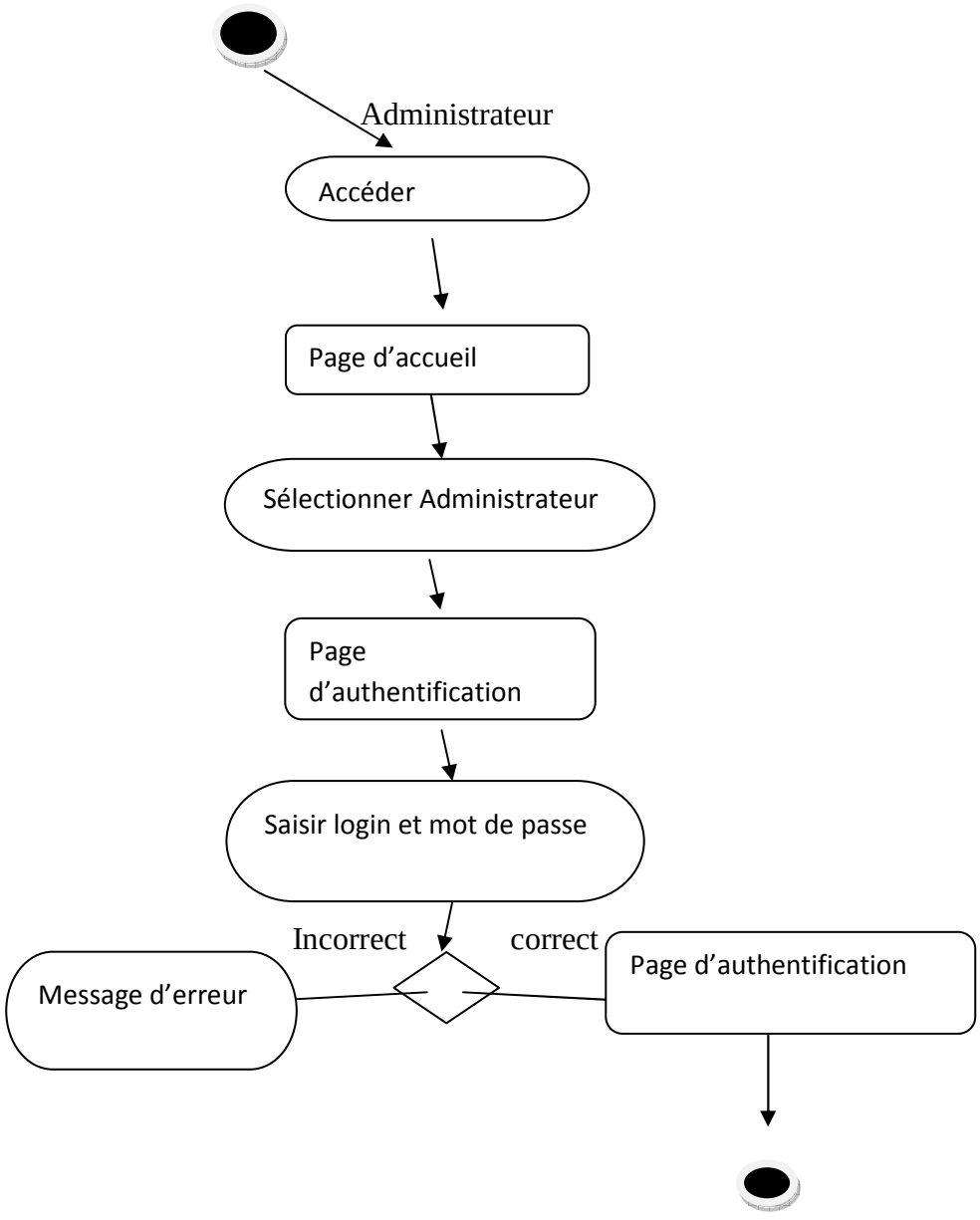



**Figure (IV.13) : diagramme d'activités du cas d'utilisation <<authentification Administrateur>>.**

- ✓ **Diagramme d'activités du cas d'utilisation << gérer les enseignants (ajouter)>> pour l'administrateur:**

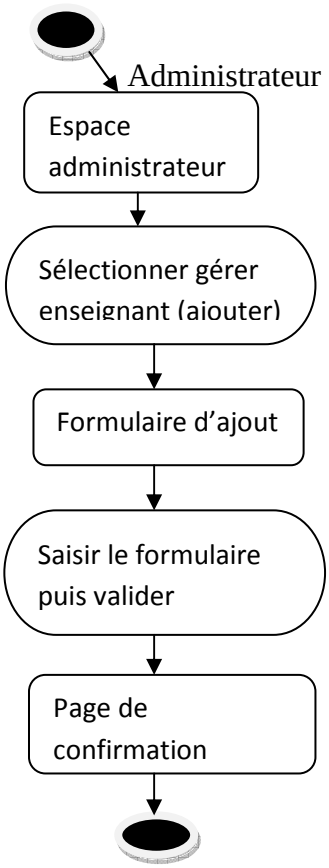



**Figure (IV.14) : Diagramme d'activités du cas d'utilisation<<ajouter enseignant>> pour l'administrateur.**

- ✓ **Diagramme d'activités du cas d'utilisation <<consulter les formations>> pour l'étudiant :**

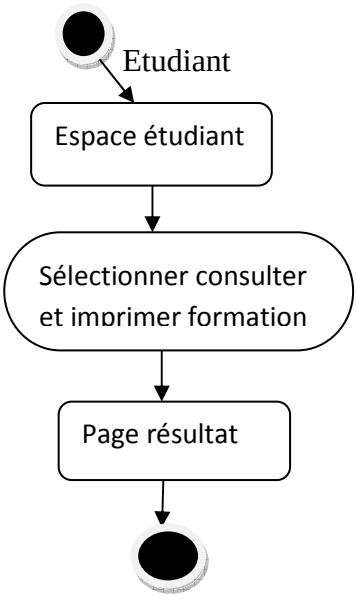



**Figure (IV.15) : Diagramme d'activités du cas d'utilisation << consulter les formations disponible>> pour l'étudiant.**

- ✓ **Diagramme d'activités du cas d'utilisation <<changement du mot de passe>> pour l'enseignant:**

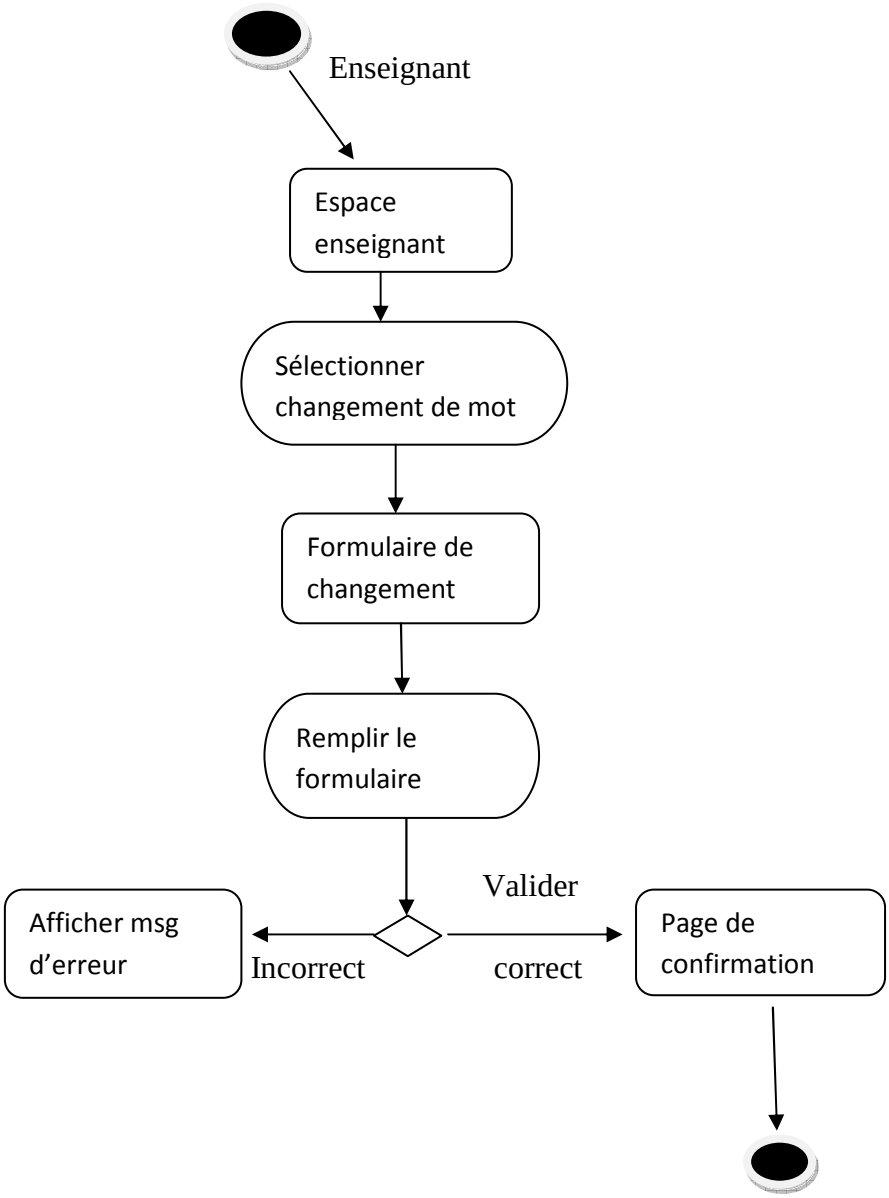



**Figure (IV.16) : Diagramme d'activités du cas d'utilisation <<changement du mot de passe>> pour l'enseignant.**

- ✓ **Diagramme d'activités du cas d'utilisation << éditer les listes des étudiants >> pour l'agent de scolarité :**

