Les services web

4 Introduction

Tout a commencé avec l'explosion de web, que les chercheur en commencé de développée des logiciels pour faciliter la communication entre les machine et les applications connecté via internet ces logiciel est nommé « web service » en français les services web. En quelques années, les services web sont devenus le nouveau point de convergence technologique de l'industrie du logiciel.

Dans ce chapitre, nous verrons les principales caractéristiques des web services, ce qui les composent et ces aspects technologiques, ces types comme SOAP et REST avec leur avantage et inconvénient, une définition RESTFUL API et une comparaison SOAP vs REST, en termine avec une conclusion.

4.1 Définition

La Définition du W3C (World Wide Web Consortium) définie les web services comme suit :
« 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. »

Un service web est donc une application client-serveur faiblement couplée, identifiée par un URI (Uniform ressource identifier) faisant communiquer des programmes, bases de données, objets, processus d'affaire, et traitent les données en laissant accéder parfois à une partie du système. Exemple d'URI : une URL (pour un site web), un URN (l'identifiant ISBN d'un livre).

Ces applications, indépendantes de tout langage, de l'implémentation, de l'OS de la plate-forme, de l'architecture sous-jacente (.NET, JEE, …), sont destinées à être utilisées par d'autres applications et moins par des humains. Elles utilisent les

protocoles basiques d'internet, SMTP, HTTP. Bien qu'échangeant en XML, un web service peut être implémenté dans différents langages (java, c++, VB etc.).
Contrairement aux EDI, (L'Échange de Données Informatisées) fortement couplés, les web services sont faiblement couplés, donc on peut modifier L'API et l'application qui l'utilise sans que cela dégrade l'interaction.(26)

4.2 Caractéristiques des web services

Le web service doit pouvoir interagir avec le client et le serveur, et pour être réutilisable, il a donc certaines caractéristiques :

- Un contrat entre les deux parties.
- Le fournisseur du service doit utiliser ou mettre en place des standards et normes le définissant.
- Le web service doit proposer une interface d'échanges au client (interface web par exemple).
- Il doit pouvoir être décrit et être découvert.
- Le web service doit être sécurisé.
- Le web service doit être fiable au niveau de son implémentation.(26)

4.3 Avantages

Les avantages du Web services sont nombreux, comme le démontre la liste ci-dessous :

- Une intégration facilitée d'un système d'informations avec une plateforme marchande
- Ses composants sont réutilisables
- Une Interopérabilité permet de lier les différents systèmes entres eux
- Réduction de couplage entre les systèmes.
- Un périmètre fonctionnel étendu mis à la disposition des marchands : Import, Inventaire, Gestion de la commande, Pricing, Après-Vente…
- Met en relation des systèmes hétérogènes
- Interconnecte des middlewares/ou permet de les installer
- Compatible avec tous langages
- Il permet de faire communiquer serveurs et machines,

- Puissance de calcul nécessaire réduite
- Multi-utilisateur, sans perturber les sources
- Mise à jour des composants facile
- Maintenance réduite (comme tout outil de big data)
- Il n'est lié à aucun système d'exploitation ou langage de programmation

Tous ces avantages font qu'aujourd'hui les Web Services sont encore utilisés aux seins des entreprises, et des sites web.(26)

4.4 Architecture des web services

Les web services peuvent être définis sous différentes formes, en fonction de la demande.

