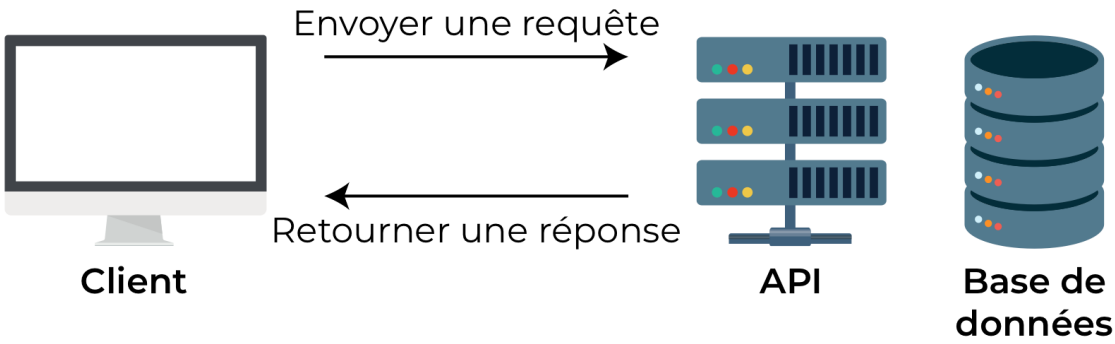
SOMMAIRE

1. Introduction :

API REST (Representational State Transfer Application Program Interface) est un style architectural qui permet aux logiciels de communiquer entre eux sur un réseau ou sur un même appareil. Le plus souvent les développeurs utilisent des API REST pour créer des services web. Souvent appelés services web RESTful, REST utilise des méthodes HTTP pour récupérer et publier des données entre un périphérique client et un serveur.

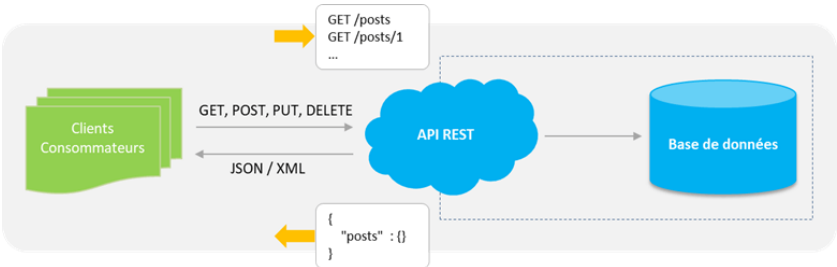
En utilisant le protocole HTTP, les API REST permettent aux logiciels d'un appareil de communiquer avec les logiciels d'un autre appareil (ou du même appareil) même s'ils utilisent des systèmes d'exploitation et des architectures différents. Le client peut demander des ressources avec un langage que le serveur comprend, et le serveur renvoie la ressource avec un langage que le client accepte. Le serveur renvoie la ressource au format JSON (JavaScript Object Notation), XML (Extensible Markup Language) ou texte, mais de nombreuses API prennent en charge d'autres langages.



2. Qu'est ce qu'une architecture REST.

REST est un ensemble de principes directeurs auxquels un développeur doit adhérer avant de pouvoir considérer son API comme « RESTful ». Les principes ne disent rien sur la façon de mettre en œuvre l'API.

- **Architecture client-serveur**: Les clients de l'API utilisent des appels HTTP pour demander une ressource (une méthode GET) ou envoyer des données au serveur (une méthode POST), ou l'une des autres méthodes HTTP prises en charge par l'API. GET et POST sont les méthodes les plus fréquemment utilisées, mais d'autres méthodes comme HEAD, PUT, PATCH, DELETE, CONNECT, OPTIONS ET TRACE peuvent également être prises en charge. La documentation de l'API montre les méthodes disponibles prises en charge par l'API.



- **Sans état**: Une application sans état ne maintient pas de connexion ni ne stocke d'informations entre deux requêtes du même client. Un client fait une requête, l'API exécute l'action définie dans la requête et répond. Une fois que l'API a répondu, elle se déconnecte et ne conserve aucune information sur le client dans sa mémoire active. L'API traite chaque requête comme une première demande.

- **Interface uniforme**: Le client interagit avec le serveur selon une manière définie, indépendamment de l'appareil ou de l'application.