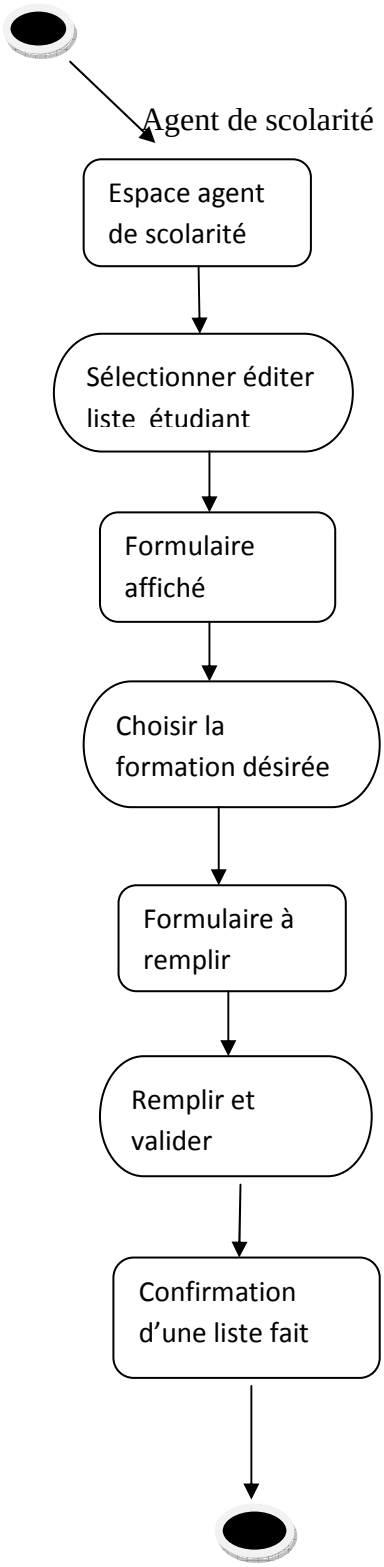


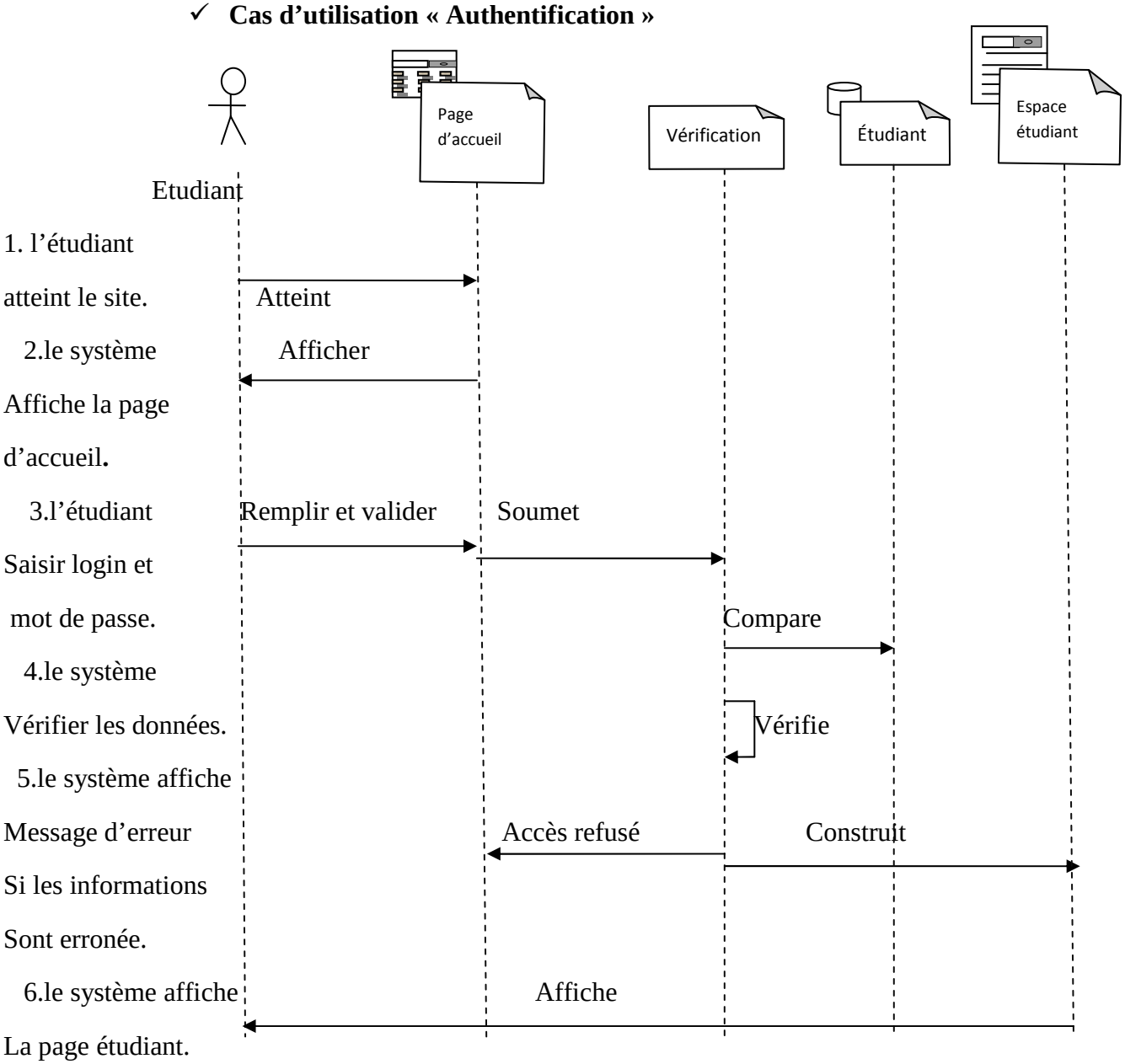

**Figure (IV.17) : Diagramme d'activités du cas d'utilisation << éditer les listes des étudiants >> pour l'agent de scolarité.**

### IV.4.2. Diagrammes de séquences de réalisation des cas d'utilisations:

Selon le formalisme UML (Unified Modeling Language), les diagrammes de séquences détaillés représente graphiquement les interactions (acteurs-systèmes) dans l'évolution temporelle. Les acteurs interagissent donc avec les pages clients et les pages serveurs qui sont utiles pour les interactions avec les ressources du serveur.

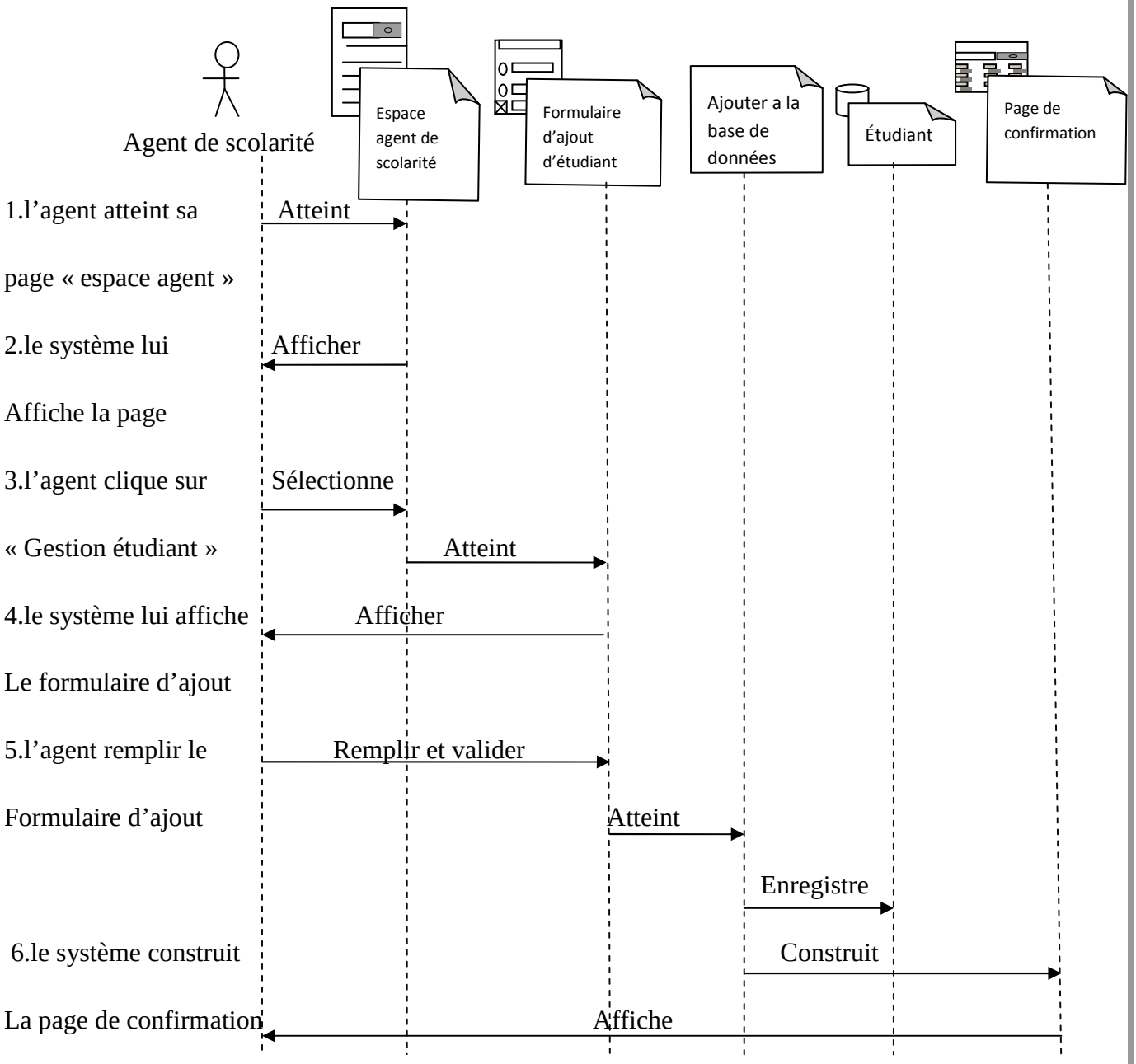
Nous allons décrire ci-dessous quelques diagrammes de séquences de cas d'utilisation important.

