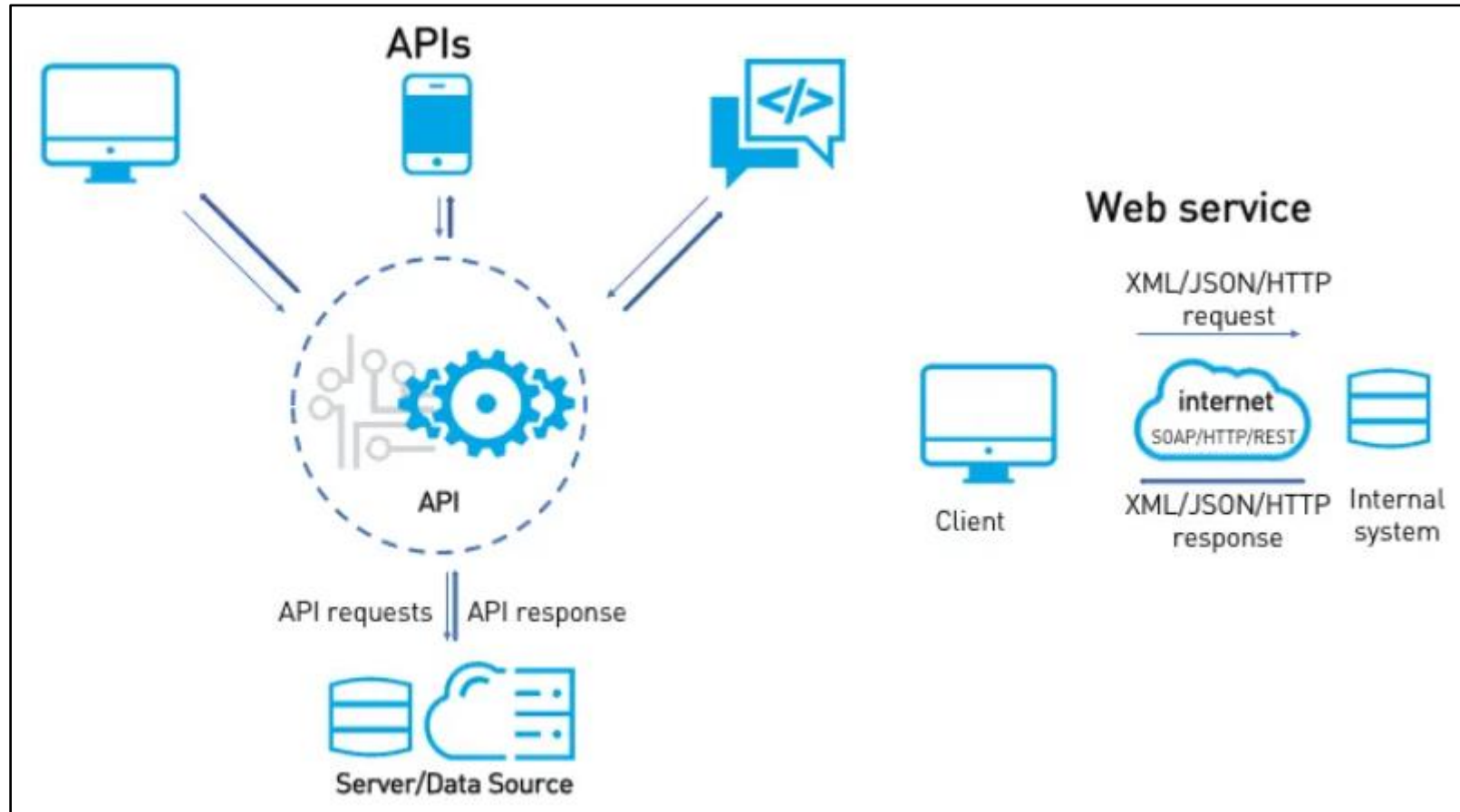


Introduction to NodeJS

Instructor: Nguyễn Trung Nghĩa

REST API VS Web Services



Layout

I. Introduction

II. Installation

III. NodeJS & ExpressJS

1. Install package & config
2. Create GET & POST request
3. Work with router and handler

IV. Create DB using MySQL

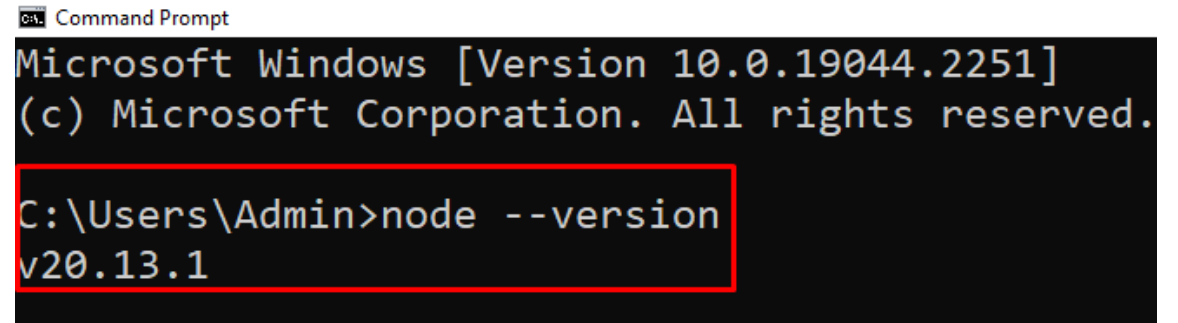
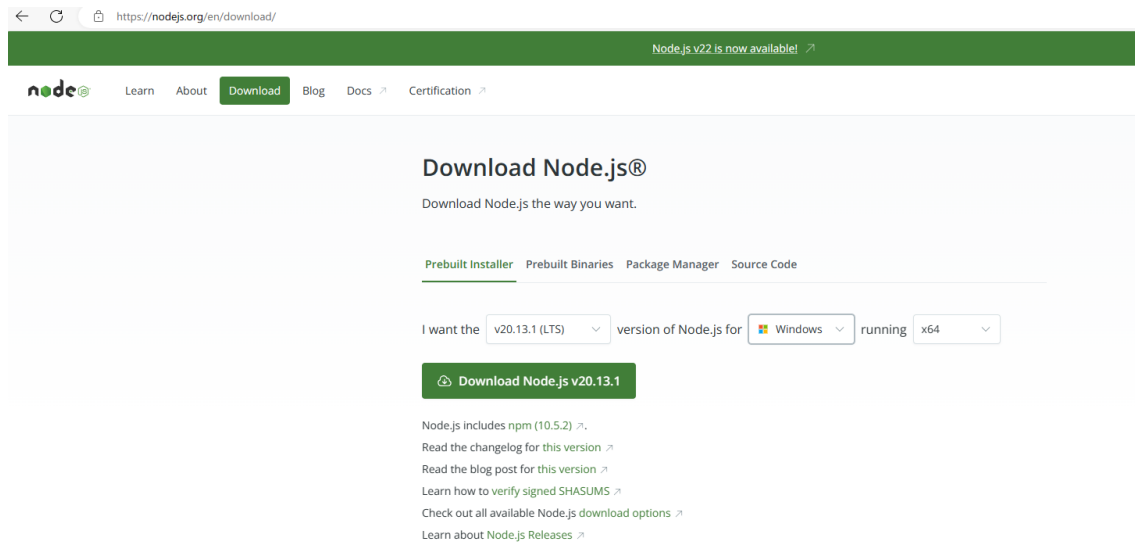
V. Connect to ExpressJS

Tools



Installation

[Node.js — Download Node.js®
\(nodejs.org\)](https://nodejs.org)



Download and install Postman

The screenshot shows the Postman website's download page. The browser address bar displays `https://www.postman.com/downloads/`. The website header includes navigation links: Product, Pricing, Enterprise, Resources and Support, and Public API Network. On the right, there are buttons for Contact Sales, Sign In, and Sign Up for Free. The main heading is "Download Postman", followed by the text: "Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman."

The Postman app

Download the app to get started with the Postman API Platform.

[Windows 64-bit](#)

By downloading and using Postman, I agree to the [Privacy Policy](#) and [Terms](#).

[Release Notes](#) →

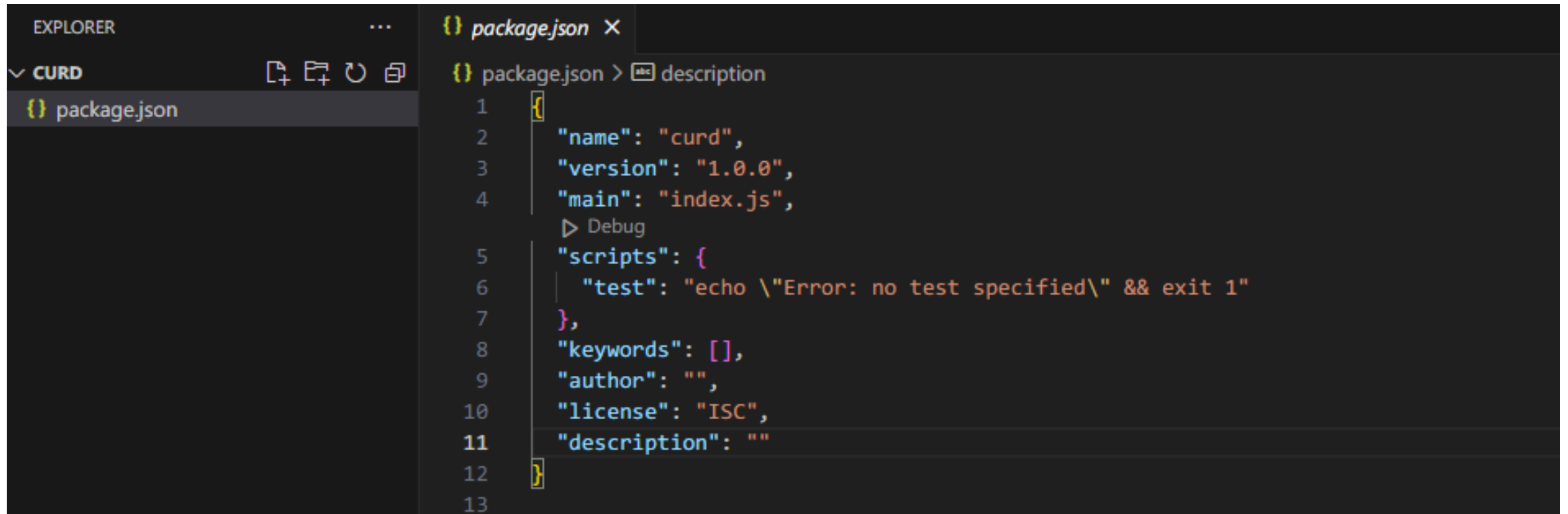
Not your OS? Download for Mac ([Intel Chip](#), [Apple Chip](#)) or Linux

The right side of the image shows a preview of the Postman application interface. It features a sidebar with "Collection", "Environment", and "History" sections. The main workspace displays a "Notion's Public Workspace" with a "New" button and an "Import" button. A search bar labeled "Search Postman" is present. The active workspace shows a "GET Retrieve a database" endpoint with a URL of `https://api.notion.com/v1/databases/:id`. Below the URL bar, there are tabs for Params, Auth, Headers(10), Body, Scripts, and Settings. A "Send" button is visible. The bottom section shows a table with columns: KEY, VALUE, DESCRIPTION, and Bulk edit. The table contains one row with the values "Key", "Value", and "Description".

[Download Postman | Get Started for Free](#)

Create project

```
PS D:\Desktop\curd> npm init --y
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a folder named 'CURD' containing a file named 'package.json'. The main editor area displays the contents of 'package.json' with the following JSON structure:

```
1 {  
2   "name": "curd",  
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
```

Install packages

```
npm i mysql2 express  
npm i -D nodemon
```

```
PS D:\Desktop\curd> npm i mysql2 express
```

```
added 75 packages, and audited 76 packages in 6s
```

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

```
found 0 vulnerabilities
```

```
PS D:\Desktop\curd> npm i -D nodemon
```

```
added 29 packages, and audited 105 packages in 4s
```

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


VS Code interface showing the file explorer on the left and the code editor on the right. The file explorer shows the project structure with `package.json` selected. The code editor displays the `package.json` file content. Red boxes highlight the `"type": "module"` field and the `"scripts"` object. Red arrows point from text annotations to these boxes.

```
1 {
2   "name": "curd",
3   "version": "1.0.0",
4   "main": "index.js",
5   "type": "module",
6   "scripts": {
7     "dev": "nodemon index.js",
8     "start": "node index.js",
9     "test": "echo \"Error: no test specified\" && exit 1"
10  },
11  "keywords": [],
12  "author": "",
13  "license": "ISC",
14  "description": "",
15  "dependencies": {
16    "express": "^4.19.2",
17    "mysql2": "^3.9.7"
18  },
19  "devDependencies": {
20    "nodemon": "^3.1.0"
21  }
22 }
```

Annotations:

- use type module to import lib (points to `"type": "module"`)
- Tell server which file will be run when server start (points to `"start": "node index.js"`)

Terminal window showing the command `npm run dev` being executed. The output shows the nodemon process starting and watching for changes.

```
To see a list of supported npm commands, run:
npm help
PS D:\Desktop\curd> npm run dev

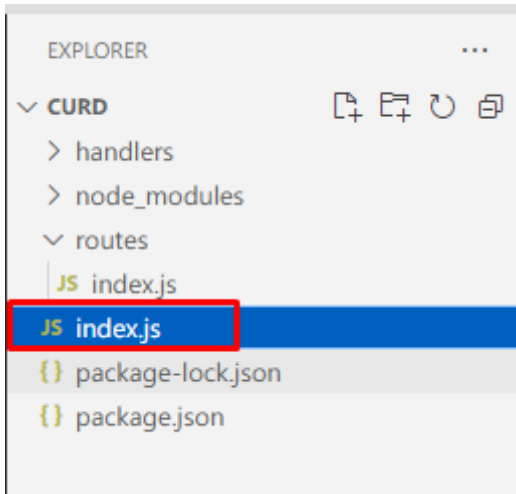
> curd@1.0.0 dev
> nodemon index.js

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Hello
[nodemon] clean exit - waiting for changes before restart
```

Left sidebar of the terminal window:

- > OUTLINE
- > TIMELINE

Create Server



```
import express from "express" // import ExpressJS framreword

const app = express(); // use the expressJS

const PORT = process.env.PORT || 5000; // define PORT
app.listen(PORT, () => console.log("Server Open At port: ", PORT));
```

```
GET
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
Server Open At port: 5000
```



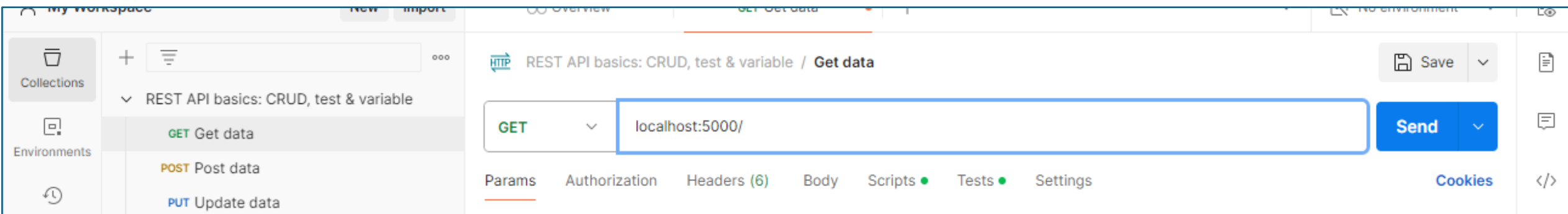
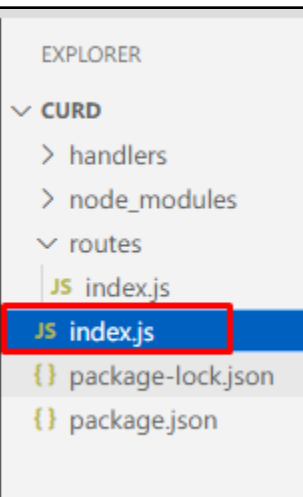
Create GET request

```
import express from "express" //

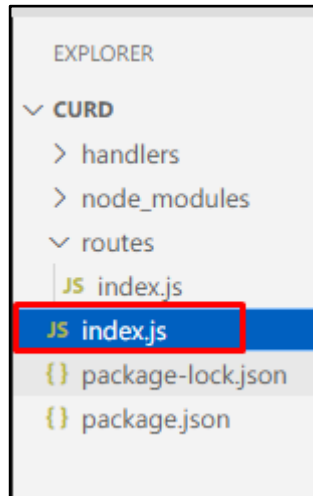
const app = express(); // use the expressJS

const PORT = process.env.PORT || 5000; // define PORT
app.listen(PORT, () => console.log("Server Open At port: ", PORT));

app.get("/", (req, res, next) => {
  console.log(req.method)
  res.send("<h1>Hello</h1>"); // send response
}); // create GET request
```

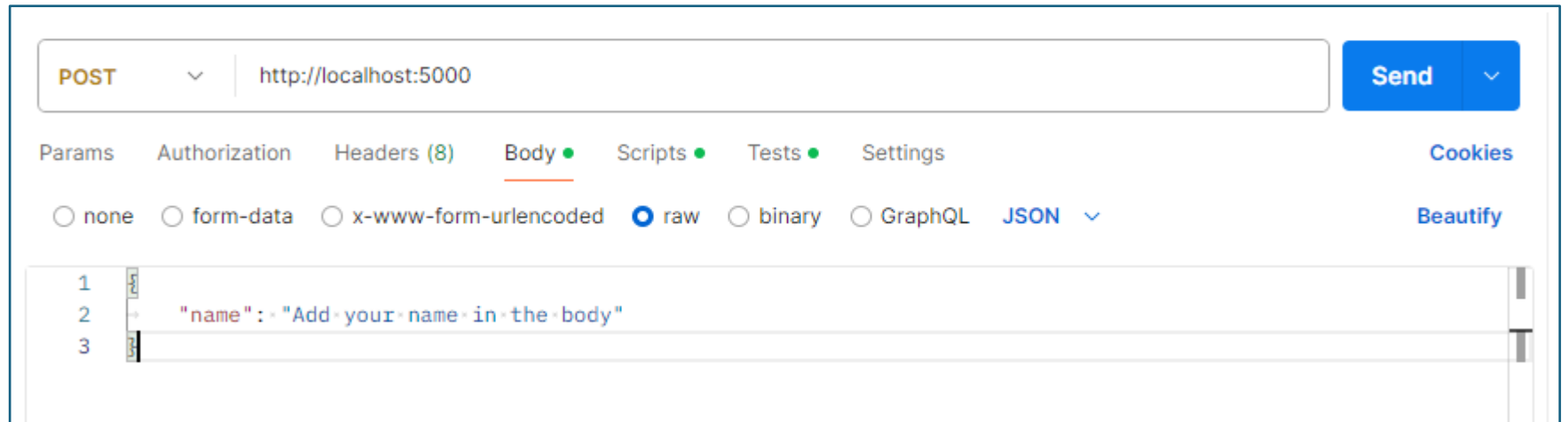


Create POST request



```
app.use(express.json()); // use library to work with JSON data

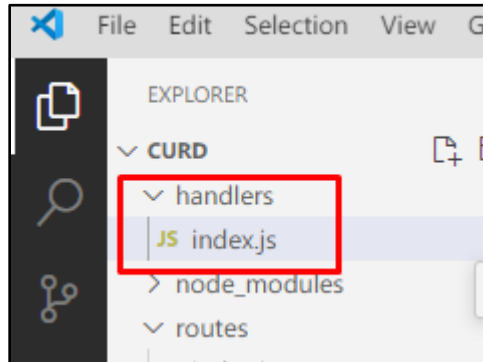
app.post('/', (req, res, next) => {
  console.log(req.body);
  res.send("POST");
});
```



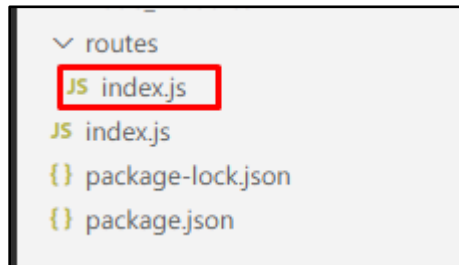
Using Postman to test

Work with Routes

Create Routes & Handlers

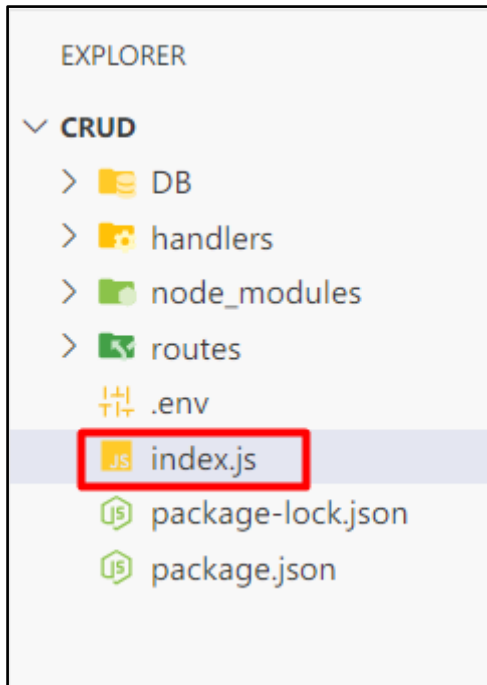


```
export const getAllProducts = async (req, res) => {  
    return res.send("GET ALL PRODUCTS");  
};
```



```
import { Router } from "express";  
import { getAllProducts } from "../handlers/index.js";  
  
const appRouter = Router();  
  
appRouter.get("/", getAllProducts)  
  
export default appRouter;
```

Update the main index.js



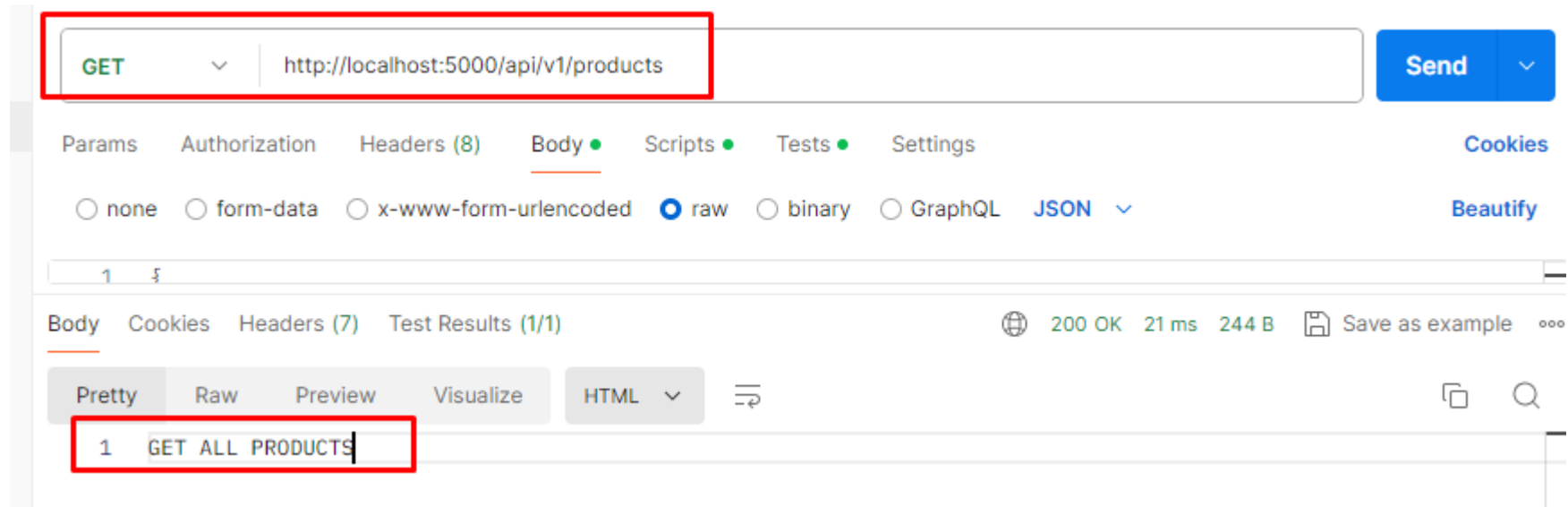
```
import express from "express";
import appRoute from "../routes/index.js";

const app = express();

app.use(express.json());
app.use("/api/v1/products", appRoute);

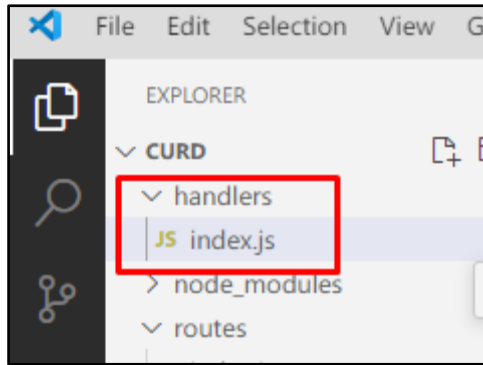
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log("Server Open At PORT: ", PORT));
```

Time to test



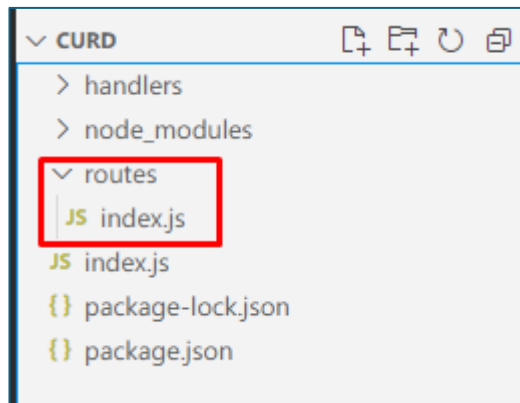
So, what happen after all?

Let's complete Handlers



```
export const getAllProducts = async (req, res) => {  
    return res.send("GET ALL PRODUCTS");  
};  
  
export const getProduct = async (req, res) => {};  
export const createProduct = async (req, res) => {};  
export const updateProduct = async (req, res) => {};  
export const deleteProuct = async (req, res) => {};
```

... and complete route



```
import { Router } from "express";
import { createProduct,
        deleteProuct,
        getAllProducts,
        getProduct,
        updateProduct } from
    "../handlers/index.js";

const appRouter = Router();

appRouter.get("/", getAllProducts);
appRouter.get("/:id", getProduct);
appRouter.post("/create", createProduct);
appRouter.put("/update/:id", updateProduct);
appRouter.delete("/delete/:id", deleteProuct);

export default appRouter;
```

V. Working with DB

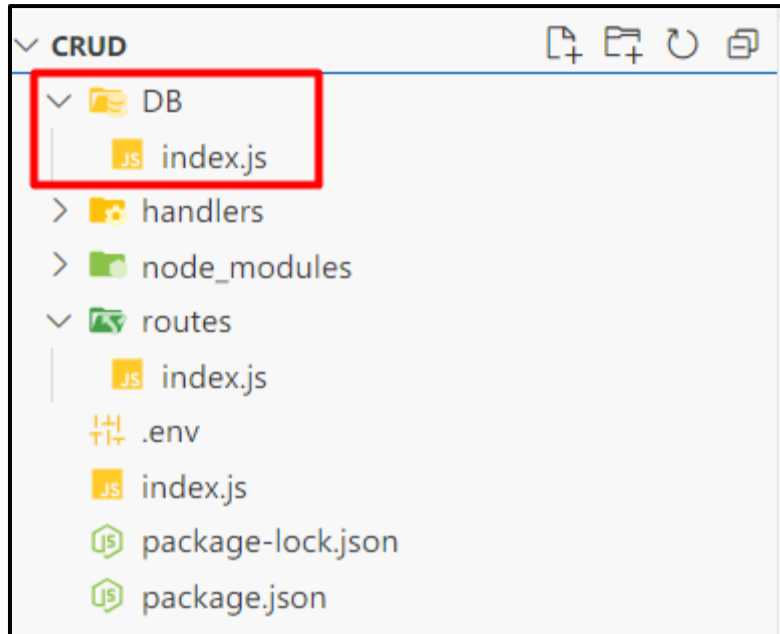
If you use MAC run mySQL by docker

```
PS F:\Yt Projects\Node\Node mysql> docker run --name sqldb -d -p 3306:3306 --rm -v  
mysqldata:/var/lib/mysql -e MYSQL_ROOT_PASSWORD='test' mysql:latest
```

0. Create a new DB

- Create DB using MySQL and named is as WADLab05

1. Create connection pool

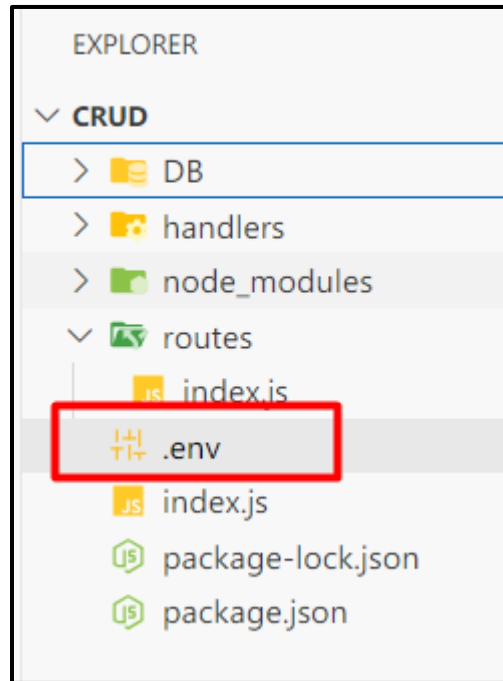


```
import { createPool } from "mysql2";

const pool = createPool({
  port: 3306,
  password: "root"
});
```

2. Create .env (/index.js)

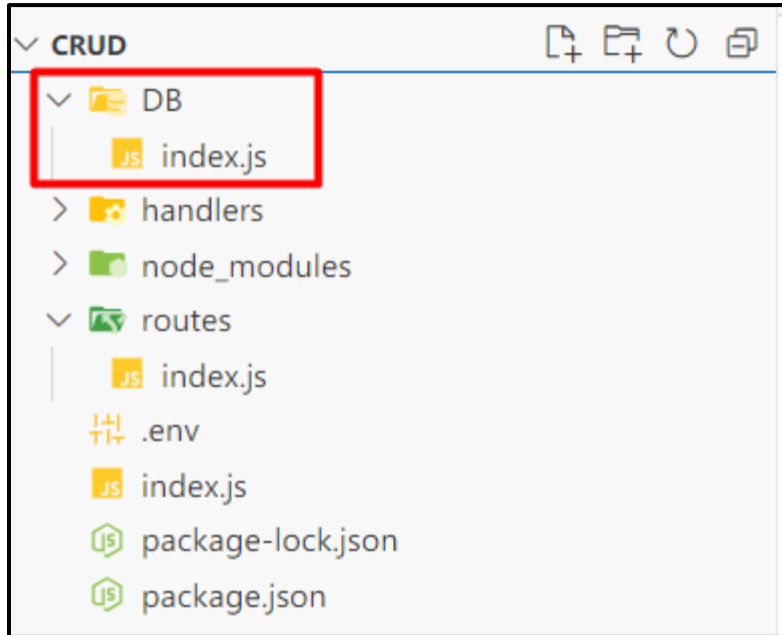
1. Run `npm i dotenv`
2. Create .env file



```
MYSQL_PASSWORD = "root",  
MYSQL_DATABASE_NAME = "WADlab05"  
MYSQL_HOST = "localhost"  
MYSQL_USER = "root"  
MYSQL_PORT = 3306
```

```
PORT = 5000
```

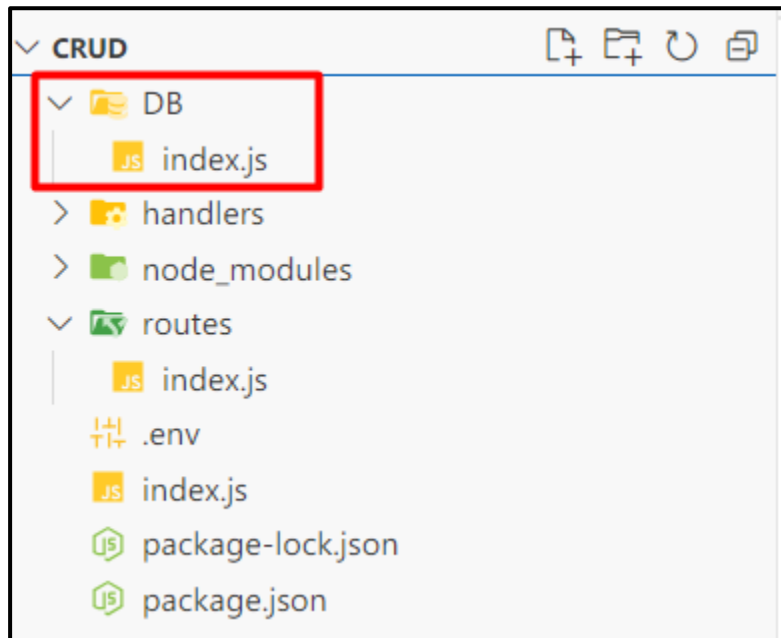
4. Update DB connect using env (/DB/index.js)



```
config() // load .env file

const pool = createPool({
  user: process.env.MYSQL_USER,
  port: process.env.MYSQL_PORT,
  password: process.env.MYSQL_PASSWORD,
  host: process.env.MYSQL_HOST,
  database: process.env.MYSQL_DATABASE_NAME,
});
```


5. Connected to DB (same file with the above)



```
// other code

const connectToDatabase = async () => {
  try {
    await pool.getConnection();
    console.log("MySQL Connection Successful");
  } catch (error) {
    console.log("Database Connection Error");
    console.log(error);
    throw error
  }
};

export {connectToDatabase, pool};
```

6. Call the connection (/index.js)

```
15  const PORT = process.env.PORT || 5000;
16
17  connectToDatabase()
18    .then(() => {
19      app.listen(PORT, () => console.log("Server Open At port:", PORT));
20    })
21    .catch((err) => {
22      console.log("Error occurred with mysql connection. Error = ", err);
23      process.exit(0);
24    });
25
```

Now you need to follow the video to create API for database

- <https://youtu.be/xwjlBGTGDc4?si=7dGWVIXoAmOQd5GG&t=3839>

Reference

1. [Node.js — Introduction to Node.js \(nodejs.org\)](https://nodejs.org/en/about/what-is-node-js)
2. <https://www.youtube.com/watch?v=xwjlBGTGDc4>