Javascript, Web API et JSON

Partie 2 - Web API et JSON

Valentin RIBEZZI

Sommaire

01

Javascript, Web API et JSON

Partie 2 - Web API et JSON

Introduction aux API Web

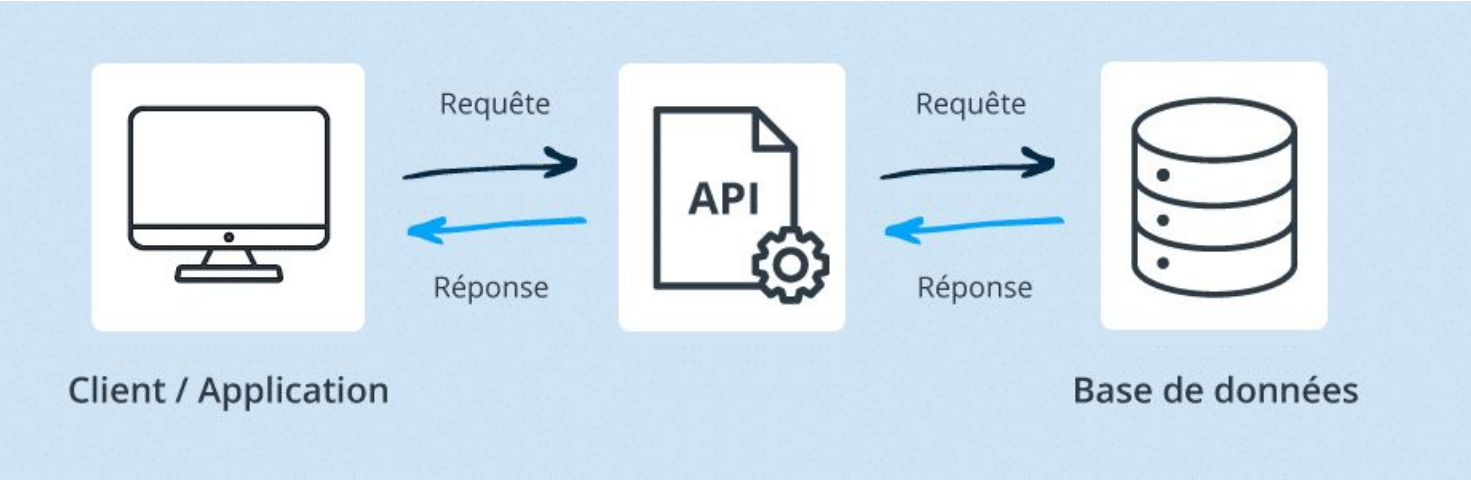
Introduction aux API Web

Définition et concepts de base

Qu’est-ce qu’une API ?

Introduction aux API Web

Définition et concepts de base



Interface qui permet à différentes applications de communiquer entre elles.

Introduction aux API Web

Définition et concepts de base

Protocole basé sur XML, plus lourd et strict.

Style architectural basé sur des principes simples, utilisant souvent JSON.

Introduction aux API Web

Définition et concepts de base

Caractéristique	REST	SOAP
Architecture	Style architectural	Protocole
Protocoles	HTTP	HTTP, SMTP, TCP, etc…
Format de messages	JSON, XML	XML
Simplicité des messages	Plus légers et simples	Plus lourds et complexes
Fonctionnement	*Stateless*	*Stateful ou stateless*
Méthodes	HTTP (GET, POST, PUT, etc…)	Enveloppe SOAP
Performances	Souvent plus rapide	Généralement plus lent
Scalabilité	Meilleure scalabilité	Moins scalable
Popularité	Largement utilisé	Utilisé pour des besoins spé.
Sécurité	HTTPS, OAuth, JWT	WS-Security

Introduction aux API Web

Définition et concepts de base

Différence entre Stateless et Stateful

Stateless

Chaque requête est indépendante. Toutes les informations nécessaires sont envoyées à chaque requête, et le serveur ne conserve pas l'état entre les requêtes.

Exemple : Un distributeur automatique qui exécute une action sans prendre en compte les anciennes transactions effectuées.

Stateful

Le serveur conserve des informations sur les interactions précédentes. Les requêtes suivantes peuvent dépendre de l'état conservé par le serveur, ce qui nécessite une gestion de session.

Exemple : Un panier sur un site stocke les articles ajoutées précédemment.

