Table de matière

Table de matière

Liste des Figures

.

Liste des Figures

Liste des Figures

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.

le style d'architecture REST est aussi largement utilisé. Le protocole de communication SOAP est typiquement utilisé pour faire appel à des procédures situées sur d'autres ordinateurs distants.

Le style d'architecture REST est employé par les applications Web pour communiquer de l'information.

En se référant à la gestion des hôtels. Ce genres des travaux ne s'effectuent plus à la main, mais par les machines et des logiciels.

L'outil informatique est un élément qui lui facilite une bonne gestion pour une meilleure prise des décisions .

Notre application est une simple application web service. Le serveur est implémenté en utilisant Tomcat comme serveur web et base de donnée Postgres .

Le document est organisé comme suit :

Le chapitre 1 : décrit la définition d'un objet intelligent.

Le chapitre 2 : décrit les notions de web des objets.

Le chapitre3 : : décrit les différents protocoles services web.

Le chapitre4 : décrit les détails de service REST

Le chapitre 5 : présente les détails d'implémentation de notre application.

Et en fin une conclusion générale qui résume notre travail

Chapitre I

I Introduction

Les objets connectés existent depuis déjà plusieurs années, mais la conjonction de l'intégration technologique, de l'évolution des technologies de communication, des nouvelles capacités à traiter les données captées et l'apparition des nouveaux services qui en découlent, a accéléré leur développement pour les rendre indispensables.

Aujourd'hui cette technologie est en train de devenir une réalité qui va drainer le commerce et l'industrie du futur. Et cela s'accélère avec pour conséquence que les actions qui sont, ou vont être, entreprises aujourd'hui vont avoir des impacts rapides et importants sur l'ensemble des entreprises. Tous les secteurs sont ou seront concernées et leur écosystème en sera aussi affecté.

Cette accélération du marché des objets connectés met au jour diverses complexités qui n'étaient pas apparues lorsque l'écosystème était encore embryonnaire. Le CIGREF a souhaité aujourd'hui s'intéresser à la compréhension de ces objets et s'est posé la question de définir les sujets à explorer lorsque l'on souhaite mettre en œuvre des objets connectés.[1]

II Un objet connecté

II.1 Définition

Un objet connecté apporte un service à valeur ajoutée allant au-delà de sa fonction première.

Grâce à un ou plusieurs capteurs, il peut produire des données ou peut apporter un service en utilisant des données extérieures. Une fois analysées, ces données produites ou captées délivrent des informations utiles :

• Détection de présence/mouvement
• Pression
• Humidité, lumière
• Température
• Position

II.2 Le fonctionnement d'un objet connecté

Un objet connecté est équipé d'un système permettant de transmettre et recevoir des informations. Il peut ainsi envoyer des données issues de ses capteurs ou interagir avec l'utilisateur.

Un objet connecté est un équipement électronique doté de capacités très hétérogènes en communication, traitement, mémoire, énergie et collecte de données ambiantes. Dans ce nuage d'objets qui nous entoure, on distingue un objet très personnel - le Smartphone - du fait qu'il est intimement associé à un individu et que chaque activité du Smartphone peut être rapporté à un individu ; en effet, le Smartphone le suit dans ses déplacements, est toujours à portée de main, sait se rendre indispensable par la qualité des applications qu'il propose, et enfin dispose de capacités importantes comparativement à d'autres objets connectés. Par ailleurs, le Smartphone permet souvent d'interagir avec d'autres objets en champ proches, comme la montre connectée, ou distants (la maison connectée comprenant l'alarme, les volets roulants…), l'interaction étant directe ou via le Cloud. Enfin, on identifie les objets connectés autonomes, comme la voiture connectée (Tesla, Google car) qui peut conduire toute seule.

III Un objet intelligent

Un objet connecté peut prendre une décision (piloter) ou alerter l'utilisateur en fonction des données captées. Certains sont capables de prétraiter des informations et de transmettre uniquement les plus pertinentes.

III.1 Définition

les objets intelligents comme des objets du monde réel doués d'une capacité de communication. Ils sont également appelés « objets connectés », « objets communicants » ou « objets acteurs ».

III.2 Caractéristiques

Les objets intelligents sont généralement limités en puissance (CPU, RAM, flash, énergie). Typiquement, sur 1 cm^2, on trouvera un microprocesseur, très peu de RAM(quelques milliers d'octets), un peu de mémoire flash (quelques douzaines de kibioctets), des interfaces et un module radiofréquence ou CPL. Lorsque ces objets disposent de piles et sont connectés en sans-fil, le point critique est leur consommation en énergie. Les flux de données sont souvent extrêmement limités (quelques paquets par minute, voire par mois), mais chaque bit transmis a un coût énergétique, et l'objet intelligent doit rester autonome (sans remplacement de pile) pendant 5 à 10 ans. C'est face à ce problème d'économies d'énergie que la notion d'objet « intelligent » et les techniques de gestion de l'énergie deviennent primordiales. Ceci ne

s'applique pas, bien entendu, aux objets raccordés au courant électrique alternatif comme ceux connectés via CPL (ceux-ci peuvent cependant avoir des contraintes de mémoire, CPU, etc.).

Parmi les domaines d'application de ces objets, on peut citer : les réseaux de capteurs déployés dans nos villes modernes, les rendant plus intelligentes et adaptatives. Il y a également la domotique qui permet à nos nouveaux téléviseurs, radio-réveils, réfrigérateurs ou cadres photos de nous rendre la vie plus facile et d'optimiser notre communication ou notre consommation d'énergie. De façon similaire, l'industrie bénéficie de robots et de machines de plus en plus intelligentes et les biens de consommation sont équipés d'étiquettes électroniques ou de codes-barres liés à des sources d'informations virtuelles permettant de nouveaux cas d'utilisation.

IV Le but du web des objets

Le but du Web des objets étant d'amener le maximum d'objets intelligents au plus proche du Web, certaines caractéristiques des réseaux de capteurs sans-fil les rendent possiblement intéressant : leur capacité à être fiable, hétérogènes, évolutifs, et robuste. Pour atteindre cet objectif, il faut une conception rigoureuse. Particulièrement dans un contexte ou les ressources des objets peuvent être limitées et où la topologie des réseaux est évolutive.

Pour répondre aux services de haute disponibilité mentionnés ci-dessus, sans affecter l'hétérogénéité des architectures de réseaux de capteurs sans fil existant, une couche middleware est nécessaire.

Le middleware peut être considéré comme une infrastructure logicielle qui fait la liaison entre le réseau de capteur, le système d'exploitation, la pile réseau, et les applications. Une solution middleware complète devrait contenir un environnement d'exécution qui supporte et coordonne des applications multiples et des services systèmes standardisés. Ex : l'agrégation de données ou la gestion et la manipulation des règles des applications cibles. L'architecture logicielle du middleware devra également fournir un mécanisme efficace et adapté à l'utilisation des ressources systèmes. Des ressources systèmes bien maitrisées permettent une dépense d'énergie adaptée et de ce fait, cela participe à prolonger la vie du réseau de capteur.

V Conclusion

Les objets connectés sont en train de devenir un vecteur de développement du *business* des entreprises. Et nombre d’entre elles ont déjà pris le virage des objets connectés. Dans certains secteurs économiques ce sont des millions d’objets qui seront très rapidement déployés.

Des objets intelligents et communicants rendent certains scénarios possibles :**bâtiments et maisons intelligentes, voitures autonomes et connectées….** des nouvelles interfaces entre l’homme et la machine. Des nouvelles questions de respect de la vie privée et de sécurité se posent, surtout face aux défis que notre société actuelle connaît : réchauffement climatique, pollution liée à notre mode de consommation alimentaire et au type d’agriculture pratiquée…

Chapitre II

I. Introduction

Le **Web des objets** désigne l'intégration de tout appareil interrogeable ou contrôlable à distance, dans le monde du World Wide Web.

La création de réseaux d'objets intelligents à grande échelle, provenant du monde réel (dotés par exemple de puces RFID, faisant partie de réseaux de capteurs sans-fil, ou encore les systèmes embarqués) est devenu le but de nombreuses activités de recherches récentes et variées. Bien plus qu'une représentation de données et fonctionnalités par des concepts de systèmes verticaux, les objets intelligents font partie intégrante du Web.

Dans le Web des objets, les technologies populaires du Web (HTML, JavaScript, Ajax…) peuvent être utilisées pour développer des applications qui font appel à des objets intelligents. Les utilisateurs peuvent se servir des mécanismes Web bien connus (la navigation, la recherche, l'étiquetage, la mise en cache, les liaisons) pour interagir avec eux.