- **Etudiant :**
  - ✓ **Cas d'utilisation « Authentification »**



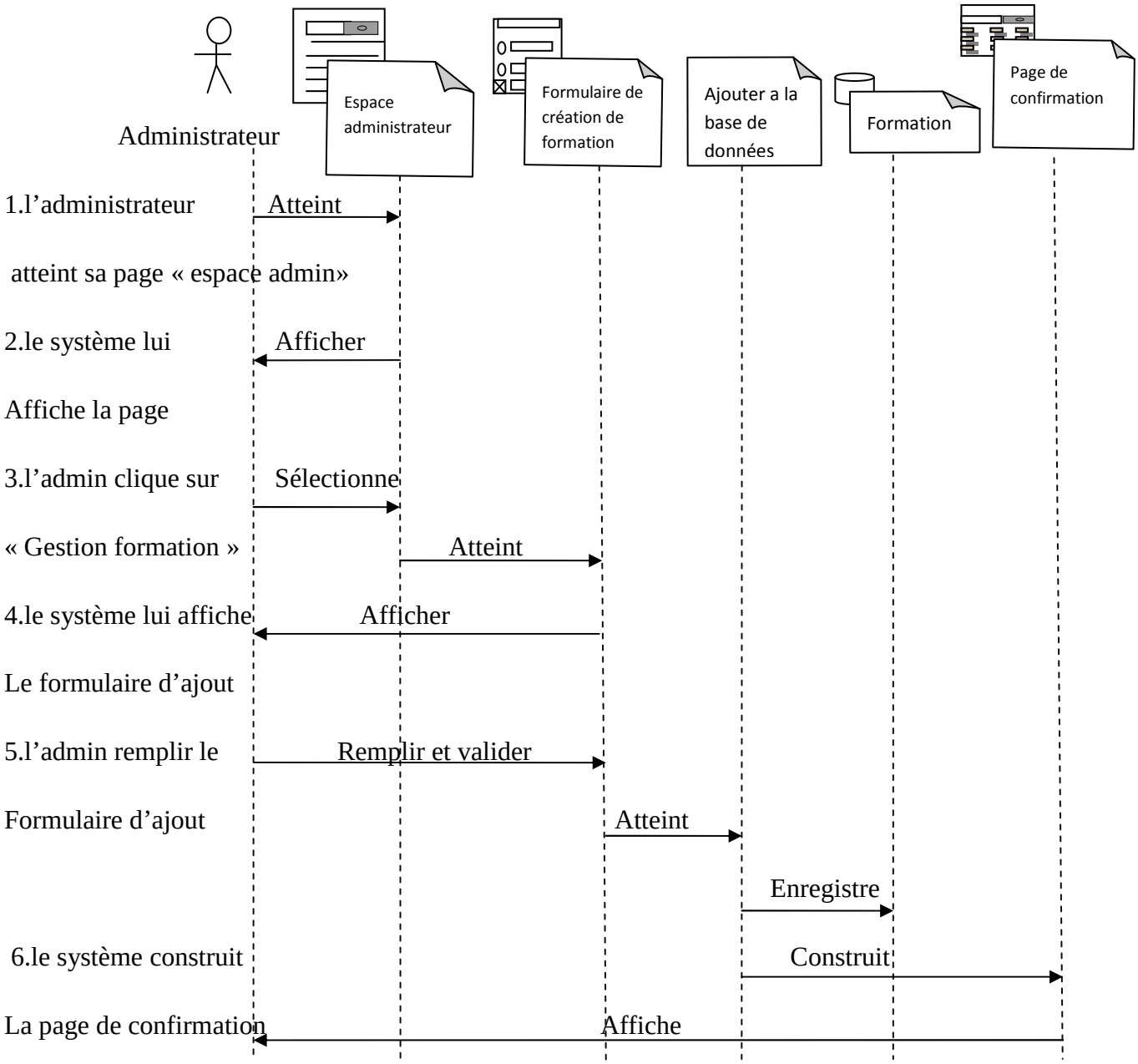

**Figure(IV.18) : Diagramme de séquence de réalisation de cas d'utilisation «Authentification Etudiant ».**

- **Agent de scolarité :**
  - ✓ **cas d'utilisation : « ajouter étudiant »**



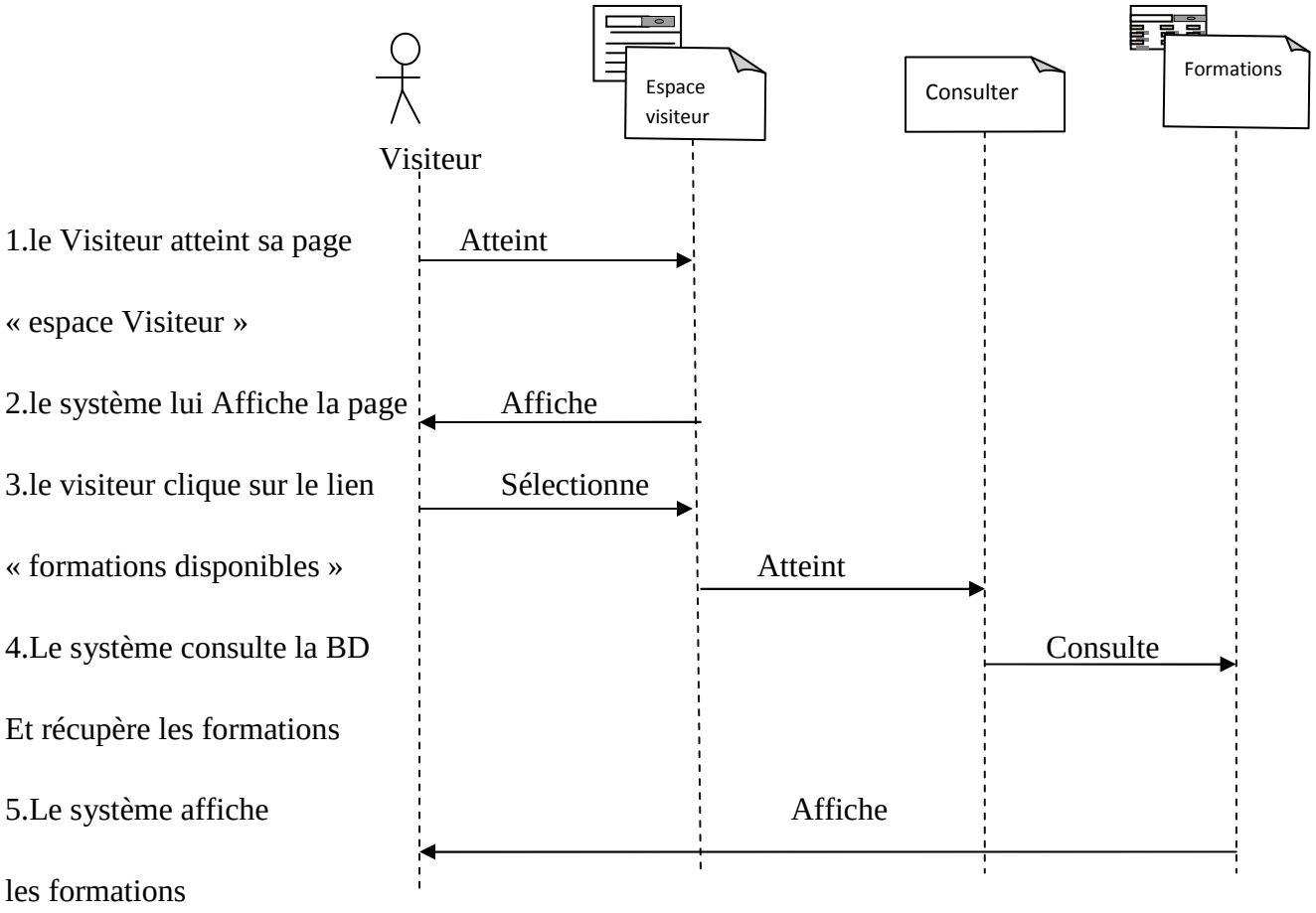

**Figure(IV.19) : Diagramme de séquence de réalisation de cas d'utilisation «ajouter étudiant ».**

- **Administrateur :**
- ✓ **Cas d'utilisation « création formation»**




**Figure(IV.20) : Diagramme de séquence de réalisation de cas d'utilisation « ajouter une formation ».**

- **Visiteur :**
- ✓ **Cas d'utilisation « voir les formation disponible»**




**Figure(IV.21) : Diagramme de séquence de réalisation de cas d'utilisation « ajouter une formation ».**

### IV.4.3. diagramme de classe globale de donnée :

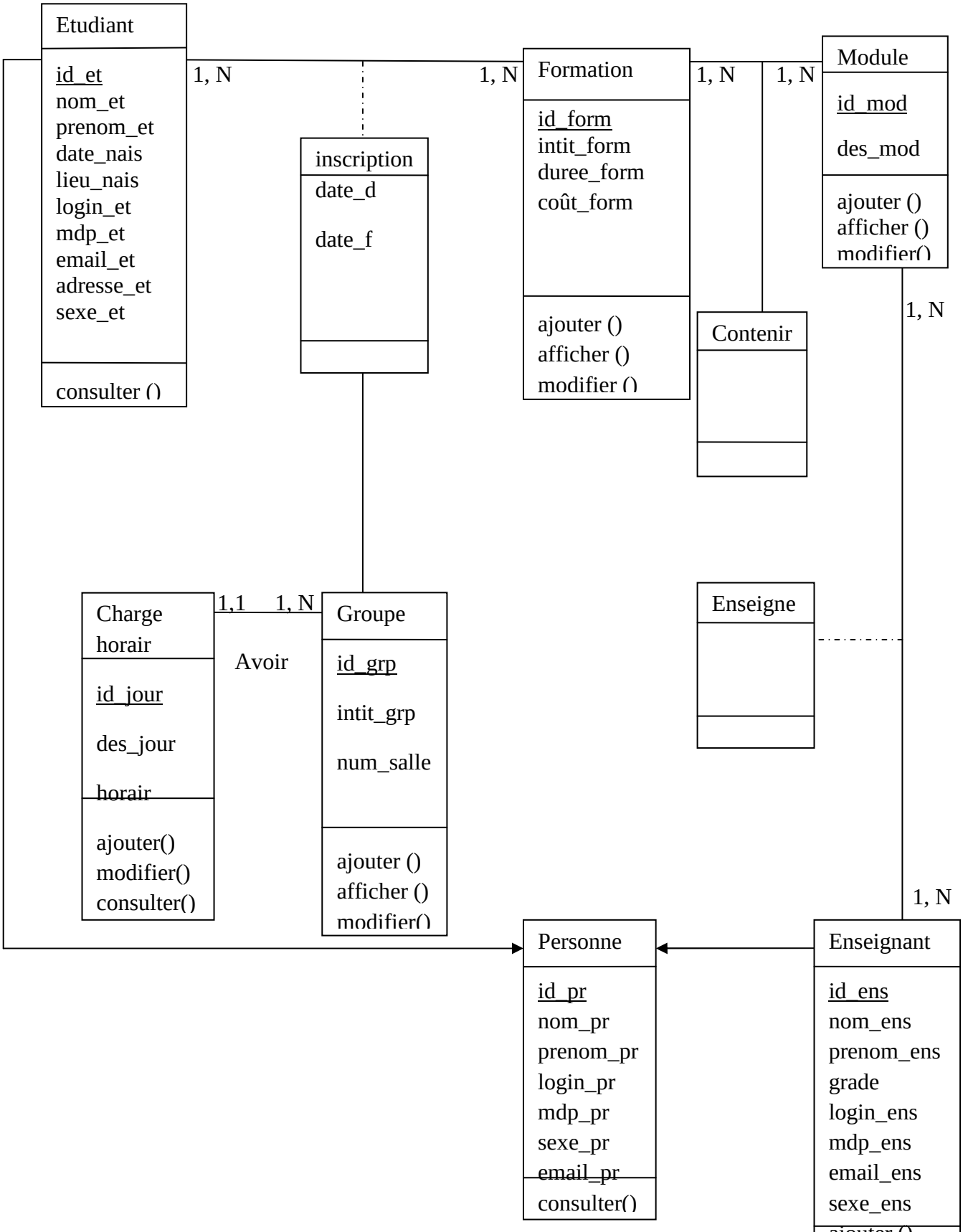



**Figure (IV.22) : diagramme de classe globale de donnée.**

## IV.5. Conclusion :

Dans ce chapitre, nous avons présenté l'analyse et la conception de notre 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, et pour conclure ce chapitre une représentation de diagramme de classe globale de donnée (la base de données) utilisée par notre application est nécessaire.

Dans le chapitre suivant, nous allons définir les outils nécessaires au développement de l'application. Ainsi qu'une présentation des interfaces de l'application réalisée.

## V.1. Introduction :

On s'intéressera dans ce chapitre à l'implémentation et la réalisation de notre application et les différentes fonctionnalités qu'elle nous offre ainsi qu'aux différents outils que nous allons utilisés lors de sa réalisation. En effet, nous y verrons les langages utilisés ainsi que les différentes technologies utilisées pour son développement (Système d'exploitation, Outils de conception web, IDE etc…), et nous aborderons ensuite son fonctionnement et ses différentes interfaces.

## V.2. Les langages de programmation :

### V.2.1. HTML (Hyper Text Markup Language): [10]

HTML est le format de données conçu pour représenter les pages web. Il permet notamment d'implanter de l'hypertexte dans le contenu des pages et repose sur un langage de balisage, d'où son nom. HTML permet aussi de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des éléments programmables tels que des applets.

HTML permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade).

### V.2.2. LE langage de programmation JAVA : [11]

Java est un langage de programmation informatique orienté objet crée 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 plate forme qui garanti la portabilité de Java. Il suffit qu'un système ait une machine virtuelle Java pour que tout programme écrit en ce langage puisse fonctionner.

- ✓ **Les servlets (Technique de développement) :** Une servlet est un programme java qui fonctionne sur un serveur Web et dont le rôle consiste a apporter une réponse a une requête. Elle constitue l'outil idéal pour l'implémentation du tiers médian dans les applications 3-tiers. Une servlet peut ainsi recevoir une requête envoyée par un

navigateur Web, négocier les informations demandées avec une base de donnée et renvoyer le résultat de la requête au navigateur.

Pour exécuter une servlet, il suffit de taper son URL dans la zone d'adresse du navigateur ou de l'interroger dans une page Web. Une servlet peut être invoquée plusieurs fois en même temps pour répandre a plusieurs requêtes simultanées. La servlet se positionne dans une architecteur Client/Serveur trois tiers dans le tiers du milieu entre le client léger chargé de l'affichage et la source de données.

- ✓ **Les jsp :** JSP= Java Server Pages, c'est un fichier contenant du code HTML et des fragments de code java exécutés sur le moteur de sevlets comparable aux langages cotés serveur de type PHP, ASP, les pages JSP sont converties en servlet par le moteur de servlets lors du premier appel a la JSP.

### V.2.3. Java Script : [12]

Javascript (initialement appelé Live script) a été développé par Netscape puis a été repris par la firme Sun.

Les scripts sont des instructions (des lignes de code) interprétées par les navigateurs Netscape et Internet explorer. Le JavaScript n'a cessé d'évaluer avec les versions des navigateurs. JavaScript est un langage de Script qui s'incorpore au langage HTML de présentation des pages Web. Les Scripts vont permettre de rendre une page web plus dynamique.

### V.2.4. SQL :

Le langage SQL (Structured Query Language) peut être considéré comme un langage d'accès normalisé aux bases de données. Il est aujourd'hui supporté par la plupart des produits commerciaux que ce soit par les systèmes de gestion de bases de données micro tel qu'Access ou par les produits plus professionnels tels qu'Oracle ou Sybase. Il a fait l'objet de plusieurs normes ANSI/ISO dont la plus répandue aujourd'hui est la norme SQL2 qui a été définie en 1992. Le succès du langage SQL est dû essentiellement à sa simplicité et au fait qu'il s'appuie sur le schéma conceptuel pour énoncer des requêtes en laissant le SGBD responsable de la stratégie d'exécution. Le langage SQL propose un langage de requêtes ensembliste.

Néanmoins, ce dernier ne possède pas la puissance d'un langage de programmation : entrées/sorties, instructions conditionnelles, boucles et affectations. Donc on a assurée ces traitements grâce au servlet qui contiennent des requêtes SQL et les traitements sur ces dernières.

