

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 Teste de *Stress* Aplicável a Sistemas de Ensino a Distância

Anápolis  
Junho, 2022

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 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. Ma. Walquiria Fernandes Marins.

Anápolis  
Junho, 2022

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 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:**

---

Me. Walquiria Fernandes Marins.

**ORIENTADOR**

---

Me. Natasha Sophie Pereira

**AVALIADOR**

---

Me. Viviane Carla Batista Pocivi

**AVALIADOR**

**Anápolis, 21 de junho de 2022**

## FICHA CATALOGRÁFICA

ABADIO, Alexandre Kamimura Vieira; CAIXETA, Davi Pereira; SILVA, Edilson Pereira da; ANDRADE, Gustavo Campos de. **Processo para Teste de *Stress* Aplicável a Sistemas de Ensino a Distância.** [Anápolis] 2022. (Universidade Evangélica de Goiás – UniEvangélica, Engenheiros de *Software*, 2022).

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*; Testes Não Funcionais; Processo De Testes Não Funcionais.

## REFERÊNCIA BIBLIOGRÁFICA

ABADIO, Alexandre Kamimura Vieira; CAIXETA, Davi Pereira; SILVA, Edilson Pereira da; ANDRADE, Gustavo Campos de. **Processo para Teste de *Stress* Aplicável a Sistemas de Ensino a Distância.** Anápolis, 2022. 65 p. Monografia - Curso de Engenharia de *Software* Universidade Evangélica de Goiás - UniEvangélica.

## CESSÃO DE DIREITOS

NOMES DOS AUTORES: Alexandre Kamimura Vieira Abadio, Davi Pereira Caixeta, Edilson Pereira da Silva, Gustavo Campos de Andrade.

TÍTULO DO TRABALHO: Processo para Teste de *Stress* Aplicável a Sistemas de Ensino a Distância.

GRAU/ANO: Graduação / 2022

É concedida à Universidade Evangélica de Goiás - UniEvangélica, permissão para reproduzir cópias deste trabalho, emprestar ou vender tais cópias para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste trabalho pode ser reproduzida sem a autorização por escrito do autor.

---

Alexandre Kamimura Vieira Abadio  
Anápolis, 21 de junho de 2022.

---

Davi Pereira Caixeta  
Anápolis, 21 de junho de 2022.

---

Edilson Pereira da Silva  
Anápolis, 21 de junho de 2022.

---

Gustavo Campos de Andrade  
Anápolis, 21 de junho de 2022.

## **AGRADECIMENTOS**

Primeiramente, à Deus por tornar tudo possível, mesmo com todas as dificuldades enfrentadas durante o desenvolvimento desta pesquisa e nos ajudar a conquistar nossos objetivos acadêmicos em todos os anos de faculdade.

À professora e orientadora Walquiria Marins por seu tempo à disposição e ajudas constantes, que nos auxiliou a elaborar grande parte das questões técnicas e teóricas necessárias. Ao professor Eduardo que nos forneceu todas informações, materiais e recursos necessários para execução dos testes. Ao Israel, líder técnico da FTT que nos permitiu executar os testes da ferramenta no sistema Virtoo e ofereceu todo suporte necessário.

Às avaliadoras da nossa banca Viviane e Natasha pelas considerações e elogios e à todas as pessoas, diretamente ou indiretamente, que contribuíram positivamente à criação deste trabalho e que transformaram toda nossa caminhada de sucesso acadêmico.

## RESUMO

Este trabalho propõe a construção de um processo de teste de *stress* que possa ser utilizado para verificar a qualidade de serviços de ensino a distância (EAD). Desse modo, faz-se a elaboração de um processo de teste de *stress* geral, como um guia para introduzir ao leitor a necessidade de se realizar testes de *stress* em uma plataforma de aulas, conteúdos e avaliações *on-line*. Inicialmente, o trabalho conduz um estudo bibliográfico sobre os conceitos e fundamentos do teste de *stress*. Em seguida, apresenta-se os processos de teste e sua importância durante o desenvolvimento, destaca-se a ferramenta mais viável para realizar os testes de *stress* e ressaltase a forma mais produtiva de se utilizar a ferramenta. Como exemplo de sistemas de ensino a distância, os ambientes virtuais de aprendizagem da Unievangélica, conhecido como AVAs, foram escolhidos como alvos de aplicação do processo de testes, para a obtenção de métricas de teste de *stress*. Posteriormente, com a realização dos testes, foi feita a coleta de dados das características de qualidade de cada plataforma, referentes à carga limite de requisições e tempo de resposta. Por fim, este trabalho propõe um processo de teste de *stress* padronizado e extensível a ser aplicado em ambientes para EAD.

**Palavras-chave:** Ambiente de Educação a Distância; Teste de *Stress*; Teste de Performance; Ferramenta de Teste de *Stress*.

## **ABSTRACT**

This work proposes the creation of a stress test process that can be used to verify the quality of distance learning platforms. Thereby, a general stress test process is developed, as a guide to introduce the reader to the necessity to perform stress tests on a platform for classes, contents and on-line exams. Initially, the work brings a bibliographic study about the concepts and fundamentals of the stress test. Then, the test processes and their importance during development are presented, the most viable tool to perform stress tests is highlighted and the most productive way to use this tool is emphasized. As an example of distance learning systems, Unievangélica's virtual learning platform, known as AVAs, were chosen as targets for the application of the testing process, to obtain stress test metrics. Subsequently, with the tests production, data about the quality characteristics of each platform, referring to the limit load of requests and response time, was collected. Finally, this work proposes a standardized and extensible stress test process to be applied in distance learning services.

**Palavras-chave:** Distance Learning Platforms; Stress Test; Performance Test; Stress Test Tool.

## LISTA DE ILUSTRAÇÕES

<b>Imagem 1</b> - Download do Script com BlazeMeter.....	25
<b>Imagem 2</b> - Quantidade de Matrículas na modalidade EAD no período 2010-2020 .....	27
<b>Imagem 3</b> - Protótipo do Processo. ....	31
<b>Imagem 4</b> - Grupo de usuários.....	32
<b>Imagem 5</b> - Relatório de Sumário. ....	33
<b>Imagem 6</b> - Erro no 1º teste (AVA v.2).....	35
<b>Imagem 7</b> - Gerenciador de tarefas que indica problemas de rede. ....	36
<b>Imagem 8</b> - Relatório de Sumário do 2º teste (AVA v.2).....	36
<b>Imagem 9</b> - Relatório de Sumário 3º teste (AVA v.2).....	37
<b>Imagem 10</b> - Relatório de Sumário 4º teste (AVA v.2).....	38
<b>Imagem 11</b> - Relatório de Sumário 5º teste (AVA v.2).....	38
<b>Imagem 12</b> - Relatório de Sumário 6º teste (AVA v.2).....	38
<b>Imagem 13</b> - Relatório de Sumário 7º teste (AVA v.1).....	39
<b>Imagem 14</b> - Relatório de Sumário 8º teste (AVA v.1).....	40
<b>Imagem 15</b> - Relatório de Sumário 9º teste (AVA v.1).....	40
<b>Imagem 16</b> - Relatório de Sumário 10º teste (AVA v.1).....	41
<b>Imagem 17</b> - Processos de Testes Não Funcionais .....	44
<b>Imagem 18</b> - Processo de teste de performance. ....	45
<b>Imagem 19</b> - Processo de teste de stress.....	45
<b>Imagem 20</b> - Processo de teste UNIPAMPA .....	54
<b>Imagem 21</b> - Processo Geral de Teste .....	55



## **LISTA DE TABELAS**

<b>Tabela 1</b> - Quantidade de Usuário Matriculados até Junho de 2022.....	34
<b>Tabela 2</b> - Configuração da máquina 1. ....	35
<b>Tabela 3</b> - Configuração da máquina 2. ....	36
<b>Tabela 4</b> - Configuração da máquina 3. ....	37
<b>Tabela 5</b> - Resultados Gerais dos Dois Ambientes. ....	43

# Sumário

1. INTRODUÇÃO .....	11
1.1. Problema da Pesquisa.....	12
1.2. Objetivos .....	12
Objetivo Geral: .....	12
Objetivos Específicos: .....	12
1.3. Justificativa.....	12
2. FUNDAMENTAÇÃO TEÓRICA .....	14
2.1. Ciclo de Vida do <i>Software</i> .....	14
2.2. Qualidade de <i>Software</i> .....	17
2.3. Processos de Teste .....	19
2.4. Testes Não Funcionais .....	21
2.5. Teste de <i>Stress</i> vs. Teste de Carga.....	22
2.6. Ferramentas de Teste de Performance .....	23
2.6.1. JMeter: .....	24
2.6.2. BlazeMeter: .....	24
2.7. Trabalhos Relacionados .....	25
2.8. Ambientes Virtuais de Aprendizagem (AVAs).....	26
2.8.1. AVA Versão 1 (2020/1 – 2021/1) .....	27
2.8.2. AVA Versão 2 (2021/2 – atual) .....	27
3. METODOLOGIA DA PESQUISA .....	29
4. ABORDAGEM PROPOSTA.....	31
4.1. Protótipo do Processo .....	31
4.2. Jmeter e BlazeMeter.....	32
5. RESULTADOS.....	34
5.1. AVA Versão 2.....	34
5.2. AVA Versão 1.....	39
5.3. Resultados Gerais Dos Testes.....	41
5.3.1. Comparativo Entre As Duas Versões Do Ava.....	41
5.4. Processo de Teste .....	43
6. CONSIDERAÇÕES FINAIS.....	47
7. REFERÊNCIAS .....	49
8. ANEXOS.....	54
9. APÊNDICES.....	55

## 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 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 evitar situações em que o acúmulo de acessos traz problemas aos sites afetados, dado que um sistema congestionado gera insatisfações nos seus usuários e impossibilita o serviço completo do site referenciado. Percebe-se que existem serviços *on-line* que são essenciais e não podem tornar-se indisponíveis em momentos críticos, como é o caso de sistemas de educação a distância (EAD) em época de avaliações.

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 que visa 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 acontecem em determinados períodos, como por exemplo, um sistema acadêmico no momento de realização de provas na modalidade EAD. Considerando um alto número de acessos simultâneos em situações adversas, como um processo de teste de *stress* seria aplicado para reduzir os impactos causados por conexões concomitantes em sistemas de ensino a distância?

### 1.2. Objetivos

#### 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.

#### Objetivos Específicos:

1. Analisar situações de indisponibilidade ou *stress* do sistema;
2. Selecionar uma ferramenta adequada para o teste de *stress*;
3. Estruturar um protótipo de processo de teste de *stress*;
4. Executar os testes de *stress* em Ambientes Virtuais de Aprendizagem (AVAs);
5. Analisar os resultados e impactos causados a partir do teste realizado;
6. Evidenciar a necessidade de um teste de *stress*;
7. Estruturar um processo de teste de *stress* aplicável a sistemas EAD.

### 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, pois 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 primeiro sistema de educação a distância (EAD) da mantenedora Associação Educativa Evangélica (AEE), cuja sua Universidade Evangélica de Goiás (UniEvangélica) que, em época de prova, não suportou a grande quantidade de acessos paralelos dos acadêmicos (SOUZA, E. 2022). 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.

Os sistemas escolhidos para serem utilizados como experimento de aplicação do processo foram o atual Ambiente Virtual de Aprendizagem (AVA) e a primeira versão do AVA, ambos da Universidade Evangélica de Goiás. Um dos principais motivos de escolha destas plataformas foi a identificação de alguns cenários de falhas que mostraram pertinentes durante a aplicação de algumas provas síncronas que valem 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 usuários finais.

## 2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção da pesquisa serão exibidos os conceitos e assuntos relevantes e fundamentais para a construção do trabalho. Sendo assim, será abordado conteúdos sobre ciclo de vida do *software*, qualidade de *software*, processos de teste, testes não funcionais e ferramentas de testes de performance.

### 2.1. Ciclo de Vida do *Software*

Um ciclo de vida de um software abrange todas as etapas da construção de um sistema, desde os processos, atividades e tarefas que são importantes para produzir e finalizar um software. Para exemplificar, destaca-se os diversos modelos de ciclo de vida tradicionais existentes no mercado, tais como modelo Cascata, Evolutivo, Espiral e Modelo em V. Por outro lado, alguns modelos de ciclo de vida ágeis que estão em alta são: Scrum, Kanban, XP e OpenUP. Nos dias atuais, praticamente todas as empresas utilizam processos e estruturas de metodologias próprias, podendo adaptar da melhor maneira que os atendem.

Existem modelos de desenvolvimento de software padrões que estabelecem fases genéricas que são adotadas em toda construção de software no mercado. O modelo Cascata é um exemplo de um processo de construção de software tradicional, sendo um dos mais utilizados do mercado. Segundo Maschietto (2020, p.14), o modelo Cascata possui “apresentação de fases bem definidas e documentação constante e presente em todas as etapas do processo.” Em linhas gerais, em modelos tradicionais existem 3 fases básicas de um ciclo de software: definição, desenvolvimento e operação. (GUEDES, 2018).

