

# Rapport de TP 2 (Technologies JS)

## TP (Express.JS)

**Encadré par :**  
**Pr Ourdou**

**Présenté par:**  
doaa BISSASSE

# Partie 1 : Introduction à Express.js

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

Express.js est un framework web minimaliste et flexible pour Node.js, qui fournit un ensemble robuste de fonctionnalités pour créer des applications web et mobiles. Il simplifie la création de serveurs web et d'API en offrant divers outils et utilitaires pour gérer les requêtes et réponses HTTP. Avec Express, on peut créer :

- Des API RESTful
- Des applications web (par exemple des applications single-page ou multi-page)
- Des services de middleware
- Des applications web en temps réel combinées avec WebSockets (par exemple, des applications de chat)

## 2. Qu'est-ce que les middlewares et comment sont-ils utilisés dans Express.js ?

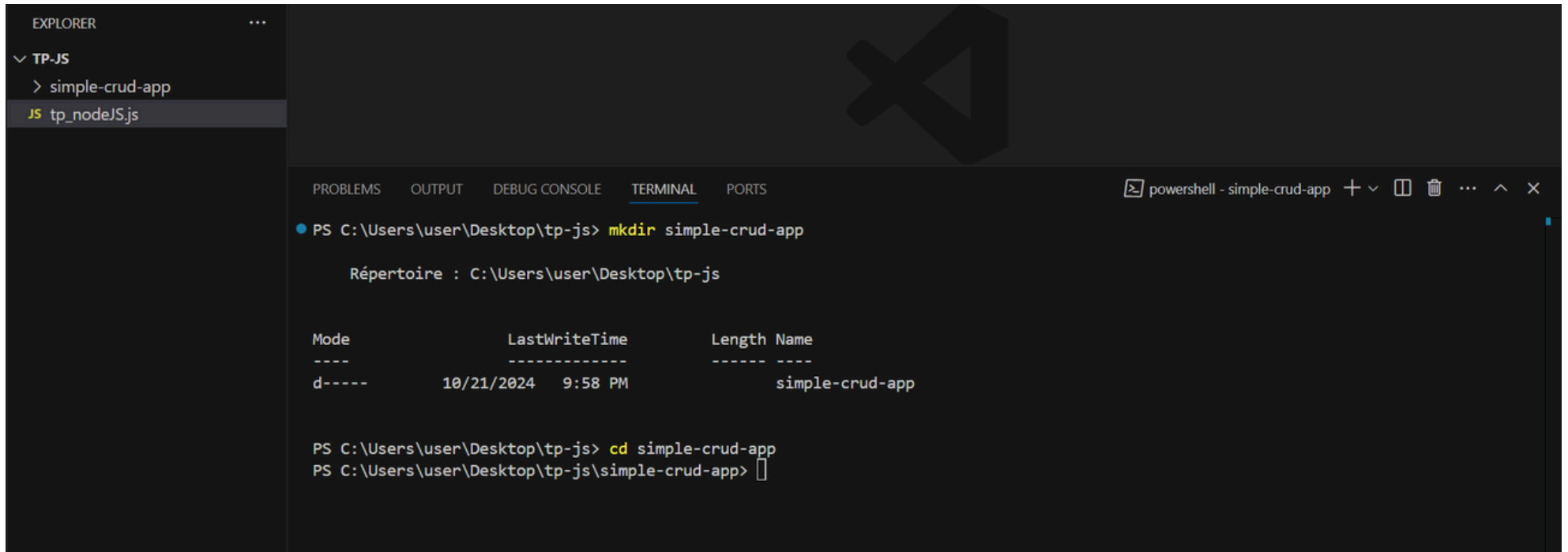
Les middlewares sont des fonctions qui ont accès aux objets req (requête) et res (réponse), ainsi qu'à la fonction next() dans le cycle de requête-réponse de l'application. Ils peuvent effectuer diverses tâches comme la journalisation, l'authentification, la gestion des erreurs ou la modification des objets req et res avant d'envoyer une réponse.

parmi les middlewares :

- **Middleware de journalisation** : Ce middleware enregistre les détails de chaque requête effectuée sur le serveur.
- **Middleware de gestion des erreurs** : Ce middleware intercepte les erreurs survenant dans l'application.

# Partie 2 : Création d'une application CRUD simple

## Étape 1 : Créer un répertoire de projet



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'TP-JS' with a subdirectory 'simple-crud-app' and a file 'tp\_nodeJS.js'. The main editor area shows a PowerShell terminal window with the following commands and output:

```
PS C:\Users\user\Desktop\tp-js> mkdir simple-crud-app

Répertoire : C:\Users\user\Desktop\tp-js

Mode                LastWriteTime         Length Name
----                -
d-----         10/21/2024   9:58 PM             simple-crud-app

PS C:\Users\user\Desktop\tp-js> cd simple-crud-app
PS C:\Users\user\Desktop\tp-js\simple-crud-app>
```

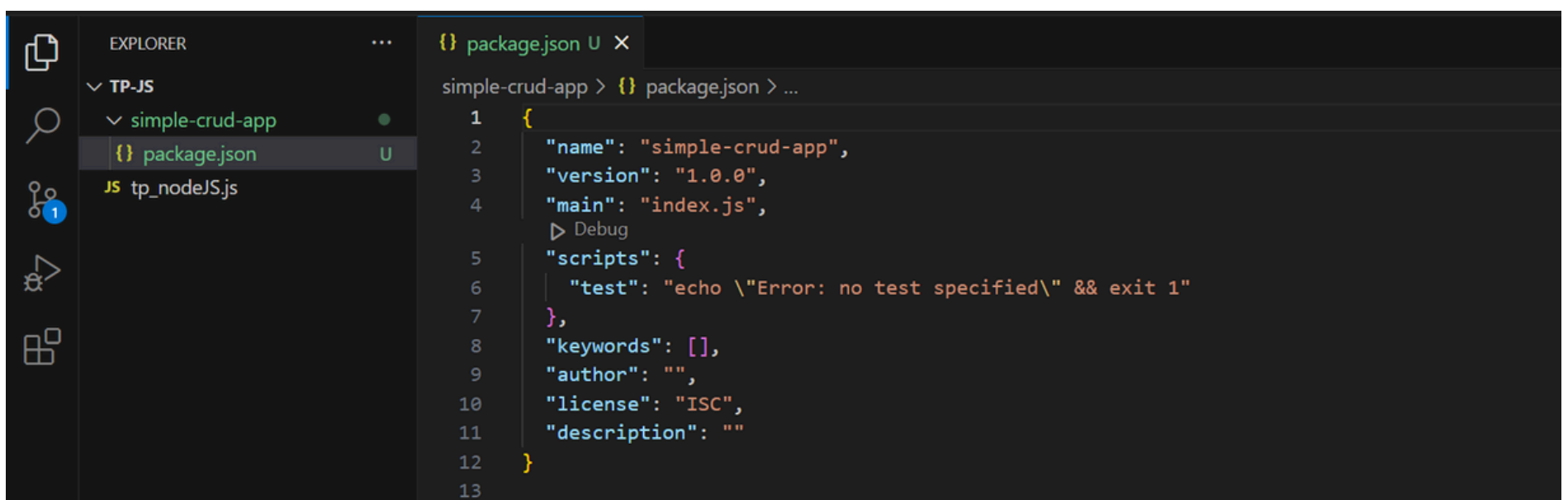
## Étape 2 : Initialiser un projet Node.js



