Université Sultan Moulay Slimane
Ecole Nationale des Sciences Appliquées
- ENSA Khouribga -

Rapport de TP

TP2: Express.js

Rédigé par:
Abdelkabir Elhamoussi

Encadré par:
Pr. Amal OURDOU

Année universitaire 2023-2024

1 Introduction

Express.js est un framework minimaliste pour Node.js qui facilite la création d'applications web et d'API. Il permet de gérer facilement les requêtes HTTP, les routes, et les middlewares. Express.js est utilisé pour créer des API RESTful, des applications CRUD, et des applications temps réel.

2 Qu'est-ce qu'Express.js et ses utilisations

Express.js est un framework léger qui repose sur Node.js. Voici quelques exemples d'utilisations possibles :

- Création d'API RESTful.
- Développement d'applications web dynamiques.
- Intégration de services tiers (authentification OAuth, etc.).

3 Middlewares dans Express.js

Les middlewares dans Express.js sont des fonctions qui interceptent les requêtes avant qu'elles n'atteignent les routes ou après la génération d'une réponse.

3.1 Exemple 1: Middleware pour parser les requêtes JSON

[language=JavaScript] app.use(express.json()); Ce middleware permet de parser le corps des requêtes au format JSON et de le rendre disponible dans `req.body`.

3.2 Exemple 2: Middleware personnalisé pour journaliser les requêtes

[language=JavaScript] app.use((req, res, next) =¿ console.log(‘*req.method*req.url‘); next(); // Passe au middleware suivant); Ce middleware affiche la méthode HTTP et l'URL de chaque requête dans la console, puis appelle `next()` pour continuer le traitement.

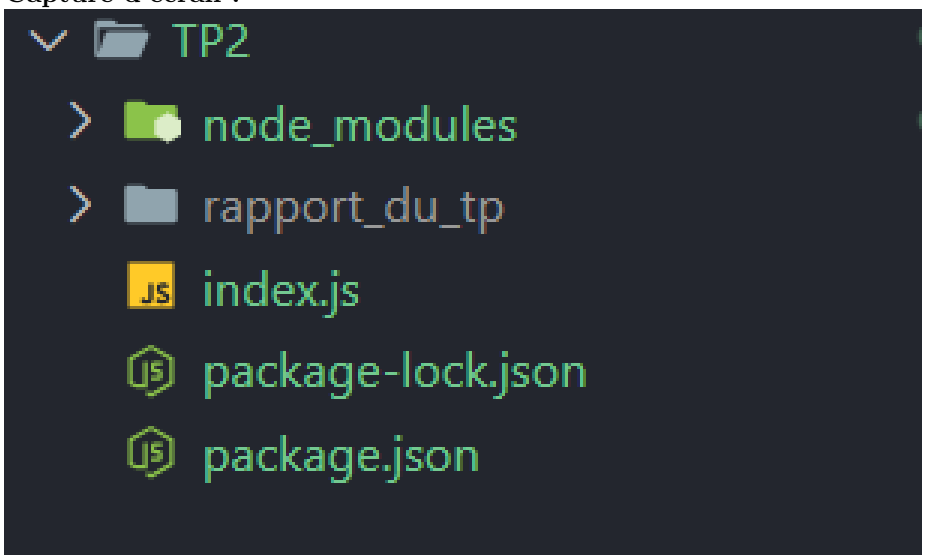
4 Création d'une application CRUD simple avec Express.js

Cette section présente les étapes pour développer une application CRUD simple avec Express.js.

4.1 Étape 1: Créer un répertoire de projet

```
mkdir express-crud
cd express-crud
```

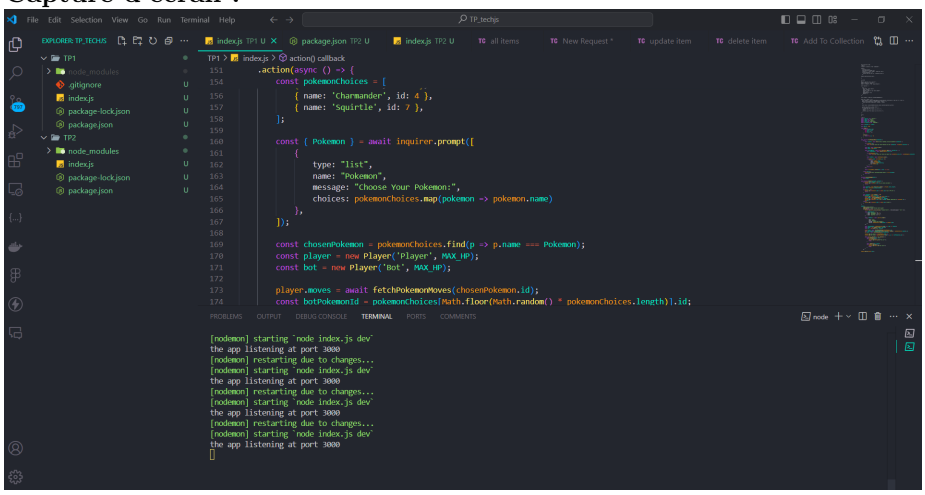
Capture d'écran :



