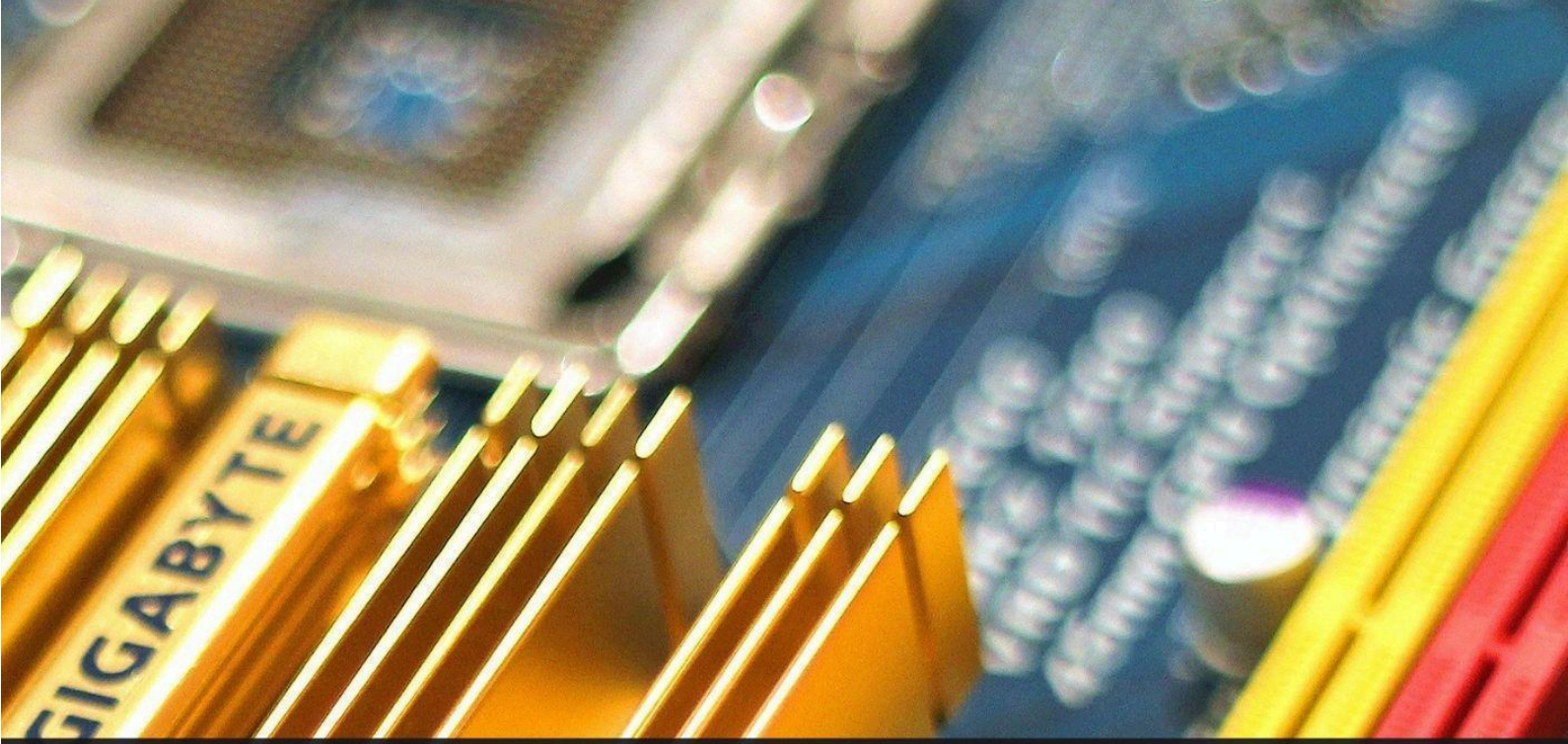
