Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

Mémoire de fin d'études

pour l'obtention du diplôme de Master en Informatique

Option: *Système d'Information et de Connaissances (S.I.C)*

Thème

Réalisation d'un service web selon l'architecture REST

Réalisé par :

- **BOUKAIS Nouria**

Présenté le 18 Juin 2017 devant le jury composé de MM.

- *Mme DIDI Fedoua* *(Président)*
- *Mr BELHOCINE Amin* *(Examinateur)*
- *Mme ILES Nawel* *(Encadreur)*

Année universitaire : 2016-2017

RESUME

REST est un style d'architecture réseau pour les services web qui met l'accent sur la définition de ressources identifiées par des URI, et utilise les messages du protocole HTTP pour définir la sémantique de la communication client/serveur: GET pour le rapatriement d'une ressource, POST pour une création, PUT pour une modification/création, DELETE pour un effacement.

Le mémoire décrit la représentation des ressources dans le domaine des web des objets en expliquant comment représenter les services web avec l'architecture REST. Pour cela, on évoque les technologies et techniques de développement connexes à l'utilisation de Web Services REST, comme la technologie des web des objets, les services web ainsi que les différents protocoles utilisés.

ABSTRACT

REST is a network architecture style for web services that focuses on defining resources identified by URIs, and uses HTTP protocol messages to define the semantics of client / server communication: GET for repatriating d A resource, POST for a creation, PUT for a modification / creation, DELETE for an erase.The paper describes the representation of resources in the web of objects by explaining how to represent web services with the REST architecture. To do this, we talk about the technologies and development techniques related to the use of Web Services REST, such as web object technology, web services and the various protocols used.

ملخص

REST هي شبكة النمط المعماري لخدمات الويب التي تركز على تحديد الموارد التي حددها محددات، وتستخدم HTTP الرسائل لتحديد دلالات اتصال العميل /الخادم: GET الحصول على لإعادة مورد ,POST إلى خلق ,PUT لتعديل / إنشاء ,DELETE للحذف.

وتصف المذكرة تمثيل الموارد في مجال الأجسام الويب شرح كيفية تمثيل خدمات الويب مع .REST لهذا، يثير التكنولوجيات ذات الصلة وتقنيات التطوير لاستخدام خدمات ويب REST، مثل الكائنات على شبكة الإنترنت والتكنولوجيا، وخدمات الشبكة والبروتوكولات المختلفة المستخدمة.

Remerciement

Nous remercions avant tout le Bon Dieu de nous avoir donné la volonté de finir ce mémoire

Nous remercions jamais assez la personne qui nous a guidé, conseillé, orienté pas à pas dans notre travail ; notre sincère gratitude envers **Mme Iles Nawel** qui en tant qu'un encadreur qui nous a apporté écoute et aide avec une patience prodigieuse ; que le tout puissant lui soit d'appuis là où ses pas prendront part dans la vie.

Nos gratitudes vont à l'ensemble des enseignants de département d'informatique qui nous ont transmis leurs savoirs et leurs connaissances tout au long de nos études.

Nos remerciements vont aux membres du jury qui ont accepté de sacrifier une part de leurs temps pour examiner notre travail.

Enfin, à tous nos proches, familles et amis pour le soutient et leur présence.

Dédicaces

En signe d'amour et de respect éternel je dédie mon modeste travail

aux êtres les plus chères : Mes parents pour leurs sacrifices, que

Dieu les protègent et puisse les rendre fiers de moi,

A Mon mari Abdedaim pour son soutient,

Ma fille Amel Fareh et mon Fils Iyad Yasser que Dieu les protègent

A Mes soeur

A tous mes amies pour le réconfort qu'ils m'ont apporté toute au

long de ces années .

A tout la famille BOUKAIS et BABAKHALI.

A tout qui m'a aidé a réalisé ce travail.

Table Des Matières

Introduction Générale

Ces dernières années, le développement du Web a été caractérisé par l'utilisation accrue de plusieurs périphériques tels que : les capteurs, les actionneurs et d'autres appareils mobiles. Ainsi, le Web des objets permet d'interconnecter tous les types de périphériques (par exemple, des capteurs médicaux, dispositifs de surveillance, capteurs de température, compteurs intelligents, etc.) et des objets du monde réel, et aussi une partie du Web et assurer une interopérabilité globale.

Dans le but de contribuer à la réalisation du Web des objets, on se base sur les standards du Web par exemple, Uniform Identificateurs de ressources (URI), Représentational State Transfer (REST) et Hypertext Transfer Protocol (HTTP)) pour la bonne représentation des services web.

Les web services sont devenus une technique incontournable pour construire des systèmes distribués faiblement couplés. L'architecture Orientée service a été largement employée dans plusieurs domaines tel que dans les systèmes e-business, e-gouvernement, systèmes automobiles, services multimédia, finances et dans beaucoup d'autres domaines.

Deux styles différents de web services ont été identifiés dans la littérature ; les Services web de types SOAP qui reposent sur le protocole SOAP (à base XML) pour assurer une communication hétérogène, et les Services de type REST qui reposent sur le protocole Http.

Dans ce travail, il nous semble très utile de développer un service web d'une architecture REST. Le service web assure l'interfaçage entre les deux architectures. Ce modeleur garantie le passage des données échangés entre services hétérogènes d'une façon cohérente.

Ce mémoire contient quatre chapitres : le chapitre I s'intéresse au web des objets, le chapitre II passe en revue les services WEB après un chapitre qui représente le web service REST et son architecture et le dernier chapitre pour conception et réalisation de l'application et on conclue par une conclusion générale.
Ce mémoire termine avec une conclusion générale.

Chapitre I : Web des objets

I.1 Introduction

Au cours des dernières années, l'Internet des objets (IoT) a évolué à une vitesse exceptionnelle, connectant un nombre important d'objets hétérogènes (capteurs, actionneurs, smartphones, applications, etc.). Malgré les différents protocoles de communication, il a été difficile d'interconnecter ces objets entre eux. Ainsi, au-dessus de l'Internet, il y'a eu l'apparition du web des objets faisant référence à l'applicatif, qui fonctionne sur cette infrastructure, en lien avec les technologies et standards du Web, il permet à l'utilisateur d'interagir avec les services (simples ou composites). Un objet physique est alors vu comme un ensemble de services accessibles au travers du web et dont l'environnement physique permet de préciser le contexte.

I.2 Définition

La notion du Web des objets est définie par une architecture commune et très utilisée telle que le World Wide Web afin d'y intégrer des objets physiques, permettant ainsi de combler le fossé entre les mondes physiques et numériques [3]. Ainsi tout objet connecté devient alors une ressource disponible sur le Web. Il peut donc à son tour être utilisé dans n'importe quelle application basée sur le Web, conçue pour interagir avec le monde physique.

I.3 But du Web des objets

Le Web des objets consiste essentiellement dans le développement de concepts, d'outils et de systèmes pour la création et l'exploitation de réseaux d'objets associés à des ressources embarquées (puces RFID, capteurs et actionneurs, installations informatiques complexes) accessibles par des services web . Le principal avantage du Web des objets est l'utilisation de normes et de protocoles Web comme le protocole de transfert hypertexte (HTTP) ou les

identificateurs uniformes de ressources (URI), le protocole REST pour bien représenter les services Web.

Il existe différents types de WoT :

- **Web social** : partage des objets, des données ou des fonctionnalités vers une utilisation participative et collaborative.
- **Web physique** : applications de géolocalisation.
- **Web sémantique** : en-tête de métadonnées analysées et indexées par des moteurs de recherche pour permettre à des agents logiciels de partager, de réutiliser ou de combiner ces informations.
- **Real-Time Web** : informations en temps réel livrées en temps opportun.
- **Web programmable** : accès à des données brutes avec une interaction avec les objets physiques par le biais d'API ouvertes.

I.4 Architecture du Web des objets

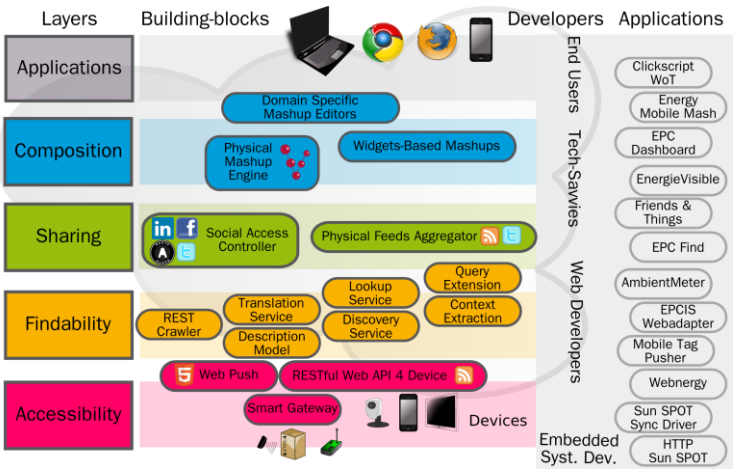
Une architecture axée sur les ressources, a été proposé dans (4) dans le domaine du Web des objets (figure I.1). Elle permet l'intégration des objets à travers les services Cette proposition est élaborée sous la forme d'une architecture en couches, structurée autour de cinq couches:

1- Couche d'accessibilité: traite l'intégration des objets dans le Web.
2- Couche de recherche : effectue la recherche et la localisation de services pertinents dans le WoT.

3- Couche de partage : traite de la gestion de l'accès aux objets (réseaux sociaux).

4- Couche de composition : permet la composition de services et introduit la notion de mashups physiques.

5- Couche Application.



FigureI I.1 : Architecture d'un WoT (4)

L'objectif de cette architecture est de faciliter l'intégration des objets intelligents avec les services existants sur le Web et de faciliter la création d'applications Web en utilisant ces objets.

Pour la construction du Web des objets, on applique les mêmes outils, techniques modèles et langages utilisés dans les applications Web. Mais pour les applications concernant les objets à ressources limitées, d'autres techniques doivent être adoptées tel que : technique AJAX (Asynchronous JavaScript and XML), technique mashups (physique-physique, physique-virtuel), approche Event Driven (3).

