UNIVERSIDADE FEDERAL DO PARÁ
FACOMP - FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Eduardo Bernardo da Silva de Assis

ELABORAÇÃO DE TESTES DE DESEMPENHO DO TIPO CAIXA PRETA PARA A FUNCIONALIDADE DE RESERVA DE ESPAÇOS DO SISTEMA DE AGENDAMENTO DO CAMPUS DE CASTANHAL

Castanhal-PA
2021

UNIVERSIDADE FEDERAL DO PARÁ
FACOMP - FACULDADE DE COMPUTAÇÃO
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Eduardo Bernardo da Silva de Assis

ELABORAÇÃO DE TESTES DE DESEMPENHO DO TIPO CAIXA PRETA PARA A FUNCIONALIDADE DE RESERVA DE ESPAÇOS DO SISTEMA DE AGENDAMENTO DO CAMPUS DE CASTANHAL

Trabalho de Conclusão de Curso submetido
ao colegiado da Faculdade de Computação
da Universidade Federal do Pará, como
requisito parcial para a obtenção do grau de
bacharel em Sistemas de Informação

Orientadora: Profa. Dra. Yomara Pinheiro Pires

Castanhal-PA
2021

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

D278e de Assis, Eduardo Bernardo da Silva.
Elaboração de testes de desempenho do tipo caixa preta para a funcionalidade de reserva de espaços do sistema de agendamento do campus de Castanhal / Eduardo Bernardo da Silva de Assis. — 2021.
35 f. : il. color.

Orientador(a): Profª. Dra. Yomara Pinheiro Pires
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Pará, Campus Universitário de Castanhal, Faculdade de Sistemas de Informação, Castanhal, 2021.

1. Teste de desempenho. 2. Teste de software. 3. Engenharia de software. I. Título.

CDD 005.14

Eduardo Bernardo da Silva de Assis

ELABORAÇÃO DE TESTES DE DESEMPENHO DO TIPO CAIXA PRETA PARA A FUNCIONALIDADE DE RESERVA DE ESPAÇOS DO SISTEMA DE AGENDAMENTO DO CAMPUS DE CASTANHAL

Trabalho de Conclusão de Curso apresentado ao Colegiado da Faculdade de Computação (FACOMP) da Universidade Federal do Pará do campus de Castanhal, como requisito parcial para a obtenção do Grau de bacharel em Sistemas de Informação.

Profa. Dra. Yomara Pinheiro Pires
Orientadora-UFPA/FACOMP

Prof. Dr. Bruno Souza Lyra Castro
Membro da Banca-UFPA/FACOMP

Prof. Dr. Igor Ruiz Gomes
Membro da Banca-UFPA/FACOMP

Prof. Dr. João Claudio Chamma Carvalho
Diretor(a) da Faculdade de Computação - FACOMP

Castanhal-PA
2021

DEDICATÓRIA

Dedico este trabalho àqueles que sempre me amaram e fizeram o melhor por mim, aos meus pais José Félix e Maria Eliane.

AGRADECIMENTOS

Agradeço, Primeiramente, a Deus por tantas bênçãos ao longo de toda a minha trajetória nesta instituição que resultou no projeto aqui concretizado neste trabalho. Agradeço também a todos os meus outros familiares que de uma forma ou de outra acabaram por me ajudar. Agradeço aos meus amigos e professores da universidade pelas experiências vividas, como também a todos os demais colegas aos quais conheci ao longo dessa jornada. Gostaria de realizar um agradecimento especial à minha orientadora Yomara Pinheiro Pires, a qual sou muito grato, por ter me guiado com excelência e dedicação.

A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.
(Arthur Schopenhauer - 1851)

Resumo

O presente trabalho tem como objetivo o estudo e elaboração de testes de desempenho automatizados do tipo caixa preta para a funcionalidade de reserva de espaços do sistema de agendamento de locais do campus, dando destaque a utilização da ferramenta *open source* Jmeter. Os resultados dos testes realizados indicam que a quantidade de requisições simultâneas que o sistema suporta são em torno de 200 requisições, o que foi considerado satisfatório pelas médias diárias de demanda que o sistema recebe.

Palavras-chave: Teste de desempenho. Teste de software. Engenharia de software.

Abstract

This work aims to study and the elaboration of automated black box performance tests for the space reservation functionality of the campus locations scheduling system, highlighting the use of the open source tool Jmeter. The results of the tests performed indicate that the number of simultaneous requests that the system supports are around 200 requests, which was considered satisfactory by the daily averages of demand that the system receives.

Keyword: Performance testing. Software testing. Software engineering.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

FACOMP	Faculdade de Computação.
UFPA	Universidade Federal do Pará.
ABNT	Associação Brasileira de Normas Técnicas.
UFOPA	Universidade Federal do Oeste do Pará.
UENP	Universidade Estadual do Norte do Paraná.
ISO	Organização Internacional para Padronização. Em inglês: International Organization for Standardization.
IEC	Comissão Eletrotécnica Internacional. Em inglês: International Electrotechnical Commission.
NBR	Norma Brasileira.
HTTP	Protocolo de Transferência de Hipertexto. Em inglês: Hypertext Transfer Protocol.
FTP	Protocolo de Transferência de Arquivos. Em inglês: File Transfer Protocol.
SMTP	Protocolo de Transferência de Correio Simples. Em inglês: Simple Mail Transfer Protocol.
POP	Protocolo dos Correios. Em inglês: Post Office Protocol.
IMAP	Protocolo de acesso a mensagem da internet. Em inglês: Internet Message Access Protocol.
URL	Se refere ao endereço de rede no qual se encontra algum recurso informático. Em inglês: Uniform Resource Locator.
IP	Rótulo numérico atribuído a cada dispositivo conectado a uma rede de computadores. Em inglês: Internet Protocol.
QOS	Tratam-se de métricas para se obter qualidade em algum tipo de prestação de serviço. Em inglês: Quality of Service.

Sumário

1 Introdução

A informatização de atividades é uma tendência que vem ganhando espaço mundo afora, atividades que outrora exigiam um alto grau de esforço de diversos profissionais ou que demandavam muito espaço gerando diversos arquivos pelas mesmas; podiam ser simplesmente consideradas impossíveis de serem realizadas devido: fatores correlacionados a distâncias geográficas, tempo de execução ou mesmo a dificuldade de gerenciar uma enorme quantia de pessoas envolvidas para realizá-las.

Na atualidade essas atividades estão passando por um processo de simplificação proporcionado por avanços tecnológicos que naturalmente estão remodelando o mundo, as estruturas e até mesmo nossa sociedade tal como a conhecíamos. E como não poderia ser diferente esse processo proporcionado pelos avanços tecnológicos não seria uma exceção ao campus de Castanhal da Universidade Federal do Pará.

Diante deste contexto proporcionado pelos avanços tecnológicos advindos da modernidade, é importante que o desenvolvimento de sistemas que auxiliem nas atividades administrativas diárias, possuam características comuns e fundamentais que tragam benefícios para os seus utilizadores. Características essas que giram em torno: da redução de tempo, melhoria de desempenho, redução de gastos, automatização de processos; para assim poder eliminar erros manuais e melhorar a execução das atividades exercidas pelos usuários.

Os sistemas desenvolvidos precisam que suas funcionalidades sejam testadas para se obter o melhor desempenho, por isso os testes de software são importantes, pois garantem que defeitos de qualquer ordem sejam descobertos e solucionados antes da utilização pelo usuário, contribuindo assim para um uso eficaz do software e suas respectivas funcionalidades.

O presente trabalho trata do estudo e elaboração de testes de desempenho automatizados do tipo caixa preta para uma ferramenta web de reserva dos espaços do campus de Castanhal, que ainda se encontra em processo de desenvolvimento.

Ficou evidenciado a necessidade no que tange a utilização de uma ferramenta robusta e satisfatória que propicie uma melhora no processo de organização e reserva dos espaços, o que acarretaria também em uma redução no desencontro de informações sobre a disponibilidade dos espaços de um campus de tamanho considerável como este.

O campus da UFPA de Castanhal possui aproximadamente cerca de quatrocentos e cinquenta usuários contando: discentes, docentes e servidores. Diante deste contexto não há um processo organizado e satisfatório o suficiente a respeito da atividade de reserva de

espaços, visto que, para além das aulas em sala, parte de uma formação superior sólida, é também relacionada a atividades extraclasse que demandam a utilização de espaços da universidade como: laboratórios, quadra poliesportiva, auditório e outros.

A alocação desses espaços do campus é simplesmente feita através de planilhas do google drive; sendo assim facilmente tendencioso a ocorrência de falhas ligadas à: colisão de horários por solicitantes que desejam reservar o mesmo espaço no mesmo momento, falta de comprovante de reserva de espaço e ausência de aviso às partes sobre a reserva do requerido espaço para os demais.

