CENTRO UNIVERSITÁRIO DE ANÁPOLIS – UniEVANGÉLICA
BACHARELADO EM ENGENHARIA DE COMPUTAÇÃO

PROCESSO MÍNIMO DE TESTE DE PERFORMANCE PARA FÁBRICAS DE SOFTWARE

JOÃO VITOR MOREIRA DE SOUSA
LUCAS NUNES SANTANA

ANÁPOLIS
2020

JOÃO VITOR MOREIRA DE SOUSA
LUCAS NUNES SANTANA

PROCESSO MÍNIMO DE TESTE DE PERFORMANCE PARA FÁBRICAS DE SOFTWARE

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. Ma. Walquíria Fernandes Marins.

ANÁPOLIS
2020

JOÃO VITOR MOREIRA DE SOUSA
LUCAS NUNES SANTANA

PROCESSO MÍNIMO DE TESTE DE PERFORMANCE PARA FÁBRICAS DE SOFTWARE

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

Aprovado(a) pela banca examinadora em 07 de dezembro de 2020, composta por:

_____
Prof. Ma. Walquíria Fernandes Marins
Orientador

_____
Prof. Ma. Luciana Nishi

_____
Prof. Me. Lucas de Almeida Ribeiro

AGRADECIMENTOS

Agradecemos primeiramente a Deus pelo dom da vida, sabedoria e por nos permitir chegar até aqui.

Aos nossos pais e familiares por sempre estarem conosco e nos proporcionar a possibilidade de estudar, não seria possível sem o esforço deles.

A nossa orientadora professora Walquíria Fernandes Marins pelo apoio e colaboração na elaboração deste trabalho, sempre nos ajudou com a maior boa vontade possível.

Aos nossos amigos de curso, que estão presentes no nosso dia a dia, ajudando uns aos outros.

Aos docentes da UniEvangélica, em especial do nosso curso, e todos os membros da FTT.

Resumo

A sistematização de um processo requer conhecimento uma fundamentação teórica sólida e confirmação de eficiência através de numerosos testes, para que possa ser implementada de maneira efetiva e agregar valor na qualidade do produto. No contexto da gestão da qualidade de *software*, é escassa a aplicação de processos para articulação de testes de *performance* nos sistemas, embora apresentem impacto relevante sobre o resultado final. Este tipo de teste não funcional contribui para aspectos de desempenho de uma aplicação, tais como tempo de resposta e carga de usuários suportada. O presente trabalho tem como objetivo formalizar e aplicar um processo de teste de *performance* que seja útil para ambientes de fábrica de *software*. A intervenção é avaliada através de um experimento na Fábrica de Tecnologias Turing (FTT), com a realização de testes utilizando ferramentas na qual são coletados dados através de questionários e indicadores de testes para comparações de eficiência da aplicação do processo.

Abstract

The systematization of a process requires knowledge of a solid theoretical foundation and confirmation of efficiency through numerous tests, so that it can be implemented effectively and add value to the quality of the product. In the context of software quality management, the application of processes to articulate performance tests in systems is scarce, although they have a relevant impact on the final product. This type of non-functional test contributes to performance aspects of an application, such as response time and supported user load. This work aims to formalize and apply a performance test process that is useful for software factory environments. The intervention is evaluated through an experiment in the Turing Technology Factory (Fábrica de Tecnologias Turing - FTT), with tests performed using tools in which data are collected through questionnaires and test indicators for comparing the efficiency of the process application.

LISTA DE ILUSTRAÇÕES

Figura 1 – Relacionamento entre os documentos de um processo de teste. 14
Figura 2 – Interface inicial do JMeter.. 19
Figura 3 – Interface inicial da extensão do BlazeMeter. 20
Figura 4 – Etapas da pesquisa..... 23
Figura 5 – Processo de desenvolvimento/teste da FTT. 26
Figura 6 – Processo adaptado de desenvolvimento/teste da FTT. 27
Figura 7 – Processo de teste de *performance*.. 28
Figura 8 – Interface de exportação de do BlazeMeter. 31
Figura 9 – Estrutura do plano de teste.. 32
Figura 10 – Grupo de usuários 33
Figura 11 – Requisição HTTP. 34
Figura 12 – Visualização de resultados. 35
Figura 13 – Conhecimento em teste de *performance*.. 37
Figura 14 – Tempo de trabalho na FTT. 38
Figura 15 – Período do curso dos integrantes da equipe de teste.. 38
Figura 16 – Conhecimento acerca das ferramentas de teste 39
Figura 17 – Requisito Turma com 1 usuário. 41
Figura 18 – Requisito Turma com 10 usuários. 42
Figura 19 – Requisito Turma com 30 usuários. 42
Figura 20 – Requisito Disciplina com 1 usuário. 43
Figura 21 – Requisito Disciplina com 10 usuários. 43
Figura 22 – Requisito Disciplina com 30 usuários. 44
Figura 23 – Tempo de resposta x Carga de usuários – Disciplina. 45
Figura 24 – Tempo de resposta x Carga de usuários – Turma. 46
Figura 25 – Tempo de resposta x Velocidade de internet - Disciplina. 47
Figura 26 – Tempo de resposta x Velocidade de internet - Turma. 48
Figura 27 – Tempo de trabalho na FTT - Q1 e Q2. 49
Figura 28 – Equipes dos integrantes - Q1 e Q2. 49
Figura 29 – Documentação não funcional – Q1. 50
Figura 30 – Documentação não funcional – Q2. 50
Figura 31 – Atividades de teste de *performance* na FTT– Q1. 51
Figura 32 – Atividades de teste de *performance* na FTT – Q2. 52
Figura 33 – Experiência com o JMeter– Q1. 52
Figura 34 – Experiência com o JMeter – Q2. 53

Figura 35 – Novo processo de teste – Q1 e Q2 53

SUMÁRIO

1 INTRODUÇÃO 10
2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 Qualidade de *software* 13
2.2 Processo de teste 14
2.3 Testes de desempenho (*performance*) 15
2.4 Ferramenta JMeter 18
2.4 Ferramenta BlazeMeter 19
2.5 Fábricas de *software* 20
3 MÉTODO DE PESQUISA 22
4 ABORDAGEM PROPOSTA 25
4.1 Fábrica de Tecnologias Turing (FTT) 25
4.2 Novo processo de teste 28
4.3 Abordagem BlazeMeter 30
4.4 Abordagem JMeter 31
4.5 Projeto de experimento 36
5 RESULTADOS ALCANÇADOS 40
5.1 Resultados JMeter 40
5.2 Comparação de questionários 48
6 CONSIDERAÇÕES FINAIS 55
REFERENCIAS BIBLIOGRÁFICAS 57

1 Introdução

O processo de desenvolvimento de *software* envolve uma série de atividades através das quais são aplicadas técnicas, métodos e ferramentas para a produção de artefatos – documentação, código fonte e demais elementos. Ainda que seja aplicado um processo para produção de artefatos de *software*, podem ocorrer erros, defeitos ou falhas no produto final. Os procedimentos de Garantia de Qualidade de *Software* são introduzidos ao longo de todo o processo de desenvolvimento, entre elas, as atividades de VV&T – Verificação, Validação e Teste com o objetivo de melhorar a qualidade dos artefatos produzidos e minimizar a ocorrência de erros e riscos associados (VINCENZI et al. 2016).

De acordo com Souza e Gasparotto (2013), na fase de teste é verificado se o comportamento do sistema está de acordo com o especificado nos requisitos funcionais - que são as funcionalidades do sistema - e não funcionais - que são os atributos de qualidade e restrições do sistema - que foram levantados junto ao cliente. O principal objetivo da realização do teste é reduzir a probabilidade de ocorrência de erros quando o sistema estiver em produção e evitar que os defeitos cheguem ao usuário final ou mesmo inviabilizem a continuidade do projeto.

Os testes funcionais segundo Resende, Santos e Neto (2010) são os mais comuns, entretanto, há outros testes que nem sempre são aplicados, mas que podem trazer um grande diferencial na utilização do produto sob diversas perspectivas no ambiente de produção, especialmente na *performance* da aplicação. O teste de *performance* contém alguns subtipos de testes, dentre estes, destacam-se os testes de desempenho, carga e estresse.

O teste de *performance* no geral tem por objetivo avaliar o comportamento do sistema submetido a uma determinada carga em um ambiente de teste específico. Ele fornece indicadores de desempenho que são utilizados para avaliar o quanto um sistema ou componente de um sistema é capaz de cumprir os requisitos de desempenho, tais como tempo de resposta ou a vazão de dados (*throughput)*, além de identificar os possíveis gargalos que podem estar degradando o desempenho do sistema (GUARIENTI et al. 2014). Visando uma qualidade total, a utilização deste teste deve estar em pauta durante a elaboração do planejamento de teste de um projeto de *software*.

Para realizar o teste de *performance* é recomendado o uso de alguma ferramenta para auxiliar os testes. Existem diversas ferramentas que podem auxiliar neste processo,

como o WebLoad, LoadRunner, JMeter, além de muitas outras. Segundo Fonseca (2012) um componente fundamental da área de teste de *software* para aplicações *web* é a utilização de boas ferramentas, pois elas irão otimizar o processo de teste de uma maneira extremamente eficiente, proporcionando resultados precisos e reais que irão influenciar diretamente no desempenho da aplicação.

Segundo Chaves (2019) algumas fábricas relutam em utilizar os testes de *performance* no desenvolvimento de seus produtos. A maioria delas se baseia em motivos como: a) falta de processo adequado e alta complexidade; b) no tempo gasto na elaboração das atividades de design, execução, análise dos resultados; c) a estratégia usada na simulação de ambientes extremos e o mais próximo da realidade que o sistema irá enfrentar em uso; d) a falta de documentação adequada sobre os requisitos não funcionais da aplicação.

Nesse contexto, qual pode ser um processo de testes de *performance* mínimo adequado para ambientes de fábricas de *software*?

Ruffato (2010) afirma que o processo de testes de *software* é basicamente construído dentro de quatro etapas: planejamento, projetos dos casos de testes, execução e por fim a avaliação dos resultados dos testes. Com isso, Coser (2012) argumenta 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 para garantir que o sistema esteja adequado para todos os tipos de usuários.

Essa pesquisa tem como objetivo central sistematizar um processo para a realização dos testes de *performance* em fábricas de *software*. Para tanto, os específicos são: i) Analisar os processos de teste utilizados em fábricas de *software*; ii) Identificar as necessidades em relação aos testes de *performance*; iii) Analisar cenários que aplicam os testes de *performance*; iv) Analisar os métodos e processos de teste do ambiente de experimento; v) Implementar o novo processo de teste em um ou mais projetos; vi) Avaliar os resultados da implementação deste processo.

Além de buscar os objetivos mencionados, outro motivo para elaboração desta pesquisa é a possibilidade de gerar conhecimento e colaborar para integrantes das áreas de teste e afins, com a formalização de um processo que muitas vezes não é abordado no desenvolvimento de um sistema de *software*. Consequentemente, quanto mais pesquisas

direcionadas para a área de teste, maior a contribuição para que fábricas de *software* continuem evoluindo seus projetos a níveis satisfatórios de qualidade.

Após esta seção introdutória, o trabalho subdivide-se em mais 5 seções. A Seção 2 apresenta a fundamentação teórica com as principais informações oriundas da leitura e revisão de textos, livros, revistas, artigos e outros, que serviram de orientação para a apresentação das informações coletadas para a pesquisa. São abordados conceitos fundamentais como a qualidade de *software* até mesmo principais ferramentas adotadas como JMeter. Na Seção 3, Método de Pesquisa, são abordadas informações sobre a estruturação da pesquisa, da intervenção aplicada e método de análise dos dados obtidos. Na Seção 4 a sistematização do processo e detalhes de sua aplicação em uma fábrica de *software*. Na Seção 5 são detalhados os resultados obtidos com a aplicação do processo. Por fim, as considerações apresentam as conclusões finais, dificuldades encontradas e trabalhos futuros.

2 Fundamentação Teórica

Nesta seção serão apresentados os conceitos fundamentais de qualidade de *software* relacionados ao trabalho, bem como ferramentas e abordagens que inspiram esta pesquisa. Na subseção 2.1 abaixo, será abordado conteúdo acerca da qualidade de *software* e sua importância no desenvolvimento de sistemas.

2.1 Qualidade de *software*

Desde o início do século XXI com a popularização da tecnologia, sistemas de *software* estão presentes em praticamente todas as coisas, desde celulares inteligentes até equipamentos médicos, entre outros. Com esta facilidade de acesso a novos sistemas, surgiu a necessidade cada vez maior de se preocupar com a qualidade durante o desenvolvimento de projetos de *software*.

Na engenharia de *software*, assim como em outras disciplinas de engenharia, é necessário considerar variáveis como esforço, produtividade, tempo e custo de desenvolvimento. Essas variáveis são afetadas negativamente quando artefatos defeituosos são produzidos, devido ao retrabalho para corrigir tais defeitos. Sabe-se, ainda, que o custo do retrabalho aumenta na medida em que o processo de desenvolvimento progride (KALINOWSKI e SPÍNOLA, 2008).

A qualidade de *software* é uma das áreas de conhecimento da engenharia de *software* e está presente em todo e qualquer produto de *software* desenvolvido. Pode ser definida como um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos e produtos (MELO, 2009). Ainda sobre a definição de qualidade, Chaves (2019) argumenta 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 à qualidade que se espera ao adquirir um produto.

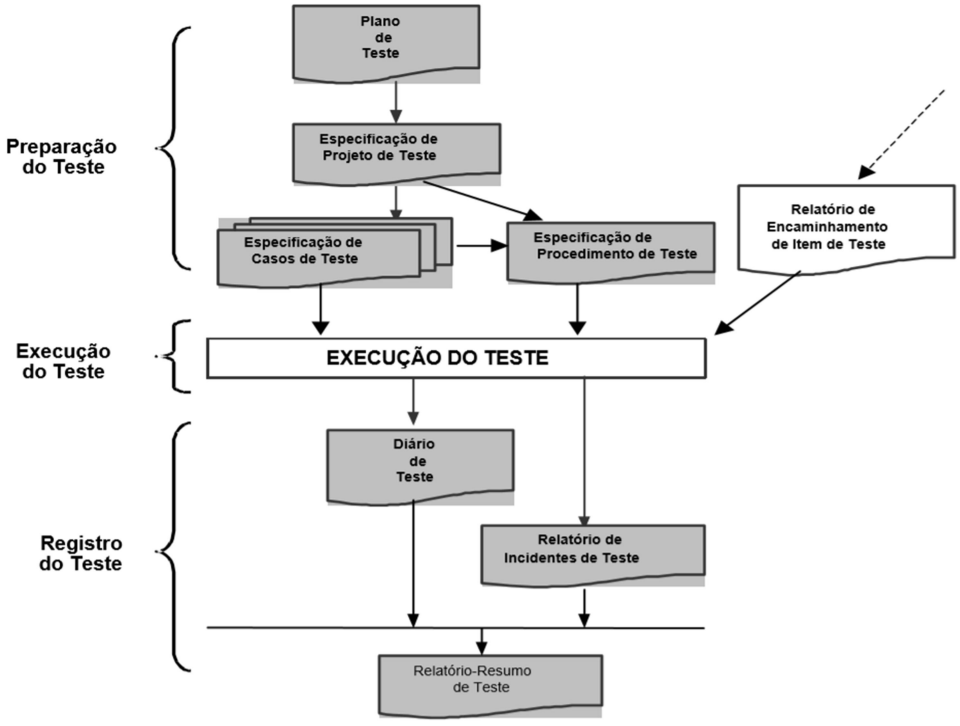
O teste de *software* é uma atividade crítica, porém, fundamental na garantia da qualidade do *software*. Os seres humanos são passíveis a erros, que podem ser encontrados desde a especificação do sistema até a entrega final do produto. Os testes podem ser escalonados em paralelo ou vinculados a todas as etapas do projeto, de acordo com metodologia de desenvolvimento escolhida (CÉRGOLI, 2017).

De acordo com Araújo e Pimenta (2013), o primeiro passo para a busca pela qualidade é a aceitação pela alta administração da empresa em investir em novos processos na área de testes de *software*. A escolha da metodologia a ser implementada vai de encontro com os objetivos estratégicos da empresa. Não importa qual a metodologia escolhida e aplicada, o importante é que a busca constante pela qualidade faça parte da cultura da organização. Na subseção 2.2 a seguir, serão abordados conceitos a respeito de processos de teste no geral, apresentando documentos básicos para estruturação do mesmo.

2.2 Processo de teste

A Norma IEEE 829 descreve um conjunto de documentos para as atividades do processo de teste de um produto de *software*. Os oito documentos definidos pela norma, que cobrem as tarefas de planejamento, especificação e relatórios de testes, são apresentados na Figura 1 (CRESPO et al. 2004):

Figura 1 – Relacionamento entre os documentos de um processo de teste.



Fonte: Crespo et al. (2004).

De acordo com Crespo et al. (2004), com estes documentos é possível registrar e gerenciar todo o processo de teste, desde o planejamento até a análise dos resultados, gerando possíveis métricas para melhorar a gestão e garantir que o produto tenha a

qualidade desejada. O processo de teste desenvolvido nesta pesquisa será adequado para atender fábricas de *software* que utilizam tanto a metodologia ágil quanto a tradicional, devido ser um processo de fácil adaptação ao ciclo de desenvolvimento da empresa.

Como mencionado em Guarienti et al. (2014) o desempenho de um sistema *web* é uma característica determinante para o seu sucesso. Fatores como a velocidade do tempo de resposta, a carga máxima de usuários suportados e a estabilidade durante diferentes cargas de acessos simultâneos são essenciais para o bom funcionamento da aplicação. Portanto, as empresas que não acompanham e testam o desempenho de seu sistema regularmente, tendem a ter perca e/ou insatisfação de seus usuários. Na Subseção 2.3 a seguir, são apresentados conceitos sobre testes de *performance* e a importância de sua aplicação.

2.3 Testes de desempenho (*performance*)

A *performance* é uma qualidade do *software* que é afetado desde suas camadas inferiores, como sistema operacional, *middleware*, *hardware* e redes de comunicação. Um sistema possui bom desempenho quando apresenta um tempo de resposta adequado e aceitável, mesmo se submetido a um volume de processamento próximo de situações reais ou de pico (FREITAS, 2013).

Diversas empresas já sofreram com a lentidão ou queda de seus sistemas quando há um grande número de usuários simultâneos utilizando a aplicação ou uma grande massa de dados no banco de dados, muitas vezes isto poderia ser evitado se a empresa realizasse os testes de *performance* corretamente, assim poderia se precaver e evitar o aborrecimento de seus clientes e usuários.

De acordo com Resende, Santos e Neto (2010) o cenário relacionado à execução de testes de *software* pode ser resumido da seguinte forma: boa parte das organizações realiza o teste funcional, uma vez que ele envolve conceitos mais difundidos, mas por outro lado, poucas organizações realizam os testes *performance*, embora seja notoriamente reconhecida sua importância, principalmente quando se trata de sistemas para *web*.

Santos e Neto (2008) afirmam que o teste de *software* é a verificação dinâmica do funcionamento de um programa utilizando um conjunto finito de casos de teste, devidamente escolhido dentro de um domínio de execução infinito, contra seu comportamento esperado. Para isto, necessita-se de um processo de teste adequado, que

gerencie todas as etapas desde a preparação do ambiente até a amostragem de resultados. Nesse cenário 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 e tempo;

• Comportamento esperado: para que um teste possa ser executado é necessário saber o comportamento esperado, para que ele possa ser comparado com o resultado obtido.

Como já mencionado, o teste de *performance* possui alguns subtipos de testes relacionados, como desempenho, carga e estresse. Neste trabalho será tratado apenas o teste de desempenho, no entanto, estudos futuros podem adaptar o processo para abranger os outros tipos de teste.

Segundo Santiago (2011), o teste de desempenho tem por objetivo específico verificar se o *software* cumpre as exigências específicas de desempenho, como por exemplo, capacidade de usuários simultâneos, tempo de resposta, vazão e confiabilidade sob uma dada carga.

Os testes de *performance* necessitam de determinados requisitos não funcionais previamente definidos para serem analisados com os resultados obtidos nos testes. Estes requisitos são discutidos por analistas e arquitetos ao iniciar o planejamento de um sistema e documentados, gerando artefatos como o documento de requisitos não funcionais e/ou documento de arquitetura do sistema.

Dada a sua importância, os testes de *performance* devem ser realizados por todos os tipos de empresas que prezem pela qualidade de seu sistema. O objetivo do teste pode variar de acordo com o a necessidade de cada empresa, como optar por mais velocidade no tempo de resposta das requisições ou uma carga maior de usuários simultâneos. Mesmo empresas de pequeno porte com sistemas simples podem realizar tais testes, buscando garantir confiabilidade e eficiência a seus projetos.

Conforme menciona Rocha (2008), apesar de ser de extrema utilidade e importância, a atividade de documentação no âmbito de um projeto de desenvolvimento de *software* é, não raras vezes, desleixada, conduzindo ao aumento dos custos e do tempo de

desenvolvimento do projeto devido ao retrabalho, originados por erros e omissões existentes na documentação. Isso pode ser evitado caso a empresa leve em consideração os requisitos não funcionais durante a fase de análise inicial do sistema.

Quando abordada de forma organizada e sistemática, esta atividade permite produzir artefatos que facilitam a melhoria, extensão e atualização do *software* a ser desenvolvido. Assim, os indivíduos relacionados com o projeto têm uma base comum de conhecimento e informação, garantindo a passagem de conhecimento no caso de entrada e saída de colaboradores (ROCHA, 2008).

Os requisitos não funcionais são de suma importância para a realização dos testes de *performance,* e variam de acordo com o objetivo que a empresa busca com o sistema. Caso a empresa não possua o documento de requisitos não funcionais, podem-se tomar as seguintes atitudes para o levantamento das informações:

a) Técnicas de estimativas como, por exemplo, a técnica PERT (*Program Evaluation and Review Technique*), que visa utilizar medidas de tempo otimista, pessimista e mais provável para a obtenção dos requisitos (GUARIENTI et al. 2014).
b) Análise de sistemas semelhantes, visando mapear as características de desempenho.
c) Utilizar requisitos de sistemas previamente desenvolvidos pela empresa, desde que os sistemas sejam similares.

Não existem requisitos que sejam universais e que possam ser utilizados no teste de *performance* de qualquer *software*, cada sistema é único e o seu desempenho pode variar de acordo com diversos fatores, desde a forma como é codificado até a escolha do servidor utilizado (MANZOR, 2012).

