Documentação - Teste de Carga/Estresse
On Catalog

Registro:

Início	Fim
12/10/2024 - 13:00h	12/10/2024 - 15:03h

Descrição: Este documento descreve o teste de estresse realizado no OnCatalog em ambiente de produção, utilizando a ferramenta K6. O teste teve como objetivo avaliar o desempenho da aplicação ao acessar o endpoint de um dos catálogos disponíveis publicamente:

https://verificacao-e-validacao-production.up.railway.app/catalog_detail/?csrfmiddlewaretoken=IYlQwX02UTR4HTkRRjXqXWYUjtkTwuHrkg3P8SIEMr1QuVxNvhJ2XCnvT2DZeUzj&catalog_id=510387de-9b6f-4642-bd9d-982209b15818

O objetivo deste teste é simular um número crescente de usuários simultâneos acessando o serviço, monitorar o comportamento sob alta carga e verificar se o sistema suporta o aumento de tráfego sem degradação severa de performance ou falhas.

Requisito Associado:

RNF004			
Nome:	Desempenho e Escalabilidade		
Descrição:	O sistema deve ser capaz de suportar pelo menos 100 usuários simultâneos e responder a requisições em até 3 segundos, para que o usuário consiga usar o sistema mesmo em condições não previstas.		
Prioridade:	1	**Anexo:**	----
Entradas e pré-condições:	1. Carga estimada de usuários durante picos de acesso. 2. Estrutura de infraestrutura disponível (servidores, banco de dados, etc.).		
Saídas e pós-condições:	1. O sistema é capaz de atender às requisições de forma eficiente, mesmo em situações de alta demanda.		
Critérios de aceitação:			
1. O tempo de resposta médio para requisições deve ser inferior a 2 segundos em 95% das interações. 2. O sistema deve suportar pelo menos 100 usuários simultâneos sem perda de desempenho. 3. Em situações de carga máxima, o sistema deve ser capaz de escalar automaticamente para acomodar a demanda. 4. Testes de carga devem ser realizados periodicamente para garantir que o sistema possa lidar com picos de usuários sem comprometer o desempenho.			

Caso de teste 01

Código do Teste:

```
import http from 'k6/http';
import { sleep } from 'k6';

export const options = {
    // Config
    stages: [
      { duration: '2m', target: 100 }, // aumento gradual para 100 usuários em 2 miN
      { duration: '5m', target: 100 }, // manutenção de 100 usuários por 5 min
      { duration: '2m', target: 200 }, // aumento gradual para 200 usuáriOS
      { duration: '5m', target: 200 }, // manutenção de 200 usuários por 5 min
      { duration: '2m', target: 300 }, // aumento gradual para 300 usuários
      { duration: '5m', target: 300 }, // manutenção de 300 usuários por 5 min
      { duration: '2m', target: 400 }, // aumento gradual para 400 usuários
      { duration: '5m', target: 400 }, // manutenção de 400 usuários por 5 min
      { duration: '10m', target: 0 },  // redução gradual para 0 usuários
    ],
    thresholds: {
        // Testa se 95% das requisições estão abaixo de 3 segundos
        http_req_duration: ['p(95)<3000']    }
};

export default () => {
  const urlRes = http.get('endpoint de um catálogo aleatório');
  sleep(1);
};
```

Descrição:

- O teste começa com 0 usuários e gradualmente aumenta para 100 usuários em 2 minutos.
- Mantém 100 usuários ativos por 5 minutos.
- Em seguida, aumenta para 200 usuários ao longo de 2 minutos e mantém essa carga por 5 minutos.
- O mesmo padrão se repete para 300 usuários e, por último, para 400 usuários, aumentando gradualmente e mantendo essa carga por um período de 5 minutos.
- Após atingir o pico de 400 usuários, o teste reduz a carga de forma gradual para 0 usuários ao longo de 10 minutos.

Métrica de desempenho:

- O teste verifica se 95% das requisições feitas ao endpoint retornam com um tempo de resposta abaixo de 3 segundos.

Resultados do Caso de teste 01:

```
silvio@TERMINAL MINGW64 ~/OneDrive/Área de Trabalho/k6 teste


execution: local
script: k6_test.js
output: -

scenarios: (100.00%) 1 scenario, 400 max VUs, 38m30s max duration (incl. graceful stop):
  * default: Up to 400 looping VUs for 38m0s over 9 stages (gracefulRampDown: 30s, gracefulStop: 30s)

data_received..................: 275 MB 121 kB/s
data_sent......................: 3.6 MB 1.6 kB/s
http_req_blocked...............: avg=2.17ms   min=0s       med=0s     max=2.46s   p(90)=0s      p(95)=0s
http_req_connecting............: avg=984.94µs min=0s       med=0s     max=236.9ms p(90)=0s      p(95)=0s
✗ http_req_duration..............: avg=12.11s   min=227.32ms med=11.19s max=23.41s  p(90)=21.45s  p(95)=22.03s
    { expected_response:true }...: avg=12.11s   min=227.32ms med=11.19s max=23.41s  p(90)=21.45s  p(95)=22.03s
http_req_failed................: 0.00%  0 out of 39524
http_req_receiving.............: avg=9.42ms   min=0s       med=1.44ms max=5.09s   p(90)=3.86ms  p(95)=6.04ms
http_req_sending...............: avg=81.53µs  min=0s       med=0s     max=4.51ms  p(90)=263.6µs p(95)=305.78µs
http_req_tls_handshaking.......: avg=1.19ms   min=0s       med=0s     max=2.28s   p(90)=0s      p(95)=0s
http_req_waiting...............: avg=12.1s    min=226.15ms med=11.19s max=23.2s   p(90)=21.45s  p(95)=22.02s
http_reqs......................: 39524  17.330299/s
iteration_duration.............: avg=13.11s   min=1.22s    med=12.2s  max=24.41s   p(90)=22.46s  p(95)=23.03s
iterations.....................: 39524  17.330299/s
vus............................: 1      min=1          max=400
vus_max........................: 400    min=400        max=400

running (38m00.6s), 000/400 VUs, 39524 complete and 0 interrupted iterations
default ✓ [====================================] 000/400 VUs  38m0s
ERRO[2281] thresholds on metrics 'http_req_duration' have been crossed
```

