

---

# **LAPORAN PRAKTIKUM MAHASISWA PEMROGRAMAN WEB FRAMEWORK**

---

## **Membuat RESTful API (NODEJS dan MySQL)**



**Oleh:**

**NAMA : Haldian**  
**NPM : 20753050**  
**KELAS : Manajemen Informatika 4B**

**Dosen:**

**Tri Sandhika Jaya, S.Kom., M.Kom.**

**D3 - MANAJEMEN INFORMATIKA  
JURUSAN EKONOMI DAN BISNIS  
POLITEKNIK NEGERI LAMPUNG**

April 2022

## A. Tujuan (Capaian Pembelajaran)

Setelah praktikum ini, praktikan mahasiswa diharapkan dapat:

Membuat Restful Application Programming Interface(API)

## B. Peralatan yang digunakan

1. Web Editor : Visual Studio Code
2. Web Browser : Google Chrome
3. OS PC : Windows 10
4. XAMPP : versi 7.2
5. NodeJs : 16.14.0 LTS
6. GitBash

## C. Hasil Praktikum

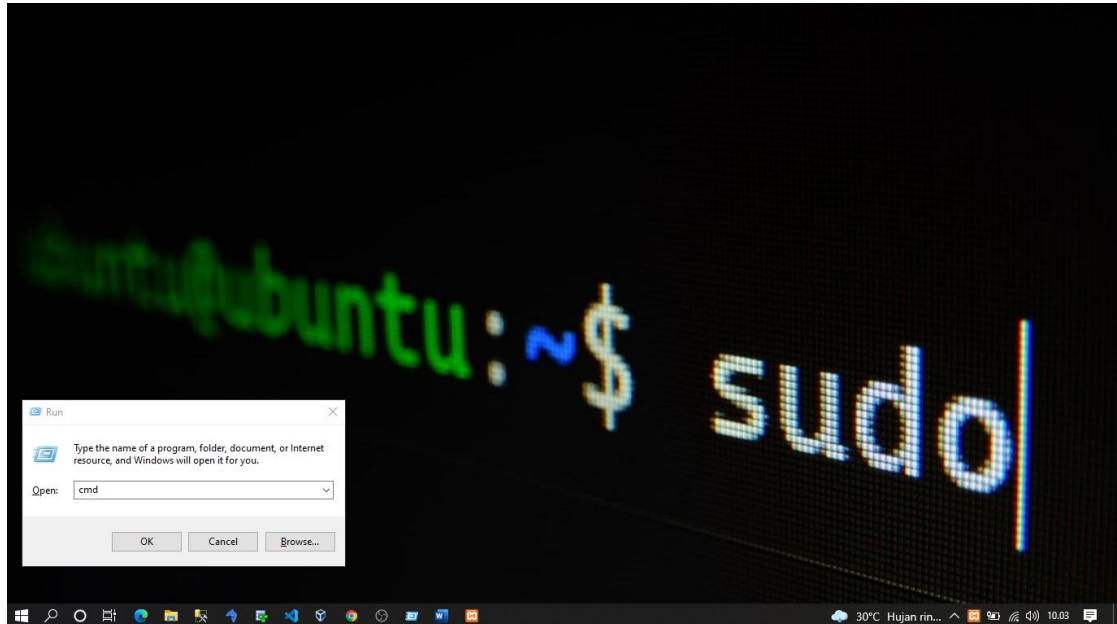
### Step #1 Design RESTful API

Mendefinisikan dulu EndPoint dari RESTful API yang akan dibuat. EndPoint merupakan routes dari API yang akan kita buat. RESTful API menggunakan HTTP verbs. HTTP verbs yang umum digunakan adalah GET, POST, PUT, dan DELETE. GET untuk mendapatkan data dari server atau lebih dikenal dengan istilah READ, POST untuk meng-CREATE new data, PUT untuk UPDATE data, dan DELETE untuk menghapus data. Atau lebih dikenal dengan istilah CRUD (Create Read Update Deletedari suatu table di database yaitu table products.

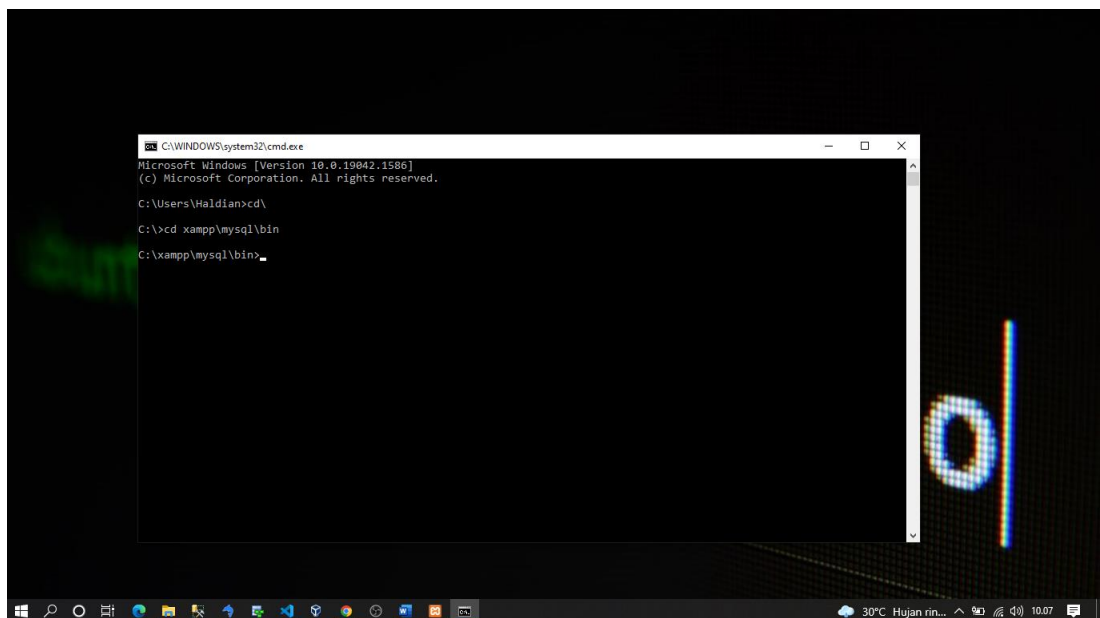
Method	EndPoint	Description
GET	api/products	List of products
GET	api/products/{id}	View a product
POST	api/products	Create new product
PUT	api/products/{id}	Update a product
DELETE	api/products/{id}	Delete a product

## Step #2. Buat Database dan Table

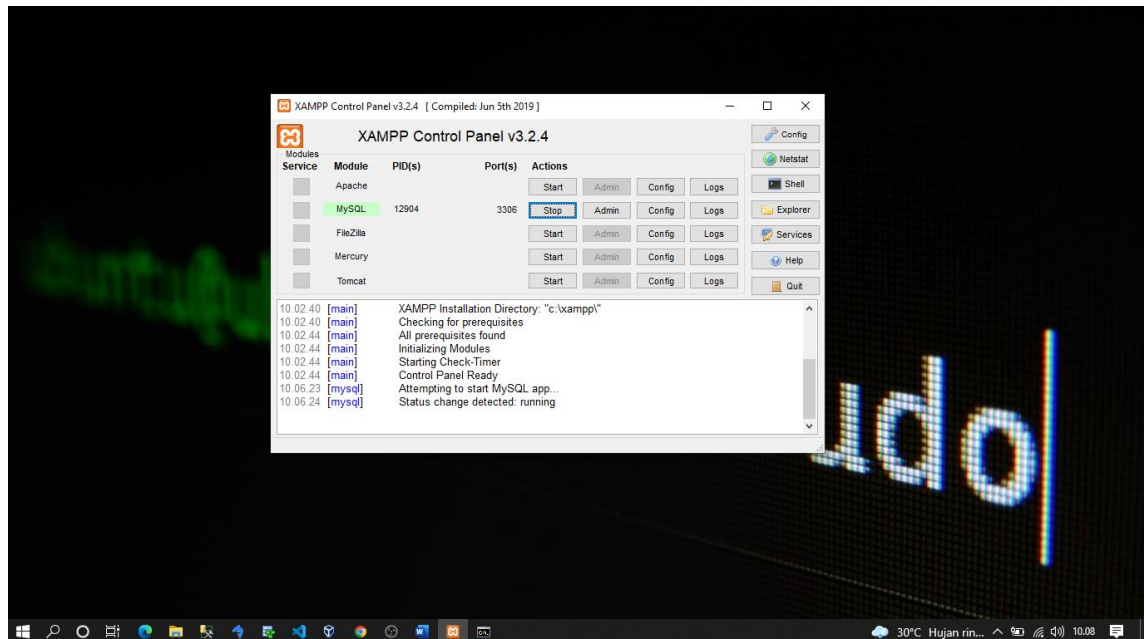
1. Kita buka cmd kita dengan win + r maka akan seperti gambar di bawah ini



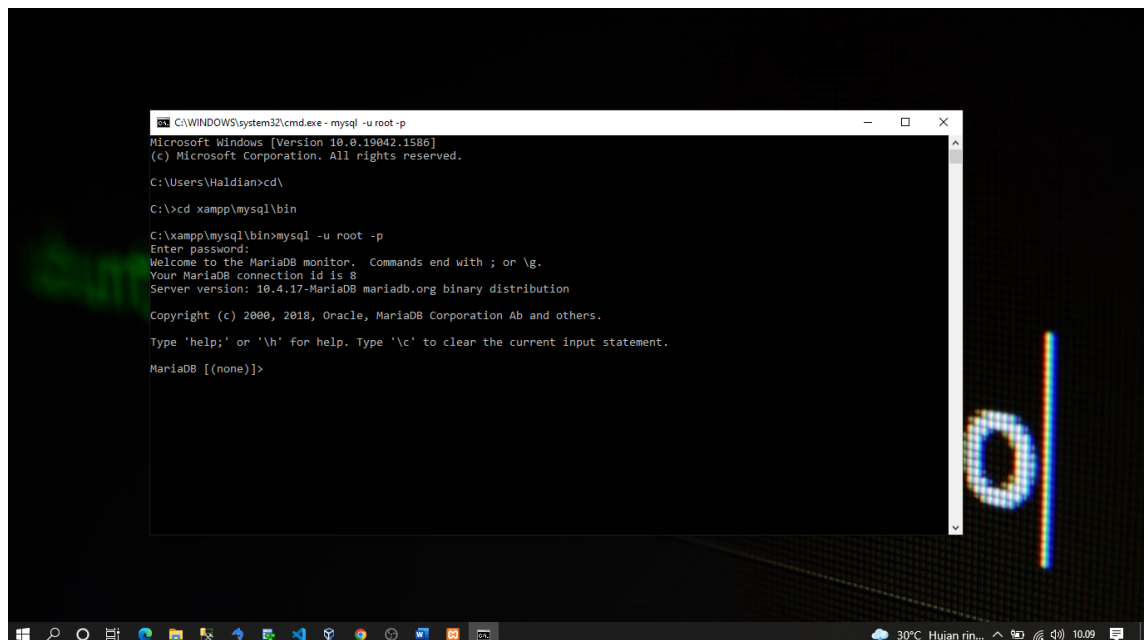
2. Kemudian kita masuk ke cd xampp\mysql\bin seperti gambar di bawah ini



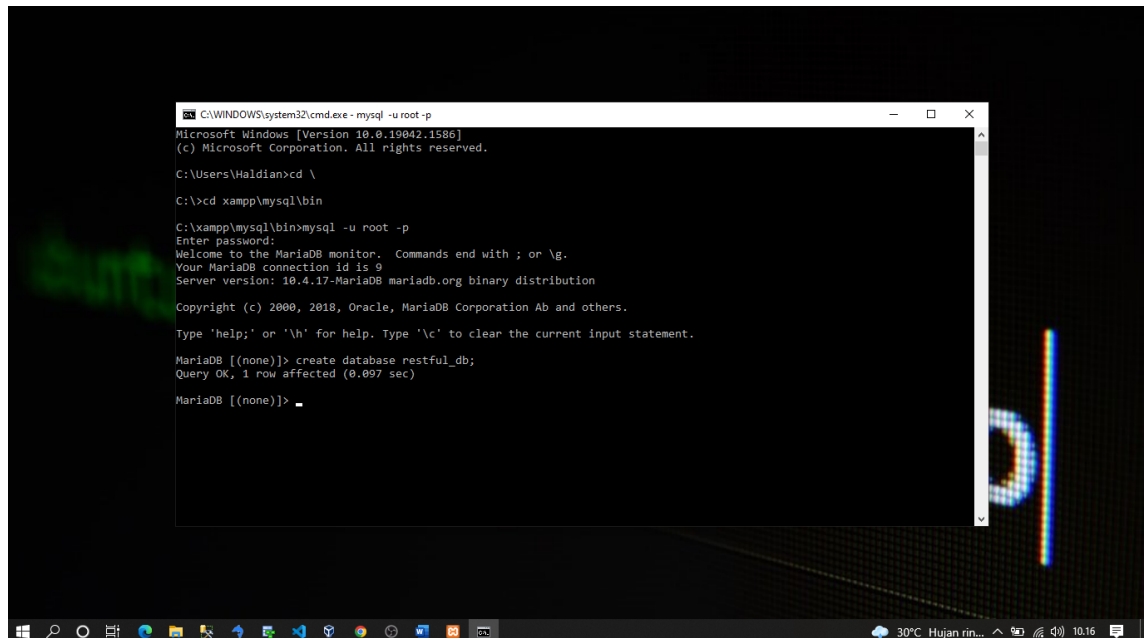
3. Selanjutnya kita buka xampp kita dan kita hidupkan mysql servernya seperti gambar di bawah ini



4. Untuk mengakses mysql server kita kita ketikkan mysql -u root -p kemudian enter pada console kita seperti gambar di bawah ini



5. kemudian membuat database dengan cara create database restful\_db seperti gambar di bawah ini



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Haldian>cd \
C:\>cd xampp\mysql\bin
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.4.17-MariaDB mariadb.org binary distribution

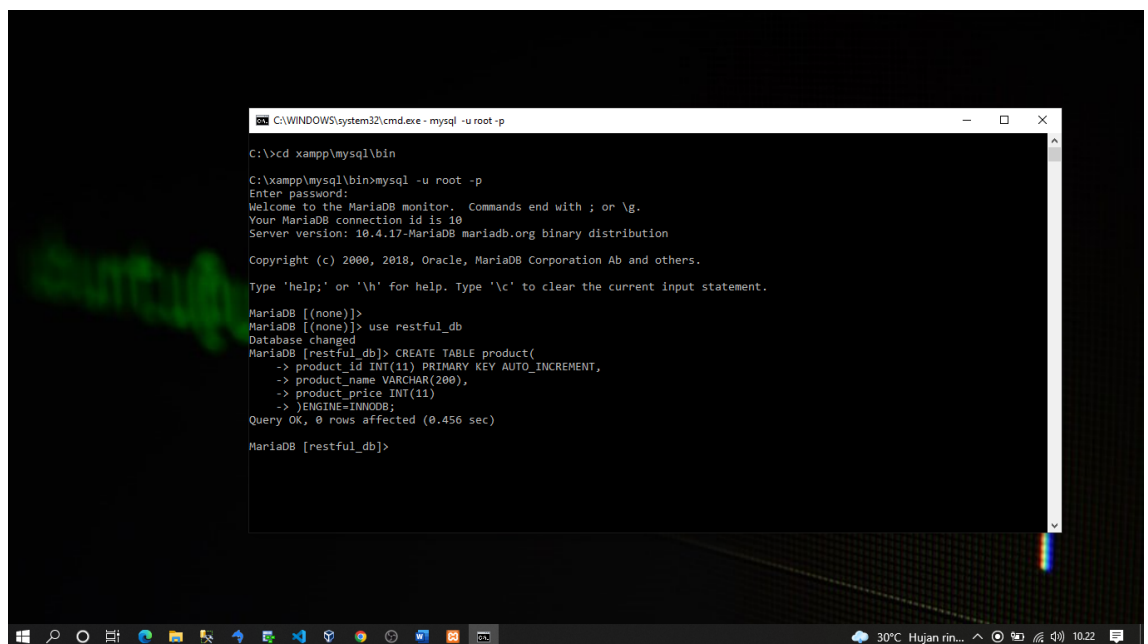
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database restful_db;
Query OK, 1 row affected (0.097 sec)

MariaDB [(none)]>
```

6. Selanjutnya kita create table product seperti gambar di bawah ini



```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
C:\>cd xampp\mysql\bin
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 10
Server version: 10.4.17-MariaDB mariadb.org binary distribution

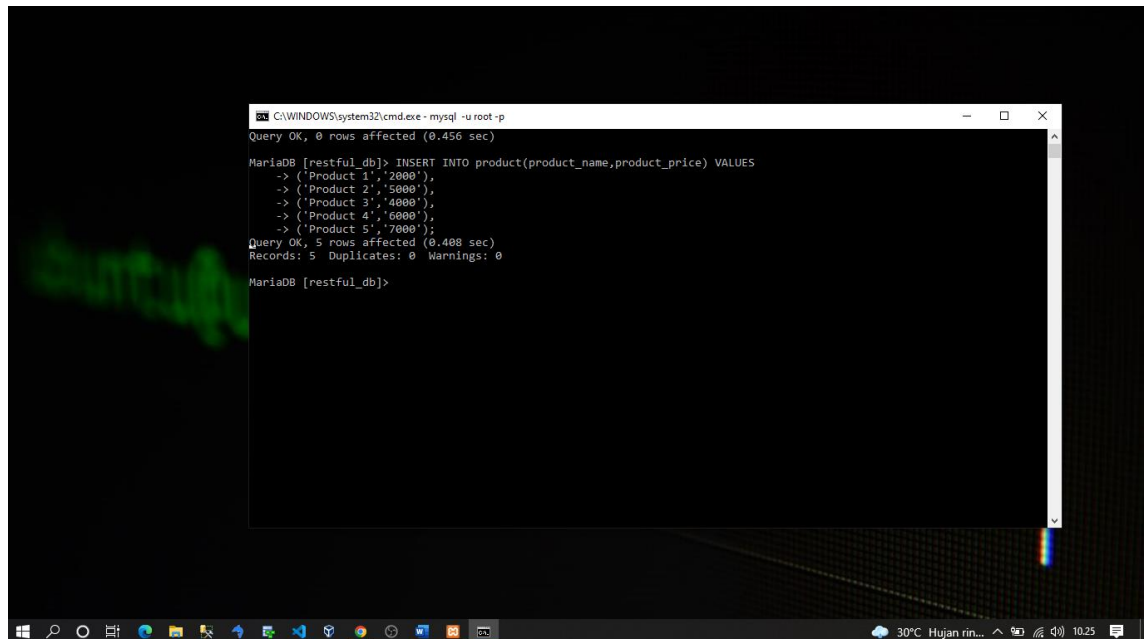
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
MariaDB [(none)]> use restful_db
Database changed
MariaDB [restful_db]> CREATE TABLE product(
-> product_id INT(11) PRIMARY KEY AUTO_INCREMENT,
-> product_name VARCHAR(200),
-> product_price INT(11)
-> )ENGINE=INNODB;
Query OK, 0 rows affected (0.456 sec)

MariaDB [restful_db]>
```

7. Selanjutnya kita insert data kedalam table product dengan mengeksekusi query dimana akan menginsert 5 data pada table product seperti gambar di bawah ini



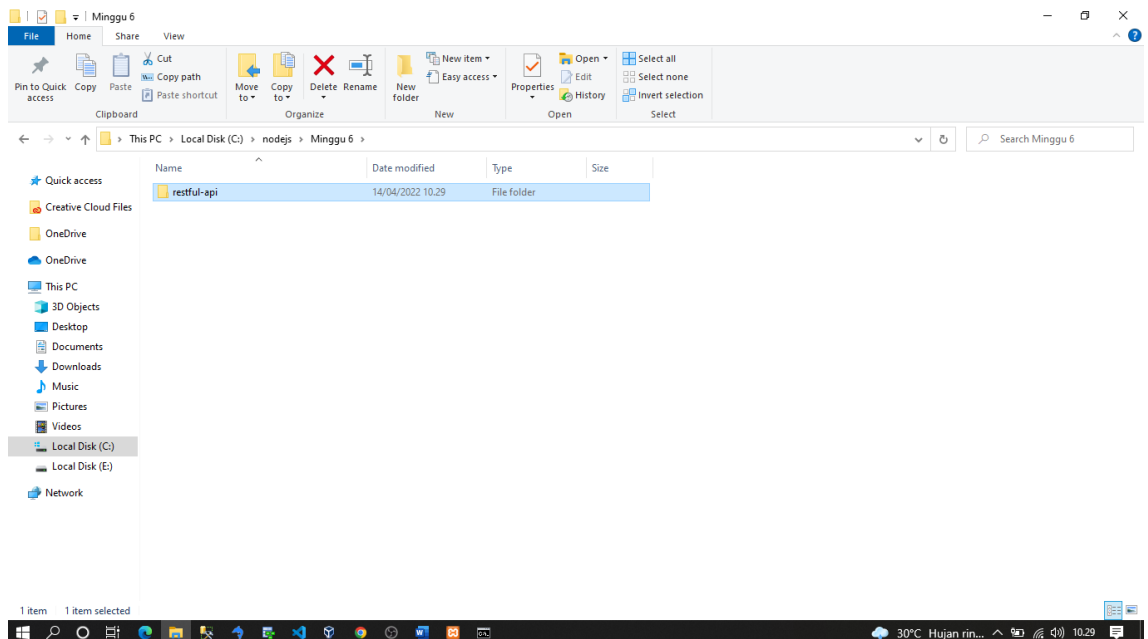
```
C:\WINDOWS\system32\cmd.exe - mysql -u root -p
Query OK, 0 rows affected (0.456 sec)

MariaDB [restful_db]> INSERT INTO product(product_name,product_price) VALUES
-> ('Product 1','2000'),
-> ('Product 2','5000'),
-> ('Product 3','4000'),
-> ('Product 4','6000'),
-> ('Product 5','7000');
Query OK, 5 rows affected (0.408 sec)
Records: 5 Duplicates: 0 Warnings: 0

MariaDB [restful_db]>
```

### Step #3. Install Dependencies

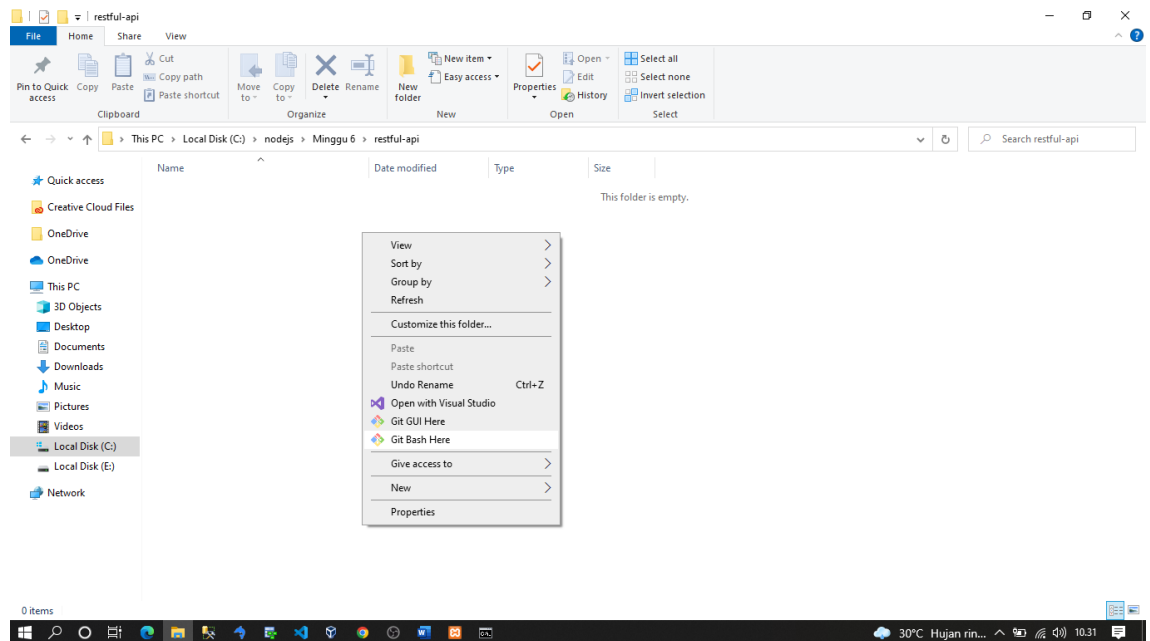
1. Sebelum menginstall dependencies kita perlu membuat sebuah folder dengan nama restful-api seperti gambar di bawah ini.



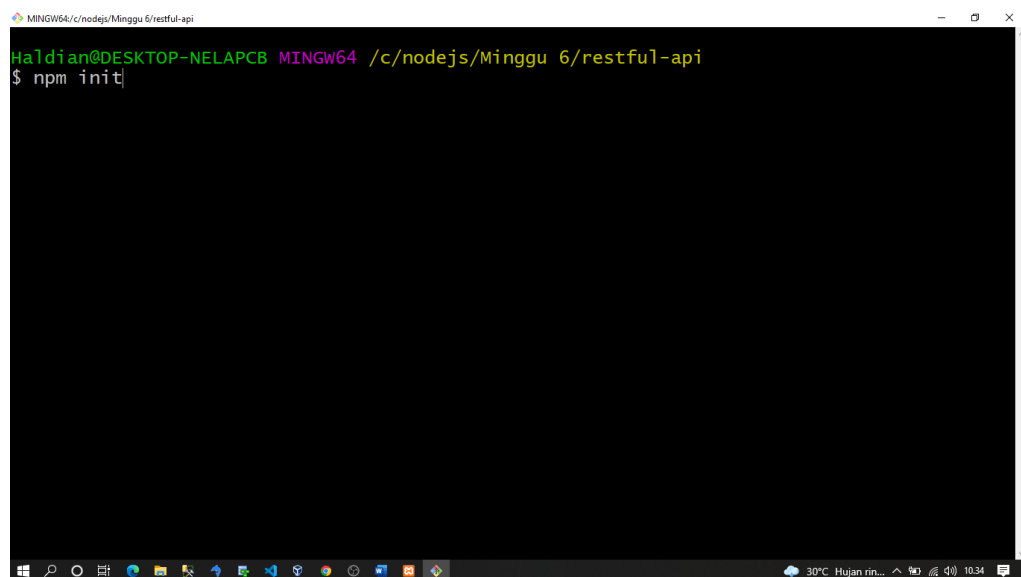
kita membutuhkan 3 dependencies yaitu:

1. Express (node.js framework)
2. MySQL (driver mysql untuk node.js)
3. Body-parser (middleware untuk handle post body request)

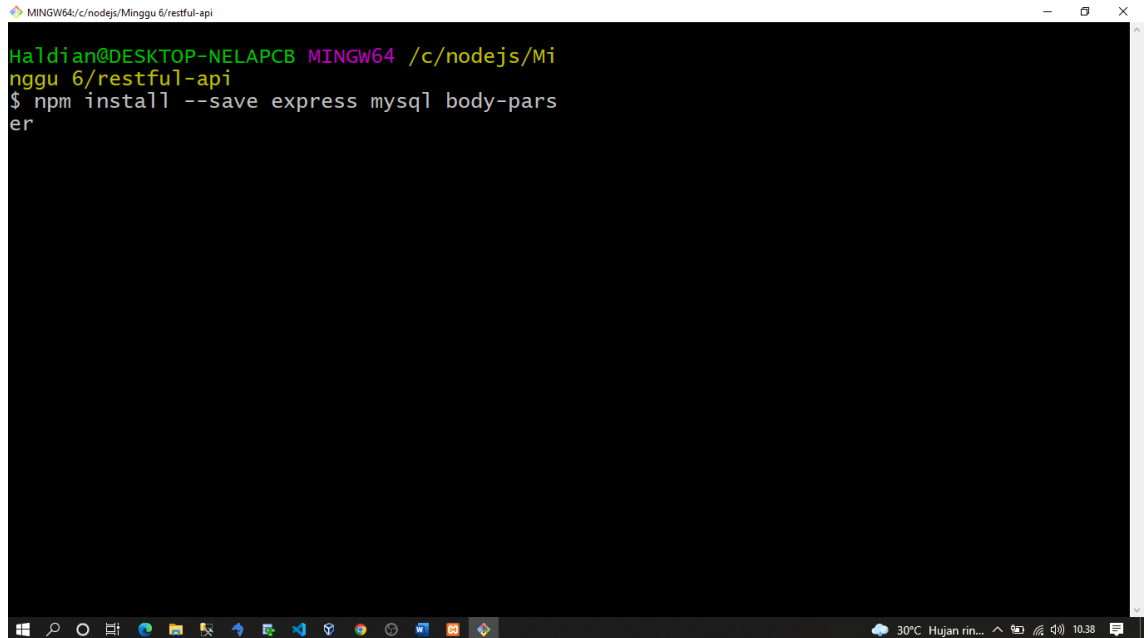
2. Kemudian kita masuk pada folder restful-api lalu klik kanan dan pilih Gitbash here seperti gambar di bawah ini



3. Kemudian kita perlu membuat package.json dengan otomatis pada gitbash kita dengan mengetikkan `npm init` seperti gambar di bawah ini

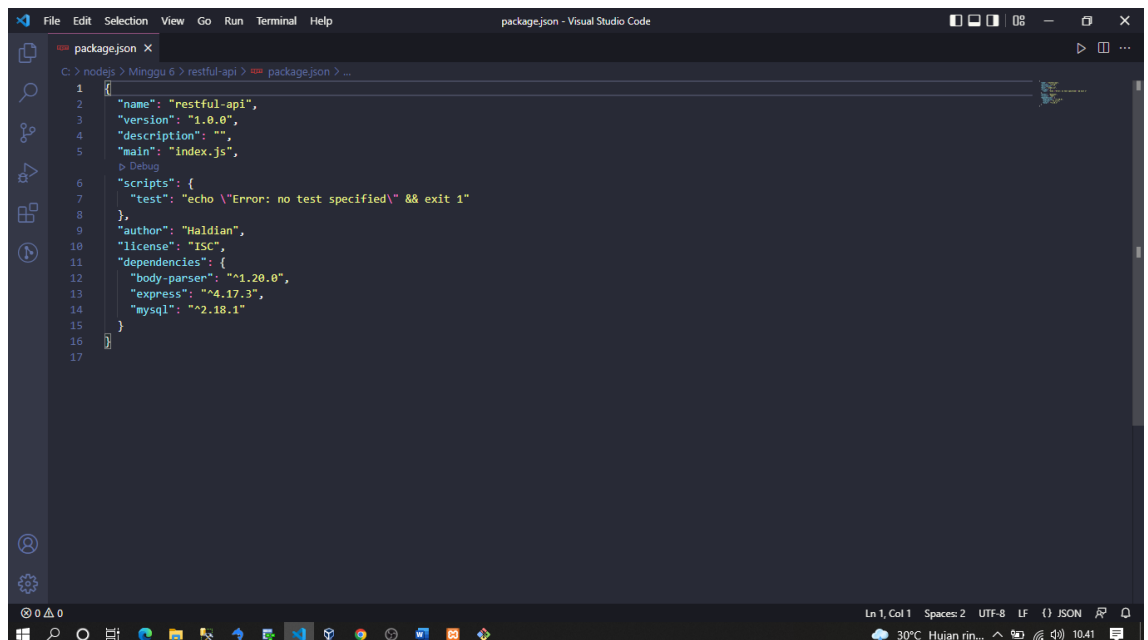


4. Selanjutnya kita Install semua dependencies dengan perintah `npm install --save express mysql body-parser` yang dibutuhkan dengan mengetikkan perintah seperti gambar di bawah ini



```
MINGW64/c:/nodejs/Minggu 6/restful-api
Haldian@DESKTOP-NELAPCB MINGW64 /c:/nodejs/Minggu 6/restful-api
$ npm install --save express mysql body-parser
```

5. Jika file `package.json` di buka maka kan seperti gambar di bawah ini

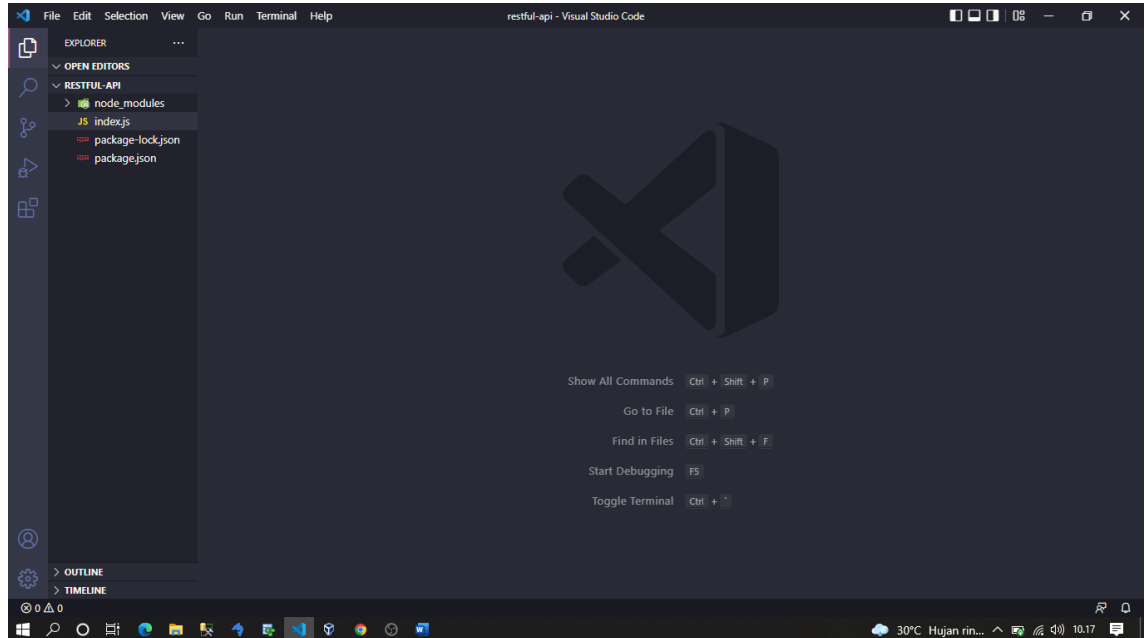


```
package.json
1 {
2   "name": "restful-api",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "author": "Haldian",
10  "license": "ISC",
11  "dependencies": {
12    "body-parser": "^1.20.0",
13    "express": "^4.17.3",
14    "mysql": "^2.18.1"
15  }
16 }
17
```

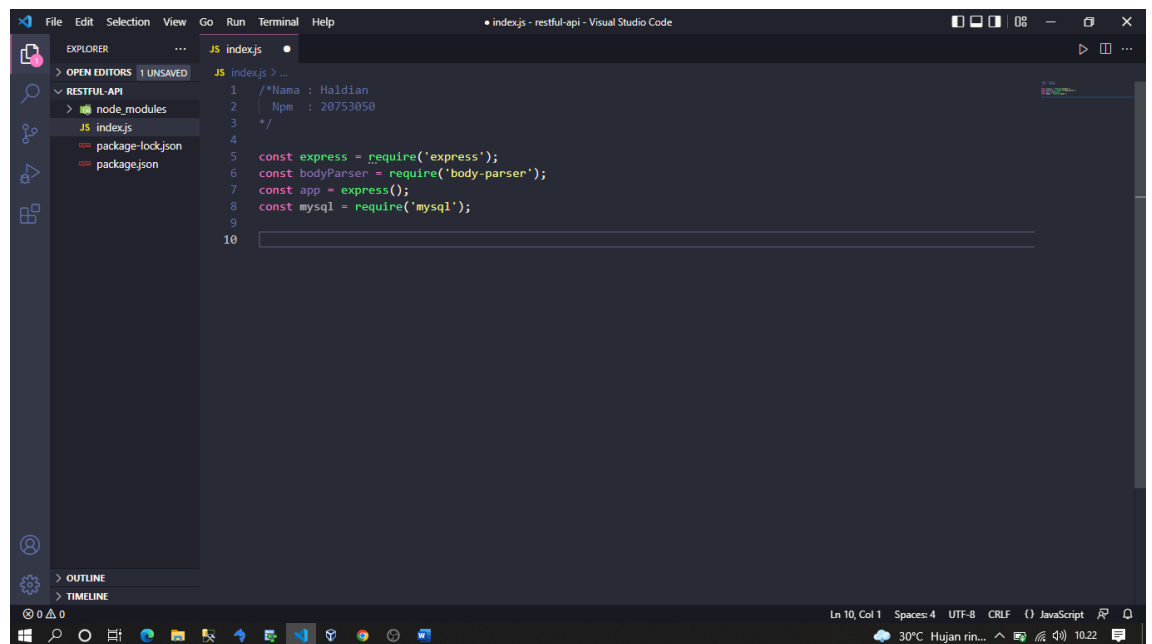


## Step #4. Buat file Index.js

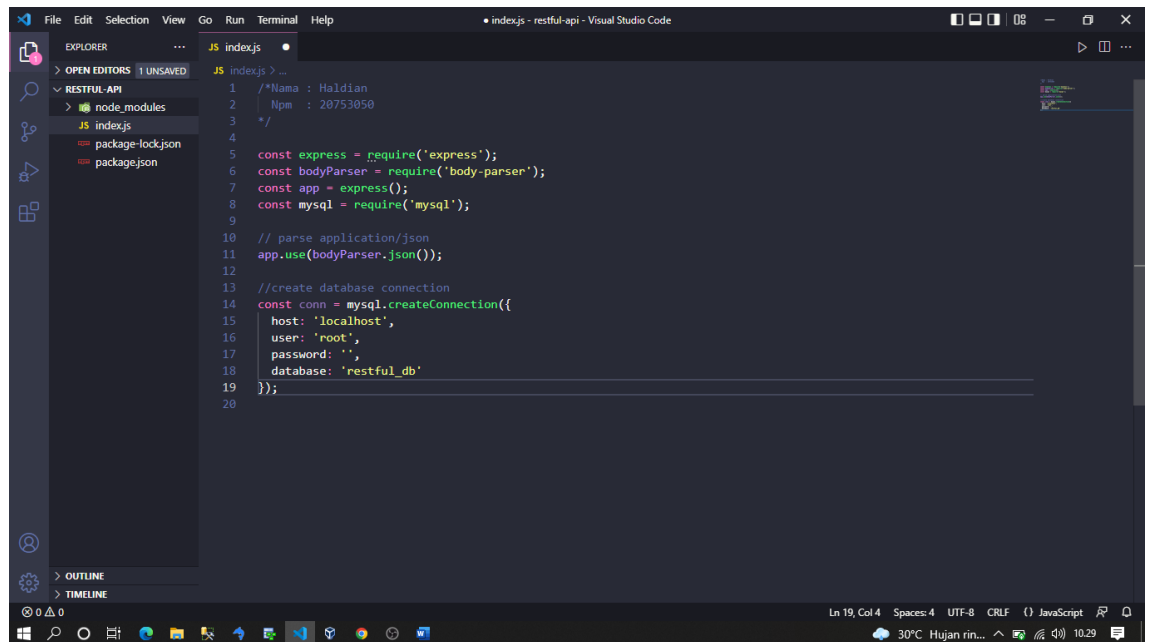
1. Pertama kita buat sebuah file dengan nama indeks seperti gambar di bawah ini



2. Kemudian kita ketikkan script pada file index seperti gambar di bawah ini
  - a. Yang pertama dimana kita memerlukan file kita dengan semua dependencies yang telah kita install sebelumnya seperti gambar di bawah ini

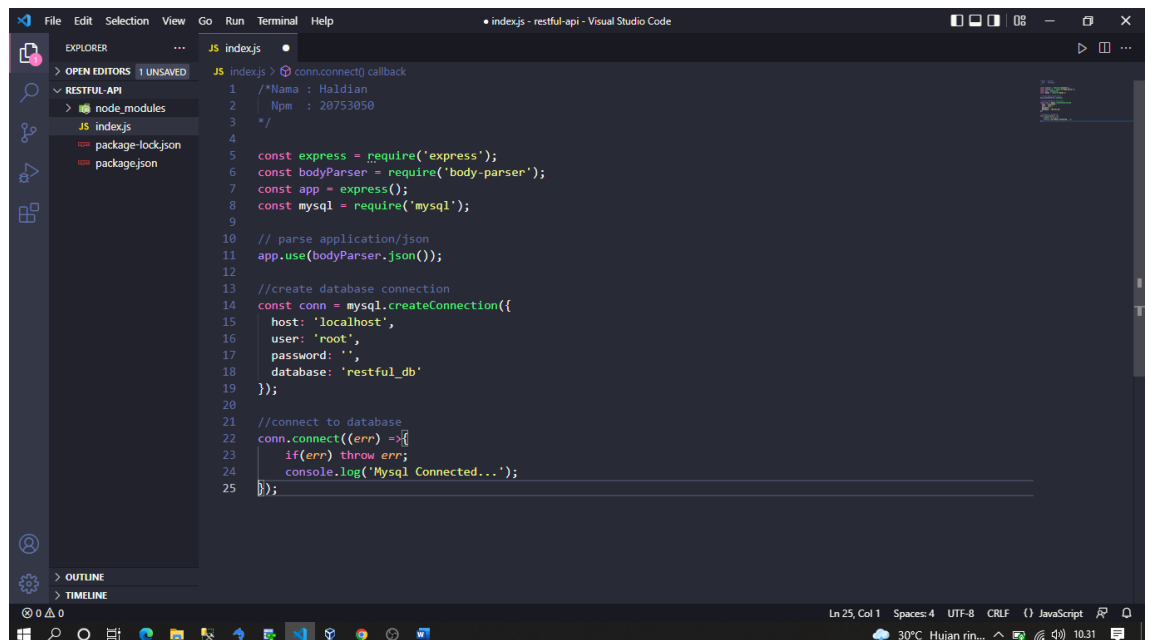


- b. Kemudian kita buat script untuk uraikan json dengan `app.use(bodyParser.json())` serta kita lakukan pembuatan script database untuk koneksi seperti gambar di bawah ini



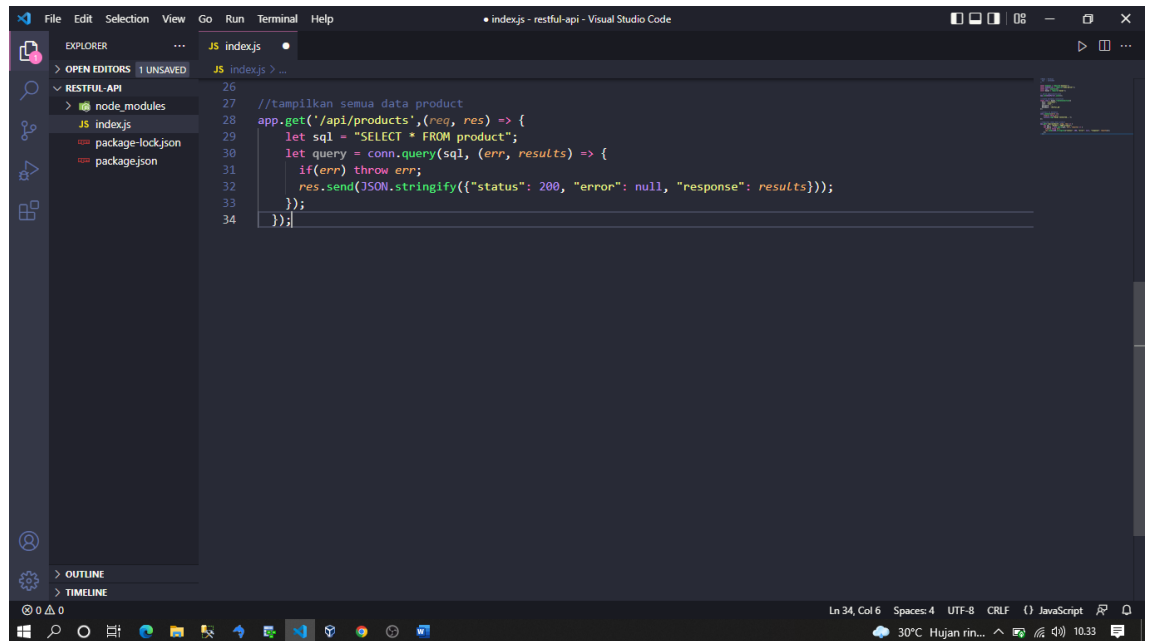
```
1  /*Nama : Haldian
2  Npm : 20753050
3  */
4
5  const express = require('express');
6  const bodyParser = require('body-parser');
7  const app = express();
8  const mysql = require('mysql');
9
10 // parse application/json
11 app.use(bodyParser.json());
12
13 //create database connection
14 const conn = mysql.createConnection({
15   host: 'localhost',
16   user: 'root',
17   password: '',
18   database: 'restful_db'
19 });
```

- c. Kemudian buat script untuk konek ke database seperti gambar di bawah ini



```
1  /*Nama : Haldian
2  Npm : 20753050
3  */
4
5  const express = require('express');
6  const bodyParser = require('body-parser');
7  const app = express();
8  const mysql = require('mysql');
9
10 // parse application/json
11 app.use(bodyParser.json());
12
13 //create database connection
14 const conn = mysql.createConnection({
15   host: 'localhost',
16   user: 'root',
17   password: '',
18   database: 'restful_db'
19 });
20
21 //connect to database
22 conn.connect((err) =>{
23   if(err) throw err;
24   console.log('Mysql Connected...');
25 });
```

- d. Kemudian kita buat script query untuk mengambil semua data product pada database kita dengan api seperti gambar di bawah ini

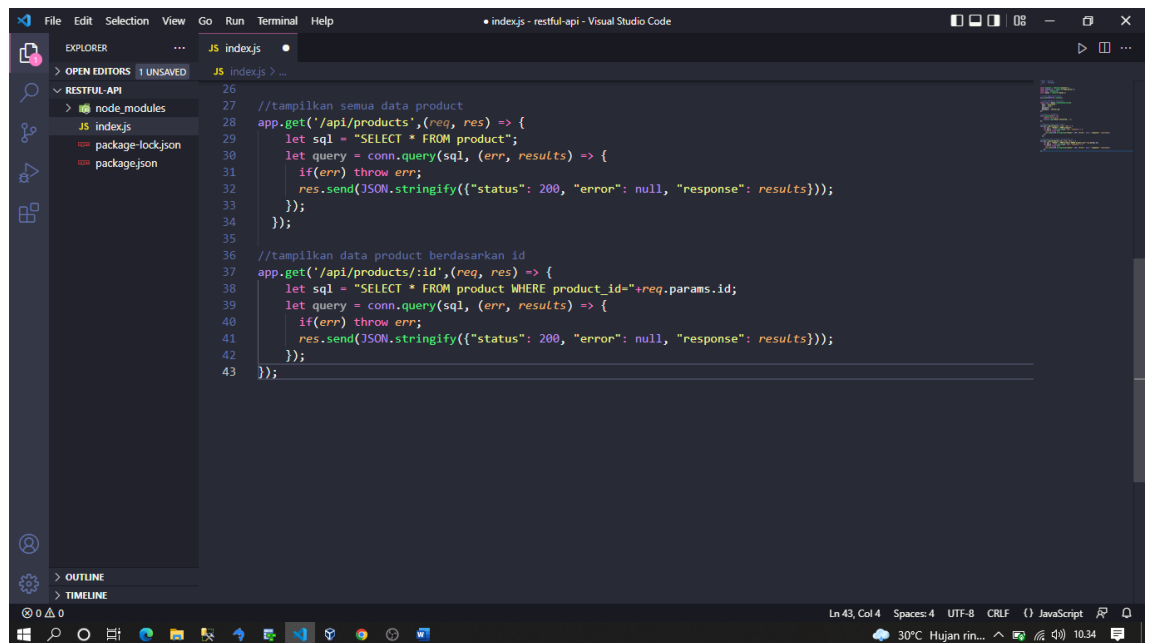


The screenshot shows the Visual Studio Code interface with a RESTful API client. The Explorer sidebar on the left shows the project structure with 'RESTFUL-API' and 'index.js' selected. The main editor displays the following JavaScript code:

```
26
27 //tampilkan semua data product
28 app.get('/api/products',(req, res) => {
29   let sql = "SELECT * FROM product";
30   let query = conn.query(sql, (err, results) => {
31     if(err) throw err;
32     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
33   });
34 });
```

The status bar at the bottom indicates the file is 'index.js' in the 'restful-api' project, using UTF-8 encoding and CRLF line endings.

- e. Kemudian kita buat script query untuk menampilkan semua data product berdasarkan ide pada database kita dengan api seperti gambar di bawah ini

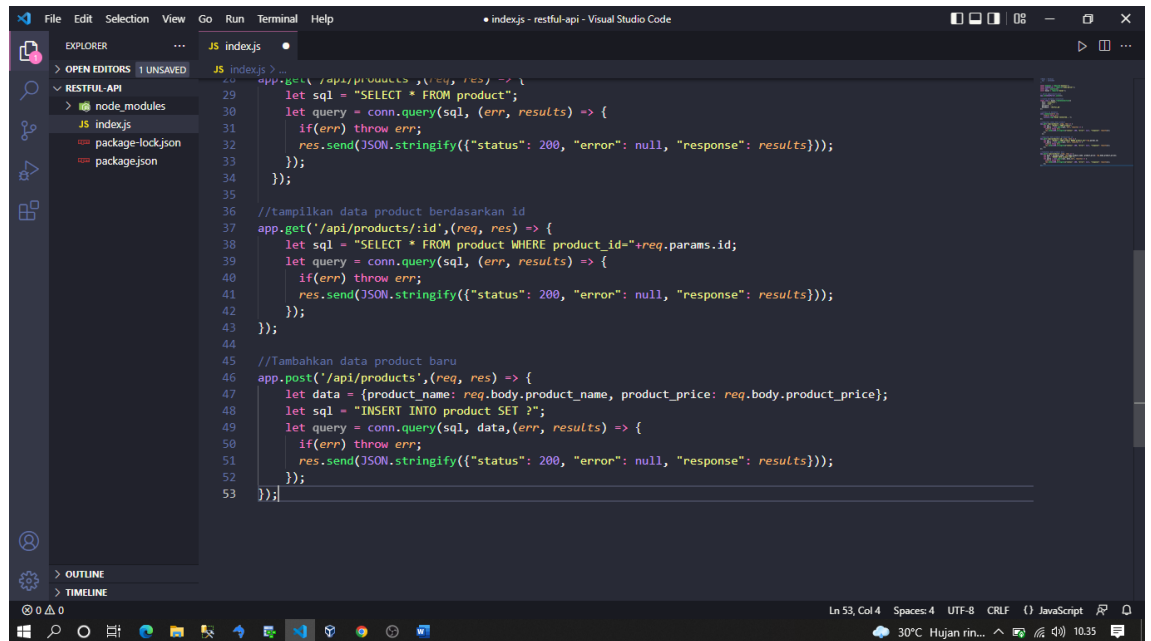


The screenshot shows the Visual Studio Code interface with a RESTful API client. The Explorer sidebar on the left shows the project structure with 'RESTFUL-API' and 'index.js' selected. The main editor displays the following JavaScript code:

```
26
27 //tampilkan semua data product
28 app.get('/api/products',(req, res) => {
29   let sql = "SELECT * FROM product";
30   let query = conn.query(sql, (err, results) => {
31     if(err) throw err;
32     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
33   });
34 });
35
36 //tampilkan data product berdasarkan id
37 app.get('/api/products/:id',(req, res) => {
38   let sql = "SELECT * FROM product WHERE product_id="+req.params.id;
39   let query = conn.query(sql, (err, results) => {
40     if(err) throw err;
41     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
42   });
43 });
```

The status bar at the bottom indicates the file is 'index.js' in the 'restful-api' project, using UTF-8 encoding and CRLF line endings.

- f. Kemudian kita buat script untuk menambah data product baru ke dalam database kita dengan api seperti gambar di bawah ini

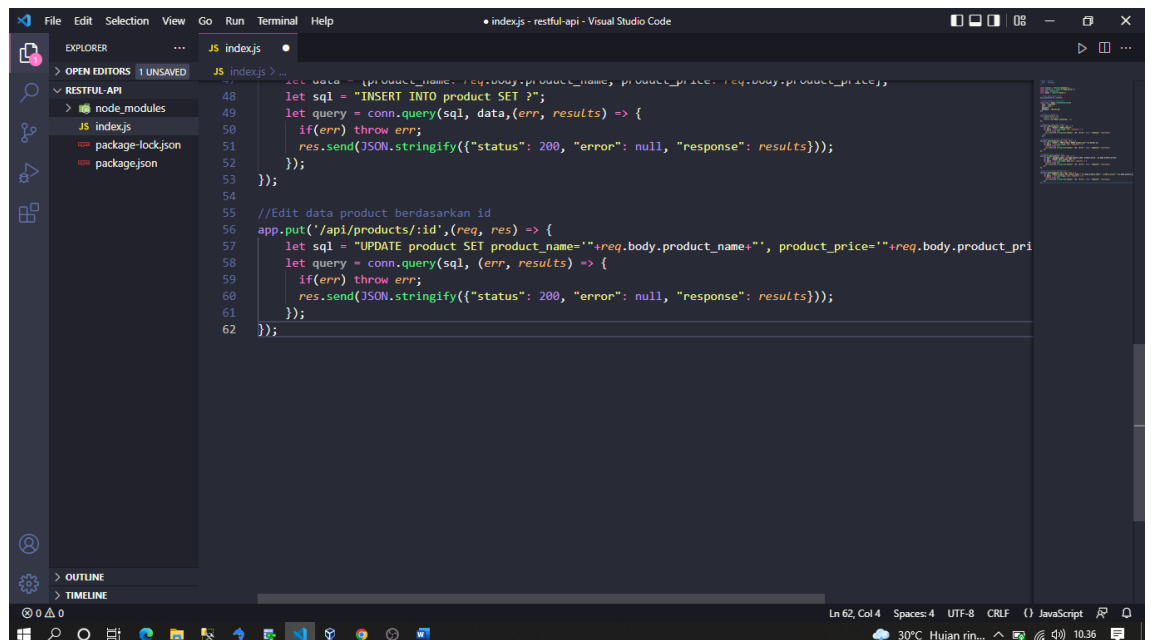


```
File Edit Selection View Go Run Terminal Help
indexjs - restful-api - Visual Studio Code

EXPLORER
> OPEN EDITORS 1 UNSAVED
  RESTFUL-API
  JS indexjs
  package-lock.json
  package.json

JS indexjs
28 app.get('/api/products', (req, res) => {
29   let sql = "SELECT * FROM product";
30   let query = conn.query(sql, (err, results) => {
31     if(err) throw err;
32     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
33   });
34 });
35
36 //tampilkan data product berdasarkan id
37 app.get('/api/products/:id', (req, res) => {
38   let sql = "SELECT * FROM product WHERE product_id="+req.params.id;
39   let query = conn.query(sql, (err, results) => {
40     if(err) throw err;
41     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
42   });
43 });
44
45 //Tambahkan data product baru
46 app.post('/api/products', (req, res) => {
47   let data = {product_name: req.body.product_name, product_price: req.body.product_price};
48   let sql = "INSERT INTO product SET ?";
49   let query = conn.query(sql, data, (err, results) => {
50     if(err) throw err;
51     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
52   });
53 });
```

- g. Kemudian kita buat script untuk mengedit data product berdasarkan ide pada database kita dengan api seperti gambar di bawah ini

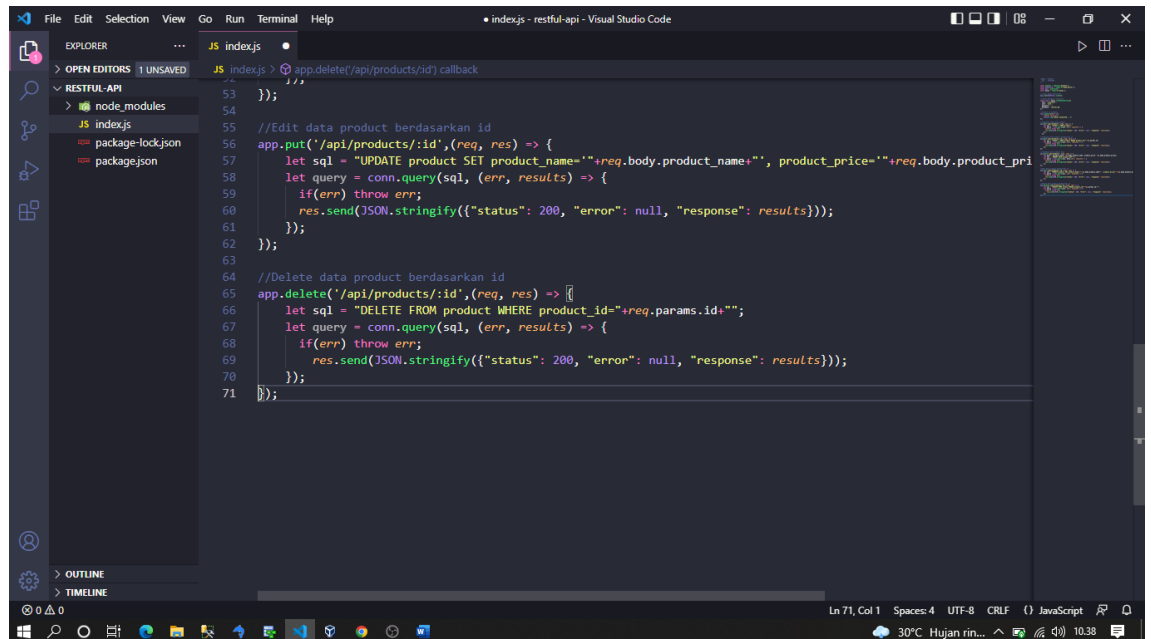


```
File Edit Selection View Go Run Terminal Help
indexjs - restful-api - Visual Studio Code

EXPLORER
> OPEN EDITORS 1 UNSAVED
  RESTFUL-API
  JS indexjs
  package-lock.json
  package.json

JS indexjs
47 let data = {product_name: req.body.product_name, product_price: req.body.product_price};
48 let sql = "INSERT INTO product SET ?";
49 let query = conn.query(sql, data, (err, results) => {
50   if(err) throw err;
51   res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
52 });
53
54
55 //Edit data product berdasarkan id
56 app.put('/api/products/:id', (req, res) => {
57   let sql = "UPDATE product SET product_name="+req.body.product_name+", product_price="+req.body.product_price;
58   let query = conn.query(sql, (err, results) => {
59     if(err) throw err;
60     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
61   });
62 });
```

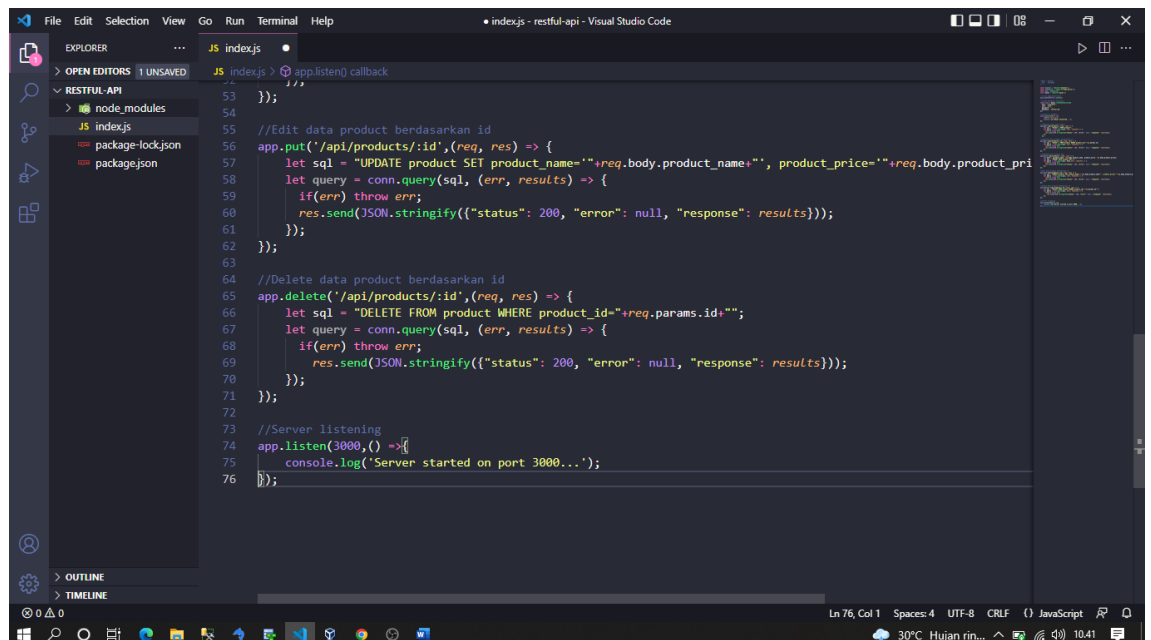
- h. Kemudian kita buat script untuk delete data product berdasarkan ide pada database kita dengan api seperti gambar di bawah ini



The screenshot shows the Visual Studio Code editor with a file named `index.js` open. The code implements a RESTful API with two endpoints: `PUT` for updating a product and `DELETE` for deleting a product. The `DELETE` endpoint is highlighted, showing a function that takes a request and response object, constructs a SQL query to delete a product by ID, and returns a JSON response with status 200 and the results. The Explorer sidebar on the left shows the project structure with `RESTFUL-API` and `node_modules` folders. The status bar at the bottom indicates the file is at line 71, column 1, using UTF-8 encoding and CRLF line endings.

```
53 // app.delete('/api/products/:id') callback
54
55 //Edit data product berdasarkan id
56 app.put('/api/products/:id',(req, res) => {
57   let sql = "UPDATE product SET product_name='"+req.body.product_name+"', product_price='"+req.body.product_price+"';"
58   let query = conn.query(sql, (err, results) => {
59     if(err) throw err;
60     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
61   });
62 });
63
64 //Delete data product berdasarkan id
65 app.delete('/api/products/:id',(req, res) => {
66   let sql = "DELETE FROM product WHERE product_id='"+req.params.id+"';"
67   let query = conn.query(sql, (err, results) => {
68     if(err) throw err;
69     res.send(JSON.stringify({"status": 200, "error": null, "response": results}));
70   });
71 });
```

- i. Selanjutnya kita buat script untuk server local untuk menjalankan file kita seperti gambar di bawah ini

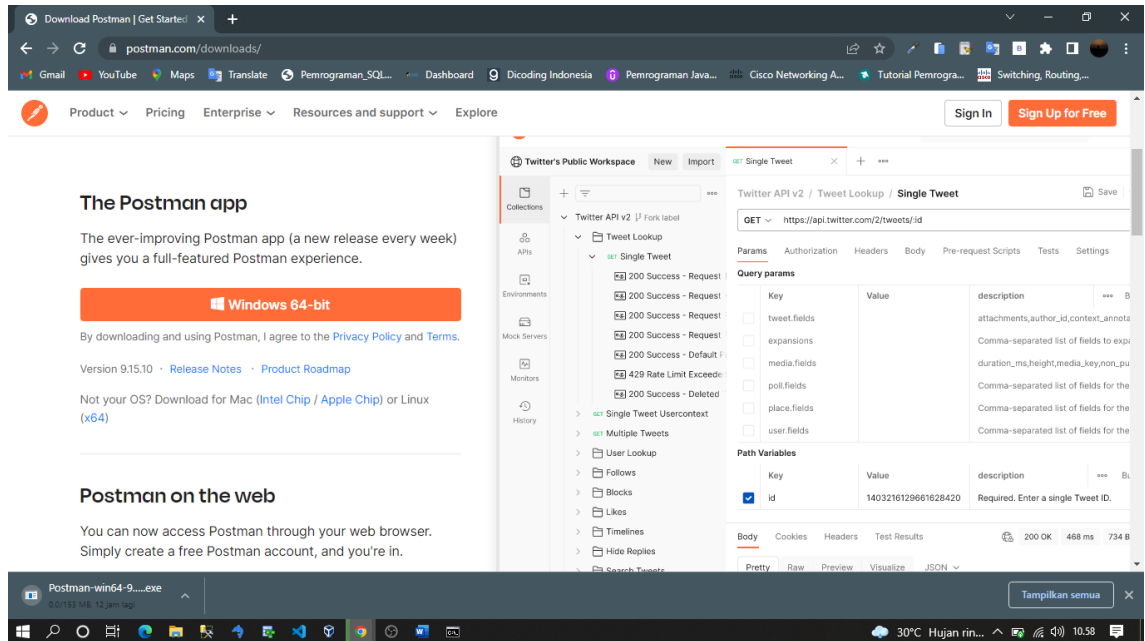


This screenshot shows the same `index.js` file with the `app.listen()` method added at the bottom. The code now includes a `listen` function that starts the server on port 3000 and logs a message to the console. The Explorer sidebar and status bar are also visible, showing the file is now at line 76, column 1.

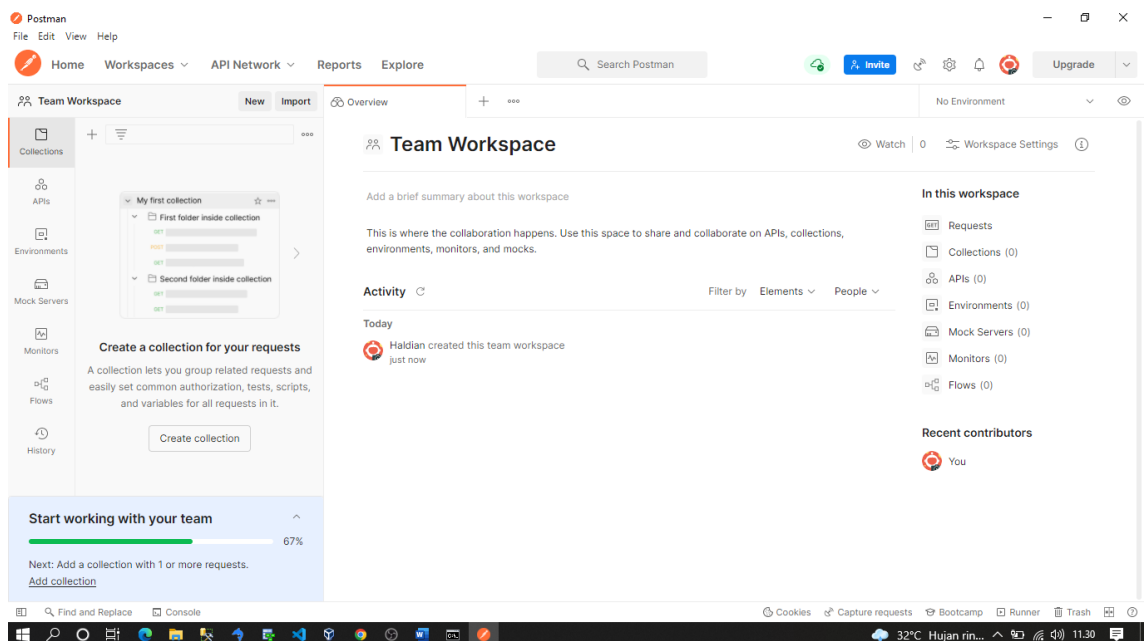
```
73
74 //Server listening
75 app.listen(3000,() =>{
76   console.log('Server started on port 3000...');
77 });
```

## Step #5. Testing

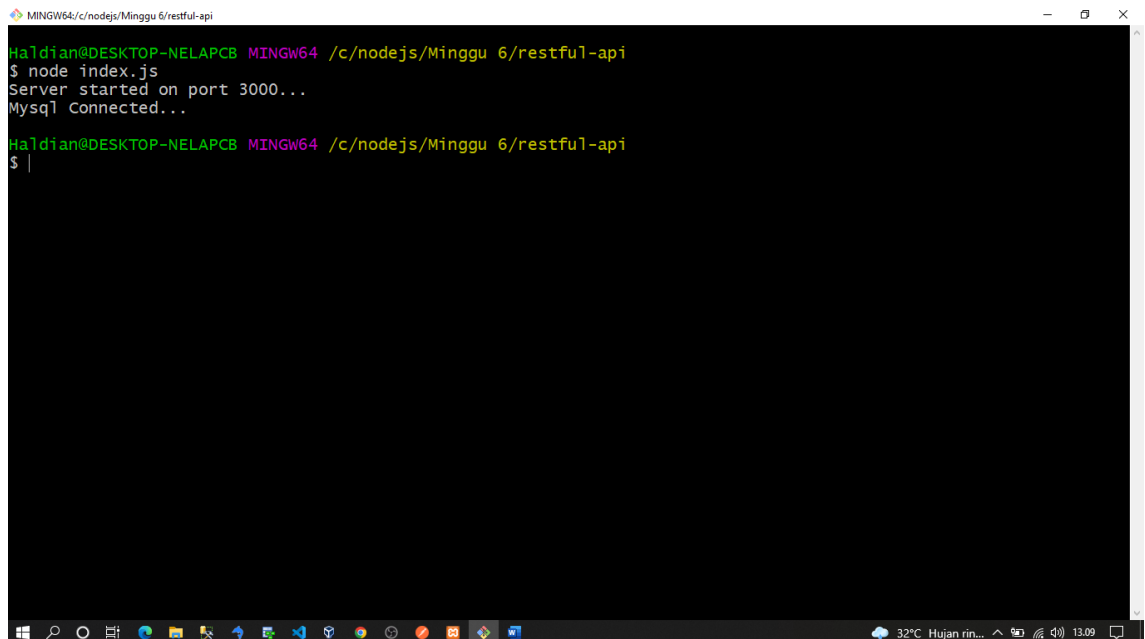
1. Untuk melakukan uji coba API yang telah kita buat yang pertama yang harus kita lakukan yaitu mendownload POSTMAN pada situs resminya yaitu <https://www.getpostman.com/> kemudian kita download seperti gambar di bawah ini



2. Setelah kita download dan install maka ini dia tampilan dari POSTMAN kita seperti gambar di bawah ini



3. Selanjutnya kita coba run di gitbash kita dengan cara node index.js jika seperti gambar di bawah ini



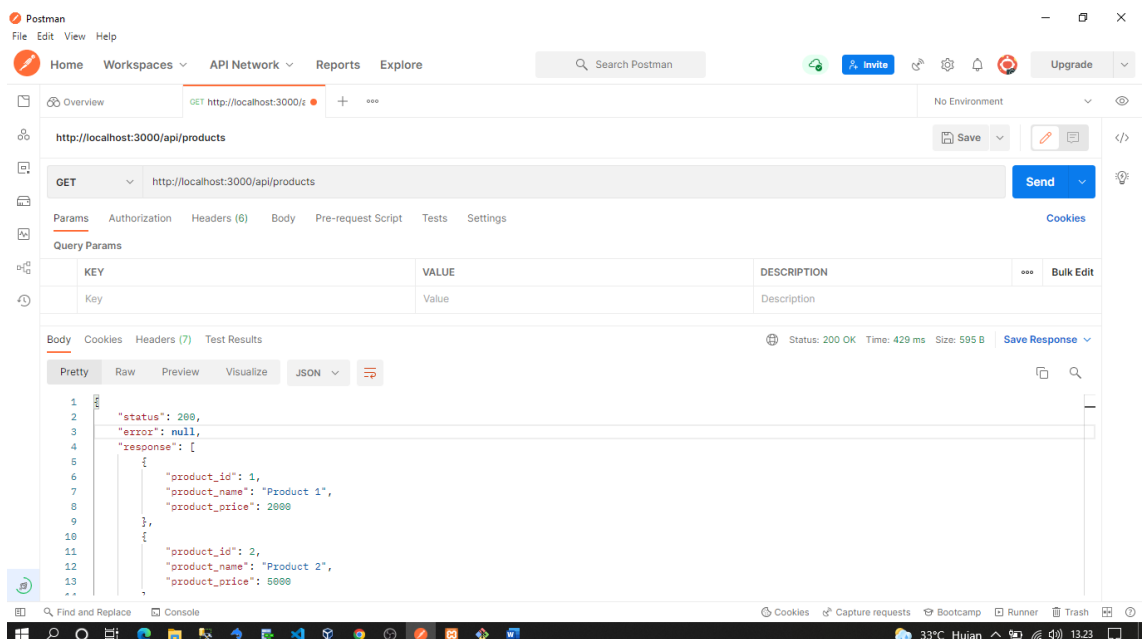
```
MINGW64/c:/nodejs/Minggu 6/restful-api
Haldian@DESKTOP-NELAPCB MINGW64 /c:/nodejs/Minggu 6/restful-api
$ node index.js
Server started on port 3000...
Mysql Connected...

Haldian@DESKTOP-NELAPCB MINGW64 /c:/nodejs/Minggu 6/restful-api
$ |
```

4. selanjutnya kita uji EndPoint-nya satu per satu.

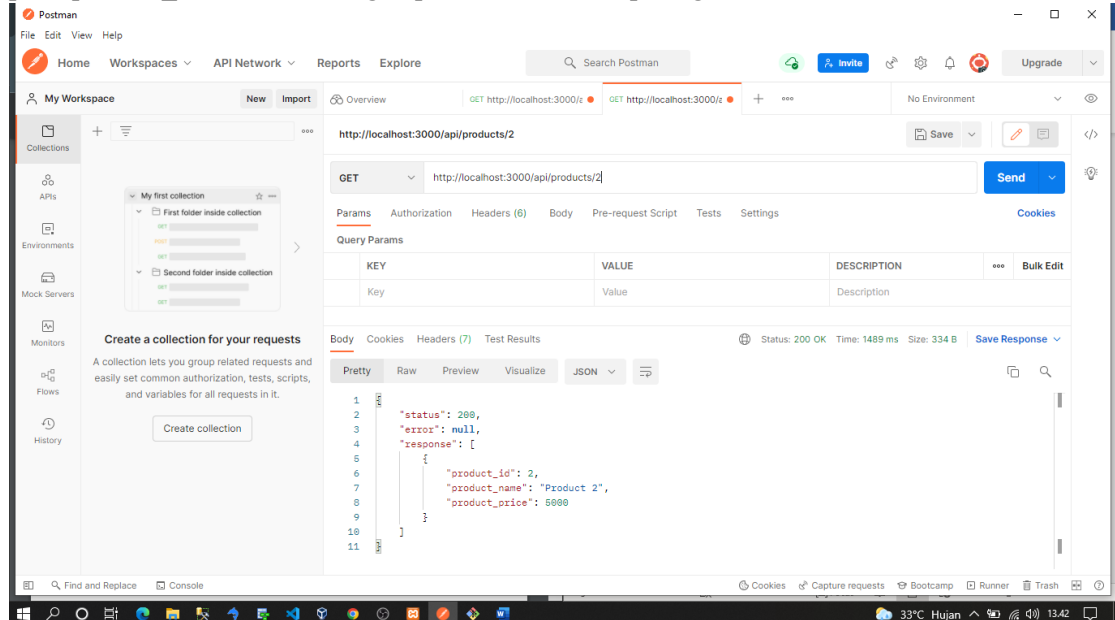
### #1. Get All Product (GET)

Pertama kita buka POSTMAN kita kemudian kita ketikan URL <http://localhost:3000/api/products> kemudian kita pilih Method GET, dan klik tombol Send kemudian kita pilih JSON, jika seperti gambar di bawah maka data berjalan dengan baik seperti gambar di bawah ini.



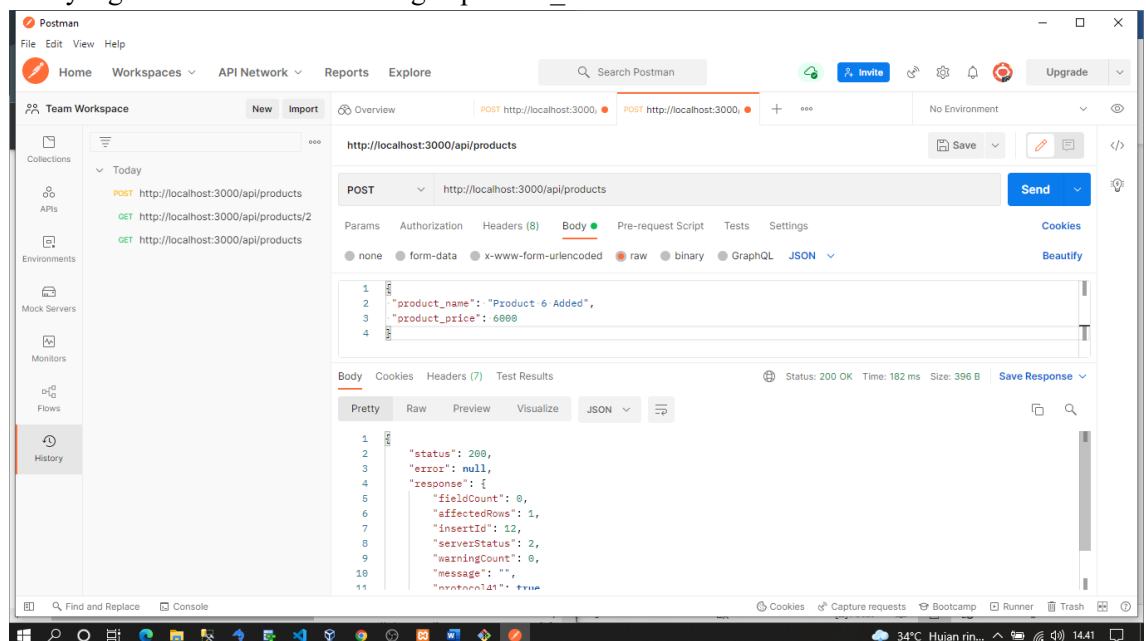
## #2. Get Single Product (GET)

Ketika URL <http://localhost:3000/api/products/2> masukan pada POSTMAN dan kita pilih Methodm GET kemudian kita Send maka terlihat data product yang ditampilkan pada product\_id='2' sesuai dengan parameter URL seperti gambar di bawah ini.



## #3. Create New Product (POST)

Selanjutnya kita ketikkan URL <http://localhost:3000/api/product> pada Method POST kemudian kita masukan data `{ "product_name": "Product 6 Added", "product_price": 6000 }` pada kolom JSON(application/json) kemudian kita Send, jika kita perhatikan pada bagian response, terdapat “affectedRows”: 1, dan “insertId”: 6. Itu berarti terdapat satu data yang diinsert ke database dengan product\_id='6’.

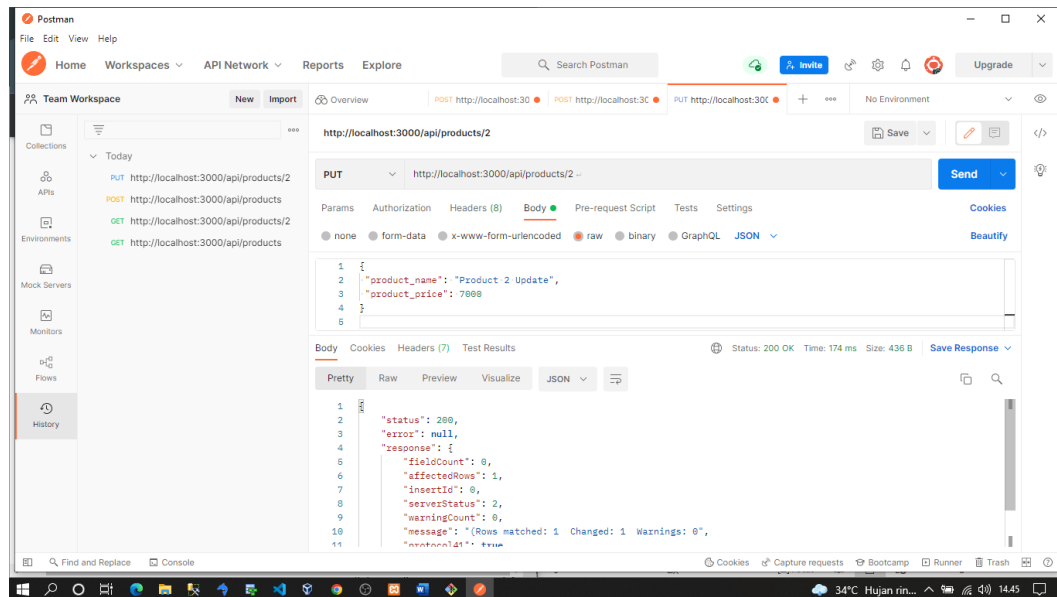




#### #4. Update Product (PUT)

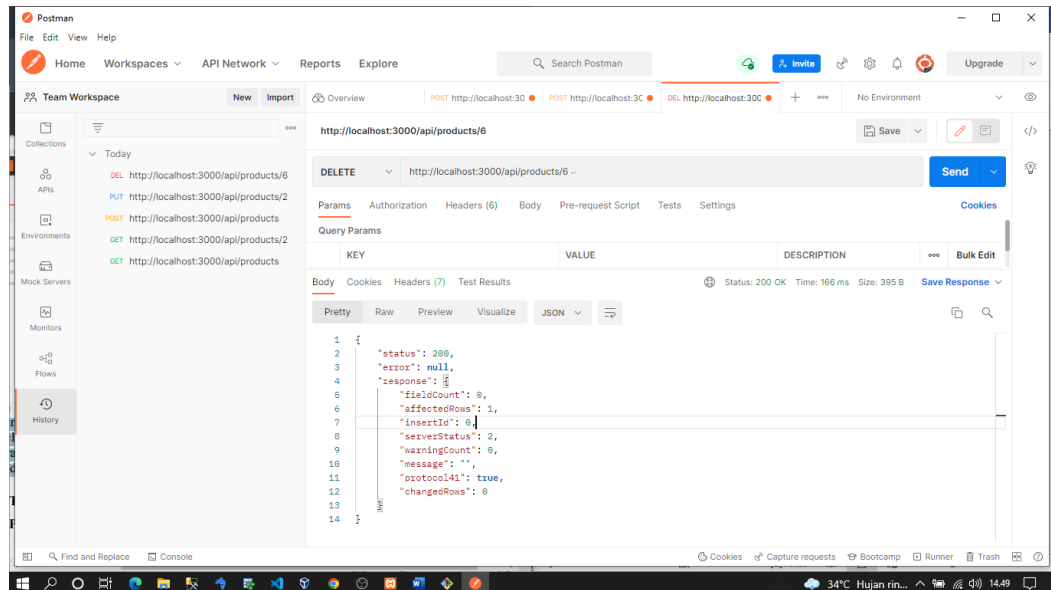
Selanjutnya kita ketikkan URL <http://localhost:3000/api/products/2> kemudian kita pilih Method PUT dan kita masukan data pada a kolom JSON(application/json)

{ "product\_name": "Product 2 Update", "product\_price": 7000 } setelah itu kita klik tombol Send. Jika kita perhatikan pada bagian response, terdapat "affectedRows": 1, dan "changedRows": 1. Itu berarti terdapat satu data yang diupdate ke database dengan product\_id='2' sesuai dengan parameter pada URL.



## #5 Delete Product (DELETE)

Selanjutnya kita ketikkan URL <http://localhost:3000/api/products/6> kemudian kita pilih Method Delete kemudian klik tombol Send, Jika kita perhatikan pada bagian response, terdapat “affectedRows”: 1, “insertId”: 0 ,dan “changedRows”: 0. Itu berarti terdapat satu data yang dihapus (delete) ke database dengan product\_id=’6’ sesuai dengan parameter pada URL maka Akan terlihat hasilnya seperti gambar di bawah ini.



## E. Daftar Pustaka

1. TSJ TV  
[www.youtube.com/c/TSJTvPro](http://www.youtube.com/c/TSJTvPro)