

# Tutorial Swagger

Equipe: Felipe Braga, Raquel Moreira, Artur Alves, Matheus Motta e Caio Daniel.

---

## Introdução -

Olá, tudo bem? Este é um tutorial de como se deve usar o swagger editor para documentar uma API e uma breve documentação da documentação que nossa equipe fez de uma API.

Neste tutorial lhe será apresentado o passo a passo para você documentar sua API. Você verá diversas imagens de exemplo para conseguir acompanhar todo o processo.

## Aviso importante -

Neste tutorial não será necessário a instalação de nenhum recurso para o acompanhamento do tutorial a seguir. Mas, caso você deseje ter a ferramenta em sua máquina, também é possível.


## Primeiro-passo

- Primeira coisa que vamos fazer é acessar o "Swagger editor", então vamos acessar qualquer navegador e inserir o caminho "editor.swagger.io" na barra de navegação, reparem que já de cara ele mostra para gente um exemplo de Swagger, o "Petstore", do lado esquerdo da tela nós temos um script Swagger onde fala de versão, info, várias coisas e aqui do lado direito nós temos a documentação interativa que é gerada a partir desse script que nós estamos usando.



editor.swagger.io



 **Swagger Editor** Supported by SMARTBEAR File Edit Generate Server Generate Client

```
1 swagger: "2.0"
2 info:
3   description: "This is a sample server
4     Petstore server. You can find out more
5     about Swagger at [http://swagger.io]
6     (http://swagger.io) or on [irc.freenode
7     .net, #swagger](http://swagger.io/irc/).
8     For this sample, you can use the api
9     key `special-key` to test the
10    authorization filters."
11  version: "1.0.0"
12  title: "Swagger Petstore"
13  termsOfService: "http://swagger.io/terms/"
14  contact:
15    email: "apiteam@swagger.io"
16  license:
17    name: "Apache 2.0"
18    url: "http://www.apache.org/licenses
19    /LICENSE-2.0.html"
20 host: "petstore.swagger.io"
21 basePath: "/v2"
22 tags:
23   - name: "pet"
24     description: "Everything about your Pets"
25     externalDocs:
26       description: "Find out more"
27       url: "http://swagger.io"
28   - name: "store"
29     description: "Access to Petstore orders"
```

## Swagger Petstore 1.0.0

[ Base URL: [petstore.swagger.io/v2](http://petstore.swagger.io/v2) ]

This is a sample server Petstore server. You can find out more about Swagger at <http://swagger.io> or on [irc.freenode.net](http://irc.freenode.net), [#swagger](http://irc.freenode.net). For this sample, you can use the api key **special-key** to test the authorization filters.

[Terms of service](#)


[Contact the developer](#)

[Apache 2.0](#)

[Find out more about Swagger](#)