The screenshot shows a PowerShell terminal window with the following output:

```
PS C:\Users\user\Desktop\tp-js\simple-crud-app> npm init -y
Wrote to C:\Users\user\Desktop\tp-js\simple-crud-app\package.json:

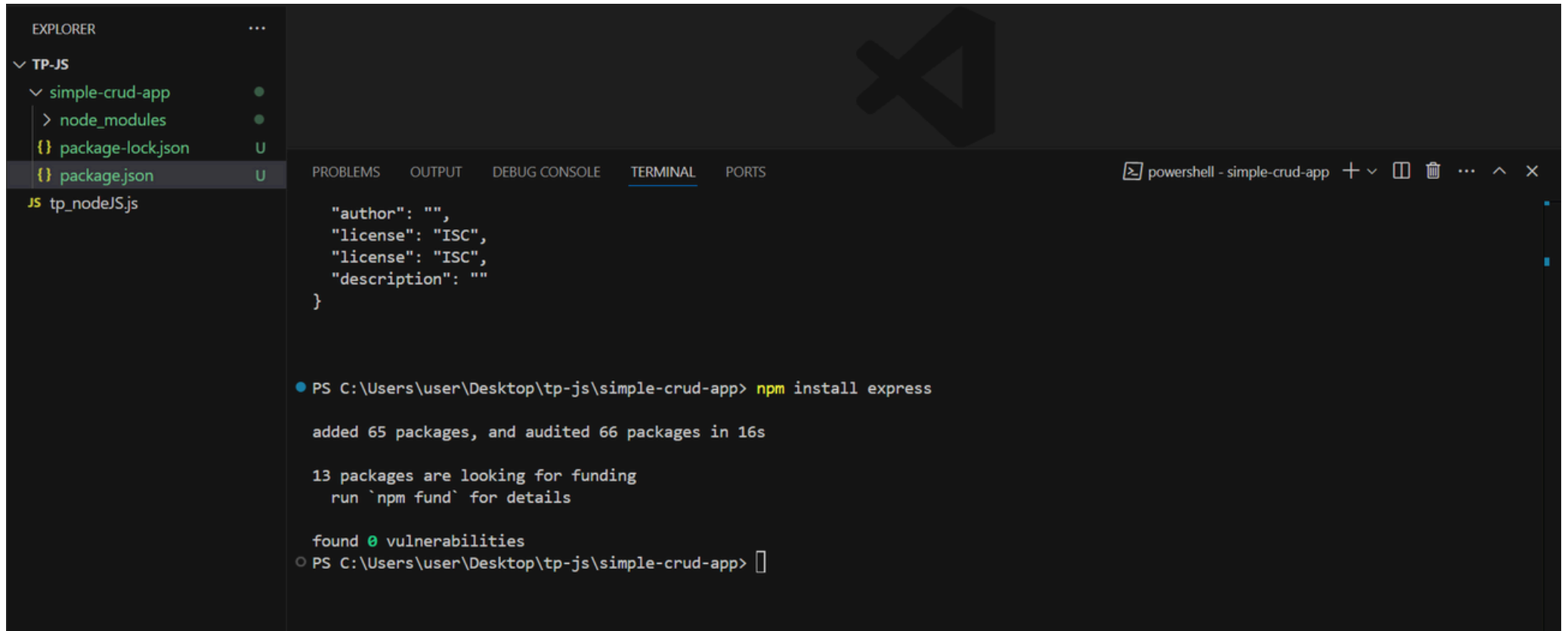
{
  "name": "simple-crud-app",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```



The screenshot shows the Visual Studio Code interface with the 'package.json' file open in the editor. The Explorer sidebar on the left shows the project structure with 'package.json' selected. The editor displays the following content:

```
1 {
2   "name": "simple-crud-app",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "test": "echo \"Error: no test specified\" && exit 1"
7   },
8   "keywords": [],
9   "author": "",
10  "license": "ISC",
11  "description": ""
12 }
13
```

## Étape 3 : Installer Express



```
    "author": "",
    "license": "ISC",
    "license": "ISC",
    "description": ""
  }

