Une API, qu'est-ce que c'est ?

Mis à jour 5 février 2024 • 13 minutes (temps de lecture) Copier l'URL

API : définition

Une API, ou interface de programmation d'application, est un ensemble de définitions et de protocoles qui facilite la création et l'intégration des applications.

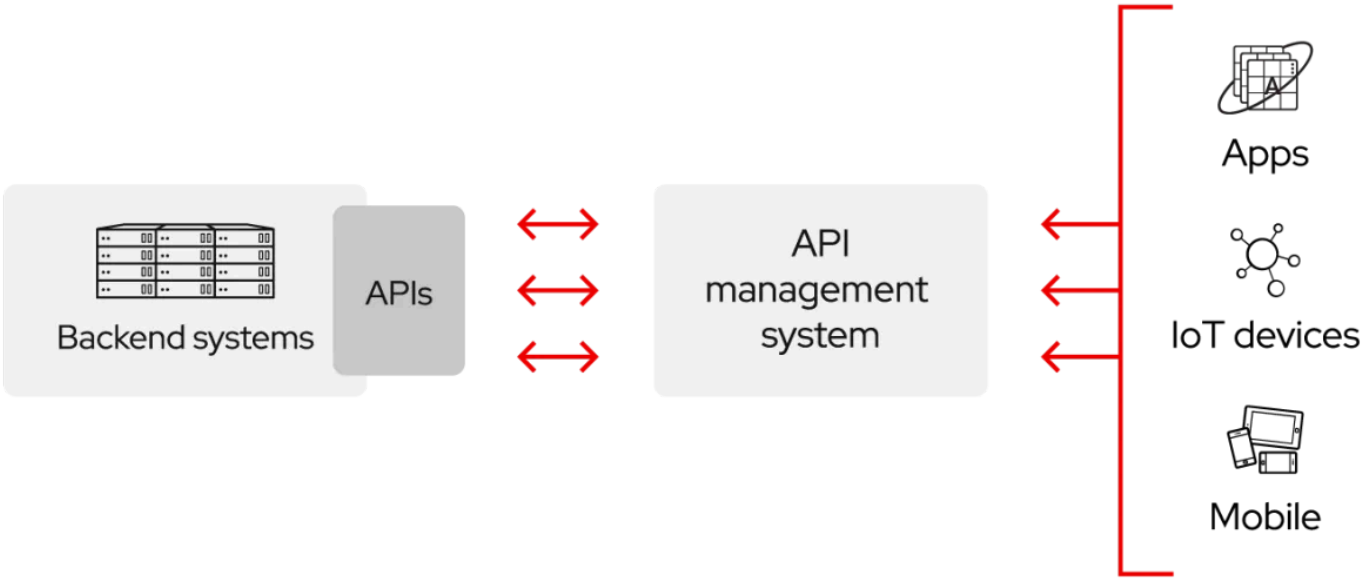
Fonctionnement des API

Les API permettent à votre produit ou service de communiquer avec d'autres produits et services sans connaître les détails de leur mise en œuvre. Elles simplifient le développement d'applications et vous font ainsi gagner du temps et de l'argent. Lorsque vous concevez de nouveaux outils et produits, ou que vous assurez la gestion de ceux dont vous disposez déjà, les API vous offrent plus de flexibilité, simplifient la conception, l'administration et l'utilisation, et vous donnent les moyens d'innover.

Les API sont parfois considérées comme des contrats, avec une documentation qui constitue un accord entre les parties : si la partie 1 envoie une requête à distance selon une structure particulière, le logiciel de la partie 2 devra répondre selon les conditions définies.

Parce que les API simplifient la façon dont les développeurs intègrent de nouveaux composants d'applications dans une architecture existante, elles facilitent la collaboration entre les équipes informatiques et métier. Souvent, les besoins des entreprises changent rapidement face à l'évolution constante des marchés numériques, où de nouveaux concurrents peuvent bouleverser tout un secteur avec une nouvelle application. Afin de conserver leur compétitivité, il est important pour ces entreprises de soutenir le développement et le déploiement rapides de services novateurs. Le développement d'applications cloud-native est un moyen évident d'augmenter la vitesse de développement. Il repose sur la connexion d'une architecture d'applications de type microservices via des API.

