Material de aula consumo API.

Criar novo projeto angular

NG new consumoAPI

Instalar Jason-server

**Passo a passo para simular uma API**

**1 – Começando com JSON Server**

**Instalação**

npm install -g json-server

**Criando o arquivo de banco de dados**

Crie um arquivo de nome **db.json** em qualquer pasta do seu computador.

{

"produtos": [

{

"id": 1,

"nome": "Hambúrguer",

"descricao": "Pão, bife de hambúrguer 90g, salada e batata.",

"preco": 8.5,

"categoria\_id": 1

},

{

"id": 2,

"nome": "X-Búrguer",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, salada e batata.",

"preco": 10.5,

"categoria\_id": 1

},

{

"id": 3,

"nome": "X-Bacon",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, 2 fatia de bacon, salada e batata.",

"preco": 12.5,

"categoria\_id": 1

},

{

"id": 4,

"nome": "X-Tudo",

"descricao": "Pão, 2 bifes de hambúrguer 90g, 2 fatias de queijo, 4 fatias de bacon, salada e batata.",

"preco": 14.5,

"categoria\_id": 1

},

{

"id": 5,

"nome": "Coca cola 350ml",

"descricao": "",

"preco": 5.5,

"categoria\_id": 2

},

{

"id": 6,

"nome": "Coca cola 600ml",

"descricao": "",

"preco": 7.5,

"categoria\_id": 2

}

],

"categorias": [

{

"id": 1,

"nome": "Hambúrgueres"

},

{

"id": 2,

"nome": "Refrigerantes"

}

]

}

**Iniciando o servidor**

json-server --watch db.json

Lembrando que por padrão a API vai funcionar no enderço: [http://localhost:3000](http://localhost:3000/)

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**2 – Rotas**

As rotas definem como você vai acessar a API.  
As rotas são:

| **Request** | **URL** | **Detalhes** |
| --- | --- | --- |
| GET | /produtos | Busca todos os produtos |
| GET | /produtos/1 | Busca um produto |
| POST | /produtos | Salva um produto |
| PUT | /produtos/1 | Atualiza dos os dados do produto |
| PATCH | /produtos/1 | Atualiza parte dos dados do produto |
| DELETE | /produtos/1 | Remove um produto |

As rotas vão ser compostas pelo endereço base (localhost:3000) mais o recurso que você quer acessar com por exemplo “produtos”.  
Para executar os requests de exemplo abaixo você pode usar a ferramenta **Postman** (<https://www.getpostman.com/>) ou alguma outra de sua preferencia.  
Para ver os resultados na pratica assista o vídeo acima ou execute os exemplos no seu computador.

**Exemplos:**

**Exemplo 1:** Buscar todos os produtos

Dados de envio

GET http://localhost:3000/produtos/

Content-Type: application/json

Retorno

[

{

"id": 1,

"nome": "Hambúrguer",

"descricao": "Pão, bife de hambúrguer 90g, salada e batata.",

"preco": 8.5,

"categoria\_id": 1

},

{

"id": 2,

"nome": "X-Búrguer",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, salada e batata.",

"preco": 10.5,

"categoria\_id": 1

},

{

"id": 3,

"nome": "X-Bacon",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, 2 fatia de bacon, salada e batata.",

"preco": 12.5,

"categoria\_id": 1

},

{

"id": 4,

"nome": "X-Tudo",

"descricao": "Pão, 2 bifes de hambúrguer 90g, 2 fatias de queijo, 4 fatias de bacon, salada e batata.",

"preco": 14.5,

"categoria\_id": 1

},

{

"id": 5,

"nome": "Coca cola 350ml",

"descricao": "",

"preco": 5.5,

"categoria\_id": 2

},

{

"id": 6,

"nome": "Coca cola 600ml",

"descricao": "",

"preco": 7.5,

"categoria\_id": 2

}

]

**Exemplo 2:** Buscar apenas um produtos

Dados de envio

GET http://localhost:3000/produtos/1

Content-Type: application/json

Retorno

{

"id": 1,

"nome": "Hambúrguer",

"descricao": "Pão, bife de hambúrguer 90g, salada e batata.",

"preco": 8.5,

"categoria\_id": 1

}

**Exemplo 3:** Salvar um produto

Dados de envio

POST http://localhost:3000/produtos/

Content-Type: application/json

{

"nome": "Hambúrguer de frango",

"descricao": "Pão, bife de hambúrguer 90g de frango, salada e batata.",

"preco": 9.5,

"categoria\_id": 1

}

Retorno

{

"nome": "Hambúrguer de frango",

"descricao": "Pão, bife de hambúrguer 90g de frango, salada e batata.",

"preco": 9.5,

"categoria\_id": 1,

"id": 7

}

**Exemplo 4:** Alterar um produto

Dados de envio

PUT http://localhost:3000/produtos/1

Content-Type: application/json

{

"nome": "Hambúrguer de frango",

"descricao": "Pão, bife de hambúrguer 90g de frango, salada e batata.",

"preco": 10.5,

"categoria\_id": 1

}

Retorno

{

"nome": "Hambúrguer de frango",

"descricao": "Pão, bife de hambúrguer 90g de frango, salada e batata.",

"preco": 10.5,

"categoria\_id": 1,

"id": 1

}

**Exemplo 5:** Alterar apenas o nome de um produto

Dados de envio

PATCH http://localhost:3000/produtos/1

Content-Type: application/json

{

"nome": "Hambúrguer de frango artezanal"

