République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Saâd DAHLAB de Blida

Faculté des Sciences
Département d'informatique.

Mémoire présenté par :

- **Djellali Mohamed.**
- **Tidafi Salim**

En vue d'obtenir le diplôme de Master

Domaine : Mathématique et Informatique.

Filière : Informatique.

Spécialité : Informatique.

Option : ingénierie du logiciel.

Sujet:

Un Système Coopératif de gestion des formations

Rapporteur : Dr Djamel Benouar

2011/2012

Résumé

ملخص

يندرج هذا المشروع في اطار انشاء برنامج الكتروني موجه لتسير ادارة التكوينات العلمية الذي يتناسب مع مختلف انماط التكوينات (الجامعة , الثانوية , المتوسطة و الابتدائية) مع اضافة نظام اتصالات لتوصيل مختلف المؤسسات لتبادل المعلومات الخاصة بالمتعلمين.

كلمات المفاتيح :ادارة التكوينات,نظام اتصالات

Résumé :

Ce projet entre dans le cadre de la réalisation d'une application web dédie a la gestion administratif des formations qui s'adapte a n'importe quelle institution tel que l'université , lycée ,CEM, primaire et les centres de formation professionnel, ainsi d'ajouter une couche de communication qui permet au différents institutions de communiquée pour l'échange de divers informations concernant les apprenants.

Mots clés : SOA , eFormation ,Système de communication ,service web.

Abstract :

This project is part of the implantation of web application dedicate to administrative training that adapted to any institutions satch as height school ,primary college .. And a layer of communication that allow for a different institutions to communicate to exchange information about a various learners.

Keyword: SOA, eFormation, Communication system Web service.

Remerciements

Nos remerciements vont à :

Allah le tout puissant, qui nous a aidés à mener à terme ce modeste travail.

Nos remerciements vont aussi à notre promoteur DR Djamel Bennouar, pour l'honneur qu'il nous a fait en acceptant d'être notre promoteur et pour tous les conseils et l'aide qu'il nous a donnés, afin de pouvoir mener à terme ce travail.

Nous tenons à remercier l'ensemble des enseignants du département d'informatique pour avoir assuré notre formation, le long des cinq années d'études, et pour nous avoir transmis leur savoir sans réserve.

Nous tenons aussi à remercier les membres du jury pour l'honneur qu'ils nous font en acceptant de juger ce travail.

Enfin, nous tenons à remercier tous nos amis et collègues pour leur soutien moral tout au long de la préparation de ce mémoire, et plus particulièrement Melle hadjer Yekhlef.

Djellali Mohamed

Tidafi Salim

Dédicaces

Je dédie ce travail à mes parents pour leur patience, leur soutien et surtout leurs sacrifices, merci infiniment et que dieu vous garde.

À mes frères Nadjib, Aniss et Wassim.et toute la famille TIDAFI

À mon binôme Djellali Mohamed et toute sa famille.

À tous mes amis et collègues de la promotion surtout Benmouffok Aissa, Sadek, Rahim, oussama, abou bakr, ninas,Hatem, lotfi.

A tout ceux qui mon aider de prêt ou de loin pour la réalisation de ce travail

Salim

Dédicaces

Je dédie ce travail à mes parents pour leur patience, leur soutien et surtout leurs sacrifices, merci infiniment et que dieu vous garde.

A Mes grands parents.

À mes frères samir et M'hamed.

À mes sœurs Sakina et Fatima

À mon binôme TIDAFI salim et toute sa famille.

À tous mes amis et collègues de la promotion surtout Benmouffok Aissa, Sadek, Rahim, abou bakr, ninas,Hatem, lotfi.

A tout ceux qui mon aider de prêt ou de loin pour la réalisation de ce travail.

mohamed

Table des matières

Partie II : Conception

Chapitre 3 :Conception de l'aspect métier de système

Chapitre 4 :conception de l'aspect communication de système

Partie III : Réalisation

Chapitre 5 : Implémentation de l'application et son test

Table des figures

Liste des tableaux

Introduction et Motivation

La généralisation de l'exploitation du réseau internet et la disponibilité au niveau des organisations de réseaux Intranet ont eu un impact important sur le développement des logiciels. C'est ainsi que nous observons depuis plusieurs années un intérêt de plus en plus croissant pour la réalisation de logiciel centré sur le Web, pour rendre plus aisée cette tache de conception réalisation de logiciels centrés sur le web, plusieurs langage et plateformes ont été défini.

Parmi les applications qui ont été conçues (ou parfois reconçues) pour exploiter les facilités d'Internet, notamment le web, il y a les logiciels orientés vers la gestion des divers aspects liés à la formation. C'est ainsi que plusieurs logiciels de gestion de la scolarité ont été conçu et parfois réalisé. Le nombre de logiciels ayant été réellement exploité par les organismes de formation ne semble pas être important. A titre d'exemple le département d'Informatique, censé être le département consommateurs des dernières innovations en terme de logiciels, ne dispose pas actuellement une solution permettant de mettre les diverses fonctionnalité de la gestion de la scolarité sur Internet. Le département d'Informatique de l'Université Saad Dahlab de Blida disposait au départ d'un logiciel réalisé qui n'était pas vraiment flexible. A titre d'exemple, les enseignants ne pouvaient pas donner leur note sous forme numérique. Les notes étaient soumise à un opérateur sur papier et l'opérateur se chargeait de faire la saisie avec tous les risques d'erreur du principalement à l'inattention. Une fois les informations saisies, il fallait imprimer les notes et donner à l'enseignant ces notes sur papier pour les vérifier. Ce processus est réitéré jusqu'à ce que les notes imprimées soient déclarées sans erreur par l'enseignant. Cette déclaration ne voulait nullement dire que l'information déclarée comme correcte par l'enseignant ne contenait pas d'erreur. D'ailleurs rare les situations ou les étudiants n'introduise pas un recours écrit pour corriger les note. Il faut noter que ce logiciel était orienté vers une scolarité dite classique.

Depuis quelques années, le département d'informatique fut doté d'un autre logiciel de gestion de scolarité. Celui-ci semble être plus intéressant que le premier en terme de fonctionnalité. Cependant, le processus de vérification de note est toujours le même. De plus ce logiciel est spécifique au système LMD. Le changement des critères de passage d'une année à une autre pourrait nécessiter une mise a jour du logiciel. Ce logiciel peut être exploité en réseau, selon les dires, mais en réalité il n'est exploité qu'en monoposte. De ce fait, nous remarquons que pour gérer plusieurs formations par plusieurs personnes (chaque personne gère une ou deux formation), il est nécessaire d'installer pour chaque personne une version du logiciel

Ce qui est remarquable c'est que ces deux logiciels ne donnent aucune garantie concernant la validité de l'information. C'est ainsi qu'il est fort probable

qu'une note soit modifiée intentionnellement ou par erreur après les délibérations. Ces deux logiciels ne donnent aucun accès direct ou indirect aux enseignants dans le processus de saisie de note ou de suivi des étudiants.

Problématique :

Plusieurs logiciels traitant de la gestion de la scolarité ou des aspects liés à la scolarité ont été proposés. Rares sont ceux qui sont fonctionnels et plus rare sont ceux qui sont accessibles à travers Internet ou fonctionnent dans le contexte d'un intranet.

Les logiciels de gestion de la scolarité ou de la formation en général sont souvent figés pour ne supporter qu'un seul type de formation (système classique à l'Université, système LMD, lycée, centre de formation professionnel etc..) et une seule approche d'évaluation, notamment l'évaluation à la fin d'une période de formation.

Les systèmes actuels ne sont pas dotés d'un système de communication leur permettant de coopérer dans le contexte du suivi des étudiants. Même dans le contexte d'un même type de formation, ces logiciels ne sont pas capables de s'échanger des informations sur les étudiants tels que le vrai cursus suivi, les notes, les appréciations d'enseignant et les états disciplinaires. Si cette situation n'est pas supporté par des logiciels adressant le même type de formation, que dire alors de la communication entres des logiciels traitant de formation différentes.

Objectifs :

La situation actuelles des logiciels de gestion de formation est principalement du à une approche très cloisonnée dans le contexte de la conception d'une application informatique. Ainsi, la conception d'une gestion de scolarité pour un lycée se basera directement sur les concepts très liés au monde du lycée. De même pour un logiciel dédié au primaire, ou un logiciel dédié au LMD.

Dans la réalité ces logiciels traite en grande partie des concepts similaires mais ayant juste des noms distincts dans chaque institution. A titre d'exemple au niveau universitaire on parle d'étudiant, au niveau primaire, moyen et secondaire nous parlons d'élève, au niveau d'une entreprise de formation nous parlons de stagiaire. En réalité les étudiants, ou élève ou stagiaire sont des apprenants.

L'étude de la scolarité dans les diverses institution de formation public ou privé nous a mené à la conclusion suivante : Il y a beaucoup de similarité entre les diverses formations. La pluparts des concepts sont les même et souvent ce n'est que l'appellation qui change. Cette remarque a mené à l'idée suivante : la formation peut être considérée comme un domaine d'application et les divers logiciels sont des cas particuliers d'applications de formation. Cette idée nous mène à l'approche

suivante : Réfléchir de manière globale à la gestion de la scolarité, que nous appellerons dorénavant gestion de formation. Cette réflexion devra nous se concrétiser par la conception d'une application hautement paramétrable. L'instanciation d'une application pour une institution particulière consistera à fixer les paramètres à l'installation. C'est en fait une sorte de dérivation d'une application particulière à partir d'une application générale. L'application générale est basée sur les concepts communs et un vocabulaire commun et homogène

La paramétrisation permettra de personnaliser le vocabulaire commun de l'application et de ce fait donner, au niveau de chaque institution de formation, le libellé adéquat à chaque élément du vocabulaire commun. Si le mot *apprenant* fait par partie du vocabulaire commun, dans un lycée, le libellé associé est élève.

Plus encore, dans le contexte de l'existence de modèle de formation reconnu et largement utilisé, nous pouvons citer à l'installation le 9modèle de comportement de l'application. A titre d'exemple de modèles, nous avons le primaire, le secondaire, le moyen, la formation universitaire classique, le LMD etc..

L'utilisation d'un vocabulaire homogène permettra non pas à des applications traitant le même modèle de formation de communiquer, mais à des applications gérant des modèles de formations distincts de communiquer entre elle. Ainsi il serait possible à l'université de tracer le cursus pédagogique d'un étudiant à partir du primaire et voir quels sont les stages et formations spécifique qu'il a suivi. Ceci sera rendu possible grâce au système de communication de l'application. Un seul système de communication sera ainsi réalisé et qui s'adaptera à n'importe quel modèle vu qu'il se base sur un vocabulaire commun et homogène.

L'objectif de notre travail est donc de réaliser une application dédié à la gestion des formations. Cette application doit être hautement paramétrable pour permettre d'adapter son installation à n'importe quelle institution de formation. Cette application sera dotée d'une couche de communication qui permettra aux diverses instances (les applications installées dans les diverses institution) de communiquer entre elles pour l'échanges de diverses informations concernant les apprenants.

Organisation du mémoire :

Le présent mémoire comporte cinq chapitres. Le premier chapitre est dédié à l'étude des techniques de communication. Cette étude permettra de déterminer la technologie de communication à adopter pour la partie communication de notre application. Le chapitre 2 sera consacré à l'étude et la présentation de quelques logiciels de gestion de la scolarité ou de formation. Le chapitre 3 sera consacré à la conception de notre système pour l'aspect métier pur (gestion des formations). Le 4ème chapitre traitera la conception de Système de communication. Le chapitre 5 sera consacré à la description de l'implémentation de notre application et son test.

Etat de l'art sur les techniques de communication

1. Introduction :

Le réseau des réseaux (internet) a pris, durant ces dernières années une évolution considérable dans le domaine informatique, depuis la petite entreprise jusqu'au grand groupes internationales. Cette évolution est du car les besoins ont changé et la migration vers une architecture orienté service qui permet l'interconnexion et l'échange des informations entre les différente applications. L'ensemble de ces applications qui coopèrent entre eux via un système de communication forme ce qu'on appelle un système répartie.

Il existe plusieurs standards permettant la réalisation de cette tache notamment les architectures basées sur CORBA, RMI de JAVA et LES SERVICES WEB.

L'objectif de ce chapitre est de présenter ces différentes architectures et les technologies utilisées par ces derniers, ainsi qu'une comparaison pour choisir la meilleure approche qui s'adapte a notre projet.

1.2 Architecture oriente service(SOA) :

1.2.1 Définitions

Il existe plusieurs façons de percevoir et de définir une Architecture Orientée Services. Voici une liste de définitions proposées et issues de plusieurs sources. Ces définitions sont intéressantes puisqu'elles illustrent plusieurs points de vue sur la SOA.

« L'architecture orienté service permet de publier (sur Internet, Extranet ou encore sur Intranet) un ensemble d'offres (existant ou non) réalisant un service bien précis (par exemple un service pour crypter des données, un autre pour authentifier une personne, etc...).Une application dans ce cadre appelle ces différents services quand elle en a besoin et peut être elle-même déployée comme un nouveau service (remplissant alors un plus grand rôle dans l'architecture métier) » [1].

« La SOA est un style d'architecture qui permet la réorganisation du Système d'Information. Elle permet l'encapsulation des fonctionnalités d'un système d'information en un ensemble de services faiblement couplés appartenant à la fois au niveau métier et au niveau technique de l'entreprise. Les services munis d'un contrat d'utilisation et d'une interface de description seront publiés dans des registres de services afin qu'ils puissent être invoqués par d'autres services » [2].

On se basant sur ces différentes définitions, nous définissons la SOA de la manière suivante:

Une architecture orienté service est une approche architecturale permettent la création des systèmes basé sur une collection de services développer dans différentes langages de programmation, héberger sur différentes plateformes avec divers modèles de sécurité et processus métier .

1.2.2 Les Concepts de la SOA

Dans une Architecture Orientée Services trois rôles clés sont communément identifiés :

Le fournisseur de services : Il permet l'accès à son service a travers une interface.

Le client du service : il désigne une personne, un serveur, ou une autre publication qui accède et l'invoque à travers son interface.

L'annuaire de services : il joue le rôle d'intermédiaire entre le fournisseur de service et le client .Les fournisseur y enregistrent leurs services, et les clients y cherchent le service satisfaisant leurs besoins.

L'interaction entre ces trois rôles est décrite par la Figure 4.1. Le fournisseur de services crée le service, puis publie sa description dans un annuaire de services. Cette description précise à la fois les opérations disponibles et leur mode d'invocation. Le client du service accède à l'annuaire pour effectuer des recherches afin de trouver les services désirés. Ensuite, une liaison s'établit entre le client et le fournisseur de service afin d'assurer l'invocation du service en question.

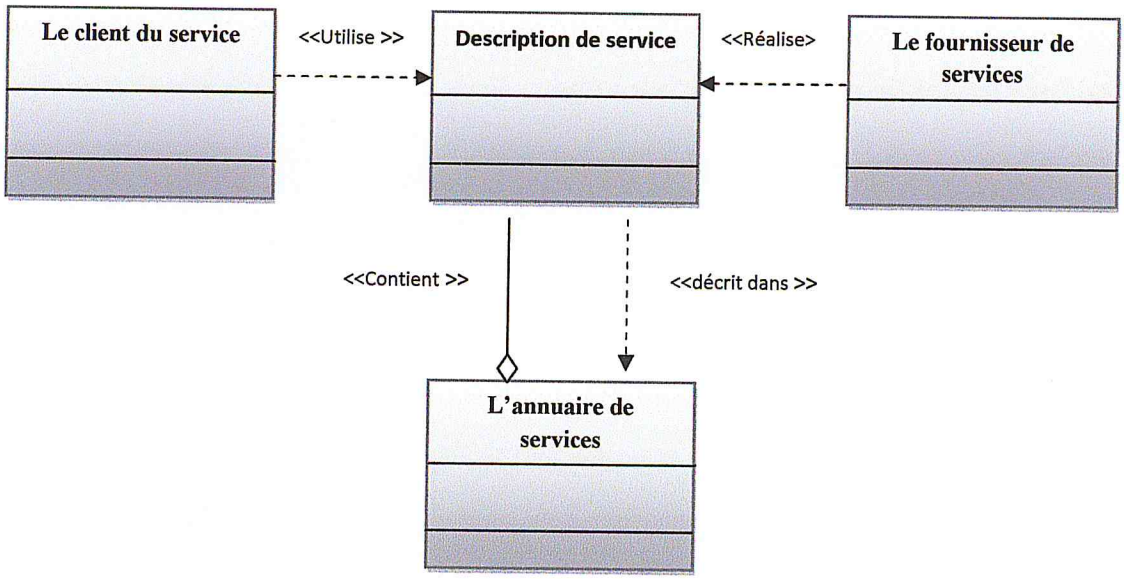


Figure 1 : Méta-modèle de l'architecture orientée service.

2.2.3 Les services web : une instance de SOA :

Le SOA repose sur différents concepts que nous avons présentés précédemment. La principale implémentation de ces concepts repose sur les **services web**.

2.2.4 Définitions :

Il existe plusieurs définitions des services web, nous adoptons dans un premier temps celle de W3C [3] :
« Un service web est un logiciel identifié par une URI (Uniform Resource Identifier), dont l'interface est publique et les liaisons sont définies et décrites en utilisant XML. L'environnement du service offre le moyen à d'autres logiciels de découvrir celui-ci. Ce service interagit avec les autres services web en respectant cette définition, donc en utilisant des messages basés sur XML acheminés par des protocoles Internet. »

Selon IBM : « "Un service Web est une collection de fonctions qui sont emballés comme une entité unique et publié sur le réseau pour une utilisation par d'autres programmes ... auto-descriptif, autonomes, des applications modulaires ... » [4].

Selon Microsoft: « "Les services Web sont un modèle très général pour construire des applications et peut être mis en œuvre pour tout système d'exploitation qui prend en charge la communication sur Internet et représentent une boîte noire fonctionnalité qui peut être réutilisé sans se soucier de la façon dont le service est mis en œuvre"» [5].

A partir de ces définitions on peut conclure que les services web sont des applications qui relient des programmes, des objets, des bases de données ou des processus d'affaires à l'aide de XML et de protocoles Internet standards. Les Web Services sont des compléments aux programmes et applications existantes, développées à l'aide de langages tel que Visual Basic, C, C++, C# (C sharp), Java ou autre, et servent de pont pour que ces programmes communiquent entre eux.

2.2.5 Les langages et protocoles utilisées par les services web :

La description fonctionnelle précédente montre l'utilisation de nombreux langages et protocoles durant le déploiement et l'invocation du service web. Ce sont ces principaux langages et protocoles que cette section va décrire.

Le protocole SOAP :

Les communications entre les différentes entités impliquées dans le dialogue avec le service web se font par l'intermédiaire du protocole SOAP (Simple Object Access Protocol) normalisé par W3C. Le protocole SOAP est une sous couche de la couche application du modèle OSI des réseaux. Le protocole applicatif le plus utilisé pour transmettre les messages SOAP est HTTP, mais il est également possible d'utiliser les protocoles SMTP ou FTP. La norme n'impose pas de choix. Le choix de l'utilisation de protocoles applicatifs, comme par exemple HTTP, est lié aux

protocoles tel que HTTP, permettant ainsi le passage sans problème à travers les différents réseaux des messages générés par l'utilisation du protocole SOAP.la figure 2 illustre le schéma de protocole SOAP [6].

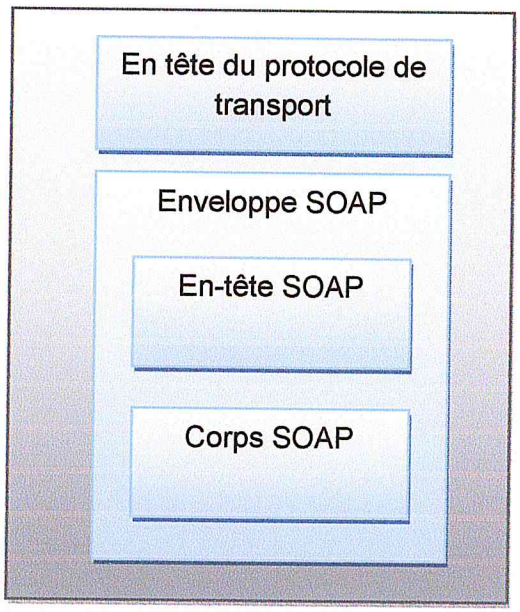


Figure 2 : Schéma d'un message SOAP.

Dans un souci d'interopérabilité, et donc dans la continuité des différents langages et protocoles issus des services web, les messages échangés lors de l'utilisation du protocole SOAP sont basés sur le méta-langage XML [9]. Ces messages comportent plusieurs parties (voir la figure 2) et sont donc encapsulés dans des protocoles de niveau applicatif.

L'en-tête du protocole de transport :

Cette partie dépend du protocole de transport utilisé. Par exemple, si le protocole HTTP est utilisé, cet en-tête contient :

- la version de HTTP utilisée,
- la date de génération de la page (qui est ici le message SOAP lui-même),
- le type d'encodage du contenu (ici, l'encodage est généralement le type text/xml).

Les messages SOAP (l'enveloppe) :

La partie principale d'un message SOAP est l'enveloppe (symbolisée par la balise envelope). Cette enveloppe (SOAP Envelope) est elle même subdivisée en deux sous-parties :

- la partie en-tête
- la partie corps du message.

Elle permet également de spécifier des environnements de noms XML utilisés dans la suite du message. L'en-tête du message SOAP (*SOAP Header*) est optionnel et extensible. Les balises XML permettant de symboliser cette partie sont **<env:Header>** et **</env:Header>.** En fait, l'en-tête permet principalement d'ajouter des informations sur le comportement que doivent avoir les différents nœuds intermédiaires, lors du traitement du message. Un nœud étant un intermédiaire SOAP, incluant le récepteur et l'émetteur SOAP, désignable depuis un message SOAP. Son rôle est de traiter l'en-tête (et d'effectuer les actions qui y sont décrites) et ensuite de transférer le résultat (le message SOAP modifié) à un autre intermédiaire (qui peut être le récepteur final).

Le corps du message SOAP :

Les données spécifiques à l'application se trouvent dans le corps du message SOAP (*SOAP Body*). Enfin, les balises symbolisant cette partie sont <env:Body> et </env:Body>. Les données doivent donc être sérialisées selon l'encodage choisi. L'utilisation du XML 1.0 à l'intérieur des blocs XML <element> permet d'envoyer absolument tous les types de documents comme par exemple des feuilles de styles, des documents XML, des images au format binaire, etc... En plus des données, cette partie peut transporter un type spécial(les messages d'erreurs *SOAP* Fault). Comme précédemment on peut ajouter un attribut dans la balise <env:Fault> permettant d'indiquer un type d'encodage des données, mais également d'autres attributs permettant la gestion de l'erreur, comme Code, Reason, Node, Role et Detail.

Le protocole UDDI :

UDDI est une norme d'annuaire de services web appelée via le protocole SOAP [7]. Pour publier un nouveau service web, il faut générer un document appelé **Business Registry**, il sera enregistré sur un **UDDI Business Registry Node**. Le Business Registry comprend 3 parties [8]:

Pages blanches : noms, adresses, contacts, identifiants des entreprises enregistrées.

Pages jaunes : informations permettant de classer les entreprises, notamment l'activité, la localisation, etc.

Pages vertes : informations techniques sur les services proposés.

Le protocole d'utilisation de l'UDDI contient trois fonctions de base :

publish *:* pour enregistrer un nouveau service,

find : pour interroger l'annuaire,

bind : pour effectuer la connexion entre l'application cliente et le service.

Comme pour la certification, il est possible de constituer des annuaires UDDI privés, dont l'usage sera limité à l'intérieur de l'entreprise.

Le langage WSDL :

Le langage de description WSDL (Web Services Description Language) a été créé dans le but de fournir une description unifiée des services web. Il se présente comme un standard actuel dans ce domaine, et il est, de plus, normalisé par le W3C. Il est issu d'une fusion des langages de descriptions NASSL (Network Acessibility Service Specification Language - IBM) et SCL (*Service* Conversation Langage - Microsoft). Son objectif principal est de séparer la description abstraite du service de son implémentation même [1].

Le langage de description WSDL se comporte donc comme un langage permettant de décrire l'interface visible (ou publiée) du service web. Il décrit, à l'aide du langage de balises XML, les différents éléments du service :

Les messages :

Ils sont composés de plusieurs parties. Ces parties correspondent, par exemple, dans le paradigme objet aux différents champs d'une structure. Ils sont décrits en XML et un type leur est associé. Les types de base XML peuvent être utilisés (entier, chaîne de caractères, etc.), mais également des types complexes définis dans un fichier WSDL (fichier identique à celui de la description du service ou externe).

Les opérations :

Cette partie associe les messages aux opérations (une opération peut être vue comme une méthode). Les différents messages étant les différents paramètres de cette méthode. Un mot clé permet de distinguer le mode :

- input : mode entrée ou paramètre de donnée
- output : mode sortie ou paramètre de résultat

Plusieurs opérations sont associées pour former un PortType, utilisé par la suite.

Les liaisons :

Elles permettent d'associer les opérations (ou un ensemble d'opérations identifiés par un (PortType) au protocole de transport de niveau inférieur. Ainsi, dans le cas d'un service web utilisant le protocole de communication SOAP :

- pour chaque opération, on définit le type d'échange utilisé (mode document ou mode XMLRPC)
- pour chaque message (composant les opérations), on décrit l'espace de nom associé au type de message à transporter.

Le service :

Le service lui-même est décrit dans la partie **service** de cette description WSDL. Le mot clé **port** permet d'associer des adresses Internet (URL) à chaque PortType : ces adresses seront utilisées par le client pour l'invocation du service. Il est également possible d'ajouter des informations permettant de trouver une

documentation pour le service. C'est également cette partie qui peut être étendue par un autre langage que WSDL.

Invocation d'un service web:

Le processus d'implémentation d'un service web est similaire à celle de toute application distribuée utilisant la technologie CORBA ou RMI. Les étapes les plus importantes de l'invocation d'un service web sont les suivantes (voir figue 1.2) [2].

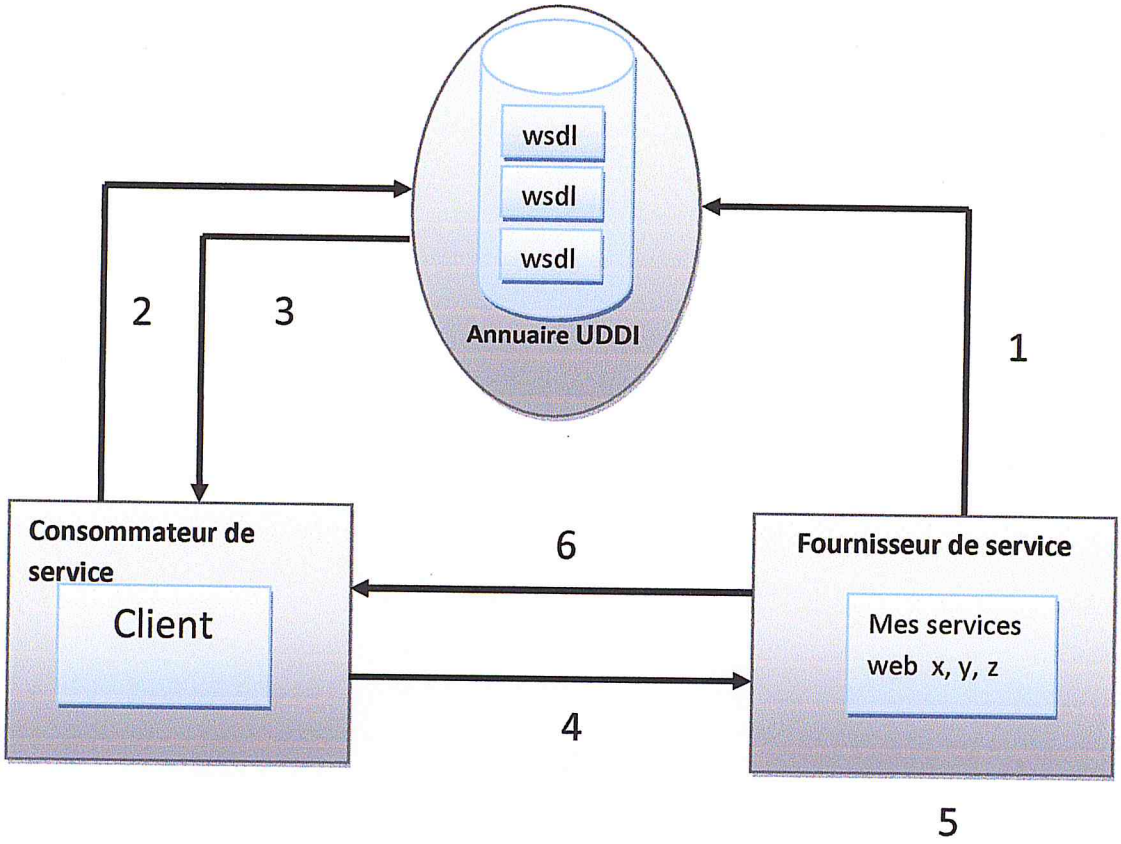


Figure 3 : Les étapes d'invocation d'un service web [11].

1. Le fournisseur de service se charge de l'enregistrement et de la publication des services auprès d'un serveur UDDI, et cela en envoyant un message SOAP à l'annuaire UDDI. Ce message regroupent la localisation du service, la méthode d'invocation (et les paramètres associés) ainsi que le format de réponse. Toutes ces informations seront formalisées ensuite à l'aide de WSDL.
2. Un client désirant consulter un service, interroge en premier lieu le serveur UDDI dont il connaît l'adresse afin de savoir quels sont les services disponibles correspondant à ses besoins. Le serveur lui retourne la liste des possibilités parmi lesquelles il sélectionne une. A ce stade, l'utilisateur ne possède qu'une URL (Uniform Resource Locator) identifiant le service sélectionné.
3. Le client de service récupère ensuite une interface WSDL qui contient la description de service, qui lui permet de savoir comment interagir avec celui-ci.

4. le client de service fait appel à la méthode distante sous forme d'une requête SOAP dans laquelle seront inclus les paramètres fournis par l'utilisateur. Ces paramètres seront empaquetés grâce à la méthode standard de présentation des données, ce qui permet d'assurer la compatibilité entre machines. Cette requête est ensuite émise vers l'URL désignant le service web.
5. Sur la machine hébergeant le service, la requête est réceptionnée puis ouverte par un Stub.
6. Une fois la requête est comprise, une réponse SOAP est construite puis émise en direction de l'expéditeur initial.

1.3 Architecture CORBA :

CORBA est une architecture pour la création et la gestion des applications orientées objet distribuées sur un réseau. La spécification de CORBA est indépendante d'une implémentation et quelconque par rapport du langage de programmation et du système d'exploitation. La figure suivante synthétise cette architecture [12].

1.3.1Différentes composantes de CORBA

1. IDL :

Son but est de permettre la description des services fournis par un objet CORBA, de manière indépendante du langage de programmation utilisé pour son implémentation. Un compilateur d'IDL permet alors de traduire une interface IDL vers un langage de programmation particulier. Cette étape va générer des portions de code parmi lesquelles le *stub* et le *skeleton*, qui assurent de liaison entre l'environnement d'exécution, l'implémentation proprement dite de l'objet serveur et le client [13].

2. ORB

L'ORB constitue l'élément central de l'architecture CORBA. C'est en effet lui qui transmet les requêtes des applications vers les objets concernés, puis qui achemine les éventuels résultats vers l'application cliente [13].

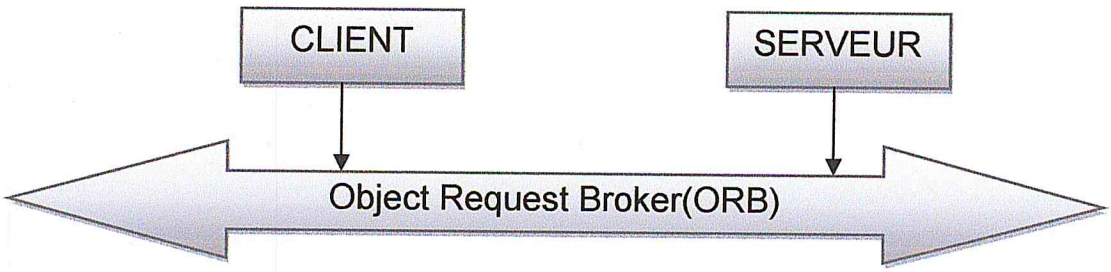


Figure 4: Le modèle d'exécution de CORBA

L'ORB est constitué de deux parties :

- Un ORB Core qui fournit la base pour les représentations des objets et la communication des requêtes ;
- d'un ensemble de composants qui fournissent un ensemble d'interfaces.

1.3.2 Interfaces :

Les interfaces permettent de définir les caractéristiques et le comportement du ou des objets CORBA qui vont être utilisés pour exécuter les requêtes [13].
On distingue :

• **les interfaces d'invocations dynamiques clientes (Dynamic Invocation Interface DII)** :

Est l'interface d'invocations dynamiques permettant de construire dynamiquement des requêtes vers n'importe quel objet CORBA sans générer/utiliser une interface SII [13].

