CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

Nayara Marcela Chaves

UMA ABORDAGEM DE APLICAÇÃO DE TESTE DE PERFORMANCE NA FÁBRICA DE TECNOLOGIAS TURING – UNIEVANGÉLICA

ANÁPOLIS
2019-02

Nayara Marcela Chaves

UMA ABORDAGEM DE APLICAÇÃO DE TESTE DE PERFORMANCE NA FÁBRICA DE TECNOLOGIAS TURING - UNIEVANGELICA

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a conclusão da disciplina de Trabalho de Conclusão de Curso II do curso de Bacharelado em Engenharia de Computação do Centro Universitário de Anápolis – UniEVANGÉLICA.

Orientador(a): Prof. Mestre Walquíria Fernandes Marins

Anápolis
2019-02

NAYARA MARCELA CHAVES

UMA ABORDAGEM DE APLICAÇÃO DE TESTE DE PERFORMANCE NA DE TECNOLOGIAS TURING - UNIEVANGÉLICA

Trabalho de Conclusão de Curso II apresentado como requisito parcial para a c
grau do curso de Bacharelado em Engenharia de Computação do Centro Univ
Anápolis – UniEVANGÉLICA.

Aprovado(a) pela banca examinadora em 28 de novembro de 2019, compos

Walquíria Fernandes Marins
Walquíria Fernandes Marins
Presidente da Banca

Kleber Silvestre Diogo
Kleber Silvestre Diogo

Agradecimentos

Foi uma longa caminhada até aqui, pensei varias vezes que o cansaço iria me ganhar, mas agora vejo que foi gratificante todo esse tempo.

Eu não teria conseguido sem a graça de Deus que me iluminou em toda caminhada, sou muito grata aos meus pais Edmilson Chaves e Coraci Marcela que me deram todo apoio necessário para iniciar e ir atrás dos meus sonhos e também a minha companheira da vida Anália de Lucca Furlan que me encoraja em todos meus momentos de fraqueza apenas me mostrando que sou capaz de conquistar meus sonhos.

Quero agradecer a minha orientadora Walquíria Marins pela sua competência e paciência nesta jornada, e também a todos os docentes da UniEVANGELICA que contribuíram para este trabalho.

RESUMO

Este trabalho de pesquisa tem como tema central a abordagem sobre a importância dos testes de desempenho em software visando obter uma melhor qualidade no produto final para os clientes. Sabe-se a criação de um software é um trabalho árduo e que requer a máxima atenção profissional de seus desenvolvedores. É preciso ressaltar que são vários os problemas que podem ocorrer em um projeto, seja ele de natureza técnica ou humana. Por isso, erros devem ser localizados a fim de evitar a falta de qualidade no produto final. Sendo assim, o objetivo desta pesquisa é contribuir para a qualidade final do produto, e desta forma minimizar problemas futuros. Foi realizada uma pesquisa utilizando o projeto VIRTOO - Sistema de Gerenciamento Acadêmico desenvolvido na Fábrica de Tecnologias Turing (FTT), localizado na UniEVANGELICA Centro Universitário de Anápolis, com o aplicativo JMeter para testar o desempenho dos requisitos disponíveis.

Palavras Chave: *Software*. Teste de Performance. Qualidade.

ABSTRACT

This research study's central theme is the approach to the importance of software performance testing in order to achieve a better quality of the final product for customers. The development of a software is known to be an arduous work that requires the maximum professional attention of its developers. It is necessary to emphasize that there are several problems that might occur in a project, be it of technical or human origin. Therefore, errors must be found in order to avoid a lack of quality in the final product. Therefore, the aim of this research is to contribute to the final quality of the product, and consequently minimize future problems. A research was carried out using the VIRTOO Project - Academic Management System developed at the Turing Technology Factory (Fábrica de Tecnologia Turing - FTT), located at UniEVANGELICA University Center of Anapolis, with the application JMeter to test the performance of available requirements.

Keywords: *Software*. Performance test. Quality.

LISTA DE ABREVIATURAS

CAPES	-	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
FTT	-	Fábrica de Tecnologia Turing
IEEE	-	*Institute of Electrical and Electronic Engineers*
ISO	-	International Organization for Standardization
ISPVida	-	Instituto Superior Politécnico Vida
ISTEL	-	Instituto Superior de Teologia Evangélica no Lubango
P&D	-	Pesquisa e Desenvolvimento
Scielo	-	*Scientific Electronic Library Online*
VIRTOO	-	Sistema de Gerenciamento Acadêmico para as faculdades
VV&T	-	Verificação, Validação e Teste

LISTA DE ILUSTRAÇÕES

Sumário

1. INTRODUÇÃO

Conforme apresentou Barbosa et al. (2016), as fábricas de *software* costumeiramente apresentam a abordagem de desenvolvimento dinâmico, iterativo e ágil. Possuem atualmente um papel relevante no desenvolvimento social e empresarial, sendo necessário frisar que seus produtos acima de tudo devem oferecer a qualidade que os clientes esperam.

A preocupação é quanto a possíveis prejuízos que as empresas desenvolvedoras de *software* possam sofrer quando não são capazes de realizar os testes de qualidade, seja por conta dos altos custos ou mesmo causado pelo problema da manutenção de uma equipe de trabalho especializada e, assim, deixam de avaliar o desempenho de seus produtos. Assim, o que se questiona nessa pesquisa é: Como contribuir para a qualidade do projeto VIRTOO, com a realização dos testes de carga, desempenho e estresse na Fábrica de Tecnologias Turing (FTT)?

Essa pesquisa tem como objetivo geral avaliar qual o processo de teste de *performance* adequado às fábricas de *software* ágeis. Para tanto, os específicos são: i) estudar as principais técnicas e ferramentas para o teste de *performance*; ii) identificar as características e necessidades das fábricas de *software* ágeis em relação aos testes de *performance* de carga, estresse e desempenho; iii) propor um processo de teste de performance de carga, estresse e desempenho e v) analisar os resultados da aplicação do processo proposto.

De acordo com Sommerville (2011) na sociedade atual, onde os processos cotidianos cada dia mais estão automatizados, o *software* é a peça chave para que tais relações, sejam elas sociais ou comerciais, aconteçam com o máximo de qualidade que se espera. E dentro dessa sistemática, o desenvolvimento de *software* deve ser ágil para atender o máximo de necessidades do seu fabricante.

Cobra (2007) ressalta que os produtos são considerados como prontos a atenderem as expectativas da clientela quando os mesmos possuem a qualidade que deles se espera. Então, para que um *software* possa ser considerado pronto para ser utilizado é imprescindível que o mesmo tenha sido testado de maneira exaustiva e seus problemas e erros de configuração detectados, para isso é imprescindível a escolha do tipo de teste que melhor se adeque às especificações de cada produto visando ao final conceder a necessária qualidade.

Sabe-se que a criação de um *software* é um trabalho árduo e que requer a máxima atenção e profissionalismo de seus desenvolvedores. Independentemente do tamanho e funcionalidades do produto o cuidado em testar sucessivamente o *software* é muito importante, bem como, a escolha pelo tipo de teste mais adequado. É preciso ressaltar que vários são os problemas que podem ocorrer a um projeto, seja ele técnico ou humano o erro deve ser localizado e desta forma evitar a falta de qualidade do produto final com o modelo de teste correto para cada circunstância (DELAMARO; MALDONADO, JINO, 2016). Entretanto, o executar teste funcional não é suficiente para assegurar o aumento da qualidade, são necessários também os testes acerca de requisitos não funcionais, muitas vezes chamados de requisitos de qualidade.

Para tanto essa pesquisa se justifica baseando-se na afirmativa de Coser (2012) ao dizer que ao se encontrar com os problemas de *performance* do *software* é preciso saber quais elementos influenciam positivamente e negativamente no seu desempenho e utilizar o teste certo dentro da grande variedade a disposição dos técnicos e responsáveis. Em Ruffato (2010) observa-se que no processo de testes de produtos de *software* são basicamente construídos dentro de quatro etapas: o planejamento, projetos dos casos de testes, a execução e por fim a avaliação dos resultados dos testes.

No teste de *performance* de especificação (teste caixa-preta) visa-se encontrar se o produto satisfaz as especificações primárias para as quais o *software* foi concebido. Esse tipo de teste pode em certos casos ser utilizado em qualquer uma das etapas e contextos, por sua característica de informalidade e sem que haja a necessidade de modificações. Já os testes especificados como de programa (teste caixa-branca) necessitam de inspeção do código fonte. Neste segundo caso, o teste é utilizado em casos específicos em que se deve exercitar partes do código apenas (KOSCIANSKI; SOARES, 2007).

Nesta pesquisa foi dada prioridade a três tipos de testes. O primeiro, de carga, que na visão de Santos et al. (2010) têm por intuito ratificar o comportamento geral do *software*, verificando os dados e funcionalidades do componente, desde a entrada até a saída das informações. O segundo, teste de desempenho tem como finalidade primordial afirmar que ao final da construção do *software* esse produto atende rigorosamente aos seus requisitos e ambiente em que será utilizado. Por fim, o teste de estresse que em alguns momentos é confundido com o de desempenho por trabalhar conjuntamente com aquele, mas se diferencia por ser aplicado em situações de máxima complexidade em ambientes críticos tendo por princípio verificar sua tolerância.

Enquanto que Vilas Boas (2007) ressalta o caráter complementar do processo de testes, como também, ensina que para cada caso deve ser utilizado a melhor técnica de teste de *performance* que se adeque ao *software* avaliado, visando assim obter maior vantagem e qualidade no produto final, com baixo custo. É através da sistemática dos testes que as fábricas de *software* conseguem atingir a eficácia uma abordagem metodologicamente correta conforme a fundamentação teórica, sendo desta forma, consideradas como ferramentas de auxílio e avaliação com alto grau de necessidade em busca da qualidade que o cliente espera de seu futuro produto.

