TABLE DES MATIERES

Introduction générale

Introduction générale

Un service web est le fait de mettre des ressources à disposition (gratuite ou non) sur Internet, via un protocole d'échanges standardisé, pour des programmes écrits dans des langages quelconques.

Dans les dernières années, les services web occupent un espace très important dans l'informatique parce qu'ils représentent une solution largement répondue à plusieurs problèmes posés.

Notre application est une simple application web service. Le serveur est implémenté en utilisant Tomcat comme serveur web et la bibliothèque Axis2 comme moteur de web service, le serveur fournit un service de conversion de devises. Le côté client et une simple application Java qui comporte deux types de clients : un client utilisateur, à travers lequel les utilisateurs de l'application bénéficient du service de conversion offert par le serveur et un client administrateur, utilisé par l'administrateur de l'application pour effectuer les différentes tâches d'administration tel que la mise à jour du taux de change.

Le document est organisé comme suit :

Le chapitre 1 : décrit quelques concepts des différents middlewares existants.

Le chapitre 2 : décrit les notions de base relatives aux services web et ses différents protocoles.

Le chapitre 3 : présente les détails d'implémentation de notre application.

Et en fin une conclusion générale qui résume notre travail.

Chapitre I :
Les middlewares

I. Introduction :

Ce chapitre contient quelques notions sur les middlewares, il commence par une petite historique décrivant l'évolution du middleware, vient ensuite une motivation pour cette dernière avec une description de ses principales fonctions.

I.1. Historique :

Au début de l'informatique, le dialogue entre machines nécessite une connaissance approfondie des protocoles réseau et parfois même du matériel réseau.

La programmation orientée objet a permis le développement des architectures distribuées en fournissant des bibliothèques de haut-niveau pour faire dialoguer des objets répartis sur des machines différentes entre eux, ce qui a considérablement allégé le travail des programmeurs. Les objets distribués sur le réseau communiquent par messages en s'appuyant sur l'une des technologies suivantes :

- Common Object Request Broker Architecture ou CORBA : ce standard de l'Object Management Group permet de faire communiquer des objets écrits dans des langages différents (C++, Java, Smalltalk) et même d'encapsuler des programmes écrits dans des langages procéduraux pour les faire passer pour des objets.
- Remote Method Invocation ou RMI : cette technologie de Sun permet de faire communiquer très simplement des objets java distribués sur le réseau.
- Les services web XML.
- .NET Remoting : vous permet de créer facilement des applications largement distribuées, si les composants de l'application sont tous sur un même ordinateur ou répartis à travers le monde entier.
- Windows Communication Foundation (WCF) : est une technologie de développement d'applications basées sur une architecture orientée services (SOA). [1]

Un autre middleware appelé service web, est apparu au début des années 2000. Il permet l'interconnexion des applications à l'aide des protocoles d'internet.

Ce middleware a été favorisé par le développement et la démocratisation du haut débit, l'expansion du langage de structuration XML et le manque d'interopérabilité. Nous notons que certaines entreprises publiaient déjà de l'information via des sites web, utilisaient la messagerie

et faisaient du commerce électronique, mais avec l'avènement des services web l'échange inter et intra entreprise sera plus organisé et plus automatisé.

La première société à avoir introduit un concept de services distribués, proche de ce qui existe aujourd'hui, est **Hewlett-Packard** avec son produit **e-Speak**, et ce dès 1999. La suite fut une grande course entre les principaux acteurs du marché qui ont progressivement, par soucis d'interopérabilité, adopté les principaux standards des services web que sont **SOAP**, **WSDL** et **UDDI**. On notera que e-Speak possède un modèle de sécurité très solide, reposant sur le standard SPKI (Simple Public Key Infrastructure) de l'IETF. [2]

II. ÉVOLUTION des middlewares :

Dans cette partie, on va donner les différents types de middleware et leurs avantages et inconvénients.

- Remote Procedure Call
- Message Oriented Middleware
- Objets Distribués
- Database Oriented Middleware.

Dans la suite de ce paragraphe, on va étudier en détail les différentes caractéristiques de ces Middlewares, leurs avantages, leurs inconvénients, et on va arriver à trouver la différence entre l'utilisation des ces anciens Middlewares et l'utilisation des nouvelles technologies (à savoir les services web).

II.1. Motivations : évolution des middlewares - RPC1 :

RPC (*Remote Procedure Call*) est un protocole réseau permettant de faire des appels de procédures sur un ordinateur distant selon le modèle client serveur. [3]

Principe :

Le principe de RPC est :

- On différencie le coté appelant (client) du coté appelé (serveur).
- Appelé offre la possibilité à des éléments distants d'appeler une ou plusieurs fonctions chez lui.
- Le coté client appelle localement la fonction sur un élément spéciale qui relayera la demande d'appel de fonction coté serveur.

[1] Remote Procedure Call

- Coté serveur, un élément spécial appellera la fonction et renverra le résultat coté client.
- Eléments spéciaux : talons (ou *stubs*). [4]

Le graphe suivant permet d'expliquer les étapes d'un appel de procédure distante par messages : [5]

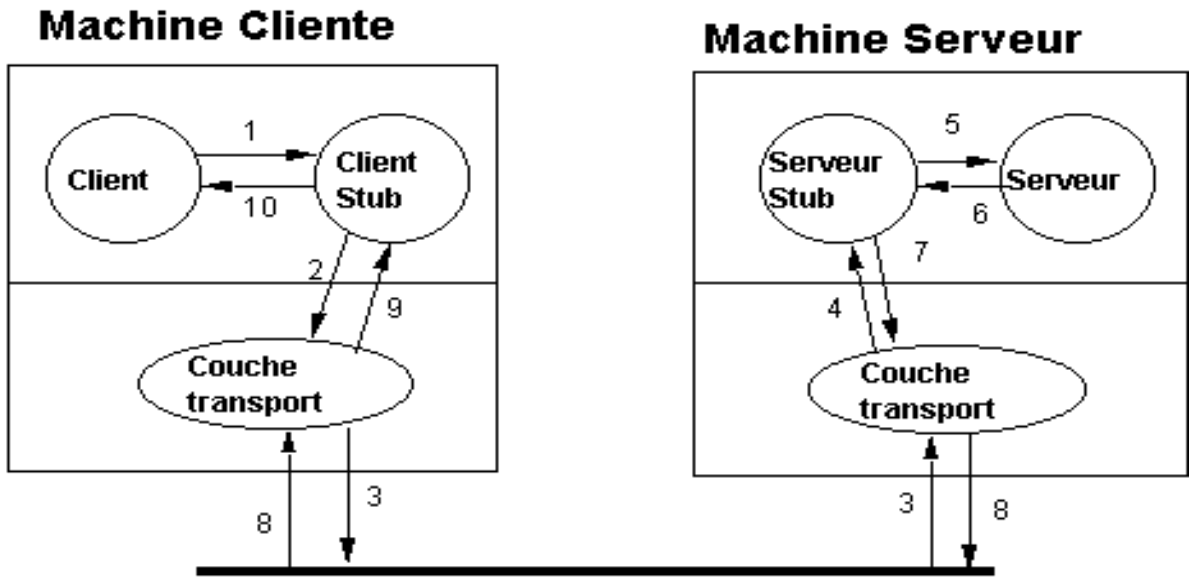

Figure I.1: RPC (Remote Procedure Call)

II.1.1 Les avantages :

Parmi les principaux avantages de RPC, il y a :

- Efficace en cas de nombreux appels**.**
- Efficace si tout le code et les données ne sont pas visités.
- Résout le problème de l'utilisation des pointeurs (références d'adresses en mémoire).

II.1.2. Les inconvénients :

RPC a des inconvénients, on peut citer quelques uns :

- Univers de systèmes homogènes.
- Volume de codes et de données à échanger pages par pages.
- Problèmes de partage selon cohérence de la mémoire répartie.

II.2 .Motivations: évolution des middlewares - MOM2 :

Il est fondé sur l'utilisation de messages asynchrones pour faire communiquer les applications

Principe :

Les MOM ont deux modes de fonctionnement principaux :

- Point à point: une application produit des messages et une application les consomme. Les messages ne sont lus que par un seul consommateur. Une fois qu'un message est lu, il est retiré de la file d'attente.
- Publish Subscribe (par abonnement) : les applications consommatrices des messages s'abonnent à un topic (sujet, catégorie de messages). Les messages envoyés à ce topic restent dans la file d'attente jusqu'à ce que toutes les applications abonnées aient lu le message.[4]

II.2.1. Les avantages :

Les avantages de MOM sont représentés par :

- Intégration de multiples protocoles et des multiples plateformes
- Messages définis par les utilisateurs
- GMD : Guaranteed Message Delivery
- Equilibrage de charge
- Tolérance de pannes
- Support pour plateformes hétérogènes
- Gestion et configuration sur interfaces graphiques. [6]

II.2.2. Les inconvénients :

Les inconvénients de MOM sont les suivantes :

- Les MOM sont plus une boîte à outils qu'un outil de répartition transparent comme CORBA ou l'annexe des systèmes répartis d'Ada
- Risque de surcharge du système
- Implémentation propriétaire (messages + architecture)
- Peu portable et peu interopérable.

[2] Message Oriented Middleware

II.3. Motivations: évolution des middlewares – CORBA :

CORBA (Common Object Request Broker Architecture) qui désigne une norme de gestion d'objets distribués conçue par l'OMG pour concurrencer le COM de Microsoft, cette architecture rend possible la communication entre plusieurs applications développées dans des langages différents et installées sur des machines différentes.

CORBA est une norme de communication utilisée pour l'échange entre objets logiciels hétérogènes. Un langage IDL (Interface Definition Language) décrit les traitements effectués et les formats de données en entrée et en sortie. Un bus applicatif, ORB (Object Request Broker) constitue le cœur de CORBA par lequel les requêtes sur les objets transitent.