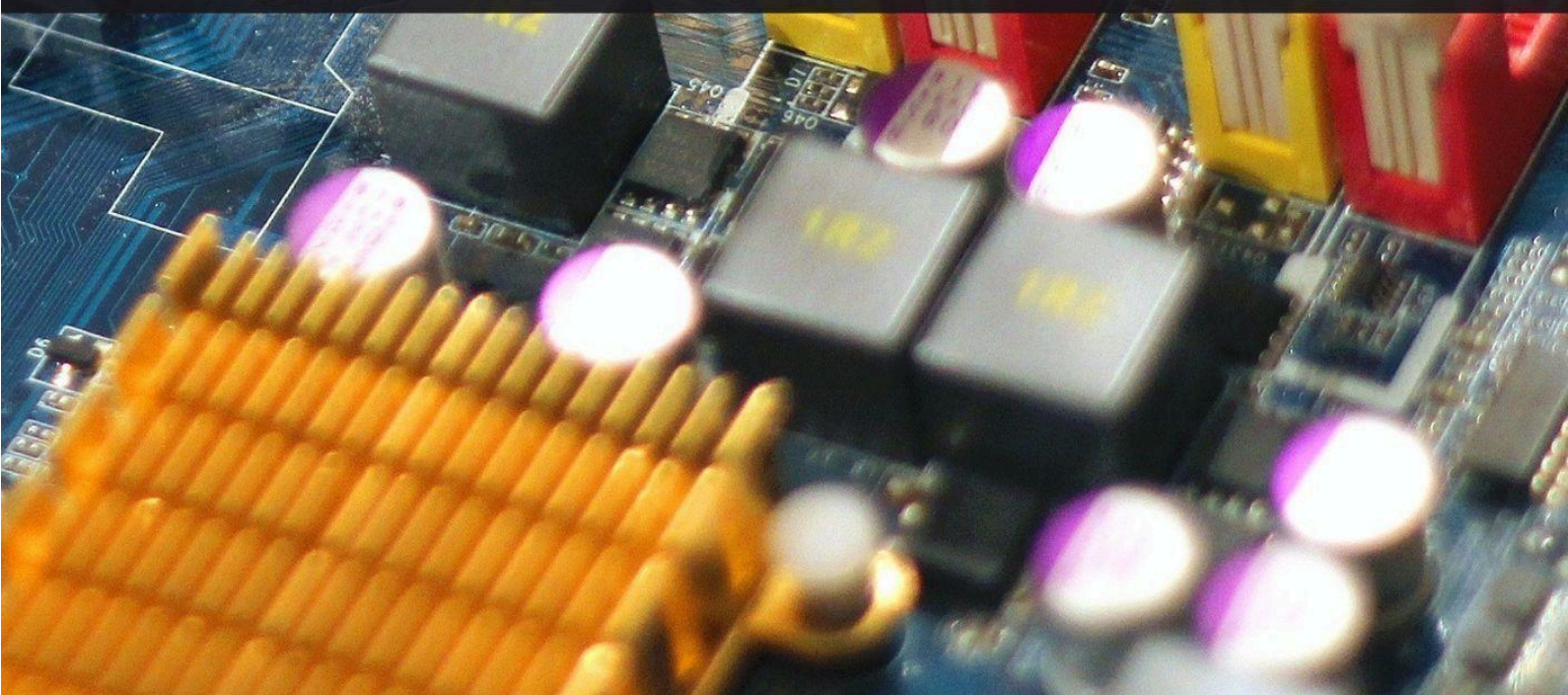