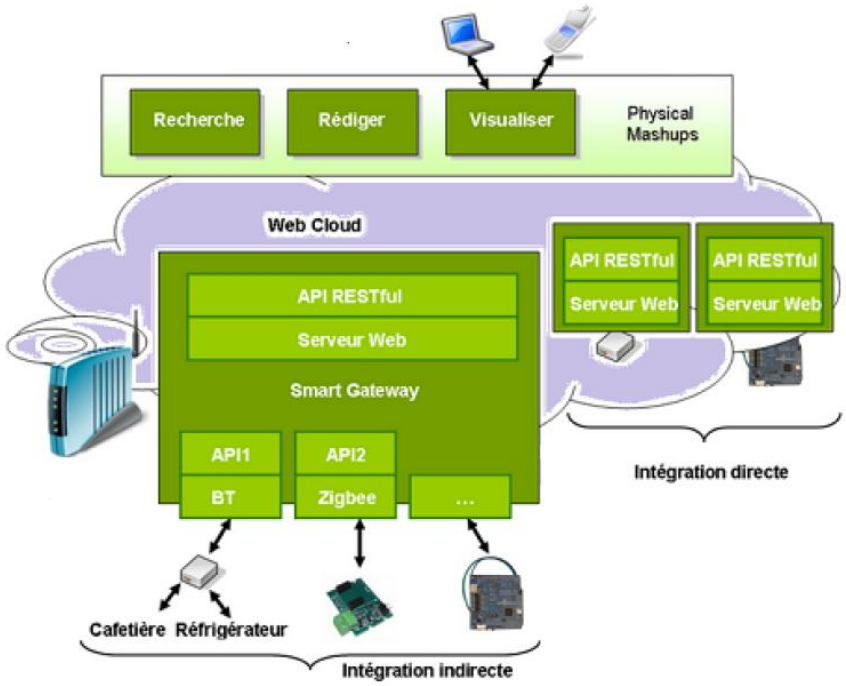
La réalisation du WoT nécessite d'étendre le Web existant afin que les objets du monde réel doivent être intégrés, soit de façon directe ou indirecte dans celui-ci. L'utilisation des services Web est nécessaire dans le but d'exploiter les données et les fonctionnalités des objets physiques en tant que service sur le Web.

I.5 Intégration des objets au Web

Le Web des objets propose d'embarquer des serveurs web dans les environnements systèmes qui sont très contraints et ne disposent pas d'écran. Une des particularités communes à ces serveurs web embarqués est qu'ils utilisent le concept d'AJAX9. Ce

modèle d'application Web permet de construire des applications Web et des sites Web dynamiques interactifs depuis un poste client à travers le protocole http (figureI.2).

Dans le cas des objets intelligents limités en ressources, notamment ceux qui n'ont pas de connexion filaire, les besoins des protocoles TCP/IP et HTTP ne sont pas adaptés car trop consommateurs en termes d'énergie, de calcul, de mémoire et de bande passante. De plus, certains objets intelligents ne les supportent pas nativement. C'est généralement le cas des réseaux de capteurs sans-fil. Dans ce cas, l'intégration du monde physique (objets intelligents) au Web passe par l'utilisation d'un reverse-proxy. Il sert de passerelle entre le réseau interne (les objets qui ne communiquent pas via IP) et le Web.



FigureII.2 : Intégration des objets au Web[1]

I.6 Conclusion:

L'un des buts du Web des objets est d'interconnecter les différents objets physiques et de faciliter leurs utilisations dans des applications composites afin de créer facilement des applications légères et dynamiques utilisant plusieurs services du Web.

Dans le chapitre suivant, on va expliquer la notion des web services ainsi que les différents protocoles utilisés.

Chapitre II : Les services web

II.1 Introduction

Avec la grande utilisation du web, les chercheurs ont développé des logiciels pour assurer et simplifier la communication entre les machines et les applications connectées via le réseau, ces logiciels sont appelés «web services».

II.2 Définition :

Un service web est un système logiciel identifié par un URI, dont les interfaces publiques et les « bindings » sont définies et décrites en XML. Sa définition peut être découverte [dynamiquement] par d'autres systèmes logiciels. Ces autres systèmes peuvent ensuite interagir avec le service web d'une façon décrite par sa définition, en utilisant des messages XML transportés par des protocoles Internet[5] .

II.3 Principales caractéristiques:

Les principales caractéristiques d'un service web sont : [18]

- Fonctionnalité utilisable via Internet
- Inter-opérables: Interface publique décrite d'une manière interprétable par tous
- Systèmes faiblement couplés, client ne connaît pas forcément le fournisseur
- Le transport des données repose sur des protocoles du WEB":HTTP, FTP, SMTP,..
- Standard ouvert: Échange de données s'effectue dans un format standard XML, JSON, HTML, Text, ...
- Le client est chargé d'analyser, traiter et/ou afficher les données reçues
- Indépendante des plates-formes et des langages

II.4 Objectifs

Les services web ont été mises en places afin de répondre à un certain nombre de besoins :

- Remplacer les protocoles actuels (RPC, DCOM, RMI) par une approche entièrement ouverte et interopérable, basée sur la généralisation des serveurs Web avec scripts CGI
- Faire interagir des composants hétérogènes, distants, et indépendants avec un protocole standard (SOAP)
- Simplifier la communication entre ces composants
- Ne pas créer de nouvelles technologies, mais se baser sur celles qui existent déjà (XML, HTTP)

Les services web sont grandement utilises par les entreprises, ce qui leur permet d'exposer un certain nombre de services et d'échanger les informations entre elles [6] .

II.5 L'architecture des services WEB :

L'architecture générale d'n service web est représentée par la figure suivante :

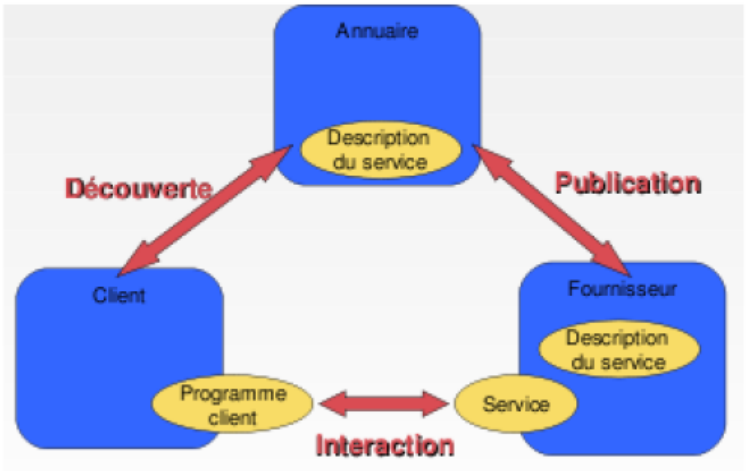


Figure II.1 : Architecture des web service [10]

D'après la Figure II.1 on constate l'ordonnancement des actions suivantes :

1. Définition, déploiement et description du service : c'est-à-dire chercher comment trouver la fonctionnalité fournie et comment y accéder (description WSDL) .

2. Publication de la description du service : Envoi de la description dans un registre (annuaire).

3. Recherche du service : Le client envoie une requête définissant ses besoins au registre, il reçoit en retour une liste de services

4. Récupération de la description du service : Le client récupère par le registre le lien vers le fichier décrivant le service sélectionné. Il ”sait” maintenant comment accéder au service (comment ”l'invoquer”) .

5. Exécution (invocation) du service Web

•Le client peut directement envoyer une requête au service pour réaliser la fonctionnalité.

• Il peut aussi récupérer plusieurs descriptions de services différents et les composer pour obtenir une fonctionnalité avancée.

Les acteurs

1. **Le client**

- Utilise, invoque des services web. On obtient la description du service satisfaisant sa requête

2. **Le fournisseur**

- Est représenté par un serveur d’applications.
- Définit le service.
- Publie sa description dans l’annuaire.
- Réalise les opérations.

3. **L'annuaire**

- Reçoit et enregistre les descriptions de services publiées par les fournisseurs
- Reçoit et répond aux recherches de services lancées par les clients

II.6 Architecture orientée service

Pour mieux appréhender la terminologie liée au Web Service et également situer tous les acteurs entrant en jeu dans les Web Service, nous allons donner une brève description de l'architecture orienté objet.

Une architecture orientée services (notée SOA pour Services Oriented Architecture) est une architecture logicielle s'appuyant sur un ensemble de services simples. Lorsque l'architecture SOA s'appuie sur des web services, on parle alors de WSOA, pour Web Services Oriented Architecture).

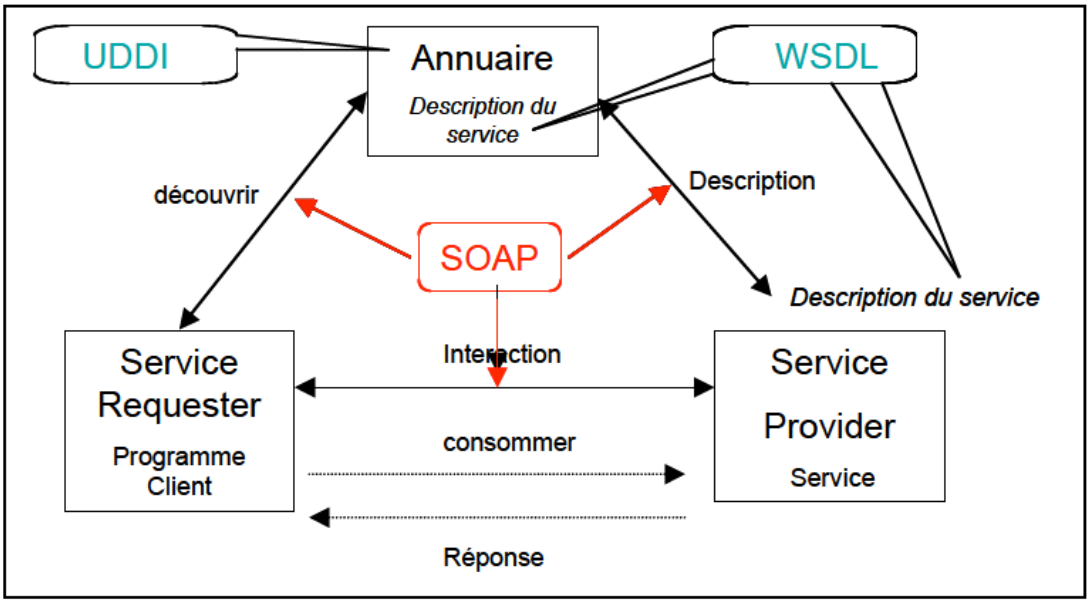


Figure II.2 : Architecture WSOA [6]

Les acteurs dans une architecture SOA :

- **Annuaire – Service Registry**

L'annuaire de services référence l'ensemble des services (et des contrats associés) disponibles au sein du SI, il participe ainsi activement à la mise en œuvre d'une cartographie dynamique du SI. Dans un modèle de bus, l'annuaire peut être auto-alimenté par le service (enregistrement). Les annuaires UDDI forment aujourd'hui le standard de référencement des services.
L'annuaire peut avoir une portée au niveau d'une application, d'une entreprise ou mondial.

- **Service Provider : (Fournisseur de service)**

L'application s'exécute sur un serveur et comporte un module logiciel accessible en XML.

- **Service Requester**

Application cliente se liant à un service et invoquant ses fonctions par des messages XML (REST, XML-RPC,SOAP) [20]

II.6.1 Terminologie dans une architecture SOA

- WSDL (Web Services Description Language) :

Le langage WSDL permet de décrire au format XML des Web Services en précisant les méthodes pouvant être invoquées, leur signature et le point d'accès (URL, port, etc..).
La structure d'un document WSDL est assez complexe et on y retrouve un espace de nom spécifique à WSDL, un espace de nom SOAP, l'espace de nom des messages et l'espace de nom XML Schema permettant de spécifier des types normalisés.

Le WSDL décrit une Interface publique d'accès à un Service Web, notamment dans le cadre d'architectures de type SOA (Service Oriented Architecture).

C'est une description fondée sur le XML qui indique « comment communiquer pour utiliser le service »;La constitution et la génération d'un fichier WSDL peut être réalisées par outils tel que DIA +UML2PHP5 ou ZendStudio.