• **l'interface d'invocation dynamique du côté serveur (Dynamic Skeleton Interface DSI)** :

Est l'interface de squelettes dynamiques qui permet d'intercepter dynamiquement toute requête sans générer une interface SSI [13].

• **les Stub clients** pour les invocations statiques du côté client [12].

• **les skeletons** pour les invocations statiques du côté serveur [12].

• **l'adaptateur d'objet (Object Adapter OA) :**

Est l'adaptateur d'objets qui s'occupe de créer les objets CORBA, de maintenir les associations entre objets CORBA et implantations et de réaliser l'activation automatique si nécessaire [12].

• **Répertoire d'interface** (Interface Repository IFR) :

Est le référentiel des interfaces contenant une représentation des interfaces OMGIDL accessible par les applications durant l'exécution.

Mécanisme d'invocation:

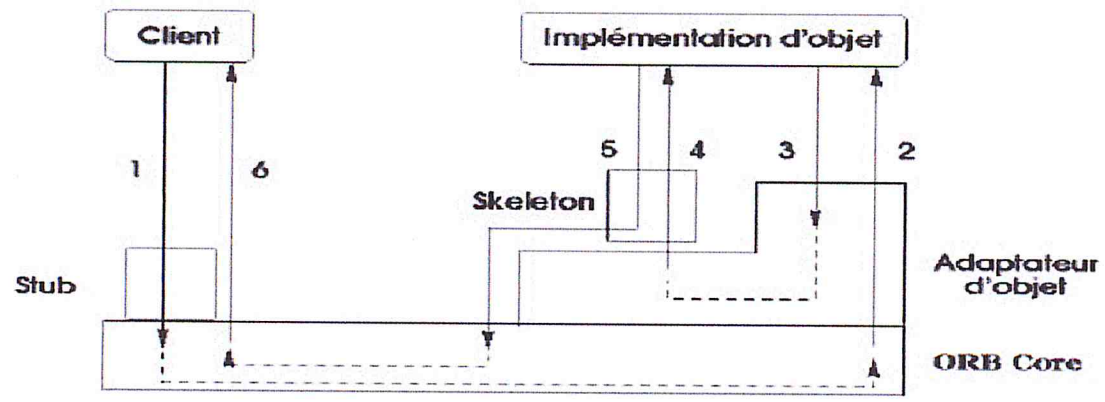


Figure 5 Les mécanismes de CORBA.

1. Le client invoque une méthode au travers du stub client ;
2. L'ORB transmet la requête à l'adaptateur d'objet qui active l'implantation d'objet ;
3. l'implantation objet avertit l'adaptateur d'objet qu'elle est prête à recevoir la requête ;
4. l'adaptateur d'objet transmet la requête à l'implantation via le skeleton ;
5. l'implantation d'objet retourne le résultat ou l'exception à l'ORB ;
6. le résultat est éventuellement retourné au client si la méthode le spécifie.

1.4 Architecture RMI :

1.4.1 Définition :

RMI (Remote Method Invocation) est une architecture de type RPC, dédié à la plate-forme Java. Elle est de ce fait orientée objet et supportée par toutes les environnements pour lesquels il existe une Machine Virtuelle Java. RMI ne définit pas de langage IDL, mais réutilise les interfaces propres au langage Java [14].

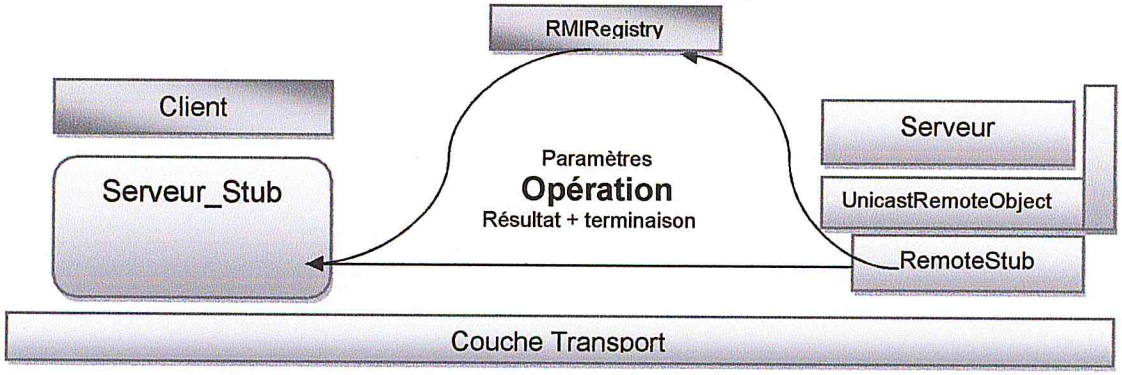


Figure 6 : Architecture RMI

1.4.2 Principe général du fonctionnement de Java RMI :

Le compilateur RMI :

Un compilateur RMI (***rmic***) est utilisé pour générer les souches côté client (***stub***), ou côté client et serveur (***skeleton***).

Le service *RMIRegistry* :

Est le service de nommage propre à RMI. Il permet aux objets distants d'être référencés dans un annuaire, et aux clients de retrouver les références des objets distants à partir de leur nom de service associé.

1.4.3 Mécanisme d'invocation:

Phase de Définition/Génération :

La déclaration des services distants s'effectue à l'aide d'une interface en langage Java qui étend, de manière directe ou indirecte, l'interface JAVA.RMI.REMOTE. Un objet serveur doit implémenter cette interface pour être utilisable avec RMI. De manière optionnelle, il peut également étendre la classe unicastRemoteObject, qui propose quelques méthodes d'automatisation des tâches de mise en liaison avec le bus RMI. A partir de cette implémentation, le compilateur RMIC génère un STUB qui sera utilisé côté client.

Mise en place d'un service et invocation :

Lorsqu'une instance de Serveur est créée, celle-ci doit être exportée sous la forme d'une *référence distante*. L'objet UnicastRemoteObject propose des méthodes dédiées à cette tâche. Lors de l'enregistrement du serveur dans la RMIRegistry, la référence distante ainsi créée est transmise, accompagnée du nom utilisé pour référencer le service. Pour accéder au serveur, le client fait également appel à la RMIRegistry, à laquelle il transmet le nom du service sous lequel le serveur est enregistré. La référence distante est alors transmise au STUB, qui se charge de transmettre les invocations du client vers le serveur. L'encodage et le décodage des données sont réalisé par sérialisation, mécanisme standard à la plate-forme Java pour la transmission d'objets à travers des flux, de quelque nature qu'ils soient.

1.5 Comparaison:

Fonction	Java RMI	CORBA	Services Web
Description	Interfaces en Java	IDL-CORBA	WSDL
Annuaire	rmiregistry	CORBA Services	UDDI
Protocole de Communication	JRMP	GIOP	SOAP Message
Représentation de donnée	Sérialisation	CDR	XML
Protocole de transport	RMI	HTTP	HTTP

Table 1 : Un comparatif entre RMI, CORBA et les Services Web

1.5.1 RMI et web services :

Java RMI est un intergiciel spécifique à la plate-forme Java. Les interfaces sont définies en Java. L'annuaire est très limité et se résume à un service de nom. Le protocole de communication JRMP (Java Remote Method Protocol), la représentation d'informations et le protocole de transfert sont spécifiques à Java.
En fait, Java RMI est une adaptation de RPC au monde Java [14].
Dans [15], l'auteur présente une étude détaillée de performance entre Java RMI et les Services Web. Cette étude a démontré que l'utilisation de Java RMI est 8.5 plus performante que l'utilisation de services Web sur un réseau local.

Cependant, l'auteur remarque que sur l'Internet, l'utilisation de Java RMI demande souvent une technique de tunneling comme HTTP-to-port, HTTP-to-CGI et HTTP-to-Servlet. L'effort pour déployer et configurer Java RMI et les composants de tunneling est plus significatif que dans le cas de services Web. De plus, Java RMI et les techniques de tunneling ont présenté une performance inférieure aux services Web. L'utilisation de Java RMI et HTTP-to-port a été quatre fois moins performante que les services Web, Et l'utilisation de Java RMI et HTTP-to-servlet a été trois fois moins performante que les services Web.
Selon les résultats obtenus, l'auteur conseille l'utilisation de services Web au détriment de Java RMI avec la technique de tunneling, et il recommande l'utilisation de Java RMI (sans tunneling) pour les scénarios qui n'ont pas des problèmes avec les pares-feu ou qui demandent plus de performance que d'interopérabilité. Dans [15], l'auteur a montré que, dans certains contextes et avec l'utilisation de techniques d'optimisation, les services Web peuvent être plus performants que Java RMI (sans tunneling).

1.5.2 CORBA et service web :

CORBA peut supporter plusieurs types d'applications distribuées, mais n'est pas efficacement adaptable à l'Internet. Il définit la notion de service, ainsi qu'une description d'interface indépendante de la plate-forme, c.-à-d. IDL-CORBA.
Le protocole GIOP est un protocole conçu pour être adapté à d'autres protocoles spécifiques au domaine .CDR est une représentation binaire d'information. Ceci rend l'interopérabilité réalisable, mais sans extensibilité. IIOP est le protocole de transport défini dans GIOP pour prendre en charge le TCP/IP. D'une part, à travers les services CORBA, un annuaire pour les services peut être fourni [16].
De plus, les services Web présentent des avantages significatifs par rapport à CORBA. Le WSDL contient deux parties, l'une abstraite et l'autre concrète. La partie abstraite contient le contrat du service, et la partie concrète contient les liaisons (bindings) et la localisation du service. IDL-CORBA ne fournit par contre que la partie abstraite, c.-à-d. le contrat du service [16].
SOAP est un protocole de communication plus riche que IIOP. Comme SOAP est basé sur XML, il permet la création de types de données plus flexibles et non prédéfinis. De plus, SOAP peut être étendu pour répondre à des nouvelles exigences (par exemple, sécurité, transaction) sans modifier la normalisation de SOAP. En fait, SOAP est plus simple pour être adapté aux besoins d'un contexte [14].
Ainsi comme les autres technologies de services Web, UDDI peut être étendu pour répondre à des nouvelles exigences. Par rapport à CORBA, UDDI fournit en plus la possibilité de classer un service par ses caractéristiques techniques.
XML est un mécanisme plus riche que CDR pour représenter des données. XML a été conçu pour être extensible et compréhensible par les humains et aussi par les ordinateurs. En contrepartie, CDR est une représentation binaire qui n'est pas extensible et ni compréhensible par les humains [14].

A partir de cette comparaison nous avons choisi d'utiliser les services web puisque ces derniers assurent :

- l'indépendance de plate-forme
- un environnement universel pour les systèmes d'information distribués
- l'utilisation de plusieurs protocoles de transfert (par exemple HTTP, SMTP et FTP)
- le codage des messages en XML
- un comportement compatible aux pare-feu
- la localisation par URI (Uniform Resource Identification)

1.6 Conclusion :

Dans ce chapitre nous avons montré les différentes architecture utilisé pour l'interconnexion des éléments d'un système distribué (CORBA, RMI et les SERVICES WEB).d'après une étude comparative nous avons décidé d'utiliser les services web qui sont les plus utilisés actuellement, puisque ils offrent plusieurs avantage en terme d'indépendance de plateformes et l'utilisation d'un standard commun pour le codage et le formatage des informations(XML).

Etat de l'art sur les systèmes de gestion de formation

2.1 Définition de la formation :

Une formation est un ensemble de connaissances acquises, en principe avant d'entrer dans la vie active en tant que élève dans un école primaire, un CEM ou un lycée, étudiant dans un département université ou un stagiaire dans un centre de formation professionnel

2.2 Un système de gestion de la formation :

C'est un système qui assiste a la conduite des formations présentai dans les centres d'enseignements, il assure le pilotage de toutes les aspects liées a la gestion des formations qui peut être :

- La gestion des notes
- Possibilité de saisir les appréciations
- Gestion enseignants
- Gestion des inscriptions des apprenants
- Gestion des modules
- Les critères de passage
- La consultation des notes
- Gestion des utilisateurs

Dans la section suivante on va détailler quelques projets déjà réalisés dans ce qui concerne la gestion de la formation :

a. Projet S.E.E.S :

S.E.E.S : Suivi des Etudiants et des Enseignements Supérieurs qui a été exploité dans les établissements universitaires à l'échelle nationale depuis l'année universitaire 2007/2008.[25]

Fonctionnalité :

Assurer la gestion et le suivi d'une année universitaire par :

- Les inscriptions administratives des étudiants,
- L'articulation des unités d'enseignements et des matières par semestre et par année,
- Les différents traitements (saisie de notes, calcul de résultats, délibérations, etc…),
- L'édition de plus de soixante dix (70) états différents,
- Des éditions statistiques,
- Des analyses de synthèse à diverses échelles (département, faculté, établissement, conférences régionales ou conférence nationale)
- Le passage automatique à l'année universitaire suivante.

b. Projet d'un site web dynamique pour des activités pédagogiques et scientifiques de la DPGR (Direction de la Post Graduation et de la Recherche) :

Il a été réalisé au niveau de l'école supérieur de d'informatique, qui consiste a mettre en place un système de gestion pour le suivi des activités pédagogiques et scientifiques [26].

Fonctionnalités :

- Facilité la couverture de l'ensemble de cycle d'un poste -graduant
- Effectuer le suivi pédagogique de l'école doctorale.
- Disposer à tout moment d'une trace de toutes les activités scientifiques et pédagogiques organisés (formations, stages et congés scientifique) et des projets de recherche menés au sein de la direction
- Prendre en charge les différentes postes graduation existants depuis la création de l'école

c. PWGESCO :

Est un programme de gestion de scolarité spécialement conçu pour l'enseignement primaire en Algérie. il englobe aussi bien les aspects administratifs que pédagogiques avec un suivi détaillé de toutes les missions assumées par le directeur d'école [27].

Fonctionnalité :

- Paramétrage complet des matières enseignées.
- Calcul instantané des moyennes mensuelles trimestrielles et annuelles.
- Comparatif très pertinent des résultats scolaires.
- Disponibilité immédiate des informations statistiques.

d. WGESCO :

Est le logiciel pour une gestion de scolarité. Il s'adapte à tous les types d'enseignements (Moyen - Secondaire -Technique) et répond correctement aux exigences de tout responsable soucieux de moderniser et d'optimiser ce volet important de la gestion des établissements. Durant la conception et le développement de WGESCO, un soin particulier a été accordé à la facilité de mise en œuvre et d'utilisation afin d'assurer une prise en main rapide et de mettre à la portée du chef d'établissement une maîtrise quasi permanente de tous les aspects administratifs et pédagogiques de la scolarité [27].

2.3 Critique :

A partir de la section précédente, nous avons remarqué les insuffisances des systèmes de gestion de formation qui sont les suivantes :

- La communication et les échanges d'informations utilisant les technologies de web est rare dans les projets réalisés se qu'il engendre une difficulté d'avoir le cursus de l'apprenant dans les institutions d'origine dans le cas d'un transfère par exemple
- Manque d'un mécanisme en ce qui concerne le processus de validation des notes
- Absence de sécurité l'ors des échanges d'informations (configuration de http au niveau de serveur web)
- Les projets réalisés sont destinés a un type bien précis d'institution ou parfois pour une institution spécifique

2.4 Suggestion :

Notre suggestion consiste à réaliser un système ou un modèle générique pour la gestion administrative des formations c'est-à-dire une procédure claire peut être adaptée à chaque type d'institution (primaire, CEM, lycée, université, centre de formation) dont le but est de fournir les services suivants :

- gestion des Inscriptions des étudiants ;
- gestion des notes : consulter, modifier, valider des notes ;
- Ouverture (inscription) automatique d'une phase ;
- Fermeture (délibération) automatique d'une phase ;
- Demande de documents officiels (certificat de scolarité, relevé de note…) ;
- gestion des étudiants et des enseignants.

Le deuxième objectif consiste à intégrer les techniques de communications pour relier les différentes institutions sur le l'échelle national et facilité les échanges des informations entre eux, dans ce qui concerne :

- Retrait de relevé de note et l'état disciplinaire à partir des différentes institutions appartiennent au cursus de l'étudiant lors d'un transfert ;
- Retrait de certificat de scolarité à l'institution actuelle dans le cas d'un centre professionnel.

Conception de l'aspect métier de système

3.1 Introduction :

Après avoir achevé notre étude de l'existant, nous abordons l'étape suivante qui consiste à concevoir notre système.

L'objectif de cette étape est de déterminer de façon détaillée et précise ce que le système devrait faire afin de répondre aux objectifs établis lors de l'analyse de l'existant. La réussite d'un projet repose sur cette phase, du moment que si une erreur s'y glisse, elle aura des répercussions néfastes sur le système [17].

Il existe plusieurs outils de conception, et nous avons choisi la modélisation orientée objet avec UML qui est un langage unifié et puissant.

Le choix de cet outil a trouvé origine dans le fait qu'il est caractérisé par la stabilité de la modélisation par rapport au monde réel et la réutilisation des objets [18].

3.2 Modélisation des besoins :

3.2.1 Identification des acteurs :

Un acteur représente un rôle joué par une entité externe (utilisateur humain, diapositive matériel ou autre système) qui interagit directement avec le système étudié [18].

Suivant les rôles que peuvent jouer des entités externes, nous avons identifié les acteurs suivants :

1. Apprenant : tous les apprenants de centre de formation.
2. Enseignant : tous les enseignants de centre de formation.
3. Service scolarité : tous les membres de la scolarité.
4. Administrateur : la personne chargé d'administration de système d'EFormation.
5. Visiteur : toute personne souhaitant consulter le système
6. Responsable de la formation : la personne qui gère la formation.
7. Chef de centre de formation : la personne chargé de gérer le centre de formation.
8. Tuteur : la personne qui surveille un apprenant ou un certain nombre d'apprenants il peut êtres un enseignant ou un parent d'apprenant.

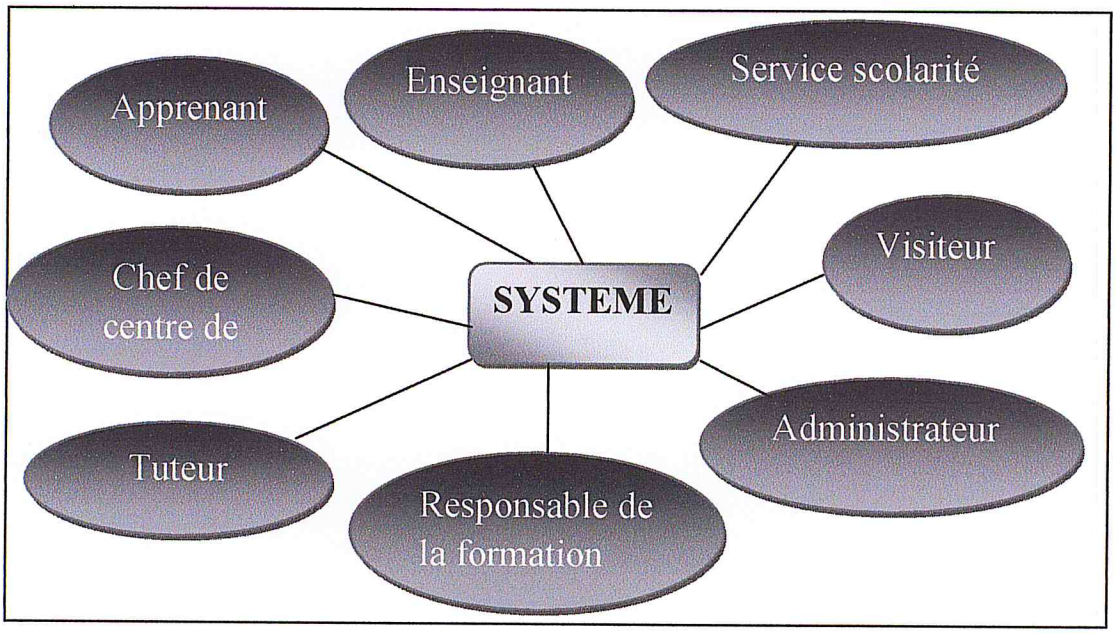


Figure 7 : diagramme de contexte

3.2.2 Identification des cas d'utilisation :

Définition : un cas d'utilisation (use case) représente un ensemble de séquence d'action réaliser par le système et produisant un résultat observable intéressant pour un acteur particulier. Un cas d'utilisation modélise un service rendu par le système, il permet de décrire ce que le future système devra faire, sans spécifier comment il le fera .l'ensemble des cas d'utilisation doit décrire exhaustivement les exigences fonctionnelles du système [19].

Le tableau suivant résume les cas d'utilisations de notre système :

N°	Cas d'utilisation		Acteur
01	authentification	Connexion	Touts les acteurs sauf le visiteur
		déconnexion	
02	Afficher liste des comptes utilisateur		Administrateur de système
03	Gestion des profils utilisateurs	Ajouter, supprimer, modifier, consulter liste profils	Administrateur de système
		Affecter des actions au profil	
04	Gestion de formation	Ajouter nouvelle formation	Chef de centre de formation
		Afficher liste des formations par type	
		Ajouter phase de formation	Responsable de la formation
		Ajouter unité de formation	
		Ajouter module	

05	Gestion des types de contrôles	Ajouter un type de contrôle	Service scolarité
		Affecter type de contrôles aux modules	
0 6	Gestion des notes de contrôle	Ajouter note	Service scolarité, Enseignant
		Afficher liste des notes des apprenants	Service scolarité, Enseignant, apprenant
		Modifier note	Enseignant, Responsable de la formation, Service scolarité
		Valider une note	
07	Gestion des promotions	Ajouter, modifier, consulter la liste des promotions	Service scolarité
		Affecter des apprenants dans des promotions	
08	Gestion des remarques	Ajouter une remarque	enseignant
		Afficher liste des remarques de l'apprenant	Enseignant, service scolarité, tuteur
09	Gestion des groupes	Ajouter un groupe	Service scolarité
		Affecter des apprenants aux groupes	
10	Gestion des sections	Ajouter, modifier, consulter liste des sections	Service scolarité
11	Gestion des apprenants	Inscrire un apprenant	Service scolarité
		Supprimer un apprenant	
		Affecter un apprenant au PSG (promotion, section, groupe)	
		Consulter la liste des apprenants	Service scolarité, Enseignant, apprenant
12	Gestion des enseignants	Ajouter un nouveau enseignant	Service scolarité
		Affecter un module a un enseignant	
		Affecter une section et un groupe a un enseignant	
13	Ajouter une nouvelle année pédagogique		Service scolarité
14	Ouverture d'une phase de formation		Service scolarité
15	Clôture d'une phase de formation (délibération)		Service scolarité
16	Afficher résultat de la délibération		Service scolarité

Tableau 2 : description des cas d'utilisations

3.2.3 Description détaillée des cas d'utilisations du système :

Cas d'utilisation N°1 « Authentification » :

Description de sommaire	
Titre	Authentification
But	Permettre à un utilisateur de se connecter au système
acteurs	Tous les acteurs de système sauf le visiteur
Description des enchainements	
Pré-conditions	L'utilisateur saisit ses droits d'accès (nom d'utilisateur et mot de passe).
Enchainements	1. L'utilisateur demande une connexion au système. 2. le système demande le login et le mot de passe 3. L'utilisateur entre le login et le mot de passe puis il valide. 4. Le système vérifie l`existence de l`utilisateur. 5. Le système ouvre une session et affiche l'espace personnel
Post-conditions	L'utilisateur se connecte au système et peut ainsi accéder aux rubriques correspondantes à son profil.

Diagramme :

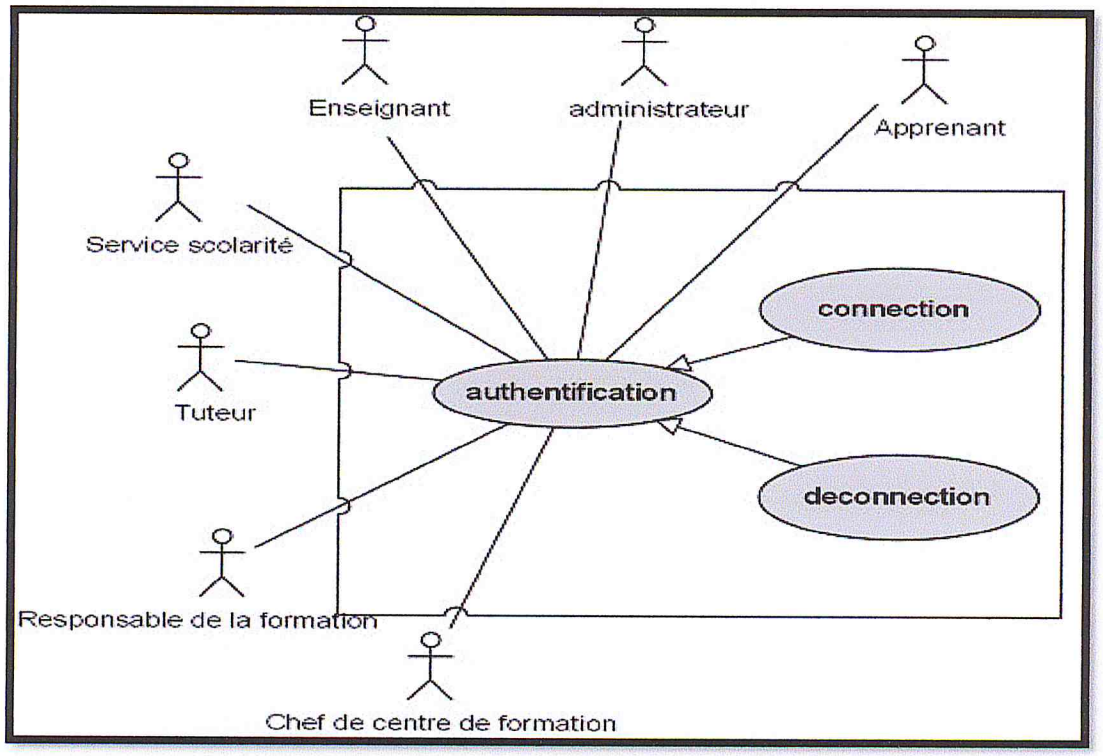


Figure 8 : Cas d'utilisation N°1 « Authentification »

Cas d'utilisation N°2 «Afficher liste des comptes utilisateur» :

Description de sommaire	
Titre	Afficher liste des comptes utilisateur
But	permet à l'administrateur de consulter la liste des comptes utilisateurs
Acteurs	administrateur
Description des enchainements	
Pré-conditions	L'administrateur est authentifié.
Enchainements	1. L'administrateur s'authentifie. 2. L'administrateur accède à l'espace des comptes utilisateurs 3. L'administrateur peut consulter la liste des utilisateurs. 4. L'administrateur se déconnecter.
Post-conditions	L'administrateur est déconnecté

Diagramme

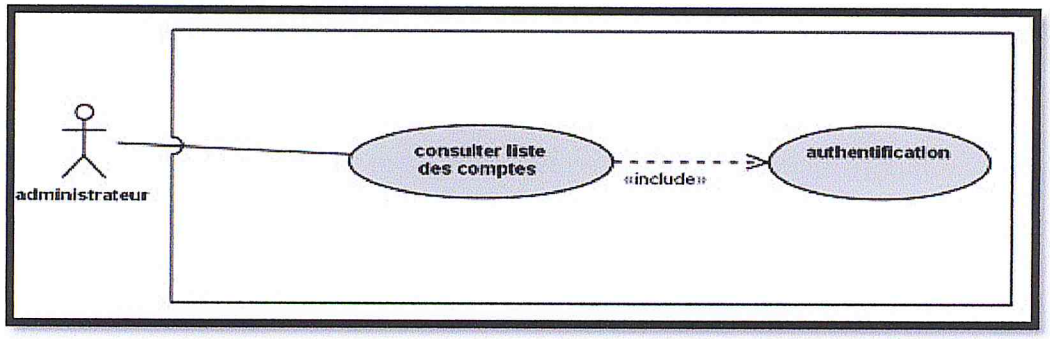


Figure 9 : Cas d'utilisation N°2 «Afficher liste des comptes utilisateur»

Cas d'utilisation N°3 «Gestion des profils utilisateurs» :

- Scénario <<ajouter nouveau profil>>

Description de sommaire	
Titre	ajouter nouveau profil
But	permet à l'administrateur d'ajouter un nouveau profil utilisateur
Acteurs	administrateur
Description des enchainements	
Pré-conditions	L'administrateur est authentifié.