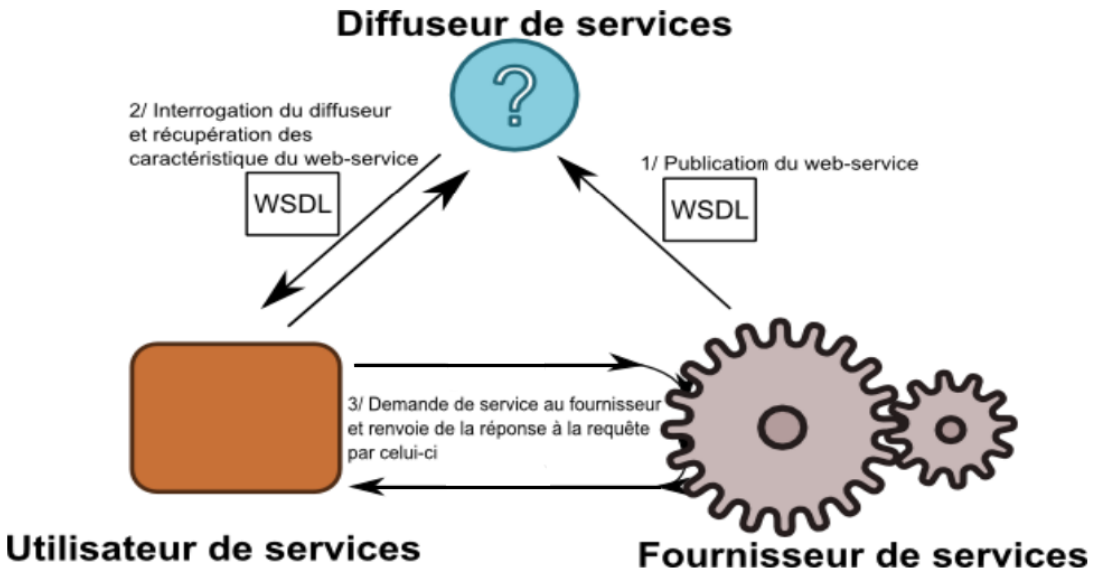

Figure 4-1 : Principe des échanges web services(27)

4.4.1Les applications format client-serveur

Un utilisateur envoie d'une machine client (ordinateur, ou autre) des requêtes à un serveur qui lui répond, ce serveur de niveau 2 (serveur d'applications) puise souvent dans un serveur de niveau 3 (serveur de bases de données). On peut résumer comme suit :

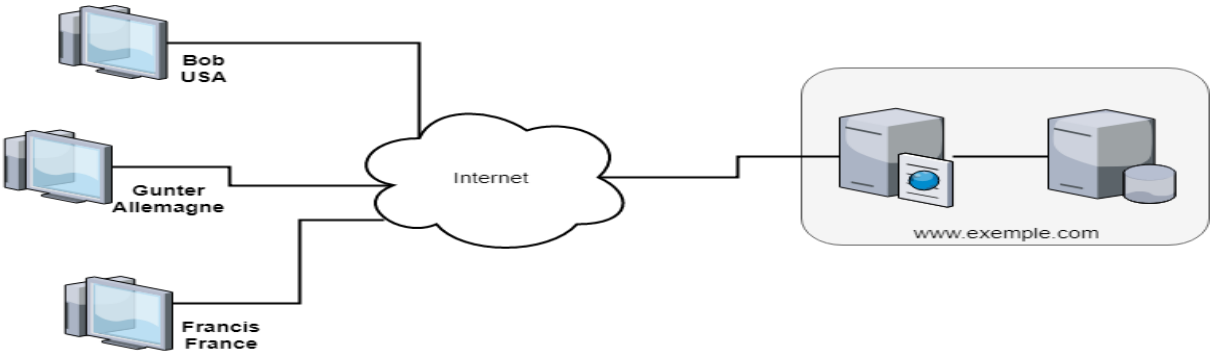

Figure 4-2 : Liaison entre clients et serveur web. (19)

4.4.2Les applications format faiblement couplées

On appelle faiblement couplées des applications peu dépendantes d'autres applications, et qui échangent peu d'information entres elles. Il existe 7 niveaux de couplage, allant du plus faible au plus fort :

1. **Sans couplage :** Aucun échange n'est effectué entre les composants
2. **Par données :** Les données échangées se font uniquement par commande avec des paramètres 'simple', comme des nombres, des tableaux ou autres.
3. **Par paquet :** même chose que par données, sauf que les paramètres sont cette fois-ci 'composé', comme des structures ou des classes
4. **Par contrôle :** les logiciels s'échangent leur contrôle en renvoyant leur 'verrou', ou drapeaux entre eux.
5. **Par liaison Externe :** Une communication externe (comme un fichier ou un lien de communication) est mise en place entre les deux composants.
6. **De type commun :** un ensemble de données communs entre les deux services est utilisé.
7. **Par contenu :** niveau le plus élevé, chaque service à accès aux données de l'autre, et vient donc puiser directement à l'intérieur.

La bonne pratique est d'avoir un niveau de couplage le plus bas possible, surtout dans les web services, car cela rend beaucoup plus difficile d'utiliser le service ailleurs, et complique le suivi des données.(26)

4.4.3Les applications format distribuées

Une application distribuée s'installe sur deux machines distantes afin de les faire communiquer, souvent par le biais d'objets. SOAP, et bien avant, CORBA (OMG), DCOM (MS), RMI (SUN) et bien d'autres utilisent tous cette technologie.(26)

4.4.4Les architecture SOA (orientée services)

Une architecture orientée services fait communiquer entre eux des objets en transcrivant la demande du serveur vers le client. Le message est donc traduit pour répondre à la demande.(26)

4.5 Les aspects technologiques

Le web service a besoin de plusieurs couches pour fonctionner :

- La couche invocation décrivant la structure des messages échangés
- La couche découverte pour trouver le web service au sein de l'annuaire de web services (nom de société, utilité de chaque service)
- et la couche description, contenant paramètres des fonctions, typages des données du web services.

