Université Sultan Moulay Slimane
École Nationale des Sciences Appliquées
-Khouribga-

Filière : Informatique et ingénierie des données

TP (Express.JS)

Élève :
Salwa Faraj

Enseignant
Amal Ourdo

21 octobre 2024

Table des matières

Table des figures

1 Introduction

L'introduction de ce TP vise à explorer les principes fondamentaux du développement d'applications web à l'aide d'Express.js, un framework populaire pour Node.js. Dans ce projet, nous allons créer une application CRUD (Créer, Lire, Mettre à jour, Supprimer) qui permettra de gérer une collection d'éléments. Nous mettrons en place des points d'entrée HTTP pour interagir avec ces éléments, en utilisant les méthodes GET, POST, PUT et DELETE. À travers ce TP, nous apprendrons non seulement à manipuler des données, mais aussi à comprendre les bonnes pratiques en matière de conception d'API RESTful, tout en renforçant nos compétences en JavaScript et en développement backend. Cette expérience sera essentielle pour développer des applications plus complexes et enrichissantes dans le futur.

2 Partie 1

1. **Qu'est-ce qu'Express.js et que peut-on en faire ?**

Express.js est un framework d'application web minimal et flexible pour Node.js, conçu pour simplifier le processus de construction d'applications web et d'API. Il fournit un ensemble robuste de fonctionnalités pour les applications web et mobiles, facilitant la gestion du routage, le traitement des requêtes et des réponses, et le service de fichiers statiques. Express.js est construit sur Node.js, tirant parti de sa nature asynchrone pour gérer efficacement plusieurs requêtes.

Que peut-on faire avec Express.js ?

- **Applications Web** : Vous pouvez créer des applications web complètes avec Express, y compris l'authentification des utilisateurs, la gestion des sessions et le templating.
- **APIs RESTful** : Express est souvent utilisé pour créer des APIs RESTful, permettant à différentes parties d'un système de communiquer via HTTP. Cela est particulièrement courant dans les applications à page unique (SPA) et les applications mobiles.
- **Applications en Temps Réel** : Avec des bibliothèques supplémentaires comme Socket.io, vous pouvez créer des applications en temps réel, telles que des applications de chat.
- **Applications à Page Unique (SPA)** : Servir des frameworks côté client comme Angular, React ou Vue.js en fournissant une API backend.
- **Service de Fichiers Statiques** : Servir facilement des fichiers statiques, tels que HTML, CSS, JavaScript et images.
- **Intégration de Middleware** : Intégrer facilement des middleware tiers pour des fonctionnalités supplémentaires, telles que la journalisation, la sécurité et la gestion des erreurs.

2.Qu'est-ce que le Middleware et comment est-il utilisé dans Express.js ?

Middleware dans Express.js est une fonction qui a accès à l'objet de requête (req), à l'objet de réponse (res), et à la prochaine fonction middleware dans le cycle de requête-réponse de l'application. Les middlewares peuvent exécuter du code, modifier les objets de requête et de réponse, terminer le cycle de requête-réponse, ou appeler la prochaine fonction middleware dans la pile.

Utilisation des Middlewares dans Express.js

Les middlewares sont utilisés pour ajouter des fonctionnalités à une application Express de manière modulaire. Ils peuvent être appliqués à l'ensemble de l'application ou à des routes spécifiques.

Exemples de Middleware dans Express.js :

Middleware de Journalisation : Ce middleware enregistre les détails de chaque requête entrante, ce qui est utile pour le débogage.

```
57  app.use(express.json());
58  app.post('/data', (req, res) => {
59      console.log(req.body);
60      res.send('Données reçues !');
61  });
62
```

FIGURE 1 – Exemple de middleware de journalisation

3 Partie 2 :Créer une application CRUD simple

3.1 Question 1 :

FIGURE 2 – Projet

3.2 Question 2 :

Pour initialiser un projet Node.js, la première étape consiste à créer un fichier package.json, qui contient des informations essentielles sur le projet, telles que son nom, sa version, sa description et ses dépendances. Cela se fait en utilisant la commande npm init dans le terminal. Cette commande guide l'utilisateur à travers une série de questions pour configurer les détails du projet

FIGURE 3 – Initialisation

3.3 Question 3 :

Pour ajouter le module Express à notre projet Node.js, nous utilisons la commande npm install express. Cette commande télécharge et installe Express, un framework web minimaliste, en tant que dépendance.