Para o desenvolvimento de *software web* em que os usuários podem realizar múltiplas tarefas, pode-se estimar que o tempo de resposta seja de no máximo 2 segundos, mas isto pode variar conforme as situações mencionadas acima. Existem outros requisitos importantes que devem ser levados em consideração, como por exemplo, o máximo de transações por segundo, o espaço utilizado durante a execução do sistema e a porcentagem de erros durante a requisição de dados.

Alguns trabalhos que já abordaram assuntos relacionados ao teste de *performance*, como disponível em Chaves (2019) e Resende, Santos e Neto (2010), apenas

demonstraram dados dos testes realizados e os procedimentos para realizar o teste, porém, não são uma melhoria para o processo da equipe de fato. Guarienti et al. (2014) apresentaram um processo de abordagem baseado em modelos, cujo propósito é analisar unicamente o tempo de resposta para aplicações *web* utilizando MBT (*Model Based Testing*) para gerar artefatos de teste, no entanto, não se estende aos demais subtipos de teste de *performance*, como teste de carga e estresse.

Dentre as ferramentas que podem ser aplicadas para esta finalidade, uma opção completa que pode cobrir uma ampla área do teste de *performance* é o JMeter, ferramenta *open source* utilizada por grandes empresas, que permite a criação de testes para vários protocolos (HTTP, JDBC, FTP, SOAP, entre outros) além de permitir a obtenção de diversos tipos de dados com a realização dos testes, como o tempo de resposta e a porcentagem de erro de requisições (FONSECA et al. 2012).

As subseções 2.4 e 2.5 a seguir, descreve os princípios básicos das ferramentas utilizadas neste trabalho, são elas, JMeter e BlazeMeter.

2.4 Ferramenta JMeter

Conforme apresenta Fonseca et al. (2012) o JMeter é uma aplicação *desktop*, feita em Java, projetada para a realização de testes de desempenho em aplicações cliente/servidor, tais como aplicações *web*. A ferramenta funciona basicamente simulando um grupo de usuários enviando solicitações para um servidor de destino e retorna estatísticas que mostram o desempenho do servidor/aplicativo de destino através de tabelas, gráficos, etc (TUTORIALSPOINT, 2015).

Ainda segundo Fonseca et al. (2012) os testes podem ser escritos manualmente, por meio da criação manual das requisições HTTP, ou gravados automaticamente enquanto o usuário navega na aplicação simulando um usuário real (por meio de um *Proxy Server*), também existem programas como o BadBoy ou o BlazeMeter, que podem gerar *scripts* de teste e exportá-los para o JMeter.

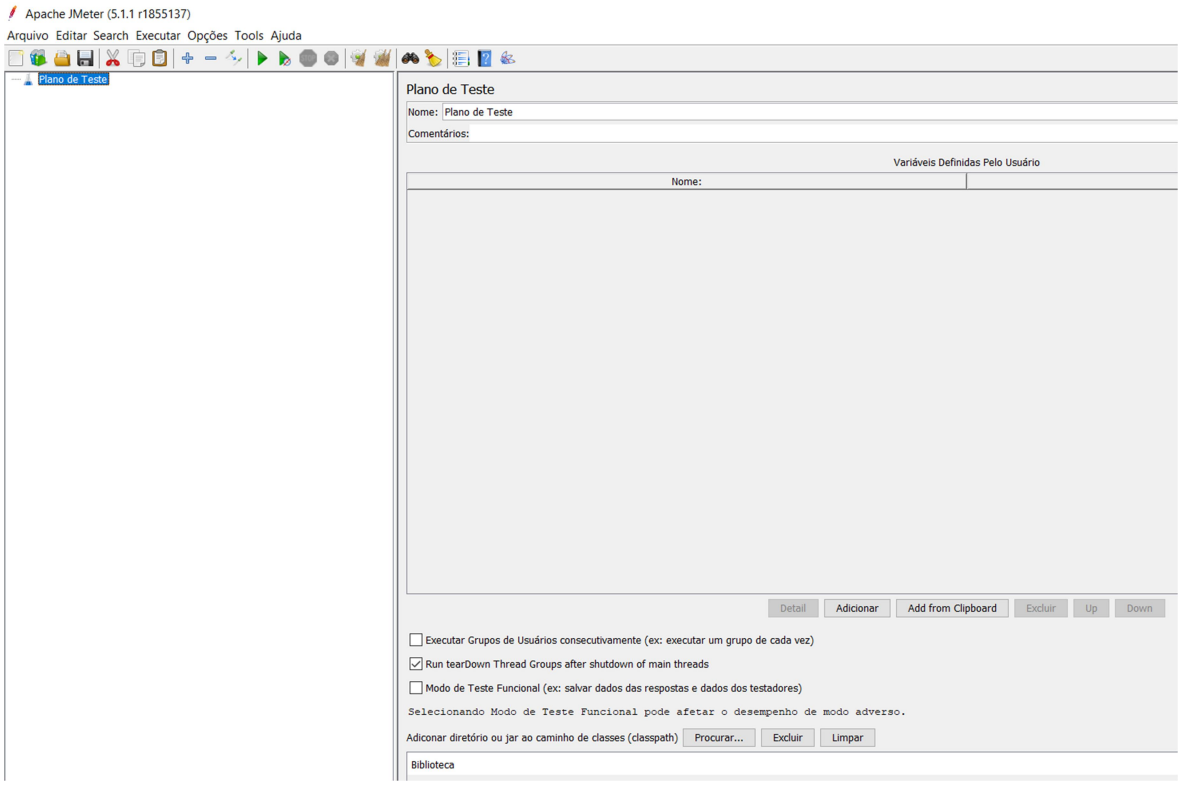
Santos e Neto (2008, p. 03), discorrem sobre as funções do JMeter com a seguinte afirmação:

> No JMeter, a organização dos elementos que compõe o teste é feita através de uma árvore hierárquica, cuja raiz é o plano de teste (TestPlan). Alguns elementos da árvore de teste são hierárquicos, como os ouvintes (Listeners) e as asserções (Assertions), pertencendo e referenciando a outros elementos da ordem hierárquica superior. Outros elementos, como as requisições (Samplers), são

> primariamente ordenados e, portanto, a ordem na qual aparecem verticalmente na árvore determina sua ordem de execução. Essa organização também é refletida no arquivo XML gerado pela ferramenta para a persistência dos elementos.

A Figura 2 abaixo apresenta a interface de início do JMeter, com o plano de teste em primeiro plano.

Figura 2 – Interface inicial do JMeter.



Fonte: Elaborado pelos autores.

Esta ferramenta foi primeiramente utilizada para realizar testes em aplicações *web*, mas tem expandido suas funcionalidades, podendo realizar testes funcionais, testes em bancos de dados entre outros (PAPPE, 2013). Outra ferramenta que pode ser utilizada para complementar a análise dos resultados obtidos com o JMeter é o GTmetrix, cujo objetivo é verificar a velocidade de carregamento de páginas *web*, além de fornecer dados e notas sobre a *performance* da aplicação.

2.5 Ferramenta BlazeMeter

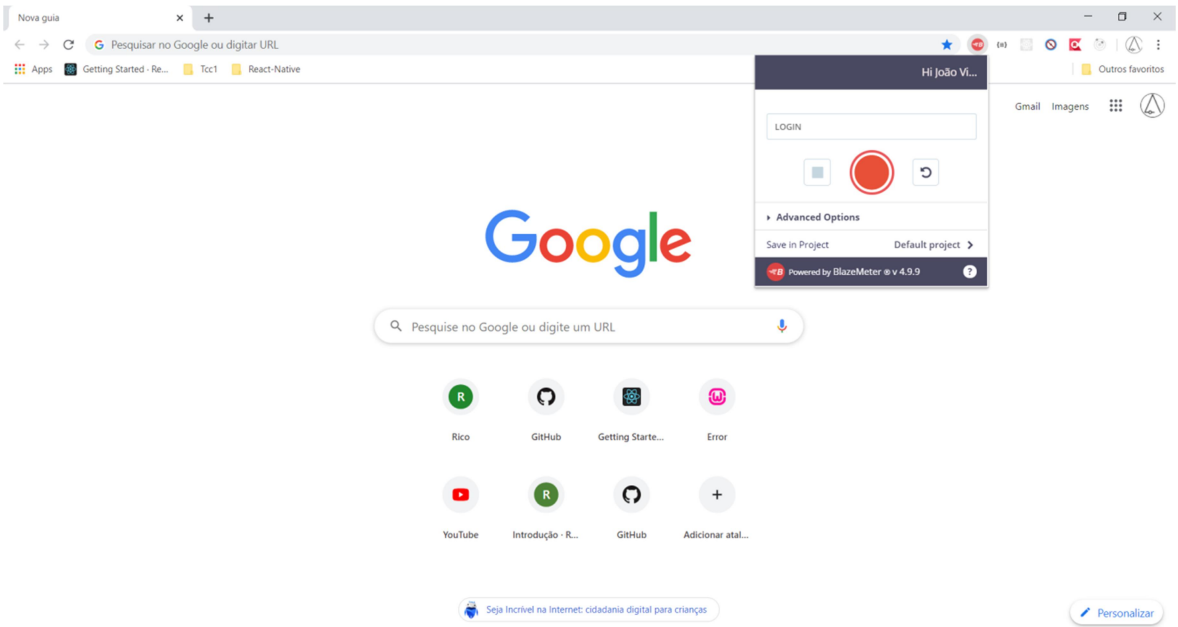
BlazeMeter é uma plataforma de testes contínuos, que fornece uma maneira rápida e fácil de encontrar e corrigir gargalos de desempenho em todas as etapas do ciclo de entrega de *software*. Possui uma extensão no programa de buscas Google Chrome que

permite a gravação dos testes através do próprio navegador, possibilitando que o mesmo cenário seja gravado e exportado tanto para a ferramenta Selenium - ferramenta utilizada para automação de testes - quanto para o JMeter em uma única sessão sincronizada.

Esta ferramenta permite a criação de testes sem a necessidade de utilizar *scripts* complexos e processos manuais. Algumas das funcionalidades são o *drag-and-drop* para facilitar o uso, a automatização de *scripts* de testes de *performance* de *websites*, APIs (*Application Programming Interface*) e GUI (*Graphical User Interface*) e a integração com ferramentas como JMeter, Jenkins, Selenium e WireMock. Desta forma, reduzindo a manutenção para gerenciar componentes e objetos reutilizáveis, permitindo gravar e capturar objetos de teste em tempo real sem a necessidade de configuração (BLAZEMETER, 2020).

A Figura 3 abaixo, representa a interface inicial da extensão do BlazeMeter para o Google Chrome. Nesta extensão é realizada a captura dos passos de teste em tempo real e o *script* gerado através desta captura, poderá ser exportado para utilização na ferramenta JMeter.

Figura 3 – Interface inicial da extensão do BlazeMeter.



Fonte: Elaborado pelos autores.

A subseção 2.6 a seguir apresenta informações acerca de fábricas de *software*, seu funcionamento e principais metodologias utilizadas.

2.6 Fábricas de *software*

De acordo com Lutz (2017) as fábricas de *software* tem como foco o desenvolvimento de sistemas específicos para clientes, ou seja, contratações pontuais que acabam no momento da entrega do *software*. O paradigma de fábrica de *software* busca a obtenção de qualidade e produtividade no desenvolvimento e manutenção de *software* por meio de padronização de processos, reúso de artefatos e controle do sistema de produção (SOUSA, 2018).

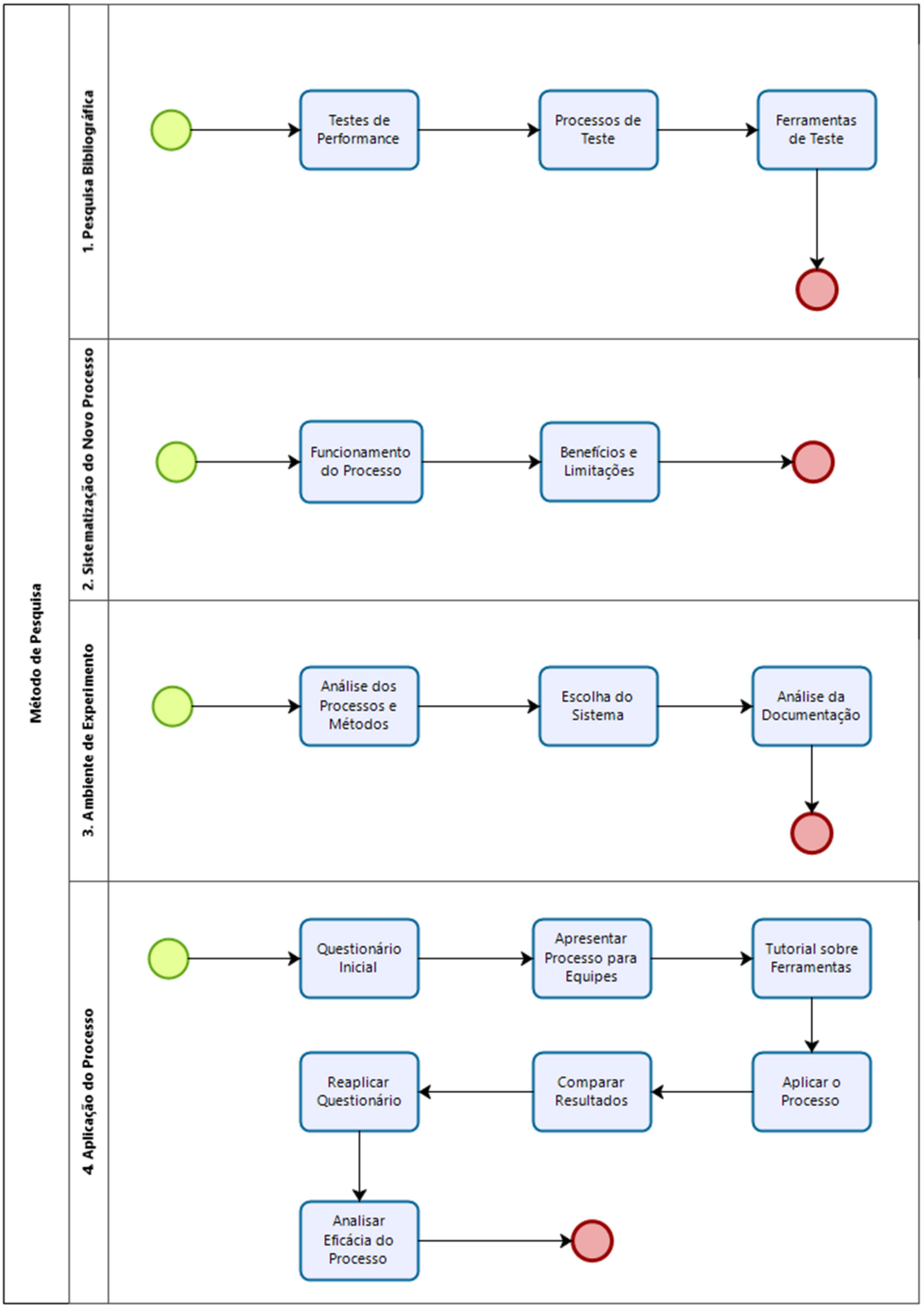
Os principais modelos de desenvolvimento de *software* conhecidos podem ser categorizados em dois grupos: tradicionais e ágeis. O modelo tradicional é conceitualmente mais antigo e caracteriza-se por ter um processo sequencial, que contempla um esforço maior de análise no início do projeto, seguido por 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 (LUTZ, 2017).

Segundo Lutz (2017) 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 propõe 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. Neste sentido, a seção a seguir detalha a estrutura de método de pesquisa definido para que seja possível alcançar os objetivos específicos determinados.

3 Método de Pesquisa

Com o intuito de responder à pergunta de pesquisa e estruturar uma estratégia para atingir os objetivos, este trabalho se trata de uma pesquisa exploratória que tem como finalidade aprofundar o conhecimento já adquirido em testes de *software* durante outros trabalhos com a sistematização de um processo para testes de *performance*, através de pesquisas bibliográficas e cursos explicativos, e avaliar como este processo pode impactar em uma fábrica de *software*. A Figura 4 abaixo representa a divisão das etapas do método de pesquisa:

Figura 4 – Etapas da Pesquisa.



Fonte: Elaborado pelos autores.

O método de pesquisa deste trabalho foi organizado em quatro etapas, sendo:

a) Pesquisa bibliográfica em artigos, livros, cursos em meio eletrônico e revistas de reconhecimento científico entre outras fontes, a respeito de testes de *performance*. Também acerca dos modelos de processos de teste, bem como sobre ferramentas disponíveis e suas funcionalidades.

b) Sistematização do processo com base no conhecimento adquirido nas pesquisas. Detalhes sobre o seu funcionamento, benefícios e limitações.

c) Escolha do ambiente de experimento do novo processo. Após a escolha do ambiente, serão analisados seus processos e métodos de teste, assim como a documentação do sistema escolhido.

d) Aplicação do novo processo. Após o prévio estudo do sistema, será elaborado um questionário para avaliar informações sobre o ambiente de experimento e as equipes responsáveis, posteriormente será apresentado o processo para as equipes e tutoriais sobre as ferramentas que serão utilizadas para funcionamento correto do modelo. Após a realização dos testes por parte da equipe, os resultados serão analisados de forma quantitativa, visando obter possíveis métricas, como tempo de resposta e porcentagem de erros de requisições, que justifiquem a eficácia e necessidade da utilização de testes de *performance*.

Subsequentemente haverá a reaplicação do questionário para determinar a eficiência desta pesquisa na garantia de qualidade do sistema e dos processos gerais da equipe de teste. O sucesso deste processo se dará quando os testes realizados indicarem se a *performance* do sistema está de acordo com as métricas esperadas na documentação e se houver aumento da eficácia na realização dos testes através deste novo processo.

4 Abordagem Proposta

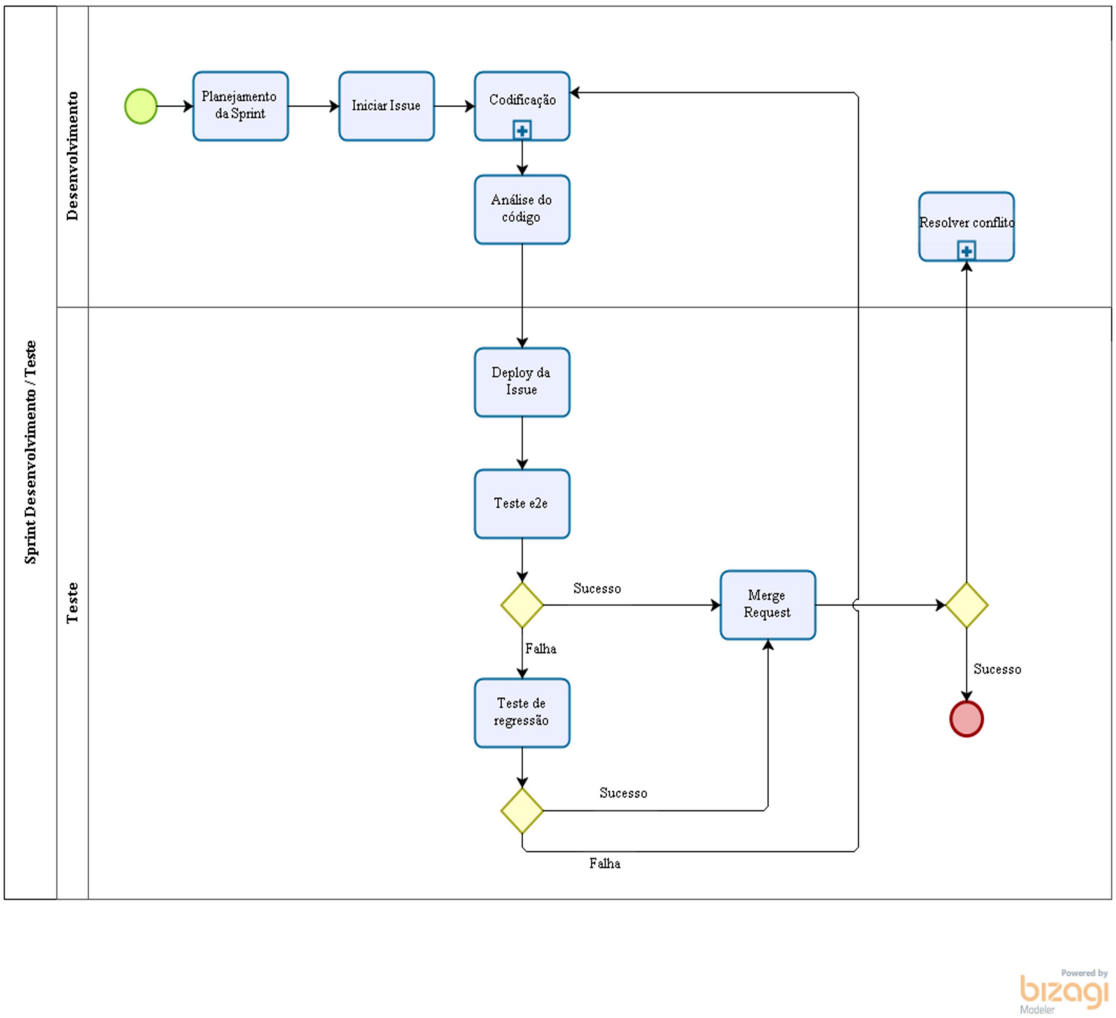
Após conhecer o estado da arte da literatura do assunto abordado na seção 2, esta seção aplica as estratégias descritas no método de pesquisa da seção 3. Para atingir os objetivos definidos, a subseção 4.1 apresenta o ambiente de experimento. Em seguida, na subseção 4.2 descreve a análise e estruturação do processo de teste proposto. Logo após, serão detalhadas as abordagens nas ferramentas JMeter e BlazeMeter, e por fim, será apresentado o projeto escolhido para a aplicação do processo.

4.1 Fábrica de Tecnologias Turing (FTT)

O ambiente selecionado para a experimentação é a Fábrica de Tecnologias Turing (FTT), uma fábrica de desenvolvimento de *software* do Centro Universitário de Anápolis – UniEvangélica, que é vinculada aos Cursos Bacharelados de Computação. É um ambiente onde os acadêmicos podem vivenciar experiências de desenvolvimento de projetos reais com o acompanhamento do corpo docente dos cursos. A FTT foi escolhida devido aos seus projetos *web* que necessitam ter uma *performance* adequada, e por possuir uma equipe de teste familiarizada com o teste de *performance*, assim facilitando o entendimento do novo processo e da ferramenta utilizada.

Por ser uma fábrica de *software* que possui alta rotatividade e constantemente novos membros com pouca ou nenhuma experiência, reforça ainda mais a necessidade de formalização de um processo estruturado e bem definido.

Figura 5 – Processo de desenvolvimento/teste da FTT.



Fonte: Equipe de Teste da FTT (2018).