DESENVOLVIMENTO DE APIs E MICROSSERVIÇOS



12

Aprofundamento em sistemas distribuídos

Andréia Cristina dos Santos Gusmão

Resumo

Para finalizar nossa disciplina, vamos conhecer o 'ngrok', que permite a comunicação do servidor flask em outros computadores da rede ou até mesmo pelo serviço https, em um celular, por exemplo.

12.1. Servidor em Flash executando localmente

Como já vimos em aulas anteriores, podemos facilmente criar um servidor em flask para ser executado na máquina local (**localhost**).

Considere como exemplo, o código do arquivo aula12_exemplo1.py na Figura 12.1.

Figura 12.1. Arquivo aula12_exemplo1.py.

```
from flask import Flask, request

app = Flask(__name__)

@app.route('/', methods=['POST'])
def boasVindas():
  nome = request.get_json(force=True)
  return 'Seja bem-vindo, {}, na aula de Desenvolvimento de APIs e Microsservi
ços.\n'.format(nome['nome'])

if __name__ == '__main__':
    app.run(host = 'localhost', port = 5002, debug = True)
```

Fonte: do autor, 2022.

A única diferença que pode ser vista neste código, em relação aos demais já estudados, é na linha 7 da Figura 12.1 nome = request.get_json(force=True), uma outra forma de transformar dados recebidos pelo POST em dicionário. Embora estamos utilizando o método POST, vamos pegar o valor digitado pelo usuário e apresentar na tela, sem salvar em um arquivo, dicionário, database.

No *prompt de comando*, acessamos a pasta onde está nosso arquivo de código e digitamos python aula12_exemplo.py, conforme mostra a Figura 12.2.

Figura 12.2. Execução do arquivo aula12_exemplo.py.

```
C:\dam>python aula12_exemplo1.py
 * Serving Flask app "aula12_exemplo1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 119-118-421
 * Running on http://localhost:5002/ (Press CTRL+C to quit)
```

Fonte: do autor, 2022.

Enviando requisição à API

Um outro modo ainda não visto em aula, é enviar requisição abrindo um novo terminal do *prompt de comando*.

No segundo terminal digitamos:

curl -H \Content-Type:application/json\ -X POST --data "{\"nome\":\"Andreia\"}" http://localhost:5002/

Explicando cada parte:

curl -H \Content-Type:application/json\ -X: padrão de execução

POST: verbo/método que iremos executar

--data: padrão quando temos dados de corpo de requisição

"{\"nome\":\"Andreia\"}": "nome" é a variável (linha 7 da Figura 12.1) que irá receber o valor do corpo da requisição, que aqui é "Andreia"

http://localhost:5002/: endereço do servidor que está sendo executado (de acordo com a Figura 12.2).

Figura 12.3. Requisição enviada no segundo terminal: aula12_exemplo1py.

```
C:\dam>curl -H \Content-Type:application/json\ -X POST --data "{\"nome\":\"Andreia\"}"
http://localhost:5002/

Seja bem-vindo, Andreia, na aula de Desenvolvimento de APIs e Microsserviços.
```

Fonte: do autor, 2022.

O comando apresentado na Figura 12.3 é para execução no Windows, no Linux, o comando é: curl -H \Content-Type: application/json\ -X POST --d "{"nome":"Andreia"}" http://localhost:5002/. Na figura 12.4, mostramos o segundo terminal, que mostra o resultado da execução.

Figura 12.4. Resultado da execução do arquivo aula12_exemplo1.py.

```
C:\dam>curl -H \Content-Type:application/json\ -X POST --data "{ \"nome\":\"Andreia\"}"
http://localhost:5002/

Seja bem vindo, Andreia, na aula de Desenvolvimento de APIS e Microsserviços
```

Fonte: do autor, 2022.

Bom, e com o nosso exemplo da aula passada, com MVC, como será que fica a execução em dois terminais? É possível, com uma pequena alteração. E essa alteração é apenas no arquivo sala_aula_server.py, não vamos alterar nada em controller e nem em model. Esse arquivo é o que estudamos na aula passada.

Exemplo inserir aluno, do código em MVC

Na função def inserir() do arquivo mvc/sala_aula_server.py alteramos a linha aluno = request.jon para aluno = request.get_json(force=True).

```
@app.route("/alunos", methods=["POST"])
def inserir():
  aluno = request.get_json(force=True)
  return aluno_controller.inserirAluno(aluno)
```

Executamos no primeiro terminal do *prompt de comando* python sala_aula_server.py (Figura 12.5).

Figura 12.5. Execução do arquivo sala_aula_server.py.

```
C:\dam\mvc_aula12\mvc>python sala_aula_server.py
 * Serving Flask app "sala_aula_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 119-118-421
 * Running on http://localhost:5002/ (Press CTRL+C to quit)
```

Fonte: do autor, 2022.

