Teste de Performance: Estudo de Viabilidade na Fábrica de Tecnologias Turing (FTT)

Bruno Silveira Ramos, Tales Víctor Gonçalves de Santana

Bacharelado em Engenharia de Software – Centro Universitário de Anápolis (UniEVANGÉLICA) – Anápolis - GO

bruno.ramos@aluno.unievangelica.edu.br, talesvictorsa@gmail.com

Abstract. *The report aims to show the importance of a performance test in Academic Software Factory and a possible application of it in the Turing Technology Factory (FTT), where the same search for quality software that should display or defects and failures. Bibliographic and exploratory researches were employed, where the objectives was to seek theoretical foundation in magazines, articles, books already published on the subject.*

Keywords*: Software, Performance, Test, Quality.*

Resumo. *O relatório tem como objetivo mostrar a importância de um teste de performance em uma Fábrica de Software Acadêmica e uma possível aplicação do mesmo na FTT (Fábrica de Tecnologias Turing), onde a mesma busca um software de qualidade que deve apresentar o mínimo possível de defeitos e falhas. Foram empregadas pesquisas bibliográficas e exploratórias, onde o objetivo foi buscar fundamentação teórica em revistas, artigos, livros já publicados sobre o assunto.*

Keywords: *Software, Performance, Teste, Qualidade.*

1. Introdução

Atualmente, devido à grande disputa no mercado, cada vez mais as empresas estão buscando a qualidade de *software*, uma vez que o mesmo deve apresentar mínimo possível de defeitos e falhas. Para alcançar essa qualidade, é necessário realizar testes de *software*, onde o objetivo é encontrar o máximo de erros no sistema desde o início do desenvolvimento, para evitar gastos desnecessários ao longo da construção. Para isso, os requisitos funcionais e não funcionais podem ser observados e avaliados.

No que se refere aos requisitos não funcionais, uma das métricas para atingir a alta qualidade é a boa *performance* ou desempenho do *software*. Diante disso, é primordial a execução do teste de *performance*, no qual são avaliadas algumas características como: i) o tempo médio de resposta, ii) a quantidade de usuários simultâneos no sistema e iii) possíveis gargalos, entre outras.

Segundo Pressman (2011),

> O teste de desempenho é usado para descobrir problemas de desempenho que podem resultar da falta de recursos no lado do servidor, largura de banda na

> rede inadequada, recursos de banco de dados inadequados, recursos deficientes do sistema operacional, funcionalidade da WebApp mal projetada e outros problemas de hardware e software que podem causar degradação de desempenho cliente-servidor. Em outras palavras, o sistema desenvolvido pode não ter aderência ao contexto (equipamentos, recursos e tarefas) do usuário.

Sendo assim, o objetivo desse relatório técnico é mostrar a importância da execução do teste de *performance* em uma Fábrica de *Software* Acadêmica e uma possível aplicação do mesmo na FTT (Fábrica de Tecnologias Turing). Também será abordado sobre a ferramenta Apache JMeter, utilizada na prática do teste de *performance*.

Além desta seção introdutória, esse artigo possui outas 7 seções. A Seção 2 aborda sobre qualidade de *software*, teste de *software* e *performance*. A Seção 3 mostra o método de pesquisa. A Seção 4 discute a respeito da abordagem proposta, com o foco do estudo na FTT. A Seção 5 expõe os resultados e discussões acerca do estudo produzido. A seção 6 apresenta as considerações finais. Por fim, a Seção 7 revela as referências utilizadas na produção desse relatório.

2. Referencial Teórico

Essa seção apresenta os principais pontos a serem discutidos no relatório. Primeiramente, é abordado sobre o que é qualidade de *software* e suas principais métricas. Logo após é discorrido acerca do que é teste de *software* e sua importância. Em seguida, é explicado o que é teste de *performance*, suas principais características e sua relevância dentro de uma Fábrica de *Software* Acadêmica.