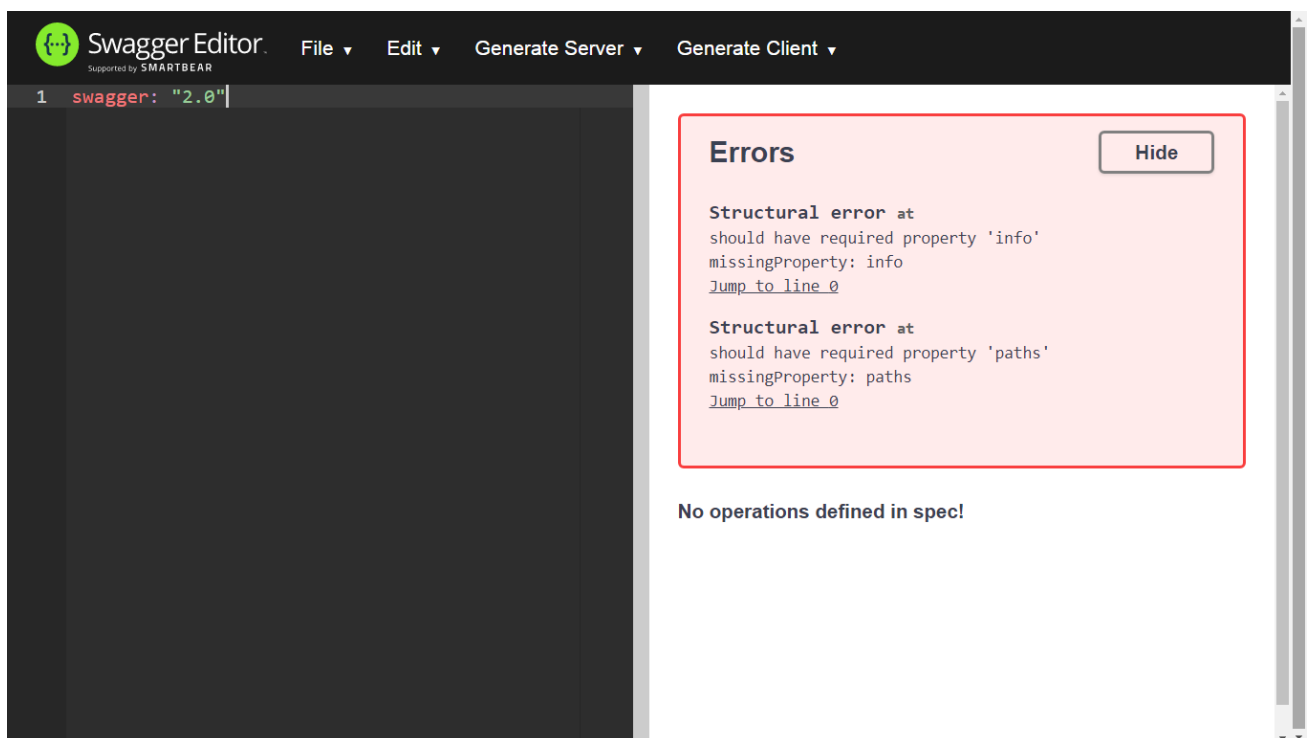
Schemes

HTTPS

Authorize 

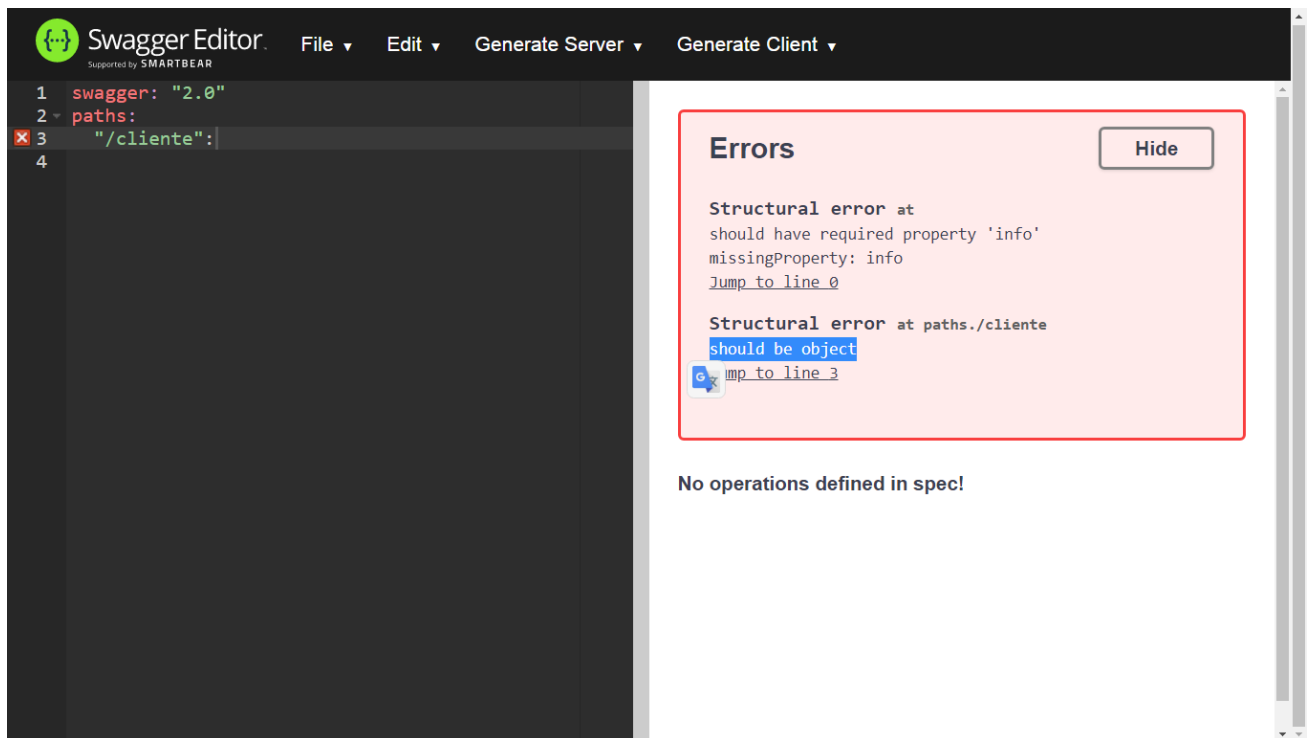
## Segundo-passo

- Agora, no lado esquerdo da tela, nós iremos zerar o script do "Petstore" e inserir nosso próprio script. Para isso, selecione o script todo pressionando "Ctrl + A" e depois pressione "Delete" para deletar o script ainda visível em sua tela. Após deletar o antigo script, vá no campo vazio do lado esquerdo e insira seu script que deseja utilizar. É necessário que no início você digite "swagger: '2.0' ou '3.0'" para definir a versão do swagger que você irá utilizar. Do lado direito da tela já é possível ver alguns erros, isso tudo acontece de forma interativa. Com isso, será necessário introduzir também a "info" e os "paths".