Enchainements	1. L'administrateur s'authentifie. 2. L'administrateur accède à l'espace de gestion des profils 3. L'administrateur ajoute un nouveau profil. 4. Le système demande une confirmation l'ajout. 5. L'administrateur confirme l'ajout puis il se déconnecter.
Post-conditions	Profil MAJ

Diagramme

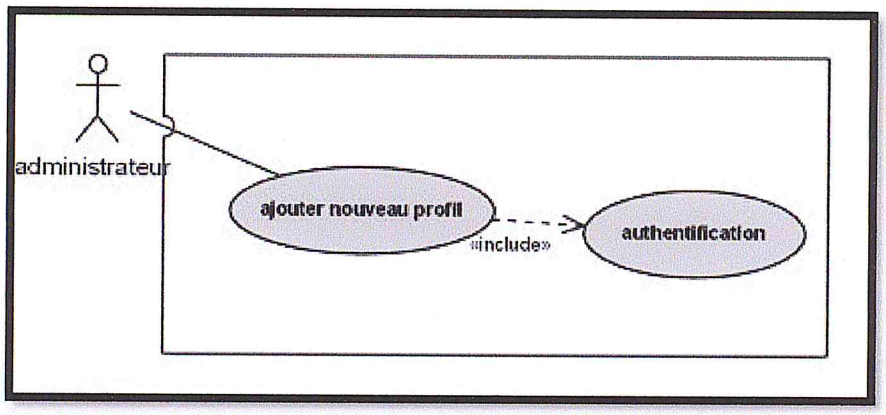


Figure 10 : Cas d'utilisation N°3 «ajouter nouveau profil»

- Scénario << affecter actions au profil >>

Description de sommaire	
Titre	affecter actions au profil
But	permet à l'administrateur d'affecter une liste d'actions au profil
Acteurs	administrateur
Description des enchainements	
Pré-conditions	L'administrateur est authentifié.
Enchainements	1. L'administrateur s'authentifie. 2. L'administrateur accède à l'espace de gestion des profils 3. L'administrateur sélectionne un profil. 4. L'administrateur sélectionne l'ensemble des actions à affecter. 4. Le système demande une confirmation l'ajout. 5. L'administrateur confirme l'ajout puis il se déconnecter.
Post-conditions	Profil MAJ

Diagramme :

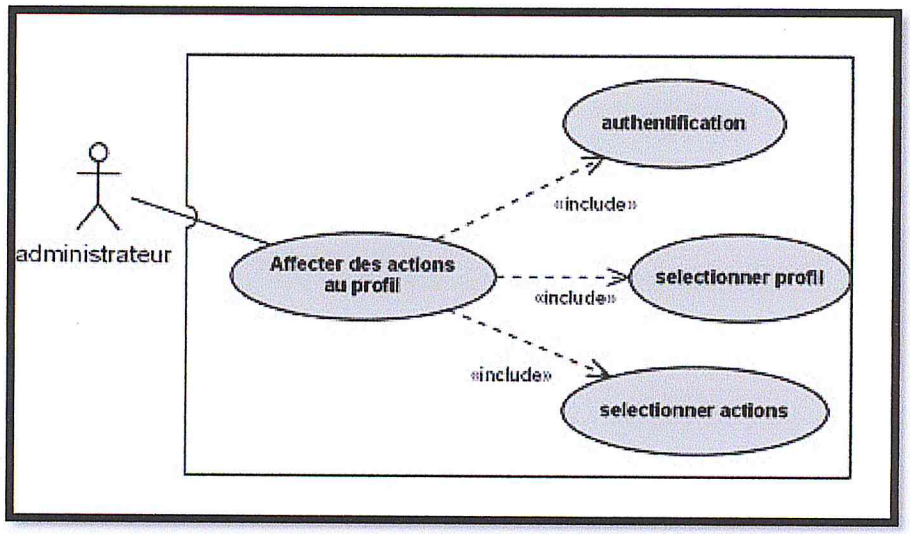


Figure 11 : Cas d'utilisation N°4 «affecter actions au profil»

Cas d'utilisation N°5 «Gestion de formation» :

Description de sommaire	
Titre	Gestion de formation
But	permet de gérer les formations
acteurs	Responsable de la formation, chef de centre de formation
Description des enchainements	
Pré-conditions	L'administrateur est authentifié.
Enchainements	1. L'administrateur s'authentifie. 2. L'administrateur accède à l'espace de gestion de la formation 3. L'administrateur peut modifier, ajouter une formation racine, une phase de formation, une unité de formation, un module
Post-conditions	Formation MAJ

Diagramme

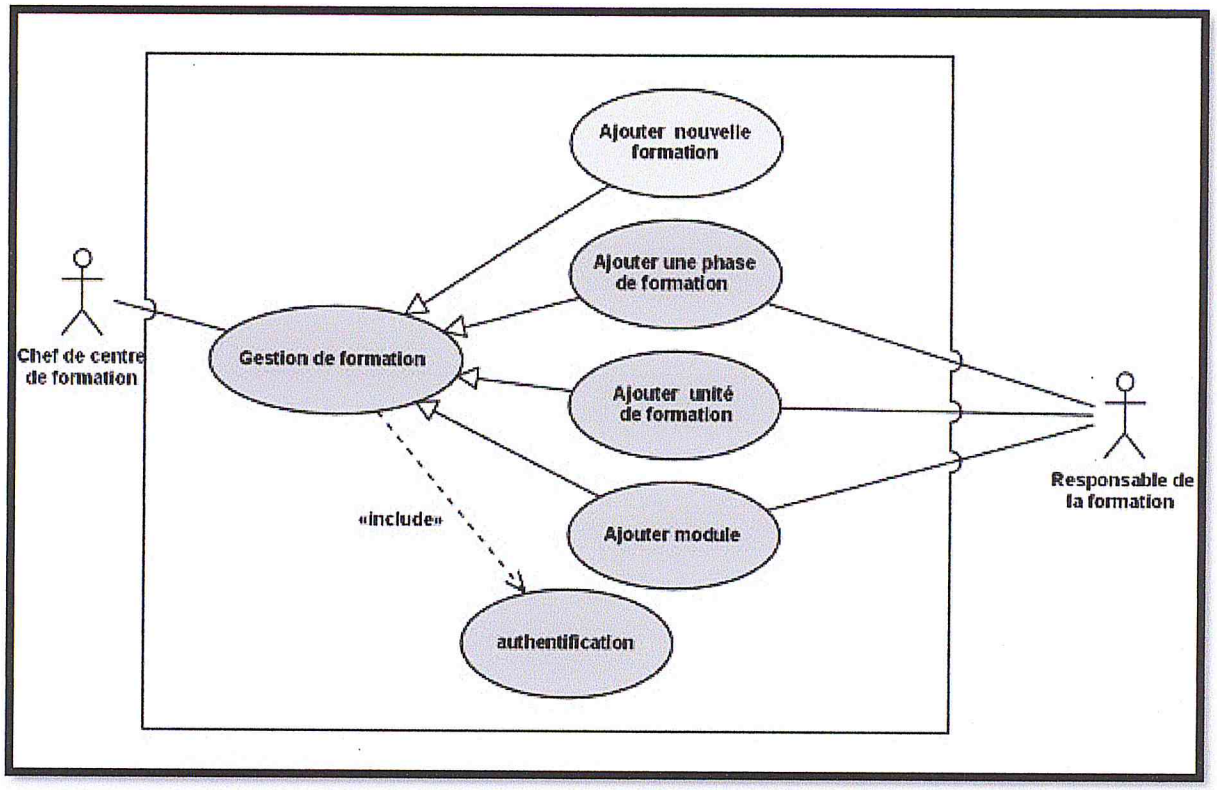


Figure 12 : Cas d'utilisation N°4 «Gestion de formation»»

Cas d'utilisation N°5 «Gestion des types de contrôles» :

- Scenario : <<ajouter un type de contrôle>>

Description de sommaire	
Titre	Ajout des types de contrôles
But	permet d'ajouter les types de contrôles
acteurs	Service scolarité
Description des enchainements	
Pré-conditions	Type de contrôle MAJ
Enchainements	1. L'administrateur s'authentifie. 2. L'administrateur accède à l'espace d'ajout des types de contrôles 3. L'administrateur peut ajouter un type de contrôle.
Post-conditions	Type de contrôle MAJ

Diagramme :

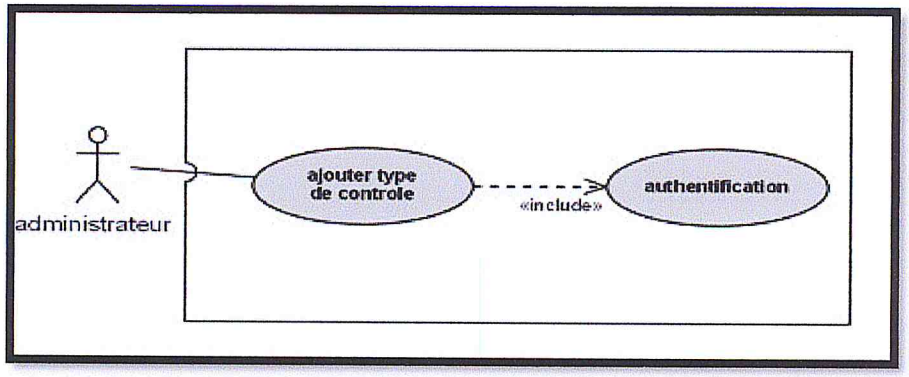


Figure 13 : Cas d'utilisation N°5 «ajouter un type de contrôle»

❖ Scenario : <<affecter les types de contrôles aux modules>>

Description de sommaire	
Titre	Affecter les types de contrôles aux modules
But	permet d'Affecter les types de contrôles aux modules
acteurs	Service scolarité
Description des enchainements	
Pré-conditions	Type de contrôle MAJ
Enchainements	1. L'administrateur s'authentifie. 2. L'administrateur accède à l'espace d'affectation des types de contrôles aux modules 3. L'administrateur sélectionne un module. 4. L'administrateur sélectionne un ensemble des types de contrôle. 5. L'administrateur affecte les types de contrôles au module sélectionné
Post-conditions	Les types de contrôle sont affectés au module

Diagramme :

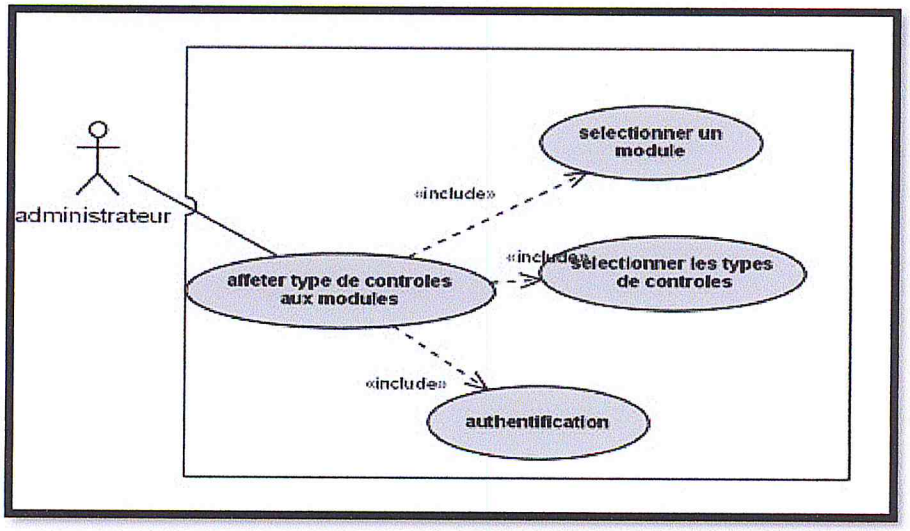


Figure 14 : Cas d'utilisation N°5 «affecter type de contrôle aux modules»

Cas d'utilisation N° 6 «Gestion des notes de contrôle» :

Description de sommaire	
Titre	Gestion des notes de contrôle
But	permet de gérer les notes de contrôle
acteurs	Service scolarité, enseignant, apprenant, responsable de la formation.
Description des enchainements	
Pré-conditions	Les acteurs (Service scolarité, enseignant, apprenant, responsable de la formation.) sont authentifiés.
Enchainements	2. L'acteur accède à l'espace des notes de contrôle 3. l'agent de Service ou un enseignant peut ajouter une note, afficher la liste des notes, modifie une note, valider une note 4. l'apprenant peut voir la liste des notes 5. le responsable de formation peut valiser les notes
Post-conditions	Note MAJ

Diagramme

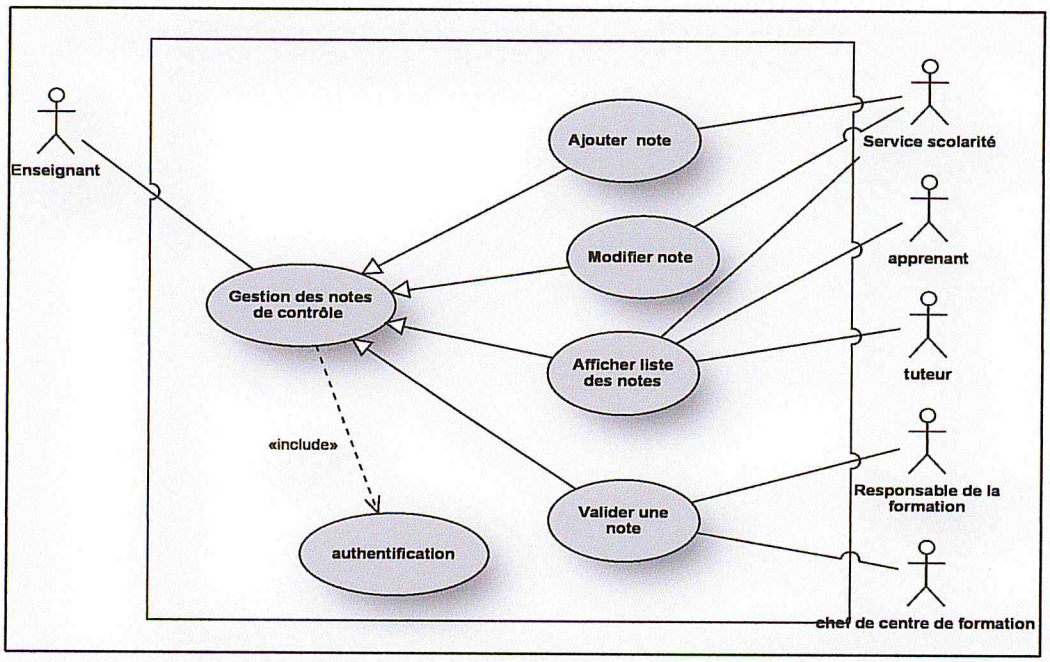


Figure 15 : Cas d'utilisation N° 6 «Gestion des notes de contrôle»

❖ Scénario <<valider une note >>

Description de sommaire	
Titre	valider une note
But	permet de valider et modifier les notes de contrôle
acteurs	enseignant, responsable de la formation, chef de centre de formation
Description des enchainements	
Pré-conditions	Les acteurs (enseignant, responsable de la formation, chef de centre de formation sont authentifiés.
Enchainements	1. L'acteur accède à l'espace des notes de contrôle 2. l'acteur modifie et valide la note de contrôle.
Post-conditions	Note validée

Diagramme :

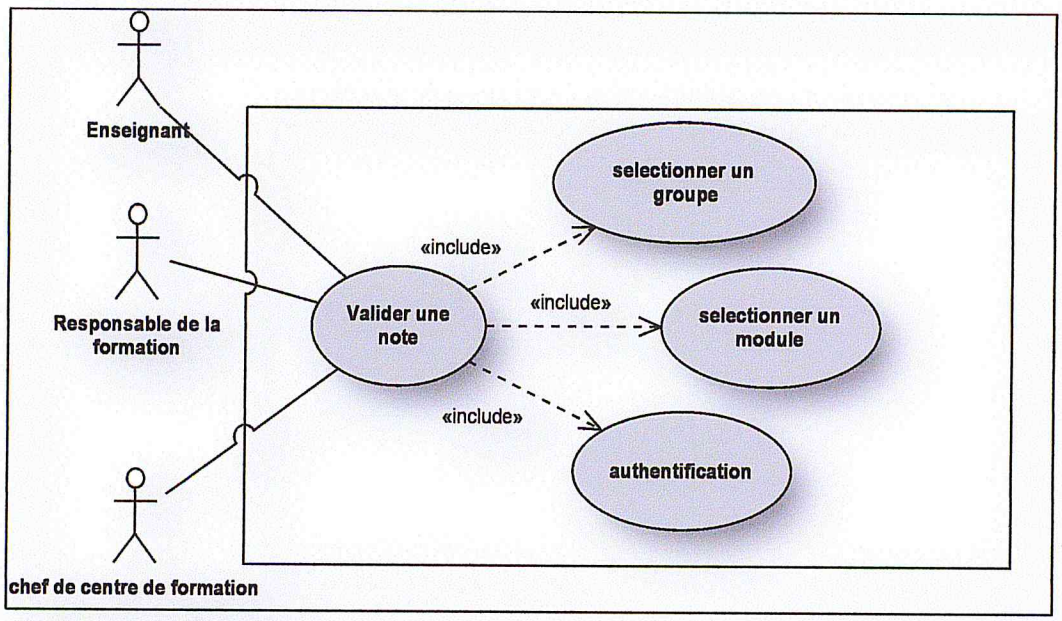


Figure 16 : Cas d'utilisation N° 6 «valider une note »

Cas d'utilisation N°7 «Gestion des promotions» :

Description de sommaire	
Titre	Gestion des promotions
But	permet de gérer les promotions
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	2. L'agent de service scolarité accède à l'espace de gestion des promotions 3. L'agent de service scolarité peut ajouter, modifier, consulter la liste des promotions, affecter les apprenants aux promotions
Post-conditions	Promotion validée

Diagramme

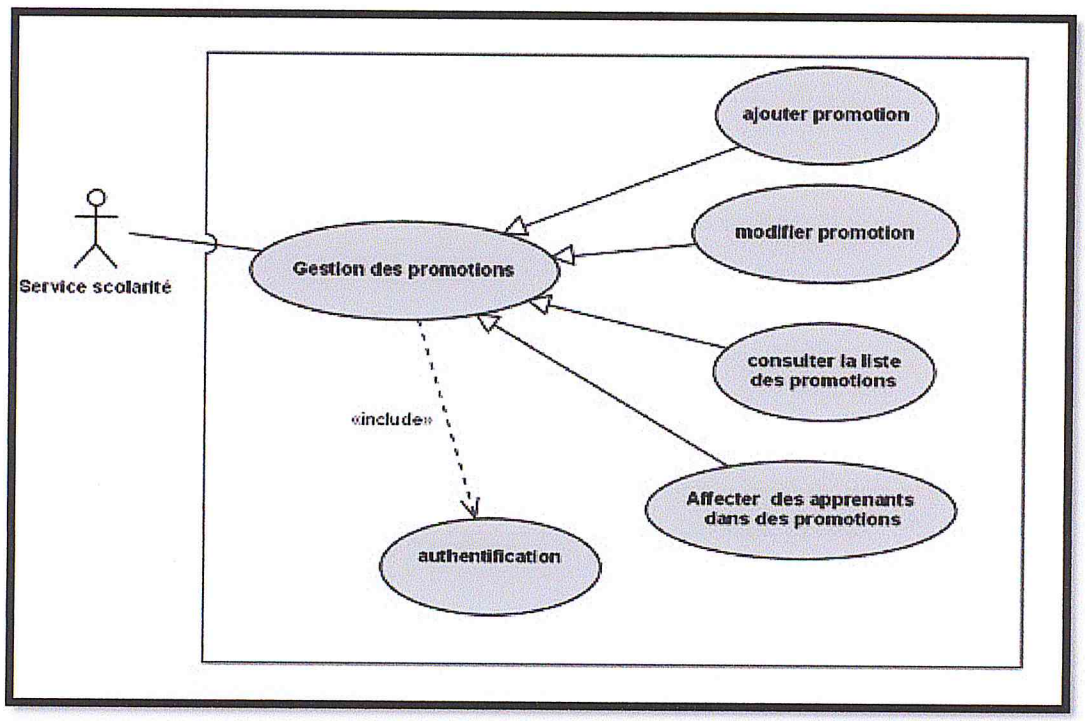


Figure 17 : Cas d'utilisation N°7 «Gestion des promotions»

Cas d'utilisation N°8 «ajouter une remarque a un apprenant» :

Description de sommaire	
Titre	ajouter une remarque a un apprenant
But	permet d'ajouter une remarque a un apprenant
Acteurs	enseignant
Description des enchainements	
Pré-conditions	enseignant s'authentifie.
Enchainements	1. enseignant accède à l'espace 'ajouter une remarque a un apprenant 2. enseignant peut ajouter une remarque a un apprenant
Post-conditions	Remarque MAJ

Diagramme :

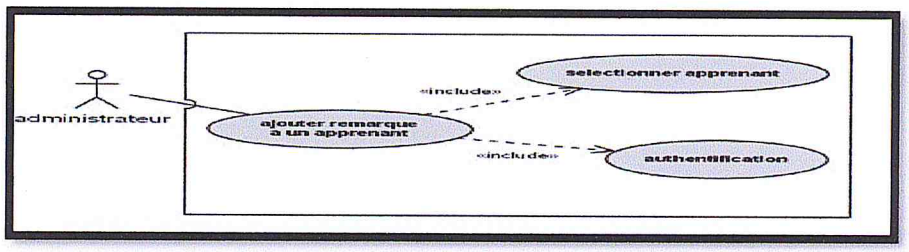


Figure 18 : Cas d'utilisation N°8 «ajouter remarque»

Cas d'utilisation N°8 «Gestion des groupes» :

Description de sommaire	
Titre	Gestion des groupes
But	permet de gérer les groupes
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	2. L'agent de service scolarité accède à l'espace de gestion des groupes 3. L'agent de service scolarité peut ajouter un groupe, affecter les apprenants au groupes
Post-conditions	Groupe MAJ

Diagramme

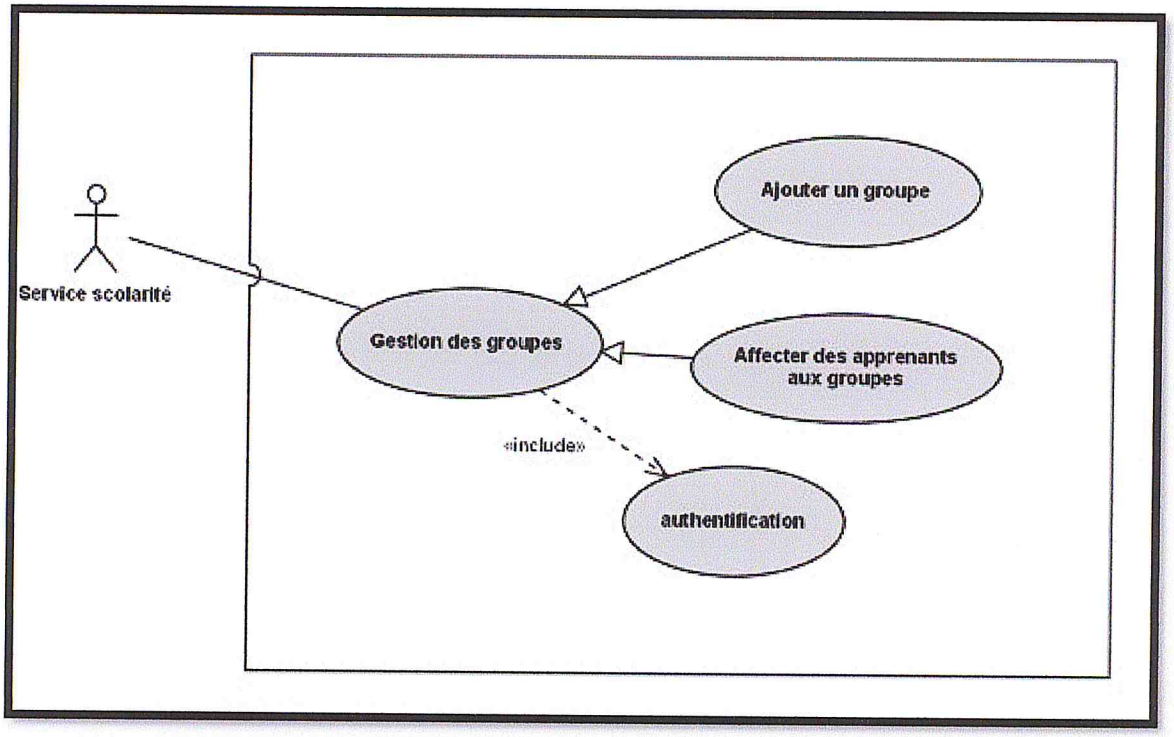


Figure 19 : Cas d'utilisation N°8 «Gestion des groupes»

Cas d'utilisation N°9 «Gestion des sections» :

Description de sommaire	
Titre	Gestion des sections
But	permet de gérer les sections
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	2. L'agent de service scolarité accède à l'espace de gestion des sections 3. L'agent de service scolarité peut ajouter une section, modifier une section, consulter liste des sections.
Post-conditions	Section MAJ

Diagramme :

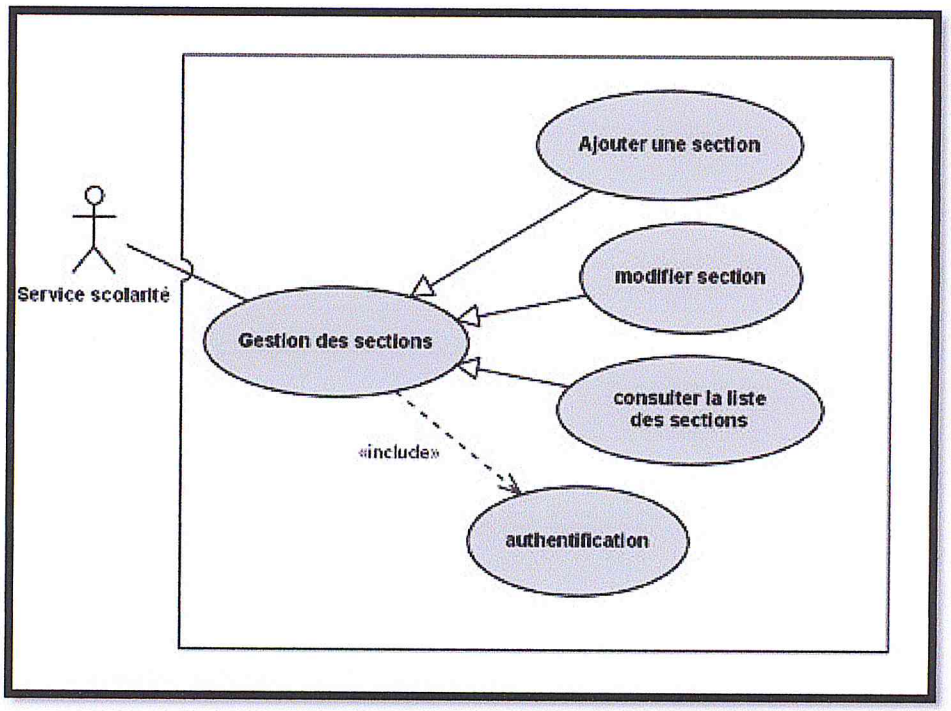


Figure 20 : Cas d'utilisation N°9 «Gestion des sections»

Cas d'utilisation N°10 «Gestion des apprenants » :

Description de sommaire	
Titre	Gestion des apprenants
But	permet de gérer les apprenants
acteurs	Service de scolarité, enseignant, apprenant
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	2. L'agent de service scolarité accède à l'espace de gestion des apprenants 3. L'agent de service scolarité inscrit un apprenant 4. L'agent de service scolarité affecte un apprenant dans une promotion, section et groupe
Post-conditions	Apprenant inscrit

Diagramme :

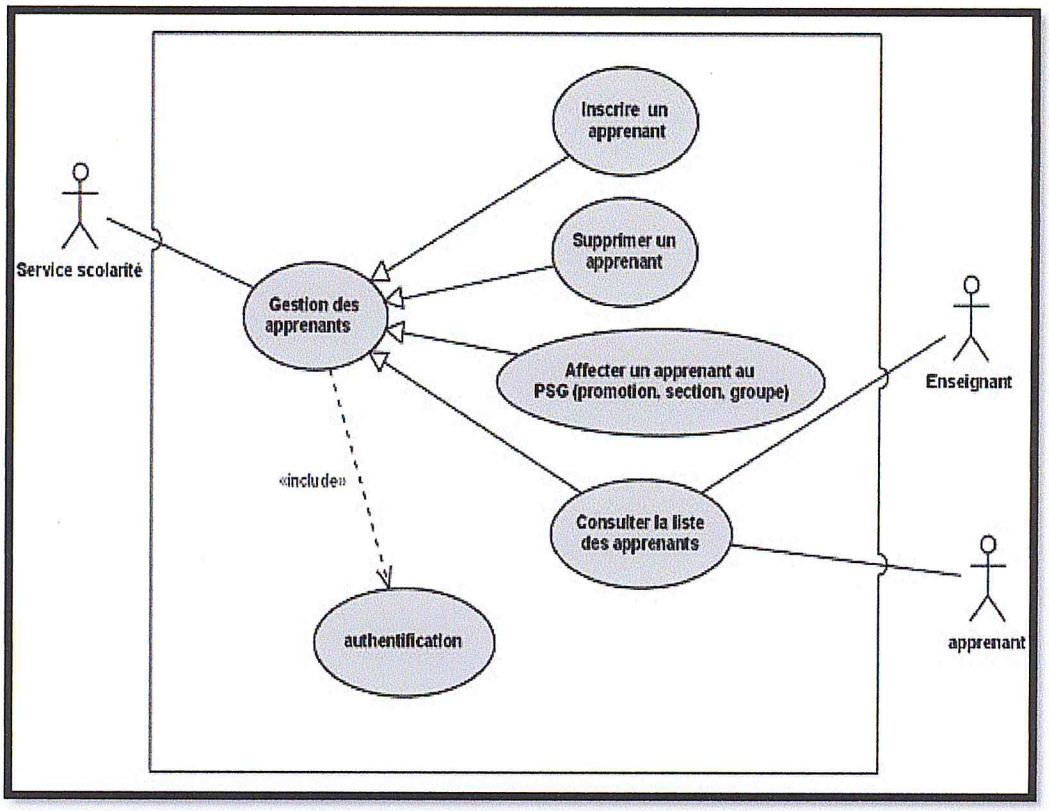


Figure 21 : Cas d'utilisation N°10 «Gestion des apprenants »

Cas d'utilisation N°11 «Gestion des enseignants» :

Description de sommaire	
Titre	Gestion des enseignants
But	permet de gérer les enseignants
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	2. L'agent de service scolarité accède à l'espace de gestion des enseignants 3. L'agent de service scolarité ajoute un nouveau enseignant 4. L'agent de service scolarité affecte un module à un enseignant 5. l'agent de scolarité affecte une section ou un groupe à un enseignant.
Post-conditions	Enseignant ajouté

Diagramme :

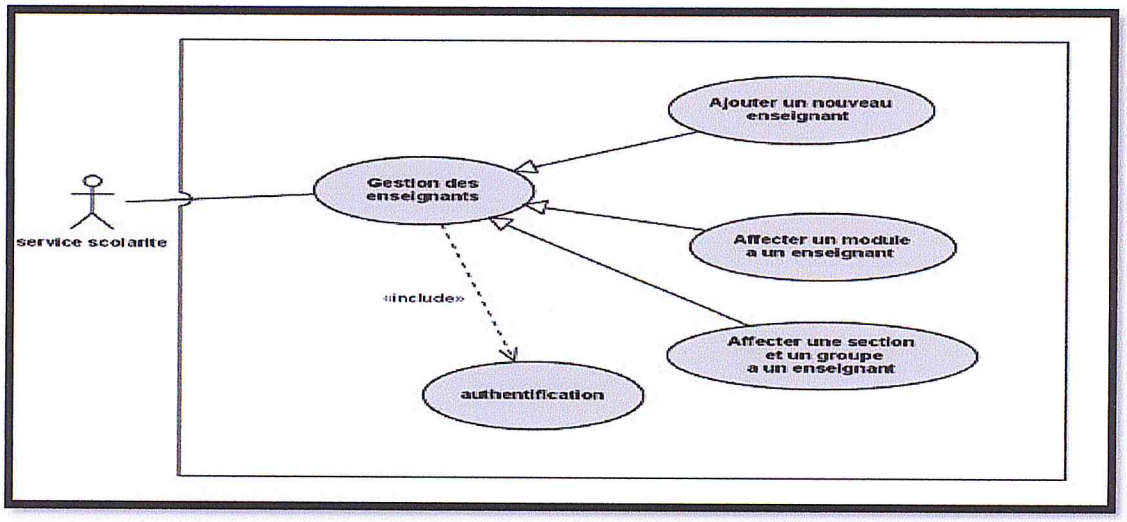


Figure 22 : Cas d'utilisation N°11 «Gestion des enseignants»

Cas d'utilisation N°12 «Ajouter une nouvelle année pédagogique» :

Description de sommaire	
Titre	Ajouter une nouvelle année pédagogique
But	permet d'ajouter une nouvelle année pédagogique
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	2. L'agent de service scolarité accède à l'espace d'ajout d'une nouvelle année pédagogique 3. L'agent de service scolarité ajoute une année pédagogique
Post-conditions	année pédagogique ajoutée

Diagramme :

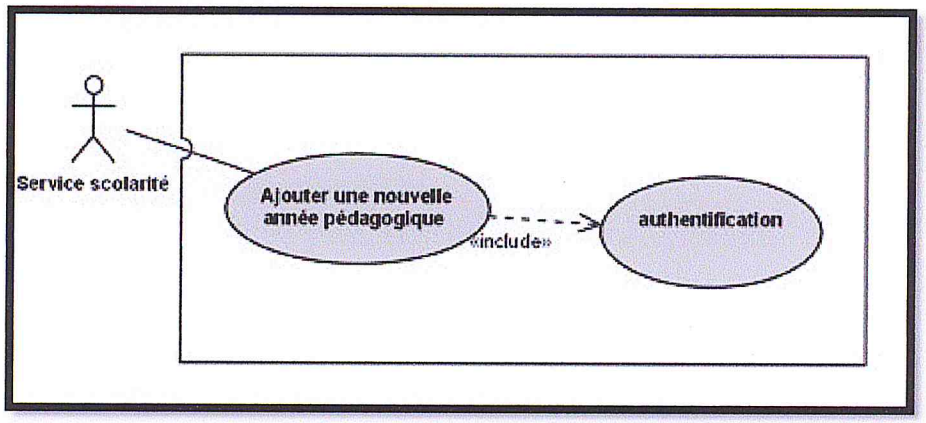


Figure 23 : Cas d'utilisation N°12 «Ajouter une nouvelle année pédagogique»

Cas d'utilisation N°12 «Ouverture d'une phase de formation» :

Description de sommaire	
Titre	Ouverture d'une phase de formation
But	permet d'ouverture d'une phase de formation
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	1. L'agent de service scolarité accède à l'espace d'ouverture d'une phase de formation 2. L'agent de service scolarité sélectionne une promotion et une phase de formation 3. L'agent de service scolarité ouvre la phase sélectionnée à la promotion
Post-conditions	La promotion est dans la nouvelle phase de formation

Diagramme :

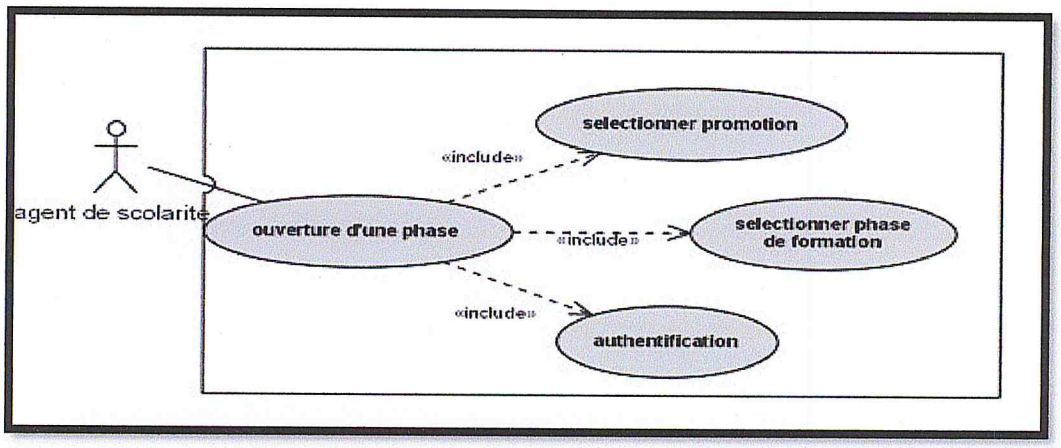


Figure 24 : Cas d'utilisation N°13 «Ouverture d'une phase de formation»

Cas d'utilisation N°12 «Clôture d'une phase de formation (délibération)» :

Description de sommaire	
Titre	Clôture d'une phase de formation (délibération)»
But	permet de clôturer la phase pour une promotion
acteurs	Service de scolarité
Description des enchainements	
Pré-conditions	L'agent de service scolarité s'authentifie.
Enchainements	1. L'agent de service scolarité accède à l'espace de délibération 2. L'agent de service scolarité sélectionne une promotion et une phase de formation 3. L'agent de service scolarité délibère le résultat de promotion et la ferme la phase pour la promotion.
Post-conditions	La phase est fermée pour la promotion

Diagramme :

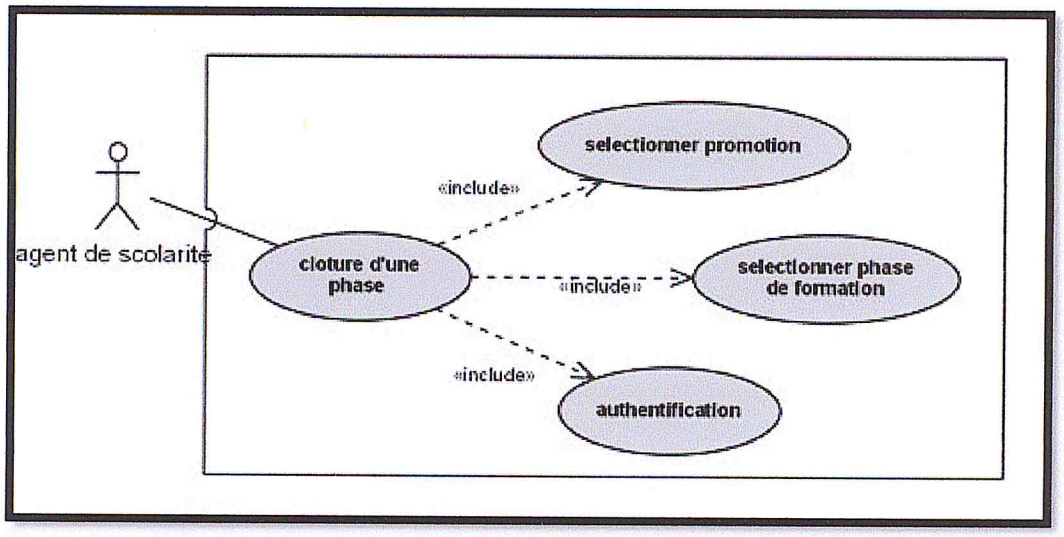


Figure 25 : Cas d'utilisation N°14 «Clôture d'une phase de formation (délibération)»

3.2.4 Les diagrammes de séquence :

Cas d'utilisation <<authentification>>

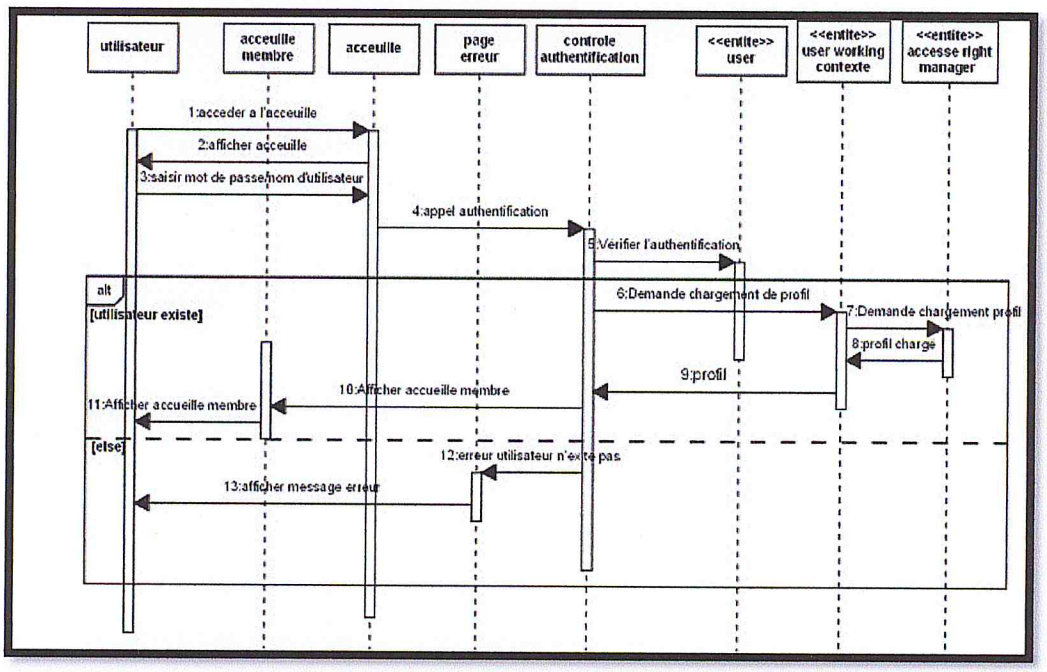


Figure 26 : diagramme de séquence de scenario «authentification».