2.1. Qualidade de *Software*

No contexto de desenvolvimento de um *software*, a qualidade caracteriza quanto um *software* atende aos requisitos especificados de um determinado produto, de modo que atenda às necessidades implícitas e explicitas. Segue alguns princípios básicos como: tentar prevenir defeitos ao invés de concertá-los, ter a certeza que os defeitos encontrados sejam corrigidos o mais rápido possível, auditar o trabalho de acordo com padrões e procedimentos previamente estabelecidos, estabelecer e eliminar as causas, bem como os sintomas dos defeitos.

Desde a descoberta da tecnologia até os dias de hoje, a busca pela qualidade de um *software* é constante, buscando atender a completa necessidade do usuário, caso não atenda estes requisitos o *software* terá uma vida curta e será responsável por varias dores de cabeça.

Um sistema com algum defeito, erro ou falha pode gerar diversos problemas, mesmo para aqueles que acham que nunca utilizaram um na vida. Imagine, em nosso dia-a-dia, se por um problema qualquer, ficarmos impossibilitados do uso de nosso celular ou de nosso cartão do banco. Por vezes este problema pode ser de ordem que não podemos controlar, como a falta de energia em uma agência bancária ou a queda do aparelho celular, impossibilitando seu uso.

Segundo Cecília, Massao e Luiz (2006),

> Poderemos ficar insatisfeitos com o fato, mas ele terá uma solução tangível. Agora imagine um problema em que não podemos ter o domínio sobre ele ou que seja intermitente ou possa até causar um dano irreparável a algumas

pessoas. Alguns exemplos, muitos de nós já vivenciamos e outros já causaram catástrofes para muitas pessoas.

Em um desenvolvimento de um *software* a qualidade não se obtém de forma espontânea, é necessário varias etapas de processos para se obter a qualidade adequada, que atende todas indigências imposta pelo usuário. Para que tal importância seja alcançada em uma fabrica de desenvolvimento de *software*, são realizados diversos tipos de teste, tais como, teste de *performance*, carga, *stress*, entre outros, com objetivos de prevenir erros para que sejam corrigidos, para que seja enviado ao cliente um *software* de qualidade.

Existem alguns modelos de avaliação, como o modelo da norma qualidade ISO 25010, que é organizada em 8 características e 32 características. É destacado a característica "Eficiência no desempenho" onde mostra os atributos sobre o teste de *performance* que se mostra eficiente para o processo de desenvolvimento de um *software* que atende seus *stakeholders*.

Figura 1: Características de qualidade de produtos de *software* presentes na norma ISO 25010 [ISO/IEC 2011].

2.2. Teste de *Software*

O teste de um *software* tem como objetivo prevenir erros, defeitos e falhas, por meio de analises dinâmicas de programas para que seja feita a correção o mais rápido possível antes de ser entregue ao cliente, garantindo assim uma maior qualidade do produto.

Trata-se de uma analise dinâmica porque há necessidade de colocar o *software* em execução para avaliar se determinados dados de entrada geram determinados dados de saída.

Os testes são divididos entre funcionais e não funcionais, teste funcional também conhecido como teste de "caixa-preta", testando definitivamente a funcionalidade do sistema, simulando um cenário de produção e identificando possíveis problemas na interface do mesmo. Os testes não funcionais, diferentemente do anterior, não são relacionados à funcionalidade da aplicação em si, pois simulam os cenários de acessos, performance e estresse, testando os atributos de um componente do sistema efetuando uma verificação mais profunda e analítica do servidor.

Para que se tenha uma boa qualidade de software que atende aos *stakeholders*, é necessário todo um processo dentro de uma fabrica de desenvolvimento de *software*, desde o processo de documentação até o processo de desenvolvimento. Durante cada processo é necessário um teste para que tal qualidade seja mantida, ajudando também no tempo de processo do desenvolvimento, pois com os erros já revelados pelos analistas de teste não teria tempo perdido em tentar achar o erro e corrigi-lo.

A equipe tem a liberdade de usar ferramentas de automação para otimizar seu trabalho, tendo um ganho de tempo muito maior, pois as ferramentas realizam teste pré-gravados de forma automática e rápida.