Les API constituent un moyen simplifié de connecter votre propre infrastructure au travers du développement d'applications cloud-native . Elles vous permettent également de partager vos données avec vos clients et d'autres utilisateurs externes. Les API publiques offrent une valeur métier unique, car elles peuvent simplifier et développer vos relations avec vos partenaires, et éventuellement monétiser vos données (l'API Google Maps en est un parfait exemple).



Imaginez, par exemple, un distributeur de livres. Ce distributeur pourrait fournir à ses librairies clientes une application cloud qui leur permettrait de vérifier la disponibilité des livres auprès du fournisseur. Toutefois, le développement de cette application risque d'être coûteux et de prendre du temps, alors que l'application finale risque d'être limitée par la plateforme et de nécessiter une maintenance continue.

Le distributeur peut aussi fournir une API pour vérifier la disponibilité des stocks. Cette approche présente plusieurs avantages :

- En accédant aux données via une API, les clients ont la possibilité de centraliser les informations sur leur inventaire.
- Le distributeur peut modifier ses systèmes internes sans impacter l'expérience de ses clients, tant que le comportement de son API ne change pas.
- Avec une API publique, les développeurs qui travaillent pour le distributeur, pour les librairies ou pour d'autres entreprises peuvent développer une application qui aide les clients à trouver les livres qu'ils souhaitent acheter. Ainsi, les distributeurs peuvent augmenter leurs ventes ou saisir de nouvelles opportunités commerciales.

En bref, avec les API, vous ouvrez l'accès à vos ressources, sans sacrifier le contrôle et la sécurité. Après, c'est vous qui choisissez les ressources que vous souhaitez partager, et avec qui. La sécurité des API est essentiellement une question de bonne gestion de celles-ci, ce qui implique l'utilisation d'une passerelle d'API. La connexion des API et la création des applications qui utilisent les données ou les fonctionnalités exposées par les API peuvent se faire par l'intermédiaire d'une plateforme d'intégration distribuée qui connecte tout, y compris les systèmes existants et l'Internet des objets (IoT).

Types d'API

API privées

L'API n'est utilisable qu'en interne. Cette approche permet de garder un contrôle total sur l'API.

API partenaires

L'API est partagée avec certains partenaires de l'entreprise. Cette approche peut générer de nouveaux flux de revenus sans compromettre la sécurité.

API publiques

L'API est accessible à tous. Cette approche autorise les tiers à développer des applications qui interagissent avec votre API et peut devenir source d'innovations.

Innovation par les API

En rendant vos API accessibles à vos partenaires ainsi qu'aux tiers, vous pouvez :

- générer de nouveaux canaux de revenus ou étendre ceux qui existent déjà ;
- étendre la portée de votre marque ;
- stimuler l'innovation Open Source ou améliorer l'efficacité grâce au développement et à la collaboration externes.

Intéressant, non ? Mais quel est le rôle des API dans tout ça ?

Cas d'utilisation des API

Reprenons notre exemple du distributeur de livres.

Imaginez qu'un partenaire de l'entreprise développe une application qui permet aux clients de localiser les livres qu'ils cherchent dans les rayons. En améliorant

l'expérience client, cette application va attirer d'autres clients dans la librairie, elle-même cliente du distributeur, qui en fin de compte aura étendu son canal de revenus.

Une entreprise tierce peut aussi utiliser une API publique pour développer une application afin que les clients puissent acheter des livres directement auprès du distributeur, sans passer par la librairie. Dans ce cas, le distributeur aura ouvert un nouveau canal de revenus.

Une entreprise peut tirer parti du partage de ses API avec certains de ses partenaires ou avec le monde entier. Chaque partenariat vous permet d'étendre la notoriété de votre marque en parallèle de vos campagnes marketing. En rendant vos technologies publiques, avec une API publique par exemple, vous encouragez les développeurs à créer un écosystème d'applications autour de votre API. Plus les gens utilisent vos technologies, plus ils sont susceptibles de faire affaire avec vous.

Vous pouvez ainsi obtenir des résultats aussi inattendus qu'intéressants en ouvrant l'accès à vos technologies et ces résultats peuvent parfois bouleverser un secteur tout entier. Dans le cas de notre distributeur de livres, de nouvelles entreprises, comme un service de location de livres, peuvent provoquer un changement fondamental dans son fonctionnement. Les API publiques et partenaires vous permettent de profiter des innovations d'une communauté de développeurs plus large. Les idées novatrices peuvent germer n'importe où. Les entreprises doivent se tenir au courant des changements qui s'opèrent sur leur marché et se préparer à y faire face. C'est là que les API interviennent.

