Validação de Carga em Testes de Desempenho de APIs de Cadastro de Usuários: Garantindo Qualidade e Eficiência com Docker e K6

Pedro Ventura Lucunde Chimuco [1] Ana Claudia Garcia Barbosa. [2]

Resumo: Garantir a qualidade e eficiência das APIs é fundamental para o sucesso das aplicações web em ambientes competitivos. Este estudo analisa técnicas de validação de carga em testes de desempenho de uma API de cadastro de usuários, utilizando um ambiente docker-compose com armazenamento de dados no Cloud MongoDB. Testes com diferentes cargas de usuários virtuais (VUs) avaliaram o desempenho da API. Utilizando ferramentas como K6 para simulação de carga, InfluxDB para armazenamento de dados de desempenho e Grafana para visualização, os resultados mostraram uma alta taxa de sucesso das requisições, embora o tempo médio de resposta tenha aumentado consideravelmente com o aumento da carga. O estudo discute esses resultados, destaca as principais descobertas e apresenta recomendações para otimização e escalabilidade da API em cenários de alta demanda.

Palavras-chave: teste de desempenho; validação de carga; APIs; qualidade de software; alta demanda.

ABSTRACT: Ensuring the quality and efficiency of APIs is fundamental to the success of web applications in competitive environments. This study analyzes load validation techniques in API performance tests using a user registration API in a docker-compose environment with data storage in Cloud MongoDB. Tests were conducted with different loads of virtual users (VUs) to evaluate the API's performance. Using K6 for load simulation, InfluxDB for storing performance data, and Grafana for visualization, the results showed a high success rate of requests, although the average response time increased considerably with the load. The study discusses these results, highlights the main findings, and presents recommendations for optimizing and scaling the API in high-demand scenarios.

Keywords: performance testing; load validation; APIs; software quality; high demand.

[1] ppedroventura@unesc.net
[2] agb@unesc.net

1 INTRODUÇÃO

Atualmente, o bom desempenho de uma aplicação web é um dos principais objetivos no desenvolvimento, crucial para a satisfação do usuário e sucesso do negócio. O desempenho inadequado pode causar perda de consumidores, danos à marca, prejuízos financeiros e perda de confiança (Stejskal, 2018, tradução nossa). Em contextos competitivos, como a véspera da Black Friday em sistemas web para e-commerce, é crucial atender plenamente às expectativas do cliente; caso contrário, o sistema será rapidamente superado pelos concorrentes (Santos I. S. e Neto, 2011).

Momentos de maior uso da aplicação são também os mais importantes, nos quais o fracasso não é uma opção. Clark (2018, tradução nossa) enfatiza que a validação da carga em testes de desempenho de um sistema antes de sua implementação é de extrema importância, uma vez que muitos sistemas são projetados para um determinado número de usuários; porém, no momento do uso, não atendem aos requisitos, causando interrupções, travamentos, perda de dados e lentidão no processamento, ou no sistema web como um todo (Correia; Couto, 2015).

Testes de carga são essenciais para entender o comportamento de aplicações web sob diferentes níveis de estresse. Segundo Menascé (Menascé, 2002, tradução nossa), esses testes podem identificar pontos de falha potencial e áreas que precisam de otimização, garantindo assim a estabilidade e a capacidade de resposta do sistema. Ademais, Hendayun, Ginanjar e Ihsan (Hendayun Mokhamd e Ginanjar, 2023, tradução nossa) destacam que os testes de carga são essenciais para identificar gargalos e pontos críticos que podem comprometer a eficiência da aplicação em situações de alta demanda. Simular cenários de uso realistas é vital para obter resultados precisos.

O teste de software é uma parte fundamental no desenvolvimento de produtos de software de qualidade. Comparado a outras fases do ciclo de vida do software, o teste pode representar até 60% do esforço total de desenvolvimento, em termos de tempo e dinheiro (Costa, 2020, tradução nossa). Desempenho é uma das características mais importantes em produtos de software de qualidade. Souza (Souza, 2018, tradução nossa) observa que avaliar um software não é tarefa trivial, pois requer conhecimento prático de ferramentas de teste e compreensão dos aspectos teóricos relacionados. A qualidade de software é a preocupação central da Engenharia de Software, e o teste é a abordagem mais comumente usada

para assegurá-la (Costa, 2020, tradução nossa).