## Terceiro-passo

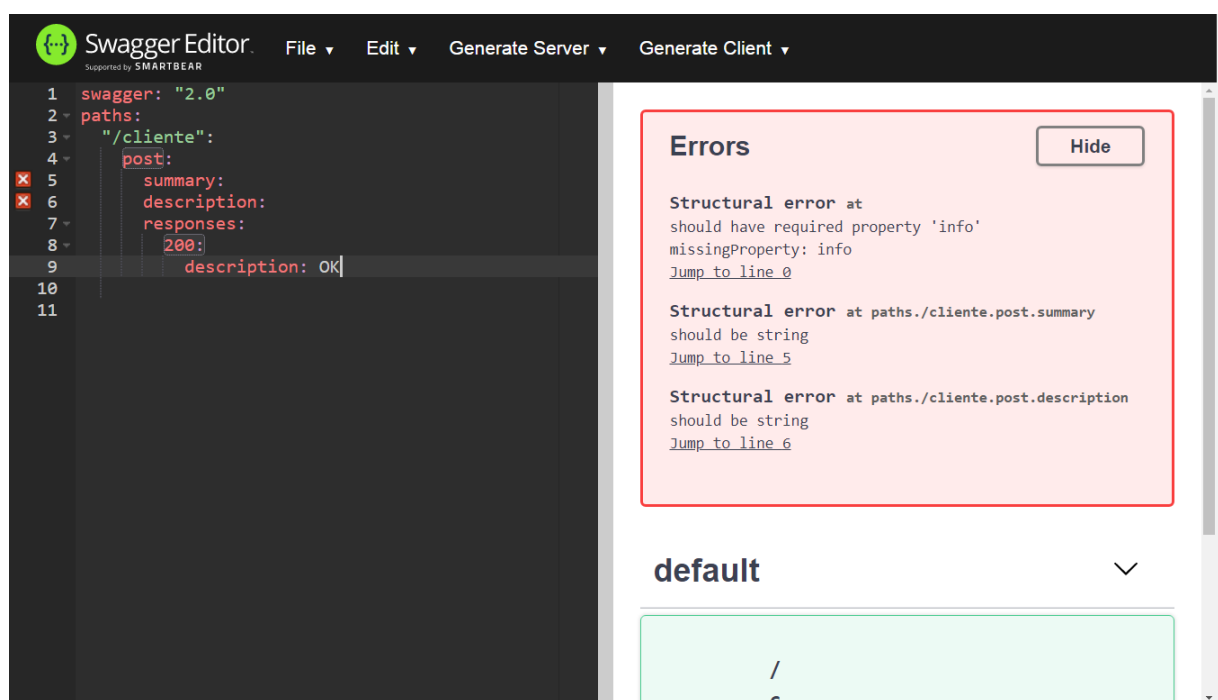
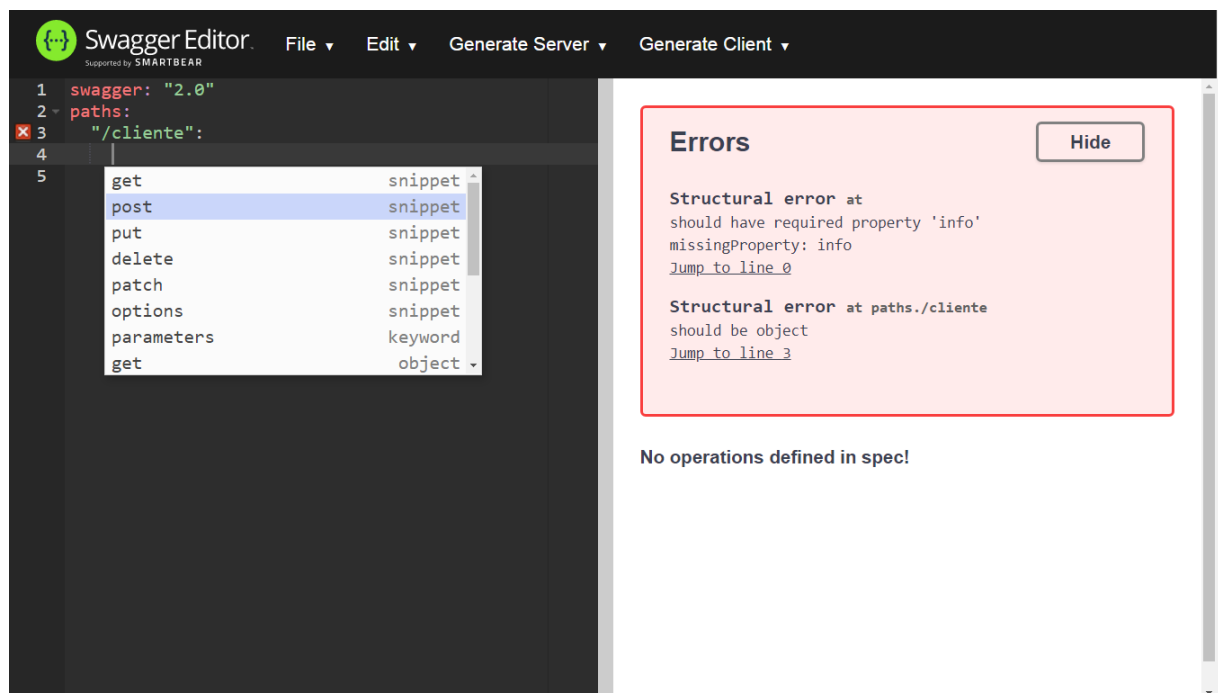
- O que é o path? São as operações que a nossa API vai oferecer, então digite "paths:" pressione "Enter" para próxima linha e um "Tab", então vá dando "Tab" para incluir uma nova propriedade dessa palavra chave. Para fazer essa primeira operação, vamos fazer primeiro o endereço, nós vamos fazer o cadastro de um cliente então aqui coloque "/cliente";



- Agora ele já deu um outro erro aqui se observamos do lado direito, ele está dizendo que a palavra /cliente, não é, e não pode ser um objeto, realmente, ele é só o nosso endereço, então vamos saltar mais uma linha dar um Tab, e agora o que vamos fazer? Vamos indicar qual é o método Rest que vamos utilizar, se vai ser o post, o get ou o put, no nosso caso vai ser o put, sempre que estivermos trabalhando com Rest vamos utilizar o post para fazer o cadastro de qualquer coisa que seja.

