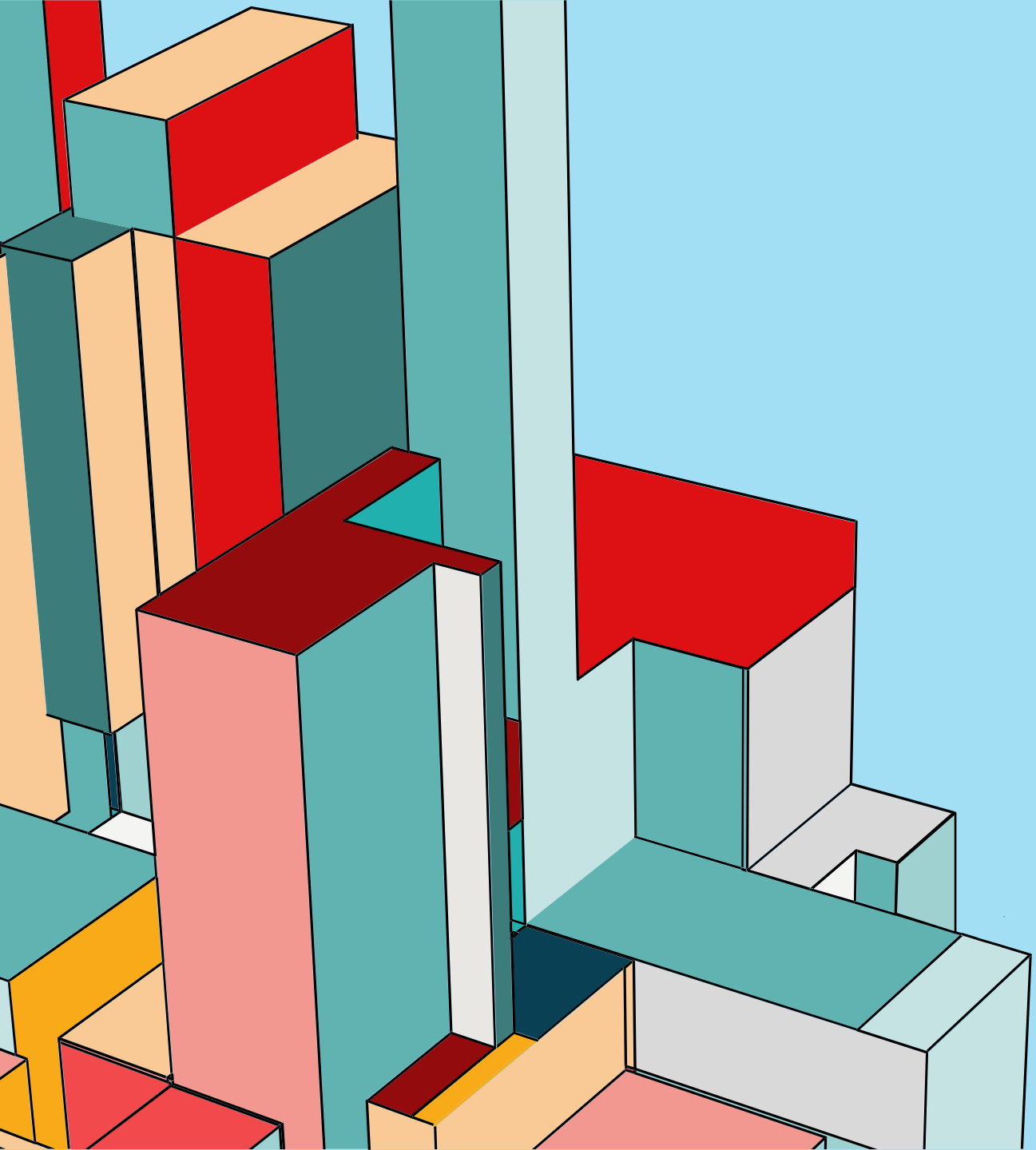


UNIVAS
UNIVERSIDADE DO VALE DO SAPUCAÍ

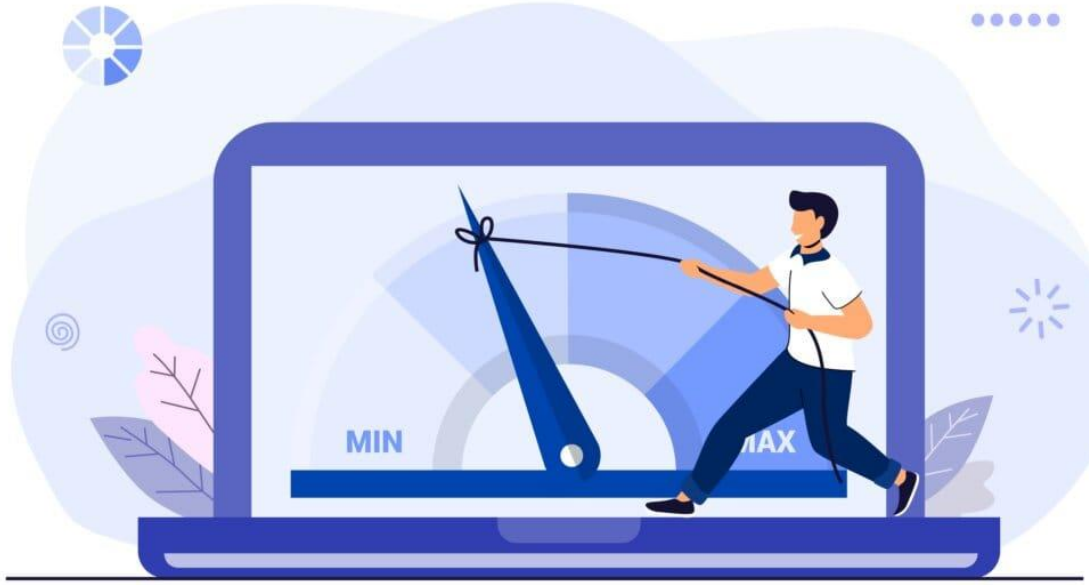
TESTE DE SOFTWARE

TESTES DE CARGA

Prof. Flávio Belizário da Silva Mota
Universidade do Vale do Sapucaí – UNIVAS
Sistemas de Informação

TESTE DE CARGA

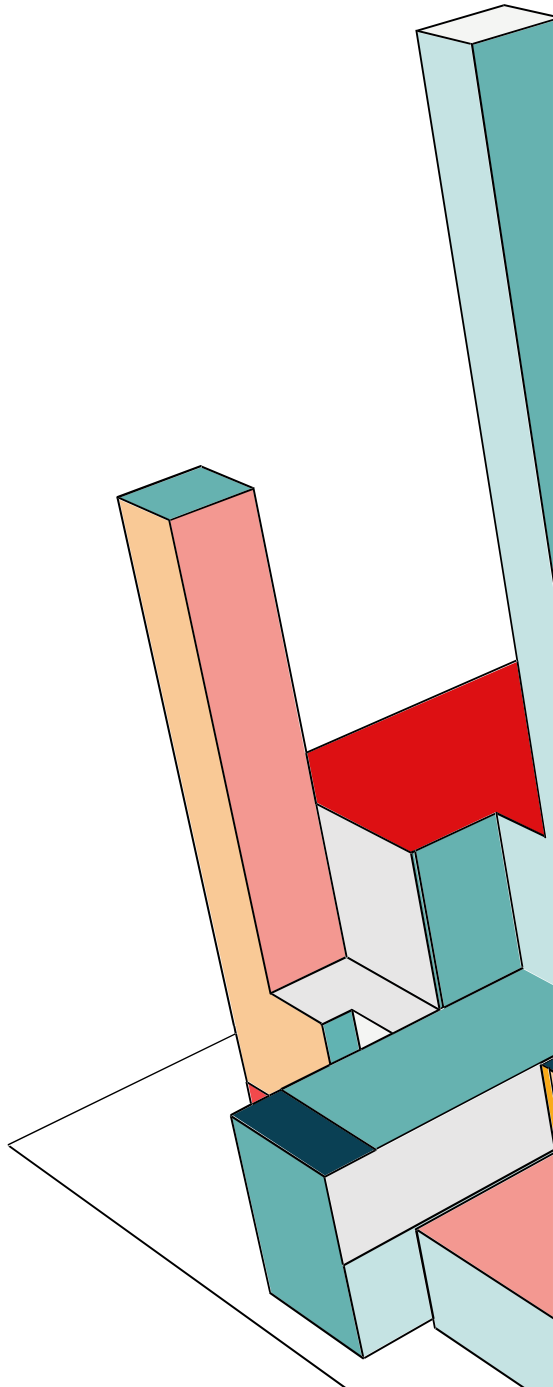
- Testes de carga verificam como o sistema se comporta sob **diferentes níveis de demanda**.
- Enquanto os testes de integração garantem funcionalidade, os de carga garantem **desempenho e estabilidade**.
- O objetivo é identificar problemas de desempenho, **como lentidão e falhas** que **podem ocorrer em condições de uso normal ou pico**.