2.3. Teste de *Performance*

Os testes de *performance* têm como finalidade verificar o desempenho do sistema em condições normais de uso, onde o foco é obter informações relevantes da utilização das principais funções. De acordo com a norma ISO/IEC/IEEE 29119-2 [ISO/IEC/IEEE 2013], um processo de testes típico costuma incluir as seguintes atividades: Planejamento de teste; Design e implementação de testes; Configuração do ambiente de teste; Execução de testes; Comunicação de incidentes; Conclusão do projeto de teste.

O teste de *performance* geralmente é feito para ajudar a identificar gargalos no sistema. (Um gargalo é uma parte do sistema que limita o desempenho ou capacidade de todo o sistema, afetando principalmente os tempos de resposta e o *throughput* [Bondi 2014]. Os gargalos em um sistema podem estar localizados no próprio sistema (como uma *query* mal elaborada ou não-otimizada), na camada de *software* que sustenta esse sistema (sistemas operacionais, servidores de aplicação, *firewalls* mal configurados) ou no *hardware*, incluindo CPU, memória, disco e rede). [SILVA, 2018].

Dentro do teste de performance existem cinco subtipos 1) Teste de carga, 2) Teste de *stress*, 3) Teste de capacidade, 4) Teste de Resistência, 5) Teste de volume, no qual somente dois vão ser retratados, que são, teste de carga e teste de *stress*. O teste de carga é geralmente realizado para identificar o comportamento do sistema sob uma carga especifica esperada, analisando se a aplicação, servidor *web* ou banco de dados em teste, mantém um bom comportamento durante sua carga habitual de trabalho. Já o teste de *stress* é realizado para verificar o comportamento do ambiente e software durante uma carga extrema. Também, para determinar se o sistema em teste irá realizar suficientes operações acima do máximo esperado, assim, avalia-se até quando o *software* pode ser exigido e quais as falhas (se existirem) decorrentes do teste.

De forma geral, o teste de desempenho avalia se o sistema irá responder de forma positiva, com vários usuários acessando o site (não necessariamente simultaneamente), ou com diversos tipos de dados cadastrados no banco de dados do sistema. Para realizar tal teste, existem algumas ferramentas, como o JMeter, que será abordado mais amplamente na seção 4.

3. Método de Pesquisa

Para o estudo sobre teste de *performance* e sua possível aplicação na FTT, foram empregadas pesquisas bibliográficas e exploratórias, onde o objetivo foi buscar fundamentação teórica em revistas, artigos, livros já publicados sobre o assunto.

Segundo Engel e Tolfo (2009),

> A pesquisa bibliográfica é feita a partir do levantamento de referências teóricas já analisadas, e publicadas por meios escritos e eletrônicos, como livros, artigos científicos, páginas de web sites. Qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto. Existem porém pesquisas científicas que se baseiam unicamente na pesquisa bibliográfica, procurando referências teóricas publicadas com o objetivo de recolher informações ou

> conhecimentos prévios sobre o problema a respeito do qual se procura a resposta (FONSECA, 2002, p. 32).

Conforme Gil (2008),

> As pesquisas exploratórias têm como principal finalidade desenvolver, esclarecer e modificar conceitos e ideias, tendo em vista a formulação de problemas mais precisos ou hipóteses pesquisáveis para estudos posteriores. De todos os tipos de pesquisa, estas são as que apresentam menor rigidez no planejamento. Habitualmente envolvem levantamento bibliográfico e documental, entrevistas não padronizadas e estudos de caso. Procedimentos de amostragem e técnicas quantitativas de coleta de dados não são costumeiramente aplicados nestas pesquisas.

4. Abordagem Proposta / Estudo na FTT

4.1. Apache JMeter

De acordo com a organização responsável pela ferramenta (Apache *Software* Foundation, 2019), a ferramenta Apache JMeter é um *software* de código aberto, desenvolvido na linguagem Java, empregada para executar testes de desempenho. Possui funções para apoiar a realização de testes funcionais, teste de conexão com o banco de dados JDBC, teste de Serviços da *Web* SOAP / REST, entre outros.