4.2 Étape 2: Initialiser un projet Node.js

```
npm init -y
```

Capture d'écran :



4.3 Étape 3: Installer Express

```
npm install express
```

Capture d'écran :

```
PS C:\Users\info\Desktop\TP_techjs\TP2> npm i nodemon express

added 94 packages, and audited 95 packages in 7s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

4.4 Étape 4: Configurer Express et démarrer le serveur

[language=JavaScript] const express = require("express"); const app = express(); const port = 3000;

app.listen(port, () =¿ console.log(‘the app listening at port *port*‘);); **Capture d'écran :**

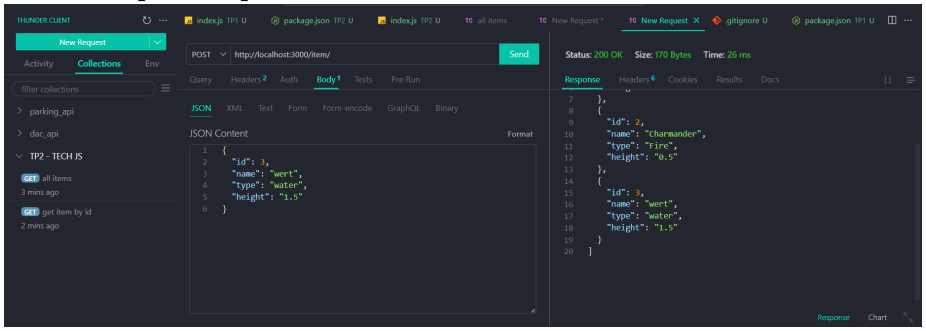
```
found 0 vulnerabilities
PS C:\Users\info\Desktop\TP_techjs\TP2> npm start dev

> tp2@1.0.0 start
> nodemon index.js dev

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js dev`
the app listening at port 3000
[nodemon] restarting due to changes...
[nodemon] starting `node index.js dev`
the app listening at port 3000
```

4.5 Étape 5: Créer un point d'entrée POST pour ajouter des éléments

[language=JavaScript] app.post("/item", (req, res) =¿ const item = "id": req.body.id, "name": req.body.name, "type": req.body.type, "height": req.body.height, ; list.push(item); res.send(list);); **Capture d'écran :**

```
app.post("/item", (req, res) => {
    const item = {
        "id" : req.body.id,
        "name" : req.body.name,
        "type" : req.body.type,
        "height" : req.body.height,
    }
    list.push(item);
    res.send(list);
});
```

test de l'api avec postman :

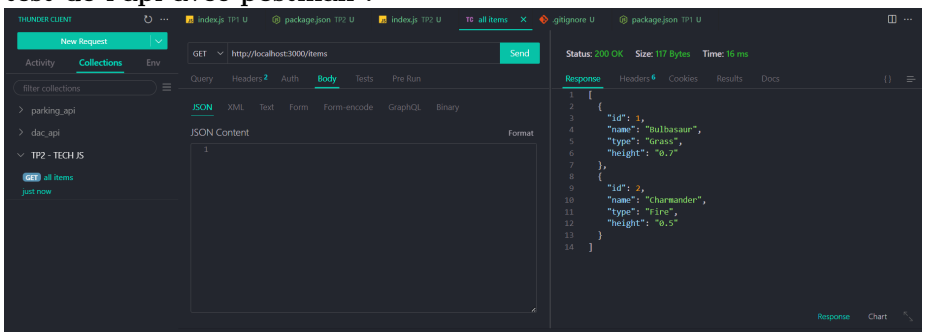


4.6 Étape 6: Créer un point d'entrée GET pour récupérer tous les éléments

[language=JavaScript] app.get("/items", (req, res) =¿ res.send(list);); **Capture d'écran :**