Nous verrons dans ce point les différents aspects technologiques entourant les web services.(26)

4.5.1Transport : SOAP (Simple Object Acces Protocol)

SOAP est un protocole de communication d'ordinateur à ordinateur sous HTTP très simple, écrit en XML. Il permet l'échange de données, quel que soit les systèmes d'exploitation. Les messages SOAP sont des transmissions en sens unique d'un émetteur vers un récepteur.(26)

4.5.2Découverte : UDDI (Universal Description, Discovery and Integration)

UDDI est une norme d'annuaire de services Web appelée via le protocole SOAP. Pour publier un nouveau service Web, il faut générer un document appelé **Business Registry**, il sera enregistré sur un UDDI Business Registry Node (IBM, Microsoft et SAP en hébergent chacun un). Les nodes sont répliqués entre eux suivant un mécanisme analogue aux DNS. Le Business Registry comprend 3 parties :

- **Pages blanches** noms, adresses, contacts, identifiants des entreprises enregistrées.

- **Pages jaunes** informations permettant de classer les entreprises, notamment l'activité (classifications NAICS, UNSPSC...), la localisation...
- **Pages vertes** informations techniques sur les services proposés.

Le protocole utilise 3 fonctions de base :

- **publish** pour enregistrer un nouveau service,
- **find** pour interroger l'annuaire,
- **bind** pour effectuer la connexion entre l'application cliente et le service.

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

UDDI est maintenant un standard employé.(26)

4.5.3Description : WSDL (Web Services Description Language)

WSDL, basé sur XML, permet de décrire le service Web, en précisant les méthodes disponibles, les formats des messages d'entrée et de sortie, et comment y accéder. (28)

Pour écrire le programme d'invocation du service, on utilisera les informations suivantes :

- **types :** définitions de types de donnée échangées.
- **message :** définition abstraite des données transmises
- **type de port :** ensemble d'opérations correspondant chacune à un message entrant ou sortant.
- **rattachement (binding) :** protocole de communication et format des données échangées pour un port.
- **port :** adresse (assure l'unicité du rattachement).
- **service :** regroupe un ensemble de ports. (28)

4.6 Les types de service web

Il existe différentes méthodes pour fournir des services Web, mais les plus courantes sont SOAP et REST :

4.6.1SOAP (Simple Object Access Protocol)

SOAP signifie Simple Object Access Protocol. Créé en 1998, l'objectif est de l'utiliser sur le marché des entreprises. En particulier, SOAP doit créer une logique d'application en tant que service. L'objectif est d'être un nouveau protocole de communication.(29)

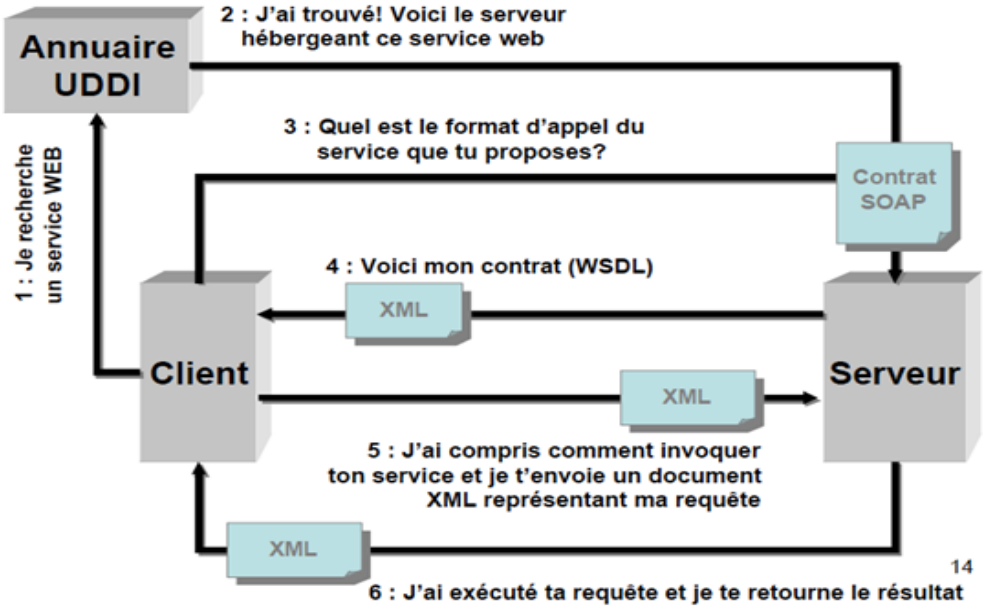

Figure 4-3 Architecture d'un Service WEB SOAP(26)

