UNIVERSIDADE EVANGÉLICA DE GOIÁS – UNIEVANGÉLICA
ENGENHARIA DE SOFTWARE

Alexandre Kamimura Vieira Abadio
Davi Pereira Caixeta
Edilson Pereira da Silva
Gustavo Campos de Andrade

Processo para um teste de *stress* aplicável a sistemas de ensino a distância

Anápolis
Agosto, 2021

UNIVERSIDADE EVANGÉLICA DE GOIÁS – UNIEVANGÉLICA
ENGENHARIA DE SOFTWARE

Alexandre Kamimura Vieira Abadio
Davi Pereira Caixeta
Edilson Pereira da Silva
Gustavo Campos de Andrade

Processo para um teste de *stress* aplicável a sistemas de ensino a distância

Trabalho apresentado ao Curso de Engenharia de Software da Universidade Evangélica de Goiás – UniEVANGÉLICA da cidade de Anápolis-GO como requisito parcial para obtenção do Grau de Bacharel em Engenharia de Software.

Orientador(a): Prof. Ms. Alexandre Moraes Tannus.

Anápolis
Agosto, 2021

UNIVERSIDADE EVANGÉLICA DE GOIÁS – UNIEVANGÉLICA
ENGENHARIA DE SOFTWARE

Alexandre Kamimura Vieira Abadio
Davi Pereira Caixeta
Edilson Pereira da Silva
Gustavo Campos de Andrade

Processo para um teste de *stress* aplicável a sistemas de ensino a distância

Monografia apresentada para Trabalho de Conclusão de Curso de Engenharia de Software da Universidade Evangélica de Goiás - UniEVANGÉLICA, da cidade de Anápolis-GO como requisito parcial para obtenção do grau de Engenheiro(a) de Software.

Aprovado por:

_____

Alexandre Moraes Tannus, Mestre, UniEVANGÉLICA

(ORIENTADOR)

_____

(AVALIADOR)

Anápolis, ____ de _____ de 2021.

FICHA CATALOGRÁFICA

ABADIO, Alexandre Kamimura Vieira; CAIXETA, Davi Pereira; SILVA, Edilson Pereira da; ANDRADE, Gustavo Campos de. **Processo para um Teste de *Stress* Aplicável a Sistemas de Ensino a Distância. [**Anápolis] 2021. (Universidade Evangélica de Goiás – UniEVANGÉLICA, Engenheiros de Software, 2021).

Monografia. Universidade Evangélica de Goiás, Curso de Engenharia de Software, da cidade de Anápolis-GO.

Palavras-chave: Testes de software; Testes de *stress* de software; Qualidade de software; Ferramenta para teste de *stress*; Jmeter.

REFERÊNCIA BIBLIOGRÁFICA

ABADIO, Alexandre Kamimura Vieira; CAIXETA, Davi Pereira; SILVA, Edilson Pereira da; ANDRADE, Gustavo Campos de. **Processo para um Teste de *Stress* Aplicável a Sistemas de Ensino a Distância.** Anápolis, 2021. 32 p. Monografia - Curso de Engenharia de Software Universidade Evangélica de Goiás - UniEVANGÉLICA.

CESSÃO DE DIREITOS

AGRADECIMENTOS

Primeiramente, à Deus por tornar tudo possível e nos ajudar a conquistar nossos objetivos acadêmicos em todos os anos de faculdade.

Às professoras Pollyana dos Reis e Walquiria Marins por seu tempo à disposição e ajudas constantes. Ao orientador Alexandre Tannus por suas orientações desde o começo do trabalho, que nos auxiliou a elaborar grande parte das questões técnicas necessárias.

À todas as pessoas, diretamente ou indiretamente, que contribuíram positivamente à criação desta monografia e que transformaram toda nossa caminhada de sucesso acadêmico.

RESUMO

Este trabalho busca construir um processo de testes de *stress* que possa ser utilizado para verificar a qualidade de sistemas de ensino a distância. Sendo assim, busca-se produzir um processo de teste de *stress* mais generalizado, servindo como um guia para introduzir o leitor a necessidade e o porquê em realizar os testes de *stress* em uma plataforma de aulas *on-line*. A primeira parte do trabalho trata-se de um estudo bibliográfico sobre os conceitos e fundamentos do teste de *stress*, como forma de obter conhecimentos acerca do tema. Em seguida, apresenta-se os processos de teste e sua importância durante o desenvolvimento, além de referenciar o uso da ferramenta Jmeter para realizar os testes de *stress*, além de se destacar a forma mais produtiva de se utilizar a ferramenta. Posteriormente, aplicar métricas no ambiente virtual de aprendizagem da Unievangélica, conhecido como AVA, para coletar dados acerca de suas características de qualidade referentes à sua carga limite de requisições. Por fim, como um resultado esperado, este trabalho busca desenvolver um comparativo entre o antigo e o novo Ambiente Virtual de Aprendizagem, destacando informações que ajudem a produzir um correto processo de teste de *stress* em ambientes de educação a distância.

Palavras-chave: Ambiente de educação a distância; Teste de *stress*; Jmeter.

Lista de Ilustrações

SUMÁRIO

1. INTRODUÇÃO

O século XXI trouxe um contexto de importância dos meios digitais para a sociedade como um todo. Novas tecnologias surgiram para suprir a distância entre as pessoas e seus possíveis meios de se comunicar, visto que manter um diálogo distante era imprescindível na atualidade. Em todos os segmentos do coletivo humano, a incorporação da tecnologia no mercado aumentou exponencialmente o acesso à internet, uma vez que, com isso, surgiu o mercado digital: inovação no modo como as compras são feitas, com a introdução da rede de internet para compras *on-line*.

Nesse contexto, mostra-se fundamental remediar as situações em que o acúmulo de acessos traz problemas aos sites afetados. Um sistema congestionado gera insatisfações nos seus usuários e impossibilita o serviço completo do site referenciado.

Mediante a situação acima, deve-se destacar todas as etapas da construção de um software. De acordo com Maschietto (2020), as etapas compreendem por especificação, projeto, validação e evolução. Por outro lado, um modelo de processo de software é uma forma adaptável de práticas que envolvem a construção do software. Exemplos de modelos de processo de software são o modelo cascata, o desenvolvimento evolucionário e o desenvolvimento baseado no reuso.

Levando em conta a etapa de validação, emerge-se um atributo importante para todo sistema produzido: a qualidade do software. Nesse caso, segundo Zanin (2018):

> A área de qualidade de produtos tem por objetivo garantir a qualidade do produto tecnológico gerado durante o ciclo de desenvolvimento. Para esse fim, são realizadas atividades com o objetivo de estressar as funcionalidades do sistema, identificando o comportamento dele nesse contexto. Essas atividades são chamadas de testes de software.

