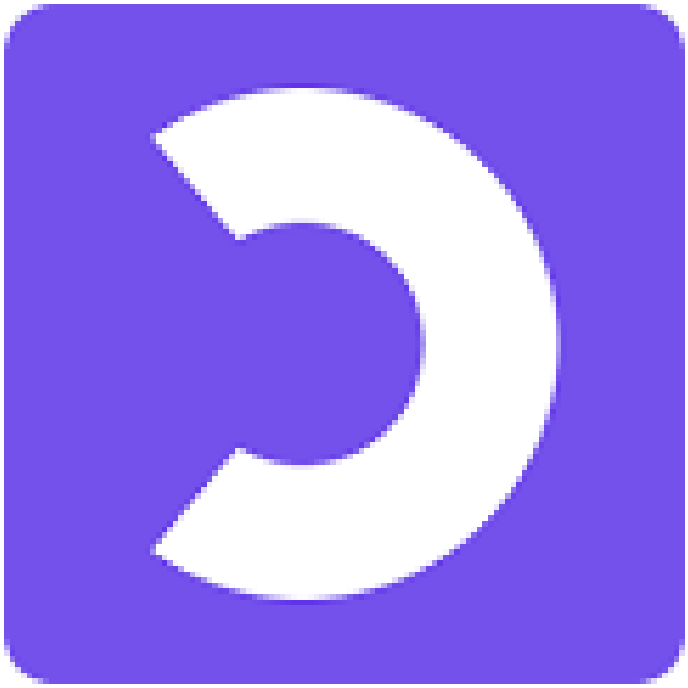
API REST



Alexandre Goux
Et
Sébastien Szollosi

Mis à jour le 14 mars 2023

Table des matières

Découvrez les API

Qu'est-ce qu'une API ?

Une API (Interface de Programmation d'Application) est un ensemble de classes, de fonctions et de méthodes qui agissent comme une interface pour un logiciel. Elle permet à d'autres logiciels d'accéder aux services du logiciel en question. L'API facilite la communication entre deux produits ou services, tels qu'une application et un service de géolocalisation, en leur permettant d'échanger des données sans connaître les détails de leur mise en œuvre.

L'API permet aux développeurs de gagner du temps et aux entreprises de réaliser des économies. Elle permet également d'ouvrir un produit au monde extérieur. Par exemple, lors de la consultation des avis sur un restaurant, en cliquant sur un restaurant sur une carte dans un navigateur web, une fenêtre s'ouvre avec les informations sur le restaurant, telles que l'adresse, le téléphone, les avis et les menus. Ce service d'affichage des informations est rendu possible grâce à l'API.

Quelques exemples d'API

De nombreux logiciels populaires tels que Facebook, Twitter, Snapchat, Shazam, Netflix et ArcGIS utilisent des API. Ces logiciels proposent également leurs propres API, permettant à d'autres services d'accéder à leurs fonctionnalités. Ainsi, ces logiciels s'appuient sur les API d'autres services pour enrichir leurs propres fonctionnalités et offrir une meilleure expérience utilisateur.

Comme fonctionne une API

Une API nécessite à la fois un serveur et un client pour fonctionner :

- Le serveur est un ordinateur puissant qui exécute l'API. Il agit en tant que fournisseur du service, offrant au client l'accès à ses données ou services.
- Le client est un programme qui échange des données avec le serveur via l'API. Il est l'utilisateur de l'API, interagissant avec le serveur pour obtenir les informations ou effectuer des actions nécessaires. Cette architecture est communément appelée architecture client-serveur.

Ainsi, le serveur et le client travaillent en collaboration via l'API pour permettre la communication, l'échange de données et l'accès aux services proposés par le serveur.

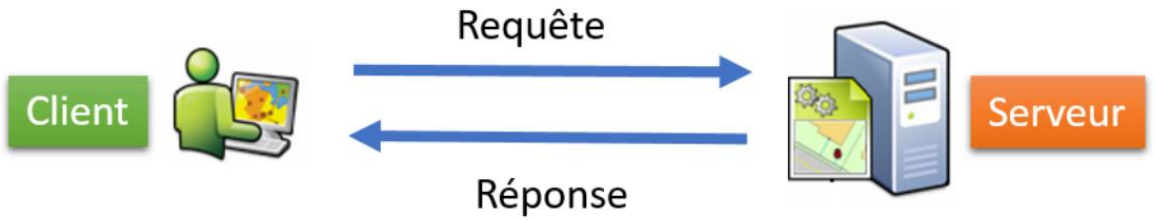
Rôle du client :
Le client est l'interface utilisateur qui présente les réponses du serveur de manière conviviale, comme une carte avec des monuments où l'utilisateur peut cliquer sur un monument pour obtenir des informations spécifiques.
Le client est responsable de l'initiation de la communication avec le serveur. Il peut demander l'exécution de services, tels que récupérer des informations en saisissant un mot dans un moteur de recherche.
Le client envoie donc des requêtes au serveur et attend une réponse en retour.