- UDDI (Universal Description Discovery and Integration) :

UDDI est un annuaire de service fondé sur XML et plus particulièrement destiné aux services Web. Un annuaire UDDI permet de localiser sur le réseau le service Web

recherché, il s'agit d'un élément clé dans les spécifications de Services, car il permet l'accès au répertoire des utilisateurs potentiels de services Web.

- SOAP (Simple Object Protocol communication) :

SOAP est un protocole de communication basé sur XML permettant la transmission de messages en différent service Web[6].

II.7 Communiquer avec les Web Service

II.7.1 Méthode REST (Representational State Transfer)

REST permet de construire une application pour les systèmes distribués comme le Web. Ce n'est pas un protocole ou un format mais une architecture (celle de http). On identifie alors :

- Une URI (Uniform Ressource Locator) qui permet d'identifier la ressource à laquelle on souhaite accéder .
- Une méthode http (POST ou GET) pour définir quel opération on souhaite effectuer sur la ressource .
- Des entête http pour gérer les métadonnées et les informations sur le transport ;

Consommer un service Web REST revient a appeler une simple URL en http, le serveur renvoie sa réponse, la plupart du temps en XML.

Exemple d'échange de données en 'mode REST' (figure II.3) :

- Le client émet la requête http suivante :

http://ws.ct-goat.com/getCityInfos.asp ?uID=xxxxxxxxxxxxx&comID=562

- le client adresse sa requête à l'URI ws.ct-goat.com/getCityInfos.asp en lui transmettant les paramètres uID=xxxxxxxxxxxxx&comID=562

Le provider de service (le serveur de Web Service) retourne un fichier XML [6] .

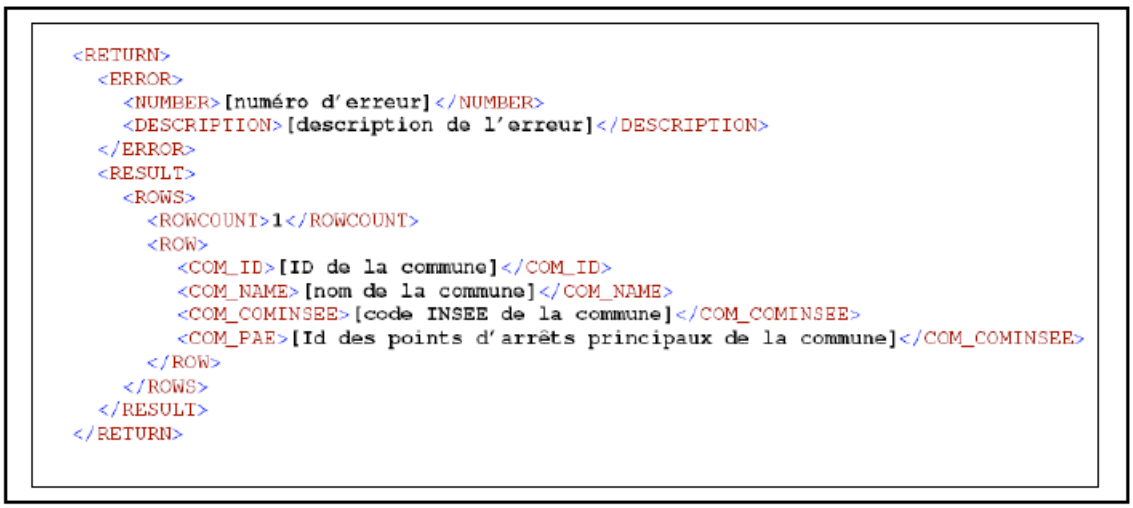


Figure II.3 Exemple de réponse au format XML [6]

- Dans l'exemple ci dessous un service Web 'getCityInfos' retourne les éléments d'une commune à partir d'un code INSEE. Cet exemple à été repris du site 'transisère.fr '
- Cette architecture est très utilisée pour la réalisation de service Web destinés à la communication entre machine.
- Il a l'avantage d'être simple. Il s'agit en réalité de l'utilisation du protocole http.

II.7.2 XML-RPC

XML-RPC est un protocole RPC (Remote procedure call), une spécification simple et un ensemble de codes qui permettent à des processus s'exécutant dans des environnements différents de faire des appels de méthodes à travers un réseau.

Le XML-RPC permet la transmission du point A au point B. Il n'y a qu'un seul appel de méthode par message de requête et une seule valeur renvoyée, sous forme de tableau ou de structure pour transmettre plusieurs valeurs, par message de réponse.

Les processus d'invocation de service à distance utilisent le protocole http pour le transport des données et le langage XML pour leur codage (voir figure II.4). Il est toutefois possible de faire du XML-RPC sur un autre protocole que http.

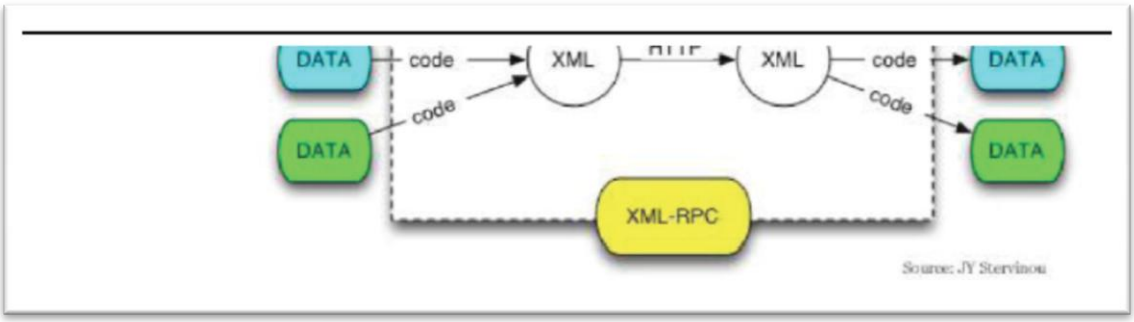


Figure II.4 Schéma représentatif du XML- RPC [6]

XML-RPC permet d'appeler une fonction sur un serveur distant à partir de n'importe quel système (Windows, MacOSX, Linux) et avec n'importe quel langage de programmation. Le serveur est lui même sur n'importe quel système et est programmé dans n'importe quel langage.

II.7.3 SOAP :

SOAP (Simple Object Access Protocol) est un protocole de communication, supportant un RPC (Remote procedural Call) ou encore des messages de type document.

Le SOAP est un protocole d'architecture SOA. Il est produit de Microsoft et IBM, sa première version a été acceptée par le W3C en 2000. Il permet l'échange des documents.

Sa caractéristique principale c'est qu'il est entièrement basé sur le langage XML pour définir la structure du message (l'enveloppe) et les données véhiculées. Le fait que SOAP utilise des protocoles de transport standards de l'Internet est une autre caractéristique essentielle [9] de type XML. Il permet aussi l'envoie des messages via le protocole de transport http.

Le but de SOAP c'est l'échange des différents types d'informations via le net sur des différents environnements en suivant trois (03) majeur éléments :

- La spécification de l'enveloppe SOAP.
- Les règles de codage des données.
- La convention de RPC.

A. Structures de message SOAP :

Chaque message SOAP contient trois (03) éléments, une enveloppe (Figure I.3), un entête et un corps.

Un message soap est composé d'une enveloppe. Ce dernier contient deux parties une entête et un corps.

- **L'entête**

L'entête ou bien le header c'est un élément optionnel s'il existe il doit être le premier élément dans l'enveloppe.

- **Le corps**

Le corps c'est un élément nécessaire dans l'enveloppe où il doit contenir des demandes et des réponses de SOAP.

III.8 Tableau comparatif entre REST et SOAP

Après étude des différents architectures utilisées dans les web services, on a remarqué que l'architecture REST est la plus adaptée et facile pour la description des ressources dans le domaine du web des objets. Pour cela, un tableau comparatif entre l'architecture REST et SOAP, qui représente les points forts de REST.

SOAP	REST
Un protocole de message basé sur XML	Un protocole de style d'architecture

Utilise WSDL pour la communication entre le consommateur et le fournisseur	Utilise XML ou JSON pour envoyer et recevoir des données
Invoque les services en appelant la méthode RPC	Appelle simplement les services via un chemin d'URL
Ne renvoie pas le résultat lisible par l'homme	Le résultat est lisible, ce qui est tout simplement XML ou JSON
Le transfert est sur http, utilise également d'autres protocoles tels que SMTP, FTP, etc.	Le transfert est uniquement via HTTP
Javascript peut appeler SOAP, mais il est difficile à mettre en œuvre	Facile à appeler depuis javaScript
La performance n'est pas formidable à REST	La performance est beaucoup mieux par rapport à SOAP - moins de CPU intensif, plus de code, etc.

II.8 Conclusion

Un des grands avantages des services Web c'est qu'il se repose sur les standards de l'internet et notamment le protocole http. De ce fait l'utilisation des Web Service n'est pas bloquée par proxy et Firewall, L'implémentation des différentes spécifications sur la majorité des langages de programmation à pour avantage de rendre extrêmement simple l'utilisation des Web Service.

Par la suite, et vu l'importance de l'architecture REST, on va l'expliquer plus en détail dans le chapitre suivant.

Chapitre III. L'architecture REST

III.1 Introduction :

REST est un style d'architecture réseau pour Web Services qui met l'accent sur la définition de ressources identifiées par des URI, et utilise les messages du protocole HTTP pour définir la sémantique de la communication client/serveur: GET pour le rapatriement d'une ressource, POST pour une création, PUT pour une modification/création, DELETE pour un effacement.

III.2 Historique

REST est l'acronyme de "Representational State Transfer" inventé par Roy T. Fielding dans sa dissertation "an architecture style of networked systems". Il existe une traduction en français ici. Roy T. Fielding participe de puis 1994 aux travaux du W3C sur les sujets URI, HTTP, HTML et WebDAV et a été le co-fondateur du projet Apache, le serveur Web qui équipe 70% des sites Web de tout l'Internet (IIS de Microsoft n'a que 20%). REST décrit les caractéristiques du Web qui en ont fait son succès. L'explication de la signification de REST telle que donnée par Roy T. Fielding est la suivante : "Representational State Transfer évoque l'image du fonctionnement d'une application Web bien construite : un réseau de pages Web (une machine à états finis virtuelle) où l'utilisateur progresse dans l'application en cliquant sur des liens (transition entre états) ce qui provoque l'affichage de la page suivante (représentant le nouvel état de l'application) à l'utilisateur qui peut alors l'exploiter"[19].

III.3 Définition :

On a plusieurs définitions sur l'architecture REST, on cite quelque unes :

- Selon Roy Thomas Fielding: «*REST is a hybrid style derived from several of the network-based architectural styles and combined with additional constraints that define a uniform connector interface*» [11].

Roy Thomas Fielding a créé cette architecture « REST » selon sa définition le REST est un modèle hybride dérivé de plusieurs modèles basés sur les concepts réseau.

