

## Criando uma REST API com JSON Server – parte 3

Com o nosso *backend JSON Server* rodando, vamos utilizar a extensão *Thunder Client* para consumir nossa *REST API*. Lembrando que esse *REST Client* não é exclusivo para o *JSON Server*, ele pode ser usado para consumir e testar qualquer *API* que o seu aplicativo estiver usando.

O propósito desse tutorial é fornecer conhecimentos básicos sobre a ferramenta para a sua inicialização nos conceitos de *REST API*, deixando que você explore mais detalhes acessando a sua documentação pelo link <https://github.com/rangav/thunder-client-support#setenv> .

Levando em conta que você já instalou a extensão demonstrada no tutorial anterior, localize e clique no ícone do *Thunder Client* na *Action Bar* do *VSCode* conforme mostrado na figura 1.

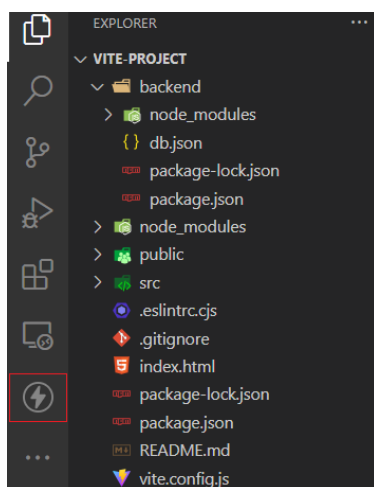


Figura 1 - Extensão do VSCode para Rest Api

Caso não apareça o ícone em sua *Action Bar*, certifique-se que tenha instalado a extensão ou clique nas reticências para mostrar mais opções.

Ao clicar no ícone do *Thunder Client*, irá aparecer no lado esquerdo do *VSCode* suas opções de tarefas para serem executadas conforme demonstrado na figura 2.

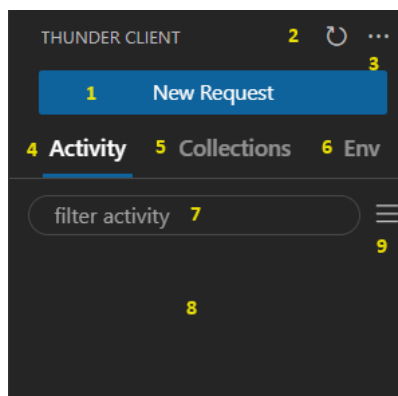


Figura 2 - Interface do Thunder Client

- 1 – Botão para adicionar uma nova requisição;
- 2 – Botão para atualizar o resultado de uma requisição;
- 3 – Menu com diversas opções para serem executadas no *Thunder Client*;
- 4 – Aba *Activity* (Atividade), onde é listada todas as requisições feitas na ferramenta;
- 5 – Aba *Collections* (Coleções), onde você poderá criar e gerenciar grupos de requisições;
- 6 – Aba *Environment Variables* (Variáveis de ambiente), onde você poderá criar e gerenciar variáveis locais e globais que poderão ser usadas nas requisições;
- 7 – Local para filtrar a lista de atividades, coleções e variáveis de ambiente;
- 8 – Local onde serão exibidas as atividades, coleções e variáveis de ambiente salvas na ferramenta;
- 9 – Menu com opções para serem executadas nas atividades, coleções e variáveis de ambiente.

## Realizando a primeira requisição

Tomando como base o arquivo *JSON* criado no primeiro tutorial, usaremos os *endpoints* e os métodos *HTTP* conforme demonstrado na tabela 1.

Tabela 1- Métodos HTTP e os endpoints

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.

Clique no botão “**New Request**”, aparecerá um formulário com campos a serem preenchidos conforme a requisição que desejamos realizar.