Rôles du serveurs :

Le serveur est la partie invisible du système. Il peut prendre différentes formes telles qu'un serveur de calcul, un serveur d'applications ou un serveur de bases de données.

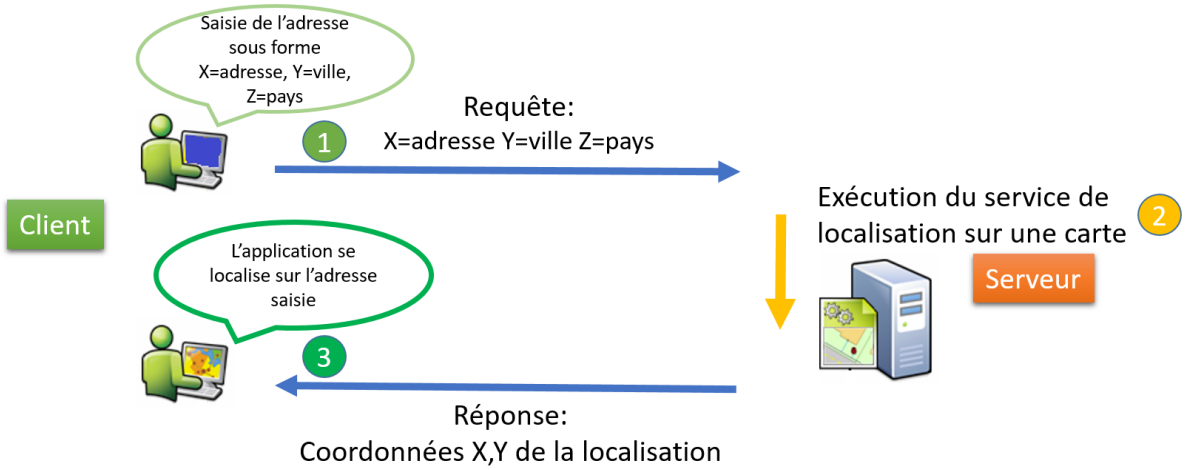
Le serveur fournit le service demandé, par exemple en envoyant les informations d'un monument. Cependant, c'est le client qui se charge de la mise en forme de ces informations. Le serveur est constamment à l'écoute des requêtes des clients et peut répondre à plusieurs clients simultanément.

Client -> Requête -> Serveur -> Réponse -> Client



Comment le client et serveur communiquent-ils ?

Le client et le serveur communiquent via un protocole client-serveur. Le client envoie une requête au serveur et reçoit une réponse en retour. La communication peut se faire entre deux machines distinctes ou sur la même machine.
La communication utilise un protocole spécifique, tel que Bluetooth, POP (messagerie mail), ou HTTP (web), qui définit les règles de communication entre les objets connectés.
Le client envoie une requête, qui peut être un message codé, des paramètres d'appel ou une configuration de service. Le serveur renvoie ensuite une réponse, par exemple en fournissant des coordonnées X et Y pour une adresse demandée.



Utilisez les API dans vos projets de développement

L'utilisation d'une API est recommandée pour exécuter ou utiliser des services web. Par exemple, si vous développez une application qui nécessite des informations sur le trafic routier, il est conseillé d'utiliser l'API de Waze plutôt que de tout développer vous-même. De nombreuses entreprises proposent des services de qualité accessibles via leur API, profitez-en !
La plupart des API disponibles sur Internet nécessitent une inscription préalable pour obtenir une clé ou un jeton d'accès aux données, également appelés ressources. Cette clé peut être un identifiant/mot de passe, une chaîne de caractères ou un ensemble de nombres.

Résumé

Une API est une interface logicielle permettant d'accéder aux données d'autres services. Le client envoie des requêtes au serveur (fournisseur du service via l'API) et ce dernier renvoie les données demandées, sous réserve de leur accessibilité et des droits d'accès du client. La communication entre le client et le serveur s'effectue via un protocole tel que HTTP.

Passez à l'architecture REST

Qu'est-ce qu'une API REST ?

REST (REpresentational State Transfer) est un type d'architecture d'API inventé par Roy Fielding dans les années 2000. Cela a été une période importante pour la reconnaissance du potentiel des API web. L'objectif de REST était de fournir des méthodes simples pour accéder à des services web.

REST est un type d'API qui facilite la communication entre des logiciels incompatibles qui utilisent des langages différents. Il peut être considéré comme un langage commun pour ces logiciels. Par exemple, une API REST peut être implémentée en Java ou en .NET.