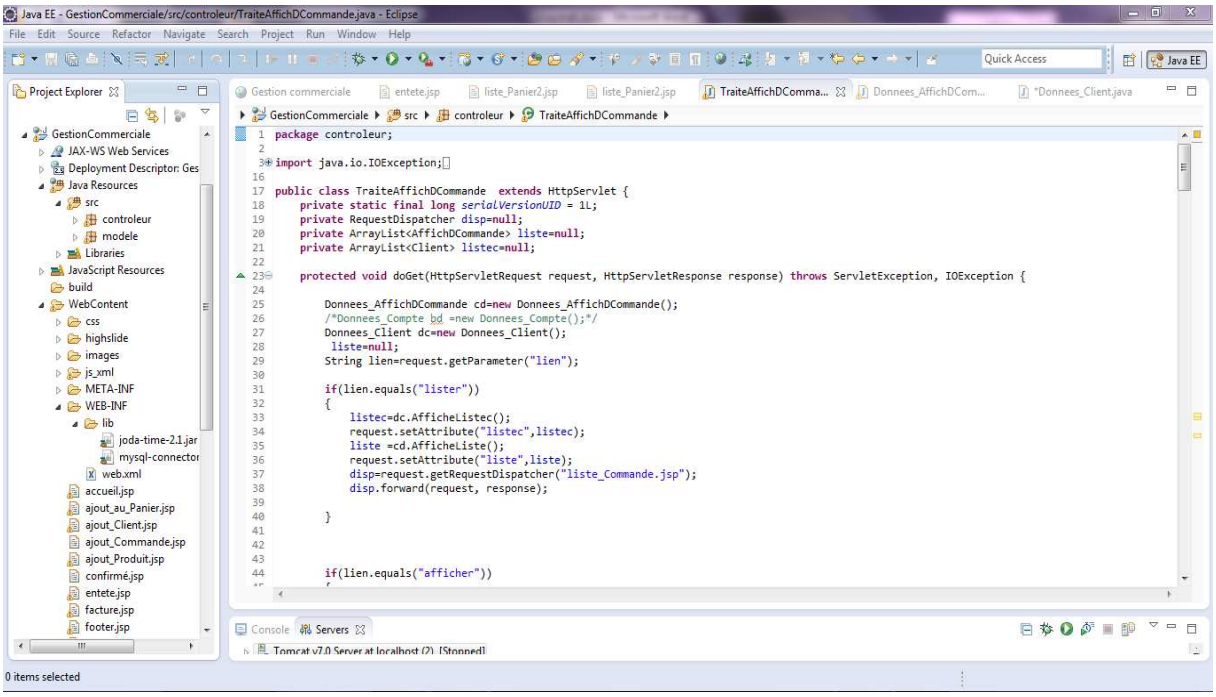
## V.3. Environnement et outils de développement :

### V.3.1. Le Système d'exploitation Windows7 :

Malgré sa récente sortie sur le marché des systèmes d'exploitation, cette dernière version éditer par le géant Microsoft est très apprécié du grand public pour sa rapidité, son efficacité et surtout sa maniabilité.

### V.3.2. Eclipse : [13]

Eclipse est un environnement de développement intégré (IDE) java, développé par IBM extensible, universel et polyvalent, permet de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Il supporte de nombreux outils de développement de haut niveau très complets : un IDE complet Java (JDT) et C/C++ (CDT), un environnement de création de plug-in (PDE), .. etc.il est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), il est extensible par l'intégration des plug-ins.




**Figure(V.1) : L'interface de l'IDE Eclipse.**

### V.3.3. Le serveur Tomcat :

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

### V.3.4. Système de gestion de bases de données MySQL:

Le SGBD MySQL, avec sa configuration client/serveur, est l'un des plus connu dans le monde du web et du logiciel libre. C'est un véritable serveur de base de données SQL (Structured Query Language) qui est un langage de requêtes vers les bases de données exploitant le modèle relationnel.

MySQL est caractérisé par :

- ✓ Une implémentation libre et populaire.
- ✓ Facile à mettre en œuvre.
- ✓ Rapide à apprendre, robuste et convivial.
- ✓ Support multiplateforme.
- ✓ Fiable et rapide.

## V.4. Création de la base de données :

### V.4.1. définition d'une base de données :

Une base de données ( BDD) est un ensemble de données structurées, avec une redondance calculée et accessibles aisément par plusieur programmes, qui les utilisent selon des objectifs distincts, les propriétées en utilisant les concepts proposés par le modèle de données,cette organisation à pour objectif de pouvoir effectuer des intérférences sur ces données sous forme de requétes.les bases de données sont manipulées par un système de gestion de base de données (SGBD).

### V.4.2. passage au modèle relationnel :

Apartir de diagramme de classe globale, nous construisons le modèle relationnel qui vas nous permettre de créer la base de donnée sur la quelle s'appuie notre application :

Etudiant(id_et,nom_et,prenom_et,date_nais,lieu_nais,login_et,mdp_et,email_et,adresse_et, sexe_et) ;
Enseignant (id_ens,nom_ens,prenom_ens,grade,login_ens,mdp_ens,email_ens,sexe_ens) ;
Module (id_mod,des_mod) ;

Enseigne (id_ens*,id_mod*) ;

Formation (id_form,intit_form,duree_form,coût_form) ;
Groupe (id_grp,intit_grp,num_salle) ;

Contenir (id_mod*,id_form*) ;

Inscription (id_et*,id_form*,date_d,date_f,) ;

Personne (id_pr,nom_pr,prenom_pr,login_pr,mdp_pr,sexe_pr,email_pr) ;

Charge horair(id_jour,des_jour,horair);

- ✓ **Table etudiant :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_et, | identifiant etudiant | INT(15) | primaire |
| nom_et | nom etudiant | VARCHAR(20) | |
| prenom_et | prenom etudiant | VARCHAR(20) | |
| date_nais | Date de naissance etudiant | Date | |
| lieu_nais | Lieu de naissance etudiant | VARCHAR(25) | |
| login_et | Login etudiant | VARCHAR(10) | |
| mdp_et | Mot de passe etudiant | VARCHAR(10) | |
| email_et | Email etudiant | VARCHAR(20) | |
| adresse_et | Adresse etudiant | TEXT | |
| sexe_et | sexe | VARCHAR(10) | |

- ✓ **Table enseignant :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_ens | identifiant enseignant | INT(15) | primaire |
| nom_ens | nom enseignant | VARCHAR(20) | |
| prenom_ens | prenom enseignant | VARCHAR(20) | |
| grade | grade de l'enseignant | VARCHAR(10) | |
| login_ens | Login enseignant | VARCHAR(20) | |
| mdp_ens | Mot de passe enseignant | VARCHAR(20) | |
| email_ens | Email enseignant | VARCHAR(20) | |
| Sexe_ens | Sexe enseignant | VARCHAR(10) | |

- ✓ **Table module :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_mod | identifiant module | INT(11) | primaire |
| Des_mod | Description du module | VARCHAR(10) | |

- ✓ **Table enseigne :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_mod | identifiant module | INT(11) | étrangère |
| id_ens | Identificateur enseignant | INT(15) | étrangère |

- ✓ **Table formation :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_form | identifiant du formation | INT(20) | primaire |
| intit_form | Intitulé de la formation | VARCHAR(40) | |
| duree_form | Durée de la formation | VARCHAR(40) | |
| coût_form | Cout de la formation | VARCHAR(40) | |

- ✓ **Table contenir :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| Id_form | identifiant du formation | INT(20) | Etrangère |
| id_mod | Code du module | INT(11) | Etrangère |

- ✓ **Table personne :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_pr | identifiant personne | INT(15) | primaire |
| nom_pr | Nom personne | VARCHAR(20) | |
| prenom_pr | Prenom personne | VARCHAR(20) | |
| login_pr | Login personne | VARCHAR(20) | |
| mdp_pr | Mot de passe personne | VARCHAR(20) | |
| sexe_pr | Sexe personne | VARCHAR(10) | |
| email_pr | Email personne | VARCHAR(20) | |

- ✓ **Table charge horair :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_jour | identifiant jour | INT(11) | primaire |
| des_jour | Description jour | VARCHAR(10) | |
| horair | Les horaires | VARCHAR(40) | |

- ✓ **Table inscription :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| date_d | Date début | Date | primaire |
| date_f | Date fin | Date | |
| id_et | identifiant etudiant | INT(15) | Etrangère |
| id_form | Identifiant du formation | INT(20) | Etrangère |

- ✓ **Table groupe :**

| Nom du champ | Description du champ | Type du champ | clé |
|---|---|---|---|
| id_grp | Identifiant de groupe | INT(20) | primaire |
| intit_grp | Intitulé de groupe | VARCHAR(40) | |
| num_salle | Numéro de la salle | INT(11) | |

## V.5. Présentation de quelques interfaces de l'application :

Nous allons vous présenter dans cette partie les interfaces principales de notre application :

### V.5.1. L'interface principale « page d'accueil » :

L'interface d'accueil, contient les liens nécessaires et indispensables pour la bonne exploration de l'application par les utilisateurs.


**Figure(V.2) : page d'accueil principale.**

### V.5.2. L'interface principale Visiteur :

C'est une page qui permet aux visiteurs d'avoir des informations sur l'école (les formations disponibles,…)


**Figure(V.3) : page d'accueil visiteur.**

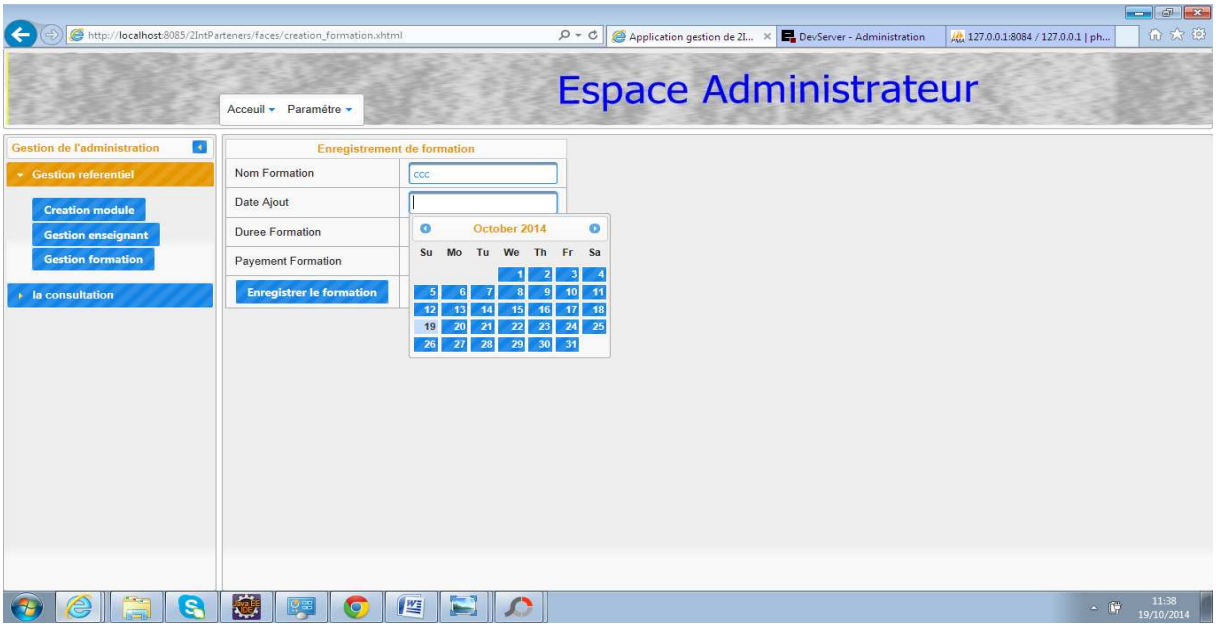
### V.5.3. L'interface principale Administrateur :

C'est un espace réservé à la personne charger de la gestion de (module, formation, enseignant) :