}

Retorno

{

"nome": "Hambúrguer de frango artezanal",

"descricao": "Pão, bife de hambúrguer 90g de frango, salada e batata.",

"preco": 10.5,

"categoria\_id": 1,

"id": 1

}

**Exemplo 6:** Excluir um produto

Dados de envio

DELETE http://localhost:3000/produtos/1

Content-Type: application/json

Retorno

Status: 200 OK

**3 – Filtros**

Para executar filtros é bem simples, você só precisa usar o nome das propriedades do objeto que você quer pesquisar.  
No exemplo abaixo estou pesquisando por produtos que o nome é **“X-Búrguer”**

Dados de envio

GET http://localhost:3000/produtos?nome=X-Búrguer

Content-Type: application/json

Retorno

[

{

"id": 2,

"nome": "X-Búrguer",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, salada e batata.",

"preco": 10.5,

"categoria\_id": 1

}

]

**4 – Paginação**

Para efetuar a paginação você vai usar o parâmetro **“\_page”** e por padrão cada pagina vão ter 10 itens. Se você quer alterar a quantidade de itens é retornada por página, é só usar o parâmetro **“\_limit”**.  
No exemplo abaixo vou buscar a pagina 1 e quero 2 itens por página.

Dados de envio

GET http://localhost:3000/produtos/?\_page=1&\_limit=2

Content-Type: application/json

Retorno

[

{

"id": 2,

"nome": "X-Búrguer",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, salada e batata.",

"preco": 10.5,

"categoria\_id": 1

},

{

"id": 3,

"nome": "X-Bacon",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, 2 fatia de bacon, salada e batata.",

"preco": 12.5,

"categoria\_id": 1

}

]

**5 – Ordenação**

Para ordenar os dados, você pode usar os parâmetros **“\_sort”** para informar qual campo vai sofrer a ordenação e **“\_order”** para definir se ascendente ou descendente. Lembrando que é possível ordenar por mais de um campo.  
No exemplo abaixo vou ordenar os produtos por nome e em ordem descendente.

Dados de envio

GET http://localhost:3000/produtos/?\_sort=nome&\_order=desc

Content-Type: application/json

Retorno

[

{

"id": 4,

"nome": "X-Tudo",

"descricao": "Pão, 2 bifes de hambúrguer 90g, 2 fatias de queijo, 4 fatias de bacon, salada e batata.",

"preco": 14.5,

"categoria\_id": 1

},

{

"id": 2,

"nome": "X-Búrguer",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, salada e batata.",

"preco": 10.5,

"categoria\_id": 1

},

{

"id": 3,

"nome": "X-Bacon",

"descricao": "Pão, bife de hambúrguer 90g, 1 fatia de queijo, 2 fatia de bacon, salada e batata.",

"preco": 12.5,

"categoria\_id": 1

},

{

"nome": "Hambúrguer de frango",

"descricao": "Pão, bife de hambúrguer 90g de frango, salada e batata.",

"preco": 9.5,

"categoria\_id": 1,

"id": 7

},

{

"id": 6,

"nome": "Coca cola 600ml",

"descricao": "",

"preco": 7.5,

"categoria\_id": 2

},

{

"id": 5,

"nome": "Coca cola 350ml",

"descricao": "",

"preco": 5.5,

"categoria\_id": 2

}

]

* Limpar o app.component.html
* Criar um servisse: ng g s service/Produto
* Injetar após a criação

Abrir app.component.ts

Texto

Descrição gerada automaticamente

Declarar o método para obter todos os produtos (GET)

Texto

Descrição gerada automaticamente

Criar em produto.service.ts o método obterTodos ()

Sempre que fizer um **REQUEST é esperado um RESPONSE** é interessante criar uma interface para ajudar a reconhecer algo que esteja faltando.