De multiples prototypes utilisent ces principes, dans un environnement de capteurs, de systèmes de supervision d'énergie et d'objets RFID sur le Web. Des applications ad-hoc telles que les *Applications Composites Physiques* émergent de plus en plus de par la facilité de développement et d'intégration (faible taille) .

II. Définition le Web des objets

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. Il s'agit d'une amélioration de l'Internet des objets qui intègre des objets intelligents, non seulement dans l'Internet, mais aussi sur le Web (c'est-à-dire au niveau de la couche application).

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.

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. [3]

III. Les types du web

Depuis les années 1990, divers outils et techniques ont transformé le Web, Vlad Trifa a présenté cinq piliers qui forment la base du Web moderne :

- Le Web social
- Le Web physique
- Le Web sémantique
- Le Web temps réel
- Le Web programmable.

Le Web des objets [2] est représenté par l'intersection de ces cinq tendances, et il est proposé comme une évolution du Web qui arrive dans le monde physique.

1. Le **Web social**

Il fait référence à une vision d'Internet considéré comme un espace de socialisation, un lieu dont l'une de ses fonctions principales est de faire interagir les utilisateurs entre eux afin d'assurer une production continuelle de contenu, et non plus uniquement la distribution de documents.

Il est considéré comme un aspect très important du Web 2.0. En particulier, il est associé à différents systèmes sociaux tels que le réseautage social, les blogs ou les wikis.

2. Web physique

En dehors des serveurs et des navigateurs, divers objets sont connectés au Web, tels que des entrepôts (chaque section contient des balises RFID permettant le suivi d'objets). Grâce à la prolifération (1,1 milliard en 2012) des smartphones avec capteur GPS et connexion à Internet, les applications de géolocalisation sont florissantes et des services permettent de créer un Web physique. Cela étend la portée des informations sur le Web de l'état des objets physiques du monde réel.

3. Le Web sémantique, ou toile sémantique

il est une extension du Web standardisée par le World Wide Web Consortium (W3C). Ces standards encouragent l'utilisation de formats de données et de protocoles d'échange normés sur le Web, avec comme format de base le Resource Description Framework (RDF).

Selon le W3C, « le Web sémantique fournit un modèle qui permet aux données d'être partagées et réutilisées entre plusieurs applications, entreprises et groupes d'utilisateurs ». L'expression a été inventée par Tim Berners-Lee (inventeur du Web et directeur du W3C), qui supervise le développement des technologies communes du Web sémantique. Il le définit comme « une toile de données qui peuvent être traitées directement et indirectement par des machines pour aider leurs utilisateurs à créer de nouvelles connaissances ». Pour y parvenir, le Web sémantique met en œuvre le Web des données qui consiste à lier et structurer l'information sur Internet pour accéder simplement à la connaissance qu'elle contient déjà.

4. Real-Time Web

Des finances aux journaux en passant par les réseaux sociaux, de nombreux domaines reposent sur des informations en temps réel livrées en temps opportun. Les services en ligne ont de plus en plus besoin de grandes quantités d'informations pour être livrés de manière sûre aussi vite que possible.

Divers outils et techniques ont fait leur apparition pour transmettre des informations en temps réel sur le Web.

Un moteur de recherche pouvant agréger toutes ces données rapidement a été créé pour le Web des objets.

5. Web programmable

Une des principales caractéristiques du Web 2.0 est la possibilité d'accéder à des données brutes à partir d'applications Web via une API Web. Le Web des objets correspond au Web 2.0 auquel on ajoute l'interaction avec les objets physiques. De nombreuses entreprises ont pris conscience des avantages à rendre leurs données et leurs services accessibles par voie de programme et pas seulement dans la fourniture de produits finis. En 2011, des milliers de sites Web mettent à disposition leurs données par le biais d'API ouvertes.

IV. Conclusion

Le Web des objets est un protocole d'application de haut niveau conçu pour maximiser l'interopérabilité dans l'IdO, et nous espérons que cette courte introduction vous a donné un avant-goût de son potentiel. Les technologies Web sont très populaires et offrent toute la souplesse et les fonctionnalités nécessaires à la majorité des futures applications IdO, y compris la découverte, la sécurité et la messagerie en temps réel.

Chapitre III

I Introduction

Les **services web** (en anglais web services) représentent un mécanisme de communication entre applications distantes à travers le réseau internet indépendant de tout langage de programmation et de toute plate-forme d'exécution :utilisant le protocole HTTP comme moyen de transport. Ainsi, les communications s'effectuent sur un support universel, maîtrisé et généralement non filtré par les pare-feux ;employant une syntaxe basée sur la notation XML pour décrire les appels de fonctions distantes et les données échangées ;organisant les mécanismes d'appel et de réponse.

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

Les services web facilitent non seulement les échanges entre les applications de l'entreprise mais surtout permettent une ouverture vers les autres entreprises. Les premiers fournisseurs de services web sont ainsi les fournisseurs de services en ligne (météo, bourse, planification d'itinéraire, pages unes, etc.), mettant à disposition des développeurs des API (Application Programmable Interface) payantes ou non, permettant d'intégrer leur service au sein d'applications tierces.

II Définition de service Web

C'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.

Cela nécessite :

- un encodage (toujours XML) ;
- un transport (souvent HTTP) ;
- une organisation des requêtes et des réponses.

La procédure de fonctionnement d'un service Web est la suivante :

1. le service Web définit un format pour les requêtes et les réponses ;
2. un ordinateur demandeur effectue une requête ;
3. le service Web effectue une action, et renvoie la réponse à l'ordinateur demandeur.

Un service Web peut par exemple :

- faire une demande automatiquement mise à jour d'un prix ;
- accéder à un calendrier universel faisant les conversions entre calendriers internationaux et connaissant, pour chaque pays, les dates des jours fériés ;
- traduire un passage
- valider un numéro international de code postal...

Les possibilités sont donc nombreuses.

Pour pouvoir utiliser un service Web, plusieurs étapes sont nécessaires :

- il faut savoir le trouver...
- ... puis connaître la méthode pour y accéder...
- ... enfin savoir l'utiliser correctement.

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. [4]

III La présentation des services web

Les services web sont des composants distribués qui offrent des fonctionnalités aux applications au travers du réseau en utilisant des standards ouverts. Ils peuvent donc être utilisés par des applications écrites dans différents langages et exécutées dans différentes plate-formes sur différents systèmes.

Les services Web utilisent une architecture distribuée composée de plusieurs ordinateurs et/ou systèmes différents qui communiquent sur le réseau. Ils mettent en oeuvre un ensemble de normes et standards ouverts qui permettent aux développeurs d'implémenter des applications distribuées internes ou externes en utilisant des outils différents fournis par les fournisseurs.

Un service web permet généralement de proposer une ou plusieurs fonctionnalités métiers qui seront invoquées par un ou plusieurs consommateurs.

Il existe deux grandes familles de services web :

- les services web de type SOAP
- les services web de type REST

IV L'intérêt d'un Service Web

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde. ;)

Les services Web sont **normalisés** car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes. C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des services Web puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les services Web n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les services Web ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plate-forme CORBA qui définit des ponts spécifiques entre le langage de définition IDl et différents langages de programmation. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents .
Les services Web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants. [5]

V Les caractéristiques d'un service Web

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

Un service Web possède les caractéristiques suivantes :

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

VI Architecture d'un service web

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

1. REST

REST (Representational State Transfer) est une architecture de services Web. Élaborée en l'an 2000 par Roy Fiedling, l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres travaux fondamentaux, REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

1. Principe

Le principe de REST est d'utiliser HTTP pour l'implémentation d'un Web Service, non plus comme simple protocole de transport, mais également pour définir l'API (Application Programming Interface) de chaque service c'est à dire la définition même des messages entre clients et serveur.

Paradoxalement, c'est donc un retour aux sources vu que la spécification de HTTP 1.1, la dernière version en date, est légèrement antérieure (1997) aux premiers Web Services basés sur les RPC (1998).

2. Avantages de REST

Les avantages sont les suivants

•Utilisation d'HTTP comme protocole applicatif et non protocole de transport;

•cache réseau → en respectant les entêtes et les requêtes préconisés dans la norme HTTP, on permet ainsi l'utilisation efficace de serveurs caches entre le serveur et les clients de l'application;

•interface uniforme → chaque Web Service REST a une interface orientée autour des messages de HTTP.

2. XML-RPC

XML-RPC est un protocole simple utilisant XML pour effectuer des messages RPC. Les requêtes sont écrites en XML et envoyées via HTTP POST. Les requêtes sont intégrées dans le corps de la réponse HTTP. XML-RPC est indépendant de la plate-forme, ce qui lui permet de communiquer avec diverses applications. Par exemple, un client Java peut parler de XML-RPC à un PerlServer !

1. Principe

XML-RPC est le plus simple des formats d'échange. Le principe de base est le suivant :

- sur le poste client, une bibliothèque encode les paramètres de la requête en XML ;
- sur le poste serveur, une (autre) bibliothèque les décode et les transmet à l'application.

La procédure inverse a lieu lors de l'envoi de la réponse à la requête vers le poste client. En définitive, le programmeur n'a jamais besoin de coder lui-même le format de sortie en XML, puisque, dans le cadre du langage de programmation qu'il utilise, des fonctions se chargent des opérations à sa place. Il n'en voit que le résultat final.

Le transfert des données se fait selon le protocole HTTP.

Il existe des bibliothèques en Perl, C, Python, Java, VB/.Net, PHP...

3. SOAP

SOAP [6] est un protocole adopté par le Consortium W3C. Le Consortium W3C crée des standards pour le Web : son but est donc de créer des standards pour favoriser l'échange d'information.

SOAP veut dire Simple Object Access Protocol. Si l'on voulait traduire cette définition en français cela donnerait Protocole Simple d'Accès aux Objets. En effet, le protocole SOAP consiste à faire circuler du XML généralement, via du http sur le port 80 (ou en utilisant un autre protocole). Cela facilite grandement les communications, car le XML est un langage standard et le port utilisé est le port 80, qui ne pose donc pas de problème pour les firewalls(pare-feu) de l'entreprise, contrairement à d'autres protocoles.

1. Ses caractéristiques

- SOAP permet une normalisation des échanges de données. Les données sont encodées en XML et échangées par des appels de procédures à distance [7] en utilisant HTTP/SMPT/POP comme protocole de communication.
- Simple, extensible et permet le diagnostic des erreurs.
- Message unidirectionnel (Requête ->Réponse).
- Fonctionne de manière synchrone et asynchrone.
- Indépendant de la plate-forme et du langage.
- N'est pas perturbé par un pare-feu.

2. Principes de SOAP

SOAP codifie simplement une pratique existante, à savoir l'utilisation conjointe de XML et http. SOAP est un protocole minimal pour appeler des méthodes sur des serveurs, services,composants, objets. Il a pour avantages de ne pas imposer une API ou un runtime,ni l'utilisation d'un ORB (CORBA, DCOM, ...) ou d'un serveur web particulier(Apache, IIS, ...) et non plus de ne pas imposer un modèle de programmation.

Une des volontés du W3C vis à vis de SOAP est de "ne pas réinventer une nouvelle technologie". SOAP a été construit pour pouvoir être aisément porté sur toutes les plates-formes et les technologies .

3. Structure d'un message SOAP

Un message SOAP est composé de deux parties obligatoires : l'enveloppe SOAP et le corps SOAP ; et une partie optionnelle : l'en-tête SOAP.

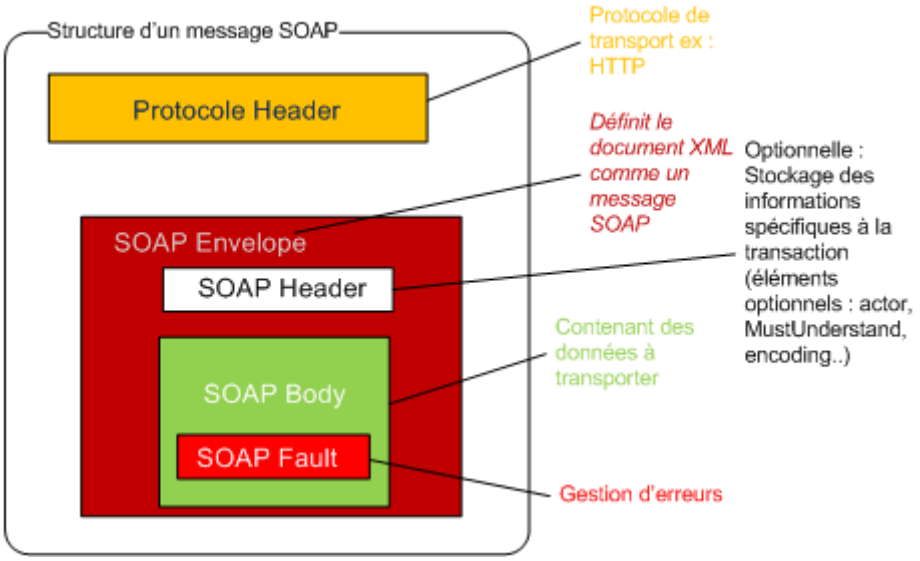

Figure III.1 : Structure d'un message SOAP **[7]**

- **SOAP envelope** (enveloppe) est l'élément de base du message SOAP. L'enveloppe contient la spécification des espaces de désignation (namespace) et du codage de données.
- **SOAP header** (entête) est une partie facultative qui permet d'ajouter des fonctionnalités à un message SOAP de manière décentralisée sans agrément entre les parties qui communiquent. C'est ici qu'il est indiqué si le message est mandataire ou optionnel. L'entête est utile surtout, quand le message doit être traité par plusieurs intermédiaires.
- **SOAP body** (corps) est un container pour les informations mandataires à l'intention du récepteur du message, il contient les méthodes et les paramètres qui seront exécutés par le destinataire final.
- **SOAP fault** (erreur) est un élément facultatif défini dans le corps SOAP et qui est utilisé pour reporter les erreurs. [8]

a) **L'enveloppe SOAP**

l'enveloppe contient un seul message constitué d'un entête facultatif (SOAP header) et d'un **corps obligatoire** (SOAP body).

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

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"

    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">



        <soap:Header>

<!-- en-tête -->

        </soap:Header>



        <soap:Body>

<!-- corps -->

        </soap:Body>

</soap:Envelope>
```

b) Le corps SOAP

Le corps SOAP est un élément obligatoire dans le message SOAP. Il contient l'information destinée au receveur.

c) L'en-tête SOAP

L'en-tête SOAP est un élément facultatif dans un message SOAP.

Le format de l'en-tête n'est pas défini dans le cahier des charges et par conséquent, il est à la disposition des clients et des services pour leur propre usage.

L'en-tête d'un message SOAP commence avec la balise <soap:Header> et se termine avec la balise </soap:Header>, je vous rappelle qu'on peut aussi faire <sopa-env:Header></sopa-env:Header>.

4. 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 III. 2 donne une vue global sur SOAP RPC :

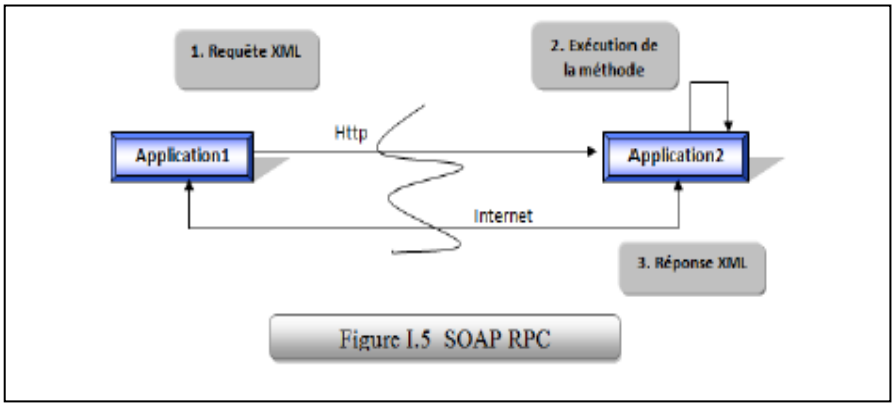

Figure III.2 : SOAP RPC **[9]**

A ses débuts, SOAP était destiné à être un protocole fournissant un mécanisme de RPC, inter-opérable, car utilisant XML & HTTP (il est maintenant –entre autre–un protocole de communication entre services web par échanges de messages XML).

Lors d'un appel RPC, un message SOAP doit contenir:

•l'adresse du destinataire du message ;

•le nom de la méthode à exécuter ;

•les paramètres à passer à la méthode.

En devenant un RPC, les services web en SOAP peuvent être vus comme un point d'entrée dans les applications « lourdes » : par exemple, un service web peut permettre une connexion entre un client sur Internet et une application à base d'EJB.

De nombreuses API (Application Programmer Interface) permettent de créer des Bstubs de méthodes exposées dans des services web.

5. La gestion des erreurs

l'approche SOAP permet de bon usage de la balise <soap:fault>

Cette balise est utilisée pour communiquer un problème qui a eu lieu dans la tentative de réalisation de la demande adressée au service Web. L'élément d'erreur est facultatif et figure uniquement dans les messages de réponse. La balise <soap:fault> peut contenir quatre autres balises facultatives :

- **faultcode** : cet élément est requis par le cahier des charges. Il contient un code indiquant la nature du problème.
- **faultstring** : est la version lisible par l'homme de la balise faultcode. C'est la traduction en langage naturel du code d'erreur.
- **faultactor** : indique le service qui a généré l'erreur. Cela est important lorsqu'une chaîne de services a été utilisée pour traiter la demande.
- **detail** : cet élément doit contenir autant d'informations que possible sur l'état du serveur à l'instant de l'apparition de l'erreur. Il contient souvent des valeurs de variables au moment de l'échec.

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 « plus » , 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>
```

