TP (Express.JS)

1. What is Express.js and What Can We Make with It?

Express.js is a minimalist and flexible web application framework for **Node.js**. It provides a robust set of features for building web and mobile applications, allowing developers to create both APIs and full-stack applications. Express simplifies tasks such as routing, handling HTTP requests and responses, managing middleware, and more.

You can use **Express.js** to:

- Create **RESTful APIs**
- Build **web applications**
- Serve **static files** (like HTML, CSS, JS)
- Manage real-time data with frameworks like **Socket.io**
- Build **single-page applications** (with tools like React or Angular)

2. What are Middlewares and How Are They Used in Express.js?

Middleware functions in **Express.js** are functions that have access to the request (`req`), response (`res`), and the next middleware function in the application's request-response cycle. They can modify `req` and `res` objects, execute any code, and end the request-response cycle or call the next middleware function.

Two Examples of Middleware in Express.js:

1. **Custom Middleware Example**: A middleware function that logs every request made to the server.

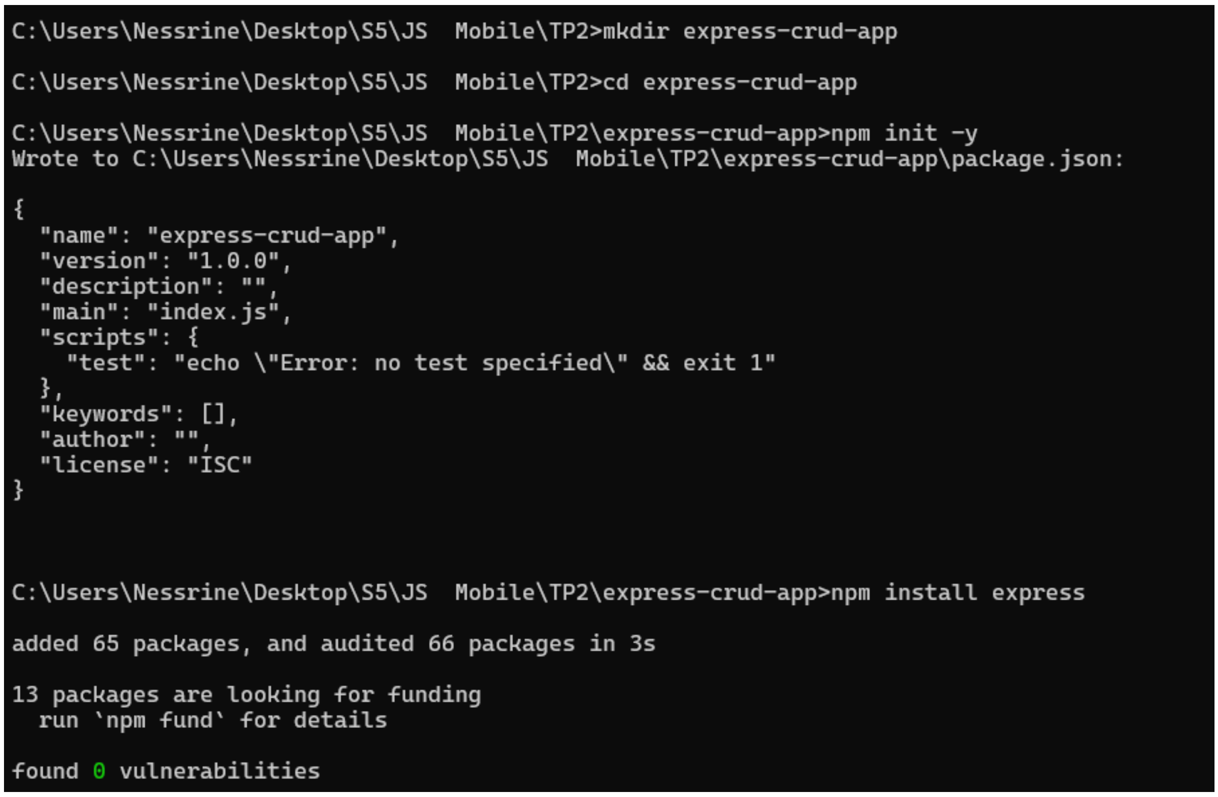
```
const express = require('express');
const app = express();

// Custom middleware to log each request
app.use((req, res, next) => {
  console.log(`${req.method} request for '${req.url}'`);
  next(); // Call the next middleware or route handler
});

app.get('/', (req, res) => {
  res.send('Hello, World!');
});

app.listen(3000, () => {
  console.log('Server running on port 3000');
});

```

2. **Built-in Middleware Example**: Using the built-in middleware `express.json()` to parse JSON request bodies.

