# API

Une API (Application Programming Interface) est une interface pour que les applications s’interagissent entre eux-mêmes.

# API REST

REST (Representational State Transfer) est une API qui imite la façon dont le web marche dans les échanges entre un client et un serveur. Une API REST est :

* Orienté client-serveur
* Sans état (le serveur n’a aucune idée de l’état du client entre deux requêtes. Du point de vue du serveur, chaque requête est une entité distincte des autres)
* Cacheable (un client doit être capable de garder en mémoire des informations sans avoir constamment besoin de demander tout au serveur).
* Avec une interface uniforme
* Avec un système de couche
* Un code à la demande (optionnel).

# Service

Un service consiste en une fonction ou fonctionnalité bien définie. C'est aussi un composant autonome qui ne dépend d’aucun contexte ou service externe.

Un service est une entité de traitement qui respecte les caractéristiques suivantes :

* Large granularité : les opérations proposées par un service encapsulent plusieurs fonctions et opèrent sur un périmètre de données large.
* Interface : un service peut implémenter plusieurs interfaces, et aussi plusieurs services peuvent implémenter une interface commune.
* Localisable : avant d’appeler un service, il faudra le trouver.
* Instance unique : un service a une instance unique. Il correspond au design pattern Singleton.
* Couplage faible : les services sont connectés aux clients et aux autres services via des standards. Ces standards assurent le découplage, c'est-à-dire la réduction des dépendances. Ces standards sont des documents XML comme dans les web services.
* Synchrone ou asynchrone : attente de réponse après l’invocation d'un service ou non.

# Service web

### Définition

Un service Web est un service basé sur le modèle SOA accessible au moyen d'Internet, qui utilise un système de messagerie standard XML, et n'est lié à aucun système d'exploitation ou langage de programmation. Le service Web est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes.

### Les principaux avantages d'un service Web :

* Son interface décrite d'une manière interprétable par les machines, qui permet aux applications clientes d'accéder aux services de manière automatique ;
* Son utilisation de langages et protocoles indépendants des plates-formes d'implantation, qui renforcent l'interopérabilité entre services ;
* Son utilisation des normes actuelles du Web, qui permettent la réalisation des interactions faiblement couplées et favorisent aussi l'interopérabilité.

### 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.

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. :)

### Les caractéristiques d'un service 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.

### 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.

Le service Web utilise WSDL (un métalangage XML) comme langage de description, un annuaire UDDI pour en permettre la localisation et un protocole de transport comme HTTP dans l'architecture REST ou SOAP pour l'architecture SOA.

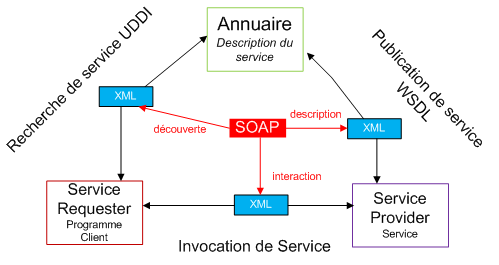
WSDL (Web Services Description Language) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service Web tels 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.

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.

REST (Representational State Transfer) est une architecture de services Web. REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

### Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant :



**Service provider service**

Le fournisseur de service met en application le service Web et le rend disponible sur Internet.

**Service requester programme client**

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP).

**Annuaire service registry**

Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver. Les interactions entre ces trois acteurs suivent plusieurs étapes :

* La publication du service : le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
* La recherche du service : le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.
* L'invocation du service : une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

# SOA

## Définition

Une architecture orientée service se conforme à divers principes de gestion des services influençant directement le comportement intrinsèque d’une solution logicielle et le style de sa conception :

* L’encapsulation des services.
* Le faible couplage des services avec la maintenance d’une relation réduisant les dépendances.
* Le contrat de service adhère à un accord de communication, collectivement défini avec un ou plusieurs documents de description.
* L’abstraction des services dissimulant la logique du service à l’extérieur.
* La réutilisation des services partageant la logique entre plusieurs services avec l’intention de promouvoir la réutilisation.
* La composition des services.
* L’autonomie des services.
* L’optimisation des services.
* La découverte des services depuis leur description extérieure.

L’architecture orientée service représente un moyen technique d’intégration des divers systèmes d’information de l’entreprise considérant chaque ressource informatique comme un service. Elle permet de construire les buildings blocks qui composeront l'urbanisme du système d'information.

La notion d’interface est importante dans l’approche orientée service. En effet, elle représente le point d’entrée unique vers les fonctionnalités de la solution logicielle et assure la communication grâce à l’échange de messages. Chaque message est porteur de la sémantique particulière à la solution logicielle. De plus ce message est rédigé dans un langage compréhensible aux deux parties en présence. Les services proposés d’une architecture agile décrivent la structure des messages qu’ils attendent du client.