A etapa de definição está relacionada na identificação do problema, busca de solução, modelagem de processos e análise do sistema. Em contrapartida na etapa de desenvolvimento, é feito o design, prototipagem, codificação, teste e entre outras etapas necessárias de acordo com a solução já definida. Por fim, a etapa de operação é a qual o software já estará em produção e deve ser feito o suporte corrigindo possíveis falhas e promovendo treinamentos ao usuário (GUEDES, 2018).

O modelo Ágil, diferentemente do tradicional, é uma abordagem incremental e iterativa onde cada iteração é dividida em miniprojetos, com a duração pré-definida incluindo todas as fases para implementá-lo, como: Levantamento de requisitos, projetos, desenvolvimento, teste e documentação. Ao final de cada iteração é feita uma entrega de uma nova versão ao cliente com as novas funcionalidades (MACORATTI, 2017).

O Manifesto para o Desenvolvimento Ágil de Software foi assinado em 2001 e trouxe a valorização de alguns pontos dentro da construção de um sistema. São eles: Indivíduos e interações acima de processos e ferramentas; Software operacional acima de documentação completa; Colaboração dos clientes acima de negociação contratual; Respostas a mudanças acima de seguir um plano (PRESSMAN, 2011, p.81).

Em linhas gerais, os ciclos de vida possuem um processo dividido em: Fase de requisitos, projetos, implementação, testes e produção. A primeira fase, na parte de levantamento 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 etapa de Análise de requisitos, ainda na fase 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.

A fase 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. Tal esforço gera tempo para ser corrigido, em razão da falta de conhecimento sobre o código escrito ou a sua desorganização.

Sucessivamente, encontra-se a fase 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 já na fase de produção, 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, ainda na fase de produção, 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: A primeira destacada é 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. Além disso, a 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.

Outro atributo é a eficiência, que tem o foco no estado de uso dos recursos e no 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, a portabilidade é a 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 dessa pesquisa em questão leva-se em conta soluções baseadas no teste de *stress*, um tipo de teste muito importante 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, conseqüentemente, 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 se 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 ISO e a IEEE. 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 seja fácil a detecção e correção de erros no *software*. Com isso, para que um *software* atinja o potencial de qualidade do mesmo, é necessário que tenha um plano de ação que aborda testes funcionais e não funcionais. Sendo assim, para que chegue ao nível esperado de qualidade, é bom 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, para garantir que a equipe vá seguir o processo, é necessário que haja um Gerente de Projetos que será responsável por promover treinamentos para reforçar as etapas que devem ser seguidas e garantir que os prazos sejam 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, mostra-se na Imagem 20 encontrada em Anexos ao final do trabalho, um processo sugerido por Primão (2012).

A Imagem 20 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.

No processo da Primão et al. (2012) Imagem 1, não apresenta com clareza quais são os tipos de testes realizados no sistema. Uma possibilidade interessante é introduzir essa etapa dentro da Fase 2, onde é feita a execução dos testes, envolvendo tanto testes funcionais quanto testes não funcionais. Desta forma, o processo de teste fica mais minucioso, transparente e colabora para uma redução de falhas.

#### **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 e até mesmo o número de acessos simultâneos no sistema. Em vista disso, tem como objetivo não apenas encontrar erros, mas eliminar gargalos de

desempenho do sistema. 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. Por outro lado, o teste de carga é realizado a fim de verificar a capacidade do sistema em suportar inúmeras transações geradas pela quantidade de usuários ativos. Ademais, o teste de capacidade é utilizado para identificar gargalos exibindo a capacidade de quantos usuários ou transações o sistema será capaz de suportar.

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* vs. Teste de Carga**

Apesar do significado ser relativamente parecido entre *stress* e carga, existem diferentes subtipos destes testes. Entre eles, tem-se 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, T. (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.

De outro modo, 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. Esse subtipo utiliza-se de uma carga que aumenta continuamente ao longo do tempo.

Outro subtipo é 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. Ferramentas de Teste de Performance

Existem algumas ferramentas responsáveis pela execução de testes de *stress* que executam os testes de modo automatizado, que servem para testar os recursos de carga simulando pessoas reais dentro do site ou sistema. Algumas são pagas, pois dependem da complexidade buscada pelo usuário, entretanto existem ferramentas que são totalmente gratuitas. 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.

### 2.6.1. 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. Sua versão atual é a 5.4.3 e ter o Java 8 ou versões mais recentes instalado na máquina é necessário.

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). Entretanto, para a execução de testes funcionais, há no mercado ferramentas mais qualificadas para tal.

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).

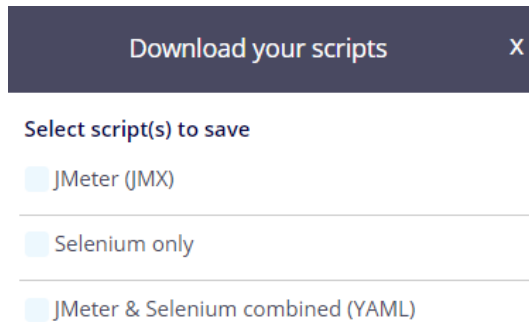
### 2.6.2. BlazeMeter:

A ferramenta BlazeMeter possui uma extensão instalável do navegador Google Chrome que permite realizar testes de turno. Ao instalar a extensão, é recomendado cadastrar-se na



plataforma da ferramenta, para que seja possível explorar tudo que ela oferece. Essa extensão registra todas as solicitações HTTP/S e as interações reais do usuário e cria o script pronto para ser executado pelo Apache JMeter ou Selenium, como pode ser observado na Imagem 1.

**Imagem 1** - Download do Script com BlazeMeter



Fonte: Imagem da ferramenta BlazeMeter

De acordo com Mascheroni e Irrazába (2013), o BlazeMeter é classificado como TaaS (Test as a Service), ou Teste como um Serviço, um modelo no qual o teste é realizado por um provedor de serviços. O trabalho de realização de testes, utilizando essa ferramenta, é realizado por uma rede de servidores em algum lugar do mundo, diferentemente de ser realizado manualmente por uma pessoa sem a ferramenta. Outras ferramentas similares são: Sauce Labs e SOASTA CloudTest.

## 2.7. Trabalhos Relacionados

Chaves (2019) fez um trabalho de conclusão de curso para 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. Essa ideia de testar a performance da aplicação

serve-se como um acréscimo para a equipe de testes da FTT, visto que aprimora a qualidade e confiança do produto final.

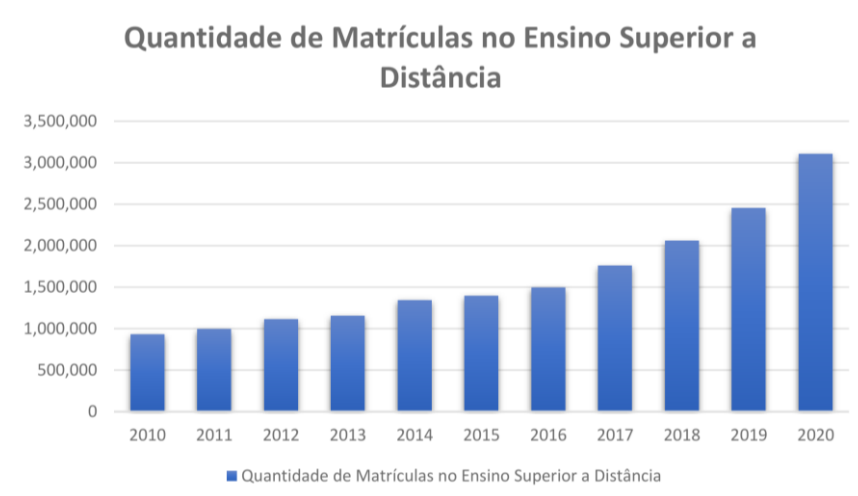
Sousa e Santana (2020) elaboraram um trabalho de conclusão de curso na UniEvangélica para o curso de Engenharia de Computação que propôs um processo de teste de performance para ambientes de fábrica de *software*. Para isso, eles utilizaram um projeto real chamado Virtoo, pertencente à Fábrica de Tecnologias Turing (FTT), que se situa dentro da UniEvangélica. Sendo assim, foram aplicados testes de performance em dois módulos do sistema, para coletar informações sobre o nível de desempenho da aplicação e, como outra métrica de resultado, foi coletado informações sobre o conhecimento da equipe de testes sobre a ferramenta Jmeter e seus usos.

Nas considerações finais, Sousa e Santana (2020) destacam-se que esse trabalho proposto foi uma continuação do trabalho de Chaves, porém com o acréscimo de propor um processo a ser seguido. Sendo assim, ficaria mais compreensível entender como funciona e como é feito o processo de teste de performance pela equipe responsável. O trabalho também destaca a importância efetiva de se abordar questões de performance em um sistema, uma vez que pode não estar adequado para suportar o quantitativo ideal de requisições simultâneas esperadas.

## **2.8. Ambientes Virtuais de Aprendizagem (AVAs)**

Ambientes Virtuais de Aprendizagem são ambientes *on-line* que proporcionam fins educacionais para instituições de ensino fundamental, médio ou superior. Nessas plataformas, as escolas ou faculdades podem oferecer aulas *online*, cursos, provas e conteúdos para qualquer usuário cadastrado. Além disso, serve como aproximador entre professor e aluno, visto que possibilita uma interação e solução de dúvidas de cada aluno específico.

O gráfico da Imagem 2 mostra um levantamento de dados referentes a notas estatísticas feitas pelo INEP (Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira), sobre a quantidade de matrículas feitas na modalidade de ensino a distância na década. A junção dos dados de 2010 até 2020 mostra que houve um aumento gradativo sobre o número de pessoas que ingressaram nos cursos de graduação EAD (Ensino a Distância), exemplificando a importância e a massiva procura por ambientes virtuais de aprendizagem nos últimos anos. Nesse período descrito no gráfico, as matrículas de cursos de graduação a distância aumentaram 233,9%, ao passo que na modalidade presencial o crescimento foi de apenas 2,3% (INEP, 2020).

**Imagem 2** - Quantidade de Matrículas na modalidade EAD no período 2010-2020

Fonte: Elaborado pelos autores.

### 2.8.1. AVA Versão 1 (2020/1 – 2021/1)

O primeiro ambiente virtual de aprendizagem da UniEvangélica foi principalmente utilizado por cursos de graduação presencial na modalidade *on-line* entre o começo de 2020 e o primeiro semestre de 2021, como medida momentânea em virtude da pandemia do COVID-19. O servidor da 1ª versão do AVA é hospedado em um data center da UniEvangélica, em uma única máquina virtual VMWare (Souza, E., 2022).

Por ser hospedado em apenas uma máquina virtual, o 1º AVA possui menos recursos disponíveis para garantir a disponibilidade do site. Entretanto, há uma estratégia de escalonamento vertical, o que aumenta a CPU, a memória RAM e o HD para diminuir os danos causados por sobrecargas no site. Além disso, adotou-se o banco de dados SQL Server, em detrimento do MySQL, para suportar um maior fluxo de dados pelo servidor.

### 2.8.2. AVA Versão 2 (2021/2 – atual)

O atual ambiente virtual de aprendizagem da UniEvangélica foi construído sobre a plataforma Moodle para possibilitar um meio de ensino que facilite a integração entre professores e alunos. Segundo Moodle (2006), a versão 1.0 foi lançada no dia 20 de agosto de 2002, criada por Martin Dougiamas que ainda continua na liderança do projeto. Em 2003 foi criada a empresa moodle.com para oferecer hospedagem gerenciada, consultorias e outros serviços.

A plataforma Moodle foi escrita em PHP, é gratuita e de código aberto sob a Licença Pública Geral GNU. A plataforma é utilizada em muitas universidades e escolas, tanto públicas

quanto privadas, sendo de acordo com Estúdio Site (2021) a plataforma de ambiente virtual de aprendizagem gratuita mais conhecida atualmente, com cerca de 260 milhões de usuários em 241 países. Alguns exemplos de outras universidades que utilizam esta plataforma são a Universidade Federal da Bahia (UFBA), Universidade Federal de Brasília (UnB) e Universidade de São Paulo (USP).

De acordo com Souza, E. (2022), o Moodle do atual AVA é disponibilizado pela empresa Open LMS, uma organização que oferta soluções de ambientes de aprendizagem para instituições de ensino superior, médio e fundamental. Tal empresa utiliza o código aberto para viabilizar uma melhor experiência e satisfação para sua comunidade e seus usuários. Além disso, utiliza-se do serviço em nuvem da AWS (*Amazon Web Services*), uma plataforma de armazenamento da Amazon, para proporcionar a disponibilidade dos dados e conteúdos aos seus clientes.

Ainda segundo Souza, E. (2022):

A disponibilização técnica e suporte do AVA é fornecida pelo departamento de Gestão de Ambientes Virtuais de Aprendizagem (AVAs) da Associação Educativa Evangélica (AEE), que tem em sua concepção a proposta de atuar de forma incessante para a incorporação de recursos modernos no processo de ensino-aprendizagem, objetivando promover com excelência o conhecimento, por meio da educação em seus diferentes níveis.