E no segundo terminal: curl -H \Content-Type:application/json\ -X POST --data "{\"id\":5, \"media\":8, \"nome\":\"Aluno Teste\"}" http://localhost:5002/alunos.

"{\"id\":5, \"media\":8, \"nome\":\"Aluno Teste\"}" são os dados de um novo aluno que queremos incluir no database.

Figura 12.6. Resultado da execução do arquivo sala_aula_server.py para o método POST no segundo terminal.

```
C:\dam>curl -H \Content-Type:application/json\ -X POST --data "{\"id\":5, \"media\":8,
\"nome\":\"Aluno Teste\"}" http://localhost:5002/alunos
[
  {
   "id": 1,
   "media": 8.5,
   "nome": "Andreia"
  },
  {
   "id": 2,
   "media": 10,
   "nome": "Arthur"
  },
  {
   "id": 3,
   "media": 10,
   "nome": "Pedro"
  },
  {
   "id": 4,
   "media": 7,
   "nome": "Ana"
  },
  {
   "id": 5,
   "media": 8,
   "nome": "Aluno Teste"
  }
]
```

Fonte: do autor, 2022.

Observe na Figura 12.6, que após a execução, o aluno de id = 5 foi inserido e a lista de alunos atualizada no terminal.

Exemplo alterar aluno, do código em MVC

Para alterar, assim como já apresentamos para inserir, só vamos alterar uma linha na função def alterar(): aluno = request.get_json(force=True).

```
@app.route("/alunos/<int:id_alterar>", methods=["PUT"])
def alterar(id_alterar):
  aluno = request.get_json(force=True)
  return aluno_controller.alterarAluno(id_alterar, aluno)
```

O teste será feito com dois terminais (Figura 12.7), em que o primeiro é onde digitamos python sala-aula_server.py No segundo terminal, vamos digitar o código de enviar a requisição para alteração: C:\dam>curl -H \Content-Type:application/json\ -X PUT --data "{\"id\":5, \"media\":10, \"nome\":\"Heitor\"}" http://localhost:5002/alunos/5.

http://localhost:5002/alunos/5 quer dizer que estamos passando o parâmetro 5, ou seja, vamos alterar (PUT) o aluno com id = 5 e os novos dados serão: id = 5, media = 10 e nome = Heitor.

Figura 12.7. Resultado da execução do arquivo sala_aula_server.py para o método PUT no segundo terminal.

```
C:\dam>curl -H \Content-Type:application/json\ -X PUT --data "{\"id\":5, \"media\":10, \"nome\":\"Heitor\"}"
http://localhost:5002/alunos/5
[
  {
    "id": 5,
    "media": 10,
    "nome": "Heitor"
  }
]
```

Fonte: do autor, 2022.

Na Figura 12.7, podemos ver que a alteração foi bem sucedida, pois mostrou os dados do aluno id=5 atualizado.

Exemplo excluir aluno, do código em MVC

Para excluir um aluno, não temos alteração do que já apresentamos para o método DELETE no arquivo sala_aula_server.py, isto porque, não temos corpo na requisição.

O comando digitado no segundo terminal é: C:\dam>curl -H \Content-Type:application/json\ -X DELETE http://localhost:5002/alunos/4 .

Queremos excluir o aluno de id = 4, por isso, http://localhost:5002/alunos/4.

Na figura 12.8, podemos ver que o aluno de id = 4 foi excluído, pois não aparece mais nos dados cadastrados e apresentando no terminal.

Figura 12.8. Resultado da execução do arquivo sala_aula_server.py para o método DELETE no segundo terminal.

```
C:\dam>curl -H \Content-Type:application/json\ -X DELETE  http://localhost:5002/alunos/4
[
  {
    "id": 1,
    "media": 8.5,
    "nome": "Andreia"
  },
  {
    "id": 2,
    "media": 10,
    "nome": "Arthur"
  },
  {
    "id": 3,
    "media": 10,
    "nome": "Pedro"
  }
]
```

Fonte: do autor, 2022.