Neste estudo, exploramos e aplicamos técnicas de validação de carga em testes de desempenho de uma API de cadastro de usuários, utilizando um ambiente docker-compose, com armazenamento de dados no Cloud MongoDB. A API deve cumprir requisitos específicos para garantir um desempenho adequado: o cadastro de usuários deve ser realizado com sucesso em até 2 segundos, mesmo sob a carga de até 100 usuários simultâneos, com uma margem de erro de pelo menos 1%.

Os resultados destes testes são cruciais para identificar os pontos onde a API necessita de otimizações. A análise dos dados obtidos permite desenvolver estratégias para melhorar a eficiência e escalabilidade da API, garantindo que ela atenda às expectativas dos usuários mesmo em cenários de alta demanda. Este artigo apresenta os métodos utilizados, os resultados dos testes e recomendações para a otimização e escalabilidade da API.

2 TRABALHOS CORRELATOS

A validação de carga e testes de desempenho são áreas amplamente estudadas na Engenharia de Software. Diversos estudos exploram a importância desses testes para garantir a robustez e eficiência de sistemas web sob diferentes níveis de demanda.

Menascé (2002) discute a importância dos testes de carga para entender o comportamento de aplicações web sob diferentes níveis de estresse. O autor destaca que tais testes são cruciais para garantir que a aplicação possa lidar com o tráfego esperado sem comprometer seu desempenho. Este estudo fornece uma base teórica sólida que reforça a necessidade do trabalho, pois a metodologia de Menascé é empregada para projetar os cenários de carga utilizados nos testes. Enquanto Menascé se concentra em aplicações web de forma geral, esse trabalho foca especificamente em APIs de cadastro de usuários. Além disso, utilizamos ferramentas modernas como Docker e K6, que não estavam disponíveis na época de Menascé, permitindo uma abordagem mais atual e prática.

Hendayun, Ginanjar e Ihsan (2023) analisam o desempenho de aplicações utilizando métodos de teste de carga e estresse em serviços de API. Os autores ressaltam que esses testes ajudam a identificar gargalos e pontos críticos que podem comprometer a eficiência da aplicação em situações de alta demanda. Este estudo é particularmente relevante para o trabalho, pois utilizamos uma metodologia similar para identificar gargalos

na API de cadastro de usuários. Embora ambos os estudos utilizem testes de carga em serviços de API, esse trabalho avança ao integrar Docker-Compose para gerenciar o ambiente de teste, proporcionando maior controle e replicabilidade dos resultados. Além disso, utilizamos InfluxDB e Grafana para uma análise de dados mais detalhada e visualmente intuitiva.

Santos, Santos Neto e Resende (2011) exploram a relevância do desempenho de sistemas web em contextos de e-commerce, especialmente durante eventos de alta demanda como a Black Friday. Os autores mostram que falhas de desempenho nesses momentos críticos podem resultar em significativas perdas financeiras e danos à reputação da marca. O estudo enfatiza a necessidade de uma preparação rigorosa e de testes extensivos para garantir que os sistemas possam suportar picos de tráfego sem degradação de desempenho. Enquanto o estudo de Santos et al. se concentra em sistemas web para e-commerce, esse trabalho foca especificamente em APIs de cadastro de usuários, um componente essencial em muitos sistemas web. Este estudo também contribui com uma análise mais técnica e detalhada utilizando ferramentas modernas de simulação de carga e monitoramento de desempenho.

Correia e Couto (2015) discutem casos em que sistemas web falharam devido à falta de testes adequados, resultando em interrupções, travamentos e perda de dados. Esses estudos sublinham a importância de uma abordagem rigorosa na validação de carga para assegurar a qualidade e a eficiência das aplicações. Os autores sugerem que a realização de testes de carga deve ser uma prática padrão no ciclo de desenvolvimento de software para evitar falhas catastróficas em produção.

