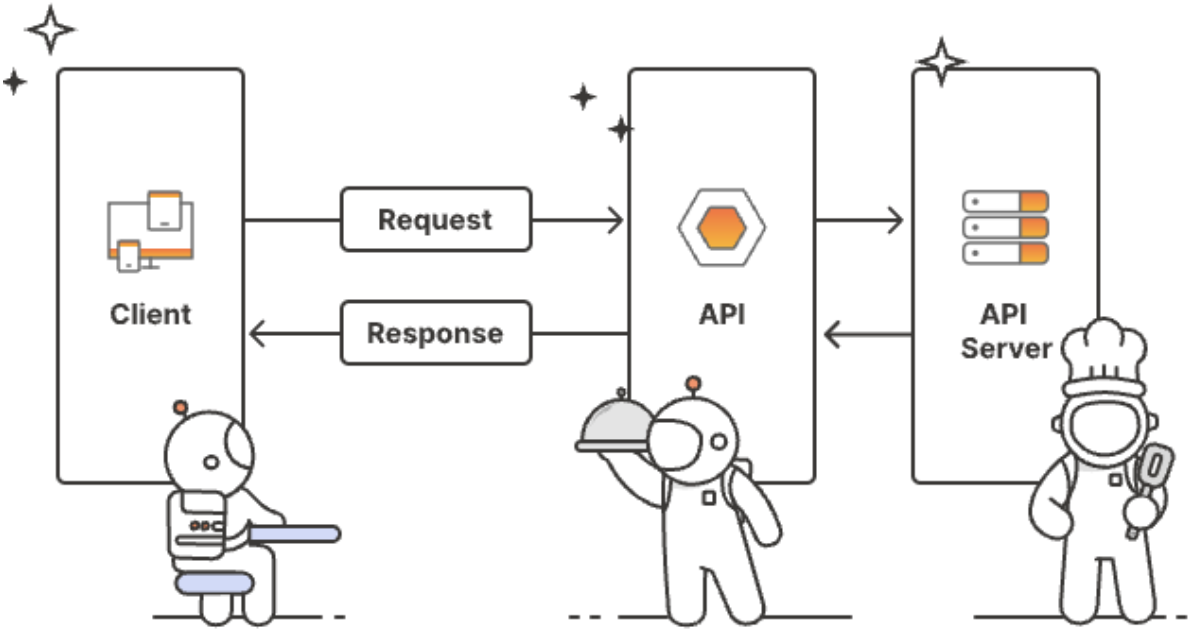
API REST

API

Une API (Application Programming Interface) est un ensemble de règles, de protocoles et d'outils qui permettent à différentes applications ou systèmes de communiquer entre eux.



Points clés sur les API

> Les API définissent comment les composants logiciels doivent interagir, **facilitant ainsi l'intégration** de différentes technologies.

> Elles permettent aux développeurs **d'accéder à des fonctionnalités** ou à des données d'autres applications sans avoir à comprendre leur fonctionnement interne.

> Dans le contexte du web, les API sont souvent utilisées pour permettre à des applications de **communiquer avec des serveurs** et **d'échanger des données**.

Qu'est-ce qu'une API REST ?

Une API REST (Representational State Transfer) est un style d'architecture pour la conception d'applications en réseau. Elle définit un ensemble de contraintes pour créer des services web scalables et interopérables.

interopérables ??? 😉

```
Interopérable signifie...
capable de fonctionner et d'interagir avec d'autres systèmes
ou applications sans restriction particulière.

ça implique que différents services web
peuvent communiquer et échanger des données facilement,
indépendamment de leur plateforme ou technologie sous-jacente
```

- Architecture client-serveur

💡 L'architecture client-serveur est un modèle de conception dans lequel les tâches sont réparties entre les fournisseurs de ressources ou de services (serveurs) et les demandeurs de services (clients). Dans le contexte d'une API REST, cette architecture permet une séparation claire des responsabilités, facilitant la scalabilité et la maintenance du système.

- stateless

💡 "Stateless" signifie que chaque requête du client vers le serveur doit contenir toutes les informations nécessaires pour comprendre et traiter la demande. Le serveur ne conserve aucune information sur l'état du client entre les requêtes. Cette approche améliore la scalabilité et la fiabilité du système.