O JMeter pode ser usado para obter informações sobre a *performance* em recursos estáticos e dinâmicos. Através dele é possível simular diferentes cargas em um servidor ou grupo de servidores, rede ou objeto para analisar o desempenho geral em diferentes cenários.

4.1.1. Plano de Teste (*Test Plan*)

O Plano de Teste (Figura 2) é o principal componente, que é responsável por especificar as configurações gerais de um teste. Nele são acrescentados os componentes convenientes aos testes que serão realizados.

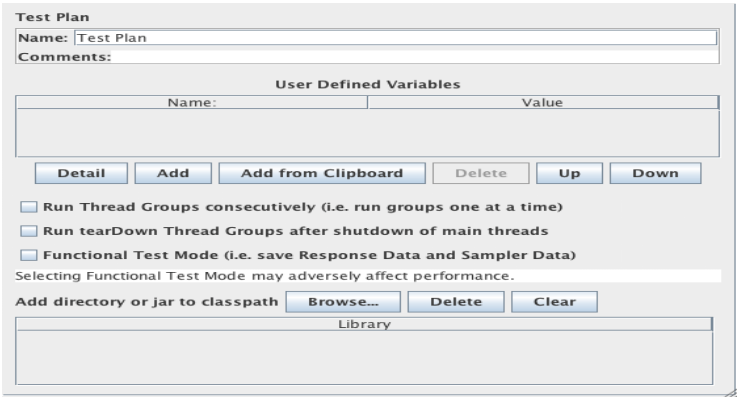

Figura 2: Painel do Plano de Teste [Apache *Software* Foundation, 2019].

4.1.2. Grupo de Usuários (*Thread Group*)

O Grupo de Usuários (Figura 3) é o componente que executa os demais componentes (cenários). Nele é possível simular ações de usuários virtuais, como, definir quantos usuários estão conectados ao mesmo tempo no sistema, o período dessa conexão e qual ação tomar após a ocorrência de um erro.

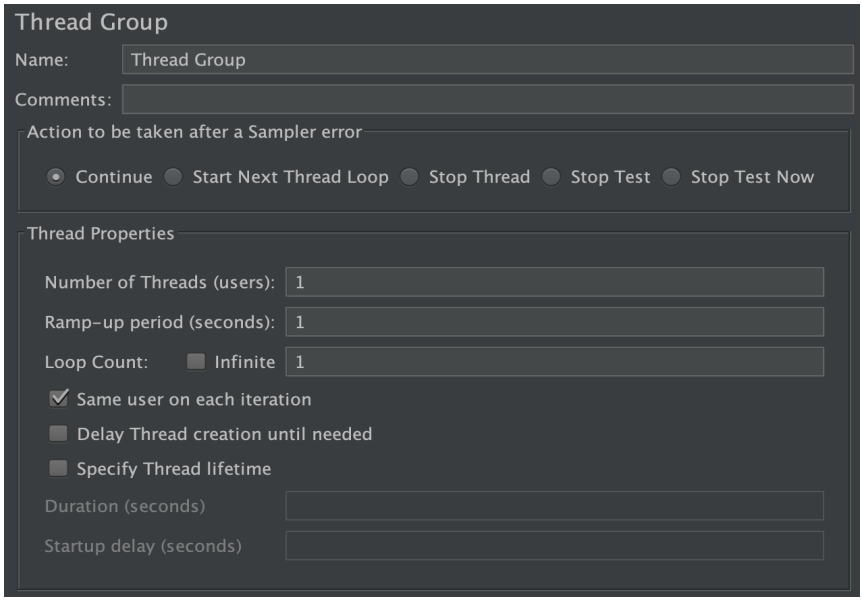

Figura 3: Painel do Grupo de Usuários [Apache Software Foundation, 2019].

4.1.3. Requisição HTTP (*HTTP Request*)

Esse componente (Figura 4) permite gerenciar o envio de requisições HTTP a qualquer página *web*. Esse envio pode ser feito por vários métodos, mas os que serão destacados são os métodos GET e POST. Ambos enviam requisições ao servidor, porém, o GET é um método de leitura ou consulta, já o POST é um método de envio de dados para inclusão ou edição de registros.