Santos et al. (2011) propõem um método para a geração de testes de desempenho e estresse a partir de testes funcionais, visando popularizar esses testes essenciais para sistemas web. A pesquisa concluiu que o uso do método e da ferramenta pode gerar ganhos em termos de redução de esforço na criação desses testes de requisitos não-funcionais. Embora Santos et al. ofereçam uma abordagem para a geração de testes, esse trabalho foca na aplicação prática de testes de carga específicos para APIs de cadastro de usuários, utilizando um conjunto de ferramentas que melhora a eficiência e precisão dos testes. Este estudo também inclui uma análise de dados mais avançada e recomendações práticas para otimização e escalabilidade.

3 MATERIAIS E MÉTODOS

Esta pesquisa explora a validação de carga em testes de desempenho de APIs utilizando uma API de cadastro de usuários em um ambiente docker-compose. Para garantir a replicabilidade e controle do ambiente de teste, foram utilizadas diversas ferramentas e metodologias descritas a seguir.

3.1 Configuração do Ambiente de Teste

A configuração do ambiente de teste foi realizada utilizando Docker, com o seguinte arquivo docker-compose.yml:

Código 1: Pseudocódigo de configuração do ambiente docker-compose

```
version: '3.9'
services:
  api:
    image: vntura/api_users:1.0
    environment:
      - NODE_ENV=production
    ports:
      - "3333:3333"
    cpus: '0.5'
    mem_limit: 512M
  k6:
    image: grafana/k6:latest
    network_mode: host
```

Fonte:Do autor (2024) - código.

3.2 Ferramentas Utilizadas

- K6: Ferramenta de teste de carga e desempenho de código aberto, utilizada para simular a carga de usuários e medir o desempenho da API.
- InfluxDB: Banco de dados time-series utilizado para armazenar os dados de desempenho coletados durante os testes.
- Grafana: Plataforma de visualização de dados, utilizada para criar painéis e gráficos a partir dos dados armazenados no InfluxDB.
- Docker: Plataforma de contêineres que facilita a criação e gestão do ambiente de teste, garantindo que todos os componentes necessários estejam devidamente configurados e isolados.

3.3 Procedimentos de Teste

Para cada teste de carga, foram seguidos os seguintes procedimentos:

1. Preparação do Ambiente: Configuração e inicialização dos containers Docker para InfluxDB, Grafana, API e K6.

2. Desenvolvimento de Scripts de Teste: Criação de scripts de teste no K6 para simular diferentes cargas de usuários (100, 200, 400, 800, 1200, 1600, 2000 VUs).

Código 2: Exemplo de Scripts de Teste de carga

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

export const options = {
  // Key configurations for avg load test in this section
  stages: [
    { duration: '5m', target: 100 }, // traffic ramp-up from 1 to 100 users over 5 minutes.
    { duration: '30m', target: 100 }, // stay at 100 users for 30 minutes
    { duration: '5m', target: 0 }, // ramp-down to 0 users
  ],
};