O objetivo dessa pesquisa é desenvolver um processo de testes de *stress* aplicável a sistemas de educação a distância visando identificar problemas a partir do alto número de conexões simultâneas. A proposta desta pesquisa é prover um processo de testes de *stress* que detecta as inconsistências do sistema no âmbito de performance e retorne dados significativos para que a equipe responsável pelo desenvolvimento da plataforma EAD consiga solucionar os problemas encontrados.

1.1. Problema da Pesquisa

A demanda por ambientes voltados à educação a distância tornou-se primordial com a evolução da possibilidade de aprendizagem remota. É indiscutível a necessidade de trabalho e estudo, aplicativos de lazer, serviços e as mais diversas atividades *on-line*. Alguns destes sistemas não estão preparados para lidar com grande volume de acessos que acontece em alguns períodos específicos, como por exemplo, um sistema acadêmico no momento de realização de provas de forma remota. Considerando um alto número de acessos simultâneos em sistemas de ensino a distância, como poderia ser um processo de teste de *stress* para reduzir os impactos do alto volume de conexões concomitantes?

1.2. Objetivos

1.2.1. Objetivo Geral:

Desenvolver um processo de teste de *stress* aplicável a sistemas de ensino a distância, a fim de reduzir os impactos do alto volume de conexões simultâneas.

1.2.2. Objetivos Específicos:

1. Identificar a necessidade de um teste de *stress*.
2. Selecionar uma ferramenta adequada para o teste de *stress*.
3. Estruturar um processo de teste de *stress*.
4. Experimentar o modelo de processo no Ambiente Virtual de Aprendizagem da UNIEVANGÉLICA (AVA).
5. Analisar os resultados e impactos causados a partir do teste a ser realizado.

1.3. Justificativa

Muitas empresas atualmente utilizam da tecnologia para automação de serviços e vendas e parte desse aumento está relacionado com a viabilização da *Internet* que compartilha rapidamente as informações (ANATEL, 2021). Com o aumento do consumo tecnológico, empresas têm investido em qualidade de sistemas que visam menor taxa de erros e que aumente a satisfação do usuário. O desempenho do sistema interfere bastante quando se fala de erros, a grande quantidade de dados recebidos e enviados por servidores pode causar certos impactos a um sistema caso não esteja preparado.

Um exemplo bem claro é o antigo serviço de educação a distância (EAD) da UNIEVANGÉLICA que, em época de prova, não suportou a grande quantidade de acessos paralelos dos estudantes. Sendo assim, os alunos não conseguiram realizar as provas no tempo

proposto, possibilitando transtornos tanto aos alunos quanto à plataforma, que recebeu um *feedback* bem negativo.

O sistema escolhido para ser utilizado como experimento de aplicação do processo foi o atual Ambiente Virtual de Aprendizagem (AVA) da Universidade Evangélica de Goiás. Um dos principais motivos de escolha desta plataforma *on-line* foi a identificação de alguns cenários de falhas que mostraram pertinentes durante a aplicação de algumas provas síncronas que vale a investigação, a fim de detectar alguns problemas específicos que a plataforma possa apresentar.

Dessa maneira, o trabalho se torna útil a equipes de teste que possam aplicar o processo de teste de *stress* em seus ambientes de trabalho para que consigam um produto final de qualidade e que não gere problemas futuros para sua empresa ou sistemas entregues. Sendo assim, é justificável apresentar uma alternativa necessária ao alto número de acessos simultâneos que possibilite ao software usado uma estabilidade de conexão e um *feedback* positivo por parte dos clientes.

1.4. Cronograma

	TCC I										TCC II											
	2021/2										2022/1											
	Ago		Set		Out		Nov		Dez		Jan		Fev		Mar		Abr		Mai		Jun	
Atividades	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2	quinzena 1	quinzena 2

Pesquisa do tema e fichamentos	X	X	X	X	X																	
Delimitação do tema e definição do problema	X																					
Entrega do Termo de Aceite e Compromisso	X																					
Identificar a necessidade de um teste de *stress*.			X	X	X																	
Selecionar uma ferramenta adequada para o teste de *stress*					X	X	X	X														
Justificativa		X	X																			

Referencial teórico			X	X	X	X	X	X													
Metodologia					X	X	X	X	X	X	X	X	X	X							
Resultados esperados							X														
Estruturar um processo de teste de *stress*									X	X	X	X	X	X	X						
Experimentar o modelo de processo no Ambiente Virtual de Aprendizagem da UNIEVANGÉLICA (AVA).															X	X	X	X	X		
Analisar os resultados e impactos causados a partir do teste															X	X	X	X	X		

a ser realizado.																					
Resultados Obtidos																			X		
Análise e Discussão																X	X	X	X		
Consideraçõe s Finais																				X	X

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Ciclo de vida do software

Nos dias atuais, praticamente todas as empresas utilizam processos e estruturas de metodologias próprias, podendo adaptar da melhor maneira que os atendem. Entretanto, existem modelos de desenvolvimento de software padrões que norteiam os projetos e estabelecem um modelo para qualquer produção de softwares no mercado. Em linhas gerais, o processo de ciclo de vida do software é dividido em: Levantamento de requisitos, Análise de requisitos, Projeto do sistema, Implementação, Testes, Implantação e Manutenção.

Primeiramente, a parte de elicitação de requisitos refere-se a compreender o problema em questão, descobrir as funcionalidades que o software terá e suas possíveis regras de negócio. “Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada.” (SOMMERVILLE, 2011, p.57).

A seguir, a etapa de Análise de requisitos é necessária para verificar se os requisitos levantados realmente são aquilo que o cliente quer. Nessa etapa, faz-se uma verificação e uma validação acerca dos modelos da etapa anterior, para que possíveis problemas nos requisitos sejam eliminados e não haja um retrabalho para refazê-los. Segundo Sommerville (2011, p.76), o custo para modificar um componente de software com problemas em uma fase final é bem maior do que consertar o erro na parte de elicitação de requisitos, visto que seria necessário alterar o projeto e a implementação, além de ser testado novamente.

Posteriormente, a parte de Projeto do sistema é uma modelagem de como funcionará internamente o software. Nesse sentido, há a definição de componentes concretos do projeto, como qual linguagem de programação será utilizada, qual sistema gerenciador de banco de dados (SGBD) será aplicado e toda a arquitetura do sistema em si. Além disso, cria-se modelos que ajudam os desenvolvedores a compreenderem todo o processo envolvido, utilizando-se diagramas UML (em português, Linguagem de Modelagem Unificada). Entre os diagramas mais utilizados estão o Diagrama de caso de usos, Diagrama de classes e Diagrama de sequência. Conforme diz Booch (2006, p.5):

> Projetos de software malsucedidos falham em relação a aspectos únicos e específicos de cada projeto, mas todos os projetos bem-sucedidos são semelhantes em diversos aspectos. Existem muitos elementos que contribuem para uma empresa de software de sucesso; um desses componentes é a utilização da modelagem.