1.1 Justificativa

Justifica-se a elaboração deste trabalho pela necessidade em reduzir os conflitos de agendamento de espaços do campus de Castanhal, que são realizados de forma manual através do preenchimento de planilhas. A forma como são realizadas essas reservas de espaços do campus não possui muito rigor o que acarreta em eventuais falhas de comunicação, que resultam em maus entendidos para os que solicitam a reserva de algum espaço para realizar alguma atividade.

Com o objetivo de resolver esse problema, a empresa júnior da UFPA de Castanhal, trabalha no desenvolvimento de um sistema web de reserva de espaços do próprio campus para sanar os problemas advindos da utilização de planilhas, na execução desta atividade, como foi citado anteriormente.

Alguns testes foram realizados no sistema em questão pela equipe da empresa júnior da UFPA de Castanhal, contudo ainda não se obteve um diagnóstico relacionado ao desempenho do sistema, sobretudo a sua funcionalidade de reserva de espaços. Este trabalho visa preencher essa lacuna, elaborando e realizando os testes de desempenho na funcionalidade de reserva de espaços do sistema web de reserva de espaços do campus de Castanhal, através da elaboração e execução de testes automatizados e de caixa preta.

Os testes de caixa preta, são uma categoria de testes de software que refletem os requisitos funcionais do mesmo. Como não há conhecimento sobre a operação interna do programa, o avaliador se concentra nas funções que o software deve desempenhar. A partir da especificação são determinadas as saídas esperadas para certos conjuntos de entradas de dados.

Para a resolução desta situação é imprescindível que ela tenha que perpassa pela utilização de um software e que o mesmo possua uma qualidade adequada para se obter a performance e os resultados almejados, a qualidade de um software está intimamente atrelada aos eventuais defeitos que o mesmo possa apresentar. Grande parte desse processo de obtenção de qualidade deve-se a utilização de testes padronizados e adequados ao

domínio do problema para assim poder de fato achar eventuais gargalos e outros problemas que impeçam a aplicação de obter o desempenho esperado.

No processo de ciclo de vida de um sistema, quanto mais cedo um defeito for encontrado, mais barato será o custo para repará-lo e consequentemente menos tempo será gasto, sobrando assim mais tempo livre para empregar em outras atividades demandadas ao mesmo, de acordo com Myers, autor do livro *The art of software testing* (1979), citado por (PONTES, 2009) "Tratar um problema na fase inicial do projeto pode chegar a ser cem vezes mais barato do que corrigir um defeito que se propagou até o ambiente de produção", a principal maneira de se encontrar eventuais falhas decorrentes no software é através de testes voltados a sua realidade de funcionalidades e de desempenho do mesmo.

Em suma, este trabalho de conclusão de curso justifica-se pela necessidade de explorarmos soluções tecnológicas e automatizadas, para a testagem da funcionalidade de reserva de espaços do sistema de agendamento do campus, com o desígnio de obter a resolução desta situação. É válido ressaltar que o sistema é desenvolvido pela empresa júnior de tecnologia da Universidade Federal do Pára de Castanhal (Link Jr).

1.2 Objetivos

1.2.1 Objetivos gerais

Os objetivos gerais deste trabalho, concentram-se em: novos testes específicos para o sistema em produção, contribuir no ganho de qualidade do produto de software, viabilizar a sua utilização por todos e extrair ao máximo o seu potencial. Todos esses objetivos gerais, no fim, acarretam em sanar e evitar os contratempos, anteriormente citados, para as partes envolvidas nas atividades de reserva de espaços do campus.

1.2.2 Objetivos específicos

- Realização de testes, usando a ferramenta Jmeter, relacionados ao desempenho da aplicação com o intuito de identificar algum problema.
- Obter um diagnóstico, por meio de testes de desempenho da funcionalidade de reserva de espaços do sistema de agendamento do campus de Castanhal.
- Gerar documentação sólida para futuras manutenções do sistema de agendamento do campus de Castanhal, com o intuito de facilitar a busca por alguma informação.

- Colaborar para a solução do problema das reservas dos espaços do campus, apontando os problemas presentes através da testagem.
- Inferir soluções com mais eficácia e obter um ganho de qualidade no software proposto.

1.3 Trabalhos correlacionados

Na literatura, encontrou-se alguns trabalhos correlacionados a esta pesquisa, como por exemplo o trabalho de (FARIAS et al., 2016) e de (FERRARI, 2008), ambos apresentam avaliações de desempenho de websites institucionais de suas respectivas universidades e se valendo de testes de desempenho proporcionados pela ferramenta *open source* Jmeter. Distintamente os autores tinham o mesmo objetivo em comum que era avaliar como os websites se comportavam a partir dos acessos simultâneos de um número de usuários virtuais em um ambiente controlado.

Os dois trabalhos correlacionados, igualmente, apresentam muitas métricas, porém apenas algumas foram adotadas que são: vazão, média de uma requisição e porcentagem de erro. Isso se fez necessário devido a grande quantidade de métricas proporcionadas pela ferramenta e como também, por essas elucidarem o objetivo do trabalho por completo.

É importante mencionar que os autores (FARIAS et al., 2016) e (FERRARI, 2008), realizam seus testes em mais de uma funcionalidade dos sites institucionais, enquanto no trabalho realizado apenas são feitos testes na funcionalidade de reserva de espaços do sistema de agendamento do campus de Castanhal.

Ambos os trabalhos criaram cargas de usuários diferentes, porém em dado momento do desenrolar dos trabalhos, os dois autores avaliaram os seus respectivos websites institucionais com a carga de cem usuários simultâneos, que também foi um dos valores de carga de acesso simultâneo utilizado neste trabalho para se medir o desempenho do sistema em produção. Em posse dessas informações, foi elaborada uma planilha que compara os resultados obtidos do trabalho atual com os resultados obtidos dos trabalhos correlacionados.

O trabalho de (FARIAS et al., 2016) faz a análise da página do edital,primeira linha da planilha, e da página inicial, segunda linha da planilha, do site institucional da Universidade Federal do Oeste do Pará (UFOPA). As outras três linhas seguintes que compõem a planilha referem-se aos resultados obtidos pelo teste na funcionalidade de reserva de espaços do sistema de agendamento, em turnos distintos.

As três linhas seguintes da planilha citadas abaixo, contém os resultados obtidos por esse trabalho. A primeira linha mostra os resultados para cem usuários simultâneos só que pelo período da manhã, a segunda linha mostra os resultados para cem usuários simultâneos pelo período da tarde e a terceira linha mostra os resultados para cem usuários

simultâneos pelo período da noite.

O trabalho realizado por (FARIAS et al., 2016) foca na análise de desempenho do site institucional da Universidade Federal do Oeste do Pará (UFOPA), utilizando cem usuários virtuais.

Obteve-se para a página de edital uma vazão, que é a média do número de transações por segundos em relação ao número de usuários ativos, de 10,7 segundos e para a página inicial obteve-se uma vazão de 7,8 segundos, o que é relativamente próximo ao que se encontrou ao realizar os testes de que tratam este trabalho. É válido ressaltar que a média de requisição entre os trabalhos se diferem pouco e a porcentagem de erro é constante em ambos, o que denota que ambos os sistemas web suportam uma carga de cem usuários realizando acessos ao mesmo tempo, a tabela 1.1 compara as diferenças encontradas no trabalho de (FARIAS et al., 2016) e (FERRARI, 2008) com o atual, assim como é exibido na tabela 1.1.

Tabela 1.1 – Comparação entre os trabalhos de (FARIAS et al., 2016) e (FERRARI, 2008) com o trabalho atual

Trabalho	Usuários	Turno	Vazão	Média da Requisição	% de Erro
(FARIAS et al., 2016)	100	-	10,7/seg	20 segundos	0%
(FARIAS et al., 2016)	100	-	7,8/seg	20 segundos	0%
Atual	100	manhã	4,3/seg	17,48 segundos	0%
Atual	100	tarde	4,8/seg	16,9 segundos	0%
Atual	100	noite	5,3/seg	14,16 segundos	0%
(FERRARI, 2008)	100	-	31,5/seg	1,104 segundos	0%
(FERRARI, 2008)	100	-	34,2/seg	0,436 segundos	0%
Atual	100	manhã	4,3/seg	17,48 segundos	0%
Atual	100	tarde	4,8/seg	16,9 segundos	0%
Atual	100	noite	5,3/seg	14,16 segundos	0%

Fonte: Própria (2020)

O trabalho de (FERRARI, 2008) faz a análise da página inicial, sexta linha da planilha, e da página de regulamento, sétima linha da planilha, do site institucional da Universidade Estadual do Norte do Paraná (UENP) campus Luiz Meneghel. As outras três últimas linhas que compõem a planilha referem-se aos resultados obtidos pelo teste na funcionalidade de reserva de espaços do sistema de agendamento.

O trabalho realizado por (FERRARI, 2008) foca na análise de desempenho do site institucional da Universidade Estadual do Norte do Paraná (UENP), utilizando cem usuários virtuais.