Cas d'utilisation << Gestion des profils>>

❖ Le scénario « ajouter un profil »

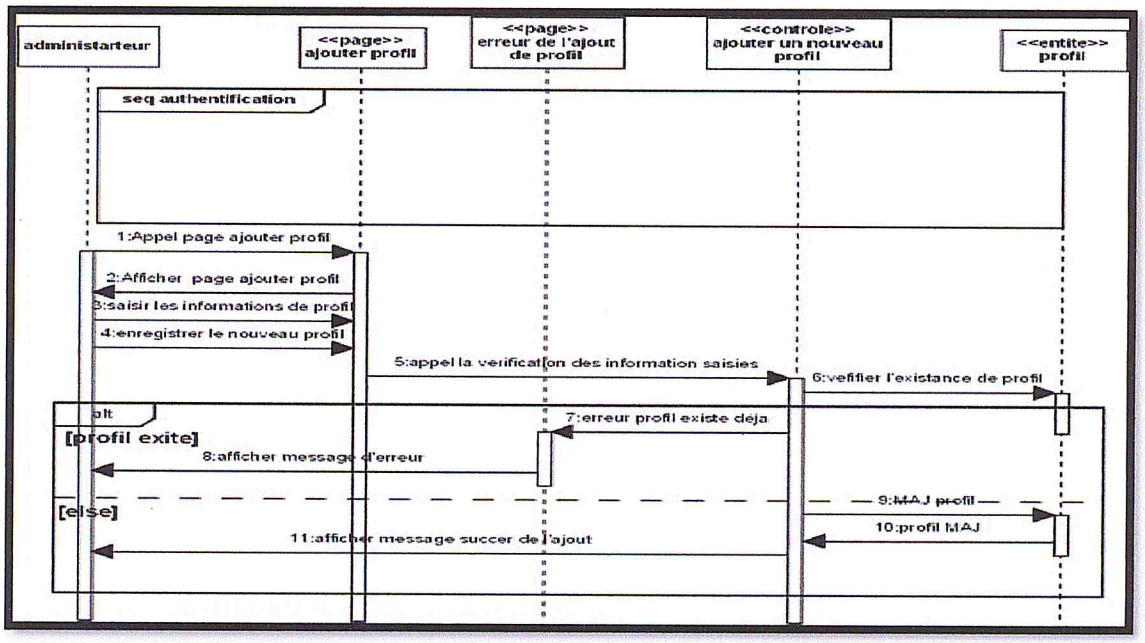


Figure 27 : diagramme de séquence de scenario «ajouter un profil»

Cas d'utilisation «Gestion de formation»

❖ Le scénario « ajouter une formation racine »

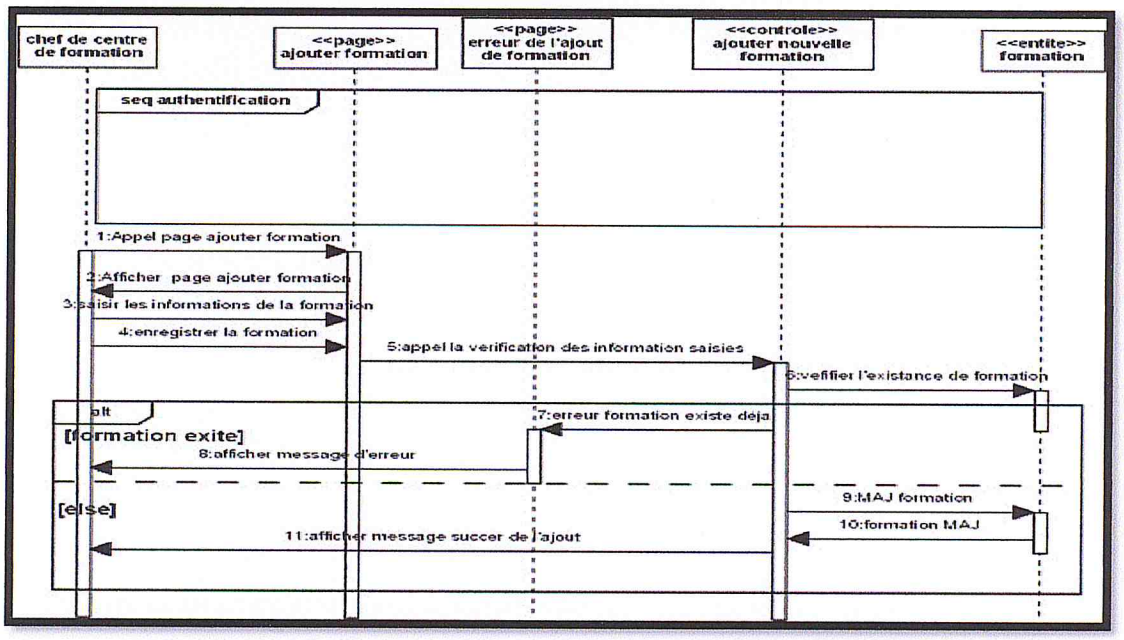


Figure 28 : diagramme de séquence de scenario «ajouter une formation racine».

❖ Le scénario « ajouter une phase de formation »

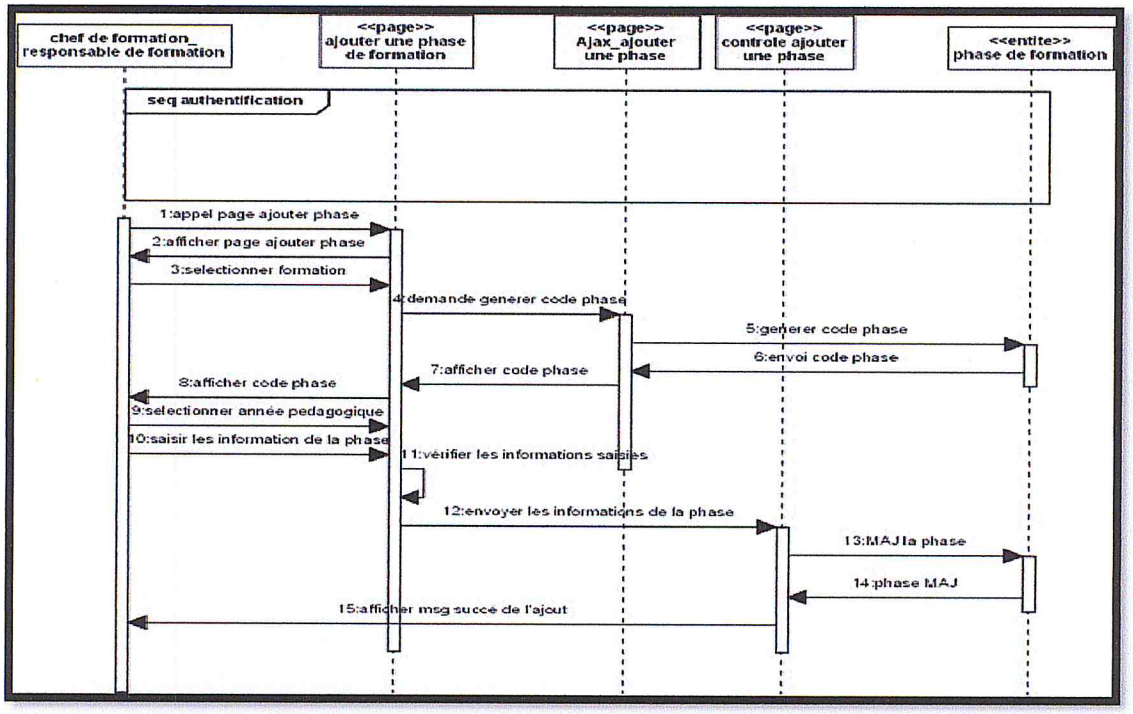


Figure 29 : diagramme de séquence de scenario «ajouter une formation racine».

❖ Le scénario « ajouter une unité de formation »

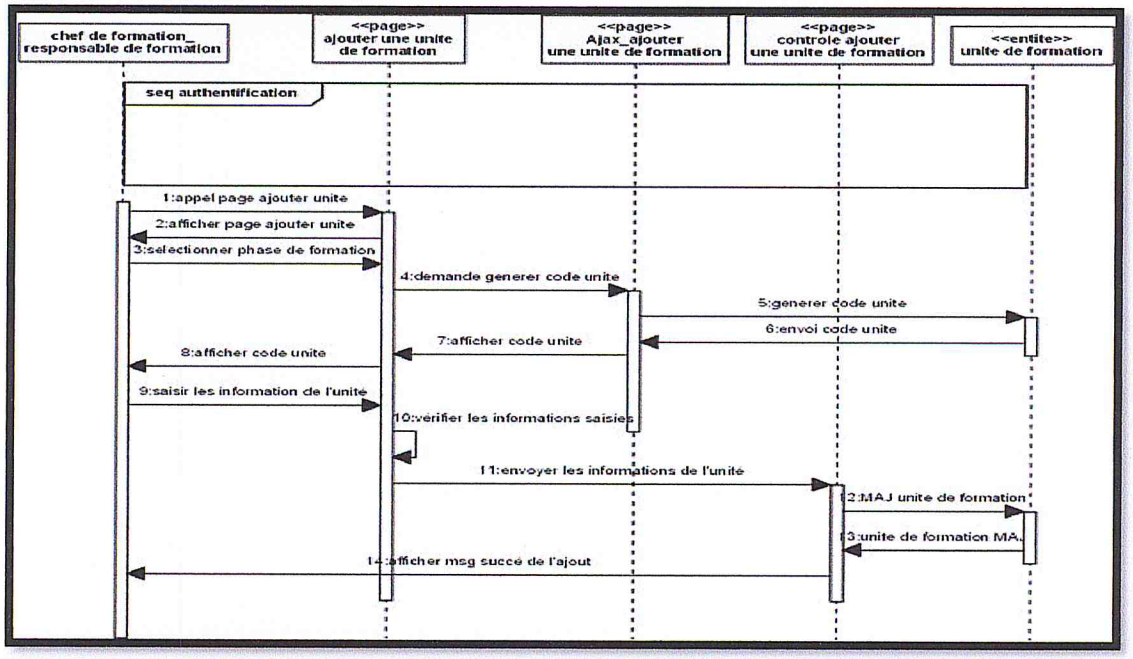


Figure 30 : diagramme de séquence de scenario «ajouter une unité de formation ».

❖ Le scénario « ajouter un module »

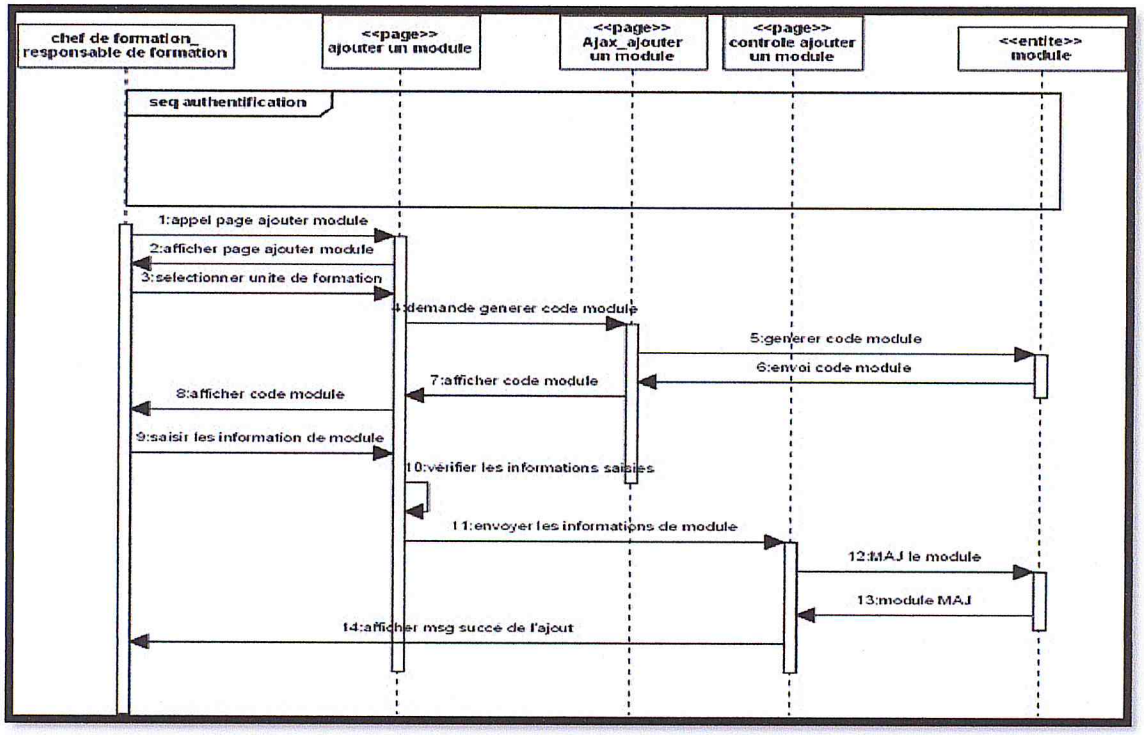


Figure 31 : diagramme de séquence de scenario «ajouter un module ».

Cas d'utilisation «gestion des types de contrôle»

❖ Le scénario « ajouter un type de contrôle »

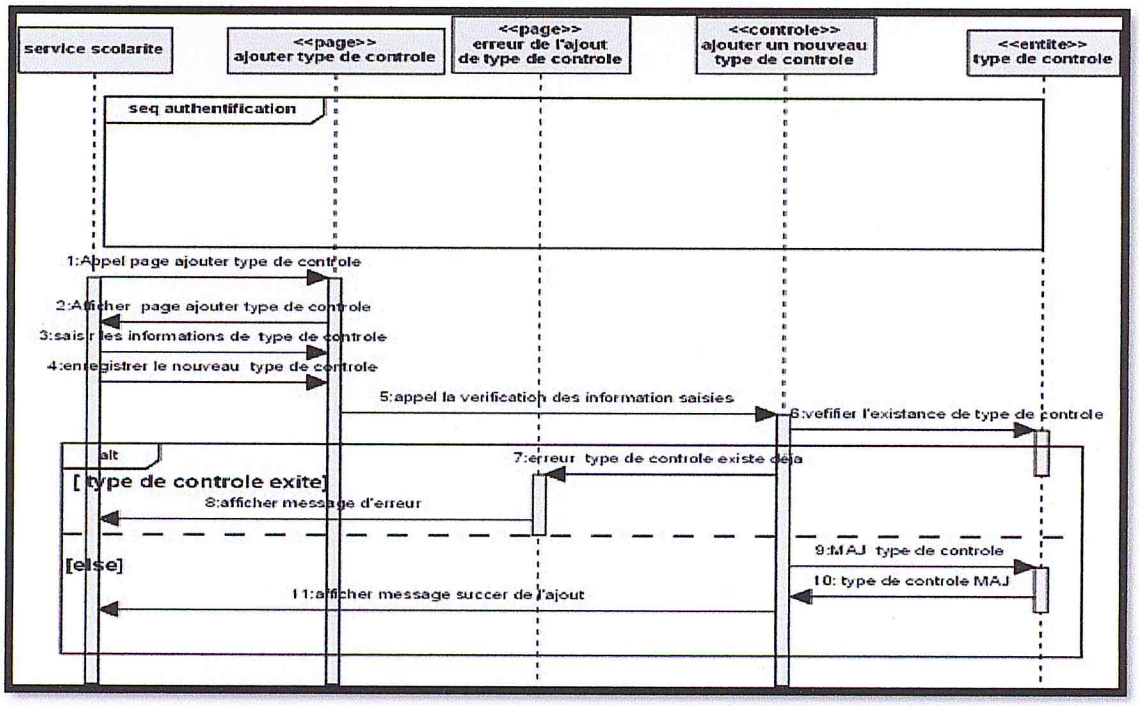


Figure 32: diagramme de séquence de scenario «ajouter un type de contrôle ».

❖ Le scénario « affecter un type de contrôle au modules »

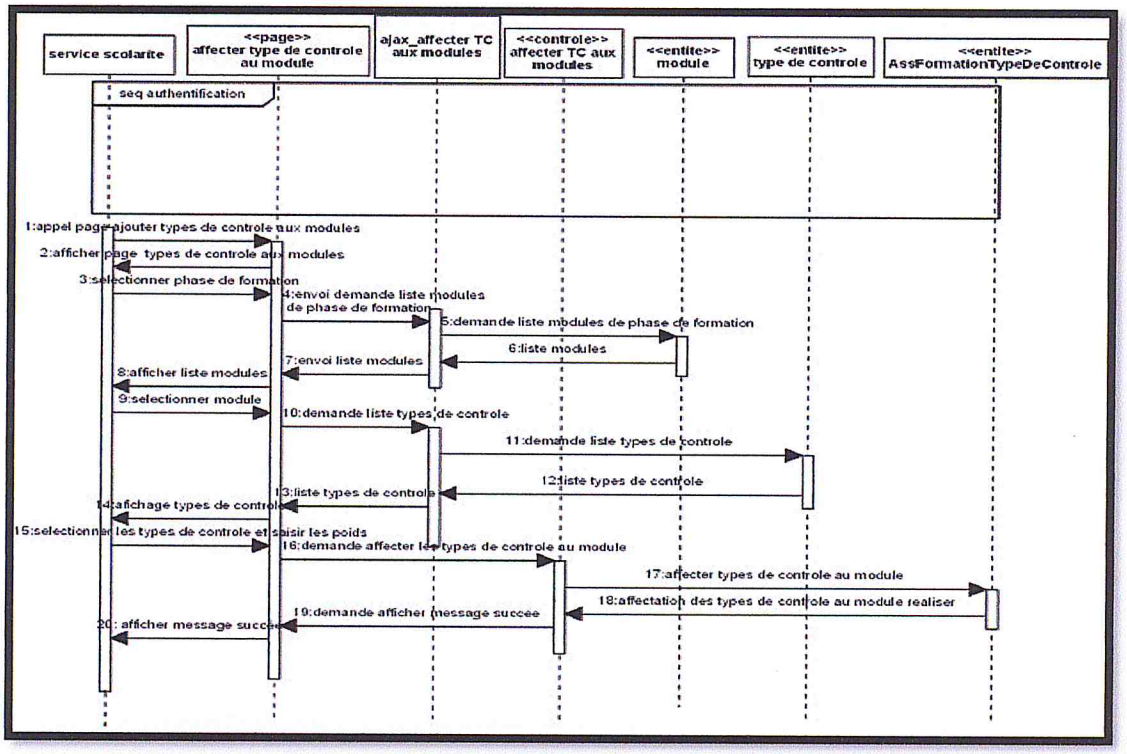


Figure 33 : diagramme de séquence de scenario «affecter un type de contrôle au modules».

Cas d'utilisation «Gestion des notes de contrôle»

❖ Le scénario « affecter une note de contrôle a un étudiant »

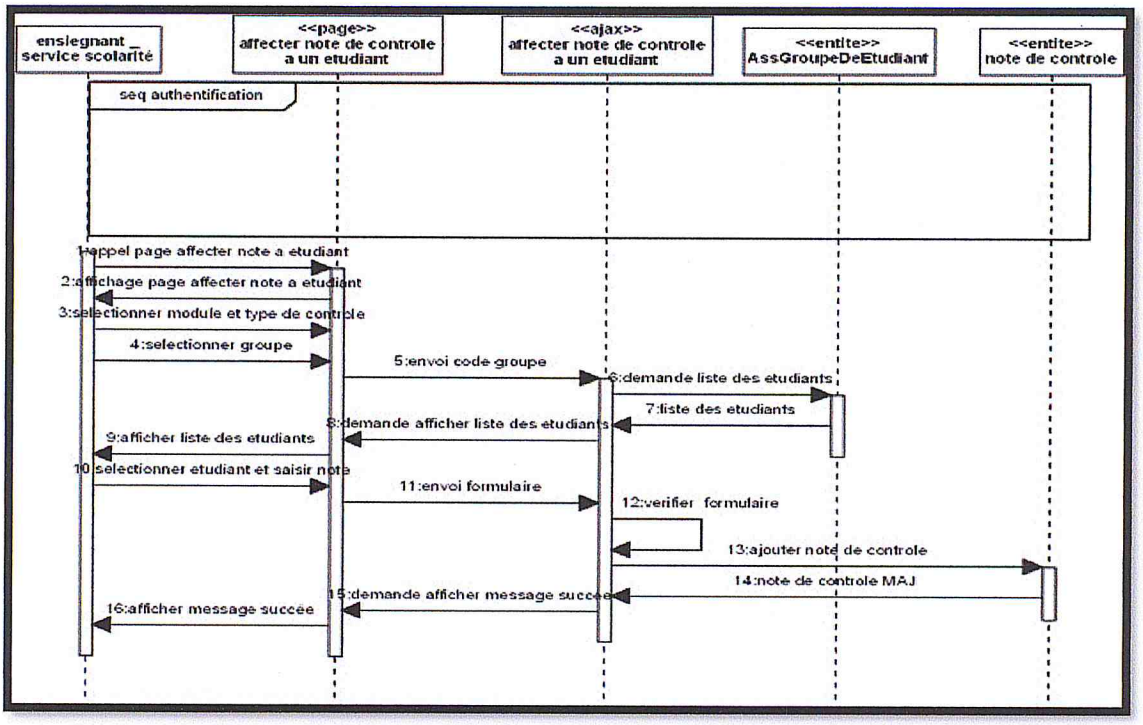


Figure 34 : diagramme de séquence de scenario «affecter une note a un étudiant».

❖ Le scénario « affichage des notes de contrôles »

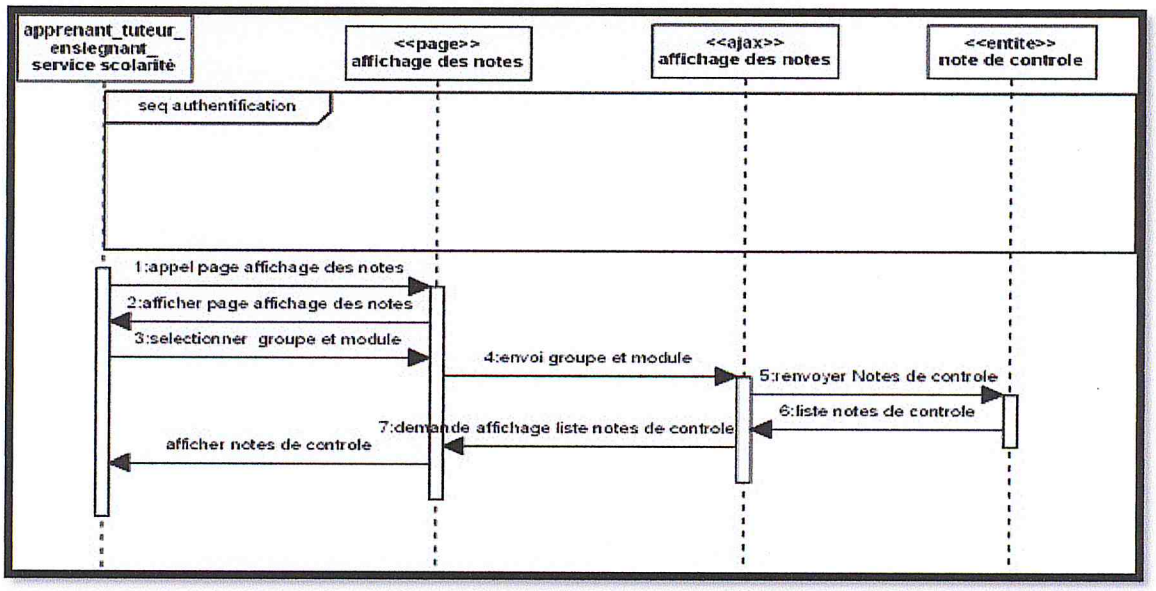


Figure 35 : diagramme de séquence de scenario «affichage des notes de contrôles».

❖ le scenario « valider les notes de contrôles »

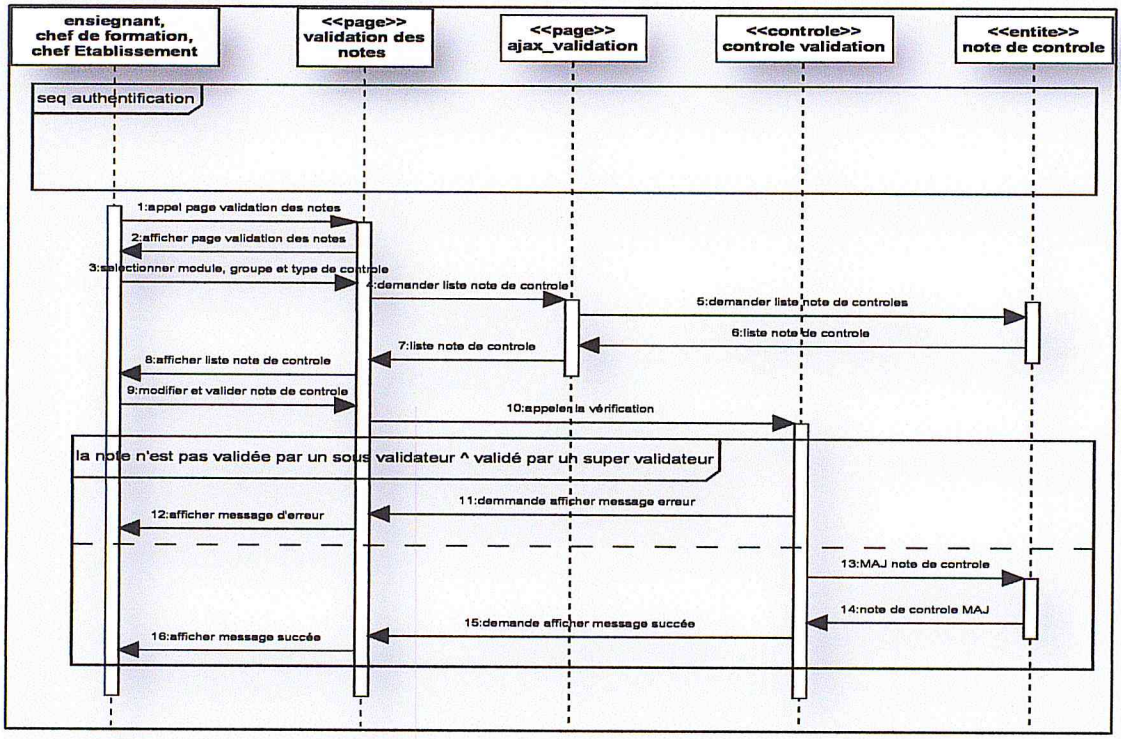


Figure 36 : diagramme de séquence de scenario «valider les notes de contrôles».

Cas d'utilisation «Gestion des promotions »

❖ Le scénario « ajouter une promotion»

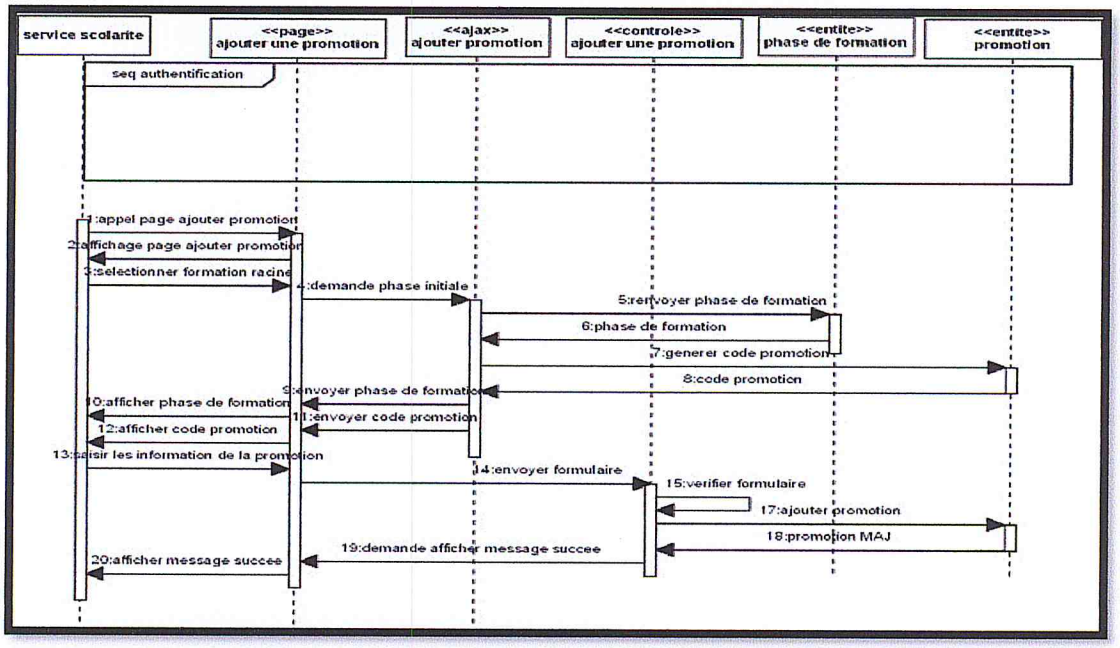


Figure 37 : diagramme de séquence de scenario «ajouter une promotion».