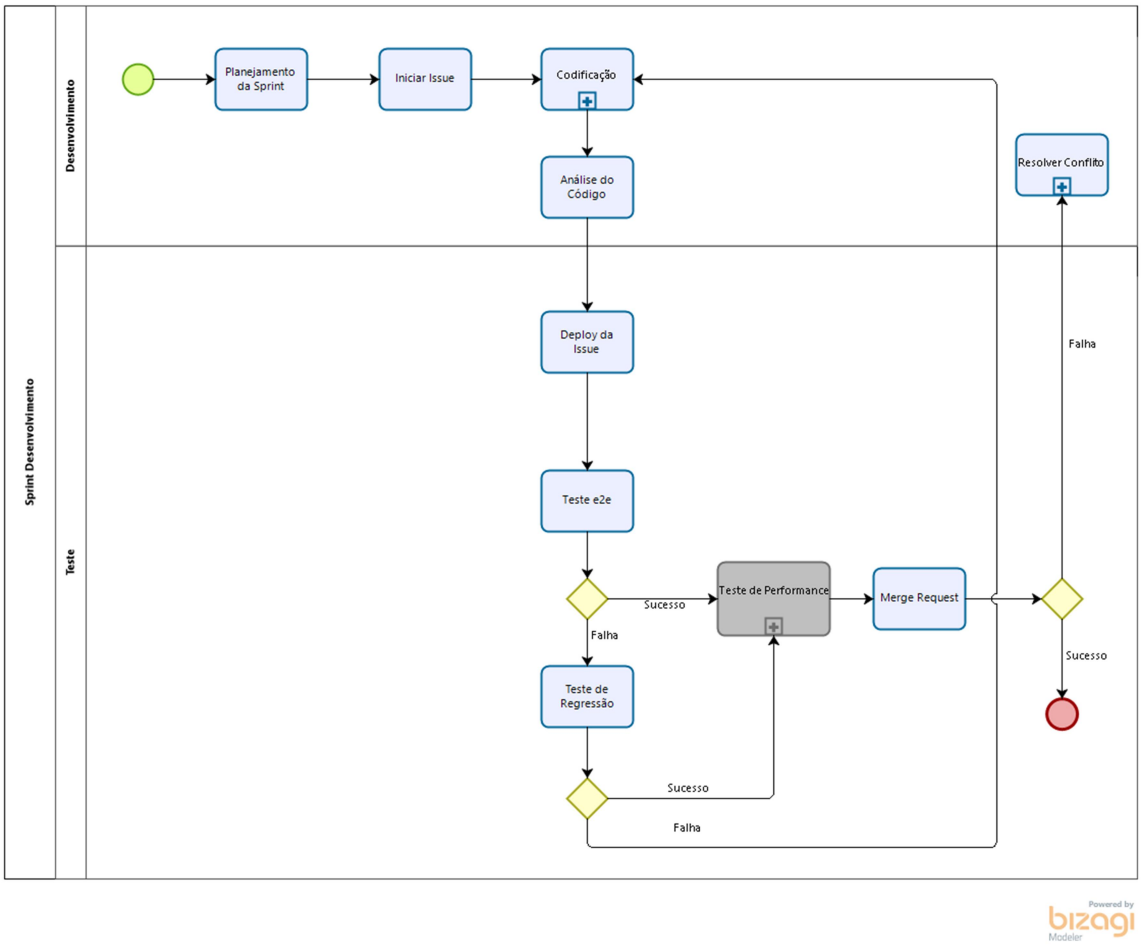
O atual processo de testes da FTT apresentado na Figura 5 acima, envolve a utilização de testes ponta a ponta e testes de regressão, testando assim a funcionalidade em si do sistema. Contudo, o processo não engloba os testes de *performance*, portanto os sistemas desenvolvidos estão suscetíveis aos defeitos e falhas de desempenho. Sendo assim, além de garantir a melhoria do processo de teste na FTT e na qualidade geral do sistema, um processo com a inclusão dos testes de *performance* também capacitará os membros da equipe a trabalharem com a ferramenta JMeter, que de acordo com Apache (2019) é utilizada e/ou patrocinada por diversas empresas como Google, Microsoft, Facebook, entre outras.

Para a aplicação do novo processo foi levado em consideração um estudo sobre os atuais processos e métodos de teste utilizados pela equipe de teste, também da documentação do sistema, como documento de requisitos não funcionais e documento de

visão, buscando facilitar o entendimento das funcionalidades do projeto escolhido para experimento.

Com base nos objetivos propostos, foi iniciada a execução da metodologia através da qual, verificou-se possível adaptar o processo de teste da FTT para incluir a nova etapa, conforme destacado em cinza na Figura 6, abaixo.

Figura 6 – Processo adaptado de desenvolvimento/teste da FTT.



Fonte: Elaborado pelos autores.

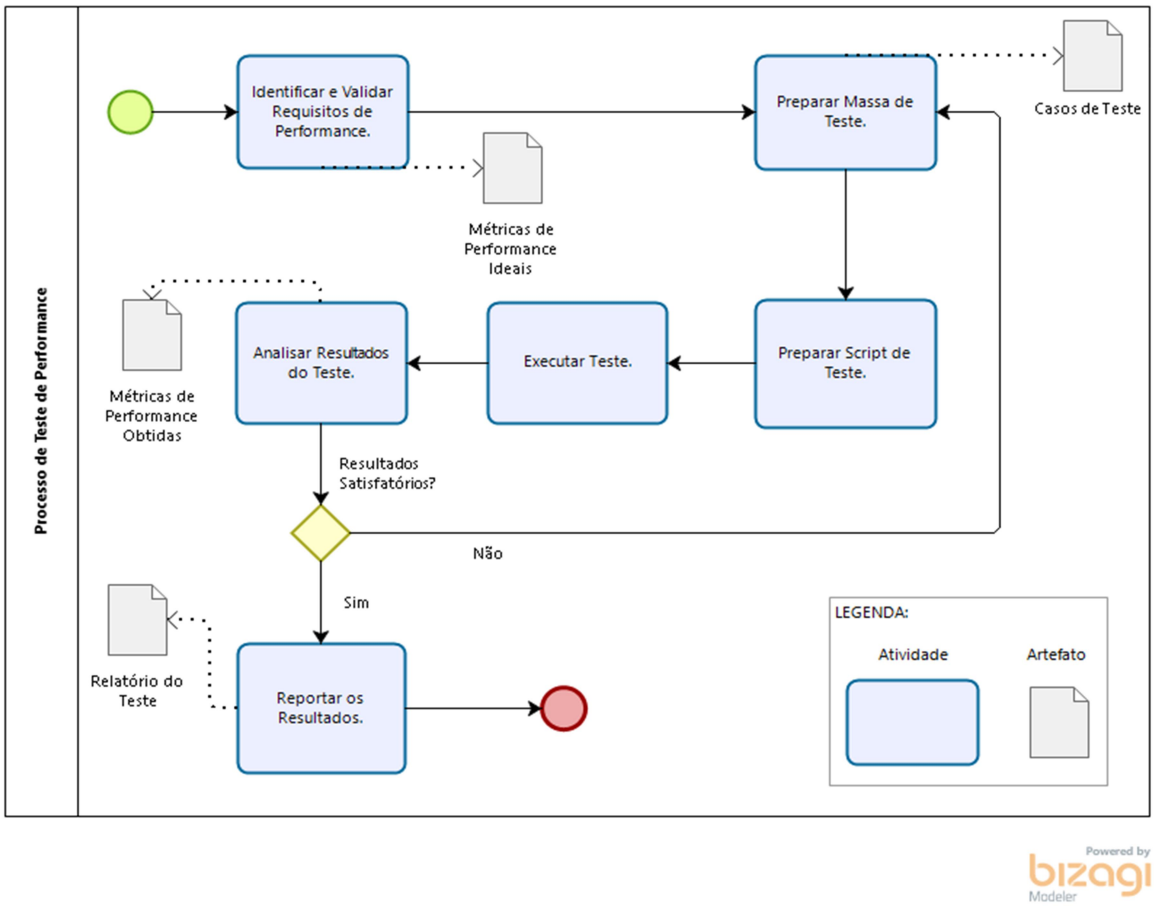
O novo processo de teste de *performance* ocorrerá após o teste funcional e2e (“end to end”, ponta a ponta na tradução livre) ou quando o teste de regressão estiver sido realizado com sucesso. Portanto, o requisito desenvolvido estará qualificado para a etapa dos testes de *performance*, e assim que estes estiverem completos, poderão ser adicionados ao restante do sistema com o *merge request*. Desta forma, o processo garante que o requisito testado esteja com a *performance* mínima aceitável, para o bom funcionamento do sistema como um todo.

A FTT visa um ambiente de constantes melhorias, considerado fundamental o desenvolvimento empírico em seus projetos. Utilizando de tecnologias atuais no mercado como Métodos Ágeis. Por isto é relativamente simples adaptar as equipes a novos processos e ferramentas, como o proposto nesta pesquisa. Esta pesquisa é útil para fábricas de *software* no geral e também para a FTT, pois o novo processo sistematizado será utilizado por equipes de teste, a fim de garantir uma melhor qualidade na *performance* geral dos projetos desenvolvidos.

4.2 Processo de teste

O resultado da pesquisa bibliográfica e da análise de diversos processos de teste foi este apresentado na Figura 7 abaixo.

Figura 7 – Processo de teste de *performance*.



Fonte: Elaborado pelos autores.

Primeiramente, antes de iniciar o processo em si, é necessário identificar o ambiente que será realizado os testes e definir uma ferramenta adequada para a situação. A

identificação do ambiente é útil para evitar imprecisão com os resultados obtidos, caso os testes sejam realizados em máquinas com diferentes configurações.

O processo se divide em seis etapas principais:

a) Identificar e Validar Requisitos de *Performance*. Nesta etapa são identificados e validados os documentos de requisitos não funcionais, visando obter as métricas para realização do teste de *performance.* O analista deve validar as métricas contidas no documento de requisitos não funcionais e caso possuam discrepâncias com a realidade do sistema, encaminhar para correção da mesma.

b) Preparar Massa de Teste. O planejamento da massa de teste deve conter todos os casos de teste do requisito a ser testado, visando especificar o objetivo de cada tipo de teste realizado em cada funcionalidade do sistema. Também deve envolver as métricas obtidas na etapa anterior, com a utilização de tabelas ou *checklist*.

c) Preparar *Script* de Teste. Nesta etapa será realizado o teste na ferramenta BlazeMeter utilizando os casos de teste preparados na etapa anterior, cujo objetivo é buscar a praticidade da ferramenta, realizando o teste de forma rápida e eficaz, facilitando a criação do *script* para ser utilizado na próxima etapa.

d) Executar os Testes. Após as devidas configurações básicas, como grupo de usuários e tipos de amostragem de resultados, o *script* gerado na etapa anterior deverá ser importado e executado na ferramenta JMeter. Apenas um caso de teste deve ser executado por vez, visando não prejudicar os resultados.

e) Analisar os Resultados do Teste. Nesta etapa serão analisados os resultados obtidos. Para isto, a equipe deve possuir um *checklist* ou uma tabela para validar as métricas alcançadas com os testes. Caso tenha ocorrido algum erro ou os resultados estiverem discrepantes com o documento de requisitos, os testes devem ser planejados novamente.

f) Reportar os Resultados. Caso a comparação das métricas obtidas no teste com os resultados esperados estiverem aceitáveis, o analista responsável deve reportar os resultados através de um documento contendo anexos do teste realizado e das métricas obtidas.

Este processo descreve as principais etapas para realização do teste de *performance* em uma fábrica de *software*. Os resultados alcançados dependem do grau de qualidade em que os testes foram planejados e executados, portanto, a equipe responsável deve estar

preparada para realizar os testes e a ferramenta escolhida deve atender bem ao seu propósito.

O processo é adaptável, ou seja, de acordo com a necessidade da empresa pode-se modificar ou acrescentar etapas, visando garantir a qualidade ao sistema que será testado. O processo visa ser o mínimo viável para sistemas *web*, para avaliar critérios comuns de desempenho da aplicação. Portanto, para que os requisitos não funcionais sejam testados completamente, pode ser necessária a utilização de outros processos em conjunto.

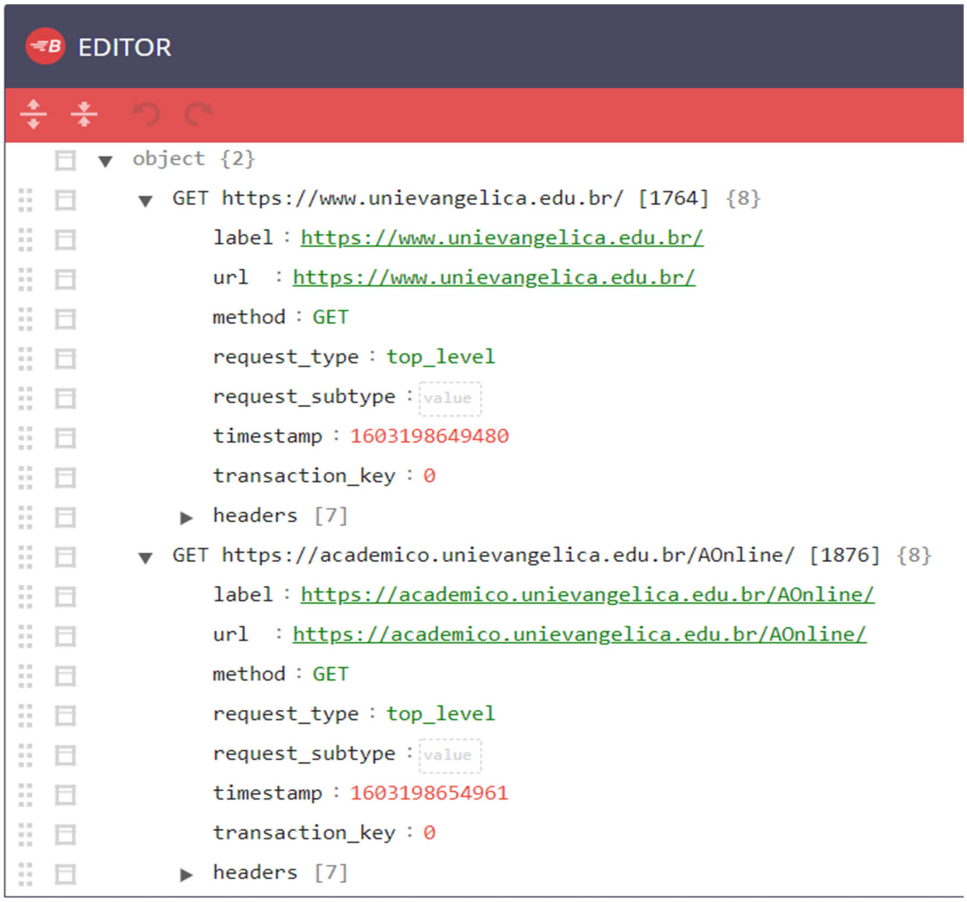
Segundo Souza (2018), o teste de desempenho mede e avalia o grau em que um item de teste desempenha suas funções designadas dentro de determinados limites de tempo. Dessa forma, pretende-se propor uma solução que vá de encontro às necessidades das empresas e, ao mesmo tempo, evite as motivações mencionadas pelas empresas para não aplicar esta etapa de teste. Portanto, uma solução adaptável que possa ser utilizada em conjunto com outras metodologias já aplicadas no desenvolvimento de *software*.

No entanto, antes da aplicação do processo, foi realizado um questionário inicial com os membros da equipe de teste, *scrum master* e *product owner*, almejando obter informações sobre a eficácia dos atuais modelos de testes da FTT e também questões técnicas a respeito do teste de *performance* (Apêndice A). Da mesma forma, foi realizado outro questionário após a aplicação do processo (Apêndice B), visando obter dados concretos a respeito da intervenção proposta nesta pesquisa.

4.3 Abordagem BlazeMeter

Conforme definido nas etapas do processo, os analistas aplicaram os casos de teste com a ferramenta BlazeMeter para criação dos *scripts* de forma automatizada, posteriormente utilizaram a ferramenta JMeter para executar os testes e obter os resultados. Na Figura 8 abaixo, é apresentado um exemplo de *script* cujo formato é o mesmo que foi utilizado pelos integrantes da equipe de teste na execução do processo.

Figura 8 – Interface de exportação de *script* do BlazeMeter.



Fonte: Elaborado pelos autores.

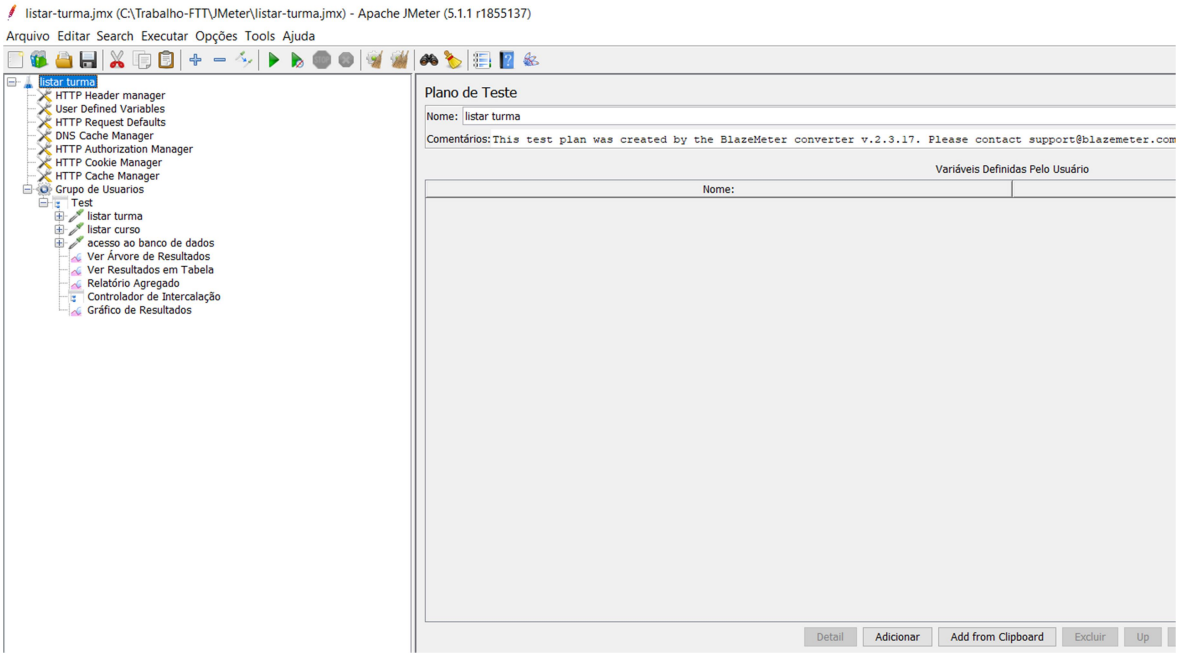
O BlazeMeter também possui uma versão *web* que possibilita a realização de testes de *performance*, no entanto, neste trabalho será utilizado o JMeter para este quesito por possuir uma gama maior de funcionalidades. Com isso, após a importação do *script*, será criado um plano de teste conforme destacado na Figura 12 da subseção 4.3 a seguir. Este plano de teste contém informações e funcionalidades para a aplicação do processo.

4.4 Abordagem JMeter

Com a obtenção do *script* de forma automatizada com o BlazeMeter, é necessário importar e configurar o mesmo, de forma que consiga suprir todas as necessidades do teste escolhido. A importação é realizada de forma simples pelo JMeter, com isso, toda a configuração é feita de forma automática com a criação do plano de testes. A Figura 9 abaixo demonstra o plano de teste configurado para realização do teste de desempenho.

Pode-se notar que existe um plano para o *script* gerado, ou seja, este plano inclui todas as configurações globais, grupo de usuários, requisições e ouvintes.

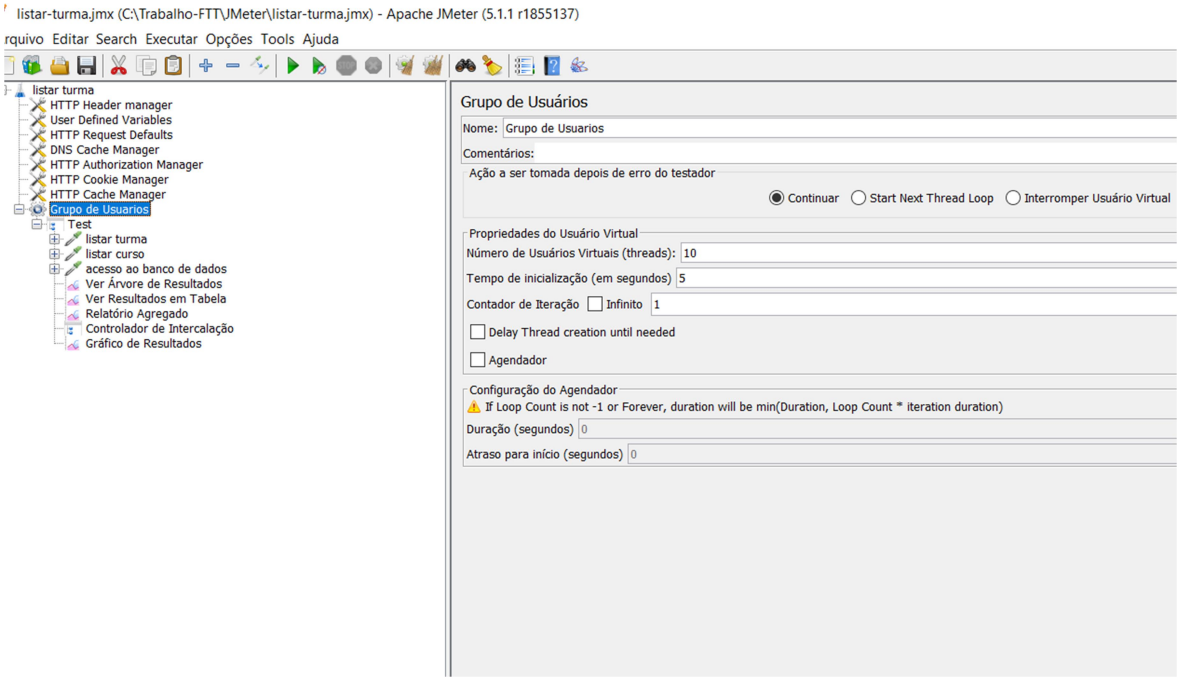
Figura 9 – Estrutura do plano de teste.



Fonte: Elaborado pelos autores.