Obteve-se, para a página inicial uma vazão de 31,5 segundos e para a página de regulamento uma vazão de 34,2 segundos, um resultado relativamente acima do que se

encontrou nos testes deste trabalho. É válido ressaltar que a média de requisição entre os trabalhos se diferem e a porcentagem de erro é constante em ambos, o que denota que também nesta comparação entre o trabalho atual e o trabalho de (FERRARI, 2008), os sistemas web suportam uma carga de cem usuários realizando acessos simultâneos, a tabela 1.1 também compara as diferenças encontradas no trabalho de (FERRARI, 2008) com o atual.

O uso destes trabalhos correlacionados como referência comparativa ao trabalho que foi realizado, servem para ilustrar como se decorreu a execução dos testes de desempenho em diferentes ambientes. Os trabalhos correlacionados e o atual possuem algumas similaridades, assim como o trabalho atual em comparação aos trabalhos correlacionados possui diferenças pontuais, como: não se resumir a apenas uma carga de cem usuários simultâneos como fazem os trabalhos de (FARIAS et al., 2016) e (FERRARI, 2008).

É importante destacar, que foram encontrados outros trabalhos que salientam acerca da importância e das etapas que se fazem presentes na avaliação de desempenho do processo de software, ou mesmo da atividade de teste no desenvolvimento de software, como os trabalhos de: (GONÇALVES, 2014) e (FUKUMORI; SANTOS; MORRO, 2008); porém, neles não foram demonstrados aplicações práticas através do uso da ferramenta Jmeter.

2 Fundamentação Teórica

2.1 Relação entre testes de software e a aquisição de qualidade do produto

Ao consultar a bibliografia referente ao assunto deste trabalho, fica evidente a correlação entre teste de software e obtenção de qualidade no produto, posto isto, para se alcançar os objetivos propostos, é importante mencionar questões alusivas às maneiras de obtenção de qualidade.

Quando pretendemos estabelecer testes em um software, é importante lembrar da integração dos componentes para possibilitar a edificação do sistema. Planejar de forma cautelosa nos dá um projeto e organização robustos que implicam diretamente na garantia de seguranças em estar fazendo o processo de testagem adequadamente.

Na literatura existem vários autores que nos alertam sobre esta questão, porém nenhum deles chega a ter uma análise tão incisiva quanto (PFLEEGER, 2004) "Cada etapa do processo de teste deve ser planejada. Na verdade, o processo de teste tem vida própria no ciclo de desenvolvimento e pode ser realizado em paralelo com muitas outras atividades de desenvolvimento, especificamente, devemos planejar cada uma dessas etapas de teste: estabelecendo os objetivos do teste, projetando os casos de teste, escrevendo casos de teste, testando os casos de teste, executando os testes e avaliando os resultados dos testes".

Tudo que é mencionado por Pfleeger de certa forma relata um aspecto importantíssimo proporcionado pelos testes de software, que é a qualidade, segundo Herman G. Weinberg, mencionado por (KOSCIANSKI; SOARES, 2007) “A qualidade é relativa. O que é qualidade para uma pessoa pode ser falta de qualidade para outra”. A definição pode parecer simples, contudo se observada mais detalhadamente pode-se perceber sua complexidade, pois, um dos principais desafios enfrentados pelos profissionais de teste de software está em definir o que é qualidade no contexto do produto atual.

Para ajudar nessa questão de ganho de qualidade, a *international organization standardization* (ISO), define algumas normas, neste sentido, dentre elas a norma ISO/IEC 9126, cuja brasileira corresponde é a NBR 13596, para padronizar a avaliação da qualidade do produto de software a nível mundial. O motivo da existência de normas é para que se tenha uma uniformização nas formas com que são realizados os processos, garantindo assim uma obtenção de resultados mais fidedigna à realidade estudada, de acordo com o site (DFILITTO, 2019) "A norma define um conjunto de parâmetros que caracterizam os

atributos de qualidade distribuídos em seis características principais que são: Funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade; com cada uma dividida em algumas sub características.".

Segundo o site (DFILITTO, 2019) "As normas de qualidade da ISO/IEC 9126 giram em torno dos itens: processo de desenvolvimento, onde se tem a relação entre a qualidade conceitual e a qualidade do software gerado; o produto, compreendendo as características de qualidade de software gerado; qualidade em uso, onde se emprega a comparação da qualidade do software nos contextos específicos do usuário". Ela deixa entendido que o processo poderá ser aplicado a cada fase apropriada do ciclo de vida de cada componente de software, onde destacam-se três estágios: definição dos requisitos da qualidade, preparação da avaliação e avaliação em si.

O que foi mencionado anteriormente na obra de Pfleeger e o que se encontra contido na norma ISO/IEC 9126, que trata sobre a obtenção de qualidade, torna possível traçar uma equivalência entre essas duas informações com o manifesto ágil de software, que engloba tanto o manifesto ágil de desenvolvimento de software (o precursor das metodologias ágeis de desenvolvimento de software) quanto o manifesto ágil de teste de software.

Nas entrevistas com um dos desenvolvedores do sistema, que irá ser testado, foi percebido o uso tanto do manifesto ágil de desenvolvimento de software quanto o manifesto ágil de teste de software. Sendo que o último foi incorporado como referencial teórico, com o objetivo de nortear a execução do trabalho. É importante mencionar que o trabalho não faz uso de nenhuma metodologia de desenvolvimento ágil ou metodologia de teste ágil, e que o manifesto ágil de teste foi usado como referência, ou seja, apenas foi abordado os seus aspectos para somar ao trabalho.

2.2 Manifesto de teste ágil e testes de software convencionais

Como o sistema web de reserva de espaços do campus se encontra em desenvolvimento, é salutar a adoção de uma perspectiva voltada para o manifesto de teste ágil ao invés de uma voltada a metodologia tradicional de testes. A opção por esta perspectiva para compor o embasamento teórico dos testes de software se dá muito em virtude de o desenvolvimento da aplicação fundamentar-se no manifesto ágil e suas respectivas metodologias ágeis de desenvolvimento, como também pelos benefícios agregados em sua constituição que lhe permitem driblar eventuais problemas no ambiente de produção como: documentação ruim ou inexistente, retrabalho e por fim até mesmo testar sob pressão de prazo. Nas palavras de (PONTES, 2009) “A escola ágil enfatiza fontes de automação de testes. Vê testes como um complemento do desenvolvimento, uma atividade que indica que o desenvolvimento está finalizado. Geralmente usa desenvolvimento dirigido a testes”.

Na abordagem tradicional de desenvolvimento de testes existe uma equipe voltada exclusivamente a identificar qualquer defeito ou inconformidade do sistema antes de ser liberado para o uso, o que com o passar do tempo na visão de (PRESSMAN, 2009) "Torna-se uma atividade mais burocrática e pouco prática para fins de se obter os resultados concretos gerados pelo processo de testes, muito disso deve-se ao excedente de documentação gerada, o que tornar-se com o passar do tempo um artefato de documentação cuja origem remonta ao processo anterior", diferentemente, no manifesto ágil de testes e suas respectivas metodologias onde a documentação é feita sob medida evitando excedente de documentação gerada e focando mais em resultados para assim desburocratizar o processo.

É válido ressaltar que o desenvolvimento de testes ágeis não espera até o fim da produção do sistema para poder iniciar os testes como em metodologias tradicionais, a ideia é desenvolver e ir testando, o que para um sistema ainda em produção, como este, é excepcional. Há uma grande possibilidade de se reduzir o retrabalho de procurar e consertar erros ocasionados em fases anteriores, e os testes ágeis por estarem presentes desde o início e se encontrarem no decorrer das etapas de desenvolvimento envolvem tanto o responsável pelos testes quanto a equipe que codificou o sistema, visto que, a troca de informação pelas partes ajuda na dinamicidade geral do processo.

É importante mencionar que o que se questiona neste ponto é apenas a forma com que geralmente os testes, sob uma perspectiva convencional, são empregados e sua inviabilidade para o sistema em questão, e é apontado por meio de uma perspectiva ligada aos valores do manifesto ágil de testes, sem se aprofundar em uma metodologia específica de teste, uma referência de como guiar esse processo.

Não é de interesse da perspectiva ágil reinventar os tipos de testes de software, mesmo porque os tipos são os mesmos da metodologia tradicional o que muda é a forma e o valor empregado ao teste, isso fica evidenciado na obra de (CRUZ, 2018) "Um importante detalhe que precisa ser reforçado é que requisitos e metodologias de teste não se diferem muito entre testes convencionais e testes ágeis; o que os diferencia mais é a importância dada aos testes e os processos em que estes são aplicados". Como, por exemplo, um dos modelos de metodologia tradicional mais populares, o modelo *waterfall*, que tem uma fase de testes localizada apenas no final do projeto, diferenciando da perspectiva ágil no sentido de caber ao profissional de testes participar ativamente de todo o processo, desde o planejamento, ao desenvolvimento e finalmente os testes propriamente ditos, ao invés de serem realizados todos no final do projeto (ou mesmo no final da iteração).