Avantages

- capable de travailler sur n'importe quel protocole.
- Expliquer le service avec WDSL (Web Service Description Language)
- Fiable Lorsqu'un problème survient, il peut être réessayé.
- La sécurité existe déjà, à la fois l'authentification, l'autorisation et le cryptage des données.(29)

Désavantage

- Difficile à développer. Rendre populaire pour le Web et le mobile.
- formats de données pris en charge, XML seul.
- parce que c'est une norme, il y a la limitation, mais en raison de ses nombreuses parties, il y a une surcharge ou le besoin de bande passante est plus élevé que le REST.(29)

Quand Utiliser SOAP

- lors de la gestion des transactions lorsque vous travaillez avec plusieurs systèmes.
- Lors de la connexion à une connexion stricte entre client / serveur
- Lorsque, par exemple, un service financier et un service de télécommunication,(29)

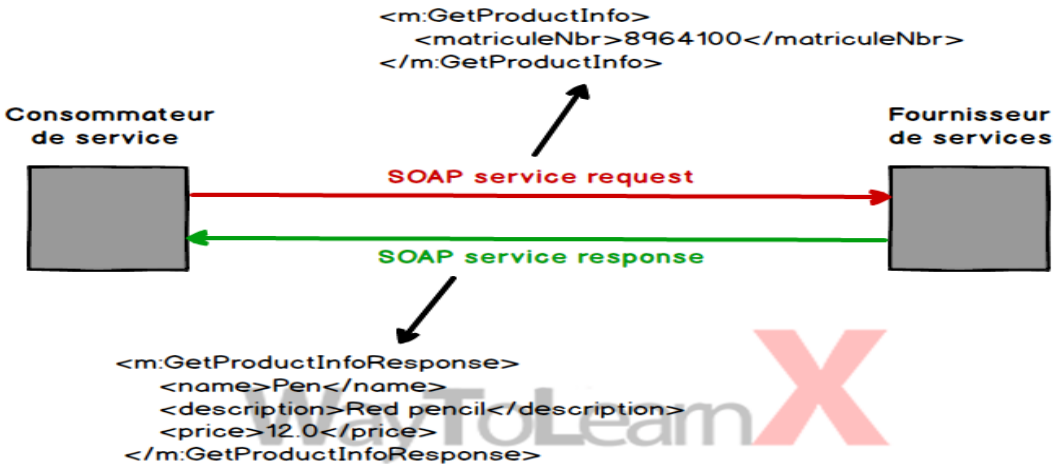

Figure 4-4 : Exemple d'une demande (request) dans le service SOAP(30)

4.6.2Method REST (Representational State Transfer)

REST signifie Représentationnel State Transfer et c'est un style architectural qui permet la communication entre les systèmes. Le terme REST a été inventé pour la première fois par Roy T. Fielding dans son doctorat. Dissertation.(31)

REST a été créé depuis 2000. Le but est d'être l'une des conceptions de technologie Web ouverte qui souhaite mettre les données sous la forme d'une ressource. Les actions suivent le verbe HTTP ou la méthode HTTP (GET, POST, PUT, DELETE).(29)