**Figure(V.4) : page d'accueil administrateur.**

### V.5.4. La page « ajouter formation » :

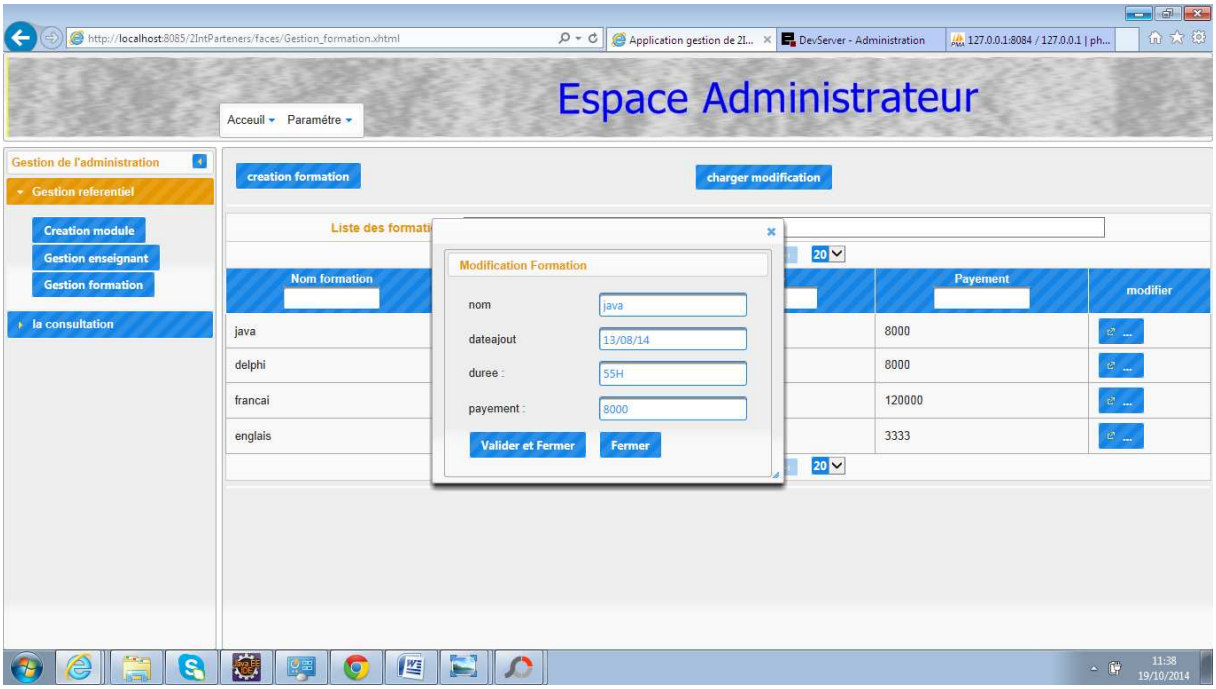
Elle peut être atteinte par l'administrateur, après l'authentification et le choix de gestion formation, pour modifier des formations précisé :




**Figure(V.5) : page ajouter une formation.**

### V.5.5. La page « modifier formation » :

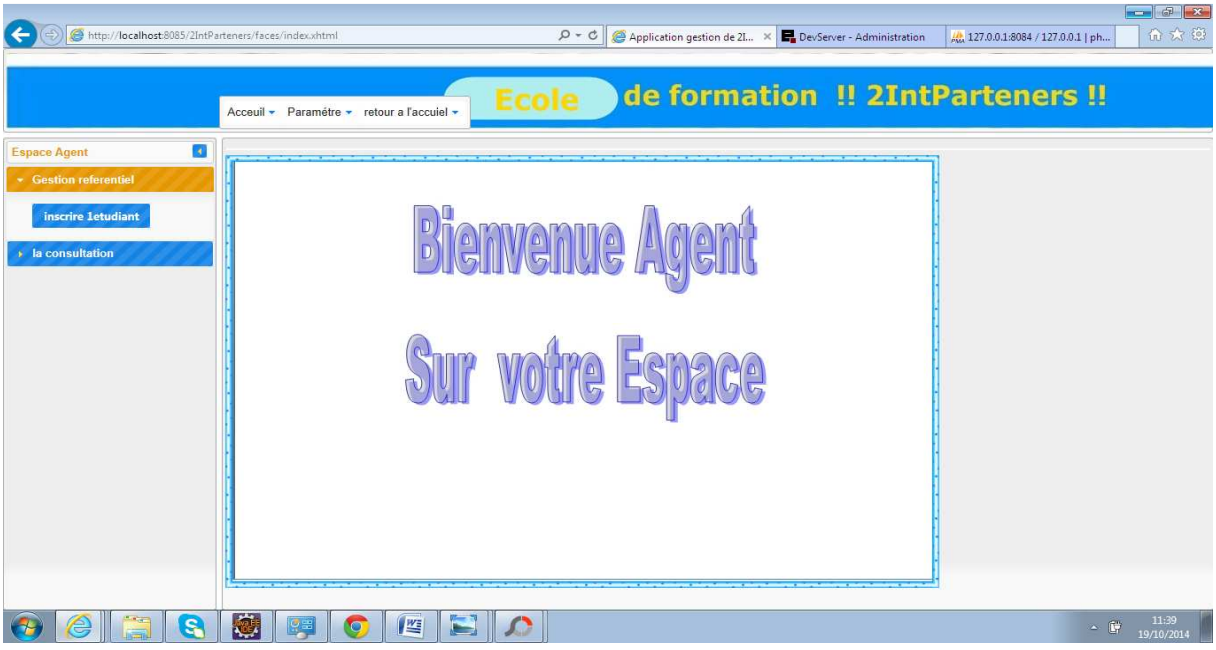
Elle permet à l'administrateur de modifier des formations :




**Figure(V.6) : page modifier une formation.**

### V.5.6. L'interface principale agent de scolarité :

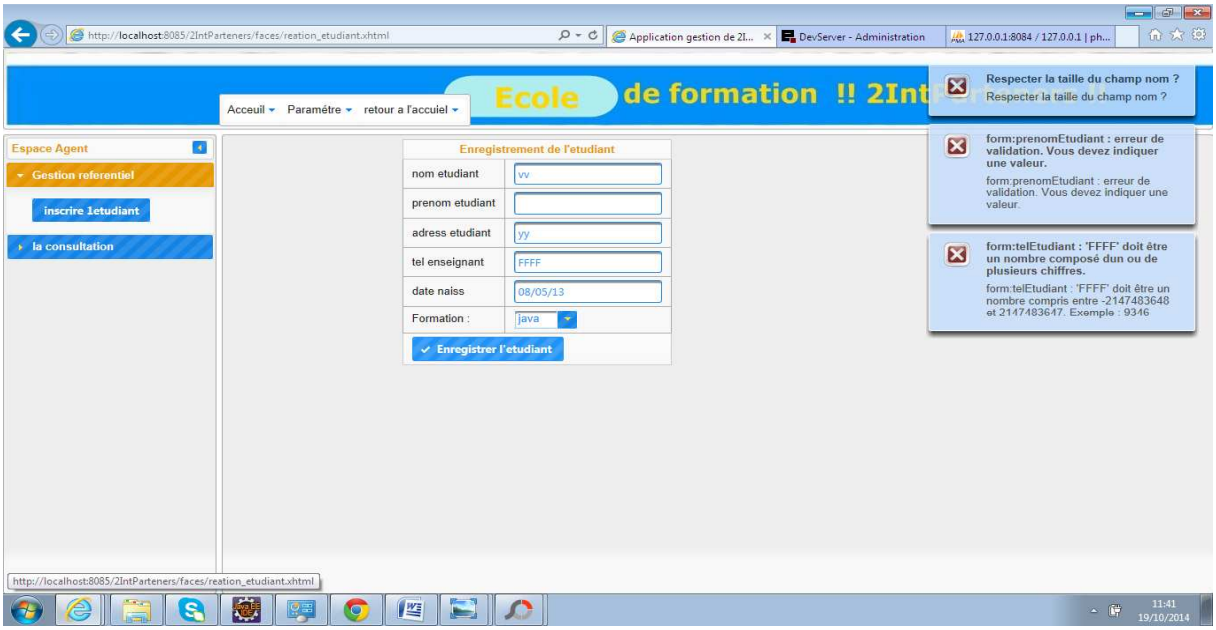
Elle peut être atteint par l'agent après l'avoir authentifié et saisir le mot de passe :




**Figure(V.7) : page d'accueil d'agent de scolarité.**

### V.5.7. La page « ajouter étudiant » avec des fautes l'heur de remplissage :

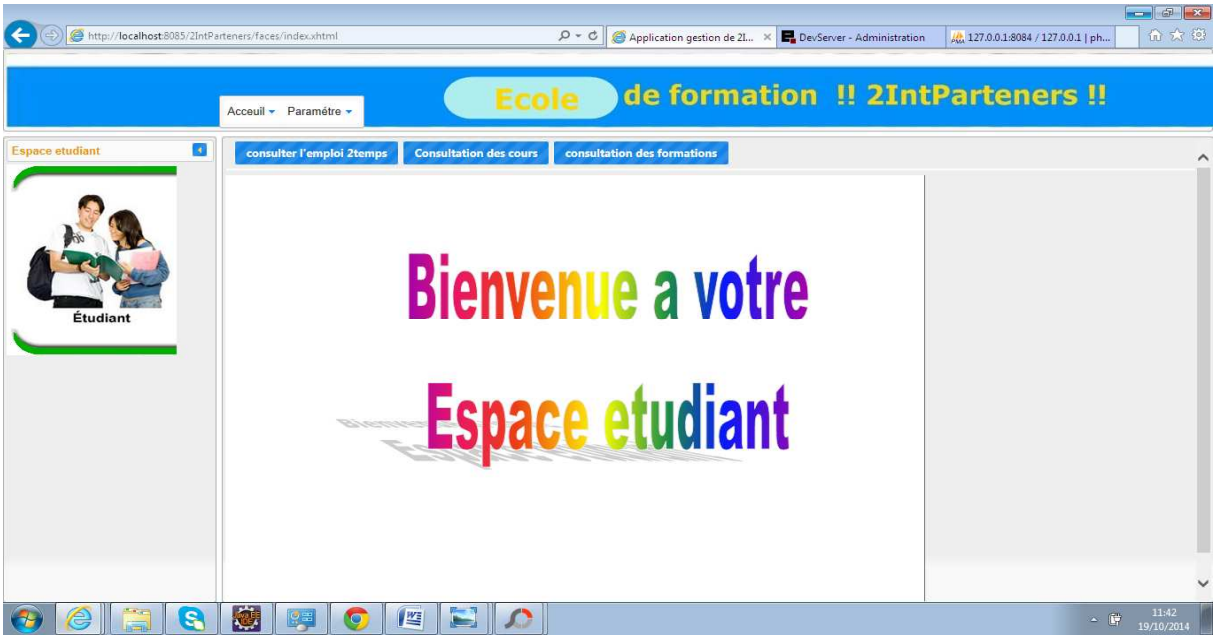
Elle peut être atteinte par l'agent, après l'authentification et après avoir sélectionné ajouter étudiant, elle renvois des messages d'erreurs s'il y on a après le contrôle :




**Figure(V.8) : page ajouter étudiant.**

### V.5.8. L'interface principale étudiant :

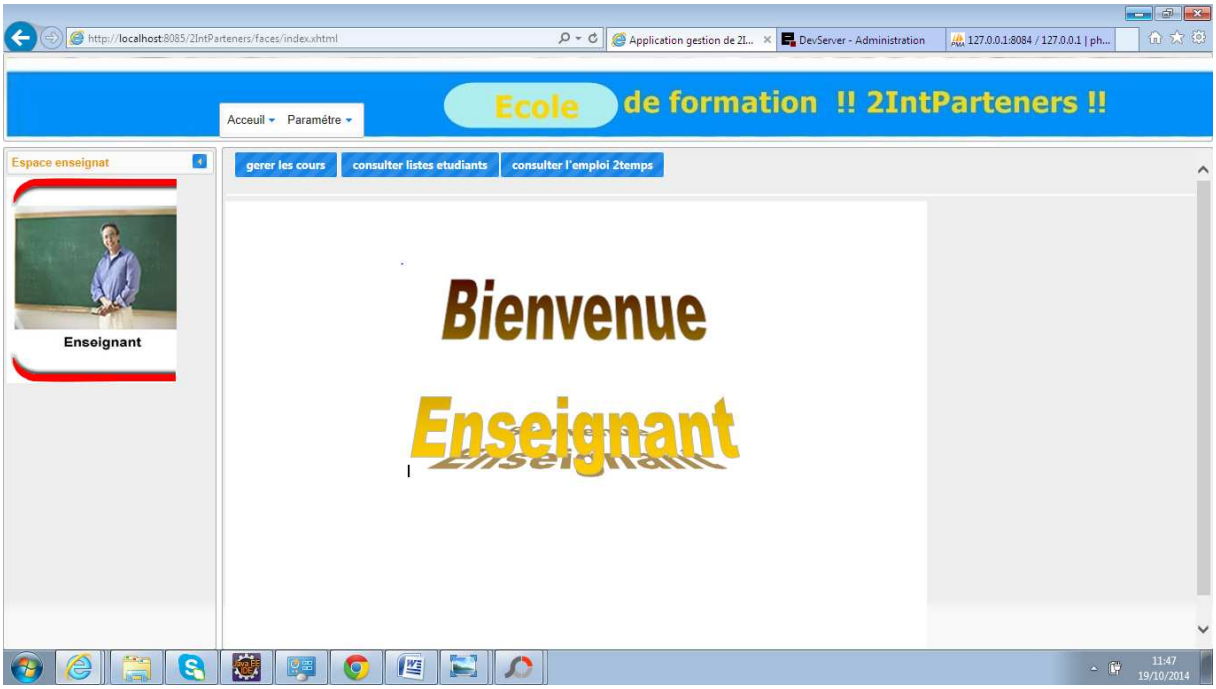
Elle peut être atteinte par l'étudiant après le saisir de son login et mot de passe :




**Figure(V.9) : page d'accueil d'étudiant.**

### V.5.9. L'interface principale enseignant :

Elle peut être atteinte par l'enseignant après avoir saisir son login et mot de passe :




**Figure(V.10) : page d'accueil enseignant.**

## V.6. Conclusion :

Dans ce chapitre, nous avons présenté au premier lieu les différents langages de programmation utilisées, puis nous avons présentées l'environnement de développement de notre application et les outils utilisés pour l'implémentée, Ensuit nous avons fait une représentation des tables de la base de données utilisée par notre application, pour conclure nous avons présenté quelques interfaces de notre application.