Exemplo buscar um aluno por id, do código em MVC

Para obter dados de todos os alunos, ou um único aluno, não temos alteração nos códigos do arquivo sala_aula_server.py.

O comando digitado no segundo terminal é: curl -H \Content-Type:application/json\ -X GET http://localhost:5002/alunos/1. Em que, http://localhost:5002/alunos/1 quer dizer que estamos buscando (GET) os dados da rota /alunos do aluno de id = 1.

Figura 12.9. Resultado da execução do arquivo sala_aula_server.py para o método GET com o parâmetro do id do aluno, no segundo terminal.

```
C:\dam>curl -H \Content-Type:application/json\ -X GET  http://localhost:5002/alunos/1
{
  "Aluno": {
    "id": 1,
    "media": 8.5,
    "nome": "Andreia"
  },
  "Status": "Aluno encontrado"
}
```

Fonte: do autor, 2022.

Na Figura 12.9, podemos ver os dados do aluno com id=1, que foi o aluno buscado.

12.2. Tunelamento na nuvem com ngrok

O flask, como vimos na seção anterior, cria um servidor que é executado localmente no tempo de execução, ou seja, permite comunicação apenas na mesma rede local. Para expor o servidor ao tráfego externo ou torná-lo acessível fora do tempo de execução globalmente em HTTP, usamos o **ngrok** (https://ngrok.com/). Nesse caso, vamos importar o módulo flask-ngrok do python.

A execução de aplicativos flask na máquina local é muito simples, porém, se tratando de compartilhamento de link do aplicativo com outros usuários, você precisa configurar o aplicativo inteiro em outro computador ou celular.

Vamos ver aqui, uma forma de enviar o link do nosso servidor, para que outras pessoas possam testá-lo, antes de implantá-lo.

Primeiro passo é que precisamos instalar o pacote/biblioteca **flask-ngrok** em Python, digitando no *prompt de comando*: pip install flask-ngrok e logo em seguida, instalar o pacote **ngrok**, digitando no *prompt de comando*: pip install ngrok

Observação: até em 2021, não era necessário instalar ngrok, apenas flask-ngrok. Outra alteração, que é agora é obrigatório gerar um token de autenticação para rodar os códigos com flask e ngrok.

No site https://dashboard.ngrok.com/get-started/setup, é preciso fazer um cadastro. Logo em seguida, será mostrado o texto **Connect your account**, em que `275JW9XPjdA0oXh9lcsl7NiTchd_6S3nK3dU59S8EvQ5MFHQJ` é o token gerado para o cadastro realizado. Esse token será adicionado nos códigos. Não esqueça de fazer seu cadastro.

2. Connect your account

Running this command will add your authtoken to the default `ngrok.yml` configuration file. This will grant you access to more features and longer session times. Running tunnels will be listed on the endpoints page of the dashboard.

```
ngrok config add-authtoken
275JW9XPjdA0oXh9lcsl7NiTchd_6S3nK3dU59S8EvQ5MFHQJ
```

Pronto! Depois de instalado o pacote, podemos utilizar ngrok em nossas aplicações. Vamos criar nosso primeiro código com ngrok, salvando o novo arquivo com o nome aula12_exemplo2.py, (Figura 12.10).

Figura 12.11. Código do arquivo aula12_exemplo2.py.

```
from flask import Flask
from flask_ngrok import run_with_ngrok
from pyngrok import ngrok

app = Flask(__name__)
ngrok.set_auth_token("275JW9XPjdA0oXh9lcsl7NiTchd_6S3nK3dU59S8EvQ5MFHQJ")
run_with_ngrok(app)

@app.route('/')
def start():
  return 'Seja bem-vindo na aula de Desevolvimento de APIs e Microsserviços.'

if __name__ == '__main__':
    app.run()
```

Fonte: do autor, 2022.

