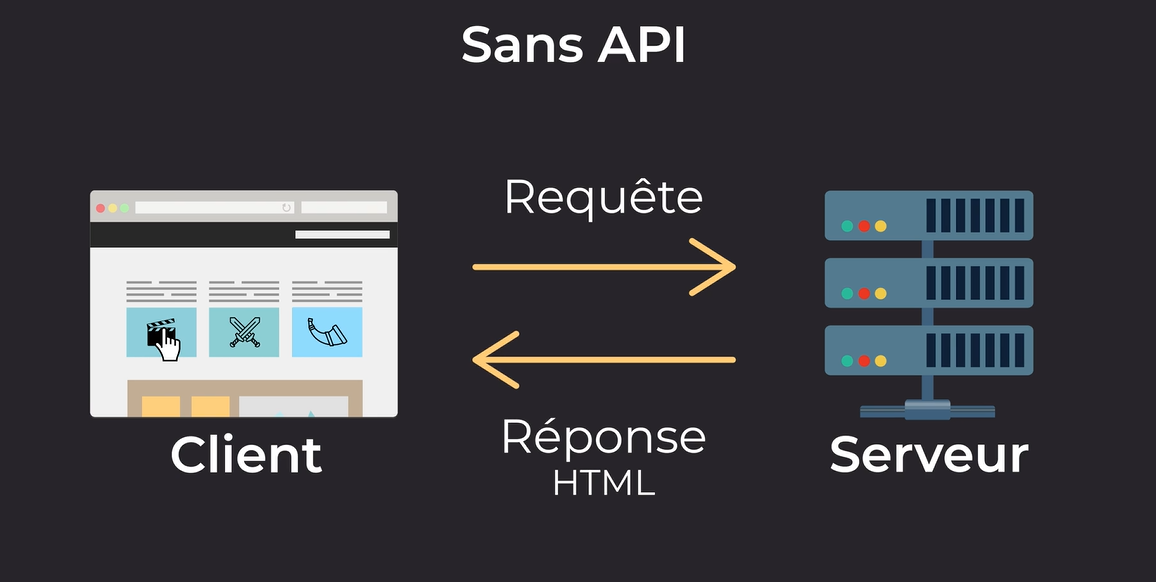
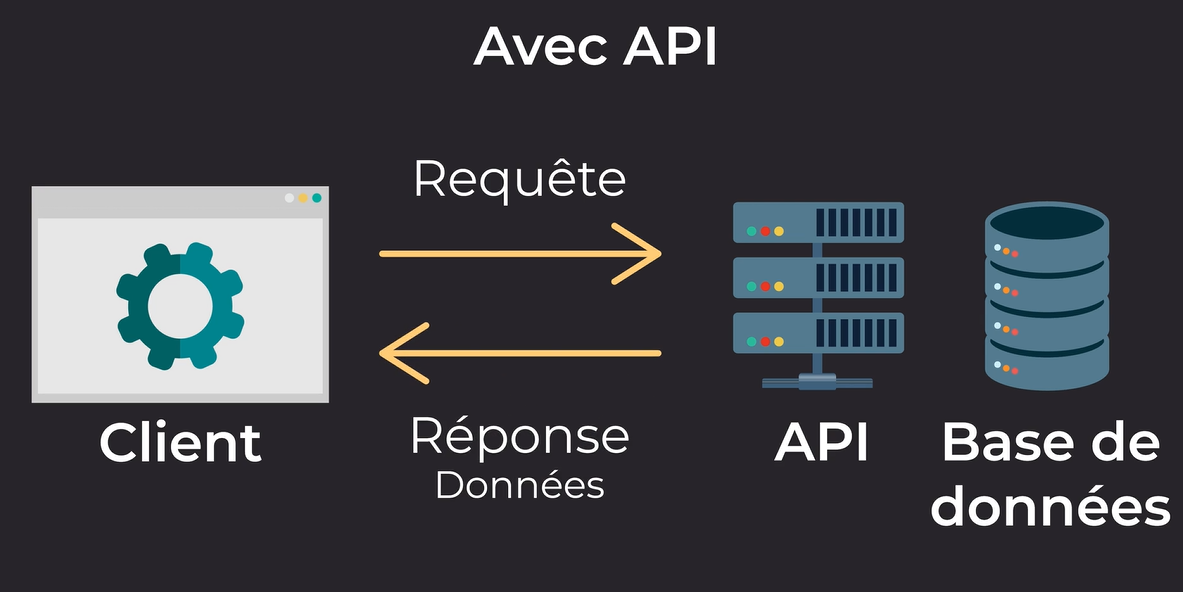
**Api:**

Api = application programming interface – interface de programmation d’application. Son but est de faciliter la communication entre 2 logiciels en servant d’intermédiaire peu importe le support.