Cas d'utilisation «ouverture d'une phase pour une promotion »

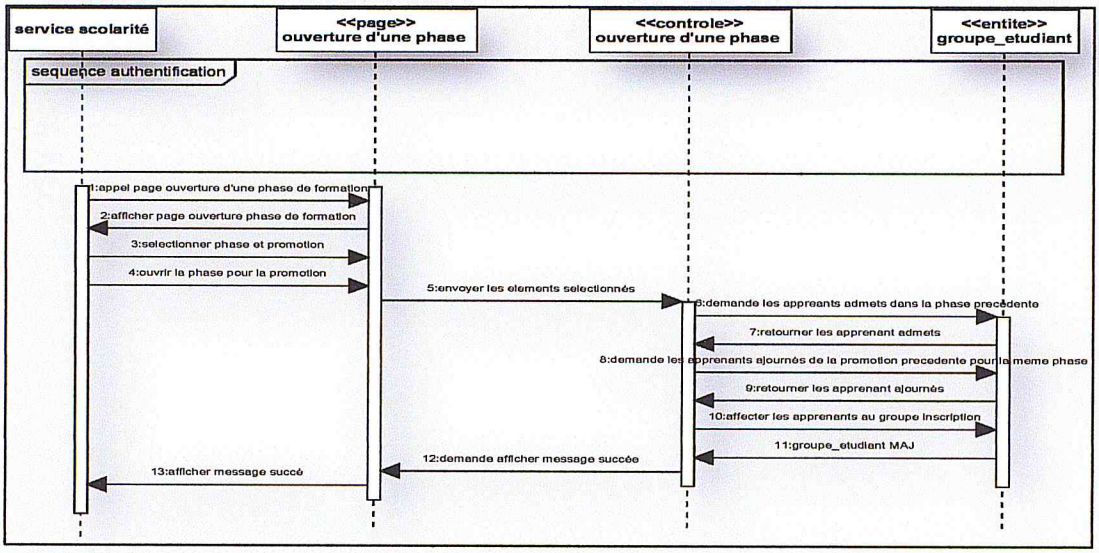


Figure 38 : diagramme de séquence de scenario «ouverture d'une phase pour une promotion».

Cas d'utilisation «fermeture phase (délibération) »

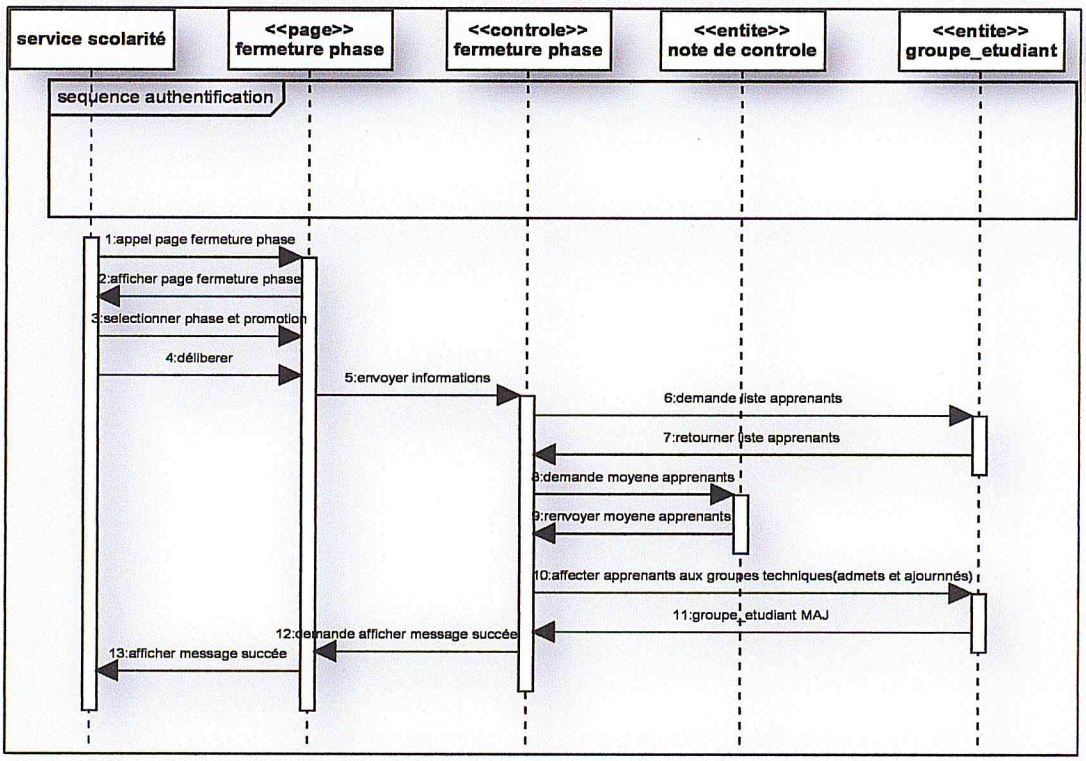


Figure 39 : diagramme de séquence de scenario «fermeture phase».

3.3 Conception:

3.3.1 Description détaillée des attributs :

a)Les attributs des classes d'objets :

Classes	Attributs	désignation	taille
Type de contrôle	code_tc nom_tc descr	Code type de contrôle Nom type de contrôle Description type de contrôle	VARCHAR(4) VARCHAR(30) VARCHAR(30)
Formation racine	codeFr code_racine code_parent code_absolu nomFr poids domaine filiere specialite date_debut date_fin credit	Identifiant de formation Code formation racine Code formation parente Code absolue Nom formation Poids de formation Domain de formation Filière de formation La spécialité Date début de formation Date fin de formation Crédit de formation	VARCHAR(09) VARCHAR(09) VARCHAR(09) VARCHAR(09) VARCHAR(50) INT VARCHAR(50) VARCHAR(50) VARCHAR(50) DATE DATE DATE
Phase de Formation	codePf code_racine code_parent code_absolu nomPf poids credit	Identifiant phase de formation Code formation racine Code formation parente Code absolue Nom phase de formation Poids phase de formation Crédit de formation	VARCHAR(15) VARCHAR(9) VARCHAR(9) VARCHAR(15) VARCHAR(50) INT INT
Unité de formation	codeUf code_racine code_parent code_absolu nomUf poids credit	Identifiant unité de formation Code formation racine Code formation parente Code absolue Nom phase unité de formation Poids Crédit de l'unité	VARCHAR(19) VARCHAR(09) VARCHAR(15) VARCHAR(19) VARCHAR(50) INT INT
module	codeMd code_racine code_parent code_absolu nomMd poids credit	Identifiant module Code formation racine Code formation parente Code absolue Nom module Poids Crédit de module	VARCHAR(22) VARCHAR(09) VARCHAR(19) VARCHAR(22) VARCHAR(50) INT INT
Année pédagogique	code_annee nom_annee	Code année pédagogique Nom année pédagogique	VARCHAR(14) VARCHAR(50)

Promotion	codePr nomPr description date	Code promotion Nom promotion Description de la promotion Date de création de promotion	VARCHAR(16) VARCHAR(16) VARCHAR(50) DATE
Section	codeSec nomSec description taille	Code section Nom section Description section Taille de section	VARCHAR(36) VARCHAR(50) VARCHAR(40) INT
Groupe	codeGr nomGr description activite taille	Code groupe Nom groupe Description Activité de groupe Taille de groupe	VARCHAR(40) VARCHAR(50) VARCHAR(50) VARCHAR(50) INT
Etudiant	puid pere_puid mere_puid matricule nom prenom date_naissance lieu_naissance inst_origine matricule_origine	Identifiant national étudiant Identifiant père de l'étudiant Identifiant mère de l'étudiant Matricule étudiant Nom étudiant Prénom étudiant Date de naissance Lieu de naissance Institue origine Matricule origine	INT INT INT VARCHAR(20) VARCHAR(30) VARCHAR(30) DATE VARCHAR(80) VARCHAR(40) VARCHAR(20)
Date	date	date	DATE
Enseignant	id_ens nom prenom sex adresse n_telephone grade fonction	Identifiant enseignant Nom enseignant Prénom enseignant Sexe Adresse Numéro téléphone Grade Fonction	INT VARCHAR(30) VARCHAR(30) VARCHAR(10) VARCHAR(80) INT VARCHAR(30) VARCHAR(30)
Profil	identifiat_pr type_pr	Identifiant profil Type de profil	INT VARCHAR(20)
Action	id_action nom_action	Identifiant action Nom action	INT VARCHAR(50)
Compte utilisateur	nom_util mot_passe	Nom utilisateur Mot de passe	VARCHAR(40) VARCHAR(40)

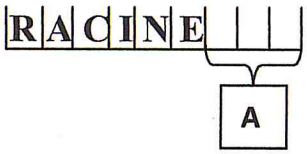
Tableau 9 : description détaillée des attributs des classes d'objets.

b) Les attributs des classes d'associations

Classes	Attributs	désignation	Type/taille
mdl_typeC ontr	code_mdl code_type_contr poids	Code module Code type contrôle poids	VARCHAR(22) VARCHAR(4) INT
notes_contr ole	matricule val appre code_mdl code_tc code_vltr_01 date_01 etat_01 code_vltr_02 date_02 etat_02 code_vltr_03 date_03 etat_03 code_prom	Matricule Etudiant Valeur Appréciation Code module Code type de contrôle Code validateur N°1 Date validation N°1 Etat de validation N°1 Code validateur N°2 Date validation N°2 Etat de validation N°2 Code validateur N°3 Date validation N°3 Etat de validation N°3 Code promotion	VARCHAR(20) FLOAT VARCHAR(40) VARCHAR(22) VARCHAR(4) VARCHAR(40) DATE VARCHAR(20) VARCHAR(40) DATE VARCHAR(20) VARCHAR(40) DATE VARCHAR(20) VARCHAR(16)
Prfl_act	identifiant_pr id_action	Identifiant profil Identifiant action	INT INT
promotion_ section_pha se	code_promo code_sction code_phase	Code promotion Code section Code phase	VARCHAR(16) VARCHAR(36) VARCHAR(15)
group_etd	code_grp matr_etd	Code groupe Matricule étudiant	VARCHAR(40) VARCHAR(20)
ens_mod_s ect	code_ens code_mod code_sec	Code enseignant Code module Code section	INT VARCHAR(22) VARCHAR(36)
Remarque	mat_enseign matrc_etud code_mod date description	Code enseignant Matricule étudiant Code module Date Description de la remarque	INT VARCHAR(20) VARCHAR(22) DATE VARCHAR(50)
ens_mod_g rp	code_ens code_mod code_grp	Code enseignant Code module Code groupe	INT VARCHAR(22) VARCHAR(40)
Appartient	code_et code_gr	Code étudiant Code groupe	VARCHAR(20) VARCHAR(40)

3.3.2 Codification :

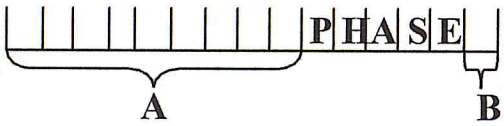
1. Code de la formation racine



A : désigne le numéro séquentiel de la formation racine sur 3 positions numériques.

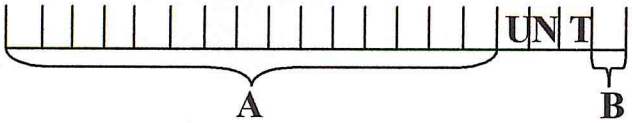
Exemple : « RACINE143 »

2. code formation phase



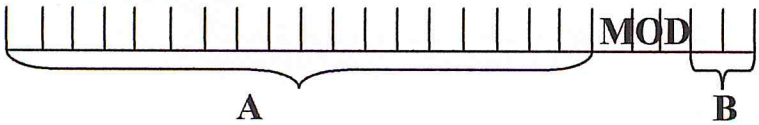
A : désigne le code de la formation racine sur 9 positions
B : désigne le numéro séquentiel de la phase de formation sur 1 position numérique.
Exemple : «« RACINE004PHASE3»»

4 code unité de formation :



A : désigne le code de la phase de formation sur 15 positions
B : désigne le numéro séquentiel de l'unité de formation sur 1 position numérique.
Exemple : ««RACINE013PHASE1UNT2 »»

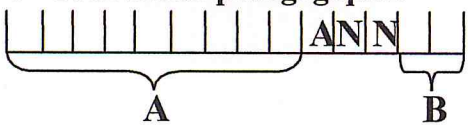
5 code module :



A : désigne le code de l'unité de formation 18 positions

B : désigne le numéro séquentiel des modules sur 1 position

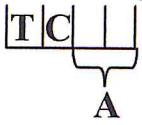
6 Code année pédagogique :



A : désigne le code de la formation racine sur 9 positions

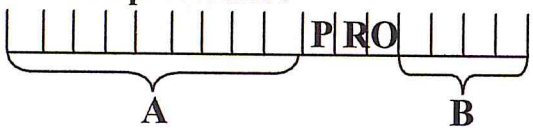
B : désigne le numéro séquentiel des années pédagogique sur 2 positions.

7 Code type de contrôle :



A : désigne le numéro séquentiel de type de contrôle sur 2 positions

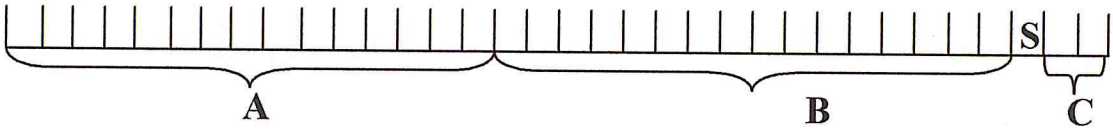
8 Code promotion :



A : désigne le code la formation racine sur 9 positions

B : désigne le numéro séquentiel sur 4 positions.

9 Code section :

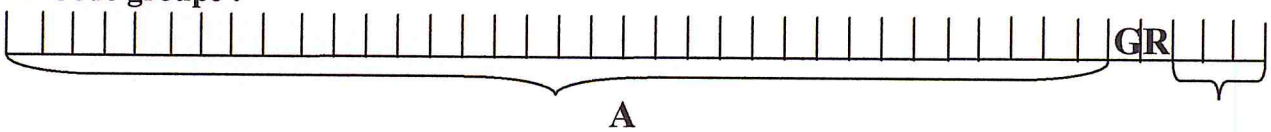


A : désigne le code de la phase sur 17 positions

B : désigne le code de la promotion sur 16 positions

C : désigne le numéro séquentiel sur 2 positions.

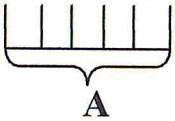
10 Code groupe :



B

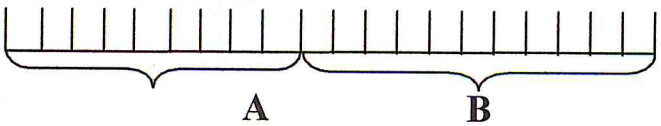
A : désigne le code de la section sur 36 positions
B : désigne le numéro séquentiel sur 3 positions

11 Code enseignant :



A : désigne le numéro séquentiel sur 5 positions

12 Code étudiant :



A : désigne le code de la formation racine sur 9 positions.
B : désigne un numéro séquentiel sur 11 positions.

13 Code profil :

A : désigne le code de profil sur 3 positions.

Exemple : ««etd»» signifier le profil étudiant, ««ens»» signifier le profil enseignant

14 Code action:

A : désigne le code de l'action sur 3 positions.

3.2.3 Les méthodes des classes d'objets et des classes d'associations :

Classe	Méthodes	Description
Type de contrôle	Persist () createTable() getTypeDeControle()	Permet d'ajouter un type de contrôle Permet de créer la table type de contrôle Permet de retourner une liste de type de controle
Formation racine	Persist () getListeFormation	Permet d'ajouter une formation racine Permet de retourner une liste de formation racine
Phase de Formation	Persist () getListePhase	Permet d'ajouter une phase de formation Permet de retourner une liste de formation racine
Unité de formation	Persist () getListeUnité	Permet d'ajouter une unité de formation Permet de retourner une liste d'unités de formation
module	Persist () getListeModules	Permet d'ajouter un module Permet de retourner une liste de modules
Année pédagogique	Persist () getListeAnnéePedagogique	Permet d'ajouter une année pédagogique Permet de retourner une liste d'années pédagogique

Promotion	Persist () getListePromotion	Permet d'ajouter une promotion Permet de retourner une liste de promotions
Section	Persist () getListeSectionDePromotion	Permet d'ajouter une section Permet de retourner une de sections de la promotion
Groupe	Persist () getListeGroupeDeSection getListeGroupeDeResponsable	Permet d'ajouter une groupe Retourne les groupes de la section Retourne les groupes de responsable
Etudiant	Persist () getEtudiantDeTuteur() getEtudiantDePromotion() getEtudiantDeSection() getEtudiantDeGroupe() modifierEtudiant() supprimer Etudiant()	Permet d'ajouter un étudiant Retourne une liste d'étudiants de tuteur Retourne liste d'étudiants de promotion Retourne liste d'étudiants de section Retourne liste d'étudiants de groupe Permet de modifier l'étudiant Permet de supprimer l'étudiant
Enseignant	Persist() getListeDensiegnant	Permet d'ajouter un enseignant Retourne la liste des enseignants
Profil	Persist() getListeProfil() modifierProfil()	Permet d'ajouter un profil Retourne la liste des profils Permet de modifier le profil
Action	Persist() getListeAction()	Permet d'ajouter une action Retourne la liste des actions
Compte utilisateur	Persist() ModifierCompte() SupprimerCompte() getListeCompte()	Permet d'ajouter un compte Permet de modifier un compte Permet de supprimer un compte Retourne liste des comptes
mdl_typContr	Persist() getTypeDeControleDeModule() getListeTypeControle	Permet d'affecter un TC au module Renvoi les type de contrôle de module Renvoi la liste des types de contrôle
notes_controle	Perist() getNoteControle()	Permet d'affecter une note de contrôle à un étudiant Renvoie la note d'un étudiant

Prfl_act	Perist() getActionDeProfil()	Permet d'affecter un TC au module Renvoi les actions de profil
promotion_section_phase	Perist() getSectionDePromotion()	Permet d'affecter une promotion a une phase Renvoi liste des sections de la promotion
ens_mod_sect	affecterSetionEnsiegnant() affecterModuleEnsiegnant()	Permet d'affecter une section a une enseignant Permet d'affecter un module a une enseignant
Remarque	Persist() affecterRemarqueEtudiant()	Permet d'ajouter une remarque Permet d'ajouter une remarque à un étudiant
ens_mod_grp	affecterGroupeEnsiegnant() affecterModuleEnsiegnant()	Permet d'affecter un groupe à une enseignant Permet d'affecter un module à un enseignant
Appartient	Perist() getEtudiantDeGroupe()	Permet d'affecter un groupe a un etudiant() Permet de renvoyer les etudiants de groupe

3.2.4 Diagramme des classes :

Cette partie est consacrée à la présentation de diagramme de classes du domaine. En analyse ce diagramme a pour objectif la description de la structure des entités manipulées par les utilisateurs en décrivant les relations entre ces classes.

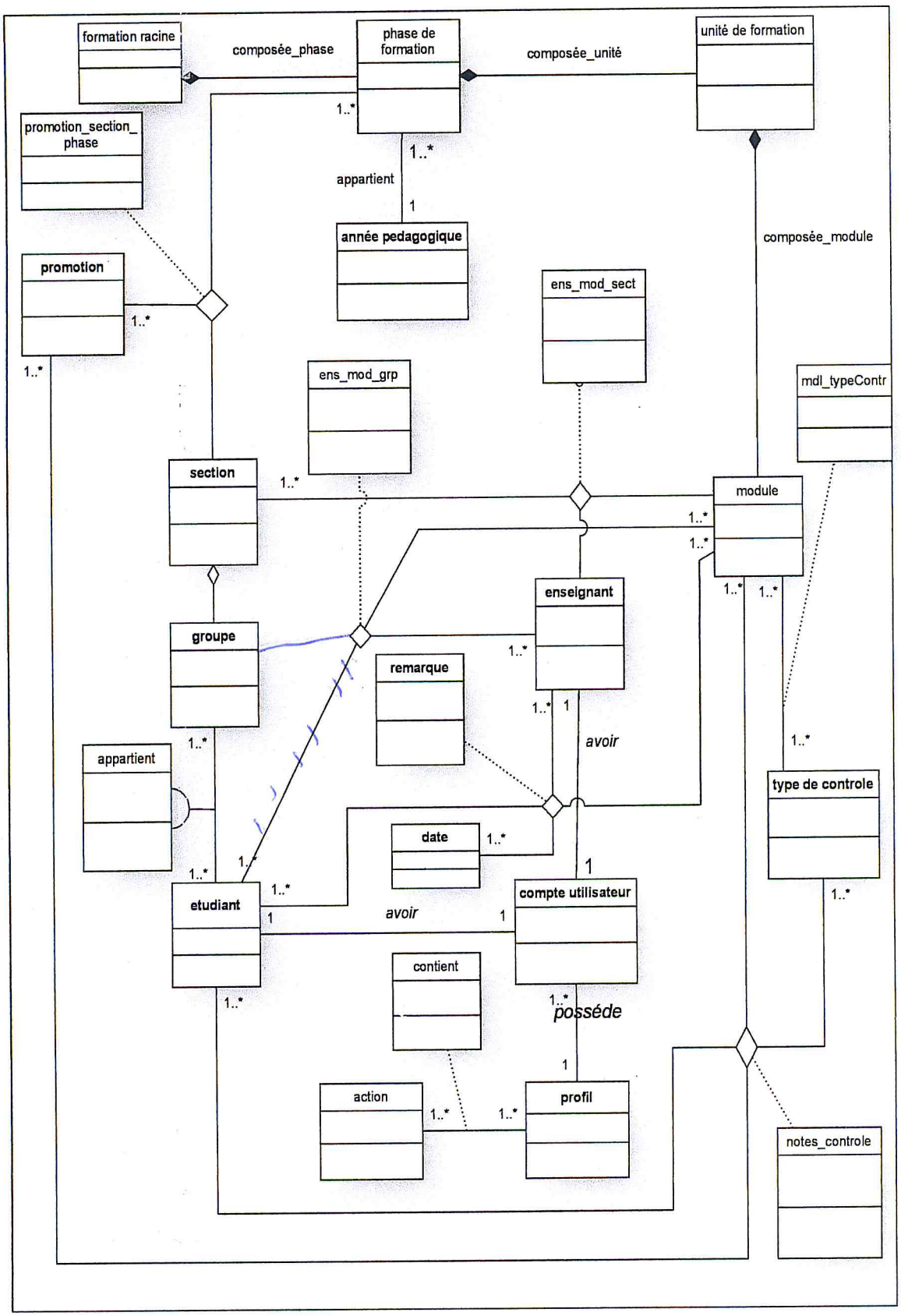


Figure 40 : diagramme des classes de l'application EFormation.

3.2.5 Le passage au modèle relationnel :

- **Type de contrôle** (code_tc, nom_tc, descr).
- **Formation racine** (codeFr, code_racine, code_parent, code_absolu, nomFr, poids, domaine, filiere, specialite, date_debut, date_fin, credit).
- **Phase de Formation** (codePf, code_racine, *code_parent, code_absolu, nomPf, poids, credit, *code_annee).
- **Unité de formation** (codeUf, code_racine, *code_parent, code_absolu, nomUf, poids,credit).
- **Module** (codeMd, code_racine , *code_parent, code_absolu, nomMd, poids, credit).
- **Remarque** (mat_enseign, matrc_etud, code_mod, date, description)
- **Année pédagogique** (code_annee, nom_annee).
- **Promotion** (codePr, nomPr, description, date).
- **Section** (codeSec, NomSec, description, taille).
- **Groupe** (codeGr, nomGr, description, activite, taille, *codeSec).
- **Etudiant** (matricule, puid, pere_puid, mere_puid, nom, prenom, date_naissance, lieu_naissance, inst_origine, matricule_origine).
- **Enseignant** (id_ens, nom, prenom, sex, adresse, n_telephone, grade, fonction).
- **Profile** (Identifiant_Pr, Type_Pr).
- **Action** (Id_Action, Nom_Action).
- **Compte utilisateur** (Nom_util, Mot_passe,*Identifiant_Pr).
- **mdl_typeContr** (code_mdl, code_type_contr, poids).
- **notes_controle** (matricule, val, appre, code_mdl, code_tc, code_vltr_01, date_01, etat_01, code_vltr_02, date_02, etat_02, code_vltr_03, date_03, etat_03, code_prom).
- **Prfl_act** (identifiant_pr, id_action).
- **Promotion_section_phase** (code_promo, code_section, code_phase).
- **group_etd** (code_grp, matr_etd).
- **ens_mod_sect** (code_ens, code_mod, code_sec).
- **ens_mod_grp** (code_ens, code_mod, code_grp).
- **Appartient** (code_et, code_gr).
-

3.3 Conclusion:

Nous avons présenté dans cette partie l'étude conceptuelle que nous avons modélisé avec les diagrammes UML suivant : modèle fonctionnels (use case), dynamique (diagramme de séquence), et statique (diagramme des classes).

Conception de l'aspect communication de système

4.1 Introduction :

A partie de l'étude comparative entre les déférents systèmes de communication vue dans les chapitres précédents, nous avons choisis de réaliser notre système de communication selon l'architecture orienté service, qui est la plut adapter a notre conception.

Dans ce présent chapitre, nous allons traiter la conception du système de communication en utilisant langage SoaML(Service-oriented architecture Modéling language).

4.2 Présentation du langage SoaML :

SoaML est une spécification présentée par l'organisation OMG en 2009[20], dans le but de fournir à l'utilisateur du langage UML (unified modeling language) les moyens de modéliser une architecture orienté services, comprenant donc des notions de consommateur et de fournisseur de service, ainsi que la notion de contrats [23].

SoaML propose plusieurs types de diagrammes :

- diagramme d'architecture des services
- diagramme de processus métier
- diagramme de contrat des services
- diagramme d'interface des services.

4.3 Présentation de la méthode de conception :

Cette méthode exige la construction de trois modèles :

- Modèle d'architecture métier.
- Modèle d'architecture system.
- Modèle de plateforme spécifique (JEE, Service Web….).

4.4 Outil de conception :

Les diagrammes présentés dans ce chapitre ont été établit en utilisant l'outil « **Modélio**» avec le module « **Modélio SoaML Designer**» [21].

Modelio est un outil de modélisation des systèmes et logiciels, qui propose une distribution open source depuis octobre 2011. Il comprend plusieurs fonctionnalités intéressantes, y compris le support de tous les standards actuels de modélisation (UML, BPMN, SOA, ...), et la génération de code Java [22].

4.5 Architecture de système :

Nous allons proposer de construire une architecture orienté service tel que chaque entité de système global (instance de e_Formation, Synchroniser) expose et consomme des services et occupe un niveau bien précis dans cette architecture, La figure suivante schématise la solution proposée :

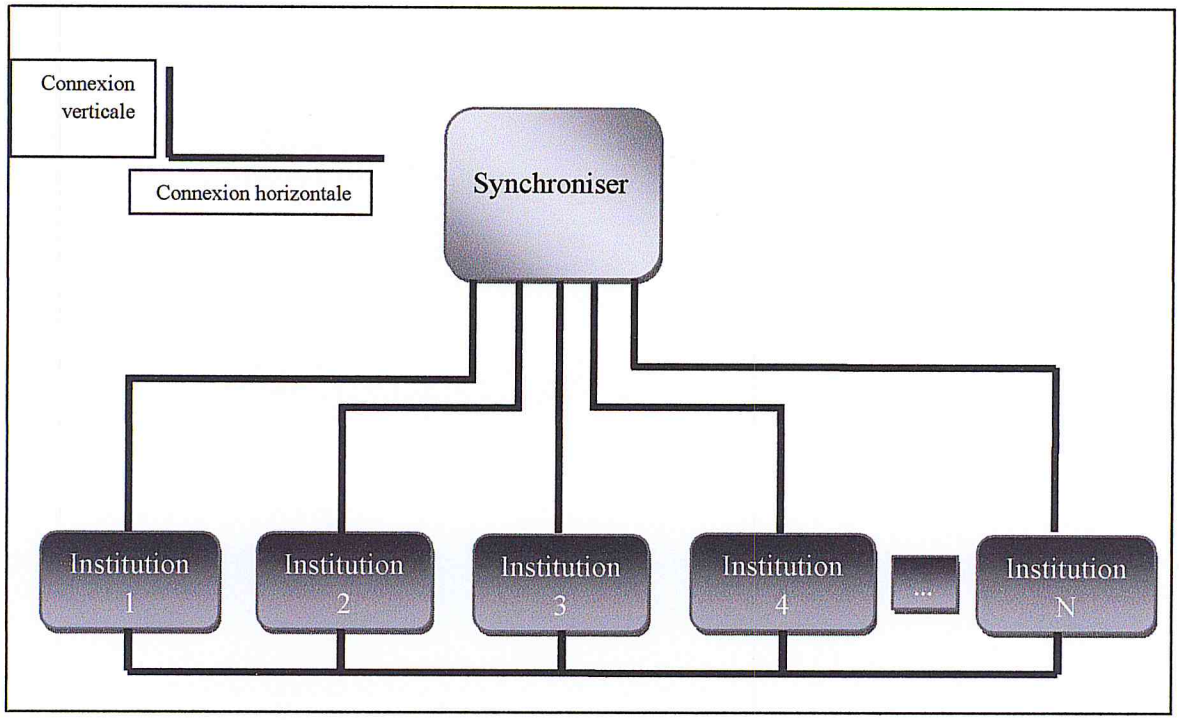


Figure41 : Architecture globale du système de communication

4.5.1 Les entités du système :

Nous avons classifié les entités du système selon la nature des services fournis en deux niveaux :

1. **Synchroniser :** cette entité offre un service nommé <<**Synchroniser**>>, ce dernier guide et surveille la communication entre toutes les autres entités.
2. **Institution**(e_Formation) **:** cette entité représente soit un département de l'université, un lycée, un CEM, un primaire ou un centre de formation professionnelle, chaque institution offre les même services : retrait des relevés de notes, de certificat scolarité, certificat disciplinaire…etc.

4.5.2 Types de communication :

- Horizontale : c'est la communication entre les entités de même nature (e_Formation).
- Verticale : c'est la communication entre deux entités de nature différente (entre les applications e_Formation et le **synchroniser**).

4.6 Conception du système :

On devise les services en deux : Services Institution et synchroniser :

4.6.1 Service Institution (e_Formation) :

L'application e_Formation expose les services suivants :

- Service de retrait des relevés de notes.
- Service de retrait de certificat de scolarité.
- Service de retrait des certificats disciplinaires.

I. Description textuelle :

a. Service retrait de relevé de notes :

Ce service permet de récupérer un relevé de note via l'identifiant de l'étudiant et les phases sélectionnées :

- **getReleveDeNoteIdInst** (in matricule ,in []phase, in idInstiution) :[]relevé de notes

➔ Récupère touts les relevés de notes pour associer aux phases envoyer dans l'institution idInstitution.

- **getAllReleveDeNote**((in matricule, in []phase) :[] relevé de notes

➔ Récupère touts les relèves de notes associer aux phases données dans touts les institutions ou il est étudier.

- **getAllReleveDeNoteAdmis**((in matricule, in []phase) :[] releve de note

➔ Récupère touts les relèves de notes admis associer au phases données dans touts les institutions ou il est étudier.

b. Service de retrait de certificat de scolarité :

Il fournit les opérations qui renvoient une certificat de scolarité

- **getCertificatScolarite**(in matricule, in idInstitution) :certificat de scolarité de l'institution « idInstitution ».

c. Service de retrait des certificats disciplinaires :

- **getCertificatDesciplinaireIdInst** (in matricule ,in idInstitution) : certificat disciplinaire de l'institution idInstitution.

- **getCertificatDesciplinaireTypeInst**(in matricule, in typeInstitution) :[] certificat disciplinaire

➔ Récupère tout les certificats disciplinaires des institutions de type «typeInstitution».

➢ **getCertificatDesciplinaireAll**(in matricule) :[] disciplinaire

➔ Récupère tout les certificats disciplinaires des institutions depuis son première inscription.

II. Modèle architecture métier :

1) les participants dans ce modèle :

Au niveau de ce modèle, nous allons identifier trois participons :

➢ **Utilisateur** : chef de formation, agent de scolarité, chef d'institution, administrateur

➢ **e_Formation** (Institution) : il joue deux rôles

- Consommateur du service :
  - Administrateur : dans le cas ou l'action à accomplir nécessite une présence humaine
  - e_Formation : dans le cas ou l'action à réaliser est automatique et ne nécessite aucune surveillance.
- Fournisseur de service (Fe_Formation):

➢ synchroniser **:** joue le rôle de fournisseur des services.

2) Diagramme d'architecture des services :

La figure suivante représente l'architecture globale des services proposés par F_eformation.