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

// Middleware to parse JSON request bodies
app.use(express.json());

app.post('/data', (req, res) => {
  console.log(req.body); // Access the parsed JSON body
  res.send('Data received');
});

app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

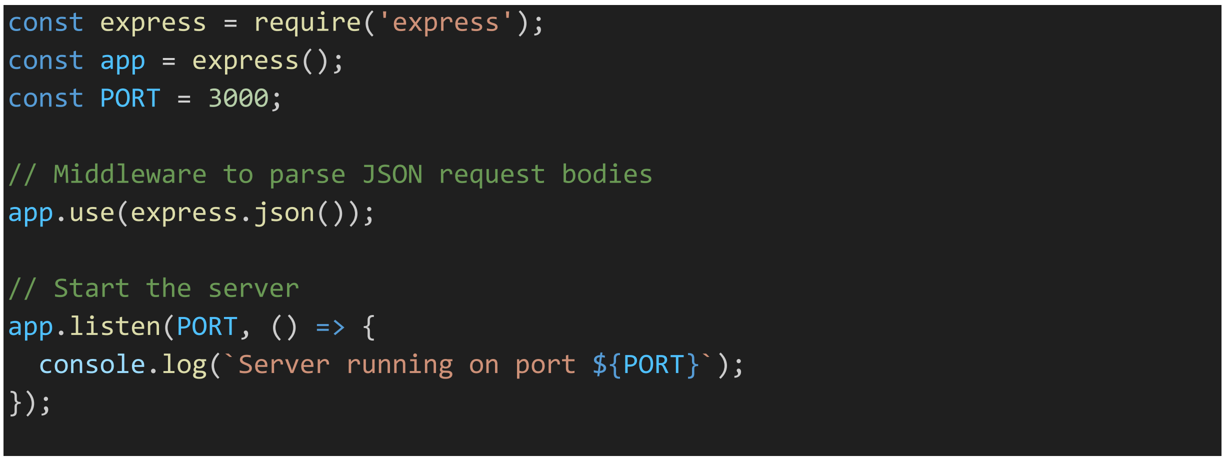
Creating a Simple CRUD Application with Express.js

1. Create a Project Directory
2. Initialize a Node.js Project
3. Install Express



```
C:\Users\Nessrine\Desktop\S5\JS  Mobile\TP2>mkdir express-crud-app

C:\Users\Nessrine\Desktop\S5\JS  Mobile\TP2>cd express-crud-app

C:\Users\Nessrine\Desktop\S5\JS  Mobile\TP2\express-crud-app>npm init -y
Wrote to C:\Users\Nessrine\Desktop\S5\JS  Mobile\TP2\express-crud-app\package.json:

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



C:\Users\Nessrine\Desktop\S5\JS  Mobile\TP2\express-crud-app>npm install express

added 65 packages, and audited 66 packages in 3s

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

found 0 vulnerabilities
```

4- Set Up Express: Run the Server Using *“app.listen”*



```
const express = require('express');
const app = express();
const PORT = 3000;

// Middleware to parse JSON request bodies
app.use(express.json());

// Start the server
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

5- Create a POST Endpoint (Add an Item)

```
let items = []; // Local variable to store items

// POST endpoint to add an item
app.post('/items', (req, res) => {
  const newItem = req.body;
  items.push(newItem);
        res.status(201).send('Item added');
      });
```

6. Create a GET Endpoint (Retrieve All Items)

```
// GET endpoint to retrieve all items

app.get('/items', (req, res) => {
    res.json(items);
  });
```

7. Create a GET Endpoint by ID (Retrieve a Specific Item)