L’api renvoi une donnée qui peut être interprété par d’autre composants de l’application.

Api permet à des services d’interagir entre eux.

Les API créent des méthodes **standardisées**et **réutilisables**qui permettent aux développeurs d’accéder à des données spécifiques lors de la construction d’applications.

Prenons un exemple. Quand vous sortez manger, le menu du restaurant offre une grande quantité d’options déjà prédéterminées. Cela vous simplifie la tâche car vous savez ce que vous pouvez commander, et donc obtenir plus rapidement votre plat. Cela donne également une meilleure compréhension de ce que vous voulez pour le chef. Finalement, vous n’avez qu’à demander à la serveuse (API) un plat du menu qu’elle transmettra en cuisine, la cuisine prépare votre plat, le remet à la serveuse qui vous le ramène.

Il existe 2 types d’API : privé et public.

Les API **privées** garantissent que les personnes en dehors de votre entreprise ou de votre application n’ont pas accès aux données disponibles de votre base de données.

Une API **privée** permet *uniquement*aux utilisateurs **autorisés** au sein de votre entreprise ou de votre application d'utiliser l’API qui peut accéder à la base de données.

Contrairement aux API privées, les API que l’on appelle **publiques** sont utilisables par d’autres personnes, qu’elles soient sur votre application ou non. Elles permettent aux développeurs de récolter les données d’une autre application pour améliorer ou enrichir leurs propres projets sans autorisation stricte.

Imaginons que vous vouliez construire un site web qui répertorie les conditions météo des stations de ski. Plutôt que de collecter vos propres données météorologiques, vous pouvez utiliser une [API de météo](https://openweathermap.org/api) et y trouver vos données !

**En résumé**

* Les API permettent de communiquer des données.
* Elles permettent la communication entre différents composants de votre application **et** entre votre application et d’autres développeurs, par l’utilisation de**requêtes** et de **réponses**.
* Elles donnent un moyen d’accès aux données de façon réutilisable et standardisée.
* Les développeurs peuvent utiliser des API publiques et privées.
* Les API publiques sont utilisables par tous sans restriction.
* Les API privées sont utilisables seulement par ceux qui ont un accès et y sont autorisés

En développement web, il existe 2 types d’API :

API SOAP: simple object access protocol.

API **REST** (restful): Representation State Transfer- representation d’un transfert d’état.

REST : ensemble de normes et de styles d’architecture pour les APIs  qui structurent la façon de communiquer les données entre votre application et le reste du monde, ou entre différents composants de votre application..