O que temos de diferente do código da Figura 12.10 para o código da Figura 12.1? Basicamente, cinco linhas:

- Precisamos importar no nosso código from flask_ngrok import run_with_ngrok e from pyngrok import ngrok para que possamos nosso código com ngrok (linhas 2 e 3 da Figura 12.10);
- ngrok.set_auth_token("275JW9XPjdA0oXh9lcsl7NiTchd_6S3nK3dU59S8EvQ5MFHQJ"): aqui, estamos setando o token referente ao cadastro que fizemos, para usar ngrok free (linha 6 da Figura 12.10);
- run_with_ngrok(app): a função run_with_ngrok () recebe o objeto da classe Flask, armazenado em app. Ele anexa o ngrok ao aplicativo flask para que, quando executarmos o aplicativo, uma URL possa ser gerada e acessível fora do tempo de execução ngrok (linha 7 da Figura 12.10);
- app.run(): executamos nosso servidor, sem especificar os parâmetros host, port e debug (linha 14 da Figura 12.10).

No *promp de comando*, digitamos então, python aula12_exemplo2.py para execução do código. Podemos ver na Figura 12.11(b), várias informações, como por exemplo, que a sessão está online, região, qual usuário está conectado (quer dizer, o

token se refere a qual cadastro), dentre outras. Em *forwarding*, mostra dois endereços, em que os finais são **.ngrok.io**.

Figura 12.12. Executando o arquivo aula12_exemplo2.py.

```
C:\dam>python aula12_exemplo2.py
 * Serving Flask app "aula12_exemplo2" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

(a)

```
ngrok by @inconshreveable
Session Status                online
Account                       andreia.gusmao@faculdadeimpacta.com.br (Plan: Free)
Version                       2.3.40
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://77f4-170-244-246-26.ngrok.io -> http://localhost:5000
Forwarding                    https://77f4-170-244-246-26.ngrok.io -> http://localhost:5000
Connections                   ttl     opn     rt1     rt5     p50     p90
0       0       0.00    0.00    0.00    0.00
* Running on http://77f4-170-244-246-26.ngrok.io
```

(b)

Fonte: do autor, 2022.

Como executar? Abrimos o nosso navegador e digitamos https://77f4-170-244-246-26.ngrok.io/, pois nossa rota definida no arquivo aula12_exemplo2.py não tem nome.

Figura 12.13. Resultado da execução do arquivo aula12_exeplo2.py no navegador com ngrok.

Seja bem-vindo na aula de Desenvolvimento de APIs e Microsserviços.

Fonte: do autor, 2022.

Bom, qual a novidade? Já testamos dessa maneira no navegador sem o ngrok. A novidade é que agora podemos copiar a URL com https e executar em 'qualquer lugar', como por exemplo, no celular, mesmo a aplicação sendo executada no computador.

Um novo teste

Vamos pegar o mesmo código aula12_exemplo1.py, apresentado na Figura 12.1 e vamos adaptá-lo para usar ngrok, salvando o novo arquivo com o nome aula12_exemplo3.py (Figura 12.13). As adaptações são as mesmas mostradas na Figura 12.10 para o código aula12_exemplo2.py.

Figura 12.13. Código do arquivo aula12_exemplo3.py.

```
from flask import Flask, request
from flask_ngrok import run_with_ngrok
from pyngrok import ngrok

app = Flask(__name__)
ngrok.set_auth_token("275JW9XPjdA0oXh9lcsl7NiTchd_6S3nK3dU59S8EvQ5MFHQJ")
run_with_ngrok(app)  # inicia ngrok quando app está executando

@app.route('/', methods=['POST'])
def boasVindas():
  nome = request.get_json(force=True)
  return 'Seja bem-vindo, {}, na aula de Desevolvimento de APIs e Microsserviç
os.\n'.format(nome['nome'])

if __name__ == '__main__':
    app.run()
```

Fonte: do autor, 2022.

Precisamos executar o código no *prompt de comando*: python aula12_exemplo3.py. E logo após será mostrada a tela da Figura 12.14(a). Esse teste vamos executar com dois terminais.

Então, vamos abrir o segundo terminal e digitar: curl -H \Content-Type: application/json\ -X POST -d "{\"nome\":\"Andreia\"}" http://00f2-2804-14d-7897-93d7-cd2d-3fb0-a6d5-ae33.ngrok.io (Figura 12.15(b)). O resultado e a explicação são os mesmos da Figura 12.2 (arquivo aula12_exemplo1.py)

Figura 12.14. Executando o arquivo aula12_exemplo3.py em dois terminais, com ngrok.