PS C:\Users\user\Desktop\tp-js\simple-crud-app> npm install express

added 65 packages, and audited 66 packages in 16s

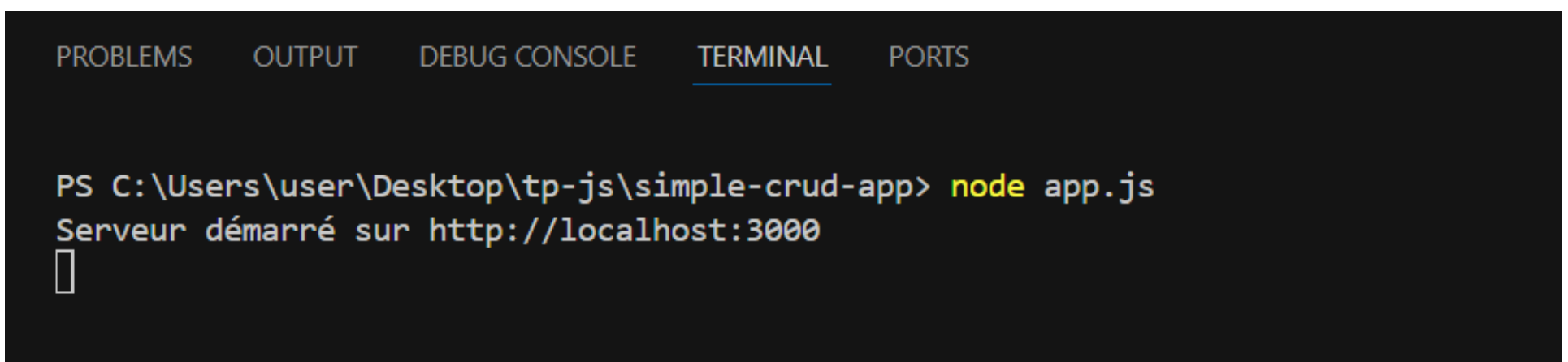
13 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\user\Desktop\tp-js\simple-crud-app>
```

## Étape 4 : Configurer Express et démarrer le serveur



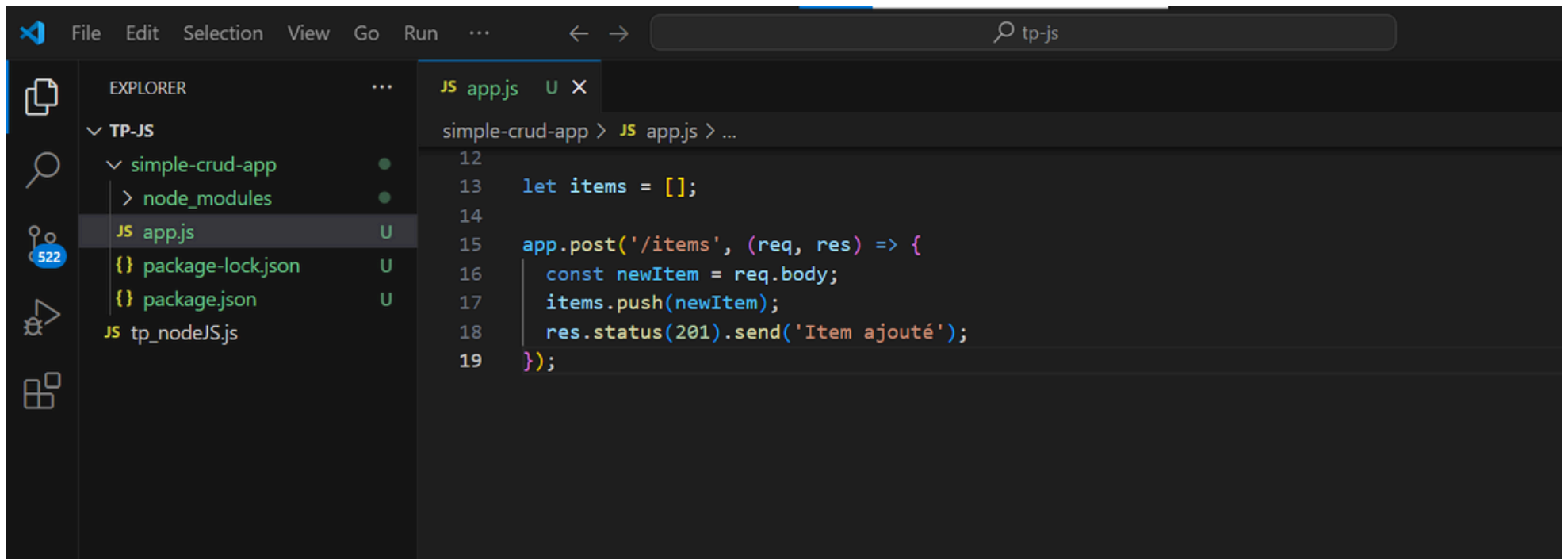
```
1  const express = require('express');
2  const app = express();
3  const PORT = 3000;
4
5  // Pour analyser les données JSON dans les corps de requêtes
6  app.use(express.json());
7
8  // Démarrer le serveur
9  app.listen(PORT, () => {
10    console.log(`Serveur démarré sur http://localhost:${PORT}`);
11  });
12
```



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\user\Desktop\tp-js\simple-crud-app> node app.js
Serveur démarré sur http://localhost:3000
```

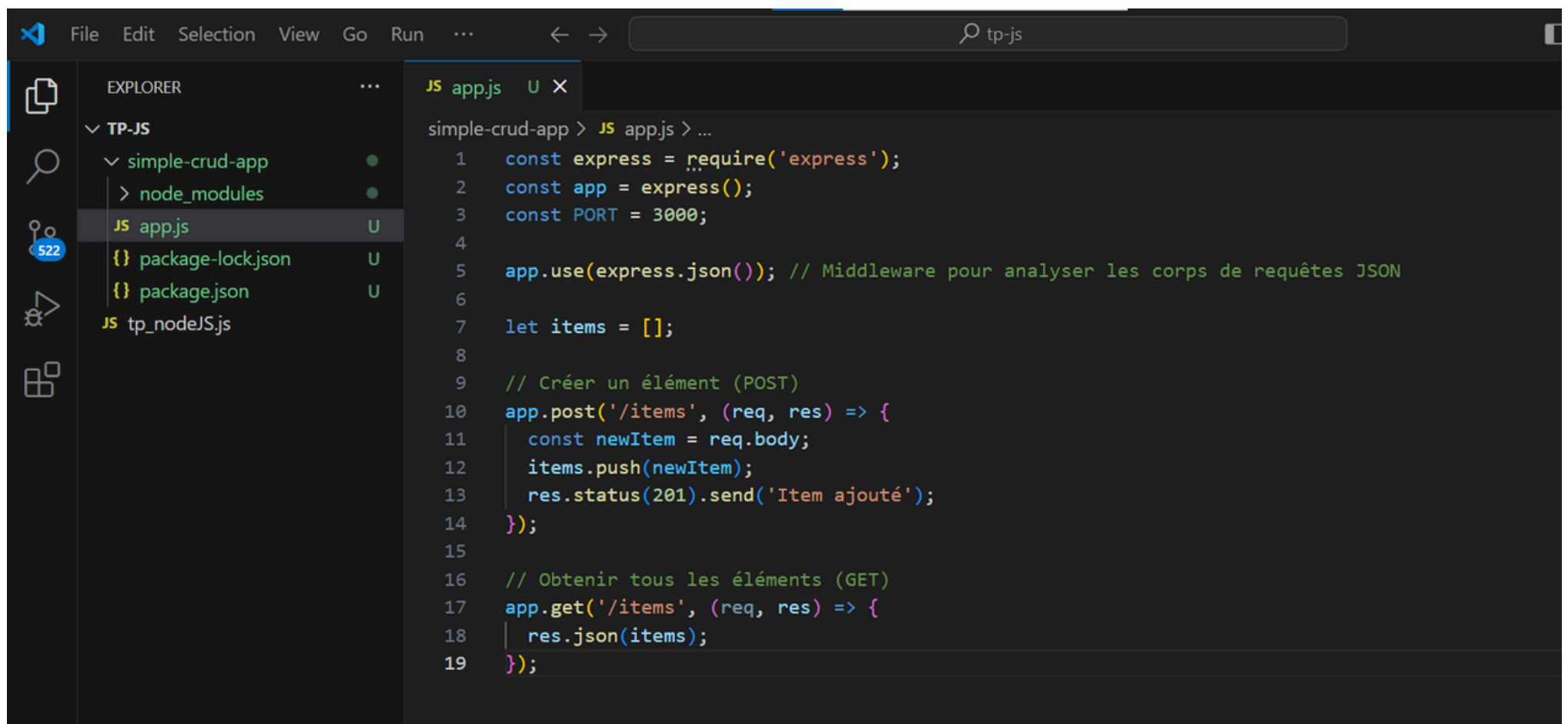
## Étape 5 : Créer un point de terminaison POST (Ajouter un élément)