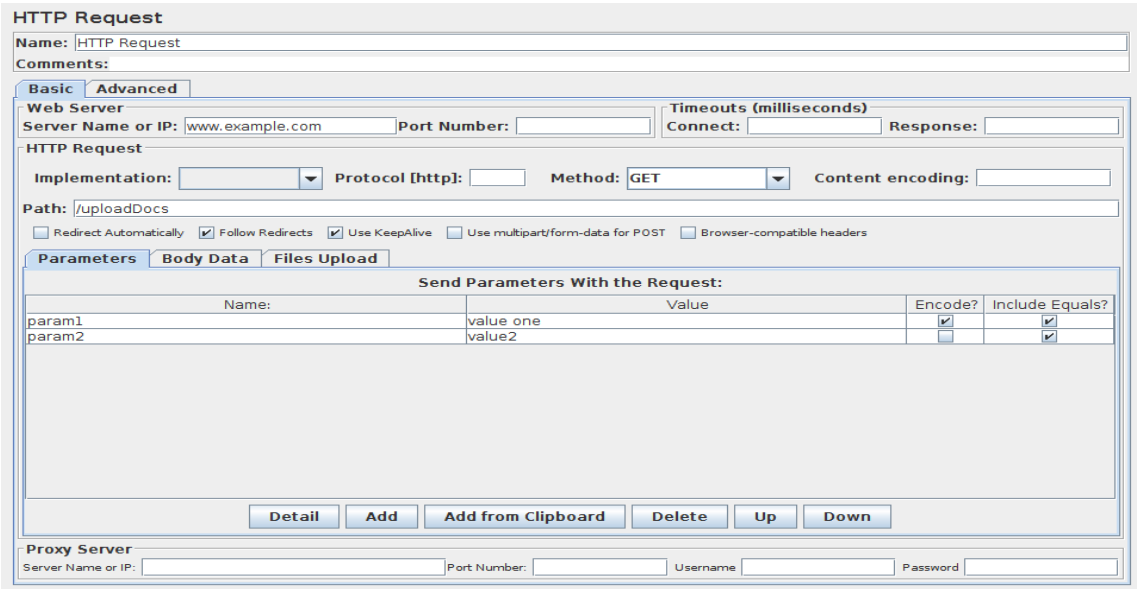

Figura 4: Painel da Requisição HTTP [Apache *Software* Foundation, 2019].

4.1.4. Relatório de Sumário (Summary Report)

O Relatório de Sumário (Figura 5) é o componente que informa os principais dados da realização do teste, dentre os mais importantes são:

- Rótulo que foi adicionado ao Plano de Teste, no caso é a Requisição HTTP (*Label*).
- Quantidade de amostras (usuários), que executaram determinado rótulo (*Samples*).
- Tempo médio de resposta, em milissegundos (*Average*).
- Tempo mínimo de resposta, em milissegundos (*Min*).
- Tempo máximo de resposta, em milissegundos (*Máx*).
- Porcentagem de erros nas amostras executadas (*Error %*).

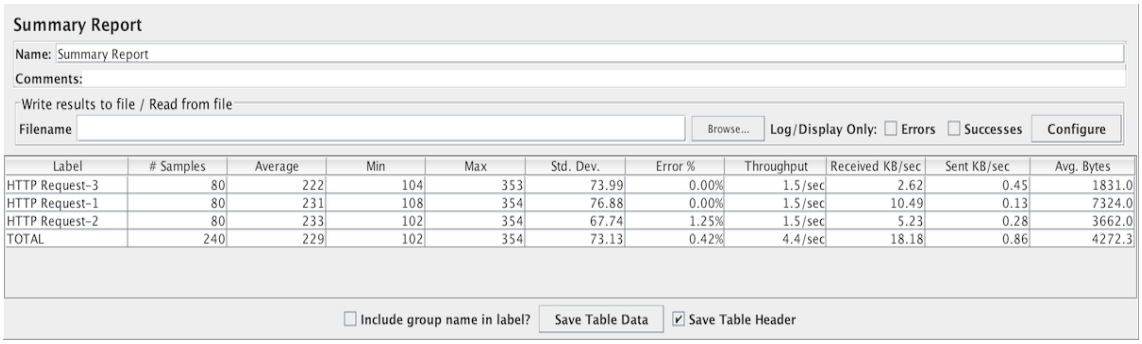

Figura 5: Painel do Relatório de Sumário [Apache Software Foundation, 2019].

