Node JS API: Construire une API REST avec Node JS et Express

Ceci est un guide pour vous aider à comprendre comment construire une API REST.

Etape 1 : Créer un serveur Express

Votre Node JS API est avant tout un serveur web à l'écoute des requêtes HTTP entrantes. Pour démarrer ce serveur web, nous allons utiliser le framework Express.

Démarrage du projet Node JS API

1. Créez votre répertoire de votre future API et naviguez à l'intérieur
2. Saisissez la commande npm init et répondez aux questions
3. Créer un fichier index.js

Vous aurez maintenant dans votre répertoire un fichier package.json, qui va reprendre différentes informations du projet et qui contiendra les dépendances qu'on va y installer.

Ajout d'Express à notre Node JS API

Retournez maintenant à votre terminal et tapez la commande suivante: *npm install express*

Cette commande a pour but de télécharger depuis la registry NPM puis d'installer la librairie express ainsi que l'ensemble des librairies dont express a besoin pour fonctionner dans votre répertoire de travail, dans le répertoire node_modules. NPM va également l'ajouter dans votre package.json dans l'objet dependencies.

Création du serveur Express dans notre fichier index.js

Maintenant qu'Express est disponible dans notre projet, nous pouvons créer le serveur. Commençons par intégrer la librairie express dans notre fichier index.js:

```
const express = require('express')

const app = express()
```

Le *require('express')* est une façon d'importer la librairie *express* et ses fonctions dans notre code. La constante *app* est l'instanciation d'un objet Express, qui va contenir notre serveur ainsi que les méthodes dont nous aurons besoin pour le faire fonctionner.

Pour le moment, votre serveur est préparé mais pas encore lancé. Si vous vous rendez sur *localhost:8080* depuis votre navigateur, vous devriez avoir une erreur.

Lorsque vous cherchez à joindre votre serveur alors qu'il n'est pas encore lancé, votre navigateur vous retourne une erreur.

Pour que notre serveur puisse être à l'écoute il faut maintenant utiliser la méthode listen fournie dans app et lui spécifier un port. Le plus souvent en développement nous utilisons 8080, 3000 ou 8000. Ça n'a pas d'importance tant que vous n'avez pas d'autres applications qui tournent localement sur ce même port.

```
app.listen(8080, () => {  console.log('Serveur à l'écoute)      })
```

En lançant la commande node index.js dans votre terminal, vous verrez qu'il affichera que votre serveur est à l'écoute. Cela veut dire que tout fonctionne bien. S'il y a une erreur, vous aurez droit à un message d'erreur sur votre terminal.

Si vous vous rendez sur votre navigateur à l'adresse localhost:8080 (ou l'autre port que vous aurez choisi), votre serveur répond à votre navigateur. N'ayant pour l'instant aucune route de configurée, il vous retourne cette erreur Cannot GET / mais il est bel et bien fonctionnel.

Etape2 :Définir une ressource et ses routes

Pour notre exemple, nous prendrons le cas d'une société exploitant des parkings de longue durée et qui prend des réservations de la part de ses clients. Nous aurons besoin des fonctionnalités suivantes:

- Créer un parking
- Lister l'ensemble des parkings
- Récupérer les détails d'un parking en particulier
- Supprimer un parking
- Prendre une réservation d'une place dans un parking
- Lister l'ensemble des réservations
- Afficher les détails d'une réservation en particulier
- Supprimer une réservation

Le but de ce guide est de vous aider à comprendre le bon fonctionnement d'une API. Nous n'allons pas connecter de vraie base de données dans ce guide. Nous allons à la place utiliser un fichier JSON contenant un échantillon de données pour manipuler notre API.

```
[
    {
        "id": 1,
        "name":"Parking 1",
        "type": "AIRPORT",
        "city": "ROISSY EN FRANCE"
    },
    {
        "id": 2,
        "name":"Parking 2",
        "type": "AIRPORT",
        "city": "BEAUVAIS"
    },
    {
        "id": 3,
        "name":"Parking 3",
        "type": "AIRPORT",
        "city": "ORLY"
    },
    {
        "id": 4,
        "name":"Parking 4",
        "type": "AIRPORT",
        "city": "NICE"
    },
    {
        "id": 5,
        "name":"Parking 5",
        "type": "AIRPORT",
        "city": "LILLE"
    }
]
```

Créer le fichier json parkings.json et placer-le à la racine de votre projet

```
const express = require('express')

const app = express()

app.get('/parkings', (req,res) => {

res.send("Liste des parkings")})

app.listen(8080, () => {

 console.log("Serveur à l'écoute")})
```

Dans ce code, à l'arrivée d'une requête GET sur l'URL *localhost:8080/parkings*, le serveur a pour instruction d'envoyer la String « Liste des parkings ».

Coupez votre serveur node s'il tourne encore (avec la commande ctrl+c dans le terminal) et relancez la commande node index.js pour prendre en compte les modifications.

Maintenant que notre route fonctionne et est capable de recevoir la requête entrante, nous allons pouvoir renvoyer la donnée des parkings au lieu d'avoir simplement une chaîne de caractères:

```
const express = require('express')

const app = express()

const parkings = require('./parkings.json')

app.get('/parkings', (req,res) => {

res.status(200).json(parkings)})

app.listen(8080, () => {    console.log("Serveur à l'écoute")})
```

Votre API retourne la donnée que vous lui avez demandé