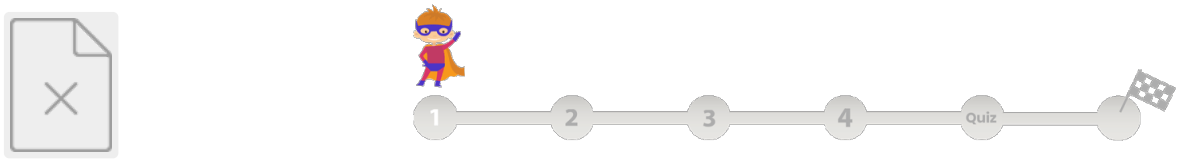
1 - Premières notions d'API




# 1. Premières Notions d’API

## 1.1 Définition d’une Interface de Programmation Applicative

En informatique, une **interface de programmation applicative** ou **API** (pour "**Application Programming Interface**") désigne une **interface structurée** permettant d'**offrir des services** à un autre logiciel.

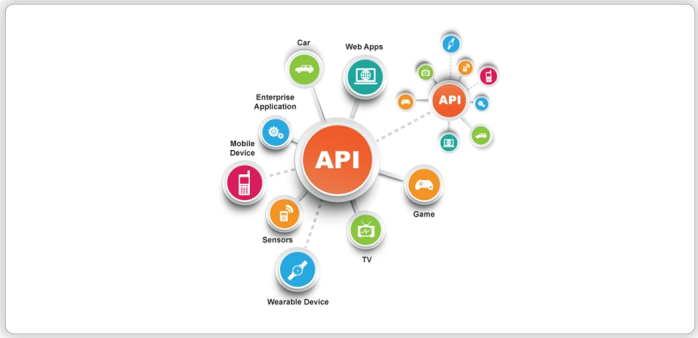



*Figure 1 : Exemple d'utilisation d'API par d'autres services*

De nos jours, vous trouverez de plus en plus d’entreprise qui exposent leur **Système d’Information à travers des API publiques ou privées**. Cette dernière approche est très rare, car toutes les API dignes de ce nom doivent supporter un **mécanisme d’authentification** qui permettent de **sécuriser les communications** inter-API.

Les API sont déployées avant tout afin **d’exposer des ressources internes au système.**

Vous pourrez donc **déporter le développement d'une interface graphique** en faisant appel à des UX/UI designer ou à d'autres métiers, car il y a peu de chance que vous soyez compétent sur **toutes les technologies**.

## 1.2 Émergence des architectures orientées web – WOA

Vous avez sûrement déjà entendu parler des services web ou **web service**s. L’API peut être vue comme **une évolution du service web**, pour certaines personnes c’est uniquement un renommage marketing, mais, selon moi, il existe une **légère différence** entre les deux approches.

Je considère que le terme de service web désigne une approche qui offre beaucoup de **souplesse d’implémentation**, car il n’y a pas vraiment de cadre de communication codifié. En contrepartie, si les services ne sont pas bien documentés, c’est presque impossible de communiquer avec. Il est très fréquent de trouver une équipe travaillant de chaque côté des web services afin de **limiter les problèmes de communication**.

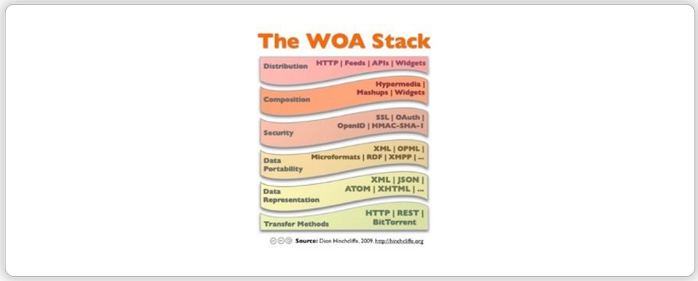



*Figure 2 : La stack technique WOA*

Les API quant à elles, font toujours référence à des **architectures codifiées**, il en existe principalement deux que vous retrouverez tout le temps : **REST** et **SOAP**.
Il faut également noter qu'une API n'est pas forcément une API web, mais c'est malheureusement souvent **un défaut de langage**. Lorsque vous entendrez parler d'API, vous serez confronté à **95% d**u temps à une **API web**.

**Quelle est la différence entre les deux ?**
C'est très simple, une **API web** utilise un **serveur web**, qui utilisera le **protocole de communication HTTP ou HTTPS**.

Une guerre est alors apparue, opposant les géants du Web qui utilisaient massivement les API "REST", aux entreprises dites classiques qui utilisent des architectures SOA principalement basées sur le protocole SOAP. Il faut attendre le début des années 2000, pour voir la communauté se scinder en deux suite aux divergences de chaque camp.

C'est ainsi qu'est né le terme de **WOA**, désignant **les architectures orientées Web**, car après tout, les **APIs REST** sont une forme **d'architecture orientée service, mais avec la particularité d'utiliser le protocole HTTP/HTTPS.**

## 1.3 Simple Object Access Protocol alias SOAP

Ce protocole a été imaginé par **Microsoft** pour résoudre les problèmes qu'ils rencontraient sur leurs services web. Il existait alors deux concurrents :

- **DCOM**

**Distributed Component Object Model** est une technique propriétaire de Microsoft qui permet la communication entre des composants logiciels distribués au sein d'un réseau informatique.