Subsequentemente, a fase de Implementação é o desenvolvimento do software em si, a codificação de todas as funcionalidades que o sistema terá. Nessa fase, as equipes de desenvolvedores atuam separadamente ou em conjunto — de acordo com a metodologia usada — utilizando ferramentas e bibliotecas da linguagem definida anteriormente. Sendo assim, é recomendável construir um planejamento para seguir, possuir um gerenciamento de mudanças que possibilite que as alterações no sistema não impactem no restante da aplicação e seguir o cronograma definido para que a entrega ocorra dentro do prazo.

Segundo destaca Siciliano (2014), para construir um bom código, é necessário que ele não possua valores excessivos para ser mantido. Muitos dos programadores não gostam de fazer manutenções em códigos escritos por outros, visto que muitas vezes gera um retrabalho desnecessário.

Sucessivamente, encontra-se a etapa de testes, que se resume em testar todas as funcionalidades do sistema, levando em consideração a especificação do projeto. Os testes de software são uma função de controle de qualidade com um objetivo principal — descobrir erros (PRESSMAN, 2011, p.389). Sendo assim, os testes de software são importantes para garantir o controle da qualidade de software, uma vez que devem garantir que as solicitações do cliente sejam atendidas da maneira correta.

Um ponto primordial da utilização de testes de software é a minimização de custos financeiros, em razão de correções de erros que custam bem mais recursos e tempo em fases de produção do que em fases de teste. Além disso, o uso de técnicas de aprimoramento de qualidade de software e de reparo de falhas tornou-se imprescindível, já que o *feedback* de novos usuários se faz fundamental para a continuação da empresa no mercado.

Em relação à construção de testes, Gonçalves (2019, p.14) diz:

> Para a realização dos testes, utiliza-se um plano de teste, que é um documento de planejamento do projeto de teste. O plano de teste deve conter todas as etapas de validação e verificação do software a serem observadas. A validação compreende o processo de examinar se o software satisfaz ou não a necessidade do usuário. Valida-se o software que é compatível com os requisitos solicitados. Por sua vez, a verificação de software é o processo que confirma que o programa satisfaz todos os requisitos.

Outra parte do ciclo de vida de um software é a Implantação, que diz respeito à instalação correta do software no ambiente do usuário. Essa fase inclui a preparação do ambiente, o treinamento de usuários referente à utilização do sistema, a execução em si do software certificando-se de fazer correções de implantação e acompanhar o uso do sistema em produção. Cosentino (2020) diz que o treinamento de usuários impacta na produtividade de muitas empresas, dado que podem cumprir com suas obrigações sem se preocuparem com gargalos técnicos, já que foram instruídos da maneira correta sobre a usabilidade do sistema.

Definitivamente, a adoção da capacitação de usuários na fase de Implantação afeta positivamente a fase seguinte de Manutenção, uma vez que diminui o número de chamados por suporte, já que os próprios usuários podem se ajudar. Vale destacar que outro ponto positivo é referente à proteção de dados relacionada à segurança, posto que um maior

conhecimento do sistema induz uma ocorrência de falhas bem menores, reduzindo o prejuízo por ação humana dentro da empresa.

Por fim, encontra-se a etapa de Manutenção de software, na qual um processo constante de melhorias e correções de um software desenvolvido é feito. “Depois que o sistema é implantado, para que ele se mantenha útil é inevitável que ocorram mudanças. Um software criado e que não possui manutenção pode ser pouco utilizado ou até mesmo inutilizado.” (MORAIS, 2020, p. 147).

Nota-se que essa constante atualização de sistemas é altamente importante para manter o software atualizado de acordo com novas tendências. Sendo assim, a necessidade de manter a aplicação estável e em concordância com as novas tecnologias faz com que empresas delimitem tempo e recursos para melhorias e correções de erros. O *feedback* de usuários é altamente essencial nesta etapa, visto que auxilia os desenvolvedores com a percepção do que necessita ser corrigido ou acrescentado no sistema.

Schach apud Maschietto (2020, p. 246) explica que existem três tipos de modificações em software: a manutenção corretiva é quando tem-se a necessidade de se corrigir uma falha dentro do sistema que poderia distorcer seu funcionamento; por outro lado, existe a manutenção de aperfeiçoamento, que consiste em melhorar algum módulo do software para aumentar a eficácia, a velocidade ou adicionar alguma funcionalidade nova; por fim, há a manutenção adaptativa, que se resume em modificar o produto para adequar-se ao ambiente em que se encontra, a um novo sistema operacional ou a uma legislação local.

2.2. Qualidade de software

Hoje, o conceito de qualidade é fundamental para qualquer empresa. A atenção à qualidade deixou de ser um diferencial competitivo e passou a ser um pré-requisito básico para a participação no mercado. Na indústria de software, a situação não é diferente. O uso extensivo de software em todos os campos, incluindo monitoramento, controle e gerenciamento de funções-chave, aumentou a importância da qualidade do software. Na década de 1990, as empresas líderes perceberam que bilhões de dólares eram desperdiçados todos os anos em software que não fornecia os recursos e funções prometidos.

De acordo com Garvin apud Pressman (2011, p.359):

> Qualidade é um conceito complexo e multifacetado que pode ser descrito segundo cinco pontos de vista diferentes. A visão transcendental sustenta que qualidade é algo que se reconhece imediatamente, mas não se consegue definir explicitamente. A visão do usuário vê a qualidade em termos das metas específicas de um usuário final. Se um produto atende a essas metas, ele apresenta qualidade. A visão do fabricante define qualidade em termos da especificação original do produto. Se o produto atende às especificações, ele apresenta qualidade. A visão do produto sugere que a qualidade pode ser ligada a características inerentes (por exemplo, funções e recursos) de um produto. Finalmente, a visão baseada em valor mede a qualidade tomando como base o quanto um cliente estaria disposto a pagar por um produto. Na realidade, qualidade engloba todas essas visões e outras mais.

De fato, para que um software funcione da maneira esperada, é necessário que ele tenha um certo nível de qualidade. Nesse caso, segundo Garvin (1984), a coexistência dessas dimensões apresentadas faz-se necessário para se ter uma harmonia na qualidade do produto. Nesse caso, abordar uma visão em detrimento de outra pode causar brechas de qualidade e propiciar uma abertura de ação de produtos e serviços concorrentes.

Tratando-se de qualidade de software, a ISO/IEC 9126 é uma norma que estabelece parâmetros e padronizações para se medir a qualidade de um produto de software. Ela trata a qualidade externa e interna e a qualidade em uso, aplicando métricas definidas em seis atributos de qualidade principais: Funcionalidade, Confiabilidade, Usabilidade, Eficiência, Manutenibilidade e Portabilidade. Cada característica proposta possui subcaracterísticas definidas que compõem um ramo específico de mensuração da qualidade de um software.