A segunda versão ambiente virtual de aprendizagem, por ser hospedado na AWS, conta com serviços de *auto-scaling*, que se resume em ajustar a capacidade e a escalabilidade para disponibilizar seu serviço mesmo com alto tráfego de pessoas. Além disso, conta com uma técnica chamada *load balance*, que se explica por manter a estabilidade do site e evitar a sobrecarga em momentos de pico.

### 3. METODOLOGIA DA PESQUISA

Como método de pesquisa deste trabalho, 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.

Sousa, Oliveira, Alves (2021) afirmam que:

A pesquisa bibliográfica é o levantamento ou revisão de obras publicadas sobre a teoria que irá direcionar o trabalho científico o que necessita uma dedicação, estudo e análise pelo pesquisador que irá executar o trabalho científico e tem como objetivo reunir e analisar textos publicados, para apoiar o trabalho científico.

Neste trabalho proposto 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.

Além da pesquisa qualitativa, também foi utilizada a pesquisa exploratória que foi usada para identificar as necessidades reais do trabalho, como por exemplo pesquisas sobre ferramentas de testes e suas funcionalidades, junto com a necessidade de se obter resultados sobre a plataforma em teste e suas limitações. O estudo de caso sobre a realidade das plataformas também foi abordado nesta 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 seguir, busca identificar e escolher ferramentas de teste que auxiliem na identificação de erros referentes a falhas de *performance* no sistema. Sendo assim, para a abordagem completa e padronizada do trabalho, foi criado um protótipo do processo que foi seguido para que não ocorressem falhas ou ausência de passos importantes que seriam abordados no processo final.

Em sequência, para a execução dos testes de *stress*, foram usados os sistemas EAD da UniEvangélica, a fim de identificar inconsistências com a concomitância de vários usuários conectados ao mesmo tempo executando uma requisição. Para que seja possível a execução do teste no sistema, utilizou-se a ferramenta Jmeter, que consiste em simular uma vasta quantidade de usuários acessando o sistema simultaneamente. Antes de realizar os testes nos ambientes EAD, foi realizado com o Israel, Líder Técnico da Fábrica de Tecnologias Turing (FTT), testes

em seu ambiente VIRTUO, um sistema acadêmico de Angola com a finalidade de garantir o entendimento nos testes utilizando as ferramentas.

Para analisar os defeitos encontrados, são utilizados os dados extraídos da ferramenta Jmeter que apontam a porcentagem de erros durante a execução das amostras e o tempo médio de requisições realizadas pela ferramenta. Os resultados encontrados são de suma importância para identificar possíveis melhorias no sistema em teste e identificar qual módulo consome mais memória do servidor.

Como objetivo principal desta pesquisa, o último ponto foi proposto um processo de teste de *stress*, para que seja possível identificar defeitos relacionados ao mau desempenho do sistema. Sendo assim, é necessário que consiga expor processos extensivos e claros, abordando métricas e testes não funcionais para compreender melhor os gargalos no sistema e que solucione problemas encontrados e proponha estratégias de disponibilidade em situações adversas.

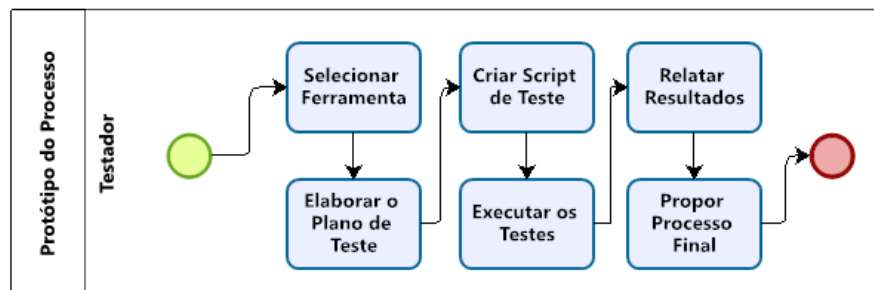
#### 4. ABORDAGEM PROPOSTA

Para alcançar os objetivos do trabalho, as abordagens levantadas na Seção 3 são aplicadas para elaborar o protótipo do processo de teste de *stress* utilizado na construção dos testes. Em seguida, expõe as ferramentas aplicadas no decorrer do processo que, para a realização dos testes de *stress* e posterior obtenção e análise dos resultados, foram utilizadas as ferramentas JMeter executando o script fornecido pela ferramenta BlazeMeter.

##### 4.1. Protótipo do Processo

Conforme a evolução do trabalho, tornou-se necessário a criação de um protótipo do processo, conforme Imagem 3, onde são abordados alguns dos passos seguidos para a realização dos testes. Inicialmente, como serão usados os sistemas de educação a distância da UniEvangélica, foi necessário realizar uma entrevista com a equipe do AVA para adquirir as informações necessárias e expectativa dos resultados dos testes. Em seguida, a análise sobre a seleção das ferramentas para a realização dos testes de *stress*.

Imagem 3 - Protótipo do Processo.



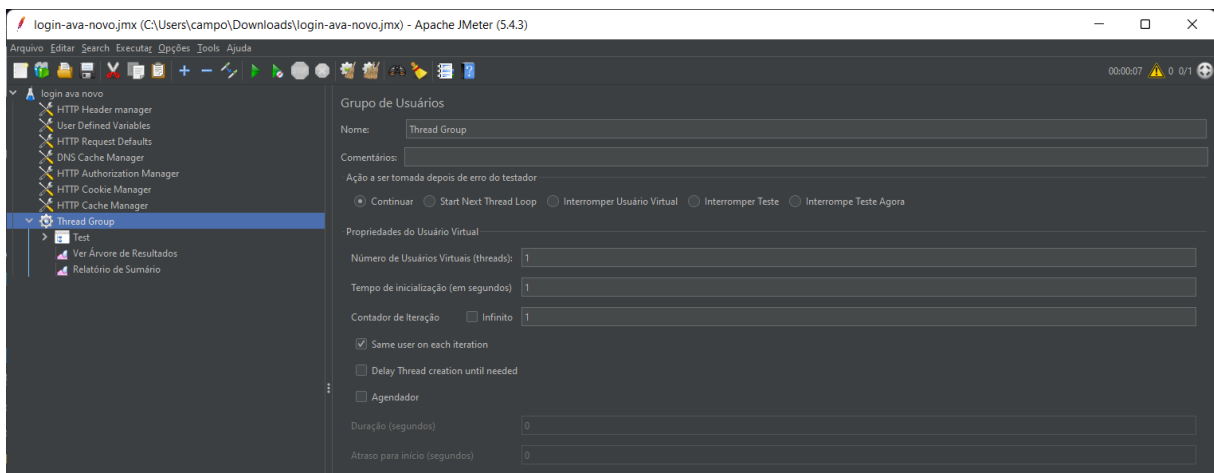
Fonte: Elaborado pelos Autores

Após selecionar a ferramenta, foi criado um Plano de Teste para guiar os analistas durante o processo de teste. É fundamental utilizar um plano de teste que contenha todos os passos e propósitos para concluir com excelência uma entrega de resultados esperados. Para executar os testes automatizados, deve-se criar o script de teste, executável na ferramenta escolhida. Ao finalizar as etapas anteriores, foi feita a execução dos testes nas versões 1 e 2 do AVA. Por fim, foi relatado o resultado dos testes que podem ser observados na Seção 5 deste trabalho. Ao finalizar o protótipo, o processo final padronizado e extensível foi adaptado para ser aplicado em sistemas EAD.

## 4.2. Jmeter e BlazeMeter

As ferramentas Jmeter e BlazeMeter foram escolhidas para serem alvo dos testes desta pesquisa. Primeiramente as escolhas foram feitas por ser dois *softwares* de código aberto, ou seja, gratuitos para todos os usuários que queiram utilizar a ferramenta. Para efeito de comparação, o LoadNinja e o LoadRunner citados neste trabalho são ferramentas pagas. Outro ponto levantado é o fato do Jmeter ser intuitivo e fácil de ser utilizado, além do conhecimento técnico prévio sobre a ferramenta pelos autores. Por outro lado, o principal motivo da escolha do BlazeMeter é por ser uma ferramenta que grava os passos executados no site e cria um script executável no Jmeter.

**Imagem 4 - Grupo de usuários.**



Fonte: Elaborado pelos autores

Após criar o script de teste pelo BlazeMeter e extraí-lo para o Jmeter, é necessário fazer as devidas configurações. Na tela de Grupo de Usuários, de acordo com a Imagem 4, é configurado a quantidade do número de usuários virtuais, o tempo de inicialização em segundos que representa o tempo de espera entre os usuários e o contador de iterações, que envia a quantidade de iterações que cada usuário realiza no script de teste. Após tudo configurado, o testador deve clicar no botão Iniciar no menu superior, para executar o teste.



**Imagem 5 - Relatório de Sumário.**

login-ava-novo.jmx (C:\Users\campo\Downloads\login-ava-novo.jmx) - Apache JMeter (5.4.3)

Relatório de Sumário

Nome: Relatório de Sumário

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo:  Procurar... Apenas Logar/Exibir  Erros  Sucessos Configurar

Rótulo	# Amostras	Média	Min.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
https://avas.un...	1	4740	4740	4740	0,00	0,00%	12,7/min	655,02	3,81	3179330,0
https://avas.un...	1	1815	1815	1815	0,00	0,00%	33,1/min	1,48	0,54	2748,0
https://avas.un...	1	81	81	81	0,00	0,00%	12,3/sec	6,66	13,91	552,0
https://avas.un...	5	84	79	89	3,32	0,00%	11,8/sec	6,56	12,97	571,2
https://avas.un...	1	49	49	49	0,00	0,00%	20,4/sec	8,31	23,26	417,0
https://avas.un...	1	84	84	84	0,00	0,00%	11,9/sec	7,25	14,82	624,0
Test	1	7191	7191	7191	0,00	0,00%	8,0/min	412,77	3,72	3186527,0
TOTAL	11	1307	49	7191	2310,45	0,00%	1,5/sec	825,53	7,45	579368,5

Fonte: Elaborado pelos autores

Ao finalizar todos os testes configurados, os resultados dos testes podem ser observados no Relatório de Sumário (Imagem 5), o qual refere-se à requisição realizada, por exemplo de um site HTTPS, as amostras são os valores da quantidade de usuários simuladas durante o teste. A média apresenta os valores médios em milissegundos, que são calculados de acordo com todas as execuções simultâneas. Por fim, é evidenciado a porcentagem de erros obtidos durante a execução dos testes na coluna de “% de Erro”.

## 5. RESULTADOS

Para levantar os dados necessários de performance dos ambientes virtuais de aprendizagem da UniEvangélica, foram executados testes de *stress* em cada um dos sistemas. As execuções dos testes foram feitas em três máquinas diferentes, para que as configurações de hardware fossem levadas em questão e fosse evidenciado a diferença que uma configuração de máquina faz nos acessos ao sistema.

Primeiramente, foi acordado com a equipe de Análise do AVA da UniEvangélica a realização dos testes no ambiente atual e na versão 1, a qual foi utilizada pelos acadêmicos entre os anos de 2019 e 2021. De acordo com Souza, E. (2022), tanto para o AVA v.1 como para o atual, cada unidade da AEE utiliza uma própria instalação do Moodle. O AVA v.2 utiliza instalações de infraestrutura individuais, diferentemente do AVA v.1 que possui apenas um servidor para todas as instituições.

Souza, E. (2022) foi questionado o que esperava com os resultados dos testes e sua resposta foi que a primeira versão do AVA com 600 usuários simultâneos começa a apresentar instabilidade, porém os testes podem falhar devido ao *firewall*. Em contrapartida, o atual é esperado que o sistema comece a apresentar lentidão a partir de 5 mil usuários, mas não fica indisponível. Na Tabela 1 consta a quantidade de alunos matriculados em cada unidade da mantida AEE.

**Tabela 1** - Quantidade de Usuário Matriculados até Junho de 2022.

Unidade	QTD
UniEvangélica - Universidade Evangélica de Goiás (Anápolis)	8633
Faculdade Evangélica de Goianésia (FACEG)	1526
Colégio Couto Magalhães	1398
Centro Universitário de Anápolis - EAD	974
Faculdade Evangélica de Ceres (FECER)	706
Universidade Evangélica de Goiás	552
Faculdade Evangélica de Rubiataba (FER)	464
Colégio Álvaro de Melo	462
UniEvangélica - Universidade Evangélica de Goiás (Ceres)	408
Faculdade Evangélica Raízes	385
Escola de Enfermagem Florence Nightingale	266
Faculdade Evangélica de Senador Canedo (FESCAN)	146
Colégio Couto MAGalhães - Goianésia	145

Fonte: SOUZA, E. 2022

### 5.1. AVA Versão 2

Os primeiros testes executados não ocorreram como esperado. O teste foi feito no AVA v.2, com uma arquitetura estruturada para suportar um número de carga elevada no sistema.