Quelques exemples

L'API REST est utilisée dans de nombreux services populaires :

- Instagram : L'API d'Instagram vous permet d'accéder à des comptes utilisateurs, des photos, des tags, etc.
- GitHub : L'API REST de GitHub vous permet de suivre l'activité d'un utilisateur, consulter les bugs d'un dépôt et même créer un dépôt depuis votre application.
- Gmail : L'API Gmail vous permet de lire, envoyer et gérer des messages, ainsi que d'utiliser d'autres fonctionnalités telles que les brouillons, les pièces jointes et la recherche de messages.
- ArcGIS : L'API ArcGIS développée par Esri est utilisée pour collecter, organiser, analyser et diffuser des informations géographiques, permettant par exemple de vous localiser sur une carte ou de calculer un itinéraire.

Utilisez les API REST facilement dans votre projet

REST a été conçu pour répondre aux besoins du web et est basé sur le standard URI (Uniform Resource Identifier). Il utilise le protocole HTTP, ce qui le rend simple à utiliser pour les logiciels. Les API REST suivent les contraintes de l'architecture REST, ce qui contribue à des applications de meilleure qualité.

L'API REST est performante car elle permet à de nombreux clients de se connecter au serveur simultanément. Utiliser REST dans notre projet informatique offre une mise en œuvre simple, une tolérance aux pannes accrue (un serveur en panne peut être remplacé par un autre) et donc une grande fiabilité.

Programmez simplement avec l'API REST

L'API REST utilise les verbes HTTP (POST, GET, PUT, DELETE) pour manipuler les ressources. Ces verbes correspondent aux opérations CRUD (Create, Read, Update, Delete) :

- **POST** : Créer (Create) une ressource.
- **GET** : Afficher (Read) une ressource.
- **PUT** : Mettre à jour (Update) la valeur d'une ressource.
- **DELETE** : Supprimer (Delete) une ressource du serveur.

Ainsi, en utilisant ces verbes HTTP, nous pouvons effectuer les opérations nécessaires sur les ressources via l'API REST.

Exemples :

Voici des exemples d'URL pour gérer des timbres en ligne :

- Pour créer un timbre : POST http://monsiteweb.fr/stamps Le serveur vous fournira probablement l'identifiant du timbre créé, par exemple 183
- Pour afficher le timbre 183 : GET http://monsiteweb.fr/stamps/183
- Pour mettre à jour le timbre 183 : PUT http://monsiteweb.fr/stamps/183
- Pour supprimer le timbre 183 : DELETE http://monsiteweb.fr/stamps/183

Profitez des avantages du REST

L'API REST est la plus utilisée du web.

Transformez votre API en API RESTful

Une API est qualifiée de RESTful lorsqu'elle respecte les principes d'architecture REST pour les services web. Les principales contraintes proposées par Roy Fielding sont les suivantes :

- Client-serveur : La communication se fait selon un modèle client-serveur avec une séparation claire des rôles entre les deux.
- Stateless server : Le serveur ne conserve pas l'état de la session client. Chaque requête doit contenir toutes les informations nécessaires pour être traitée.
- Cache : Le serveur doit permettre la mise en cache des réponses côté client pour une meilleure efficacité et performance.
- Uniform interface : L'interface de communication entre le client et le serveur doit être uniforme et basée sur l'utilisation d'URL.

Les clés de l'API REST

L'architecture REST repose sur 4 principes clés :

1. Mécanisme client-serveur : Il y a une séparation claire des rôles entre le client et le serveur, permettant une meilleure évolutivité et indépendance des composants.
2. Identifiant de la ressource : Chaque ressource est identifiée de manière unique par une URL (Uniform Resource Locator) et est accessible via le protocole HTTP.
3. Verbes HTTP : Les opérations sur les ressources sont effectuées à l'aide des verbes HTTP tels que GET (récupération), POST (création), PUT (modification) et DELETE (suppression). Ces verbes décrivent les actions à entreprendre sur la ressource.
4. Représentation de la ressource : La réponse renvoyée par le serveur peut avoir plusieurs représentations possibles, telles que HTML, JSON ou XML. Cela permet à différents clients de choisir la représentation qui leur convient le mieux.

Ces principes clés de l'architecture REST favorisent la simplicité, la scalabilité et l'interopérabilité des systèmes.

Analysez les codes de retour des verbes HTTP !

Lorsque vous effectuez une requête auprès d'un serveur, celui-ci vous renvoie une réponse qui inclut un code d'état HTTP. Ce code d'état n'est pas visible dans votre navigateur, mais il indique si la requête a été traitée avec succès ou non. Le protocole HTTP définit quarante codes d'état standard, composés de trois chiffres, qui sont utilisés pour transmettre les résultats de la demande du client.
Ces codes sont regroupés en cinq catégories :

- **1xx** : Informatif, communique une information au client.
- **2xx** : Succès, indique que la demande du client a été acceptée avec succès.
- **3xx** : Redirection, indique que le client doit prendre des mesures supplémentaires pour compléter sa demande.
- **4xx** : Erreur du client, indique une erreur de la part du client.