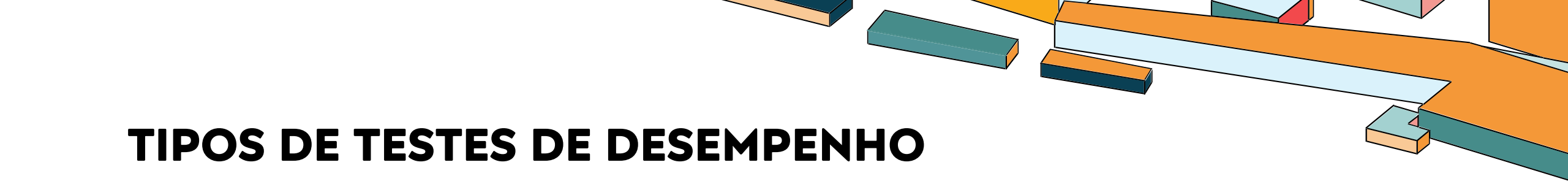


TESTE DE CARGA

Testes de carga ajudam a responder perguntas como:

- Quantos usuários simultâneos o sistema suporta?
- O tempo de resposta cresce de forma aceitável?
- Existe gargalo em alguma parte da aplicação?





TIPOS DE TESTES DE DESEMPENHO

- Teste de carga (load test): mede o comportamento sob carga esperada.
- Teste de estresse (stress test): ultrapassa o limite de carga até a falha.
- Teste de volume: avalia grandes volumes de dados.
- Teste de endurance (soak): mantém carga constante por longo período.

JMETER

- O **Apache JMeter** é uma ferramenta open source para testes de desempenho e carga.
- Simula usuários enviando requisições simultâneas (HTTP, JDBC, SOAP, FTP, etc.) e **coleta métricas como tempo médio de resposta, throughput e taxa de erros**.

JMETER

- O JMeter trabalha com **planos de teste (test plans)** compostos por elementos:
  - **Thread Groups:** simulam usuários virtuais e definem a carga.
  - **Samplers:** definem as requisições (HTTP, JDBC, etc.).
  - **Config elements:** armazenam variáveis e parâmetros.
  - **Listeners:** exibem resultados e gráficos.
  - **Assertions:** verificam se a resposta é válida.