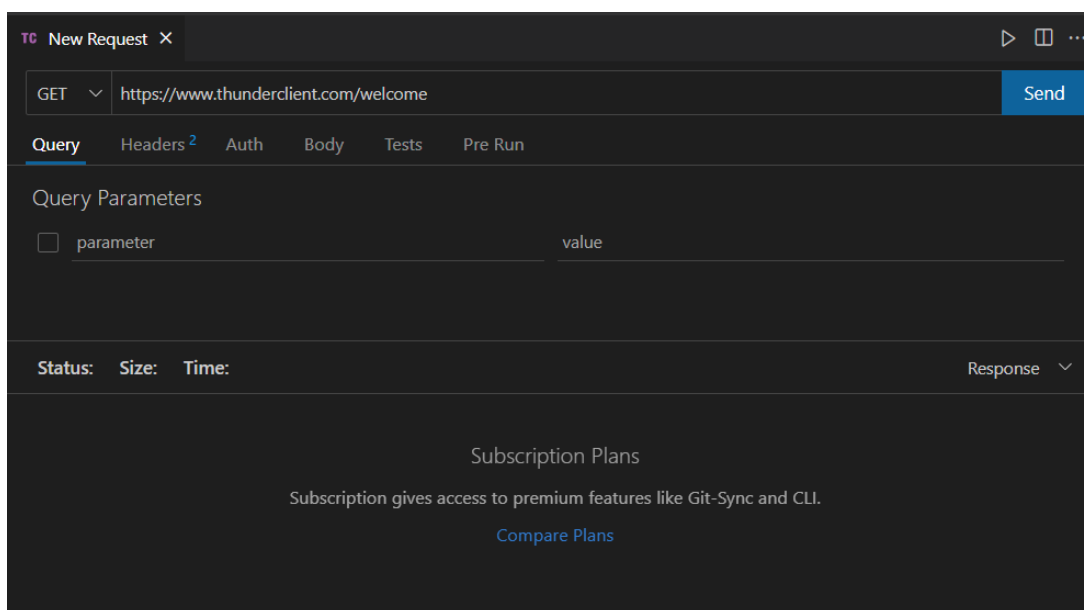


Figura 3 - Interface para nova requisição

Seguindo a tabela 1, vamos executar uma requisição com o método *GET* para obter todos os produtos. Para tal, altere a *url* ao lado da palavra *GET* para a *url* do nosso *JSON Serve* <http://localhost:3001/products> e em seguida clique no botão “**Send**”.

Perceba na figura 4 que iremos ter como resposta um erro informando que a conexão foi recusada pelo servidor (*Connection was refused by the server*).

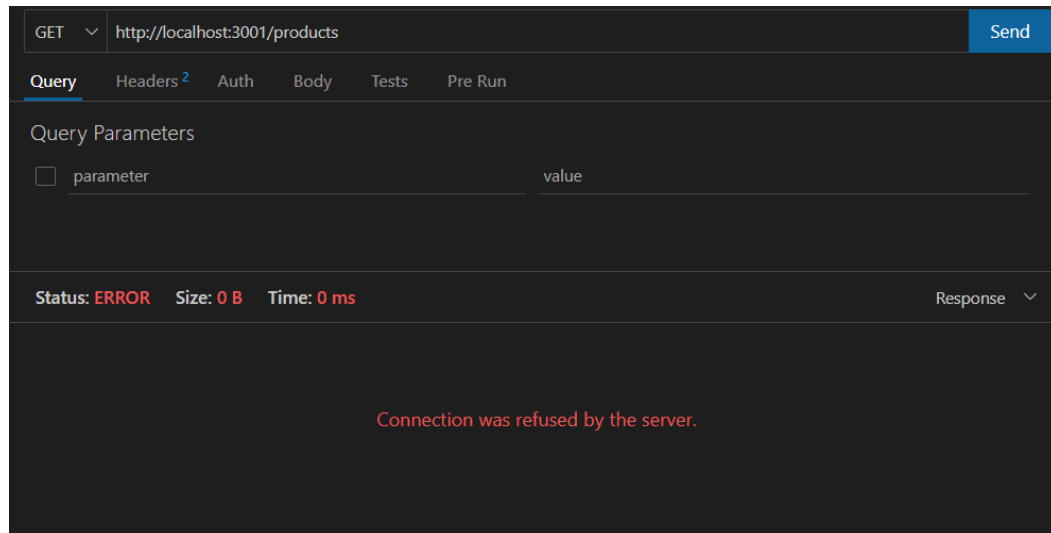


Figura 4 - Resposta da requisição informando o erro de conexão

Por algum motivo o *Thunder Client* não consegue acessar o *localhost*. Uma forma que podemos contornar esse problema é substituir a palavra *localhost* por `[::1]`.

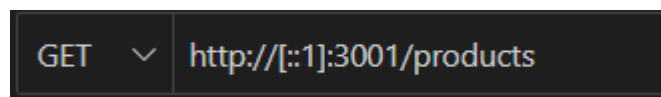


Figura 5 - Requisição para obter todos os produtos

Após a substituição, clique novamente no botão “**Send**” e obteremos como resposta um *JSON* com todos os produtos conforme mostrado na figura 6.