```
ngrok by @inconshreveable
Session Status          online
Account                 andreia.gusmao@faculdadeimpacta.com.br (Plan: Free)
Version                 2.3.40
Region                  United States (us)
Web Interface           http://127.0.0.1:4040
Forwarding              http://6d3c-170-244-246-26.ngrok.io -> http://localhost:5000
Forwarding              https://6d3c-170-244-246-26.ngrok.io -> http://localhost:5000
HTTP Requests
POST /                  200 OK
```

(a)

```
C:\dam>curl -H \Content-Type: application/json\ -X POST -d "{\"nome\":\"Andreia\"}"
http://6d3c-170-244-246-26.ngrok.io

curl: (6) Could not resolve host: application

Seja bem-vindo, Andreia, na aula de Desenvolvimento de APIs e Microsserviços.
```

(b)

Fonte: do autor, 2022.

Para esse exemplo, não conseguimos demonstrar o resultado no navegador do celular, pois como o método tem corpo na requisição, requer os dados de entrada, por um formulário, por exemplo, e não desenvolvemos essa parte visual.

Testando ngrok com nosso código em MVC

Vamos criar o arquivo sala_aula_server_ngrok.py, seguindo as modificações já realizadas no arquivo sala_aula_server.py (para executar requisições em terminais), somente com as adaptações para usar ngrok (as adaptações são as mesmas mostradas na Figura 12.10 para o código aula12_exemplo2.py).

Na Figura 12.15, apresentamos o código do arquivo sala_aula_server_ngrok.py, com destaque para as implementações para uso da biblioteca flask_ngrok e as funções estão omitidas, apenas para mostrar o que realmente teve alterações.

Os testes, seguem o mesmo padrão já apresentado. As rotas com corpo de requisição (POST e PUT) devem ser testadas no POSTMAN (pois não desenvolvemos a parte visual, por exemplo um formulário, para que o usuário entrasse com os dados de forma correta e simples). As demais, podem ser testadas no navegador, POSTMAN ou celular.

A seguir, apresentamos o resultado da execução para a rota /alunos, com o método GET, que permite buscar um aluno pelo seu id. Lembrando que é necessário executar o código python no *prompt de comando*: python sala_aula_server_ngrok.py. Após essa execução, irá mostrar no terminal a url para testes.

Considere para esse exemplo a seguinte url: http://2e20-2804-14d-7897-93d7-cd2d-3fb0-a6d5-ae33.ngrok.io.

Como queremos testar a rota alunos que requer o parâmetro id do aluno, devemos então, chamar essa url como http://2e20-2804-14d-7897-93d7-cd2d-3fb0-a6d5-ae33.ngrok.io/alunos/1, em que 1 aqui significa que estou pesquisador o aluno com id = 1.

Figura 12.15. Testando a rota /alunos para inserção de um aluno.

```
from flask import Flask, jsonify, request
from flask_ngrok import run_with_ngrok
from pyngrok import ngrok

import controller.aluno_controller as aluno_controller

app = Flask(__name__)
ngrok.set_auth_token("275JW9XPjdA0oXh9lcsl7NiTchd_6S3nK3dU59S8EvQ5MFHQJ")
run_with_ngrok(app)  # inicia ngrok quando app está executando

@app.route('/alunos')
def getAlunos():
    return aluno_controller.listar()

@app.route("/alunos/<int:id_consulta>", methods=["GET"])
def getAlunoId(id_consulta):
    return aluno_controller.localizaPorId(id_consulta)

@app.route("/alunos/maior_media", methods=["GET"])
def getAlunoMaiorMedia():
    return aluno_controller.localizarPorMaiorMedia()