PARÂMETROS PRINCIPAIS DE CARGA

- Em um Thread Group, configuramos:
  - **Número de usuários virtuais (threads)** simultâneos o JMeter deve simular. Cada *thread* representa um cliente virtual independente, que envia requisições ao servidor da mesma forma que um usuário real faria.
  - Tempo de subida (**Ramp-up period**). O Ramp-up define em quantos segundos o JMeter deve "chegar" ao total de *threads* configurado. Controla a velocidade com que os usuários virtuais entram no sistema.
  - Número de execuções (**loops**). Define quantas vezes cada *thread* executa a sequência de requisições

MÉTRICAS ANALISADAS

- É possível coletar dados como:
  - **Throughput:** número de requisições processadas por segundo.
  - **Latência:** tempo para receber a primeira resposta.
  - **Tempo de resposta médio** e percentil (**p95/p99**).
  - **Erro (%)** e **TPS (transactions per second)**.

PASSO A PASSO: JMETER PARA TESTE DE CARGA

1) Instalar JMeter*

- Baixe o ZIP do JMeter no site oficial (https://jmeter.apache.org/download_jmeter.cgi)
- Extraia o ZIP para algo como `C:\Tools\apache-jmeter-5.6.3\`
- Rode a GUI:
  - Dê duplo clique em `C:\Tools\apache-jmeter-5.6.3\bin\jmeter.bat` (ou rode no PowerShell).

*Necessário ter um JDK/SDK instalado.

PASSO A PASSO: JMETER PARA TESTE DE CARGA

2) Organizar arquivos do projeto

- Crie no projeto:
  - **backend/tests/jmeter/plans/** – arquivos .jmx
  - **backend/tests/jmeter/data/** – CSVs/payloads
  - **backend/tests/jmeter/props/** – propriedades (ex.: local.properties)
  - **backend/tests/jmeter/results/** – arquivos .jtl
  - **backend/tests/jmeter/reports/** – relatórios HTML

PASSO A PASSO: JMETER PARA TESTE DE CARGA

Não é uma boa prática rodar todos os testes do JMeter via interface gráfica (consome parte da memória e afeta os resultados). Sendo assim, vamos usar o GUI para criar os planos e o CLI para rodar os comandos.

Podemos criar um arquivo de propriedades para orientar o teste:

3) Dentro de **backend/tests/jmeter/props/** crie um arquivo **`local.properties`**

```
baseUrl=http://localhost:3001

users=100

rampUp=10

loopCount=1
```

PASSO A PASSO: JMETER PARA TESTE DE CARGA

4) Criar um plano básico na GUI (backend HTTP)

1. Abra o JMeter GUI.
2. Criar um Test Plan → Add → Threads (Users) → Thread Group
    1. Number of Threads: ${__P(users,100)}
    2. Ramp-up: ${__P(rampUp,10)}
    3. Loop Count: ${__P(loopCount,1)}
3. Dentro de Thread Group → Add → Config Element → HTTP Request Defaults
    1. Server Name or IP:
        1. Server: localhost
        2. Port: 3001

PASSO A PASSO: JMETER PARA TESTE DE CARGA

4) Criar um plano básico na GUI (backend HTTP)