Introduction aux API Web

Définition et concepts de base

Il existe plusieurs composants pour illustrer le fonctionnement des API Web.

Endpoint

Requêtes et Réponses

Headers

Introduction aux API Web

Définition et concepts de base

Il existe plusieurs composants pour illustrer le fonctionnement des API Web.

Endpoints

Les URL où l'API peut être accédée.

Requêtes et Réponses

Les clients envoient des requêtes et les serveurs envoient des réponses.

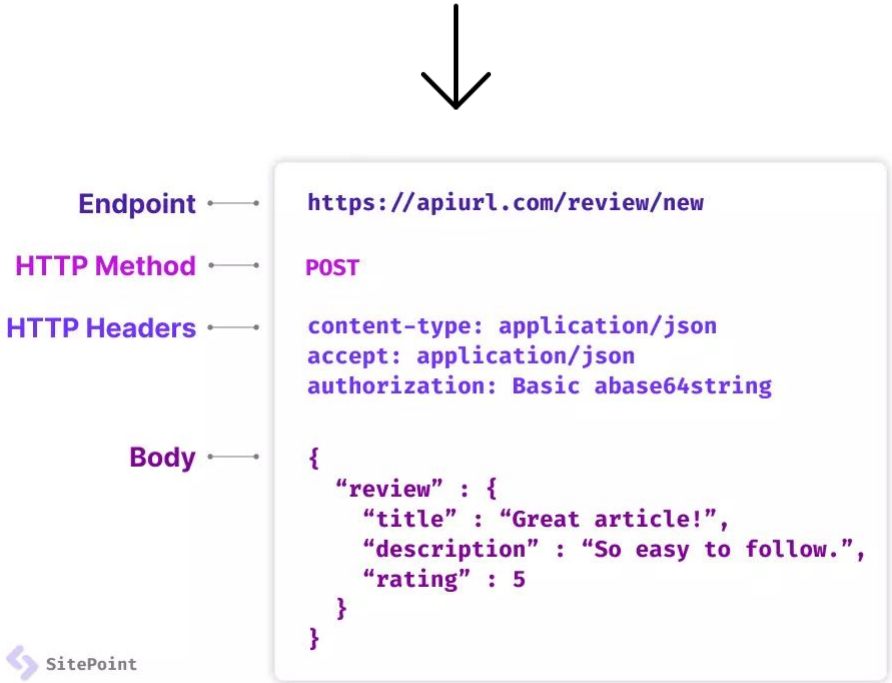
Headers

Métadonnées pour les requêtes et réponses (authentification, type de contenu, etc.).

Introduction aux API Web

Définition et concepts de base

Il existe plusieurs composants pour illustrer le fonctionnement des API Web.



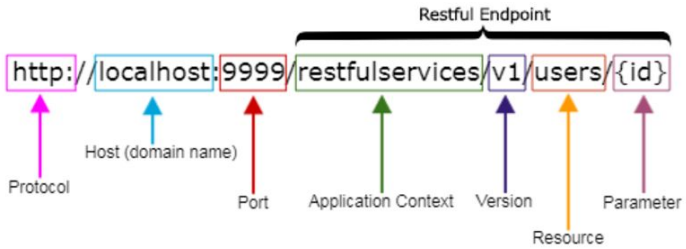
Parlons maintenant de REST.

C'est le type d'API que nous allons utiliser pour le reste du cours.

Introduction aux API Web

Architecture REST

Les principes de REST	
Stateless	Chaque requête contient toutes les informations nécessaires.
Resource-based	Manipulation des ressources via URI.
URI (Uniform Resource Identifier)	Identificateur unique pour chaque ressource.



URI

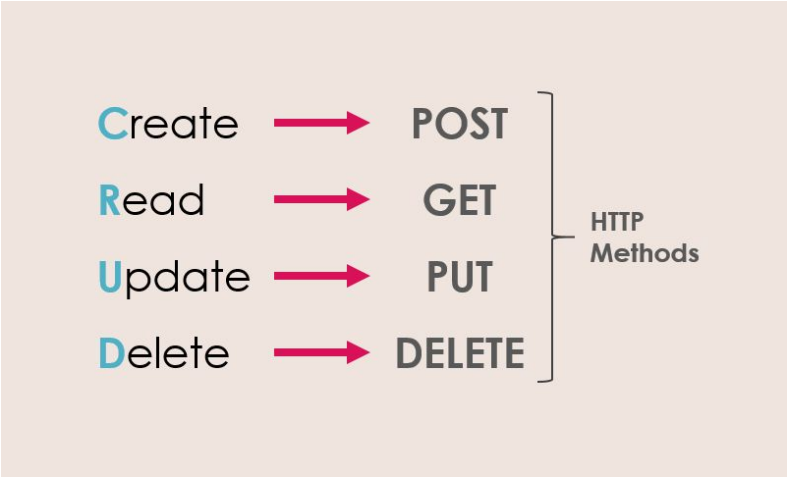
Mais concrètement, comment utiliser cet URI pour agir sur des données ?