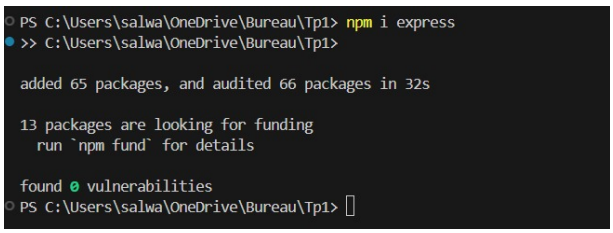


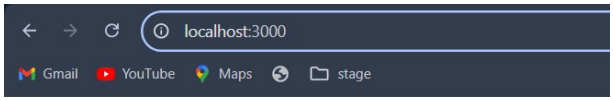
FIGURE 4 – Instalation de Express

3.4 Question 4 :

Dans ce code, j'ai d'abord importé le module Express, qui est essentiel pour développer des applications web en Node.js. Ensuite, j'ai créé une instance de l'application Express. J'ai défini un point d'extrémité pour la méthode GET sur la racine de l'application (/), qui envoie le message "Bonjour Salwa Faraj" lorsque cette route est accédée. J'ai ensuite

spécifié le port 3000 sur lequel le serveur écoutera les requêtes entrantes. Enfin, j'ai démarré le serveur en l'écoutant sur le port défini, avec un message dans la console indiquant que le serveur est en cours d'exécution et précisant l'URL à laquelle il est accessible.

FIGURE 5 – Enter Caption



Bonjour Salwa Faraj !

FIGURE 6 – Navigateur

3.5 Question 5 :

Dans ce code, j'ai d'abord importé le module Express et créé une instance de l'application. Ensuite, j'ai utilisé un middleware pour analyser les corps des requêtes au format JSON. J'ai initialisé un tableau items pour stocker les éléments ajoutés. J'ai défini un point d'extrémité GET sur la racine de l'application qui renvoie un message de bienvenue. Pour permettre l'ajout d'éléments, j'ai créé un point d'extrémité POST /items qui vérifie si le corps de la requête contient un nouvel élément. Si l'élément est valide, il est ajouté au tableau items, qui est ensuite affiché dans la console. Enfin, le serveur écoute les requêtes sur le port 3000 et affiche un message confirmant son bon fonctionnement.