Outro motivo para a elaboração desta pesquisa se baseia na busca acadêmica em que os experimentos se tornem uma constante, assim ao abordar tal tema tem-se como incentivo poder, mesmo que de maneira ínfima, colaborar para que os colegas de curso também busquem evoluir nas pesquisas visando o aprimoramento dos testes e, por conseguinte desonerar essa etapa tão importante na fabricação de novos *software*s.

Esta pesquisa tem como metodologia se tratar de um estudo de caso, onde de início foram realizadas a coleta de informações através da revisão de literatura. Posteriormente deu-se a aplicação de uma pesquisa experimental qualitativa com os integrantes da FTT.

Este trabalho se dividiu em capítulos. O primeiro serviu para o levantamento bibliográfico referente ao assunto abordado. O segundo apresenta a metodologia utilizada na construção desta pesquisa. No terceiro a abordagem de estudo para atingir os objetivos propostos. O Terceiro capítulo traz os resultados alcançados pela pesquisa direta.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 INDÚSTRIA TECNOLÓGICA DE *SOFTWARE*

De acordo com Santos (2014) todo território nacional quando das primeiras implementações na área da informática veio ao longo dos anos passando por um processo intenso de informatização, sobretudo, o crescimento de redes informacionais e aparelhos como: computadores, *smartphones*, *tablets* e uma centena de outros itens usuais no dia a dia da sociedade. Sendo que em todos os casos existe uma dependência clara em relação aos *software*s para a execução de suas funções, por meio da manipulação da informação neles contida.

Ainda Santos (2014, p. 15), "tudo se informatiza, mas no território esse fenômeno é ainda mais marcante na medida em que o trato do território supõe o uso da informação", assim, compreende-se que a medida que a evolução da informatização adentra o espaço territorial do país de forma qualificada, mais existe a necessidade do suporte técnico das atividades sejam elas empresariais, industriais, prestação de serviços e do cotidiano social.

> A partir dos processos de informatização e modernização, é possível identificar porções do território brasileiro que se destacam pela alta densidade informacional. Em 1982, durante o início da difusão dos microcomputadores, o grau de concentração regional era da ordem de 75% dos parques de computadores instalados nos estados do Sudeste, sendo 46% em São Paulo, 22% no Rio de Janeiro e 5% em Minas Gerais, seguidos da região Sul, com 13% dos equipamentos. A região Nordeste contava com 6% do total de computadores instalados, e o Norte com apenas 1%. Atualmente, identificamos, por exemplo, aglomerações de empresas produtoras de *software* (STEDA, 2015, p. 28).

Para Tenório e Valle (2012, p. 60) e diante das grandes revoluções tecnológicas ocorridas na sociedade mundial para se construir um *software* é preciso percorrer um caminho em sua produção com fases definidas como: "levantamento; projeto lógico; modelagem básica de dados; definição da arquitetura; especificação; programação; e testes, para então ser reproduzido". Deve-se ressaltar que o *software* como deverá ser concebido somente pode ser entendido após a fase de sua programação, anteriormente processada por um período em que se estudam modelos, a sua concepção, descrição de algoritmos entre outros elementos.

Depois de produzido o *software* passa para a fase de distribuição do programa que conforme ensina Silva (2009), ela pode ser realizada de maneira condicionada à compra com direito de exclusividade (que neste caso não permite a redistribuição a terceiros);

shareware que significa que o registro foi devidamente adquirido, mas seu uso é considerado aberto a mais avaliações; demo em que o uso é livre por tempo previamente determinado; *adware* onde o seu uso é atrelado condicionalmente à propagandas; *freeware* em que o uso é gratuito e onde o código fonte não será alterado; domínio público em que os direitos autorais são protegidos por leis adequadas ao tema; semi-livre em que as cópias, modificações e distribuição são permitidas somente para fins não lucrativos.

Steda (2015) levantou uma classificação tendo como base quatro atividades consideradas como cruciais, da seguinte forma: em empresas especializadas em serviços de informática, como consultoria, manutenção, reparação e comercialização de equipamentos; firmas que devem se responsabilizar pela distribuição e comercialização de *software*-produto, pronto para uso; o grupo dos serviços em *software* de baixo valor agregado corresponde àqueles relacionados à internet ou de criação, processamento e manutenção de dados e os serviços em *software* de alto valor agregado envolve o desenvolvimento de *software* sob encomenda (ou customizável) e de projetos e modelagens de bases de dados.

As empresas que tem como ramo de atuação a engenharia de *Software* evoluíram muito nas últimas décadas visando criar novas técnicas, critérios, métodos e ferramentas na produção de seus produtos de informática, isso se deve à grande procura e utilização que o mercado consumidor tem necessitado para fazer frente às mudanças sociais, nos modos de comunicação e entretenimento. Essas mudanças atingem a todas as áreas da atividade humana e que por decorrência se baseiam na computação (JORGE et al., 2015).

Britto e Stallivieri (2010, p. 320) ensinam ainda que além de todas as características positivas aqui já apresentadas, também é preciso ressaltar que a produção de *software* gera novos postos de emprego com mão de obra qualificada, sendo que em algumas situações em atividades intensivas em P&D.

As fábricas de *software* adotam dois modelos distintos de produção, o tradicional e o ágil e conforme os dizeres de Lutz (2017, p. 13) se diferenciam da seguinte forma:

> O modelo tradicional é conceitualmente mais antigo e caracteriza-se por ter um processo sequencial, que contempla um esforço maior de análise no início do projeto, seguido por uma fase de desenvolvimento pesada e após, um período de testes e homologação, sendo que o cliente recebe uma entrega de *software* concentrada ao final dos trabalhos.
>
> O grupo das metodologias ágeis surgiu justamente para oferecer dinamicidade aos projetos, propondo novas técnicas de execução das etapas de trabalho, seguindo fluxos alternativos e flexíveis. Basicamente esse modelo de trabalho propõem que as fases sejam executadas conforme a sua necessidade, ampliando a capacidade de adaptabilidade do processo às características do projeto, ou seja, a análise, o desenvolvimento e os testes podem andar em paralelo, enquanto o cliente recebe pequenas entregas parciais. Esse modelo exige maior participação

> do cliente no projeto, garantindo um nível de assertividade maior tanto para a empresa quanto para o cliente.

2.2 QUALIDADE DE *SOFTWARE*

De acordo com Batista (2014) o conceito referente a qualidade se trata de um aglomerado de elementos que tornam um produto eficaz para determinada função a qual tenha sido criado. Entre essas funções estão pontos como atendimento das necessidades do cliente, ser coerente e seguro de acordo com os padrões técnicos exigidos pela legislação, outras qualidades podem ser agregadas aos produtos tais como: o aspecto moral; sua qualidade deve poder ser intrínseca; a questão da entrega dentro do prazo acordado, custo acessível e segurança que ofereça.

Especificamente a um *software* Pressman (2016) entende que a qualidade parte do princípio de seguir a risca os requisitos de funcionamento e seu desempenho, obedecer aos padrões legais relativos ao termo de respeito a qualidade que se espera ao adquirir um produto.

Todas essas determinações retro mencionadas fazem parte do que está exposto na ISO 9000:2005 (ISO 2005), documento esse que cuida de legislar sobre a qualidade como um princípio na construção e desenvolvimento de produtos. Por isso, se torna tão necessário realizar a bateria de testes que somente a partir de sua exaustão se pode ter a certeza de que o *software* obedece aos requisitos e características ao qual tenha sido desenvolvido (CÉRGOLI, 2017).

2.2.1 Norma ISO/IEC 25000 - SQuaRE

De acordo com Yang (2012) existem vários modelos de avaliação da qualidade de *softwares*, sua utilização depende de aspectos característicos a cada um dos *softwares* dentro da cadeia hierárquica. Nesse sentido varias normas internacionais foram criadas no sentido de regular a criação e busca pela qualidade esperada, entre elas se encontra a ISO/IEC 25000 (*Software Quality Requirements and Evaluation, SQuaRE*), que é um modelo dividido em cinco partes e foi desenvolvida como fornecedora do suporte que definem os requisitos e qual o processo a ser escolhido nos testes de *software*.

Conforme Moraes e Lima Júnior (2017, p. 243) esse modelo descrito na ISO/IEC 25000 se relaciona a duas dimensões relativas à qualidade de *software*:

> A primeira delas é a qualidade em uso, que especifica características de qualidade relacionadas à interação humana com o sistema. [...] A segunda dimensão proposta pela ISO/IEC 25010 é a qualidade do produto, que define um conjunto de oito características da qualidade relacionadas à atributos internos e externos do *software*.

E quanto a essas oito características o Quadro 1 traz cada uma delas e suas discriminações:

Quadro 1 – Características da ISO/IEC 25000

Característica	Descrição
Adequação funcional	Capacidade do software de prover funções que atendam às necessidades implícitas e explícitas quando usado em condições especificadas.
Eficiência de desempenho	Desempenho em relação à quantidade de recursos utilizados.
Confiança	Capacidade de manter um nível de desempenho especificado em um determinado período de tempo.
Usabilidade	Capacidade de ser usado para atingir metas específicas com efetividade, produtividade e satisfação do usuário.
Manutenibilidade	Capacidade de ser modificado visando à melhoria, correção ou adaptação a mudanças no ambiente ou nos requisitos.
Portabilidade	Capacidade de ser transferido de um hardware, sistema operacional ou ambiente de uso para outro.
Compatibilidade	Capacidade de trocar informações com outros *softwares* e desempenhar as funções que lhe forem requeridas enquanto compartilha recursos de *hardware* e *software*.
Segurança	Capacidade do software de proteger informações para que pessoas ou outros softwares tenham o nível de acesso apropriado aos seus níveis de permissão.