- Selon Martin Kalin: « *REST is a style of software architecture for distributed hypermedia systems; that is, systems in which text, graphics, audio, and other media are stored across a network and interconnected through hyperlinks*. »[13]

On conclut du point de vue de " Martin Kalin" que REST est un style d'architecture pour les systèmes hypermédia distribués, c'est un système auquel le texte, les graphiques,

III.4 Architecture

La conception de l'architecture REST est basée sur les contraintes suivantes [11]:

III.4.1Le modèle vide

Le modèle vide représente simplement un ensemble vide de contraintes décrit un système dans lequel aucune frontière distincte n'existe entre les composants. C'est le point de départ pour notre description de REST.

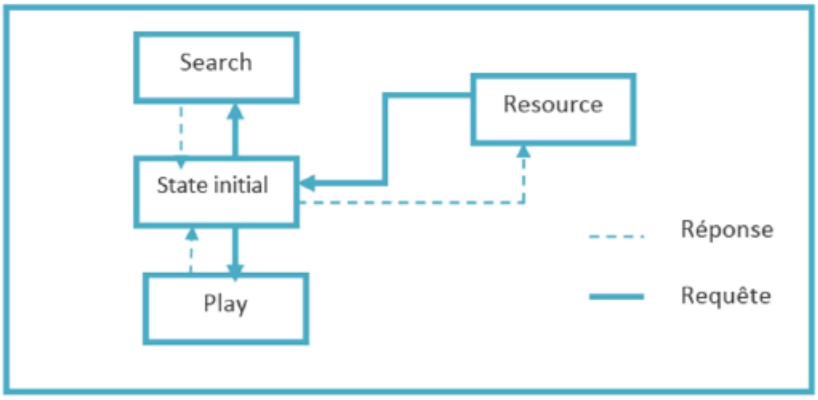
III.4.2Le modèle client-serveur

C'est la première contrainte ajoutée au REST pour séparer les concepts. Cette séparation permet aux composants d'évoluer indépendamment.

III.4.3Le modèle sans état

C'est un modèle concerne l'interaction entre le client et le serveur pour chaque requête du client vers le serveur doit contenir toutes les informations nécessaires et il ne peut pas profiter d'aucun contexte sur le serveur.

Cette contrainte assurer la visibilité, la fiabilité et faculté de montée en charge. Mais cette contrainte diminue la performance du réseau en répétant les données (surplus par interaction) envoyées dans une série de requêtes. La figure III.1 représente un schéma du modèle sans état.



FigureIII.1: Schéma représentant un modèle sans état[11]

III.4.4 Le modèle cache

Cette contrainte est basée sur la mise en cache des donnés d'une réponse de façon implicite ou explicite pour améliorer la performance de réseau.

III.4.5 La contrainte interface uniforme

Le point fondamental qui distingue le modèle d'architecture REST des autres modèles est la mise en exergue d'une interface uniforme entre les composants.

Cette contrainte permet la simplicité de système, l'amélioration de la visibilité d'interaction et la mise en œuvre sera être découplées des services qu'elles fournissent. Mais elle n'est pas optimale pour d'autre forme d'interaction architecturale.

III.5.6 Le modèle Système en couche

Le modèle de système en couches permet à une architecture d'être composé de couches hiérarchiques en contraignant le comportement des composants, pour offrir un équilibrage de charges pour les services.

III.4.7 La contrainte code à la demande

Un client a le droit d'accès à un ensemble de ressources, mais il ne sait pas comment ils sont traités. Il envoie une requête au serveur traitant qui est distant et il l'exécute localement.

Les avantages de code à la demande incluent la possibilité d'ajouter des fonctionnalités à un client déployé, ce qui permet d'améliorer l'extensibilité. Mais le code à demande réduit la visibilité, ce manque de visibilité conduit à des problèmes de déploiement si le client ne peut pas faire confiance aux serveurs c'est pour ça cette contrainte est optionnelle.

III.5 Principes de REST

L'architecture REST repose sur les principes décrits dans les sous sections :

1. Adressage

L'adressage est l'idée que tous les objets et les ressources de votre système est accessible par un identifiant unique . Cela semble à une évidence, mais si on pense à ce sujet, l'identité de l'objet normalisé n'est pas disponible dans de nombreux environnements.

Ce n'est pas une grosse affaire pour une application, mais avec la nouvelle popularité de la SOA, on se dirige vers un monde où les applications disparates doivent être intégrer et interagir. Ne pas avoir quelque chose d'aussi simple que le service d'adressage normalisé ajoute toute une dimension complexe aux efforts d'intégration [12].

2. Interface

Le principe de REST d'une interface limitée est peut-être la pilule la plus difficile pour un développeur COBRA ou SOAP expérimenté à avaler. L'idée derrière cela est qu'on s'en tient à l'ensemble fini des opérations du protocole d'application lequel vous distribuez vos services sur.

Cela signifie qu'on n'a pas un paramètre "action" dans URI et on utilise uniquement les méthodes de HTTP pour les services web. HTTP a un petit ensemble fixe de méthodes opérationnelles.

a) GET

Le but de la commande GET (Figure III.2) est de récupérer une représentation d'une ressource. Cette méthode ne devrait jamais causer d'effets secondaires.

Le risque d'abuser le « GET » est que les ressources seront modifiés ou involontaire conséquences pourraient corrompre ou affecter les ressources de façon indéterminable.

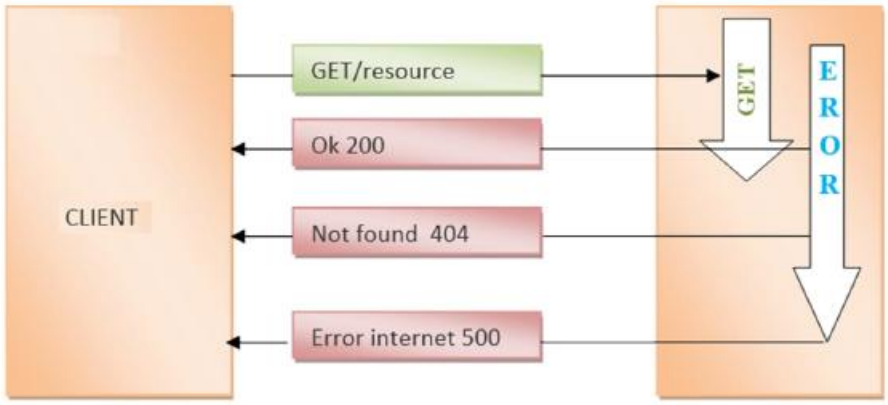


Figure III.2 Méthode http GET[12]

b) POST

La méthode POST permet de crée une nouvelle ressource sur le système POST Cette méthode a essentiellement deux objectifs : l'un qui s'inscrit dans les contraintes de REST, et l'autre qui va REST extérieur et introduit un élément de style RPC [14].

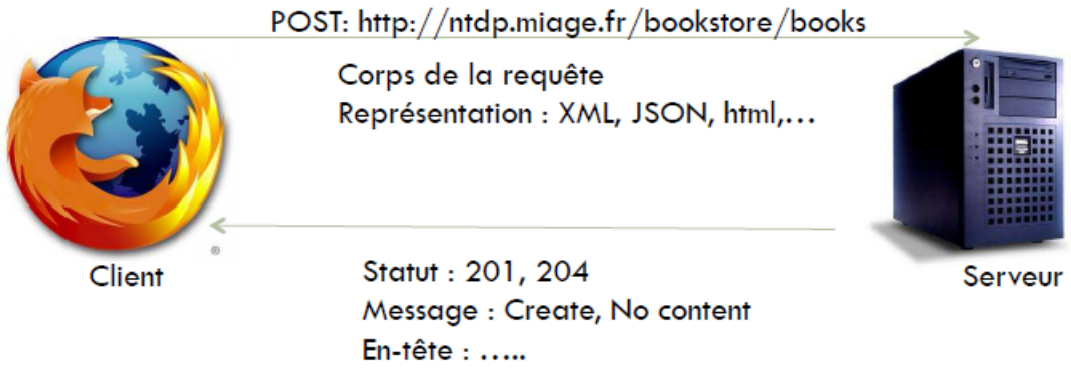


Figure III.3 Méthode http POST [15]

La méthode POST est conçu pour :

- Annoter des ressources existantes ;
- Afficher un message à un babillard, forum de discussion, liste de diffusion, ou un groupe similaire des articles.
- Fournir un bloc de données, tel que le résultat de la soumission d'un formulaire, à un traitement de données processus ;
- Extension d'une base de données via une opération d'ajout.

Le mécanisme de POST [14] :

- Un client Java effectue une requête à l'URI « http://restfuljava.com/ étudiants /Jane,»
- La demande POST porte la charge utile sous la forme d'un fichier XML.
- Le serveur reçoit la demande et le cadre de REST permet à manipuler le code dans le pour stocker la représentation
- Une fois le stockage de la nouvelle ressource est terminée la réponse est renvoyée :

Si c'est un succès, nous envoyons un code de 200 ; sinon nous envoyons le cas échéant code d'erreur.

c) DELETE

La méthode DELETE supprime une ressource. Le serveur devrait retourner une réponse indiquant le succès ou l'échec de l'opération. Les réponses à la méthode DELETE ne sont pas mises en cache.

- Un client soumet une demande de suppression d'une ressource. DELETE /utilisateurs / john HTTP/1.1 Hôte : www.example.org
- Le serveur crée une nouvelle ressource et retourne une représentation indiquant l'état de la tâche.
- Le client peut interroger l'URI http://www.example.org/task/1 pour connaître l'état de la demande.

```
HTTP/1.1 202 Accepted
Content-Type: application/xml;charset=UTF-8
<status xmlns:atom="http://www.w3.org/2005/Atom">
<state>pending</state>
<atom:link href="http://www.example.org/task/1" rel="self"/>
<message xml:lang="en">Your request has been accepted for  processing.</message>
<created>2009-07-05T03:10:00Z</ping>
<ping-after>2009-07-05T03:15:00Z</ping-after>
</status>
```

Figure III.4 Exemple de la méthode Delete[12]

d) PUT

Cette méthode est pour créer des nouvelles ressources uniquement lorsque le client peut contrôler une partie de l'URI. Par exemple, un serveur de stockage peut attribuer une URI racine de chaque client et de laisser les clients à créer de nouvelles ressources à l'aide de cette racine URI comme un répertoire racine sur un système des fichiers.

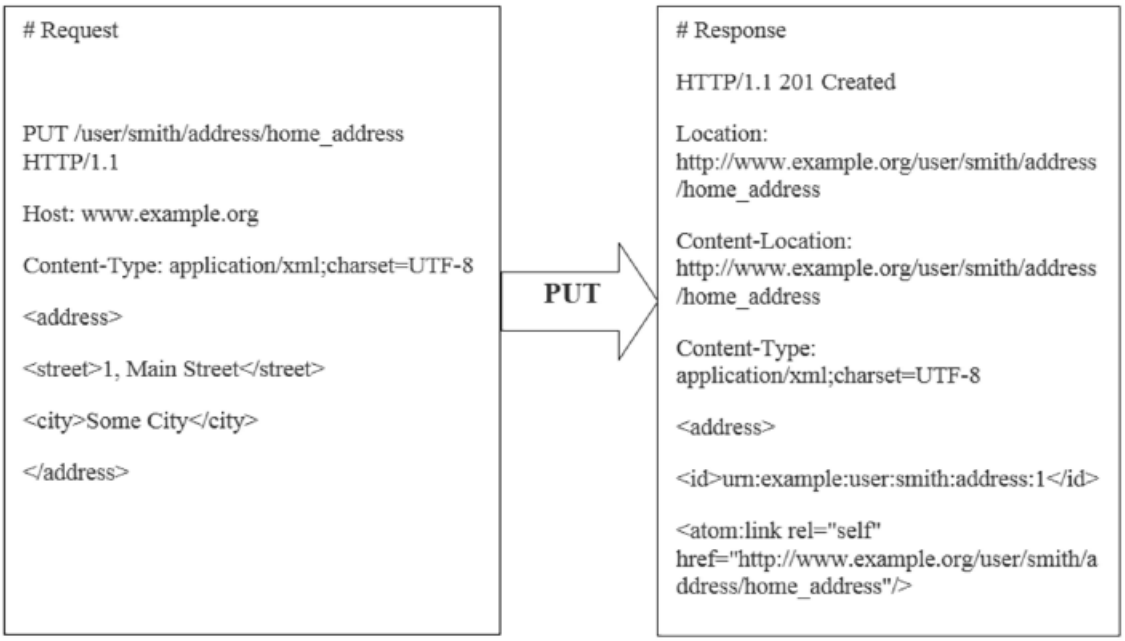


Figure III.5 Exemple de la méthode PUT[12]

3. Représentation des ressources

Parce que REST et HTTP ont une approche multidimensionnelle à l'adressage , la méthode choix, et le format de données, on a un protocole beaucoup plus découplées qui permet à un service d'interagir avec une grande variété de différents clients d'une manière cohérente.

III.6 Fonctionnalités des services RESTFUL

Parmi les fonctionnalités et les spécificités des Services de type REST, on trouve entre autres :

1. Dans une demande, l'appariement d'un verbe HTTP telles que GET avec un URI comme http://.../

2. Le service utilise des codes de statut HTTP tels que 404 (ressource non trouvée) et 405 (méthode non autorisé) pour répondre aux mauvaises demandes.

3. Si la demande est bonne, le service répond avec une représentation XML capturant l'état de la ressource demandée. Jusqu'à présent, les honneurs de services GET demandent, mais les autres verbes CRUD seront ajoutés à la prochaine révision.
4. Le service ne profite pas des types MIME.
5. La mise en œuvre des services RESTFUL ne contraint pas de la même manière comme un service SOAP basé précisément parce qu'il n'y a pas de contrat de service formel.
6. La mise en œuvre est flexible mais bien sûr même pour un ad hoc.

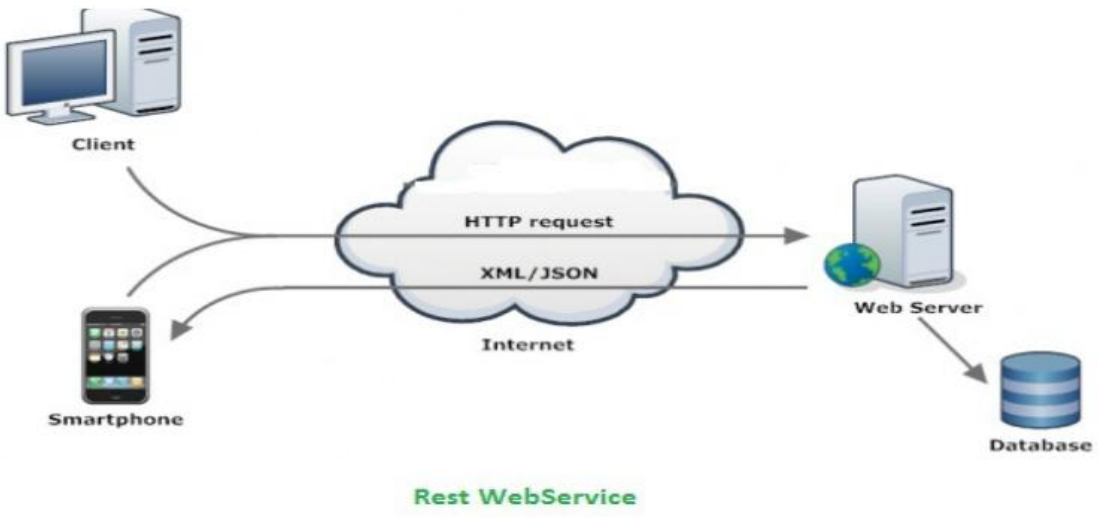


Figure III.6 web service REST [18]

III.7 SOAP VS REST

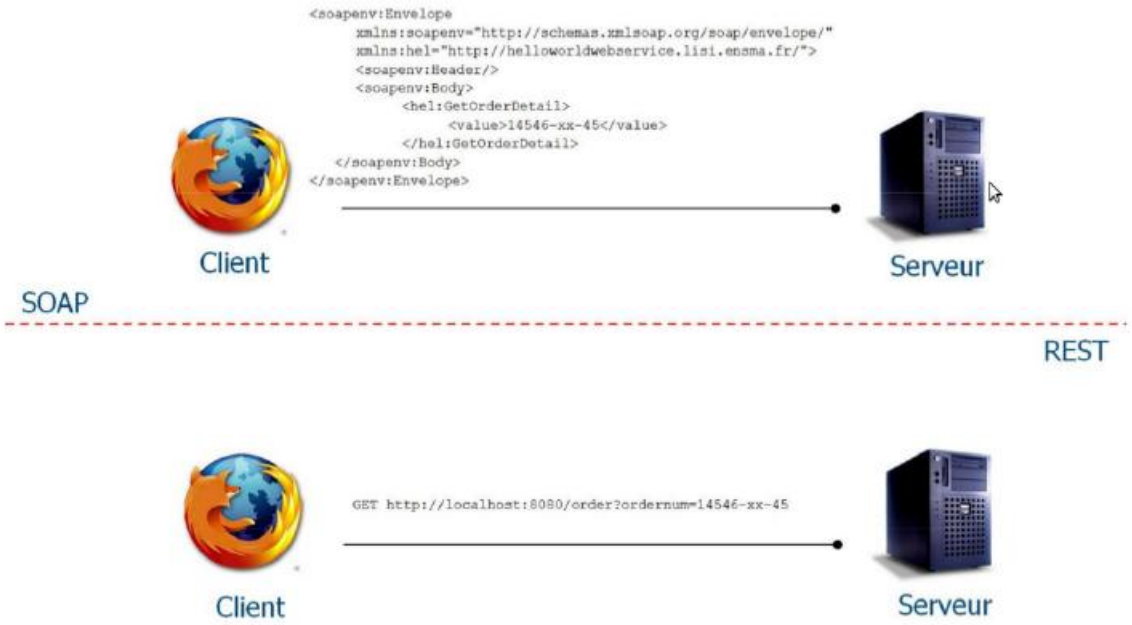


Figure III.7 SOAP VS REST [15]

D'après cette architecture on conclue que l'architecture REST plus facile a utiliser

III.7 Conclusion

Après cette étude, on résume que REST est :

- Structurant: à cause du nommage des URI
- Efficace: utilise bien la machinerie HTTP
- Évolutif: en termes de nouvelles URIs ou représentations
- Indépendant des mises en œuvre (jsp, restlet, etc.)

REST est plus simple à implémenter, les services web l'utilisent afin de gérer des ressources distantes avec toutes les phases classiques, de création, de récupération, suppression et de modification .

Le chapitre suivant décrit la phase conception et réalisation d'un service web avec REST, en utilisant une application de gestion de banque.

IV – Conception et réalisation de l'application

IV.1 Introduction

L'objectif de notre mémoire est la réalisation d'un service Web en se basant sur l'architecture REST. Ce qui va nous aider à représenter n'importe quel objet physique dans le monde du « web des objets ». Le service web qu'on a choisi concerne la gestion d'un compte bancaire. Les cas d'utilisation sont les suivants :

- Créez un compte
- Verser vers un compte
- Retirer de l'argent d'un compte
- Virement entre 2 comptes

Pour cela nous utilisons l'architecture JAVAEE avec EJB qui permet de créez notre projet. Cette architecture facilite le travail, vu qu'elle est dotée des quatre fonctions : Get, Put, Delete, Post , et elle nous permet d'avoir notre web service de manière plus simple et pratique.

Donc nous allons décrire dans ce chapitre, la phase de conception ainsi que l'architecture suivie et les outils utilisés.

IV.2 La démarche UML :

UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par « Langage de modélisation unifié ». La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc.

Réaliser ces diagrammes revient donc à modéliser les besoins du logiciel à développer.

.

1. Diagramme de classe :

Les diagrammes de classes permettent de spécifier la structure et les liens entre les objets dont le système est composé : ils spécifient qui sera à l'œuvre dans le système pour réaliser les fonctionnalités décrites par les diagrammes de cas d'utilisation.

Les classes qui sont utilisé dans notre travail sont :

La classe compte : contient les entités code, solde, date création, et elle est constituée des méthodes suivantes :

- Ajouter un compte
- Afficher les informations d'un compte
- Afficher la liste des comptes
- Retirer un montant
- Verser un montant
- Virement d'un montant d'un compte à un autre compte

La classe banque local : elle est constituée des méthodes suivantes :

- Add compte
- Get compte
- Listes comptes
- Verser
- Retirer
- Virement

La classe banque service : elle contient les méthodes suivantes :

- Banque service
- Créez un compte
- Verser vers un compte
- Retirer de l'argent d'un compte
- Virement entre 2 comptes

La figure IV.1 décrit le diagramme des classes décrites ci-dessus :

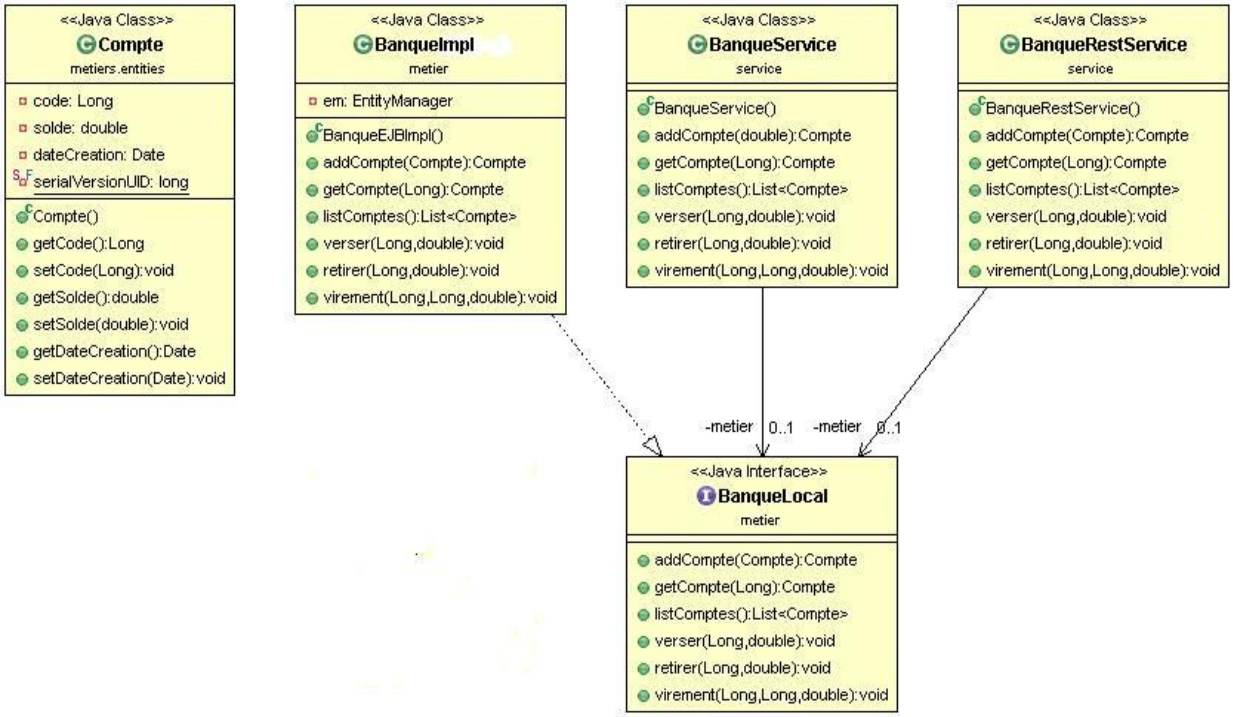


Figure IV.1 diagramme de classe « gestion d'une banque » avec UML.