- **5xx** : Erreur du serveur, indique une erreur de la part du serveur.

Ces codes d'état sont importants pour comprendre le résultat de votre requête et prendre les mesures appropriées en cas d'erreur ou de succès.

Séparez le client du serveur

La séparation des responsabilités entre le client et le serveur est essentielle pour une API efficace :

- Le serveur gère les règles métier et les données.
- Le client se concentre sur l'interface utilisateur.

En séparant ces deux parties, vous améliorez la portabilité et la scalabilité de votre application. Chaque composant peut évoluer indépendamment. Par exemple, vous pouvez refaire le design d'un site web sans avoir à modifier l'API côté serveur.

Rendez votre API REST fiable et évolutive

Vous pouvez tout à fait faire évoluer votre API REST existante ou en créer une nouvelle. Chaque API REST peut être enrichie de nouvelles fonctionnalités.

L'API REST est indépendante du type de plateforme ou des langages

L'un des avantages clés de l'API REST est qu'elle fonctionne indépendamment du langage de programmation ou de la plateforme utilisée par le client et le serveur :

- Indépendant du langage : Un service web développé en Java doit pouvoir être utilisé par un client codé en Python, en C++, ou dans tout autre langage.
- Indépendant de la plateforme : Un service web fonctionnant sur une plateforme Linux peut être accessible par un client utilisant Windows, un système d'exploitation mobile ou toute autre plateforme.

l'API REST offre une grande flexibilité et interopérabilité, permettant à des applications développées dans différents langages ou sur différentes plateformes de communiquer et d'échanger des données de manière transparente.

Transmettez l'information avec JSON

JSON

JSON est l'acronyme de JavaScript Object Notation.

Le format JSON est utilisé pour stocker et organiser les données de manière lisible. Il utilise des paires "clé - valeur" pour représenter les informations et est couramment utilisé dans le développement web.


```json
{
    "currentVersion":10.3,
    "folders":
    [
        "Canvas",
        "Demographics",
        "Elevation",
        "Ocean",
        "Polar",
        "Reference",
        "Specialty",
        "Utilities"
    ],
    "services":
    [
        {
            "name":"NatGeo_World_Map",
            "type":"MapServer"
        },
        {
            "name":"USA_Topo_Maps",
            "type":"MapServer"
        },
        {
            "name":"World_Imagery",
            "type":"MapServer"
        },
        {
            "name":"World_Physical_Map",
            "type":"MapServer"
        },{
            "name":"World_Shaded_Relief",
            "type":"MapServer"
        },
        {
            "name":"World_Street_Map",
            "type":"MapServer"
        },
        {
            "name":"World_Terrain_Base",
            "type":"MapServer"
        },
        {
            "name":"World_Topo_Map",
            "type":"MapServer"
        }
    ]
}
```


- les clés en vert avant les : ; ce sont toujours des chaînes de caractères ;
- les valeurs en jaune, de l'autre côté des : ; elles peuvent être de plusieurs types, des chaînes de caractères, mais aussi des nombres, des tableaux, des objets...

Les documents JSON sont relativement légers et leur traitement côté serveur web est donc rapide.

XML

XML (eXtensible Markup Language) est un langage de balisage utilisé pour décrire et communiquer des informations entre différentes applications informatiques. Il est extensible, ce qui signifie qu'il peut être utilisé pour décrire des langages dérivés du XML avec leurs propres balises et règles. XML est largement utilisé pour l'échange de données structurées entre systèmes, permettant une communication efficace et une interopérabilité entre différentes applications.

```
<NOM>ARCGIS</NOM>
<CREATEUR>Jack Dangermond</CREATEUR>
```

- les balises <NOM> ou <CREATEUR> permettent de délimiter les informations correspondant au nom et au créateur.
- <NOM> annonce le début des informations concernant le nom.
- </NOM> annonce la fin des informations concernant le titre (notez le slash /).

Un des avantages du XML est que c'est l'un des rares formats qui peuvent être lus à la fois par un humain et par un ordinateur.

JSON ou XML, lequel choisir pour l'API REST ?

