TP (Express.JS)

Filière : Informatique et Ingénierie des Données

Niveau : 3ème année

21 octobre 2024

Made By :
Meriem Fattah
Haitam Mabrouk

What is ExpressJs ?

Express.js is a popular web application framework for Node.js, the JavaScript runtime environment. It provides a set of features and tools that simplify the process of building web applications and APIs using Node.js. With Express.js, developers can quickly create server-side logic, handle HTTP requests, manage routes, and render dynamic content.

What are middlewares ?

Middlewares are pieces of code that get executed between receiving the request and returning the response. They sit in the middle, or "in the middleware", of the request-response cycle.
The key points about middlewares in Express.js are :

- They have access to the request object (req) and the response object (res).
- They can perform various tasks, such as logging, parsing request bodies, adding response headers, executing custom logic etc

Creating a simple CRUD application

1 - We've created a project directory called 'express-crud-app'
2 - We've initialized the Node Project using the command :

```
PS C:\Users\MSI\Desktop\express-crud-app> npm init -y
Wrote to C:\Users\MSI\Desktop\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"
}
```

3 - We've installed Express using the command :

```
PS C:\Users\MSI\Desktop\express-crud-app> npm i express

added 65 packages, and audited 66 packages in 3s

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

found 0 vulnerabilities
```

Creating a simple CRUD application

4 - We've setup our Express server and bind it to a port 3000 :

server.js > ...

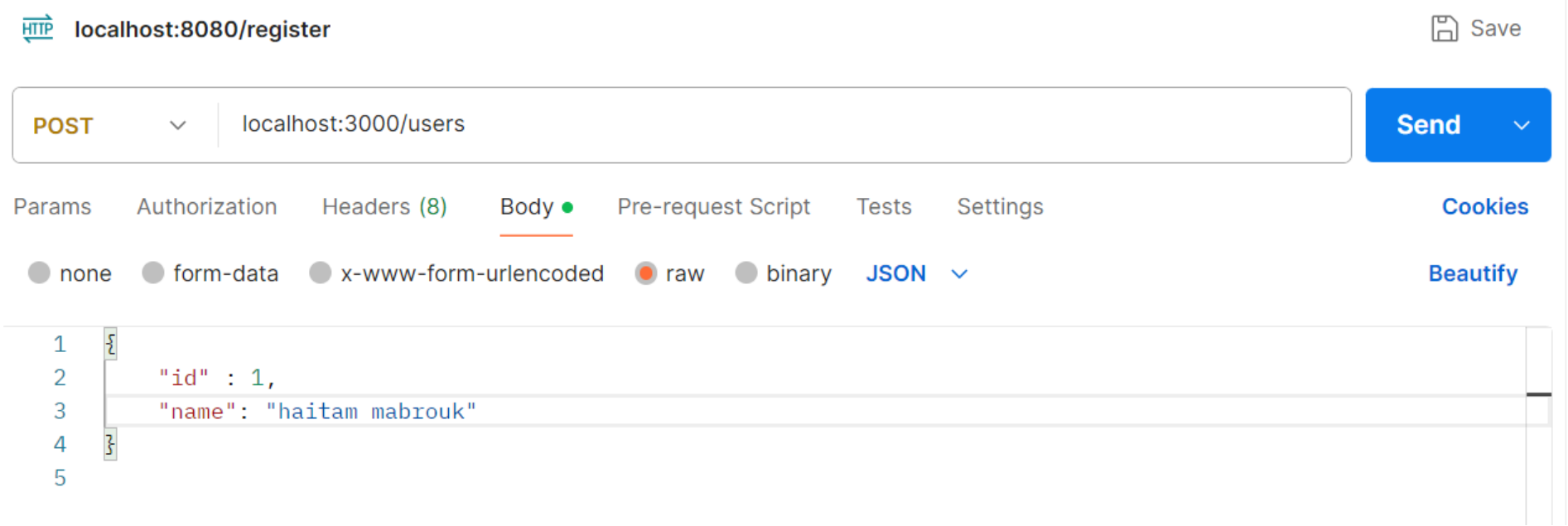
```
1   const express = require('express')
2   const app = express()
3
4   app.listen(3000)
```

