diginamic FORMATION
« Votre passeport pour l'emploi numérique »

Définition d'une API

Sommaire

Pourquoi avons-nous besoin d’API ?

Les applications modernes sont **modulaires**, **connectées** et souvent **distribuées**. Elles sont composées de plusieurs couches ou services qui doivent **échanger des données** entre eux.

🔧 Quelques architectures typiques

Architecture	Description	Exemple
Monolithique	Une seule application contenant toutes les fonctionnalités (frontend, backend, logique métier, base de données)	Une app PHP avec tout dans le même dossier
Client-Serveur	Séparation entre une interface utilisateur (client) et un serveur (qui traite les données)	Un site React qui interagit avec un backend Python
Microservices	Plusieurs petits services indépendants qui communiquent entre eux	Un service pour les utilisateurs, un pour les produits, un pour les commandes
Serverless / Cloud Functions	Fonctions déployées indépendamment dans le cloud, qui s’activent à la demande	Une fonction AWS Lambda qui envoie un email

Quel problème résolvent les API ?

Dans tous ces cas, **les différents composants doivent dialoguer**. Mais comment faire communiquer un client React avec un service Python ? Ou une app mobile avec une base de données ?

👉 C’est là que les **API** entrent en jeu : elles offrent une **interface standardisée** pour permettre à des composants logiciels **hétérogènes** (différents langages, plateformes, environnements) de **communiquer** de manière claire, sécurisée et documentée.

Qu’est-ce qu’une API ?

Une **API (Application Programming Interface)** est un **contrat d'échange** entre deux logiciels : elle décrit **ce que l’on peut faire**, **comment on peut le faire**, et **quel type de réponse on peut attendre**.

Une API définit par exemple :

- Des **points d’accès** (appelés *endpoints*) accessibles via une URL
- Les **méthodes** que l’on peut utiliser (GET, POST, etc.)
- Les **formats de données** acceptés (JSON, XML…)
- Les **codes de retour** selon le résultat

🤔 Métaphore – L'API comme menu de restaurant

Imaginez un restaurant :

- Le **client** (votre application) fait une commande.
- Le **menu** (l'API) vous indique ce que vous pouvez commander.
- Le **serveur** (le backend) prépare le plat et vous le livre.
- Vous n'avez **pas besoin de savoir comment le plat est cuisiné**, vous utilisez simplement l'interface (le menu) pour interagir avec le système.

À quoi sert une API dans la pratique ?

Les API permettent :

- De **dissocier les responsabilités** (ex. : un frontend web qui appelle une API backend)
- De **rendre des fonctionnalités accessibles** à d'autres applications (authentification, paiements, gestion de fichiers, etc.)
- De **standardiser les échanges** entre différents services, langages, plateformes
- D'**intégrer des services tiers** (ex : Stripe pour les paiements, GitHub pour le code, SendGrid pour les emails…)

Exemples de cas concrets

- Un site e-commerce frontend (React) appelle une **API pour gérer les produits et les commandes**
- Une app mobile appelle une API pour **authentifier un utilisateur**
- Un service backend appelle l'API de GitHub pour **récupérer des dépôts publics**
- Un microservice de facturation expose une API REST pour **générer des factures**