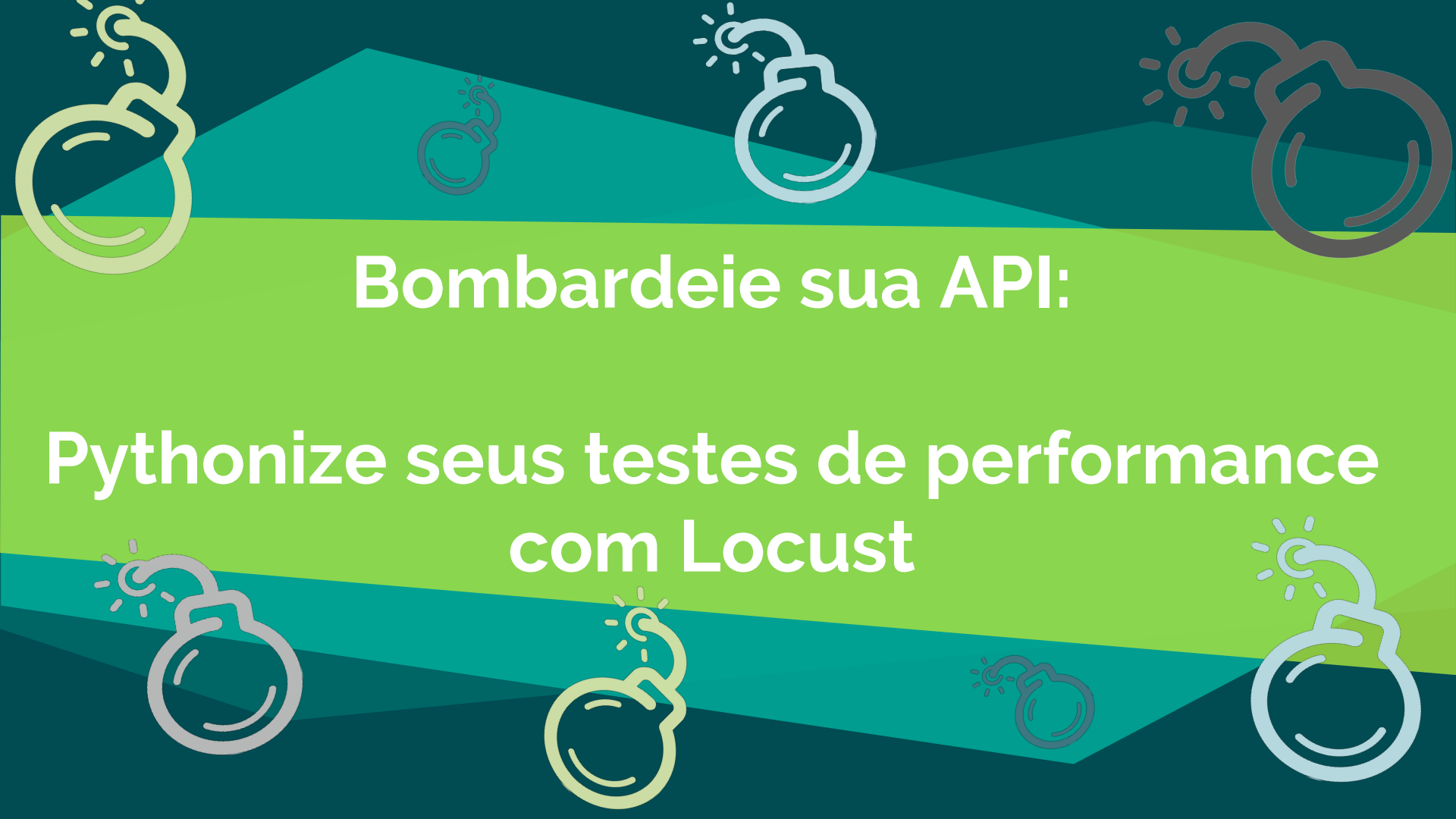

Bombardeie sua API:
Pythonize seus testes de performance com Locust

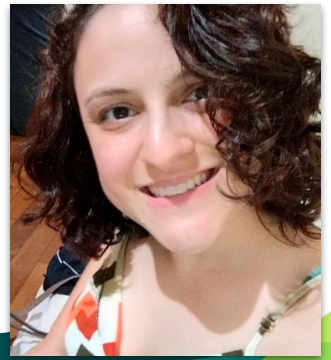
Hello World! 👋

Élysson MR

Desenvolvedor Python/GO/NodeJS atuando com micro-serviços na Softplan, curioso por natureza e padawan em Arquitetura de Software.