Segundo Chaves (2019), o grupo de usuários representa o número de usuários virtuais que executa uma ação no sistema, já o tempo de inicialização representa o tempo que o JMeter leva para iniciar cada novo usuário. A Tabela 1 descrita no final desta subseção informa a quantidade de usuários virtuais simultâneos e o tempo de inicialização para cada teste que será realizado. A Figura 10 a seguir demonstra um exemplo de grupo de usuários com dez usuários simultâneos, justamente um dos cenários escolhidos para esta pesquisa.

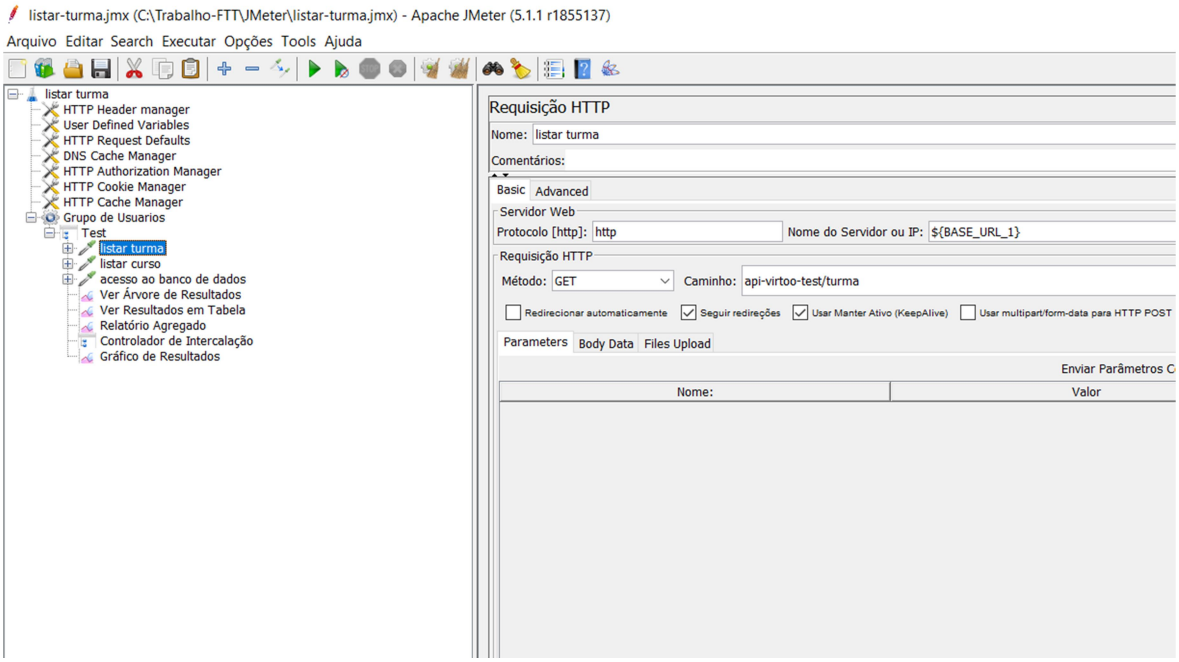
Figura 10 – Grupo de usuários.



Fonte: Elaborado pelos autores.

A requisição HTTP permite o envio de requisições HTTP/HTTPS ou arquivos para um servidor *web,* também permite a adição de parâmetros como o tipo de protocolo, tipo de método e nome do servidor ou IP. Com a utilização do BlazeMeter, todas as configurações de requisições são feitas de forma automatizada, proporcionando uma maior eficiência ao realizar o teste. Na Figura 11 tem-se um exemplo de uma requisição adicionada a um plano de teste.

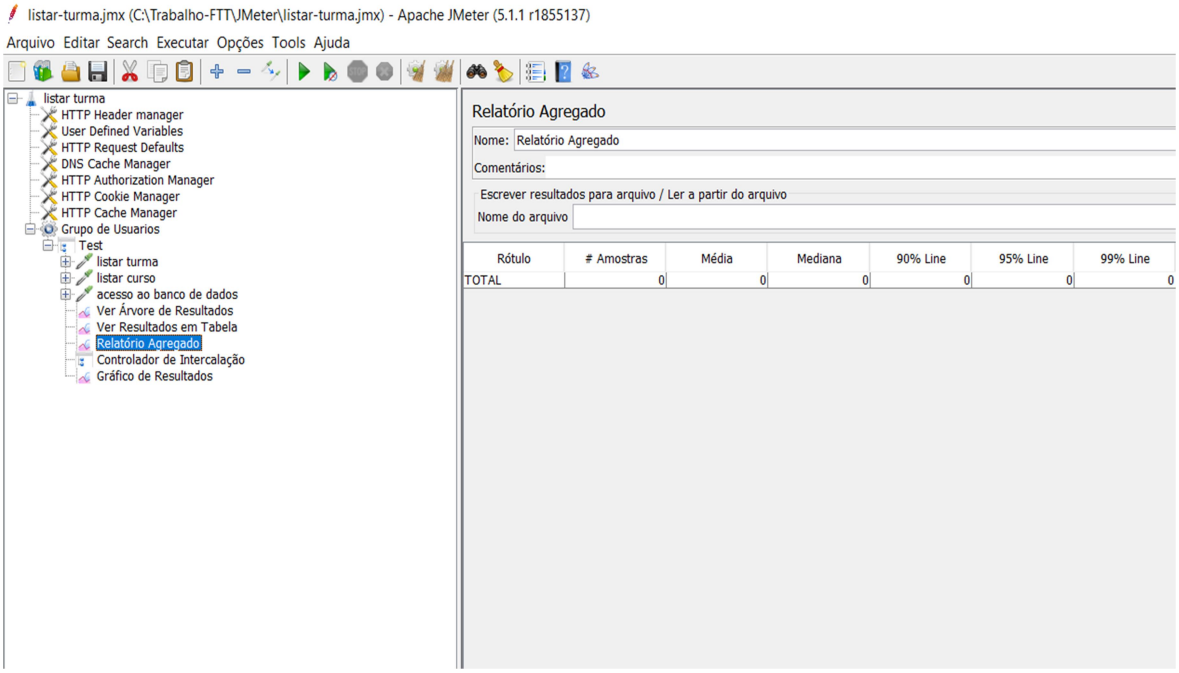
Figura 11 – Requisição HTTP.



Fonte: Elaborado pelos autores.

Para visualização dos resultados dos testes, são utilizados os ouvintes, que são responsáveis por capturar as informações durante a execução dos testes e assim criar gráficos e relatórios, possibilitam também a geração de arquivos para exportar os resultados, conforme destacado na Figura 12 abaixo. Nesta pesquisa serão utilizados os ouvintes: a) Ver árvore de resultados, b) Relatório agregado, c) Ver resultados em tabela, pelo fato de apresentarem dados de forma simplificada, facilitando o entendimento do analista responsável pelo teste.

Figura 12 – Visualização de resultados.



Fonte: Elaborado pelos autores.

A ferramenta JMeter foi configurada de forma que possibilite a realização de diferentes cenários de teste, os números apresentados na Tabela 1 abaixo, foram implementados na função de grupo de usuários, determinando o número de vezes que os usuários devem enviar requisições ao servidor e o tempo de inicialização entre cada usuário.

Tabela 1 – Relação entre usuários simultâneos e tempo de inicialização.

Número de Usuários Virtuais Simultâneos	**Tempo de Inicialização em Segundos**
1	1
10	5
30	15

Fonte: Elaborado pelos autores.

O padrão apresentado na tabela acima permite a realização do teste de maneiras distintas, viabilizando a obtenção de dados diferenciados, desta forma os analistas podem investigar em qual contexto o sistema está com o desempenho mais deficitado. A relação número de usuários x tempo de inicialização foi projetado com base no padrão apresentado por Santos e Neto (2008) por se tratar de um teste com poucos usuários. A ferramenta

JMeter possibilita analisar diversos tipos de dados, no entanto, o foco desta pesquisa é apresentar a importância do teste de *performance* juntamente com a utilização de um processo definido, com a concentração em comparar a média do tempo de resposta das requisições HTTP utilizadas nos testes.

4.5 Projeto de experimento

Após análises dos projetos em andamento da FTT, o sistema escolhido para esta pesquisa foi o projeto VIRTOO - Sistema de Gerenciamento Acadêmico e Financeiro para as faculdades angolanas Instituto Superior Politécnico Vida (ISPEL) e Instituto Superior de Teologia Evangélica no Lubango (ISTEL). Este projeto foi escolhido devido estar em uma fase mais avançada de desenvolvimento, possibilitando a realização dos testes de *performance* com maior eficiência.

Os requisitos selecionados para aplicar o processo foram Manter Disciplina e Manter Turma, desenvolvidos no projeto Virtoo. Para este experimento, as funcionalidades selecionadas foram as de listagem de ambos os requisitos no ambiente de homologação disponibilizado, por possuírem características parecidas, assim facilitando a comparação e entendimento dos resultados. No entanto, não foram encontrados dados acerca dos requisitos não funcionais do projeto Virtoo, esta característica é justamente uma das mais comuns em diversos projetos de *software,* com isso proporciona um grau maior de dificuldade na comparação dos resultados dos testes.

O experimento foi realizado por cinco integrantes da equipe de teste da FTT. Estão descritos na Tabela 2 abaixo as especificações utilizadas durante os testes:

Tabela 2 – Especificações dos analistas participantes.

Analista	**Processador**	**Memória RAM (em gigabytes)**	**Velocidade de internet (em megabytes)**
Analista 1	Intel® Core™ i5-7600	24GB	50MB
Analista 2	Intel® Core™ i3-7020	8GB	20MB
Analista 3	Processador: Intel®	8GB	3MB

	Core™ i7-7700		
Analista 4	Intel® Core™ i5-3210	8GB	20MB
Analista 5	Intel® Core™ i5-7600	8GB	10MB

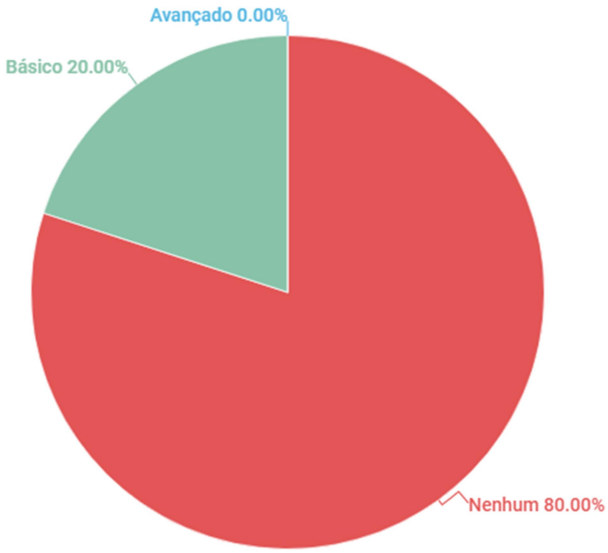
Fonte: Elaborado pelos autores.

Com base em informações coletadas com a ajuda de membros das equipes de *product owner* e *scrum master*, levantou-se os dados abaixo sobre a possível configuração do cenário do cliente final do projeto Virtoo, contudo, estas especificações são de máquinas utilizadas dentro das instituições acadêmicas parceiras da FTT.

- Processador: Intel® Core™ i3 ou Intel® Core™ i5;
- Memória RAM: 4GB;
- Velocidade de Internet: 20MB (toda a instituição compartilha desta velocidade).

Os gráficos das figuras abaixo apresentam informações acerca dos cinco membros da equipe de teste, coletados com a realização do questionário inicial e com o workshop antes da aplicação do processo. Estas informações são importantes para avaliar o nível de conhecimento dos participantes a respeito de processos e ferramentas de testes, visando levantar dados para futuras comparações de resultados. A Figura 13 a seguir apresenta dados acerca do conhecimento dos integrantes sobre teste de *performance*.

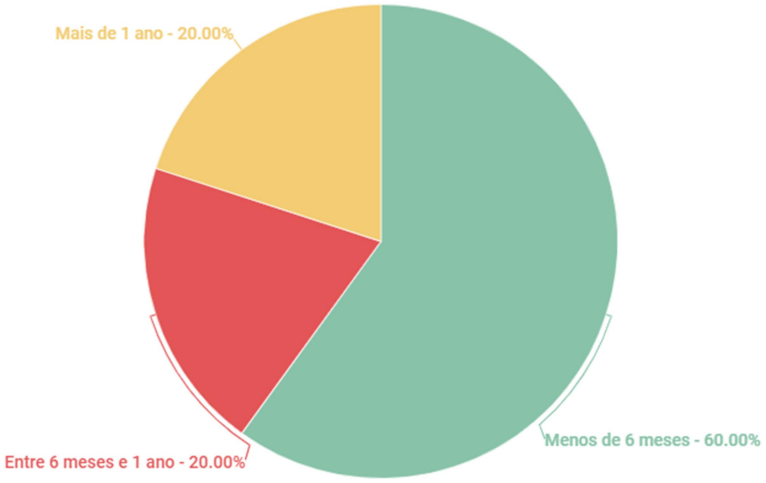
Figura 13 – Conhecimento em teste de *performance*.



Fonte: Elaborado pelos autores.

Inicialmente foi questionado sobre o conhecimento dos integrantes acerca da realização do teste de *performance*, apenas um declarou conhecer alguns aspectos da realização deste tipo de teste. A Figura 14 abaixo exibe informações sobre a quanto tempo cada membro trabalha na FTT.

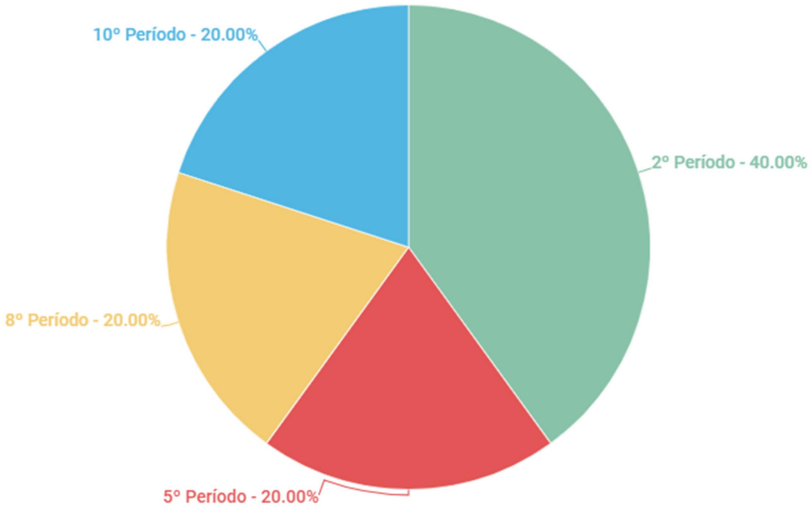
Figura 14 – Tempo de trabalho na FTT.



Fonte: Elaborado pelos autores.

Nota-se que há uma diferença de tempo em que cada individuo está na FTT, isso ocorre devido a alta rotatividade já mencionada no ambiente de trabalho. A Figura 15 abaixo apresenta informações acerca do período de estudo no qual cada membro pertence.

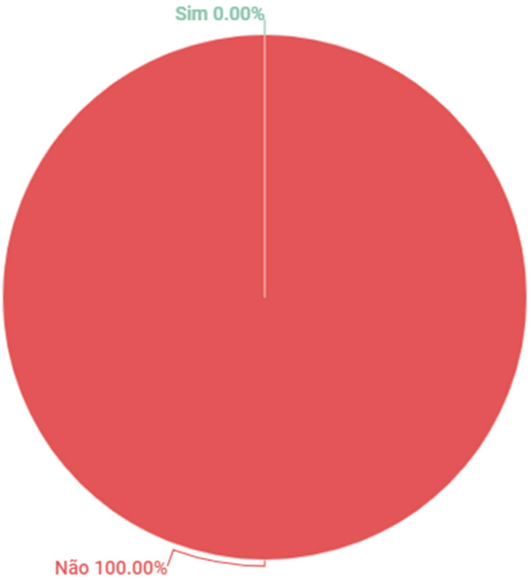
Figura 15 – Período do curso dos integrantes da equipe de teste.



Fonte: Elaborado pelos autores.

A FTT possui a característica de acolher estudantes de diversos períodos, em todas as etapas de suas carreiras. Permitindo assim, uma transmissão de experiências muito importante para a formação profissional destes alunos. A próxima Figura, 16, exibe dados a respeito do conhecimento dos integrantes sobre as ferramentas para realização dos testes de *performance*.

Figura 16 – Conhecimento acerca das ferramentas de teste.



Fonte: Elaborado pelos autores.

De todos os participantes, nenhum possui conhecimento sobre as ferramentas JMeter e BlazeMeter, portanto, este gráfico destaca a importância da realização do workshop e dos tutoriais prévios antes da aplicação do processo.

5 Resultados Alcançados

Devido a alta rotatividade da FTT, diversas vezes o conhecimento não é transmitido de membros antigos para novos membros, por isso a necessidade da padronização de todos os métodos de testes utilizados, visando otimizar o desenvolvimento de projetos da FTT e aumentar a eficácia das capacitações e entradas de novos integrantes na equipe.

Com base em informações coletadas com membros da FTT durante o *workshop* de aplicação, o teste de *performance* já foi bastante estudado pela equipe de testes, no entanto, sempre foi aplicado em forma de estudos ou pesquisas e não com um processo definido. Desta forma, esta pesquisa busca salientar a importância da realização deste teste, possibilitando diagnosticar em quais requisitos o sistema tem um baixo desempenho em questões de *performance*.

Seguindo a abordagem proposta, foi realizada uma pesquisa direta com os membros da equipe de teste, *scrum master e product owner,* com a realização de um questionário inicial (Apêndice A) e um questionário final (Apêndice B), visando obter dados para análise e comparação do cenário de pré e pós-aplicação do processo, com o objetivo de identificar características a respeito de testes de desempenho e dos processos da FTT.

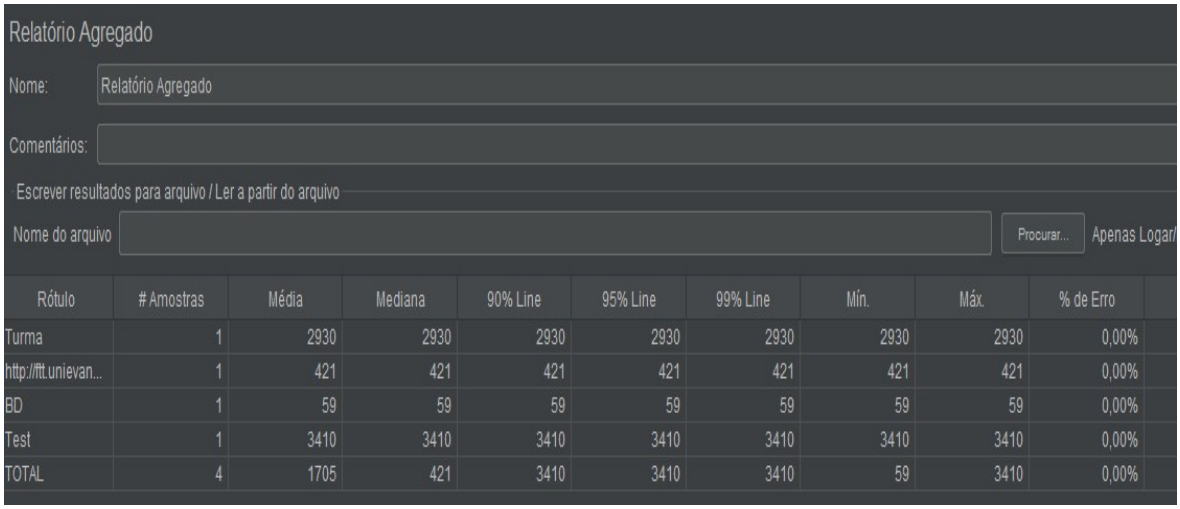
Após a apuração dos resultados do questionário, foram disponibilizados tutoriais em formato de documento e de vídeo, com a finalidade de facilitar a compreensão dos objetivos, do funcionamento do processo e das ferramentas utilizadas. Posteriormente, foi realizado um *workshop* online com os membros da equipe de teste para auxiliar na execução e sanar dúvidas. Foram realizados seis testes, três em cada requisito, por cada analista, totalizando trinta testes executados. Para maximizar a amostragem de resultados, na subseção 5.1 a seguir serão exibidos os resultados de um analista contendo os seis testes realizados. O restante dos testes foram organizados em forma de gráfico, permitindo uma melhor apresentação dos resultados e comparações de dados.

5.1 Resultados JMeter

Foi utilizado o ouvinte Resultado Agregado da ferramenta JMeter como principal ouvinte, que possibilita a visualização de diversos dados, como a média, mediana, valor mínimo, valor máximo, porcentagem de erros nas requisições, estas serão as métricas com prioridade nesta pesquisa. Os valores de média correspondem ao tempo médio de resposta das requisições, a mediana localiza o ponto central em ordem ascendente para realizar as comparações. Caso seja somente uma requisição, os valores de média e mediana serão iguais. Os dados de valor mínimo e máximo fazem justificativa aos termos e apresentam o menor e maior tempo de resposta, respectivamente, de cada requisição.

Conforme afirma Chaves (2019), estatísticas apontam que os usuários de aplicações *web* consideram um tempo médio ideal até dois segundos para obter uma resposta da aplicação. No entanto, este tempo ideal pode variar de acordo com a complexidade do requisito a ser testado e a forma como foi desenvolvido, também deve ser levado em consideração o servidor utilizado pelo sistema. As Figuras 17, 18 e 19 a seguir, representam os testes no requisito Turma, a ferramenta BlazeMeter gerou três requisições e um controlador de transação (Test), o controlador é encarregado de gerenciar todas as outras requisições, por isso será levado em consideração os seus valores de média para as próximas comparações.

Figura 17 – Requisito Turma com 1 usuário.