Introduction aux API Web

Architecture REST

Nous allons utiliser des méthodes HTTP.

Il en existe plusieurs, mais nous allons voir les 4 principales.



Voyons chaque méthode avec un exemple d'endpoint.

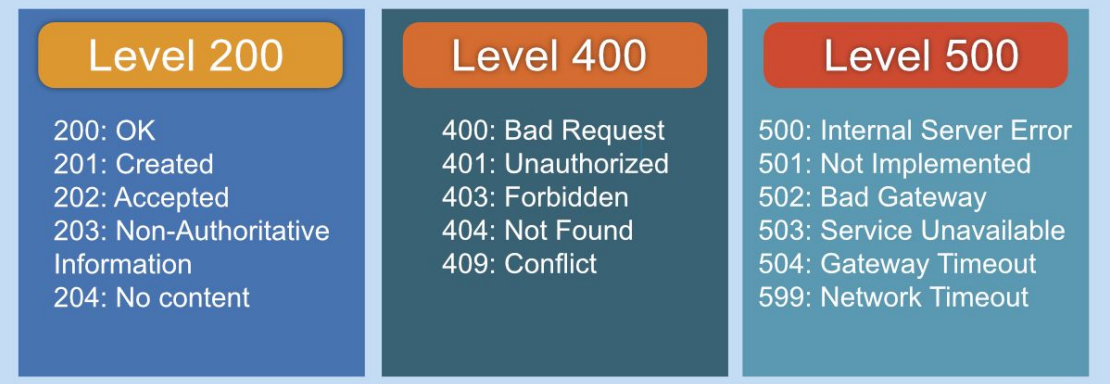
Introduction aux API Web

Architecture REST

Endpoint / URI	Méthode	Explications / Rendus
https://monapi/api/employes	GET	On récupère la liste des employés.
https://monapi/api/employes/1	GET	On récupère la liste d'un employé en fonction de l'identifiant spécifié (1).
https://monapi/api/employes	POST	On crée un nouvel employé.
https://monapi/api/employes/2	PUT	On modifie un employé en fonction de l'identifiant spécifié (2).
https://monapi/api/employes/3	DELETE	On supprime un employé en fonction de l'identifiant spécifié (2).

Introduction aux API Web

Architecture REST



Les statuts HTTP les plus courants

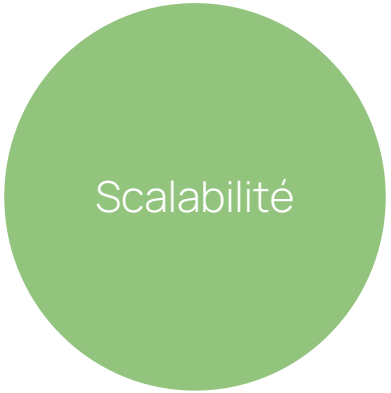
200 OK : Succès.
201 Created : Ressource créée avec succès.
400 Bad Request : Requête mal formée.
401 Unauthorized : Authentification requise.
404 Not Found : Ressource non trouvée.
500 Internal Server Error : Erreur du serveur.

Quels sont les avantages des API ?

Introduction aux API Web

Les avantages

Permet la communication entre différentes technologies et plateformes.



Facilement extensible pour gérer plus de demandes.

Permet d'intégrer de nouvelles fonctionnalités sans affecter les systèmes existants.