Para melhor detalhar as divisões da ISO/IEC 9126, vale destacar as respectivas características de cada uma:

Primeiramente, a funcionalidade, que se define em ser a condição de um software de fazer o que ele realmente é proposto a fazer, de satisfazer o usuário com suas necessidades aparentes. Suas subcaracterísticas são: adequação, acurácia, interoperabilidade, segurança e conformidade.

Outra característica é a confiabilidade, onde a qualidade de um software de manter um nível de desempenho adequado mesmo em condições adversas. Seus subatributos são: maturidade, tolerância a falhas, recuperabilidade, conformidade.

Usabilidade, uma das características mais importantes, diz que um software deve ser compreensível e claro para qualquer usuário, sendo fácil de se usar. Seus subatributos são: inteligibilidade, apreensibilidade, operacionalidade, acessibilidade e conformidade.

A eficiência, tem o foco do estado de uso dos recursos e do tempo de execução de um software conforme especificado. Suas subcaracterísticas são: tempo, recursos utilizados e conformidade.

Além disso, a manutenibilidade, se resume na facilidade de se fazer manutenção corretiva ou melhorias no software. Seus subatributos são: analisabilidade, modificabilidade, estabilidade, testabilidade e conformidade.

Por fim, portabilidade, ela possui característica do software de ser movido para outro ambiente sem grandes impactos. Suas subcaracterísticas são: adaptabilidade, facilidade de instalação, facilidade de substituição e conformidade.

Nota-se que o subatributo Conformidade aparece em todas as características de qualidade de um software. Nesse caso, o subatributo mede-se o quanto o software cumpre com todo tipo de padronização no contexto inserido e com requisitos de legislação em cada característica.

O problema de pesquisa em questão leva-se em conta soluções baseadas no teste de *stress*, um tipo de teste muito utilizado em qualquer software. Nesse contexto, destaca-se os atributos de Confiabilidade e Eficiência, que serão altamente abordados na pesquisa, já que os testes de *stress* relacionam-se diretamente com estes citados.

Um software de alta qualidade agrega uma série de benefícios aos usuários e a empresa fabricante em questão. Em primeiro ponto, os usuários ganham com agilidade em seus diversos processos e atividades feitos no sistema, com uma maior resposta e entrega de seus resultados. Além disso, um software que está sempre disponível, não sofre com quedas e possui um tempo de resposta bom, gera uma satisfação a mais no cliente e, consequentemente, um *feedback* positivo. Por outro lado, a alta qualidade do software possibilita aos desenvolvedores uma menor preocupação com manutenções, correções de erros e suporte ao cliente, dedicando o tempo para aplicações novas e melhorias ao software.

2.3. Processos de teste

Para obter a qualidade do software, várias instituições criaram normas, processos e até mesmo ferramentas que realizam testes automatizados nos softwares garantindo o bom desempenho. Todas as empresas que desejam alcançar sistemas com uma boa qualidade e que se destaquem no mercado, é recomendado seguir tais normas, como a IEEE, ISO, entre outras.

A ISO *(International Organization for Standardization*) é uma federação mundial de Organismos Nacionais de Normalização, tendo somente um representante por país, objetivando facilitar as trocas internacionais de bens e serviços e desenvolver a cooperação nos campos da atividade intelectual, científica, tecnológica e econômica. (ABNT/CB-26, 2012).

A IEC (*International Electrotechnical Commission*) é a organização mundial líder que prepara e publica Normas Internacionais para as áreas elétrica, eletrônica e tecnologias relacionadas, além de disciplinas como terminologia, compatibilidade eletromagnética, performance, segurança e meio ambiente, incluindo trabalhos na otimização da eficiência energética e desenvolvimento de normas para energias renováveis (ABNT/CB-26, 2012).

A fundação IEEE (*Institute of Electrical and Electronic Engineers*) responsável por promover o conhecimento nas áreas de engenharia elétrica, eletrônica e computação, define padrões para diversas áreas e práticas presentes na engenharia de software. (MIGUEL, 2012).

De acordo com Adriana apud Oliveira (2017):

> O objetivo da ISO/ IEC/ IEEE 29119-2 é definir um processo genérico para teste de software que possa ser usado em qualquer ciclo de desenvolvimento de software. O modelo especifica processos de teste que podem ser usados para governar, gerenciar e implementar testes de software em qualquer organização, projeto ou atividade de teste.

O processo de teste é imprescindível para que o software seja fácil para a detecção e correção de erros. Com isso, para que um software atinja o potencial de qualidade do software, é necessário ter um plano de ação para que chegue a esse nível, o que direciona a estruturar um processo que se adapte a qualquer software e cumpra todas as etapas de teste.

Para definir um processo eficiente, é necessário que haja uma boa estratégia para atacar todas as partes do projeto, desde o início, no levantamento de requisitos até sua

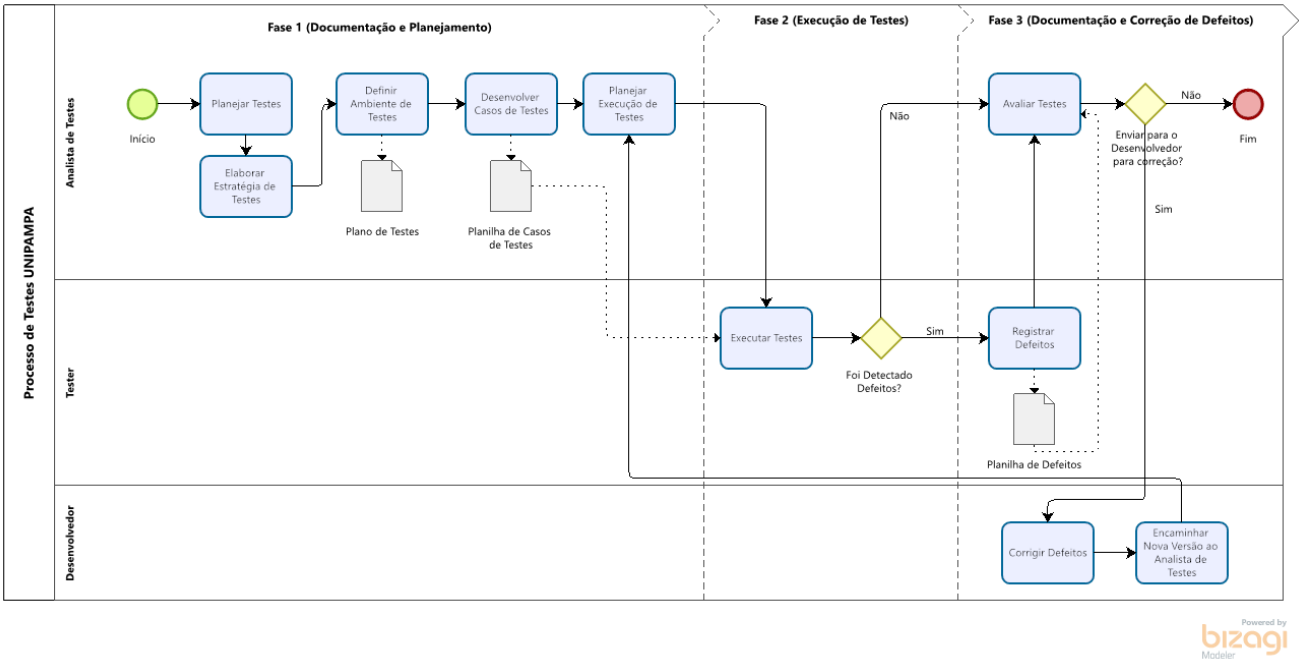
implantação final. Deve-se pensar em requisitos funcionais e não funcionais, regras para documentar e codificar, softwares para auxiliar nos testes das funcionalidades levando o sistema ao seu limite e até mesmo ultrapassá-lo.