Figure III. 3 : la réponse SOAP

6. Avantages et Inconvénients

- SOAP permet grâce au descripteur d'identifier la liste des services et objets exposés, alors que REST est sans état, ce qui lui permet d'avoir une plus grande indépendance entre le client et le serveur.
- SOAP s'adapte à différents protocoles de transport en évitant les problèmes de communication alors que REST n'utilise que le protocole HTTP.
- REST utilise l'URI comme représentation de ses ressources avec sa flexibilité d'évolution alors qu'on voit un fort couplage client / Serveur côté SOAP.

4. WSDL

1. définition

WSDL (Web Services Description Language) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées. [10]

2. 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.
- Les outils de développement, s'appuyant sur ces standards, permettent la création automatique de programmes utilisant les services Web existants.

3. Inconvénients

- Les normes de services Web dans certains domaines sont actuellement récentes.
- Les services Web souffrent de performances faibles comparée à d'autres approches de l'informatique répartie telles que le RMI, CORBA, ou DCOM.

5. UDDI

UDDI (Universal Description, Discovery and Integration) est un annuaire de services. Il fournit l'infrastructure de base pour la publication et la découverte des services Web. UDDI permet aux fournisseurs de présenter leurs services Web aux clients.

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

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

1. Structure UDDI

La structure des données UDDI est exprimée en utilisant les schémas XML. Ils sont divisés en 4 types de structures de données :

- Business Entity : C'est l'équivalent des pages blanches de l'annuaire. Cette structure inclut donc les informations concernant l'entreprise ou l'entité qui publie son Web service. Cette structure peut inclure un BusinessKey (clé d'affaire)qui peut prendre la forme d'une clé UUID (Universally unique identifiers) assurant la protection de l'information contenu dans le Web service.
- BusinessService : C'est l'équivalent des pages jaunes de l'annuaire. On y retrouve lesinformations concernant le nom et la description du Web service offert.
- BindingTemplate : C'est l'équivalent des pages vertes citées plus haut. Il contient les informations concernant les points d'accès au Web service et les aspects techniques permettant la liaison entre le Web service et l'API du requérant de celui-ci. Il fait référence à un ou plusieurs tModel.
- tModel : Il s'agit du mécanisme permettant l'échange de métadonnée à propos du Web service. Il peut s'agir d'un pointeur à un fichier WSDL, mais il peut aussi pointer sur d'autres types de fichiers comme sur des fichiers ebXML ou Rosetanet par exemple. Mais chaque tModel ne définira qu'un type d'interface avec laquelle elle peut interagir. Cependant un fichier UDDI peutcontenir plusieurs tModel.

2. Avantages et inconvénients:

Les services web ont des avantages et inconvénients

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. [11]

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 . [12]

VII Conclusion

Il est nécessaire de faire le point sur la technologie des services Web. Les services Web est un terme qui décrit un ensemble de protocoles standards utilisés pour établir un domaine d'intégration des applications.

L'un des facteurs ayant contribué au succès des services Web est sans doute l'utilisation des standards Internet tels que XML et HTTP. En conséquence, tout système capable d'analyser du texte et de communiquer via un protocole de transport Internet standard peut communiquer avec un service Web. Le principe de REST est d'utiliser HTTP pour l'implémentation d'un Web Service, non plus comme simple protocole de transport, mais également pour définir l'API (Application Programming Interface) de chaque service c'est à dire la définition même des messages entre clients et serveur.

Chapitre IV

I. Introduction

REST (Representational State Transfer) est l'un de ces acronymes qui représente une non technologie comme peuvent l'être Ajax, DHTML, Web 2.0 et autres.

REST est un style d'architecture qui repose sur le protocole HTTP : On accède à une ressource (par son URI unique) pour procéder à diverses opérations (GET lecture / POST écriture / PUT modification / DELETE suppression), opérations supportées nativement par HTTP.

II. Architecture REST

REST (REpresentational State Transfer) est un style d'architecture crée par Roy. T.Fielding en 2000 dans sa thèse de doctorat "Architectural Styles and the Design of Network-based Software Architectures .

1. Définition de REST

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» .
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. »
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, l'audio et autre média sont stockés sur un réseau et reliés entre eux par des hyperliens.

2. Contrainte d'architecture REST

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

2.1. Le 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.
Conception et réalisation d'un modeleur de composition de service web SOAP et REST

2.2. le modele client-serveur

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

2.3. Le 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 **IV.1** représente un schéma du modèle sans état .

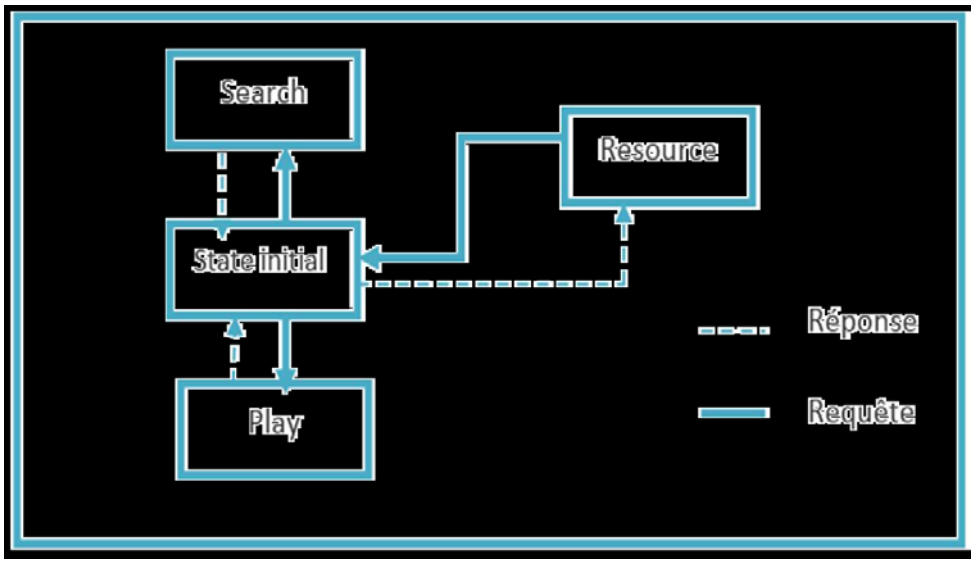

Figure IV.1. Schéma représentant un modèle sans état

2.4.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.

2.5. 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 oeuvre sera être découplées des services qu'elles fournissent. Mais elle n'est pas optimale pour d'autre forme d'interaction architecturale.

2.6.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.

Conception et réalisation d'un modeleur de composition de service web SOAP et REST

2.7. 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. [13]

3. Principes de REST

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

3.1. 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.

3.2. 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 **IV.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.

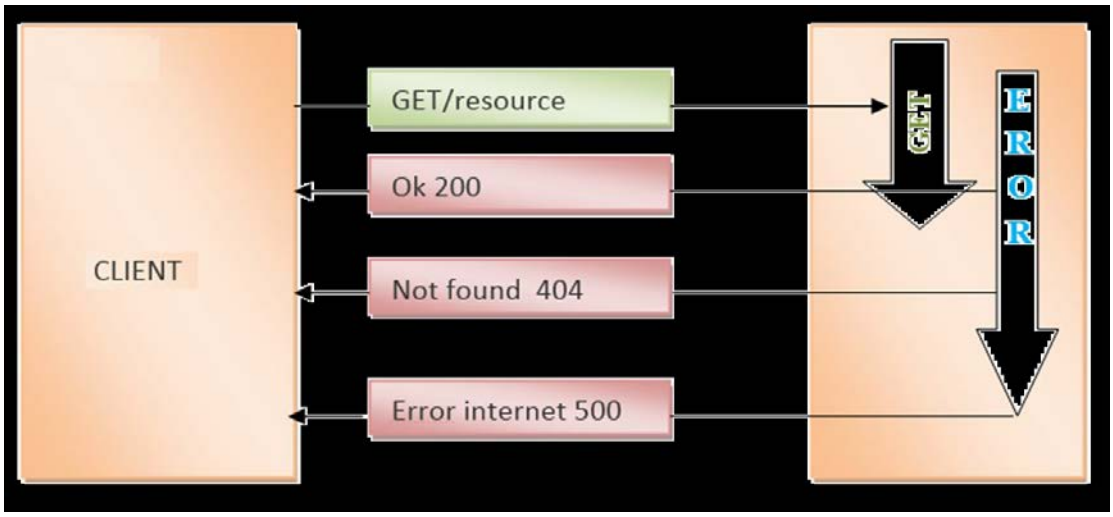

Figure IV.2 : Méthode http GET

b) POST

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. 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 :

- 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 IV.3 :Exemple de la méthode Delete

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.