Além disso, foi combinado com a equipe de análise do AVA para ser feito em um horário que não gerasse danos aos alunos da universidade. Sendo assim, os testes foram feitos em um notebook com as seguintes especificações de máquina, conforme Tabela 2.

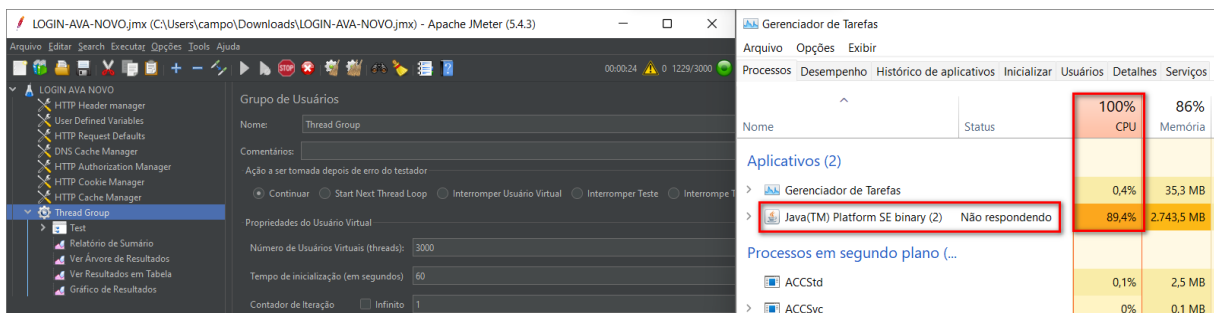
**Tabela 2** - Configuração da máquina 1.

Componente/Versão	Especificação
<b>Processador</b>	Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz
<b>RAM instalada</b>	8,00 GB (utilizável: 7,84 GB)
<b>Tipo de sistema</b>	Sistema operacional de 64 bits, processador baseado em x64
<b>Sistema Operacional</b>	Edição Windows 11 Home Single Language Versão 21H2
<b>Versão Apache JMeter:</b>	5.4.3

Fonte: Elaborada pelos autores.

A realização do primeiro teste foi feita com 3000 usuários simultâneos, tempo de inicialização de 60 segundos e com execução de uma requisição por usuário. A velocidade de rede no momento do teste era de 150Mbps e a de upload era 41Mbps. Entretanto, o processador da máquina não suportou a quantidade de processos do Jmeter e travou a ferramenta, conforme pode ser observado na Imagem 6.

**Imagem 6** - Erro no 1º teste (AVA v.2).



Fonte: Elaborada pelos autores.

Durante a execução dos testes, foram encontrados alguns empecilhos que dificultaram a atividade de teste. Devido ao alto número de *threads* (número de usuários), a máquina apresentada na Tabela 2 não conseguiu executar o teste com uma grande carga de usuários. Sendo assim, a CPU do computador não foi suficiente para suportar tanta quantidade de dados e a interface do JMeter não conseguiu finalizar a execução do plano de teste.

Após a tentativa sem sucesso com a máquina 1, foi disponibilizado uma segunda máquina mais potente para a realização de outro teste e, por ser um equipamento superior, o esperado era que suportasse com facilidade todas as requisições do Jmeter. O segundo teste foi feito no computador do laboratório da UniEvangélica com as especificações de máquina mostradas na Tabela 3.

Tabela 3 - Configuração da máquina 2.

Componente/Versão	Especificação
Processador	Intel(R) Core(TM) i7-7700H CPU @ 3.60GHz 3.60 GHz
RAM instalada	16.00 GB (utilizável: 15.4 GB)
Tipo de sistema	Sistema operacional de 64 bits, processador baseado em x64
Sistema Operacional	Edição Windows 10 Pro Versão 20H2
Versão Apache JMeter:	5.4.3

Fonte: Elaborada pelos autores.

Diferentemente das tentativas anteriores, a máquina suportou melhor sem travar por causa de CPU ou memória. Definiu-se 300 usuários virtuais com tempo de inicialização de 60 segundos e executando 4 requisições/iterações. O teste de velocidade na rede foi feito e marcou 95Mbps de download e upload de 90Mbps. Ao executar o teste, a rede indicou 100%, conforme mostra a Imagem 7, além de não terminar as últimas 12 execuções de testes, sendo feitas apenas 288. Com isso, tem-se o seguinte resultado deste experimento, exibido na Imagem 8.

Imagem 7 - Gerenciador de tarefas que indica problemas de rede.

Nome	Status	86% CPU	50% Memória	0% Disco	100% Rede
> Java(TM) Platform SE binary		81,2%	4.662,8 MB	0 MB/s	94,9 Mbps

Fonte: Elaborada pelos autores.

Imagem 8 - Relatório de Sumário do 2º teste (AVA v.2).

Rótulo	# Amostras	Média	Min.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
https://avas.un...	889	56427	1391	872970	76513,48	11,25%	59,1/min	3020,40	18,82	3138840,8
https://avas.un...	879	985	464	15900	1059,47	0,00%	2,3/sec	5,54	2,22	2482,3
https://avas.un...	879	1055	42	818134	27576,81	0,11%	58,8/min	0,53	1,10	554,8
https://avas.un...	878	101	35	3519	196,19	0,00%	2,3/sec	0,93	2,61	417,0
https://avas.un...	878	125	53	4257	183,30	0,00%	2,3/sec	1,40	2,85	624,0
https://avas.un...	4390	126	56	3592	158,78	0,00%	11,5/sec	6,39	12,63	571,2
Test	878	56048	2451	348769	60805,65	11,05%	2,3/sec	7000,62	66,05	3179267,2
TOTAL	9671	10539	35	872970	37551,08	2,05%	10,7/sec	6047,98	57,05	577801,4

Fonte: Elaborada pelos autores.

A Imagem 8 mostra os resultados do teste de *login* a partir da ferramenta *JMeter* juntamente com a ferramenta *BlazeMeter*, onde 11,25% das requisições realizadas pela ferramenta foram falhas, com uma média de 56,42 segundos para tempo de resposta de uma requisição para outra. O tempo mínimo de uma requisição apontada pelo *JMeter* foi de 1391 ms para que o *login* seja efetivo durante o teste. Esse tempo é equivalente a 1,3 segundos para a realização da ação.

Em razão do problema de rede no segundo teste, foi necessário realizar um terceiro teste com configurações de internet e de hardware diferentes. Sendo assim, a velocidade de internet passou a ser de 200 Mbps e a Tabela 4 exhibe as configurações da máquina na qual foi realizada

a tentativa, juntamente com as ferramentas propostas. Os testes foram executados com excelência sem apresentar problemas durante a sua operação.

**Tabela 4** - Configuração da máquina 3.

Componente/Versão	Especificação
<b>Processador</b>	Intel(R) Core(TM) i5-7600 CPU @ 3.50GHz 3.50 GHz
<b>RAM instalada</b>	16,0 GB (utilizável: 15,1 GB)
<b>Tipo de sistema</b>	Sistema operacional de 64 bits, processador baseado em x64
<b>Sistema operacional</b>	Edição Windows 10 Pro Versão 21H1
<b>Versão Apache JMeter</b>	5.4.1

Fonte: Elaborada pelos autores.

A Imagem 9, descrita abaixo, mostra os resultados a partir da execução do teste de *Login* com uma máquina necessária para uma execução melhor dos testes. Com isso, os números de usuários simulados neste teste foram de 200 usuários virtuais, com o tempo de inicialização de 5 segundos de um usuário para outro e um contador de iteração com o valor de 25 iterações.

**Imagem 9** - Relatório de Sumário 3º teste (AVA v.2).

Rótulo	# Amostras	Média	Min.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
https://avas.unievangelica.edu.br/login/index.php	4945	48316	1482	1100540	39503,09	1,64%	3,5/sec	10913,91	63,15	3175125,1
https://avas.unievangelica.edu.br/theme/snap/rest...	4945	1119	514	11805	687,53	0,00%	3,6/sec	9,66	3,50	2746,8
https://avas.unievangelica.edu.br/lib/ajax/service.p...	4945	163	51	18197	334,97	0,00%	3,6/sec	1,94	4,05	552,0
https://avas.unievangelica.edu.br/lib/ajax/service.p...	24725	176	63	8390	182,37	0,00%	18,0/sec	10,04	19,78	571,2
https://avas.unievangelica.edu.br/lib/ajax/service-n...	4945	141	49	4595	164,63	0,00%	3,6/sec	1,47	4,09	417,0
https://avas.unievangelica.edu.br/lib/ajax/service.p...	4945	170	60	4867	164,85	0,00%	3,6/sec	2,20	4,47	624,0
Test	4945	50793	2957	1103029	39728,88	1,64%	3,5/sec	10928,78	98,14	3182320,9
TOTAL	54395	9235	49	1103029	25437,06	0,30%	38,7/sec	21857,57	196,28	578603,8

Fonte: Elaborada pelos autores.

Após os resultados finais dos testes, o *login* apresentou uma porcentagem de erro de 1,64% e uma média de tempo de resposta de 48316 ms, que é equivalente a 48 segundos. O tempo mínimo para a resposta de requisição HTTP foi de 1482 ms que equivale a 1,4 segundos. Observa-se que o sistema começa a ter um tempo de resposta maior que o esperado com uma carga muito grande, porém não falha e suporta a quantidade de requisições feita.

**Imagem 10** - Relatório de Sumário 4º teste (AVA v.2).

Relatório de Sumário

Nome: Relatório de Sumário

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo:     Erros  Sucessos

Rótulo	# Amostras	Média	Min.	Máx.	Desvio Padrão	% de Erro	Vazão	KB/s	Sent KB/sec	Média de Bytes
https://avas.unievangelica.edu.br/login/index.php	8339	66753	1451	1909950	62034,19	1,97%	3,5/sec	10963,38	63,47	3173529,6
https://avas.unievangelica.edu.br/theme/snap/rest...	8339	1405	498	44334	1393,38	0,00%	3,6/sec	9,64	3,50	2746,8
https://avas.unievangelica.edu.br/lib/ajax/service.p...	8339	185	45	9730	298,11	0,00%	3,6/sec	1,94	4,04	552,0
https://avas.unievangelica.edu.br/lib/service.p...	41695	195	63	21421	282,19	0,00%	18,0/sec	10,04	19,77	571,2
https://avas.unievangelica.edu.br/lib/ajax/service.n...	8339	157	42	9092	228,51	0,00%	3,6/sec	1,47	4,09	417,0
https://avas.unievangelica.edu.br/lib/ajax/service.p...	8339	193	63	14472	299,96	0,00%	3,6/sec	2,19	4,46	624,0
Test	8339	71675	2620	1913350	62397,43	1,97%	3,5/sec	10983,16	98,68	3180725,4
TOTAL	91729	13031	42	1913350	37830,66	0,36%	38,9/sec	21966,32	197,35	578313,7

Fonte: Elaborada pelos autores.

Na Imagem 10, foi testado da seguinte forma: quantidade de 300 usuários realizando 30 requisições cada em intervalos de 5 segundos. Mostra-se uma porcentagem de 1,97% de erro ao tentar realizar *login* na plataforma. A média de tempo de requisição foi com a mínima de 1,4 segundo e a média de 68,7 segundos. Com esse resultado, o sistema não consegue suportar essa quantidade de dados e acaba sofrendo lentidão, aumentando o tempo de *login* no sistema.

**Imagem 11** - Relatório de Sumário 5º teste (AVA v.2).

Relatório Agregado

Nome: Relatório Agregado

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo:     Erros  Sucessos

Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Min. ↓	Máx.	% de Erro	Vazão	KB/s	Sent KB/sec
https://avas.unievangelica.edu.br/mod/quiz/view.php?id=629496	4855	31076	24552	53127	70807	118271	0	1101261	3,09%	3,7/sec	9969,06	33,05
https://avas.unievangelica.edu.br/mod/quiz/statattempt.php	4853	7732	5260	11458	15800	53966	0	908986	2,84%	3,7/sec	1304,27	38,77
Test	4853	38812	31503	65219	83568	149700	0	1103366	5,48%	3,6/sec	11194,50	71,13
TOTAL	14561	25874	19934	52676	68350	118271	0	1103366	3,80%	10,9/sec	22393,13	142,28

Fonte: Elaborada pelos autores.

Na Imagem 11, o teste foi executado com uma carga esperada de 5000 amostras, ou seja, 200 usuários realizando 25 requisições cada num intervalo de 5 segundos. Após a finalização do teste, o JMeter retornou uma porcentagem de erro de 3,09% para a ação de ver uma questão da prova e 2,84% de erro na execução da ação de iniciar uma prova. A média de tempo para a requisição de iniciar uma prova foi de 7732 ms correspondendo a 7,73 segundos de resposta. Já a ação de ver uma questão, teve uma média de 31076 ms equivalente a 31,07 segundos de tempo de resposta.