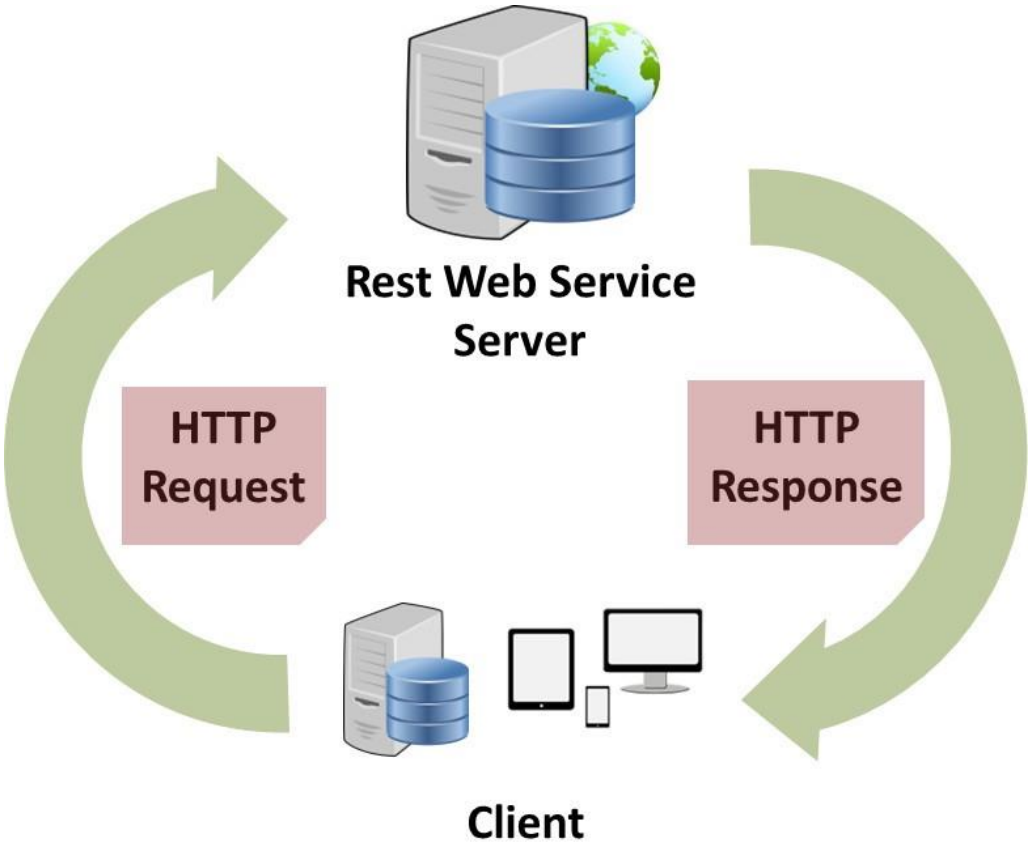

Figure 4-5 Architecture REST (32)

Avantages

- Il est basé sur HTTP et est conforme aux normes HTTP, il peut donc être facilement développé.
- Prend en charge de nombreux formats de données tels que XML, JSON, texte brut et bien d'autres.
- Prise en charge de Easy to expand du système
- Bonne performance
- Prend en charge la mise en cache des données.(29)

Désavantages

- Cela fonctionne uniquement avec le protocole HTTP.
- Il n'y a pas de sécurité et de fiabilité intégrée, alors faites-le vous-même
- formats de données envoyés entre client-serveur Il n'y a pas de restrictions.(29)

Quand utiliser REST

- lorsque vous souhaitez réduire la quantité de données et la quantité de bande passante utilisée
- lorsque cela est nécessaire lorsque vous travaillez sur des systèmes Web et mobiles, par exemple, un service de médias sociaux, un service de chat Web, il n'est donc pas étrange de savoir pourquoi les systèmes Web et API Via le Web est donc REST.(29)

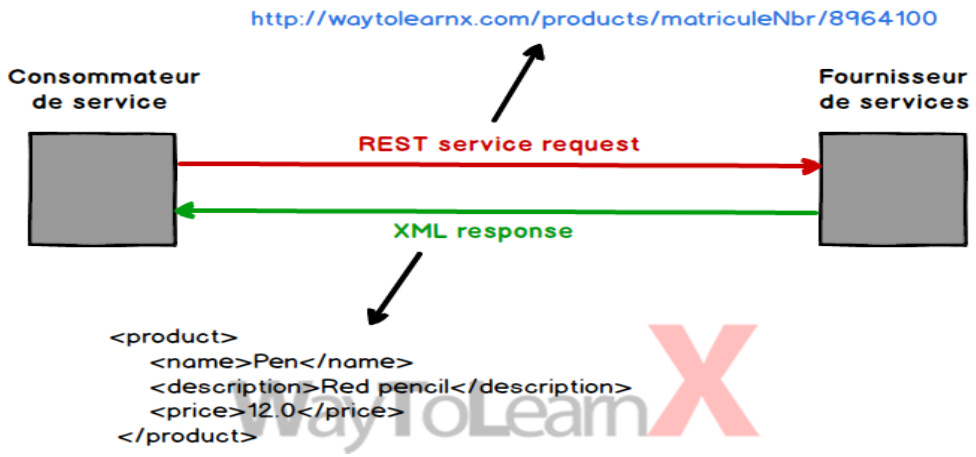

Figure 4-6 : Exemple d'une demande (request) dans le service REST(30)

4.6.3Tableau comparatif entre REST et SOAP

Après étude de différentes architectures utilisées dans les web services, on a remarqué que l'architecture REST est la plus adaptée pour notre application. Pour cela, on va montrer dans un tableau, qui représente les comparatifs entre l'architecture REST et SOAP, et qui montre les points forts de REST.