Fonte: Adaptado de Moraes e Lima Júnior (2017, p. 243-244)

Pode-se observar pelo descrito no quadro acima e diante das informações levantadas a ISO/IEC 25000 se trata de uma norma regulamentadora que tem um único princípio. Através de um processo sério de construção e desenvolvimento as empresas sejam obrigadas a conceder ao cliente a tão almejada qualidade dos produtos a serem adquiridos, somente a partir dessa sistemática os *software* poderão levar aos seus usuários a confiança de que aquilo a que se propõe o produto será atingida.

2.3 TESTES DE *PERFORMANCE* EM *SOFTWARE*

De acordo com Barbosa et al. (2016) deve-se ressaltar que mesmo com todos os cuidados necessários à qualidade na produção de *software*s, em todas suas etapas e

atividades, erros no produto devem ser esperados. Para diminuir seus efeitos sobre a qualidade do *software* as indústrias deste produto têm efetivado um processo que percorre toda a sistemática produtiva de desenvolvimento.

Na busca por evolução as fábricas de *software* vêm ao longo dos anos aprimorando suas técnicas, critérios, avaliando seus métodos e ferramentas, tudo isso observando como as necessidades humanas se alteram, trazendo dessa forma ao mundo da computação uma alta demanda por qualidade e produtividade, seja no que concerne às impressões dos clientes, quanto ao que o produto pode oferecer (JORGE et al., 2015).

Conforme Barbosa et al. (2016) neste caso a realização de testes constantes relativos à qualidade e operacionalidade do *software* produzido passa a ser primordial, antes que o cliente possa adquiri-lo para seu uso pessoal. Ázara (2013) ressalta que a realização dos testes de *performance* são atividades importantes no processo de produção de *software*s, pois é a partir deles que os erros podem ser eliminados e o nível de qualidade pode aumentar.

É através do conjunto de informações coletadas nos testes de *performance* que as atividades de depuração, manutenção e estimativa relativos ao nível que se pode obter do *software* podem ser dimensionadas. Ázara (2013) relembra que para muitos estudiosos, o processo de realização dos testes é considerado como sendo a parte mais onerosa em todo conjunto de produção de um novo *software*, mas também alerta que tal fato ainda não pode ser definitivamente tachado desta forma por haverem ainda pontos a serem melhor estudados e maximizados os custos.

Enquanto que Delamaro et al. (2016) ressaltam o caráter complementar do processo de testes, como também, ensina que para cada caso deve ser utilizado a melhor técnica de teste de *performance* que se adeque ao *software* avaliado, visando assim obter maior vantagem e qualidade no produto final, com baixo custo. É através da sistemática dos testes que as fábricas de *software* conseguem atingir a eficácia uma abordagem metodologicamente correta conforme a fundamentação teórica, sendo desta forma, consideradas como ferramentas de auxílio e avaliação com alto grau de necessidade em busca da qualidade que o cliente espera de seu futuro produto.

2.3.1 Teste de Carga

De acordo com Campos (2011) o teste de carga é um tipo de teste no qual quando realizado se tem por finalidade verificar o volume de dados que é suportado por certo sistema, ou seja, sua capacidade de processamento. Esse modelo de teste tem como característica a possibilidade de monitorar um sistema em um ambiente onde existe uma grande carga de acessos ao mesmo tempo.

O teste de carga pode ser realizado de três formas, conforme se descrimina no Quadro 2:

Quadro 2 – Tipos de testes de carga e suas características

Tipo	Característica
Constante	Quando se realiza um tipo de teste invariável de carga por um período de tempo determinado.
Por etapa	Quando de forma controlada se vai aumentando a carga de acessos e ao mesmo tempo se monitora o desempenho do sistema.
Meta	Quando se traça uma meta máxima que o sistema deve suportar, assim se pode ter a clara noção do número máximo de usuários que podem ter acesso ao mesmo tempo.

Fonte: Adaptado de Campos (2011)

Souza (2018) define o teste de carga como sendo aquele que forma maneira geral são utilizados para se verificar qual a carga máxima e dessa forma avaliar o seu desempenho dentro de situações mínimas ou máximas de uso compartilhado de um dado sistema.

2.3.2 Teste de Estresse

Modelo de teste de *software* no qual ele é aplicado visando colocar o sistema a situações extremas. Ou seja, ao se colocar um sistema em funcionamento com uma carga máxima suportável se verifica quais poderiam ser os problemas apresentados e dessa forma buscar analisar e realizar as correções necessárias (MENEZES, 2018).

Esse tipo de teste é uma ferramenta fundamental em sistemas que requeiram uma grande eficiência sob um ambiente de constantes e múltiplos acessos, tais como: servidores de arquivos e da web; em aplicação dentro de grandes corporações empresariais e jogos de computador online (MENEZES, 2018). Sparano (2012) aponta que a maior dificuldade na aplicação do teste de estresse é no cuidado necessário que deve haver na configuração adequada da plataforma a ser testada.

2.3.3 Teste de Desempenho

Para Gomes (2015) o teste destinado a verificar o desempenho de *software* visa conhecer a melhor aplicação e finalidade do teste de carga. Esse tipo de teste analisa o fluxo de execução, o tempo de resposta e a qual o limite máximo suportado.

> Sabendo que os objetivos de desempenho precisam expressar as medidas capazes para determinar o quão rápido o sistema deve ser executado, qual o limite aceitável para seu desempenho, qual o consumo de processador e memória aceitável e que número de usuários concorrentes deve ser suportado. Os objetivos e foco de teste de desempenho devem estar claramente estabelecidos nas metas do projeto do sistema, pois somente assim poderão nortear os esforços do desenvolvimento para se saber quando eles foram atingidos. (GUARIENTI et al., 2014, p. 05).

Segundo Campos (2010) quando se utiliza o modelo de teste que objetiva verificar o desempenho do *software* a princípio tem-se por iniciativa conhecer os níveis confiabilidade, tempo de resposta, *throughput*, ou a escalabilidade de um sistema que esteja sob análise dentro de um ambiente de carga alta de trabalho.

De acordo com Campos (2010) o teste de desempenho é uma ferramenta importantíssima no conjunto do desenvolvimento do *software*, trazendo elementos positivos na qualidade final do produto. Para realizá-los, devem ser utilizadas as ferramentas certas para o monitoramento e análise dos dados conseguidos pela pesquisa.

2.4 A IMPORTÂNCIA DO TESTE DE *PERFORMANCE* EM *SOFTWARE*

Para Delamaro et al. (2016) construir um *software* não é uma tarefa fácil, principalmente se as dimensões e utilidades forem de grandes proporções. E as possibilidades de que o produto não possua a qualidade que se espera torna-se ainda maior, por erro humano, que mesmo podendo dispor de todas as ferramentas da engenharia de *software* são passíveis de acontecer.

Ressalta Coser (2012) que para conseguir diminuir a possibilidade acima descrita é preciso que o desenvolvedor tenha conhecimento para lidar com este desafio. Lidar com os problemas de *performance* do *software* precisa necessariamente saber quais elementos influenciam positivamente e negativamente no desempenho.

Conforme Freitas (2013), a performance do *software* é uma das qualidades do produto que pode ser afetada ainda em seus primeiros passos de construção, o sistema operacional. *middleware*, *hardware* e redes de comunicação. E a partir da medição do tempo de resposta que se tem uma noção da qualidade de *performance*, mesmo quando exposto a um volume de processamento considerado como exaustivo.

Jorge et al. (2015) afirmam que a responsabilidade pelo desenvolvimento dos testes de qualidade dos *softwares* é dos seus desenvolvedores. E esses testes são compostos por três atividades básicas: planejamento na qual é definida a estratégia que tem por princípio minimizar os riscos do negócio; o projeto é a segunda atividade nela se deve estudar qual os tipos de testes a serem utilizados de acordo com a constituição do produto que está sendo fabricado e por último a execução onde os testes escolhidos na atividade anterior são colocados em funcionamento e como os mesmos atuam no produto e suas respostas passam a ser conhecidas.

Coser (2012), traz que a *performance* do produto é tratada de maneira comum como sendo a forma como se dá a eficiência, a efetividade e os aspectos gerenciais deste *software*. A *performance* pode ser medida por duas categorias: i) subjetiva que depende apenas do potencial de conhecimento de seus desenvolvedores e; ii) objetiva que coloca em questão elementos quantificáveis como o custo, esforço e atraso.

Algumas fábricas relutam em utilizar os testes de *performance* no desenvolvimento de seus produtos. A maioria delas se baseia em motivos como: alta complexidade; no tempo gasto na elaboração das atividades de design, execução, análise dos resultados; A estratégia usada na simulação de ambientes extremos e o mais próximo da realidade que o sistema irá enfrentar em uso (CAMPOS, 2010).

As fábricas que no tardar sofrem com problemas de *performance* no software que já está em produção pois, precisa de uma tratativa mais complexa, devido os requisitos já estarem em produção o custo, esforço e tempo são maiores.

Por fim, há que se acrescentar que toda a revisão de literatura apresentada teve como objetivo salientar a importância dos testes de *performance* em *software*, indo de encontro ao que se propõe neste trabalho.

3. MÉTODO DE PESQUISA

Para atingir os objetivos propostos de forma a encontrar respostas para o problema de pesquisa identificado, o método de pesquisa deste trabalho foi organizado em etapas, a saber: a) Revisar bibliografia e analisar diligência; b) Examinar o processo de teste atual do local escolhido para experimentação; c) Aplicar *workshop* para aplicação de teste de *performance* para a equipe de testes; d) Validação da utilização do processo proposto; e) Retratar aprendizado após análise de resultados.