4. Thread Group → Add → Sampler → HTTP Request:

- Testar o método /health da API através de um GET

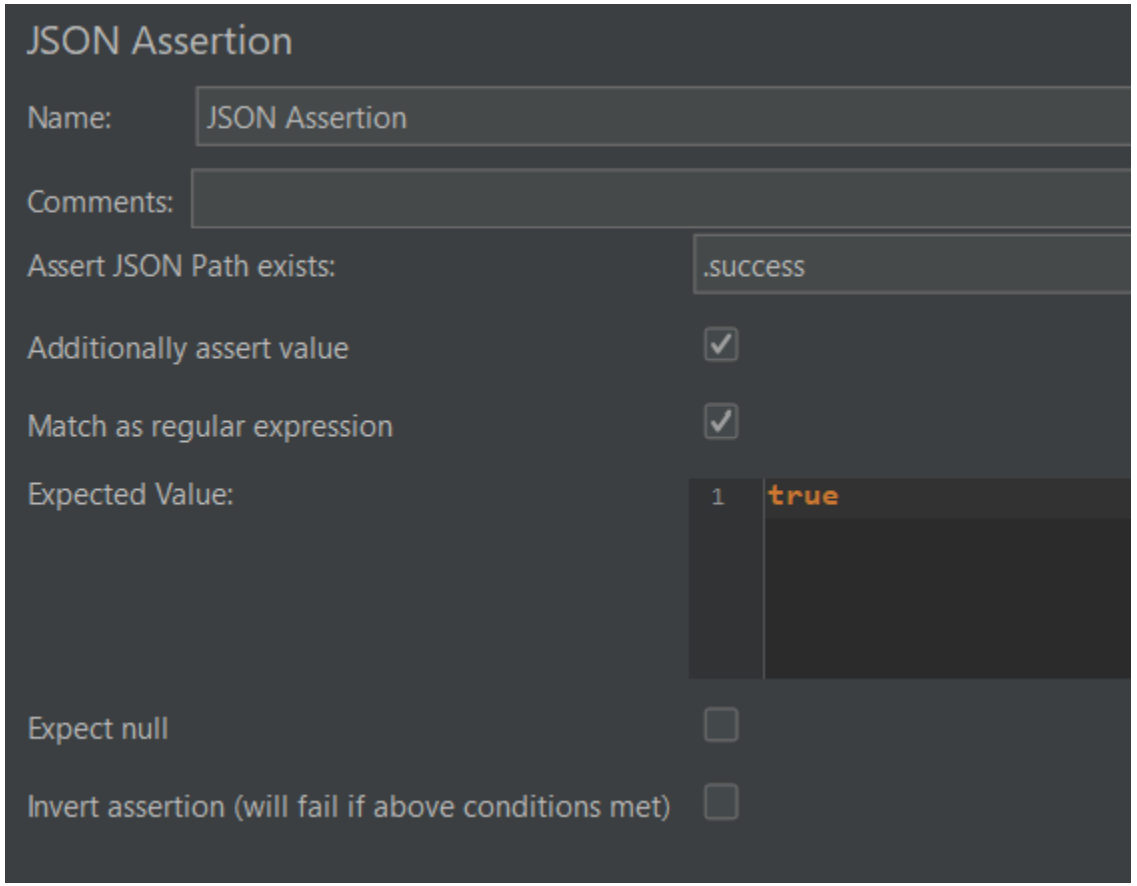
5. Adicionar um componente View Results in Table, um Graph Results e um Aggregate Report.