ANÁLISE DOS RESULTADOS:

- O tempo médio de resposta foi de 12.11 sec, que excede o limite de 2 sec em 95% das interações, conforme estipulado nos critérios de aceitação.
    - Erro presente em: `http_req_duration`
- Taxa de Erro: Não houve falhas nas requisições (0.00%).
- Carga Máxima: O teste alcançou um pico de 400 usuários simultâneos, conforme planejado, mas o desempenho não atendeu aos critérios definidos.

CONCLUSÃO:

Como previsto, os resultados indicam que a aplicação não atende ao critério de desempenho estabelecido e previstos no RNF004, com um tempo de resposta médio muito superior ao limite aceito.

Resultado esperado: o caso de teste 01 falha.
Resultado obtido: o caso de teste 01 falha.

Caso de teste 02

Código do Teste:

```
import http from 'k6/http';
import { sleep } from 'k6';

export const options = {
      //config
       stages: [
        { duration: '2m', target: 100 }, // aumento gradual para 100 usuários em 2 min
        { duration: '5m', target: 100 }, // manutenção de 100 usuários por 5 min
        { duration: '2m', target: 50  }, // redução gradual para 50 usuários
        { duration: '2m', target: 0   }, // redução gradual para 0 usuários
    ],
    thresholds: {
        // Testa se 95% das requisições estão abaixo de 3 segundos
        http_req_duration: ['p(95)<3000']
    }
};

export default () => {
  const urlRes = http.get('endpoint de um catálogo aleatório');
  sleep(1);
};
```

Descrição:

- O teste começa com 0 usuários e aumenta gradualmente para 100 usuários em 2 minutos.
- Mantém 100 usuários ativos por 5 minutos.
- Posteriormente, reduz a carga para 50 usuários ao longo de 2 minutos.
- Por fim, reduz gradualmente a carga para 0 usuários em 2 minutos.

Métrica de desempenho:

- O teste verifica se 95% das requisições feitas ao endpoint retornam com um tempo de resposta abaixo de 3 segundos.

Resultados do Caso de teste 02:

```
silvio@TERMINAL MINGW64 ~/OneDrive/Área de Trabalho/k6 teste

execution: local
        script: k6_test.js
        output: -

     scenarios: (100.00%) 1 scenario, 100 max VUs, 11m30s max duration (incl. graceful stop):
              * default: Up to 100 looping VUs for 11m0s over 4 stages (gracefulRampDown: 30s, gracefulStop: 30s)

     data_received..................: 75 MB  114 kB/s
     data_sent......................: 978 kB 1.5 kB/s
     http_req_blocked...............: avg=2.65ms   min=0s       med=0s     max=5.62s    p(90)=0s      p(95)=0s
     http_req_connecting............: avg=935.39µs min=0s       med=0s     max=319.47ms p(90)=0s      p(95)=0s
   ✓  http_req_duration..............: avg=2.47s    min=224.87ms med=2.46s  max=2.90s    p(90)=2.77s   p(95)=2.79s
     http_req_failed................: 0.00%  0 out of 10825
     http_req_receiving.............: avg=12.21ms  min=0s       med=1.52ms max=9.93s    p(90)=4.06ms  p(95)=6.04ms
     http_req_sending...............: avg=85.93µs  min=0s       med=0s     max=1.2ms    p(90)=253.9µs p(95)=290.7µs
     http_req_tls_handshaking.......: avg=1.71ms   min=0s       med=0s     max=5.31s    p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=2.46s    min=223.89ms med=2.48s  max=6.81s    p(90)=4.13s   p(95)=4.42s
     http_reqs......................: 10825  16.373174/s
     iteration_duration.............: avg=3.47s    min=1.22s    med=3.49s  max=12.3s    p(90)=5.14s   p(95)=5.44s
     iterations.....................: 10825  16.373174/s
     vus............................: 1      min=1          max=100
     vus_max........................: 100    min=100        max=100


running (11m01.1s), 000/100 VUs, 10825 complete and 0 interrupted iterations
default ✓ [======================================] 000/100 VUs  11m0s
```

ANÁLISE DOS RESULTADOS:

- O tempo médio de resposta foi de 2.46 sec, que está abaixo do limite de 3 sec em 95% das interações, conforme estipulado nos critérios de aceitação.
- Taxa de Erro: Não houve falhas nas requisições (0.00%).
- Carga Máxima: O teste alcançou um pico de 100 usuários simultâneos, conforme planejado.

CONCLUSÃO:

Os resultados indicam que a aplicação atende ao critério de desempenho estabelecido e previsto no RNF004, com um tempo de resposta médio dentro do limite previsto.

Resultado esperado: o caso de teste 02 passa.
Resultado obtido: o caso de teste 02 passa.

REFERÊNCIAS:

https://k6.io/open-source/