```
app.get("/items", (req, res) => {
    res.send(list);
});
```

test de l'api avec postman :

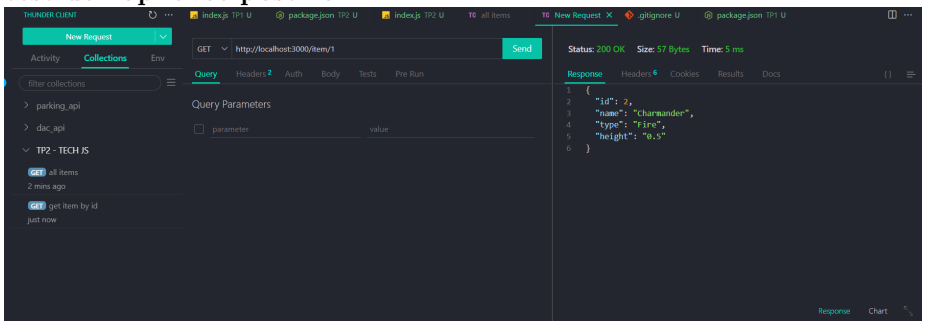


4.7 Étape 7: Créer un point d'entrée GET pour récupérer un élément spécifique par ID

[language=JavaScript] app.get("/item/:id", (req, res) =¿ const id = req.params.id; res.send(list[id]);); **Capture d'écran :**

```
app.get("/item/:id", (req, res) => {
    const id = req.params.id;
    res.send(list[id]);
});
```

test de l'api avec postman :

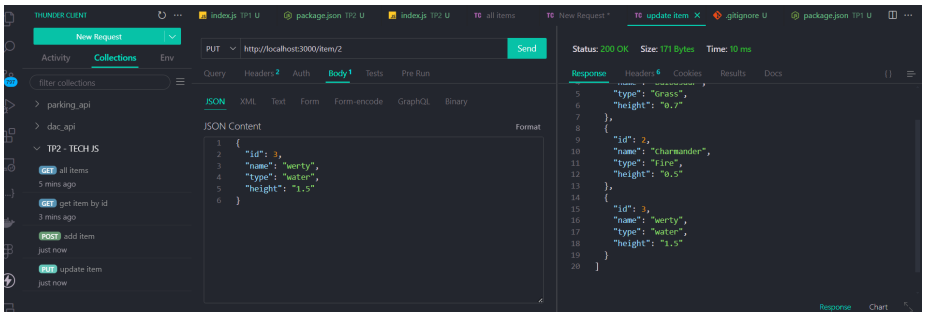


4.8 Étape 8: Créer un point d'entrée PUT pour mettre à jour un élément existant

[language=JavaScript] app.put("/item/:id", (req, res) =¿ const id = req.params.id; list[id] = req.body; res.send(list);); **Capture d'écran :**

```
app.put("/item/:id", (req, res) => {
    const id = req.params.id;
    list[id] = req.body;
    res.send(list);
});
```

test de l'api avec postman :



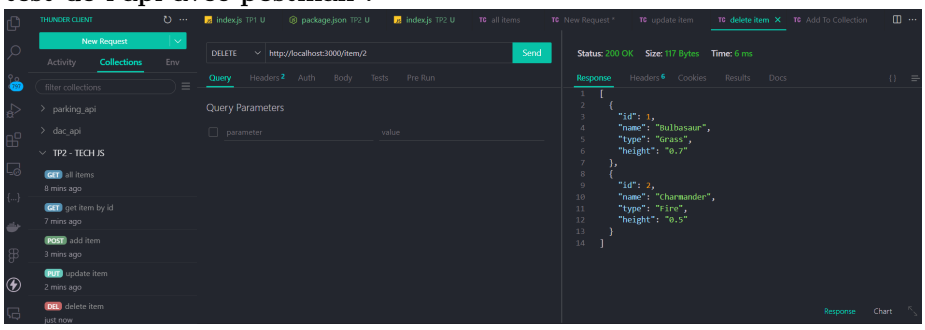
4.9 Étape 9: Créer un point d'entrée DELETE pour supprimer un élément

[language=JavaScript] app.delete("/item/:id", (req, res) =¿ const id = req.params.id; if (!list[id]) res.send("Item not found"); else list.splice(id, 1); res.send(list););

Capture d'écran :

