IV SIMGETEC: SIMPÓSIO DE GESTÃO E TECNOLOGIA

Centro Estadual de Educação Tecnológica Paula Souza
Faculdade de Tecnologia de Carapicuíba
13 a 16 de Junho de 2018
ISSN: 2526-6241

AUTOMAÇÃO DE TESTE DE DESEMPENHO

Alexandre Girão Bezerra de Menezes[1] – FATEC Carapicuíba

Leandro Ribeiro de Souza[2] – FATEC Carapicuíba

Luiza Barbosa Lustri[3] – FATEC Carapicuíba

Prof. Me. Jadir Custódio Mendonça Júnior[4] – FATEC Carapicuíba (Orientador)

RESUMO

Grande parte das companhias modernas utilizam aplicações *web* para tornar viável seu negócio. Estatísticas mostram que os usuários de aplicações web consideram um tempo ideal até dois segundos para obter uma resposta da aplicação. Sendo assim, as empresas têm se preocupado cada vez mais com o desempenho e a qualidade de suas aplicações, realizando testes para garantirem uma experiência ainda melhor para seus usuários. A aplicação de testes de desempenho garante agilidade na tomada de decisões e consequentemente rápida atuação aos problemas que os usuários possam ter. O intuito do projeto é apresentar a viabilidade e a facilidade que uma empresa pode ter ao automatizar processos de testes de desempenho mediante os processos de qualidade de *software*.

Palavras-chave: Automação. Testes de desempenho. Qualidade de Software.

ABSTRACT

A large majority of modern companies use web applications to make their business viable. Statistics show that web application users consider an ideal response time of up to 2 seconds for an application response. As such, companies have been increasingly concerned with the performance and quality of their applications, doing tests to ensure an even better experience for their users. An application of performance tests, ensure agility in decision making and consequently fast acting on the problems that users eventually have. This project intends to show the feasibility and easiness that a corporation can have when using automating performance testing processes through software quality processes.

Keywords: *Automation. Performance tests. Software Quality.*

[1] - Análise e Desenvolvimento de Sistemas – *e-mail*: alexgirao@gmail.com
[2] - Análise e Desenvolvimento de Sistemas – *e-mail*: souza.ribeiro92@gmail.com
[3] - Análise e Desenvolvimento de Sistemas – *e-mail*: luiza_lustri@hotmail.com
[4] - Me. Engenharia Elétrica – *e-mail*: jadir.junior@fatec.sp.gov.br – Orientador

1 INTRODUÇÃO

Usuários de websites estão cada vez menos pacientes com a baixa velocidade em que as páginas carregam. De acordo com John Mueller, Analista *Webmaster* de Tendências da Google, considera-se carregamento ideal de uma página web entre dois e três segundos.

Segundos a mais no carregamento de uma aplicação pode tornar-se fator decisivo na escolha de um fornecedor. Com a alta concorrência entre as empresas, não basta apenas oferecer os mesmos serviços, é preciso oferecer também alta disponibilidade e desempenho das aplicações para garantir a vantagem competitiva.

De acordo com uma pesquisa realizada pela Akamai, empresa de *Internet* estadunidense, juntamente com a Forrester Consulting, grande empresa de pesquisa e consultoria mundial, dos 1.048 consumidores *online* entrevistados, 52% afirmam que a velocidade está diretamente associada com a fidelidade a um determinado *site*. Ainda segundo a pesquisa, "outro fator relevante é o quanto uma página lenta tem o poder de distrair os consumidores, tirando o foco do processo de venda". Dentre estes usuários, 14% irão visitar outro *site* e 23% irão interromper a navegação e sairão de seus computadores.

O objetivo do projeto é desenvolver uma solução de qualidade de *software* integrado com ciclo de vida da aplicação para proporcionar rápidas interações de desenvolvimento e testes, assim como diminuição de bugs, instabilidades e indisponibilidade em sistemas críticos.