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The Explorer shows a project named 'TP-JS' containing a folder 'simple-crud-app' and files 'app.js', 'package-lock.json', 'package.json', and 'tp\_nodeJS.js'. The 'app.js' file is selected and open in the editor. The code in the editor shows the implementation of a POST endpoint for adding a new item to a list.

```
12
13 let items = [];
14
15 app.post('/items', (req, res) => {
16   const newItem = req.body;
17   items.push(newItem);
18   res.status(201).send('Item ajouté');
19 });
```

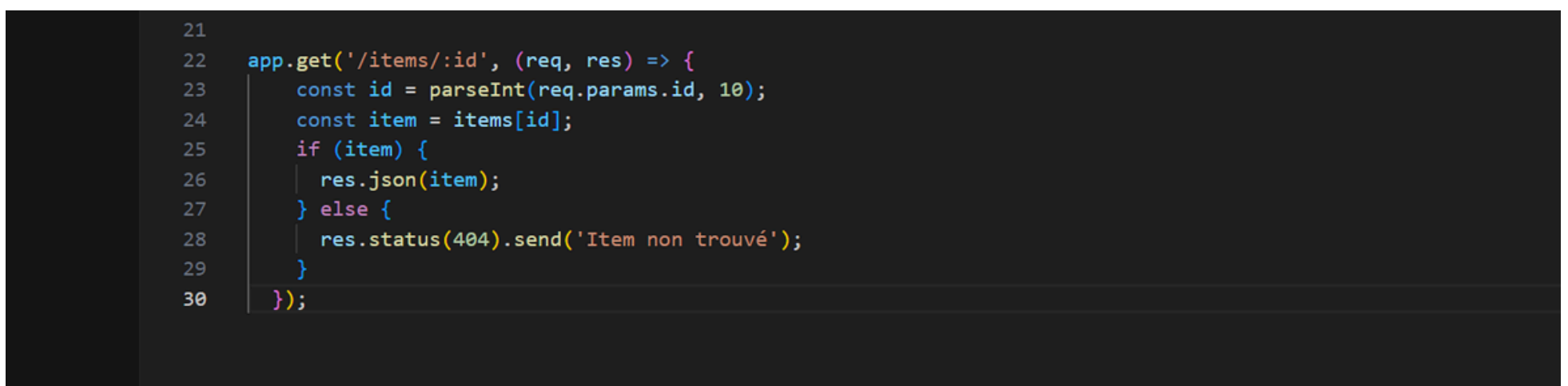
## Étape 6 : Créer un point de terminaison GET (Récupérer tous les éléments)



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The Explorer shows a project named 'TP-JS' containing a folder 'simple-crud-app' and files 'app.js', 'package-lock.json', 'package.json', and 'tp\_nodeJS.js'. The 'app.js' file is selected and open in the editor. The code in the editor shows the implementation of a GET endpoint to retrieve all items from a list.

```
1 const express = require('express');
2 const app = express();
3 const PORT = 3000;
4
5 app.use(express.json()); // Middleware pour analyser les corps de requêtes JSON
6
7 let items = [];
8
9 // Créer un élément (POST)
10 app.post('/items', (req, res) => {
11   const newItem = req.body;
12   items.push(newItem);
13   res.status(201).send('Item ajouté');
14 });
15
16 // Obtenir tous les éléments (GET)
17 app.get('/items', (req, res) => {
18   res.json(items);
19 });
```

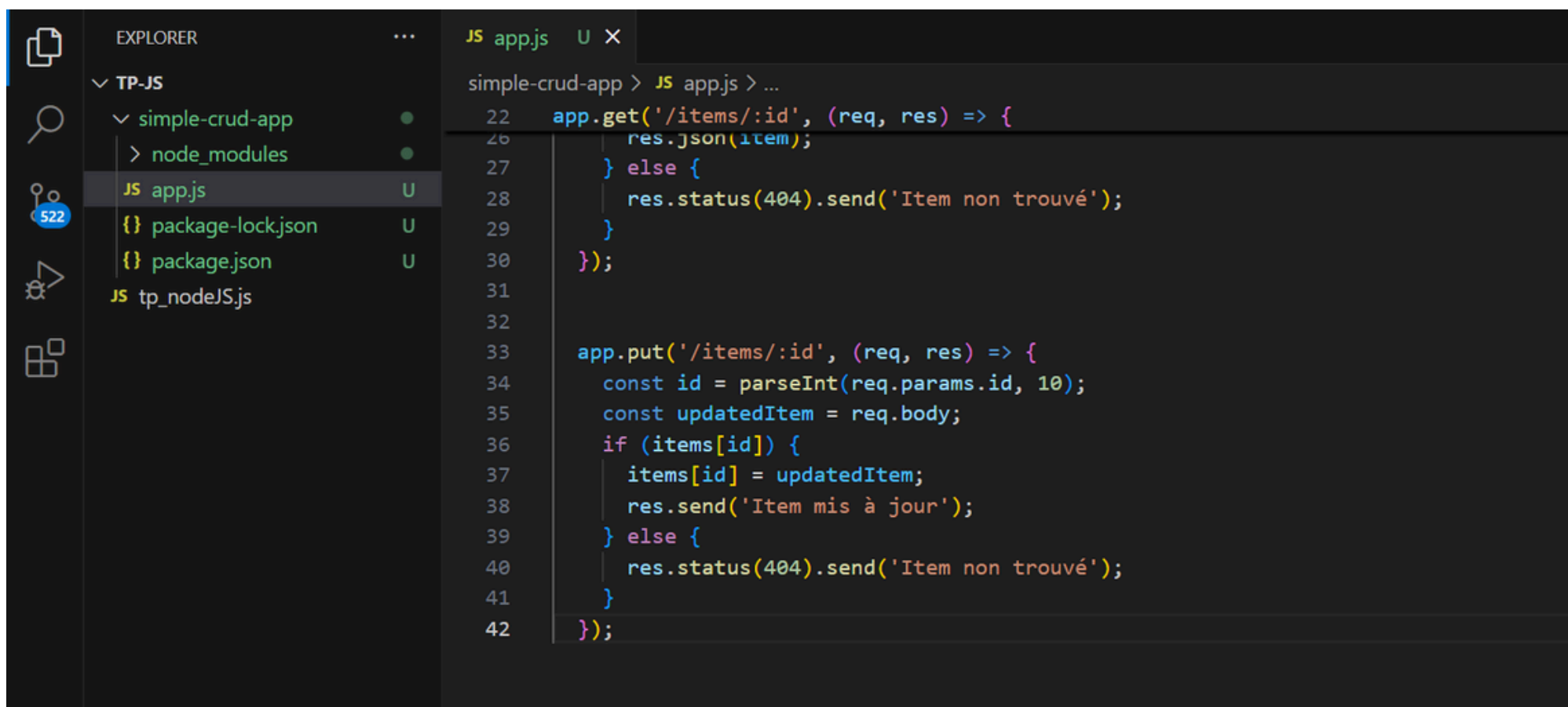
## Étape 7 : Créer un point de terminaison GET par ID (Récupérer un élément spécifique)