**Imagem 12** - Relatório de Sumário 6º teste (AVA v.2).

Relatório Agregado

Nome: Relatório Agregado

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo:     Erros  Sucessos

Rótulo	# Amostras	Média ↑	Mediana	90% Line	95% Line	99% Line	Min.	Máx.	% de Erro	Vazão	KB/s	Sent KB/sec
https://avas.unievangelica.edu.br/mod/quiz/statattempt.php	8273	9470	7336	13898	21434	48363	463	185083	0,48%	3,3/sec	1025,54	36,88
https://avas.unievangelica.edu.br/mod/quiz/view.php?id=629496	8275	58000	45916	104689	133432	226045	718	2135697	1,11%	3,4/sec	9241,75	31,00
Test	8273	67465	55345	118133	148272	241778	1257	2137979	1,54%	3,4/sec	10241,65	67,10
TOTAL	25121	44079	24151	97406	123712	205143	463	2137979	1,04%	10,2/sec	20685,52	134,20

Fonte: Elaborada pelos autores.

Na Imagem 12, os resultados mostrados são referentes a uma carga de 9000 amostras, sendo 300 usuários realizando 30 iterações cada num intervalo configurado de 5 segundos de uma requisição a outra. Ao acessar uma prova e iniciá-la, o *JMeter* retornou a porcentagem de erro de 0,48% com uma média de 9,47 segundos de tempo de resposta. Por outro lado, ao ver uma questão da prova, retornou uma porcentagem de 1,11%, com uma média de 58 segundos para ver uma questão. Diferentemente da Imagem 11, a quantidade de iterações e usuários foi aumentada e o tempo de resposta da plataforma acaba ficando abaixo do esperado.

## 5.2. AVA Versão 1

Os testes no primeiro ambiente virtual de aprendizagem da UniEvangélica foram realizados a partir da máquina que é apresentada na Tabela 4. As configurações utilizadas no *JMeter* para esses testes foram menores do que no AVA v.2, sendo que a estabilidade do AVA v.1. é menor e apresenta maior porcentagem de erro durante a execução dos testes.

**Imagem 13** - Relatório de Sumário 7º teste (AVA v.1).

Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Min.	Máx.	% de Erro	Vazão	KB/s	Sent KB/sec
https://avagraduacao.unievangelica.edu.br/disciplinasonline/login/index.php	2829	38625	35201	61419	73463	107441	1	398717	9,33%	5,6/sec	9497,96	46,68
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	2829	1998	1158	2883	4146	10348	0	53814	1,31%	5,6/sec	200,42	3,88
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	2829	196	108	372	526	1941	0	9600	0,85%	5,6/sec	3,53	3,82
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	8486	242	113	524	694	1480	0	13623	0,05%	16,9/sec	57,83	13,50
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	2829	562	356	1006	1450	3097	72	19959	0,04%	5,6/sec	71,17	5,65
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	5658	169	103	363	408	1316	0	10942	0,27%	11,3/sec	3,06	8,13
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	2829	157	104	170	397	1043	0	14793	0,11%	5,6/sec	1,95	5,12
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	2829	148	104	166	399	1001	0	3038	0,07%	5,6/sec	3,72	4,48
Test	2828	42357	38901	66293	79499	112819	1767	400261	11,63%	5,6/sec	9819,17	90,89
TOTAL	33946	7058	120	33432	46070	72979	0	400261	2,00%	67,2/sec	19641,94	181,80

Fonte: Elaborada pelos autores.

Como mostra a Imagem 13, o teste de *login* no AVA v.1. foi realizado com as seguintes configurações: 300 usuários efetuando 10 requisições a cada 5 segundos. O retorno dos testes exibido pelo *JMeter* mostra uma porcentagem de erro de 9,33% com uma média de tempo de requisição de 38,62 segundos e uma mínima de tempo de 1ms. Tais dados mostram-se um percentual de erro grande referente a quantidade de requisições feitas e um tempo médio relativamente alto por conta do suporte do servidor.

**Imagem 14** - Relatório de Sumário 8º teste (AVA v.1).

Relatório Agregado

Nome: Relatório Agregado

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo  Procurar... Apenas Logar/Exibir  Erros  Sucessos  Configurar

Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Min.	Máx.	% de Erro	Vazão	KB/s	Sent KB/sec
https://avagraduacao.unievangelica.edu.br/disciplinasonline/login/index.php	4069	57459	49804	93996	117664	192899	0	672535	21,45%	5,8/sec	9286,83	47,38
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	4056	2718	1368	5389	8919	24181	0	112902	1,08%	5,8/sec	206,20	4,01
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	4055	683	111	2232	4174	8605	0	17258	2,32%	5,9/sec	3,81	3,93
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	12157	305	115	624	945	2544	0	49989	0,14%	17,6/sec	60,38	14,07
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	4052	743	374	1422	2410	5300	0	48564	0,12%	5,9/sec	74,19	5,89
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	8102	180	104	367	451	1789	0	8939	0,69%	11,8/sec	3,28	8,46
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	4051	192	105	376	442	1730	0	42069	0,10%	5,9/sec	2,04	5,35
https://avagraduacao.unievangelica.edu.br/disciplinasonline/lib/ajax/servic...	4051	177	105	364	433	1669	0	10298	0,05%	5,9/sec	3,88	4,69
Test	4051	63155	54797	104398	126646	206722	1474	674572	24,54%	5,7/sec	9686,80	92,30
TOTAL	48644	10548	131	46998	68165	116665	0	674572	4,29%	68,7/sec	19410,79	184,81

Fonte: Elaborada pelos autores.

Ao analisar a Imagem 14, temos o teste executado no AVA (versão 1), com a funcionalidade de *login*. Aumentou-se a carga do JMeter para a seguinte configuração: 400 usuários realizando 10 requisições cada num intervalo de 5 segundos. Como mostra a imagem, o resultado do teste teve 21,4% de erro de requisições, uma média de 57,45 segundos de tempo de resposta com um mínimo de 0 ms. Ao examinar os resultados, pode-se observar que o sistema apresentou bastante instabilidade com uma carga de 4000 amostras.

**Imagem 15** - Relatório de Sumário 9º teste (AVA v.1).

Relatório Agregado

Nome: Relatório Agregado

Comentários:

Escrever resultados para arquivo / Ler a partir do arquivo

Nome do arquivo  Procurar... Apenas Logar/Exibir  Erros  Sucessos  Configurar

Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Min.	Máx.	% de Erro	Vazão	KB/s	Sent KB/sec
https://avagraduacao.unievangelica.edu.br/disciplinasonline/mod/quiz/view.php?id=640745	5737	17091	10548	38502	45460	64788	0	888304	7,53%	6,0/sec	5382,62	49,24
https://avagraduacao.unievangelica.edu.br/disciplinasonline/mod/quiz/startattempt.php	2861	7469	5428	12610	16880	32578	0	902798	3,57%	3,1/sec	134,97	29,98
Test	2861	41994	38115	60351	70536	110236	2528	947022	17,79%	3,0/sec	5506,66	77,97
TOTAL	11459	20682	12352	46419	55366	80720	0	947022	9,10%	12,0/sec	11018,25	155,88

Fonte: Elaborada pelos autores.

Durante a execução do 9º teste referenciado na Imagem 15, foi observado um problema durante a análise de resultados. Ao tentar realizar a execução do início de prova, o JMeter não conseguiu iniciar metade das amostras previstas e, com isso, ele apresentou uma porcentagem de erro de 3,57% em uma média de 7,4 segundos de uma requisição a outra. Por outro lado, na ação de ver questões da prova, foram realizadas todas as execuções previstas com o resultado de 7,53% de erro de requisições com uma média de 17 segundos de tempo de resposta. Todo o teste foi executado com 300 usuários virtuais realizando 10 iterações cada num tempo de 5 segundos.



**Imagem 16** - Relatório de Sumário 10º teste (AVA v.1).

Rótulo	# Amostras	Média	Mediana	90% Line	95% Line	99% Line	Min.	Máx.	% de Erro	Vazio	KB/s	Sent KB/sec
https://avagraduacao.unievangelica.edu.br/disciplinasonline/mod/quiz/view.php?id=640745	9014	32464	15190	65422	88265	165558	0	2083301	17,08%	4,3/sec	3779,09	33,84
https://avagraduacao.unievangelica.edu.br/disciplinasonline/mod/quiz/startattempt.php	4400	16835	7711	19610	27636	64933	0	1986035	6,00%	2,1/sec	129,64	20,51
Test	4400	72456	32400	122182	161466	427465	1896	2072975	33,73%	2,1/sec	3887,29	33,76
TOTAL	17814	38482	16602	78921	107970	220863	0	2083301	18,46%	8,5/sec	7793,04	107,73

Fonte: Elaborada pelos autores.

O mesmo problema foi identificado no teste da Imagem 16. O início de prova não conseguiu alcançar a quantidade de amostras previstas apresentando 6,0% de erros na execução dos testes. A configuração de execução desse teste foi de 600 usuários realizando 10 iterações cada com 5 segundos de intervalo de um para o outro. De outro modo, a visualização da questão da prova apresentou mais instabilidade com 17,08% de erros com uma média de 32,46 segundos de tempo de resposta.

### 5.3. Resultados Gerais Dos Testes

Ao analisar todos os resultados dos testes realizados, foi identificado um problema no teste em uma máquina com configurações inferiores, sendo que o JMeter não executava os testes e parava de responder durante a ação, como ficou evidenciado no teste com a máquina da Tabela 2. Outro problema encontrado foi a conexão de *internet*, já que a ferramenta JMeter simula usuários virtuais e utiliza bastante a conexão e exige uma *internet* melhor para execução de testes de larga escala.

Após executar todos os testes, foi obtido um retorno esperado para cada funcionalidade. Como as versões do AVA suportam cargas diferentes de usuários, foi utilizado a estratégia de acordo com a disponibilidade do servidor e, com isso, os testes foram executados com cargas diferentes de um servidor para outro.

#### 5.3.1. Comparativo Entre As Duas Versões Do Ava

A segunda versão do AVA, apresentou um desempenho melhor em relação a quantidade de usuários simulados. Com uma taxa de erro de 1,64% com cerca de 5 mil usuários realizando iterações simultâneas, ele consegue suportar a quantidade média prevista pela instituição. Por outro lado, o AVA v.1, apresentou alguns problemas durante a execução dos testes, visto que cerca de 3 mil usuários foi o suficiente para gerar 9,33% de erros. Sendo assim, esse resultado revela uma melhora importante do sistema AVA e que possui um bom suporte para a quantidade de alunos matriculados na instituição de ensino.

Outro dado importante é a simulação de uma maior quantidade de usuários, simulando assim um *stress* maior nos servidores disponíveis para teste. Como citado acima, na Imagem 14, o JMeter não foi capaz de gerar cargas suficientes no AVA v.1, e segundo Sousa, E. (2022), possivelmente os testes poderiam ser barrados por conta da proteção do *firewall*. Tendo essa informação em mente, a porcentagem de erro do AVA v.1 foi de 21,4% com uma carga de 4 mil usuários simultâneos realizando a ação de *login* no sistema.

Na primeira perspectiva, os testes no AVA atual retornaram resultados de extrema importância em comparação a carga aplicada e o retorno de erros. Com uma quantidade de 9 mil usuários, a porcentagem de erro foi de 2,06%, mostrando assim uma melhora significativa em relação a quantidade de erros do AVA v.1. Ao testar a execução de uma prova, foram utilizados os seguintes casos de teste: Acessar uma prova e ver uma questão. Ao executar os testes, teve o seguinte retorno: 3,09% de erro para a ação de iniciar uma prova e 2,84% na ação de ver uma questão da prova. Com isso, a simulação realizada foi de 5 mil amostras, ou seja, 5 mil alunos executando a ação de abrir uma prova e visualizar a primeira questão da prova.

Por outro lado, no AVA versão 1, a ação de ver uma prova e abrir uma questão foi observado uma porcentagem de erro de 3,57% na execução de iniciar uma prova e 7,53% de erro para ação de ver uma questão. Entretanto, a quantidade de usuários foi de 3 mil, simulando 3 mil alunos executando uma prova num intervalo de 5 segundos no AVA v.1. Porém, como dito anteriormente, possivelmente o *firewall* do sistema barrou a execução dos testes no primeiro Ambiente Virtual de Aprendizagem.

O nível de *stress* do servidor do AVA v.2 é bem aceitável quando trata-se da execução das provas simultâneas dos usuários. Com uma carga de 9 mil usuários, foi obtido o resultado de um total de 0,48% para iniciar a prova e 1,11% para a visualização da prova. Já o AVA v.1 mostra um nível de *stress* que apresentou problemas durante a execução dos testes, com uma carga de 6 mil usuários. Assim, para iniciar a prova, foram 6,0% de erros e para visualizar a questão da prova foram 17,08% de erros.

Com esses resultados, tendo como comparação a execução das provas, o AVA v.1 apresentou bastante instabilidade e mostrou problemas esperados durante a realização das provas. Por outro lado, na segunda versão do AVA, as provas podem ser executadas de maneira síncrona e não apresentará instabilidade. A Tabela 5 abaixo, resume os testes feitos anteriormente.

Tabela 5 - Resultados Gerais dos Dois Ambientes.