02

Javascript, Web API et JSON

Partie 2 - Web API et JSON

JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation)

Introduction

Le JSON

Format léger d'échange de données, facile à lire et à écrire pour les humains et les machines.


```
{
  "longitude": 47.60,
  "latitude": 122.33,
  "forecasts": [
    {
      "date": "2015-09-01",
      "description": "sunny",
      "maxTemp": 22,
      "minTemp": 20,
      "windSpeed": 12,
      "danger": false
    },
    {
      "date": "2015-09-02",
      "description": "overcast",
      "maxTemp": 21,
      "minTemp": 17,
      "windSpeed": 15,
      "danger": false
    },
    {
      "date": "2015-09-03",
      "description": "raining",
      "maxTemp": 20,
      "minTemp": 18,
      "windSpeed": 13,
      "danger": false
    }
  ]
}
```

JSON (JavaScript Object Notation)

Introduction

La structure de JSON

Valeur

Une valeur associé à une clé. “latitude” : “256.2039”

Objet

Ensemble de paires clé-valeur. { "nom": "John", "âge": 30 }

Tableau

Liste ordonnée de valeurs. ["rouge", "vert", "bleu"]

JSON (JavaScript Object Notation)

Introduction

Exemple de JSON

```
{
    "menu": {
        "id": "file",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New", "onclick": "CreateNewDoc()" },
                { "value": "Open", "onclick": "OpenDoc()" },
                { "value": "Close", "onclick": "CloseDoc()" }
            ]
        },
        "informations": [".html", "15mo"]
    }
}
```

JSON (JavaScript Object Notation)

Introduction

Équivalent au format XML

```
<menu id="file" value="File">
   <popup>
      <menuitem value="New" onclick="CreateNewDoc()"/>
      <menuitem value="Open" onclick="OpenDoc()"/>
      <menuitem value="Close" onclick="CloseDoc()"/>
   </popup>
 </menu>
```

03

Javascript, Web API et JSON

Partie 2 - Web API et JSON

Utilisation des API Web

Utilisation des API Web

Appel d'API REST

Utilisation de cURL pour tester les API

```
curl -X GET "https://api.exemple.com/ressource"
```

Utilisation des API Web

Appel d'API REST

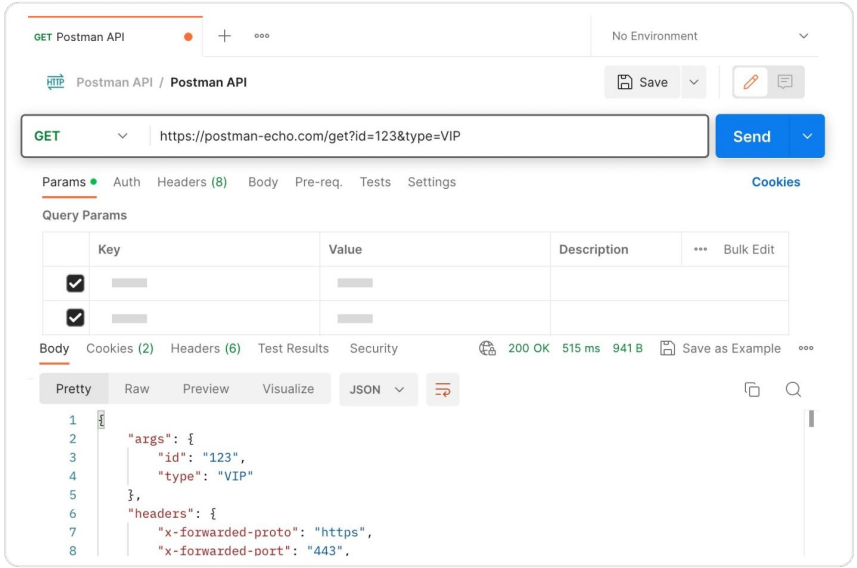
Utilisation de cURL pour tester les API

```
curl -X POST "https://api.exemple.com/ressource"
     -H 'Content-Type: application/json'
     -d '{"login":"my_login","password":"my_password"}'
```

Utilisation des API Web

Appel d'API REST

Utilisation de Postman pour tester les API



Utilisation des API Web

Intégration d'API dans une Application

En Nuxt :

```
<template>
  <div>
    <h1>Data from API</h1>
    <div v-if="data">
      <p>{{ data }}</p>
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      data: null
    };
  },
  async fetch() {
    const response = await this.$axios.$get('https://api.example.com/data');
    this.data = response;
  }
};
</script>
```

Utilisation des API Web

Intégration d'API dans une Application

En React :

```
import React, { useEffect, useState } from 'react';

function App() {
  const [data, setData] = useState(null);

  useEffect(() => {
    fetch('https://api.example.com/data')
      .then(response => response.json())
      .then(data => setData(data))
      .catch(error => console.error('Error fetching data:', error));
  }, []);

  return (
    <div>
      <h1>Data from API</h1>
      <div>{data && <p>{JSON.stringify(data)}</p>}</div>
    </div>
  );
}

export default App;
```