Principe :

CORBA est basé sur la technologie objet. L'ensemble des développements réalisés dans un environnement CORBA doit se faire dans cet esprit, bien que d'autres technologies puissent être intégrées.

CORBA repose sur une architecture client/serveur et un modèle de communication de type RPC. Cependant, CORBA se veut très universel. D'autres modèles de communications peuvent être instanciés dans une architecture CORBA, notamment la communication par messages. Un point essentiel est l'indépendance de la plateforme aux contextes de développement et d'exécution des applications. [7]

II.3.1. Les avantages :

CORBA présente des avantages importants pour les systèmes distribués comme :

- La transparence ;
- La portabilité ;
- L'interopérabilité ;
- L'adaptabilité ;
- La disponibilité ;
- La stabilité. [8]

II.3.2.Les inconvénients :

Les inconvénients de CORBA sont représentés par :

- La complexité;
- Le prix élevé

- Une formation spécialisée pour les développeurs.[8]

II.4. Motivations: évolution des middlewares – RMI :

RMI (Remote method invocation), Interface pour permettre à des machines java virtuelle de communiquer entre elles.

Principe :

Afin de pouvoir lier l'implémentation présente sur le serveur avec les interfaces du client il existe plusieurs alternatives. Nous pouvons générer des classes Java qui vont implémenter les interfaces, se connecter au serveur, puis déléguer l'appel vers le serveur au travers d'un socket.

Nous pouvons aussi éviter de générer du code grâce au proxy dynamique pour rediriger l'appel du code client vers le serveur dynamiquement. Quelle que soit l'alternative choisie nous voyons là émerger deux agents importants :

- l'objet côté client qui va rediriger l'appel vers le serveur (le Stub) ;
- l'objet côté serveur qui sera atteint par l'objet dynamique côté client (le Skeleton). [9]

La figure suivante décrit la structure de RMI : [9]

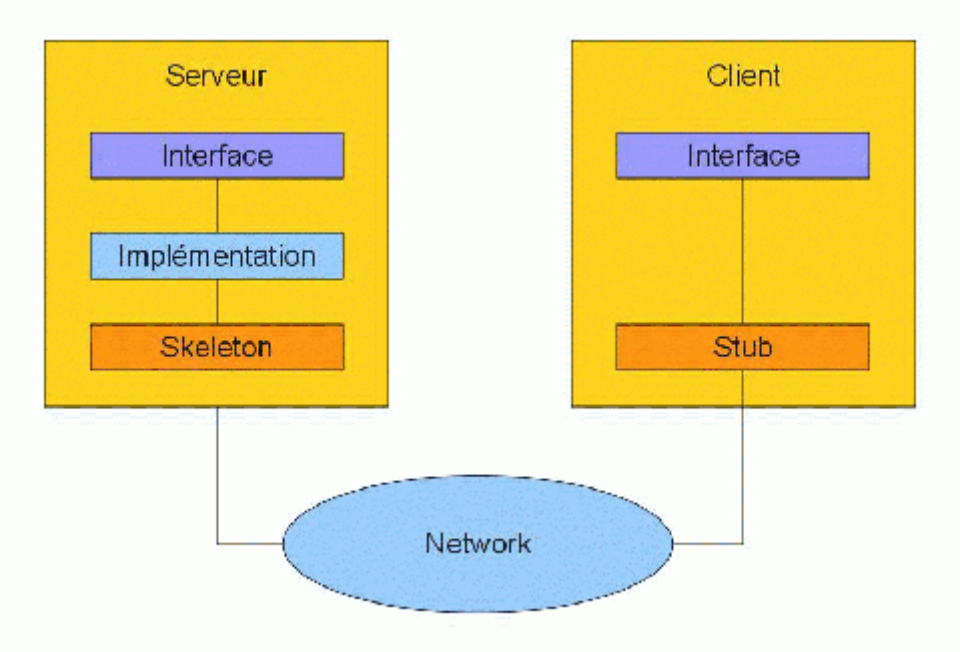

Figure I.2 : Architecture générale de RMI

II.4.1.Avantages :

Parmi les avantages de RMI :

- Il ya une apparence superficielle de la simplicité
- la distance peut être traitée comme si elle était locale
- la simplicité de programmation
- son utilisation est sécuritaire
- il élimine le coulage de mémoire. [10]

II.4.2. Inconvénients :

Les inconvénients importants de RMI sont :

- Utilisation exclusive avec JAVA.
  - Pas d'interopérabilité avec d'autres langages de programmation.
- Relative lenteur à l'exécution.
  - Due aux coûts de la sérialisation des paramètres.
- défaillance du réseau.
- Architecture fortement couplé.

II.5. Motivations: évolution des middlewares - COM/DCOM[3] :

Component Object Model (COM) est une interface standard pour les composants logiciels mis en place par Microsoft en 1994. Il est utilisé pour permettre la communication interne et la dynamique de création de l'objet dans un large éventail de langages de programmation.

DCOM (Distributed Component Object Model) est un logiciel middleware de type ORB : un logiciel moteur qui permet à des objets qui s'exécutent sur un ordinateur d'échanger des services avec des objets exécutés par un autre ordinateur. L´*ORB* fait appel à un *ORB* exécuté par l'autre ordinateur. Il permet en particulier au programmeur de se servir de l'objet sans tenir compte du fait qu'il est sur un autre ordinateur.[11]

Principe :

Un des objectifs de *COM* est de faciliter la création de programmes par assemblage de composants, où chaque composant peut être mis à jour ou amélioré indépendamment des autres. Ceci est rendu possible par le fait que tous les composants peuvent être assemblés selon la même technique. *COM* unifie la manière dont les composants mettent à disposition leurs fonctionnalités et la manière dont les autres programmes

[3] Component Object Model

vont les rechercher et s'en servir5. Les programmes qui adhèrent à la spécification *COM* pourront être réutilisés dans plusieurs langages de programmation, tels que C, Visual Basic, Java, Delphi, FoxPro ou COBOL.

DCOM traitant automatiquement les communications entre les deux ordinateurs, y compris l'application du protocole Remote Procedure Call, l'authentification, la sérialisation et l'utilisation de la mémoire.[11]

II.5.1. Les avantages :

Parmi les avantages de COM/DCOM :

- Simple (coté utilisation).
- Une migration rapide
- La rapidité

II.5.2. Les inconvénients :

On peut donner aussi quelques inconvénients de COM/DCOM :

- Une portabilité discutable.
- Une faible sécurité.
- Spécifique à Microsoft.

La figure suivante décrit l'architecture de COM/DCOM : [12]

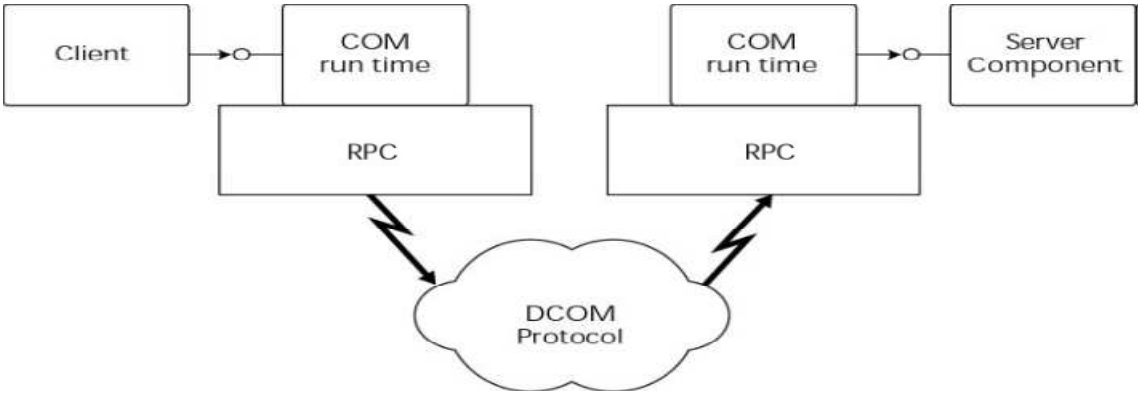

Figure I.3 : Architecture générale de COM/DCOM

II.6. Motivations: évolution des middlewares – Composants :

Un composant est un élément de base dans la construction d'applications. C'est une unité logicielle autonome offrant des interfaces spécifiées permettant entre autre de configurer le composant lui-même. Le composant dispose aussi de fonctionnalités

additionnelles comme la gestion du déploiement, de la composition et sa gestion dynamique. [2]

On a deux modèles très utilisés :

- CORBA Component Model
- J2EE EJB (Entreprise Java Beans).

II.6.1. Intérêt des composants :

On a plusieurs intérêts des composants, parmi ces intérêts :

– Indépendance des composants à la Compilation

– Granularité: Faciliter la création de grandes applications

– Réutilisation: Composants = boites noires indépendantes

– Programmation: Faciliter (parfois !)

– Extensibilité: Compléter un composant sans effets de bord

– Exécution: Maîtrise du cycle de vie

II.7. Motivations : positionnement des services web :

II.7.1. Les besoins des entreprises :

L'entreprise a des besoins au coté informatique, parmi ses besoins on a la flexibilité et l'indépendance dont la possibilité de réorganiser le système de façon rapide, efficace et peu chère, d'autre part on a le partage d'informations et d'applications entre partenaires, clients ou services, on a aussi l'interopérabilité pour que tout le monde n'a pas la même organisation, les mêmes systèmes d'exploitation, les mêmes architectures matérielles, et le dernier besoin est la sécurité et la confidentialité pour garder la garantie de livraison .