@app.route("/alunos", methods=["POST"])
def inserir():
  aluno = request.get_json(force=True)
```

```
  return aluno_controller.inserirAluno(aluno)

@app.route("/alunos/<int:id_deletar>", methods=["DELETE"])
def excluir(id_deletar):
  return aluno_controller.excluirPorId(id_deletar)

@app.route("/alunos/<int:id_alterar>", methods=["PUT"])
def alterar(id_alterar):
  aluno = request.get_json(force=True)
  return aluno_controller.alterarAluno(id_alterar, aluno)

@app.route('/')
def start():
    return "Construindo nossa aplicação com MVC"

if __name__ == '__main__':
    app.run()
```

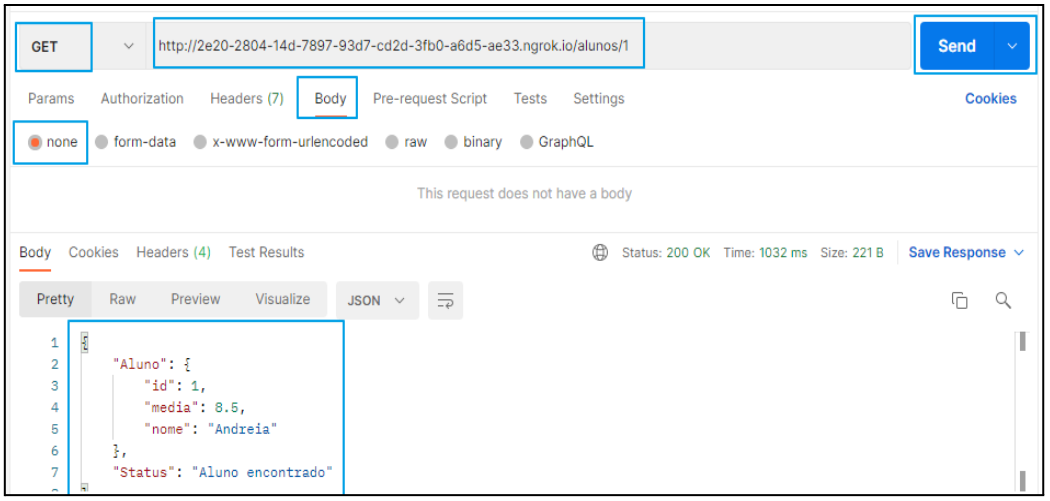
Fonte: do autor, 2022.

Na Figura 12.16 temos o resultado dessa execução em duas formas diferentes: no navegador (Figura 12.16(a)) e no POSTMAN (Figura 12.16(b)). Podemos perceber que em todas, temos a mesma saída, que são os dados do aluno com id = 1.

Figura 12.16. Resultado da execução da rota /alunos para busca de um aluno pelo seu id, com ngrok no arquivo de código com padrão MVC.

{“Aluno”: {“id”: 1, “media”:8.5,”nome”:Andreia}, “Status”: “Aluno encontrado”}

(a)



(b)

Fonte: do autor, 2022.

Para praticar, recomendamos que teste as demais rotas do arquivo com o código fonte, compartilhe a url para testes em outros computadores, celulares, etc.

12.3. Código completo

Os arquivos encontram-se como material complementar desta aula. Temos aula12_exemplo1.py, aula12_exemplo2.py, aula12_exemplo3.py e a pasta mvc, em que as pastas controller e model seguem as mesmas da aula passada, o diferencial são dois servidores: sala_aula_server.py e sala_aula_server_ngrok.py.

Referências

Como executar o aplicativo Flask online usando Ngrok?. Disponível em: https://acervolima.com/como-executar-o-aplicativo-python-flask-online-usando-ngrok/. Acesso em 29 out 2021.

Tutorial do framework Flask. Disponível em: https://edisciplinas.usp.br/pluginfile.php/5258505/mod_resource/content/1/Tutorial.pptx.pdf. Acesso em 29 out 2021.