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The Explorer shows a project named 'TP-JS' containing a folder 'simple-crud-app' and files 'app.js', 'package-lock.json', 'package.json', and 'tp\_nodeJS.js'. The 'app.js' file is selected and open in the editor. The code in the editor shows the implementation of a GET endpoint to retrieve a specific item from a list by its ID.

```
21
22 app.get('/items/:id', (req, res) => {
23   const id = parseInt(req.params.id, 10);
24   const item = items[id];
25   if (item) {
26     res.json(item);
27   } else {
28     res.status(404).send('Item non trouvé');
29   }
30 });
```

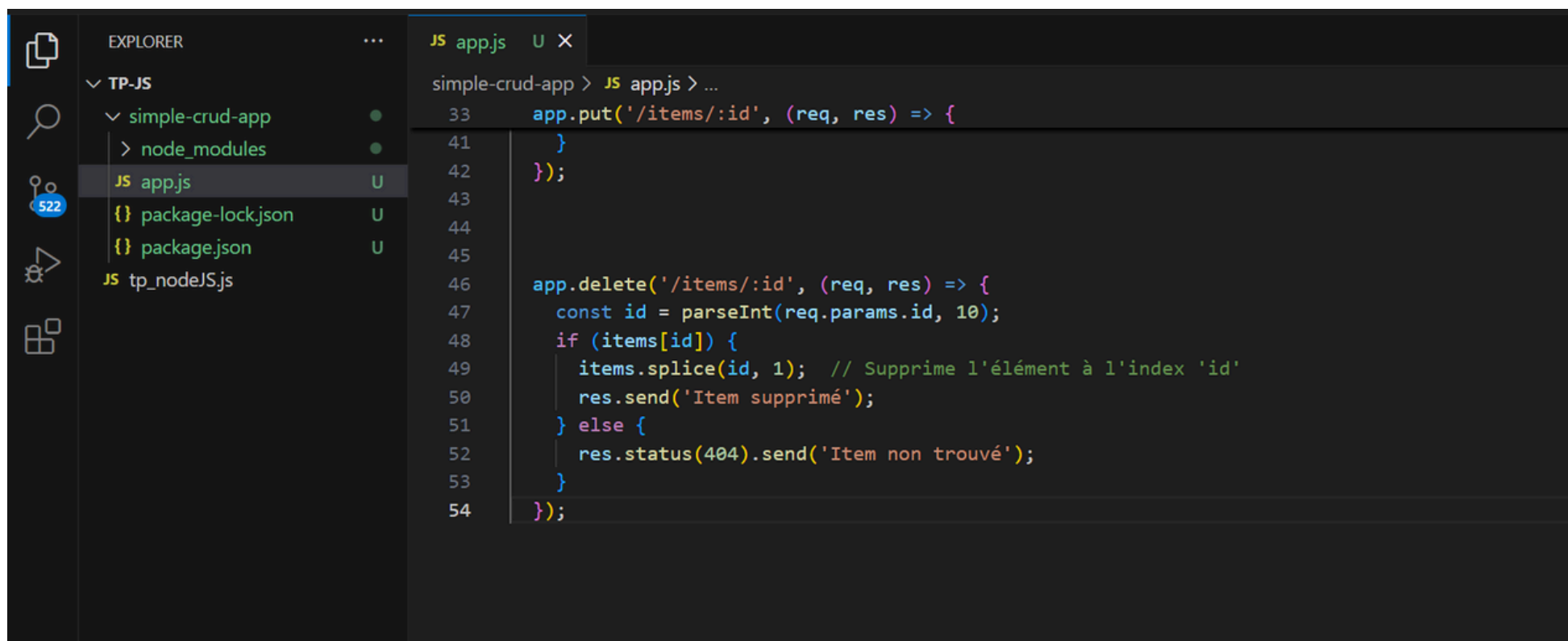
## Étape 8 : Créer un point de terminaison PUT (Mettre à jour un élément)



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of a project named 'TP-JS'. The file 'app.js' is selected. The main editor window shows the code for the PUT endpoint at line 33. The code implements a function to update an item in a collection. It uses `parseInt` to convert the item ID to a number. If the item exists in the `items` array, it is updated with the new body from the request. If not, a 404 status is returned with the message 'Item non trouvé'.

```
simple-crud-app > JS app.js > ...
22  app.get('/items/:id', (req, res) => {
26      res.json(item);
27  } else {
28      res.status(404).send('Item non trouvé');
29  }
30  });
31
32
33  app.put('/items/:id', (req, res) => {
34      const id = parseInt(req.params.id, 10);
35      const updatedItem = req.body;
36      if (items[id]) {
37          items[id] = updatedItem;
38          res.send('Item mis à jour');
39      } else {
40          res.status(404).send('Item non trouvé');
41      }
42  });
```

