# FastAPI - Introduction aux APIs

30 minutes Beginner

Machine status

Ubuntu Server 18.04 LTS SSD Volume Type 64-bit x86

Online

34.245.135.23

Connect

Reset Stop

DataScientest • com

# APIs avec FastAPI

## 1. Introduction aux APIs

### a. Introduction

API signifie littéralement **Interface de Programmation d'Application** et est définie comme un ensemble de définitions de sous-programmes, de protocoles de communication et d'outils qui servent d'intermédiaire à la communication entre deux applications. Pour le Data Engineer et le Machine Learning Engineer, l'utilisation d'API va se révéler très utile lors des phases de déploiement et d'automatisation.

Une API est en réalité utilisée pour dé-corréler le service d'un client qui souhaite l'utiliser. Elle permet d'abstraire un fonctionnement complexe en donnant une liste de protocoles d'utilisation de ce service.

On peut penser à une API comme une prise de courant. Une prise de courant définit un protocole pour son utilisation: l'appareil que l'on souhaite brancher doit comporter deux broches, espacées d'un certain écart. Si cet écart est respecté alors la prise délivrera un courant de 220V à une fréquence de 50Hz. Du point de vue du client, c'est-à-dire de l'appareil électrique, la production de l'électricité n'a pas d'importance. Il peut s'agir de toutes sortes de centrale. De l'autre côté, c'est-à-dire du point de vue de la prise, on n'a pas besoin de s'adapter à l'appareil électrique: si le protocole est respecté, l'énergie nécessaire est fournie.

service.

Il existe donc de nombreux types d'APIs possibles. Nous allons en évoquer quelques unes:

- la librairie pandas peut en fait être vue comme une API. On retrouve un ensemble de règles qui permettent d'utiliser des fonctions complexes à partir d'une documentation simplifiée.
- l'API OpenWeatherMap est une API de données. À partir d'une requête HTTP, on reçoit des données météorologiques dans le lieu demandé, à la date demandée.
- l'API Google Cloud Translate permet d'utiliser la fonction de traduction de Google pour traduire facilement les textes que l'on utilise.

On peut voir que certaines APIs peuvent être très simples et simplement renvoyer certaines données alors que d'autres peuvent cacher l'utilisation de fonctions très complètes. On remarquera que dans le cas des APIs Google Translate et OpenWeatherMap, il faut s'identifier et éventuellement souscrire à un plan pour pouvoir accéder au service.

C'est en effet un des avantages très importants des APIs. En ré-écrivant clairement les règles d'utilisation d'une API, nous pouvons introduire l'utilisation de token d'authentification. On peut alors donner certains droits à certains utilisateurs, définir des limites dans l'utilisation de l'API et éventuellement vendre l'accès à l'API.

De plus, ces APIs s'inscrivent dans la philosophie de l'architecture micro-service, dans laquelle les composants d'un système doivent être les plus petits possibles et les plus isolés pour pouvoir être facilement mis à jour sans changer l'architecture globale du système.

Imaginons, par exemple, une entreprise qui propose des vélos à la location. Cette entreprise stocke ses données sur la disponibilité des vélos à la location dans une base de données. Cette base de données est interrogée régulièrement par différents outils: le site internet de l'entreprise, l'application mobile, qui permettent tous deux de réserver des données, par les bornes de location qui notifient du retour des vélos ainsi que par un dashboard interne. Si l'entreprise n'emploie pas une API, alors il faudra mettre à jour tous les outils lorsque l'entreprise sera amenée à changer sa base de données (par exemple pour permettre une utilisation distribuée de celle-ci). Avec une API, il suffit de mettre à jour l'API qui jouera le rôle d'intermédiaire. De plus, on pourra développer certaines fonctionnalités qui permettront à des acteurs extérieurs d'accéder à certaines données (on peut penser aux autorités locales qui pourraient utiliser ces données pour faire un suivi du service).

De la même façon, si on construit un algorithme d'analyse de sentiment performant, c'est-à-dire qui arrive à prédire correctement le sentiment positif ou négatif d'une phrase ou d'un texte, on pourra cacher le fonctionnement de cet algorithme par une API. Ainsi, nous pourrons construire des abonnements avec diverses limites d'utilisation de notre algorithme. De plus, le jour où l'algorithme est remplacé par un autre plus puissant, son utilisation ne change pas pour l'utilisateur final.

l'interface.

Certaines entreprises reposent fortement sur ce principe d'APIs. Un exemple très célèbre est Amazon: en 2002, Jeff Bezos aurait demandé à toutes ces équipes de toujours créer des APIs à chaque fois qu'ils créent une fonctionnalité ou qu'ils exposent de la données.

On peut voir que les APIs sont donc très intéressantes d'un point de vue architectural mais aussi d'un point de vue plus métier car elles offrent des perspectives économiques.

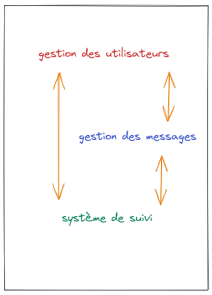
Enfin, il faut noter que pour garantir l'utilisation de ces APIs, les protocoles de communications doivent être très clairs et bien documentés sur les paramètres à fournir ainsi que sur leurs effets.

## b. Architecture micro services

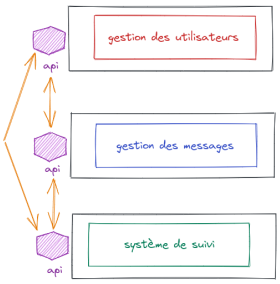
Avec des applications de plus en plus complexes, les organisations se tournent de plus en plus vers des architectures micro-services, c'est-à-dire des architectures dans lesquelles les services sont très découplés.

Prenons l'exemple d'un réseau social. L'entreprise a besoin d'un système d'identification et de gestion des utilisateurs, un système qui permet de créer et poster des messages, de systèmes de monitoring, etc... Une solution classique consisterait à créer une sorte de script très complexe qui permet de tout gérer d'un coup: de la réception des données jusqu'à leur stockage en passant par l'implémentation des droits. On parle alors d'architecture monolithique.

L'autre approche consisterait à créer une architecture de stockage. On va séparer la gestion des utilisateurs de la gestion des posts, etc... Ces différents services pourront communiquer entre eux grâce à des APIs.




architecture monolithique




architecture micro-service

L'architecture micro-service présente de nombreux avantages:

- les différents services sont **autonomes**, peuvent être développés par différentes équipes, dans différents langages puisqu'elles peuvent communiquer entre elles via leur API.
- chaque service est **spécialisé** dans la résolution d'une tâche unique ce qui permet de plus facilement identifier des problèmes ou de faire évoluer un service
- comme chaque service est une petite unité indépendante du reste du système et spécialisée, on peut facilement modifier ce service donnant ainsi des cycles de développement plus rapides et plus agiles.
- si certains services requièrent des ressources (stockage, calcul) plus importantes, ou si leur besoin en ressources évolue au fil du temps,

d'autres projets. L'API permet en effet d'avoir une documentation précise de l'utilisation du service qui permet à d'autres équipes d'inclure le service dans leurs applications.
- on peut enfin facilement identifier d'où viennent les pannes dans un tel système et réparer ces pannes rapidement ou même de modifier le service pour que la panne ne réapparaisse plus.

Validated