No manifesto ágil de desenvolvimento temos a declaração de valores que com o decorrer se tornam princípios essenciais no desenvolvimento de aplicações e do ciclo de vida ágil, *the testing manifesto* é o manifesto voltado a práticas de testes dinâmicas durante o ciclo de vida, os valores declarados pelo manifesto de teste ágil são: testar por todas as etapas versus testar no fim, prevenir *bugs* versus encontrar *bugs*, testar o entendimento

versus checar funcionalidades, construir o melhor sistema versus quebrar o sistema, time responsável pela qualidade versus responsabilidade dos testadores; e esses mesmos valores se tornaram princípios que nortearam a elaboração tanto do plano de testes quanto da realização do teste em si neste trabalho.

2.3 Aspectos gerais sobre aplicação web

A aplicação web diz respeito a uma solução que é executada diretamente no *browser* (ou navegador), não sendo preciso realizar uma instalação na máquina do usuário, as aplicações web podem ser definidas de acordo com o site (DIEGOMACEDO, 2018), "Em essência, uma aplicação web é um software que é instalado em um servidor web e é projetado para responder a solicitações, processar informações, armazenar informações e dimensionar as respostas de acordo com a demanda e, em muitos casos, é distribuído em vários sistemas ou servidores.". As redes sociais e o sistema de agendamento de espaços do campus, são alguns dos exemplos que se enquadram nesse perfil.

Uma aplicação web pode ser categorizada tendo como princípio a utilização de um servidor (computador que é utilizado por computadores comuns e por outros aparelhos que acessam a internet para prover os mais variados serviços possíveis), exemplo: ao entrar no sistema de agendamento de espaços do campus, a página acessada antes de vir para seu navegador, é processada em um servidor ligado a internet que irá retorna o processamento das regras de negócio em questão. Sendo assim a função do servidor web é receber uma solicitação (requisição) e devolver (resposta) algo para a máquina que o solicitou.

Para que uma aplicação web funcione, ela depende de um servidor web, de solicitações realizadas pelos usuários, do uso de protocolos e métodos (normalmente o HTTP) e da resposta do protocolo.

A aplicação deve permitir que as pessoas usuárias consigam fazer uma solicitação e receber algo em resposta. Ou seja, elas precisam mediar essa interação de forma natural, devolvendo o que a pessoa deseja como resultado, por exemplo, se a pessoa pede para abrir uma foto, é preciso que isso seja devolvido à ela, e não a abertura de uma página aleatória.

Segundo o blog (BETRYBE, 2020), "O servidor tem por função receber a solicitação do público e devolver uma resposta para a aplicação. A resposta pode ser a abertura de uma nova página, imagens, documentos, entre outros. Outro componente da aplicação web é o protocolo HTTP. Ele é como uma espécie de “linguagem” que determina o padrão pelo qual a solicitação realizada se comunica com o servidor. Outros protocolos envolvidos em aplicações web podem ser: FTP (transmissão de arquivos), SMTP (envio de mensagens para um servidor de email), POP e IMAP (acesso a mensagens de email eletrônico).".

Uma das principais vantagens é que, como a maioria domina o uso dos navegadores, a

usabilidade dessas soluções é tranquila para o público, sem a necessidade de um treinamento detalhado sobre como utilizar a aplicação.

Todas as atualizações necessárias são feitas por meio de um único servidor central, não sendo preciso baixar aplicações ou realizar reinstalações. Logo, temos tudo feito de forma centralizada, sem maiores problemas, bastando realizar a adaptação no servidor.

2.4 Visão geral acerca dos testes voltados para aplicações web

Quando nos referimos a obtenção de qualidade em um projeto é imprescindível que o mesmo passe por processos de testes. Acima de tudo, a realização de testes tem por objetivo encontrar erros que foram ocasionados durante o processo de codificação do sistema. segundo (BARTIÉ, 2002) citado por (BRAGA et al., 2013) "Os testes são um processo de detecção de erros de software que verificam o comportamento do software através de diversos cenários de testes, descobrindo bugs. Ele também afirma que o ideal é executar os testes em todas as fases da engenharia de software para garantir um produto com qualidade. Kaner; Falk e Nguyen acreditam que ao realizar os testes de software, um dos principais benefícios é que o resultado é a melhora da qualidade, onde bugs e erros são concertados.".

No decorrer do processo inevitavelmente o foco dos casos de teste deve mudar entre os diferentes componentes da arquitetura do software em questão. Porém, devido a grande quantidade de conteúdo que um sistema possui e os seus diferentes módulos, deve-se estabelecer um ponto de partida para o começo dos testes.

O ponto de partida para os testes de software, geralmente são os testes que envolvem o código fonte da aplicação, também conhecidos como: testes de caixa branca, com o objetivo de analisar a performance ideal das funcionalidades estabelecidas. Nas palavras de braga (BRAGA et al., 2013) "Podem existir erros lógicos que podem ser infiltrados quando implementado funções, condições ou controles que estejam fora da função principal e, ao testarmos situações cotidianas o processamento fica bem entendido, porém podem ocorrer casos especiais que poderiam ser detectados ao realizar teste. Podem existir situações que acreditamos que um caminho lógico nunca será executado quando na verdade é executado regularmente, onde o fluxo lógico do programa pode ser contra intuitivo e pressupomos que os fluxos de dados seguirá determinado caminho, mas nos enganamos e só descobrimos quando se realiza os testes. Também podem ocorrer erros de digitação que passam sem ser detectados pelos mecanismos de verificação de sintaxe e só serão detectados ao iniciar os testes."

Em se tratando do sistema de agendamento do Campus, os testes de funcionalidade e integração já foram realizados pelos desenvolvedores durante o processo de codificação, e o sistema já se encontra operando, possivelmente os testes necessários em suas funcionalidades

lograram exito, o que nos força a mover os esforços para outras categorias de testes de software, como os testes de desempenho.

2.5 Fundamentos para um teste em uma aplicação

Todo bom teste começa com a definição de um excelente plano de testes. O plano de testes visa colaborar com o desenvolvimento de um software através da organização e documentação eficaz dos componentes: técnicos, funcionais e estruturais; que serão verificados e validados, de modo a garantir o bom funcionamento do programa junto ao usuário final. Sendo assim, um plano de testes de software tem como foco garantir a confiabilidade e segurança de um software, identificando possíveis erros e falhas durante a sua confecção.

Independentemente de qual o tipo de teste de software pretendido o fundamento principal deles é possuir alta probabilidade de encontrar erros, mas além disso um bom teste deve mesclar outras características gerais como: não ser redundante, ser o mais completo possível e não ser muito longo como também não ser muito curto. O tipo de teste adotado tem que possuir algumas propriedades comuns a outros tipos de testes o que garante que os mesmos permitam lograr êxito na busca por o maior número de erros possíveis, abaixo seguem-se algumas dessas propriedades que para (PRESSMAN,2009) são fundamentais para a atividade de realização de testes de uma aplicação.

- Compreensibilidade. Característica ligada à informação, quanto mais informações possuirmos, mais inteligente será o teste. A documentação técnica é instantaneamente acessível, bem organizada, específica, detalhada e precisa neste tipo de característica.
- Estabilidade. Característica que um teste deve possuir que diz respeito ao controle do número de alterações realizadas no software, as alterações devem ser controladas justamente para não atrapalhar a execução do teste.
- Decomponibilidade. Diz respeito ao controle do escopo do teste com o intuito de isolar os problemas rapidamente a fim de se realizar o novo teste de forma mais racional. Tendo em mente que o sistema é construído em módulos separados e que os mesmos podem ser testados da mesma maneira.
- Controlabilidade. Característica ligada ao controle do software. Todas as possíveis saídas podem ser geradas por meio de alguma combinação de entrada, e os formatos de entrada e saída são consistentes e estruturados.

Todos os testes de software levam em consideração em algum nível as propriedades comuns a todos citadas acima. Existem dois grandes grupos de testes mais abrangentes,

que servem de base teórica e a partir deles se originam os testes mais específicos, são eles: testes de caixa preta e testes de caixa branca.

2.6 Testes de caixa preta e testes caixa branca

O teste de caixa branca, também conhecido como teste orientado à lógica ou teste estrutural, é um dos dois grandes grupos de casos de testes. Usa as estruturas pertencentes ao projeto como componentes para poder assim derivar os casos de testes. Partem do aspecto interno do programa, o código fonte, para poder realizar a avaliação dos componentes. Para (BARTIÉ, 2002) citado por (BRAGA et al., 2013) "Os testes de caixa branca são baseados na arquitetura interna do software. Eles empregam técnicas que tem como objetivo identificar defeitos nas estruturas internas de programas.". Geralmente itens como: fluxo de dados, estruturas de controle, ciclos e condições lógicas são objetos aos quais esse grupo de teste se direciona.