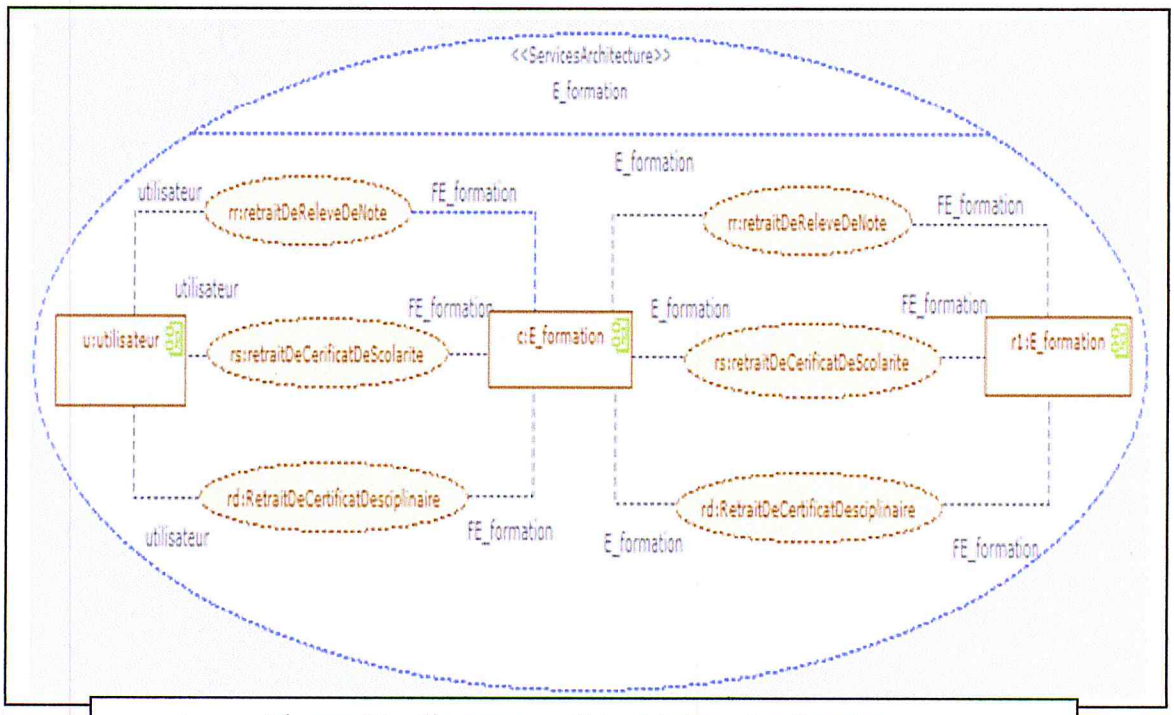


Figure 43 : diagramme d'Archicture des Services

«Processus de retrait des relèves de notes durant le cursus d'un étudiant » :

Les activités utilisées :

- instOrigine : code de l'institution d'origine de l'étudiant
- matOrigine : matricule de l'étudiant dans l'institution d'origine

Ce processus est récursif, il fait appel au service de localisation d'institution et à lui-même avec le scénario suivant :

1. Le système récupère les relèves de notes existantes localement pour la formation ou il est inscrit d'étudiant
2. Il fait appel au synchroniser pour avoir l'adresse de son institution origine (l'institution d'origine fait le même travail avec la 2eme institution d'origine...).Les appels s'arrêtent lorsqu'on trouve une institution où l'origine de l'étudiant inscrit est une formation différente que celle demandée au début.
3. A la fin la première institution récupère tous les relevés de notes appartient au cursus de l'étudiant pour la formation demandée.

« Processus de retrait des certificats disciplinaires durant le cursus d'un étudiant » :

Le même scénario que le premier sauf qu'il récupère des certificats disciplinaires

4) **Diagramme de contrats des services :** nous allons décrire les services suivant :

- Service de retrait de relevé de note
- Service de retrait de certificat disciplinaire
- Service de retrait de certificat de scolarité

Pour chaque rôle d'un participant un diagramme de contrat associé est établi:

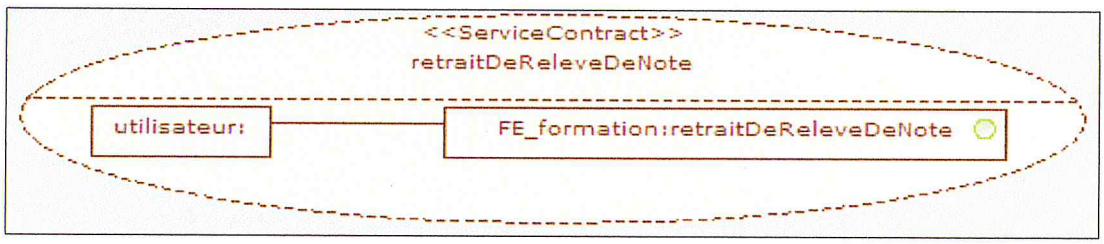


Figure 43 : Contrat du service de retrait de relevé de note pour le client du rôle utilisateur

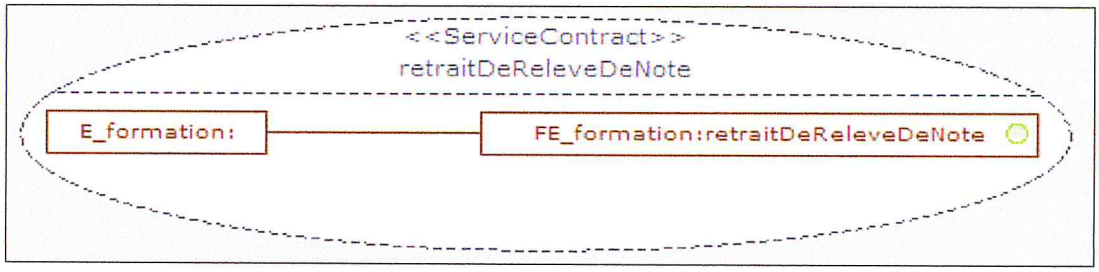


Figure 44 : Contrat du service de retrait de relevé de note pour le client du rôle e_Formation

Figure 45 : Contrat du service de retrait de certificat disciplinaire pour le client du rôle utilisateur

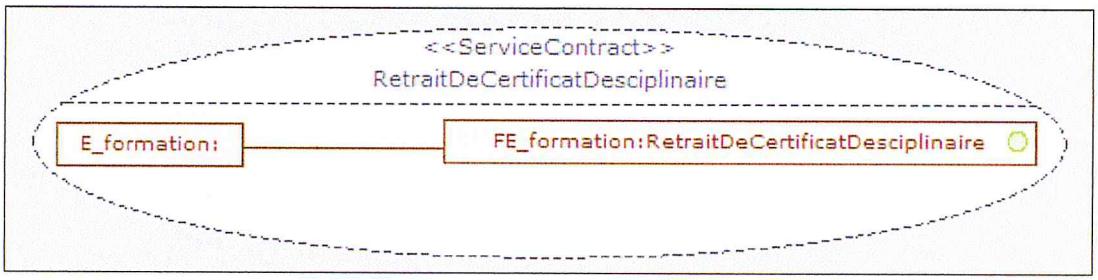


Figure 46 : Contrat du service de retrait de certificat disciplinaire pour le client du rôle e_Formation

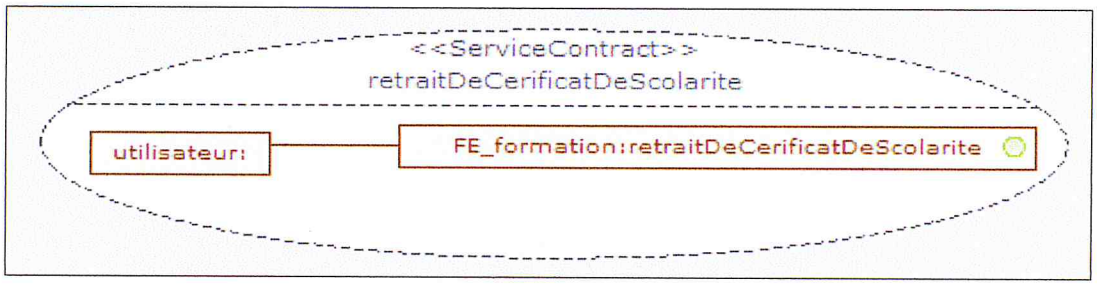


Figure 47 : Contrat du service de retrait de certificat scolarité pour l'utilisateur

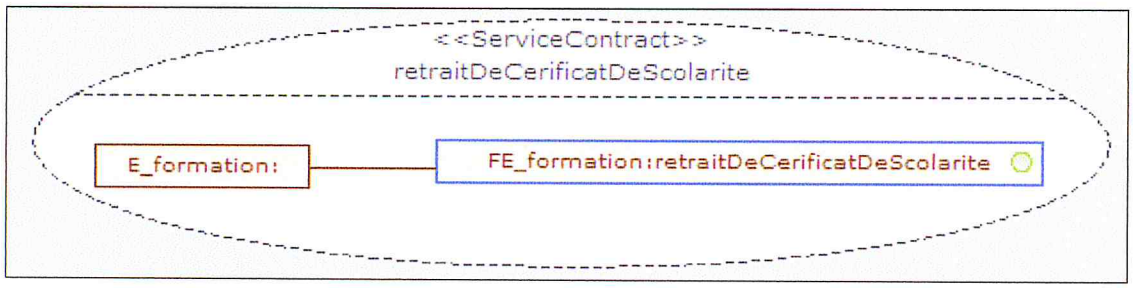


Figure 48 : Contrat du service de retrait de certificat disciplinaire pour le client du rôle e_Formation

III. Modèle architecture système :

1) Diagramme d'interface du service :

Dans cette section, on décrit les services suivants :

- Service de retrait de relevé de note
- Service de retrait de certificat disciplinaire
- Service de retrait de certificat de scolarité

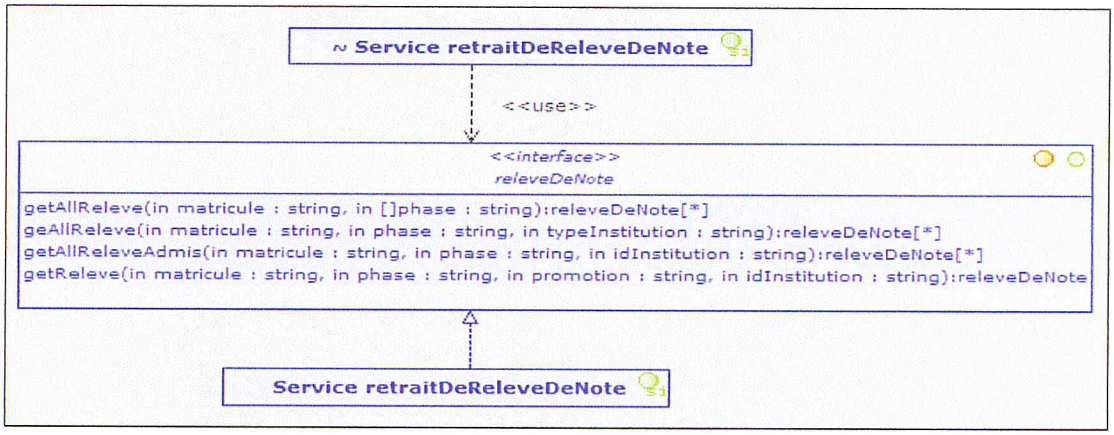


Figure 49 : Interface de service de relève e note

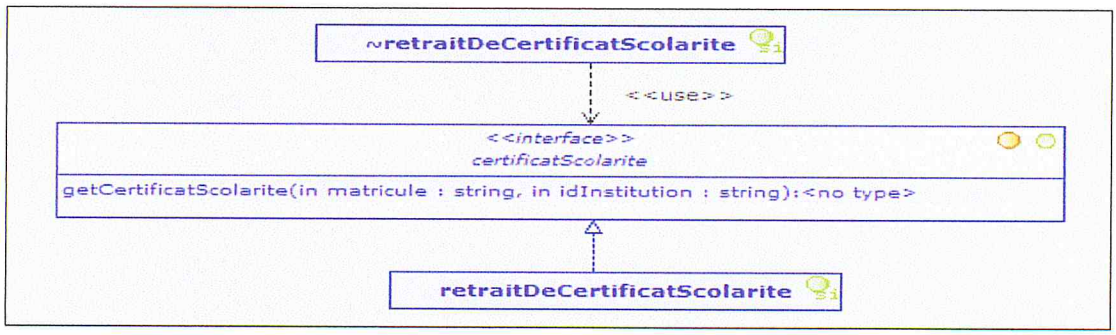


Figure 50 : Interface de service de certificat de scolarité

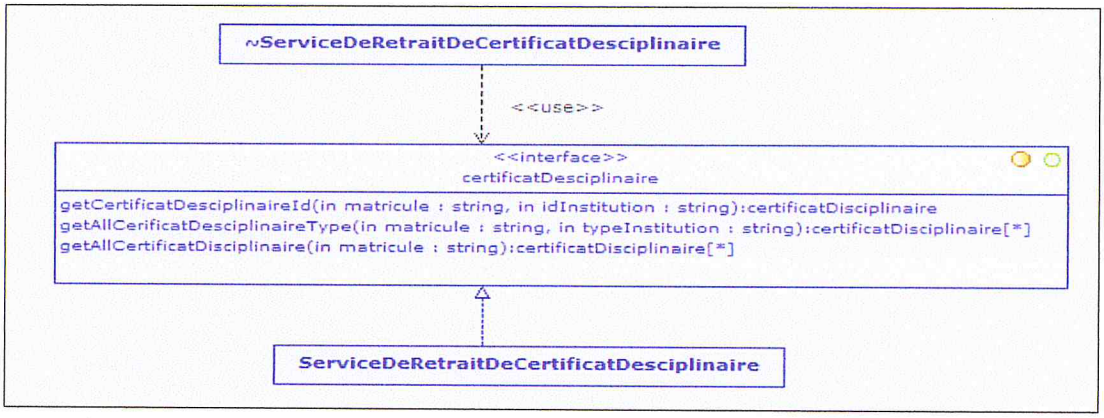


Figure 51 : Interface de service de certificat disciplinaire

2) **Diagramme des messages :** Dans cette section, on décrit les services suivants :
   - Service de retrait des relevés des notes.
   - Service de retrait de certificat de scolarité.
   - Service de retrait des certificats disciplinaires.
   - Service consultation des notes.

Dans cette partie on s'intéresse a étudier la structure et la forme des messages échangés en représentant quelques diagrammes des messages.
Le tableau suivant donne pour chaque opération la structure globale des messages requête et réponse du **service de retrait des relevés de notes**.

Opération	Type de message	Forme globale	Structure détaillée
getReleveDeNoteIdInst	Requête	Chaine de caractères	---
	Réponse	**ReleveDeNote**	Voir figure
getAllReleveDeNote	Requête	Chaine de caractère	---
	Réponse	**ReleveDeNote**	Même figure
getAllReleveDeNoteAdmis	Requête	Chaine de caractères	---
	Réponse	**ReleveDeNote**	Même figure

Tableau 3 : Diagramme des messages pour le service retrait de relevé de note.

Diagramme de message : Relevé de notes

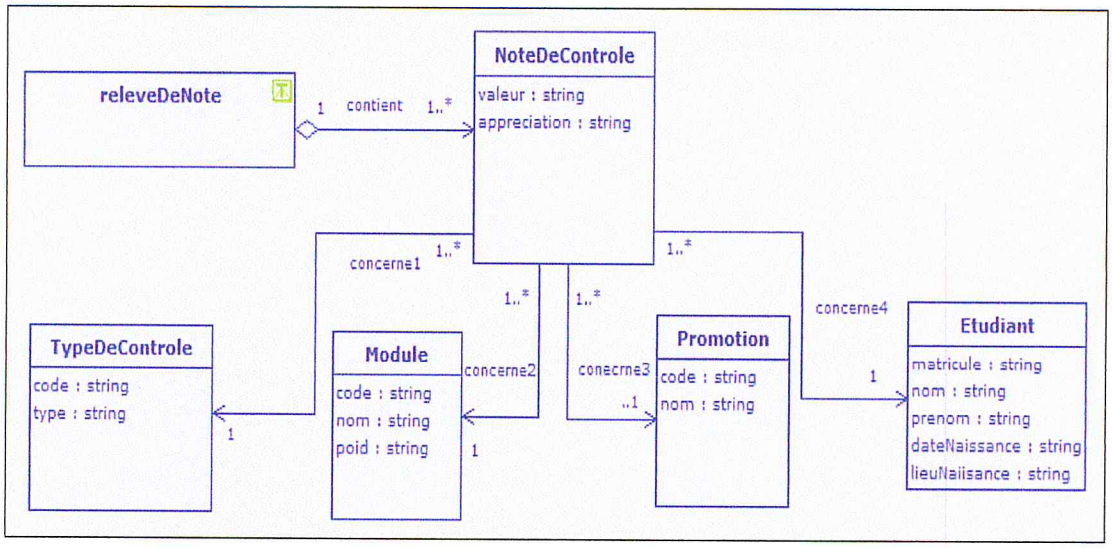


Figure 52 : Type de message Relevé de note

Le tableau suivant donne pour chaque opération la structure globale des messages requête et réponse du **service de retrait de certificat de scolarité.**

Opération	Type de message	Forme globale	Structure détaillé
getCertificatScolarite	Requête	Chaine de caractère	----
	Réponse	**CertificatScolarite**	Voir figure

Tableau 4 : structure globale des messages requête et réponse du service de retrait de certificat de scolarité

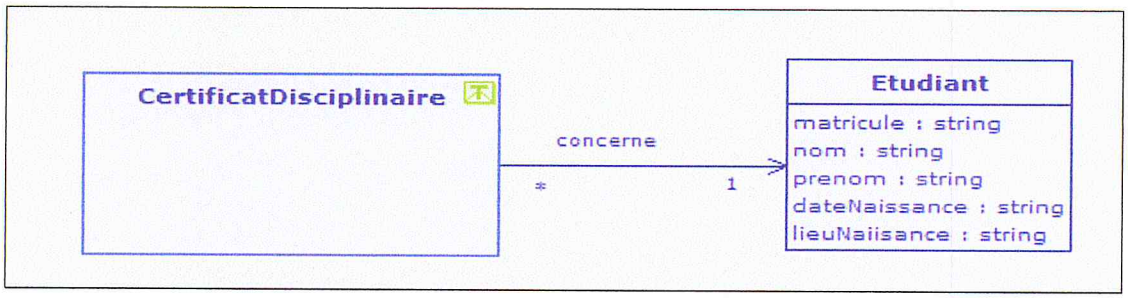


Figure 53 : Type de message certificat disciplinaire

Le tableau suivant donne pour chaque opération la structure globale des messages requête et réponse du service de retrait de certificat disciplinaire

Opération	Type de message	Forme globale	Structure détaillée
getCertificatDesciplinaireIdInst	Requête	Chaine de caractères	---
	Réponse	**CertificatDesciplinaire**	Voir figure
getCertificatDesciplinaireTypeInst	Requête	Chaine de caractère	---
	Réponse	**CertificatDesciplinaire**	Même figure
getCertificatDesciplinaireAll	Requête	Chaine de caractères	---
	Réponse	**CertificatDesciplinaire**	Même figure

Tableau 5 : structure globale des messages requête et réponse du service de retrait de certificat disciplinaire

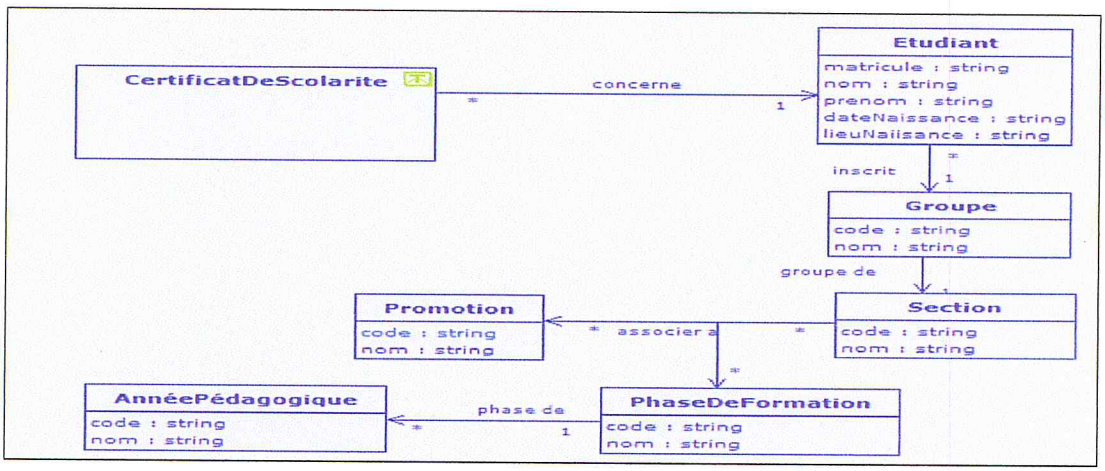


Figure 54 : Type de message Certificat de scolarité

4.6.2 Service Synchroniser :

L'application synchroniser fournit les services suivants :

- Service d'allocation de configuration réseau pour les applications e_Formation.
- Service de localisation des applications e_Formation.

I. Description textuelle :

a. Service d'allocation de configuration réseau :

Son but est de sauvegarder la configuration réseau d'une application e_Formation dans le synchroniser lors de son installation.

allocatee_Formation(in ide_Formation, in nome_Formation , in typeE_formation, in adressee_Formation, in dbNamee_Formation, in adresseDbe_Formation):

Renvoi une chaine de caractère :

- Ok : si l'enregistrement a été fait avec succès
- Erreur : s'il ya une erreur lors d'enregistrement

b. **Service de localisation des applications e_Formation :**

Son but est de fournir la configuration réseau de l'application identifiée par ide_Formation

localisee_Formation(in ide_Formation) :-> configuration réseau sous forme d'une chaine de caractères

II. Modèle architecture métier :

a. Diagramme d'architecture des services : la figure représente l'architecture globale des services de synchroniser.

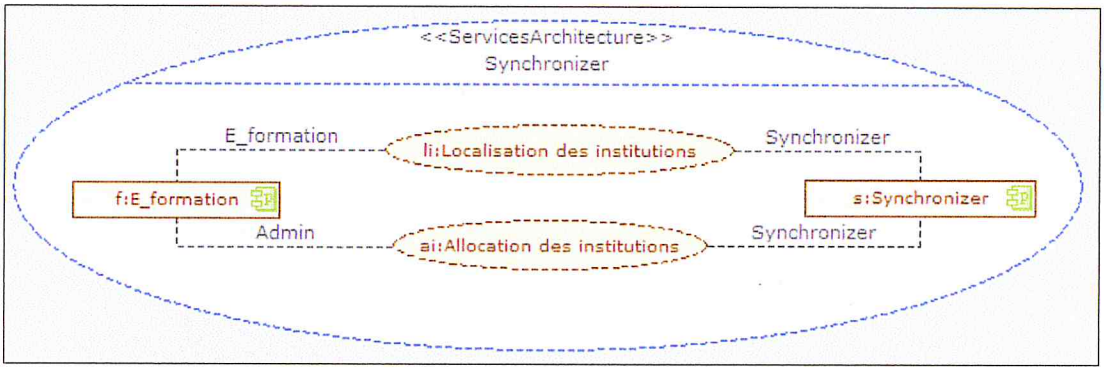


Figure 55 : l'architecture globale des services de synchroniser.

b. Diagramme de contrat des services :

La figure suivante représente le contrat du service d'allocation de configuration réseau pour l'application e_Formation lors de l'installation

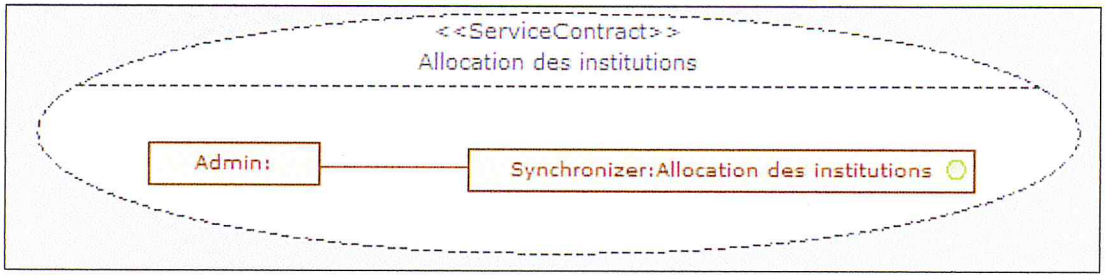


Figure 56 : contrat du service d'allocation de configuration réseau

La figure suivante représente le contrat du service de localisation d'une application e_Formation .

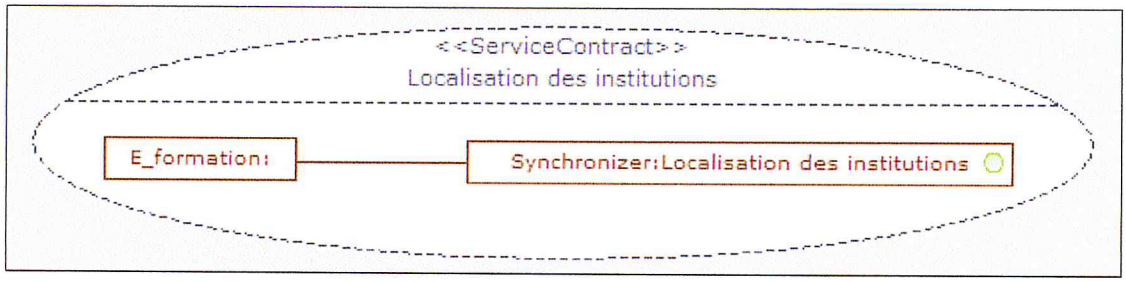


Figure 57 : contrat du service de localisation d'une application e_Formation

III. modèle architecture système :

a. Diagramme d'interface du service :

La figure suivante décrit l'interface service d'allocation d'une application e_Formation

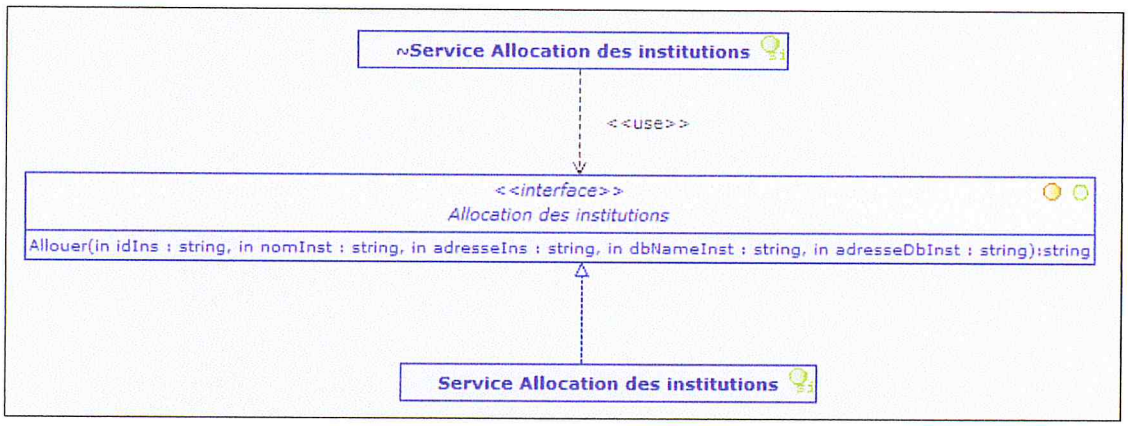


Figure 58 : l'interface service d'allocation d'une application e_Formation

La figure suivante décrit l'interface service de localisation d'une application e_Formation en question

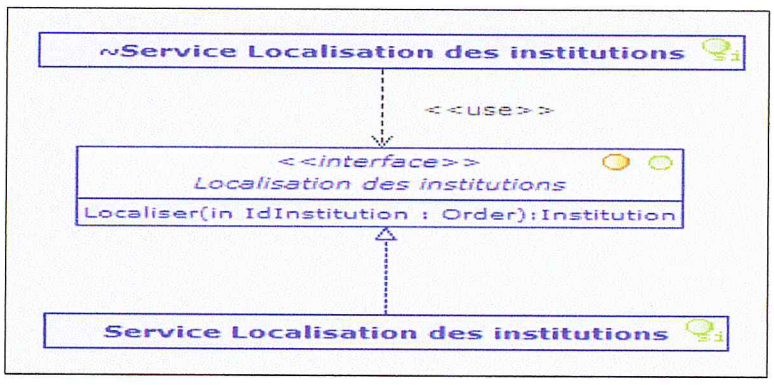


Figure 59 : localisation d'une application e_Formation

Diagramme des messages : Vu que ces services ne manipulent que des chaines de caractères, nous n'avons pas des diagrammes de messages à établir.

Diagrammes des participants

Au niveau de cette partie, nous avons identifié en générale les participants ainsi que les services étudiés dans la conception de système de communication.

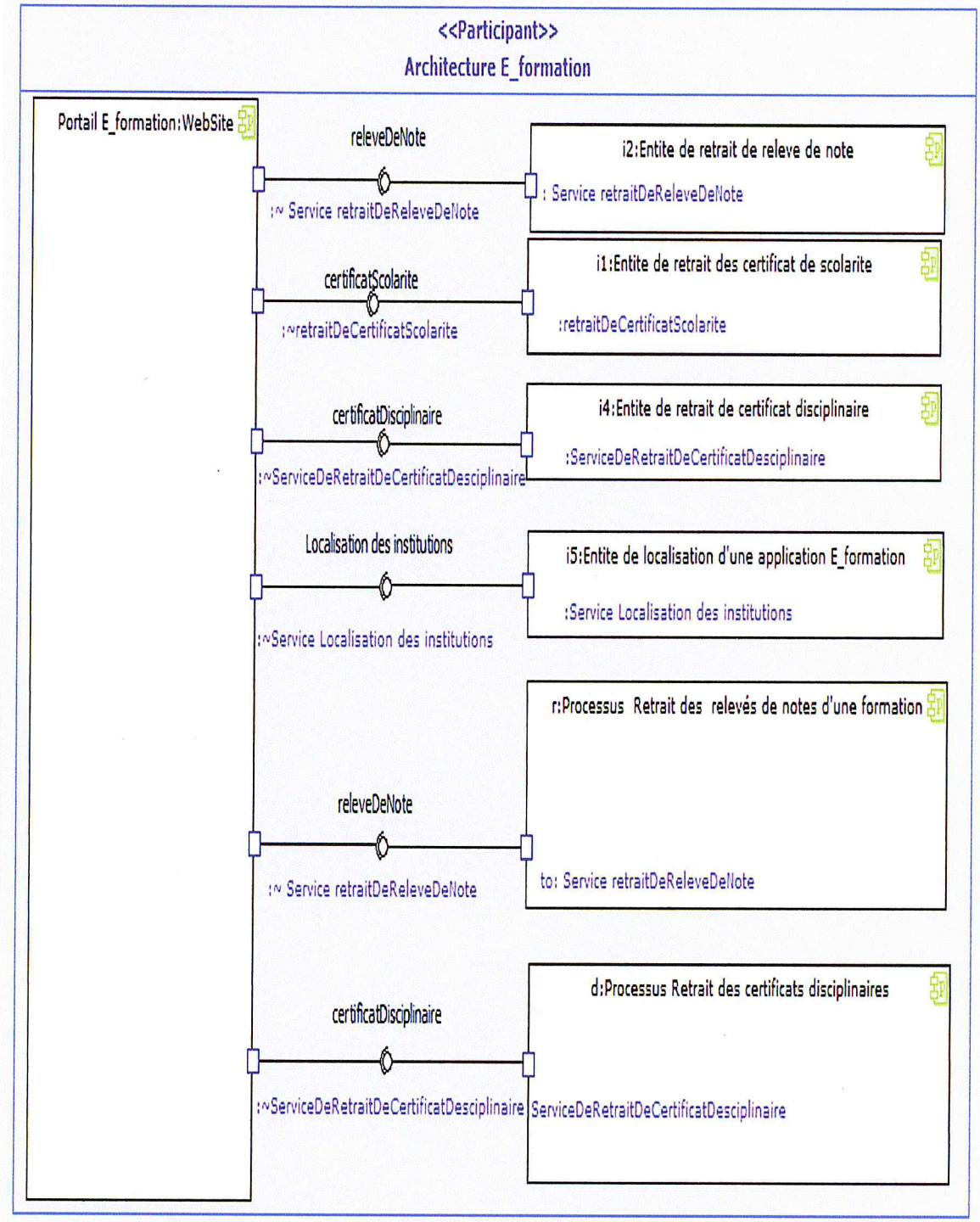


Figure 60 : Diagramme des composants du système.