Dans le contexte d'un service web, vous pouvez utiliser à la fois le format JSON et XML pour recevoir des données. Par exemple, si vous souhaitez représenter les informations sur les employés de votre société (John Doe, Arlette Smith et Jean Dupont), voici à quoi cela pourrait ressembler dans les deux formats :

json Copy code

```json
{
  "employes": [
    {
      "nom": "John Doe"
    },
    {
      "nom": "Arlette Smith"
    },
    {
      "nom": "Jean Dupont"
    }
  ]
}
```

xml Copy code

```xml
<employes>
  <employe>
    <nom>John Doe</nom>
  </employe>
  <employe>
    <nom>Arlette Smith</nom>
  </employe>
  <employe>
    <nom>Jean Dupont</nom>
  </employe>
</employes>
```

En comparant XML et JSON, on constate que JSON est plus facile à analyser grâce à sa compatibilité avec les objets JavaScript, tandis qu'XML nécessite un analyseur spécifique. De plus, JSON est plus rapide car il est moins verbeux et son analyse est plus simple.

Transmettez l'information

Lors de l'appel de l'API REST via la requête GET http://masociete.com/employees/all, le serveur renverra la réponse au format JSON contenant la liste des employés de la société. Cette réponse sera plus facilement analysable grâce à la compatibilité de JSON avec les objets JavaScript, offrant une simplicité d'analyse et une meilleure performance par rapport à XML.

```json
{
    "employees":
    [
        {
            "prenom":"John",
            "nom":"Doe"
        },
        {
            "prenom":"Arlette",
            "nom":"Smith"
        },
        {
            "prenom":"Jean",
            "nom":"Dupont"
        }
    ]
}
```

L'information du serveur vers le client est transmise grâce au protocole HTTP et à travers un format d'échange, ici JSON car c'est le plus courant pour l'API REST.

Configurez votre URL pour transmettre les paramètres

3 symboles sont utilisés pour ajouter une chaîne de paramètres à une URL :

- ? concatène l'URL et la chaîne de paramètres ;
- & sépare les paramètres multiples ;
- = assigne une valeur à un paramètre.

Par exemple, l'URL suivante comporte les paramètres Bbox, BboxSR, Layers, layerDefs, etc.

```
http://sampleserver6.arcgisonline.com/arcgis/rest/services/Wildfire/MapServer/export?bbox=-1.2800616666100327E7%2C3803571.1214327915%2C-1.1144321686843704E7%2C5687408.222440326&bboxSR=&layers=1&layerDefs=&size=&imageSR=&historicMoment=&format=png&transparent=false&dpi=&time=&layerTimeOptions=&dynamicLayers=&gdbVersion=&mapScale=&rotation=&datumTransformations=&layerParameterValues=&mapRangeValues=&layerRangeValues=&f=html
```

La seule limite est dans la taille de l'URL : 2 083 caractères.

Informez-vous sur les API existantes

Avant toute chose, il faut se renseigner sur les API et leurs utilisations

Analysez une API existante par rapport à votre besoin

L'utilisation d'une API dans un projet informatique nécessite de prendre en compte les droits d'usage qui lui sont associés. Ces droits peuvent inclure des restrictions quant à l'utilisation de l'API dans un projet commercial, des considérations relatives aux droits d'auteur, et d'autres aspects similaires. Il est donc essentiel de se familiariser avec les conditions d'utilisation de l'API choisie afin de garantir une utilisation légale et appropriée dans le cadre du projet.

Certaines API nécessitent des licences pour les utiliser. Ces licences peuvent être **gratuites** ou **payantes**.
D'autres API nécessitent une **authentification** (utilisateur/mot de passe) afin de pouvoir y accéder.

Lors de l'utilisation d'une API dans un projet informatique, il est important de respecter les règles d'utilisation. Cela peut inclure des restrictions liées à un usage commercial ou aux droits d'auteur. Il est également crucial de ne pas divulguer d'informations confidentielles qui pourraient être utilisées de manière frauduleuse, comme des données bancaires. Soyez vigilant et protégez vos informations !

Distinguez API publique et API privée

L'API Gmail est une API publique, ce qui signifie qu'elle est ouverte aux développeurs externes à Google. Elle permet à ces développeurs d'accéder aux fonctionnalités de Gmail dans le cadre de leurs projets informatiques. Les API publiques, également connues sous le nom d'"Open API", sont conçues pour être utilisées par des communautés de développeurs externes à l'entreprise qui fournit l'API.

une API doit répondre à certaines contraintes, notamment en termes de sécurité.

Contrairement à une API publique, une API privée est limitée à une utilisation interne au sein de l'entreprise qui l'a développée. Elle n'est pas accessible au grand public.

Utilisez une API en toute sécurité

Vérifiez la provenance de l'API pour éviter les mauvaises surprises

Par exemple, lors de la création d'une boutique en ligne, il est crucial de vérifier la fiabilité d'une API avant de l'utiliser pour gérer les paiements en ligne. Il existe des risques potentiels tels que l'utilisation d'une API frauduleuse qui pourrait entraîner des conséquences désastreuses, notamment le vol de données bancaires des utilisateurs lors des transactions.