<b>Login</b>					
<b>AVA 1</b>			<b>AVA 2</b>		
<b>QNTD de Requisições</b>	<b>% de Erro</b>	<b>Média em S</b>	<b>QNTD de Requisições</b>	<b>% de Erro</b>	<b>Média em S</b>
3000	9,33%	38,62	5000	1,64%	48
4000	21,40%	57,45	9000	2,06%	70,8
<b>Ver Prova</b>					
<b>AVA 1</b>			<b>AVA 2</b>		
<b>QNTD de Requisições</b>	<b>% de Erro</b>	<b>Média em S</b>	<b>QNTD de Requisições</b>	<b>% de Erro</b>	<b>Média em S</b>
3000	7,53%	17	5000	2,84%	31,07
6000	17,08%	32,46	9000	1,11%	58
<b>Iniciar Prova</b>					
<b>AVA 1</b>			<b>AVA 2</b>		
<b>QNTD de Requisições</b>	<b>% de Erro</b>	<b>Média em S</b>	<b>QNTD de Requisições</b>	<b>% de Erro</b>	<b>Média em S</b>
3000	3,57%	7,4	5000	3,09%	7,73
6000	6,00%	5	9000	0,48%	9,47

Fonte: Elaborado Pelos Autores

#### 5.4. Processo de Teste

O objetivo principal desta pesquisa é propor um processo de teste de *stress* para ser utilizado em ambientes virtuais de aprendizagem. Tais processos são importantes em todo ciclo de vida de um projeto e é extremamente necessário existir a etapa de testes funcionais e testes não funcionais. Os responsáveis pelos testes são nomeados Analista de Teste, sendo responsáveis por assegurar a qualidade do produto.

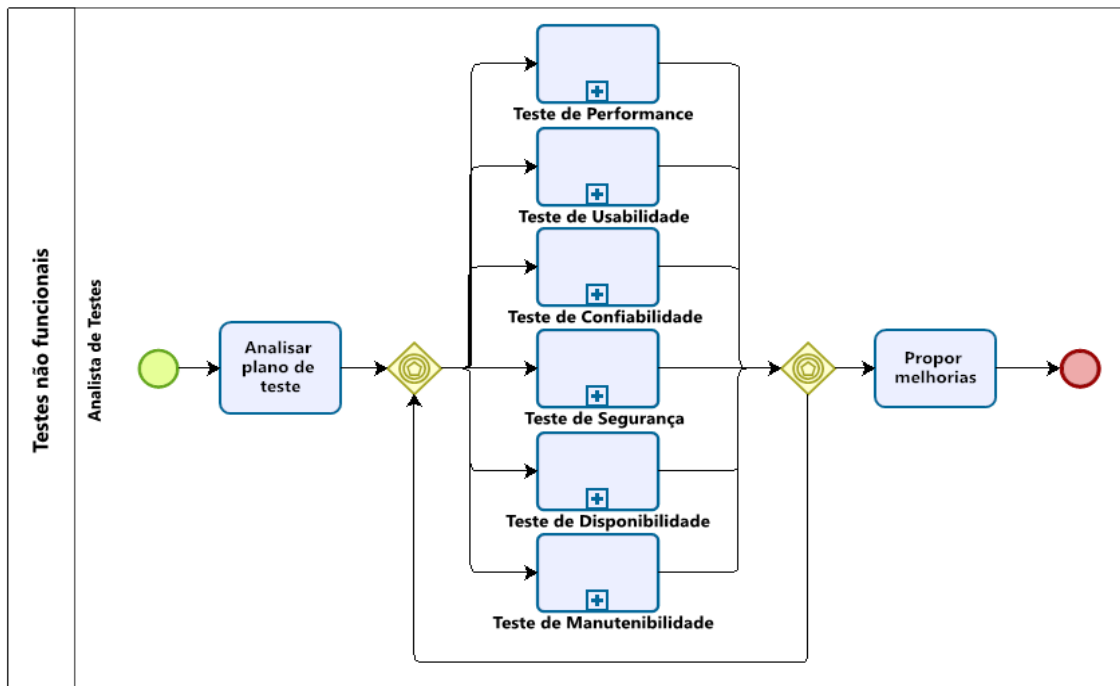
Sendo assim, no momento em que a equipe de projetos cria o *BackLog* do produto, a equipe de teste produz o plano de teste que será seguido para realização dos testes. Na Imagem 18 encontrada no Apêndice ao final do trabalho, o processo de teste abordado no referencial teórico com as adaptações sugeridas, na qual é proposto dois subprocessos um para os Testes Funcionais e outro para os Testes Não Funcionais para o Analista de Teste execute os testes com exatidão.

A Imagem 21, encontrada no Apêndice A no final do trabalho, mostra um processo geral de testes, que é feito após o desenvolvedor codificar a funcionalidade e enviar ao testador para

que ele produza o plano de teste na Fase 1. Em seguida, o testador desenvolve o caso de teste da funcionalidade em questão e segue para a próxima fase do processo. Posteriormente, na Fase 2, tem-se o subprocesso de testes funcionais, onde os testes são realizados utilizando o caso de teste referente e a documentação do requisito.

Subsequentemente, o testador evidencia a necessidade de testes comportamentais e de qualidade na aplicação e segue para o subprocesso de testes não funcionais, como mostrado abaixo na Imagem 17. Esse subprocesso lida-se com as regras de negócio expostas na documentação do requisito e com o plano de testes para definir atributos de qualidade, segurança e confiabilidade para o sistema e testa tais métricas para atribuir características qualitativas para a aplicação.

**Imagem 17** - Processos de Testes Não Funcionais

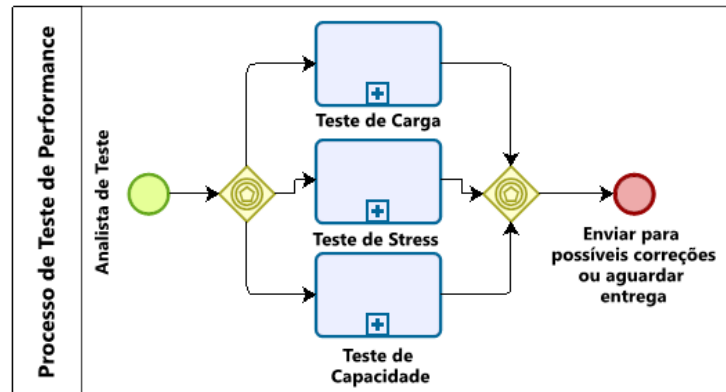


Fonte: Elaborado pelos autores

Como apresentado na Imagem 17, após o analista de testes examinar o plano de teste em busca de características de qualidade do sistema, ele decide qual tipo de teste não funcional será feito e sua respectiva ordem. A imagem destaca diversos tipos de testes não funcionais, tais como teste de performance, teste de usabilidade, teste de confiabilidade, teste de segurança, teste de disponibilidade e teste de manutenibilidade. Ao passar por um tipo de teste específico, o testador pode regressar a outro tipo de teste, até que todos os aspectos de qualidade do sistema

tenham sido testados. Em seguida, abordando o foco do trabalho, é proposto o subprocesso de realização de Teste de Performance.

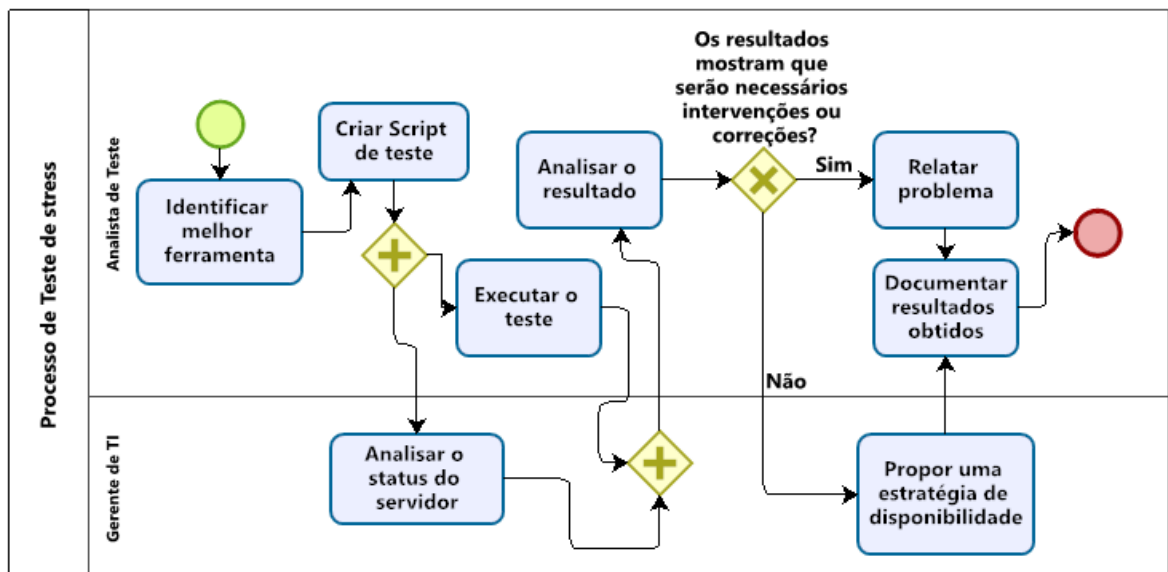
**Imagem 18** - Processo de teste de performance.



Fonte: Elaborado pelos autores.

No processo da Imagem 18, é proposto a execução de alguns dos testes existentes em testes de performance. Para testar o sistema de acordo com a carga conciliada com o cliente, o testador precisa executar o subprocesso de Teste de Carga. Por outro lado, para medir a capacidade global do sistema e determinar até que ponto o tempo de resposta torna-se inaceitável, deve ser elaborado o subprocesso de Teste de Capacidade. Por fim, para testar o limite do sistema e quanto suportar em situações extremas, ultrapassando a carga definida, é feito o teste de *stress*. Como teste de *stress* que é o foco desta pesquisa, será abordado somente este subprocesso.

**Imagem 19** - Processo de teste de stress.



Fonte: Elaborado pelos autores.

Por fim, tem-se o processo de teste de *stress* Imagem 19. Por conseguinte, o testador deve identificar qual a melhor ferramenta para o teste de *stress* e, em seguida, criar o script de teste baseado nos casos de testes feitos anteriormente. Logo em seguida, deve-se executar o script na ferramenta selecionada em paralelo, o Gerente de TI deve analisar o status do servidor analisando a porcentagem do uso durante a execução do teste.

Quando o teste finalizar, o testador analisa o resultado do teste identificando a possível necessidade de correção/intervenção. Caso precise de correções, o testador deve relatar o problema, caso não precise de correções, o Gerente de TI deve propor uma estratégia de disponibilidade. Por fim, o testador deve documentar os resultados obtidos e assim, finaliza o processo de *stress* e avança para a próxima etapa do processo anterior, na Imagem 20.

## 6. CONSIDERAÇÕES FINAIS

Dentre os principais objetivos desta pesquisa, estão a criação de um processo padronizado e extensível a ser aplicável à ambientes EAD. Assim que os acadêmicos da Universidade Evangélica de Goiás vivenciaram uma indisponibilidade durante a realização das provas online durante a pandemia, foi questionado se o ambiente havia passado por testes não funcionais. Ao fazer este questionamento para a equipe de análise do AVA, a resposta foi que não havia sido feito nenhum tipo de testes não funcionais, diante disso a primeira versão do AVA não conseguia suportar a quantidade total de alunos do turno noturno.

Por meio de diversas pesquisas bibliográficas, neste trabalho afirma que se a equipe de qualidade de *software* tivesse executado testes de carga e teste de *stress* o AVA v.1, a indisponibilidade não teria acontecido. Segundo a equipe de análise, o AVA v.1 começou a apresentar indisponibilidade com 600 acessos simultâneos, isto é, se a UniEvangélica de Anápolis possui aproximadamente 6.215 alunos matriculados no turno noturno a primeira versão do AVA suportou menos de 10% do turno.

Para a execução dos testes de *stress* é necessário o uso de uma ferramenta que simula uma grande quantidade de usuários virtuais acessando simultaneamente para analisar até onde o sistema suporta, para isso, dentre as diversas ferramentas este trabalho escolheu o BlazeMeter que é uma ferramenta que cria o *script* gravando os passos acessados no sistema e o Jmeter que executa esse *script* e simula o teste com as devidas configurações necessárias. Ambas ferramentas foram escolhidas por serem gratuitas e intuitivas, além de gerar resultados claros para análise.

Em seguida, foi elaborado um protótipo do processo para a evolução para padronizar e planejar este trabalho Imagem 6, com isso, foi seguido o passo a passo que consta no protótipo na ordem apresentada. Ao seguir o protótipo e a chegada da etapa da execução dos testes de *stress* no Ambiente Virtual de Aprendizagem (AVA), os testes foram feitos na versão atual do AVA da UniEvangélica e na versão anterior, porém, na primeira versão, o *firewall* bloqueou e impediu a execução total dos testes, com isso, os resultados do ambiente v.1 são resultados parciais, onde pode ser observado na Seção 5.4, conforme a entrevista, esse barramento era esperado.