## Étape 9 : Créer un point de terminaison DELETE (Supprimer un élément)

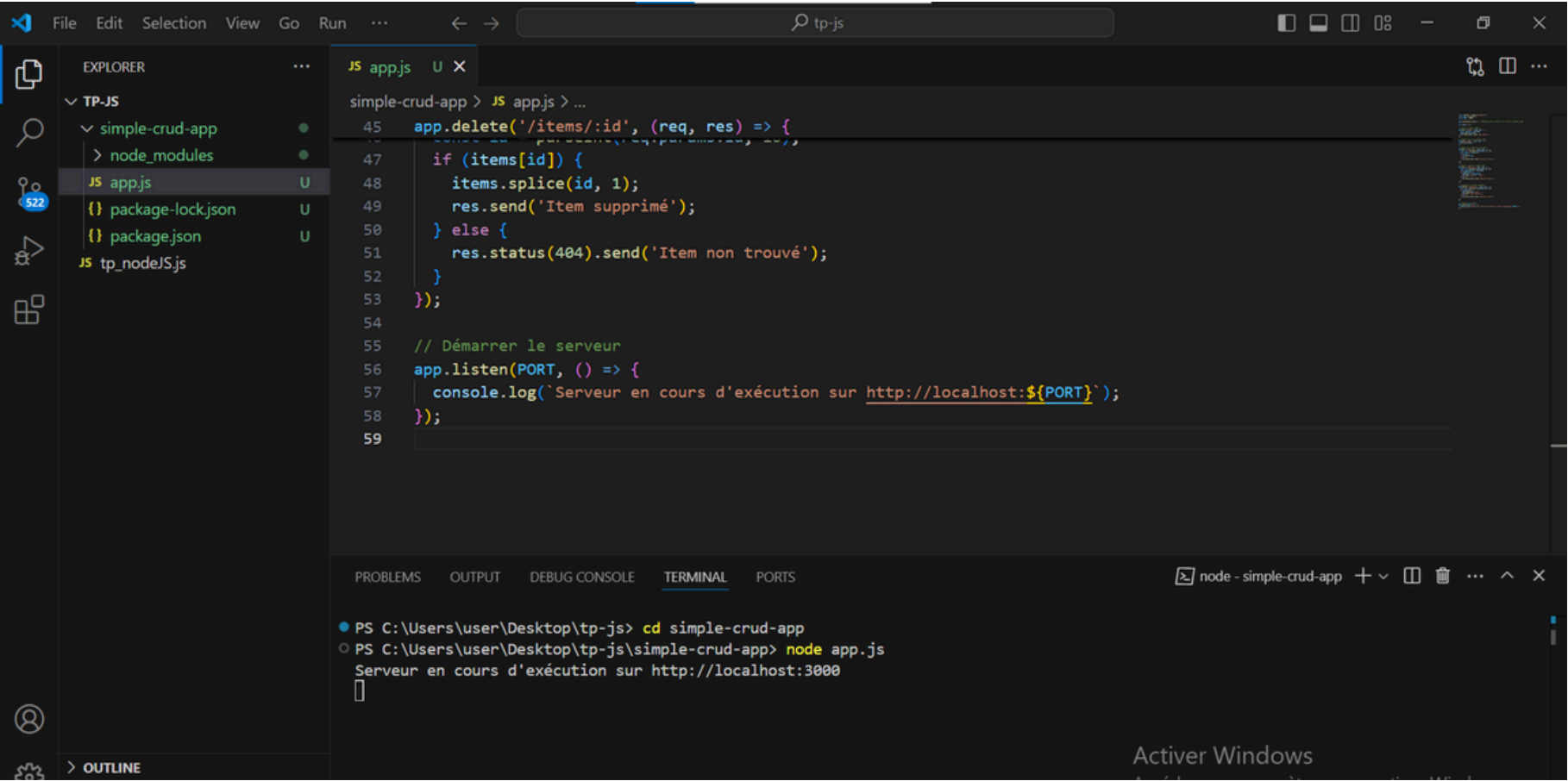


The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure. The file 'app.js' is selected. The main editor window shows the code for the DELETE endpoint at line 46. The code implements a function to delete an item from a collection. It uses `parseInt` to convert the item ID to a number. If the item exists in the `items` array, it is removed using `splice`. If not, a 404 status is returned with the message 'Item non trouvé'.

```
simple-crud-app > JS app.js > ...
33  app.put('/items/:id', (req, res) => {
41      }
42  });
43
44
45
46  app.delete('/items/:id', (req, res) => {
47      const id = parseInt(req.params.id, 10);
48      if (items[id]) {
49          items.splice(id, 1); // Supprime l'élément à l'index 'id'
50          res.send('Item supprimé');
51      } else {
52          res.status(404).send('Item non trouvé');
53      }
54  });
```