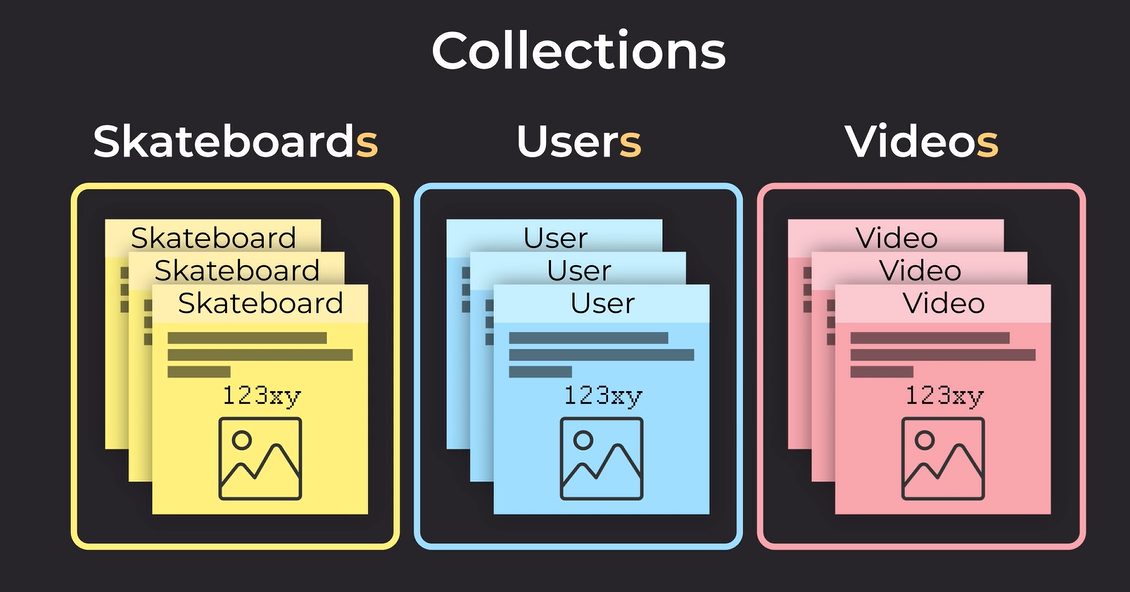
Une api est dite de type rest API REST si :

* Client\_server seperation.
* Stateless – sans état, ce qui signifie que le serveur ne sauvegarde aucune des requêtes ou réponses précédentes.
* Cacheable (avoir un système de cache de données).
* Uniforme interface.
* Layered system (avoir un système de couche).
* Code on demand architecture.

Cacheable : La réponse doit contenir l’information sur la capacité ou non du client de mettre les données **en cache**, ou de les sauvegarder. Si les données **peuvent être mises en cache**, la réponse doit être accompagnée d’un numéro de version. Ainsi, si votre utilisateur formule deux fois la même requête (c’est-à-dire s’il veut revoir une page) et que les informations n’ont pas changé, alors votre serveur n’a pas besoin de rechercher les informations une deuxième fois. À la place, le client peut simplement mettre en cache les données la première fois, puis charger à nouveau les mêmes données la seconde fois.

En résumé

* Toutes les API ne sont pas RESTful et les API REST ont des lignes directrices architecturales spécifiques.
* Les avantages clés des API REST sont les suivants :
  + la séparation du client et du serveur, qui aide à scaler plus facilement les applications ;
  + le fait d’être stateless, ce qui rend les requêtes API très spécifiques et orientées vers le détail ;
  + la possibilité de mise en cache, qui permet aux clients de sauvegarder les données, et donc de ne pas devoir constamment faire des requêtes aux serveurs.
* SOAP est un autre type d’API, mais est plus utilisé dans les grandes entreprises.

Ressources API REST

Procédons étape par étape et déterminons ensemble les ressources, leurs informations supplémentaires et les collections.

Une image contenant texte

Description générée automatiquement

Quand un client achète un skateboard en utilisant votre application web, cela donne :

* Votre API envoie la requête du navigateur (le client) aux serveurs de l’application pour l’achat d’un skateboard ;
* La requête met à jour l’inventaire pour qu’il y ait un skateboard de moins ;
* La requête met à jour l’historique de commandes du client pour ajouter le skateboard à son historique d'achats.

Le **path (ou***chemin***)** que vous donnez à votre API lui permet de savoir exactement **où**se trouvent les données que vous voulez récupérer.

Si une ressource est l’objet qui stocke vos données, pour les récupérer vous allez avoir besoin d’un identifiant de ressource uniforme, ou ***URI***pour *Uniform Resource Identifier*. L’URI est le moyen d’identifier votre ressource, comme une étiquette.

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Toutes les URL sont des URI, mais toutes les URI ne sont pas des URL. L’URI permet d’identifier une ressource tandis que l’URL permet de la localiser.

Les données des API REST peuvent utiliser deux langages : XML et JSON. Si une API renvoie un set de données en XML ou en JSON, le contenu restera le même, mais la forme change. Le format de données est différent.