Já os testes no ambiente v.2, como já esperado pela equipe de análise, os testes mostraram que o sistema consegue suportar a carga total de quantidade de alunos matriculados no turno noturno, além disso, a equipe do AVA esperava que o sistema apresentasse lentidão

após 5000 acessos e todos os testes superiores esse valor, apresentou lentidão, mas não gerou indisponibilidade.

Diante do objetivo principal deste trabalho que é criação do processo de teste sendo um processo padronizado e extensível e aplicável a sistemas EAD podendo ser acrescentados testes não mencionados neste trabalho. O processo proposto é uma adaptação do processo encontrado durante a pesquisa bibliográfica, onde foi adicionado subprocessos distinguindo os testes funcionais dos não funcionais, tornando o processo mais claro e objetivo.

Seguindo o foco, o subprocesso de testes de *stress*, nele é abordado todos os passos mencionados anteriormente finalizando com a etapa de propor estratégia de disponibilidade seguido de documentação dos resultados obtidos, onde os responsáveis devem propor estratégias para situações adversas onde o servidor se aproxima do seu limite evitando problemas e assim gerenciando os riscos de indisponibilidade.

Esta pesquisa evidencia-se a necessidade de se fazer testes de *stress* em ambientes de ensino a distância, uma vez que são plataformas que podem sofrer com o alto índice de acessos simultâneos e indisponibilizar seus serviços. Salientou-se que os dois AVAs da UniEvangélica sofreram com cargas altas de usuários em seu sistema, cada um com suas especificações, e que abordar testes não funcionais, como teste de *stress* no processo de teste pode ser uma alternativa para solucionar o problema.

Por fim, faz-se uma reflexão para futuros trabalhos, nos quais poderão acrescentar os outros tipos de testes de performance, como Teste de Carga e Teste de Capacidade, mencionados no processo da Imagem 20. Além disso, em sistemas completos, os testes não devem permanecer apenas em testes de desempenho, como evidenciado no processo da Imagem 19. Sendo assim, para que um processo de testes não funcionais esteja completo, é necessário implementar testes de segurança, testes de usabilidade, testes de confiabilidade, entre outros.



## 7. REFERÊNCIAS

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

ABNT/CB-26; **ISO**. 2012. Disponível em: <https://www.cb26.org.br/iso>. Acesso em: 12 nov. 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](https://www.gov.br/anatel/pt-br/dados/acompanhamento/relatorios-de-acompanhamento/2021#R2021_5). Acesso em: 21 maio 2021.

APACHE, S. F. **Apache Foundation**. 2019. Disponível em: [http://jmeter.apache.org/usermanual/jmeter\\_proxy\\_step\\_by\\_step.html](http://jmeter.apache.org/usermanual/jmeter_proxy_step_by_step.html). Acesso em: 22 set. 2021.

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

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (**Inep**). Censo da Educação Superior 2020: notas estatísticas. Brasília, DF: Inep, 2022.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (**Inep**). Censo da Educação Superior 2018: notas estatísticas. Brasília, 2019.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (**Inep**). Censo da Educação Superior 2016: resumo técnico. Brasília, 2018.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (**Inep**). Censo da Educação Superior 2014: resumo técnico. Brasília, 2016.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (**Inep**). Censo da Educação Superior 2012: resumo técnico. Brasília, 2014.

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. **Revista De Trabalhos Acadêmicos - Campus Niterói**. 2014. 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 out. 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/jspui/handle/aee/16995>. Acesso em: 13 dez. 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 nov. 2021.

FREITAS, F. G.; MAIA, C. L. B.; CAMPOS, G. A. L.; SOUZA, J. T., **Otimização em Teste de Software com Aplicação de Metaheurísticas**. Revista de Sistemas de Informação da FSMA n. 5 (2010) pp. 3-13. Disponível em: [http://www.fsma.edu.br/si/edicao5/FSMA\\_SI\\_2010\\_1\\_Estudantil\\_1.pdf](http://www.fsma.edu.br/si/edicao5/FSMA_SI_2010_1_Estudantil_1.pdf). Acesso em: 20 set. 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 nov. 2021.

GUEDES, Marylene. **Ciclo de vida do software: por que é importante saber?**. 2018. Disponível em: <https://www.treinaweb.com.br/blog/ciclo-de-vida-software-por-que-e-importante-saber>. Acesso em: 02 de julho de 2022.

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 nov. 2021.

HALILI, Emily H. **Apache JMeter: A practical beginner's guide to automated testing and performance measurement for you websites,** 2008. Disponível em: [http://download.51testing.com/ddimg/uploadsoft/20131113/ApacheJMeter\\_English.pdf](http://download.51testing.com/ddimg/uploadsoft/20131113/ApacheJMeter_English.pdf). Acesso: 14 out. 2021.

ISTQB. **Certified Tester Syllabus: Performance Testing.** 2018. Disponível em: [https://bstqb.org.br/b9/doc/syllabus\\_ct\\_pt\\_1.0br.pdf](https://bstqb.org.br/b9/doc/syllabus_ct_pt_1.0br.pdf). Acesso em: 18 set. 2021.

MACORATTI, José Carlos. **.NET - O ciclo de vida do desenvolvimento de Software.** 2017. Disponível em: [https://www.macoratti.net/17/09/net\\_slcd1.htm](https://www.macoratti.net/17/09/net_slcd1.htm). Acesso em: 2 jul. 2022.

MASCHERONI, Maximiliano A.; IRRAZÁBA, Emanuel. Continuous Testing and Solutions for Testing Problems in Continuous Delivery: A Systematic Literature Review. **Computación y Sistemas**, Vol. 22, No. 3, 2018, pp. 1009–1038 doi: 10.13053/CyS-22-3-2794.

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: 1 nov. 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 out. 2021.

MOODLE é um ava (ambiente virtual de aprendizado)? **Estúdio Site,** 2021. Disponível em: <https://www.estudiosite.com.br/site/moodle/o-moodle-e-um-ava>. Acesso em: 1 maio 2022.

MORAIS, Izabelly.Souares. D.; ZANIN, Aline. **Engenharia de *software***. Porto Alegre: Grupo A, 2020. 9788595022539. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595022539/>. Acesso em: 16 nov. 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 nov. 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 out. 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](https://www.researchgate.net/figure/Figura-1-Processo-de-testes-de-software-da-instituicao_fig1_267841817). Acesso em: 13 nov. 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 set. 2021.

SANTOS, Ismayle de Sousa. Neto, Pedro de Alcântara dos Santos. **Automação de testes de desempenho e estresse com JMeter**. Disponível em: <https://docplayer.com.br/2863873-Automacao-de-testes-de-desempenho-e-estresse-com-o-jmeter.html>. Acesso em: 14 out. 2021.

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

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

SOUSA, Angélica Silva de; OLIVEIRA, Guilherme Saramago de; ALVES, Laís Hilário. **A Pesquisa Bibliográfica: Princípios E Fundamentos**. 2021. Disponível em: <https://revistas.fucamp.edu.br/index.php/cadernos/article/view/2336>. Acesso em: 5 jun. 2022.

SOUSA, João Vitor Moreira de; SANTANA, Lucas Nunes. **Processo Mínimo De Teste De Performance Para Fábricas De Software Com Jmeter**. 2020. TCC (Graduação) – Curso de Engenharia de Computação, Centro Universitário de Anápolis – Unievangélica, Anápolis, 2020. Disponível em: <http://repositorio.aee.edu.br/jspui/handle/aee/17209>. Acesso em: 16 maio 2022.

SOUZA, Eduardo Ferreira de. **Informações sobre o AVA**. Entrevista concedida a Gustavo Campos de Andrade. Anápolis, 17 jun. 2022. [A entrevista encontra-se transcrita no Apêndice "C" deste trabalho].

SOUZA, Thiago Silva de. **Testes de desempenho de software: Teoria e Prática**. 2018. Disponível em: <https://sol.sbc.org.br/livros/index.php/sbc/catalog/download/7/13/391?inline=1>. Acesso em: 15 out. 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: 1 nov. 2021.