export default () => {
  const urlRes = http.get('https://test-api.k6.io');
  sleep(1);
};
```

Fonte: (K6, 2024)

3. Execução dos Testes: Execução dos scripts de teste utilizando o K6, monitorando o tempo de resposta, o número de requisições por segundo e a taxa de sucesso.

4. Coleta de Dados: Armazenamento dos dados de desempenho no InfluxDB.

5. Visualização dos Resultados: Criação de painéis no Grafana para visualizar os dados de desempenho.

6. Análise dos Resultados: Análise dos gráficos e tabelas gerados para identificar possíveis gargalos e pontos de otimização.

3.4 Análise de Dados

Os dados coletados foram analisados para avaliar o desempenho da API sob diferentes cargas de usuários. A seguir, apresentamos algumas das métricas analisadas:

- Tempo Médio de Resposta: Tempo médio que a API leva para responder a uma requisição.
- Requisições por Segundo: Número de requisições que a API pode processar por segundo.
- Taxa de Sucesso: Percentual de requisições que foram processadas com sucesso pela API.
- Os resultados dos testes são apresentados na seção de Resultados e Discussão.

4 RESULTADOS E DISCUSSÃO

Os testes foram realizados simulando diferentes cargas de usuários virtuais (VUs) para avaliar o desempenho da API. A seguir, os resultados obtidos são apresentados em formato tabular e gráfico para melhor visualização e análise.

4.1 Resultados dos Testes de Desempenho

Figura 1: Descrição da Tabela: Esta tabela mostra os resultados dos cenários de teste de desempenho.

Cenário	VUs Máx	Duração (s)	Duração Média (ms)	Requisições (req/s)	Sucesso (%)
1	100	240	30,76	72,93	100,00%
2	200	240	653,91	90,94	100,00%
3	400	240	2,17	94,64	100,00%
4	800	240	4,96	101,42	100,00%
5	1200	240	7,95	102,12	100,00%
6	1600	240	10,9	103,18	100,00%
7	2000	252	13,93	98,69	100,00%

Fonte:Do autor (2024)

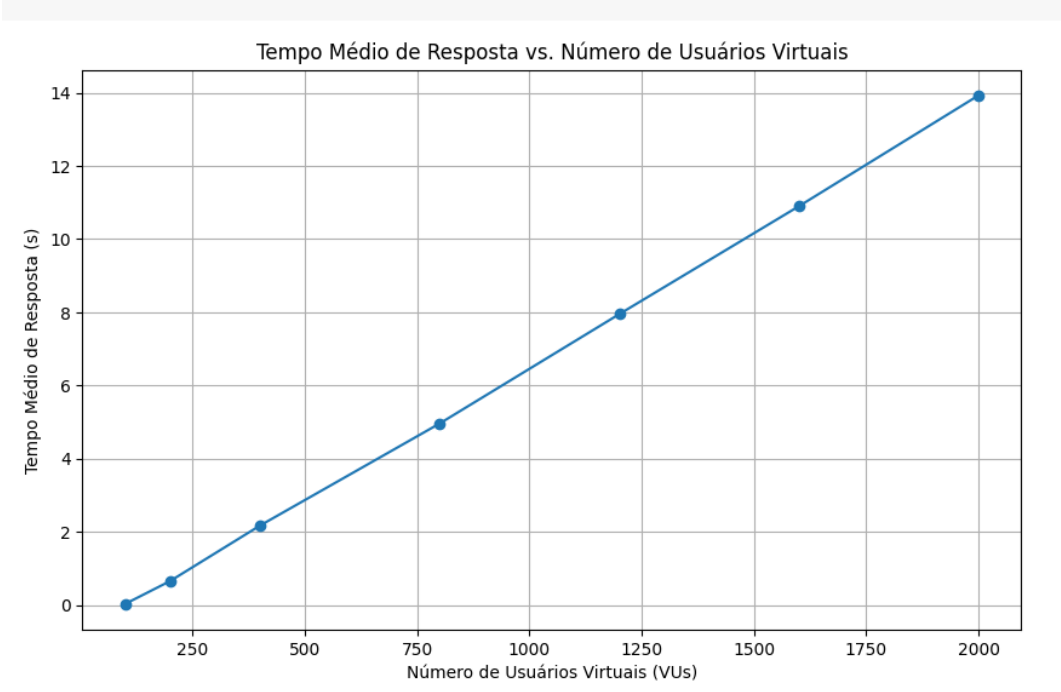
4.2 Análise dos Resultados

A análise dos resultados mostra que o tempo de resposta da API aumenta significativamente com a carga de usuários. Enquanto a API mantém uma alta taxa de sucesso de 100% em todos os testes, a latência cresce proporcionalmente à carga, o que indica a necessidade de otimizações para melhorar a eficiência sob alta demanda.

- Tempo de Resposta: O tempo de resposta aumentou significativamente com a carga de usuários, indicando a necessidade de otimização para manter a eficiência sob alta demanda. A partir de 400 VUs, o tempo médio de resposta começou a aumentar exponencialmente, atingindo 13,93ms com 2000 VUs.

Na figura 2 é possível observar o gráfico ilustra o aumento linear no tempo médio de resposta da API conforme o número de usuários virtuais (VUs) cresce. O tempo de resposta aumenta de forma constante, atingindo cerca de 14 segundos com 2000 VUs, indicando uma sobrecarga progressiva da API com o aumento da carga de usuários.

Figura 2: Tempo Médio de Resposta em Função do Número de Usuários Virtuais.