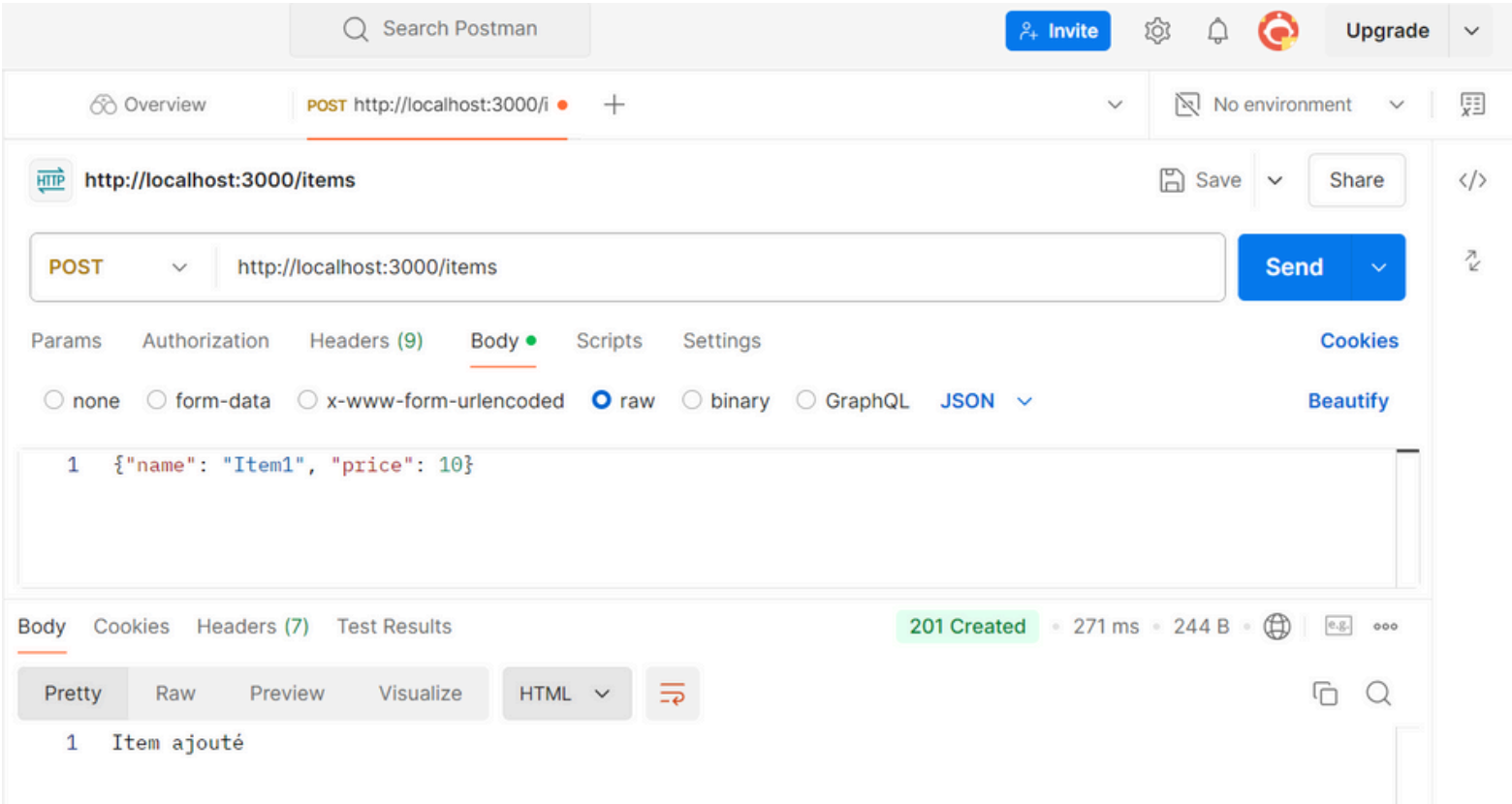


# Étape 10 : Démarrer le serveur

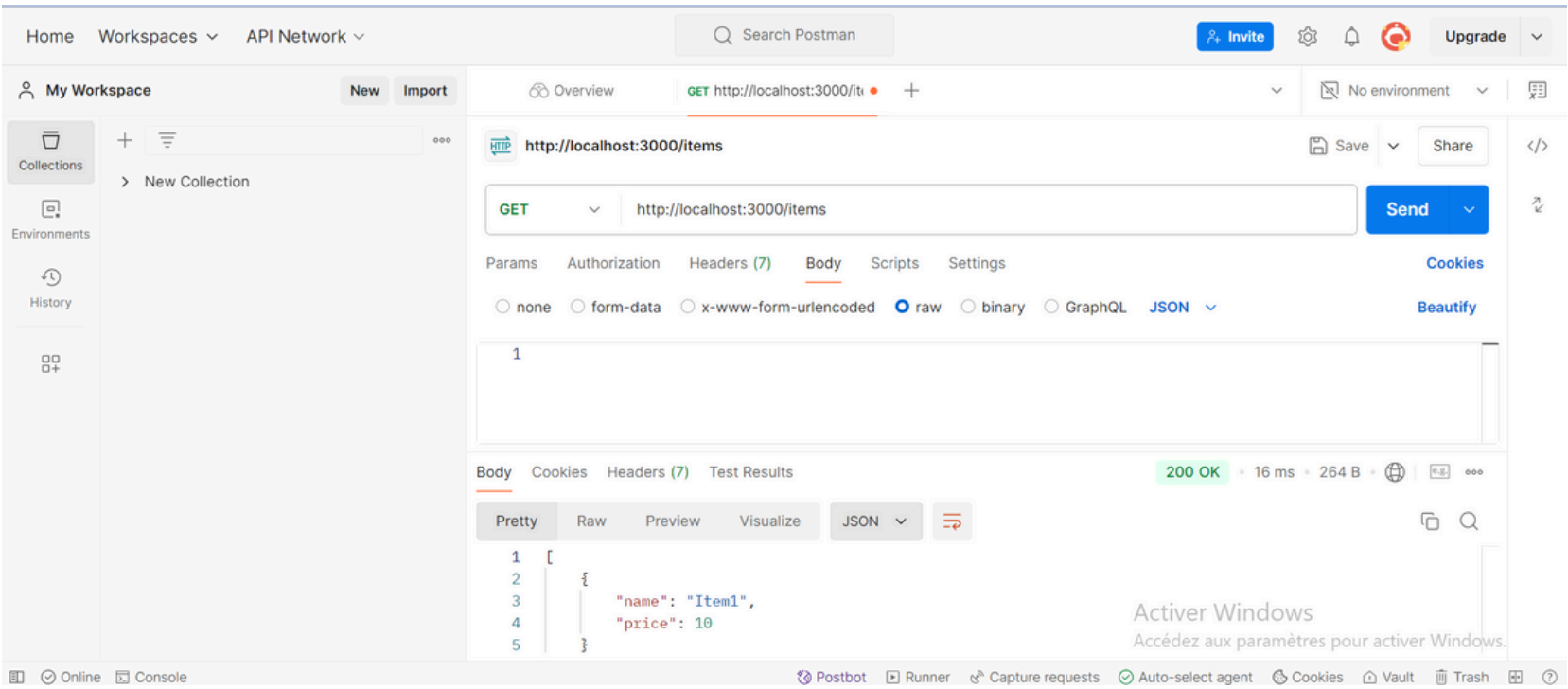


# Étape 11 : Tester les points de terminaison avec Postman

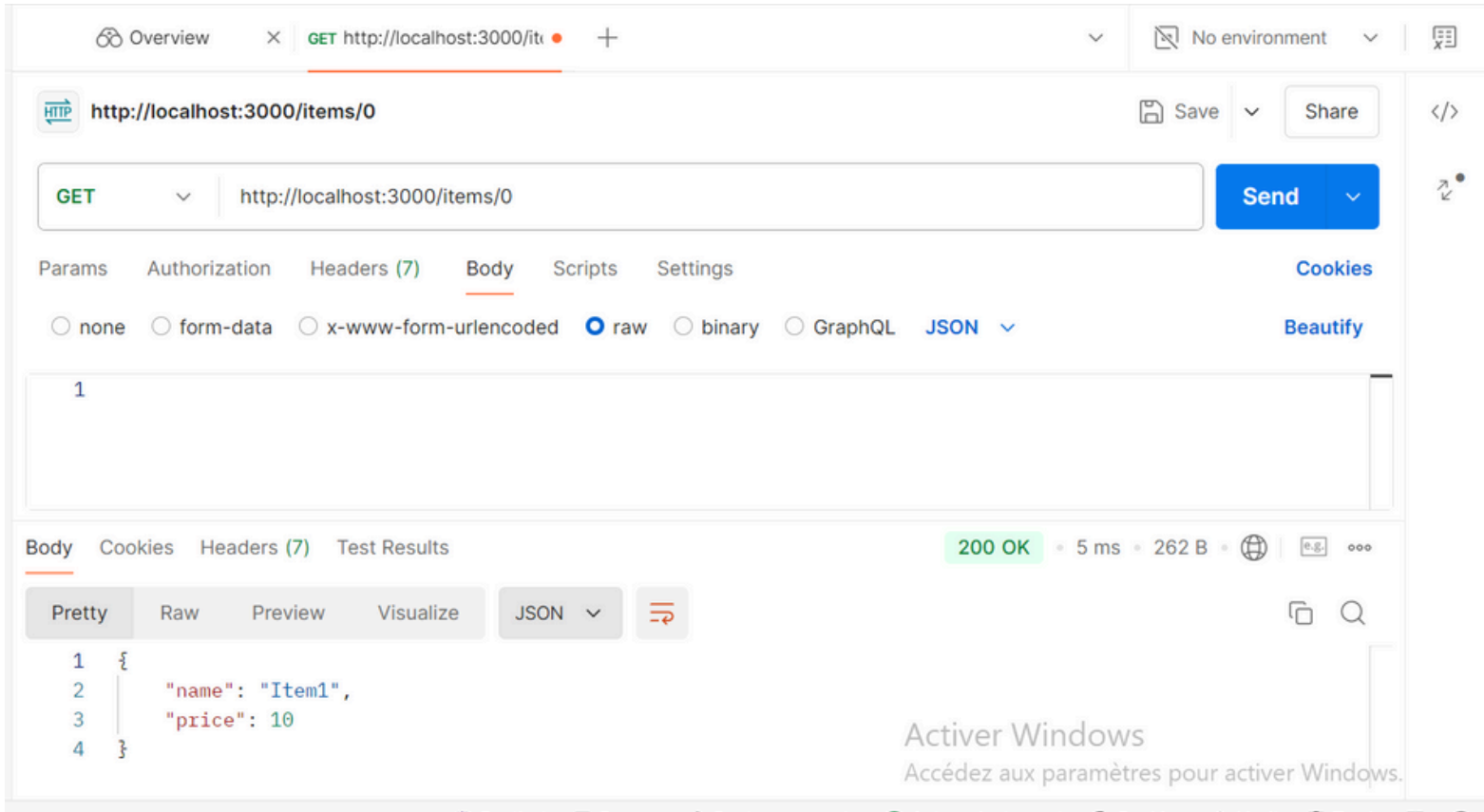
POST /items



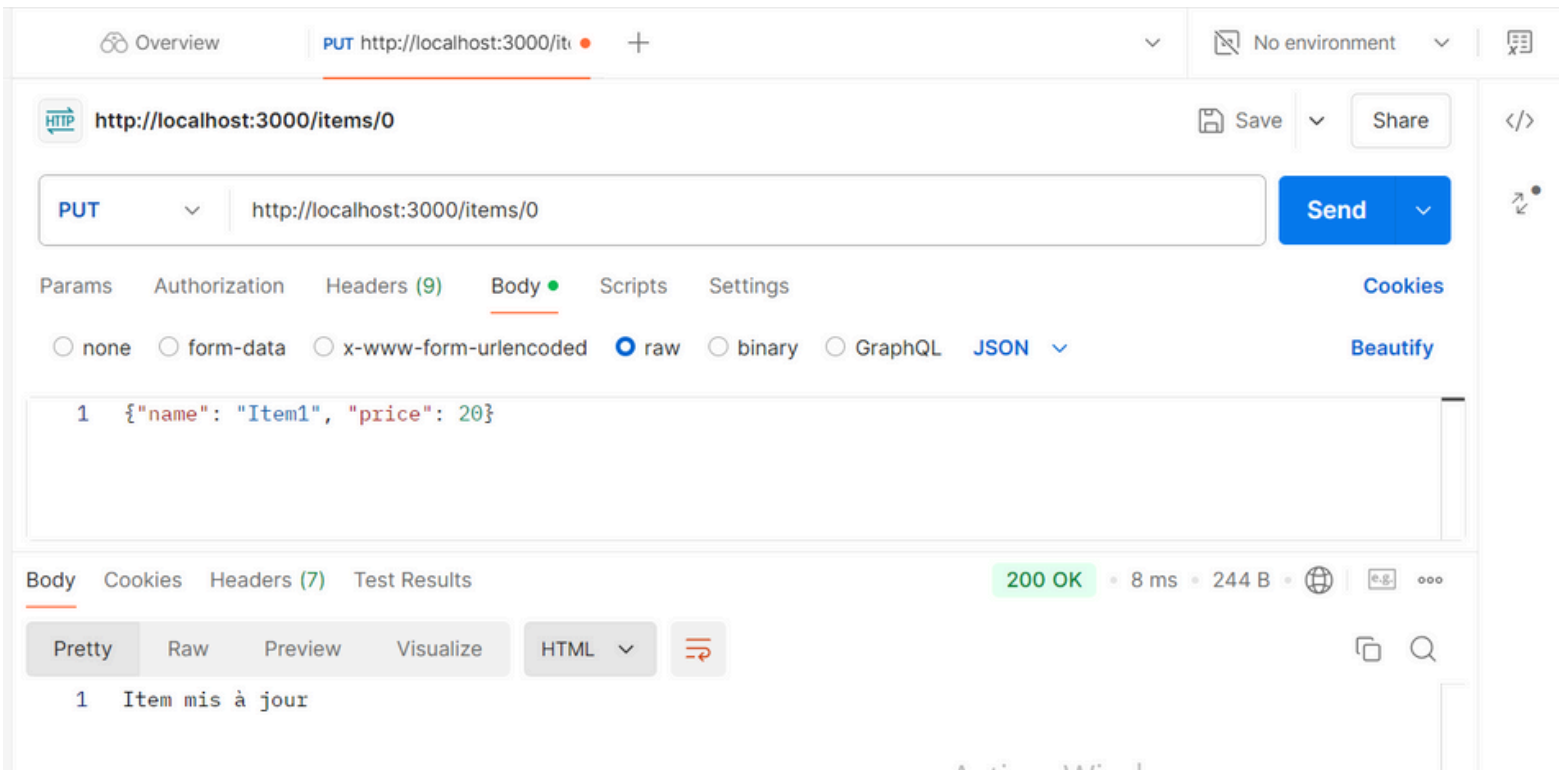
GET /items : Récupérer tous les éléments.



**GET /items/ : Récupérer un élément par son index.**



**PUT /items/ : Mettre à jour un élément existant (envoyez des données JSON).**



**DELETE /items/ : Supprimer un élément par son index.**