6. Rodar pela GUI.

PASSO A PASSO: JMETER PARA TESTE DE CARGA

5) Adicionar Assertions: para garantir que o sistema responde corretamente

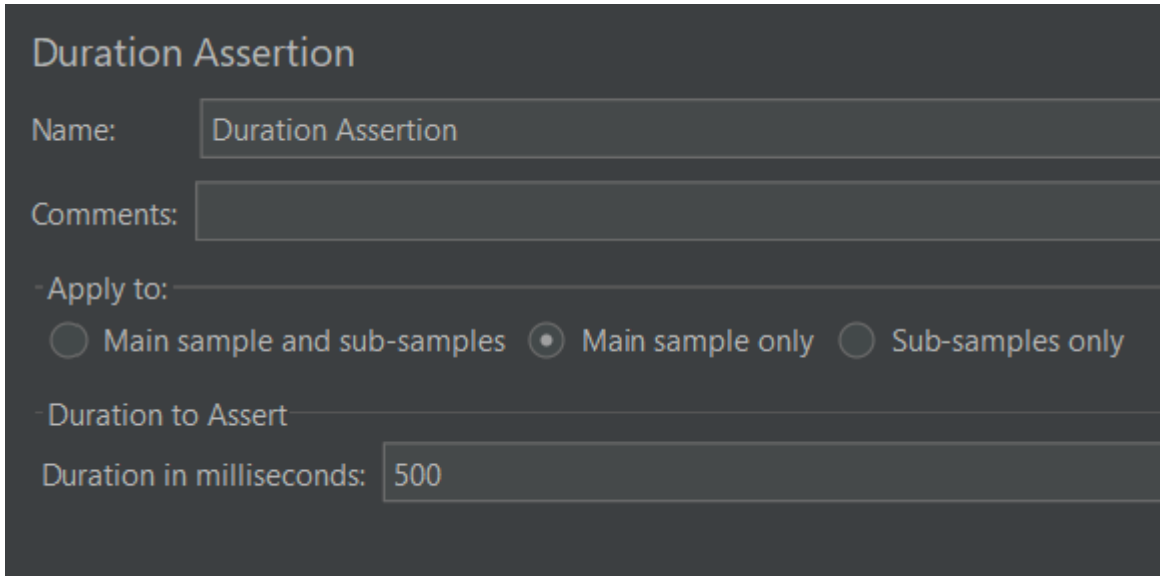
1. Add → Assertions → JSON Assertion



PASSO A PASSO: JMETER PARA TESTE DE CARGA

5) Adicionar Assertions: para garantir que o sistema responde corretamente

1. Add → Assertions → Duration Assertion



PASSO A PASSO: JMETER PARA TESTE DE CARGA

6) Salve o plano em backend/tests/jmeter/plans/ com o nome de smoke.jmx

Smoke aqui vem da ideia “ligar e ver se sai fumaça”. No hardware, um “smoke test” significava ligar o equipamento para checar se não queimava.

Em software, virou um teste rápido de sanidade que verifica se o sistema básico “sobe e responde” sem erros.

No contexto de performance/carga:

- É um teste leve (poucos usuários, curta duração).
- Serve para validar ambiente, dados, roteiros JMeter e asserts antes dos testes pesados.
- Critério típico: 0% erros e tempos aceitáveis (ex.: p95 < 500 ms) nas rotas críticas.

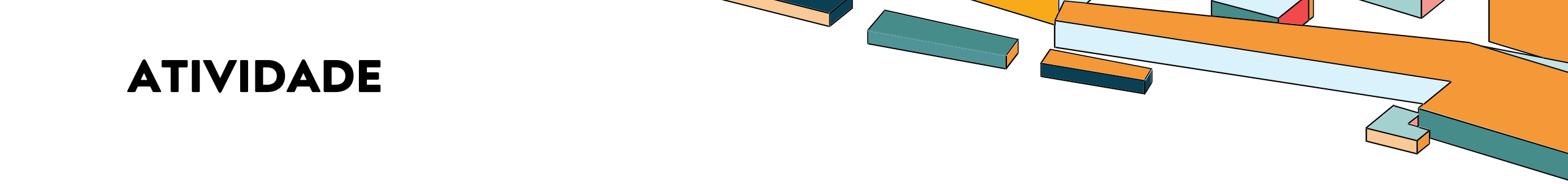
PASSO A PASSO: JMETER PARA TESTE DE CARGA

7) Precisamos rodar o comando para executar o JMeter, usando um arquivo de plano de teste, especificando as propriedades e destinando um local para salvar o resultado do teste e o relatório (que pode ser aberto em HTML)

```
C:\Tools\apache-jmeter-5.6.3\bin\jmeter.bat -n -t
backend/tests/jmeter/plans/smoke.jmx -l
backend/tests/jmeter/results/smoke.jtl -e -o
backend/tests/jmeter/reports/smoke -q
backend/tests/jmeter/props/local.properties
```

Isso vai produzir um relatório que pode ser acessado via:

```
start backend/tests/jmeter/reports/smoke/index.html
```



ATIVIDADE

Altere o número de threads e o ramp-up:

- 50 usuários em 5 segundos;
- 200 usuários em 20 segundos.

Observe como o throughput e o tempo de resposta mudam.

ATIVIDADE – COMMITAR NO GIT

Crie um plano de teste simulando 200 usuários simultâneos acessando o endpoint `/api/tasks`.

Varie o ramp-up e gere um report.