4.1.5. Gráfico de Resultados (*Graph Results*)

Esse componente (Figura 6) mostra de maneira gráfica, alguns dados em relação ao tempo de execução. Os dados mais importantes são:

- Média entre o tempo e o número de requisições (*Average*).
- A mediana mostra metade das amostras menores que a média e outra metade maior que a média, podendo ter amostras com valor igual a media (*Median*).
- Desvio de um conjunto de dados (*Deviation*).

- A vazão é o número de amostras por unidade de tempo (*Throughput*).

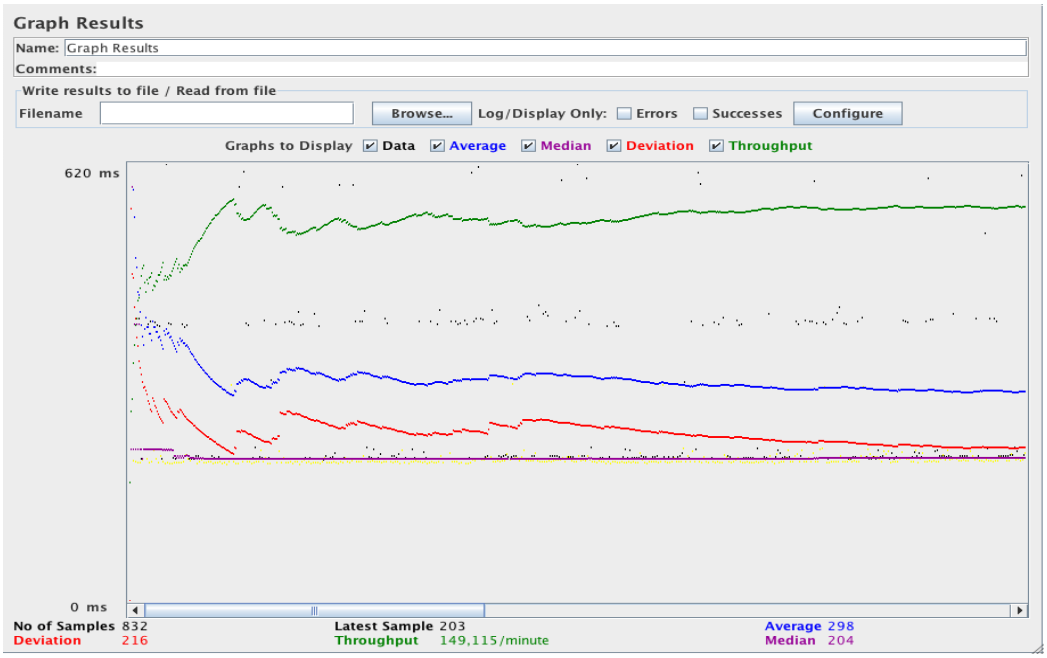

Figura 6: Painel do Gráfico de Resultados [Apache *Software* Foundation, 2019].

5. Resultados e Discussões

Espera-se que com as informações apresentadas, seja possível a implantação do teste de *performance* no processo de teste da Fábrica de Tecnologias Turing (FTT), utilizando a ferramenta Apache JMeter. Nesse sentido, esse relatório procurou mostrar a importância do teste de *performance* e exibir os principais componentes usados no JMeter para a execução desse teste.

6. Considerações Finais

No contexto que foi abordado em todo o trabalho, trazendo a ideia da implementação do teste de *performance* para a Fábrica de Tecnologias Turing (FTT), conclui-se que é viável tal implementação para a Fábrica, onde o mesmo garante uma maior qualidade nos *softwares* desenvolvidos, também garantindo uma otimização no trabalho. Por fim, com esse teste, é possível a utilização de programas para a realização do teste automático de *performance* com a ferramenta JMeter. Por fim, com a otimização, o trabalho flui melhor, e assim, é possível alcançar uma maior qualidade de *software*.