II.8.Intérêt des services web :

L'intérêt des services web est représenté par :

- XML est utilisé pour l'échange des données et messages.
- Ils permettent l'intégration de plates-formes hétérogènes via les protocoles d'internet (http ou autre)
- Les développeurs ont l'embarras du choix en matière de langage de programmation : Java, C, C++, Perl, Python, C#, et/ou Visual Basic,
- Peu ou pas de modification des applications existantes

- Permet une intégration faible, les composants sont simples mais peuvent résoudre des problèmes complexes
- L'utilisation du protocole HTTP permet de passer les firewalls.
- Il n'existe pas de client propriétaire
- Outils de développement et de déploiement fournis par les principales plateformes J2EE et Microsoft .NET
- Permet la localisation et l'invocation dynamique d'autres services à partir de registres privés ou publics

III. Conclusion :

Nous avons présenté dans ce chapitre les principales technologies dérivées du domaine des middlewares. Dans le chapitre suivant nous allons éclaircir les concepts, et les standards associés aux services web.

Chapitre II:
Les services web

I. Introduction :

En quelques années, les services web sont devenus le nouveau point de convergence technologique de l'industrie du logiciel.

Un bon nombre d'éditeurs de serveurs d'applications, d'outils de développement, ont placé ces services web au cœur de leur stratégie. Dans ce chapitre, nous décrivons les technologies liées aux services web.

II. Définition :

Au sens large, les services web sont des systèmes logiciels, permettant l'interopérabilité entre plusieurs systèmes logiciels (agents) sur un réseau informatique. Plus spécifiquement, lorsque nous utilisons comme base les normes du W3C, l'interface du système est définie par un langage lisible par un ordinateur (WSDL). D'autres système logiciels vont communiquer avec le service Web selon sa description en utilisant le langage SOAP, généralement en utilisant XML pour sérialiser les messages et HTTP comme protocole réseau.

Les Web Services rendent disponibles à plus grande échelle les intergiciels (middleware) traditionnels.[13]

III. Le concept des web services :

Le concept des Web Services s'articule actuellement autour des trois acronymes suivants:

• WSDL : Web Services Definition Language pour la description des Services Web.

• SOAP : Simple Object Access Protocol pour les appels de services à distance par échange de messages XML.

• UDDI : Universal Description, Discovery and Integration, pour le référencement des services.

Nous décrivons en détails ces trois normes dans la partie VII.

IV. l'intérêt des services web :

Les Services Web comportent de nombreux avantages, ils sont utilisables à distance via n'importe quel type de plate-forme, ils peuvent servir au développement d'applications distribuées et sont accessibles depuis n'importe quel type de clients. Les services web appartiennent à des applications capables de collaborer entre elles de manière transparente pour l'utilisateur.

V. Architecture des Web Services :

Jusqu'ici, l'accès via Internet à une ressource applicative ou à une base de données s'effectuait par l'envoi d'une requête s'appuyant sur des langages de script (PHP, JSP, …).

Il s'agissait donc d'un dialogue entre une couche de *présentation* reposant sur HTML (protocole http) et des applications installées sur un serveur distant.

Avec les Web Services, un dialogue est désormais instauré entre applications qui peuvent être installées sur des machines distantes, et ceci grâce à des standards XML. En effet, afin de dialoguer via Internet, ces applications doivent « parler » le même langage, langage basé sur le XML.

L'architecture de référence des services web se base sur les trois concepts suivants :

- **Le fournisseur de service :** c'est le propriétaire du service.
- **Le client (ou le consommateur de service) :** c'est un demandeur de service. D'un point de vue technique, il est constitué par l'application qui va rechercher et invoquer un service.
- **L'annuaire des services :** c'est un registre de descriptions de services offrant des facilités de publication de services à l'intention des fournisseurs ainsi que des facilités de recherche de services à l'intention des clients.

Les interactions de base qui existent entre ces trois éléments sont les opérations de *publication*, de *recherche*, d'*invocation* et de *lien* (bind).

Pour bien comprendre le fonctionnement de cette architecture, nous expliquons ci-dessous le rôle de chacun des éléments précédents :

La figure suivante décrit l'architecture d'un service web : [13]

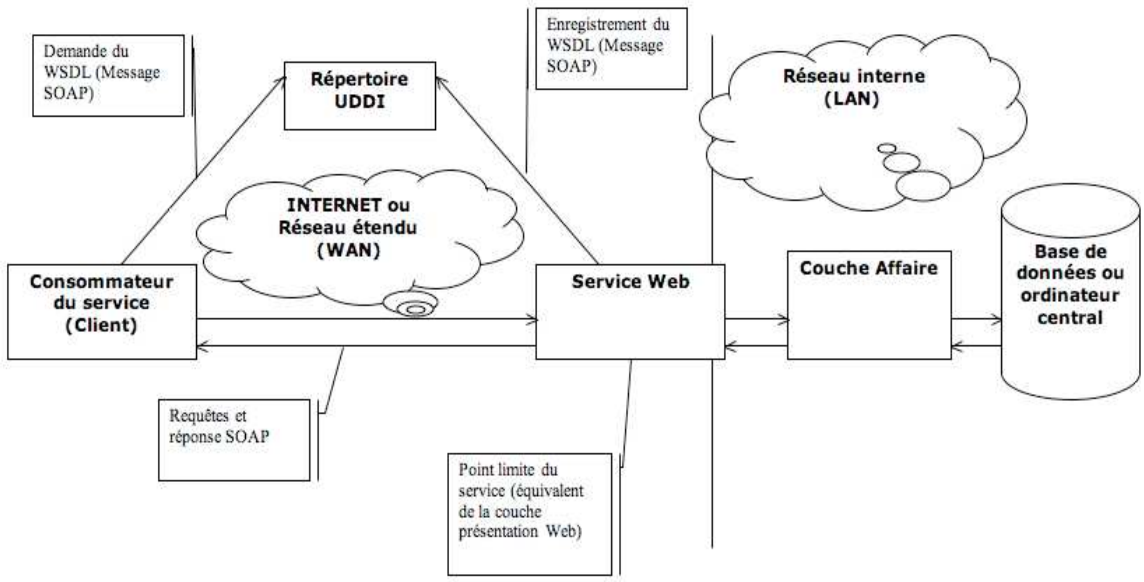

Figure II.1 : Architecture des services web

VI. Les applications des Services web :

L'application des services web est multiple, autant dans les domaines du **B2C**, **B2B** que pour des domaines de gestion, par exemple gestion de stock, gestion commerciale, etc...

B2C (Business to Consumer) : Qualifie une application, un site Internet destiné au grand public.

B2B (Business to Business) : Qualifie une application, un site Internet destiné au commerce de professionnel à professionnel. [17]

VII. Protocoles :

Les Services Web se basent actuellement sur trois protocoles : **SOAP**, **WSDL**, **UDDI**, dans ce qui suit, nous allons détailler le fonctionnement de chaque protocole :

VII.1. SOAP – Simple Object Access Protocol :

Au sein des protocoles liés aux Services Web, SOAP fait figure de pièce centrale. En effet, SOAP définit un protocole simple destiné à l'échange d'informations.

Son objectif est de permettre la normalisation des échanges de données au sein d'architectures distribuées orientées objet [14].

SOAP constitue un standard simple et neutre puisqu'il :

• N’impose pas l’utilisation d’une API particulière.

• N’impose aucun modèle de programmation.

De plus, SOAP est fondé sur un standard : le langage XML.

VII.1.1. Structure d’un message SOAP :

Tout d’abord un message SOAP est un document XML qui doit avoir la forme suivante : [14]

- La structure de l’enveloppe SOAP qui définit une structure générale décrivant le contenu, le destinataire, et la nature du message.
- Les règles d’encodage SOAP qui définissent le mécanisme de sérialisation utilisé pour échanger des objets.
- SOAP RPC qui définit, pour les utilisations synchrones, une convention de représentation des appels et des réponses des procédures distantes.

La figure suivante décrit la structure d’un message SOAP : [15]

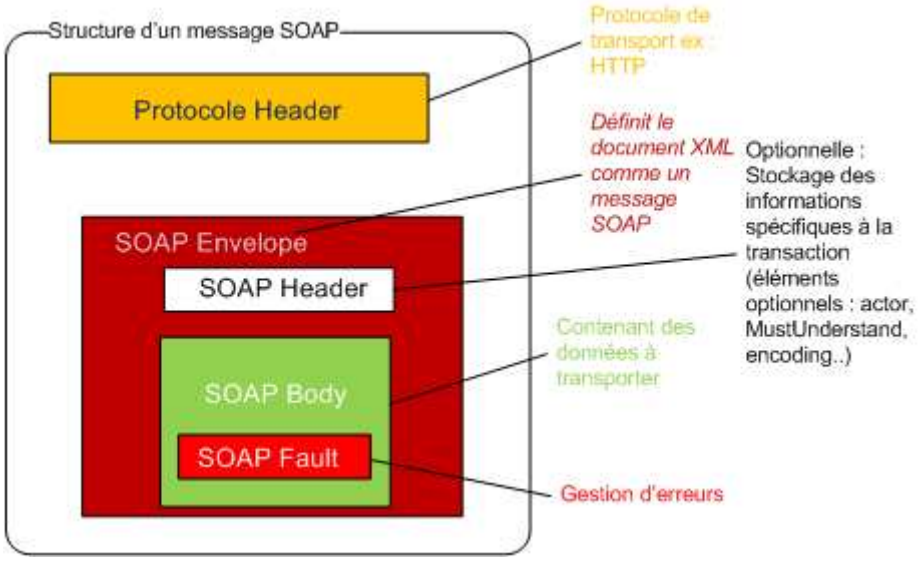

Figure II.2 : les composants d'un message SOAP

Un message SOAP est composé des parties suivantes (figure II.2): [17]

a. Le HTTP Header :

Le protocole HTTP envoie une requête POST. L’entête HTTP se trouve juste avant le message SOAP, et définit le destinataire du message, les règles d’encodage HTTP, etc.