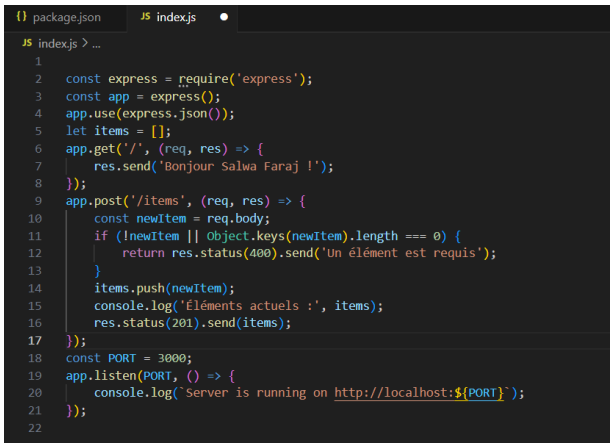


FIGURE 7 – Code

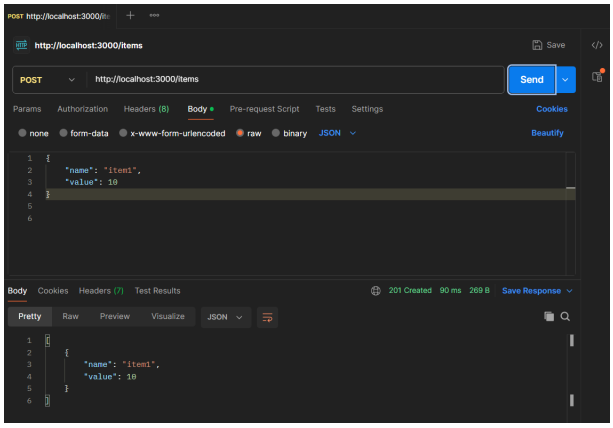


FIGURE 8 – Resultat

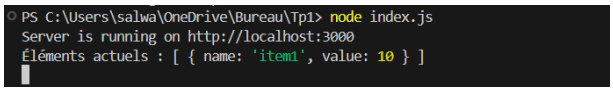


FIGURE 9 – Affichage dans le console

3.6 Question 6 :

Dans ce code, j'ai d'abord importé le module Express et créé une instance de l'application. Ensuite, j'ai utilisé un middleware pour analyser les corps des requêtes au format JSON. J'ai initialisé un tableau items pour stocker les éléments ajoutés. J'ai défini un point d'extrémité GET sur la racine de l'application qui renvoie un message de bienvenue. Pour permettre l'ajout d'éléments, j'ai créé un point d'extrémité POST /items qui vérifie si le corps de la requête contient un nouvel élément. Si l'élément est valide, il est ajouté au tableau items, qui est ensuite affiché dans la console.

De plus, j'ai ajouté un point d'extrémité GET /items qui permet de récupérer la liste complète des éléments ajoutés. Enfin, le serveur écoute les requêtes sur le port 3000 et affiche un message confirmant son bon fonctionnement.

FIGURE 10 – Code

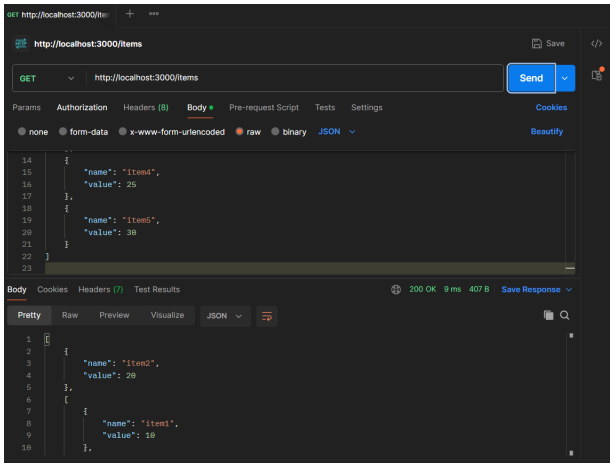


FIGURE 11 – Resultat

3.7 Question 7 :

Le point d'entrée GET par ID (/items/ :id) permet de récupérer un élément spécifique de la liste en fonction de son identifiant unique (ID).