Toute Organisation, doit suivre l'évolution de la technologie et se mettre a jour pour assurer des bon systèmes informatiques, afin d'avoir un meilleure rendement, une meilleure qualité de service et de minimise le temps de réponse et de satisfaire la clientèle qui ne cesse d'augmenter ainsi que la bonne gestion de son travail et éviter de tomber dans des situations critiques.

L'étude et le développement de ce projet informatique nous ont permis, non seulement de nous familiariser avec le domaine professionnel(les écoles des formations), mais aussi d'acquérir une certaine maitrise, a savoir les outils de développement tels que Eclipse, le serveur web Apache(tomcat) et le serveur de base données MySQL, coté conception orienté objet nous avons opté pour le langage UML, tandis que pour la réalisation nous avons utilisé le langage de programmation JAVA web(J2EE) (Servlets, JSP) pour l'application web, ainsi JavaScript et SQL.

Pour réaliser notre projet nous l'avons décomposé en deux parties :

En premier lieu nous avons fait une analyse préalable pour l'école en citant tous ses acteurs (étudiant, enseignant, agent de scolarité, Administrateur, visiteur) et leurs besoins.

Ensuite, nous avons procédé à la réalisation d'une application web avec J2EE pour la gestion de scolarité qui répond aux besoins et qui offre aux utilisateurs de site plusieurs services tel que la consultation des formations pour le visiteur, gère les cours pour l'enseignant…

Pour finir, nous espérons avoir répondu aux attentes et aux exigences de l'école, et que l'application réalisée sera utile pour ses utilisateurs.

# Bibliographie


[1] : George et Olivier Gardarin , « Le client /Serveur »,Edition EYROLLES 2000.

[2] :http://www.informatique-inside.com/index.php ?page client/serveur.

[3]: conception et realisation d'une application web de gestion de la scolarité réalisé par Mlle Djaghlouli Nadia et Mlle Zerrouki djamila, proposé par : Mr H.Acheur, Ingénieur promotion 2006/2007.

[4]:jeant Francois Pillou, Client/serveur, Edition : Dunod, 2006, PDF.

[5]: http://www.wiképédia.com.

[6]:document: J2EE vs Net Centre d'enseignement de grenoble 2008/2009 qui se trouve sur :

Membres-liglab.image.fr/plumejeaud/NFE107…/J2EE%20vsù20NET.pdf.

[7] : Guillaume CRESTA,GATCHA Charles, MOUNISSAMY Sivakumar, « Framework Hibernate », Tutorial ISIMA Octobre 2005.

[8] : Grady Booch et al, « le guide de l'utilisation d'UML », Edition EYROLLES 2003.

[9] : Pierre-allain MULLER, « Modélisation objet avec UML », Edition : Eyrolles 1997.

[10] : http://fr.wikipedia.org/wiki/html.

[11] :«Servlets et Java Server page le guide du développeur », P. Y. Saument, A.Mirecourt.

[12] : WWW.ccim.be/ccim328/js.

[13]: http//www.Wikipedia.org.

## 1. Introduction :

Dans ce chapitre, nous allons entamer le processus de développement par une analyse qui mettra en évidence les différents acteurs intervenant dans le système cible ainsi que leurs besoins. La phase conception, s'appuyant sur les résultats de la phase analyse donnera la modélisation des objectifs à atteindre. Pour ce faire, notre démarche va s'appuyer sur le langage UML, qui permet la représentation de la dynamique et la statique du système à travers les différents diagrammes qu'il offre.

## 2. Définition UML :

UML est le facteur de standardisation, car il est impossible de prétendre imposer une méthode, une manière de faire, à toute l'industrie. En revanche on peut prétendre. Définir un langage de modélisation qui, s'il est suffisamment général, sera adopté comme moyen de communication. UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration d'un modèle. Cependant, dans le cadre de la modélisation d'une application informatique, les acteurs d'UML préconise d'utiliser une démarche :

- ✓ Itérative et incrémentale.
- ✓ Guidée par les besoins des utilisateurs de système.
- ✓ Centrée sur l'architecture logicielle.

## 3. La modélisation UML :

UML fournit une panoplie d'outils permettant de présenter l'ensemble des éléments du monde objet (classe, objet….) ainsi que les liens qui les relie. Toutefois, étant donné qu'une seul représentation est trop subjective, UML fournit un moyen astucieux permettant de représenter diverse projection d'une même représentation grâce aux vues. une vue est constituée d'un ou plusieurs diagrammes.

On distingue trois types de vues :

- ✓ Les vues statique : représentant le système physiquement.
  - Diagramme de classe
  - Diagramme d'objets
  - Diagramme de composants
  - Diagramme de déploiement
- ✓ Les vue Dynamique :
  - Diagramme d'états
  - Diagramme d'activités
  - Diagramme de séquence

- Diagramme de collaboration

✓ Le vue fonctionnelle :

- Diagramme de cas d'utilisation

✓ UML suit une démarche en trois étapes :

- Analyse
- Conception
- Implémentation

✓ Les briques d'UML : composées de :

- Les éléments
- Les relations
- Les diagrammes

**4. Eléments d'UML :**

Il existe quatre types d'éléments dans UML :

✓ les éléments structurels.

✓ les éléments comportementaux.

✓ les éléments de regroupement.

✓ les éléments d'annotation.

**4.1. Les éléments structurels** : Ils représentent les parties statiques du modèle.

Ceux sont des représentations conceptuelles ou physiques d'un système

Il existe 7 types d'éléments structurels :

✓ Classe

✓ Interface

✓ Collaboration

✓ Cas d'utilisation

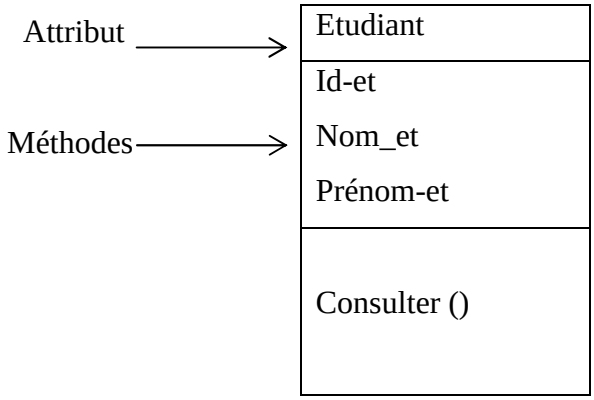
✓ Classe d'activité

✓ Composant

✓ Nœud

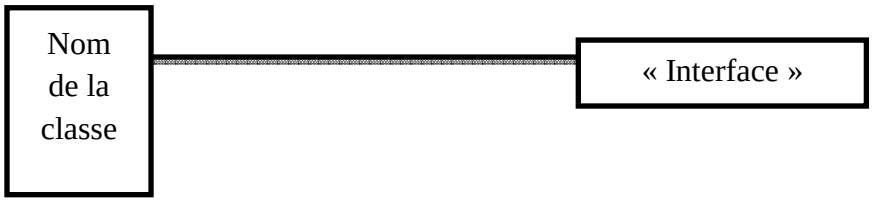
**4.1.1. Classe :** C'est un ensemble d'éléments ayant les mêmes attributs, les mêmes opérations, les mêmes relations et la même sémantique.

**Exemple :**




**4.1.2. Interface :** C'est un ensemble d'opérations définissant la fonction d'un élément ou d'un composant. Elle définit seulement la signature des opérations.

**Exemple :**

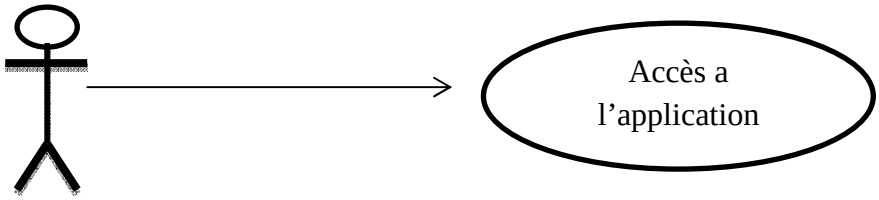



**4.1.3. Collaboration :** La collaboration Elle définit une interaction entre plusieurs éléments. Elle a un rôle structurel et comportemental**.**

**4.1.4. Cas d'utilisation :**

- ✓ Il s'agit de la solution UML pour représenter le modèle conceptuel.
- ✓ Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.
- ✓ Ils centrent l'expression des exigences du système sur ses utilisateurs : ils partent du principe que les objectifs du système sont tous motivés.
- ✓ Ils se limitent aux préoccupations "réelles" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- ✓ Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système.
- ✓ Ils permettent de classer les acteurs et structurer les objectifs du système.

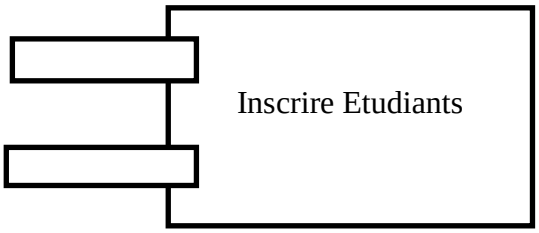
**Exemple :**




**4.1.5. Classe d'activité :** Classe particulière dont les objets possèdent un ou plusieurs processus pouvant lancer une activité de commande.

**4.1.6. Composant :** Le composant est une partie physique d'un système qui conforme a une spécification et fournit la réalisation d'un ensemble d'interface.il peut être un module.exe, dll, com javaBeans, etc. il peut aussi être constitué de plusieurs sous composantes.

**Exemple :**




**4.1.7. Nœud :** C'est un élément physique. Lors d'une exécution, il représente une ressource ayant une capacité de calcul. En règle générale, il à moins de la mémoire et souvent de capacité de traitement .Un nœud est représenté par un cube.

**Exemple :**


**4.2. Les éléments comportementaux,**

**4.2.1. Les interactions :**

Les interactions est l'ensemble des messages échanger entre les éléments du système (résultat de collaboration d'un groupe d'instances), elles sont des messages, des séquences d'actions ou bien des liens (relation entre des éléments). Une interaction peut être visualisée selon le point de vue du temps (diagramme de séquences) ou de celui d'espace (diagramme de collaboration). Un message par exemple est une interaction représenté par une ligne fléchée, indiquant le nom de son opération.

