

## Atividade 4 – API RESTFull

### SISTEMAS DISTRIBUÍDOS E MOBILE

**Joinville 06 de junho de 2023**

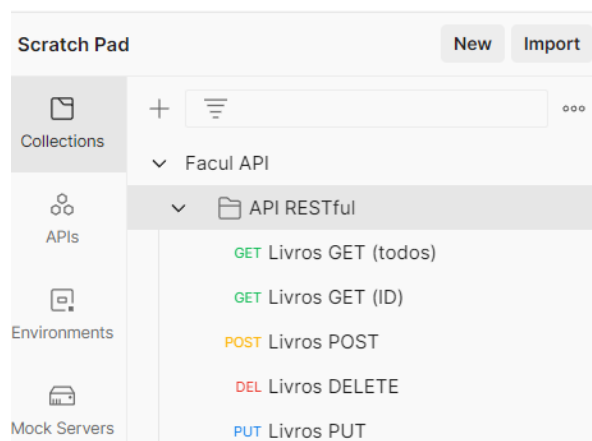
**Professora:** Marisangila Alves.

**Alunos:** Éverton Gambeta.

Jean Carlos Severino.

Criou-se um programa em python de um catálogo de livros onde pode ser listado os livros existentes (GET), listado por ID (GET), atualizados (PUT), inserir novos livros (POST) e excluídos por ID (DELETE). Para facilitar a manipulação desses registros foi utilizado o Postman.

Sendo assim, criei as cinco operações que vamos utilizar:



Código criado no **Visual Studio Code**:



Para a interação com a API foi utilizado o software **Postman**:



## Código pronto realizado em Python:

```
catalogo_de_livros_server.py X
C: > Users > evert > OneDrive > Área de Trabalho > Everton_Gambeta > catalogo_de_livros_server.py > get_livro_por_id

1  from flask import Flask, request, jsonify
2
3  app = Flask(__name__)
4
5  # Lista de Livros
6  livros = [{
7
8      'id': 1,
9      'titulo': "O Senhor dos Anéis - A Sociedades do Anel",
10     'autor': "J.R.R. Tolkien",
11
12     },
13     {
14
15         'id': 2,
16         'titulo': "Harry Potter e a Prda Filosofal",
17         'autor': "J.K. Howling",
18
19     },
20     {
21
22         'id': 3,
23         'titulo': "Interestelar",
24         'autor': "JChristopher Nolan",
25
26     },
27 ]
28 # Rota para obter todos os LIVROS
29 @app.route('/livros', methods=['GET'])
30 def get_livros():
31     return jsonify(livros), 200
32
33 # Rota para obter um LIVROS pelo ID
34 @app.route('/livros/<int:id>', methods=['GET'])
35 def get_livro_por_id(id):
36     for livro in livros:
37         if livro.get('id') == id:
38             return jsonify(livro), 200
39     return jsonify({'message': 'Este LIVRO não foi encontrado através desta ID'}), 404
40
41 # Rota para adicionar um novo LIVRO
42 @app.route('/livros', methods=['POST'])
43 def add_livro():
44     new_livro = {
45         'id': len(livros) + 1,
46         'titulo': request.json['titulo'],
47         'autor': request.json['autor']
48     }
49     livros.append(new_livro)
50     return jsonify(new_livro), 201
51
52 # Rota para atualizar um LIVRO existente
53 @app.route('/livros/<int:livro_id>', methods=['PUT'])
54 def update_client(livro_id):
55     for livro in livros:
56         if livro['id'] == livro_id:
57             livro['titulo'] = request.json['titulo']
58             livro['autor'] = request.json['autor']
59             return jsonify(livro), 200
60     return jsonify({'message': 'Este LIVRO não foi encontrado para poder Atualizar'}), 404
61
62 # Rota para excluir um LIVRO
63 @app.route('/livros/<int:livro_id>', methods=['DELETE'])
64 def delete_livro(livro_id):
65     for livro in livros:
66         if livro['id'] == livro_id:
67             livros.remove(livro), 200
68             return jsonify({'message': 'Livro deleted'})
69     return jsonify({'message': 'Este LIVRO não foi encontrado para poder Excluir'}), 404
70
71 if __name__ == '__main__':
72     app.run(debug=True)
```

Primeiramente vamos rodar o programa:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

* Serving Flask app 'catalogo_de_livros_server'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 661-841-506
█
```

## 1 – Listar todos os Livros (GET)

Previamente cadastrei três livros conforme imagem do código a seguir, desta forma vou utilizar o método GET para listar todos.