O teste de caixa preta, também conhecido como teste comportamental, é o outro dos dois grandes grupos de casos de testes. Suas técnicas são voltadas a analisar aspectos externos com o intuito de se verificar a saída dos dados usando entradas de vários tipos, diferentemente do teste de caixa branca que é voltado para o código fonte, estes servem para derivar séries de condições de entrada que utilizam por completo os requisitos funcionais em um programa.

De acordo com (BRAGA et al., 2013) "O teste de caixa preta é focado nos requisitos funcionais do software, sendo assim, ele visa garantir que o sistema esteja de acordo com a especificação funcional, de forma que, atenda todos os requisitos do sistema. Onde o principal objetivo é garantir que o que foi desenvolvido está de acordo com o esperado, sendo que apenas o conhecimento das entradas e saídas possíveis para o programa é necessário e, ao contrário do teste de caixa branca, desconsiderando a arquitetura interna do software.".

Os testes de caixa preta não são alternativas às técnicas de caixa branca, pelo contrário, são abordagens complementares que disponibilizam a possibilidade de se descobrir uma variedade de erros distintos aos que são encontrados através dos testes de caixa branca e exige que o sistema já esteja com algumas partes já feitas.

Como o sistema de agendamento do campus, está em processo de desenvolvimento e já tem partes concluídas e funcionando e também ainda não possui parâmetros que atestem de forma mais ampla o seu comportamento esperado, na prática, somente alguns tipos de testes poderão ser empregados em função da atual fase de desenvolvimento que se encontra.

Como os primeiros testes que comumente devem ser realizados para um sistema

giram em torno do grupo de testes de caixa branca, que tem como ênfase o código fonte, e os mesmo já foram realizados no sistema de agendamento do campus, como mencionado em entrevistas com os seus desenvolvedores: teste de unidade, teste de integração e o teste funcional; optou-se neste trabalho por priorizar a realização dos testes vinculados ao grupo de caixa preta. Abaixo segue-se a descrição mais detalhada de qual teste será abordado no escopo do trabalho, e que também é compatível com a atual fase que encontra-se o desenvolvimento do sistema de agendamento do campus.

2.6.1 Teste de desempenho

Com a crescente disseminação de informação e a popularização do acesso a internet, as empresas e as pessoas estão tendendo a impulsionar cada vez mais investimentos na criação de ambientes virtuais para disponibilizar informações. Com isso cresce a possibilidade de múltiplos usuários realizarem acessos simultâneos para fazerem uso de vários serviços na web, por isso esses sistemas devem possuir a capacidade de suportar uma demanda alta de acessos, e uma forma de se verificar o comportamento das aplicações é através da avaliação de desempenho.

Para o site (TIESPECIALISTAS, 2015)[1] "Além de apresentar ferramentas para analisar e verificar se os aplicativos ou sistemas conseguem suportar condições de alta carga e com um grande número de transações, sejam clientes ou volumes de dados. A fim de avaliar a sua robustez quando submetido a uma carga acima do previsto, assim como determinar o seu limite de capacidade com um tempo de resposta considerado aceitável para o sistema e sua infraestrutura.".

A partir dessa avaliação primária fornecidas pelos testes de desempenho são realizadas aferições mais complexas que envolvem: a disponibilidade do sistema, critérios de desempenho, características de desempenho de vários sistemas ou configurações de sistema, a fonte de problemas de desempenho e os níveis de rendimento. O que foi mencionado anteriormente é evidenciado ao analisar um trecho escrito por (PRESSMAN, 2009) "No começo das iterações de arquitetura, os testes de desempenho têm o foco na identificação e eliminação de gargalos de desempenho relacionados à arquitetura. Nas iterações de construção, tipos adicionais de testes de desempenho são implementados e executados para ajustar o software e o ambiente (otimizando o tempo de resposta e os recursos) e para verificar se a aceitabilidade dos aplicativos e do sistema manipula condições de alta carga e stress, como um grande número de transações, clientes e ou volumes de dados".

[1] https://www.tiespecialistas.com.br/teste-de-desempenho-de-software/

3 Metodologia

Através de uma demanda no campus de Castanhal da Universidade Federal do Pará que consistia em uma melhora no processo de alocação de espaços físicos do próprio, chegou-se a uma proposta de um modelo de testes para a funcionalidade de reserva de espaços de um sistema de agendamento, em desenvolvimento, inspirado no manifesto ágil de testes, com testes de desempenho por meio da ferramenta automatizada Jmeter. Caracterizando a natureza da pesquisa como aplicada, em virtude, de se buscar a geração de conhecimento para a aplicação prática e dirigida à solução de problemas que contenham objetivos anteriormente definidos.

As ferramentas automatizadas permitem que testes de caixa preta sejam efetuados e elaborados de forma menos invasiva a aplicação, os testes dos quais esse trabalho trata pertencem a uma categoria de testes de software conhecidos como caixa preta, cujo os alvos são as saídas dos dados usando entradas de vários tipos. Obtendo o ganho de qualidade que permitirá a utilização do sistema, assim como facilitando a elaboração de seu respectivo plano de testes que conta com o cenário e caso de testes, para poder corroborar a proposta de solução apresentada.

Através do uso da ferramenta automatizada neste trabalho, é possível obtermos os valores que utilizamos na análise de desempenho da funcionalidade de reserva de espaços do sistema de agendamento do campus, como: número de amostras, média de requisições, valor mínimo e máximo de requisições, desvio padrão, quilobytes enviados, quilobytes recebidos, porcentagem de erro em requisições, vazão e a média de bytes. Devido a todos esses dados mencionados obtidos por meio da ferramenta automatizada utilizada, a abordagem da pesquisa acaba se caracterizando como quantitativa.

Diante deste contexto, este trabalho é classificado como uma pesquisa exploratória, visto que, um dos grandes intuitos do mesmo é proporcionar mais familiaridade sobre o assunto ao qual se pretende investigar, problema de reserva de espaços da Universidade Federal do Pará no campus da cidade de Castanhal, o que na visão de (PRODANOV; FREITAS, 2013) "Tem como finalidade proporcionar mais informações sobre o assunto que vamos investigar, possibilitando sua definição e seu delineamento, isto é, facilitar a delimitação do tema da pesquisa; orientar a fixação dos objetivos e a formulação das hipóteses ou descobrir um novo tipo de enfoque para o assunto. Assume, em geral, as formas de pesquisas bibliográficas e estudos de caso.".

Para a realização da pesquisa exploratória foi montado um levantamento bibliográfico que inclui: livros, revistas e artigos; assim como entrevistas com um dos membros participantes da empresa júnior (Linkjr) da Universidade Federal do Pará do campus de

Castanhal, que são os responsáveis pelo desenvolvimento do sistema de agendamento do campus, e foi montado o estudo de caso mediante a isso.

O trabalho faz uso do estudo de caso, a partir de experimentos controlados em uma ferramenta automatizada chamada Jmeter, um estudo de caso é uma estratégia de pesquisa científica, que analisa um fenômeno real considerando o contexto em que está inserido e as variáveis que o influenciam. Analisando as variáveis fornecidas pelo mesmo poderemos atestar com mais exatidão a real situação, além da utilização de análise estatística de resultados.

Nesta pesquisa foram ouvidos alguns estudantes do campus e um dos membros da empresa Júnior de tecnologia da universidade (linkjr). O período da pesquisa se decorreu em dois messes, de maio de 2020 a julho de 2020, onde nesses dois meses, foram definidos: à analise dos materiais bibliográficos que fariam parte do trabalho, a entrevista com o membro da equipe de desenvolvimento da empresa Júnior de tecnologia da universidade (Linkjr) e a realização dos testes de desempenho no sistema.

3.1 Casos e cenários de testes

Quando se trabalha com testes de software, além de estabelecer os tipos de testes que serão exercidos é importante projetar de forma eficiente os casos e cenários de testes, sendo esses importantes aspectos levados em conta por diversos autores que os consideram como chaves para um teste bem sucedido.

Uma boa descrição para o que vem a ser um cenário de testes se encontra no site (PRIMECONTROL, 2019) "Um cenário de teste é uma descrição de um objetivo que o usuário pode encontrar ao utilizar o programa. Um exemplo seria 'Testar se um usuário consegue deslogar do programa ao fechá-lo'. Tipicamente, um cenário de teste vai precisar de diferentes tipos de testes para garantir que o objetivo tenha sido bem testado.".

Outra descrição, só que para casos de testes, pode ser encontrada no mesmo site (PRIMECONTROL, 2019) "Os casos de teste descrevem uma ideia específica a ser testada, sem detalhar os dados necessários e etapas exatas a serem executadas. Por exemplo, um caso de teste poderia ser 'Testar se um código de desconto pode ser aplicado em um produto em promoção'. Isso não descreve quantos vão ser códigos ou como serão utilizados. A forma de testar este caso pode variar de tempos em tempos.".

Fazendo uma analogia: enquanto os cenários de testes se concentram em nos dizer "o que" deverá ser testado, os casos de testes apontam para "como" irá ser testado, em estabelecer os meios e regras para efetuar o teste.