Portanto, percebe-se a importância da utilização de meios que otimizem o tempo gasto nos acessos às aplicações e proporcionem aos usuários a melhor experiência para que possam atingir seus objetivos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Teste e Qualidade de Software

Existem diversas tentativas de definição sobre o que é teste de *software*. Todas elas generalizam e conduzem ao mesmo conceito: "teste de *software* é uma das atividades que busca contribuir para a melhoria da qualidade do *software*", Barbosa e Torres (2011).

Devido a particularidades dos *softwares* - como constantes mudanças e, muitas vezes, complexidade -, há certa dificuldade na realização dos testes, caracterizada pelo alto custo dos processos, falta de profissionais especializados na área, falta de conhecimento sobre a relação custo/benefício dos testes e, sem dúvidas, a atenção com os testes somente na fase final do

projeto. Em função do tempo exigido pelas atividades de teste, as empresas de desenvolvimento de *softwares* utilizam equipes pouco especializadas ou simplesmente encurtam essa etapa para evitar maiores gastos. Segundo Eliza e Lagares (2010, apud Barbosa e Torres, 2011), "algumas empresas preferem trazer profissionais de outras áreas como Help Desk, Implantação, Desenvolvimento, para os cargos da área de Teste simplesmente por entenderem que para Testar, conhecer o negócio é mais do que suficiente". Esses fatores têm influência direta na qualidade do *software* e, quanto menos levados em consideração, maior a chance de entrega de produtos com erros, levando à insatisfação do cliente.

2.2 Automação de Teste: definição

As empresas estão cada vez mais dependentes das tecnologias, de tal forma que essa dependência determina a viabilidade do negócio. Em função disso, qualquer risco deve ser mitigado. Segundo Lages (2010), "automatizar os testes nada mais é do que repassar para o computador as atividades de testes que normalmente são realizadas de forma manual", ou seja, a automação de testes transforma uma rotina manual em *scripts* executados por *softwares*, garantindo a qualidade dos *softwares* e sistemas.

As contínuas repetições dos testes permitem simular diversos cenários específicos, evitando que ações importantes sejam ignoradas, muitas vezes por falha humana, e garantindo a abrangência dos testes em inúmeras situações. A grande vantagem dos testes automatizados é a possibilidade de criar cenários de testes bem elaborados e de maior complexidade quando comparados aos realizados manualmente. Isso acontece devido a utilização de códigos interpretados por máquinas, ou seja, programação, que agilizam o processo de elaboração das situações.

2.2.1 Automação de Teste: Apache JMeter

O Apache JMeter[5] é uma aplicação desenvolvida em Java[6] para executar testes funcionais e medir o desempenho de aplicações. Segundo Oliveira (2016), "Quando falamos em teste de performance, sempre a primeira ferramenta que nos vem à cabeça é o JMeter, isso porque é um *software* que está há um tempo no mercado e tem vários projetos de sucesso relatados na comunidade."

---

[5] JMeter: https://jmeter.apache.org/
[6] Linguagem de programação orientada a objetos. https://www.java.com/pt_BR/

O objetivo do JMeter é prover cenários de testes com maior realidade e confiabilidade aos sistemas. "Portanto, os testes de carga devem simular o mais próximo possível da realidade de utilização, cenários realistas ajudam a minimizar os efeitos da subestimação ou superestimação dos tempos de resposta das aplicações", de acordo com Urbano (2012). De forma resumida, é uma ferramenta que permite que um profissional grave em scripts as requisições enviadas durante a interação com aplicações *web* (*sites*, serviços, etc.), podendo posteriormente simular usuários virtuais interagindo com essas aplicações.