Il est important de rester vigilant et de préférer des API certifiées ou sécurisées. Si nécessaire, il est possible d'acheter une API fiable plutôt que de la développer soi-même. Il est également recommandé d'utiliser le protocole HTTPS lors de l'utilisation d'une API pour assurer une communication sécurisée. Certaines entreprises, comme CDiscount, offrent des API fiables et sécurisées, car elles les utilisent elles-mêmes pour leurs propres services.

Testez l'API dans Postman

Après avoir trouvé et vérifié la fiabilité de l'API, il est temps de la tester avant de l'intégrer dans votre programme informatique.

Postman est un logiciel utilisé pour tester et interagir avec des API. Il fournit une interface conviviale permettant d'envoyer des requêtes HTTP personnalisées aux API, de recevoir et d'analyser les réponses. Postman facilite le processus de développement, de débogage et de documentation des API en offrant des fonctionnalités telles que la création de requêtes, la visualisation des réponses, la gestion des environnements et des collections d'API.

Authentifiez votre API

Qu’est-ce que l’authentification ?

Les API fonctionnent grâce à la communication entre un client et un serveur via le protocole HTTP et des formats de données spécifiques. Pour garantir la sécurité, le serveur utilise un processus d'authentification pour vérifier l'identité du client.
L'authentification consiste à prouver son identité en fournissant des informations confidentielles connues uniquement par le client.
Une fois authentifié, le client peut accéder aux ressources de son compte sur l'API.
Différentes méthodes d'authentification sont utilisées, telles que

- l'authentification basique
- l'authentification par clé
- l'authentification ouverte (OAuth).

Authentifiez l'API basiquement

L'authentification basique est une méthode simple qui nécessite un nom d'utilisateur et un mot de passe. Le client transforme ces identifiants en une valeur unique qu'il envoie dans sa requête HTTP. Le serveur compare ensuite cette valeur unique aux identifiants enregistrés. Si les identifiants correspondent à un utilisateur dans la liste, le serveur exécute la requête. Sinon, il renvoie un code de statut spécial (401) indiquant que l'authentification a échoué et que la requête n'a pas pu être exécutée.

Authentifiez l'API avec une clé

L'authentification basique utilise les mêmes identifiants que ceux du compte utilisateur, ce qui peut poser des problèmes de sécurité. Pour remédier à cela, une solution consiste à utiliser une clé d'API unique. La clé d'API est différente du mot de passe du compte utilisateur et est utilisée pour accéder à l'API. Cela permet de limiter les permissions du client et de protéger les données du compte. Le client s'authentifie en incluant la clé API dans l'URL, et le serveur vérifie cette clé pour autoriser l'accès aux ressources. De plus, la clé d'API peut également être utilisée pour restreindre certaines fonctions administratives afin de protéger les mots de passe des utilisateurs.

Authentifiez l'API avec OAuth2

Une alternative aux clés API est l'utilisation d'OAuth2, un protocole de délégation d'autorisation largement utilisé sur le web. Au lieu de transmettre les identifiants à chaque requête, OAuth2 permet à l'application cliente d'obtenir une autorisation unique du serveur. Par exemple, lorsque vous utilisez un client Twitter, l'application ne vous demande pas votre mot de passe à chaque publication de message. Au lieu de cela, l'application a reçu une autorisation préalable pour agir en votre nom. Cela simplifie le processus et améliore la sécurité en évitant la transmission répétée des identifiants.

Découvrez d’autres API

Qu'est-ce que SOAP ?

SOAP (Simple Object Access Protocol) est un protocole de communication basé sur XML qui permet aux applications d'échanger des informations via HTTP. Il facilite l'accès aux services web et favorise l'interopérabilité des applications sur le web. Comparé à REST, SOAP est plus lent et nécessite une bande passante Internet plus importante en raison de la verbose du format XML. En résumé, si vous avez besoin de publier une API complexe vers l'extérieur, SOAP peut être utile, mais si vous souhaitez une mise en œuvre rapide et simple d'API, REST est la meilleure option.

Quelques recommandations

Pour une utilisation optimale des API, voici quelques conseils importants :

- Évitez d'utiliser des API publiques non sécurisées.
- Privilégiez les API REST, voire RESTful, qui offrent une architecture flexible et standardisée pour les communications.
- Optez pour des API qui utilisent le format d'échange JSON, largement répandu et facile à manipuler.