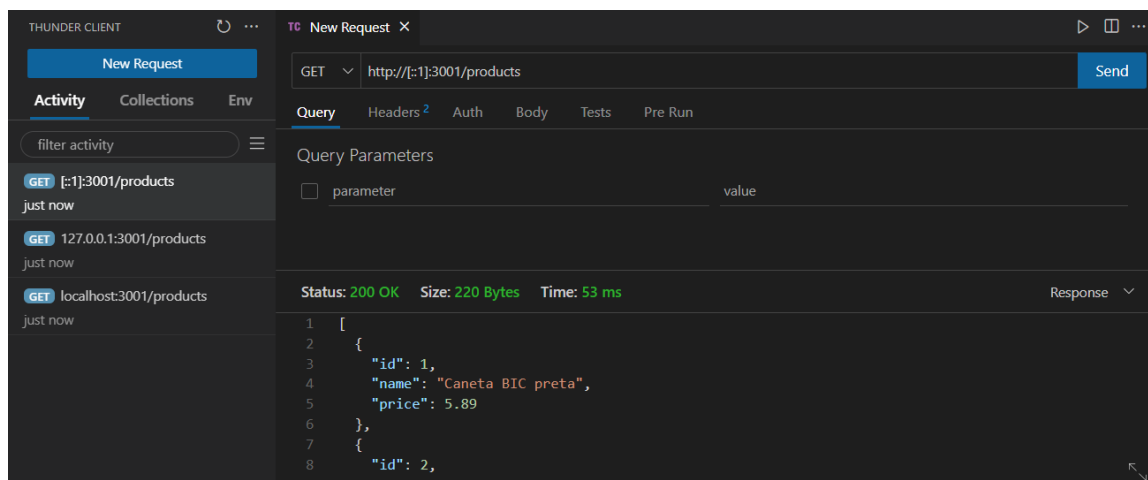


Figura 6 - Resultado da requisição

Observe também, conforme enviamos as requisições elas são armazenadas na aba *Activity* em forma de lista, onde você poderá clicar em uma delas para enviar posteriormente ou excluir da lista.

Agora vamos enviar uma requisição para obtermos apenas o produto que tem o id igual a 1. Altere a *url* da requisição conforme a figura 7 e em seguida clique no botão “**Send**”.

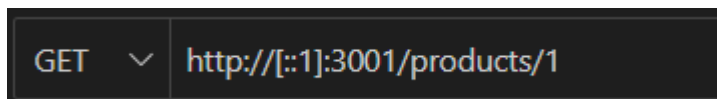


Figura 7 - Requisição para obter o produto do id igual a 1

Você verá na figura 8, a resposta trazendo apenas um produto conforme nossa requisição.