## Quarto-passo

- Se for fazer uma alteração já será outro método, então no editor aqui do swagger há a possibilidade também do 'auto complete', dê um "Ctrl + Espaço + Enter", repare que do lado esquerdo ele já indicou alguns métodos que podem ser usados, vamos clicar no "post" e ver como ele auto completa para nós. Da mesma forma é possível utilizar outros métodos como "git", "put", "delete" e alguns outros.



## API de exemplo –

- Acima lhe foi mostrado como iniciar um script no Swagger Editor. Agora irei lhe mostrar uma API que nós (equipe de Devs que montou este tutorial) consumimos e documentamos utilizando esta mesma ferramenta, inicialmente da mesma forma que foi apresentada anteriormente. Esta é uma API de moedas que exibe “ativos em alta” e “ativos em baixa”. Vamos averiguar como foi feita essa documentação.
- Olhando o script no mesmo momento que abrir, é possível ver algumas coisas, começando pela versão (Que é a mesma utilizada mais acima), schemes, host, info e algumas outras coisas.

The screenshot displays the Swagger Editor interface. On the left, a code editor shows the OpenAPI 2.0 definition for 'API Moeda'. The definition includes a base URL of 'localhost:3000', version '1.0.0', and a 'post' method for the '/ativosemAlta' endpoint. The right panel provides a visual overview of the API, including the title 'API Moeda 1.0.0', the base URL, a description, and links for terms of service, support, and license. A 'Schemes' dropdown menu is set to 'HTTP'.

```
1 swagger: '2.0'
2 schemes:
3   - http
4   - https
5 host: 'localhost:3000'
6 info:
7   version: 1.0.0
8   title: API Moeda
9   description: API que retorna os dados de ativos em
10    alta e em baixa.
11 termsOfService: 'http://localhost:3000'
12 contact:
13   name: Suporte
14   url: 'http://localhost:3000'
15   email: suporte@apimoeda.com
16 license:
17   name: MIT
18   url: 'http://opensource.org/licenses/MIT'
19 paths:
20   /ativosemAlta:
21     post:
22       tags:
23         - Métodos HTTP Ativos em alta
24       summary: Adiciona ativos em alta
25       description: Adiciona um ativos em alta
26       operationId: post
27       parameters:
28         - in: body
29           name: ativosemAlta
```

- Em “info” vão as informações importantes do script de consumo da nossa API, como por exemplo: O título que é exibido no lado direito da tela, a descrição e outras informações que são importantes para nosso script.
- Também é possível ver os “paths”, é onde vai todo o resto, como os métodos e as informações do consumo da API dentro deles.
- De primeira mão é possível ver o método “post” sendo inicializado na linha 20.

```

17     url: 'http://opensource.org/licenses/MIT'
18   paths:
19     /ativosemAlta:
20       post:
21         tags:
22           - Métodos HTTP Ativos em alta
23         summary: Adiciona ativos em alta
24         description: Adiciona um ativos em alta
25         operationId: post
26         parameters:
27           - in: body
28             name: ativosemAlta

```

- Olhe como o swagger responde de modo interativo no lado direito da tela:

The screenshot shows the Swagger UI interface. At the top, there's a section titled "Métodos HTTP Ativos em alta" with a dropdown arrow. Below it, there's a green box representing a POST endpoint: "POST /ativosemAlta Adiciona ativos em alta". Below that, there's another section titled "Métodos HTTP Ativos em alta" with a dropdown arrow. This section contains four colored boxes representing different HTTP methods: a green box for POST, a blue box for GET, an orange box for PUT, and a red box for DELETE. Each box contains the method, the path, and a brief description of the endpoint's function.

- Interessante, não é mesmo? Agora, mais abaixo eu irei mostrar como cada método foi chamado na documentação e como o swagger exibiu do lado direito da tela.