En**XML**, chaque élément de donnée à une balise ouvrante et une balise fermante qui peut également avoir des balises imbriquées :

Une image contenant texte

Description générée automatiquement

Le JSON

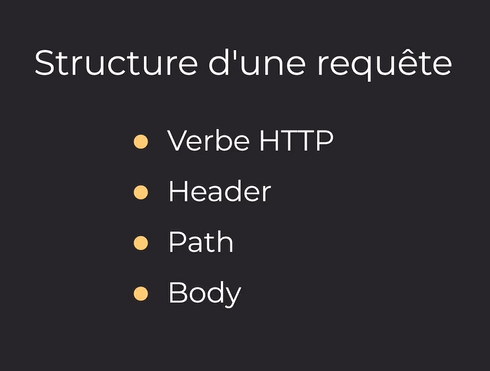
Le JSON stocke les données sous un format de clé-valeur avec comme clé le type de données, suivi de deux points : , suivi de la valeur de la donnée. Les données JSON sont entourées d'accolades {}, et chaque paire clé-valeur est envoyée comme chaîne de caractères avec des guillemets autour "".

Une image contenant texte

Description générée automatiquement

### En résumé

* Une ressource est un objet de type nominal utilisé pour sauvegarder des données dans une API.
* Une ressource peut contenir des informations supplémentaires.
* Les ressources sont regroupées en collection et sont nommées au pluriel.
* Vous pouvez accéder aux ressources dans les API avec des URI.
* Les données REST peuvent être en langage JSON ou XML, mais le JSON est le plus courant.



Verbe http : différents types d’actions qu’on peut appliquer à notre requête.

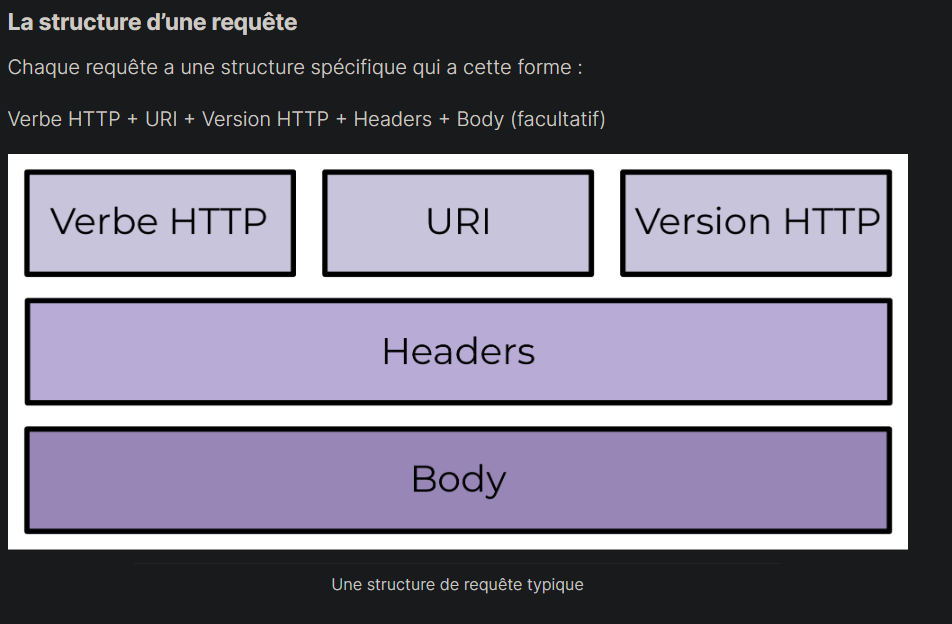


Une image contenant texte

Description générée automatiquement

Une API REST implique l’envoi de **requêtes**du client à l’API, qui passe la requête au serveur, l’API récupère la **réponse** et la renvoie enfin au client. Dans ce chapitre, nous allons voir comment formuler ces requêtes grâce à [Postman](https://www.postman.com/).

Cette interface graphique est utilisée par de nombreux développeurs. Elle facilite la construction de nos requêtes. C’est donc l’outil idéal pour tester des API sans devoir utiliser de code.



Les **verbes HTTP** correspondent à différents types d’actions que vous pouvez accomplir avec votre requête. Ceux que vous rencontrerez le plus couramment sont **GET**(obtenir),**PUT** (mettre),**POST**(publier), et **DELETE**(supprimer).