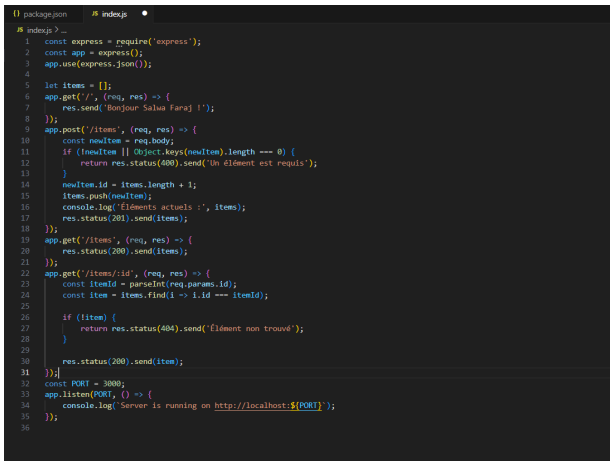


FIGURE 12 – Code

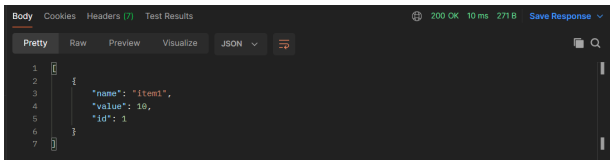


FIGURE 13 – Resultat

3.8 Question 8 :

Dans ce code, j'ai ajouté un point d'entrée PUT qui permet de mettre à jour un élément existant dans le tableau items en utilisant son id. Si l'élément est trouvé, il est mis à jour avec les nouvelles données envoyées dans la requête et renvoyé au client. Si l'élément n'existe pas, un message d'erreur "Item not found" est renvoyé avec un statut 404.

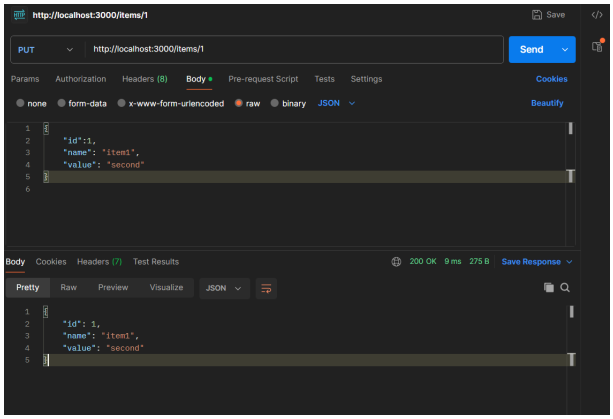


FIGURE 14 – Update

FIGURE 15 – Resultat Update

3.9 Question 9 :

Dans ce code, j'ai ajouté un point d'entrée DELETE qui permet de supprimer un élément du tableau items en fonction de son id. Si l'élément est trouvé, il est supprimé et renvoyé au client. Si l'élément n'est pas trouvé, un message d'erreur "Item not found" est renvoyé avec un statut 404.

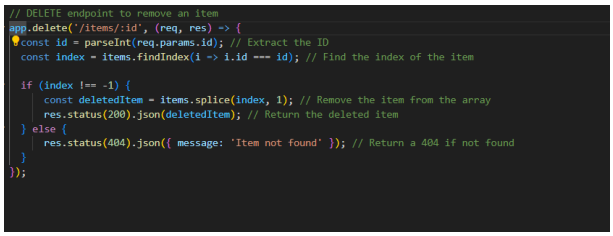


FIGURE 16 – Delete

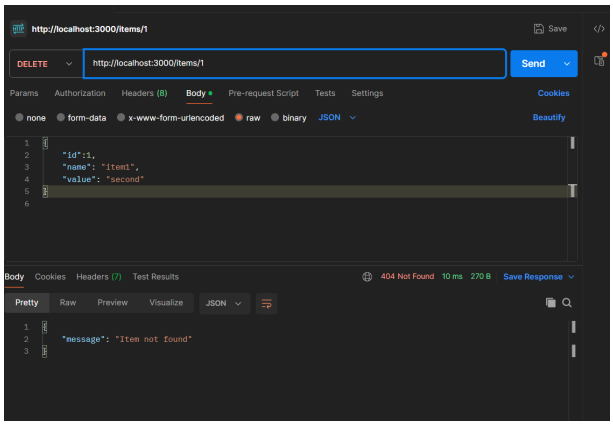


FIGURE 17 – Resultat Delete

4 Conclusion

En conclusion, ce TP a permis de créer une application CRUD simple en utilisant Express.js, où nous avons implémenté des points d'entrée pour créer, lire, mettre à jour et supprimer des éléments. Chaque fonctionnalité a été soigneusement conçue pour interagir avec un tableau d'objets en mémoire, offrant ainsi une expérience pratique sur la manière de gérer les opérations de base sur les données.