Préparez la conception de votre API

La mise en place d'une API suit un processus similaire à celui d'un projet informatique. Pour optimiser notre temps de développement et de mise en ligne de notre API, nous pouvons suivre quatre étapes clés :

1. **Concevoir l'API** : Nous devons réfléchir à la manière dont nous allons construire notre API en déterminant les ressources que nous souhaitons mettre à disposition. Une planification minutieuse et une structuration cohérente sont essentielles pour offrir une expérience utilisateur optimale.
2. **Construire l'API** : Une fois la conception établie, nous passons à l'étape de développement, où nous écrivons le code nécessaire pour implémenter les fonctionnalités de l'API et interagir avec les ressources prévues. Nous veillons à respecter les bonnes pratiques de développement et à tester rigoureusement l'API pour garantir sa stabilité et sa sécurité.
3. **Comment utiliser l'API :** Une documentation claire et complète est cruciale pour guider les développeurs dans l'utilisation de notre API. Nous expliquons les fonctionnalités offertes, les requêtes acceptées, les réponses attendues, ainsi que les paramètres et exemples pertinents. Une documentation bien rédigée facilite l'intégration de l'API par d'autres développeurs.
4. **Faire connaître l'API :** Une fois que notre API est prête, il est important de la promouvoir pour attirer des utilisateurs potentiels. Nous pouvons partager des informations sur l'API via des sites web, des blogs, des conférences ou des événements dédiés aux développeurs. La création d'une communauté autour de l'API, en fournissant un support et en écoutant les retours des utilisateurs, favorise son adoption et son succès.

Réfléchissez avant de construire votre API

En tant que développeurs, il est crucial de prendre le temps de formaliser nos besoins, spécifier et concevoir notre projet informatique avant de commencer à développer notre API. Cela nous permet de gagner du temps précieux.

Avant tout, il est essentiel de comprendre les applications et leurs contraintes d'utilisation. Les questions clés à se poser sont :

- Comment allons-nous construire notre API ?
- REST ou SOAP ?
- Quels éléments seront disponibles dans notre API ?

Par exemple, dans un projet de commerce électronique, les entités principales peuvent être les clients et les commandes. Nous pourrons inclure des fonctionnalités telles que la récupération des commandes d'un client et la consultation de l'état d'une commande.
En formalisant nos idées et nos besoins dès le début, nous optimisons notre temps de développement et répondons plus efficacement aux attentes de nos clients.

Vérifiez que l’API n’existe pas

Avant de commencer à concevoir notre API, nous devons faire des recherches pour vérifier si une API similaire existe déjà. Cela nous permettra de gagner du temps et de nous assurer que notre API est unique.

Construisez votre API

Passez à la construction !

Organisez vos données

Avant de commencer la conception de notre API, nous devons organiser nos données de manière appropriée. Les fonctionnalités de notre API dépendront largement de cette organisation. Par exemple, si nous construisons une API pour gérer les livres d'une bibliothèque nationale, les fonctionnalités seront différentes de celles destinées à une petite structure. Il est également important de penser à l'évolutivité de l'API. Si notre API peut répondre aux besoins d'une bibliothèque nationale aussi bien que ceux d'une petite structure, nous aurons une API polyvalente. En ce qui concerne la conception, le choix d'une API REST nous offre une plus grande liberté.

URL ou URI ?

L'URI (Uniform Resource Identifier) est un identifiant unique d'une ressource, tandis que l'URL (Uniform Resource Locator) est un sous-ensemble spécifique des URIs utilisé pour localiser précisément une ressource sur Internet. Toutes les URL sont des URIs, car elles identifient et localisent une ressource, mais toutes les URIs ne sont pas des URLs, car certaines peuvent uniquement identifier une ressource sans fournir d'information de localisation. Cette distinction est importante pour comprendre et utiliser correctement les concepts d'URI et d'URL dans le développement et l'accès aux ressources en ligne.

Définissez les ressources

Une ressource est une entité spécifique associée à une URL. Les endpoints (points finaux) définissent les fonctionnalités accessibles via l'API. Par exemple, dans une API de gestion de bibliothèque, les endpoints peuvent inclure :

Verbe HTTP	endpoint	Actions
GET	/commands	Lister les commandes
POST	/commands	Passer une commande
GET	/commands/123	Détails de la commande 123
PUT	/commands/123	Modifie la commande 123
DELETE	/commands/123	Annule la commande 123

Chaque ressource peut être accessible en utilisant différents formats de données, tels que HTML ou JSON. Déterminez les données à échanger entre le client et le serveur en précisant les paramètres et les formats nécessaires.

Liez les ressources entre elles

Pour compléter notre API, nous avons besoin d'ajouter un endpoint pour la ressource "consommateur" ou "client". Par exemple, nous pouvons utiliser "/customers/<id>" comme endpoint pour accéder à la ressource client de VGBurger.

De plus, pour associer les commandes à cette ressource client, nous pouvons mettre en place le endpoint suivant : POST https://vbburger.com/customers/<id>/commands/<id_command>. Par exemple, le endpoint /customers/3/commands/5 renverrait la commande 5 pour le client 3.

Optimisez la recherche des données

Au fur et à mesure que les données augmentent, les endpoints listant les ressources peuvent devenir moins adaptés. Pour éviter les problèmes de performance, il est recommandé d'utiliser des paramètres dans l'URL pour filtrer les données.

