# Lire une documentation

Comment trouver efficacement une information

## Table des matières

1. Découvrir la documentation ..... 2
2. Qu'est-ce que je veux faire ? ..... 2
3. Repérer les endpoints ..... 2
4. Repérer la méthode HTTP ..... 3
5. Repérer les paramètres ..... 3
6. Repérer les headers ..... 3
7. Repérer la structure des résultats ..... 4

## 1. Découvrir la documentation

Chaque API doit être livrée avec une **description** de ses fonctionnalités. Je ne parle pas ici d'une description textuelle du style « L'API fait ceci et cela », je parle d'une description **standardisée** qui répond à deux problématiques simples :

- La documentation doit être la plus **concise** possible
- La documentation doit être la plus **limpide** possible

Oui oui, les deux en même temps.

Cependant, une part importante de cette **limpidité**, de cette **clarté**, vient du développeur en train de la lire !

*En effet, lire une documentation ce n'est pas comme lire un livre, il y a une **méthodologie de recherche** à appliquer afin de se mettre en accord avec la structure de la documentation.*

Prenons l'exemple de l'API gratuite et sans clé d'accès : https://jsonplaceholder.typicode.com/

## 2. Qu'est-ce que je veux faire ?

Je sais, ça peut paraître bête mais le meilleur moyen de trouver une réponse efficacement est de bien connaître sa problématique. Posez-vous donc la question suivante : De quoi ai-je besoin ? Récupérer des utilisateurs ? Ok, quelles données précisément ? Sous quel format ?

Plus vous aurez de certitudes par rapport à votre besoin, plus vous serez à même de juger de la pertinence de ce que vous trouverez.

*Une API, quelle qu'elle soit, est toujours organisée par **actions***

## 3. Repérer les endpoints

Maintenant qu'on sait ce qu'on cherche, il va falloir repérer dans la documentation quelles sont les différentes options à notre portée. Ces options sont généralement représentées par différents **endpoints**, que l'ont peut retrouver dans la section « **Endpoints »** ou alors **« Routes »**, ou alors « **API Reference »**, ou alors **« Resources »**…

Dans toutes les documentations d'API du monde, il y aura toujours une section dédiée aux **endpoints**. Sans ça, l'API est inutilisable.

Quelques règles concernants les **endpoints** :

- Quand un endpoint est au pluriel, il renvoie **la plupart du temps** un TABLEAU DE RÉSULTATS MULTIPLES.
- Quand on tombe sur des endpoints contenant des {}, il s'agit **la plupart du temps** de paramètres **dynamiques**.

Exemples :

- /users
- /albums/{id}/photos
- /posts/{postId}/comments

## 4. Repérer la méthode HTTP

Chaque endpoint d'une API s'appelle avec une méthode HTTP dédiée, le plus souvent en rapport avec sa fonction même si il se peut que le même endpoint puisse être appelé avec deux méthodes différentes.

Par exemple /users/34 appelé avec GET pourra renvoyer les données de l'utilisateur 34, mais une requête PUT envoyée au même endpoint sera une **mise à jour** de l'utilisateur 34.

| **Méthode** | **Signification** |
|---|---|
| GET | Lire / récupérer |
| POST | Créer |
| PUT / PATCH | Mettre à jour |
| DELETE | Supprimer |

## 5. Repérer les paramètres

Certains endpoints peuvent avoir besoin de **paramètres** pour fonctionner, mais ce n'est pas toujours le cas. Il existe plusieurs façons de faire passer les paramètres :

- Soit dans le chemin de l'endpoint, comme ici : /posts/57
- Soit dans les paramètres de l'URL appelée : /posts?id=57

Pour les données envoyées en POST, on ne va pas appeler ça des paramètres. On va plutôt faire référence au **Request body**, ou alors au **payload**, des mots-clés que vous pouvez chercher dans l'API.

## 6. Repérer les headers

Certains headers peuvent être nécessaires pour faire fonctionner une API. Comme par exemple une authentification que l'on mettra alors traditionellement dans un header Authorization

## 7. Repérer la structure des résultats

Une fois qu'on sait comment appeler notre endpoint, il faut que l'on soit capable de comprendre sa réponse. Pour cela on va chercher les **responses** ou les **status code** que va nous renvoyer l'API. Par exemple :

- 200 OK
- 201 Created
- 400 Bad Request
- 401 Unauthorized

Pour les statuts, ou alors


```
{
  "id": 123,
  "title": "Hello"
}
```


Pour la JSON response.

*On doit savoir ce qu'on envoie et ce qu'on récupère, ce qui vient en entrée et en sortie de l'API*