Le champ ***SOAPAction*** peut être utilisé pour indiquer l'intention de la requête SOAP. Cette information peut être utilisée par un firewall pour filtrer les messages. Ce champ est obligatoire mais peut être vide si on n'indique pas l'intention de la requête.

b. L'enveloppe SOAP :

L'enveloppe contient l'espace de nommage définissant la version de SOAP utilisée, et les règles de sérialisation, et d'encodage.

c. Le header SOAP :

Cette partie du message est optionnelle. Elle sert à transmettre des informations nécessaires pour l'exécution de la requête SOAP aux dictionnaires qui recevront le message. On y précise généralement des informations liées aux transactions, à l'authentification, etc. Le header est composé d'un ou plusieurs champs : l'attribut **actor**, désignant le destinataire du header, l'attribut **mustUnderstand**, qui indique si le processus est optionnel.

L'attribut actor permet de préciser l'application à laquelle est destinée l'information contenue dans le header. L'URI http://schemas.xmlsoap.org/soap/actor/next précise en particulier que ces informations sont destinées à la première application qui reçoit le message. Dans le cas où l'actor n'est pas précisé, le header est analysé par le destinataire final du message.

Dans l'exemple suivant, on précise des informations sur l'identification de l'utilisateur et la transaction à laquelle appartient le message :

```
<SOAP-ENV :Header
SOAP-ENV :actor="http://schemas.xmlsoap.org/soap/actor/next"
SOAP-ENV:mustUnderstand="1">
<identifiant numero="124527"/>
<transaction type="compensees" numero="YU75X"/>
</SOAP-ENV:Header>
```

d. Le Body SOAP:

Le body SOAP contient toutes les informations que l'on veut transmettre à l'application distante. Le contenu du Body est normalisé dans SOAP RPC, pour modéliser une requête et sa réponse. Le body de la requête contient l'identifiant de l'objet distant, le nom de la méthode à exécuter et les éventuels paramètres. Le body de la réponse contient le résultat de l'exécution de la requête.

VII.1.2. SOAP RPC:

SOAP RPC définit les conventions permettant d'utiliser SOAP comme un RPC, et le format des messages pour effectuer une requête ou envoyer une réponse. SOAP RPC se base sur les spécifications de XML-RPC. La figure II.3 donne une vue global sur SOAP RPC :

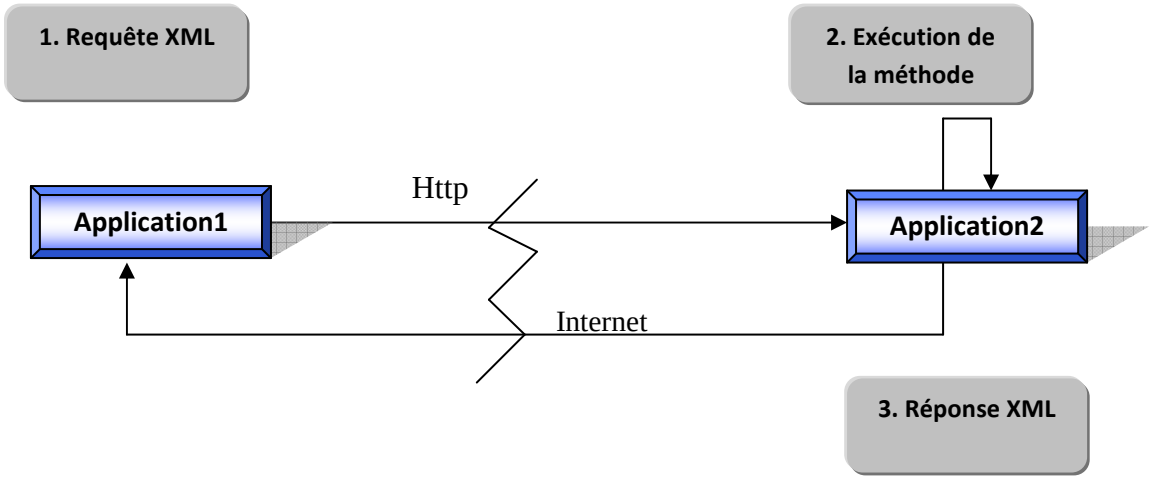

Figure II.3 : SOAP RPC

a. Préambule :

On appelle service SOAP une application dont les méthodes sont accessibles via SOAP. Ce service peut être développé dans n'importe quel langage de programmation. Un service est accessible via un identifiant unique, de type URN.

En Java toute classe peut être rendue accessible via SOAP (en utilisant Apache SOAP par exemple).

Lors de l'appel d'une méthode sur le service déployé, c'est la méthode correspondante de la classe Java qui est exécutée.

Nous donnons un exemple de classe java : « **Calculatrice.java** » que l'on désire rendre accessible via SOAP avec l'identifiant "urn: Calculatrice " :

```
public class Calculatrice {

public int plus(int a,int b) {

 return a+b;          }
```

```
public int moins(int a,int b) {

 return a-b;            }

}
```

b. Requêtes / réponses SOAP :

Pour appeler l'opération "plus", on précise dans le Body SOAP de la requête l'identifiant de l'objet distant, l'opération à exécuter et les éventuels paramètres :

```
<SOAP-ENV:Body>

<m:plus xmlns:m="urn:Calculatrice">

<a type="xsd:int">5</a>

<b type="xsd:int">8</b>

</m:plus>

</SOAP-ENV:Body>
```

Détails de la requête :

- <m:plus xmlns:m="urn:Calculatrice">: l'espace de nommage défini, a pour URN l'identifiant du service (« urn : Calculatrice »). Le nom de l'élément (plus) correspond au nom de la méthode à exécuter.
- Les paramètres d'appel de la méthode sont ensuite ajoutés les uns à la suite des autres. Ici on n'a deux paramètres a et b : de type int. On a affecté à ces deux paramètres les valeurs 5 et 8 respectivement.

Le message SOAP est alors envoyé au service SOAP. Le service exécute la méthode précisée dans la requête, ici c'est la méthode « plus », et retourne ensuite un message SOAP dont le Body contient le résultat de l'opération :

```
<SOAP-ENV:Body>

<m:plus xmlns:m="urn:Calculatrice">

<return type="int">13</return>

</m:plus>
```

</SOAP-ENV:Body>

Le résultat est contenu dans un élément **<return>** :

- L'attribut type précise le type de retour de la méthode exécutée.
- Le contenu de l'élément est le résultat de l'exécution de la méthode.

c. La Gestion des erreurs :

En cas d'erreur lors du traitement de la requête, le serveur renvoie un message SOAP donnant les raisons de l'erreur, dans un message HTTP dont le header commence par :

HTTP/1.1 500 Server Error

L'erreur est détaillée dans le Body SOAP, dans un élément **Fault**, donnant :

- faultcode : le code de l'erreur, destiné à un traitement informatique ;
- faultstring : une explication textuelle, à destination des opérateurs humains ;
- faultactor (optionnel) : en cas d'erreur dans le transport, l'acteur mis en cause peut être précisé : firewall, serveur, proxy, etc. ;
- détail (optionnel) : un détail de l'erreur (par exemple en Java la trace de l'exception renvoyée).

Par exemple, dans le cas où la signature de la méthode de la requête ne correspond pas à la signature de la méthode du service , c-à-d au lieu de la méthode « plus » , on met par exemple « plu » , la réponse SOAP sera comme suit :

<SOAP-ENV:Body>

<SOAP-ENV:Fault>

<faultcode>SOAP-ENV:Client</faultcode>

<faultstring>

Method signature does not match.

</faultstring>

</SOAP-ENV:Fault>

</SOAP-ENV:Body>

d. Pattern d'utilisation:

La norme SOAP ne définit pas les patterns d'utilisation. Ceci dit, à l'image des RPC traditionnels, on pourra être amené à utiliser un objet distant par le biais d'un proxy effectuant les requêtes SOAP, ce qui permet de manipuler l'objet distant comme un objet local :

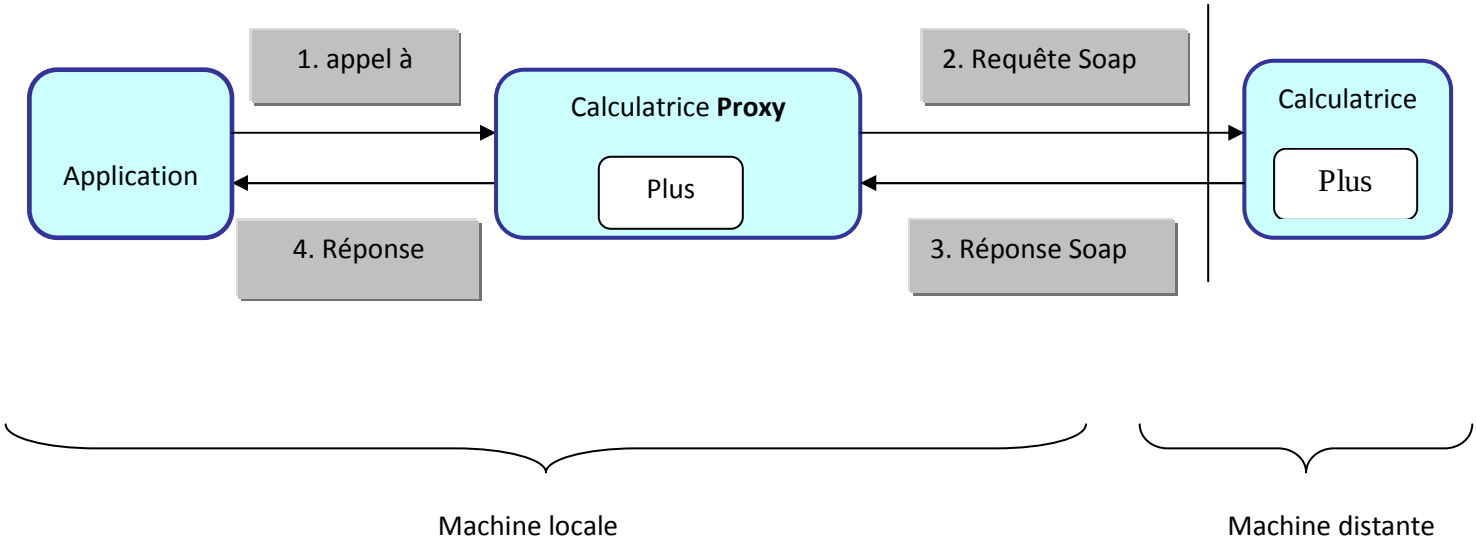

Figure II.4 : Proxy SOAP

1. Une application locale veut appeler la méthode ***plus*** de l'objet distant ***Calculatrice*** :

elle instancie un proxy et appelle sa méthode ***plus*** ;

2. Le proxy est un « raccourci » vers l'objet distant : il effectue la requête SOAP

destinée à l'objet distant.