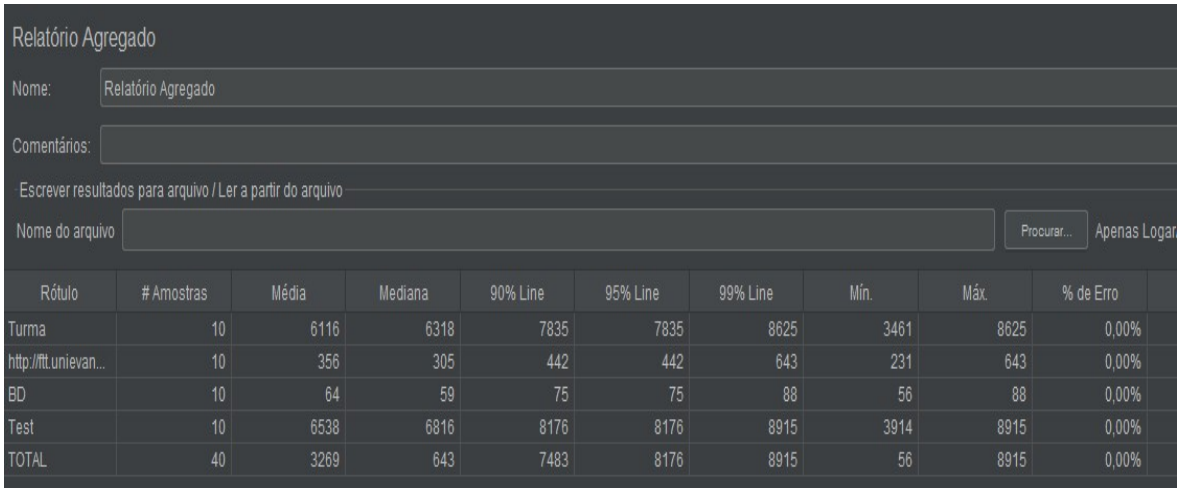


Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Turma	1	2930	2930	2930	2930	2930	2930	2930	0,00%
http://ftt.unievan...	1	421	421	421	421	421	421	421	0,00%
BD	1	59	59	59	59	59	59	59	0,00%
Test	1	3410	3410	3410	3410	3410	3410	3410	0,00%
TOTAL	4	1705	421	3410	3410	3410	59	3410	0,00%

Fonte: Analista de teste 1.

A Coluna de Rótulo na tabela de resultados representa o nome que o analista ou o BlazeMeter atribuiu para a requisição. Os resultados de apenas um usuário no requisito de Manter Turma foram de aproximadamente três segundos de média, um valor próximo do esperado. O teste foi executado com sucesso e sem nenhum erro nas requisições.

Figura 18 – Requisito Turma com 10 usuários.

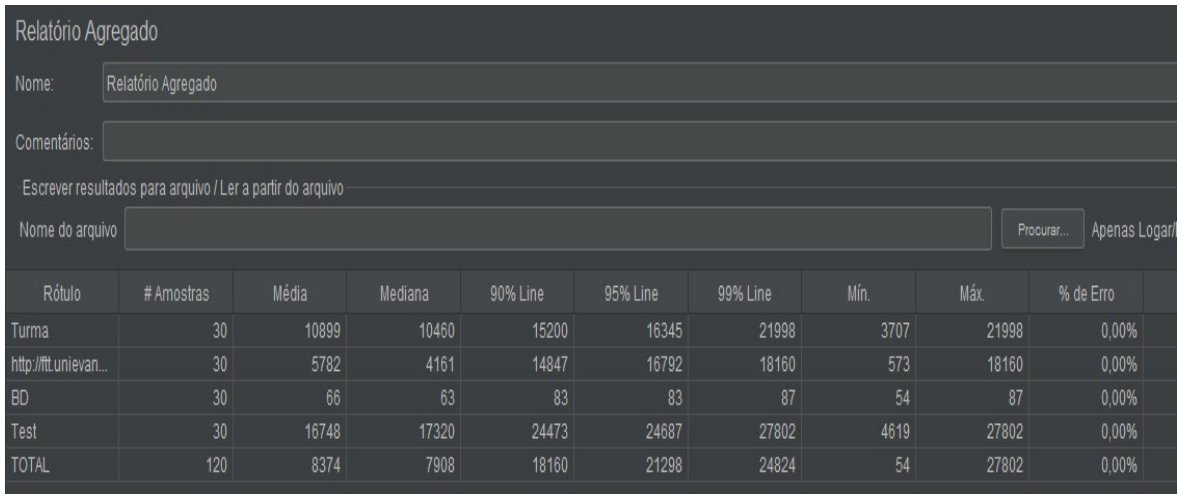


Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Turma	10	6116	6318	7835	7835	8625	3461	8625	0,00%
http://ftt.unievan...	10	356	305	442	442	643	231	643	0,00%
BD	10	64	59	75	75	88	56	88	0,00%
Test	10	6538	6816	8176	8176	8915	3914	8915	0,00%
TOTAL	40	3269	643	7483	8176	8915	56	8915	0,00%

Fonte: Analista de teste 1.

Os resultados de dez usuários simultâneos no requisito de Manter Turma foram de aproximadamente seis segundos de média, um valor acima do esperado. O teste foi executado com sucesso e sem nenhum erro nas requisições. No entanto, os valores de mínimo e máximo, foram demasiadamente excedidos.

Figura 19 – Requisito Turma com 30 usuários.



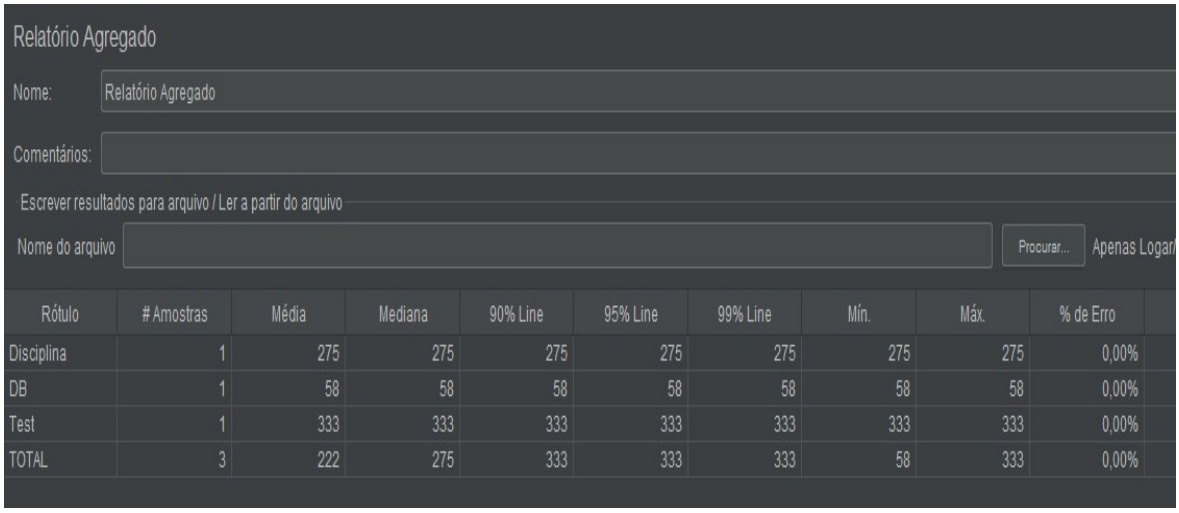
Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Turma	30	10899	10460	15200	16345	21998	3707	21998	0,00%
http://ftt.unievan...	30	5782	4161	14847	16792	18160	573	18160	0,00%
BD	30	66	63	83	83	87	54	87	0,00%
Test	30	16748	17320	24473	24687	27802	4619	27802	0,00%
TOTAL	120	8374	7908	18160	21298	24824	54	27802	0,00%

Fonte: Analista de teste 1.

Os valores de média e mediana seguem um padrão em relação à quantidade de amostras, no entanto, com trinta usuários estes valores são bastante superiores ao esperado. Os valores de mínimo e máximo também apresentaram grandes discrepâncias. Todas as requisições foram realizadas com sucesso. Contudo, seria ideal a realização do teste de carga e estresse para averiguar se este padrão se mantém com maiores quantidades de usuários.

As Figuras 20, 21 e 22 a seguir, representam os testes no requisito Manter Disciplina, a ferramenta BlazeMeter gerou novamente as três requisições e o controlador de transação (*Test*), como este requisito é menos complexo que o primeiro, os dados seguem a mesma tendência e não variam de forma abrupta.

Figura 20 – Requisito Disciplina com 1 usuário.

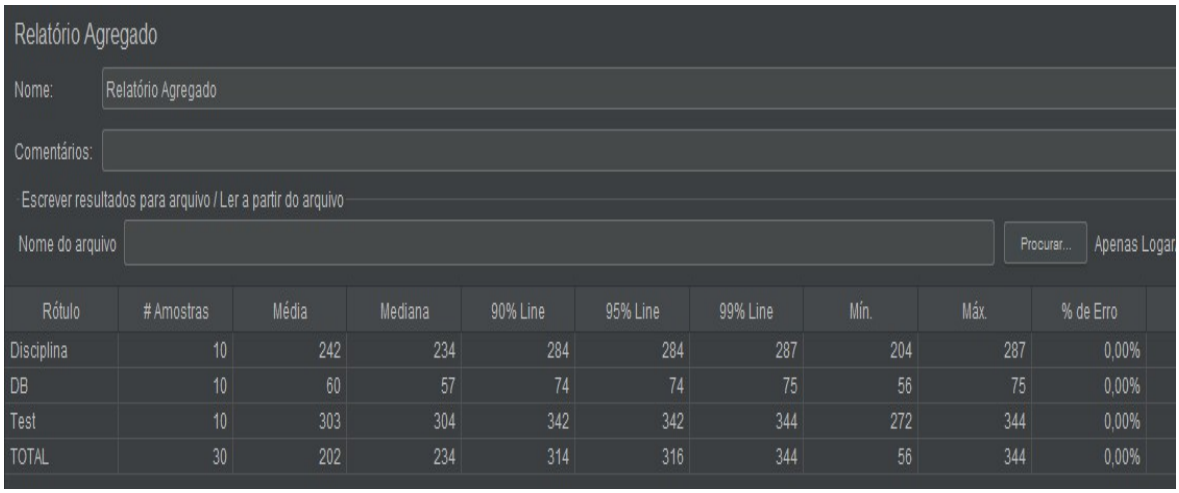


Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Disciplina	1	275	275	275	275	275	275	275	0,00%
DB	1	58	58	58	58	58	58	58	0,00%
Test	1	333	333	333	333	333	333	333	0,00%
TOTAL	3	222	275	333	333	333	58	333	0,00%

Fonte: Analista de teste 1.

Neste requisito, por ser de menor complexidade, os valores de média e mediana apresentados foram abaixo de um segundo, valores próximos do esperado. O teste foi executado com sucesso e sem nenhum erro nas requisições.

Figura 21 – Requisito Disciplina com 10 usuários.

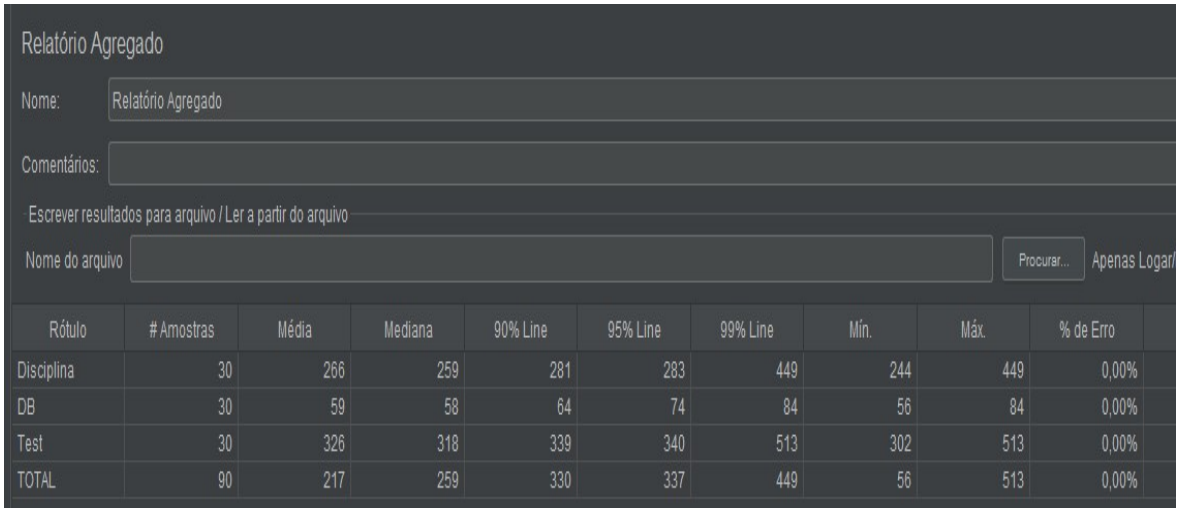


Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Disciplina	10	242	234	284	284	287	204	287	0,00%
DB	10	60	57	74	74	75	56	75	0,00%
Test	10	303	304	342	342	344	272	344	0,00%
TOTAL	30	202	234	314	316	344	56	344	0,00%

Fonte: Analista de teste 1.

Desta vez com dez usuários, os valores apresentados novamente foram próximos do esperado. A variação de mínimo e máximo foram supérfluos, apontando que o requisito permanece com a mesma velocidade de resposta com uma carga um pouco mais alta. O teste foi executado com sucesso e sem nenhum erro nas requisições.

Figura 22 – Requisito Disciplina com 30 usuários.



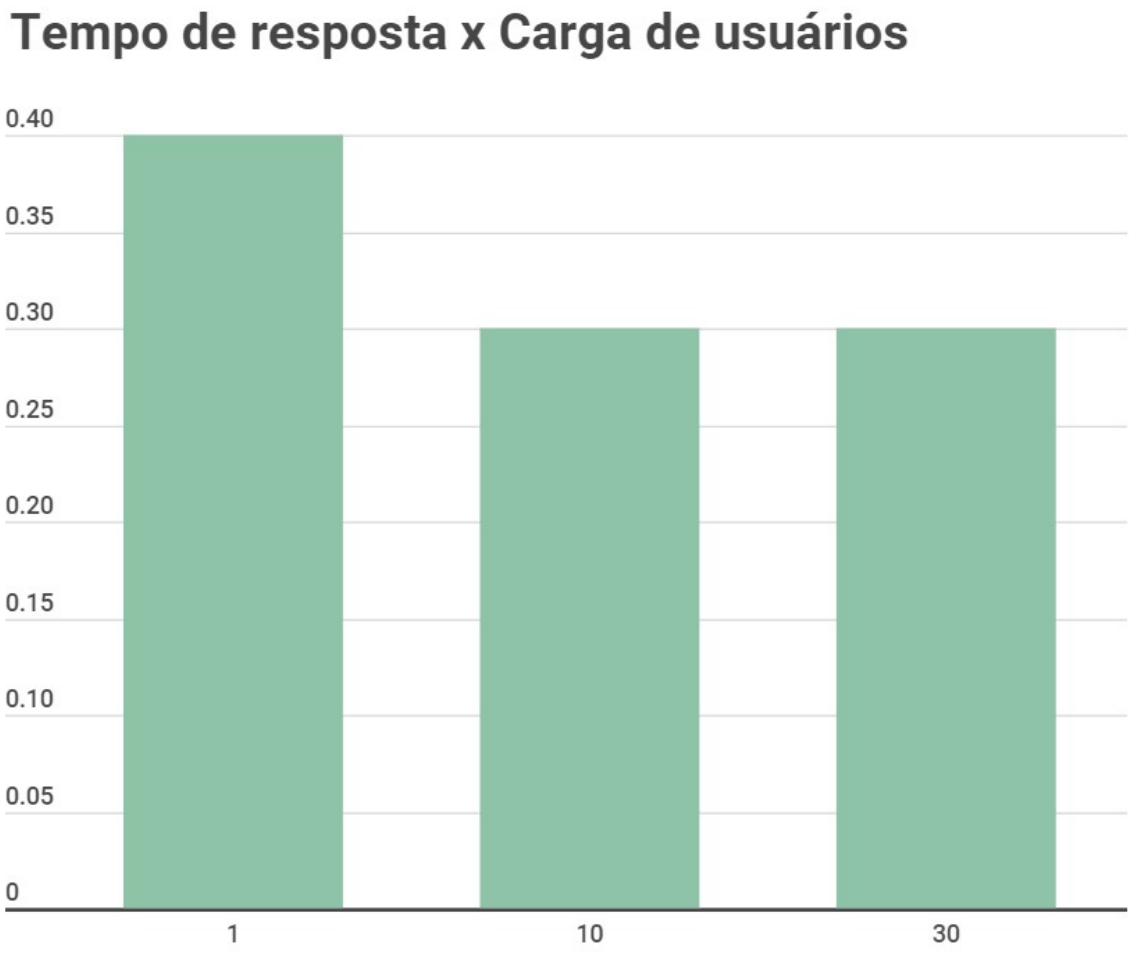
Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Mín.	Máx.	% de Erro
Disciplina	30	266	259	281	283	449	244	449	0,00%
DB	30	59	58	64	74	84	56	84	0,00%
Test	30	326	318	339	340	513	302	513	0,00%
TOTAL	90	217	259	330	337	449	56	513	0,00%

Fonte: Analista de teste 1.

De acordo com as imagens anteriores, pode ser notada a diferença entre os resultados dos primeiros testes, principalmente pela complexidade entre os requisitos. Desta vez os dados se manteram constantes, não seguindo a forma de crescimento conforme foi no requisito de Manter Turma. Como os valores de tempo de resposta se mantiveram com menos de um segundo de diferença entre os três testes, este requisito possui uma qualidade de desempenho superior ao outro requisito.

De mesma forma, os dados podem variar de acordo com as especificações da máquina utilizada para a realização do teste, bem como se o analista estiver com outros programas abertos ao mesmo tempo. A Figura 23 abaixo representa a relação entre o tempo de resposta (em segundos) e a quantidade de usuários simultâneos do requisito de Manter Disciplina.

Figura 23 – Tempo de resposta x Carga de usuários – Manter Disciplina.

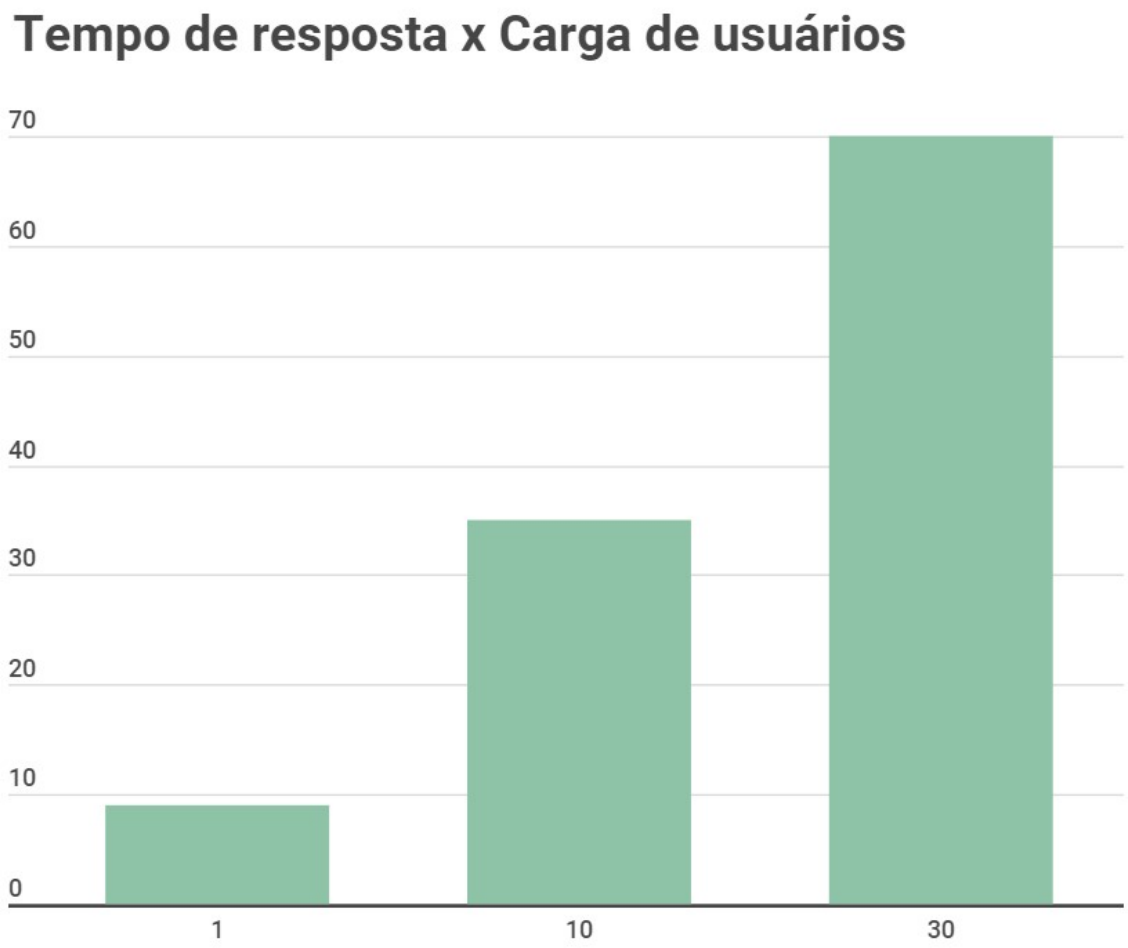


Fonte: Elaborado pelos autores.

O gráfico foi constituído da média dos resultados de todos os testes realizados pelos analistas, pode ser notado que a diferença é mínima entre os resultados, questão de milissegundos. Desta forma é possível afirmar que o requisito atende os padrões de qualidade estimados.

A Figura 24 abaixo representa a relação entre o tempo de resposta (em segundos) e a quantidade de usuários simultâneos do requisito de Manter Turma.

Figura 24 – Tempo de resposta x Carga de usuários – Turma.



Fonte: Elaborado pelos autores.

Diferentemente do primeiro gráfico, o requisito de Manter Turma apresentou uma grande divergência nos resultados. Isto é ocasionado pela complexidade do requisito, também por possuir detalhes como filtros de pesquisa e dependência com outros requisitos. Desta forma é possível afirmar que o requisito não atende completamente os padrões de qualidade estimados.

Outro tópico importante para considerar é a questão da especificação da máquina que o analista que realizou o teste, conforme já foi mencionado, todos os analistas possuem máquinas capazes de realizar os testes de forma adequada. No entanto, um ponto para ponderar é a velocidade de internet dos mesmos. As Figuras 25 e 26 abaixo representam a relação do tempo de resposta médio dos três testes realizados (em segundos) e velocidade de internet (em *megabytes*) nos requisitos de Manter Disciplina e Manter Turma respectivamente.

Figura 25 – Tempo de resposta x Velocidade de internet - Disciplina.

Fonte: Elaborado pelos autores.

É notório que a internet pode variar de forma considerável os resultados dos testes, principalmente abaixo dos 10 *megabytes*. No entanto, a variação foi pouco mais de 1 segundo no requisito de Manter Disciplina, considerado razoável no quesito de *performance*.

Figura 26 – Tempo de resposta x Velocidade de internet - Turma.

Fonte: Elaborado pelos autores.