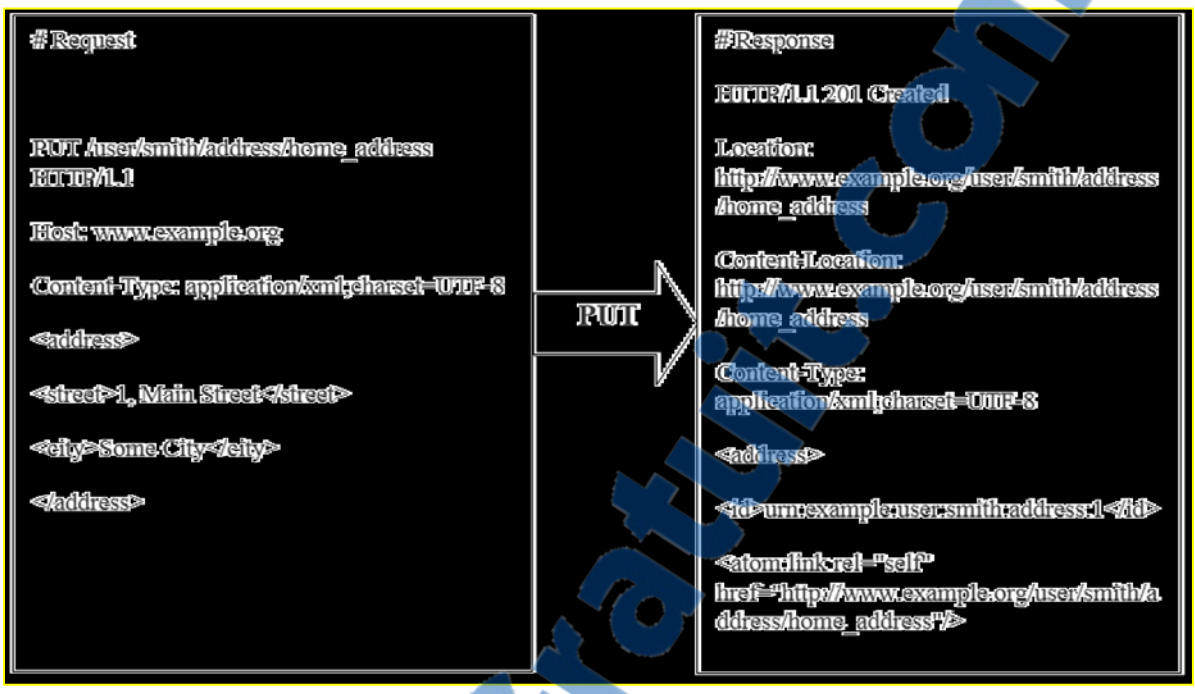

Figure IV.4 :Exemple de la méthode PUT

e) HEAD

HEAD est exactement comme GET, sauf qu'au lieu de retourner un corps de la

réponse, il ne retourne que le code de réponse et les en-têtes associés à la demande.

III. 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.

Conception et réalisation d'un modeleur de composition de service web SOAP et REST

5. La mise en oeuvre 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 oeuvre est flexible mais bien sûr même pour un ad hoc.

IV. Rest VS Restful

Si une application est sans état , utilise uniquement des méthodes HTTP ,expose sa structure dans les URI et qu'elle utilise dans ses requêtes du XML et /ou du JSON ,elle respecte les différents préceptes de REST ;elle est donc RESTful .

En résumé, une application qui ne s'expose qu'en REST est RESTful .Si elle s'expose autrement ,alors elle n'est pas RESTful.

V. Avantages

- Simplicité de mise en œuvre .
- Lisibilité par un humain .
- Evolutivité .
- Repose sur les principes du web.
- Représentations multiples (XML, JSON,...).

VI. Inconvénients

- Sécurité restreinte par l'emploi des méthodes http.
- Cible l'appel de ressources. [14]

VII. Représentation des ressources : JSON

Parce que REST et HTTP ont une approche multidimensionnelle à l'adressage la de 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.

Que ce soit au niveau du serveur ou au niveau des clients, une API RESTful manipule des ressources par une représentation de celles-ci.

Le principe en pratique est de passer de la représentation utilisée en interne, que ce soit sur le client ou le serveur, à la représentation utilisée dans le message HTTP, requête ou réponse.

Concernant la spécification du format de représentation utilisé dans le message : comme spécifié plus haut, les messages d'une application RESTful sont indépendants les uns des autres ce qui implique en particulier que le codage de la représentation des ressources peut

être différent entre deux messages et donc qu'il faut spécifier dans le message le codage utilisé.

Pour un Web Service RESTful on utilise typiquement un entête HTTP :

Content-type: text/xml

Le client doit demander un format d'encodage spécifique en utilisant l'entête Accept :

Accept: text/xml

Pour les Web Services RESTful, les encodages les plus utilisés sont XML et JSON.

XML est un standard incontesté mais souffre de quelques inconvénients:

- verbeux
- difficilement lisible par un humain
- dualité entre les attributs et les éléments.

VIII. Utilisation REST :

- Utiliser dans le développement des applications orientés ressources (ROA) ou orientées données (DOA)
- Les applications respectant l'architecture REST sont dites RESTful.

IX. Conclusion

Un Web Service RESTful est très simple à utiliser (partie cliente) et relativement simple à écrire (partie serveur) à partir du moment où l'on fait appel à des bibliothèques existantes. Le seul prérequis est la connaissance de HTTP.

Ce type d'architecture connaît un grand succès en ce moment, confirmé par l'intérêt d'acteurs tels que Google, Yahoo! Ou encore Amazon. Une des raisons principales tient à l'importance croissante des navigateurs en tant que plateforme d'exécution sur les postes clients fixes mais également sur les terminaux mobiles. Les applications web sont en train de passer d'un mode« document» (le serveur retourne une mise en forme du résultat) à un mode desktop (le client charge la partie interface de l'application, les requêtes au serveur ne font plus que charger les données) essentiellement grâce à l'utilisation d'Ajax. Dans ce contexte, fournir une interface RESTful à une application permet d'envisager une intégration relativement aisée.

Chapitre V

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 :

L'objectif essentiel de ce travail se résume à « la conception et la réalisation d'une application web pour la gestion d'un Hôtel », une application fiable et maniable afin de facilite la tâche de réservation (chambres) ; un service web développé en java, le serveur est fait à l'aide de
Pour mieux décrire notre application on va utiliser quelques diagrammes UML : le diagramme de cas d'utilisations, le diagramme de classes .

II.1. le Diagramme de classes :

Le diagramme suivant (**figure V.1)**décrit l'ensemble des classes de notre application. Nous avons utilisé des simples classes de chambres et reservation pour faciliter les choses et les relations entre les différentes classes.

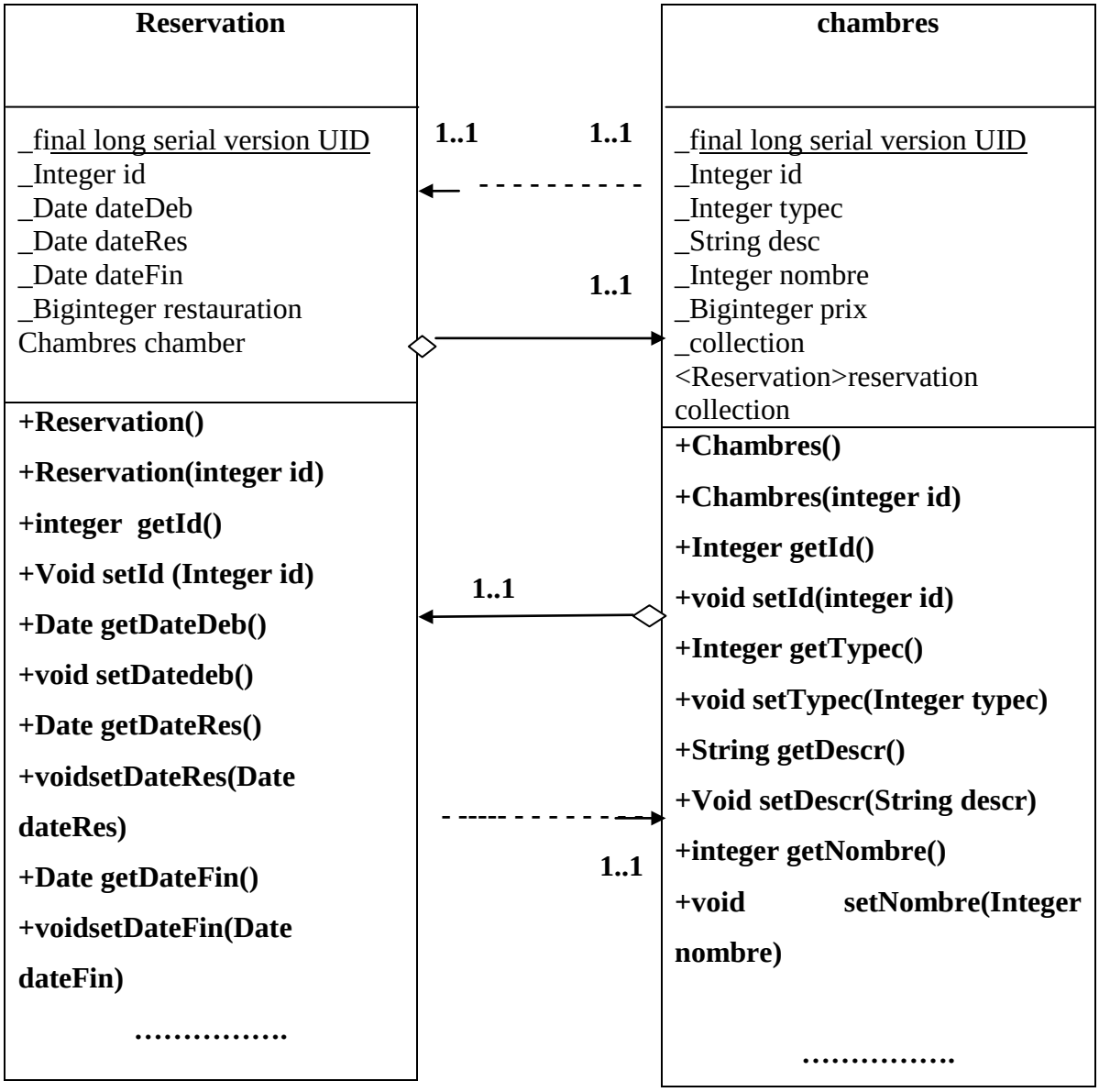

Figure V.1 :diagramme de classes

II.2.Diagramme d'activité « gestion des réservations des chambres »

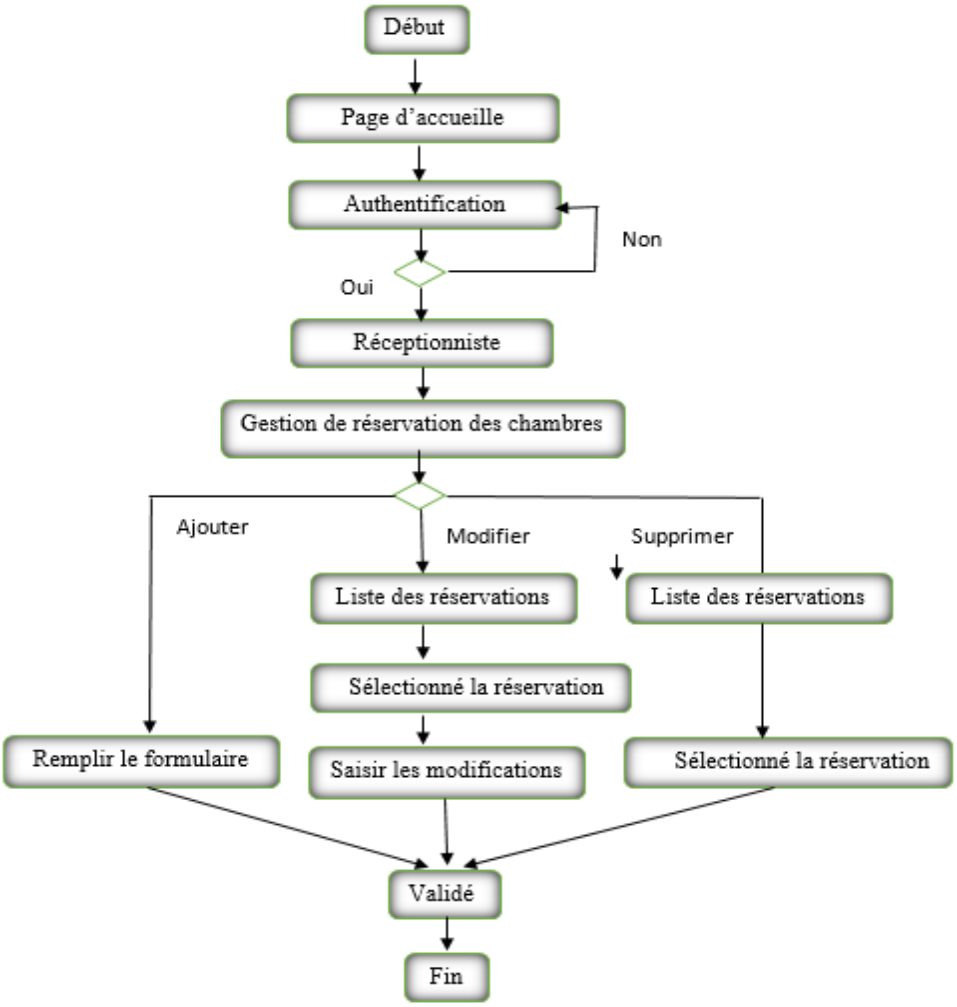

Figure V.2:diagramme d'activité [16]

II. 3.Les acteurs du système :

Un acteur représente un rôle joué par une entité externe (utilisateur humain, dispoitif matériel ou autre système) qui interagit directement avec le système étudié. Il peut consulter et/ou modifier directement l'état du système. L'acteur qui interagisse avec l'application

à concevoir est :

- **Le réceptionniste :** gestion des réservations des chambres et des clients.
- **Le client** : Cet acteur peut consulter, réserver, et suivre le déroulement d'une réservation.

III. L'implémentation :

III.1.Les différents outils utilisés :

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.

2. **TOMCAT :** L'utilisation d'un serveur Java EE est obligatoire pour le développement de pages Web dynamiques en Java EE. Un serveur HTTP classique reçoit des requêtes HTTP et renvoie des réponses mais il ne connait pas les Servlets, les JSP... Il est donc essentiel d'utiliser un programme appelé moteur de Servlets qui est contenu dans le serveur Java EE et qui permet de pallier ce manque.
3. **APACHE**

Apache est le serveur Web le plus utilisé sur Internet. Dans une architecture en production, il est recommandé d'utiliser un serveur Web en frontal d'un serveur d'applications. Ces recommandations sont également appliquées dans le cas de l'utilisation d'un conteneur Web comme Tomcat. L'utilisation d'un serveur Web en frontal est nécessaire dans ce projet pour des raisons de performance, de sécurité et de flexibilité.

4. **Coupler Tomcat et le serveur web Apache :**

L'intégration d'un serveur Tomcat avec un serveur Web se fait au travers d'un connecteur configuré au sein de Tomcat et d'une extension ajoutée au serveur Web. Un connecteur Tomcat est une classe Java qui supporte un protocole réseau spécifique et propriétaire. La librairie d'extension du serveur Web permet un dialogue entre les deux serveurs.

5. **PostgreSQL :**

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD.

Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB, MySQL et Firebird), ou propriétaires
(comme Oracle, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.

6. **La Java Persistence API :**

La **Java Persistence API** (abrégée en JPA), est une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java.

La Java Persistence API est à l'origine issue du travail du groupe d'experts JSR 220.

La persistance dans ce contexte recouvre 3 zones :

- l'API elle-même, définie dans le paquetage javax.persistence
- le langage Java Persistence Query (JPQL)
- l'objet/les métadonnées relationnelles

C'est un moteur adapté à des bases métier, donc riche en fonctionnalités et puissant. Son installation est cependant plutôt simple. Il faut juste comprendre quelques principes de base (ce que cette présentation s'efforce de faire)

III.2.Présentation des interfaces de l'application :

1. Page d'accueil

Cette page représente la page d'accueil présentée par la figure si dessous apparait lorsque l'utilisateur accède à l'application. Cette dernière a deux boutons «Book» pour accéder à la réservation, et un autre «Her» pour voir la description d'hotel Les Zianides.

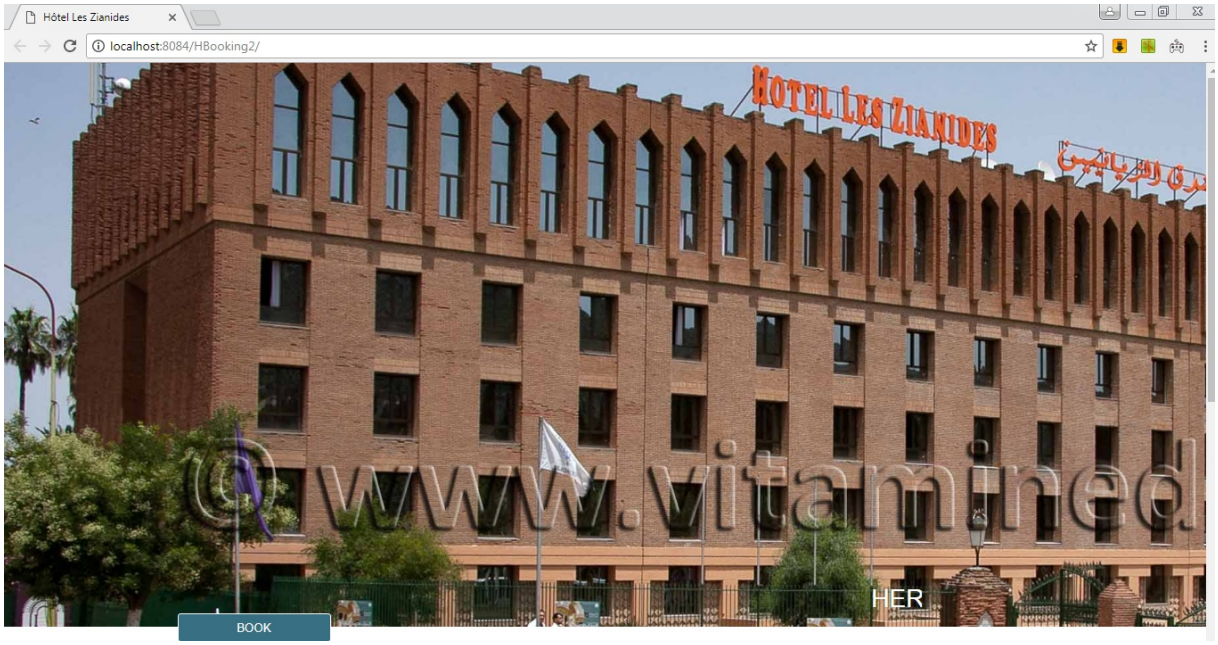

Figure V.3 : Interface page d'accueil.

Si on click HER,on trouve la description d'hotel ZIANIDES comme suit :

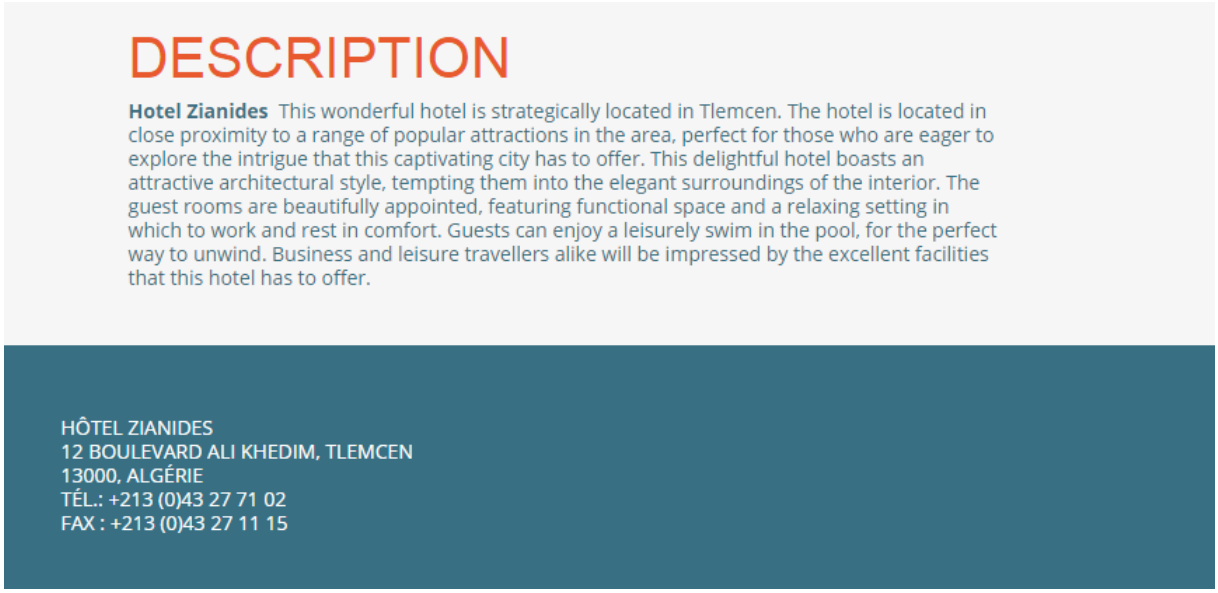

Figure V.4 :fenêtre d'une description d'un hôtel

L'objectif principal de cette application c'est la gestion d'un hôtel ,exactement la gestion de réservation des chambres si elles sont disponibles ou occupées pour une date précise ,pour cela en utilise une fenêtre spécial pour la réservation et service web REST ou SOAP.

Pour faire la reservation on click sur le boutton BOOk et on trouvera la fenêtre suivante :

Figure V.5 :formulaire d'une réservation d'une chambre

On prend par exemple un hôtel ,il est constitué d'un certain nombre de Chambres de types single , double et suite .

On suppose que cet hôtel a trois chambres single ,trois chambres double et trois chambre de type suite.

Si on veut faire une réservation on remplit les champs qui existe dans le **figure V.5** et on fait le choix entre le service web REST et service web SOAP ;la différence entre les deux on trouvera dans le tableau suivant :

SOAP/XML	**REST/JSON**
Protocole	Style architecture
Fort couplage	Faible couplage
XML implique un analyseur syntaxique	JSON est sous-ensemble de java script
Utilisation WSDL pour la communication entre consommateur et fournisseur	Utilisation XML ou JSON pour envoyer et recevoir des données
Invoque les services en appelant la méthode RPC	Simplement appelle les services via URL PATH

TableauV.1 : Comparaison SOAP et REST [15]

on remplit les champs d'après le souhaite du client et on choisit le service web REST comme suite :

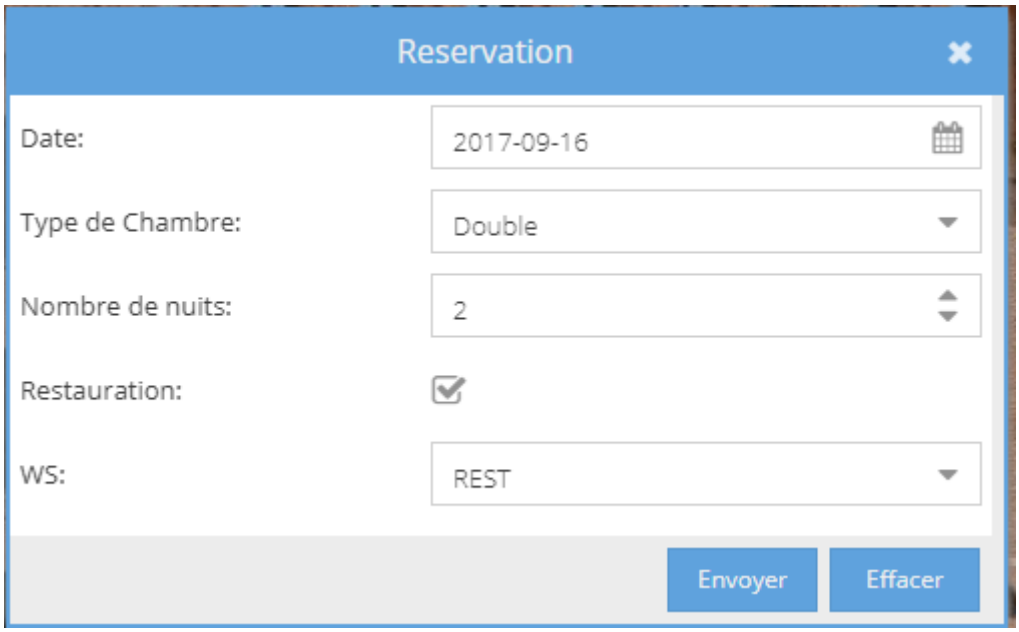

Figure V.6 :Réservation d'une chambre

si la chambre est disponible et n'est pas occupée ,le message vient comme suite :

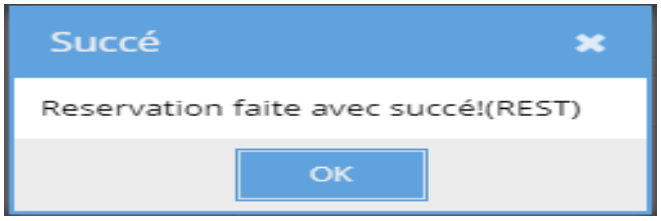

figure V.7: Réservation Succé par ws REST

Et on trouve même résultat si on choisit le service web SOAP ;la difference existe que REST utilise XML ou JSON mais SOAP utilise seulement XML .

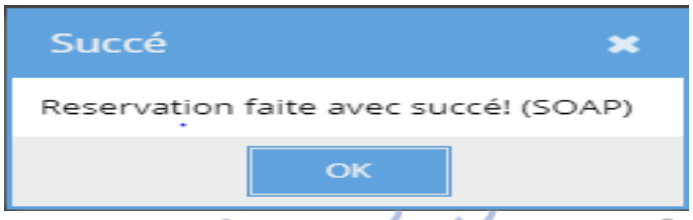

Figure V.8 : Réservation Succé par ws SOAP

si on choisit service web REST et la chambre est occupée pour cette date donc le message est comme suite :

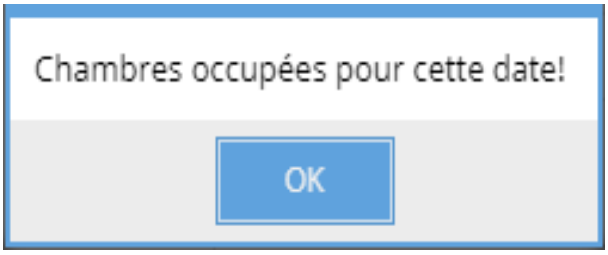

figure V.9 : Réservation échec par ws REST

IV. Conclusion

La phase de réalisation est une étape très importante dans le cycle de vie d'une application. Dans ce dernier chapitre nous avons illustré le déroulement de la réalisation de notre application gestion de réservation hôtels à base service REST et SOAP , présenté les outils et le langage de programmation. En fin, on conclut avec une présentation des différentes interfaces de notre application.

Conclusion générale

Les Web Services possèdent une simplicité de mise en oeuvre : Ils rendent en effet accessibles depuis Internet des fonctionnalités d'une application exixtante tout en ne modifiant pas en profondeur le système d'information de l'entreprise.

Les services Web avec ses protocoles et ses standards avance vers toujours plus de normalisation.

Un Web Service RESTful est très simple à utiliser (partie cliente) et relativement simple à écrire (partie serveur) à partir du moment où l'on fait appel à des bibliothèques existantes. Le seul prérequis est la connaissance de HTTP.

Déjà, le protocole d'échange de messages SOAP et le langage WSDL pour la définition de l'interface standards. Les Web Services reposent sur des bases solides(SOAP etWSDL)qui ont prouvé leur efficacité et leur maturité même si un normalisation complète n'existe pas encore.

Un des avantages principaux des Web Services est qu'ils sont basé sur Internet qui est on le sait fiable et mature.

Le travail effectué dans ce mémoire a pour objectif la réalisation d'une application pour la gestion des réservations de l'hôtel dans le but de faciliter la tâche aux personnels en leurs donnant la possibilité de gérer les chambres et les réservations.

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] Un 360°pour bien les comprendre,Décembre 2016 , Disponible sur :

http://www.cigref.fr/wp/wp-content/uploads/2016/12/CIGREF-Objets-Connectes-2016.pdf

[2] Wikipedia®,https://fr.wikipedia.org/wiki/Web_des_objets

[3] Disponible sur :https://www.techopedia.com/definition/26834/web-of-things-wot

[4] Disponible sur :

http://liris.cnrs.fr/amille/enseignements/Ecole_Centrale/projets_2006/Compte%20rendu%20WebServ

[5] Chabane Refes, Les services Web Développement web , 22févr.2017, Disponible sur :

https://openclassrooms.com/courses/les-services-web

[6] W3C Recommandation, www.w3.org/2002/07/soap-translation/soap12-part1.html.

[7] W3C Recommandation, www.w3.org/2002/07/soap-translation/soap12-part0.html.

[8] Pierrick Vanneau, Mathieu Samson et Julien Donel , SOAP – JINI , IR3 2005,

http://igm.univ-mlv.fr/~duris/NTREZO/20042005/Donel-Samson-Vanneau-Jini-SOAP.pdf

[9] DALI YAHIA Mohammed,selection de service web a base QoWs,

http://dspace.univ-tlemcen.dz/bitstream/112/930/1/DALI-YAHIA-Mohammed.pdf

[12] Disponible sur : http://www.w3.org/2001/XMLSchema-instance

[13] Roy T. Fielding, Representational State Transfer (REST),http://opikanoba.org/tr/fielding/rest/

[14] Samuel SIMON, Services Web : REST et SOAP, le 05/12/2016, https://www.supinfo.com/articles/single/3786-services-web-rest-soap

[15] Longueuil, Québec, Canada, Comparaison de plateformes logicielles pour programmation de services Web dans un environnement aux ressources limitées ,octobre 2016, Disponible sur :

https://www.usherbrooke.ca/cefti/fileadmin/sites/cefti/documents/Essais/CeFTI_-_essai_Dagenais

[16] AKOU Hamza ,HARKOUK Saïd, Conception et Réalisation d'une Application Web de Gestion d'Hôtel Sous JAVA Cas : « Hôtel ROYAL», http://www.univbejaia.dz/dspace/bitstream/handle/123456789/740/Conception%20et%20R%C3%A

Résumé

Aujourd'hui, l'informatique a atteint une prodigieuse évolution technologique dans différents domaines (réseaux informatiques, bases de données...). Cette évolution est nécessaire pour remédier aux problèmes rencontrés dans la vie quotidienne. Automatiser des informations est l'un des rôles essentiels de l'informatique. C'est ceci qui nous a poussés à créer une application pour la gestion d'un hôtel accessible par des utilisateurs.

Notre travail consiste à faire un simple service web (conversion de devises), nous avons implémenté les deux parties client et serveur, Le serveur est implémenté à l'aide des technologies suivantes : Apache Tomcat ,et PostgresSQL est le serveur de base de données de l'application tant dis que le client est une simple application Java.

Mots clés : client/serveur, Java, PostgresSQL , service web, Apache Tomcat.

Abstract

Today, computer science has reached a tremendous technological development in different fields (computer networks, databases ...). This is necessary to address the problems in now a days life.Automate information processing is one of the essential roles of the computer. It is this that led us To create a network application for the management of a hotel.

Our work focus on making a simple web service (devices conversion), we are implemented two parts client and server, The server is implemented with the aid of the following technologies:

Apache Tomcat,and PostgresSQL server database application however the client is a simple Java application.

Keywords: client/server, Java, PostgresSQL , web service, Apache Tomcat.

ملخص:

اليوم، تكنولوجيا المعلومات قد حققت التكنولوجيا المذهلة في مختلف النطاقات (شبكات الكمبيوتر وقواعد البيانات، وما إلى ذلك).

هذا التطور هو ضروري لعلاج المشاكل التي تواجهها في الحياة اليومية. أتمتة المعلومات هي واحدة من الأدوار الأساسية للحوسبة. وهذا ما دفعنا إلى إنشاء تطبيق لإدارة فندق يمكن الوصول إليه من قبل المستخدمين.

يتمثل عملنا في خدمة ويب (تحويل العملات) ,و قمنا بتنفيذ كل من العميل و الملقم ,يتم تنفيذ الملقم باستخدام تقنيات تكنولوجية حديثة مثل: اباش طوم كات وبوست قراس.اما العميل يتم تطبيق عليه برنامج جافا.

الكلمات الاساسية: العميل/الملقم_جافا_بوست قراس_اباش طوم كات.