May Fernandes

Analista de Testes com mais de 08 anos de experiência com testes de aplicações desktop e micro-serviços. Atualmente trabalho com processos de automação de testes e QAOps na Softplan.



TESTES DE PERFORMANCE

Uma breve explicação...

Os testes de performance tem como objetivo avaliar o desempenho e comportamento de uma aplicação sob condições de carga e estresse (transações e usuários simultâneos).

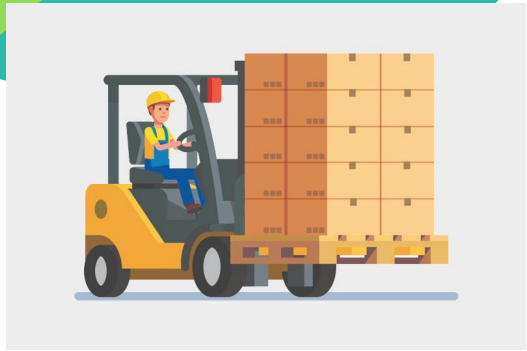
Tipos de testes de performance

Desempenho

- **Objetivo**: Avaliar o comportamento da aplicação sob uso constante ao longo do tempo.

- **Condições**: Carga em nível típico constante durante um tempo definido. Ou tempo não definido, para revelar o momento de falha e gargalos

Tipos de testes de performance



Carga

- **Objetivo**: Avaliar como a aplicação lida com níveis variáveis de carga: baixo, típico e pico.

- **Condições**: Carga inicial baixa, porém aumentada gradativamente durante um tempo definido. Ou tempo não definido, para revelar o momento de falha e gargalos.

Tipos de testes de performance

Estresse

- **Objetivo**: Avaliar como a aplicação lida com os níveis máximos de carga e determina o quanto de carga sua aplicação aguenta até a falha. Também serve para determinar os recursos mínimos para uso da aplicação.

- **Condições**: Carga inicial em nível de pico e aumentada gradualmente durante um tempo definido. Ou tempo não definido, para revelar o momento de falha e gargalos. Também pode reduzir a disponibilidade dos recursos (memória, banda larga, CPU, etc)

Requisitos não-funcionais

Mas então como eu testo?

Mas o que é ter performance?

Quando posso dizer que minha aplicação tem um bom desempenho?

Requisitos não-funcionais

- “Quero o mais rápido possível, por favor!”
- “Tem que ser muito rápida!”
- “Não pode ser lenta!”
- “As respostas têm que ser instantâneas!”

Requisitos não-funcionais

É importante que os requisitos não-funcionais sejam bem definidos já no planejamento da aplicação, considerando contexto e objetivo da aplicação e, que estejam alinhados com as expectativas dos clientes/usuários.

Requisitos não-funcionais

[RNF-Exemplo] Minha aplicação deve suportar 1.000 transações por minuto em um ambiente normal com média de 1.000 usuários simultâneos.

- **Desempenho**: A aplicação suporta 1.000 transações por minuto com 1.000 usuários simultâneos?
- **Carga**: Quantas transações serão suportadas por minuto quando aumentarmos os usuários simultâneos para 2.000, 3.000, 4.000?
- **Estresse**: Quantas transações por minuto solicitadas por 5.000, 6.000, 7.000 usuários simultâneos, serão suportadas pela aplicação sob condições não especificadas do software/hardware?

Requisitos não-funcionais

[RNF-Exemplo] Minha aplicação deve suportar 1.000 transações por minuto em um ambiente normal com média de 1.000 usuários simultâneos.

- **Desempenho**: A aplicação suporta 1.000 transações por minuto com 1.000 usuários simultâneos?
- **Carga**: Quantas transações serão suportadas por minuto quando aumentarmos os usuários simultâneos para 2.000, 3.000, 4.000?
- **Estresse**: Quantas transações por minuto solicitadas por 5.000, 6.000, 7.000 usuários simultâneos, serão suportadas pela aplicação sob condições não especificadas do software/hardware?

Requisitos não-funcionais

[RNF-Exemplo] Minha aplicação deve suportar 1.000 transações por minuto em um ambiente normal com média de 1.000 usuários simultâneos.