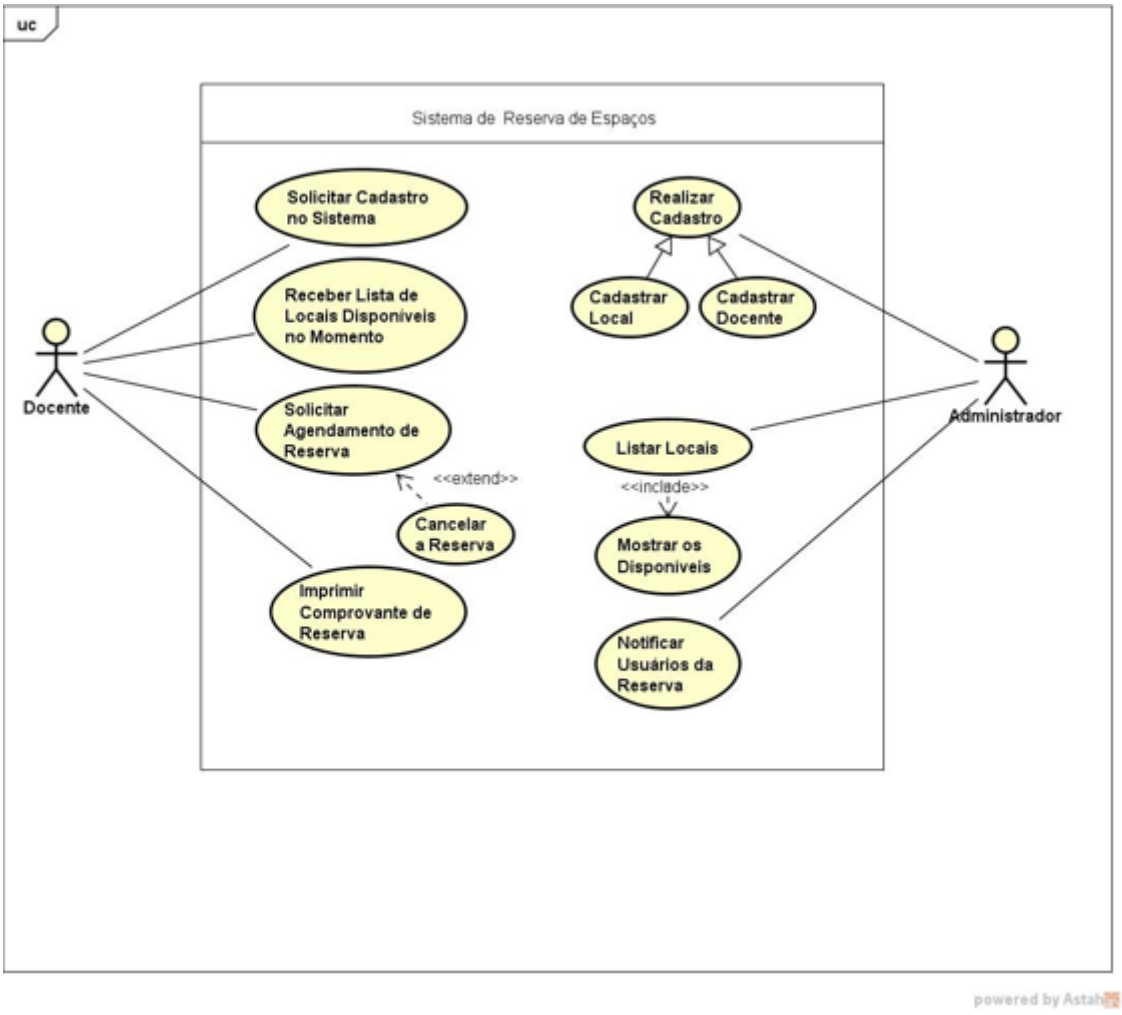
Presentemente o sistema web de reserva de espaços do campus ainda está sendo otimizado e construído. Por essa razão decidiu-se priorizar alguns testes que tem sua

origem vinculada ao grupo de testes de caixa preta, testes que elegem as saídas geradas pelo processo como foco de análise, para serem realizados em primeiro plano em vez dos de caixa branca; sobretudo pela urgência de se observar os resultados gerados pelo processo e como também pelos testes de caixa preta serem acréscimos a os de caixa branca, visto que, ao se descobrir os erros e saídas inesperadas fica mais fácil de se identificar onde se encontra as anomalias no código fonte, assim como também contribuir com a extração do potencial máximo das funcionalidades em questão.

4 Definição dos estudos de caso para os testes de desempenho no sistema

Os casos e cenários de testes por obrigação devem ser próximos o bastante para a realidade do produto ao qual se pretende testar. Elaborar todo um processo que envolve os casos e cenários sem que os mesmos não reflitam na prática o que o sistema demanda, no fim, torna-se um desperdício. Para se ter uma direção do que se deve estabelecer é preciso primeiramente observa de forma mais holística como se comporta o processo que envolve o software e seus usuários, para isso nada melhor do que um diagrama de casos de uso para poder exemplificar como se dará a relação entre os que usarão o sistema e o sistema em si, como na figura 4.1.

Figura 4.1 – Caso de uso que exemplifica o funcionamento do sistema em produção



Fonte: Própria (2020)

Após a definição dos casos de usos do sistema, é importante mostrar o sistema em si que já foi criado até o momento, conforme ilustram as figuras 4.2 e 4.3, onde mostram respectivamente a interface inicial do sistema e a funcionalidade de reserva de espaços.

Figura 4.2 – Interface inicial do Sistema de Gestão da Universidade Federal do Pará Campus Universitário de Castanhal (SisCast)

Fonte:Própria (2020)

Como mostrado na figura 4.2 está é a tela inicial exibida aos usuários do sistema. O sistema em questão trata-se do SisCast (Sistema de Gestão da Universidade Federal do Pará Campus Universitário de Castanhal), o sistema de agendamento de espaços do campus é um subsistema desse sistema maior que é o SisCast, que fica localizado no lado esquerdo, onde está escrito espaços.

Figura 4.3 – Funcionalidade de reserva

Fonte:Própria (2020)

A figura 4.3 exibe o sistema de agendamento de espaços do campus, mais precisamente onde uma de suas funcionalidade, a funcionalidade de reserva de espaços, se encontra localizada, logo abaixo do *label* solicitar agendamento.

4.1 Cenário de teste de desempenho para a funcionalidade de reserva de espaços

Antes de se efetuar os testes de desempenho serão estabelecidos os casos e os cenários respectivos a funcionalidade de reserva de espaço, para se esclarecer o que pretende-se realizar com tal atividade. A ênfase neste segmento concentra-se na funcionalidade de reserva de espaços contida no sistema de agendamento do campus, visto que, é a funcionalidade fundamental do sistema, e para qual o mesmo fora projetado e também por se encontrar já operando livremente.

Algumas informações, obtidas por meio de entrevistas com um dos membros da equipe de desenvolvimento, devem ser consideradas na confecção dos casos e cenários de testes, pois nos auxiliarão a ilustrar os mesmos virtualmente pertinentes às situações presentes no domínio do software. Na opinião de (PFLEEGER, 2004) “Se os casos de teste não forem representativos e não envolverem todas as possibilidades de exercício das funções que demonstram a correção e a validade do sistema, então, o restante do processo de testes é inútil”.

Na análise do que foi escrito pela autora fica evidenciado a sua preocupação com os casos de testes, porém para se obter casos coesos é necessário primeiramente o estabelecimento de cenários que mostram o que permeia nossa aplicação.

Dentre as informações obtidas mediante as entrevistas, algumas se destacam, como o fato de semanalmente serem realizados cerca de três ou mesmo quatro pedidos de reserva de algum espaço da universidade, claro que em circunstâncias especiais como em eventos de alguma faculdade este valor aumenta para dez ou até mais pedidos para aquele período específico.

Outra informação igualmente importante é que universidade conta com cerca de 450 usuários incluindo: discentes, docentes e servidores; como já abordado no início do trabalho, embora este número seja bem além do valor real, de usuários que esse sistema realmente atenderá no dia a dia, a simulação contendo esse valor nos dimensiona de forma coerente e segura acerca dos limites suportados pela aplicação, sejam limites esses: de carga, de disponibilidade, de stress, de volume, de escalabilidade e de tolerância.

O sistema já deve estar disponível e possuir a funcionalidade de reserva de espaços já operante, outra questão importante que deve ser levada em consideração para a realização da simulação, é o horário, para se obter uma simulação que reflita com mais realidade o que se espera, devemos cobrir durante um dia específico os turnos da manhã, tarde e noite para que seja possível prosseguir com a realização sem que haja dúvidas relacionadas a legalidade do processo, como também, que possíveis questões relacionadas à influência do tráfego da rede em determinado momento sejam elucidadas através da operação em turnos distintos, a tabela 4.1 mostra o cenário proposto para o teste de desempenho.