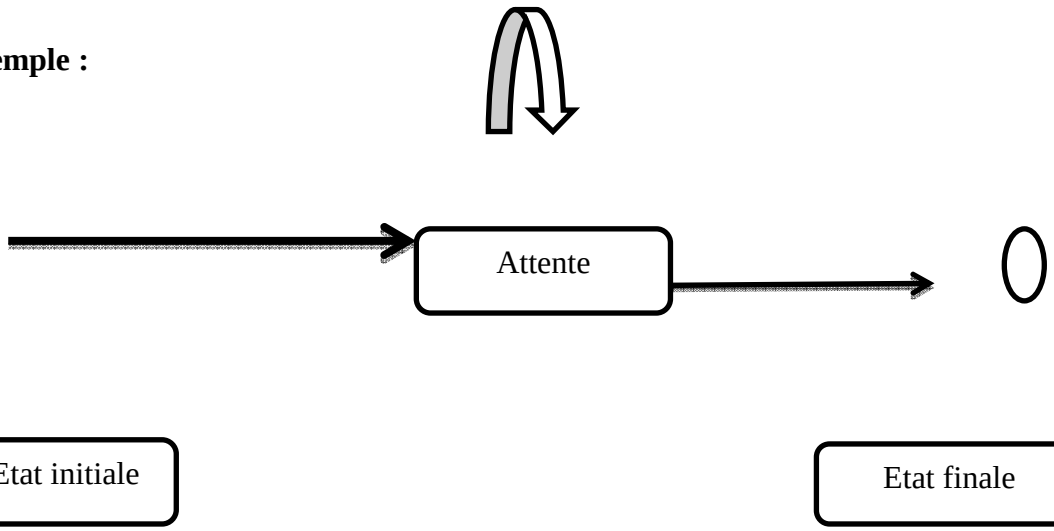
**Exemple :**

Affiche

### 4.2.2. Etat :

La machine à état spécifie le cycle de vie d'un objet quand cet objet a un comportement dynamique. On représente graphiquement un état par un rectangle au coin arrondis (au centre), et en distingue les états de départ (à gauche)et terminaux(a droite).Comme décrit dans figure suivante.

**Exemple :**




### 4.3. Les éléments de regroupement :

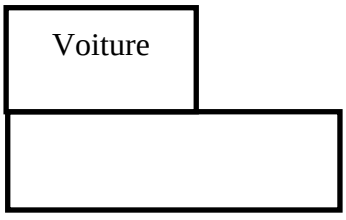
Les éléments de regroupements représentent les parties organisationnelles des modèles UML. Ce sont des boîtes dans lesquelles un modèle peut être décomposé. Il existe un seul type fondamental d'éléments de regroupement : le « paquetage ».

### 4.3.1. Package :

Un package en UML (ou paquetage en français) est un groupe d'éléments, dans le but de les grouper dans des ensembles cohérents. Un package peut contenir la plupart des éléments UML : classes, objets, cas d'utilisations, composantes, etc. Il peut également contenir des packages, créant une hiérarchie complète.

L'avantage des packages est qu'ils permettent de structurer les diagrammes et donnent une vision globale plus claire.

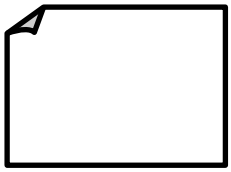
**Exemple :**



### 4.3.2. Les éléments d'annotation :

Ils représentent les parties explicatives modèles. Ceux des commentaires. Il existe 1 type d'élément d'annotions : Ils font partie des décorations.

**Exemple :**



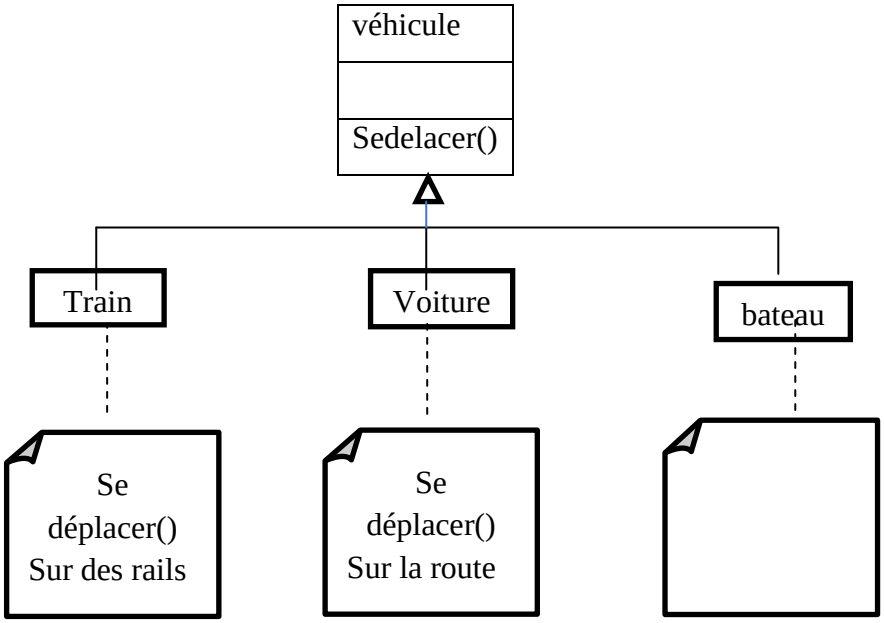
**5. Les Relations :** sont les liens entre les éléments.

Il existe Cinq types de relation interclasse sont couramment utilisés (il en existe d'autres) :

- ✓ héritage
- ✓ dépendance
- ✓ agrégation
- ✓ composition
- ✓ Association

- ✓ **Héritage :**
- ✓ L'héritage est un mécanisme de transmission des propriétés d'une classe (ses attributs et méthodes) vers une sous-classe.
- ✓ Une classe peut être spécialisée en d'autres classes, afin d'y ajouter des caractéristiques spécifiques ou d'en adapter certaines.
- ✓ Plusieurs classes peuvent être généralisées en une classe qui les factorise, afin de regrouper les caractéristiques communes d'un ensemble de classes

**Exemple** :



- ✓ **Dépendance :**

C’est un lien sémantique entre 2 éléments.Un changement de l’un peut affecter la sémantique de l’autre.

**Exemple :** 

- ✓ **Agrégation** :

L’agrégation est un cas particulier de relation d’association qui indique qu’une classe est une partie d’une autre classe. Cetterelation comporte également les ordre de multiplicité On la représente graphiquement en décorant la relation d’association par un losange.

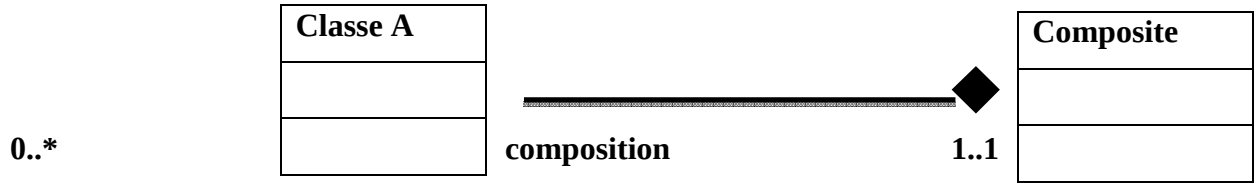
**Exemple** :


- ✓ **Composition :**

Composition Aussi appelée "agrégation forte" ou "agrégation par valeur", il s'agit en fait d'une agrégation à laquelle on impose des contraintes internes : un seul objet peut faire partie d'un composite (l'agrégat de la composition), et celui-ci doit gérer toutes ses parties. En clair, les composants sont totalement dépendants du composite.

En UML, la composition est représentée de la même manière que l'agrégation, mais le diamant est plein.

**Exemple :**

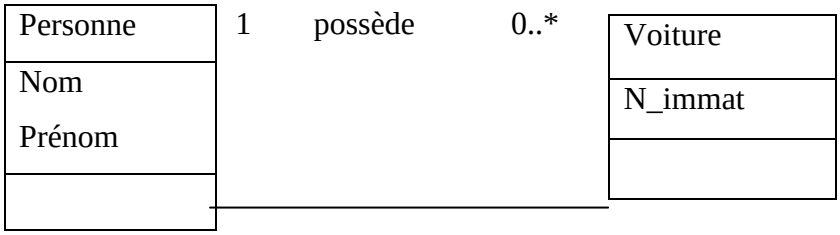



- ✓ **Association :**

C'est la relation la plus simple entre deux classes. Elle existe à partir du moment où l'une des deux classes sert de type à un attribut de l'autre, et que cet autre envoi des messages à la première (condition nécessaire pour une association). Simplement, une association indique que deux classes communiquent entre elles (dans un sens ou dans les deux sens).

En UML, une association est représentée par une ligne entre deux classes, possiblement accompagnée d'une flèche si l'association n'est pas bidirectionnelle.

**Exemple :**

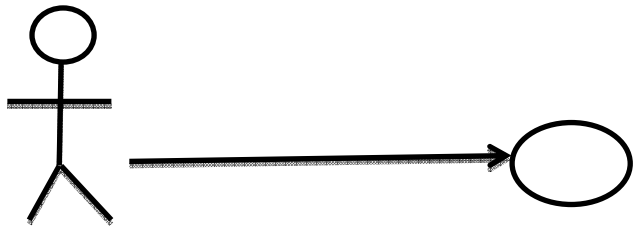



**6. Les différents types de diagrammes UML :**

**6.1. Diagramme cas d'utilisation :**

- ✓ Il s'agit de la solution UML pour représenter le modèle conceptuel.(est de comprendre et structurer les besoins du client.)
- ✓ Les use cases permettent de structurer les besoins des utilisateurs et les objectifs correspondants d'un système.
- ✓ Ils centrent l'expression des exigences du système sur ses utilisateurs : ils partent du principe que les objectifs du système sont tous motivés.
- ✓ Ils se limitent aux préoccupations "réelles" des utilisateurs ; ils ne présentent pas de solutions d'implémentation et ne forment pas un inventaire fonctionnel du système.
- ✓ Ils identifient les utilisateurs du système (acteurs) et leur interaction avec le système.
- ✓ Ils permettent de classer les acteurs et structurer les objectifs du système.
- ✓ Ils servent de base à la traçabilité des exigences d'un système dans un processus de développement intégrant UML.

**Exemple :**



Acteur A Cas d'utilisateur x

✓ **Intérêt des cas d'utilisation :**

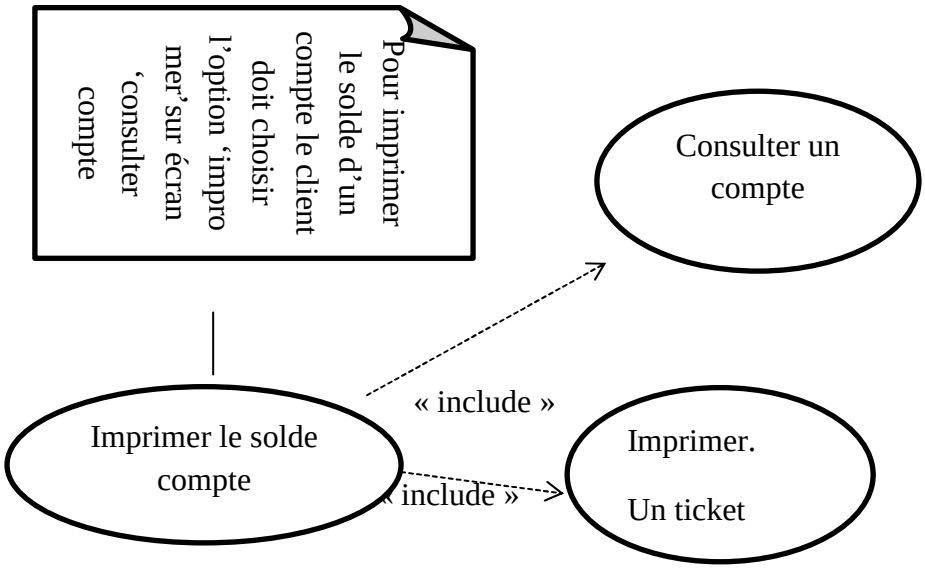
- Le but de la conceptualisation est de comprendre et structurer les besoins du client.
- Il ne faut pas chercher l'exhaustivité, mais clarifier, filtrer et organiser les besoins
- Une fois identifiés et structurés, ces besoins :
  - définissent le contour du système à modéliser (ils précisent le but à atteindre),
  - permettent d'identifier les fonctionnalités principales (critiques) du système.
- Le modèle conceptuel doit permettre une meilleure compréhension du système.
- Le modèle conceptuel doit servir d'interface entre tous les acteurs du projet.
- Les besoins des clients sont des éléments de traçabilité dans un processus intégrant UML.