4.7 REST vs SOAP

REST et SOAP proposent différentes méthodes pour appeler un service Web. REST est un style architectural, tandis que SOAP définit une spécification de protocole de communication standard pour l'échange de messages XML. Les applications REST peuvent utiliser SOAP.(33)

Les services Web REST sont sans état. Une implémentation basée sur REST est simple par rapport à SOAP, mais les utilisateurs doivent comprendre le contexte et le contenu transmis,

car il n'y a pas d'ensemble standard de règles pour décrire l'interface des services Web REST. Les services REST sont utiles pour les appareils à profil restreint, tels que les mobiles, et sont faciles à intégrer aux sites Web existants.(33)

SOAP nécessite moins de code de plomberie c'est-à-dire un code d'infrastructure de bas niveau qui relie les principaux modules de code entre eux que la conception de services REST. Le langage de description des services Web décrit un ensemble commun de règles pour définir les messages, les liaisons, les opérations et l'emplacement du service. Les services Web SOAP sont utiles pour le traitement et l'appel asynchrones. (33)

Tableau 1 : Tableau de comparaison entre les caractéristiques des Services WEB basés sur SOAP et REST

SOAP	REST
Les opérations sont définies comme WSDL les ports	Les opérations sontdéfinies dans les messages
Adresse unique pourchaque opération	Adresse unique pourchaque instance de processus
Instances des processus multiples partagent lamême opération	Chaque objet supporte les opérations(standards) définies
Couplage serré deComposants	Couplage lâche deComposants
Le débogage est possible	La liaison tardive estpossible
Les opérations complexes peuvent être cachées derrière la façade	Les instances des processus sont créées explicitement

.	
Emballage existantAPI estsimple	Le client n'a pas besoin desinformations de routageau-delà de l'initialeURI de l'usine de processus (ProcessFactory)
Confidentialité augmentée	Le client peut en avoir une interface auditeur générique pour les notifications
Les clients doivent connaître au préalable les opérations et leurs sémantiques	Grand nombre d'objets
Le client a besoin de ports dédiés pour des différents types de notification Les instances des processus sont créées implicitement	La gestion de l'URI peut devenir lourde

4.8 RESTFULL API

Nous allons voir ce qu'est RESTful API et à quoi cela sert, mais avant cela, il est nécessaire de comprendre ce qu'est une API.

4.8.1API

Une API (Application Programming Interface),en français Interface de Programmation Applicative, est un moyen de mettre à disposition des méthodes et services à des applications tierces. Cela va permettre aux développeurs de réutiliser du code existant, plutôt que de devoir réinventer la roue à chaque fois. Un service web est un type d'API, accessible via internet, souvent via le protocole http (HyperText Transfert Protocole). Par exemple, l'utilisation du paiement par Paypal dans des applications tierces se fait via l'API mise à disposition par

Paypal. Ainsi, le système d'authentification est géré, et au lieu de passer énormément de temps à développer, il suffit de créer un compte développeur et récupérer la clé d'API, il en est de même pour l'API Facebook, Google…

4.8.2Définition

Le concept de REST est défini par certaines règles, contraintes ou principes. Le système, l'application, les services ou tout ce qui satisfait à ces principes REST sont appelés RESTful.(31)

Les services Web qui suivent les principes RESTful sont des services RESTful. L'URI est utilisé pour accéder aux services RESTful afin d'obtenir les ressources.(31)Dans le glossaire RESTful, les ressources ne sont que des données et des fonctions. Nous allons donc éventuellement appeler les services Web via URI pour accéder aux fonctions et ainsi obtenir les données de ressources.(31)

4.8.3Architecture d'API de services Web RESTful :

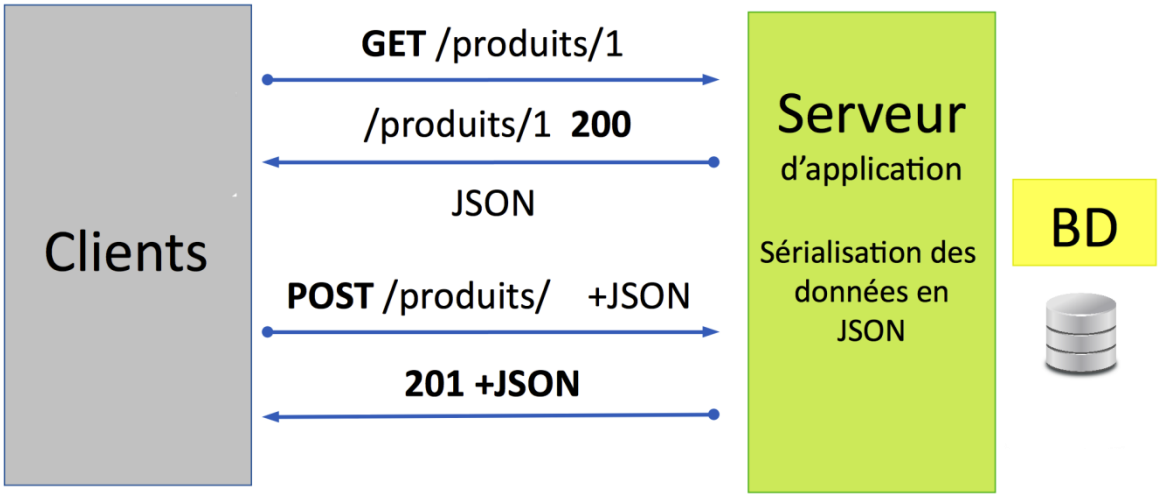

Figure 4-7 : Architecture d'une API de services web RESTful

4.8.4Fonctionnalités des services RESTFUL :