```
app.delete("/item/:id", (req, res) => {
    const id = req.params.id;
    if(!list[id]) {
        res.send("Item not found");
    }
    list.splice(id, 1);
    res.send(list);
});
```

test de l'api avec postman :

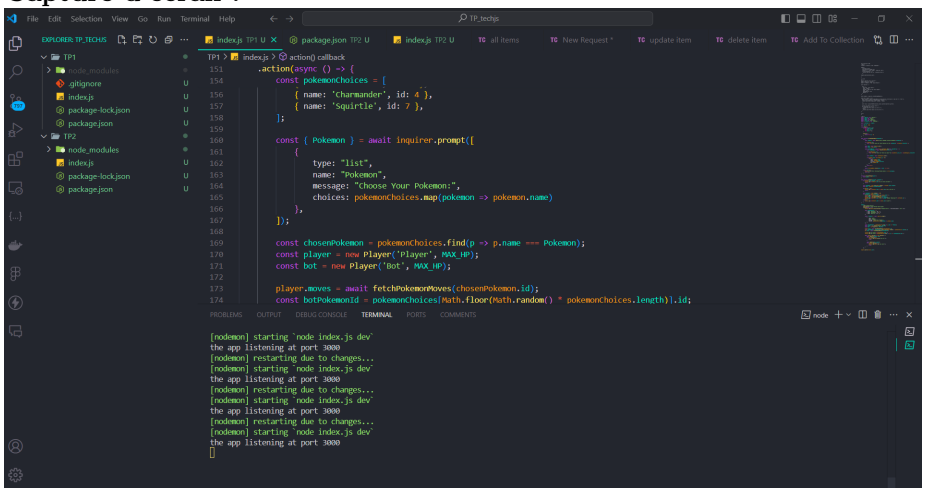


4.10 Étape 10: Démarrer le serveur

Utilisez la commande suivante pour démarrer le serveur dans le terminal :

`node index.js`

Capture d'écran :

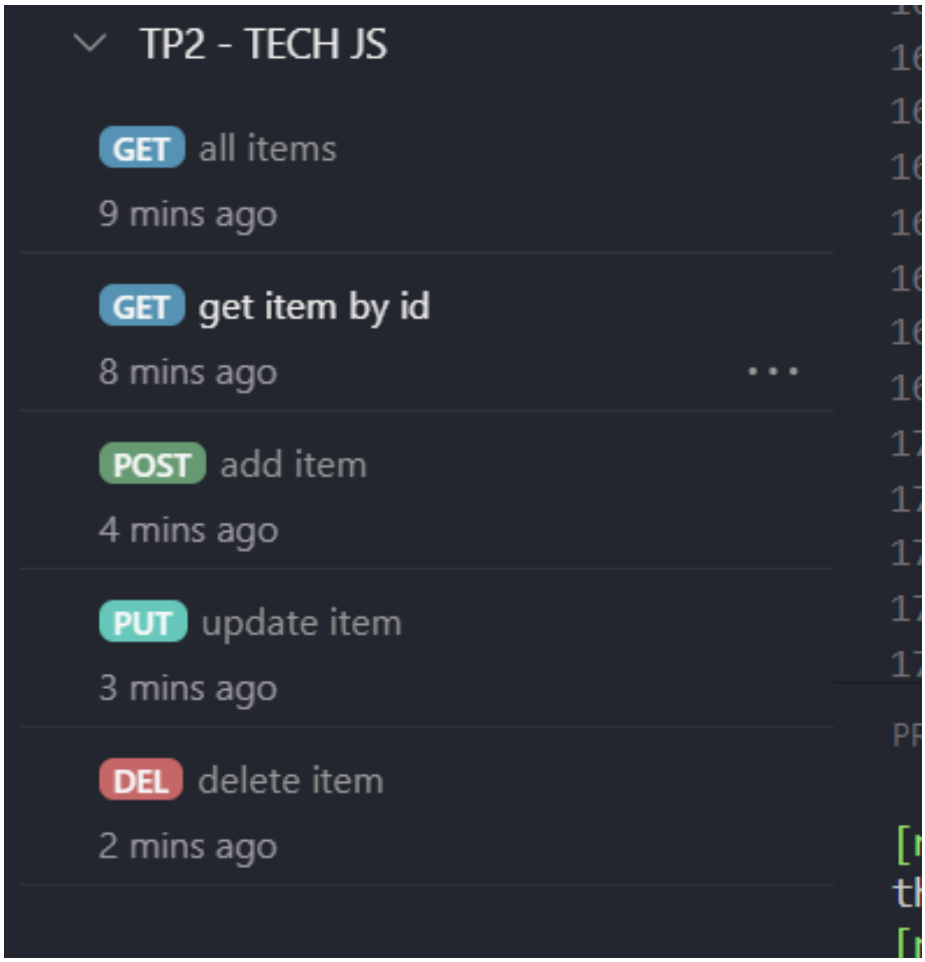


4.11 Étape 11: Tester les points d'entrée avec Postman

Utilisez Postman pour tester les points d'entrée :

- POST `/item` : Ajouter un élément.
- GET `/items` : Récupérer tous les éléments.
- GET `/item/:id` : Récupérer un élément spécifique.
- PUT `/item/:id` : Mettre à jour un élément.
- DELETE `/item/:id` : Supprimer un élément.

Capture d'écran :



5 Conclusion

Dans ce TP, nous avons appris à utiliser Express.js pour créer une application CRUD simple. Express.js est un framework puissant et flexible qui facilite le développement rapide d'API et d'applications web. Grâce aux points d'entrée CRUD que nous avons mis en place, il est possible de gérer efficacement des ressources dans une application web.