Por fim, deve ter uma pessoa para garantir que a equipe vá seguir o processo, tal pessoa será responsável por promover treinamentos para reforçar as etapas que devem ser seguidas. E garantir que os prazos serão cumpridos.

No processo, a etapa de teste do software deve ser vista com maior cuidado, investindo em tecnologias e mão de obra qualificada, em outras palavras, tendo um processo de teste bem elaborado, recursos e ferramentas, todo software terá uma qualidade excelente, tornando erros e falhas futuras quase inexistentes.

Para isso, um processo sugerido por Primão (2012) o seguinte processo:

Figura 1 - Processo de teste UNIPAMPA



Fonte: Primão et al. 2012

A Figura 1 mostra que o processo foi dividido em 3 fases, a fase 1 é focada no planejamento e documentação para dar início aos testes. Iniciando o processo, o analista de testes deve planejar os testes que serão realizados no sistema, em seguida, elaborar uma estratégia de testes, na próxima etapa, definir ambientes de teste, deve desenvolver um documento para registrar qual será o plano de teste elaborado pela equipe, onde será seguido durante toda a vida do projeto. Em seguida, definir casos de testes deve criar uma planilha de casos de teste para executar todos os testes realizados no sistema, todos os testes são passíveis aos casos de testes. Por fim planejar a execução de testes durante o ciclo de vida do projeto.

A segunda fase tem como o objetivo a execução dos testes, nele é realizado tudo que foi definido na primeira fase. Ao terminar um teste e ser detectado um defeito ou não, prossegue para a terceira e última fase, o defeito identificado, deve ser registrado e encaminhado para ser avaliado. Caso o defeito necessite de correção, deve ser encaminhado para a equipe de desenvolvimento. Ao finalizar, deve ser validada a correção, o analista de testes deve planejar a execução de novos testes verificando se o defeito foi corrigido. Caso não tenha mais nenhum defeito, os testes serão validados e finalizados conforme o processo.

2.4. Testes Não Funcionais

Os testes não funcionais são realizados no sistema ainda no começo do desenvolvimento quando se procura encontrar defeitos rapidamente antes que seja tardio e custoso para o proprietário do sistema. Esses testes são feitos para avaliar características do sistema como usabilidade, performance ou segurança definindo sobre o quão bem o sistema se comporta durante o seu uso. Podendo ser utilizado em todo e qualquer nível de teste (ISTQB, 2018).

Dentre as vantagens de se realizar teste de performance, algumas delas pode ser definida como a melhoria da qualidade do produto do ponto de vista de usuário, como por exemplo a redução dos custos de sistema e a identificação antecipada dos defeitos mais críticos da aplicação como arquitetura do sistema que levam a economizar em tempo para a produção e entrega do sistema (da Silva, 2019).

Freitas et al. (2010) diz que o teste de performance verifica se a aplicação atende as especificações do sistema e consegue retornar como resultado o tempo de exibição da página, tempo de consulta de um relatório até mesmo o número de acessos simultâneos no sistema. Tendo como objetivo não encontrar erros e sim eliminar gargalos de desempenho.

De acordo com a ISTQB – *Performance Testing*, existem vários tipos de teste de performance dentre eles o teste de *stress*, de carga e de capacidade que são definidos como:

Teste de *stress* é responsável por testar até onde o sistema consegue tolerar os picos de grandes números de acessos, sejam eles no limite ou excedentes. O teste também é utilizado para avaliar a capacidade do sistema em lidar com a falta de capacidade de processamento do computador utilizado e até mesmo a internet e memória RAM.

Teste de carga é realizado a fim de verificar a capacidade do sistema em suportar inúmeras transações geradas pela quantidade de usuários ativos. Teste de capacidade é utilizado para identificar gargalos exibindo a capacidade de quantos usuários ou transações o sistema será capaz de suportar.

Existem algumas ferramentas responsáveis pela execução de testes de *stress* que executam os testes de modo automatizado. Dentre as ferramentas existentes, algumas recomendadas para utilização são (Pathak, 2018):

A ferramenta LoadRunner foi desenvolvida pela HP e é bastante utilizada para testes de *loading.* O JMeter é uma ferramenta de teste *open source* desenvolvida pela *Apache Foundation,* utilizada para realizar uma vasta quantidade de testes, como por exemplo carregamento, *stress* e funcionalidades. A ferramenta LoadNinja é utilizada para identificar problemas de desempenho em aplicativos e permite registrar interações juntamente ao cliente para identificar problemas rapidamente.

O teste de *stress* é mais utilizado quando se precisa avaliar se o sistema está apto para receber grandes cargas de pessoas logadas ao mesmo tempo, podendo influenciar em uma má interação entre sistema-usuário.

Algumas métricas são utilizadas para mensurar a qualidade dentro dos testes realizados, como por exemplo falhas de conexão onde mostra a quantidade de conexões recusadas pelo cliente ou número de tentativas que falharam. Outros exemplos que podem ser citados são: o tempo de carregamento de uma tela ou imagem, ou o tempo necessário para carregar todas as informações contidas na página (PrimeControl, 2018).

A escalabilidade e performance do sistema também influenciam no momento do teste para mensurar métricas de teste de *stress*. Exemplificando algumas métricas: a quantidade de páginas que foram requisitadas, o tamanho dos dados de resposta solicitados e a execução em que exibe o número de vezes que os cenários de testes foram planejados versus o número de vezes que o cliente foi executado (PrimeControl, 2018).

2.5. Teste de stress x Teste de carga

Apesar do significado ser relativamente parecido entre *stress* e carga, existem diferentes subtipos destes testes. Entre eles temos o **teste de carga baixa** definido como um subtipo do teste de carga, que consiste em submeter o sistema a diversas cargas abaixo do esperado avaliando se o sistema responde como o esperado.

Já o **teste de carga típica** avalia como o sistema funciona sobre cargas esperadas e típicas do dia a dia do *software.* Outro subtipo do teste de carga, o **teste de pico de carga** ou *peak load* é utilizado para testar o sistema em condições esperadas a picos de utilização altos.