```
// GET endpoint to retrieve an item by ID
app.get('/items/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const item = items.find(i => i.id === id);
    if (item) {
      res.json(item);
    } else {
      res.status(404).send('Item not found');
    }
  });
```

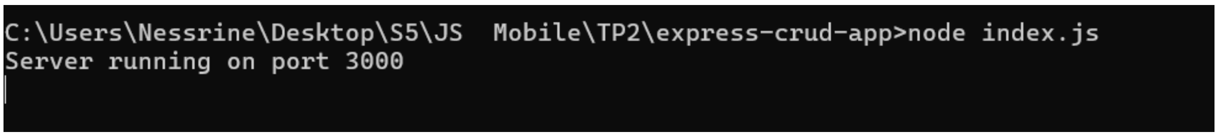
8. Create a PUT Endpoint (Update an Item)

```
// PUT endpoint to update an item by ID
app.put('/items/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const index = items.findIndex(i => i.id === id);
    if (index !== -1) {
      items[index] = req.body;
      res.send('Item updated');
    } else {
      res.status(404).send('Item not found');
    }
  });
```

9. Create a DELETE Endpoint (Delete an Item)

```
// DELETE endpoint to delete an item by ID
app.delete('/items/:id', (req, res) => {
    const id = parseInt(req.params.id);
    const index = items.findIndex(i => i.id === id);
    if (index !== -1) {
      items.splice(index, 1);
      res.send('Item deleted');
    } else {
      res.status(404).send('Item not found');
    }
  });
```

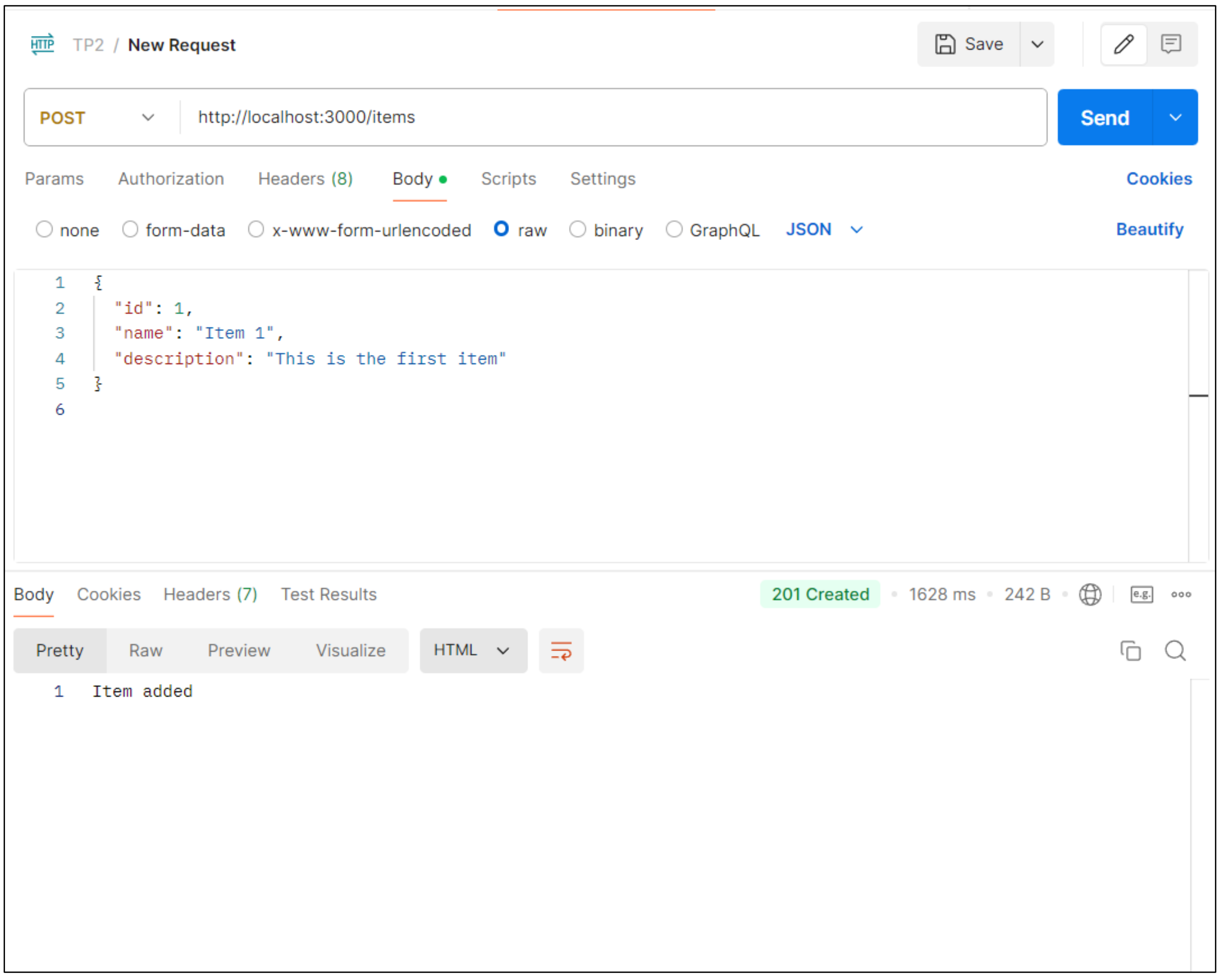
10. Start the Server