A primeira etapa consistiu em uma revisão bibliográfica para conhecer estudos relacionados, bem como, selecionar um modelo adequado de teste de *performance* de carga, desempenho e estresse para fábricas de *software*. Em seguida, foi feita uma análise da revisão bibliográfica sobre o tema, por meio do método dedutivo e comparativo (PRODANOV; FREITAS, 2013). O método dedutivo é a modalidade de raciocínio lógico visando explicar as premissas contidas no conteúdo e também se trata de uma pesquisa comparativa, pois o mesmo visa estudar um mesmo fenômeno em tempos diferentes concedendo dessa forma ao pesquisador a possibilidade de verificar a evolução de seu experimento.

As pesquisas foram realizadas em livros, artigos, revistas em meio eletrônico e analógico, para a fundamentação teórica desta pesquisa. Dando-se prioridade a revistas eletrônicas indexadas de reconhecimento científico tais como: Scielo, Institute of Electrical and Electronic Engineers (IEEE), Science Direct, Portal de Periódicos CAPES, scielo, google acadêmico e Literatura Científica e Técnica da América Latina e Caribe (Lilacs) entre outras fontes, sobre o melhor modelo de teste de *performance* de carga, desempenho e estresse adequados às fábricas de software.

Foi realizada um acompanhamento presencial com a equipe de teste com a finalidade de analisar o processo de teste utilizado, entender quais as dificuldades eles enfrentam e como a aplicação do teste de *performance* precisa ser executada dentro do processo.

Em uma das etapas um foi realizado um *worskhop* na Fábrica de Tecnologias Turing com os integrantes da equipe de teste, expondo as funcionalidades principais da ferramenta JMeter e como o processo de teste se enquadra.

Após a parametrização, o processo proposto foi analisado com o líder da equipe de teste reforçando a importância de se testar a performance do projeto VIRTOO para contribuir com a qualidade final.

Com todas as parametrizações realizadas, ambiente configurado para execução do teste de *performance* dois planos de teste foram criados para contribuir com os resultados deste projeto.

4. ABORDAGEM PROPOSTA

Para atingir aos objetivos propostos por esta pesquisa foi selecionado um ambiente de fábrica de *software* acadêmica que será descrito na subseção 4.1, em seguida, serão descritos os aspectos da aplicação da intervenção, bem como da ferramenta adotada para coleta dos dados.

4.1 FABRICA DE TECNOLOGIAS TURING (FTT)

Fundada em 2006, a FTT é um ambiente de desenvolvimento de *software* vinculada aos Cursos Bacharelados de Computação da UniEvangélica, que oferece aos graduandos uma experiência em fábrica de *software* (TURING, 2017). Neste trabalho, considerou-se o projeto de *software* VIRTOO – Sistema de Gerenciamento Acadêmico para as faculdades Instituto Superior Politécnico Vida (ISPVida) e Instituto Superior de Teologia Evangélica no Lubango (ISTEL) localizadas em Lubango na Angola, com módulos já em uso.

A Fábrica de Tecnologias Turing onde foram aplicados os testes utiliza um modelo de processo hibrido na visão de Alvares (2013) nesse tipo de processo o gerenciamento de projetos une duas ou mais metodologias ou a quantidade necessária de frameworks disponíveis no mercado. Dessa possibilidade é possível extrair somente aquilo que agregue valor e qualidade ao que estiver sendo desenvolvido aumentando dessa forma o nível do gerenciamento.

A metodologia ágil *Scrum* é utilizada na Fábrica se trata de uma ferramenta extremamente ágil e flexível, que tem como objetivo definir um processo interativo e incremental de desenvolvimento, sua aplicação é indicada a qualquer produto e no gerenciamento de atividades mais complexas. Essa ferramenta se caracteriza pela sua versatilidade e objetividade dentro de situações com alto grau de dificuldade. O *Scrum* centra sua atenção na equipe, melhorando a comunicação e a cooperação, fazendo com que todos se sintam melhores refletindo num ganho de produtividade (BISSI, 2007).

Outro processo utilizado é o Open Up que se trata de uma metodologia considerada interativa e unificada utilizada dentro de um ciclo de vida estruturado. Esse modelo de ferramenta se caracteriza por ser livre e com um baixo nível de formalidade, podendo ser usado em uma infinidade de projetos não somente no desenvolvimento de *software* (FAGUNDES, 2011).

4.2 FERRAMENTA JMETER

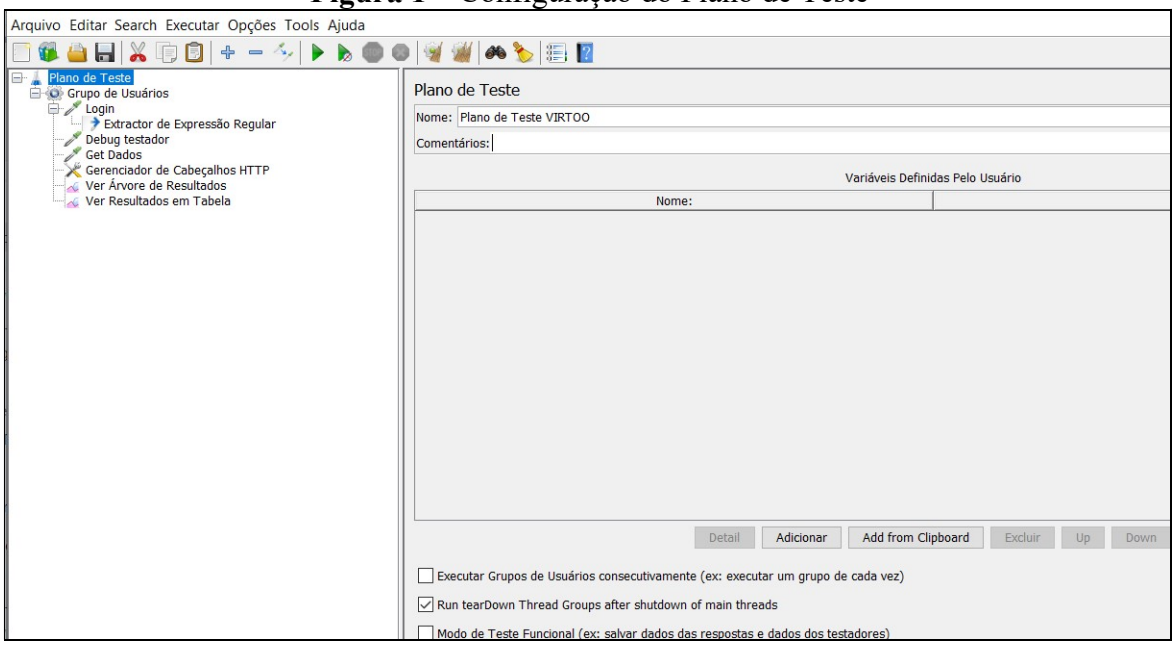
Para o desenvolvimento da pesquisa foi utilizado a ferramenta JMETER que se trata de um *software* de código aberto que serve para analisar o comportamento funcional de um modelo de teste a ser aplicado visando medir o desempenho.

As funções JMeter são valores especiais que podem preencher campos de qualquer Sampler ou outro elemento em uma árvore de teste. Existem dois tipos de funções: valores estáticos definidos pelo usuário (ou variáveis) e funções internas. Os valores estáticos .definidos pelo usuário permitem que o usuário defina as variáveis a serem substituídas pelo seu valor estático quando uma árvore de teste é compilada e enviada para execução (APACHE, 2019).

Essa substituição ocorre uma vez no início da execução do teste. Isso pode ser usado para substituir o campo DOMAIN de todas as solicitações HTTP, por exemplo - tornando simples alterar um teste para atingir um servidor diferente com o mesmo teste (APACHE, 2019). Assim por sua grande gama de utilização esse tipo de aplicativo se mostra o mais adequado para a realização dos testes que aqui se propõe (carga, estresse e desempenho) esse processo os testes serão aplicados no início de uma *sprint*.

A ferramenta permite realizar uma serie de configurações para execução dos testes, os elementos ficam em formato de árvore onde é possível estruturar o Plano de Teste (TestPlan) conforme a Figura.1:

Figura 1 – Configuração do Plano de Teste



Fonte: Elaborada pela autora

De acordo com Santos e Santos Neto (2008) na composição da árvore de testes alguns itens são hierárquicos, como os Ouvintes (relatórios) e as *Assertions* (asserções), ligados a outros elementos de uma ordem maior na ordem hierárquica. Já os *Samplers* (requisições), são ordenados de forma primária, por isso surgem em uma ordem na qual aparecem verticalmente na árvore determina sua ordem de execução.