## 8. ANEXOS

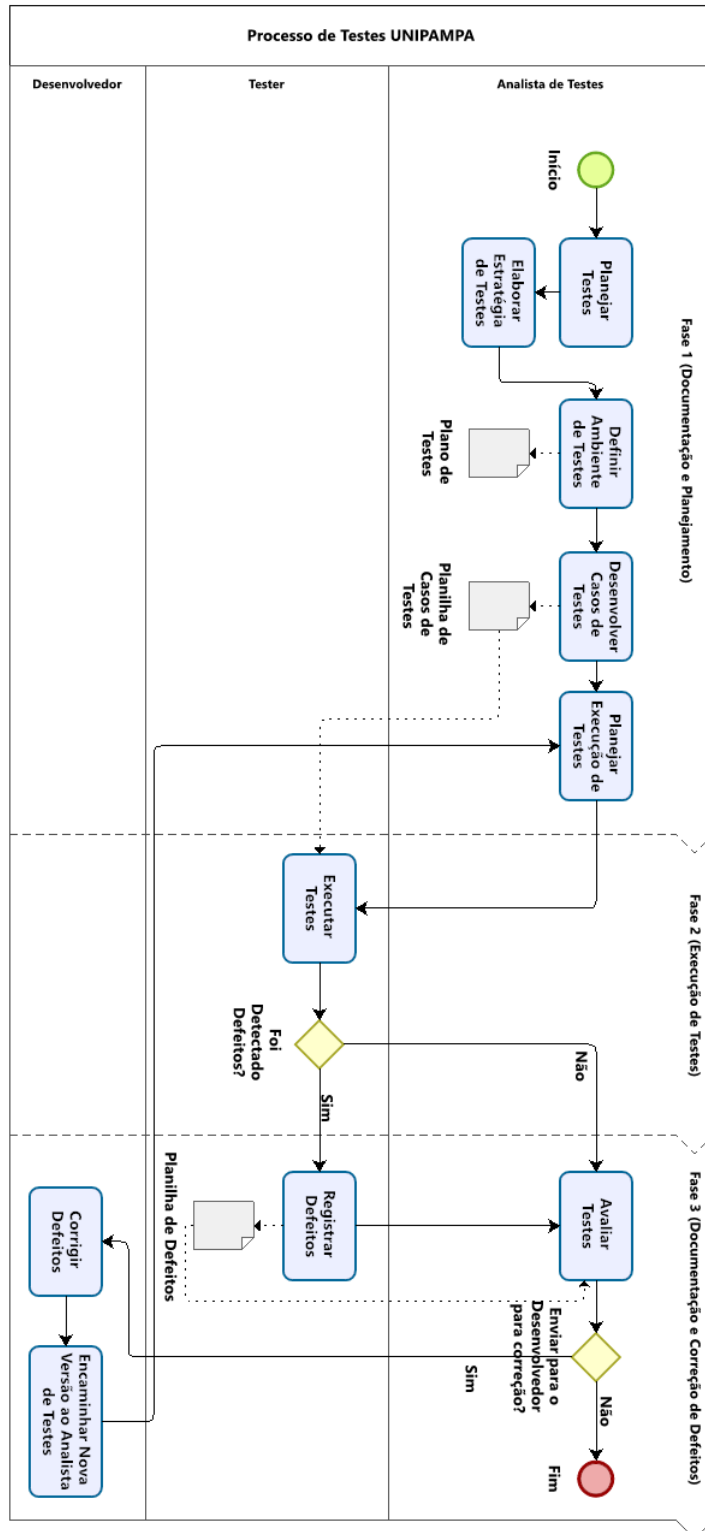


Imagem 20 - Processo de teste UNIPAMPA

9. APÊNDICES

APÊNDICE A – Processo Geral de Teste

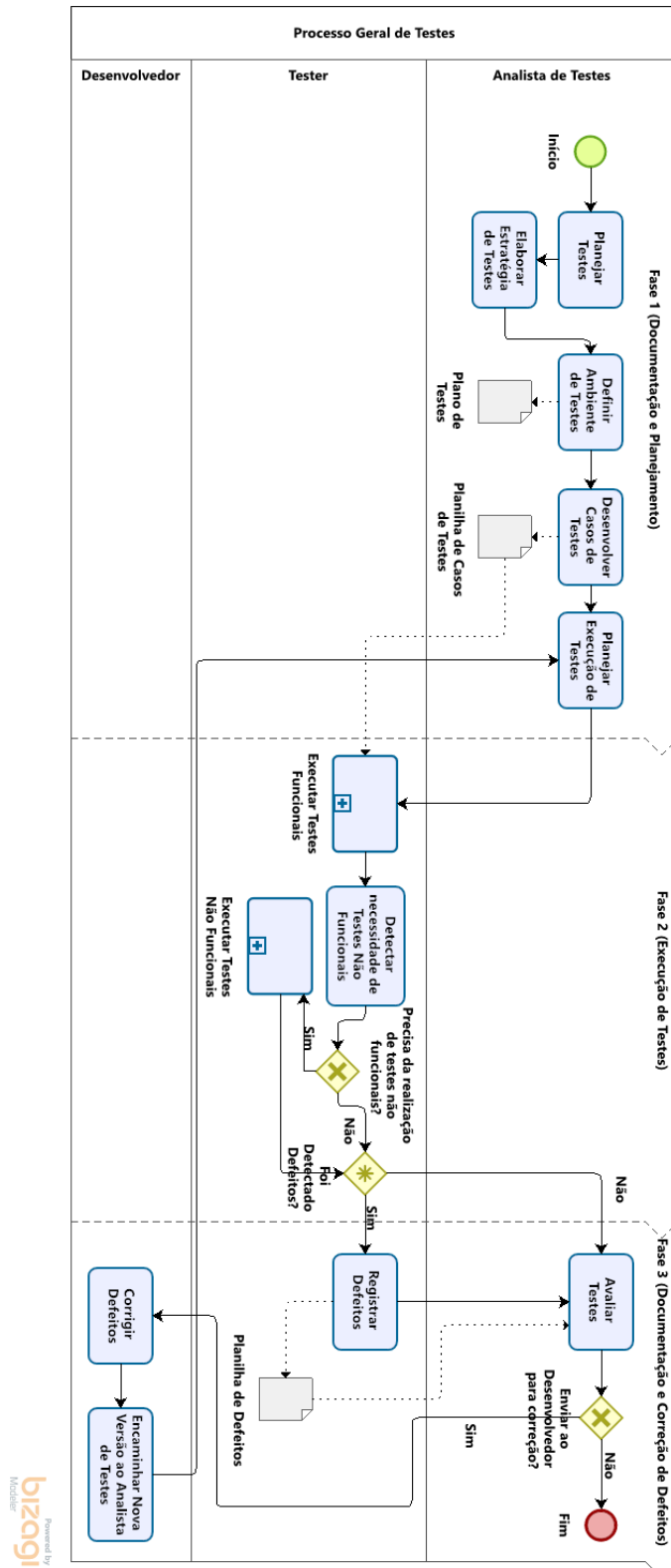


Imagem 21 - Processo Geral de Teste

**APÊNDICE B – Plano de Teste**

**UNIVERSIDADE EVANGÉLICA DE GOIÁS – UNIEVANGÉLICA  
ENGENHARIA DE SOFTWARE**

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

Plano de teste

Ambiente Virtual de Aprendizado da UniEVANGELICA

Plano de teste

Teste

Versão 1.0



**HISTÓRICO DE MODIFICAÇÕES**

<b>Versão</b>	<b>Data</b>	<b>Responsável</b>	<b>Estado atual</b>
1.0	20/01/2022	Alexandre Kamimura	Verificação de conteúdo para teste.
1.1	04/06/2022	Alexandre Kamimura	Reestruturação do plano de teste.

## **ÍNDICE**

### **Introdução**

1.1 Objetivos

1.2 Restrições do plano de teste

2.0 Teste de stress

2.1 Atividades de teste

2.2 Cronograma para atividades de teste

2.3 Responsáveis pelas atividades de teste

## **1. INTRODUÇÃO**

Afim de realizar testes no sistema AVA, esse plano de teste será utilizado para realizar o acompanhamento de atividades de teste de estresse visando identificar prováveis defeitos na aplicação do Ambiente Virtual de Aprendizagem. A ferramenta escolhida para realizar os testes é o JMeter.

### **1.1. Objetivos**

O primeiro objetivo da realização do teste define como identificar defeitos de desempenho no sistema em teste.

O segundo objetivo propõe definir as correções necessárias para uma aplicação com melhor desempenho.

O terceiro objetivo é propor melhorias a partir da identificação desses erros.

### **1.2. Restrições do plano de teste**

O sistema será testado somente nas funcionalidades mais requisitadas da aplicação, como por exemplo Login e Realização de provas, simulando a utilização dessas funcionalidades por muitos usuários ao mesmo tempo. Além disso, deve ser realizado teste de carga e teste de stress no sistema assim explorando suas inconsistências.

## 2. TESTE DE STRESS

O teste de stress será realizado para que consigamos identificar possíveis falhas na aplicação, simulando uma grande quantidade de usuários realizando várias requisições ao mesmo tempo.

<b>Técnica Utilizada</b>	Teste de carga abrangendo cargas de trabalho extremas
<b>Objetivo</b>	Esses testes são executados para compreender melhor como e em quais áreas o sistema será dividido, para que os planos de contingência e a manutenção de atualização possam ser planejados com bastante antecedência
<b>Estratégias</b>	A estratégia do teste define-se com o Teste de stress típico que consiste em simular cargas acima do esperado do sistema. O teste pico de stress é empregado para identificar em qual momento o sistema começa a apresentar falhas.
<b>Ferramentas Utilizadas</b>	A ferramenta escolhida para realizar esses testes foi o JMeter, que é uma ferramenta poderosa para realização de teste de desempenho, carga e stress.
<b>Critérios de Êxito</b>	Para que o teste tenha êxito, buscamos identificar falhas já propostas nas técnicas a fim de validar a estratégia pensada. Como por exemplo a possível e não possível falha do sistema durante a execução dos testes.

### 2.1. Atividades de teste

Para realizar os testes será definido as seguintes etapas:

1° - Identificar a quantidade máxima que o sistema suporta, a partir da quantidade de alunos matriculados na instituição e também a quantidade definida pela equipe de desenvolvimento.

2° - Após identificar essa quantia, devemos começar a realizar os testes a partir da ferramenta JMeter simulando as cargas.

3° - Elevar o sistema ao ápice para encontrar falhas.

4° - Analisar a quantidade de falhas encontradas explícitas na ferramenta JMeter a fim de ter um resultado mais concreto.

## **2.2. Cronograma para atividades de teste**

Os testes deverão ser realizados com prazo máximo de dois dias assim que é recebida a informação da quantidade máxima de usuários que podem acessar o sistema simultaneamente.

## **2.3. Responsáveis pelas atividades de teste**

O responsável pelos testes no sistema será o membro Alexandre Kamimura, que ficará responsável pela execução e leitura dos dados, retornando para a equipe os problemas encontrados e possíveis soluções pensadas.

## **APÊNDICE C – Entrevista com Eduardo**

Entrevista concedida por Eduardo Ferreira de Souza à Gustavo Campos de Andrade

### **1. Qual é o escopo do AVA?**

O Ambiente Virtual de Aprendizagem (AVA) empregado na UniEVANGÉLICA é o Moodle fornecido pela OpenLMS, em um ambiente cloud computing (computação em nuvens) garantindo altíssima disponibilidade e escalabilidade. Trata-se de uma ferramenta utilizada no mundo todo para promoção de aprendizagem à distância, fazendo uso de intuitividade e interface amigável ao usuário. Trata-se de um ambiente de ensino e aprendizagem que possibilita a apresentação de materiais, recursos e tecnologias apropriadas. O AVA faz uso de intuitividade e proporciona interface amigável ao usuário, permitindo o desenvolvimento da cooperação e reflexão. Na UniEVANGÉLICA EAD, o ambiente é personalizado e passa por avaliações periódicas, contando com um design moderno, projetado em favor de processos de aprendizagem, levando em consideração os seguintes aspectos: navegabilidade; acesso aos conteúdos e atividades; disposição de objetos de aprendizagem e cores agradáveis ao usuário.

De maneira complementar, são oportunizados no ambiente: live – momento de interação síncrona empregando metodologias ativas e exposição de conteúdo; webinar – momentos de palestras e eventos on-line de caráter transdisciplinar. Assim, a interação entre docentes, discentes e tutores é garantida de maneira satisfatória e ágil. O AVA é diretamente integrado ao sistema acadêmico utilizado pela UniEVANGÉLICA, o Lyceum. Possibilitando a disponibilização automática de disciplinas para o aluno, aproveitando os mesmos dados de acesso e transferindo para o sistema acadêmico informações de notas.

A disponibilização técnica e suporte do AVA é fornecida pelo departamento de Gestão de Ambientes Virtuais de Aprendizagem (AVAs) da Associação Educativa Evangélica (AEE), que tem em sua concepção a proposta de atuar de forma incessante para a incorporação de recursos modernos no processo de ensino-aprendizagem, por meio da utilização dos AVAs, objetivando promover com excelência o conhecimento, por meio da educação em seus diferentes níveis. A busca pela excelência, é, portanto, a principal diretriz para construção desta política, sempre primando pela formação de cidadãos comprometidos com a transformação social e a sustentabilidade.

### **2. Foram feitos testes não funcionais no processo do AVA atual?**

O AVA atual é disponibilizado por um fornecedor que é responsável pela garantia dos aspectos relacionados aos testes não funcionais (performance, disponibilidade).

**3. Foram feitos testes não funcionais no processo do AVA antigo?** Não foram realizados testes não funcionais no AVA antigo.

**4. Onde está hospedado o AVA atual?**

O AVA atual é fornecido pela empresa OpenLMS. Hospedado na AWS - amazon web services. Utilizando instâncias EC2 para servidor virtualizados, RDS para banco de dados, S3 para armazenamento antigo. Conta com loadbalance e auto-scaling.

**5. Onde está hospedado o AVA antigo?**

Data center da UniEVANGÉLICA, em uma unica máquina virtual VMWare. **6. O AVA é o mesmo para toda mantenedora?**

Cada instituição utiliza uma instalação do Moodle.

**7. O servidor é o mesmo para o AVA de toda a mantenedora?**

No antigo sim. UniEVANGÉLICA, FACEG, FER, FEJA, FECER, Colégios. Atualmente são instalações de infraestrutura individuais.

**8. Qual a composição do(s) servidor(es) envolvidos com o AVA? Quantos núcleos tem o processador? Quanto de memória alocada? Qual a velocidade de transferência da memória secundária (hd, ssd, raids de ssds hds)?**

Do atual não temos os detalhes. Pois é gerenciado pelo terceiro.

**9. Existe alguma estratégia para tentar garantir uma alta disponibilidade do AVA atual?**

Atualmente conta com auto-scaling e loadbalance.

**10. Existe alguma estratégia para tentar garantir uma alta disponibilidade do AVA antigo?**

Escalamento vertical, aumentando CPU, RAM e HD.

Também foi feito a migração de MySQL para SQL Server

**11. O AVA faz uso do swarm orquestrador de containers?**

Do atual não temos os detalhes. Pois é gerenciado pelo terceiro.

**12. O AVA possui implantação de kubernetes?**

Do atual não temos os detalhes. Pois é gerenciado pelo terceiro.

**13. O AVA possui partes construídas com base na arquitetura de microsserviços? Quais?**

Sim. A parte de integração com Lyceum é um microsserviço separado da aplicação principal. **14. Com quantos acessos simultâneos o AVA antigo começou a falhar?**

A partir de 600 usuários simultâneos havíamos registros de instabilidade. Durante o período de prova.

**15. O AVA atual disponibiliza algum recurso via cdn com o propósito de reduzir a carga do servidor? Quais?**

Não utiliza CDN.

**16. Quais as linguagens e frameworks usadas?**

O AVA é uma instalação do Moodle, software LMS open-source largamente utilizado. O Moodle é desenvolvido em PHP com uma arquitetura própria.

**17. Quais bancos de dados usados pelas mantenedoras? Qual a relação entre estes bancos de dados?**

Relacionados com AVA existe 3 banco de dados: 1) utilizado pelo moodle (Atual MySQL), 2) LYCEUM (SQL Server), 3) gerenciamento da integração entre AVA e LYCEUM. (SQL Server).

**18. Todas as mantenedoras usam o mesmo banco de dados?**

- 1) utilizado pelo moodle, -Individual por instituição
- 2) LYCEUM – Mesmo para todas
- 3) gerenciamento da integração entre AVA e LYCEUM. – Mesmo para todas

**19. É utilizada alguma solução para problemas de arquitetura e escalabilidade como por exemplo o redis, elasticsearch? Se sim quais? Para quais recursos?**

Conta com loadbalance e auto-scaling via AWS.

**20. Quantos alunos estão matriculados por turno?**

72% noturno

28% matutino ou Integral

**21. O AVA possui sistema de redundância, quais?**

Backups diários em diferentes destinos de toda base de arquivos e dados.

**22. Existem diferenças do AVA na intranet da faculdade? Melhor disponibilidade?**

Não.

**23. No AVA existe ou faz parte de um sistema de data warehouse?**

Sim, diariamente é feito cópia do banco de dados do AVA que é usado para relatórios. Não é consultado diretamente no banco em produção.



**24. Por fim, eu e meus colegas realizaremos o teste de stress e teste de carga nos dois ambientes AVA antigo e AVA novo, faremos o teste nas funcionalidades de prova e login. Qual é a expectativa do suporte do sistema?**

Antigo que a partir de 600 usuários apresente instabilidade e venha a ficar indisponível antes de 100 usuários. Possível que o firewall bloqueie. No AVA atual pode apresentar lentidão a partir de 5 mil usuários. Mas não vai ficar indisponível.