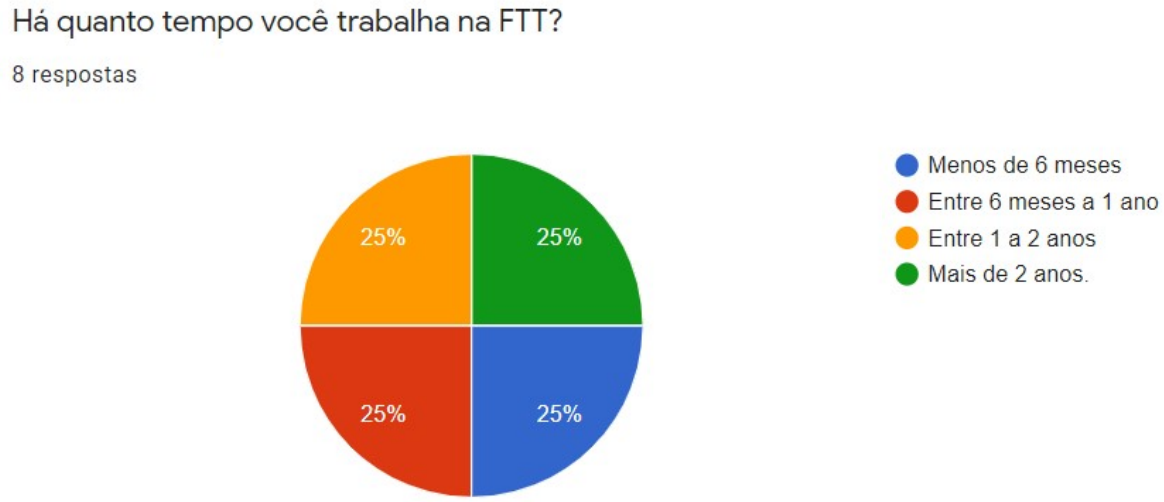
Novamente uma diferença grande entre os requisitos de Manter Disciplina e Manter Turma. Desta vez, a curva de complexidade do requisito x tempo de resposta mostrou-se justificada. A velocidade de internet é um fator importante, através dela pode ser simulado o ambiente do usuário final e comparar com mais precisão os resultados obtidos.

5.2 Comparação de questionários

Conforme apresentado no método de pesquisa, este trabalho aplicou dois questionários aos membros das equipes envolvidas com os processos de teste da FTT. O objetivo destes questionários era averiguar informações acerca dos integrantes e de seus conhecimentos sobre os temas abordados por este trabalho. Nota-se que um membro da equipe de teste não realizou os questionários por problemas pessoais, no entanto, o mesmo participou do workshop e da aplicação do processo. No total foram oito participantes, das

equipes de *scrum master* (2), *product owner* (2) e equipe de testes (4). A Figura 27 a seguir, exibe o tempo de trabalho que cada integrante possui na FTT. As imagens de comparação, a partir da Figura 29, são apresentadas na ordem de aplicação, questionário um (Q1) e questionário dois (Q2).

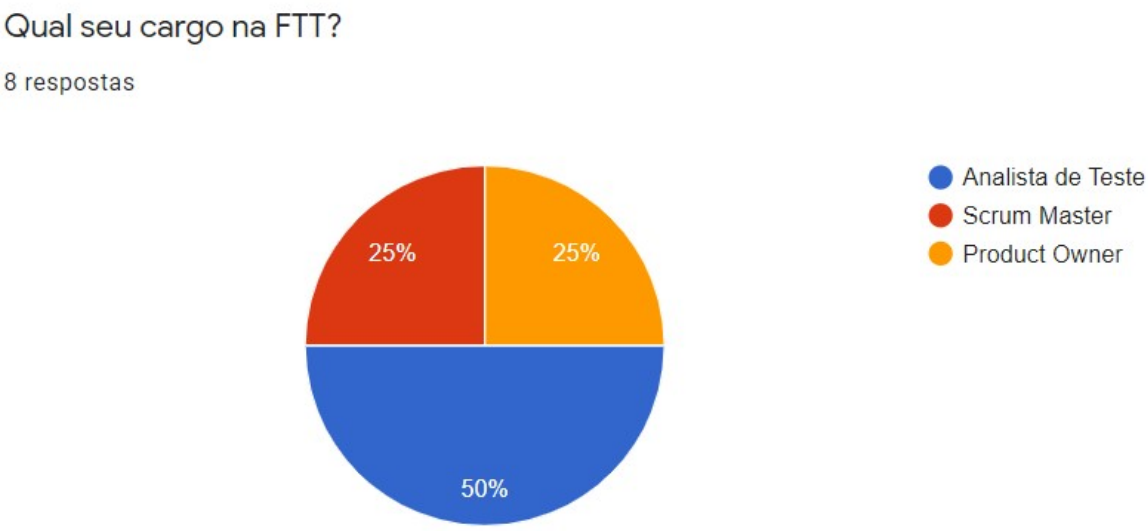
Figura 27 – Tempo de trabalho na FTT - Q1 e Q2.



Fonte: Elaborado pelos autores.

Ambos os questionários foram realizados com o auxilio da ferramenta Google Forms. Como já apresentado, a FTT possui alta rotatividade de membros, sendo uma porta de entrada para muitos que estão iniciando a sua carreira em diversas áreas da computação. Com membros de diversos períodos, a transmissão de conhecimento fortalece a interação entre as equipes. A Figura 28 abaixo apresenta as equipes dos membros participantes.

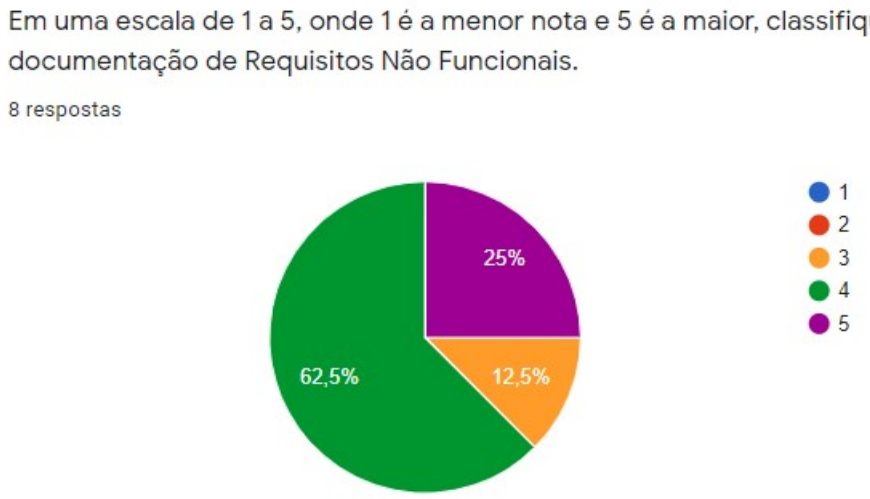
Figura 28 – Equipes dos integrantes - Q1 e Q2.



Fonte: Elaborado pelos autores.

Inicialmente a ideia era aplicar os questionários somente para a equipe de teste, no entanto, estendeu-se para as outras duas equipes, pelo fato de ambas participarem direta ou indiretamente de todo o conteúdo produzido pela equipe de teste. As Figuras 29 e 30 em sequência demonstram as opiniões dos participantes antes e depois da aplicação do processo sobre a documentação de requisitos não funcionais.

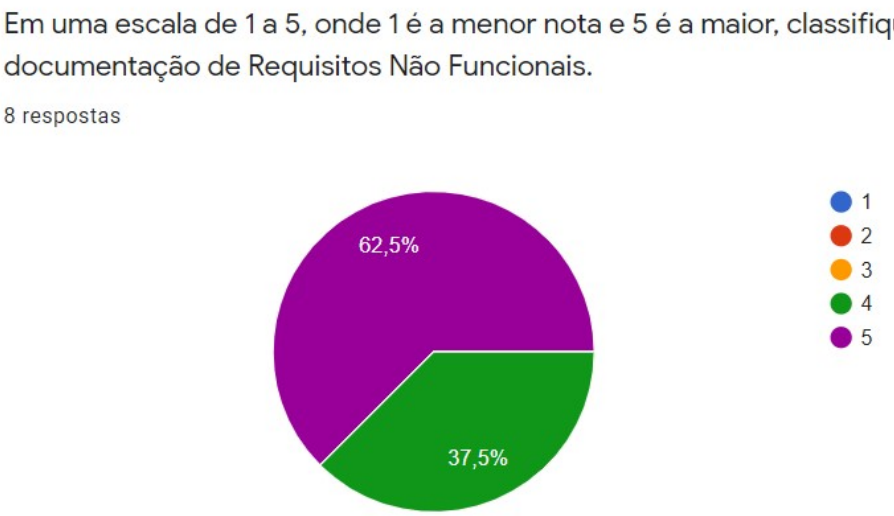
Figura 29 – Documentação não funcional – Q1.



Fonte: Elaborado pelos autores.

A documentação de requisitos não funcionais (RNF) é um aspecto fundamental para o bom entendimento das limitações e restrições de um sistema, com ela é possível identificar fatores como o tempo de resposta do sistema, o nível de segurança desejado, além da portabilidade para sistemas operacionais. A Figura 30 abaixo indica a opinião dos participantes após a aplicação do processo.

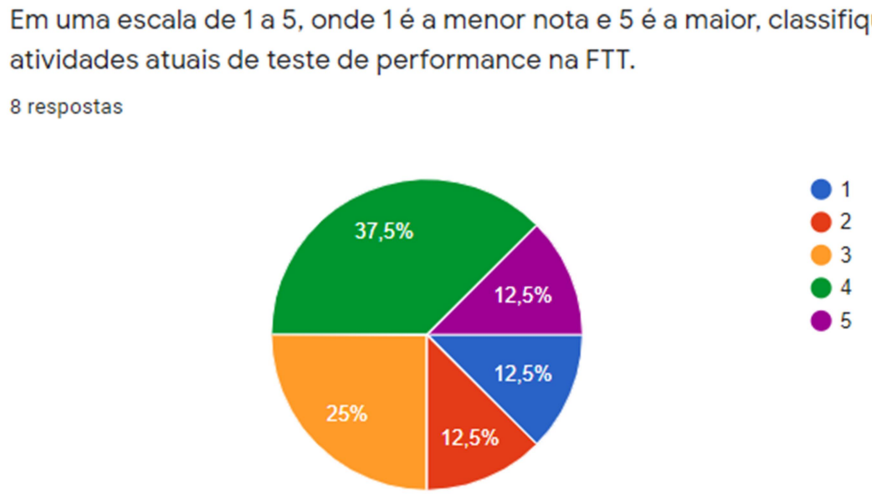
Figura 30 – Documentação não funcional – Q2.



Fonte: Elaborado pelos autores.

Inicialmente os participantes julgavam importante esta documentação, contudo, após a aplicação do processo, houve uma mudança em suas opiniões, justamente pelo fato do trabalho demonstrar que estes RNF são de extrema importância, portanto, não devem ser deixados de lado durante a fase de levantamento de requisitos de um projeto. As Figuras 31 e 32 adiante, evidenciam a eficiência das atividades de teste de *performance* na FTT.

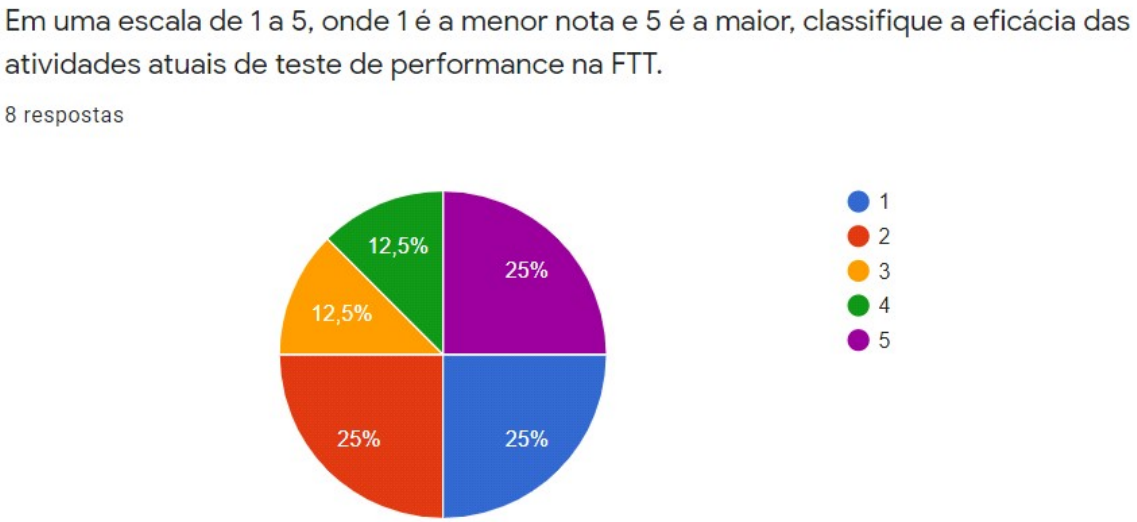
Figura 31 – Atividades de teste de *performance* na FTT– Q1.



Fonte: Elaborado pelos autores.

Com a aplicação do questionário inicial, ficou perceptível que as opiniões dos participantes foram bastante distintas. Na FTT, os membros da equipe de teste estudam diversas ferramentas e métodos de teste, porém, nem sempre são aplicados ou documentados. O mesmo acontece com atividades de teste de *performance*. A Figura 32 abaixo apresenta a opinião dos integrantes após a aplicação do processo.

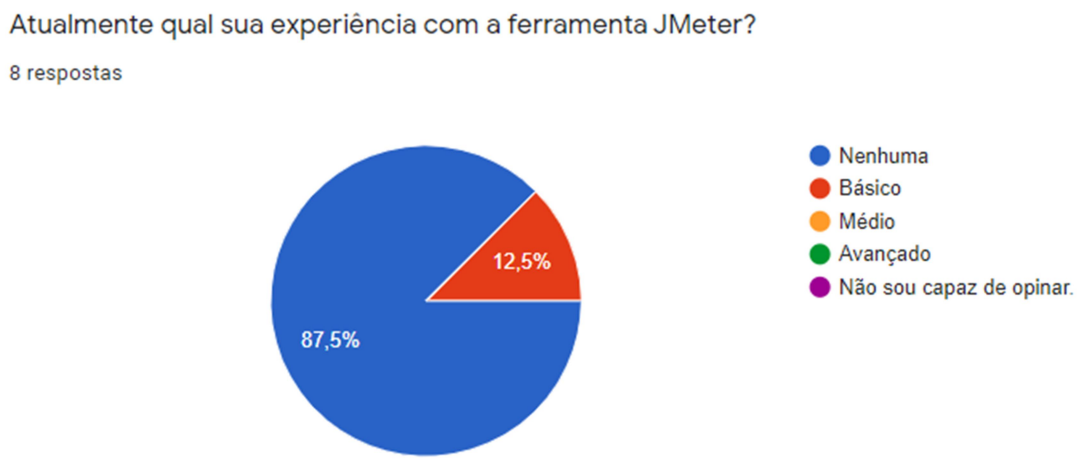
Figura 32 – Atividades de teste de *performance* na FTT – Q2.



Fonte: Elaborado pelos autores.

Com a aplicação do processo, foi observado que os participantes se atentaram justamente sobre a questão da eficácia das atividades de teste. Conforme já evidenciado anteriormente, os testes de *performance* na FTT não são realizados de forma periódica e não possuem um processo definido para a atividade. Portanto, não são eficazes de modo que melhorem a qualidade dos projetos em desenvolvimento. As Figuras 33 e 34 a seguir, destacam a experiência dos participantes com a ferramenta JMeter.

Figura 33 – Experiência com o JMeter– Q1.

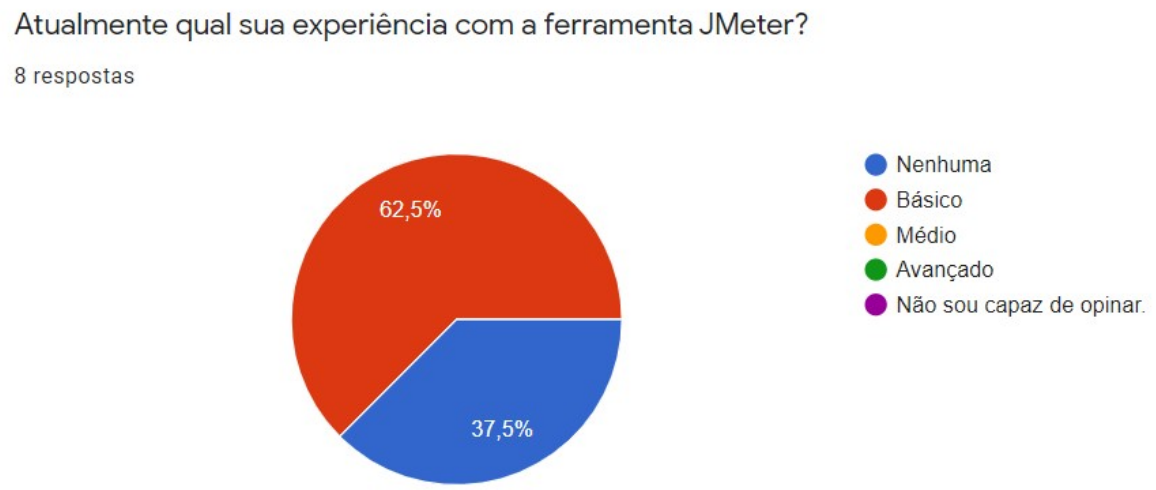


Fonte: Elaborado pelos autores.

A ferramenta JMeter amplamente discutida neste trabalho, foi a escolhida para a aplicação do processo, por apresentar dentre outros fatores, a facilidade de aprendizado, ser *open source* e possuir integração com outras ferramentas. A princípio, apenas um membro

possuía experiência com o JMeter, fato justificável através da alta rotatividade da FTT. A Figura 34 em seguida apresenta a opinião dos integrantes após a aplicação do processo.

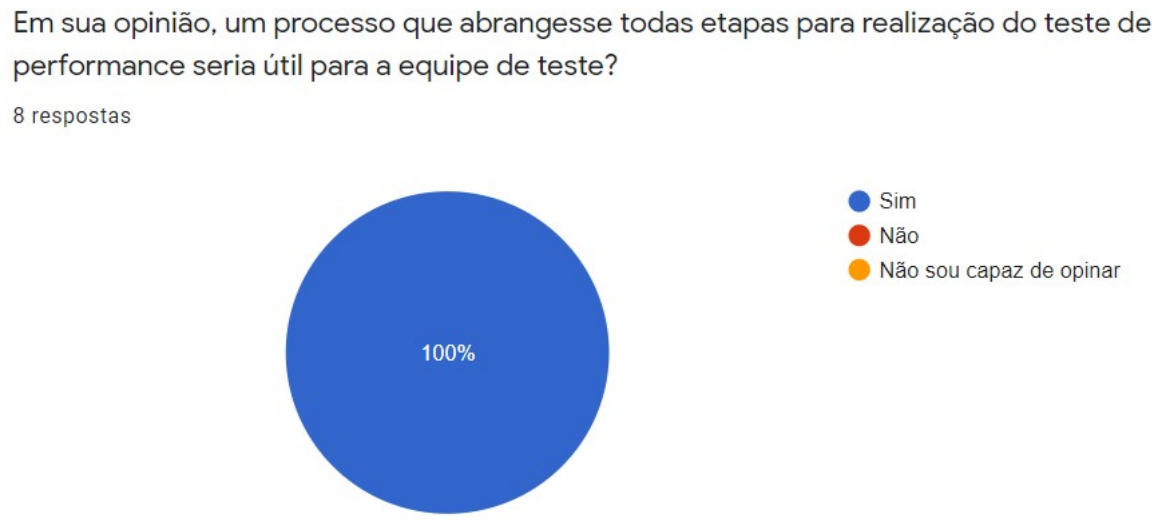
Figura 34 – Experiência com o JMeter – Q2.



Fonte: Elaborado pelos autores.

O workshop juntamente com tutoriais para auxiliar na execução dos testes, contribuiu para aprimorar a experiência dos participantes com o JMeter. Fato este que também colabora para a carreira profissional dos mesmos. Desta maneira, deve ser destacado que o conhecimento em novas ferramentas amplia o horizonte do estudante, de forma que possibilita o desenvolvimento ou adaptação de novas metodologias e técnicas de teste para serem aplicadas na FTT. A Figura 35 abaixo expõe a opinião dos participantes a respeito do processo de teste.

Figura 35 – Novo processo de teste – Q1 e Q2.



Fonte: Elaborado pelos autores.

Uma das principais questões durante a elaboração desta pesquisa foi justamente o fato da equipe de teste não possuir um processo de teste de *performance* estruturado. A opinião dos integrantes nos dois questionários foram idênticas. Há a necessidade de implantação de um processo que facilite a aplicação dos testes de *performance*, de forma que possa contribuir para a qualidade final dos projetos desenvolvidos na FTT.

6 Considerações Finais

Dentre os principais objetivos desta pesquisa estão a sistematização e aplicação de um processo de teste de *performance*. Através de extensa pesquisa bibliográfica, foi sistematizado um processo que abrange os passos fundamentais para realização deste tipo de teste. Com a realização de workshops e tutoriais os integrantes puderam estar a par do propósito de funcionamento deste processo. Por meio de questionários, pôde ser realizada a comparação da opinião dos participantes antes e depois da aplicação do novo processo.

De forma geral o objetivo central foi concluído, conforme evidenciam os questionários aplicados e os resultados alcançados, o processo sistematizado mostrou-se adequado para o modelo de desenvolvimento de *software* da FTT e adaptável para os padrões de trabalho das equipes envolvidas. Contribuindo, não somente para a melhoria da qualidade final dos projetos, mas também para a formação profissional dos integrantes das equipes.

Apesar da atividade de teste ser uma prática bem difundida nas fábricas de *software*, o teste de *performance* ainda não faz parte da rotina da maioria destes empresas e quando são realizados, são aplicados sem processos e métricas definidas. Portanto, é comum somente grande projetos fazerem uso de procedimentos como o mostrado neste trabalho.

Devido a alta rotatividade da FTT, muitas vezes os estudos desenvolvidos pelos integrantes das equipes não são transmitidos aos novos membros, visto que os mesmos não são documentados e publicados nos repositórios das equipes. Desta forma, é considerado um esforço perdido por não ser repassado aos demais companheiros.

Esta pesquisa é uma continuação do trabalho iniciado por Chaves (2019), onde foram realizados testes de *performance* em um projeto da FTT, porém, sem um processo a ser seguido. Com isso, espera-se que o processo proposto permita uma maior formalização dos testes de desempenho e entendimento da equipe responsável. Desta forma, há uma contribuição não somente técnica, mas para a gestão de conhecimento do local aplicado.