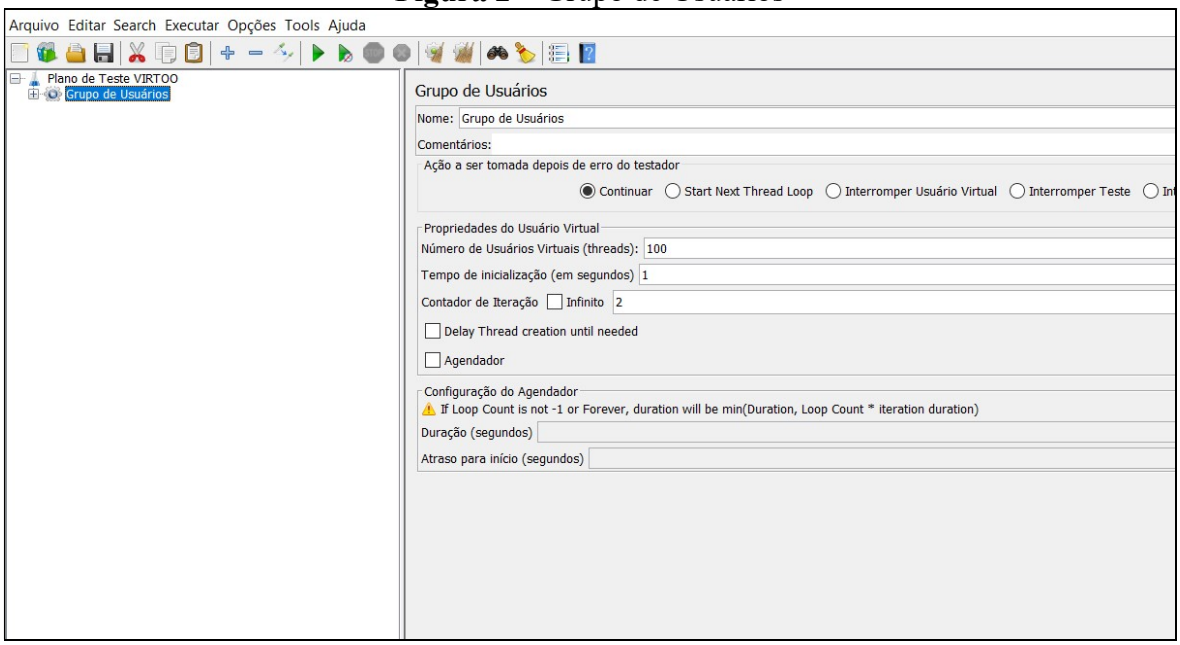
Essa visualização também é idêntica quando em modo de arquivo XML gerado pela ferramenta para a persistência dos elementos. A execução dos testes com o JMeter, pode ser efetivada de duas maneiras: em uma máquina só ou de forma distribuída, na qual o esforço do teste é distribuído dentre diversas máquinas. É através dela que os testadores podem conseguir uma maior fidelidade na recriação de cenários de teste, pois com a distribuição da execução do teste entre várias máquinas evita-se gargalos tanto de processamento quanto de caminhos na rede do sistema sob teste (SANTOS; SANTOS NETO, 2008).

Na inicial do Apache JMeter, sendo que o primeiro passo a ser realizado é a iniciação do grupo de usuários e a configuração com base nos requisitos exigidos para cada tipo de teste a ser realizado. Sendo que existe a possibilidade de se poder determinar uma repetição continua na qual é possível apurar os resultados e se ter uma noção clara do possível erro que esteja acontecendo.

No Plano de Teste (Figura 1) é possível definir um nome identificador para o Plano do Teste, adicionar comentários, criar variáveis globais, que são visíveis por todas as requisições do plano de teste em questão, além de permitir a definição de parâmetros ou comportamentos comuns a todos os testes.

O Grupo de Usuários (Figura 2) serve para representar um grupo de usuários virtuais. Sendo que é possível se ter vários grupos de usuario ligados ao Test Plan, e todos os outros elementos, com exceção do *Work Bench*, devem ser adicionados como filho de algum *Thread Group*.

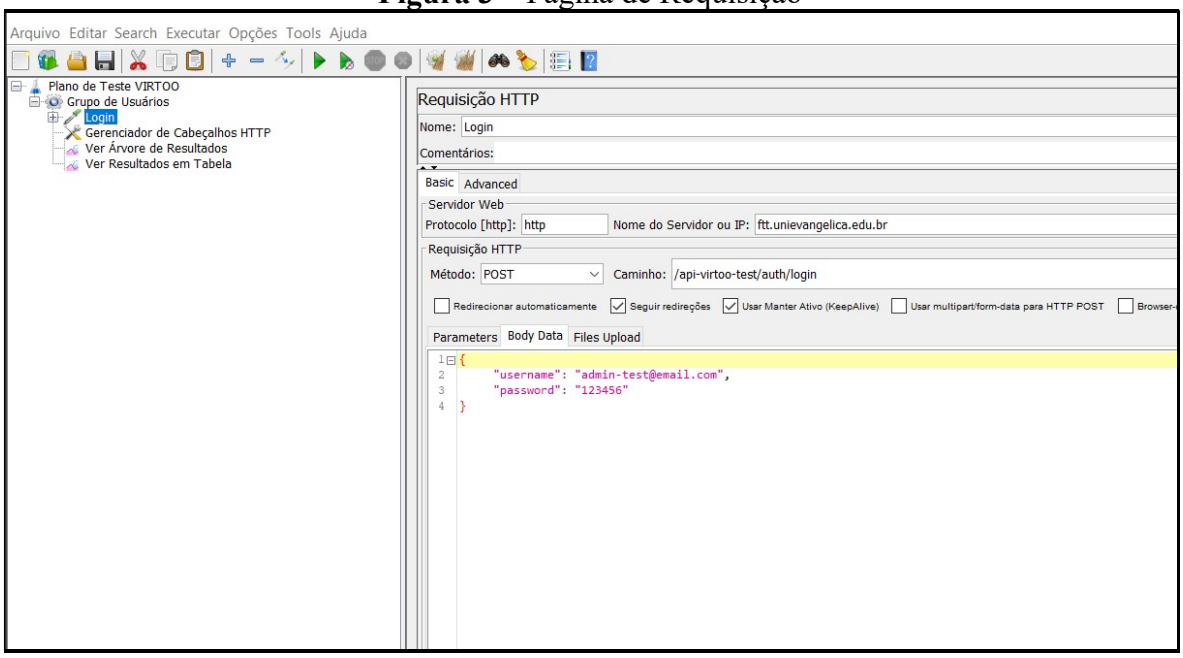
Figura 2 – Grupo de Usuários



Fonte: Elaborada pela autora

O HTTP Request permite o envio de requisições HTTP/HTTPS ou arquivos para um servidor Web. Na Figura 3 tem-se um exemplo de um HTTP Request adicionado a um Plano de Teste.

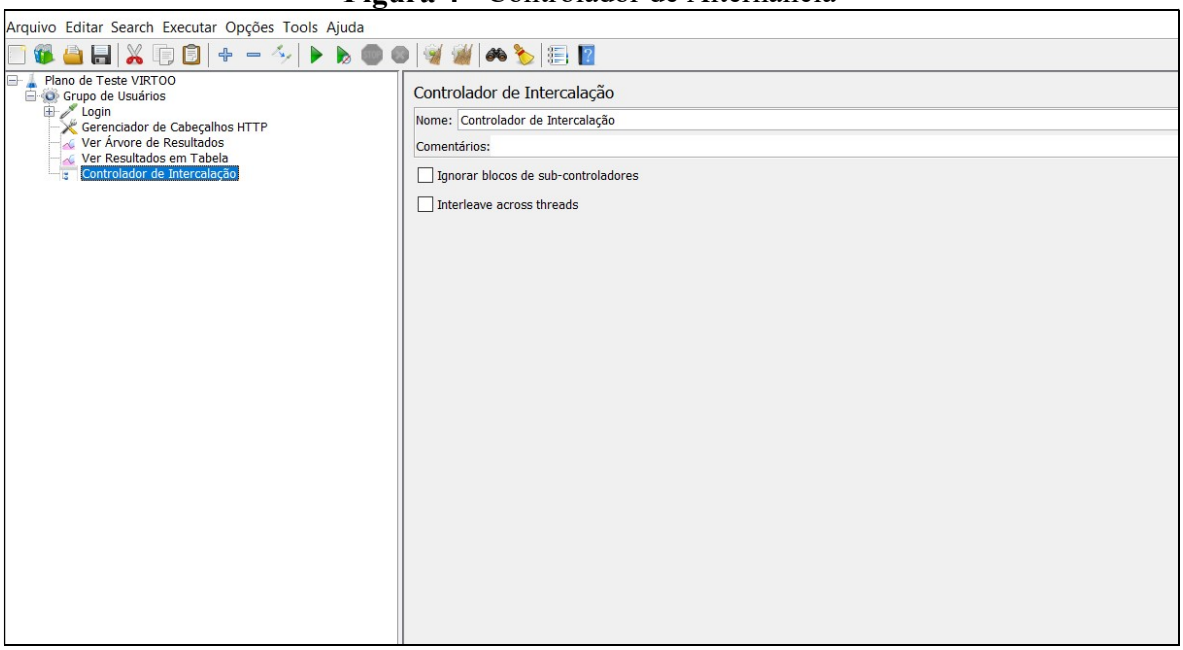
Figura 3 – Página de Requisição



Fonte: Elaborada pela autora

O JMeter possui o *Random Controller*, que é similar ao *Interleave Controller*. A única diferença entre eles é o fato desse não alternar entre as requisições com base na ordem em que foram colocadas. O *Random Controller* (Controlador de Alternância) simplesmente escolhe aleatoriamente um dos seus filhos a cada iteração, de acordo com a Figura 4.