Une image contenant texte

Description générée automatiquement

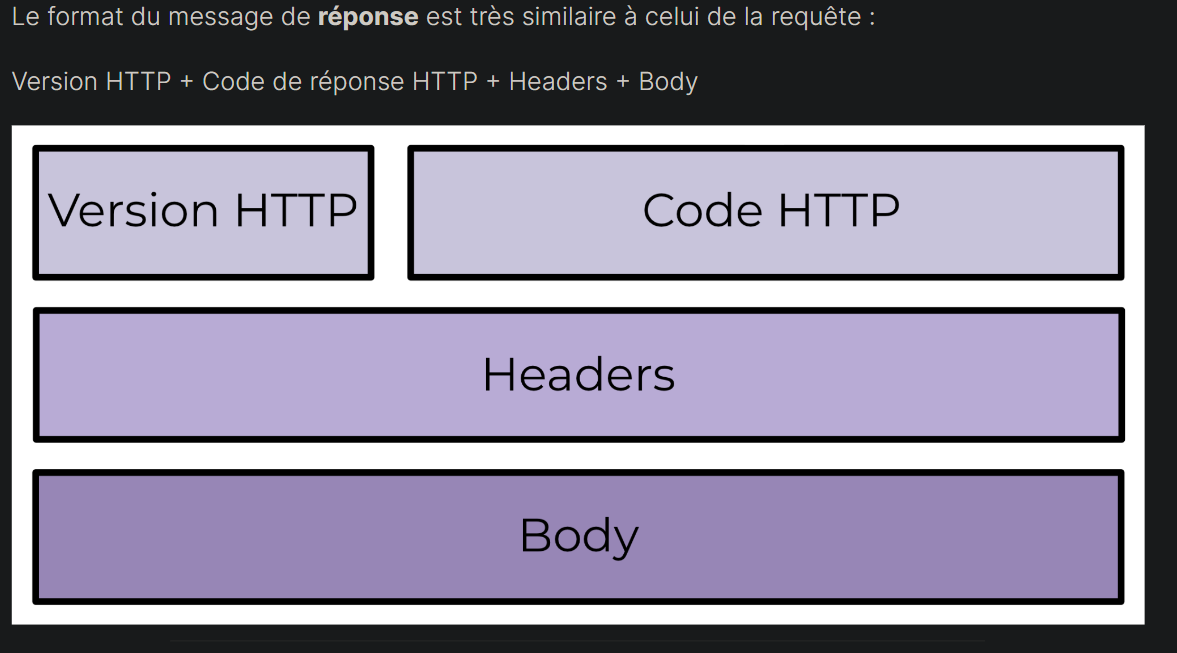
Les différentes options pour les headers :

<https://developer.mozilla.org/fr/docs/Web/HTTP/Headers>

**Le body**

Pour finir, parlons du body ! Pour formuler une requête, il n’est utilisé qu’avec PUT (mise à jour) ou POST (création). Il contient les données réelles de la ressource que vous essayez de créer ou de mettre à jour. Les données sont envoyées sous format JSON.

Prenons un exemple ! Vous voulez ajouter ou mettre à jour les informations d’un utilisateur ; vous ajouterez donc les détails de l’utilisateur (prénom, nom , adresse…) dans le body, en JSON.



Code de réponse http.

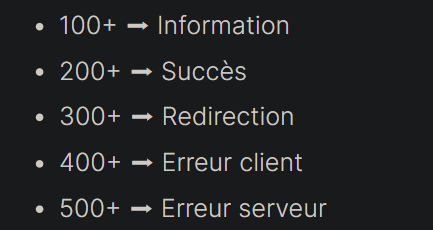
Une image contenant texte

Description générée automatiquement

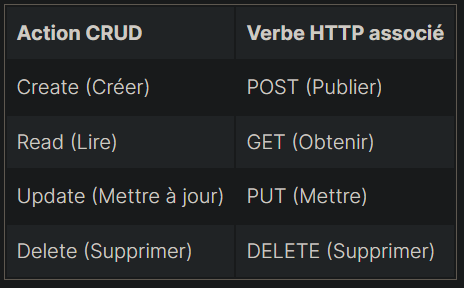
200 : requête réussie.