Tabela 4.1 – Cenário do teste de desempenho

Cenário	Requisitos	Turnos e Data de Realização	Carga Utilizada Total	Resultados Esperados (Efeitos)
CN1- Acessar simultaneamente a funcionalidade de reserva de espaços.	Sistema já disponível na rede.	Realizar no mesmo dia e nos 3 turnos distintos os testes.	450 acessos simultâneos por turno.	Acessos Simultâneos bem sucedidos e a detecção de eventuais erros.

Fonte: Própria (2020)

4.2 Caso de teste para a funcionalidade de reserva de espaços

A partir do exposto na tabela 4.1, que diz respeito ao cenário que permeia o teste de desempenho, foi possível construir a tabela 4.2, referente ao caso de teste, onde é possível visualizar com mais exatidão como se decorre todo o processo que envolve a realização

detalhada do teste.

Um caso de testes mostra os caminhos percorridos por um módulo, caso de uso ou funcionalidade dentro do projeto. Serve como base para que os responsáveis pelos testes possam executar os mesmos manualmente, mas pode ser criado, também, com o intuito de automatizar os testes. Além disso, os casos de testes devem cobrir o máximo de situações possíveis. Resumindo, um caso de testes é um conjunto de ações e os resultados esperados para elas, na tabela 4.2 observamos de forma mais completa todo o processo que envolve o teste.

Tabela 4.2 – Caso do teste de desempenho

Pré-Requisito	Procedimento	Resultado Esperado
1- Sistema disponível na rede; 2- Possuir Acesso a internet; 3- Saber utilizar o Jmeter; 4- Configurar a ferramenta para os diferentes níveis de carga; 5- Possuir noções de estatística; 6- Interpretar os resultados obtidos pela ferramenta.	1- Escolher funcionalidade a ser testada; 2- Gerar casos de testes por grupos de usuários no Jmeter; 3- Iniciar com 100 usuários virtuais e adicionar mais 50 até atingir o valor de 450; 4- Utilizar requisições HTTP para os usuários virtuais; 5- Efetuar os testes nos 3 turnos (manhã, tarde e noite);	1- Exibir os resultados de forma organizada pelo relatório de sumário; 2- Construir gráficos para comparar os resultados dos testes; 3- Realizar analise nos resultados e às aferições cabíveis.

Fonte: Própria (2020)

A tabela 4.2 exibe os casos de testes, que serão utilizados nesta pesquisa para avaliar o desempenho do sistema no que diz respeito a carga a qual este é submetido pela ferramenta Jmeter. Os casos de testes, são compostos por pre-requisitos e procedimentos necessários que precisam ser satisfeitos para o alcance dos resultados esperados.

Espera-se que analisando o desempenho do sistema, tendo sua principal funcionalidade, submetida a uma carga elevada de usuários simultâneos realizando requisições, possamos definir com mais clareza os níveis de carga suportados sob estresse, e também

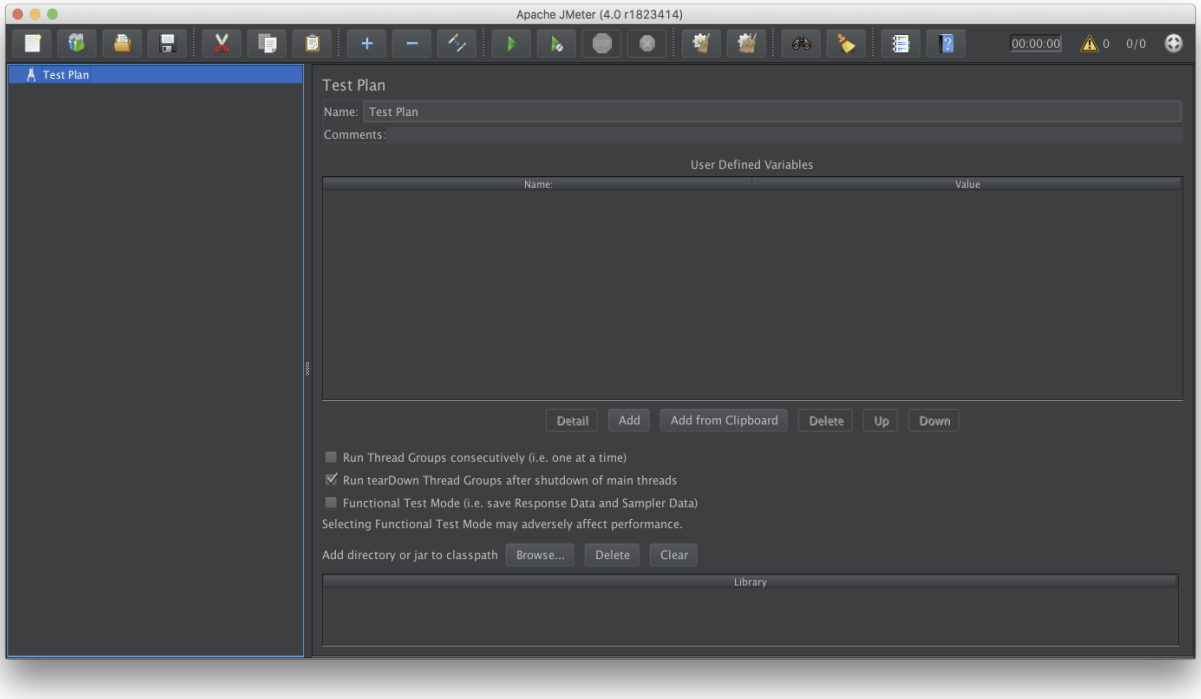
saber o que poderá acontecer com o sistema caso os limites de cargas descobertos sejam violados.

4.3 Ferramenta Jmeter

A ferramenta Jmeter[1] é uma aplicação *desktop* de código aberto, é escrita em java, e foi desenvolvida pela *Apache Foundation* para executar testes funcionais e medir o desempenho de aplicações em geral, sendo originalmente projetada para testar aplicações web. Sua utilização é indicada para realização de testes de performance, carga e stress.

É possível gravar todas as requisições que um usuário faria, simulando ações corriqueiras dentro da aplicação. Essas ações ficam gravadas no Jmeter, em uma estrutura conhecida como "grupo de teste". Após a gravação destas ações, o Jmeter possibilita disparar lotes simultâneos e numerosos destas ações, simulando um grupo de usuários. No final, a resposta do servidor para cada solicitação feita é coletada e, com base nessas respostas, as estatísticas são calculadas e as métricas de performance são geradas, obtendo-se a minimização dos drásticos efeitos da subestimação ou da superestimação dos tempos de resposta das aplicações, conforme ilustrado na figura 4.4.

Figura 4.4 – Interface inicial do Jmeter



Fonte: Merixstudio (2021)

[1] A utilização de ferramentas para testes de carga e stress é essencial para que possamos testar a performance de nossas aplicações e mantê-las com qualidade mesmo com picos de tráfego.

A figura 4.4 exibe a tela inicial da Ferramenta Jmeter, a qual possui dois ambientes: o lado esquerdo exibe todos os parâmetros de configurações possíveis da ferramenta. O lado direito da tela é onde são realizadas os ajustes e as alterações destes parâmetros pelo usuário. No canto superior desta tela temos um cronometro, onde o usuário define o tempo de duração do teste a ser realizado. Neste pesquisa não foram realizados testes com um tempo cronometrado.

4.3.1 Configuração da ferramenta Jmeter para a realização dos testes

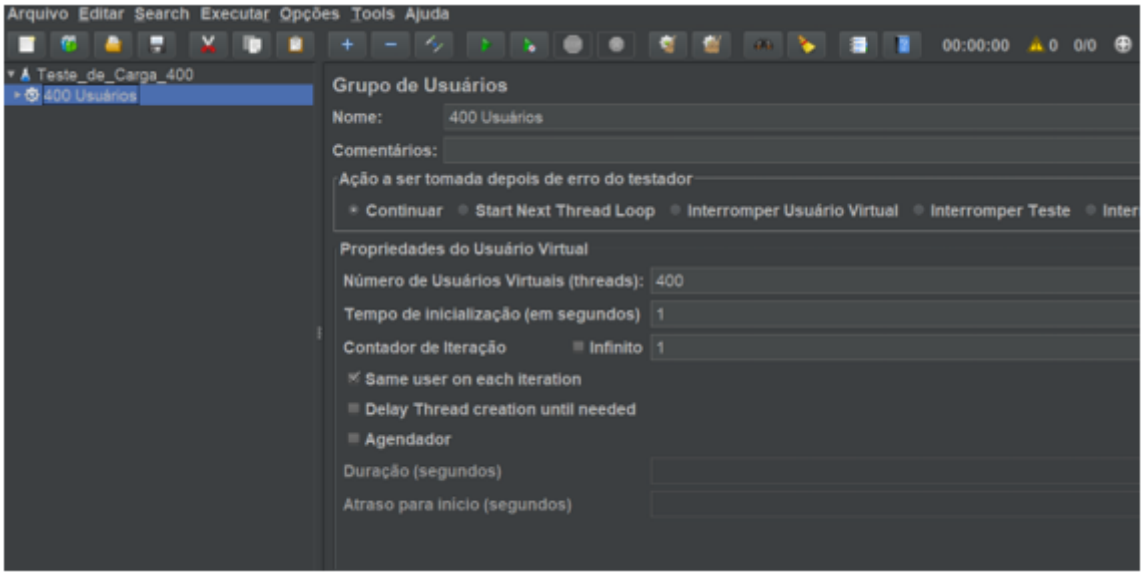
O teste que pretende-se realizar baseia-se na criação de usuários virtuais e sua utilização para simular o acesso de pessoas reais, onde são efetuadas requisições HTTP na funcionalidade de reserva de espaços do sistema de agendamento do campus de Castanhal, com o objetivo de se descobrir a quantidade de carga suportada e seu desempenho em determinados pontos, porém para isso é necessário configurar a ferramenta.