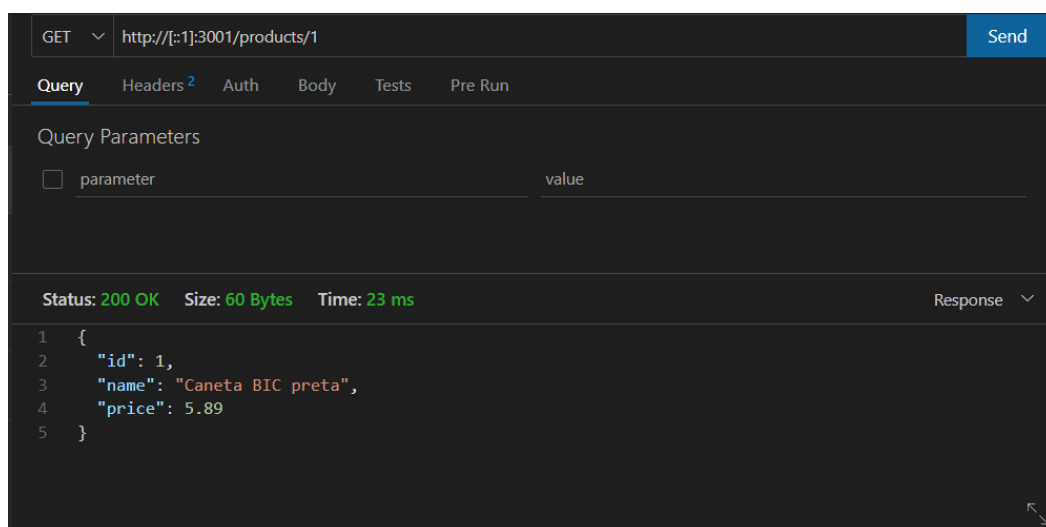


Figura 8 - Resposta da requisição trazendo apenas um produto

## Criando variáveis de ambiente

Em muitos casos quando desenvolvemos a *API*, sua *url* fica um pouco extensa como por exemplo, <https://www.meu-dominio.com/api/v1/products> até chegar no *endpoint* “**products**”. Imagine toda vez que você for enviar uma nova requisição ter que digitar esse caminho. Para resolver essa situação podemos criar uma variável de ambiente e salvar nela a *url base* e usá-la juntamente com os *endpoints*.

Clique na aba “*Env*” e em seguida no ícone de menu ao lado da palavra “**filter environment**” e depois na opção “**New Environment**” conforme demonstrado na figura 9.

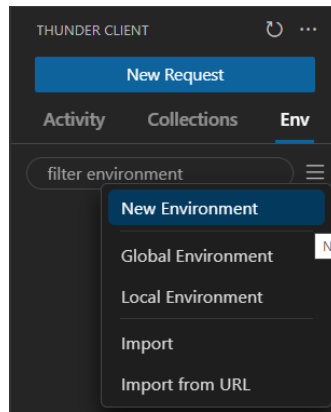


Figura 9 - Menu de opções para variáveis de ambiente

Abrirá um campo de texto acima solicitando o nome do ambiente. Escreva “**general**” e pressione **ENTER**. Irá aparecer do lado esquerdo o ambiente. Agora clique no nome do ambiente e preencha o formulário conforme a figura 10 e ao final clique no botão “**Save**”.

The image shows the 'Update Environment' form in the Thunder Client application. The form has a title 'Update Environment' and a subtitle 'filter variables'. It contains a text input field for 'Name' with the value 'general'. Below this is a table with two columns: 'Variable Name' and 'Value'. The table has two rows: one with 'url\_base' and 'http://[::1]:3001', and another with 'variable' and 'value'. At the bottom of the form, there is a checkbox labeled 'Link to .env file' and a 'Save' button.

Figura 10 - Interface de variáveis de ambiente

Dessa forma será criada uma variável de nome “**url\_base**” com o valor [http://\[::1\]:3001](http://[::1]:3001) que poderemos usá-la em nossas requisições.

Para testarmos a variável, clique no botão “**New Request**” e escreva a *url* com o método *GET* conforme demonstrado na figura 11.

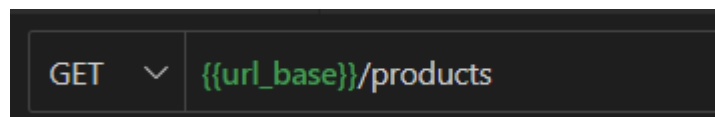


Figura 11 - Url usando uma variável

Veja que o nome da variável está envolvido por duas chaves abrindo “{{” e duas chaves fechando “}}”. Dessa forma quando executarmos a requisição a variável será substituída pelo seu valor, nos poupando de digitarmos *URLs* extensas.

Esse foi apenas um exemplo da utilidade de variáveis de ambiente, para maiores detalhes e uso consulte a documentação.

## Adicionando um produto

Para adicionar um produto devemos usar o método *POST* e enviar dados no corpo (*body*) da requisição por meio do *Form-encode*. Clique no botão “**New Request**” e preencha os dados da requisição conforme a figura 12 e ao final clique no botão “**Send**”.

The screenshot shows a REST client interface. At the top, the method is set to 'POST' and the URL is '{{url\_base}}/products'. A 'Send' button is in the top right. Below the URL bar, there are tabs for 'Query', 'Headers', 'Auth', 'Body', 'Tests', and 'Pre Run'. The 'Body' tab is selected, and within it, the 'Form-encode' sub-tab is active. The 'Form Encoded' section contains three input fields: 'name' with the value 'Computador Desktop Dell i9', 'price' with the value '2755.99', and an unchecked 'name' field with the value 'value'. An 'Import' button is in the top right of the form area.

Figura 12 - Adicionando um produto por meio do método POST

Você terá como resposta a adição do produto conforme a figura 13 e 14.