2.2.2 Automação de Teste: InfluxDB

A ferramenta InfluxDB[7] é utilizada para armazenar dados em ordem cronológica, funcionando como uma base de dados serial, muito usada em sistemas para análise, monitoramento e armazenamento das informações. Sua integração com Apache JMeter facilita o processo de automação, pois não é necessário desenvolver a integração entre as duas ferramentas, poupando tempo no processo. Além disso, o InfluxDB também possui uma integração HTTP[8], possibilitando que bibliotecas em várias linguagens integrem com esse banco de dados.

2.2.3 Automação de Teste: Grafana

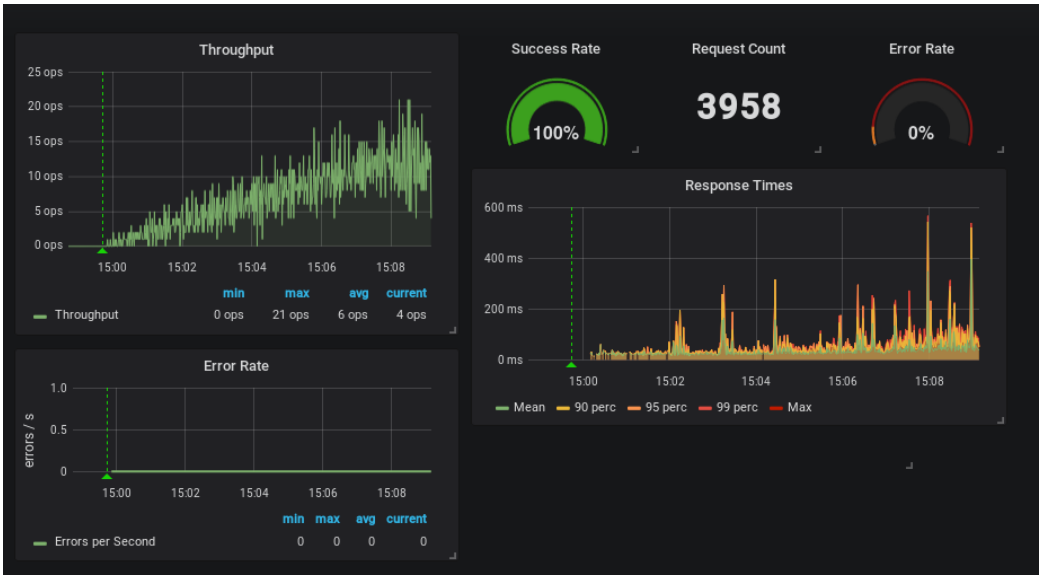
Grafana[9] é uma ferramenta de análise e visualização de métricas utilizada em diferentes mercados para visualização de dados em tempo real. Segundo Diedrich (2017), "O Grafana talvez seja uma das mais completas ferramentas para criação e exibição de gráficos, e uma ferramenta extremamente flexível e adaptável, tendo como input diversos tipos de backend", o que facilita ainda mais, neste caso, a automação do teste de desempenho.

É uma aplicação que permite fácil entendimento dos dados por utilizar gráficos para apresentá-los.

---

[7] InfluxDB: https://www.influxdata.com/
[8] HTTP é um protocolo de comunicação utilizado para transferência de dados entre redes de computadores.
[9] Grafana: https://grafana.com/

Figura 1 - Tela do Grafana apresentando telemetria dos testes



Fonte: Elaborado pelos autores

2.2.4 Automação de Teste: Jenkins

A utilização de uma ferramenta de integração contínua garante que qualquer alteração no código do *software* seja validada de forma rápida dentro da solução, possibilitando a investigação de *bugs* com agilidade, por exemplo. Segundo Fowler (2006, apud Gomes 2016), "muitos times acham que essa abordagem leva a uma significante redução nos problemas de integração e permite que um time desenvolva *software* coeso mais rapidamente".

Jenkins[10] é uma solução de integração contínua, ou seja, possibilita análise de códigos e detecção de bugs com antecedência. Esta ferramenta possui uma série de funcionalidades, dentre elas a automação de testes. Por ter código fonte aberto, pode ser usado para automatizar todos os tipos de tarefas relacionados à criação, teste ou distribuição de *software*.