Visto que os objetivos desta pesquisa são sistematizar e aplicar um novo processo, a análise profunda dos dados e possíveis melhorias de recolhimento de métricas não foram abordadas e são possibilidades de estudos futuros. Também é um potencial trabalho futuro o estudo aprofundado das ferramentas JMeter e BlazeMeter, visando identificar mais

benefícios deste *software* e diferentes maneiras de serem utilizados, notado que são ferramentas interessantes e *open source*.

Além disso, existem outros tipos e métodos de testes não funcionais, que podem ser levados em consideração num possível aprimoramento do processo descrito neste trabalho. Também são cabíveis melhorias e adaptações, de forma que possa ser mais eficiente quanto aos resultados e métricas que são obtidos. Pode-se, ainda, investigar o impacto deste e de outros testes na usabilidade do sistema.

Os resultados alcançados destacam a importância da realização do teste de *performance* para a qualidade de sistemas de *software*. Os dados apurados contribuem para a evidência da necessidade de melhorias no aspecto de *performance* do sistema e aponta que o mesmo pode não estar adequado para uma grande quantidade de usuários e requisições simultâneas.

Uma das principais questões no quesito de *performance* de um sistema, é a avaliação do cenário do cliente. Conforme foi descrito na abordagem proposta, o cenário do cliente é um pouco inferior ao dos integrantes da equipe de testes, com isso, devem ser feitas análises e buscar possíveis alternativas com a intenção de contornar esse empecilho para que possam ser realizados testes que simulem a configuração do usuário final.

Um dos problemas encontrados foi a falta de conhecimento dos membros das equipes acerca das ferramentas utilizadas e dos testes de *performance*, no entanto, foi possível aplicar o processo com a ajuda dos workshops e tutoriais desenvolvidos. Outra dificuldade da pesquisa foi encontrar informações acerca de processos de teste de *performance* e seu funcionamento, também sobre a documentação de requisitos não funcionais. Todavia, a busca pela qualidade e aprimoramento de *software* deve ser prioridade para qualquer empresa que queria manter um nível de excelência e desta forma promover aos clientes a qualidade esperada.

Referências Bibliográficas

ARAÚJO, L. B.; PIMENTA, L. C. F. **A Importância Do Teste De Software Para A Qualidade Do Projeto.** Minas Gerais. 2013. Disponível em: <https://pmkb.com.br/wp-content/uploads/2013/11/a-importancia-do-teste-de-software.pdf>. Acesso em: 19 mar. 2020.

APACHE, S. F. **Apache Foundation**. 2019. Disponível em: <https://www.apache.org/foundation/thanks.html>. Acesso em: 22 mai. 2020.

BLAZEMETER, CA. **Complete Continuous Testing for the Enterprise One Platform. 100% Open Source Compatible**. 2020. Disponível em: <https://www.blazemeter.com/product/>. Acesso em: 02 set. 2020.

CÉRGOLI, R. **Uma Análise da Aplicação de Testes no Desenvolvimento de um Sistema ERP.** Dissertação (Especialização) – Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP. São Paulo. 2017.

CHAVES, N. M. **Uma Abordagem de Aplicação de Teste de Performance na Fábrica de Tecnologias Turing – UniEvangélica**. Trabalho de Conclusão de Curso (Bacharelado) – Centro Universitário de Anápolis – UniEvangélica. Anápolis. 2019.

COSER, A. **Modelo para análise da influência do capital intelectual sobre a performance dos projetos de software**. 2012. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/99386/304292.pdf?sequence=1&isAllowed=y>. Acesso em: 28 out. 2020.

CRESPO, A. N.; SILVA, O.; BORGES, C. A; SALVIANO, C. F. **Uma Metodologia para Teste de Software no Contexto da Melhoria de Processo**. São Paulo. 2004. Disponível em:<https://www.researchgate.net/publication/237497188_Uma_Metodologia_para_Teste_de_Software_no_Contexto_da_Melhoria_de_Processo>. Acesso em: 02 out. 2019.

FREITAS, A. L. S. C. **Ontologia para Teste de Desempenho de Software**. Rio Grande do Sul. 2013. Disponível em: <http://repositorio.pucrs.br/dspace/bitstream/10923/1501/1/000449315-Texto%2bCompleto-0.pdf>. Acesso em: 16 mar. 2020.

FONSECA, T.; COSTA, L.; ARAGÃO, R.; ANDRADE, L.; DÓSEA, M. B. **Ferramentas para Teste de Desempenho JMeter x WebLoad**. Revista Mundo J. Ed. 053. Univale - Faculdades Integradas do Vale do Ivaí. 2012. Disponível em: <http://www.univale.com.br/unisite/mundo-j/artigos/53_Jmeter.pdf>. Acesso em: 11 abr. 2020.

GUARIENTI, P.; BERNARDINO, M.; ZORZO, A. F.; OLIVEIRA, F. M. **Uma Abordagem de Análise do Tempo de Resposta para Teste de Desempenho em Aplicações Web**. In. Anais do XV Workshop de Testes e Tolerância a Falhas – WTF. Florianópolis. 2014. Disponível em: <http://www.sbrc2014.ufsc.br/anais/files/wtf/ST1-2.pdf>. Acesso em 18 fev. 2020. OLHARRRRRRRRRRR

KALINOWSKI, M.; SPÍNOLA, R. **Introdução à Inspeção de Software - Aumentando a Qualidade Através de Verificações Intermediárias**. Rio de Janeiro, 2008. Disponível em: <https://www.researchgate.net/publication/256091533_Introducao_a_Inspecao_de_Software_-_Aumentando_a_Qualidade_Atraves_de_Verificacoes_Intermediarias>. Acesso em: 19 mar. 2020.

LUTZ, D. **Implantação de Novo Processo de Trabalho em uma Fábrica de Software Baseado nos Modelos Ágeis de Desenvolvimento**. 2017. Disponível em: <https://www.univates.br/bdu/bitstream/10737/1948/1/2017DouglasFernandoLutz.pdf>. Acesso em 16 mar. 2020.

MANZOR, S. **Application Performance Testing Basics**. 2012. Disponível em: <http://agileload.com/docs/default-document-library/application-performance-testing-basics-agileload.pdf?sfvrsn=0>. Acesso em 20 mar. 2020.

MELO, S. **Inspeção de software**. São Paulo. 2009. Universidade de São Paulo (USP). Disponível em: <http://www.teses.usp.br/teses/disponiveis/55/55134/tde-27022012-104952/pt-br.php>. Acesso em: 16 mar. 2020.

PAPPE, Integração. **Manual de Utilização da Ferramenta JMeter**. Goiânia, 2013. 29p. Manual Técnico. Instituto de Informática, Universidade Federal de Goiás.

RESENDE, R. S. F.; SANTOS, I. S.; NETO, P. A. S. **Geração de Testes de Desempenho e Estresse a partir de Testes Funcionais**. 2010. Revista de Informática Teórica e Aplicada. Volume 17, N. 2. Disponível em: <https://seer.ufrgs.br/rita/article/download/rita_v17_n2_p174/11208>. Acesso em: 25 mar 2020.

ROCHA, N. A. S. **Documentação de Software: Integração de Ferramentas de Modelação e Processamento de Texto**. Dissertação (Mestrado) – Faculdade de Engenharia da Universidade do Porto. Portugal. 2008. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/59642/1/000129520.pdf>. Acesso em: 22 mai. 2020.

RUFFATO, B. R. **O papel do governo brasileiro no fomento das inovações no setor das TICs: um enfoque na indústria de software**. 2010. Monografia (Graduação em Economia) – Faculdade de Ciências Econômicas, Universidade Federal do Rio Grande do Sul, Porto Alegre.

SANTIAGO, R. M. **Ensaio do SWEBOK – Software Engineering Body of Knowledge**. Trabalho de Conclusão de Curso - Universidade Gama Filho. Goiânia. 2011.

SANTOS, I. S.; NETO, P. A. S. **Automação dos Testes de Desempenho e Estresse com o Apache JMeter**. 2008. Disponível em: https://www.academia.edu/6385343/Automa%C3%A7%C3%A3o_de_Testes_de_Desempenho_e_Estresse_com_o_JMeter. Acesso em: 29 set. 2020.

SOUSA, F. C. **Análise de Técnicas para Estimativas de Esforço de Tempo em Projeto de Software Desenvolvidos na Fábrica de Tecnologias Turing – UniEvangélica**. Trabalho de Conclusão de Curso (Bacharelado) – Centro Universitário de Anápolis – UniEvangélica. Anápolis. 2018.

SOUZA, K. P.; GASPAROTTO, A. M. S. **A importância da atividade de teste no desenvolvimento de software**. In XXXIII Encontro Nacional De Engenharia De Produção. Bahia. 2013. Disponível em: <http://abepro.org.br/biblioteca/enegep2013_TN_STO_177_007_23030.pdf>. Acesso em: 02 set. 2019.

SOUZA, T. S. **Testes de Desempenho de Software: Teoria e Prática**. Minicursos da ERSI-RJ 2018 - V Escola Regional de Sistemas de Informação do Rio de Janeiro. SBC.

2018. Disponível em: <https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/7/13/39-1?inline=1>. Acesso em: 19 fev. 2020.

TUTORIALSPOINT. **Apache JMeter Asynchronous Javascript and xml**. 2015. Disponível em: <https://www.tutorialspoint.com/jmeter/jmeter_tutorial.pdf> Acesso em: 13 abr. 2020.

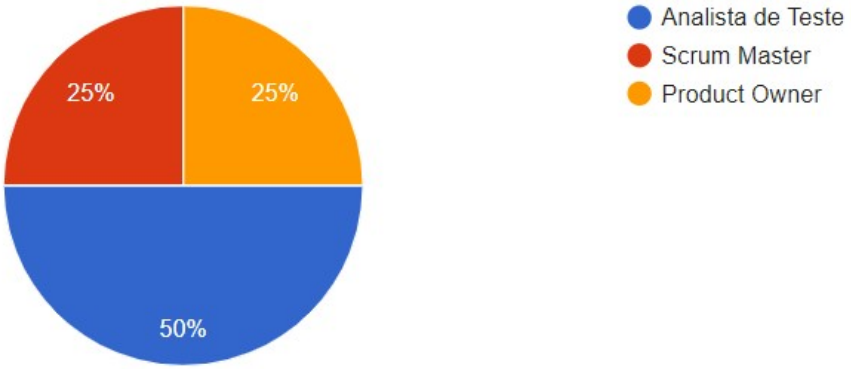
VINCENZI, A. M. R.; VALLE, P. H. D.; BARBOSA, J. R.; LANA, C. R.; DELAMARO, M. E.; MALDONADO, J. C. **Introdução ao Teste de Software**. 2016. Disponível em:<https://edisciplinas.usp.br/pluginfile.php/2266784/mod_resource/content/1/relatorio_teste.pdf>. Acesso em: 18 fev. 2020.

LISTA DE APÊNDICES

APÊNDICE A - Realizado antes da aplicação do processo

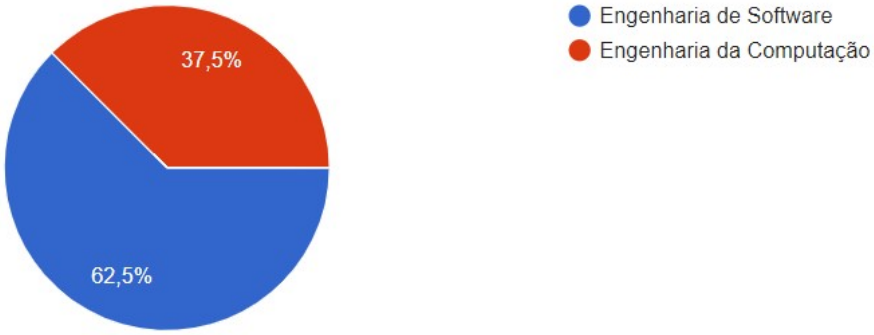
Qual seu cargo na FTT?

8 respostas



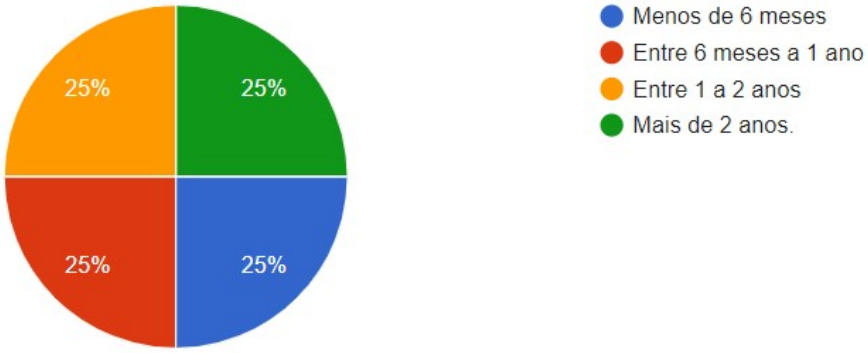
Qual o seu curso?

8 respostas



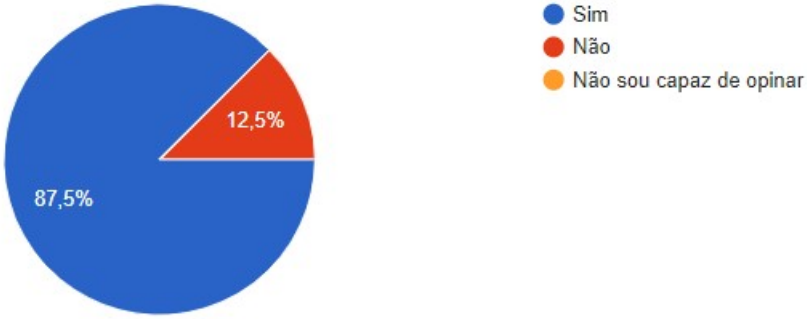
Há quanto tempo você trabalha na FTT?

8 respostas



Você sabe para qual(is) finalidade(s) são utilizados os Requisitos Não Funcionais identificados para um sistema?

8 respostas

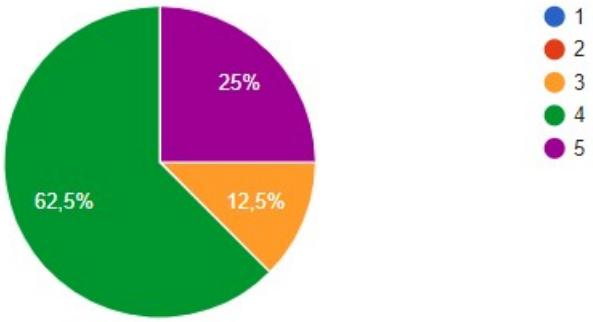


Você julga necessário a documentação de Requisitos Não Funcionais em um projeto de software?

8 respostas

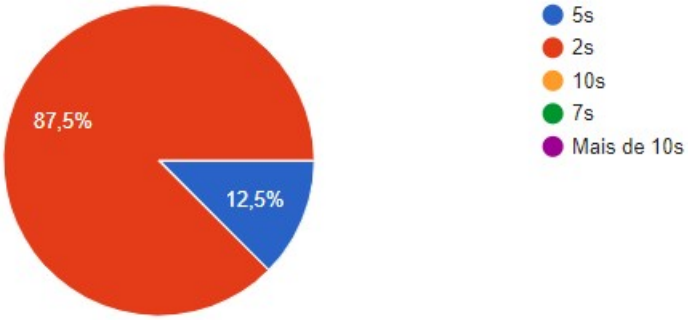
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a importância da documentação de Requisitos Não Funcionais.

8 respostas



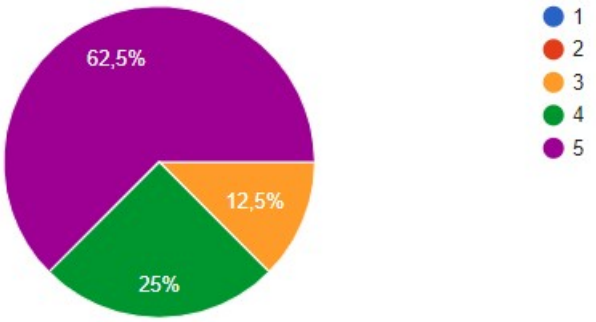
Em sua opinião, qual o tempo de resposta ideal para um sistema web? (em segundos).

8 respostas



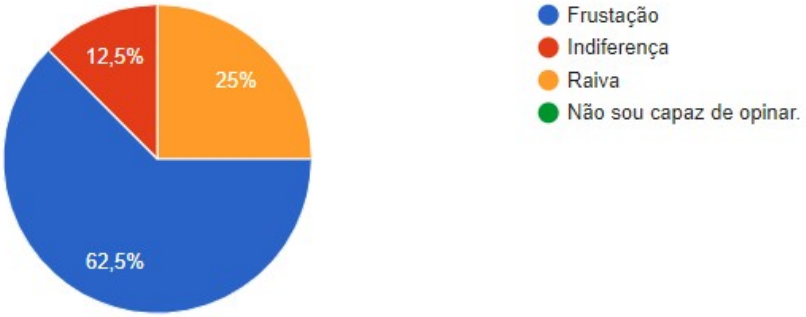
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a importância do tempo de resposta de um sistema web como um dos aspectos fundamentais para o bom funcionamento do sistema.

8 respostas



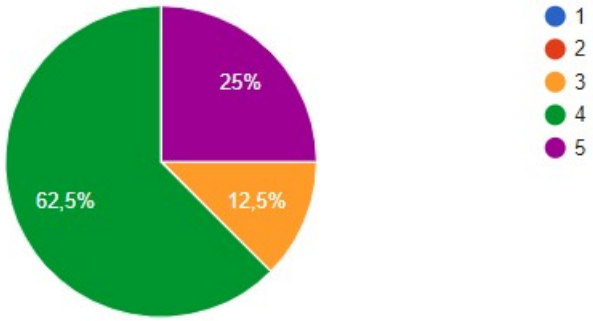
Quando você está navegando em um sistema e ele é muito lento, qual o seu sentimento em relação a isso?

8 respostas



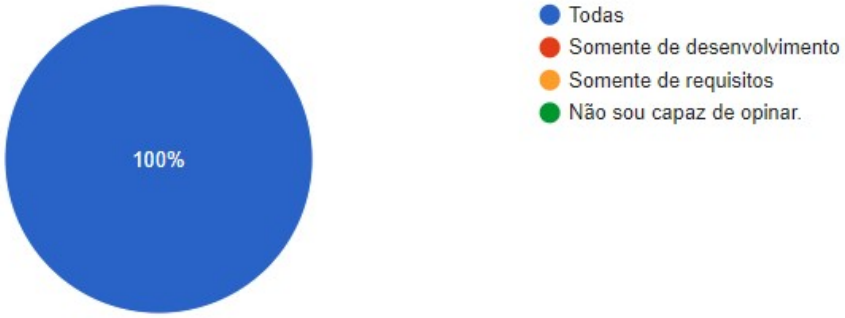
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique o nível de complexidade que você acredita ser a realização de testes de performance em um projeto de software.

8 respostas



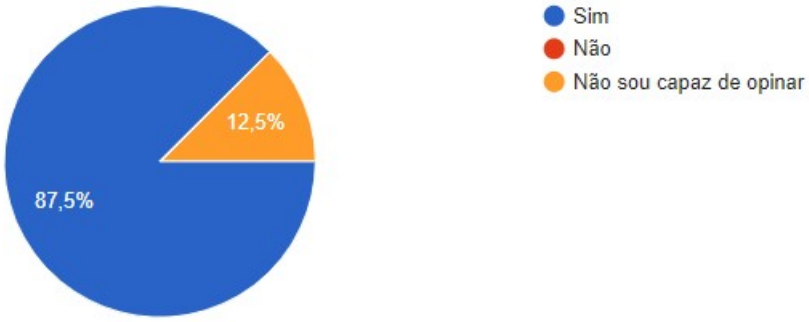
A performance de um sistema deve ser de responsabilidade de quais equipes envolvidas no projeto?

8 respostas



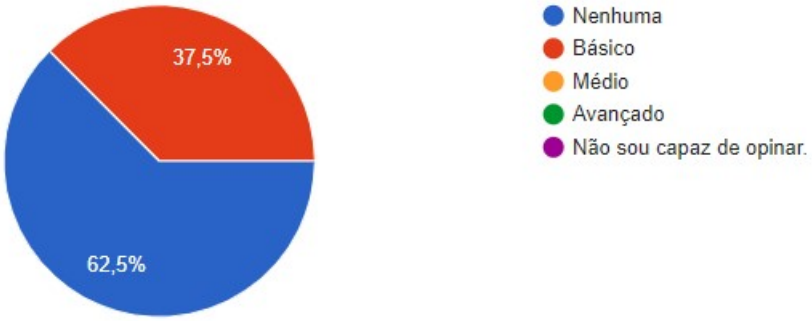
Você considera ser importante a realização de testes de performance em projetos de software?

8 respostas



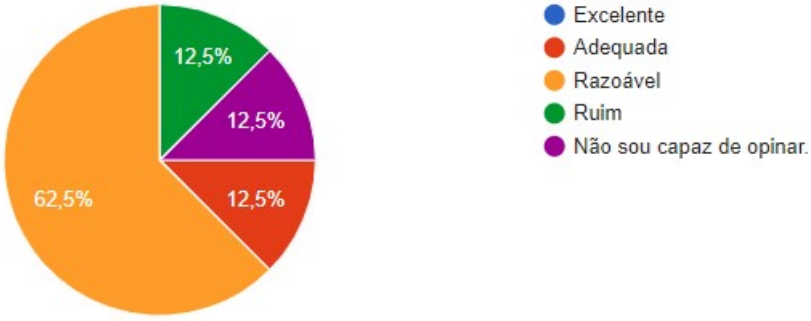
Atualmente qual sua experiência com testes de performance?

8 respostas



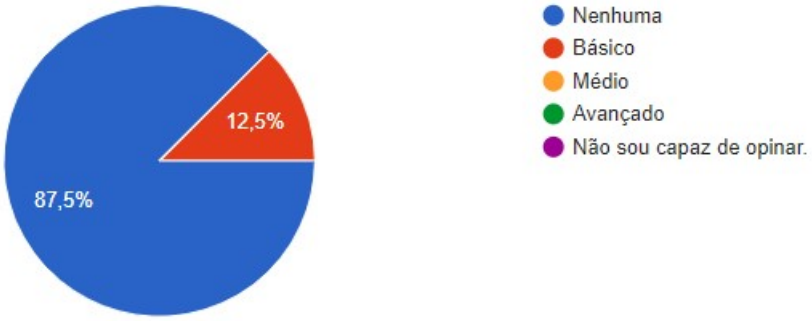
Como você avalia a performance do sistema virtoo?