Le gouvernement irlandais améliore la productivité grâce aux API →

L'histoire des API en quelques mots

Les API sont apparues à l'aube de l'informatique, avant même les ordinateurs personnels. À cette époque, elles étaient surtout utilisées en tant que bibliothèques pour les systèmes d'exploitation. Elles résidaient presque toutes en local sur les systèmes sur lesquels elles s'exécutaient, même si elles transféraient parfois des messages entre les mainframes. Presque 30 ans après, les API sont sorties de leurs environnements locaux. Au début des années 2000, elles sont devenues importantes pour l'intégration des données à distance.

API distantes

Les API distantes sont conçues pour interagir via un réseau de communication. Ici, « *distant* » signifie que les ressources manipulées par l'API ne se trouvent pas sur l'ordinateur qui formule la requête. Le réseau de communication le plus fréquemment utilisé étant Internet, la plupart des API sont conçues sur la base des normes web. Toutes les API distantes ne sont pas des API web, mais on peut supposer que toutes les API web sont distantes.

Les API web utilisent en général le protocole HTTP pour leurs messages de requête et fournissent une définition de la structure des messages de réponse. Les messages de réponse se présentent la plupart du temps sous la forme d'un fichier XML ou JSON. Ces deux formats sont les plus courants, car les données qu'ils contiennent sont faciles à manipuler pour les autres applications.

API SOAP ou REST

Pour standardiser l'échange des informations entre les API toujours plus nombreuses, il a fallu développer un protocole : le « Simple Object Access Protocol », plus connu sous le nom de SOAP. Les API conçues d'après le protocole SOAP utilisent le format XML pour leurs messages et reçoivent des requêtes via HTTP ou SMTP. SOAP a pour objectif de simplifier l'échange des informations entre les applications qui s'exécutent dans des environnements différents ou qui ont été écrites dans des langages différents.

Le « Representational State Transfer », ou REST, est une autre tentative de normalisation. Les API web qui respectent les contraintes de l'architecture REST sont appelées API RESTful. Ces deux éléments diffèrent sur un point fondamental : SOAP est un protocole, alors que REST est un style d'architecture. Cela signifie qu'il n'existe aucune norme officielle qui régit les API web RESTful. Selon la définition proposée par Roy Fielding dans sa thèse « Architectural Styles and the Design of Network-based Software Architectures », les API sont RESTful tant qu'elles respectent les six contraintes de conception d'un système RESTful :

- **Architecture client-serveur :** une architecture REST est composée de clients, de serveurs et de ressources et elle traite les requêtes via le protocole HTTP.
- **Serveur stateless :** le contenu du client n'est jamais stocké sur le serveur entre les requêtes. Les informations sur l'état de la session sont, quant à elles, stockées sur le client.

- **Mémoire cache :** la mise en mémoire cache permet de se passer de certaines interactions entre le client et le serveur.
- **Système à couches :** des couches supplémentaires peuvent assurer la médiation dans les interactions entre le client et le serveur. Ces couches peuvent remplir des fonctions supplémentaires, telles que l'équilibrage de charge, le partage des caches ou la sécurité.
- **Code à la demande (facultatif) :** un serveur peut étendre les fonctionnalités d'un client en lui transférant du code exécutable.
- **Interface uniforme :** cette contrainte est capitale pour la conception des API RESTful et couvre quatre aspects différents :
  - **Identification des ressources dans les requêtes :** les ressources sont identifiées dans les requêtes et sont séparées des représentations retournées au client.
  - **Manipulation des ressources par des représentations :** les clients reçoivent des fichiers qui représentent les ressources. Ces représentations doivent contenir suffisamment d'informations pour être modifiées ou supprimées.
  - **Messages autodescriptifs :** tous les messages renvoyés au client contiennent assez d'informations pour décrire la manière dont celui-ci doit traiter les informations.
  - **Hypermédia comme moteur du changement des états applicatifs :** après avoir accédé à une ressource, le client REST doit être en mesure de découvrir toutes les autres actions disponibles par des hyperliens.