Utilisation des API Web

Intégration d'API dans une Application

En Javascript :

```
<!DOCTYPE html>
<html lang="en">
	<head>
		<meta charset="UTF-8">
		<meta name="viewport" content="width=device-width, initial-scale=1.0">
		<title>Fetch API Example</title>
	</head>

	<body>
		<h1>Data from API</h1>
		<div id="data-container"></div>

		<script>
			fetch('https://api.example.com/data')
				.then(response => response.json())
				.then(data => {
					const dataContainer = document.getElementById('data-container');
					dataContainer.textContent = JSON.stringify(data);
				})
				.catch(error => console.error('Error fetching data:', error));
		</script>
	</body>
</html>
```

04

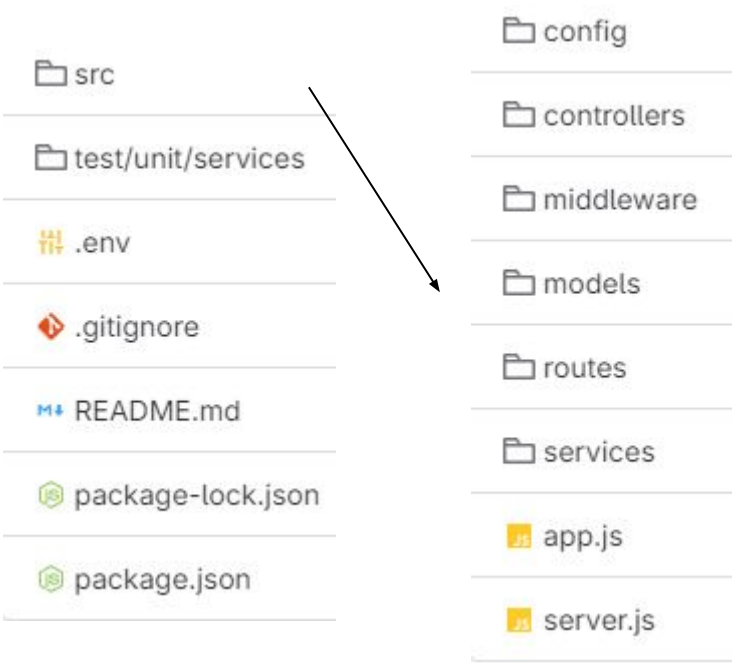
Javascript, Web API et JSON

Partie 2 - Web API et JSON

Création d'une API REST

Création d'une API REST

Arborescence du projet d'API



app.js : Point d'entrée de l'API, qui sert notamment à "générer" les routes

server.js : Fichier servant à configurer le serveur de l'API *(optionnel)*

config : dossier avec les fichiers servant à gérer des configurations (base de données, swagger…)

controllers : Interface entre les services et les routes pour les actions sur les données

middleware : Dossier regroupant les méthodes et fonctionnalités communes aux dossiers (La gestion des tokens par exemple)

models : Dossier servant à regrouper les modèles qui définissent le "design" des tables de la base de données (un modèle par table).

routes : Dossier servant à définir les fichiers pour créer chaque route/endpoint de votre API.

services : Dossier regroupant les fichiers interagissant directement avec la base de données

Création d’une API REST

Voyons cela directement sur un projet et étudions sa structure !

05

Javascript, Web API et JSON

Partie 2 - Web API et JSON

Sécurité et authentification

Sécurité et authentification

Comme pour les autres projets informatiques, les API sont confrontés à différentes menaces.

Sécurité et authentification

Les attaques par injection permettent à un attaquant d'exécuter des commandes SQL/MongoDB arbitraires en manipulant les entrées utilisateur.

Une mauvaise gestion de l'authentification et de l'autorisation peut permettre à des utilisateurs non autorisés d'accéder à des données sensibles ou d'exécuter des actions non autorisées.

Injection de scripts malveillants dans les pages web.

Et beaucoup, beaucoup, BEAUCOUP d'autres...