De modo geral, o teste de carga é utilizado para testar o sistema em condições esperadas. Já o teste de *stress* é empregado para extrapolar os limites que o sistema suporta. O resultado esperado após realizar um teste de *stress,* de acordo com Souza (2018) é a queda do servidor HTTP e outros resultados podem ser apresentados durante o teste, mas deve-se atentar durante a leitura dessas anormalidades.

O teste de *stress* também apresenta subtipos como o **teste de *stress* típico,** que deve ser utilizado para aplicar cargas acima do esperado no sistema e finalizado quando o sistema começa a apresentar anomalias, podendo ser responsável por definir o ponto de quebra do sistema.

O **teste de pico de *stress*** é empregado para identificar o momento em que o sistema começa a apresentar falhas com um aumento repentino de carga. Também pode ser utilizado para verificar a recuperação do sistema caso não suporte o aumento significativo de carga. Por último, o **teste de *stress* com recursos insuficientes,** que consiste em aplicar testes no sistema conforme uma configuração de *hardware* muito abaixo do esperado para utilização da aplicação.

2.6. Jmeter

A ferramenta JMeter criada em 2007 foi projetada para realizar testes não funcionais em sistemas *web* que possibilita testar o comportamento de um programa e medir seu desempenho. Idealizado pela empresa Apache Software Foundation, o aplicativo é um *software* de código aberto desenvolvido 100% em linguagem Java.

O aplicativo ajuda o testador a identificar os pontos mais críticos do sistema simulando diferentes cargas em cima do *software* em teste. Bem como Halili (2008) pode se dizer que as ferramentas gráficas disponibilizadas pelo JMeter auxiliam o testador a ter uma melhor análise do resultado do teste, seja ele de *stress* ou desempenho tendo uma apuração mais precisa e a certeza de que a aplicação está retornando resultados seguros e confiáveis.

Essa ferramenta também pode ser usada para execução de testes funcionais pois existem vários tipos de modelos que podem ser empregados para refinar resultados recebidos

por testes, sendo possível adotar expressões regulares para facilitar o uso da ferramenta (Carvalho et al. 2014).

Conforme é feita a execução dos testes com a ferramenta JMeter ela pode ser realizada de duas formas como define Santos et al (2008): “em uma máquina só ou de forma distribuída, na qual o esforço do teste é distribuído dentre diversas máquinas”. A divisão dessas partes pode ser muito importante para o resultado do teste, já que é através dele que os analistas podem recriar os cenários e ter resultados de teste mais sólidos evitando diferenças entre testes já realizados.

A ferramenta JMeter também consegue executar testes em modo de gravação que permite o usuário da aplicação gravar passo a passo do que a ferramenta poderá reproduzir para concluir o teste. Essa ferramenta de gravação fica disponível no navegador do analista e pode ser instalada por extensão. Após instalado e gravado os passos a extensão gera um script de teste que pode ser executado diretamente na ferramenta, aplicando as métricas do teste configurado diretamente na aplicação (Apache Foundation, 2021).

2.7. Trabalhos Relacionados

Chaves (2019) fez um trabalho de conclusão de curso no curso de Engenharia de Computação da UniEvangélica no qual foram utilizados testes de performance aplicados a um dos sistemas desenvolvidos pela Fábrica de Tecnologias Turing (FTT): o Virtoo. Foi usada a ferramenta Jmeter na aplicação dos testes no sistema, os quais retornaram dados de desempenho e erro referentes às métricas que foram empregadas. Chaves (2019) aplicou o teste em dois requisitos do sistema e acompanhou junto à equipe os resultados obtidos do Jmeter. Posteriormente, foram disponibilizados workshops e materiais de apoio para a equipe da FTT poder aplicar a ferramenta Jmeter no software Virtoo.

Como consideração final de seu trabalho acadêmico, Chaves (2019) concluiu que os testes de desempenho realizados contribuíram para a qualidade final do projeto Virtoo, uma vez que foi possível identificar o comportamento dos requisitos em determinadas situações de uso, suportando ou não as requisições esperadas.

3. METODOLOGIA DA PESQUISA

Este trabalho expõe a necessidade da qualidade e dos testes de software que buscam minimizar a quantidade de erros presentes e precaver erros futuros. A segunda proposta busca identificar e escolher uma ferramenta de teste que automatize e auxilie na identificação de erros referentes a falhas de *performance* no sistema, visando a diminuição da necessidade de realização de testes manuais que possam retardar a entrega do produto final.

O terceiro ponto desta pesquisa é propor um processo de teste de *stress,* para que seja possível identificar defeitos relacionados a um mau desempenho do sistema. Sendo assim, é necessário que consiga expor métricas para compreender melhor gargalos no sistema e que assim seja possível solucionar problemas encontrados.

Ao finalizar a criação do teste, o processo deve ser aplicado em um sistema *web* da Universidade Evangélica de Goiás chamado de Ambiente Virtual de Aprendizagem (AVA) a fim de identificar inconsistência com a simultaneidade de vários usuários logados ao mesmo tempo. Para que seja possível a execução do teste no sistema, utiliza-se a ferramenta Jmeter, que consiste em simular uma vasta quantidade de usuários acessando o sistema simultaneamente e retornando alguns pontos de melhoria.

Para realizar as pesquisas do trabalho e aprofundar ainda mais o tema, foram utilizadas pesquisas bibliográficas, como *e-books,* portais de periódicos e revistas eletrônicas com reconhecimento científico. Neles, foram utilizadas as palavras-chaves: a) Processo de Teste de Stress; b) Teste De Stress; c) Processo de Teste; d) Ferramentas de Testes Não Funcionais. Para Gil (2002) “A pesquisa bibliográfica é desenvolvida com base em material já elaborado, constituído principalmente de livros e artigos científicos.”

Como método de pesquisa, emprega-se o modo qualitativo que, de acordo com Lando (2020), “pesquisa qualitativa é uma abordagem que pressupõe que o significado dado ao fenômeno é mais importante que sua quantificação.” Tendo isso em vista, o conteúdo apresentado neste trabalho baseia-se na análise de documentos que podem ser gerados por interfaces gráficas de um aplicativo de automação de testes de performance.

4. RESULTADOS ESPERADOS

Primeiramente no ciclo do trabalho, temos como resultados esperados expor a necessidade de um sistema com qualidade juntamente com a necessidade de testes em um site. Outro ciclo define que um sistema em desenvolvimento necessita de um processo de teste, mesmo que seja pouco complexo. Desse modo, um processo de teste aplicável a sistemas de ensino a distância será produzido neste trabalho.

No próximo ciclo tem o objetivo de empregar os conhecimentos adquiridos durante os estudos, aplicando no atual Ambiente Virtual de Aprendizagem a ferramenta Jmeter, que consiste em executar testes de *stress* para encontrar falhas que possam prejudicar a experiência do usuário durante o uso do sistema em dias que o sistema fica sobrecarregado pelo número de transações.