✓ **Les relations entre cas utilisation :**

Il Ya deux type de relation possibles entre cas :

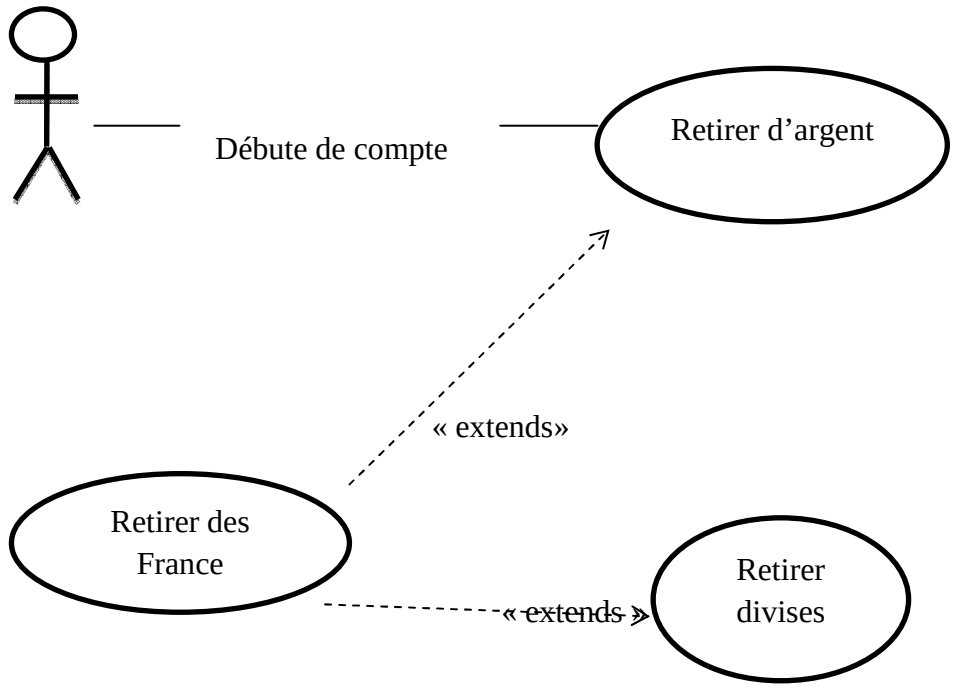
- L'inclusion :quand les cas sources comprend le cas destination
- L'extension :quand le source ajoute son comportement cas.

**Exemple :**




✓ **Relation d'utilisation** : indiqué que le cas d'utilisation source continent aussi le comportement décrit dans le cas d'utilisation destination

**Exemple :**




La relation extension indiqué que le cas d'utilisation source étend (précise) les objectifs (le comportement) du cas d'utilisation destination

**6.2. Diagramme classes :**

✓ **Diagramme de classes : sémantique**

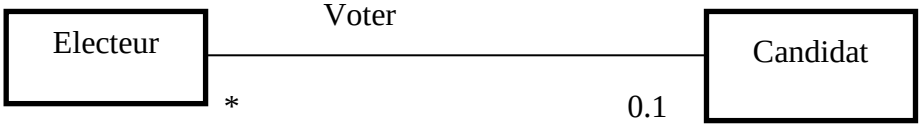
- Un diagramme de classes est une collection d'éléments de modélisation statiques (classes, paquetages...), qui montre la structure d'un modèle.
- Un diagramme de classes fait abstraction des aspects dynamiques et temporels.
- Pour un modèle complexe, plusieurs diagrammes de classes complémentaires doivent être Construits.

  On peut par exemple se focaliser sur :

  - les classes qui participent à un cas d'utilisation (cf. collaboration),
  - les classes associées dans la réalisation d'un scénario précis,
  - les classes qui composent un paquetage,
  - la structure hiérarchique d'un ensemble de classes.

  Les diagramme de classes permettent de représenter l'ensemble des informations formalisés, qui sont regrouper des classes.

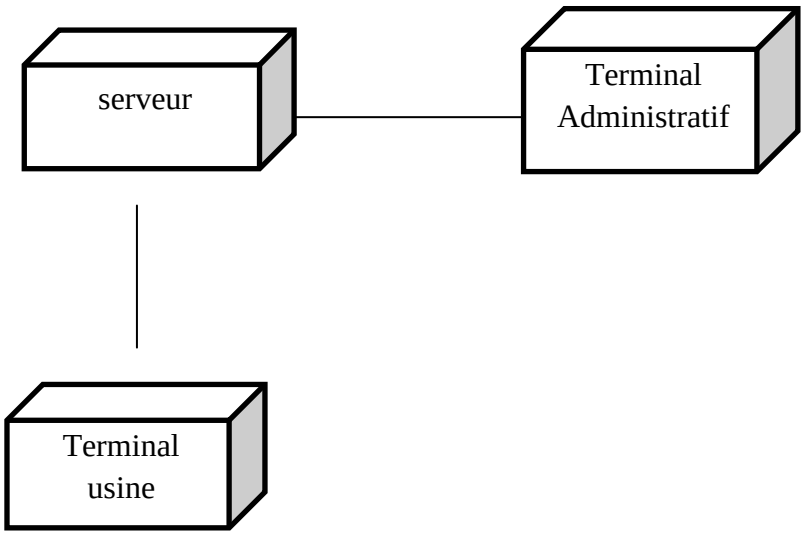
**Exemple :**



### 6.3. Diagramme de déploiement :

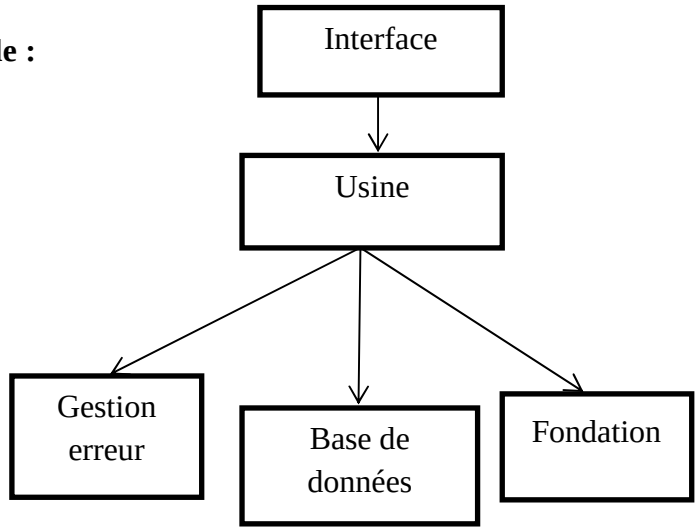
- ✓ Les diagrammes de déploiement montrent la disposition physique des matériels qui composent le système et la répartition des composants sur ces matériels.
- ✓ Les ressources matérielles sont représentées sous forme de nœuds.
- ✓ Les nœuds sont connectés entre eux, à l'aide d'un support de communication. La nature des lignes de communication et leurs caractéristiques peuvent être précisées.
- ✓ Les diagrammes de déploiement peuvent montrer des instances de nœuds (un matériel précis), ou des classes de nœuds.
- ✓ Les diagrammes de déploiement correspondent à la vue de déploiement d'une architecture logicielle .

**Exemple :**




**6.4. Diagramme de composants :** Il représente l'organisation et les dépendances des composants. Un composant est soit une (ou plusieurs) classe(s), interface(s) ou collaboration(s).

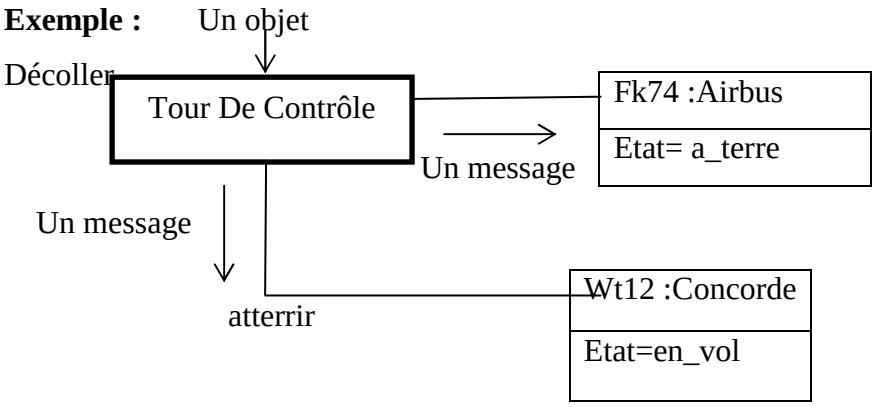
**Exemple :**



### 6.5. Diagramme de collaboration

- ✓ Les diagrammes de collaboration montrent des interactions entre objets (instances de classes et acteurs).
- ✓ Ils permettent de représenter le contexte d'une interaction, car on peut y préciser les états des objets qui interagissent.

**Exemple :**




### 6.6. Diagramme de séquence : sémantique

C'est un diagramme d'interaction. Il représente un ensemble d'objets et leurs relations, avec les messages qu'ils échangent. (aspect chronologique des messages).

#### 6.6.1. Les déférents types de messages :

- ✓ Message simple
- ✓ Message minuté
- ✓ Message synchrone
- ✓ Message asynchrone
- ✓ Message dérobant

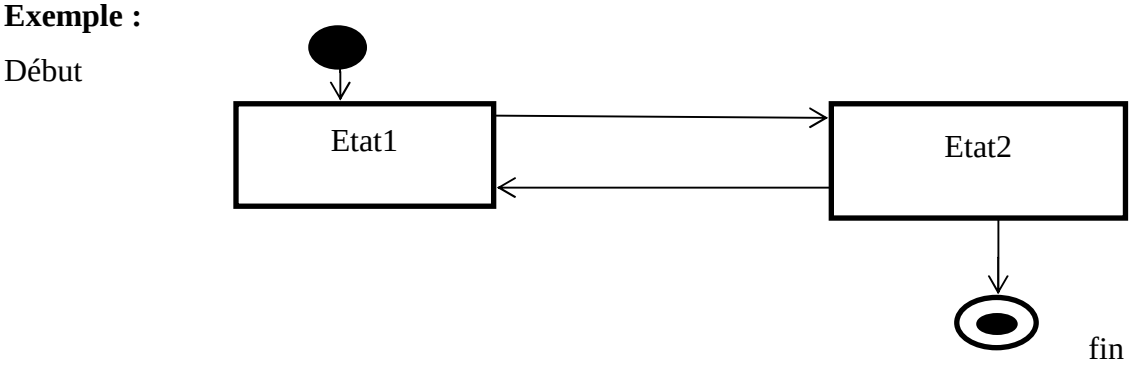
- **message simple :** Message dont on ne spécifie aucune caractéristique d'envoi ou de réception particulière.
- **message minuté (timeout) :** Bloque l'expéditeur pendant un temps donné (qui peut être spécifié dans une contrainte), en attendant la prise en compte du message par le récepteur. L'expéditeur est libéré si la prise en compte n'a pas eu lieu pendant le délai spécifié.
- **message synchrone :** Bloque l'expéditeur jusqu'à prise en compte du message par le destinataire. Le flot de contrôle passe de l'émetteur au récepteur (l'émetteur devient passif et le récepteur actif) à la prise en compte du message.

- **message asynchrone :** N'interrompt pas l'exécution de l'expéditeur. Le message envoyé peut être pris en compte par le récepteur à tout moment ou ignoré (jamais traité).
- **message dérobant :** N'interrompt pas l'exécution de l'expéditeur et ne déclenche une opération chez le récepteur que s'il s'est préalablement mis en attente de ce message.

### 6.7. Diagramme d'états-transitions : sémantique

- ✓ Ce diagramme sert à représenter des automates d'états finis, sous forme de graphes d'états, reliés par des arcs orientés qui décrivent les transitions.
- ✓ Les diagrammes d'états-transitions permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.
- ✓ Un état se caractérise par sa durée et sa stabilité, il représente une conjonction instantanée des valeurs des attributs d'un objet.
- ✓ Une transition représente le passage instantané d'un état vers un autre.
- ✓ Une transition est déclenchée par un événement. En d'autres termes : c'est l'arrivée d'un événement qui conditionne la transition.
- ✓ Les transitions peuvent aussi être automatiques, lorsqu'on ne spécifie pas l'événement qui la déclenche.

**Exemple :**




**Conclusion :**

L'UML, comme l'on a vu à travers ce chapitre, ne propose pas une démarche objet mais une notation adapté au monde de développement orienté objet. Il nous a donc permis de s'initier aux techniques de modélisation objet. La notation UML peut s'adapter a tous les projets informatiques.