- Mise en cache possible

💡 La **mise en cache possible** est une caractéristique des API REST qui permet de stocker temporairement des données fréquemment accédées, réduisant ainsi la charge sur le serveur et améliorant les performances du système. Cette technique optimise les temps de réponse et l'utilisation des ressources.

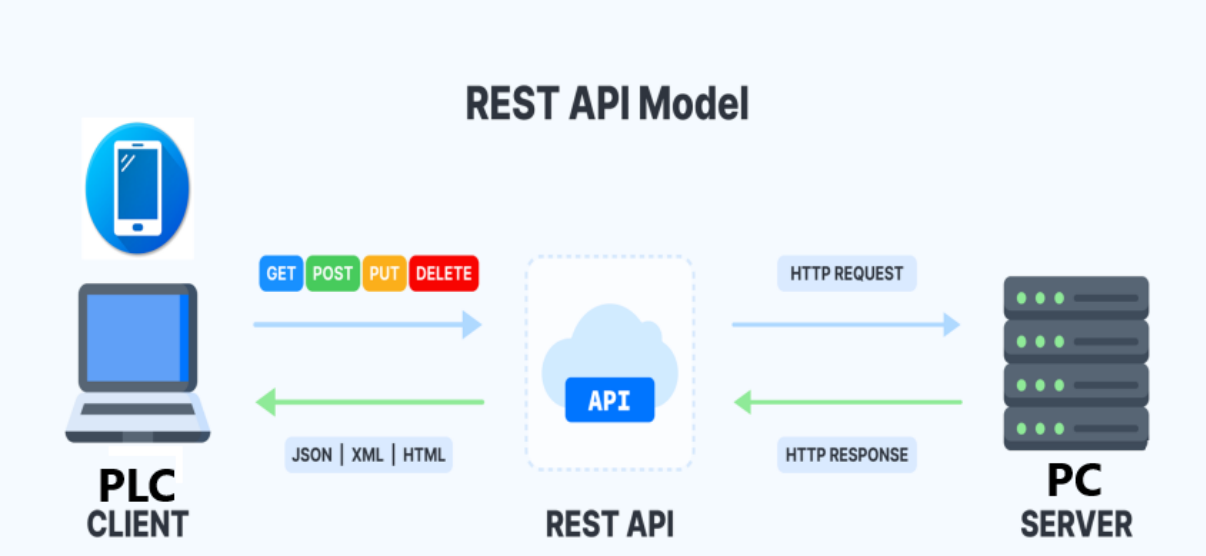
- Interface uniforme

💡 L'**interface uniforme** est un principe fondamental des API REST qui établit un ensemble standard de méthodes pour interagir avec les ressources. Cette approche assure une cohérence dans la conception et l'utilisation de l'API, simplifiant sa compréhension et son utilisation par les développeurs. De plus, elle favorise l'interopérabilité entre différents systèmes.

- Système en couches

> 💡 Le **système en couches** est un principe architectural des API REST qui structure l'application en couches hiérarchiques distinctes, chacune avec des responsabilités spécifiques. Cette approche améliore la modularité, la flexibilité et l'évolutivité du système. Elle permet également une meilleure séparation des préoccupations entre les différents composants de l'API.

Les API REST utilisent généralement le protocole HTTP pour les communications et échangent des données au format JSON ou XML.



ENDPOINT

Les méthodes GET, POST, PUT et DELETE sont des opérations HTTP fondamentales utilisées dans les API REST pour interagir avec les ressources.

- **GET** : Utilisée pour récupérer des données. C'est une opération de lecture qui ne modifie pas les ressources sur le serveur. Par exemple, pour obtenir la liste des utilisateurs ou les détails d'un produit.
- **POST** : Utilisée pour créer de nouvelles ressources. Cette méthode envoie des données au serveur pour créer un nouvel élément. Par exemple, pour ajouter un nouvel utilisateur ou créer une nouvelle commande.
- **PUT** : Utilisée pour mettre à jour une ressource existante. Cette méthode remplace entièrement la ressource ciblée par les nouvelles données fournies. Par exemple, pour mettre à jour les informations d'un utilisateur.
- **DELETE** : Utilisée pour supprimer une ressource. Cette méthode permet de retirer une ressource spécifique du serveur. Par exemple, pour supprimer un produit du catalogue.

Ces méthodes sont essentielles pour implémenter les opérations CRUD (Create, Read, Update, Delete) dans une API REST

OpenAPI