

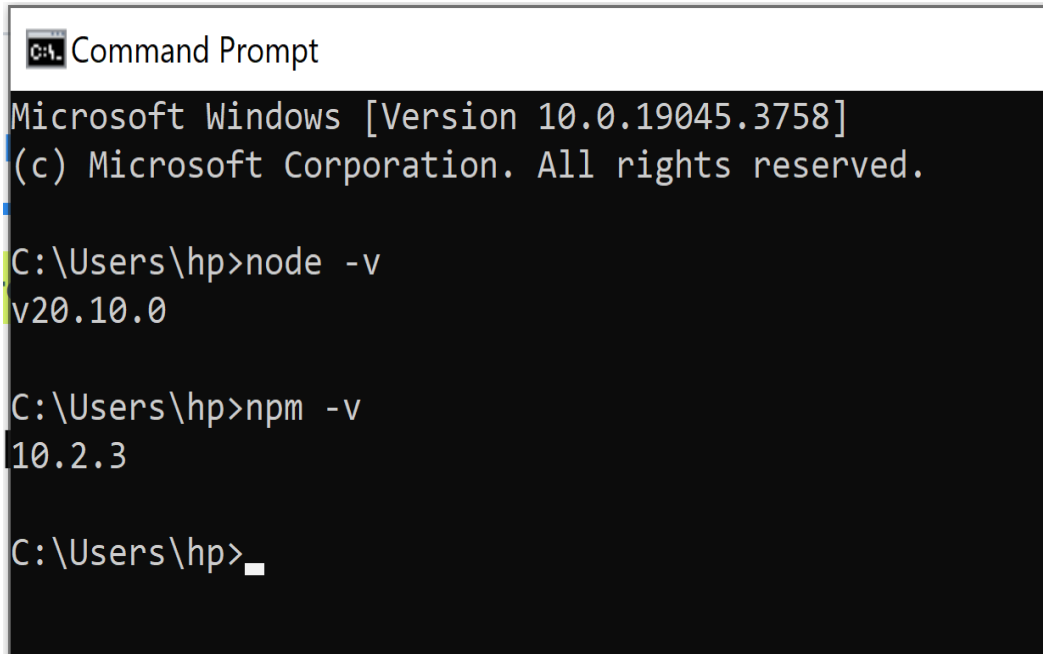
Tạo API như thế nào

1.Hướng dẫn sv cài đặt

Cài JSON SERVER

<https://nodejs.org/en/download>

1. Cài đặt , nhấn next... finish
2. Mở cmd, vào thư mục đã cài đặt, check version



```
Command Prompt
Microsoft Windows [Version 10.0.19045.3758]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hp>node -v
v20.10.0

C:\Users\hp>npm -v
10.2.3

C:\Users\hp>
```

3. Cài đặt gói sử dụng NODE package manager(NPM)

Npm install -g json-server

Hoặc có thể chỉ cài lên 1 folder (tạo folder D:\install\jsonserver

Npm init

```
D:\>cd install

D:\install>cd jsonserver

D:\install\jsonserver>npm -v
10.2.3

D:\install\jsonserver>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (jsonserver)
```

Và enter đến cuối chọn yes: (để tạo 1 project)

```
D:\install\jsonserver>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (jsonserver)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\install\jsonserver\package.json:

{
  "name": "jsonserver",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes_
```

Sau đó gõ : npm install i json-server

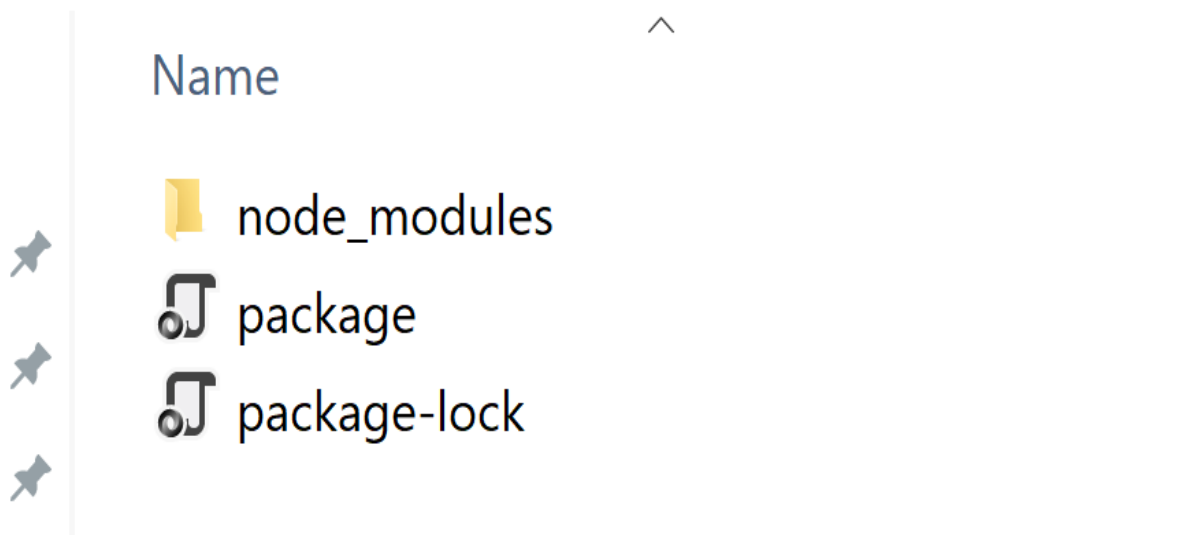
```
D:\install\jsonserver>npm install i json-server

added 117 packages, and audited 118 packages in 9s

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

found 0 vulnerabilities
```

› This PC › New Volume (D:) › install › jsonserver



2. Giải thích về json

JSON là gì? **JSON** là từ viết tắt của **JavaScript Object Notation** - Có thể hiểu là "Kí hiệu Object trong Javascript".

JSON là một định dạng trao đổi dữ liệu chuẩn, nó là một ngôn ngữ độc lập và hỗ trợ các cấu trúc dữ liệu như mảng và đối tượng.

Nếu bạn đã biết về [Object Javascript](#) thì ví dụ về **JSON** sau đây cũng chính là để biểu diễn cho một **Object** trong Js.

```
{
  "type": "laptop",
  "brand": "Asus",
  "operating system": "Windows 10 Pro",
  "graphic card": "NVIDIA"
}
```

Trong đó, gồm có 2 phần:

- **key:** *type, brand, operating system, graphic card.*
- **value:** *laptop, Asus, Windows 10 Pro, NVIDIA.*

JSON được sử dụng làm gì? JSON là một cú pháp để gửi, nhận và trao đổi dữ liệu.

Trao đổi dữ liệu: Khi trao đổi dữ liệu giữa trình duyệt và máy chủ, dữ liệu chỉ có thể là văn bản. **JSON** là văn bản, chúng ta có thể chuyển đổi bất kỳ đối tượng **JavaScript** nào thành **JSON** và gửi **JSON** đến máy chủ.

Chúng ta cũng có thể chuyển đổi bất kỳ **JSON** nào nhận được từ máy chủ thành các đối tượng **JavaScript**. Bằng cách này, chúng ta có thể làm việc với dữ liệu dưới dạng các đối tượng **JavaScript** mà không cần phân tích cú pháp và dịch phức tạp.

3. Tạo file `sp_thanh.json` thành API trên server ảo của mình

3.1 tạo file `sp_thanh.json`

```
4. {
5.   "sp_thanh": [
6.     {
7.       "id": 1,
8.       "name": "laptop thinkpad",
9.       "description": "RAM:8GB, HDD: 256GB, WIN10",
10.      "PRICE": 15000000,
11.      "category_id": 1
12.    },
13.    {
14.      "name": "DELL XPS",
15.      "description": "RAM:8GB, HDD: 512GB, WIN 10 64 BIT",
16.      "PRICE": "20000000",
17.      "category_id": "3",
18.      "id": 2
19.    },

```

```

20.   {
21.     "name": "MAC",
22.     "description": "RAM:16GB, HDD: 256GB, OS",
23.     "PRICE": 35000000,
24.     "id": 4
25.   },
26.   {
27.     "name": "MAC",
28.     "description": "RAM:16GB, HDD: 256GB, OS",
29.     "PRICE": 35000000,
30.     "id": 5
31.   }
32. ]
33.}

```

3.2 open file package.json, thêm dòng start

```

"scripts": {
  "start": "json-server --watch sp_thanh.json" ,
  "test": "echo \"Error: no test specified\" && exit 1"
},
"author": "",

```

```

D:\install\jsonserver>npm start

> jsonserver@1.0.0 start
> json-server --watch sp_thanh.json

\{^_^}/ hi!

Loading sp_thanh.json
Done

Resources
http://localhost:3000/sp_thanh

Home
http://localhost:3000

Type s + enter at any time to create a snapshot of the database
Watching...

```

Đây là API của nó (API là url chứa dữ liệu mình muốn hiển thị ra)

http://localhost:3000/sp_thanh

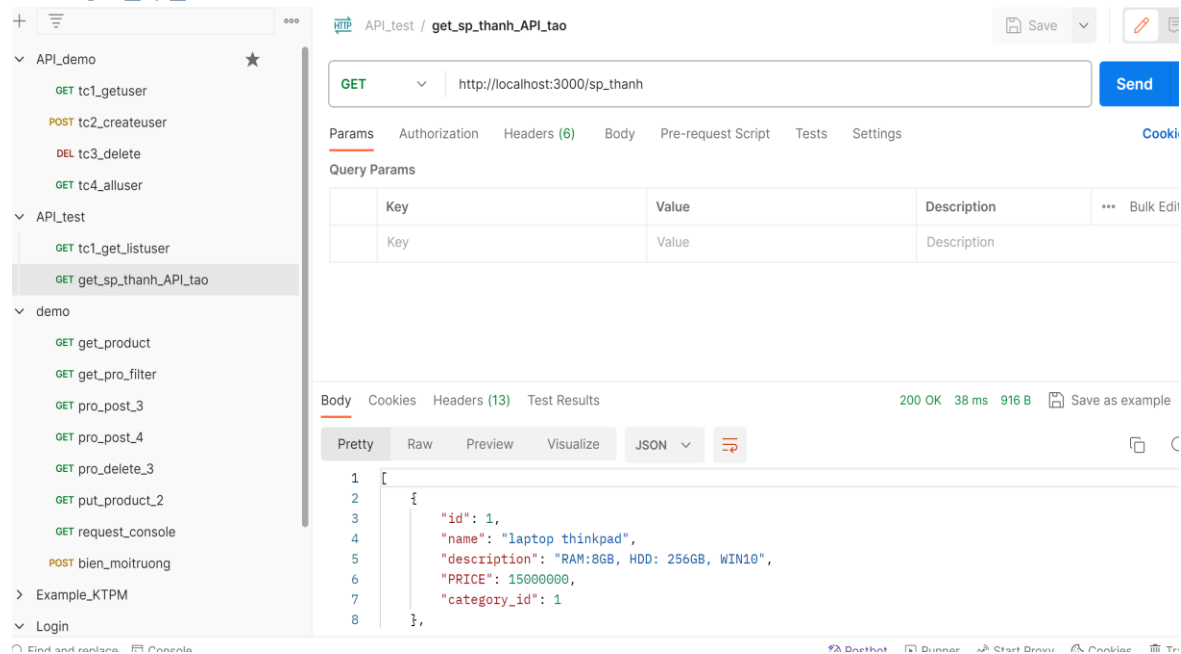
open trong chrome sẽ thấy

← → ↻ ⓘ localhost:3000/sp_thanh

```
[
  {
    "id": 1,
    "name": "laptop thinkpad",
    "description": "RAM:8GB, HDD: 256GB, WIN10",
    "PRICE": 15000000,
    "category_id": 1
  },
  {
    "name": "DELL XPS",
    "description": "RAM:8GB, HDD: 512GB, WIN 10 64 BIT",
    "PRICE": "20000000",
    "category_id": "3",
    "id": 2
  },
  {
    "name": "MAC",
    "description": "RAM:16GB, HDD: 256GB, OS",
    "PRICE": 35000000,
    "id": 4
  },
  {
    "name": "MAC",
    "description": "RAM:16GB, HDD: 256GB, OS",
    "PRICE": 35000000,
    "id": 5
  }
]
```

3.3 Test trên API vừa tạo

3.3.1 get_sp_thanh



3.3.2 Thiết lập biến môi trường trong postman

Đây là một trong các tính năng hữu ích của postman cho các biến dữ liệu giống nhau để có thể tái sử dụng nhiều chỗ. Việc thiết lập chúng thành các biến dùng chung tránh tình trạng lặp lại trong quá trình kiểm thử, cũng như giúp dễ dàng thay đổi giá trị của chúng khi thực hiện kiểm thử các giá trị khác nhau.

Các bước thiết lập:

- Click biểu tượng mắt ở trên phải. Ở vùng Globals, click liên kết Edit.
- XÂY dựng biến trong 1 project, chọn project demo, chọn Edit

DEL pro_
POST prc
PUT put_
demo
>
+
...
No Environment

Environment
Add

No active Environment

An environment is a set of variables that allow you to switch the context of your requests.

Globals
Add

No global variables

Global variables are a set of variables that are always available in a

Chọn add globals

Globals
Save

Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be edited by anyone in that workspace. [Learn more about globals](#)

Filter variables

	Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/>	demo_link	default	http://localhost:3000	http://localhost:3000
	Add new variable			

Globals
Save
Export

Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. [Learn more about globals](#)

Filter variables

	Variable	Type	Initial value	Current value	...
<input checked="" type="checkbox"/>	demo_link	default	http://localhost:3000	http://localhost:3000	
	Add new variable				

- Sử dụng biến demo_link theo cú pháp {{ demo_link }}.
- Vào get_sp_thanh_API_tao: sử dụng biến demo_link

API_test / get_sp_thanh_API_tao

GET {{demo_link}}/sp_thanh

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1

Body Cookies Headers (13) Test Results 200 OK 31 ms 1.01 KB Save as example

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 1,
4     "name": "laptop thinkpad",
5     "description": "RAM:8GB, HDD: 256GB, WIN10",
6     "PRICE": 15000000,
7     "category_id": 1
8   },

```

Postbot Runner Start Proxy Cookies

3.3.3 Viết test script trong postman

API_test / test_script_API_tao

GET {{demo_link}}/sp_thanh

Params Authorization Headers (6) Body Pre-request Script **Tests** Settings

```

1 pm.variables.get("demo_link");
2 console.log(pm.variables.get("demo_link"));

```

Body Cookies Headers (13) Test Results 200 OK 33 r

Pretty Raw Preview Visualize JSON

GET http://localhost:3000/sp_thanh

"http://localhost:3000"

3.3.4 một số thao tác thông dụng với test script

Một số thao tác thông dụng với test script

Thiết lập và lấy giá trị biến môi trường

```
pm.environment.set("variable_key", "variable_value")
pm.environment.get("variable_key")
```

Xoá biến môi trường

```
pm.environment.unset("variable_key")
```

Thiết lập và lấy giá trị biến collection

```
pm.collectionVariables.set("variable_name",
                           "variable_value")

pm.collectionVariables.get("variable_name")
```

Xoá biến collection

```
pm.collectionVariables.unset("variable_name")
```

Thiết lập và lấy giá trị biến toàn cục

```
pm.globals.set("variable_key", "variable_value")
pm.globals.get("variable_key")
```

Xoá biến toàn cục

```
pm.globals.unset("variable_key")
```

Lấy giá trị biến có trong globals hoặc một môi trường có hiệu lực

```
pm.variables.get("variable_key")
```

Xử lý response

- Kiểm tra response body bằng một chuỗi

```
pm.test("Response Handling", function() {
    pm.response.to.have.body("str")
})
```

- Kiểm tra response body chứa một chuỗi nào đó

```
pm.test("Response Handling", function() {
    pm.expect(pm.response.text()).to.include("str")
})
```

- Kiểm tra dữ liệu trả về

```
pm.test("Response Handling", function() {
    var d = pm.response.json()
    pm.expect(d.value).to.equal(1)
})
```

- Kiểm tra header có thuộc tính Content-Type không.

```
pm.test("Response Handling", function() {  
    pm.response.to.have.header("Content-Type")  
})
```

- Kiểm tra thời gian phản hồi dưới 200 mili giây

```
pm.test("Response Handling", function() {  
    pm.expect(pm.response.responseTime)  
        .to.be.below(200)  
})
```

- Kiểm tra status code của POST request là thành công

```
pm.test("Response Handling", function() {  
    pm.expect(pm.response.code)  
        .to.be.oneOf([200, 201, 202])  
})
```

Gửi request bất đồng bộ (chỉ dùng trong pre-request test script)

```
pm.sendRequest("https://postman-echo.com/get",  
    function (err, response) {  
        console.log(response.json());  
    });
```

Một số phương thức assertion thông dụng khác

- Kiểm tra kiểu dữ liệu

```
pm.expect("postman").to.be.a("string")  
pm.expect({a: 1}).to.be.an("object")  
pm.expect(undefined).to.be.an("undefined")
```

- Kiểm tra chuỗi rỗng hoặc mảng rỗng

```
pm.expect([]).to.be.empty  
pm.expect("").to.be.empty
```

- Kiểm tra kết quả chứa/không chứa key nào đó

```
pm.expect({a: 1, b: 2}).to.have.all.keys("a", "b")  
pm.expect({a: 1, b: 2}).to.have.any.keys("a")  
pm.expect({a: 1, b: 2}).to.not.any.keys("c", "d")
```

- Kiểm tra chiều dài của chuỗi hoặc mảng

```
pm.expect("foo").to.have.lengthOf(3)  
pm.expect([2, 1, 3]).to.have.lengthOf(3)
```

- Kiểm tra hai mảng có các phần tử giống nhau

```
pm.expect([1, 2, 3]).to.have.members([3, 1, 2])
```

- Kiểm tra kết quả chứa một phần tử nào đó

```
pm.expect([1, 2, 3]).to.include(2)  
pm.expect({a: 1, b: 2}).to.include({a: 1})
```

End