```
C:\Users\Nessrine\Desktop\S5\JS  Mobile\TP2\express-crud-app>node index.js
Server running on port 3000
```

11. Test the Endpoints Using Postman

- **Tester le POST Endpoint**



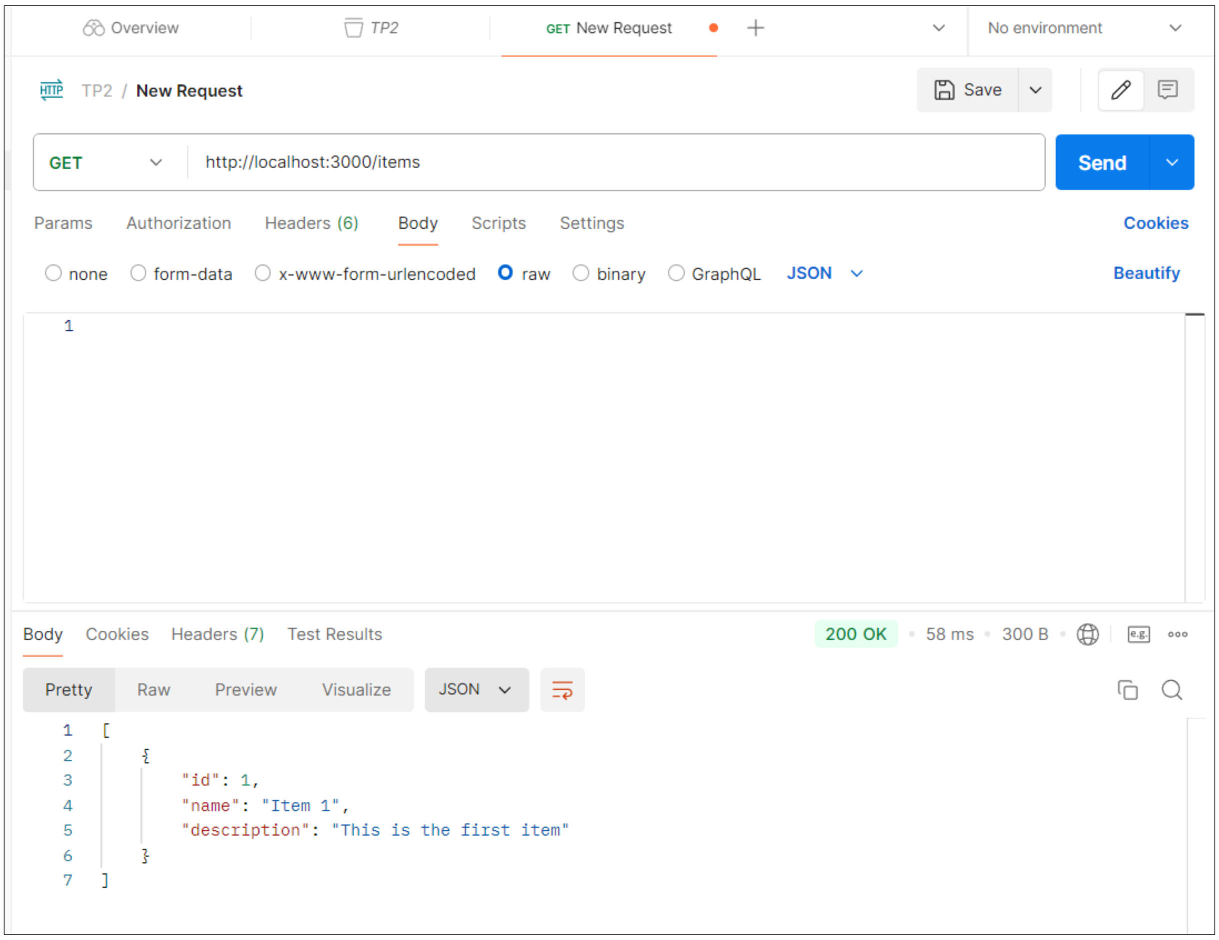
POST http://localhost:3000/items