8 respostas



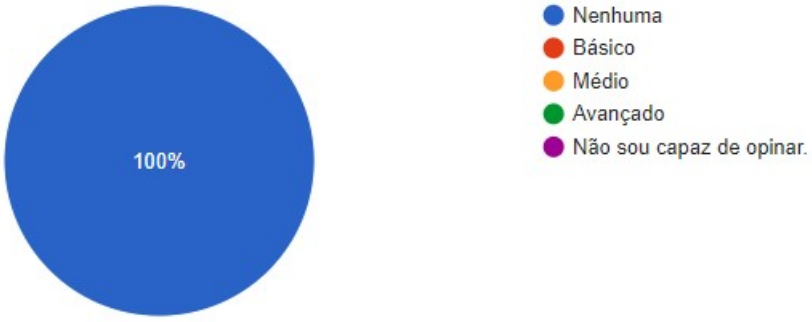
Atualmente qual sua experiência com a ferramenta JMeter?

8 respostas



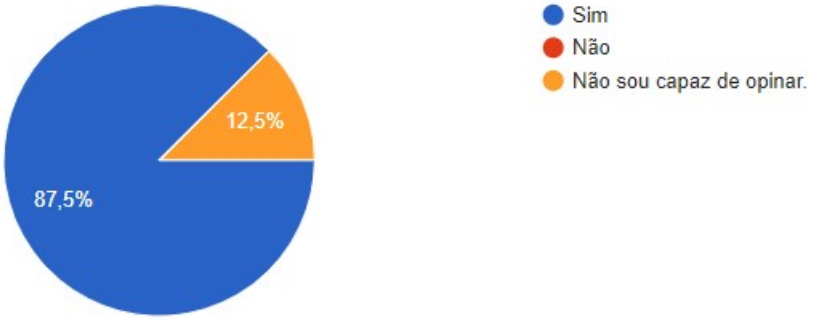
Atualmente qual sua experiência com a ferramenta BlazeMeter?

8 respostas



A FTT utiliza processos para realização de testes funcionais e validação de documentos/artefatos?

8 respostas

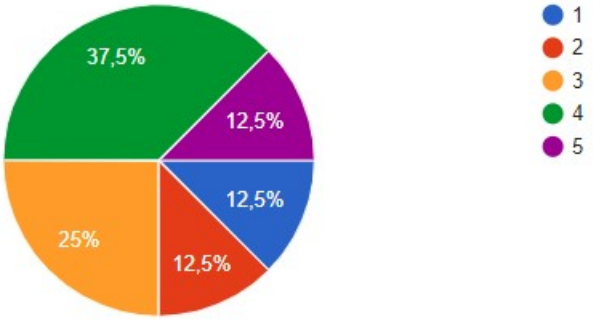


Em sua opinião, um processo que abrangesse todas etapas para realização do teste de performance seria útil para a equipe de teste?

8 respostas

Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a eficácia das atividades atuais de teste de performance na FTT.

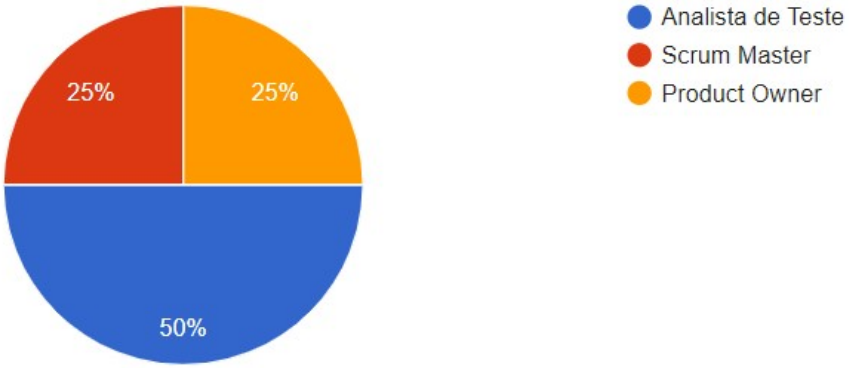
8 respostas



APÊNDICE B – Realizado após a aplicação do processo – Contém duas questões extras.

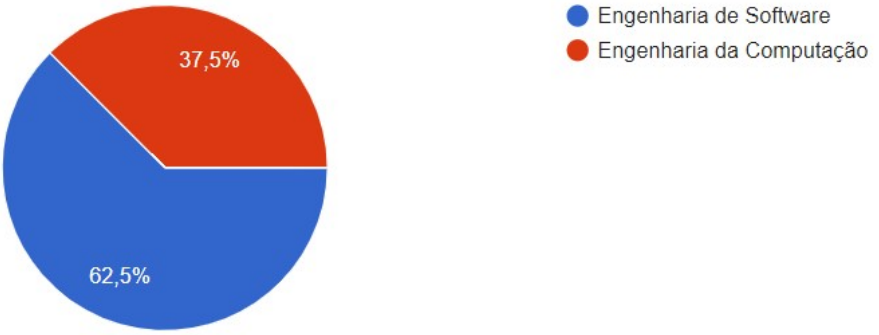
Qual seu cargo na FTT?

8 respostas



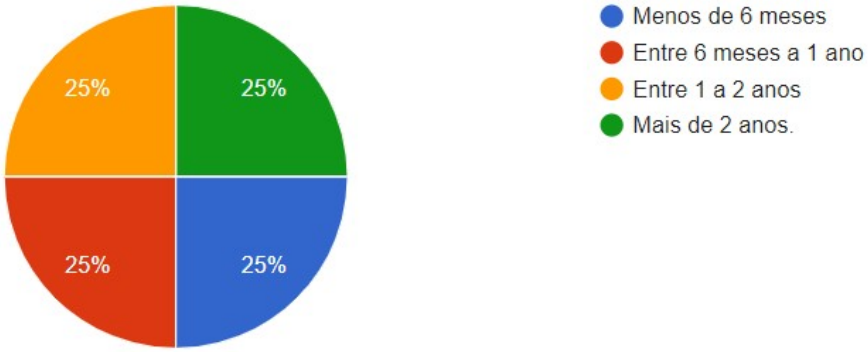
Qual o seu curso?

8 respostas



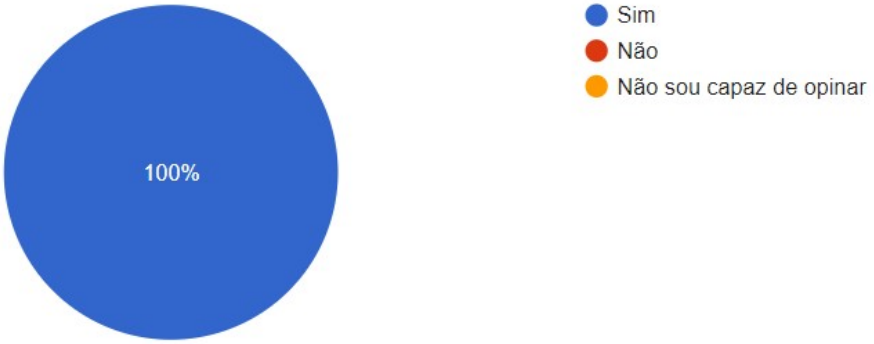
Há quanto tempo você trabalha na FTT?

8 respostas



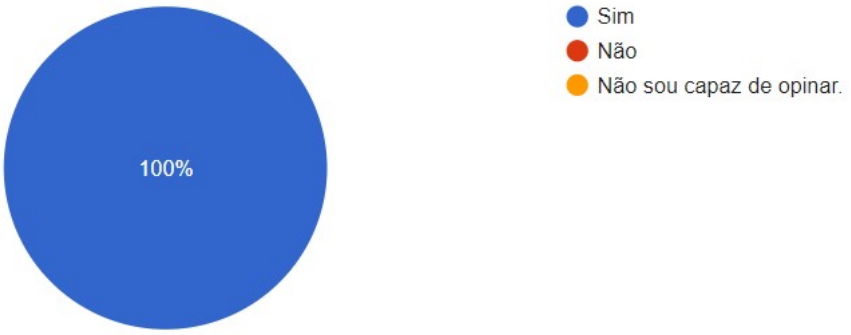
Você sabe para qual(is) finalidade(s) são utilizados os Requisitos Não Funcionais identificados para um sistema?

8 respostas



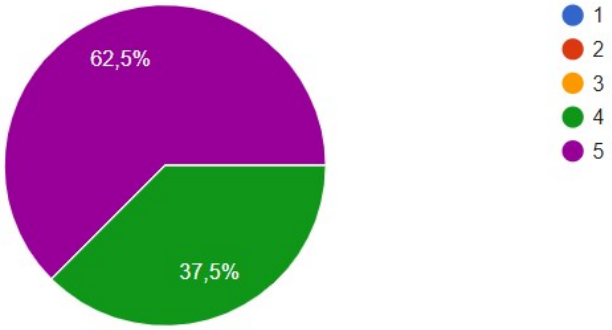
Você julga necessário a documentação de Requisitos Não Funcionais em um projeto de software?

8 respostas



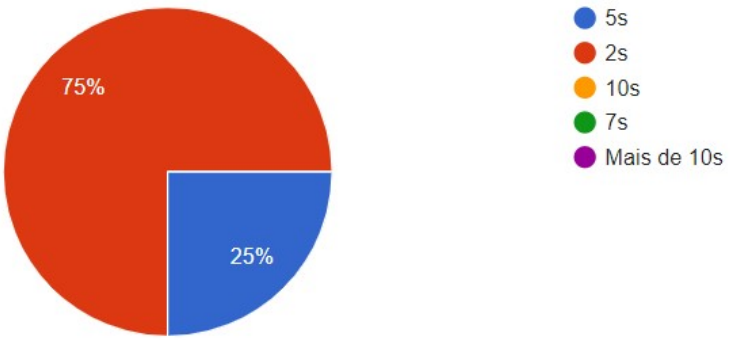
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a importância da documentação de Requisitos Não Funcionais.

8 respostas



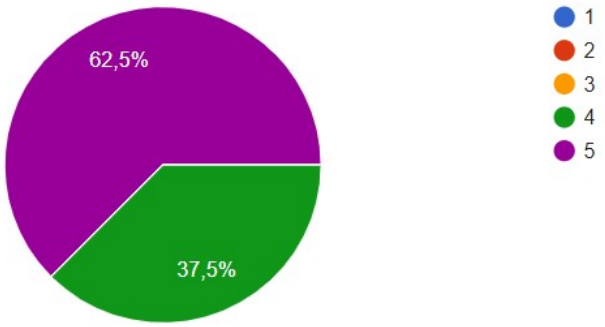
Em sua opinião, qual o tempo de resposta ideal para um sistema web? (em segundos).

8 respostas



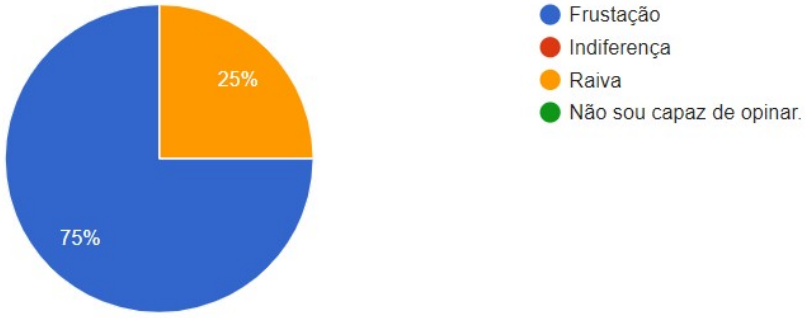
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a importância do tempo de resposta de um sistema web como um dos aspectos fundamentais para o bom funcionamento do sistema.

8 respostas



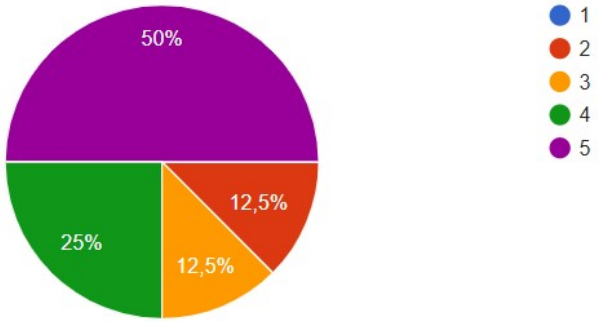
Quando você está navegando em um sistema e ele é muito lento, qual o seu sentimento em relação a isso?

8 respostas



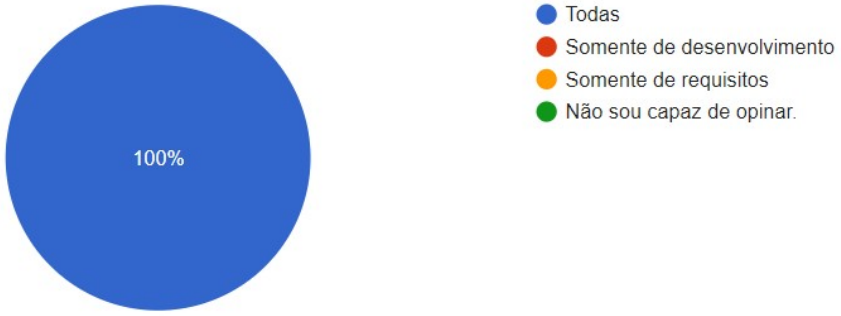
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique o nível de complexidade que você acredita ser a realização de testes de performance em um projeto de software.

8 respostas



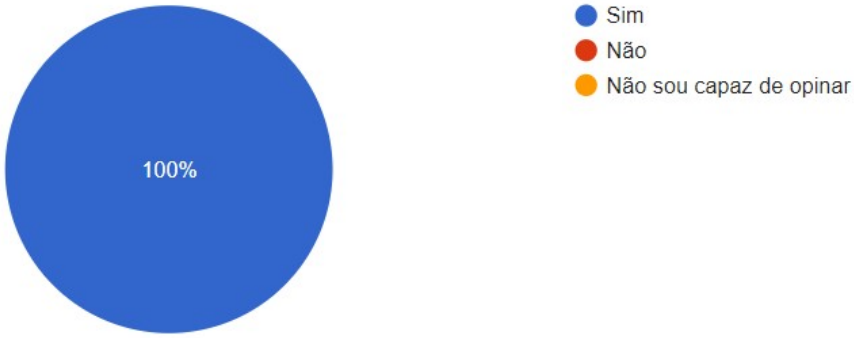
A performance de um sistema deve ser de responsabilidade de quais equipes envolvidas no projeto?

8 respostas



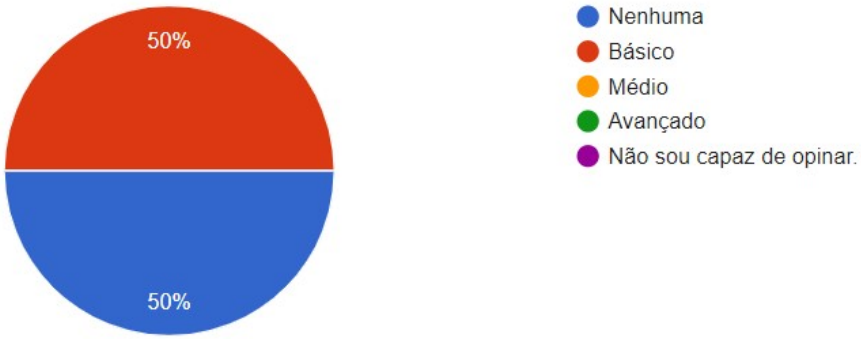
Você considera ser importante a realização de testes de performance em projetos de software?

8 respostas



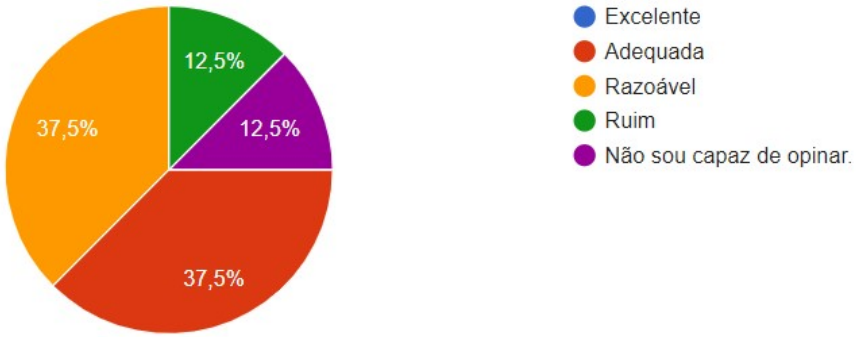
Atualmente qual sua experiência com testes de performance?

8 respostas



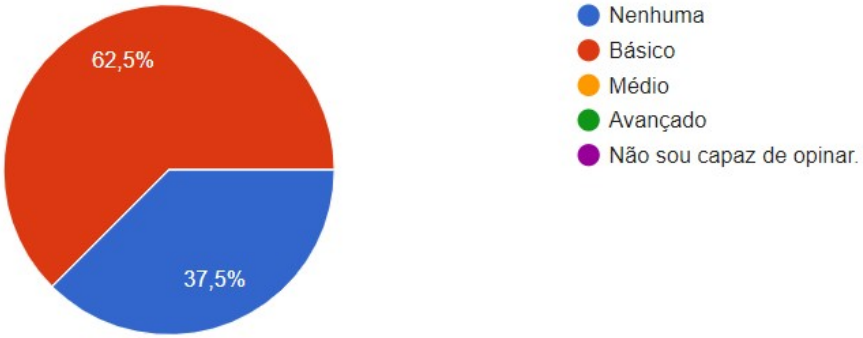
Como você avalia a performance do sistema virtoo?

8 respostas



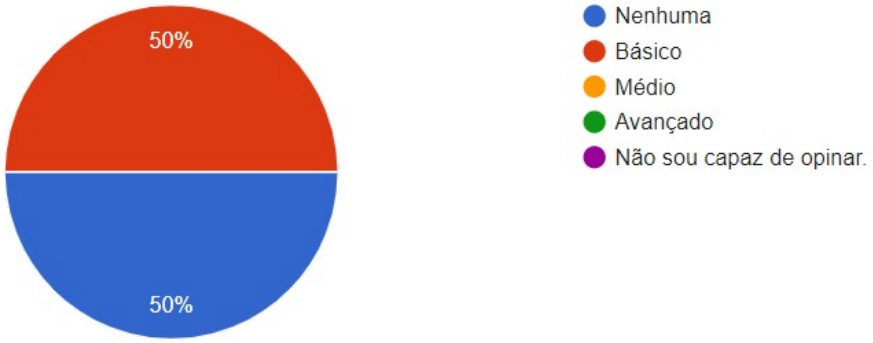
Atualmente qual sua experiência com a ferramenta JMeter?

8 respostas



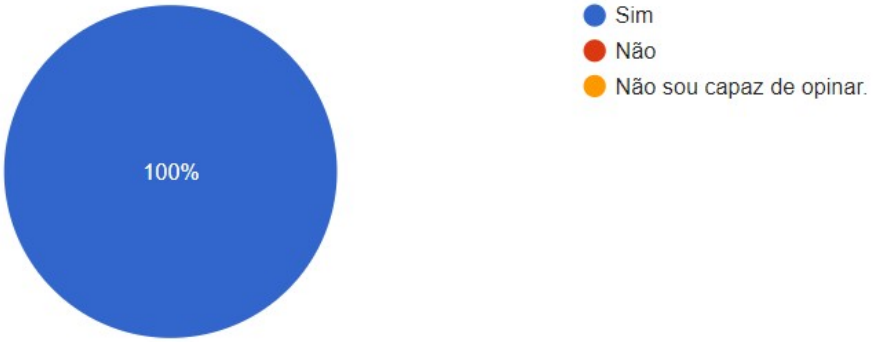
Atualmente qual sua experiência com a ferramenta BlazeMeter?

8 respostas



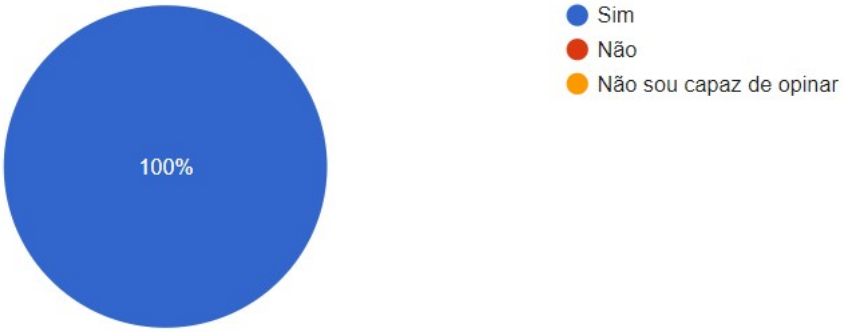
A FTT utiliza processos para realização de testes funcionais e validação de documentos/artefatos?

8 respostas



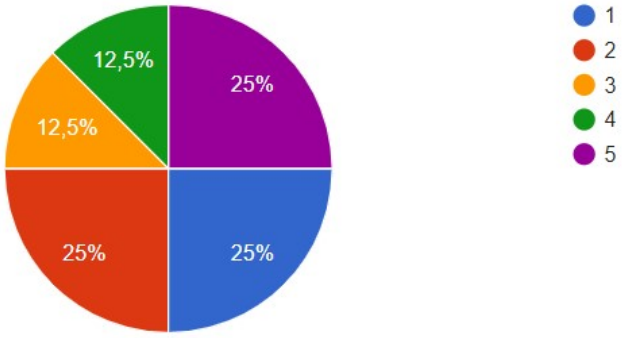
Em sua opinião, um processo que abrangesse todas etapas para realização do teste de performance seria útil para a equipe de teste?

8 respostas



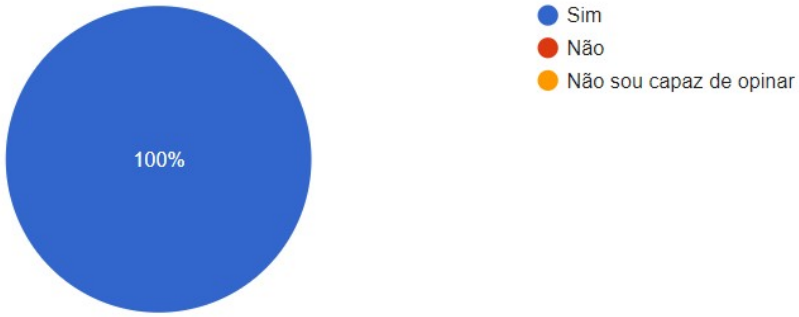
Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a eficácia das atividades atuais de teste de performance na FTT.

8 respostas



Em sua opinião, com um processo de teste de performance definido, seria possível otimizar o tempo de estudo dos novos membros da equipe em relação a este tipo de teste?

8 respostas



Em uma escala de 1 a 5, onde 1 é a menor nota e 5 é a maior, classifique a importância de se documentar processos e definir métodos de teste.

8 respostas