3 PROCEDIMENTOS METODOLÓGICOS

Para realizar um teste de desempenho é necessário levantar algumas informações importantes tanto para análise, quanto para a execução dos testes. Conhecer a arquitetura do sistema é algo fundamental em uma análise de desempenho, assim como conhecer o fluxo lógico da aplicação, as tecnologias envolvidas e monitorar o seu funcionamento.

O método de pesquisa utilizado neste projeto é o quantitativo, apoiando-se em pesquisa exploratória para o embasamento teórico do projeto. Conforme Prodanov e Freitas

[10] Jenkins: https://jenkins.io/

(2013, p. 69), a pesquisa quantitativa "considera que tudo pode ser quantificável, o que significa traduzir em números opiniões e informações para classificá-las e analisá-las. Requer o uso de recursos e de técnicas estatísticas". Para Silva e Menezes (2001 apud SILVA, 2004):

> "A pesquisa exploratória visa proporcionar maior familiaridade com o problema com vistas a torná-lo explicito ou a construir hipóteses. Envolve levantamento bibliográfico; entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; análise de exemplos que estimulem a compreensão. Assume, em geral, as formas de Pesquisa Bibliográficas e Estudos de Caso".

Quanto aos procedimentos, o projeto foi desenvolvido a partir de estudo de caso de clientes da empresa Yaman Tecnologia[11] e pesquisa bibliográfica com a utilização de artigos e livros sobre teste e qualidade de *software*.

Ainda segundo Silva e Menezes (2001 apud SILVA, 2004), "a pesquisa bibliográfica é elaborada a partir de material já publicado, constituído principalmente de livros; artigos de periódicos e atualmente com material disponibilizado na *internet*".

De acordo com Gil (1988, apud SILVA, 2004), "o estudo de caso é caracterizado pelo estudo profundo e exaustivo de um ou poucos objetos, de maneira que permita seu amplo e detalhado conhecimento".

4 DESENVOLVIMENTO

Este estudo de caso mostra a importância de automatizar testes de desempenho que são executados em determinadas aplicações de uma empresa, podendo mensurar impacto destes testes e monitorar as métricas geradas. Este cenário permitiu que a empresa Yaman Tecnologia desenvolvesse uma solução integrada entre diversas ferramentas, podendo assim, com o auxílio da solução, observar os resultados de forma rápida e auxiliar na tomada de decisões do cliente.

4.1 Requisitos Estabelecidos

Para atender a necessidade do cliente, alguns requisitos deveriam ser atendidos durante o desenvolvimento da solução. Tais foram:

---

[11] Yaman Tecnologia: https://www.yaman.com.br/

- Utilização de ferramentas de código aberto;
- Solução com o mínimo de intervenção manual.

O uso de código aberto evita que o cliente fique preso à um fornecedor específico. A automação do ciclo de construção e teste do projeto de *software* torna todo o processo direto e objetivo.

4.2 Arquitetura da solução

Para atender ao requisito de intervenção mínima, a arquitetura foi projetada utilizando contêineres Docker[12], pois a partir da sua principal funcionalidade – empacotamento de uma aplicação ou ambiente dentro de um contêiner – a administração destes ambientes isolados tornou-se mais rápida e determinística. Além desses pontos, a solução baseada em contêiner facilita a validação em ambiente controlado e a implantação em ambiente de *data center*.

4.3 Definição da solução

Após o entendimento com o cliente, foram levantadas as funcionalidades mais importantes:

- Iniciar o teste no momento em que uma versão nova da aplicação for liberada;
- O *script* de teste e os parametros de carga configurados por projeto;
- Armazenar os dados provenientes da injeção de carga;
- Gerar gráficos com as métricas do teste;
- Controle de parada automático e manual do teste.

Estabelecendo, assim, o ponto inicial do desenvolvimento da solução.