```
{
  "id": 1,
  "name": "Item 1",
  "description": "This is the first item"
}
```

201 Created • 1628 ms • 242 B

```
Item added
```

Tester le GET Endpoint pour récupérer tous les items



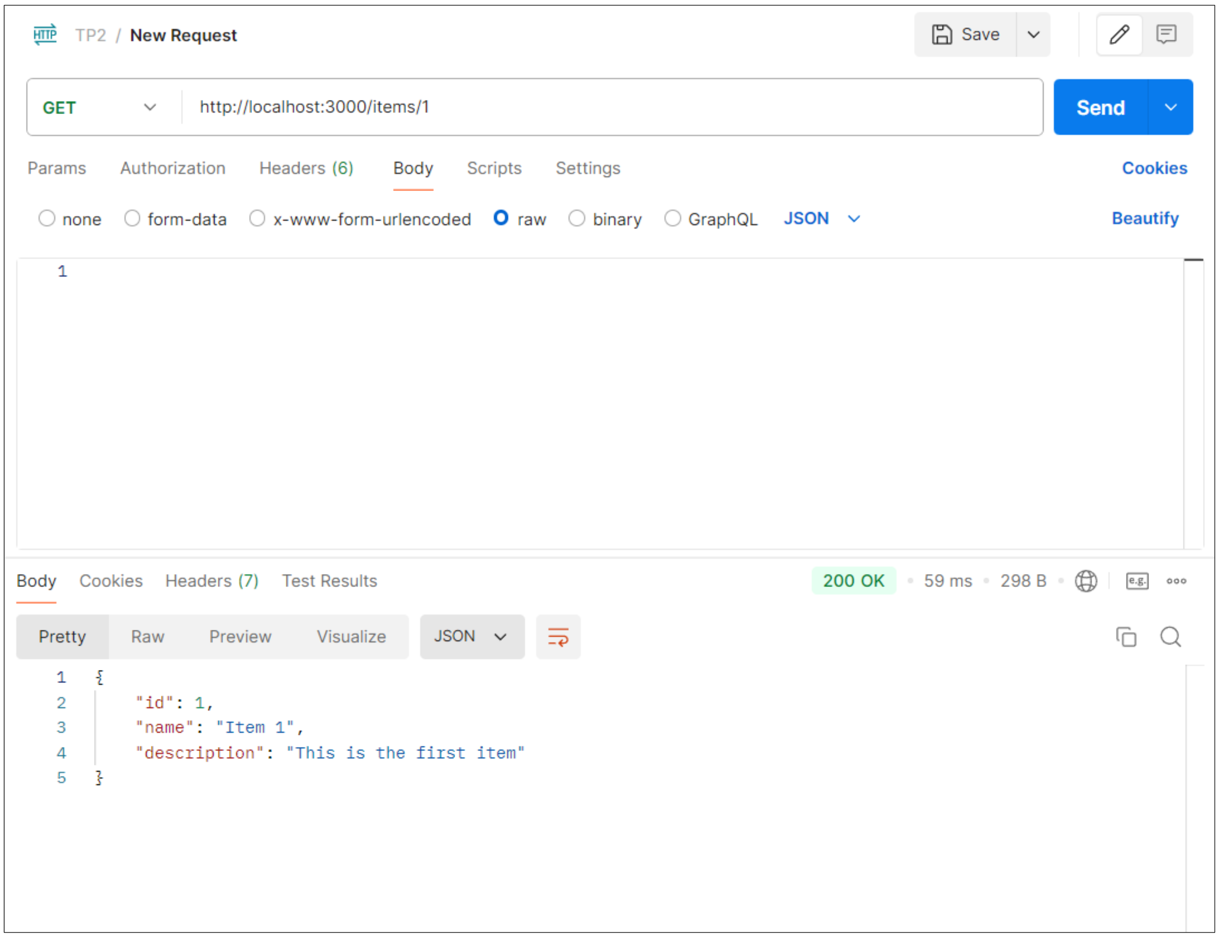
GET http://localhost:3000/items

Body: [✓ raw] JSON

200 OK • 58 ms • 300 B