Criar uma Interface

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Agora começamos o processo de criação do request

Pura injeção de dependência

Acrescente o conteúdo em produto.service.ts

Linhas 2 e 9 e 13

Texto

Descrição gerada automaticamente

Ao passar o mouse sobre o get vemos que ele possui parâmetros que devem ser passados. url, options, headers. Contexto, observe,etc... e 14 sobreescritas

Texto

Descrição gerada automaticamente

Para apontar para o endpoint get precisamos passar alguns parâmetros

1º é a url Get(URL) que obtemos olhando o endpoint do json-serve

Texto

Descrição gerada automaticamente

Vamos inserir este endereçamento em **enviroment.ts** na pasta enviroment

Criar uma constante chamada API\_PATH e passar o valor do endpoint

Tela preta com letras brancas

Descrição gerada automaticamente

Feito isto, lembre-se o primeiro parâmetro é a URL, passaremos nossa API\_PATH

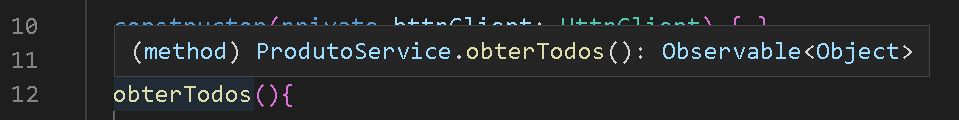
Observe que ao inserir as linhas 14 ele fará – ou deveria fazer – o import do API\_PATH na linha 1, caso não faça insira na mão...

Texto

Descrição gerada automaticamente

O uso de crases em get(`${API\_PATH}`) é chamado de template string (converter em texto o s valores passados por constantes ou variáveis) e o a tabela que desejo chamar...

obterTodos está retornando um observable



OBservables vs Promisses

Para transformar em Promisse, acrescente .toPromisse() ao final da linha (ele está descontinuado, iremos alterar para Observables mais a frente)

Texto

Descrição gerada automaticamente

Com isto ele retorna uma promisse

Interface gráfica do usuário, Texto, Site

Descrição gerada automaticamente

O que queremos é o retorno de um IProduto não de uma object... e neste caso ele traz um array... precisamos explicitar isto

Altere a linha 14 para

Ele retorna um array

Texto

Descrição gerada automaticamente

Adicioinar no app.module.ts o import do httpclient

Texto

Descrição gerada automaticamente

Fzer a chamada dentro do app.component.ts obterTodosProdutos(){}

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente

ObterPorId

Abrir produto.service.ts

Texto

Descrição gerada automaticamente

Abrir app.component.ts

Texto

Descrição gerada automaticamente

Abrir app.component.html

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Posso trazer apenas informações especificas

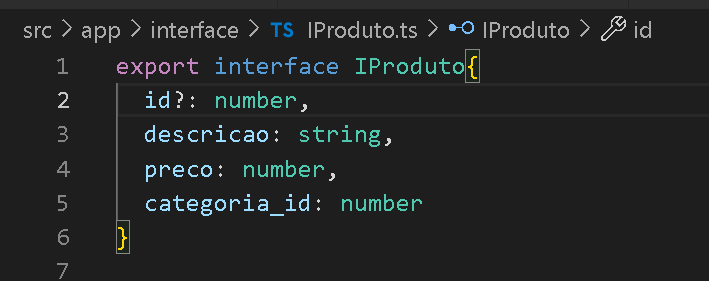
No app.component.ts

Texto

Descrição gerada automaticamente

Inserir um novo Produtos

Na Interface alterar o ID acrescentando um ? após o nome da propriedade indicando que não é obrigatório o seu uso. Linha 2

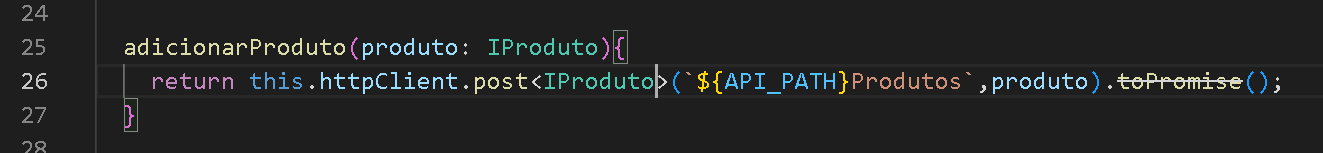


Abrir servisse produto.service.ts

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Vou criar um produto e ele tem que ser idêntico a Interface IProduto



Abrir app.component.ts

Chamar o método da servisse

Texto

Descrição gerada automaticamente

Chamar botão

App.component.html

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Clicando adicionar

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Após adicionar

Interface gráfica do usuário, Texto, Aplicativo, Word

Descrição gerada automaticamente

Atualizar dados PUT()

Produto.service.ts

Acrescentar método atualizarProduto

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Eu preciso mandar um objeto do tipo carro



Preciso passar o id que será alterado

No app.component.ts criar um método para atuializar

Texto

Descrição gerada automaticamente

Acrescentar a chamada pelo click no html

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Clicque no botão atualizar Produto

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Delete

Produto.service.ts

Texto

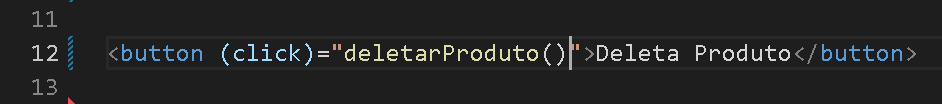
Descrição gerada automaticamente

App.component.ts

Interface gráfica do usuário, Texto

Descrição gerada automaticamente com confiança média

Acrescentar o botão para remover



Texto

Descrição gerada automaticamente