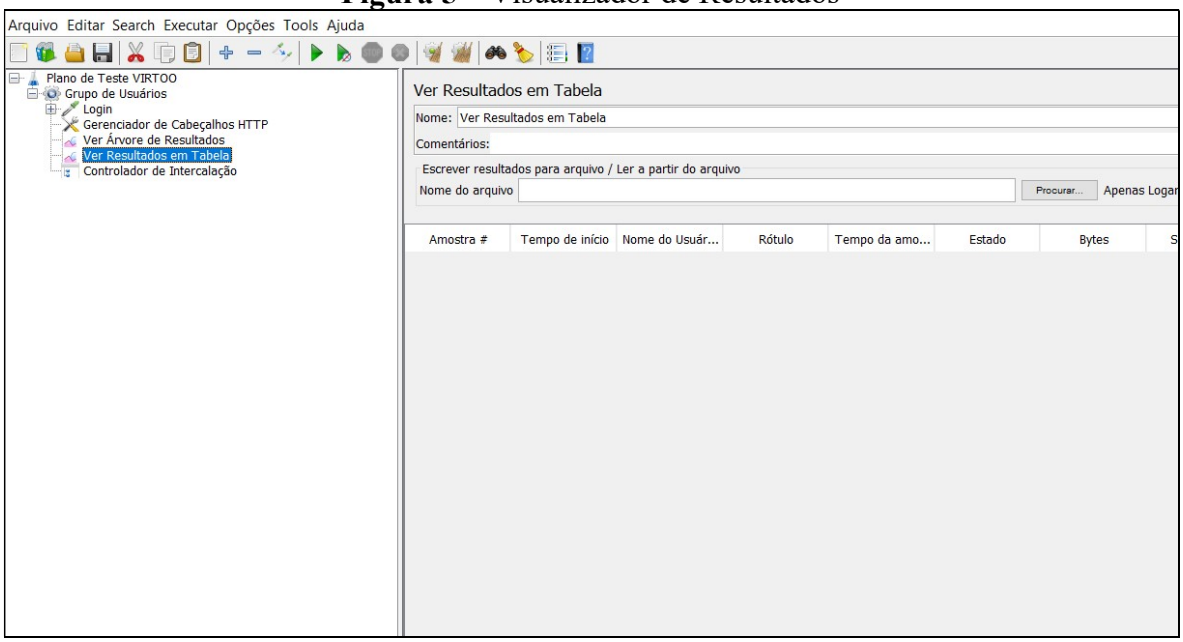
Figura 4 - Controlador de Alternância



Fonte: Elaborada pela autora

Para visualizar os resultados dos testes é necessário adicionar um Ouvinte ao plano de teste. Os Ouvintes capturam os dados das informações durante a execução dos testes e criam relatórios e gráficos com os resultados obtidos. Além de exibir os resultados, os Ouvintes também possibilitam a geração de um arquivo, que pode ser inclusive uma planilha eletrônica, contendo tais resultados (Figura 5).

Figura 5 – Visualizador de Resultados



Fonte: Elaborada pela autora

4. RESULTADOS ALCANÇADOS

Para se conseguir atingir o objetivo proposto de escolher a mais adequada abordagem de aplicação de teste de *performance* às fábricas de *software* ágeis foi desenvolvida uma pesquisa direta com a equipe da FTT.

Os resultados apurados na pesquisa dentro da FTT serviram para analisar as principais dificuldades e medidas de controle de *performance* e quais as principais ferramentas devem ser utilizadas dentro das fábricas de *software* em relação aos testes de carga, estresse e desempenho.

Um dos pilares educacionais e formadores da FTT é a ênfase na constante intenção de incentivar e aperfeiçoar seu corpo de alunos, o que vem dando resultados positivos no que concerne ao quantitativo de vagas preenchidas em empresas pelos alunos nela matriculados.

A FTT abre oportunidades para todos os alunos do curso de Engenharia de Computação da UniEVANGÉLICA, na maioria acadêmicos dos primeiros semestres sem experiência no mercado de trabalho, isto pois a fábrica serve para colocar em prática o que é aprendido em sala de aula e principalmente capacita-los para o trabalho utilizando processos e ferramentas atuais. Estes alunos após adquirirem qualificação e experiência muita das vezes são visados pelas empresas da região e logo se despedem do quadro, gerando outra característica da FTT a alta rotatividade, devido esta, o grupo precisa ser constantemente capacitado para minimizar as oscilações de velocidade no desenvolvimento dos *software*s.

Diante das demandas encontradas foi proposta esta pesquisa para apoio dos novos integrantes na aplicação do teste de *performance* para contribuir com tempo e trabalho de capacitação de processo.

Foi verificado se o modelo aqui desenvolvido é condizente com a realidade da fábrica onde a pesquisa foi desenvolvida, e como ele pode auxiliar na qualidade final do *software* desta forma pretendeu-se também colaborar cientificamente para a evolução das discussões sobre o tema aqui abordado, mesmo que de maneira pequena.

Foi realizada uma pesquisa presencialmente com o Scrum Master, P.O, líder da equipe de teste e coordenadores das equipes com a finalidade de entender os processos executados. Junto com a equipe foram definidos os requisitos que melhor se adequam para o início dos testes e foi realizada as estimativas da *performance* dos requisitos não-

funcionais conforme um o trabalho aplicado na própria FTT. A técnica PERT aborda pontos importantes nas estimativas, tal como os riscos e tem a finalidade de direcionar, viabilizar e ajudar a controlar os ciclos de desenvolvimento de software (FERREIRA, 2016).

Conforme Santos e Santos Neto (2008, p. 02) para se definir qual a *performance* dos requisitos é importante:

> O teste de *software* é a verificação dinâmica do funcionamento de um programa utilizando um conjunto finito de casos de teste, adequadamente escolhido dentro de um domínio de execução infinito, contra seu comportamento esperado. Nesse conceito existem alguns elementos chaves: dinâmica: o teste exige a execução do produto, embora algumas de suas atividades possam ser realizadas antes do produto estar operacional; conjunto finito: o teste exaustivo é geralmente impossível, mesmo para produtos simples; escolhido: é necessário selecionar testes com alta probabilidade de encontrar defeitos, preferencialmente com o mínimo de esforço; comportamento esperado: para que um teste possa ser executado é necessário saber o comportamento esperado, para que ele possa ser comparado com o obtido.

A aplicação de teste de *performance* é um processo novo para a equipe de teste, foram disponibilizados materiais de apoio para utilização da ferramenta JMeter e um workshop para auxiliar na execução e sanar dúvidas.

Os resultados desta pesquisa foram obtidos através dos requisitos Manter Login e Listar Disciplina, desenvolvidos no projeto VIRTOO através destes foram estimadas as seguintes informações conforme Tabela 1:

Tabela 1 – Estimativas dos requisitos não-funcionais

Requisito	Carga máxima de usuários simultâneos	Tempo ideal de resposta das requisições	Número de Requisições
Realizar Login	30 usuários	2 segundos	40
Listar Disciplina	30 usuários	2 segundos	40

Fonte: Elaborada pela autora

A Tabela 1 contém informações pré-definidas, para analisar o comportamento dos requisitos, precisa realizar oscilações para obter resultados do comportamento das interações.

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 (MENEZES, 2018).

A equipe de desenvolvimento disponibilizou um ambiente de homologação que possui as mesmas especificações de produção, no qual verificou-se as seguintes especificações:

- Processador: Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz;
- Memória RAM: 12 GB;
- Disco Rígido: 85.9 GB;
- Sistema Operacional: Linux CentOs 6.5;
- Versão Apache JMeter: 5.1;

No Grupo de Usuários foi definido o numero de vezes que os usuários devem enviar requisições ao servidor e o tempo de inicialização entre cada usuário.

Na quantidade de Usuários Virtuais (threads) foi atribuído o valor 30 considerado pelas estimativas uma quantidade significativa para qualidade do *software*, através da variação de usuários virtuais conseguiu-se testar a carga e estresse do *software*.

No preenchimento do campo Tempo de Inicialização (em segundos) foi definido com o valor 2, este é o tempo de demora entre o início de cada usuário, no contador de iteração e a configuração que define a quantidade de vezes que os usuários virtuais realizarão as requisições no servidor.

Na execução da ferramenta JMeter é possível analisar vários dados, mas o foco deste trabalho é analisar:

- Tempo mínimo de resposta das requisições HTTP;
- Tempo máximo de resposta das requisições HTTP;
- Tempo mediano da resposta das requisições HTTP;
- Quantidade total de requisições que a aplicação suporta.

4.1 RESULTADOS JMETER

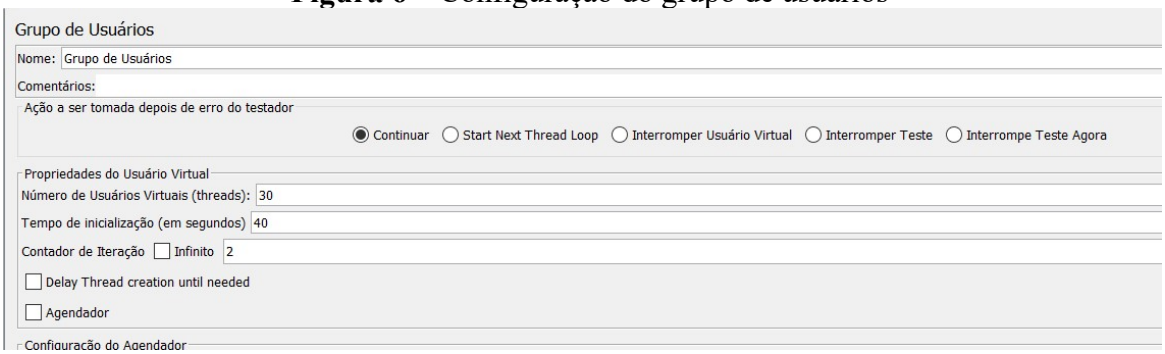
Foi realizado o acompanhamento com a equipe de teste da FTT para configuração do ambiente, entende-se que somente depois de exaustivamente discutido e testado o

aprimoramento do *software* pode ser conseguido e desta forma conceder aos clientes a qualidade necessária que se espera do produto adquirido.

Com a finalidade de se testar a *performance*, o requisito Listar Disciplina foi populado com dados nas tabelas para simular um ambiente em produção, utilizando o método GET.

O estresse e a carga trabalharam juntos, com a alternação de valores entre número de usuários virtuais e a quantidade de requisições no servidor. Inicialmente foi configurado o grupo de usuários conforme Figura 6.