Posteriormente, aplicar os testes de *stress* na antiga plataforma EAD da UniEvangélica e produzir um comparativo entre as duas plataformas, ressaltando as métricas de avaliação e retornando dados que serão usados para verificar a qualidade de desempenho das duas plataformas.

Por fim, o último ciclo do trabalho apresenta que, quando o resultado é aplicado corretamente, o sistema tende a diminuir problemas futuros e custos de produção, assim deixando o *software* com uma qualidade superior e diminuindo retrabalho.

5. REFERÊNCIAS

ABNT/CB-26; **IEC**. 2012. Disponível em: https://www.cb26.org.br/iec. Acesso em: 12 de novembro de 2021.

ABNT/CB-26; **ISO**. 2012. Disponível em: https://www.cb26.org.br/iso. Acesso em: 12 de novembro de 2021.

ANATEL. In: **MINISTÉRIO DAS COMUNICAÇÕES, Gov. Anatel: Sumário dos Relatórios** Publicados de 2021. [S. l.], 14 jan. 2021. Disponível em: https://www.gov.br/anatel/pt-br/dados/acompanhamento/relatorios-de-acompanhamento/2021#R2021_5. Acesso em: 21 de maio de 2021.

APACHE, S. F. **Apache Foundation**. 2019. Disponível em: https://www.apache.org/foundation/thanks.html. Acesso em: 22 de setembro de 2021.

APACHE, S. F. **Apache Foundation**. 2019. Disponível em: http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html. Acesso em: 22 de setembro de 2021.

BLANCO, Mariana. **Documentação de Teste Baseado na Norma IEEE 829 - Estudo de Caso: "Sistema de Apoio a Tomada de Decisão"** , Tecnologias, Infraestrutura e Software, ano 1, v. 1, n. 1, ed. 1, p. 91 - 97, jul 2012. Disponível em: http://revistatis.dc.ufscar.br/index.php/revista/article/view/18. Acesso em: 2 de outubro de 2021.

BOOCH, Grady. **UML: guia do usuário**. ed. 2. Brasil: Elsevier Brasil, 2006, 2006. 474 p. v. 2. ISBN 8535217843, 9788535217841.

CARVALHO, Fabiano Silva de. BARROS, Marissol Martins. FILHO, José Inácio Ferreira. PEDROSA, Danilo Luiz Souza. COSTA, Ronaldo Alves da. **Testes não funcionais com apoio da ferramenta JMeter**. Disponível em: http://www.revista.universo.edu.br/index.php?journal=1reta2&page=article&op=view&path%5B%5D=1401&path%5B%5D=1036. Acesso em: 10 de outubro de 2021.

CHAVES, Nayara Marcela. **Uma abordagem de aplicação de teste de performance na Fábrica de Tecnologias Turing - Unievangélica**. 2019. TCC (Graduação) – Curso de Engenharia de Computação, Centro Universitário de Anápolis - Unievangélica, Anápolis, 2019. Disponível em: http://repositorio.aee.edu.br/bitstream/aee/16995/1/2WaNayara.pdf. Acesso em: 13 de dezembro de 2021.

COSENTINO, Dorian. **Treinamento do usuário no uso de softwares: qual a importância e como fazer?** 2020. Disponível em:https://gdsolutions.com.br/gestao-de-ti/treinamento-usuario/. Acesso em: 16 de novembro de 2021.

CRESPO, Adalberto; SILVA, Odair; BORGES, Carlos; SALVIANO, Clênio; JUNIOR, Miguel; JINO, Mario. **Uma Metodologia para Teste de Software no Contexto da Melhoria**

de Processo, UNICAMP - Campinas - SP - Brasil, 2004. Disponível em: https://www.researchgate.net/profile/Adalberto-Crespo-2/publication/237497188_Uma_Metodologia_para_Teste_de_Software_no_Contexto_da_Melhoria_de_Processo/links/54e5d1040cf2cd2e028b338b/Uma-Metodologia-para-Teste-de-Software-no-Contexto-da-Melhoria-de-Processo.pdf. Acesso em: 25 de outubro de 2021.

FREITAS, Fabrício Gomes de. MAIA, Camila Loiola Brito. CAMPOS, Gustavo Augusto lima de. SOUZA, Jefferson Teixeira de. **Otimização em Teste de Software com Aplicação de Metaheurísticas**. 2010. Disponível em: http://www.fsma.edu.br/si/edicao5/FSMA_SI_2010_1_Estudantil_1.pdf. Acesso em: 20 de setembro de 2021.

GARVIN, D.A. **What Does "Product Quality" Really Mean?** MIT Sloan Management Review, Cambridge, MA, v. 16, n. 1, 1984. Disponível em: <https://sloanreview.mit.edu/article/what-does-product-quality-really-mean/>. Acesso em: 29 de novembro de 2021.

GIL, A. C. **Como elaborar projetos de pesquisa**. São Paulo, SP: Atlas, 2002.

GONÇALVES, Priscila.de. F.; BARRETO, Jeanine.dos. S.; ZENKER, Aline. M.; AL., et. **Testes de software e gerência de configuração.** Porto Alegre: Grupo A, 2019. 9788595029361. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595029361/. Acesso em: 15 de novembro de 2021.

HALILI, Emily H. Apache JMeter: **A practical begnner's guide to automated testing and performance measurement for you websites**, 2008. Disponível: http://download.51testing.com/ddimg/uploadsoft/20131113/ApacheJMeter_English.pdf. Acesso: 14 de outubro de 2021.

HEGDE, Vinayak e PALLAVI. **Web Performance Testing: Methodologies, Tools and Challenges**, 2014 .

ISTQB. **Certified Tester: Foudation Level Syllabus**. 2018. Disponível em: https://bstqb.org.br/b9/doc/syllabus_ctfl_2018br.pdf. Acesso em: 18 de setembro de 2021.

LANDRO, Felipe; **Método de pesquisa qualitativa: O que é e como fazer?**, 2020; Disponível em: https://www.academicapesquisa.com.br/post/método-qualitativo-como-fazer. Acesso em: 21 de novembro de 2021.

MASCHIETTO, Luís. G.; RODRIGUES, Thiago. N.; BIANCO, Clicéres.M. D; AL., et. **Processos de Desenvolvimento de Software**. Porto Alegre: Grupo A, 2020. 9786556900520. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786556900520/. Acesso em: 01 de novembro de 2021.

MIGUEL, Sérgio. **Padrão para Documentação de Teste de Software**, [*s. l.*], 2012. Disponível em:

https://www.devmedia.com.br/padrao-para-documentacao-de-teste-de-software/26534. Acesso em: 3 de outubro de 2021.

MORAIS, Izabelly.Soares. D.; ZANIN, Aline. **Engenharia de software.** Porto Alegre: Grupo A, 2020. 9788595022539. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595022539/. Acesso em: 16 de novembro de 2021.

