Relatório de testes de desempenho

Projeto: Teste de desempenho de API simulada em Python
Ferramenta utilizada: K6
Responsável: Ana Vitória Araújo de Souza
Data da execução: 09/11/2025
Objetivo: Avaliar tempo de resposta, capacidade e disponibilidade de uma API simulada sob carga

---

1. VISÃO GERAL

O teste foi realizado sobre uma API desenvolvida em Python, simulando um endpoint de teste para fins de avaliação de desempenho. O objetivo foi validar os seguintes requisitos:

- **SLA:** tempo de resposta p95 inferior a 2 segundos
- **Capacidade:** suportar até 100 usuários simultâneos
- **Disponibilidade:** taxa de erro inferior a 1%

Utilizou-se a ferramenta K6 para simular múltiplos usuários virtuais (VUs) com validações de sucesso, tempo de resposta e estabilidade.

2. DETALHES DOS TESTES

A API foi submetida a três cenários distintos de carga: carga progressiva (Ramp-Up), pico de demanda (Spike) e resistência prolongada (Endurance). Foram simulados três perfis de usuários com comportamentos distintos: Navegante, Comprador e Robo. Cada perfil acessou endpoints específicos com pausas realistas entre requisições.

- **Cenário executado:**
    - **Ramp-Up:** crescimento gradual de 0 → 50 → 100 VUs;
    - **Spike:** 100 VUs simultâneos por 30 segundos;
    - **Endurance:** 100 VUs constantes por 30 minutos.
- **Perfis de Usuários:**
    - **Navegante:** acessa produtos e status com pausas leves;
    - **Comprador:** realiza transações e acessa endpoints mais pesados;
    - **Robo:** monitora status e força carga de CPU;
- **Resultados principais:**

O teste revelou que a API não atende aos requisitos de desempenho definidos:

- O sistema começou a degradar a partir de 50 VUs no Ramp-Up;
- No Spike e Endurance, o tempo de resposta ultrapassou 20 segundos;
- Retornados muitos erros 503: indicam sobrecarga no backend;
- O endpoint `/pagamentos` foi o mais afetado;

Apesar de funcional, o sistema não está preparado para uso em escala real sem otimizações.

3. RECOMENDAÇÕES

Seguem algumas recomendações para melhora do sistema em relação e desempenho e suportar os usuários requisitados:

- Aumentar o número de workers no servidor (ex: `uvicorn --workers 4`);
- Implementar cache e filas de requisição para reduzir latência;
- Otimizar o endpoint `/pagamentos`, que apresentou maior lentidão;
- Monitorar uso de CPU, memória e banco de dados durante testes de carga;
- Reavaliar a arquitetura da API para garantir escalabilidade acima de 100 usuários simultâneos.