```
catalogo_de_livros_server.py X
C: > Users > evert > OneDrive > Área de Trabalho > Everton_Gambeta > catalogo_de_livros_server.py > add_livro

1  from flask import Flask, request, jsonify
2
3  app = Flask(__name__)
4
5  # Lista de Livros
6  livros = [{
7
8      'id': 1,
9      'titulo': "O Senhor dos Anéis - A Sociedades do Anel",
10     'autor': "J.R.R. Tolkien",
11
12     },
13     {
14
15         'id': 2,
16         'titulo': "Harry Potter e a Prdra Filosofal",
17         'autor': "J.K. Howling",
18
19     },
20     {
21
22         'id': 3,
23         'titulo': "Interestelar",
24         'autor': "JChristopher Nolan",
25
26     },
27 ]
28 # Rota para obter todos os LIVROS
29 @app.route('/livros', methods=['GET'])
30 def get_livros():
31     return jsonify(livros), 200
```

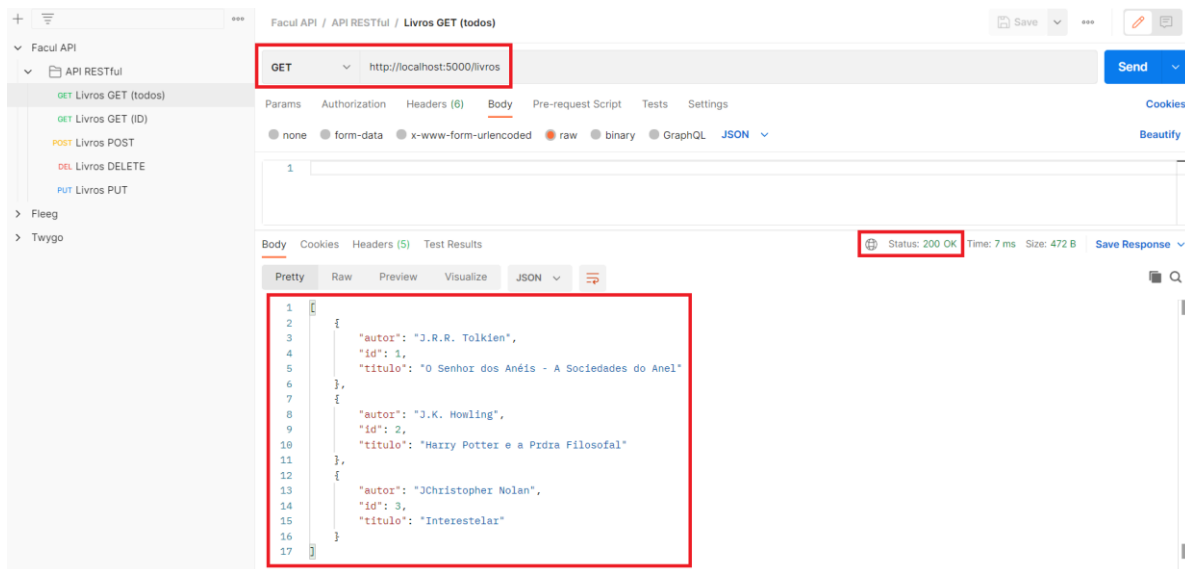
Utilizando o Postman:

**Método:** GET

**URL:** <http://localhost:5000/livros>

**Status:** 200 OK

**Resultado:** Body



Retorno no terminal também:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

* Debug mode: on
WARNING: This is a development server. Do not use it in a production
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 661-841-506
127.0.0.1 - - [04/Jun/2023 17:44:15] "GET /livros HTTP/1.1" 200 -
```

## 2 – Listar apenas um Livro (GET)

Código:

```
33 # Rota para obter um LIVROS pelo ID
34 @app.route('/livros/<int:id>', methods=['GET'])
35 def get_livro_por_id(id):
36     for livro in livros:
37         if livro.get('id') == id:
38             return jsonify(livro), 200
39     return jsonify({'message': 'Este LIVRO não foi encontrado através desta ID'}), 404
```

Utilizando o Postman:

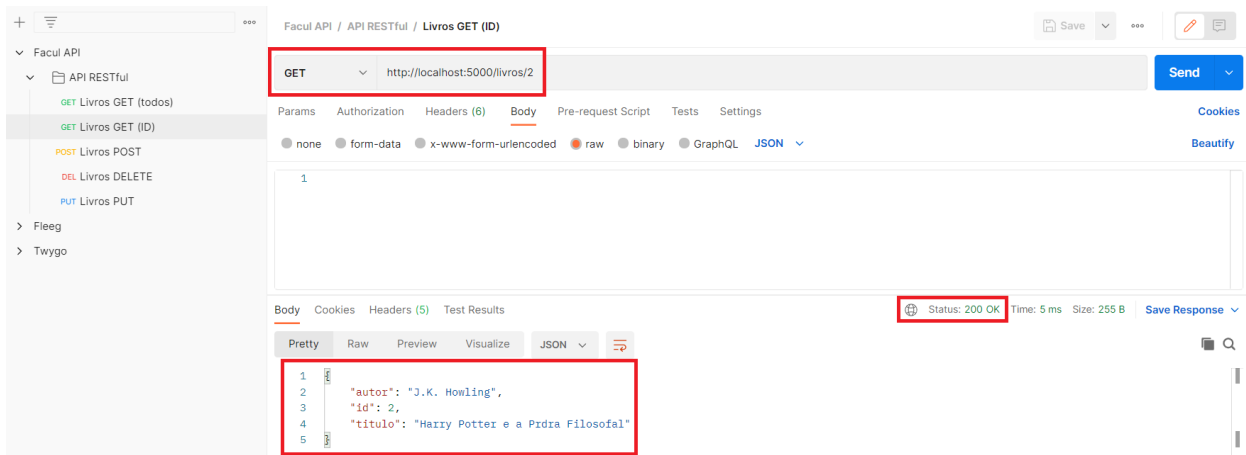
**Método:** GET

**URL:** <http://localhost:5000/livros/2>

**Status:** 200 OK

**Resultado:** Body

**Obs:** O /2 colocado no final da URL representa a ID que deve ser mostrada.



### 3 – Adicionar um novo Livro (POST)

Código:

```
41 # Rota para adicionar um novo LIVRO
42 @app.route('/livros', methods=['POST'])
43 def add_livro():
44     new_livro = {
45         'id': len(livros) + 1,
46         'titulo': request.json['titulo'],
47         'autor': request.json['autor']
48     }
49     livros.append(new_livro)
50     return jsonify(new_livro), 201
```

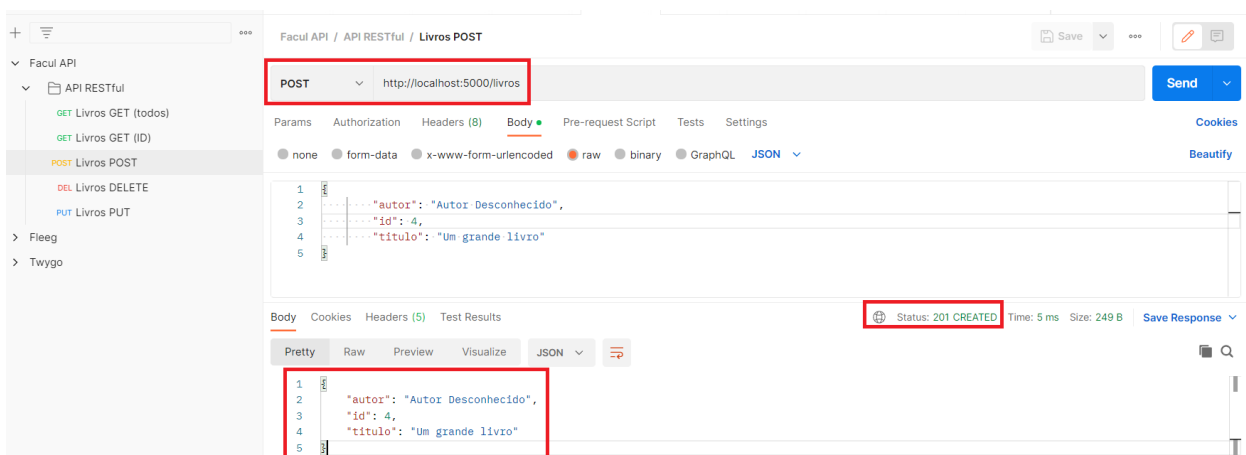
Utilizando o Postman:

**Método:** POST

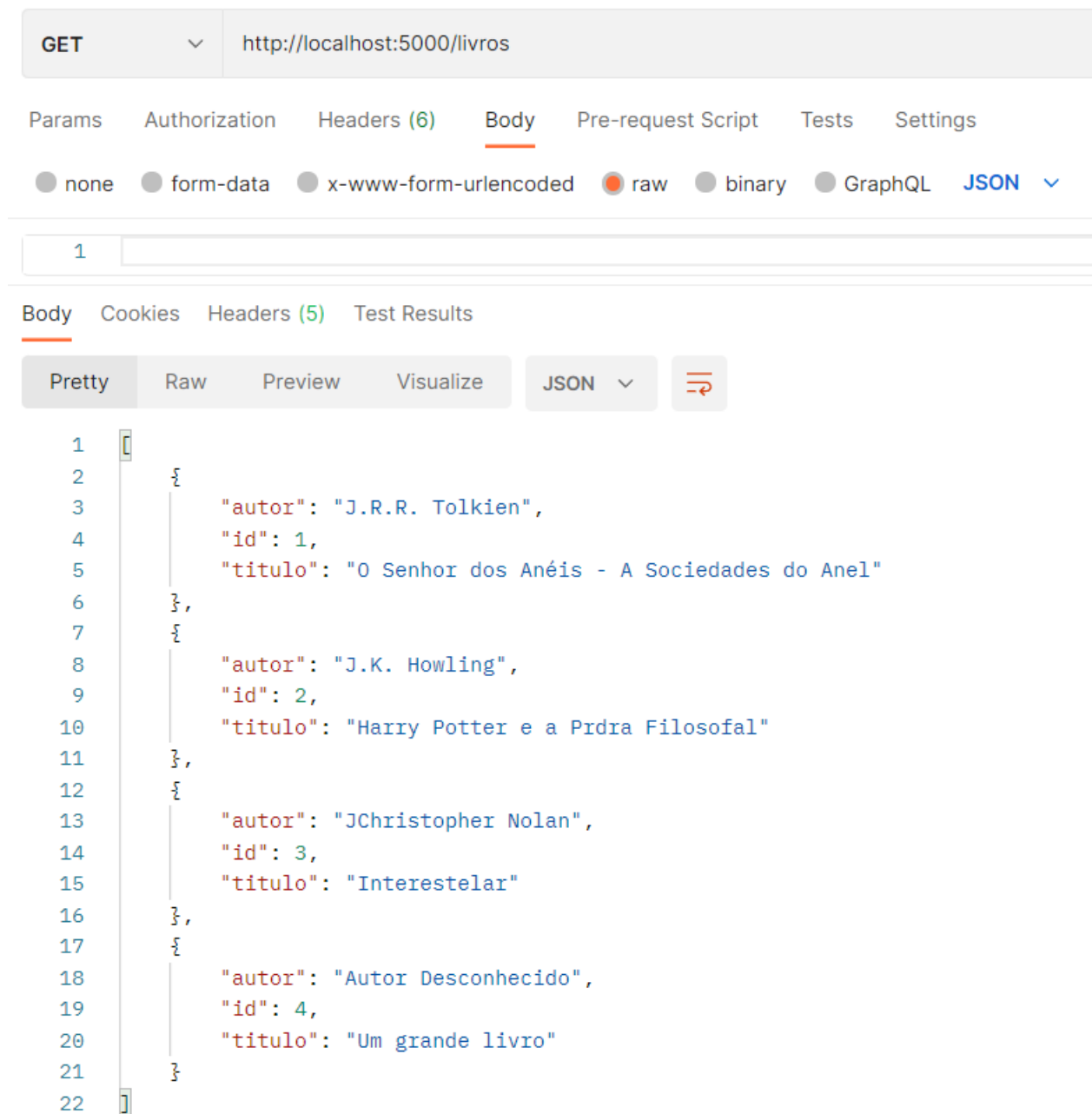
**URL:** <http://localhost:5000/livros>

**Status:** 201 CREATED

**Resultado:** Body



Para confirmar essa criação eu utilizo um GET novamente para listar e ver que foi acrescentado o item 4:



GET http://localhost:5000/livros

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** v

1

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize **JSON** v ↺

```
1 [
2   {
3     "autor": "J.R.R. Tolkien",
4     "id": 1,
5     "titulo": "O Senhor dos Anéis - A Sociedades do Anel"
6   },
7   {
8     "autor": "J.K. Howling",
9     "id": 2,
10    "titulo": "Harry Potter e a Prdra Filosofal"
11  },
12  {
13    "autor": "JChristopher Nolan",
14    "id": 3,
15    "titulo": "Interestelar"
16  },
17  {
18    "autor": "Autor Desconhecido",
19    "id": 4,
20    "titulo": "Um grande livro"
21  }
22 ]
```

#### 4 – Atualizar o registro de um Livro (PUT)

Código:

```
52 # Rota para atualizar um LIVRO existente
53 @app.route('/livros/<int:livro_id>', methods=['PUT'])
54 def update_client(livro_id):
55     for livro in livros:
56         if livro['id'] == livro_id:
57             livro['titulo'] = request.json['titulo']
58             livro['autor'] = request.json['autor']
59             return jsonify(livro), 200
60     return jsonify({'message': 'Este LIVRO não foi encontrado para poder Atualizar'}), 404
```

Utilizando o Postman:

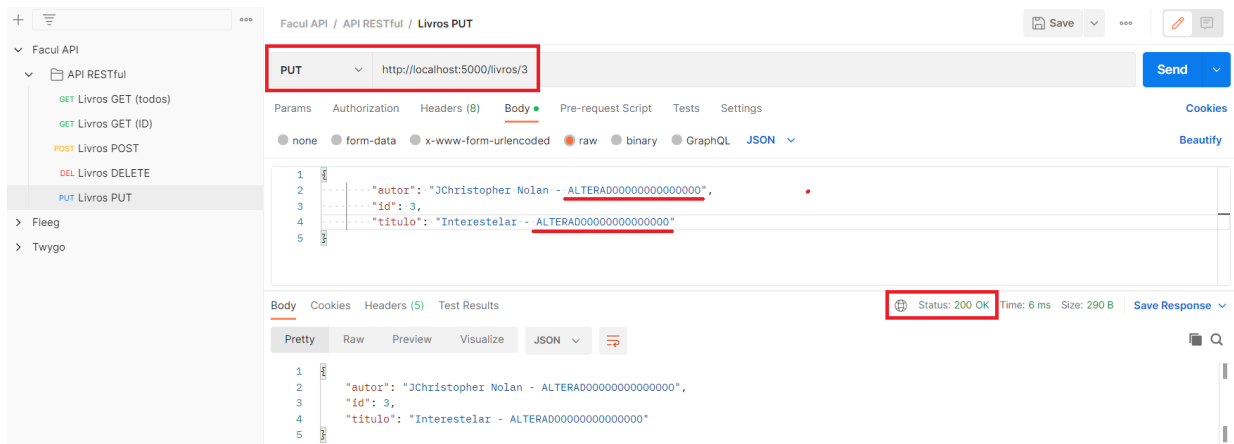
**Método:** PUT

**URL:** <http://localhost:5000/livros/3>

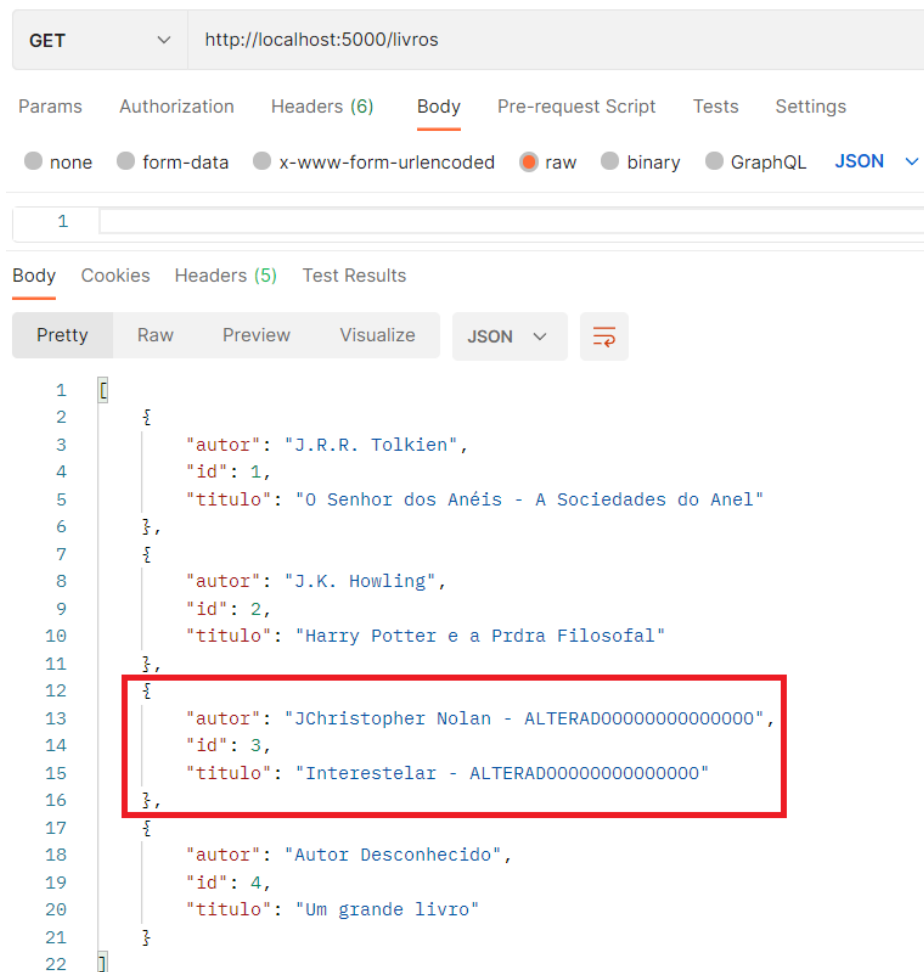
**Status:** 200 OK

**Resultado:** Body

**Obs:** O /3 colocado no final da URL representa a ID que deve ser alterada.



Listar novamente (GET) para conferir a edição realizada:



## 4 – Excluir um livro (DELETE)

Código:

```
62 # Rota para excluir um LIVRO
63 @app.route('/livros/<int:livro_id>', methods=['DELETE'])
64 def delete_livro(livro_id):
65     for livro in livros:
66         if livro['id'] == livro_id:
67             livros.remove(livro), 200
68             return jsonify({'message': 'Livro deleted'})
69     return jsonify({'message': 'Este LIVRO não foi encontrado para poder Excluir'}), 404
```

Utilizando o Postman:

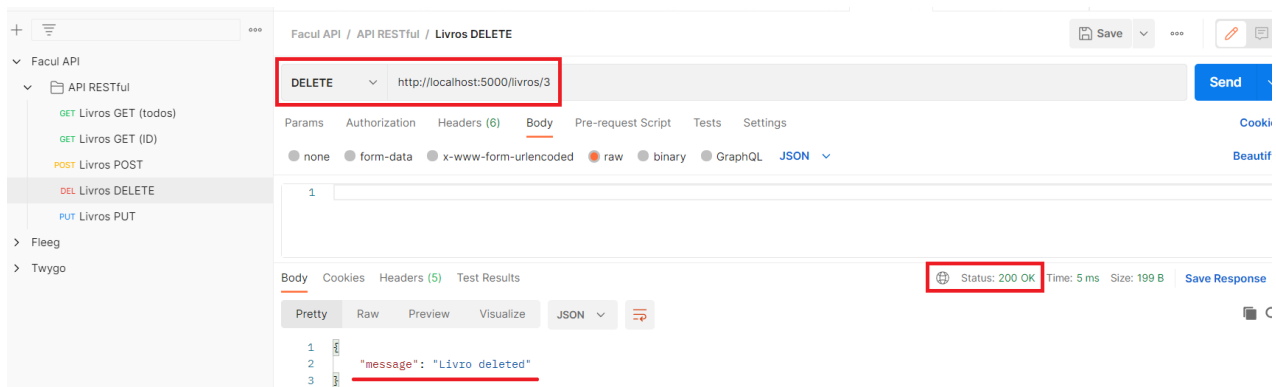
**Método:** DELETE

**URL:** <http://localhost:5000/livros/3>

**Status:** 200 OK

**Resultado:** Body (“Livro deleted”).

**Obs:** O /3 colocado no final da URL representa a ID que deve ser excluída.



Lembrando que os resultados também são mostrados no terminal:

