RELATÓRIO TÉCNICO AEROCODE

APRESENTAÇÃO

Este relatório apresenta medições de tempo de requisições de API's em milissegundos. Para isso será utilizada a ferramenta Open-Source **k6** da Grafana, que permite testes de carga para melhoria de desempenho e performance.

Os testes foram realizados em um notebook ASUS-X1504ZA, com as seguintes especificações:

- Sistema Operacional: Windows 11.
- Processador: 12th Gen Intel(R) Core(TM) i5-1235U, 1300 Mhz, 10 núcleos, 12 processadores lógicos.
- Memória principal: SSD NVMe 477GB
- Memória RAM: SODIMM 2400MT/s 20GB
- Banco de dados: MySQL

Foram utilizados scripts em JavaScript para cada quantidade de usuários: 1, 5 e 10. Para cada um também há os testes padrões feitos pelo k6, assim como das métricas de latência, tempo de resposta e tempo de processamento, durando 30segundos cada teste. Além disso são testadas 4 API's referentes ao objeto da Aeronave, que envolvem dois GET's (um para todas as aeronaves e outro para uma aeronave específica), POST e DELETE.

LEITURA DE IMAGENS:

TOTAL RESULTS: indica as checagens padrão do k6.

- checks_total: indica a quantidade de validações realizadas.
- checks_succeded: indica a porcentagem de validações que obtiveram sucesso.
- checks_failed: indica porcentagem de validações que falharam.

As escritas em verde indicam quais foram os tipos de requisição, rota da api e status HTTP.

CUSTOM: mostra as métricas de latência, tempo de processamento e tempo de resposta.

- avg: tempo médio para iniciar a resposta.
- min: tempo mais rápido que uma requisição teve.
- med: mediana de tempo das requisições
- max: tempo mais demorado que uma requisição teve.
- p(90): percentil de 90% das requisições.
- p(95): percentil de 95% das requisições.

HTTP: mostra as métricas de duração de tempo da requisição, requisições que falharam e quantidade de requisições. Utiliza os mesmos atributos que “CUSTOM”.

EXECUTION: mostra tempo de iterações e quantidades de usuários virtuais ativos.

- interation_duration: tempo de uma iteração completa no script.
- iterations: quantidade de iterações.
- vus: quantidade de usuários virtuais.
- vus_max: quantidade máxima de usuários virtuais.

NETWORK:

- data_received: quantidade de dados recebidos pelo servidor.
- data_sent: quantidade de dados enviados pelo servidor.

Primeiro teste - 1 Usuário (Virtual Users/VUs)

```
     /\      Grafana   /‾‾/
  /\  /  \     |\  __   /  /
 /  \/    \    | |/ /  /   ‾‾\
/          \   |   (  |  (‾)  |
/ __________ \  |_|\_\  \_____/

    execution: local
       script: .\backend\k6\test_1User.js
       output: -

    scenarios: (100.00%) 1 scenario, 1 max VUs, 1m0s max duration (incl. graceful stop):
            * default: 1 looping VUs for 30s (gracefulStop: 30s)


  TOTAL RESULTS

    checks_total.......: 120     3.880604/s
    checks_succeeded...: 100.00% 120 out of 120
    checks_failed......: 0.00%   0 out of 120

    ✓ GET /aeronaves 200
    ✓ POST /aeronaves 201
    ✓ GET /aeronaves/:id 200
    ✓ DELETE /aeronaves/:id 200
```

```
CUSTOM
latencia.....................: avg=5.806827 min=0.5394 med=4.04635 max=32.9183 p(90)=10.30839 p(95)=11.13739
tempo_processamento..........: avg=0.589444 min=0      med=0.48225 max=7.7362  p(90)=1.0398   p(95)=1.77333
tempo_resposta...............: avg=6.396271 min=1.7482 med=4.39305 max=33.6519 p(90)=10.93731 p(95)=12.041115

HTTP
http_req_duration..............: avg=7.27ms   min=1.74ms med=7.68ms  max=33.65ms p(90)=10.93ms  p(95)=11.89ms
  { expected_response:true }...: avg=7.27ms   min=1.74ms med=7.68ms  max=33.65ms p(90)=10.93ms  p(95)=11.89ms
http_req_failed................: 0.00% 0 out of 120
http_reqs......................: 120   3.880604/s

EXECUTION
iteration_duration.............: avg=1.03s    min=1.02s  med=1.02s   max=1.11s   p(90)=1.03s    p(95)=1.04s
iterations.....................: 30    0.970151/s
vus............................: 1     min=1        max=1
vus_max........................: 1     min=1        max=1

NETWORK
data_received..................: 51 kB 1.6 kB/s
data_sent......................: 15 kB 485 B/s
```

Segundo Teste – 5 Usuários (Virtual Users/VUs)

```
      /\      Grafana   /‾‾/
   /\  /  \     |\  __   /  /
  /  \/    \    | |/ /  /   ‾‾\
 /          \   |   (  |  (‾)  |
/ __________ \  |_|\_\  \_____/

    execution: local
       script: .\backend\k6\test_5Users.js
       output: -

    scenarios: (100.00%) 1 scenario, 5 max VUs, 1m0s max duration (incl. graceful stop):
             * default: 5 looping VUs for 30s (gracefulStop: 30s)


  TOTAL RESULTS

    checks_total.......: 580     19.283052/s
    checks_succeeded...: 100.00% 580 out of 580
    checks_failed......: 0.00%   0 out of 580

    ✓ GET /aeronaves 200
    ✓ POST /aeronaves 201
    ✓ GET /aeronaves/:id 200
    ✓ DELETE /aeronaves/:id 200
```

```
CUSTOM
latencia.....................: avg=7.664638 min=0.5264  med=6.036  max=32.5321 p(90)=15.05142 p(95)=18.06208
tempo_processamento..........: avg=0.38841  min=0       med=0.3894 max=2.8097  p(90)=0.94228  p(95)=1.17077
tempo_resposta...............: avg=8.053048 min=0.8625  med=6.2925 max=32.6496 p(90)=15.61858 p(95)=18.92645

HTTP
http_req_duration..............: avg=8.64ms   min=862.5µs med=7.79ms max=32.64ms p(90)=15.73ms  p(95)=18.63ms
  { expected_response:true }...: avg=8.64ms   min=862.5µs med=7.79ms max=32.64ms p(90)=15.73ms  p(95)=18.63ms
http_req_failed................: 0.00%  0 out of 580
http_reqs......................: 580    19.283052/s

EXECUTION
iteration_duration.............: avg=1.03s    min=1.01s   med=1.03s  max=1.09s   p(90)=1.05s    p(95)=1.05s
iterations.....................: 145    4.820763/s
vus............................: 5      min=5        max=5
vus_max........................: 5      min=5        max=5

NETWORK
data_received..................: 256 kB 8.5 kB/s
data_sent......................: 73 kB  2.4 kB/s
```

Terceiro Teste – 10 Usuários (VirtualUsers/Vus)

```
     /\      Grafana   /‾‾/
  /\  /  \     |\  __   /  /
 /  \/    \    | |/ /  /   ‾‾\
/          \   |   (  |  (‾)  |
/ __________ \  |_|\_\  \_____/

    execution: local
       script: .\backend\k6\test_10Users.js
       output: -

    scenarios: (100.00%) 1 scenario, 10 max VUs, 1m0s max duration (incl. graceful stop):
             * default: 10 looping VUs for 30s (gracefulStop: 30s)


  TOTAL RESULTS

    checks_total.......: 1160    38.319574/s
    checks_succeeded...: 100.00% 1160 out of 1160
    checks_failed......: 0.00%   0 out of 1160

    ✓ GET /aeronaves 200
    ✓ POST /aeronaves 201
    ✓ GET /aeronaves/:id 200
    ✓ DELETE /aeronaves/:id 200
```

```
CUSTOM
latencia.....................: avg=9.121543 min=0.6703  med=7.00405 max=46.9401 p(90)=18.76963 p(95)=24.408655
tempo_processamento..........: avg=0.374846 min=0       med=0       max=3.7192  p(90)=1.01206  p(95)=1.2051
tempo_resposta...............: avg=9.496389 min=0.6703  med=7.19265 max=47.3675 p(90)=19.31882 p(95)=24.767595

HTTP
http_req_duration..............: avg=9.86ms   min=670.3µs med=8.64ms  max=47.36ms p(90)=18.63ms  p(95)=22.96ms
  { expected_response:true }...: avg=9.86ms   min=670.3µs med=8.64ms  max=47.36ms p(90)=18.63ms  p(95)=22.96ms
http_req_failed................: 0.00%  0 out of 1160
http_reqs......................: 1160   38.319574/s

EXECUTION
iteration_duration.............: avg=1.04s    min=1.02s   med=1.03s   max=1.08s   p(90)=1.06s    p(95)=1.07s
iterations.....................: 290    9.579893/s
vus............................: 10     min=10        max=10
vus_max........................: 10     min=10        max=10

NETWORK
data_received..................: 534 kB 18 kB/s
data_sent......................: 145 kB 4.8 kB/s
```

RESULTADOS:

Os testes de carga realizados com 1, 5 e 10 usuários virtuais (VUs) mostram que o backend se comporta de forma estável e eficiente mesmo com aumento de carga:

1. Checks:
    - 100% das validações (checks) foram bem-sucedidas em todos os testes.
    - Isso indica que todos os endpoints (GET, POST, GET por id e DELETE) responderam corretamente, sem erros.
2. Latência e tempo de resposta:
    - Com 1 VU, o tempo médio de resposta foi ~6,4 ms; com 5 VUs, ~8 ms; e com 10 VUs, ~9,5 ms.
    - O aumento é gradual e pequeno, mostrando que a API mantém baixa latência mesmo com mais usuários.
    - Percentis 90/95 indicam que a grande maioria das requisições completou rapidamente, sem picos preocupantes.
3. Tempo de processamento do servidor:
    - Permaneceu baixo (~0,3–0,6 ms em média), indicando que o backend processa as requisições muito rapidamente e a maior parte do tempo de resposta é causada pela rede e espera inicial.
4. Execução e throughput:
    - Iterações aumentam proporcionalmente ao número de VUs (30 → 145 → 290), mostrando que o sistema consegue lidar com mais requisições simultâneas.
    - Tempo médio por iteração permanece estável (~1,03 s), evidenciando consistência.
5. Rede:

- Volume de dados enviado e recebido aumenta proporcionalmente à carga, mas permanece dentro de limites baixos, sem indicar saturação.

A API suporta bem cargas de até 10 usuários simultâneos, com baixa latência, 100% de requisições corretas e tempo de processamento muito rápido. Não há sinais de gargalo ou degradação significativa com aumento de carga.

Esses resultados indicam que o backend está bem otimizado para o volume testado e pode escalar para cargas maiores com expectativa de manutenção da performance.