Les deux dernières entités qui représentent des processus sont détaillées par les diagrammes des figures suivants:

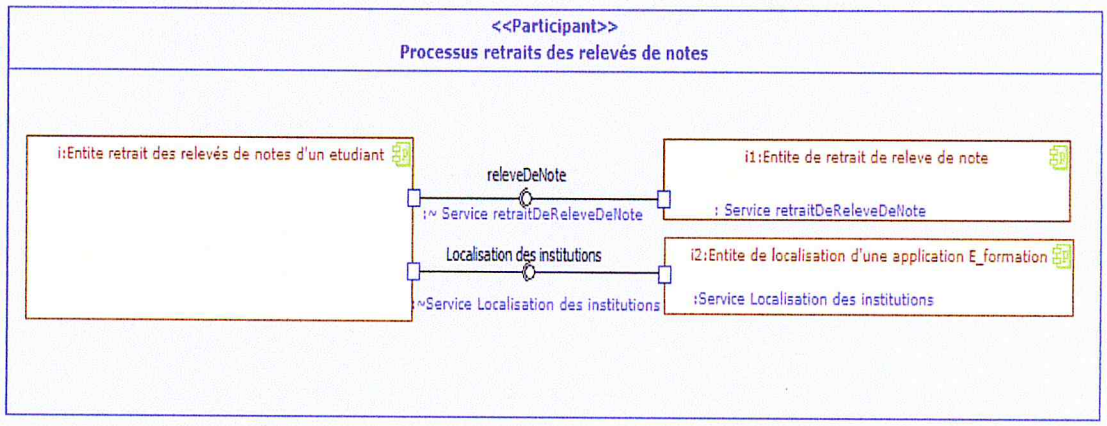


Figure 61 : Diagramme du participant » Processus retraits des relevés de notes»

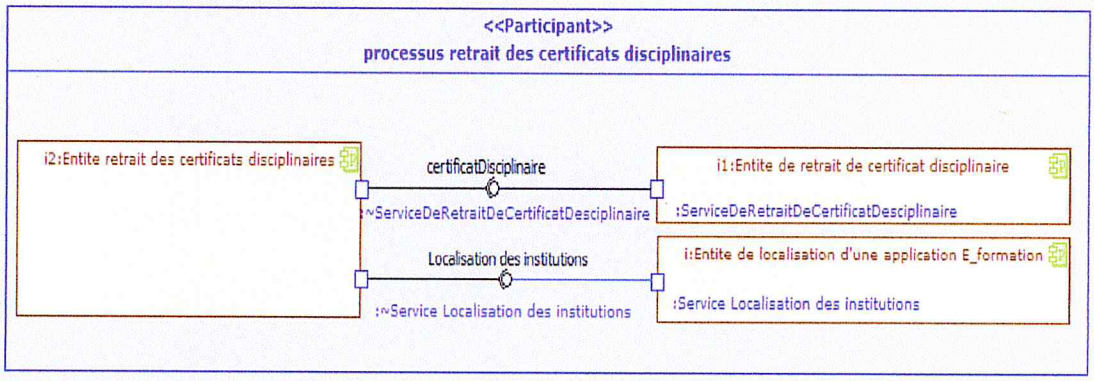


Figure 62: Diagramme du participant «Processus retraits des certificat disciplinaires »

Conclusion :

Nous avons présenté dans ce chapitre la description détaillée des services ainsi que les diagrammes et interfaces associer en utilisant le langage SoaML.

Ce chapitre nous a aidés pour structurer notre travail pratique concernant la programmation des classes java responsable a la manipulation des services web.

Implémentation de système

5.1 Introduction

Apres la présentation de notre système dans les chapitres précédents, dans cette dernière partie, nous allons décrire le modèle de déploiement et l'implémentation de notre système. Dans ce qui suit nous tenons à présenter et argumenter nos choix des technologies utilisées pour le développement de système, tel que les langages de programmation, le SGBD, etc. Nous étalerons après les différentes fonctionnalités offertes par notre système. Cela sera illustré par des prises d'écrans permettant d'expliquer au mieux le fonctionnement de notre application.

5.2 Environnement de développement

Nous présentons dans cette section, les outils utilisés pour le développement de notre système (serveur web, SGBD, langage de programmation…etc).

5.2.1 Environnement de développement logiciel :

Choix du serveur web :

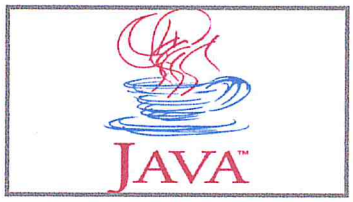
Apache Tomcat : Tomcat est un serveur web qui gère les servlets J2EE et les JSPs .Il est souvent employer en combinaison avec un serveur web apache : Apache s'occupe de toutes les pages web traditionnelles, et Tomcat uniquement les pages d'une application web java. Issu de projet JAKARTA, Tomcat implémente les spécifications des servlets et les JSPs de Sun Microsystems. Il inclut des outils pour la configuration et la gestion, mais peut également être configuré en éditant les fichiers de configuration XML. Comme Tomcat inclut un serveur http interne, il est aussi considérer comme un serveur http, il a aussi une plateforme pour le développement et le déploiement d'applications web et les web services.

Choix du SGBD :

Les SGBD libres et gratuits sont nombreux. **MySQL 5.5**, **Postgres**, en sont des exemples. Si nous avons choisi **MySQL**, c'est plus pour des raisons de performances et fonctionnalités offertes. Nous citeront dans la suite ses principaux avantages :

- **MySQL** est beaucoup moins complexe à installer et à administrer que d'autres systèmes.
- **MySQL** supporte le langage de requête **SQL** comme on peut lui accéder via **ODBC**.
- **MySQL** permet des connexions multiples en même temps et utilise différentes bases de données simultanément.
- **MySQL** dispose d'un système de contrôle qui interdit la consultation des données ceux qui n'en pas l'autorisation.

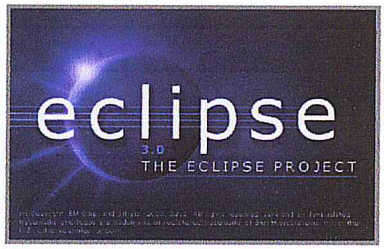
Choix des langages de programmation:



Notre choix du langage de programmation s'est porté sur le langage « Java » et cela pour diverses raisons :

- C'est un langage orienté objet dérivé du C, mais plus simple que le C++.
- Il est portable et indépendant de toutes plateforme : il est possible d'exécuter des programmes java sur tout les environnements qui possèdent une java virtuel machine.
- Java est multitâche : il permet l'utilisation des threads qui sont des unités d'exécution isolé.
- Java possède une riche bibliothèque des classes comprenant les fonctionnalités diverses telle que les fonctions standards le système de gestion des fichiers, les fonctions multimédia et beaucoup d'autre fonctionnalités.

Choix de l'Environnement de développement intégré :



Notre choix sur l'environnement de développement s'est porté sur Eclipse EE 3. 6.

Eclipse est un environnement de développement intégré libre extensible, universel et polyvalent, permettant de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions .

La spécificité d'Eclipse IDE vient du fait de son architecture totalement développée autour de la notion de plugin : toutes les fonctionnalités de cet atelier logiciel sont développées en tant que plug-in.

JSP (Java Server Page):

Le Java Server Pages ou JSP est une technique basée sur Java qui permet aux développeurs de générer dynamiquement du code HTML, XML ou tout autre type de page web. Cette technique permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique. Depuis la version 2.0 des spécifications, la syntaxe JSP est complètement XML. Les JSP sont compilées par un compilateur JSP pour devenir des servlets Java. Un compilateur JSP peut générer un Servet Java en code source Java qui peut à son tour être compilé par le compilateur Java, ou peut générer le pseudo-code Java interprétable directement.

JavaScript :

Pour avoir plus de fonctionnalités et d'interactivité avec l'utilisateur du système, nous avons utilisé Java Script, et spécialement AJAX (ou Asynchronous JavaScript And XML) qui n'est pas un langage, mais plutôt une méthode de développement web basée sur l'utilisation de JavaScript pour effectuer des requêtes à l'intérieur d'une page web sans recharger la page, qui va rendre plus interactifs notre système et offre une meilleure ergonomie.

JasperReports :

JasperReports est un outil de Reporting Open Source, offert sous forme d'une bibliothèque qui peut être embarquée dans tous types d'applications Java. JasperReports se base sur des fichiers jrxml (qui ont la structure d'un fichier XML) pour la présentation des états. Il permet d'exporter des rapports aux différentes formats (PDF, HTML, XLS, CSV, XML, RTF, TXT)[24].

Technologies des web services :

Apache Axis 2 : Apache Axis2, est la troisième génération de moteur de web services, après ces prédécesseurs apache SAOP et Apache Axis, non seulement il est plus efficace, modulaire et orienté XML, mais il est aussi flexible, extensible et il implémente des dispositifs puissants pour les entreprises, comme la sécurité et la fiabilité .Apache Axis 2 est la meilleur implémentation open source des web services ces trois dernière années.

Axis 2 puisse sa puissance et robustesse dans les quatre points suivants :

- Meilleur performance ;
- support de messagerie ;
- meilleur support pour les extensions des web services ;
- est un meilleur support de déploiement.

Pour l'implémentation des services web nous avons exploité deux plugins qu'on a installés dans l'environnement Eclipse et qui sont décrite ci-après.

a. Axis 2 Service Archive Generator Wizard :

le générateur d'archive de service, est un outil important qui permet la génération des archives d'un service web (fichier aar ou jar) qui serons déployés comme un service web dans Axis 2.

b. Apache Axis2 Code Generator Wizard :

C'est un outil qui permet de générer automatiquement d'une part le fichier wsdl a partir de code source d'un service (java2WSDL), et d'autre part, nous

utilisons l'utilitaire WSDL2java de ce plugin qui permet de générer a partir de la description wsdl d'un service les différente classes et interfaces clientes nécessaire a l'appel de ce service.

5.3 Déploiement de système :

5.3.1 Architecture technique :

L'architecture technique (également nommé architecture physique) d'écrit l'ensemble des composants matériels supportant l'application. L'architecture de notre système est décomposée en trois niveaux :

a) Architecture de l'application d'E-formation :

La figure suivante montre l'organisation en couche du l'application, chaque couche communique avec une couche adjacente.

L'intérêt de cette décomposition est de séparer le problème en plusieurs parties, chaque partie réalise une tache bien précise. Dans ce qui suit nous allons détailler le rôle de chaque couche.

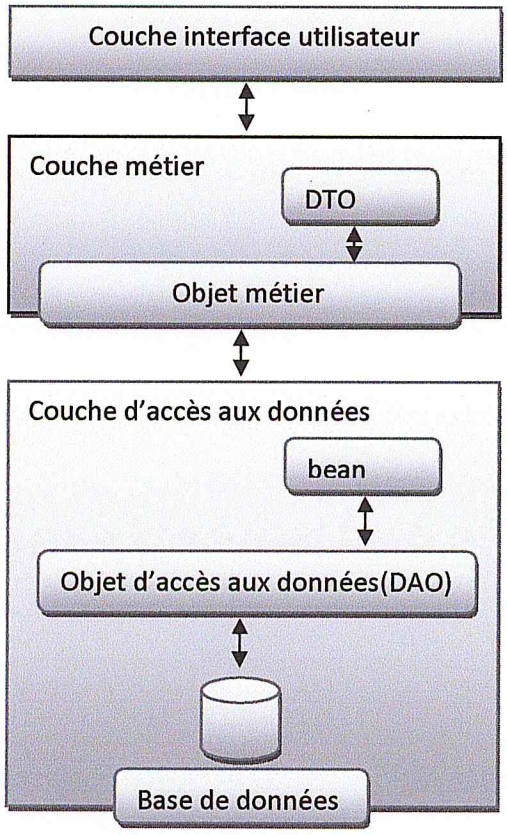


Figure 63 : Architecture de l'application

Couche d'accès aux données :

Cette couche s'occupe de l'accès à la base de données .les objets manipulés à ce niveau sont :

a. **Bean :** Les objets Bean sont les objets persistants de l'application (communicant avec la base de données).Ces objets permettent de récupérer les informations dans la base de données et mettre a jour la base données.

b. **Objets d'accès aux données (DAO) :**

Les objets d'accès aux données s'occupent de persister les objets Bean, ils réalisent toutes les fonctionnalités concernant l'axé aux données i.e. les opérations CRUD (create, retrieve, update, delete), la connexion a la base et l'exécution des requêtes.

Couche métier :

S'occupe du logique métier de l'application, cette couche est indépendante de toutes formes d'interface avec l'utilisateur, elle doit être utilisable aussi bien avec une interface console ou une interface web .c'est la couche la plus stable de l'architecture, elle ne change pas si on change l'interface utilisateur ou la façon d'accédé aux données nécessaires au fonctionnement de système.

a. **objet métier :** ils implémentent la logique métier, ces objets utilisent la couche d'accès aux données pour stocker et récupérer les informations de la base.

b. **DTO :** les objets DTO (data Transfer Object) ont pour but de transporter les informations métier complexes (combinaison de plusieurs Bean).la communication entre les Bean et les DTO peut se s'effectue dans les deux sens :

- **Bean->DTO :** dans le cas d'une opération de récupération des données .dans ceci ,le DTO sera alimenté a partir d'un ou plusieurs Bean .en effet un Bean a une structure qui correspond a une table ,et les DTO a une forme qui rassemble a un ensemble de table ,cette phase d'assemblage est assurée par la couche métier .
- **DTO->Bean :** dans le cas ou la base de données doit être mise a jour par les données provenant du client .la couche métier transforme le DTO en Bean ou en plusieurs Bean (dans le cas ou plusieurs table doivent êtres mises a jour) et fait appel a la couche d'accès aux données.

Couche interface utilisateur :

C'est l'interface (graphique souvent) qui permet à l'utilisateur de piloter l'application et d'en recevoir des informations. Cette couche contient tout sort d'interface avec l'utilisateur (pages JSP, HTML et JavaScript).

b) Architecture de Service Web :

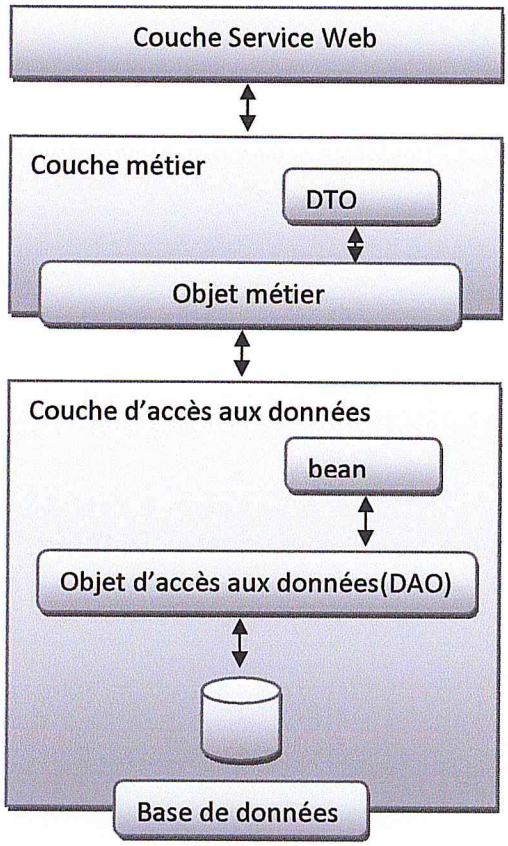


Figure 64 : Architecture de service

Couche d'accès aux données :

Elle est bien détaillée dans la section précédente et possède les mêmes fonctionnalités de la couche d'accées aux données de l'application E-Formation

Couche métier :

Joue le même rôle de la couche métier de l'application d'E-Formation .

Couche service Web :

C'est la couche la plus importante de toute l'architecture .Celle-ci constitue l'interface entre l'utilisateur et l'exécution des demandes. Elle est composée des classes qui implémentent l'interface du service, donc elle fournit les opérations nécessaires qui assurent la communication entre les différentes entités du système.

c) Architecture de client de service :

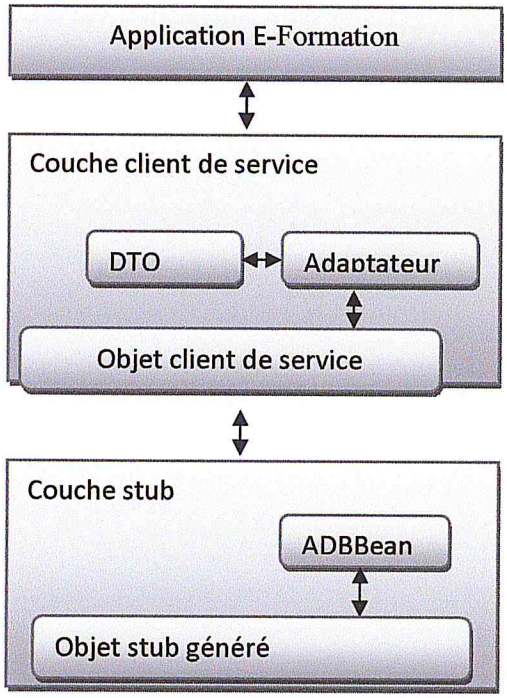


Figure 65 : Architecture de client de service

Couche Stub :

Cette couche s'occupe de gérer l'accès aux ressources réseaux .Elle réalise la sérialisation en XML, et l'envoi, la réception et la construction des messages SOAP.

a. **Objet Stub :** les objets stub sont générés par le plugin **Apache Axis2 Code Generator Wizard** (présentés dans la section précédente), à partir des fichiers WSDL. ces objets implémentent des interfaces spécifiques a Axis 2 pour générer proprement l'envoie, la réception et la construction des messages SOAP.

b. **ADBBean** : ils représentent les types d'objets échangés entre le fournisseur et le consommateur du service .Ces classes implémentent l'interface ADBBean(fournit par Axis 2) dans le but de sérialiser (dé-sérialiser)les données a envoyer (recevoir).

Couche client de service : elle constitue l'intermédiaire entre le consommateur de service(E-Formation) et la couche Stub, cette couche joue un rôle très important, elle réalise les adaptations nécessaire et gère l'appel au service web en utilisant les services de la couche Stub .

L'intérêt de la centralisation des opérations d'invocation du service est de rendre transparent a l'application de E-Formation les procédures d'appel du service.

a. **Objets client de service** : ils ont pour but de gérer l'invocation du service, ces objets offrent les mêmes opérations que le service lui-même. Ces objets collaborent avec les classes **Adaptateurs** pour gérer la conversion de :

   - ADBBean->DTO : dans le cas ou le service Web envoie une réponse, a sa réception les objets client du service récupèrent les ADBBean fournis par la couche Stub, puis les transforme en DTO par les adaptateurs, et enfin le résultat obtenu est transféré a l'application de E-Formation

b. **Adaptateurs :** ils réalisent la conversion d'ADBBean--> DTO (cette conversion est expliquée dans la section précédente).

La figure suivante montre les étapes d'exécution de la procédure d'invocation d'un service web.

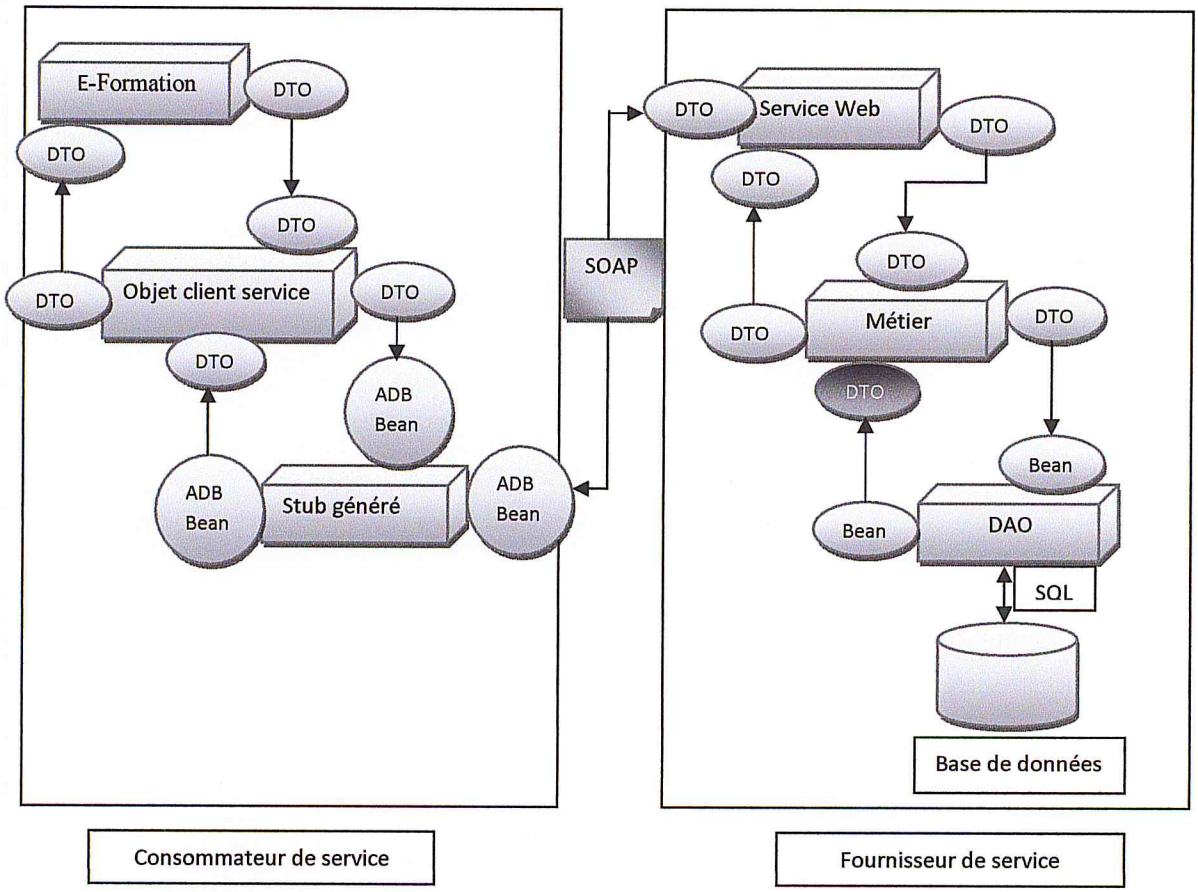


Figure 66 : Interaction entre le fournisseur et le consommateur du service

5.1.1 Diagramme de déploiement :

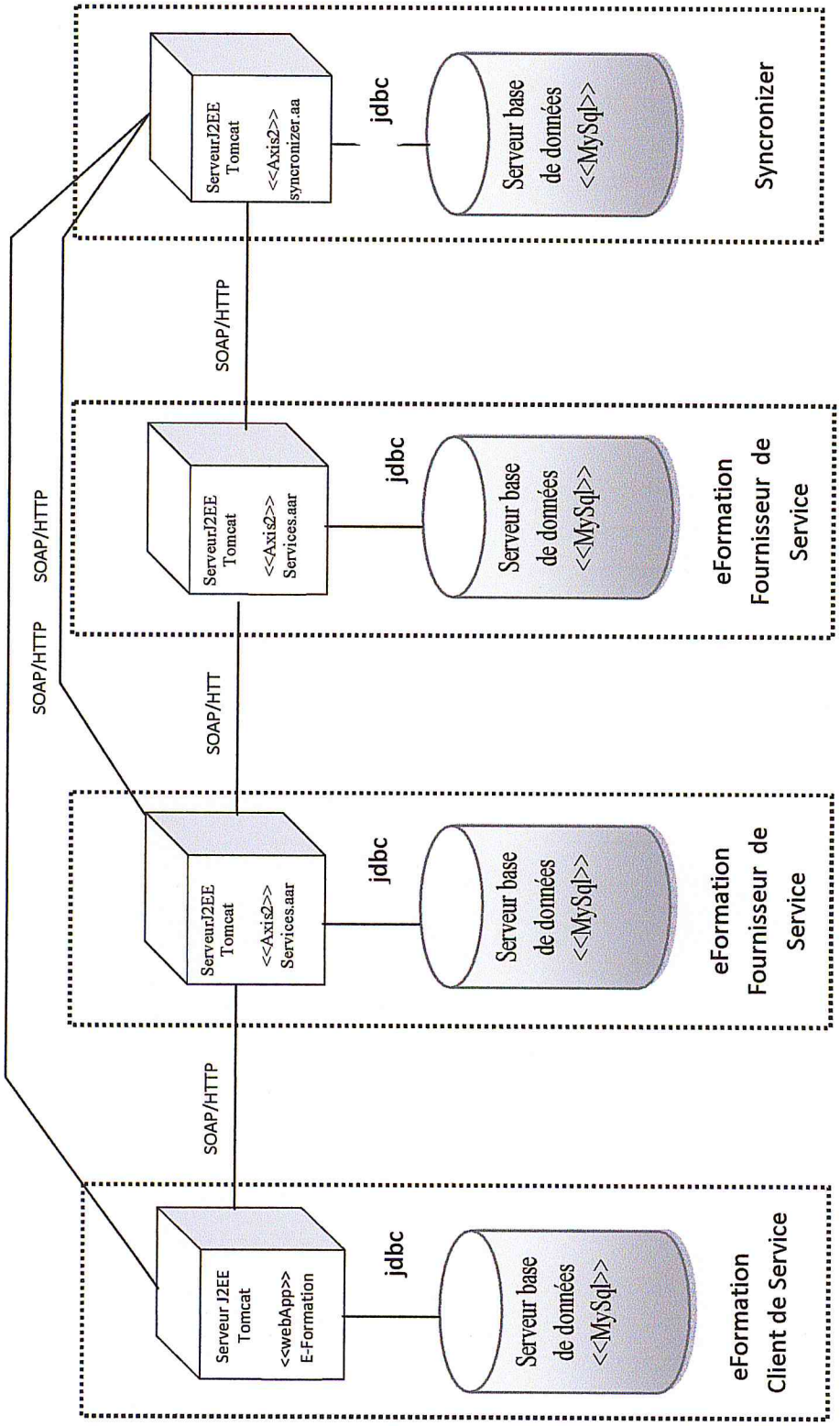


Figure 67 : Diagramme de déploiement de système de communication

5.4 Présentation de l'application:

Toute conception doit être concrétisée par la réalisation et la mise en place d'une application, qu'on considère comme l'aboutissement des étapes précédentes de développement du logiciel. Dans ce qui suit, on présente la structure de l'interface de notre application(les principales fenêtres)

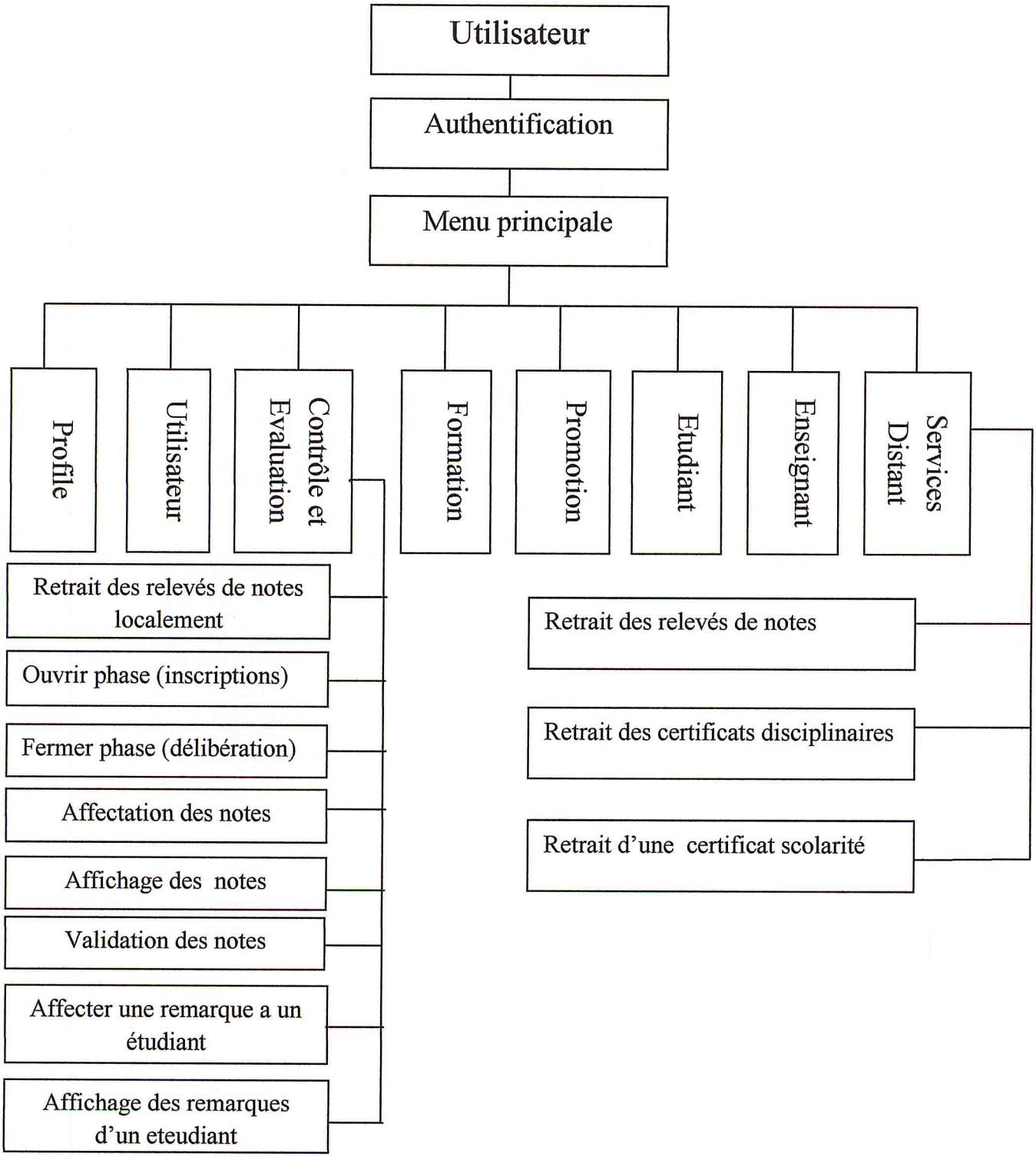


Figure 68 : Structure de l'interface de notre application

5.4.1 Interfaces et fonctionnalités du système :

Dans cette partie, nous allons présenter quelques interfaces de l'application, en suivant le processus global de la planification des emplois temps dans l'étude conceptuelle. Ces interfaces concernent entre autres :

Page d'accueil (authentification) :

Figure 69–Page d'accueille-

Fenêtre affichage des notes :

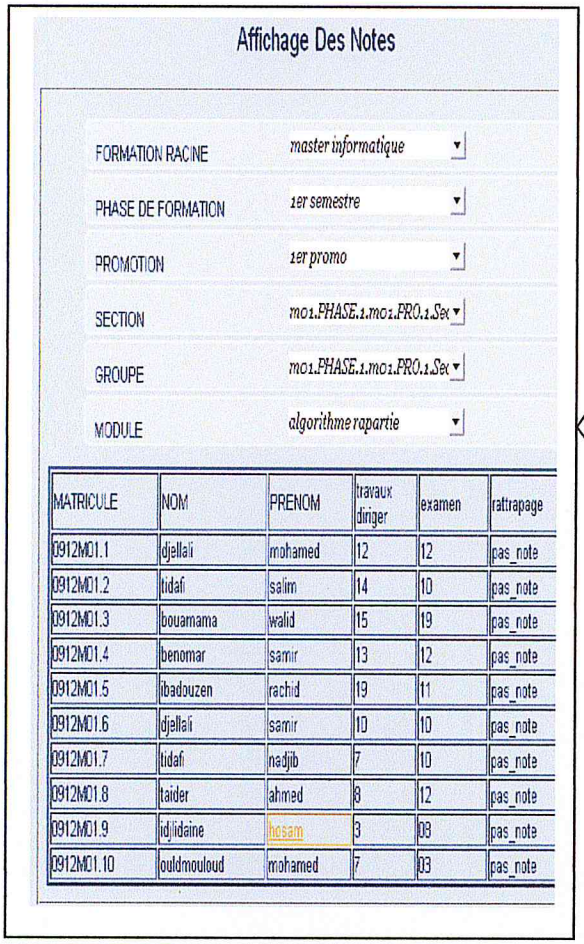


Figure 70-affichage des notes selon le profil

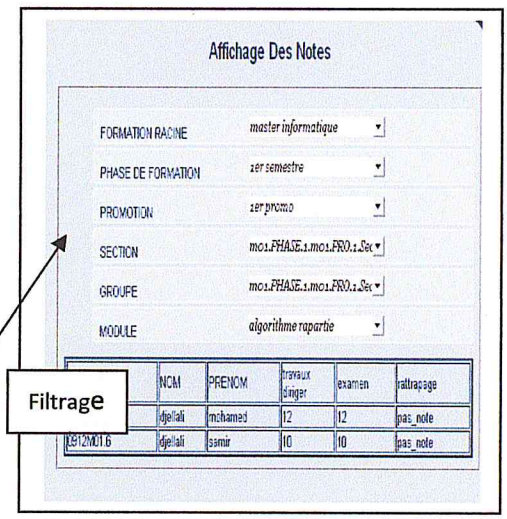


Figure 71- affichage des notes selon profil tuteur-

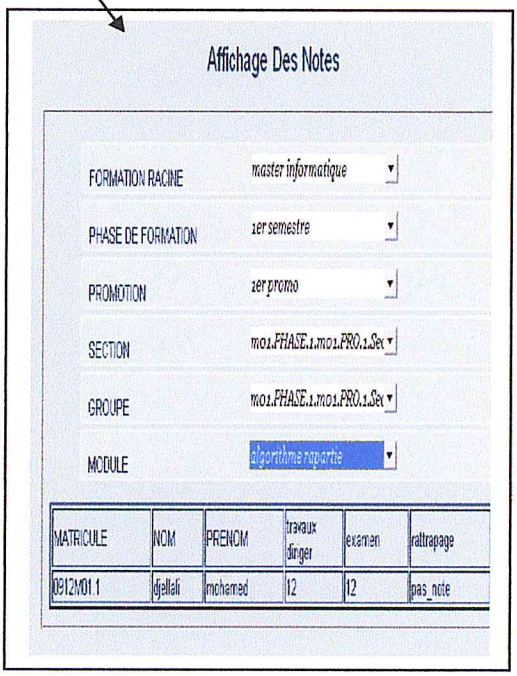


Figure72-affichage des notes selon profil étudiant-

Les figures au-dessus montrent le filtrage des notes selon les droits apportés sur les utilisateurs, le filtrage de ces notes est effectué selon des profils utilisateur.

Pour le filtrage des notes par exemple:

- Un étudiant a le droit de voir seulement ces notes

- Un parent d'élève (tuteur) peut voir les notes de ces enfants
- L'enseignant peut voir, modifier et valider les notes de ces étudiants

Page validation des notes :

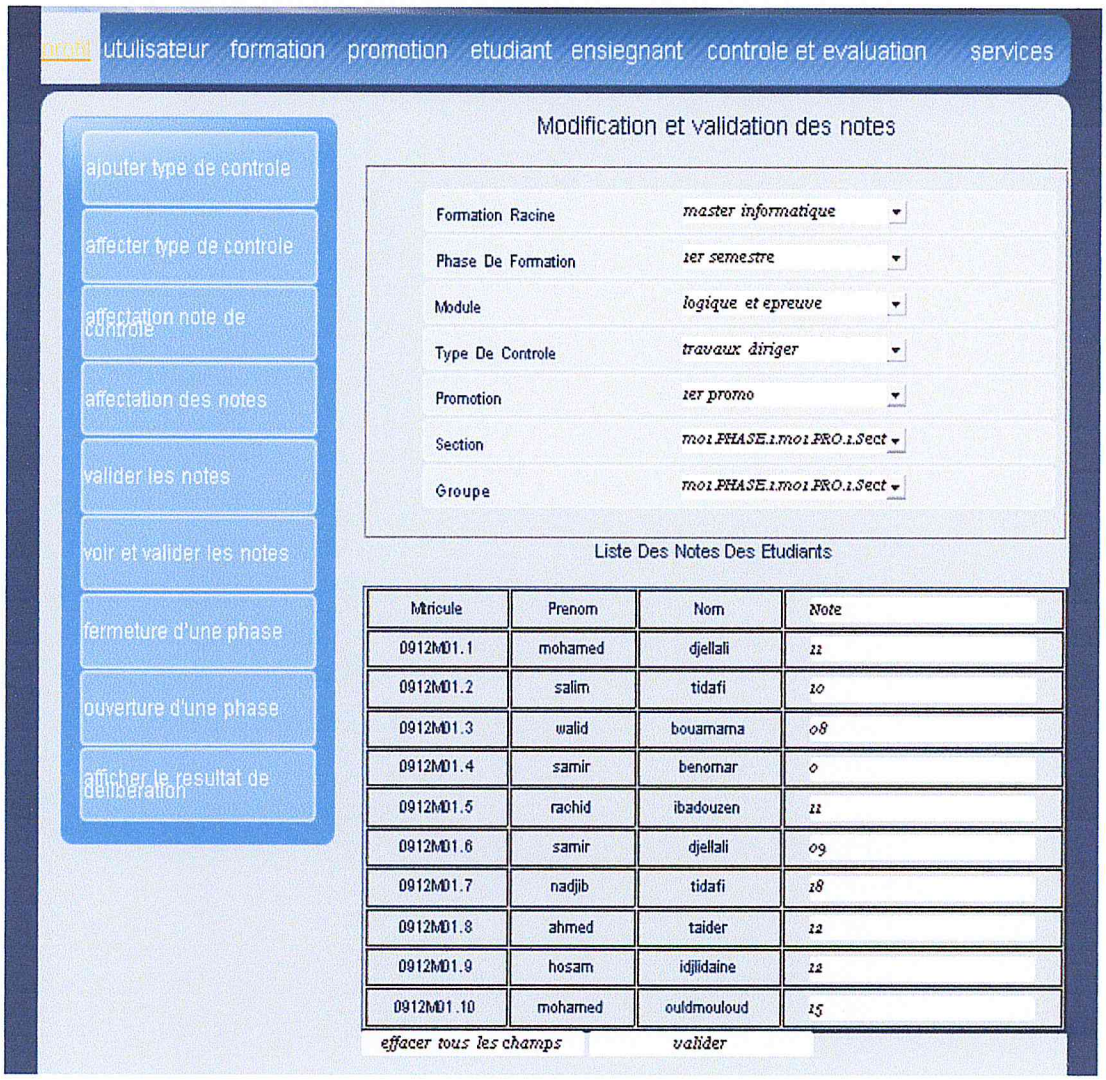


Figure 73- Page validation des notes -

Cette page est autorisé seulement pour quelques profiles tel que : agent scolarité, Enseignant, Responsable de formation et le chef d'institution.

Apres vérification et modification des notes par le validateur, il clic sur le bouton valider : quelques messages vont s'afficher tel que:

- **« Modification avec succès»** : le cas de l'agent de scolarité ou il peut seulement modifier sans valider
- **«Validation est effectuer avec succès»** : dans le cas de l'un des trois validateurs

- **«validation est échouée, attendez la validation de validateur d'ordre inferieur».**
- **« validation est échouée, attendez la validation de validateur d'ordre supérieur».**

Page fermer phase :

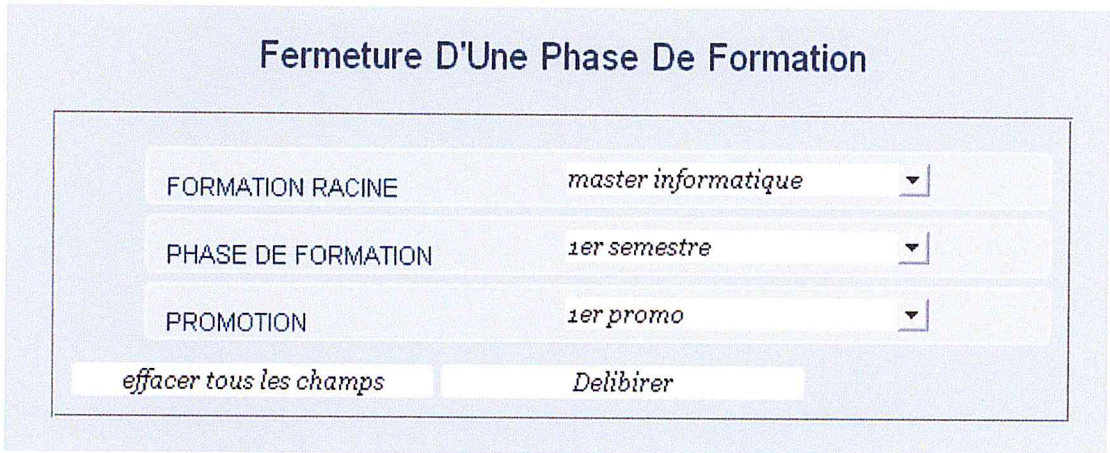


Figure74 -fenêtre fermeture de la phase de formation-

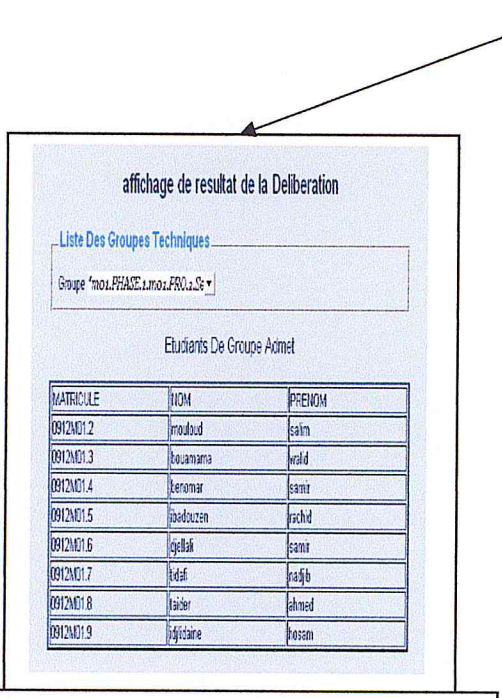


Figure 75-affichage groupe admet-

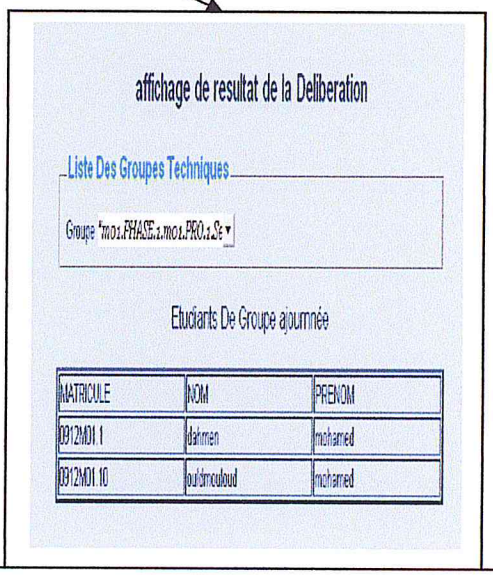


Figure 76-affichage groupe ajourné-

Cette page permet de fermer une phase de formation pour une promotion, cette fermeture consiste à chargé les apprenants dans les groupes techniques (admet, ajournée) suivant le processus suivant :

- Vérifier les notes de la phase s'elles sont validées

Sinon un message sera affiché : « il faut valider tout le notes avant de fermer la phase»

- Son but est de charger l'apprenant dans groupes : admit, ajourné, résultat et dette.
  - Si Moyenne phase>=10 : affecter apprenant au **groupe admis**
    Sinon affecter l'étudiant au **groupe ajourné**
  - Si Moyenne année>=10 : affecter l'étudiant au **groupe résultat**
  - Sinon Si Crédit de l'année>=creditMin : affecter l'étudiant au **groupe dette**

Page ouvrir phase(passage automatique d'une phase a une autre) :

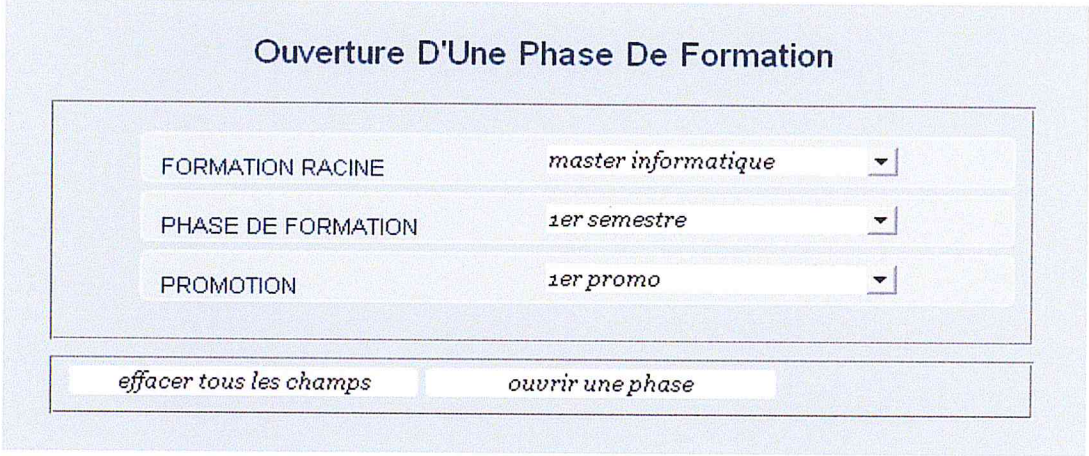


Figure77-ouverture d'une phase de formation pour une promotion-

Utilisée les groupes admis, ajourné, résultat et dette Pour ouvrir la nouvelle phase, son but est le passage automatique de la promotion a la nouvelle phase en chargeant les apprenants ajournées de la promotion précédente dans la nouvelle phase et les apprenants admets de la promotion actuelle de la phase précédente dans le groupe Inscription de la nouvelle phase.

Page retrait des relevés de notes distants :

Une étude de cas :

Un étudiant étudier son 1 er année Licence LMD au département informatique de l'université d'Oran, en 2eme année il a changé l'université vers département informatique de l'université de Bab Zouar, pour sa 3eme année il a fait une demande de transfère au département informatique de l'université de Blida.

Avant d'accepter la demande, la comité pédagogique fait une étude de son dossier tel que les notes obtenus dans ces années précédente:

La figure suivante montre la fenêtre de retrait des relevés de notes de l'étudiant, exécutée au sien de département informatique de l'université de Blida.

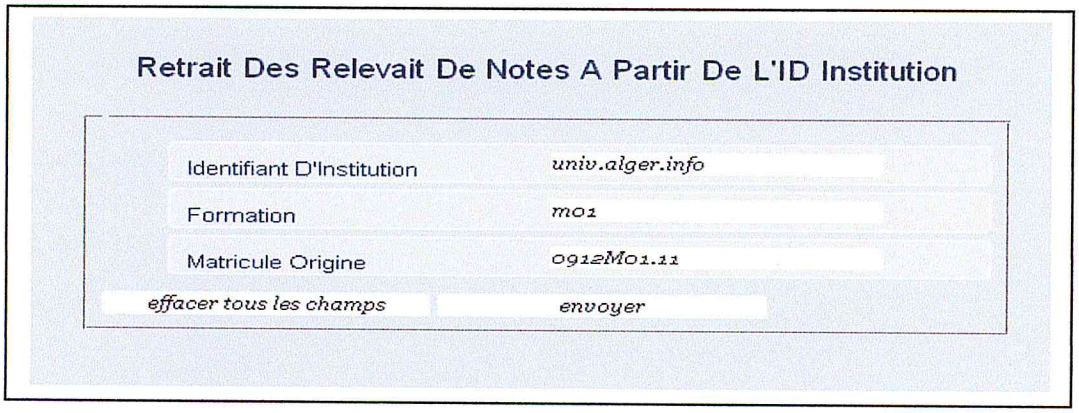


Figure 78 –formulaire retrait relevé de notes-

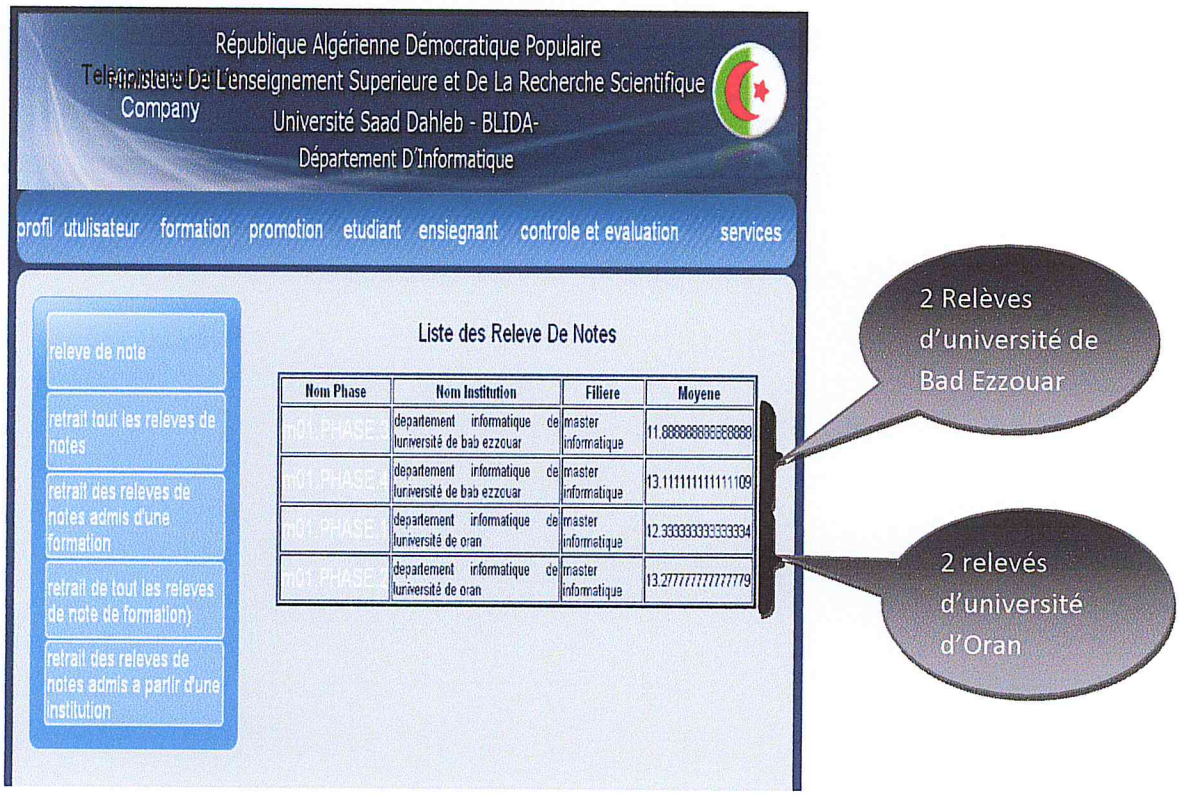
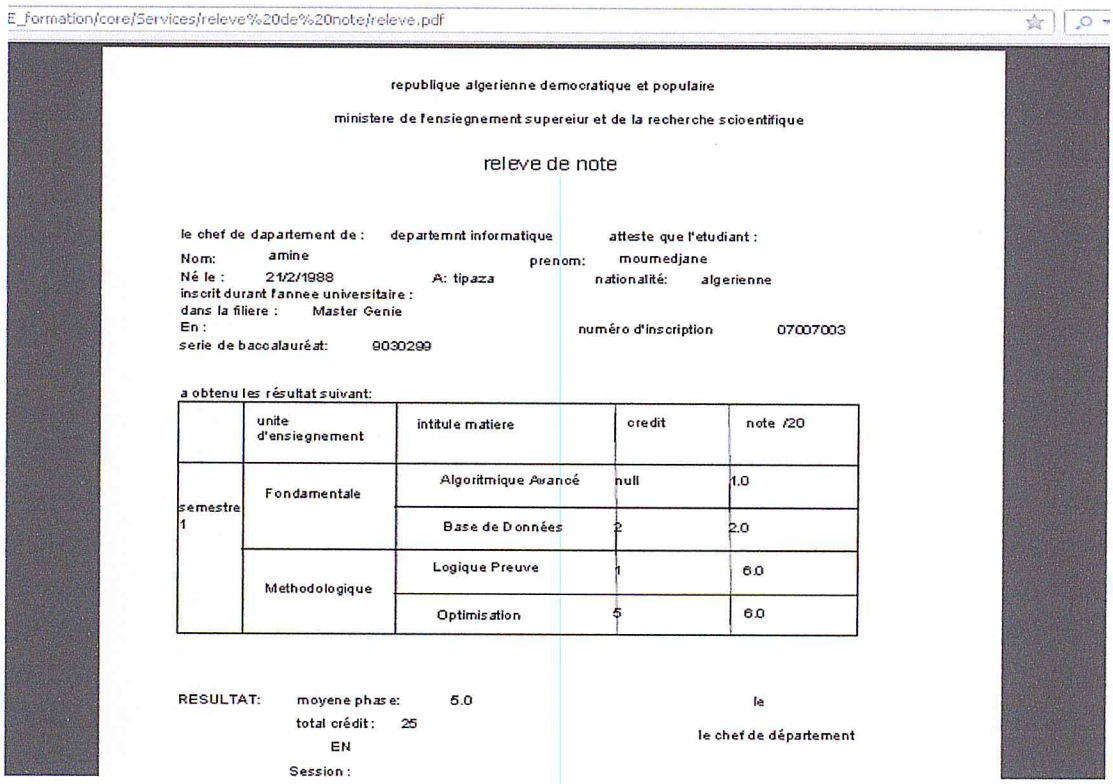


Figure 79-affichage de liste des relevés de notes distante-

Fenêtre relève de note :


E_Formation/core/Services/releve%20de%20note/releve.pdf

republique algerienne democratique et populaire

ministere de l'ensiegnement supereiur et de la recherche scioentifique

releve de note

le chef de dapartement de : departemnt informatique atteste que l'etudiant :

Nom: amine prenom: mournedjane

Né le : 21/2/1988 A: tipaza nationalité: algerienne

inscrit durant l'annee universitaire :

dans la filiere : Master Genie

En :

serie de baccalauréat: 9030299 numéro d'inscription 07007003

a obtenu les résultat suivant:

	unite d'ensiegnement	intitule matiere	credit	note /20
semestre 1	Fondamentale	Algoritmique Avancé	null	1.0
		Base de Données	2	2.0
	Methodologique	Logique Preuve	1	6.0
		Optimisation	5	6.0

RESULTAT: moyene phase: 5.0 le

total crédit: 25 le chef de département

EN

Session :

Figure 80 -document relève de note -

Conclusion :

Le travail que nous avons effectué consiste à concevoir et réaliser un système coopératif pour la gestion des formations.

Nous avons développé une application web pour la gestion administratif des formations notamment : L'inscription des étudiants, la saisie, la validation des notes, et le passage automatique d'une phase à une autre.
Cette application à une particularité d'être prestataire de services, en effet elle rend les résultats exploitables par d'autres applications rapatriées sur un réseau. Elle est toutefois consommateur de service étant donné qu'elle fait appel aux différents services externes tels que les relèves de notes, les certificats de scolarité et les certificats disciplinaires.

Pour sa réalisation nous avons suivi 3 étapes : analyse, conception et implémentation.

Dans l'analyse nous avons étudié les techniques de communication (RMI, services web..) Ainsi que les systèmes développés dans le domaine de gestion des formations afin de collecter les informations essentielles à la production de la valeur ajoutée.

Pour la partie conception nous avons présenté l'étude conceptuelle de notre système en 2 parties : conception de système de gestion des formations avec UML et celui de système de communication avec SoaML en présentant les diagrammes nécessaires a l'étude.

Et enfin dans l'implémentation nous avons présenté les technologies utilisées pour le développement de notre système: java, et MySQL pour la programmation des fonctionnalités de portail, et Axis 2 pour le système de communication.

Ce projet nous a permis :

- D'améliorer profondément nos connaissances en programmation orientée objet en java et de pratiquer les technologies des services web sous Axis2
- D'approfondir nos connaissances en conception avec l'utilisation d'un nouveau langage de modélisation SoaML
- De découvrir les architectures orientée services.

Perspectives :

Afin de compléter ce travail on peut envisager quelques travaux futurs :

- Intégration des nouveaux modules par exemple : un composant de sécurité afin de protéger les échanges d'informations entre les instances de système et le Synchroniser.
- Mise en œuvre de la signature numérique des documents manipulés.
- Ajouter la version en Arabe de système pour prendre en considération les institutions qui travaillent avec celle-ci.

- Réalisation d'une interface graphique pour la configuration de Synchroniser et les services web.
- Mise en place de nouveaux services Web comme le service de recherche des formations dont laquelle l'apprenant est inscrit

[1] Sylvain-Rampacek «Sémantique, interactions et langages de description des services web complexes » thèse de doctorat, novembre 2006

[2] Riadh Ben Halima « implantation et expérimentation d'une architecture en bus pour l'auto réparation des applications distribuées à base de services web », thèse doctorat 2009.

[3] Michael Champion, Chris Ferris, Eric Newcomer, Iona, and David Orchard «Web
Services Architecture », 14 Novembre 2002.

[4] Francisco Curbera, Yaron Goland, Johannes Klein, Frank Leymann, Dieter Roller, Satish Thatte and Sanjiva Weerawarana «Business Process Execution Language for Web Services», 2002.

[5] Mary Kirtland, a Platform for Web Services, Microsoft Developer Network, January 2001.

[6] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen « SOAP Version 1.2 Part 2 – Adjuncts », 19 Décembre 2002.

[7] IBM Tom Bellwood, Microsoft Luc Clément, IBM David Ehnebuske, IBM Andrew Hately, IBM Maryann Hondo, HP Yin Leng Husband, Microsoft Karsten Januszewski, Oracle Sam Lee, IBM Barbara McKee, Intel Joel Munter, and SAP Claus von Riegen « UDDI Version 3 » 19 Juillet 2002.

[8] July«UDDI.ORG. Universal, Description, Discovery and Integration (UDDI) Version 3.0.1 » 2003.

[9] Denivaldo Cicero Pavão LOPES, « Étude et applications de l'approche MDA pour des plates-formes de Services Web » 11 juillet 2005.

[10] Gilbert Babin, Michel Leblanc, « les web services et leur impact sur le commerce b2b », Août 2003

[11] Arnaud VEZAIN « Les services web », présentation générale. Technical report, Association HERMES, Février 2005.

[12] Eric Malville, « L'auto-organisation de groupes pour l'allocation de taches dans les systèmes multi-Agents : Application a CORBA » thèse de doctorat, Mars 1999.

[13] Elarbi Badidi, « Architecture et services pour la distribution de charge dans les systèmes distribués objet » thèse doctorat, Mai 2000.

[14] Annick Fron, « Architecture répartie en JAVA : RMI, CORBA, JMS, sockets, SOAP, Service web », édition DUNOD, 2007.

[15] N. A. B. Gray, « Comparison of web service, Java-RMI, and CORBA service implementation » Fith Australasion Workshop on Software and system Architecture , 13 Avril 2004, Malbourne, Australia.

[16] M. B. Juric, I. Rozman, M. Hericko, « Performance Comparison of CORBA and RMI » informationand Software Technology, 42, pp 915-933, 2000.

[17] DEBRAUWER L., KARAM N.; UML 2: Entraînez-vous à la modélisation ; France 2006.

[18] Pascal Roques et Franck Vallée, « UML en action de l'analyse des besoins à la conception en Java », 2002.

[19] DAHDAH Ahmed, HAMANI Mohamed, « Conception et réalisation d'un système d'information pour le suivi financier des opérations d'investissements », juin 2007.

[20] http : //www.omg.org/spec/Soaml/

[21] http://modeliosoft.com/

[22] Arne J. Barre, Dima Panfilenko, « Service Modeling with SoaML », sixth European Conference on Modelling Foundations and Applications ,15-18 June 2010, Paris,France.

[23] Yekhlef Hadjer, «Conception et réalisation d'un système de communication pour les logiciels de e-Gouvernement », juillet 2011.

[24] http://developpez.net

[25] Advanced computing technologie ACT Oran
« Suivi des Etudiants et des Enseignements Supérieurs »,2006

[26] FELLAH Abdeldjalil, HEBBACHE Khaled « Conception et réalisation d'un site web dynamique pour le suivi des activités pédagogiques et scientifiques de la DPGR de l'ESI» 2009/2010

[27] http://www.scolarex.com, 06/2012

Annexe

Le Langage SoaML

1. Présentation de langage SoaML :

SoaML est une nouvelle spécification présentée par l'organisation OMG (Object Management Group) en 2009 .le but de SoaMl est de fournir aux utilisateur de langage UML (Unified Modiling Language) les moyennes de modéliser une architecture orientés service composant des notion de consommateur et fournisseur de services, ainsi que la notion de contrat. SoaML se base sur un ensemble de concept que nous présentons brièvement dans ce qui suit[23].

1.1 Participants :

Ils sont des entités qui fournissent ou utilisent des services. Ces entités peuvent représenter des personnes, organisations ou des composants d'un système d'information.ils ont des ports de services qui sont les points de connexion ou des services sont effectivement fournis ou consommés[23].

1.2 Ports :

Les participants fournissent et consomment des services via des ports. Un port représente le point d'interaction ou le service est consommé ou fournit. Il est associés a une interface requise (marqué par le stéréotype « request ») ou offerte (désigner par le stériotype « service »)[23].

1.3 Contrat de service :

Le contrat de service représente un accord entre les participants consterné pour savoir comment le service doit être fournir et consommé. Il a pour but de définir les rôles joués par chaque participant dans le service (fournisseur et consommateur) et les interfaces qu'ils mettent en œuvre pour jouer ce rôle[23]

2. Diagrammes du langage SoaML :

Afin de modéliser une application orienté service, SoaML fournit plusieurs diagrammes qui sont :

- Diagramme d'architecture des services (service Architecture)
- Diagramme de processus métier (Bisnesse Process)
- Diagramme des Capabilités (Capabilities)
- Diagramme de contrat des services (Service Contrat)
- Diagramme d'interface de service (Service interface)
- Diagramme de message (Message diagram)

- Diagramme des chorégraphies (service choreographies)
- Diagramme des participants(ou des composant)

2.1 Diagramme de processus métier :

Ce diagramme décrit les processus pertinents du domaine dont l'architecture oriente service fait partie.il a pour objectif de proposer un moyenne simple et visuel de communication entre les différent intervenants chargés de la réalisation de projet. La spécification SoaMl propose d'utiliser la notation BPMN(business Process Modiling Notation) pour réaliser ce type de diagramme[23].

2.2 Diagramme de contrat :

Ce diagramme est dédié a la modélisation des contrat qui ont été mentionnés dans l'architecture des services, en définissant pour chaque contrat le fournisseur et le consommateur d'un tel service[23].

2.3 Diagramme d'architecture des services :

Ce diagramme est dédier a la modalisation d'une architecture orienté service, en spécifiant les différente participant et les contrats de services qui le relient.

Etapes de modélisation :

Pour construire une architecture de service avec SoaML, il faut suivre les étapes suivantes :

- Identifier les participants
- Identifier les contrats de service.
- Lier les participants aux contrats de service.

2.4 Diagramme d'interface des services :

Ce diagramme défini les interfaces et les responsabilités d'un participant afin de fournir ou consommer un service.il représente le résultat de raffinement de diagramme contrat de service [23].

Etapes de modélisation :

Afin de modéliser une interface des services avec SoaML, il faut suivre les étapes suivantes :

- Définir l'interface de service à partir des contrats.
- Définir du fournisseur avec ses opérations

- Définir l'interface du consommateur (L'interface du consommateur est définie lorsque des appels **CallBacks** sont nécessaire) ;
- Relier l'interface du fournisseur et du consommateur à l'interface de service comme suit :
  - Créer un lien de réalisation entre l'interface du fournisseur et l'interface de service.
  - Créer un lien d'utilisation entre l'interface du consommateur et l'interface de service
- Créer et relier l'interface conjugué du service comme suit :
  - Créer l'interface conjuguée du service qui utilise l'interface du fournisseur et réalise

L'interface du consommateur. A noter que le nom d l'interface conjuguée du service est précédée par « » qui représente le contraire de l'interface de service.

1.2.5 Diagramme des participants

Ce digramme permet de :

- Spécifier les composants qui réalisent les services définies par le diagramme d'architecture des services.
- Offrir une vision des composants du système.
- Décrire les relations qui existent entre les composants

Etape de modification :

Pour concevoir un diagramme des participants du service ,il est conseillé de suivre les étapes suivants

- Identification des composants de l'architecture des services par la décomposition en plusieurs composants plus spécifiques.
- Créer les ports des composants
  - Un port stéréotypé «service » a pour type l'interface de service.
  - Un port stéréotypé «request» a pour type l'interface de conjugé.
- Lier les ports par leurs interfaces fournies et requiers

1.2.6 Diagramme des messages :

Le diagramme des messages(ou type de messages)spécifie les informations échangeés entre le fournisseur et le consommateur du service .Les type de messages représentent des données pures qui peuvent être les différences parties[23]

1.3 La méthode SoaML

Cette méthode a été développée avec le soutien de l'Union européenne FP7 (Le Septième programme cadre est programme cadre actuel(2007-2013)de l'Union européenne dans la recherche et le développement technologique, il est géré par une commission européenne) dans le cadre du projet **SHAPE** .Son objectif est de fournir les étapes préliminaires du développement d'une architecture orienté service en utilisant le langage SoaML[23]. Cette méthode exige la construction de trois modèles[23]

- Modèle d'architecture métier **BAM (Business Architecture Modeling).**
- Modèle d'architecture Système **SAM(Système Architecture Modèling).**
- Modèle de la plateforme spécifique **PSM(Platform Spécific Model).**

Le deuxième modèle est le résultat de l'op ération de raffinement du premier ,à ce stade on est plus proche de l'implémentation(dernier modèle)

1.3.1 Modèle d'architecture métier « BAM »

Le BAM décrit les opérations métiers ainsi l'environnement ou l'architecture orientée service sera installé [23].Le BAM capture les besoins de l'entreprise et identifier les services par la définition de :

- Les objectifs métiers ;
- La sémantique de l'information(technologies,ontologies).
- Les processus métier.

Ce modèle décrit :

- L'architecture des services.
- Les contrats des services.

Afin de construire ce modèle ,il est nécessaire d'établir les diagrammes suivants :

- Diagramme d'architecture des services.
- Diagramme des processus métier

- Diagramme des contrats des services.
- Diagramme des Capabilités(optionnel)

1.3.2 Modèle d'architecture système(SAM) :

Ce modèle décrit l'architecture globale du système, A ce stade deux aspects sont définis[23] :

Aspect structurelle :décrit les composants ,leurs relations de dépendance(Dependency)

Et leur interfaces fournies/requise .Afin de construire ce modèle il faut établir les diagrammes suivants :

- Diagramme d'interface des services.
- Diagramme des messages.
- Diagramme de participants.

Aspect dynamique : décrit les interactions entre les composants .Le diagramme des chorégraphies du service permet de schématiser cet aspects.

1.3.3Modèle de plateforme spécifique PSM

Le modèle de plateforme spécifique(PSM) contient la conception et la mise en œuvre de l'architecture orienté service spécifiés dans la plateforme de la technologie choisie , par exemple : service Web , Java Entreprise Edition(JEE), les systèmes multi-agents(MAS),

Peer-2(peer(P2p)..ect