3. L'objet distant décode le message XML reçu et exécute sa méthode ***plus***. Il

envoie sa réponse au proxy.

4. Le proxy transmet la réponse reçue à l'application.

L'avantage de cette méthode est que l'application n'a pas à se soucier de la localisation de l'objet distant et de la construction de la requête SOAP : elle l'utilise comme un objet local.

VII.2. WSDL – Web Service Description Language :

Le WSDL est un langage dérivé d'XML permettant de fournir les spécifications nécessaires à l'utilisation d'un Service web en décrivant les méthodes, les paramètres et les types de retour.

Le WSDL est aussi l'équivalent de IDL (Interface Definition Language) pour la programmation distribuée (CORBA). Ce langage permet de décrire de façon précise les services Web, en incluant des détails tels que les protocoles, les serveurs, les ports utilisés, les opérations pouvant être effectuées, et les formats des messages d'entrée et de sortie.

Il y eut d'autres tentatives de langages pour résoudre le problème de la définition des services Web. Microsoft a d'abord proposé SDL (Service Definition Language) avec une implémentation fournie dans leur Toolkit SOAP, puis IBM a proposé NASSL (Network Accessible Service Specification Language), dont une implémentation est fournie dans SOAP4J. Microsoft modifia sa première spécification et proposa SCL (SOAP Contract Language), puis les différents acteurs s'accordèrent sur WSDL.

VII.2.1. Structure d'un document WSDL:

La figure suivante décrit la structure d'un document WSDL :

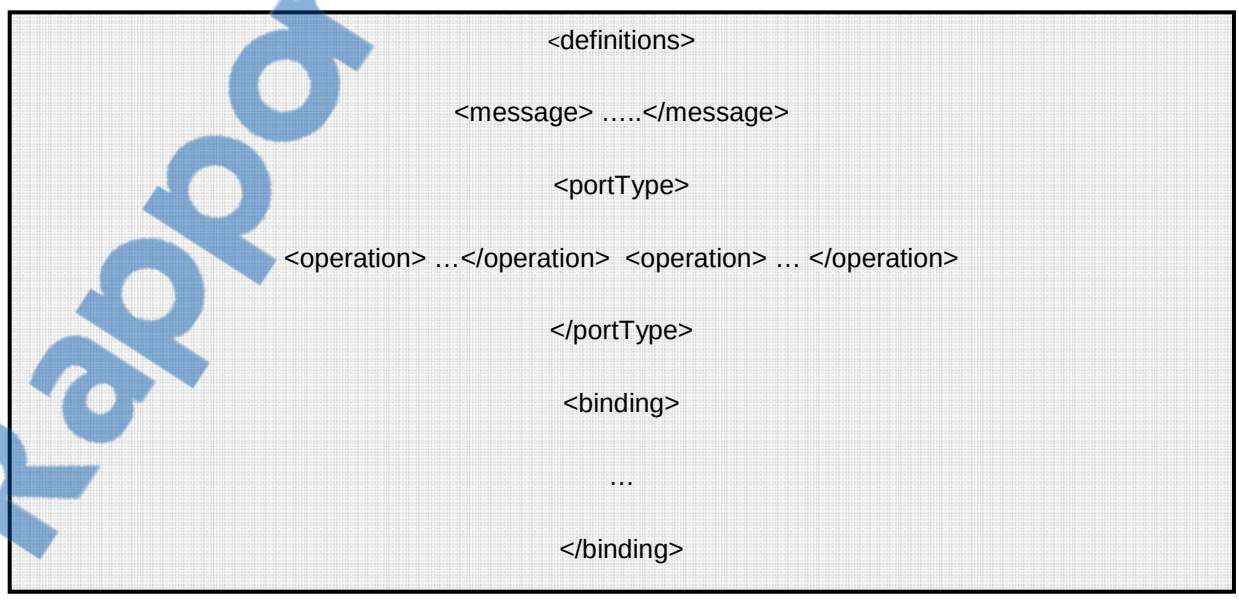

Figure II.5 : Structure d'un document WSDL

- **<definitions>** : Cet élément contient la définition du service. C'est la racine de tout document WSDL. Cette balise peut contenir les attributs précisant le nom du service, et les espaces de nommage. <definitions> contient trois types d'éléments : <message>, <prototype>, <binding> et <service>.
    - <**message**> et <**portType**> : Ces éléments définissent les opérations offertes par le service, leurs paramètres d'entrée et de sortie, etc. En particulier, un <message> correspond à un paramètre d'entrée ou de sortie d'une <operation>. Un <portType> définit un ensemble d'opérations. Une <**operation**> définit un couple message -entrée / message -sortie. Par exemple, dans le monde Java, une opération est une méthode et un portType une interface.
    - <**binding**> : Cet élément associe les <portType> à un protocole particulier. Les bindings possibles sont SOAP, CORBA ou DCOM. Actuellement seul SOAP est utilisé. Il est possible de définir un binding pour chaque protocole supporté.
    - <**service**> : Cet élément précise les informations complémentaires nécessaires pour invoquer le service, et en particulier l'URI du destinataire. Un <service> est modélisé comme une collection de ports, un <**port**> étant l'association d'un <binding> à un URI.

Il est aussi possible de définir des types de données complexes dans une balise <types> juste avant la balise <message>. Enfin, chaque élément WSDL peut être documenté à l'aide de l'élément <documentation>. Cet élément contient des informations liées à la compréhension du document par les utilisateurs humains du service.

VII.3. UDDI - Universal Description Discovery and Integration:

VII.3.1. Principe:

UDDI est une spécification définissant la manière de publier et de découvrir les Services Web sur un réseau. Ainsi, lorsque l'on veut mettre à disposition un nouveau service, on crée un fichier appelé **Business Registry** qui décrit le service en utilisant un langage dérivé d'XML suivant les spécifications UDDI.

Les informations qu'il contient peuvent être séparées en trois types :

• **Les pages blanches** incluant l'adresse, le contact et les identifiants relatifs aux Services Web.

• **Les pages jaunes** identifiant les secteurs d'affaires relatifs aux Services Web.

• **Les pages vertes** donnant les informations techniques.

En utilisant l'API **UDDI** l'utilisateur peut alors stocker ces informations dans un nœud (node) UDDI. Elles sont ensuite répliquées de noeud en noeud (principe assez proche dans l'idée des réplications de DNS).

Une fois ceci est fait, le Service Web peut alors être connu de tous ceux qui le recherchent [6].

Le modèle UDDI comporte 5 types de structures de données décrites sous forme de schéma XML.

Business entity : elle contient des informations sur l'entreprise qui publie les services dans l'annuaire, cette structure contient les autres éléments de l'UDDI.

Business service : ensemble d'informations sur les services publiés par l'entreprise (nom du service, les catégories…).

Binding Template : ensemble d'informations sur le lieu d'hébergement du service (c.à.d. l'adresse du point d'accès du service).

Tmodel : ensemble d'informations sur le mode d'accès au service (définitions wsdl), il peut s'agir aussi d'une spécification abstraite ou d'une taxonomie.

Publisher Assertion : ensemble d'informations contractuelles entre les partenaires. [17]

VII.3.2. La recherche d'un Web Service :

La recherche se fait grâce à un moteur de recherche intégré au site de l'opérateur UDDI choisi. Ce moteur de recherche vous permettra d'affiner votre recherche selon plusieurs critères :

- Nom de l'entreprise
- La localisation de l'entreprise
- Identifiant de l'entreprise

- Le nom du Web Service …

L'API de requête regroupe les appels aux sites opérateurs qui n'ont pas besoin d'authentification particulière.

Les annuaires UDDI ont pour but de localiser virtuellement des Web Services hébergés sur les serveurs du monde entier. Lors de la recherche on peut ainsi se renseigner sur tous les services mis à disposition d'une entreprise, et avoir des informations sur l'entreprise,

Les opérateurs UDDI certifient la sécurité et l'intégrité des Web Services contenus dans leurs annuaires.

Les appels aux sites des opérateurs donnent accès aux différents éléments de l'enregistrement d'un Web Service dans un annuaire UDDI:

• *find_binding:* récupère la liaison du service considéré.

• *find_business* : récupère l'identité de l'entreprise productrice du Web Service.

• *find_relatedbusiness* : récupère la liste des entreprises étant reliées (filiale, département. partenaire, ...) à l'entreprise productrice du Web Service.

• *find_service* : récupère la définition du service.

• *find_tmodel:* récupère le modèle de données associé.

• *get_bindingDetail:* récupère, par une liaison précédemment établie par *find_binding* les champs individuels.

• *get_businessDetail, getbusinessDetailExt:* récupère une entité précédemment établie par *find business* les attributs individuels.

• *get serviceDetail:* récupère un service précédemment établi par *find_service* les attributs individuels du service (prototypes des méthodes).

