NodeJS - API

Introduction

Aujourd'hui, nous allons découvrir les fondements de la création d'une API à l'aide de **Node.js** pour une application de **gestion de tâches**, communément appelée une "To-Do List".

Nous allons explorer les étapes clés pour concevoir, mettre en œuvre une API qui permettra aux utilisateurs de créer, lire, mettre à jour et supprimer des tâches de leur liste de tâches.

Dans votre répertoire nommé "**runtrack-nodeJS**", créez un dossier "**jour2**" qui doit contenir votre API. N'oubliez pas d'envoyer vos modifications dès qu'un job est avancé ou terminé et mettez des commentaires explicites lors de vos commits.

Étape 1 : Ordonner son projet

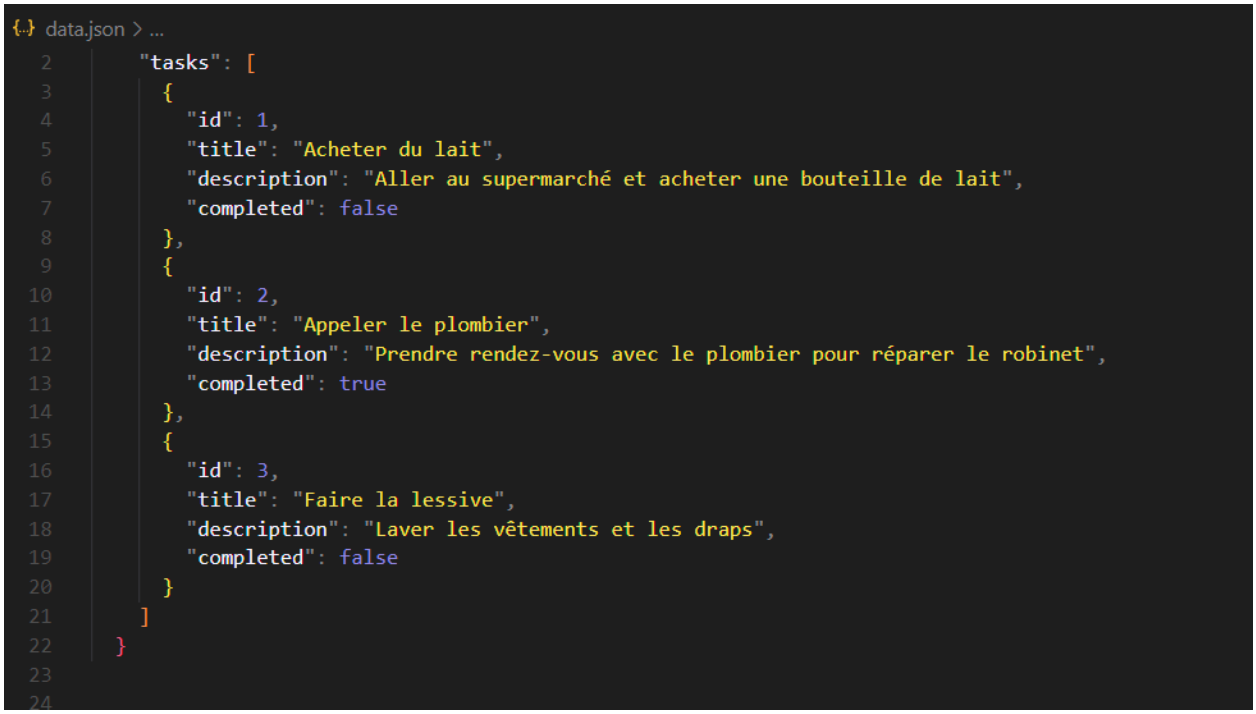
Créez les fichiers suivant en prenant le temps de penser à organiser celui-ci :

- ➔ "**index.js**" ⇒ Point d'entrée de votre API
- ➔ "**server.js**" ⇒ Contenant la configuration du serveur ainsi que le démarrage

➔ “**routes.js**” ⇒ Contenant l’ensemble des routes de votre API

➔ “**data.json**” ⇒ Fichier JSON contenant vos données

Chaque tâche doit être enregistrée dans le fichier “**data.json**”, et doit posséder un **id**, **un titre**, **une description** et **un statut**.



Étape 2 : Création du serveur

Comme vous l'aurez compris, Node.js est un environnement d'exécution JavaScript côté serveur, ce qui implique qu'il nécessite un serveur pour écouter les requêtes entrantes et exécuter du code JavaScript en réponse à ces requêtes.

Afin de rendre votre code modulaire et réutilisable, créer votre serveur dans le fichier nommé “**server.js**” et l'exporter vers le fichier “**index.js**”. Ceci permet de séparer la logique de création du serveur de son utilisation. Pour des raisons de clarté et de maintenabilité, évitez de surcharger le fichier “**index.js**” avec le code du serveur.

Étape 3 : Les routes

Mettre en place les routes suivantes respectant les bonnes pratiques de nommage des routes :

- ➔ **/tasks** ⇒ [**GET**] : Récupérer toutes les tâches de la liste.
- ➔ **/tasks** ⇒ [**POST**] : Créer une nouvelle tâche.
- ➔ **/tasks/:id** ⇒ [**PUT**] : Mettre à jour une tâche existante.
- ➔ **/tasks/:id** ⇒ [**DELETE**] : Supprimer une tâche existante.

Ces routes doivent être définies dans le fichier "**routes.js**" de votre projet et ensuite importées dans votre fichier "**index.js**" pour être utilisées par votre application.

Étape 4 : Logique métier

Maintenant que vos routes sont définies, écrivez la logique nécessaire au bon fonctionnement de chacune d'elles. Par exemple, la route **/tasks** doit pouvoir lire les données à partir du fichier JSON. Une fois la logique mise en place, assurez-vous de renvoyer une réponse appropriée au client, en incluant éventuellement un code de statut HTTP correspondant (par exemple, 200 pour succès, 201 pour création réussie, etc.).
Notez que certaines routes, telles que celles impliquant des actions de création, mise à jour ou suppression de données, ne peuvent pas être testées directement dans un navigateur.

Étape 5 : Postman

Vous avez fini de créer vos routes, mais vous ne pouvez pas toutes les tester ? Vous êtes donc à la bonne étape ! Postman va être une des solutions.

Postman permet de tester facilement et rapidement vos routes API, vous permettant ainsi de vérifier si la réponse renvoyée par le serveur correspond à vos attentes.

Commencez par vous rendre sur Postman et créez un compte si vous n'en avez pas déjà un. Ensuite, créez un espace de travail (workspace) pour votre projet. Une fois cela fait, ajoutez vos routes dans Postman. Assurez-vous de sélectionner les informations correspondantes pour chaque requête, telles que le verbe HTTP, l'URL et le corps de la requête.

Envoyez chaque requête individuellement et vérifiez les réponses renvoyées par le serveur. Assurez-vous que les codes de statut HTTP sont corrects et que les données renvoyées correspondent à ce que vous attendez pour chaque action de votre API.

Pour aller plus loin...

Implémenter une gestion des erreurs avec le bon code de statut HTTP ainsi qu'un message sur l'ensemble des routes.

Compétences visées

- ➔ NodeJS
- ➔ API

Rendu

Le projet est à rendre sur
https://github.com/prenom-nom/runtrack-nodeJs.

Base de connaissances

- Module HTTP
- Comment créer un serveur web avec node.js ?
- Qu'est-ce qu'une API ?
- Les codes HTTP
- Les méthodes HTTP
- Module fs