- **CORBA**

**Le Common Object Request Broker Architecture**, est une architecture logicielle utilisée pour le développement de composants et d'**object request broker (ORB)**. Ces composants, qui sont assemblés afin de construire des applications complètes, peuvent être écrits dans des langages de programmation distincts, être exécutés dans des processus séparés, voire être déployés sur des machines distinctes.

Le problème de ces deux approches provient du **format des données** qui transite entre les services : **le binaire**. Or, le protocole HTTP qui régit le web posait des problèmes de compatibilité avec le format binaire.

C'est dans cette optique que Microsoft a imaginé un protocole basé sur un **partage de messages au format XML**. Il est très utilisé dans tous les produits Microsoft, mais il ne fait pas forcément l'unanimité sur le web. Beaucoup lui reproche **la lourdeur du format de XML**. En effet, **le XML** est un **langage informatique** utilisant **un système de balise générique.** Chaque donnée est **encapsulée par une balise ouvrante et fermante** ce qui **alourdit considérablement les données utiles** en comparaison à un langage **comme le JSON**.

## 1.4 Representational state transfer alias REST

Vous l'aurez compris, le REST est devenue la Rolls des API web en quelques années, car il offre **une alternative plus simple**. Au lieu d'utiliser le XML pour faire effectuer une demande, le **REST** utilise **une simple requête http sur une URL**.

Pour effectuer des actions particulières, le REST peut utiliser **des méthodes HTTP spécifiques** telles que :

- **GET**
- **POST**
- **PUT**

- **PATCH**
- **DELETE**

Autre différence majeure avec SOAP, le **REST n'utilise pas le langage XML** pour fournir la réponse. Vous trouverez principalement des API REST qui utiliseront le **JavaScript Object Notation (JSON)** comme langage de sortie des données. Vous rencontrez peut-être également des données au **format CSV (Comma Separated Value)** ou encore **Really Simple Syndication (RSS)** même si c'est deux formats sont beaucoup moins répandus.

Vous pouvez donc obtenir **la sortie dont vous avez besoin** sous une **forme** qui est **facile à analyser** avec le **langage dont vous avez besoin** pour votre application.

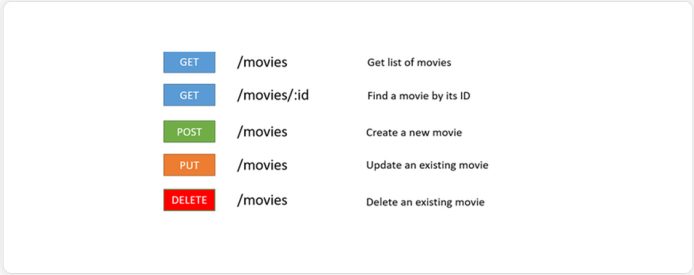



*Figure 3 : Exemple d'action REST*

Le principal attrait qui a permis au REST de **croître aussi vite** provient de sa courbe d'apprentissage très courte qui permet de rapidement le prendre en main. L'utilisation du JSON en tant que langage d'échange de données permet de le **rendre compatible nativement** avec un grand nombre de langages web ce qui n'est pas forcément le cas de l'XML.

**Exemple :**
Le gouvernement français a créé un bon nombre d'open API que vous pourrez utiliser pour consulter des informations mises à disposition publiquement.
Via votre navigateur, vous pouvez accéder directement à ces informations. Lorsque vous entrez une URL dans la barre de navigation de ce dernier, vous réalisez en réalité une requête GET sur le serveur web.

Vous pouvez essayer l'URL suivante :
https://www.data.gouv.fr/api/1/organizations/?page_size=1

Certes, le rendu n'est pas très lisible, mais le résultat en JSON que vous affiche le navigateur peut être facilement interprété.

Copiez le contenu dans un formateur de JSON comme le site https://jsonformatter.curiousconcept.com/ et vous verrez tout de suite le résultat lisible de la requête ci-dessus.

## 1.5 Bonne pratique des API

Lorsque vous souhaiterez concevoir une API, vous serez rapidement confrontés à la problématique du **« design d'API ».** Ce point constitue un enjeu majeur, dans la mesure où une API mal conçue sera vraisemblablement peu ou pas utilisée par les clients : les développeurs d'applications.

Pour être bien conçue, une API doit respecter les points suivants :

- **Respecter les standards HTTP**
- **S'inspirer des bonnes pratiques des Géants du Web, car les développeurs sont habitués à les consommer**
- **Être simple à utiliser, on parle de KISS « Keep it simple, stupid »**
- **Être versionnée**

Vous l'aurez sans doute compris, les **APIs** ont **fortement modifié le paysage informatique** au cours des **dix dernières années**, mais elles doivent leur **succès** en grande partie à un autre concept clé de la SOA, le **microservice**.