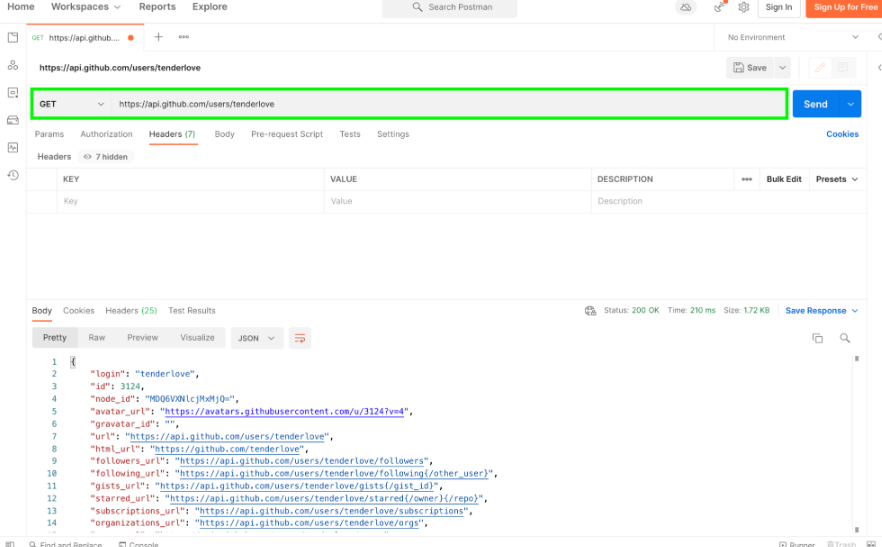
404 : échec.

 En général, les règles de base pour les codes de réponse HTTP sont les suivantes :

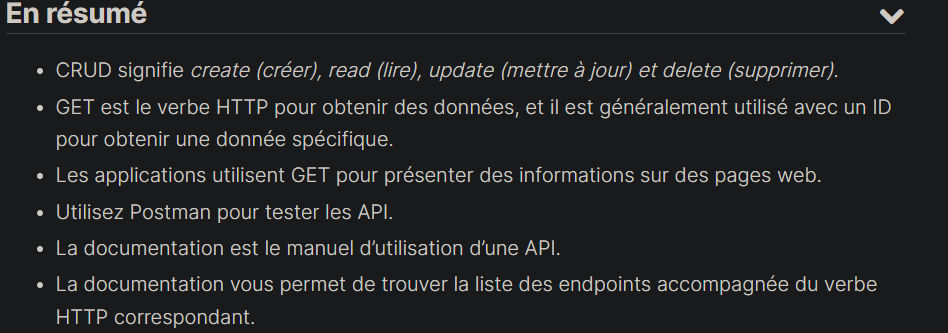


CRUD :





La documentation d’une API ressemble à un manuel d’utilisation très très détaillé. Étant donné que chaque API est différente, vous ne sauriez pas les utiliser sans une documentation claire et précise.



Endpoint = URL / URI qui fait partie d’une api.

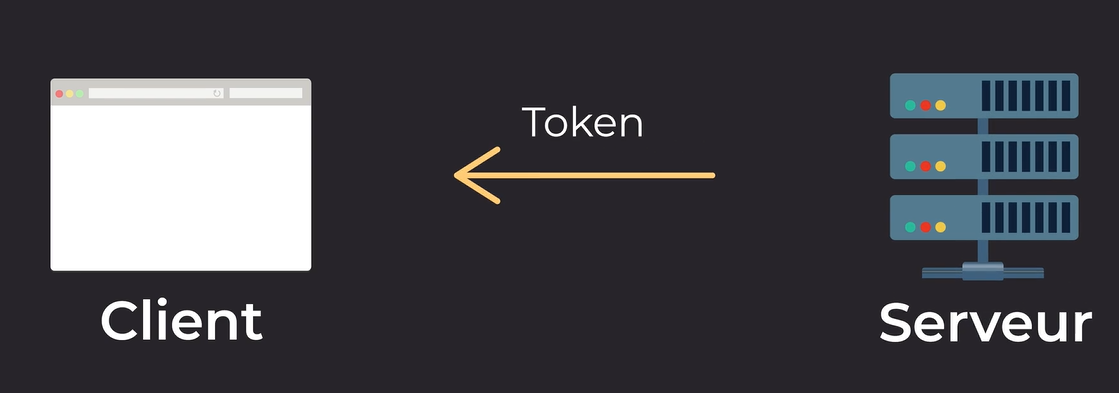
*La sécurité des données et l’authentification :*