- ## Diagramme de sequences :

  Les diagrammes de séquences permettent de décrire COMMENT les éléments du système interagissent entre eux et avec les acteurs :

  - Les objets au cœur d'un système interagissent en s'échangent des messages.

- Les acteurs interagissent avec le système au moyen d'IHM (Interfaces Homme-Machine).

Concernant les acteurs intervenants : Il existe deux acteurs utilisées qui sont :

1. **Le client**
2. **Admin du banque**

La figure IV.2 représente le diagramme de séquence :

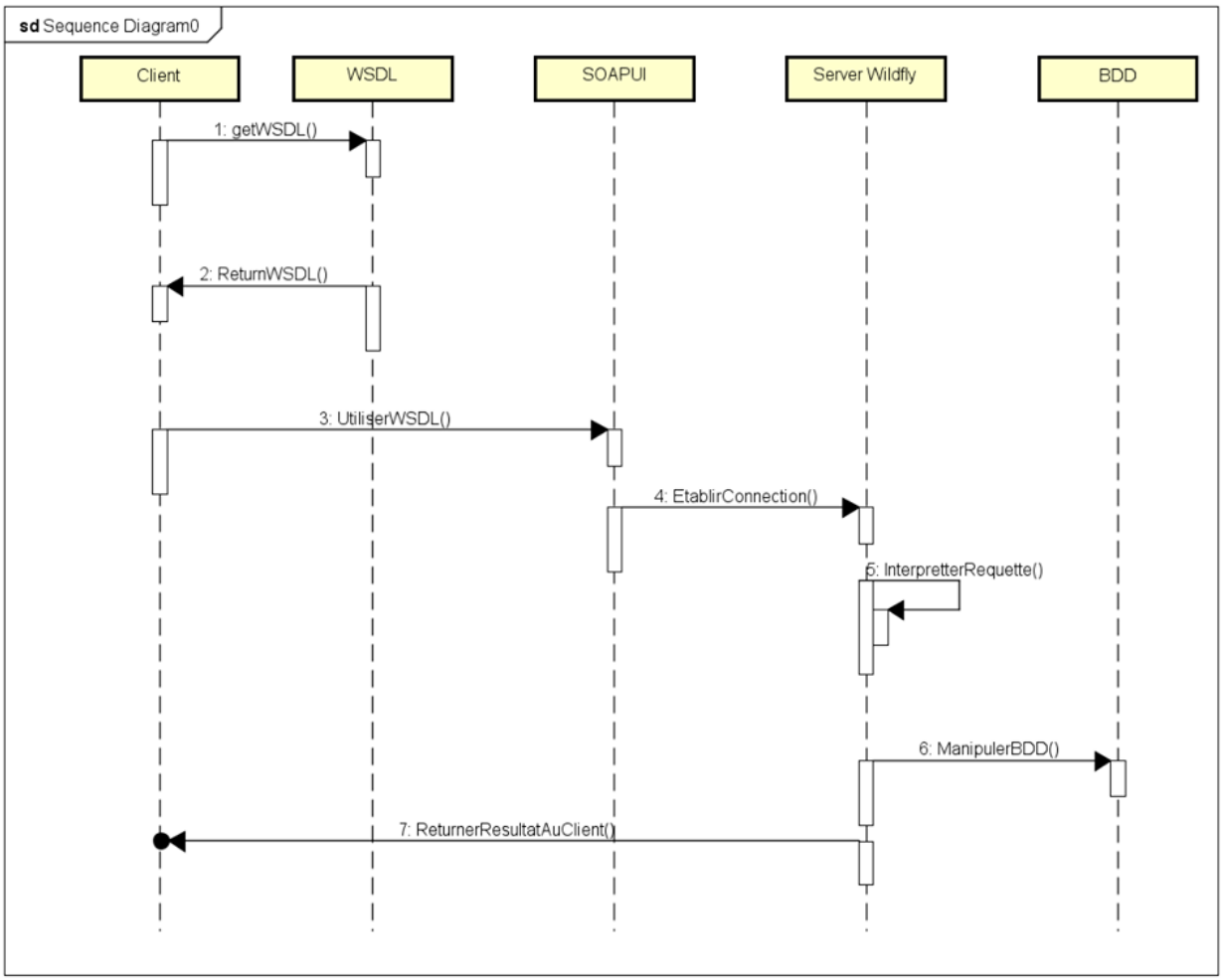


Figure IV.2 diagramme de séquence « gestion d'une banque » avec UML.

IV.3 Langages de programmation utilises

Pour l'élaboration de système conçu, Nous avons utilisé les langages de programmation suivants :

IV.3.1 Langage Java

Java est un langage de programmation et une plate-forme informatique qui ont été créés par Sun Microsystems en 1995. Beaucoup d'applications et de sites Web ne fonctionnent pas si Java n'est pas installé et leur nombre ne cesse de croître chaque jour [16].

- **Caractéristiques du langage Java**

Les caractéristiques du langage java sont :

a) Interprété : La source est compilé en pseudo code ou byte code puis exécuté par un interpréteur Java « Java Virtual Machine (JVM) ».

b) Portable : Il est indépendant de toute plate-forme. Le code reste indépendant de la machine sur laquelle il s'exécute. Cette indépendance est assurée au niveau du code source grâce à Unicode et au niveau du byte code.

c) Orienté objet : Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application. Java n'est pas complètement objet car il définit des types primitifs (entier, caractère, flottant, booléen,...).

d) Simple : Le choix de ses auteurs a été d'abandonner des éléments mal compris ou mal exploités des autres langages tels que la notion de pointeurs (pour éviter les incidents en manipulant directement la mémoire), l'héritage multiple et la surcharge des opérateurs.

e) Sûr : La sécurité fait partie intégrante du système d'exécution et du compilateur. Un programme Java planté ne menace pas le système d'exploitation.

f) Multitâche : Il permet l'utilisation de threads qui sont des unités d'exécution isolées. La JVM utilise plusieurs threads

IV.3.2 Langage XML

XML est un standard promulgué par le W3C, l'organisme chargé de standardiser les évolutions du Web. On retrouve dans XML une généralisation des idées contenues dans HTML et SGML1.

XML permet de définir des balises et de leur associer une interprétation. Dans HTML, on n'utilise les balises que pour décrire l'aspect graphique que doit revêtir la page dans le navigateur Web. Dans XML, les balises permettent d'associer toutes sortes d'informations au
fil du texte.

IV.3.3 Langage WADL

WADL est un format de document XML qui définit un vocabulaire pour décrire les applications web. WADL vise à être le WSDL des services web RESTful. Il permet aux fournisseurs de services afin de documenter les ressources offertes, les méthodes HTTP que les ressources, et les formats de représentation pris en charge. Les clients du service peuvent utiliser WADL tant pour la documentation, la configuration et la génération de code.

Lors de l'exécution, Jersey va générer automatiquement un document WADL pour une application JAX- RS. [17].

L'application WADL est disponible dans / root / application.wadl, où root est l'URI de base de l'application Web déployée. Cette WADL généré inclut toutes les ressources profondes et autant de métadonnées qui peuvent être extrait de la compilation.

IV.4 Les outils utilisés

1. Eclipse :

Eclipse est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement

sur Java.

L’environnement Eclipse a beaucoup d’avantages, on cite comme exemple : Plate-forme ouverte pour le développement d’applications et extensible grâce aux plug-ins

- Plusieurs versions d’un même plug-in peuvent cohabiter sur une même plate-forme
- Support multi-langages, multi-OS (Win, Linux, Mac)
- Très rapide à l’exécution
- Nombreuses fonctionnalités proposées par le JDT (Java Development Tool)
- Historique local des dernières modifications réalisées.

2. HeidiSQL :

HeidiSQL est un outil d'administration de base de données possédant un éditeur SQL et un constructeur de requête. Il a été développé et optimisé pour être utilisé avec le SGBD relationnel MySQL disponible commercialement ou gratuitement.

3. WildFly 8

WildFly, anciennement JBoss Application Server ou JBoss, est un serveur d'applications Java EE Libre écrit en Java, publié sous licence GNU LGPL. Étant écrit en Java, WildFly peut être utilisé sur tout système d'exploitation fournissant une machine virtuelle Java (JVM). Le nom JBoss est aujourd'hui utilisé pour JBoss EAP, produit dérivé WildFly et faisant l'objet d'un support commercial[1].

4. SoapUI

SoapUI est une application open source permettant le test de web service dans une architecture orientée services (SOA). Ses fonctionnalités incluent l'inspection des web service, l'invocation, le développement, la simulation, le mocking, les tests fonctionnels, les tests de charge et de conformité.

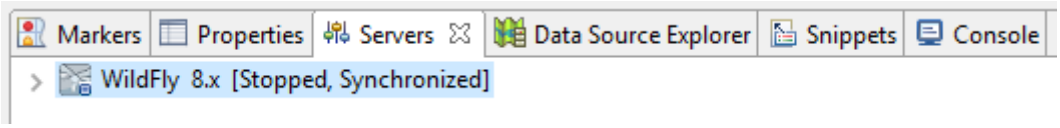
5. Navigateur web

Pour tester les services web on a besoin d'un navigateur Web. Notre choix s'est fait sur le navigateur **Mozilla Firefox**. Firefox est un navigateur Web libre et gratuit et il a connu un succès croissant depuis sa création. **Mozilla Firefox** a introduit progressivement des fonctionnalités qui l'ont rendu flexible et facile à utiliser.

IV.5 Configuration de server WildFly et bdd

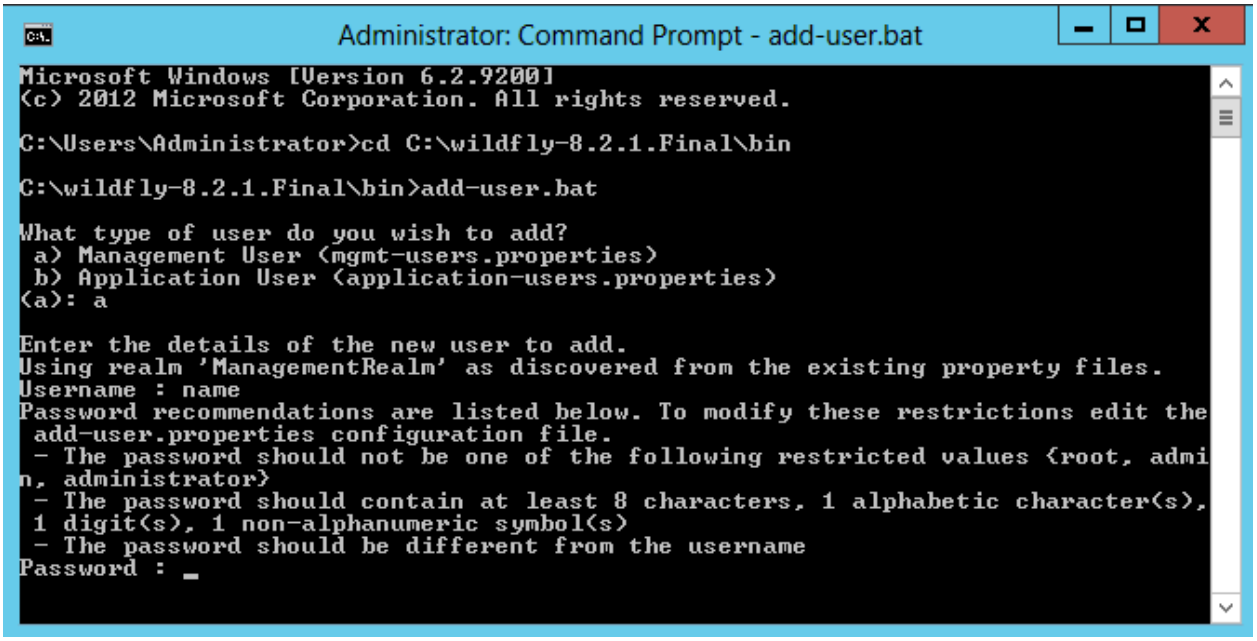
- Pour utiliser JAVAEE il faut avoir un server d'application javaee, Dans ce qui suit nous allons montrer les étapes de configuration de server wildfly.

1- Dans la zone server d'éclipse cliquer sur new choisit ensuite WildFly 8.1 cliquer sur next
2- Ensuite choisir l'emplacement de Wildfly
3- Si tous va bien tu auras le résultat suivant :

4- 

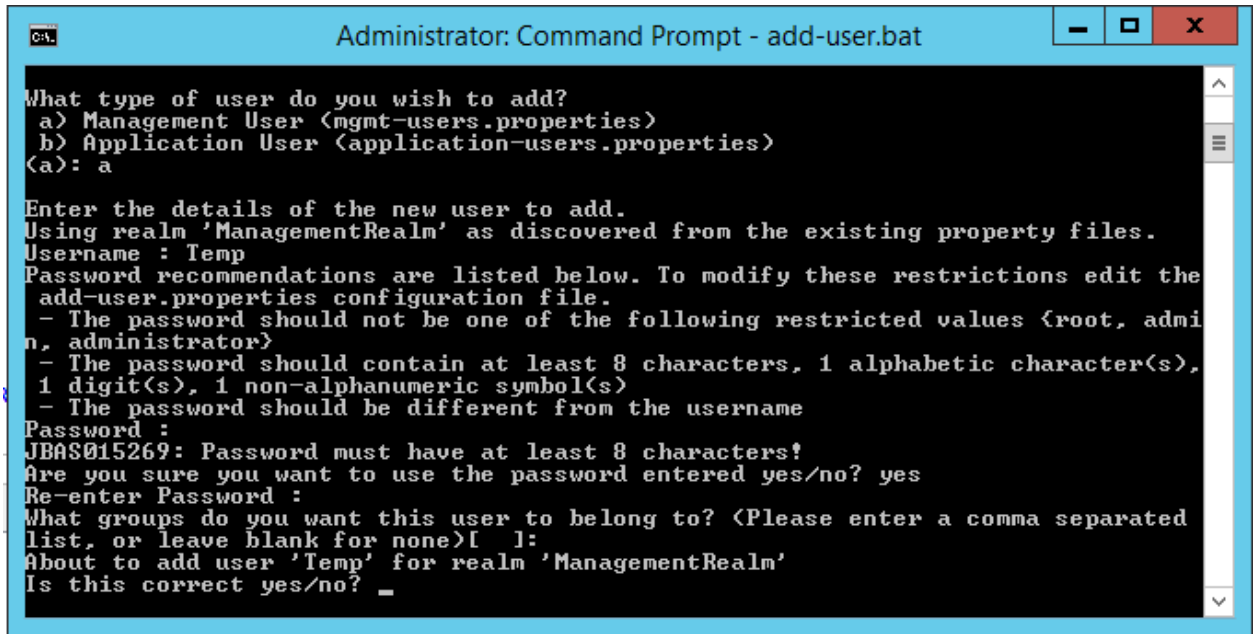
- Ensuite nous allons créer notre bdd avec le nom db_banque
- Après, il faut réaliser le server WildFly avec notre bdd pour qu'il puisse la manipuler.

1- Lancer le server wildfly dans éclipse
2- Accéder a l'URL http://localhost:8080
3- Pour utiliser wildfly, il faut ajouter un utilisateur, à travers cmd accéder a fichier de wildfly8 et taper la ligne de commande suivant add-user.bat



4- Ajouter ensuite les information de connexion

5-



6- Accede ensuite ..\wildfly-8.2.0.Final\modules\system\layers\base\com

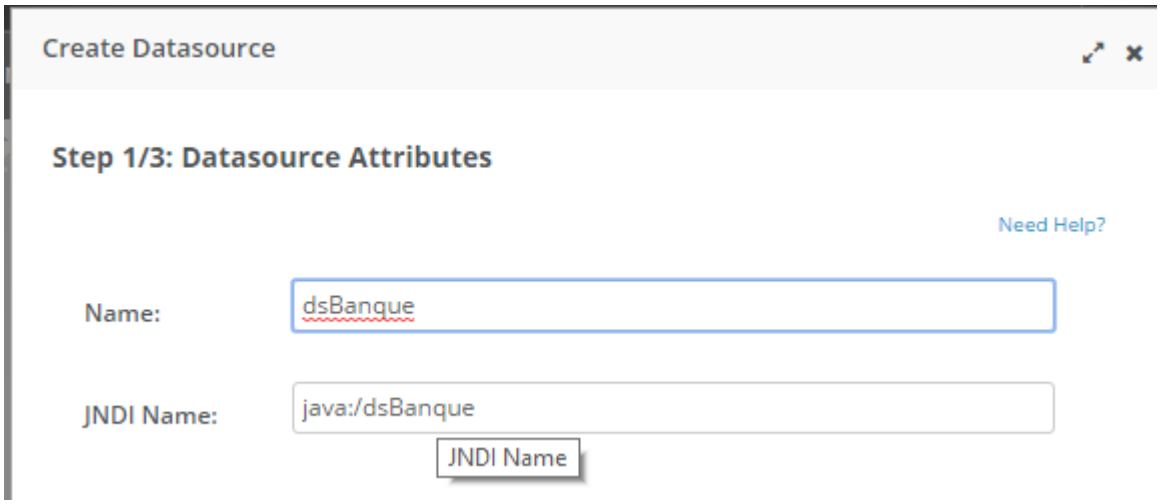
7- Creer un dossier mysql/driver

8- Ajouter mysql-connector-java-5.1.33.jar et le fichier module.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<module xmlns="urn:jboss:module:1.3" name="com.mysql.driver">
 <resources>
  <resource-root path="mysql-connector-java-5.1.33.jar"/>
 </resources>
 <dependencies>
  <module name="javax.api"/>
  <module name="javax.transaction.api"/>
 </dependencies>
</module>
```

9- Apres acceder a l'url http://localhost:9990/console/App.html

- Dans la zone configuration liquer sur add
- Saisir les informations suivantes :



Next mysql

- Ensuite saisir les informations suivantes ;

  Connexion URL: jdbc:mysql://localhost:3306/db_banque

  Saisir le nom utilisateur et le mot de passe

  Ensuite nous créons notre projet

  Apres le lancement de serveur

  Nous obtenons le fichier wsdl suivants

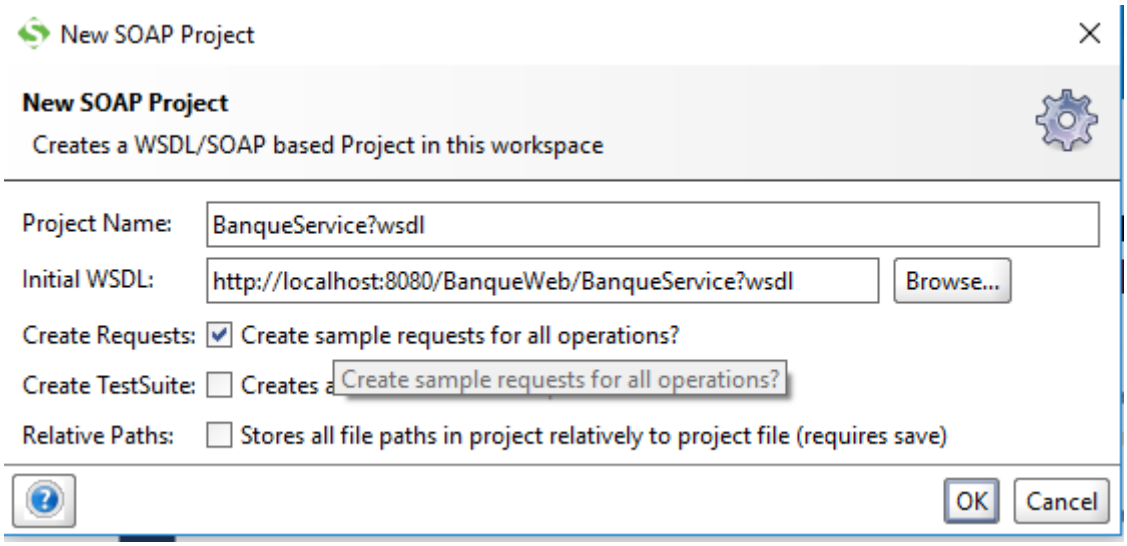
Fichier WSDL