## Post –

Esquerda:

```
18 paths:
19   /ativosemAlta:
20     post:
21       tags:
22         - Métodos HTTP Ativos em alta
23       summary: Adiciona ativos em alta
24       description: Adiciona um ativos em alta
25       operationId: post
26       parameters:
27         - in: body
28           name: ativosemAlta
29           schema:
30             type: object
31             properties:
32               ativo:
33                 type: string
34                 example: VVAR3
35               ultimo:
36                 type: string
37                 example: '18,97'
38               varDia:
39                 type: integer
40                 example: 3.74
41               valMin:
42                 type: string
43                 example: '18,72'
44               valMax:
45                 type: string
46                 example: '19,75'
47               data:
48                 type: string
49                 example: '15:09 19/08'
50               id:
51                 type: integer
52                 example: 1
53       consumes:
54         - application/json
55       produces:
56         - application/json
57       responses:
58         '200':
59           description: Atualização efetuada com sucesso
60           schema:
61             properties:
62               ativo:
```



Direita:

**POST** **/ativosemAlta** Adiciona ativos em alta

Adiciona um ativos em alta

Parameters

Try it out

Name	Description
ativosemAlta	
object (body)	<div>Example Value   Model</div> <pre>{   "ativo": "VVAR3",   "ultimo": "18,97",   "varDia": 3.74,   "valMin": "18,72",   "valMax": "19,75",   "data": "15:09 19/08",   "id": 1 }</pre> <div>Parameter content type application/json</div>

Responses

Response content type application/json

Code	Description
200	<div>Atualização efetuada com sucesso</div> <div>Example Value   Model</div> <pre>{   "ativo": "VVAR3" }</pre>

## Get –

Esquerda:

```
87  get:
88    tags:
89      - Métodos HTTP Ativos em alta
90    summary: Requisita ativos em alta
91    description: Requisita os ativos em alta
92    operationId: get
93    parameters:
94      - in: body
95        name: ativosemAlta
96        schema:
97          type: array
98          items:
99            type: object
100           properties:
101             ativo:
102               type: string
103             ultimo:
104               type: string
105             varDia:
106               type: integer
107             valMin:
108               type: string
109             valMax:
110               type: string
111             data:
112               type: string
113             id:
114               type: integer
115           example:
116             - ativo: VVAR3
117               ultimo: '18,97'
118               varDia: 3.74
119               valMin: '18,72'
120               valMax: '19,75'
121               data: '15:09 19/08'
122               id: 1
123             - ativo: AAAAAAA
124               ultimo: '77,77'
125               varDia: '77,77'
126               valMin: '77,77'
127               valMax: '77,77'
128               data: null
129               id: 2
130    consumes:
131      - application/json
132    produces:
133      - application/json
134    responses:
```

Direita:

**GET** **/ativoemAlta** Requisita ativos em alta

Requisita os ativos em alta

Parameters

Try it out

Name	Description
ativoemAlta	Example Value   Model
array[object] (body)	<pre>[   {     "ativo": "VVAR3",     "ultimo": "18,97",     "varDia": 3.74,     "valMin": "18,72",     "valMax": "19,75",     "data": "15:09 19/08",     "id": 1   },   {     "ativo": "AAAAAAA",     "ultimo": "77,77",     "varDia": "77,77",     "valMin": "77,77",     "valMax": "77,77",     "data": null,     "id": 2   } ]</pre>

Parameter content type

application/json

application/json

Responses

Response content type

application/json

## Put –

Esquerda:

```
175 put:
176   tags:
177     - Métodos HTTP Ativos em alta
178   summary: Atualiza um ativo em alta
179   description: Atualiza um ativo em alta
180   operationId: put
181   parameters:
182     - in: body
183       name: ativosemAlta
184       schema:
185         type: object
186         properties:
187           ativo:
188             type: string
189             example: WVAR3
190           ultimo:
191             type: string
192             example: '18,97'
193           varDia:
194             type: integer
195             example: 3.74
196           valMin:
197             type: string
198             example: '18,72'
199           valMax:
200             type: string
201             example: '19,75'
202           data:
203             type: string
204             example: '15:09 19/08'
205           id:
206             type: integer
207             example: 1
208   consumes:
209     - application/json
210   produces:
211     - application/json
212   responses:
213     '200':
214       description: Ativo em alta atualizado
215       schema:
216         type: object
217         properties:
218           ativo:
219             type: string
220             example: WVAR3
221           ultimo:
222             type: string
```