L’architecture orientée service est une solution logicielle distribuée. Elle propose un mécanisme d’échange de messages sécurisé entre les systèmes d’informations sous-jacents en employant des protocoles de communication standardisés. Cette approche offre à l’architecture une opportunité d’ouverture sur un large éventail de solutions logicielles existantes.

# Microservice

L’architecture microservices est une approche où une application est décomposée en plusieurs petits services. Et ces services sont souvent spécialisés dans une seule distincte tâche.

L’architecture répond aux problèmes rencontrées par les applications monolithes.

Une application monolithe est celle qui a pour ambition de traiter toutes les demandes possibles et répondre à un maximum de cas d’usage.

Avec le temps, elle a tendance à grossir car elle intègre toujours plus de nouvelles fonctionnalités mais elle supprime rarement les anciennes fonctionnalités même lorsque celles-ci deviennent inutiles ou obsolètes. Et avec le temps les différentes briques développent des interdépendances entre elles. Et avec ça, la quantité de code augmente et ce code devient de plus en plus complexe.

L’application devient difficile à maîtriser et à faire évoluer.

Enfin, plus un projet est gros, plus il est critique pour l’entreprise, et moins on va prendre de risques pour tester des nouveautés, on va donc préférer la stabilité à l’innovation.

La réponse est de découper l’application en différents modules fonctionnels, en microservices.

Chaque microservice exécute une partie spécifique et unique de l’application. Et ces services peuvent être accédés par le client via l’API du microservice correspondant. Le but de l’architecture microservices est de redonner un maximum d’agilité et d’évolutivité à une application.

Côté technique l’architecture microservices est très souvent couplé à la technologie des conteneurs, et notamment Docker.

Quant aux conteneurs, Le principe est d’avoir un conteneur par unité de microservices. Ainsi, Chaque microservice a son propre environnement d’exécution pour héberger son code. Avec les conteneurs, Il est donc très facile en fonction du besoin d’adapter le nombre de microservices, à la hausse comme à la baisse.

SI un service est plus demandé lors d’un pic de charge, il suffit de créer plus de conteneurs fournissant ce microservice. A contrario, quand la demande de ce service diminue, il suffit de ne garder que le minimum de microservices pour couvrir la demande. Dans l’architecture microservices, il suffit juste d’adapter le nombre de ce microservice au besoin alors que dans les applications monolithes, il aurait fallu modifier la totalité de l’application à cause des interdépendances.

Et c’est en étant couplé avec l’utilisation des conteneurs que chaque microservice devient autonome vis à vis des autres microservices un microservice peut être alors modifiée, supprimée ou déployée sans impacter les autres microservices.

Ce qui veut dire qu’avec une application découpée en microservices, il est plus simple de cibler les différentes parties impactées d’une application lors d’une évolution. Ce qui facilite la modification des parties concernées sans risque pour le reste de l’application non impliqué.

Du coup, les mises à jour et nouvelles fonctionnalités peuvent être déployées beaucoup plus facilement et rapidement, rendant le déploiement continu possible.

Concernant l’organisation humaine, la taille des projets est limitée à une équipe de quelques personnes. Chaque équipe est autonome et va donc gérer son organisation et sa base de code.

Une équipe est alors libre des choix techniques en fonction de leurs besoins propres.

Terminons par des points d’attention à propos des microservices

Et tout ce que l’on vient de voir doit permettre aux entreprise d’avoir un atout très important et c’est d’ailleurs ça qui va pousser à l’adoption des microservices.

Cette adoption permet d’avoir un temps au marché plus rapide. Aujourd’hui un temps au marché rapide est devenu un facteur stratégique majeur, il permet à une entreprise de prendre un avantage concurrentiel décisif sur ces concurrents en intégrant la toute dernière innovation dans ses produits ou d’adapter ses produits aux nouveaux besoins du marché.

Comme on l’a vu tout à l’heure les microservices permettent justement de déployer plus rapidement et facilement. Ce qui veut dire être capable de s’adapter plus rapidement à la demande du marché et aux nouveautés.

Par contre et il est important de souligner :

Plus les itérations sont nombreuses et rapides (notamment dans le cadre d’un déploiement continu), plus cela va demander aux équipes de se synchroniser rapidement.

Cela va demander une organisation mature pour la communication et coordination entre les équipes et une synchronisation de leurs priorités et objectifs pour qu’ils soient alignés avec les enjeux business