Par exemple, si nous voulons rechercher tous les vgburgers avec de la mozarella, nous pouvons utiliser l'URL suivante : http://vgburger.com/commands?fromage=mozarella.
Si nous voulons filtrer encore plus en recherchant les vgburgers avec de la mozarella et un pain aux céréales, l'URL pourrait être :
http://vgburger.com/commands?fromage=mozarella&pain=cereale.

La réalisation de l'API !

Les étapes pour réaliser une API pour un site web de vente en ligne de burgers végétariens, comme VGBurger, sont les suivantes :

1. Analyser le cahier des charges pour comprendre les besoins spécifiques, tels que la vente de burgers végétariens avec un choix parmi 10 ingrédients.
2. Traduire les besoins en fonctionnalités requises, telles que la gestion des commandes, le menu des burgers, la gestion des ingrédients, etc.
3. Écrire chaque fonctionnalité sous forme d'URL et créer les endpoints correspondants.
4. Coder les fonctionnalités en écrivant le programme informatique qui les réalise.
5. Tester chaque fonctionnalité pour s'assurer de son bon fonctionnement.

Ces étapes vous permettront de construire votre API de manière méthodique et de garantir son bon fonctionnement.

Nous pouvons créer une API en utilisant différentes technologies et outils, en fonction de nos préférences, de notre expérience et des besoins spécifiques de notre projet. Voici quelques options courantes :

1. Express.js : Un framework JavaScript basé sur Node.js qui permet de construire des applications web et des API RESTful.
2. Flask : Un framework Python léger et flexible pour créer des API web.
3. Django : Un framework web Python complet offrant un support pour la création d'API RESTful.
4. Ruby on Rails : Un framework Ruby qui facilite la création d'API RESTful.

5. ASP.NET Core : Un framework de développement web multiplateforme basé sur .NET, qui permet de créer des API avec C#.
6. Laravel : Un framework PHP élégant pour la construction d'applications web et d'API.
7. Spring Boot : Un framework Java qui simplifie la création d'applications web et d'API, avec une configuration et un déploiement facilités.

Règles générales dans la conception d'API

Une API utilise des verbes (GET, POST, PUT, DELETE) et des noms (arguments de l'action) pour accéder à des ressources. Il est important pour nous de générer une variable d'état lors de l'exécution d'une action, qu'elle aboutisse à un succès ou à une erreur. En cas d'erreur, nous devons fournir une description précise et sans ambiguïté du problème.

Par exemple, si nous effectuons une requête GET sur l'URL http://vgburger.com/commands?fromage=brie, le code de réponse sera 2xx (succès de la requête, qu'elle renvoie ou non des ressources).

En revanche, si nous exécutons l'URL GET http://vbburger.com/com?fromage=brie, le code de réponse sera 4xx, car le endpoint /com n'existe pas.

Il est essentiel pour nous de gérer le résultat de chaque endpoint, qu'il soit réussi ou non. Cela déterminera la qualité de notre API.

Quelques conseils

- Une API est pour un client. Elle doit proposer des fonctions simples et évolutives.
- L'API doit être sécurisée (basique, clé d'API, OAuth).
- Préférez les noms concrets pour décrire vos ressources, évitez les verbes !
- Utilisez des minuscules pour le nommage des endpoints.
- L'API doit être documentée et illustrée par des exemples

Testez votre API

Tester notre API est une étape primordiale avant de la mettre en ligne. La qualité de notre API dépendra des tests que nous effectuons.

Nous pouvons utiliser des logiciels tels que Postman pour tester notre API. Voici quelques conseils importants pour les tests :

- Créez un cahier de tests pour assurer la qualité de l'API lors des nouvelles versions ou mises à jour.
- Vérifiez que chaque résultat de requête est correctement géré, qu'il s'agisse d'un succès ou d'un échec.
- Vérifiez que chaque résultat de requête renvoie la ressource attendue ou effectue l'action appropriée.
- Testez les performances pour les requêtes renvoyant un grand nombre de résultats, comme le endpoint /commands dans l'exemple de VGBurger.

Ces conseils nous permettront de garantir la qualité et les performances de notre API lors des tests.

Partagez votre API

Pour faire connaître notre API, nous pouvons utiliser différentes stratégies :

- Participer à des hackathons, qui sont des événements où des spécialistes se réunissent sur plusieurs jours pour collaborer sur des projets de programmation informatique.
- Organiser des défis pour les développeurs en ligne, ce qui peut attirer l'attention et susciter l'intérêt pour notre API.
- Créer un portail d'API qui hébergera notre API ainsi que d'autres API, fournissant de la documentation, des outils de recherche et d'autres ressources pour faciliter son adoption et son utilisation.