Ces contraintes peuvent sembler difficiles à appliquer, mais dans les faits, elles le sont moins qu'un protocole. C'est pour cette raison que les API RESTful prennent progressivement le pas sur les API SOAP.

Ces dernières années, la spécification OpenAPI s'est imposée comme la norme commune pour définir les API REST. La norme OpenAPI permet aux développeurs de créer des interfaces d'API REST indépendantes du langage de manière à ce que les utilisateurs puissent les comprendre avec un minimum d'approximation.

demandées, et rien de plus. Utilisé à la place de REST, GraphQL permet aux développeurs de créer des requêtes qui extraient les données de plusieurs sources à l'aide d'un seul appel d'API.

En savoir plus sur les différences entre SOAP et REST →

Architectures SOA et de microservices

Les deux approches architecturales qui utilisent le plus les API distantes sont l'architecture orientée services (SOA) et l'architecture de microservices. La SOA est l'approche la plus ancienne. Elle visait à l'origine à améliorer les applications monolithiques. Par définition, une application monolithique fait tout. Mais certaines fonctions peuvent être fournies par d'autres applications faiblement couplées via un modèle d'intégration, comme un ESB (Enterprise Service Bus).

Si l'architecture SOA est plus simple qu'une architecture monolithique sous bien des aspects, elle peut potentiellement provoquer des changements en cascade au sein de l'environnement si les interactions entre les différents composants ne sont pas parfaitement comprises. En complexifiant l'environnement, l'architecture SOA réintroduit une partie des problèmes qu'elle cherchait à résoudre.

Les architectures de microservices fonctionnent d'une manière très similaire, dans le sens où elles utilisent des services faiblement couplés. Par contre, elles poussent la déstructuration de l'architecture classique encore plus loin. Les services qui composent l'architecture de microservices utilisent une structure de messagerie commune, telle que les API RESTful. Ils se servent des API RESTful pour communiquer entre eux simplement, sans convertir leurs données ni recourir à des couches d'intégration supplémentaires. L'utilisation des API RESTful permet et favorise même l'accélération de la distribution des nouvelles fonctions et mises à jour. Chaque service est distinct. Vous pouvez remplacer, améliorer ou supprimer chacun d'entre eux sans affecter les autres services de l'architecture. Cette architecture légère vous aide à optimiser les ressources distribuées ou cloud et à faire évoluer chaque service de façon dynamique.

En savoir plus sur les architectures SOA →

API ou webhook

Un webhook est une fonction de rappel basée sur le protocole HTTP. Il permet à deux API d'établir une communication légère orientée événements. Si des applications de toutes sortes les utilisent pour réceptionner de petites quantités de

données en provenance d'autres applications, les webhooks peuvent également déclencher les workflows d'automatisation des environnements GitOps.

On qualifie souvent les webhooks d'API inversées ou d'API « push », car elles confient la communication au serveur plutôt qu'au client. C'est le serveur qui envoie au client une demande HTTP POST unique dès que les données sont disponibles, et non plus le client qui envoie des requêtes en continu. Malgré leurs surnoms, les webhooks ne sont pas des API, mais ces deux systèmes fonctionnent ensemble. Pour utiliser un webhook, une application doit disposer d'une API.

En savoir plus sur les webhooks →

Pour aller plus loin

ARTICLE

Une API, qu'est-ce que c'est ?

Une API, ou interface de programmation d'application, est un ensemble de définitions et de protocoles qui facilite la création et l'intégration de logiciels d'applications.

En savoir plus →

ARTICLE

Quel est le rôle d'une passerelle d'API ?

Une passerelle d'API est un outil de gestion des interfaces de programmation d'application (API) qui se positionne entre un client et une collection de services back-end.

En savoir plus →

ARTICLE

Red Hat, un partenaire de choix en matière d'API

Nos solutions d'API privilégient la réutilisation ainsi que l'agilité informatique. Elles incluent une interface de gestion qui vous aide à analyser, surveiller et faire évoluer votre environnement.

En savoir plus →

En savoir plus sur les API

Produits | Articles liés | Ressources

Red Hat 3scale API Management

Plateforme d'infrastructure sur laquelle partager, distribuer, contrôler et monétiser les interfaces de programmation d'application (API).

En savoir plus →