Une API RESTful décompose une transaction pour créer une série de petits modules. Chaque module aborde une partie sous-jacente particulière de la transaction. Cette modularité offre aux développeurs une grande flexibilité, mais il peut être difficile pour les développeurs de concevoir leur API REST à partir de zéro. Actuellement, plusieurs entreprises fournissent des

modèles à utiliser par les développeurs; les modèles fournis par Amazon S3 , Cloud Data Management Interface (CDMI) et Open Stack Swift sont les plus populaires.(33)

Une API RESTful utilise les méthodes HTTP existantes définies par le protocole RFC 2616. Ils utilisent GET pour récupérer une ressource; PUT pour changer l'état ou mettre à jour une ressource, qui peut être un objet , un fichier ou un bloc ; POST pour créer cette ressource; et DELETE pour le supprimer.(33)

4.8.4.1 Les Méthodes HTTP

4.8.4.1.1 GET

Utilisez les requêtes GET pour récupérer uniquement la représentation / les informations des ressources - et non pour les modifier en aucune façon. Comme les requêtes GET ne changent pas l'état de la ressource, on dit que ce sont des méthodes sûres. De plus, les API GET doivent être idempotentes**,** ce qui signifie que faire plusieurs requêtes identiques doit produire le même résultat à chaque fois jusqu'à ce qu'une autre API (POST ou PUT) ait changé l'état de la ressource sur le serveur.
Si l'URI de demande fait référence à un processus de production de données, ce sont les données produites qui doivent être renvoyées en tant qu'entité dans la réponse et non le texte source du processus, à moins que ce texte ne soit la sortie du processus.(34)

Pour toute API HTTP GET donnée, si la ressource est trouvée sur le serveur, elle doit renvoyer le code de réponse http 200 (OK) avec le corps de la réponse, qui est généralement du contenu XML ou JSON (en raison de leur nature indépendante de la plate-forme).
Si la ressource n'est PAS trouvée sur le serveur, elle doit renvoyer le code de réponse http 404 (NOT FOUND) De même, s'il est déterminé que la requête GET elle-même n'est pas correctement formée, le serveur renverra le code de réponse http 400 (BAS REQUEST).(34)

Exemple d'URI de demande :

- GET http://www.appdomain.com/users
- GET http://www.appdomain.com/users?size=20&page=5
- GET http://www.appdomain.com/users/123
- GEThttp://www.appdomain.com/users/123/address

4.8.4.1.2 POST

Utilisez les API POST pour créer de nouvelles ressources subordonnées, par exemple, un fichier est subordonné à un répertoire le contenant ou une ligne est subordonnée à une table de base de données. Lorsqu'on parle strictement en termes de REST, les méthodes POST sont utilisées pour créer une nouvelle ressource dans la collection de ressources.(34)

Idéalement, si une ressource a été créée sur le serveur d'origine, la réponse devrait être un code de réponse HTTP 201(Created) et contenir une entité qui décrit l'état de la demande et se réfère à la nouvelle ressource, et un en- tête Location .

Plusieurs fois, l'action effectuée par la méthode POST peut ne pas aboutir à une ressource pouvant être identifiée par un URI. Dans ce cas, il s'agit du code de réponse HTTP 200 (OK) ou de 204 (NO Content) l'état de réponse approprié.
Les réponses à cette méthode sont pas mis en cache, à moins que la réponse inclut appropriée Cache-Control ou Expires champs en- tête. (34)

Veuillez noter que POST n'est ni sûr ni idempotent, et l'invocation de deux requêtes POST identiques se traduira par deux ressources différentes contenant les mêmes informations (à l'exception des identifiants de ressource).(34)

Exemple d'URI de demande :

- POST http://www.appdomain.com/users
- POST http://www.appdomain.com/users/123/accounts