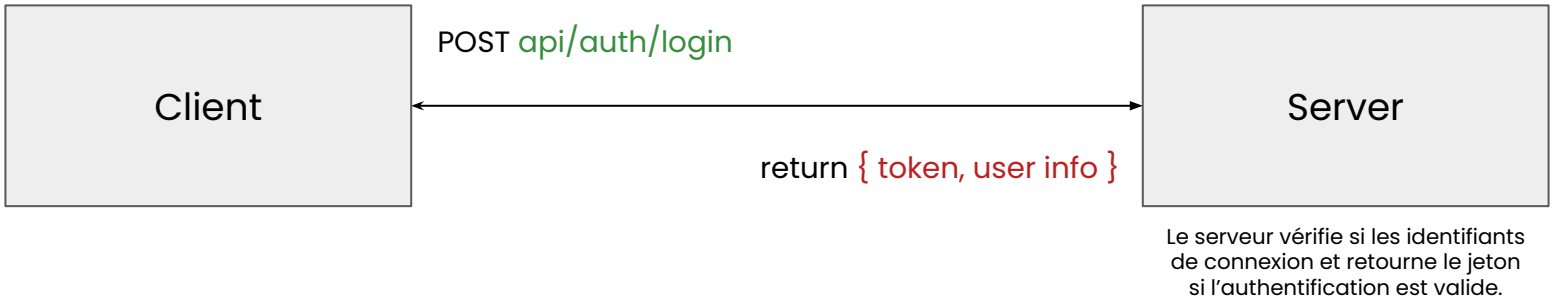
Sécurité et authentification

Compte tenu du temps pour la formation, nous allons nous concentrer sur l'authentification avec des jetons !

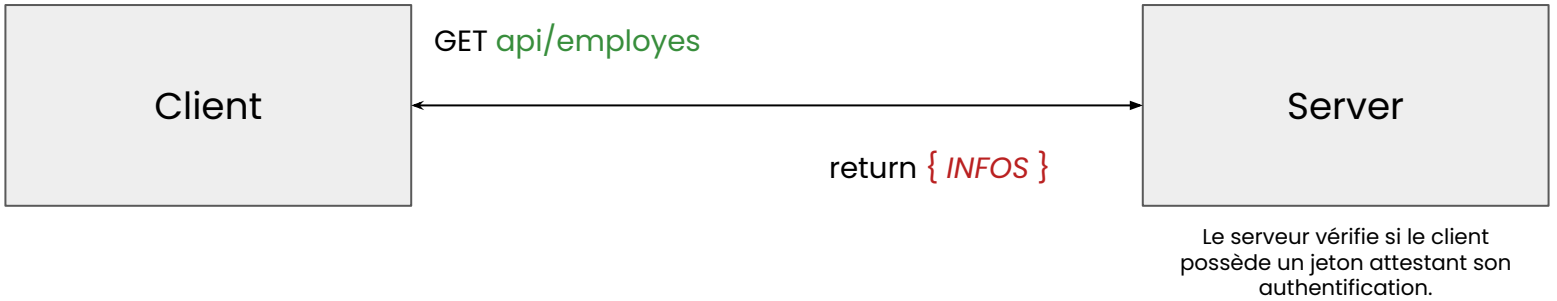
Nous allons créer des jetons d'authentifications avec JWT (JSON Web Token).

Sécurité et authentification

On crée un jeton lors de l'authentification



On ré-utilise ce jeton lors de l'appel de routes pour vérifier l'authentification



Sécurité et authentification

Middleware permettant de vérifier un token :

```
// Importation des modules
const jwt = require('jsonwebtoken');

// Middleware pour vérifier le token
function verifyToken(req, res, next) {
    const token = req.header('Authorization');
    if (!token) return res.status(401).json({ error: 'Accès refusé.' });
    try {
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        req.userId = decoded.userId;
        next();
    } catch (error) {
        res.status(401).json({ error: 'Invalid token' });
    }
};

// Exportation du middleware
module.exports = verifyToken;
```

Sécurité et authentification

Utilisation du middleware en pratique :

```
// GET
app.get('/entreprises', verifyToken, entrepriseController.getAllEntreprises);

// GET by id
app.get('/entreprises/:id', verifyToken, entrepriseController.getEntrepriseById);

// POST
app.post('/entreprises', verifyToken, entrepriseController.createEntreprise);

// PUT
app.put('/entreprises/:id', verifyToken, entrepriseController.updateEntreprise);

// DELETE
app.delete('/entreprises/:id', verifyToken, entrepriseController.deleteEntreprise);
```