• *gel_tmodelDetail:* récupère un modèle établie par *find_tmodel* les champs individuels.

VII.3.3. La publication d'un Web Service :

Comme dans WSDL, la liaison UDDI regroupe, pour un protocole de communication donné, les données techniques nécessaires à l'exploitation du Web Service (adresse IP, noms de domaines, les informations sur les modalités d'usage du service, ...)

La publication par une entreprise d'un Web Service requière que celle-ci s'authentifie auprès du site de l'opérateur UDDI. L'entreprise doit s'enregistrer chez l'opérateur si cela n'est pas déjà le cas.

Une fois le site de l'opérateur choisi, les modifications ultérieures ou la mise à jour de

cet enregistrement doivent être faites auprès du même opérateur.

Lors de l'enregistrement, vous pouvez enregistrer simultanément un ensemble d'entreprises affiliées, des relations entres entreprises indépendantes décrivant des accords croisés.

L'API se décompose en *3* groupes:

• La manipulation *(save* et *delete)* .

• L'authentification des commandes par jeton *(get_authToken* et *discard_authToken)* .

• L'ajout de relations inter entreprises *(joint_ventures).* [17]

VIII. Avantages et inconvénients:

Les services web ont des avantages et inconvénients : [16]

VIII.1. Avantages :

• Les services Web fournissent l'interopérabilité entre divers logiciels fonctionnant sur diverses plates-formes.

• Les services Web utilisent des standards et protocoles ouverts.

• Les protocoles et les formats de données sont au format texte dans la mesure du possible, facilitant ainsi la compréhension du fonctionnement global des échanges.

• Basés sur le protocole HTTP, les services Web peuvent fonctionner au travers de nombreux pare-feux sans nécessiter des changements sur les règles de filtrage.

• Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.

VIII.2. Inconvénients :

• Les normes de services Web dans certains domaines sont actuellement récentes.

• Les services Web ont de faibles performances par rapport à d'autres approches de l'informatique répartie telles que le RMI, CORBA, ou DCOM.

• en l'utilisation du protocole HTTP, les services Web peuvent contourner les mesures de sécurité mises en place à travers des pare-feu.

IX. Conclusion:

Un service web n’est qu’une transaction accessible par l’échange de documents XML entre deux URL .Ce dernier offre une réelle souplesse d'utilisation et permet l’accélération du développement d’application.

En adoptant les services web, on facilitera l’interopérabilité et la réutilisabilité du code, mais ceci n’est pas suffisant pour assurer l’automatisation total de l’interopérabilité .en effet on doit prendre en compte des aspects sémantiques lors de la description des services web.

Chapitre III:
La conception et l'implémentation

I. Introduction :

Dans ce chapitre, nous décrivons notre application développée, cette description est divisée en deux parties : La conception et l'implémentation, la première présente quelques aspects relatifs à la conception et la modélisation de l'application, tandis que la deuxième traite la phase d'implémentation ainsi que les outils de développement.

II. La Conception de l'application :

Notre objectif est de mettre en œuvre une application client –serveur constituée des éléments suivants : un service web développé en java, le serveur est fait à l'aide de l'Apache Tomcat et un client java qui invoque le service précédant.

La partie client comporte deux types de clients : un client utilisateur, à travers lequel les utilisateurs de l'application bénéficient du service de conversion offert par le serveur. Et un client administrateur, utilisé par l'administrateur de l'application pour effectuer les différentes tâches d'administration tel que la mise à jour du taux de change.

Pour mieux décrire notre application on va utiliser quelques diagrammes UML : le diagramme de cas d'utilisations, les diagrammes de séquences, le diagramme de classes et le diagramme de déploiement.

II.1. Diagramme de cas d'utilisation :

Le diagramme suivant (figure III.1) représente les vue utilisateur de notre application, il décrit les cas d'utilisation du système implémenté.

II.1.1. la description textuelle :

1. Les différents utilisateurs :

- L'administrateur : c'est l'administrateur de l'application, son rôle est de faire la création et les mises à jour des données utilisées par le serveur dans le processus de conversion de devises.
- Le client : utilise le service de conversion de devises offert par le serveur.

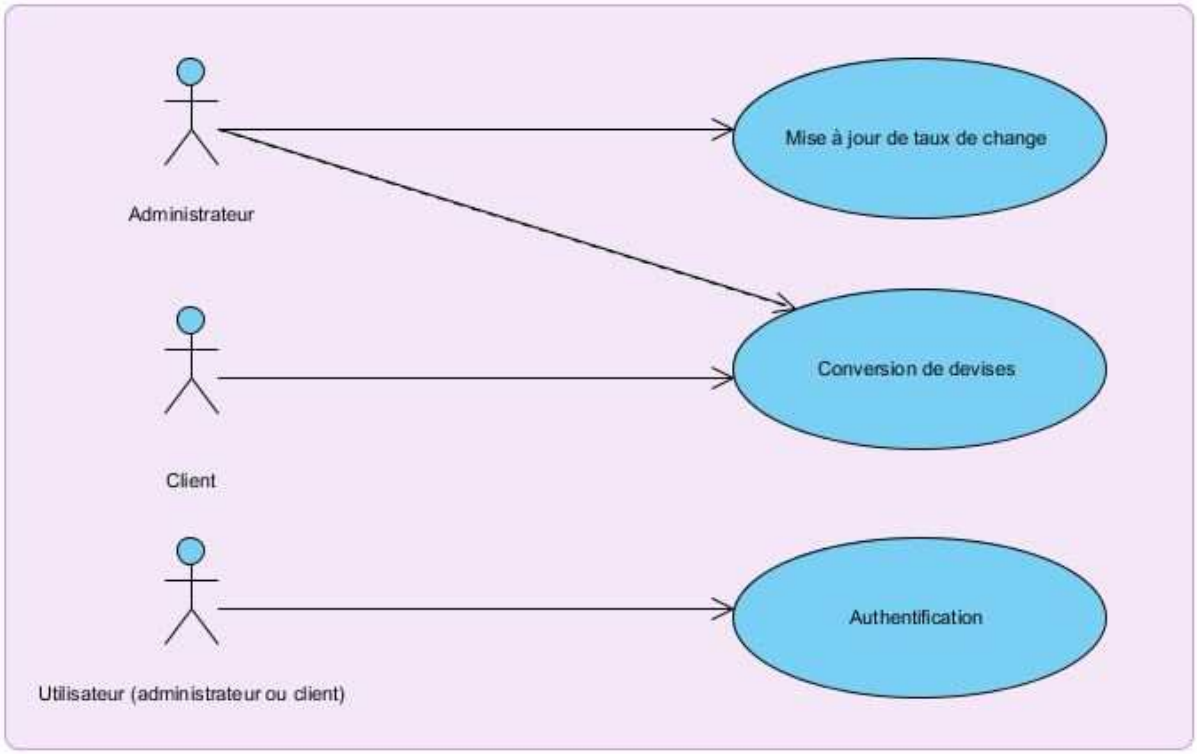

Figure III.1 : Le diagramme des cas d'utilisation

2. Les fonctionnalités du système :

- Le remplissage et mise à jour de tableau : l'administrateur de l'application effectue le remplissage et la mise à jour des valeurs de taux de change utilisées dans les calculs de conversion.
- La conversion de devises : le serveur accepte les requêtes des clients (demande de conversion), et répondre a ces requêtes en effectuant les traitements adéquats.

II.2. Les diagrammes de séquence :

Les diagrammes de séquences montrent les interactions entre les objets, la représentation se concentre sur la séquence des interactions selon un point de vue temporel.

Les diagrammes suivants représentent les diagrammes de séquence pour les cas d'utilisation précédemment décrits :

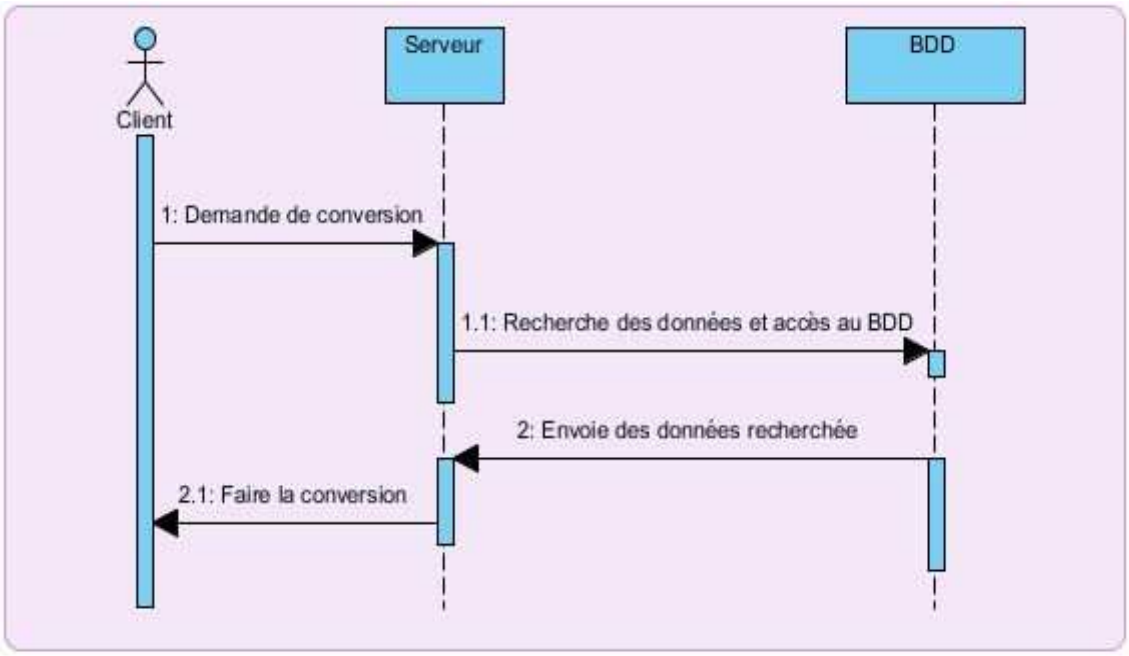

Figure III.2 : Le diagramme de séquence : la conversion de devises

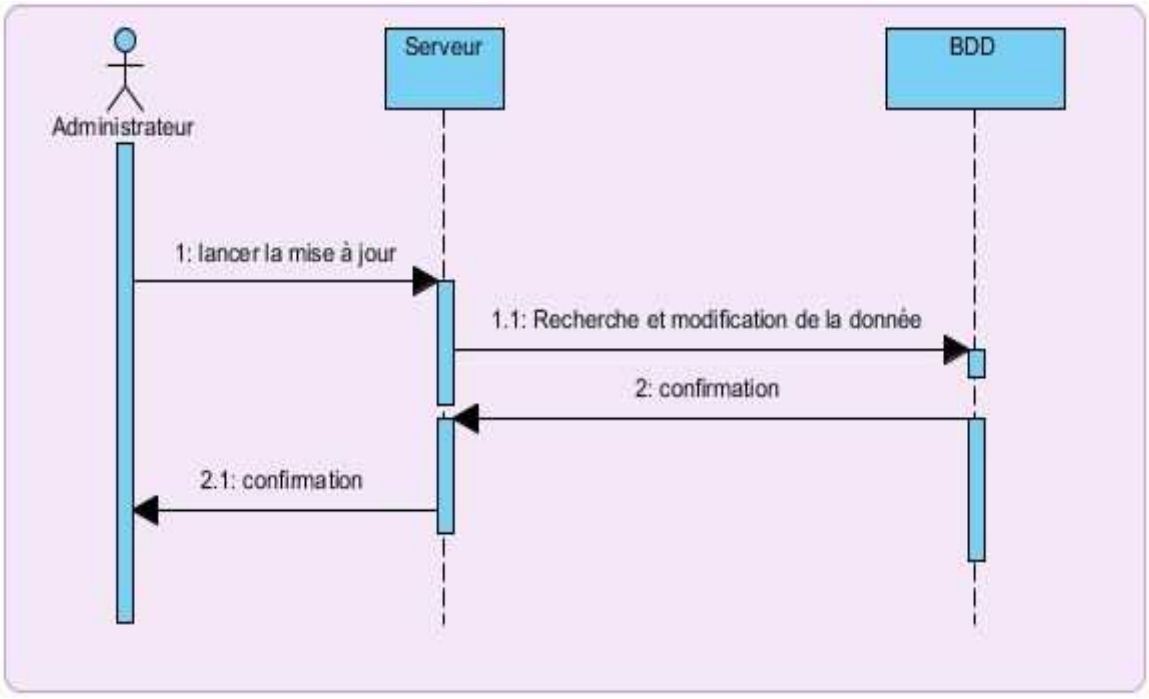

Figure III.3 : Le diagramme de séquence : la mise à jour de taux de change

II.3. le Diagramme de classes :

Le diagramme suivant (figure III.4) décrit l'ensemble des classes de notre application. Nous avons utilisé des simples classes de client et administrateur pour faciliter les choses. La relation entre les différentes classes est une simple relation dépendance.

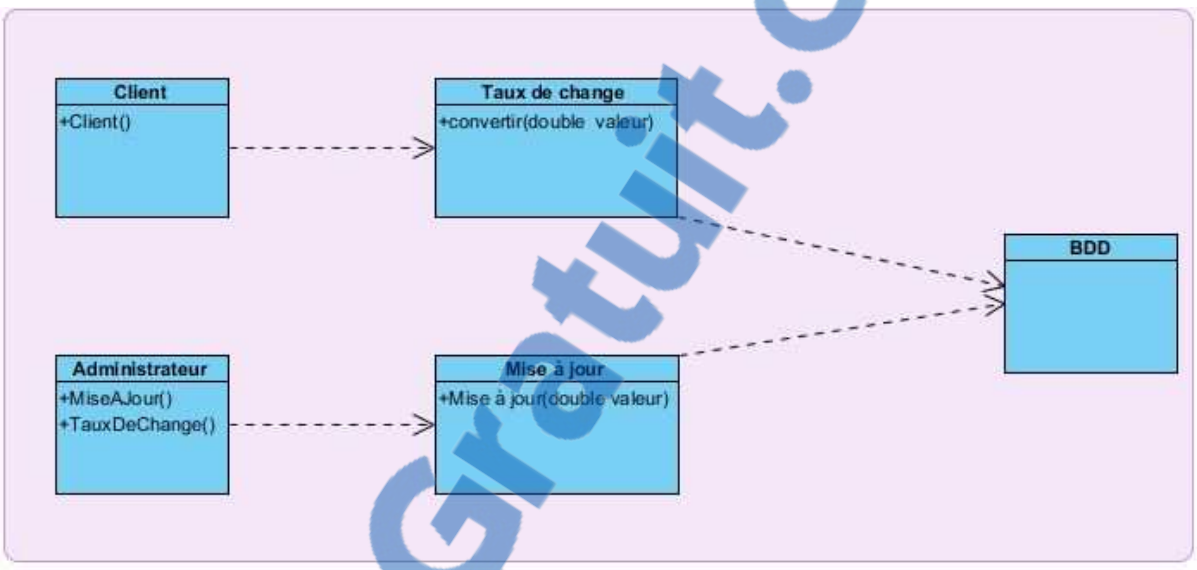

Figure III.4 : Le diagramme de classes

II.4. le Diagramme de déploiement :

Le diagramme suivant représente le déploiement de différents composants de notre application. La partie serveur et la partie client sont déployées dans des machines différentes, avec la particularité que les composants serveur s'exécutent dans un environnement particulier : le serveur d'application.

Dans notre cas nous avons utilisé seulement deux machines pour déployer la totalité des composants du système, cela n'empêche pas d'utiliser plusieurs machines pour héberger les clients et les différentes parties du serveur.

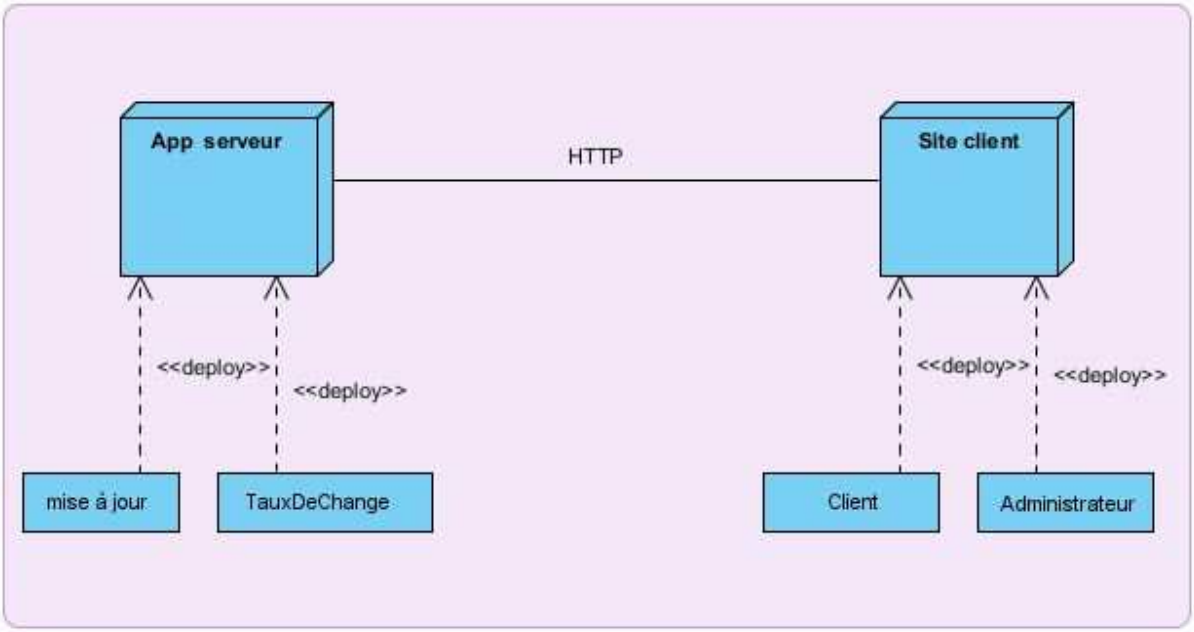

Figure III.5 : Le diagramme de déploiement

III. L'implémentation :

III.1. Les différents outils utilisés :

III.1.1.NetBeans 6.8 :

NetBeans est un environnement de développement intégré (IDE) pour Java, placé en open source par **Sun**. En plus de Java, NetBeans permet également de supporter différents autres langages, comme C, C++, XML et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditer en couleur, projets multi-langage, éditeur graphiques d'interfaces et de pages web).

NetBeans est disponible sous Windows, Linux et d'autres systèmes d'exploitation. Il est lui-même développé en Java, ce qui peut le rendre assez lent et gourmand en ressources mémoires.

III.1.2.Apache Tomcat :

Apache Tomcat est une implémentation open source des technologies Java Servlet et JavaServer Pages (JSP), on dit aussi que c'est un conteneur de servlet. Il y a différentes versions d'Apache Tomcat sont disponibles pour différentes version de spécification de Servlet et JSP.

III.1.3.Axis 2 :

Axis 2 est un moteur de services web, il vise à faciliter le développement de services web en technologie SOAP.

Il a pour objectif d'être plus efficace, plus modulaire et plus orienté XML que la version précédente (Axis).

III.2. l'implémentation côté serveur :

Nous avons basé dans l'implémentation de serveur de notre application sur TOMCAT, un plugin NetBeans nous a facilité le lien entre le serveur et l'environnement de développement.

Le serveur est implémenté en utilisant Tomcat comme serveur web et la bibliothèque Axis2 comme moteur de web service.

III.2.1. le déploiement des services :

Lorsqu'on termine l'implémentation de nos services (mise à jour et taux de change), il faut les déployer dans le serveur d'application. Après ça nous relançons notre serveur (TOMCAT) pour ajouter nos services sur le serveur.

Ce travail permet de faciliter l'accès au nos services par n'importe quel client.

III.3.l'implémentation côté client :

Le client est développé avec l'environnement NetBeans. Il fait l'appel de service à l'aide des bibliothèques de l'Apache Tomcat. Les figures suivantes présentent respectivement les interfaces graphique relatives à la partie client : le client administrateur, et le client utilisateur.

- Pour entrer dans l'application, il faut vérifier si l'utilisateur est un administrateur ou bien un simple client. L'administrateur a le droit de modifier les données de taux de change par contre le client a le droit de faire seulement la conversion de devises.

Figure III.6 : Lancement de l'application

- Interface client administrateur : remplissage des taux de changes.

Figure III.7 : Remplissage de tableau et mise à jour « taux de change »

- Interface client utilisateur pour la conversion de devises

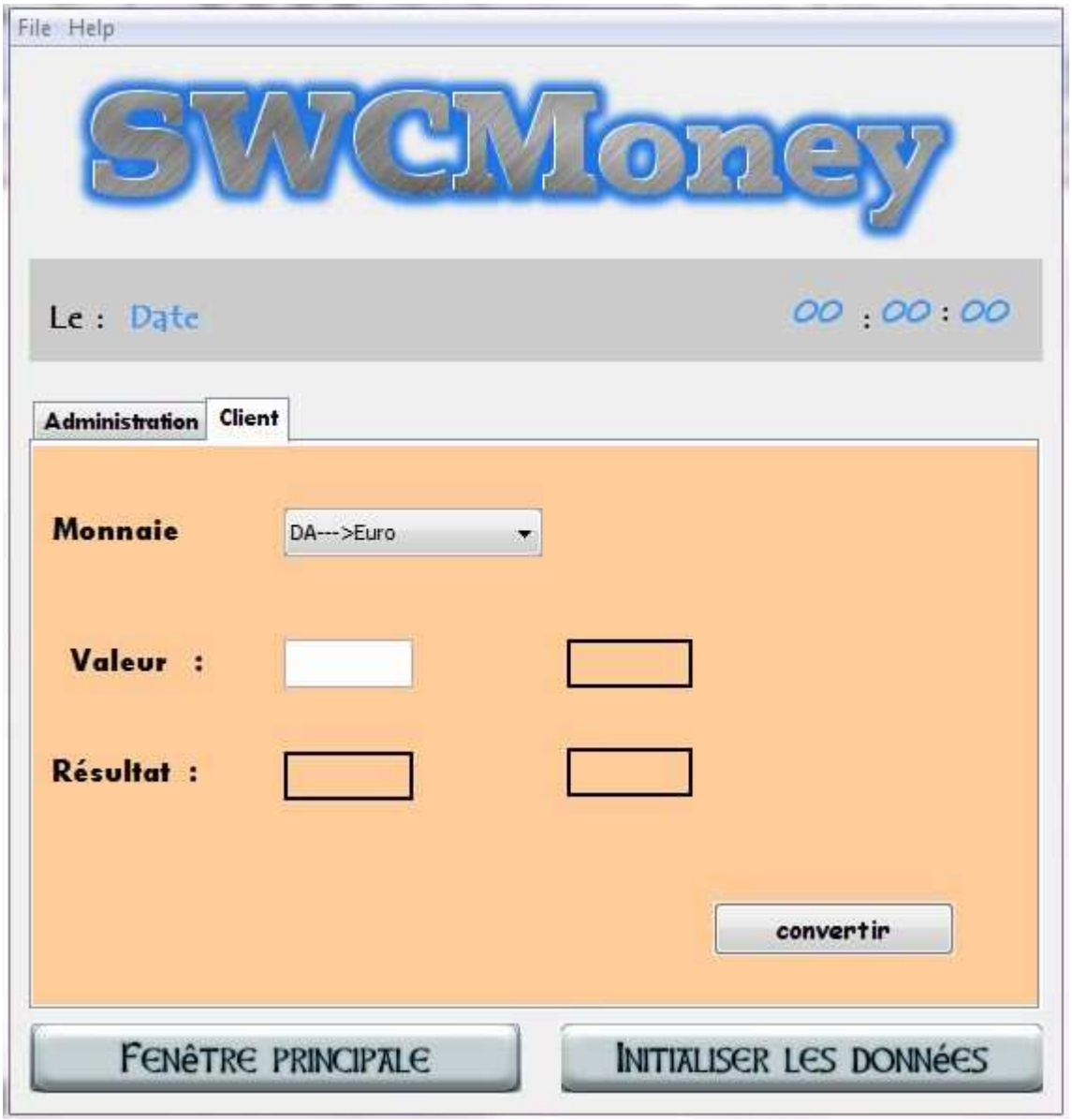

Figure III.8 : Conversion de devises

IV. Conclusion :

Dans ce chapitre nous avons présenté les principes de conception et d'implémentation des différentes parties de notre application.

L'exécution de la partie client et la partie serveur est bien déroulée, et la communication est bien réussi entre ces deux parties, aussi le service de conversion de devises a fourni des résultats fiables comme prévu.

Conclusion générale

Conclusion générale

Le travail présenté dans ce mémoire tourne autour des applications service web, l'objectif était de donner une vue générale sur les technologies suivantes : Apache Tomcat et Axis2, en introduisant quelques concepts et notions théoriques relatives à ces dernières. Quelques notions pratiques sont acquises à travers le développement d'une simple application incluant deux parties (client et serveur), dont le client est un administrateur fait la mise à jour de données de taux de change et la conversion de devises ou bien un simple client fait juste la conversion de devises. Le serveur répond aux requêtes de client.

Finalement, on a réussi à implémenter cette application d'une manière simple et compréhensible, même si elle est loin d'être utilisable sur le plan pratique mais elle reste une bonne expérience, cette dernière est un bon complément de notre formation de base, elle nous a permis d'enrichir nos connaissances théoriques et pratiques, et constitue la base de départ pour des futurs travaux.

Nos perspectives étant de continuer dans le domaine des Web services, de bâtir une base solide pour pouvoir développer des applications plus consistantes, et plus complètes.

Références Bibliographiques

[1] http://fr.wikipedia.org/wiki/Architecture_distribu%C3%A9e (17Avril2013)

[2] **Lin. D**. An information-theoretic definition of similarity. In Proceedings of 15th International Conference on Machine Learning, 1998.

[3] http://fr.wikipedia.org/wiki/Remote_procedure_call (5Mai2013).

[4] **Eric Cariou,** Appels de procédures à distance :RPC .

[5] **Gérard Florin,** L'appel de procédure distante' RPC Remote Procedure Call '.

[6] **Stéphane Frenot,** MOM Message Oriented Middleware.

[7] **Georges Linares**, Introduction à CORBA**.**

[8] **Denivaldo LOPES**, CORBA.

[9] http://alain-defrance.developpez.com/articles/Java/J2SE/micro-rmi/ (14Mai2013)

[10] http://www.trucsweb.com/aSP/trucs.asp?no=287&type=7(15Mai2013)

[11] http://fr.wikipedia.org/wiki/Component_Object_Model(15Mai2013)

[12] http://technet.microsoft.com/en-us/library/cc722925.aspx(20Mai2013)

[13] http://www.benoitpiette.com/labo/introduction-aux-web-services.html#page3(25Mai2013)

[14] **Rada, R**., **Mili, H**., **Bicknell, E**., **Blettner, M**.: Development and application of a metric on semantic nets. In : IEEE Transactions on Systems , Man and Cybernetics .(1989) 17-30.

[15] http://www.siteduzero.com/informatique/tutoriels/les-services-web/le-protocole-de-communication-soap(25Mai2013)

[16] http://connectikpeople.blogspot.com/2011/06/avantages-et-inconvenients-services-web.html#.UXFXn6LwmXM(30Mai2013)

[17] Amara Mohammed, Interopérabilité des services web hétérogène , Mémoire de projet de fin d'étude, université de Tlemcen, 2009.

LISTE DES FIGURES

Résumé

Notre travail consiste à faire un simple service web (conversion de devises), nous avons implémenté les deux parties client et serveur, le client peut être un administrateur qui fait la mise à jour des valeurs de taux de change ou bien un simple client qui a le droit de faire seulement la conversion de devises. Le serveur est implémenté à l'aide des technologies suivantes : Apache Tomcat et Axis2, tant dis que le client est une simple application Java.

Mots clés : client/serveur, Java, Axis2, service web.

Abstract

Our work focus on making a simple web service (devices conversion), we are implemented two parts client and server; the client can be an administrator that can up date change rate values, or a simple client that has the right to make only the devices conversion. The server is implemented with the aid of the following technologies: Apache Tomcat and Axis2, however the client is a simple Java application.

Keywords: client/server, Java, Axis2, web service.

ملخص:

يتمثل عملنا في خدمة ويب (تحويل العملات)، وقمنا بتنفيذ كل من العميل و الملقم ، قد يكون العميل مسؤول (مدير) و تقتصر مهمته في تحديث القيم من أسعار الصرف وفقا للبورصة العالمية أو قد يكون عميل بسيط لديه الحق بتحويل العملات فقط و يتم تنفيذ الملقم باستخدام تقنيات تكنولوجية حديثة مثل : أباش طوم كات و أكساس2. أمّا العميل يتّم تطبيق عليه برنامج جافا.

الكلمات الأساسية : العميل / الملقم - أكساس2 - أباش طوم كات - جافا.