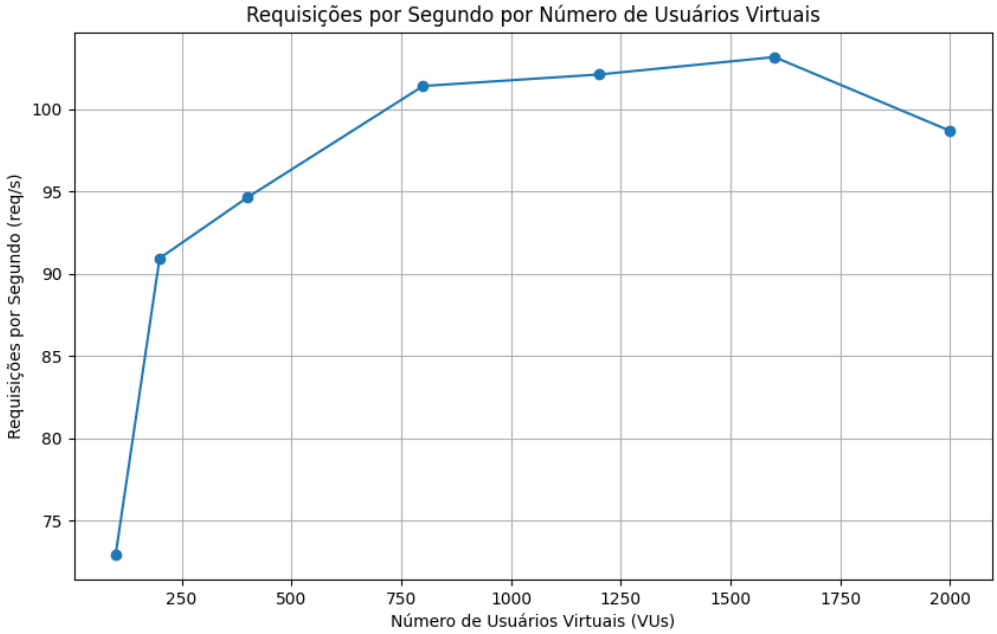


Fonte:Do autor (2024)

- Número de Requisições por Segundo: A API conseguiu manter um número constante de requisições por segundo até certo ponto, mas a eficiência começou a diminuir em cargas muito altas. O número de requisições por segundo permaneceu relativamente estável, variando entre 72,93 e 103,18, mostrando que a API pode processar um número significativo de requisições simultâneas.

Na figura 3 é possível observar o gráfico mostrando a relação entre o número de usuários virtuais (VUs) e o número de requisições por segundo (req/s) que a API pode processar. Inicialmente, o número de requisições por segundo aumenta conforme o número de VUs aumenta, atingindo um pico em torno de 103 req/s com 800 a 1200 VUs. Após esse ponto, o desempenho da API começa a declinar, com o número de requisições por segundo caindo para cerca de 95 req/s quando o número de VUs chega a 2000. Isso indica que a API consegue lidar bem com até 1200 VUs, mas apresenta gargalos e diminuição de eficiência com cargas mais altas, sugerindo a necessidade de otimizações para manter o desempenho sob alta demanda.

Figura 3: Requisições por Segundo em Função do Número de Usuários Virtuais.



Fonte:Do autor (2024)

- Taxa de Sucesso e Erro: A taxa de sucesso foi de 100% em todos os cenários, o que é um bom indicador da estabilidade da API, apesar do aumento na latência. No entanto, é importante monitorar o comportamento da API sob cargas ainda maiores para garantir que a taxa de sucesso permaneça alta.

5 CONCLUSÃO

Os testes de desempenho realizados demonstraram que a API é capaz de lidar eficientemente com até 100 usuários simultâneos, com tempos de resposta aceitáveis. No entanto, à medida que a carga aumenta, a latência das requisições também aumenta, destacando a necessidade de otimização para cenários de alta demanda.

A validação de carga mostrou-se crucial, pois os resultados dos testes revelaram como a API se comporta sob diferentes níveis de estresse. Identificou-se que, embora a API mantenha alta estabilidade com uma taxa de sucesso das requisições de 100%, a latência aumenta significativamente com a carga.

Embora os resultados sejam promissores, é importante reconhe-

cer algumas limitações que podem ter impactado os resultados. Os testes foram realizados com uma API rodando localmente, enquanto os dados eram salvos em um banco de dados MongoDB na nuvem. Essa configuração pode não refletir completamente as condições de um ambiente de produção, onde fatores externos, como variabilidade de rede e carga, podem influenciar significativamente a performance. Além disso, a configuração de hardware utilizada nos testes pode não representar um ambiente de produção típico. O desempenho da API pode variar consideravelmente em diferentes configurações de hardware, e a extrapolação dos resultados para outros ambientes deve ser feita com cautela. Os testes focaram na performance de uma única instância da API. A escalabilidade horizontal, que envolve a distribuição da carga de trabalho entre várias instâncias, não foi abordada. A implementação dessa técnica poderia fornecer insights adicionais sobre o comportamento da API em cenários de alta demanda.