- **Identification des ressources**: : L'API doit avoir un URI (identifiant de ressource uniforme) spécifique pour chaque ressource,

3. Qu'est ce qu'une ressource web.

Une ressource web est essentiellement tout ce avec quoi un client peut interagir sur le web. Le terme peut s'appliquer à un fichier tel qu'un document Word, une image, du HTML ou une vidéo, mais la ressource peut être plus abstraite et inclure des éléments réels. Une ressource peut également être un service tel que Google Maps ou des services financiers.

C'est le développeur de l'API qui décide quels formats seront pris en charge pour la réponse. Par exemple, un serveur peut répondre avec JSON, XML ou texte. L'API doit pouvoir formater la réponse en fonction des besoins du client.

4. Exemples d'API REST.

Presque tout ce qui se passe sur Internet implique des API. Les API fonctionnent en arrière-plan pour effectuer des tâches telles que la validation d'adresses, le traitement des cartes de crédit, la réservation ou la planification de rendez-vous.

Voici quelques exemples de sites Web qui proposent des API :

- OpenAI : OpenAI propose une API pour son modèle de langage GPT-3. Vous pouvez l'utiliser pour générer du texte, traduire des langues, résumer des documents, etc.
- Twitter : L'API de Twitter vous permet d'accéder à diverses fonctionnalités de la plateforme, comme la recherche de tweets, la publication de nouveaux tweets, la récupération de profils d'utilisateurs, etc.
- Google Maps : L'API de Google Maps vous permet d'intégrer des fonctionnalités de Google Maps dans votre application, comme la recherche de lieux, l'affichage de cartes, la création d'itinéraires, etc.
- Spotify : L'API de Spotify vous permet d'accéder à des informations sur les chansons, les albums, les artistes et les playlists. Vous pouvez également contrôler la lecture de la musique sur les appareils des utilisateurs.
- Weather API : Il existe plusieurs API météorologiques, comme OpenWeatherMap, Weatherstack, et Meteostat, qui vous permettent d'accéder à des informations météorologiques en temps réel, des prévisions, des données historiques, etc.
- NASA : L'API de la NASA vous donne accès à une multitude de données spatiales, y compris des images de la journée, des données sur les astéroïdes, la Station spatiale internationale, Mars Rover, etc.

- Stripe : Stripe propose une API pour intégrer des fonctionnalités de paiement dans votre application web ou mobile.
- Twilio : Twilio propose une API pour l'envoi de SMS, la réalisation d'appels téléphoniques, la vidéoconférence, et d'autres fonctionnalités de communication.

5. Exemples d'API codée en Javascript sur NodeJs.

```
const express = require('express');
const mysql = require('mysql');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

const conn = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: 'root',
  database: 'METEO',
  port:8889
});

app.get('/temperature', (req, res) => {
  let sql = 'SELECT * FROM TEMPERATURE';
  if(req.query.ID) {
    sql += ` WHERE ID=${req.query.ID} LIMIT 1`;
  }
  conn.query(sql, (err, results) => {
    if(err) throw err;
    res.send(results);
  });
});

app.post('/temperature', (req, res) => {
  const temp = req.body.TEMP;
  const sql = `INSERT INTO temperature(TEMP) VALUES('${temp}')`;
  conn.query(sql, (err, result) => {
    if(err) throw err;
    res.send({status: 1, status_message: 'Temperature added with success.'});
  });
});

app.put('/temperature/:ID', (req, res) => {
  const temp = req.body.temp;
  const sql = `UPDATE TEMPERATURE SET TEMP='${temp}' WHERE id=${req.params.ID}`;
  conn.query(sql, (err, result) => {
    if(err) throw err;
    res.send({status: 1, status_message: 'Temperature updated with success.'});
  });
});

app.delete('/temperature/:ID', (req, res) => {
  const sql = `DELETE FROM TEMPERATURE WHERE id=${req.params.ID}`;
  conn.query(sql, (err, result) => {
    if(err) throw err;
    res.send({status: 1, status_message: 'Temperature deleted with success.'});
  });
});

app.listen(3000, () => {
  console.log('Server started on port 3000...');
});
```

6. Requêtes Curl :

Requête GET en utilisant curl :

```
curl -X GET http://localhost:3000/temperature

curl -X GET http://localhost:3000/temperature?ID=857
```

Requête POST en utilisant curl :

```
curl -X POST -H "Content-Type: application/json" -d '{"TEMP": "48"}'
http://localhost:3000/temperature
```

Requête PUT en utilisant curl :

```
curl -X PUT -H "Content-Type: application/json" -d '{"temp": "11"}'
http://localhost:3000/temperature/857
```

Requête DELETE en utilisant curl :

```
curl -X DELETE http://localhost:3000/temperature/858
```