Cannot GET /

5 - 6 - We are going to create the POST method handler which allows us to add a new user to a list of users and a GET method handler which allows us to retrieve all users in the list :

```
8   router.route("/")
9       .post((req, res) => {
10          const newUser = req.body
11          users.push(newUser)
12          res.status(201).json(`The user : ${newUser.name} has been added succesfully`)
13      })
14      .get((req, res) => {
15          if (users.length > 0) {
16              return res.status(200).json(users)
17          }
18          res.status(404).json(`There is no User Yet !`)
19      })
```

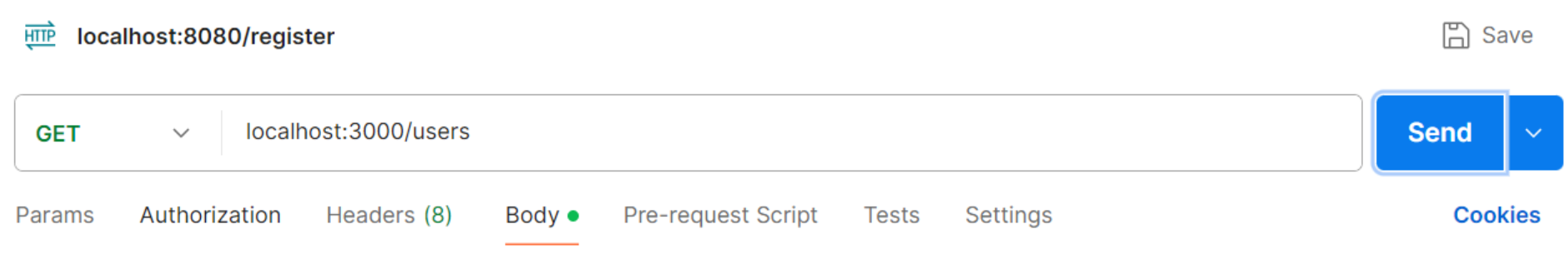
Demo (using Postman) :
- Creating a new user :
Request :



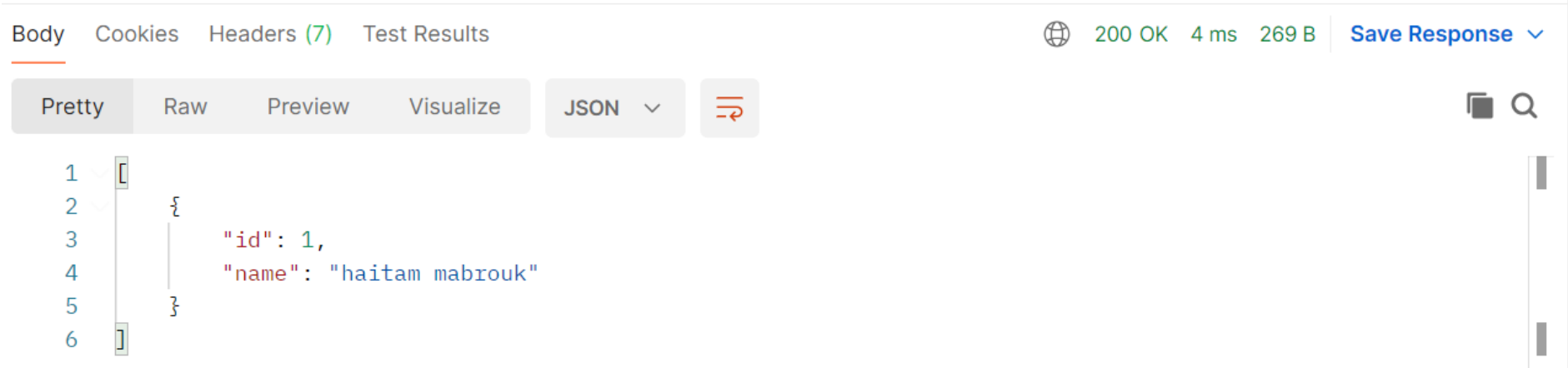
Response :

- Retrieving all users :

Request :



Response :



6 - 7 - 8 - We are going to create the GET method handler for getting the user based on it's id and a PUT method handler for updating the user's name based on the id and a DELETE method handler which allows us to delete a user based on the id :