4.4 Construção da solução

O Apache JMeter, citado anteriormente, foi utilizado para desenvolver os *scripts* que interagem com a aplicação do cliente, simulando usuários e injetando carga. Esta ferramenta possui uma funcionalidade nativa, que permite a comunicação com um banco de dados InfluxDB e também permite que os *logs* (registros) gerados pela ferramenta sejam armazenados em tabelas deste banco de dados. O Grafana, também utilizado no desenvolvimento da solução, da mesma forma que o Apache JMeter possui comunicação

---

[12] Docker: promovido pela empesa Docker, Inc., é uma tecnologia de software que, a partir de contêineres, permite o teste e a implantação de aplicações isoladamente para qualquer máquina que tenha o Docker instalado. https://www.docker.com/.

nativa com InfluxDB, permitindo que todas as métricas armazenadas em tabelas sejam exibidas em um painel de instrumentos, onde são mostradas métricas como tempo de resposta, quantidade de requisições por segundo, tempo mínimo e máximo de cada requisição e outras informações relevantes, conforme mostrado na figura 2.

Figura 1 - Métricas de testes apresentadas pela aplicação Grafana

Metrics Overview

Time	requestName	Count	Avg	Min	Median	90%	95%	99%	Max	Error Rate
2018-05-20 14:58:43	show	4.02 K	15.19	2.00	6.00	33.00	53.00	141.00	501.00	0%
2018-05-20 14:58:43	search	4.03 K	19.27	4.00	11.00	37.00	54.00	139.00	342.00	0%
2018-05-20 14:58:43	post	3.96 K	41.43	13.00	28.00	71.00	104.00	239.00	566.00	0%
2018-05-20 14:58:43	next 2	3.98 K	18.83	4.00	11.00	38.00	56.00	125.00	355.00	0%
2018-05-20 14:58:43	next 1	3.99 K	19.22	4.00	11.00	38.00	56.00	136.00	379.00	0%
2018-05-20 14:58:43	new	3.97 K	10.08	2.00	5.00	19.00	33.00	93.00	315.00	0%
2018-05-20 14:58:43	list ibm	4.01 K	19.00	4.00	11.00	38.00	53.00	127.00	353.00	0%
2018-05-20 14:58:43	list all	4.05 K	30.06	9.00	22.00	52.00	72.00	148.00	437.00	0%

Fonte: Elaborado pelos autores

Para orquestrar todo o processo, foi utilizado o Jenkins. Por esta ferramenta ser um servidor de integração contínua, existem diversas funcionalidades que permitem a comunicação com outras máquinas e também outras ferramentas. Dentro do Jenkins, foram criadas rotinas de trabalho (*jobs)*, que são tarefas onde pode ser configurado um *script* para iniciar um teste no Jmeter que está em outra máquina (em nosso contexto, outro contêiner). Estes testes foram previamente configurados por um consultor da empresa Yaman, sendo o único processo não automatizado.

Uma vez publicada uma nova versão da aplicação no Jenkins, o teste de desempenho é iniciado e o acompanhamento das métricas do teste podem ser acompanhadas no mesmo momento com um atraso de poucos segundos no painel de instrumentos do Grafana, o teste pode ser parado de forma automática caso atinja um limite pre-determinado, como tempo de resposta bem acima da linha de base conhecida ou quantidade excessiva de erros. O teste pode ser encerrado manualmente também.

Os dados dos testes ficam armazenados para efeito histórico. O expurgo dos dados ficou a critério do cliente e pode ser trivialmente implementado e implantado com agendadores de tarefa.

5 RESULTADOS E DISCUSSÃO

Durante o desenvolvimento do projeto foi possível observar resultados benéficos relacionados à automação, aumentando a confiabilidade, disponibilidade e integridade dos testes realizados. Além disso, o desenvolvimento proporcionou indicadores de linha de base para análise e rápida atuação na correção de problemas, que, conforme esperado, foram detectados com antecedência para possibilitar que um dos objetivos do projeto – a diminuição de bugs – pudesse ser concluído.