Referências

Apache Software Foundation. *Apache JMeter 5*. 2019. Disponível em: < https://jmeter.apache.org>. Acesso em: 06/12/2019.

CAMPOS, F. M. *Qualidade, Qualidade de Software e Garantia da Qualidade de Software são as mesmas coisas?*. Disponível em: <http://www.linhadecodigo.com.br/artigo/1712/qualidade-qualidade-de-software-e-garantia-da-qualidade-de-software-sao-as-mesmas-coisas.aspx>. Acesso em: 04/12/2019.

DIENY. *Teste de performance com JMeter*. DevMedia. 2016. Disponível em: <https://www.devmedia.com.br/teste-de-performance-com-jmeter/34621>. Acesso em: 28/11/2019.

DUARTE, K. C; FALBO, R. A. *Uma Ontologia de Qualidade de Software*. Mestrado em Informática – UFES. Disponível em: <https://pdfs.semanticscholar.org/f89f/a728eaa29953551badddba11cee4839800df.pdf>. Acesso em: 02/12/2019.

FONSECA, T. et al. *JMeter x WebLoad*. UFS. Disponível em: < http://www.univale.com.br/unisite/mundo-j/artigos/53_Jmeter.pdf>. Acesso em: 12/11/2019.

GERHARDT, E. T; SILVEIRA, T. D. (Org). *Métodos de pesquisa*. Coordenado pela Universidade Aberta do Brasil – UAB/UFRGS e pelo Curso de Graduação Tecnológica – Planejamento e Gestão para o Desenvolvimento Rural da SEAD/UFRGS. – Porto Alegre: Editora da UFRGS, 2009. Disponível em: < http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf>. Acesso em: 14/11/2019.

GIL, A. C. *Métodos e técnicas de pesquisa social*. 6 ed. São Paulo: Atlas, 2008.

ISO 25000. *ISO/IEC 25010*. 2019. Disponível em: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>. Acesso em: 04/12/2019.

KOSCIANSKI, A; SOARES, M. S. *Qualidade de Software*. 2 ed. Editora Novatec. Disponível em: < https://www.martinsfontespaulista.com.br/anexos/produtos/capitulos/241804.pdf>. Acesso em: 20/11/2019.

PAPPE, I. *Manual de Utilização da Ferramenta JMeter*. UFG, Instituto de Informática. Goiânia. 2013.

PRESSMAN, R. S. *Engenharia de Software: Uma Abordagem Profissional*. 7 ed. Porto Alegre: AMGH, 2011.

Qualidade de Software. PUC – Rio. Disponível em: < https://www.maxwell.vrac.puc-rio.br/16818/16818_4.PDF>. Acesso em: 02/12/2019.

RAPCHAN, F. *A Norma ISSO 9000-3*. UFES, Centro Tecnológico, Departamento de Informática. Disponível em: < http://www.geocities.ws/chicorapchan/artigos/9000-3.pdf>. Acesso em: 20/11/2019.

SILVA, K. L. *Qualidade de Software*. Uberlândia. 2007. Disponível em: < http://www.facom.ufu.br/~bacala/QS/apostila_QS_Katia.pdf>. Acesso em: 03/12/2019.

SOMMERVILLE, I. *Engenharia de Software*. 9 ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, T. S. *Testes de Desempenho de Software: Teoria e Prática*. Escola Regional de Sistemas de Informação do Rio de Janeiro, 2018. Disponível em: < https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/7/13/39-1?inline=1>. Acesso em: 13/11/2019.

ZABEU, A. C. P. et al. *A importância da qualidade no desenvolvimento de software*. 2006. Disponível em: <http://asrconsultoria.com.br/wp-content/uploads/2016/06/A-importancia-da-qualidade-no-desenvolvimento-de-software.pdf>. Acesso em: 04/12/2019.