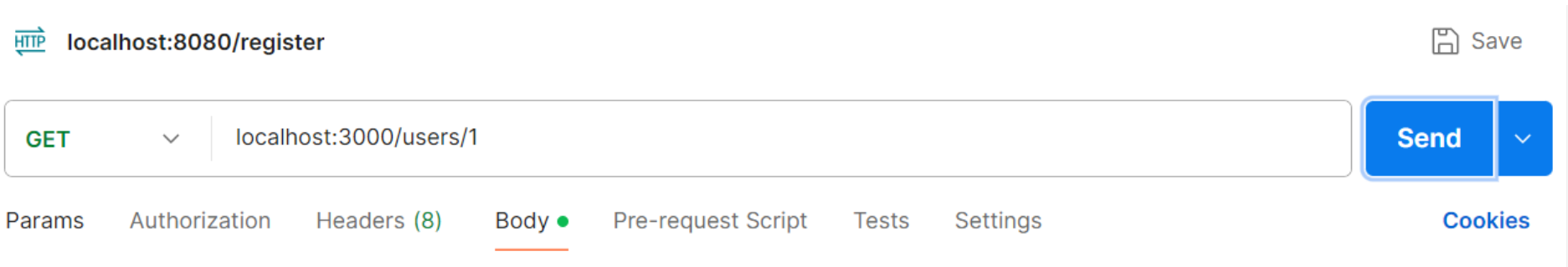
routes > users.js > get() callback

```
21  router.route("/:id")
22      .get((req, res) => {
23          const id = parseInt(req.params.id)
24          for (user of users) {
25              if (user.id === id) {
26                  return res.status(200).json(`The User that you are looking for : ${JSON.stringify(user)}`)
27              }
28          }
29          res.status(404).json(`User Not Found`)
30      })
31      .put((req, res) => {
32          const id = parseInt(req.params.id)
33          const { name } = req.body
34          for (user of users) {
35              if (user.id === id) {
36                  user.name = name
37                  return res.status(200).json(`The User has been updated succesfully : ${JSON.stringify(user)}`)
38              }
39          }
40          res.status(404).json(`User Not Found`)
41      })
42      .delete((req, res) => {
43          const id = parseInt(req.params.id)
44          for(user of users) {
45              if(user.id === id) {
46                  const index = users.findIndex(user => user.id === id)
47                  users.splice(index, 1)
48                  return res.status(200).json(`The user with the Id : ${id} has been deleted`)
49              }
50              res.status(404).json(`User Not Found`)
51          }
52      })
```

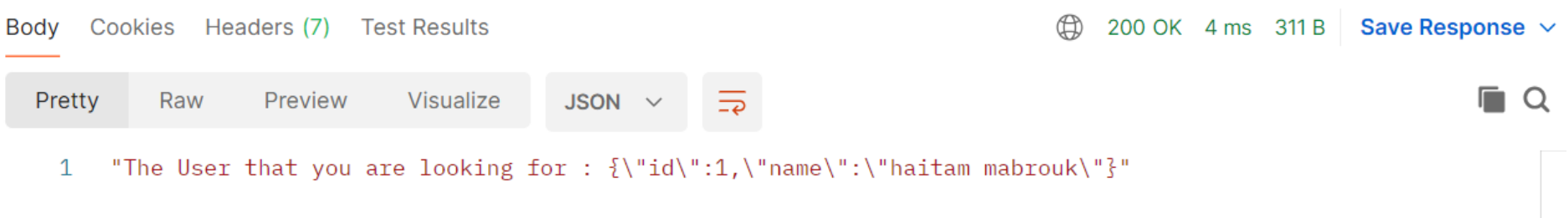
Demo (using Postman) :

- Retrieving the user based on the id :

Request :

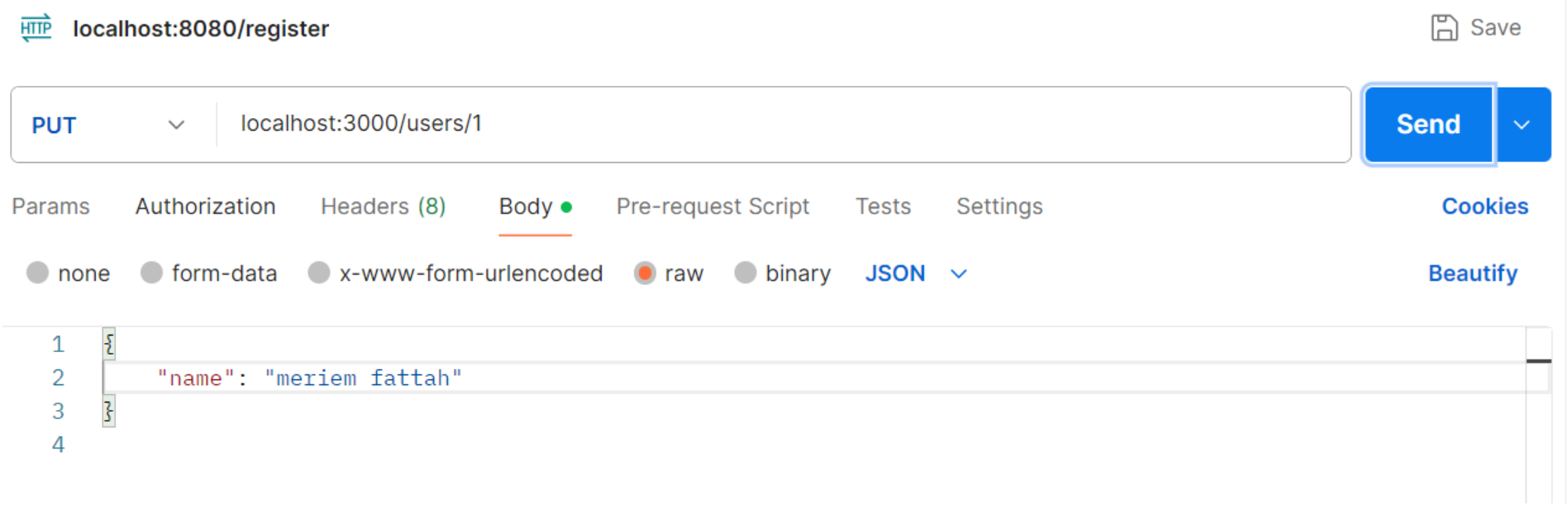


Response :



- Updating the user's name based on the id :

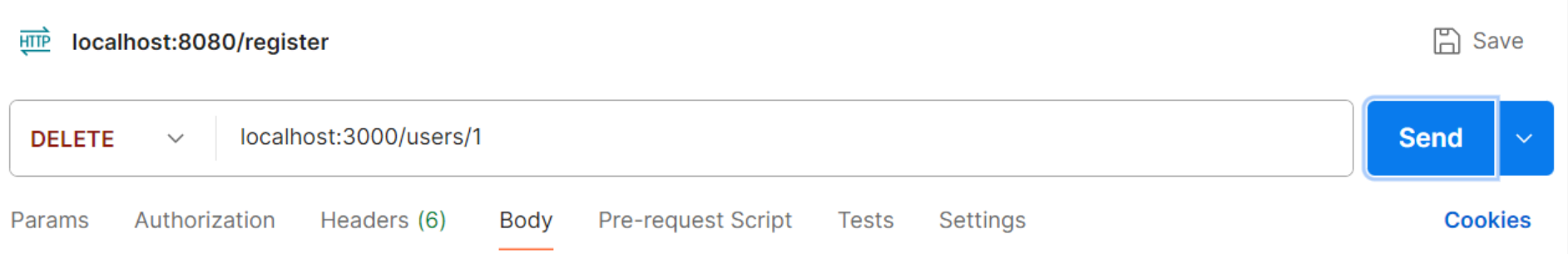
Request :



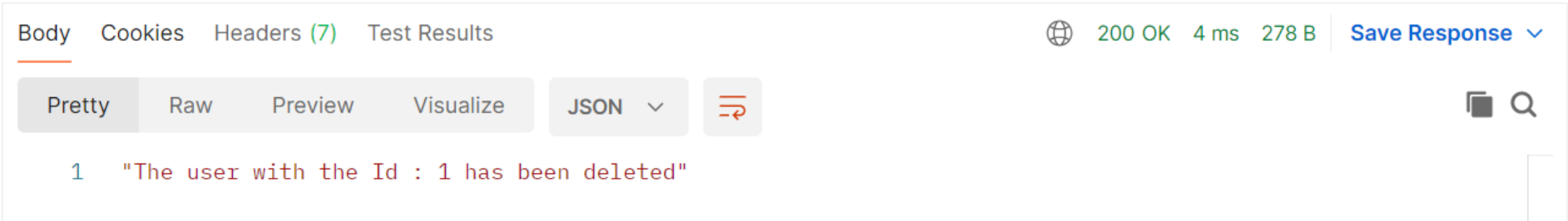
Response :

- Deleting user by id :

Request :


localhost:8080/register
Save
DELETE
localhost:3000/users/1
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies

Response :


Body
Cookies
Headers (7)
Test Results
200 OK
4 ms
278 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 "The user with the Id : 1 has been deleted"