- **Desempenho**: A aplicação suporta 1.000 transações por minuto com 1.000 usuários simultâneos?
- **Carga**: Quantas transações serão suportadas por minuto quando aumentarmos os usuários simultâneos para 2.000, 3.000, 4.000?
- **Estresse**: Quantas transações por minuto solicitadas por 5.000, 6.000, 7.000 usuários simultâneos, serão suportadas pela aplicação sob condições não especificadas do software/hardware?

Requisitos não-funcionais

[RNF-Exemplo] Minha aplicação deve suportar 1.000 transações por minuto em um ambiente normal com média de 1.000 usuários simultâneos.

- **Desempenho**: A aplicação suporta 1.000 transações por minuto com 1.000 usuários simultâneos?
- **Carga**: Quantas transações serão suportadas por minuto quando aumentarmos os usuários simultâneos para 2.000, 3.000, 4.000?
- **Estresse**: Quantas transações por minuto solicitadas por 5.000, 6.000, 7.000 usuários simultâneos, serão suportadas pela aplicação sob condições não especificadas do software/hardware?

FERRAMENTAS PARA TESTES DE PERFORMANCE

Introdução ao Locust e demonstração

Locust

Ferramenta Python para testes de performance

Bombardeie sua API

A ideia é bombardear sua aplicação com um enxame de gafanhotos (ou usuários, como quiser!) gerando subsídios para você monitorar sua aplicação através de uma interface web ou em *command line mode*, para analisar e encontrar os gargalos da sua aplicação.

Características

- **Código Simples**: Crie seus testes diretamente em código Python. E adicione recursos extras com o poder do Python ❤.
- **Distribuído e Escalável**: Suporta teste distribuído, permitindo escalabilidade de usuários conectados.
- 👍***Battletested***: O Locust foi usado para testar o *Battlelog*, app web do famoso jogo *Battlefield*!!

MÃO NA MASSA

Uma breve demonstração...

Exemplo de Teste no Locust

(simples)

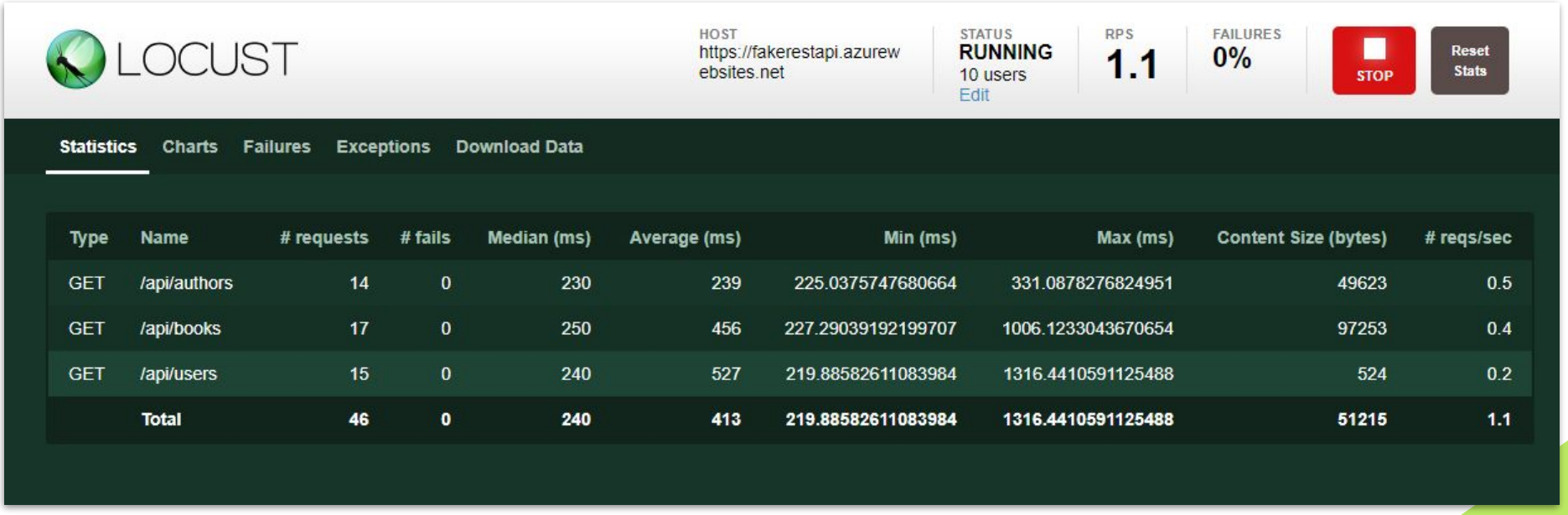
locustexample1.py

```python
from locust import HttpLocust, TaskSet, task

class APITasks(TaskSet):
    @task
    def authors(self):
        self.client.get("/api/authors")

    @task
    def books(self):
        self.client.get("/api/books")

    @task
    def users(self):
        self.client.get("/api/users")

class APIUser(HttpLocust):
    task_set = APITasks
    min_wait = 5000
    max_wait = 15000
    host = "https://fakerestapi.azurewebsites.net"
```