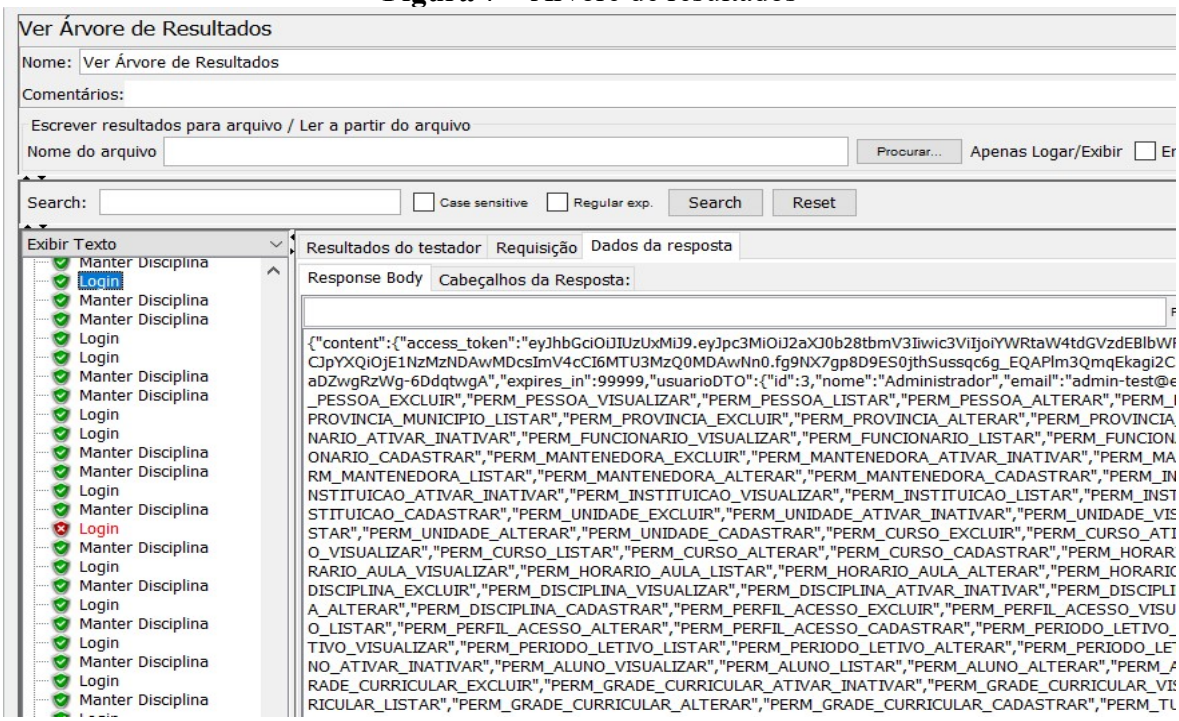
Figura 6 – Configuração do grupo de usuários



Fonte: Elaborada pela autora

Através do relatório Árvore de Resultados é possível identificar quais requisições tiverem sucesso, no teste realizado as 1.600 requisições foram realizadas, no requisito Manter Login chegou a 0,25% de erro nas requisições realizadas, no requisito Listar Disciplina todas as requisições foram realizadas com sucesso conforme figura 7. A variação de carga de usuários virtuais foi de 30 que é considerado um resultado bom para a equipe de especificação de requisitos.

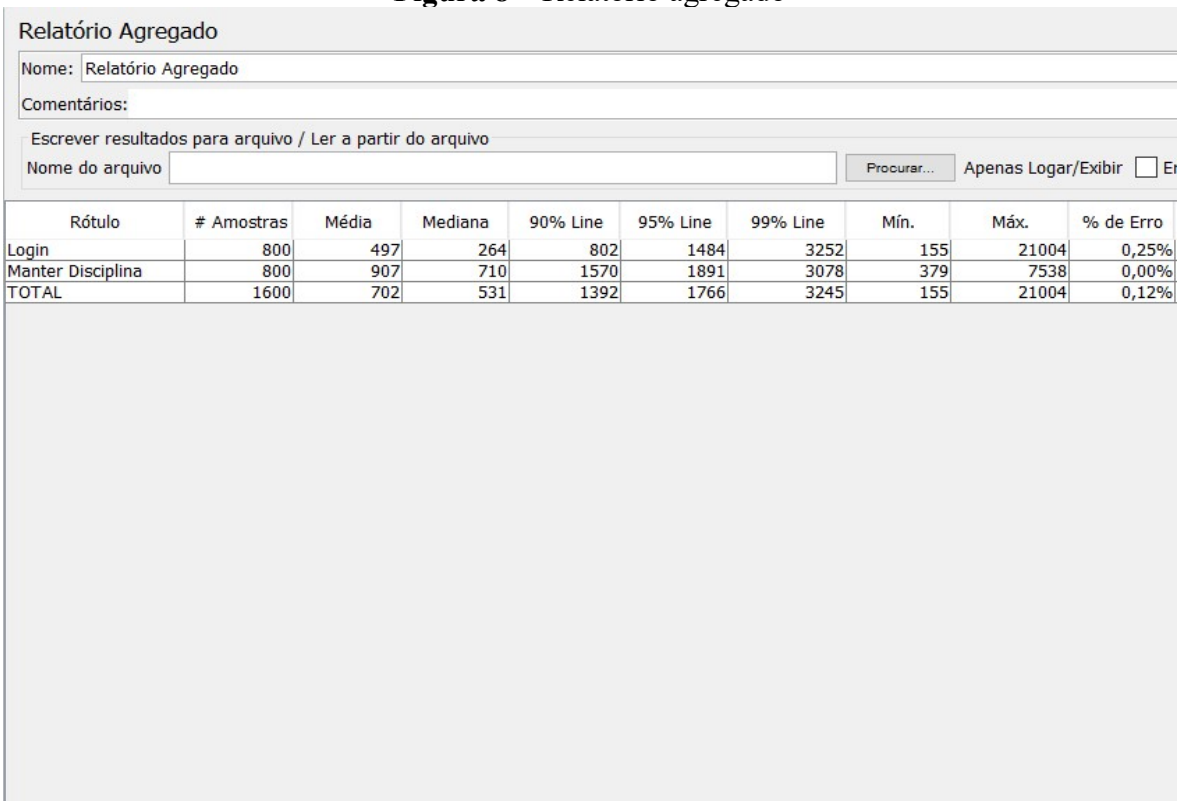
Figura 7 – Árvore de resultados



Fonte: Elaborada pela autora

Com a finalidade de obter os resultados dos questionamentos citados anteriormente no ouvinte “Relatório Agregado” que separa por linha cada requisito testado é possível analisar a quantidade de requisições na coluna # Amostras e o totalizador das requisições feitas no plano de teste.

Figura 8 – Relatório agregado



Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Login	800	497	264	802	1484	3252	155	21004	0,25%
Manter Disciplina	800	907	710	1570	1891	3078	379	7538	0,00%
TOTAL	1600	702	531	1392	1766	3245	155	21004	0,12%

Fonte: Elaborada pela autora

A média de tempo em milissegundos dos requisitos foi diferente, no requisito Manter Login teve em média 0,4 segundos na realização das requisições, já no Listar Disciplina foi de 0,9 segundos.

Na coluna Mediana pegou-se todas as ocorrências, os tempos de carregamento em ordem ascendente localizando o ponto central para realizar as comparações, no Realizar Login 50% dos usuários virtuais tiveram como variação de tempo entre 155 ms ate 264 ms, os outros 50% tiveram variação 264 ms até a máxima 21004 ms. No requisito Listar Disciplina a variação de 50% dos usuários foi de 379 ms com a mediana 710 ms e os outros 50% foi de 710 ms com variação até 7538 ms.

A coluna % de Erro apresentou 0,25% de erro ao realizar as requisições conforme informado anteriormente.

Throughput ou vazão, conforme Ribeiro (2013) é a quantidade de vezes que se consegue receber uma resposta completa de uma página por segundo. Podendo influenciar no requisito testado, a baixa vazão demonstra certa instabilidade quando a carga for maior.

5. CONSIDERAÇÕES FINAIS

Para se conseguir atingir o objetivo proposto de escolher o mais adequado tipo de teste de *performance* às fábricas de *software* ágeis foi desenvolvida uma pesquisa direta com colegas acadêmicos do curso de engenharia da Computação da UniEVANGÉLICA.

Os resultados apurados servirão para que fosse possível realizar um estudo mais aprofundado quanto a escolha da técnica e das ferramentas mais adequadas ao teste de *performance* de *software.* A partir disso chegou-se à conclusão de que no experimento aqui apresentado a ferramenta JMETER apresentou-se como o elemento mais indicado a ser aplicado, por ser um *software* de código aberto.

Os resultados alcançados na pesquisa direta com os integrantes da FTT trouxe como principal informação a importância de realizar o teste de *performance* para contribuir para qualidade final do produto desenvolvido. Os resultados colhidos serviram para entender os limites da aplicação, os requisitos não apresentaram uma *performance* com resultados totalmente satisfeitos na carga de usuários, estresse ocasionado pelo numero de requisições e o tempo relacionado a *performance* da aplicação, sendo assim a pesquisa realizada evidencia que o software não está apto a uma grande quantidade de requisições e usuários simultâneos.

A execução do teste de *performance* contribuiu para a qualidade final do projeto VIRTOO os requisitos que passaram pelo processo possuem uma visão de como irão se comportar em determinadas situações de uso. É possível identificar, analisando os relatórios, se o requisito está apto para ser usado por uma quantidade de usuários ou se irá suportar as requisições esperadas.

Um dos problemas citados foi a dificuldade da aplicação do teste de *performance* nas fabricas de software, por falta de conhecimento, custo e profissionais qualificados, este trabalho contribuiu mostrando um roteiro, a importância e como realizar a aplicação.