OLIVEIRA, Adriana Ferreira. **Um estudo sobre a aplicabilidade da série de padrões ISO/IEC/IEEE 29119 em métodos ágeis de desenvolvimento de software**, 2017. Disponível em: http://www.monografias.ufop.br/handle/35400000/625. Acesso em: 11 de novembro de 2021.

PATHAK, Amrita. **As 27 Melhores Ferramentas de Teste de Desempenho a Serem Utilizadas em 2021**. 2021. Disponível em: https://kinsta.com/pt/blog/ferramentas-teste-desempenho/. Acesso em: 5 de outubro de 2021.

PRESSMAN, Roger S. **Engenharia de software: uma abordagem profissional**. ed. 7, Local de publicação: McGraw-Hill, 2011

PRIMÃO, Aline. DA, Patric. RIBEIRO, Silva. KREUTZ, Diego. (2012). **Estudo de Caso: Técnicas de Teste como parte do Ciclo de Desenvolvimento de Software**. Disponível em: https://www.researchgate.net/figure/Figura-1-Processo-de-testes-de-software-da-instituicao_fig1_267841817. Acesso em: 13 de novembro de 2021.

PRIMECONTROL, Quality Drives Results. **Testes de estresse e carga: conheça os tipos e ferramentas**. 2018. Disponível em: https://www.primecontrol.com.br/testes-de-estresse-e-carga-conheca-os-tipos-e-ferramentas/. Acesso em: 20 de setembro de 2021.

SANTOS, Ismayle de Sousa. Neto, Pedro de Alcântara dos Santos. **Automação de testes de desempenho e estresse com JMeter**. Disponível em: http://lmialaret.orgfree.com/artigoJMeter.pdf. Acesso em: 14 de outubro de 2021.

SILVEIRA, Maicon Bernardinho da. **Conjunto de características para teste de desempenho: uma visão a partir de modelos**. 2012. Disponível em: http://tede2.pucrs.br/tede2/handle/tede/5187. Acesso em: 27 de outubro de 2021.

SICILIANO, Leandro Tavares. **Boas práticas de programação**. 2014. Disponível em: https://www.devmedia.com.br/boas-praticas-de-programacao/31163. Acesso em: 14 de novembro de 2021.

SOMMERVILLE, Ian**; Engenharia de Software;** tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; Revisão técnica Kechi Hirama - 9. ed. - São Paulo: Pearson Prentice Hall, 2011.

SOUZA, Thiago Silva de. **Testes de desempenho de software**: Teoria e Prática. 2018. Disponível em: https://pdfs.semanticscholar.org/7f61/2fa8898aacc8f984e12f504d528374810480.pdf. Acesso em: 15 de outubro de 2021.

SUBRAYA, B.M. **Integrated Approach to Web Performance Testing: A Practitioner's Guide**. 2006.

VIEGAS, Júlio. **Teste de Software: Introdução, Conceitos Básicos e Tipos de Teste.** 2019. Disponível em: https://blog.onedaytesting.com.br/teste-de-software/ . Acesso em: 15 de outubro de 2021.

ZANIN, Aline. JÚNIOR, Paulo.A. P.; ROCHA, Breno. C.; AL., et. **Qualidade de software**. Grupo A, 2018. 9788595028401. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595028401/. Acesso em: 01 de novembro de 2021.

6. ANEXOS

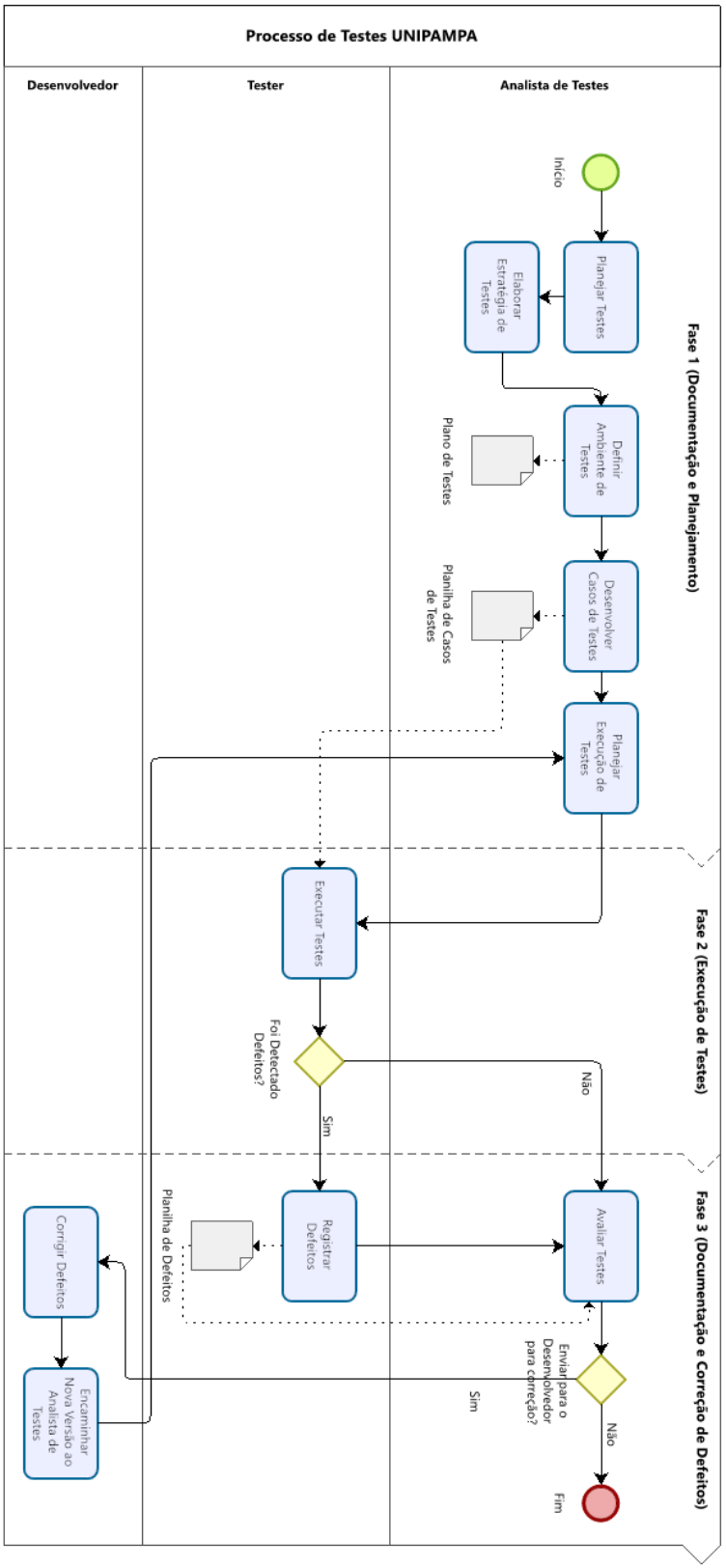


Figura 1 - Processo de teste UNIPAMPA (Primão et al. 2012)