locustexample2.py

```
> locust -f locustfile.py
```

Locust: Web mode



Type	Name	# requests	# fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Content Size (bytes)	# reqs/sec
GET	/api/authors	14	0	230	239	225.0375747680664	331.0878276824951	49623	0.5
GET	/api/books	17	0	250	456	227.29039192199707	1006.1233043670654	97253	0.4
GET	/api/users	15	0	240	527	219.88582611083984	1316.4410591125488	524	0.2
	Total	**46**	**0**	**240**	**413**	**219.88582611083984**	**1316.4410591125488**	**51215**	**1.1**

Exemplo de Teste no Locust

(com pré-requisito e pesos)

locustexample1.py | locustexample2.py

```python
from random import randint
from locust import HttpLocust, TaskSet, task

class APITasks(TaskSet):
    def on_start(self):
        for i in range(1, 6):
            book = {
                "Title": f"My Little Book {i}",
                "ID": f"{i}"
            }
            self.client.post("/api/books", json=book)

    @task(10)
    def allBooks(self):
        self.client.get("/api/books")

    @task(5)
    def especificBook(self):
        seed = randint(1, 5)
        self.client.get(f"/api/books/{seed}", name="/api/books/ID")

class APIUser(HttpLocust):
    task_set = APITasks
    min_wait = 5000
    max_wait = 15000
    host = "https://fakerestapi.azurewebsites.net"
```

```
> locust -f locustfile.py -H https://fakerestapi.azurewebsites.net
```

Exemplo Teste de Desempenho

- **100** usuários (**-c**)
- **100** usuários por segundo (**-r**) -> *Para indicar que será constante*
- durante **5** minutos (**-t**)

```
> locust --no-web -c 100 -r 100 -t 5m -H http://localhost:8000 -f
locustfile.py
```

Exemplo Teste de Carga

- **500** usuários (**-c**)
- **10** usuários por segundo (**-r**) -> *Até chegar em 500*
- durante **5** minutos (**-t**)

```
> locust --no-web -c 500 -r 10 -t 5m -H http://localhost:8000 -f
locustfile.py
```

Exemplo Teste de Estresse

- **3000** usuários (**-c**)
- **200** usuários por segundo (**-r**) -> *Até chegar em 3000*
- durante **10** minutos (**-t**)

```
> locust --no-web -c 3000 -r 200 -t 10m -H http://localhost:8000
-f locustfile.py
```

Thanks!

Perguntas?

Você também pode nos achar em:

elyssonmr@gmail.com

ammmayara@hotmail.com

https://github.com/elyssonmr

https://github.com/mayribeirofernandes

https://www.linkedin.com/in/elyssonmr/

https://www.linkedin.com/in/mayfernandes/

Referências

- https://locust.io/
- https://www.devmedia.com.br/testes-de-desempenho-carga-e-stress/26546
- https://www.slideshare.net/edlaineuem/teste-de-performance-com-jmeter-como-criar-e-executar-os-testes-em-aplicaes-web-e-como-interpretar-seus-resultados
- http://www.bstqb.org.br/uploads/syllabus/syllabus_ctal_tta_2012br.pdf
- http://www.bstqb.org.br/uploads/glossario/glossario_ctfl_3.2br.pdf
- 7Masters Automação de Testes - Escolhas durante a automação de testes de API com Julio de Lima (https://www.youtube.com/watch?v=lMpBbaqWSsA)
- MTC 2016 - Stefan Teixeira - Testes de Carga com Locust (https://www.youtube.com/watch?v=riy0z8ltFeY)