Desta forma, entende-se que o objetivo geral foi atingido, uma vez que a ferramenta JMeter se mostrou indicada a realizar testes de desempenho confiáveis e que se tornam uma opção viável dentro da FTT. Entende-se que somente depois de exaustivamente discutido e testado o aprimoramento do software pode ser conseguido e desta forma conceder aos clientes a qualidade necessária que se espera do produto adquirido. A um potencial trabalho futuro pode ser relacionado ao uso dos resultados para o mapeamento de riscos e oportunidades do projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT. Associação Brasileira de Normas Técnicas. (2005), NBR ISO 9000:2005. Disponível em: https://qualidadeuniso.files.wordpress.com/2012/09/nbr-iso-9000-2005.pdf. Acesso em: 16 set. 2019.

Alvares, D. L. S. (2013), Modelo hibrido no gerenciamento de projetos. Disponível em: https://www.devmedia.com.br/modelo-hibrido-no-gerenciamento-de-projetos/37289. Acesso em: 15 set. 2019.

Apache, S. F. (2019) JMETER visão global. Disponível em: https://jmeter.apache.org/index.html. Acesso em: 10 set. 2019.

Ázara, L. N. (2013), A indústria de *software* no contexto institucional: um estudo em dois municípios de Minas Gerais. Dissertação (Mestrado em Administração) – Universidade Federal de Lavras, Lavras.

Barbosa, E. F. et al. (2016), Introdução ao teste de *software*. Disponível em: https://edisciplinas.usp.br/pluginfile.php/2266784/mod_resource/content/1/relatorio_teste.pdf. Acesso em: 06 out. 2018.

Batista, F. de M. (2014). O que é mesmo Qualidade? Blog da Qualidade. Disponível em: http://web.archive.org/web/20170506140818/http://www.blogdaqualidade.com.br/oque-e-qualidade/. Acesso em: 09 out. 2019.

Bissi, W. (2007), Scrum - metodologia de desenvolvimento ágil. Campo Dig., Campo Mourão, v.2, n.1, p.3-6, jan/jun.

Britto, J., Stallivieri, F. (2010), Inovação, cooperação e aprendizado no setor de *software* no Brasil: análise exploratória baseada no conceito de Arranjos Produtivos Locais (APLs). Economia e Sociedade, Campinas, v. 19, n. 2 (39), p. 315-358.

Campos, F. M. (2010). Teste de desempenho: Conceitos, Objetivos e Aplicação - Parte 1. Disponível em: http://www.linhadecodigo.com.br/artigo/3256/teste-de-desempenho-conceitos-objetivos-e-aplicacao-parte-1.aspx#ixzz616suQZH7. Acesso em: 30 set. 2019.

Campos, F. M. (2011). Testes de Performance - Testes de Carga, Stress e Virtualização - Parte 3. Disponível em: http://www.linhadecodigo.com.br/artigo/3259/testes-de-performance-testes-de-carga-stress-e-virtualizacao-parte-3.aspx#ixzz616UxPOz4. Acesso em: 30 set. 2019.

Cérgoli, R. (2017). Uma análise da aplicação de testes no desenvolvimento de um sistema erp. Disponível em: https://spo.ifsp.edu.br/images/phocadownload/DOCUMENTOS_MENU_LATERAL_FIXO/POS_GRADUA%C3%87%C3%83O/ESPECIALIZA%C3%87%C3%83O/Gest%C3%A3o_da_Tecnologia_da_Informa%C3%A7%C3%A3o/PRODUCAO/2017/UmaC3%A7%C3%A3o_de_Testes_no_Desenvolvimento_de_um_Sistema_ERP.pdf. Acesso em: 09 out. 2019.

Coser, A. (2012), Modelo para análise da influência do capital intelectual sobre a performance dos projetos de *software*. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/99386/304292.pdf?sequence=1&isAllowed=y. Acesso em: 23 out. 2018.

Cobra, M. (2007), Marketing e moda. São Paulo: SENAC.

Delamaro, M. E. et al. (2016), Introdução ao teste de *software*. 2. ed. Rio de Janeiro: Elsevier.

Fagundes, R. M. (2011). Engenharia de requisitos: Do perfil do analista de requisitos ao desenvolvimento de requisitos com UML e RUP. São Paulo: Atlas, 2011.

Ferreira, F. S. (2016), Gestão ágil de projetos: método para estimativa em projeto de *software* para fábrica de *software* acadêmica. Disponível em: http://repositorio.aee.edu.br/bitstream/aee/317/1/TCC2_2016_02_FelixSoares.pdf. Acesso em: 22 set. 2019.

Freitas, A. L. S. da C. (2013), Ontologia para teste de desempenho de *software*. Disponível em: http://repositorio.pucrs.br/dspace/bitstream/10923/1501/1/000449315-Texto%2bCompleto-0.pdf. Acesso em: 24 out. 2018.

Gomes, V. (2015). Teste de desempenho de *software*. Disponível em: https://www.tiespecialistas.com.br/teste-de-desempenho-de-*software*/. Acesso em: 29 set. 2019.

Guarienti, P. et al. (2014). Uma Abordagem de Análise do Tempo de Resposta para Teste de Desempenho em Aplicações Web. Disponível em: http://www.sbrc2014.ufsc.br/anais/files/wtf/ST1-2.pdf. Acesso em 30 set. 2019.

Jorge, F. de F. et al. (2015), A evolução do jogo *iTestLearning* para o ensino das atividades de execução de testes de *software*. Disponível em: http://www.tise.cl/volumen11/TISE2015/295-305.pdf. Acesso em: 16 jun. 2019.

Koscianski, A.; Soares, M. S. (2007), Qualidade de *software*: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de *software*. 2. ed. São Paulo: Novatec.

Lutz, D. (2017), Implantação de novo processo de trabalho em uma fábrica de *software* baseado nos modelos ágeis de desenvolvimento. Disponível em: https://www.univates.br/bdu/bitstream/10737/1948/1/2017DouglasFernandoLutz.pdf. Acesso em: 05 maio 2019.

Menezes, T. (2018). Visão geral: teste de performance. Disponível em: https://medium.com/@tspm/vis%C3%A3o-geral-testes-de-performance-efa9f5e4f90. Acesso em: 30 set. 2019.

Moraes, M. H. B. M.; LIMA JUNIOR, F. R. (2017). Proposição e aplicação de uma metodologia baseada no AHP e na ISO/IEC 25000 para apoiar a avaliação da qualidade de softwares de gestão de projetos. **Revista Gestão da Produção Operações e Sistemas**, v. 12, n. 2, p. 239, abr. 2017.

Observatório Softex. (2012), *Software* e serviços de TI: A indústria brasileira em perspectiva. Versão resumida. Disponível em: http://www.softex.br/wp-content/uploads/2013/07/Ingl%C3%AAs-e-Portugu%C3%AAs-%E2%80%93-Vers%C3%A3o-Resumida-Volume-2.pdf. Acesso em 08 out. 2018.

Pressman, R. A. (2016). Engenharia de *Software*: uma abordagem profissional. McGraw Hill Brasil: 2016.

Prodanov, C. C., Freitas, E. C. de. (2013), Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico. 2. ed. Novo Hamburgo/RS: FEEVALE.

Ruffato, B. R. (2010), O papel do governo brasileiro no fomento das inovações no setor das TICs: um enfoque na indústria de *software*. Monografia (Graduação em Economia)

– Faculdade de Ciências Econômicas, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Santos, I. S. et al. (2010), Requisitos e Aspectos Técnicos desejados em ferramentas de testes de *software*: um estudo a partir do uso do SQFD. Revista Eletrônica de Sistemas de Informação, v. 9, n. 2, artigo 10.

Santos, I. S., Santos Neto, P. A. (2008). Automação dos Testes de Desempenho e Estresse com o Apache JMeter. Disponível em: https://sites.google.com/site/ismaylesantos/geracao-de-testes-de-desempenho. Acesso em: 29 set. 2019.

Santos, T. G. (2014), SIG como ferramenta para a regularização fundiária da comunidade rural do retiro - Aracruz/ES. Disponível em: http://www.geo.ufes.br/sites/geografia.ufes.br/files/field/anexo/Talles.pdf. Acesso em: 09 maio 2019.

Silva, F. J. da S. (2009), *Software* livre: conceitos, história e impactos. 54 slides. Disponível em: http://www.deinf.ufma.br/~fssilva/palestras/2009/sl.pdf. Acesso em: 07 out. 2018.

Silva, W. M., Calazans, A. T. S. (2012), Ferramentas free para teste de *software*: um estudo comparativo Universitas Gestão e TI, v. 2, n. 2, p. 57-72, jul./dez.

Sommerville, I. (2011), Engenharia de *Software*. 9. ed. São Paulo: Pearson Prentice.

Souza, T. S. (2018). Testes de Desempenho de *Software*: Teoria e Prática. Disponível em: https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/7/13/39-1?inline=1. Acesso em: 01 out. 2019.

Sparano, J. C. (2012). Teste de estresse. Disponível em: http://analistajcs.blogspot.com/2012/03/teste-de-estresse.html. Acesso em: 01 out. 2019.

Steda, M. M. V. (2015), Tecnologias da informação e território: políticas para o setor de *software* no Brasil. Disponível em: http://repositorio.unicamp.br/bitstream/REPOSIP/286568/1/Steda_MelissaMariaVeloso_M.pdf. Acesso em: 03 out. 2018.

Tenório, F. G., Valle, R. (2012), Fábrica de *software*. Rio de Janeiro: FGV.

Vilas Boas, A. L. C. (2007), Qualidade e Avaliação de Produto de *Software*. Lavras/MG: FAEPE/UFLA.

Yang, H. (2012). Measuring software product quality with ISO standards base on fuzzy logic technique. In: LUO, J. (Ed.). Affective Computing and Intelligent Interaction. New York: Springer-Verlag Berlin Heidelberg, p. 59-67.