The screenshot shows the response of a POST request. At the top, it displays 'Status: 201 Created', 'Size: 75 Bytes', and 'Time: 15 ms'. Below this, the response body is shown as a JSON object: 

```
1 {
2   "name": "Computador Desktop Dell i9",
3   "price": "2755.99",
4   "id": 4
5 }
```

Figura 13 - Resposta da requisição do método POST

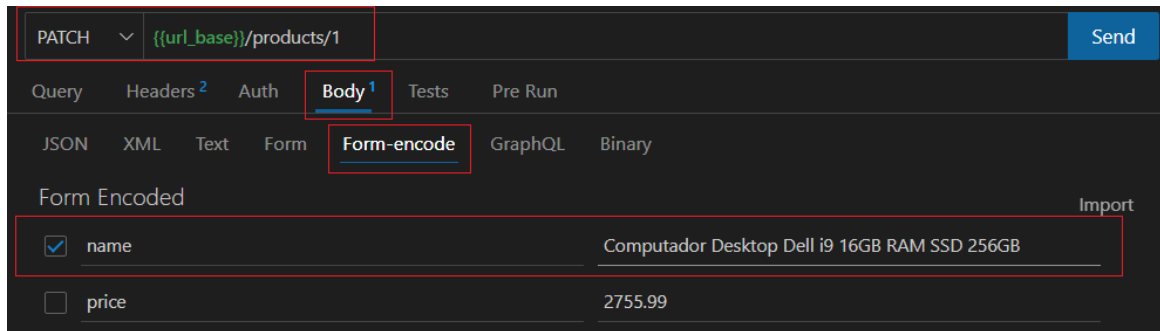
The screenshot shows a JSON file named 'db.json' with the following content: 

```
backend > {} dbjson > [ ] products > {} 2
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": 12000.59
12     },
13     {
14       "id": 3,
15       "name": "Samsung 20+",
16       "price": 5000.89
17     },
18     {
19       "name": "Computador Desktop Dell i9",
20       "price": "2755.99",
21       "id": 4
22     }
23   ]
24 }
```

Figura 14 - Arquivo db.json com o produto adicionado

## Alterando um dado do produto

Se quisermos alterar um dado de um produto específico, usamos o método *PATCH*, passar o dado no corpo (*body*) da requisição por meio do *Form-encode* e na *url* o id do produto conforme demonstrado na figura 15.



The screenshot shows the Thunder Client interface for configuring a PATCH request. The URL is set to `{{url_base}}/products/1`. The 'Body' tab is selected, and the 'Form-encode' option is chosen. The form contains two fields: 'name' with the value 'Computador Desktop Dell i9 16GB RAM SSD 256GB' and 'price' with the value '2755.99'. The 'Send' button is visible in the top right corner.

Figura 15 - Usando o método *PATCH* para alterar um dado do produto

Caso queira alterar todos os dados do produto, basta escolher o método *PUT* e passar os dados via o *Form-encode*.

## Excluindo um produto

Finalizando nossa demonstração, agora iremos excluir um produto utilizando o método *DELETE* informando o id do produto. Adicione nova requisição clicando no botão “**New Request**” e deixe a *url* de acordo com a figura 16 e clique no botão “**Send**”.



The screenshot shows the Thunder Client interface for configuring a DELETE request. The URL is set to `{{url_base}}/products/4`. The 'DELETE' method is selected.

Figura 16 - Método *DELETE* para excluir o produto de id igual a 4

## Concluindo...

Ferramentas *REST Client* são essenciais para os desenvolvedores testarem as *APIs* em desenvolvimento ou como a situação demonstrada nesse tutorial, onde utilizamos o *JSON Server* para fornecer uma *API fake* para prototipação ou apresentação do aplicativo. Existem várias ferramentas que podem ser utilizadas com essa finalidade, e aqui demonstramos de forma básica uma extensão existente no **VSCode**, o *Thunder Client*, que nos permite de forma fácil e rápida testarmos os *endpoints* sem ter que executar outro aplicativo.

Há muitas funcionalidades a serem exploradas no *Thunder Client* conforme seus conhecimentos em *REST API* vão avançando. Não pare por aqui, consulte a documentação e realize novos experimentos.

## Referências

VADHINENI, R. **Thunder Client**. Disponível em:  
<<https://github.com/rangav/thunder-client-support#setenv>>. Acesso em: 2 dez.  
2023.