```
[
    {
        "id": 1,
        "name": "Item 1",
        "description": "This is the first item"
    }
]
```

5. **Tester le GET Endpoint pour récupérer un item par ID**



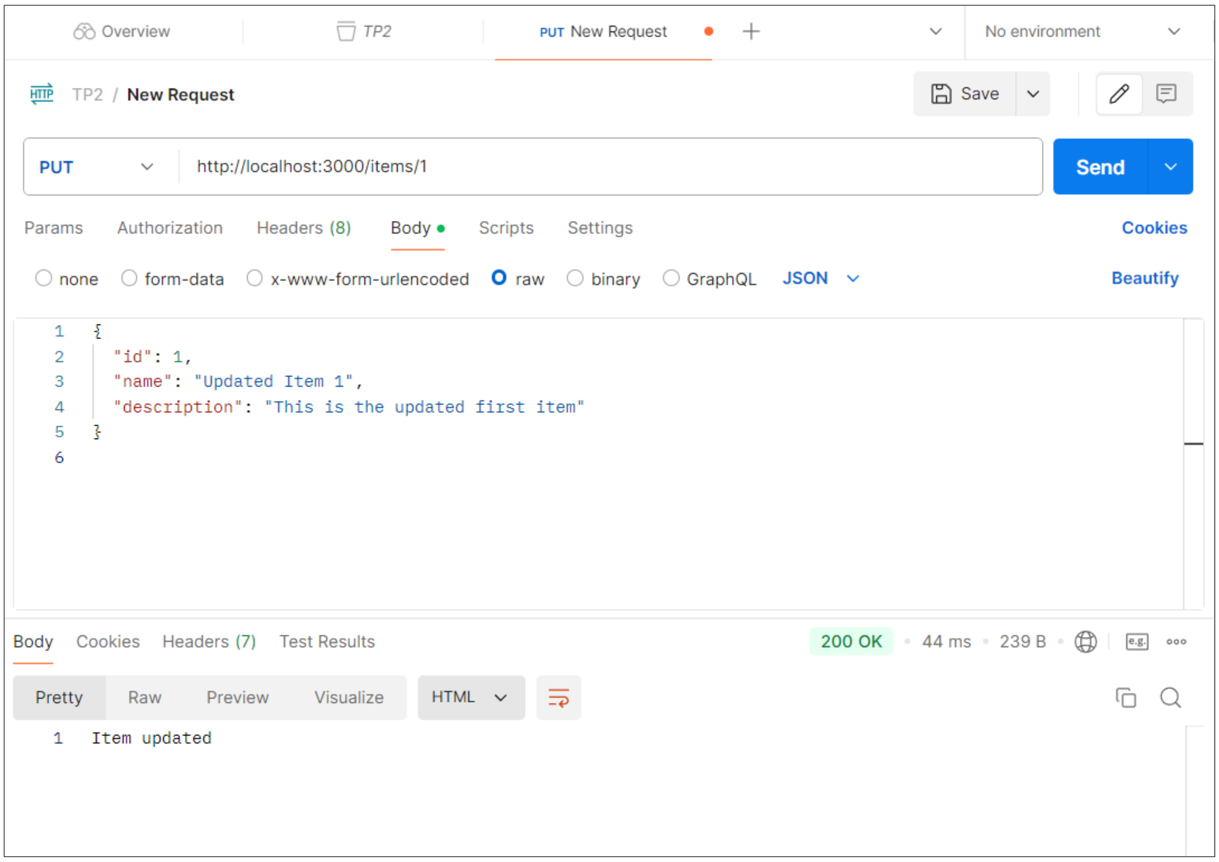
GET http://localhost:3000/items/1

Body: raw [✓] JSON

200 OK • 59 ms • 298 B