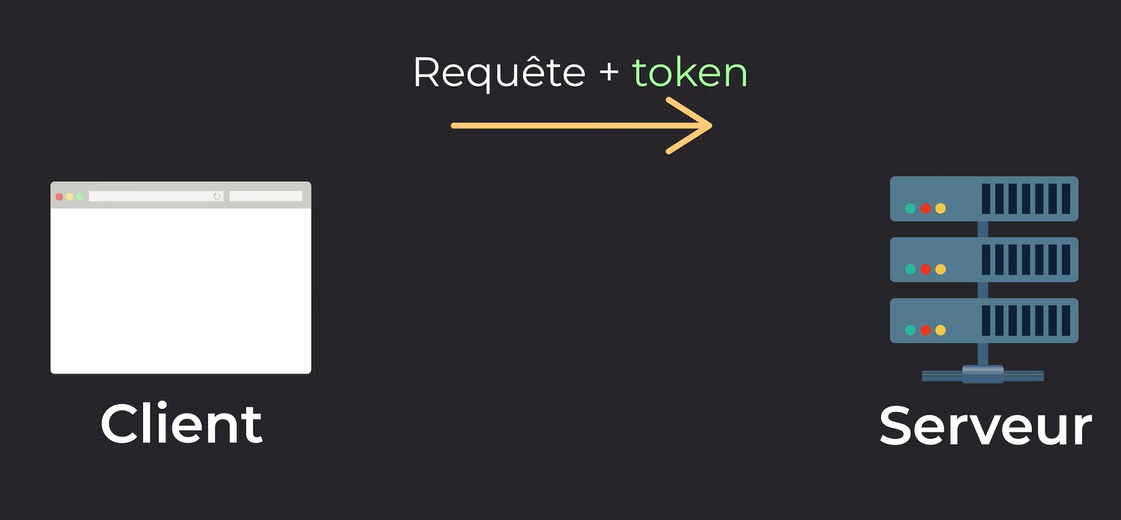
Authentification est le fait d’autoriser ou non un utilisateur d’accéder/consulter/modifier les données.

Comment ?

On signe les requêtes faites par un utilisateurs par un TOKEN









Une api est sécurisée si : elle met en place un système d’authentification/ d’autorisation, d’encodage de donnée, et qu’elle est mise à jour régulièrement.

POST / CREATE

Une image contenant texte

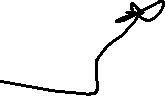
Description générée automatiquement

Une image contenant texte

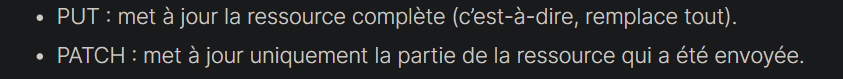
Description générée automatiquement

Une image contenant texte

Description générée automatiquement



PUT/UPDATE



Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

DELETE / DELETE

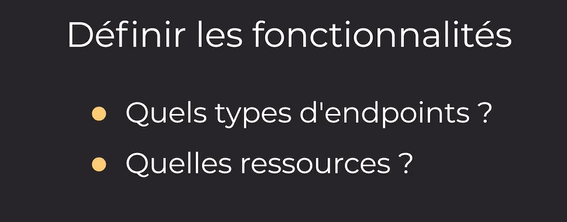
Une image contenant texte

Description générée automatiquement

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

Créer votre API REST



Une image contenant texte

Description générée automatiquement

Une documentation doit contenir :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Les questions à se poser pour créer une bonne API :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

lorsque vous effectuez une recherche Google, il n’est pas rare d’obtenir des millions de résultats ; du coup, Google ne renvoie qu’une page des résultats les plus haut classés, et vous pouvez parcourir le reste pour en voir plus.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement