

Criando uma REST API com JSON Server – parte 1

Nessa série de tutoriais, você vai aprender de uma forma simples e eficaz a implementar uma *REST API* totalmente falsa com zero codificação em pouquíssimo tempo.

Imagine você na seguinte situação. Você é um desenvolvedor *front-end* e precisa de um *back-end* rápido para prototipagem e simulação do projeto para a reunião com o cliente, e não tem tempo para esperar a equipe de *back-end* desenvolver e te entregar as funcionalidades necessárias.

Para resolver essa situação, você pode usar o *JSON Server* que é uma biblioteca capaz de criar uma *REST API fake* com todos os *endpoints* de um recurso, como os 4 principais verbos *HTTP*: *GET*, *POST*, *PUT* e *DELETE*. Dessa forma, seu *front-end* poderá consumir essa *API* simulada, possibilitando a criação de toda a camada *HTTP* da aplicação.

Agora vamos instalar o *JSON Server*. Para isso vou levar em conta que você já tenha um projeto *React* criado. Não que seja obrigatório ter um projeto, e ainda ser somente *React*. Apenas iremos aproveitar o projeto criado em aula. Mas você poderá instalar o *JSON Server* independentemente do local.

Instando o *JSON Server*

Em seu projeto, crie uma pasta chamada “**backend**” conforme a figura 1.

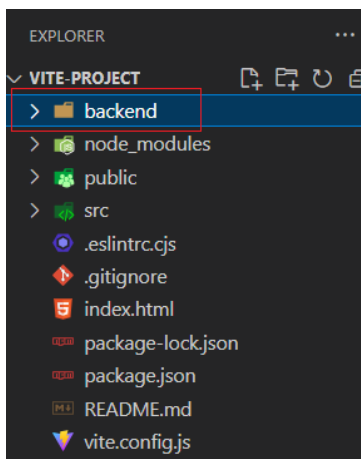


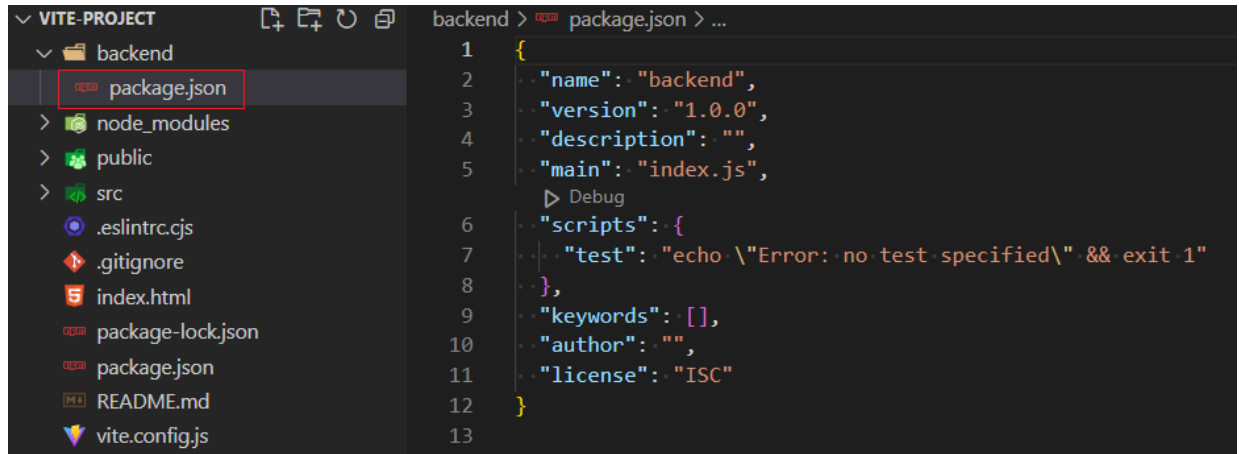
Figura 1 - criação da pasta backend no projeto.

Conforme a figura 2, abra o terminal do **VSCode** e execute o comando **cd backend** para entrar na pasta.

```
PS C:\Users\rafae\projetos\react\vite-project> cd backend
PS C:\Users\rafae\projetos\react\vite-project\backend>
```

Figura 2 - comando para entrar na pasta backend.

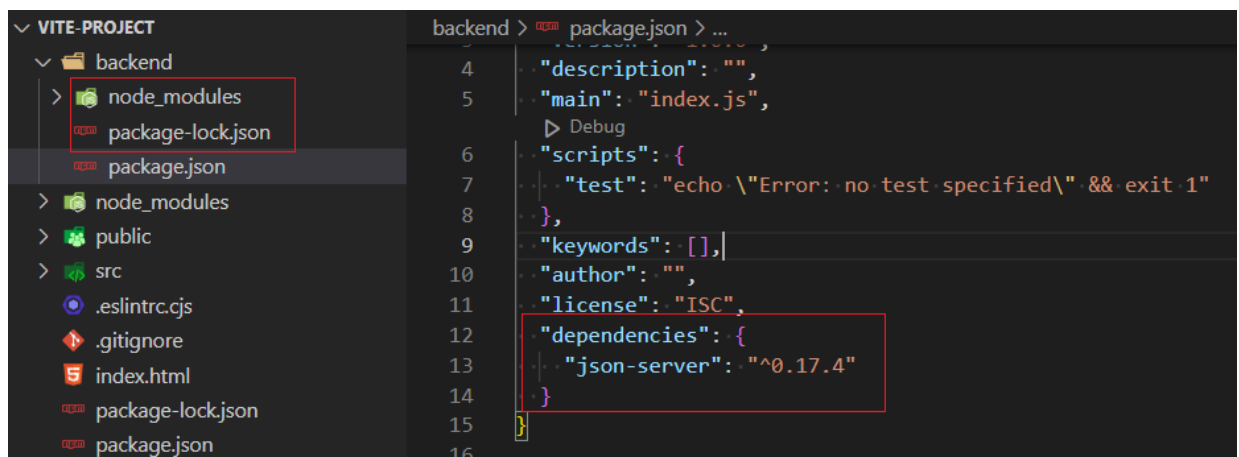
O próximo passo, execute o comando **npm init -y**, ele irá criar o arquivo “**package.json**” para gerenciar nossas dependências, mostrado na figura 3.



```
backend > npm init -y
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC"
12 }
13
```

Figura 3 - resultado após a execução do comando **npm init -y**.

Para instalar o **JSON Server**, digite o comando **npm i json-server**. Com isso ele irá criar a dependência em nosso projeto conforme a figura 4.



```
backend > npm i json-server
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "json-server": "^0.17.4"
14   }
15 }
16
```

Figura 4 - dependência do **JSON Server** criada.


Com o processo finalizado, agora iremos criar um arquivo com o nome “**db.json**” dentro da pasta “**backend**”. Esse arquivo servirá para que o **JSON Server** crie a **REST API** baseado nele. Nele iremos criar objeto **JSON** que terá todos os **endpoints** da nossa **API**.

```
backend > {} db.json > [ ] products
1  {
2    "products": [
3      {
4        "id": 1,
5        "name": "Caneta BIC preta",
6        "price": 5.89
7      },
8      {
9        "id": 2,
10       "name": "Notebook Mac Pro",
11       "price": 12000.00
12     },
13     {
14       "id": 3,
15       "name": "Samsung 20+",
16       "price": 5000.89
17     }
18   ]
19 }
```

Figura 5 - arquivo db.json com o objeto json.

Vá para o arquivo **'package.json'** e faça a alteração conforme demonstrado na figura 6.

```
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1"
8  },
```

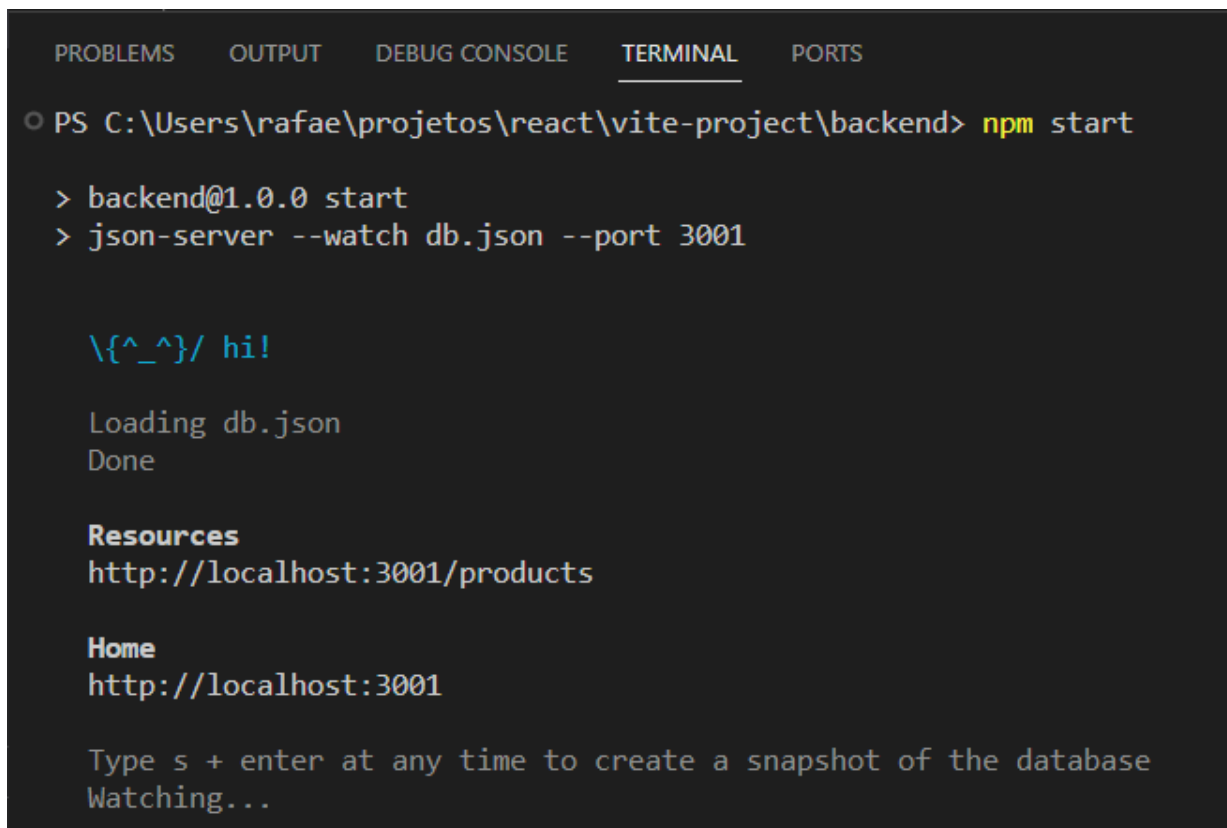


```
6  "scripts": {
7    "start": "json-server --watch db.json --port 3001"
8  },
```

Figura 6 - alteração do arquivo package.json

Basicamente estamos configurando para iniciar o **JSON Server** com o arquivo **"db.json"** na porta 3001 e fique monitorando as eventuais alterações com a opção **–watch**.

No terminal dentro da pasta **"backend"** execute o comando **npm start** para iniciar nossa API.



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rafae\projetos\react\vite-project\backend> npm start

> backend@1.0.0 start
> json-server --watch db.json --port 3001

\{^_^}/ hi!

Loading db.json
Done

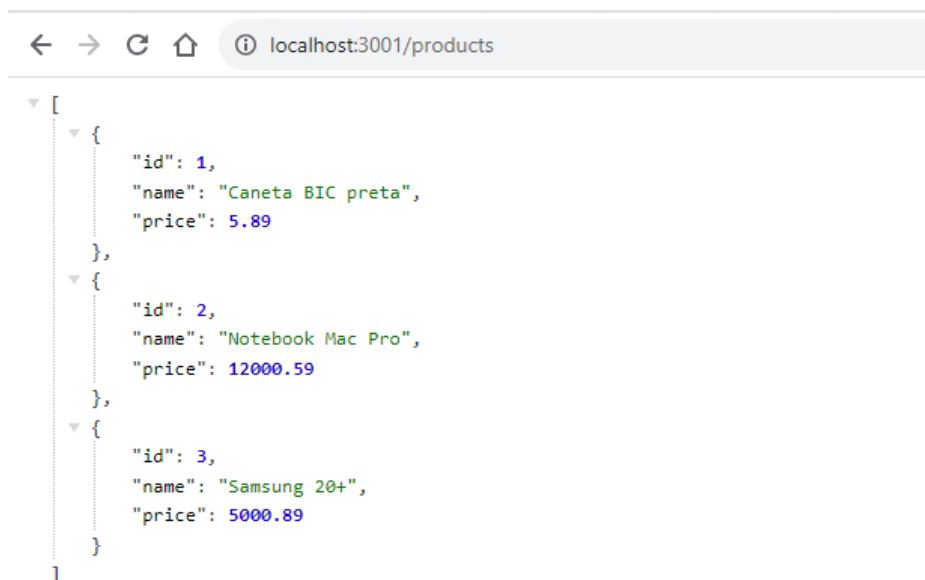
Resources
http://localhost:3001/products

Home
http://localhost:3001

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

Figura 7 - execução da API.

Para testarmos nossa API, abra o navegador e digite a url <http://localhost:3001/products> . você obterá



```
localhost:3001/products

[
  {
    "id": 1,
    "name": "Caneta BIC preta",
    "price": 5.89
  },
  {
    "id": 2,
    "name": "Notebook Mac Pro",
    "price": 12000.59
  },
  {
    "id": 3,
    "name": "Samsung 20+",
    "price": 5000.89
  }
]
```

Figura 8 - resposta da API ao digitar a url.

Digitando a url <http://localhost:3001/products/1> , você obterá apenas o produto de id igual a 1

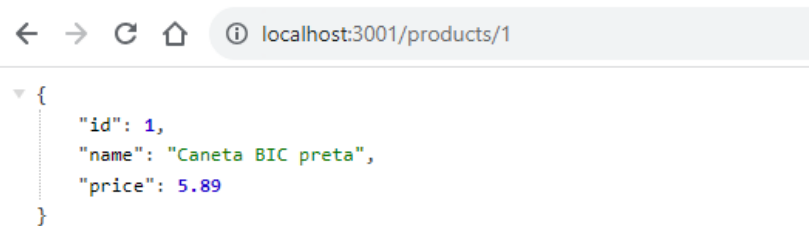


Figura 9 - resposta da API trazendo apenas um produto.

Outra maneira de testar a sua API é usar um cliente REST como o **POSTMAN** ou **INSOMNIA**. No **VSCode** existem plugins que você pode instalar e usá-los para testar os métodos como **GET**, **POST**, **PUT** e **DELETE**.

Clique o ícone de *Extensions* e procure por *Thunder Client*, conforme a figura 10, mas fique à vontade em escolher outra extensão.

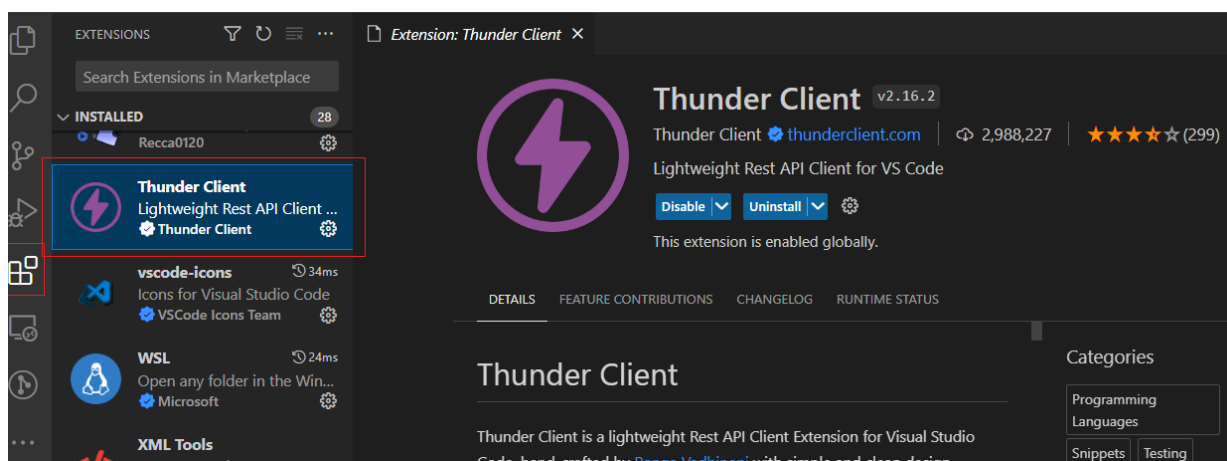


Figura 10 - extensão do VSCode para um cliente REST.

Em breve, disponibilizarei um tutorial de como utilizar o *Thunder Client*.

Rotas

Com base no arquivo **'db.json'**, aqui estão todas os métodos **HTTP** que você poderá utilizar em sua aplicação:

Tabela 1 - métodos HTTP e rotas disponíveis em nossa API.

Método HTTP	Rotas	Ação
GET	/products	Obtém todos os produtos.
GET	/products/1	Obtém o produto com id igual a 1.
POST	/products	Salva um produto.
PUT	/products/1	Atualiza todos os dados do produto com id igual a 1.
PATCH	/products/1	Atualiza parte dos dados do produto com id igual a 1.
DELETE	/products/1	Remove o produto com o id igual a 1.

Concluindo...

Muitas vezes no desenvolvimento de uma aplicação do lado do *front-end* precisamos ter a estrutura de *REST API* para apresentação, prototipação ou até mesmo realizar testes, mas que ela ainda está sendo desenvolvida pela equipe *back-end*. Não queremos criar nosso *front-end* de uma maneira e depois ter que alterar quando o *back-end* entregar a estrutura de rotas, *endpoints* e tudo que envolve uma *REST API*.

Para resolver essa situação, vimos nesse tutorial como criar uma *REST API fake* para que possamos consumi-la de tal forma como se estivéssemos consumindo a *API* real de nossa aplicação. *JSON Server* é um aplicativo *JavaScript* que nos permite realizar tal feito.

Referências

json-server. Disponível em: <<https://www.npmjs.com/package/json-server>>.
Acesso em: 29 nov. 2023.

EGGHEADIO. Expert led courses for front-end web developers. Disponível em:
<<https://egghead.io/lessons/javascript-creating-demo-apis-with-json-server>>.
Acesso em: 29 nov. 2023.

FRANCO, F. Simulando uma API REST com JSON Server de maneira simples.
Disponível em: <<https://www.fabricadecodigo.com/json-server/#:~:text=O%20que%20%C3%A9%20o%20JSON%20Server&text=JSON%20Server%20%C3%A9%20uma%20biblioteca>>. Acesso em: 29 nov. 2023.