```
{
    "id": 1,
    "name": "Item 1",
    "description": "This is the first item"
}
```

6. Tester le PUT Endpoint pour mettre à jour un item



PUT http://localhost:3000/items/1

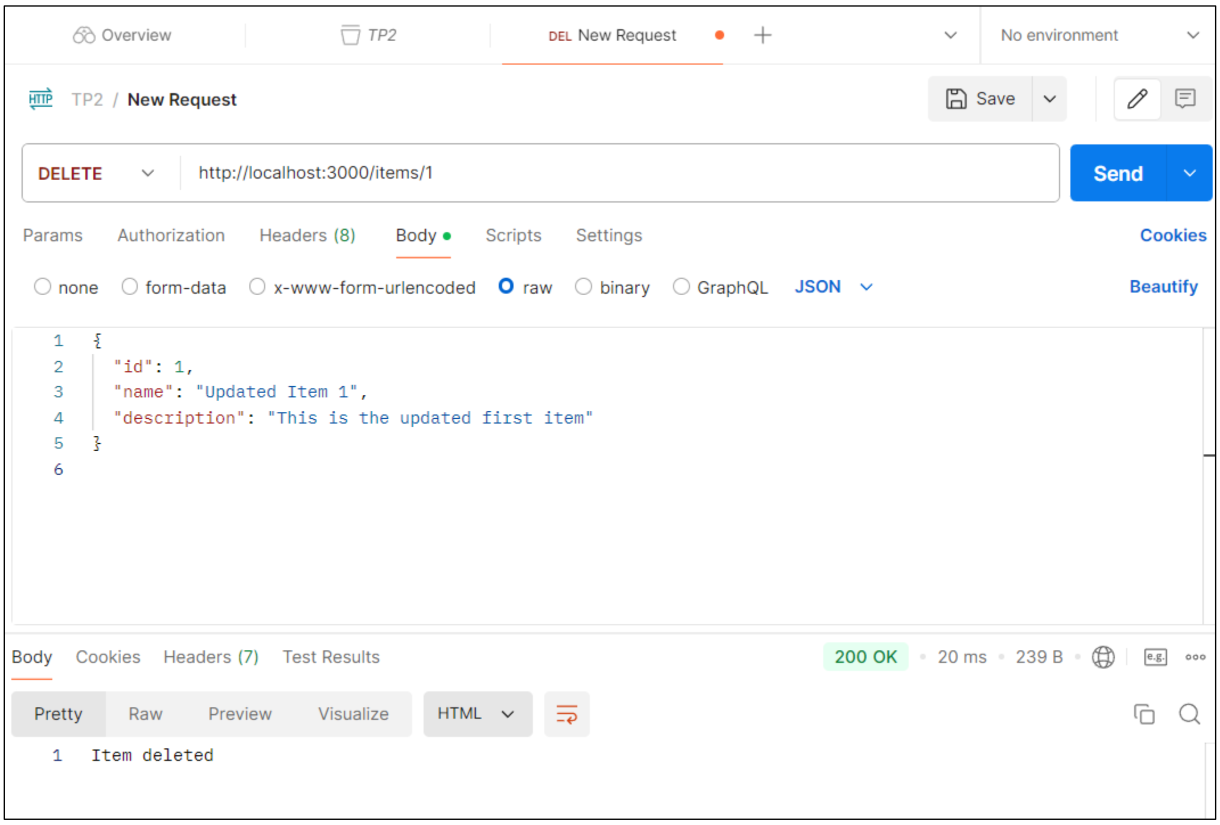
Body: [✓ raw] JSON

```
{
  "id": 1,
  "name": "Updated Item 1",
  "description": "This is the updated first item"
}

```

200 OK · 44 ms · 239 B

```
Item updated
```

7. Tester le DELETE Endpoint pour supprimer un item



DELETE http://localhost:3000/items/1

```
{
  "id": 1,
  "name": "Updated Item 1",
  "description": "This is the updated first item"
}

```

200 OK • 20 ms • 239 B

```
Item deleted
```