4.8.4.1.3 PUT

Utilisez les API PUT principalement pour mettre à jour la ressource existante (si la ressource n'existe pas, l'API peut décider de créer une nouvelle ressource ou non). Si une nouvelle ressource a été créée par l'API PUT, le serveur d'origine doit informer l'agent utilisateur via la réponse du code de 201(Created) réponse HTTP et si une ressource existante est modifiée,

les codes de réponse 200 (OK) ou 204 (NO Content) DEVRAIENT être envoyés pour indiquer la réussite de la demande. (34)
Si la demande passe par une antémémoire et que l'URI de demande identifie une ou plusieurs entités actuellement mises en cache, ces entrées devraient être traitées comme périmées. Les réponses à cette méthode ne peuvent pas être mises en cache**.**(34)

La différence entre les API POST et PUT peut être observée dans les URI de requête. Les requêtes POST sont effectuées sur des collections de ressources, tandis que les requêtes PUT sont effectuées sur une seule ressource.(34)

Exemple d'URI de demande :

- PUT http://www.appdomain.com/users/123
- PUT http://www.appdomain.com/users/123/accounts/456

4.8.4.1.4 DELLET

Comme le nom s'applique, les API DELETE sont utilisées pour supprimer des ressources (identifiées par Request-URI).(34)

Une réponse réussie aux demandes DELETE devrait être une réponse HTTP code 200 (OK)
Si la réponse comprend une entité décrivant l'état, 202 (Accepted) si l'action a été mise en file d'attente, ou 204 (NO Content) si l'action a été exécutée mais la réponse n'inclut pas d'entité.
Les opérations DELETE sont idempotentes. Si vous supprimez une ressource, elle est supprimée de la collection de ressources. L'appel répété de l'API DELETE sur cette ressource ne changera pas le résultat - cependant, appeler DELETE sur une ressource une deuxième fois renverra un 404 (NOT FOUND) car il a déjà été supprimé. Certains peuvent prétendre que cela rend la méthode DELETE non idempotente. C'est une question de discussion et d'opinion personnelle. Si la demande passe par une antémémoire et que l'URI de demande identifie une ou plusieurs entités actuellement mises en cache, ces entrées devraient être traitées comme périmées. Les réponses à cette méthode ne peuvent pas être mises en cache**.**(34)

Exemple d'URI de demande :

- DELETE http://www.appdomain.com/users/123
- DELETE http://www.appdomain.com/users/123/accounts/456

4.8.5Composants en Réseau de REST

Avec REST, les composants en réseau sont une ressource à laquelle l'utilisateur demande l'accès - une boîte noire dont les détails d'implémentation ne sont pas clairs. Tous les appels sont sans état; rien ne peut être conservé par le service RESTful entre les exécutions (33)

4.8.6Contraintes de conception et d'architecture d'API RESTful

Pour créer une véritable API RESTful, elle doit respecter les six contraintes architecturales REST suivantes:

Utilisation d'une interface uniforme (UI) : Les ressources doivent être identifiables de manière unique via une URL unique, et uniquement en utilisant les méthodes sous-jacentes du protocole réseau, telles que DELETE, PUT et GET avec HTTP, s'il est possible de manipuler une ressource. (33)

Basé sur le client-serveur : Il doit y avoir une délimitation claire entre le client et le serveur. Les problèmes liés à l'interface utilisateur et à la collecte des demandes sont du domaine du client. L'accès aux données, la gestion de la charge de travail et la sécurité sont le domaine du serveur. Ce couplage lâche du client et du serveur permet à chacun d'être développé et amélioré indépendamment de l'autre. (33)

Opérations apatrides : Toutes les opérations client-serveur doivent être sans état et toute gestion d'état requise doit avoir lieu sur le client et non sur le serveur. (33)

Mise en cache des ressources RESTful : Toutes les ressources doivent autoriser la mise en cache, sauf indication explicite que la mise en cache n'est pas possible.(33)

Système en couches : REST permet une architecture composée de plusieurs couches de serveurs. (33)

Code sur demande : La plupart du temps, un serveur renvoie des représentations statiques de Ressources sous forme de XML ou JSON . Cependant, si nécessaire, les serveurs peuvent envoyer du code exécutable au client.(33)

Conclusion

On à vue dans ce chapitre l'importance et l'avantage des services web, ces aspects technologiques, et deux type qui sont SOAP et REST avec une comparaison pour montrer quelle est la meilleur solution pour notre application, on à vue aussi c quoi une API et le RESTFUL API et en quoi le protocole REST est avantageux pour écrire des services web, notamment par rapport à SOAP.

Après cette étude, on a remarqué que l'architecture REST est la plus adaptée pour notre application on résume que REST est plus Structurant à cause du nommage des URI, Efficace et facile à développer à cause de l'utilisation de Protocol HTTP comme style de communication, ilprend en charge de nombreux formats de données tels que XML, JSON, texte brut et bien d'autres.

Le chapitre suivant décrit la phase de conception et réalisation de notre application en utilisant un système d'information qui existe déjà.