Com base no aprendizado durante esta pesquisa, sugerem-se para trabalhos futuros: realizar testes comparativos com outras técnicas de otimização e escalabilidade, como caching, balanceamento de carga e otimizações de código, para determinar as melhores práticas para a API de cadastro de usuários; executar testes de desempenho em diferentes ambientes e configurações de hardware para avaliar a robustez e a escalabilidade da API em cenários variados, incluindo a utilização de diferentes provedores de nuvem e configurações de rede; desenvolver um pipeline de CI/CD que inclua testes de desempenho automatizados e monitoramento contínuo da API para garantir que as melhorias de desempenho sejam continuamente verificadas e implementadas; avaliar o impacto de atualizações e a adição de novas funcionalidades na performance da API. Estudos de caso que acompanham a evolução da API ao longo do tempo podem fornecer insights valiosos sobre a manutenção de desempenho em longo prazo.

REFERÊNCIAS

CLARK, M. *How the BBC builds websites that scale*. 2018. Acesso em: 15 ago. 2022. Disponível em: <https://www.creativebloq.com/features/how-the-bbc-builds-websites-that-scale>. 2

CORREIA, T. d. A.; COUTO, T. Teste de performance em aplicações web: Garantindo o desempenho de uma aplicação. *Interfaces Científicas - Exatas e Tecnológicas*, v. 1, n. 3, p. 23–30. Disponível em: <https://periodicos.set.edu.br/exatas/article/view/2494>. 2, 4

COSTA, V. e. a. Taxonomy of performance testing tools: A systematic literature review. In: *Proceedings of the 35th Annual ACM Symposium on*

Applied Computing. [S.l.: s.n.]: [s.n.], 2020. p. 1997–2004. 2, 3

HENDAYUN MOKHAMD E GINANJAR, A. e. I. Y. Analysis of application performance testing using load testing and stress testing methods in api service. *JURNAL SISFOTEK GLOBAL*, STMIK Bina Sarana Global, v. 13, n. 1, p. 28. ISSN 2088-1762. 2, 3

K6. *Average-load testing: A beginner's guide*. 2024. Acesso em: 24 maio 2024. Disponível em: <https://grafana.com/blog/2024/01/30/average-load-testing/>. 6

MENASCé, D. A. Load testing of web sites. *IEEE Internet Computing*, v. 6, n. 4, p. 70–74. ISSN 10897801. 2, 3

SANTOS I. S. E NETO, P. A. S. e. R. R. Geração de testes de desempenho e estresse a partir de testes funcionais. *Revista de Informática Teórica e Aplicada*, v. 17, p. 174. ISSN 01034308. 2, 4

SOUZA, T. S. d. *Testes de Desempenho de Software: Teoria e Prática*. [S.l.]: [S.l.: s.n.], 2018. 139-171 p. ISBN 9788576694564. 2

STEJSKAL, B. J. *Performance Testing and Analysis of QPID Dispatch Router*. [S.l.]: [S.l.: s.n.], 2018. 65-71 p. 2