Direita:

**PUT** **/ativosemAlta** Atualiza um ativo em alta

Atualiza um ativo em alta

Parameters

Try it out

Name	Description
ativosemAlta	Example Value   Model
object (body)	<pre>{   "ativo": "VVAR3",   "ultimo": "18,97",   "varDia": 3.74,   "valMin": "18,72",   "valMax": "19,75",   "data": "15:09 19/08",   "id": 1 }</pre>
	Parameter content type application/json

Responses

Response content type application/json

Code	Description
200	Ativo em alta atualizado
	Example Value   Model
	<pre>{   "ativo": "VVAR3"</pre>

## Delete –

Esquerda:

```
243 ▾    '/ativoemAlta/{id}':
244 ▾        delete:
245 ▾            tags:
246 ▾                - Métodos HTTP Ativos em alta
247 ▾            summary: Deletar um ativo em alta
248 ▾            description: Deletando um ativo em alta
249 ▾            operationId: delete
250 ▾        produces:
251 ▾            - application/xml
252 ▾            - application/json
253 ▾        parameters:
254 ▾            - name: id
255 ▾              in: path
256 ▾              required: true
257 ▾              type: integer
258 ▾        responses:
259 ▾            '200':
260 ▾                description: Ativo deletado com sucesso
261 ▾            '400':
262 ▾                description: Requisição inválida
263 ▾            '500':
264 ▾                description: Erro interno no servidor
```

Direita:

**DELETE** /ativosemAlta/{id} Deletar um ativo em alta

Deletando um ativo em alta

Parameters

Cancel

Name	Description
<b>id</b> * required integer (path)	<input type="text" value="2"/>

Execute

Clear

Responses

Response content type application/xml

Curl

```
curl -X DELETE "http://localhost:3000/ativosemAlta/2" -H "accept: application/xml"
```

Request URL

```
http://localhost:3000/ativosemAlta/2
```

Server response

Code	Details
Undocumented	TypeError: Failed to fetch

Responses

Code	Description
------	-------------

## Último exemplo –

- Nos exemplos dados anteriormente, os prints exibiram o resultado do path “ativosemAlta” como você pode perceber. Há apenas dois paths, o “ativosemAlta” e o “ativosemBaixa”,



abaixo eu irei mostrar como ficou o resultado do path “ativosemBaixa”, é basicamente a mesma coisa que o outro path, só é alterado coisas como o nome da tag e etc.

```
265 ▾ /ativosemBaixa:
266 ▾   post:
267 ▾     tags:
268     - Métodos HTTP Ativos em baixa
269     summary: Adiciona ativos em Baixa
270     description: Adiciona um ativo em baixa
271     operationId: Post
272 ▾   parameters:
273 ▾     - in: body
274       name: ativosemBaixa
275 ▾     schema:
```

## Métodos HTTP Ativos em baixa

**POST** /ativosemBaixa Adiciona ativos em Baixa

**GET** /ativosemBaixa Requisita ativos em baixa

**PUT** /ativosemBaixa Atualiza um ativo em baixa

**DELETE** /ativosemBaixa/{id} Deletar um ativo em baixa

## Conclusão –

- Neste tutorial nós aprendemos a acessar o swagger, aprendemos um pouco sobre como criar um script, documentá-lo e como nosso script deverá aparecer na tela caso não seja encontrado nenhum erro.



---

## Referências –

- <https://www.alura.com.br/conteudo/swagger-crie-uma-documentacao-rest>
- <http://www.matera.com/blog/post/swagger-como-gerar-uma-documentacao-interativa-para-api-rest>