6 CONSIDERAÇÕES FINAIS

O uso de testes integrados com desenvolvimento e o ciclo de vida da aplicação mostrou-se como um fator na influência positiva na qualidade das aplicações desenvolvidas e na detecção de problemas que, caso estivessem em ambiente produtivo, causariam prejuízos com instabilidades, indisponibilidades e perda de clientes.

A representação de casos de testes através de programas possibilita a criação de cenários mais elaborados. Além disso, a automação dos testes permite à equipe de desenvolvimento certa segurança para realizar alterações em códigos, já que na etapa de testes, sabe-se que poderão ser simulados inúmeras vezes, assegurando a qualidade do *software*. Esse conjunto permite a percepção de *bugs* mais cedo e apoia os profissionais em uma rápida atuação sobre os mesmos.

REFERÊNCIAS

BERNARDO, Paulo Cheque; KON, Fabio. **A Importância dos Testes Automatizados.** 2008. 7 f. Dissertação - Curso de Engenharia, Desconhecido, São Paulo, 2008. Disponível em: <https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>. Acesso em: 07 maio 2018.

BARBOSA, Filipe Bernardes; TORRES, Isabelle Vasconcelos. O Teste de Software no Mercado de Trabalho. **Tecnologias em Projeção**, Desconhecido, v. 2, n. 1, p.49-52, jun. 2011.

DIEDRICH, Cristiano. **Coleta de métricas no Docker Swarm.** 2017. Disponível em: <https://www.mundodocker.com.br/coleta-metricas-no-docker-swarm/>. Acesso em: 17 maio 2018.

ELIZA, Renata; LAGARES, Vivian. Adotando Checklists no Teste de Software. Revista Java Magazine Ed 87 – DDD e Java, 2010.

GOMES, Fabio. **Integração contínua: uma introdução ao assunto.** Disponível em: <https://www.devmedia.com.br/integracao-continua-uma-introducao-ao-assunto/28002>. Acesso em: 19 maio 2018.

LAGES, Daniel Scaldaferri. Automação dos Testes: um lobo na pele de cordeiro?. Revista Engenharia de Software Magazine Ed 29 – Automação de testes, 2010.

OLIVEIRA, Dieny Sttefanye de. **Teste de performance com JMeter.** 2016. Disponível em: <https://www.devmedia.com.br/teste-de-performance-com-jmeter/34621>. Acesso em: 19 maio 2018.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico.** Novo Hamburgo: Feevale, 2013. Disponível em: <http://www.feevale.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book Metodologia do Trabalho Cientifico.pdf>. Acesso em: 20 maio 2018.

SILVA, Mariela Wagner da. **Evidenciação da Contabilidade Ambiental: uma análise da transparência das demonstrações contábeis de empresas nacionais do setor siderúrgico.** 2004. 68 f. TCC (Graduação) - Curso de Ciência Contábeis, Universidade Federal de Santa Catarina, Florianópolis, 2004. Disponível em: <http://tcc.bu.ufsc.br/TCC/TCC contabeis/2004/MARIELA WAGNER DA SILVA.PDF>. Acesso em: 20 maio 2018.

SILVEIRA, Iclélia. **UM MODELO PARA CAPACITAÇÃO DOS INSTRUTORES DO SISTEMA CAD PARA VESTUÁRIO E DOS MODELISTAS, COM FOCO NA GESTÃO DO CONHECIMENTO.** 2011. 304 f. Tese (Doutorado) - Curso de Desconhecido, PUC-RIO - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2011. Disponível em: <https://www.maxwell.vrac.puc-rio.br/18698/18698_6.PDF>. Acesso em: 20 maio 2018.

“O conteúdo expresso no trabalho é de inteira responsabilidade do(s) autor(es).”