```
targetNamespace="http://service/">
▼<wsdl:types>
  ▼<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://service/" elementFormDefault="unqualifi
   targetNamespace="http://service/" version="1.0">
     <xs:element name="addCompte" type="tns:addCompte"/>
     <xs:element name="addCompteResponse" type="tns:addCompteResponse"/>
     <xs:element name="getCompte" type="tns:getCompte"/>
     <xs:element name="getCompteResponse" type="tns:getCompteResponse"/>
     <xs:element name="listComptes" type="tns:listComptes"/>
     <xs:element name="listComptesResponse" type="tns:listComptesResponse"/>
     <xs:element name="retirer" type="tns:retirer"/>
     <xs:element name="retirerResponse" type="tns:retirerResponse"/>
     <xs:element name="verser" type="tns:verser"/>
     <xs:element name="verserResponse" type="tns:verserResponse"/>
     <xs:element name="virement" type="tns:virement"/>
     <xs:element name="virementResponse" type="tns:virementResponse"/>
   ▼<xs:complexType name="addCompte">
     ▼<xs:sequence>
        <xs:element name="solde" type="xs:double"/>
      </xs:sequence>
     </xs:complexType>
   ▼<xs:complexType name="addCompteResponse">
     ▼<xs:sequence>
        <xs:element minOccurs="0" name="return" type="tns:compte"/>
      </xs:sequence>
     </xs:complexType>
   ▼<xs:complexType name="compte">
     ▼<xs:sequence>
        <xs:element minOccurs="0" name="code" type="xs:long"/>
        <xs:element minOccurs="0" name="dateCreation" type="xs:dateTime"/>
        <xs:element name="solde" type="xs:double"/>
```

Pour utiliser notre service web nous allons utiliser SOAPUI

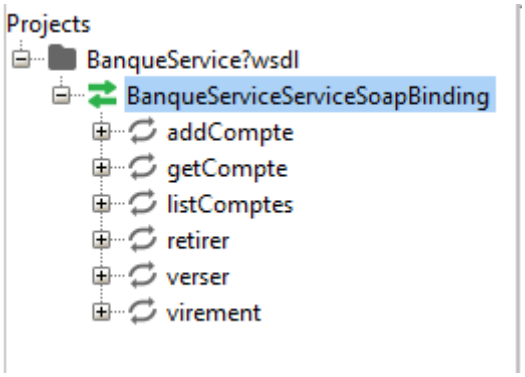
1- Service Web SOAP

- ✓ On accéde a SOAPUI en cliquant sur SOAP
- ✓ On saisit les information suivantes : projet name et intial WSDL .



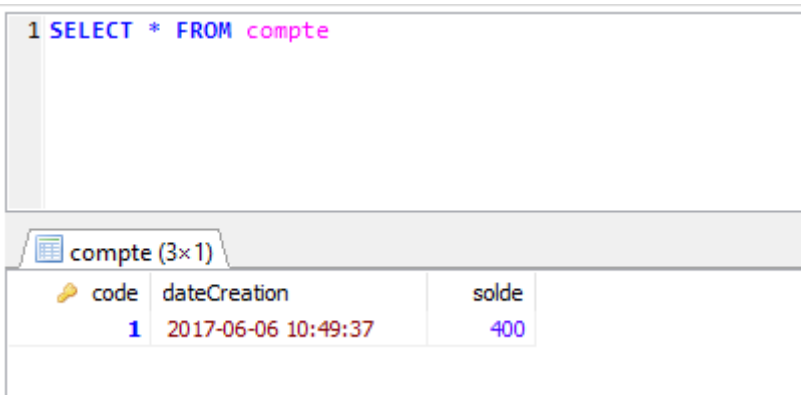
et on valide.

Nous obtenons les opérations de notre web service



- pour ajouter un compte nous cliquons sur la requête addcompte

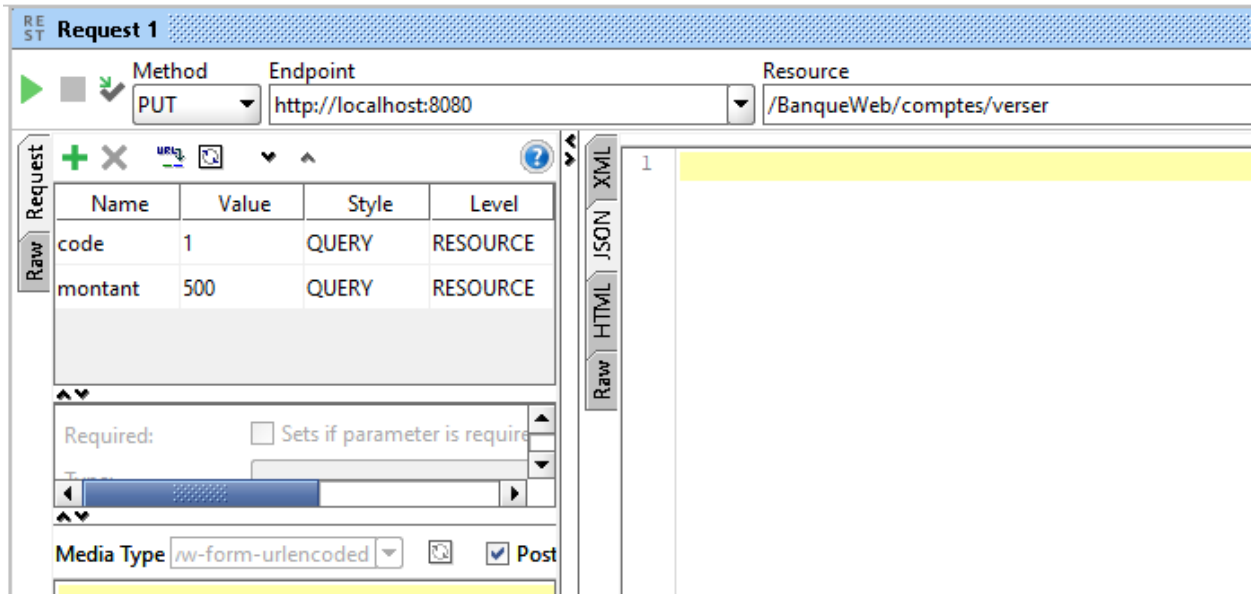
➢ Ainsi, le compte est créé.



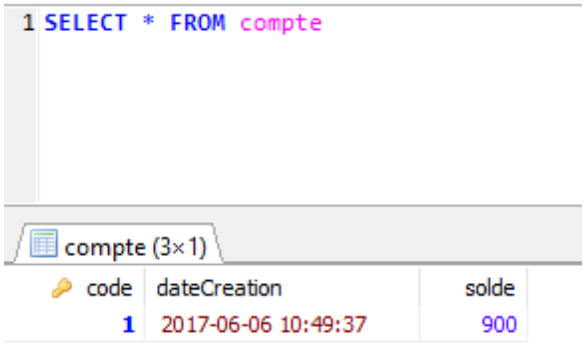
B- Le service REST

✓ Nous cliquons sur REST, et nous saisons l'URL suivant : http://localhost:8080/BanqueWeb/comptes/verser

✓ nous saisons les informations comme montre la figure suivant



- Enfin 500 da a été ajouter au compte



IV.5 Conclusion

Dans ce chapitre, on a décrit la mise en œuvre de notre modeleur. On a implémenté quelques fonctionnalités du modeleur proposé dans l'architecture et on a présenté les fonctions de notre application avec REST et SOAP pour la gestion d'une banque. En outre, nous avons présenté les langages et les outils utilisés pour la mise en œuvre d'un prototype du modeleur proposé.

Conclusion Générale

Notre projet s'inscrit dans le domaine de génie logiciel, plus particulièrement dans la technologie des services Web. Dans le premier chapitre, nous avons présenté la notion web des objets. Dans le deuxième, on a présenté les services web et ces architectures. En troisième chapitre, nous avons présentés en détail l'architecture REST vu son importance.

Finalement, nous avons présenté la conception et la réalisation de notre application. Le travail qui nous a été associé consiste à permettre la gestion d'une banque avec deux architecture SOAP et REST afin de soulever les points forts de REST.

On a rencontré beaucoup de difficulté lors de la réalisation du générateur, ceci est dû aux spécificités des langages de descriptions et du nombre des outils et du serveur à maitriser.

Comme perspective, on peut penser à la description et à la représentation sémantique d'une ressource physique dans le monde du web des objets et de gérer ses fonctionnalités par l'architecture REST.

Références Bibliographiques

[1] https://github.com/jgamblin/Mirai-Source-Code 04/04/2017

[2] Nicolas Seydoux, Mahdi Ben Alaya, Nathalie Hernandez, Thierry Monteil,Ollivier Haemmerle. Semantique et Internet des objets : d'un etat de l'art a une ontologie modulaire 2012

[3] D. Guinard, V. Trifa, F. Mattern and E. Wilde, E. "From the Internet of Things to the Web of Things: Resource oriented Architecture and Best Practices. In Architecting the Internet of Things," Springer, Berlin Heidelberg, 2011

[4] D. Guinard, "A Web of Things Application Architecture - Integrating the Real-World" 2011

[5] Sana Sellami.« les services Web ». 2016-207.

[6] Youssef BELAID. *« SERVICE WEB – SOAP »*.2012

[7] E. Stattner, " **Applications et Services WEB:** Architecture REST",2014-2015.

[8] Fierstone, «Les services Web Jeremy »,Email : fierston@essi.fr SAR5 – Novembre 2002

[9]. Walid FDHILA : « décentralisation Optimisées et synchronisation des Procèdes Métiers Inter-Organisationnels » thèse de doctorat, l'université Henri Poincaré Nancy1, 2011.

[10] Rémi Coletta "IUT de Montpellier - Programmation Une introduction aux Services Web" 2010

[11] Roy Thomas Fielding : « Architectural Styles and the Design of Network-based Software Architectures", these de doctorat, university of california, irvine, 2000.

[12]. Bill Burke: "RESTful Java with JAX-RS", O'Reilly, Copyright © 2010, USA.

[13]. Martin Kalin: "Java Web Services: Up and Running by Martin Kalin", O'Reilly,2009, USA

[14]. Leonard Richardson and Sam Ruby:"RESTful web services" O'Reilly Media, Copyright © 2007, United States of America.

[15] Amosse EDOUARD, Doctorant COMPRENDRE L'ARCHITECTURE DES WEB SERVICES REST

[16]. Oracle, Equipe Java ; Ressource de sécurité Java, Article du Web URL : http://www.java.com/fr/security/, Consulté le 01/05/2014.

[17]. Bill Burke: “RESTful Java with JAX-RS”, O'Reilly, Copyright © 2010, USA.

[18]. World Wide Web Consortium, http://www.w3.org/.

[19]. http://www.figer.com/publications/REST.htm#.WT8kZuvyh0w

[20]. Thai Tri, Hung, “Architecture Des Systèmes D‟information D‟entreprise Et Architecture Orientée Service”. institut de la Francophonie pour l‟informatique , 2005.

Liste des figures

ANNEXES

IoT : Internet of Things.

WoT : Web Of Things.

XML : Extensible Markup Language.

URI : Uniform Resource Locator.

RPC : remote procedure call.

DCOM : Distributed Component Object Model.

RMI : Remote method invocation.

REST : representational state transfer.

SOAP : Simple Object Protocol communication.

JSON : JavaScript Object Notation.

CRUD : (Create, Read, Update and Delete.

MIME : Multipurpose Internet Mail Extensions.

JAVAEE : Java Enterprise Edition.

EJB : Entreprise Java Bean.

HTTP: HyperText Transfer Protocol.

HTML : HyperText Markup Language.

WebDAV : Web-based Distributed Authoring and Versioning.