

Logo da Orange Talents

# Consumindo um endpoint REST com Micronaut de forma declarativa

## Antes de começar

Para complementar o curso, recomendamos estudar a documentação oficial do Micronaut sobre [Declarative HTTP Clients with @Client](https://docs.micronaut.io/latest/guide/index.html#clientAnnotation) caso você não tenho estudado ainda. Nela é possível ter uma idéia mais clara do poder e das possibilidades oferecidas pelo framework.

## Cenário:

Imagine que precisamos consumir uma API REST de um sistema terceiro de cartão de crédito no endereço https://api.de.terceiro.com/. Essa API em questão é responsável por bloquear um cartão de crédito de um cliente com base nas informações do cartão e titular.

Durante uma conversa com o time de desenvolvimento dessa empresa terceira formos informados que a API REST que queremos consumir se trata de uma API nova que foi construída as pressas para atender uma demanda do nosso time, por esse motivo ela ainda não possui documentação formal para compartilhar, como uma página em Swagger ou mesmo um PDF com os detalhes. Portanto, eles nos passaram via email um exemplo de requisição HTTP para consumirmos esta nova API REST exposta por eles:

POST /api/cartoes/82dd5f15-3e09-4aff-8844-84bac382e4b4/bloquear HTTP/1.1  
Host: api.de.terceiro.com  
Content-Type: application/json  
X-Request-Id: 4bdd3d64-f9fe-4ea2-9f64-9b96e96c75dc  
  
{  
 "titular": "Rafael Ponte",  
 "expiraEm": "2027-09"  
 "cvv": 123  
}

Eles também informaram que a resposta HTTP é bem simples, não possui um corpo na resposta e que os status são: - em caso de sucesso retorna status 200; - em caso de erro de validação retorna 400; - em caso de erro inesperado retorna status 500;

No exemplo acima da requisição HTTP temos a URI do endpoint, cabeçalhos e corpo da requisição HTTP. Repare que além dos dados do corpo da requisição temos também que informar o id do cartão contido na URI e o cabeçalho HTTP X-Request-Id durante o consumo da API. Entenda o X-Request-Id como um número aleatório e único que deve ser gerado por nossa aplicação a cada submissão.

Como você implementaria uma HTTP client declarativo com Micronaut para consumir a API REST na sua aplicação?

## O que seria bom ver nessa resposta?

* **Peso 6**: Declarar uma interface com um método passando os dados necessários para consumir a API e anotar a interface e método com as anotações do Micronaut para um HTTP client declarativo;
* **Peso 3**: Passar o request-id na assinatura do método e utilizar a anotação @Header para informar que se trata do cabeçalho X-Request-Id;
* **Peso 1**: Configurar o endereço do serviço no arquivo application.yml para flexibilizar a mudança em ambientes diferentes;

## Resposta do Especialista:

* Eu começaria fazendo um teste de consumo da API REST via POSTman ou Insomnia para ter certeza sobre os dados da requisição e da resposta, dessa forma eu evitaria surpresas implementando um HTTP client com Micronaut diretamente;
* Dado que eu aprendi a consumir a API, eu crio um HTTP client declarativo com Micronaut. Para isso, eu crio uma interface com um método para submeter a requisição com todos os dados necessários (incluindo o DTO do corpo da requisição) e não me preocupo muito com a resposta pois não há corpo:
* interface BloqueioDeCartaoClient {  
    
   fun bloquear(  
   requestId: UUID,   
   cartaoId: UUID,   
   request: BloqueioRequest): HttpResponse<Any>  
  }
* Em seguida eu anotaria a interface com as anotações do Micronaut para configurar um HTTP client declarativo e assim consumir a API. Alguns cabeçalhos não precisam ser informados no client pois o Micronaut já adiciona eles por padrão. Após anotar a interface, o código ficaria parecido com:
* @Client("https://api.de.terceiro.com/")  
  interface BloqueioDeCartaoClient {  
    
   @Post("/api/cartoes/{cartaoId}/bloquear")  
   fun bloquear(  
   @Header("X-Request-Id") requestId: UUID,   
   @PathVariable cartaoId: UUID,   
   @Body request: BloqueioRequest): HttpResponse<Any>  
  }
* Agora, eu levanto a aplicação para fazer alguns testes para ter certeza que o código acima funciona e está consumindo corretamente a API REST. Aproveito para habilitar os logs do Micronaut para verificar os dados de envio e recebimento da requisição. Em caso de erros eu vou corrigindo e me apoiando na documentação oficial do Micronaut;
* Com o HTTP client funcionando como esperado na aplicação, eu extraio o endereço da API REST para o arquivo application.yml da aplicação e passo-o como parâmetro para a anotação @Client("${enderecoDaApi}"). Isso é importante pois permite que a aplicação possa ser executada em diferentes ambientes bastando apenas mudar o endereço via variável de ambiente;
* Por fim, faço mais um teste para garantir que a configuração no application.yml está de fato funcionando;