Para a realização de testes são disponibilizados diversos tipos de requisições, além de controladores lógicos como *loops* e controles condicionais, para serem utilizados na construção de planos de teste. O Jmeter disponibiliza um controle de *threads*, no qual é possível configurar o número de *threads*, a quantidade de vezes que cada *thread* será executada e o intervalo entre cada execução. E, por fim, existem diversos *listeners* que, inspirando-se nos resultados das requisições, podem ser usados para gerar gráficos, tabelas e outros que exibem com detalhes o que ocorreu durante o processo de teste.

Para se criar um teste nesta ferramenta é necessário primeiramente que o sistema já esteja na rede, ou seja, que o sistema em questão seja encontrado nos motores de busca convencionais como o Google. Abaixo foi realizada a configuração de quatrocentos usuários simultâneos apenas para fins didáticos com o objetivo de se ilustrar melhor o processo de configuração dessa ferramenta.

Após definir quais funcionalidades serão testadas na ferramenta, devemos dividi-las em casos de testes, que poderão ser organizados por grupos de usuários (nomeados neste exemplo como “400 Usuarios”), então teremos o plano de testes (nomeado neste exemplo para “Teste_de_carga_400”) para agrupar os casos de testes, que podem ser executados simultaneamente ou separadamente, conforme é exemplificado na figura 4.5.

Figura 4.5 – Configuração do número de usuários simultâneos no Jmeter



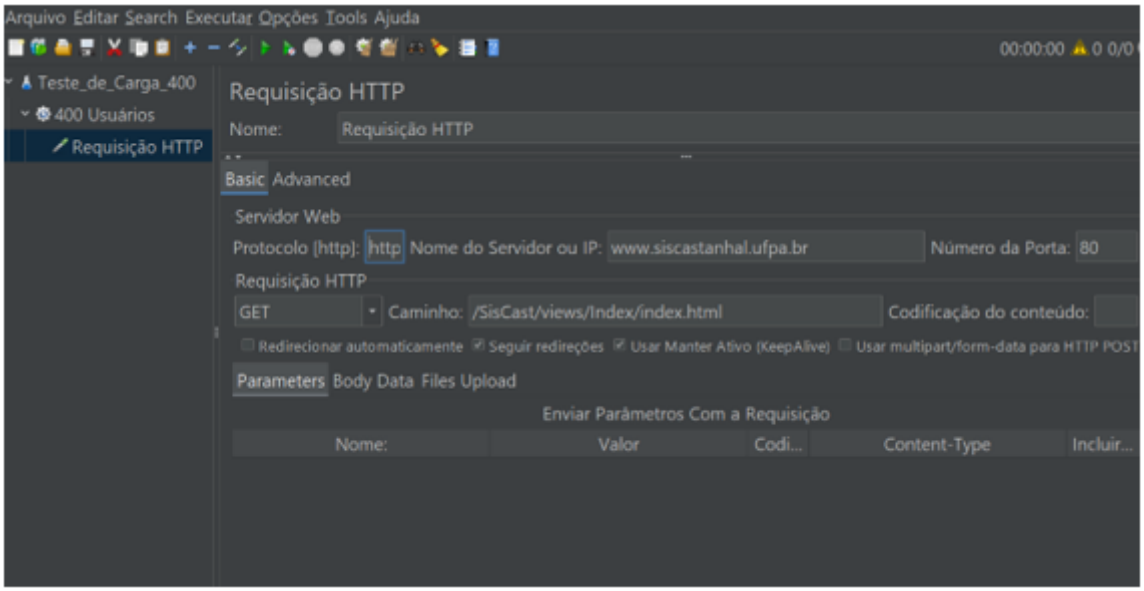
Fonte: Própria (2020)

Cada um dos mesmos irão ser configurados de forma idêntica na opção “Número de Usuários Virtuais (threads)”, diferindo-se apenas na carga de usuários simultâneos por caso de teste: primeiro com cem, depois cento e cinquenta, duzentos até o valor de quatrocentos e cinquenta que corresponde a quantidade média de usuários do campus; assim como é exibido a configuração do exemplo de quatrocentos usuários na figura 4.5.

Basta que a pessoa encarregada de realizar o teste, configure as requisições que a carga de usuários simultâneos fará, temos diversas opções para este teste, foi escolhida a requisição HTTP por ser a mais usada na web e que melhor representa o contexto de realização do teste.

O testador deve copiar a URL da página do sistema em questão para poder colar o domínio na opção (Nome do servidor ou IP), por o *path* (Caminho, onde na página se encontra a funcionalidade pretendida a ser testada), escolher o método de nossa preferência (GET ou POST) na opção (Método) que no caso tem haver com a requisição escolhida anteriormente. Por fim devemos inserir o protocolo referente a requisição usada na opção (Protocolo) que no nosso teste é o protocolo HTTP (Hypertext Transfer Protocol) e a porta referente ao protocolo HTTP (Porta 80) na opção (Número da porta), como é mostrado na figura 4.6.

Figura 4.6 – Configuração da requisição que os usuários simultâneos farão no Jmeter

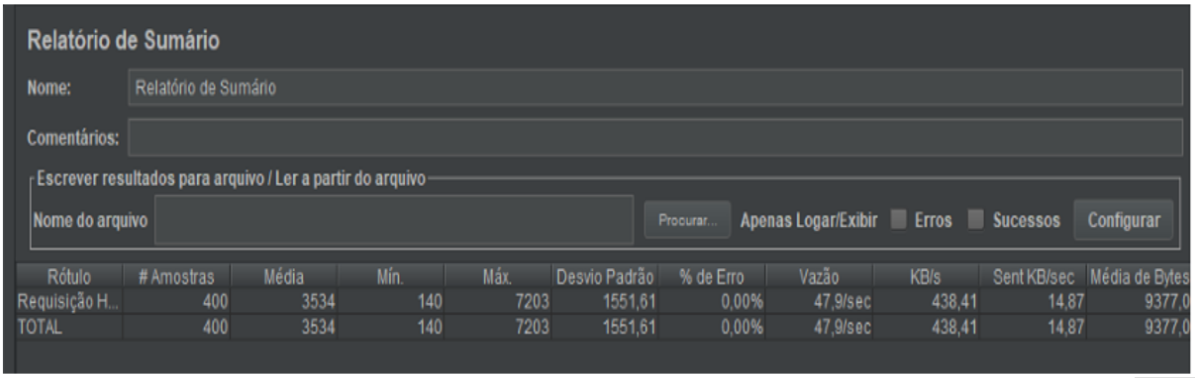


Fonte: Própria (2020)

Depois dessas configurações todas podemos gerar os *listeners*, que são a forma com que os resultados gerados pela ferramenta serão exibidos, para podermos fazer as aferições cabíveis ao contexto do teste realizado.

Conforme o procedimento iniciado anteriormente na configuração da ferramenta para a execução do exemplo com quatrocentos usuários simultâneos, podemos obter uma forma de exibir o resultado com o *listener* chamado "Relatório de sumário". Trata-se de uma tabela, onde nela é possível verificar de forma mais didática as métricas e os resultados estatísticos obtidos pela ferramenta no desenrolar do teste. Com ele é possível vislumbrar com mais exatidão as principais métricas e valores gerados, embora o *listener* relatório de sumário seja bastante elucidativo, nem todas as métricas e valores contidos nos seus resultados produzidos terão impacto significativo para o processo de aferição do teste. O que nos permite escolher algumas métricas, dentre as várias produzidas, para compor nossa análise do teste, a figura 4.7 abaixo nos mostra a exibição do resultado obtido para 400 usuários simultâneos usando o *listener* relatório de sumário.

Figura 4.7 – Relatório de sumário para 400 usuários simultâneos



Fonte: Própria (2020)

A figura 4.7 é o relatório de sumário, nela é apresentado diversos resultados obtidos nos testes em várias métricas, sendo que o destaque vai para: vazão (47,9 / sec), porcentagem de erro (0.00%) e média de uma requisição (3534). É valido destacar que o teste feito neste capítulo, foi realizado na funcionalidade de login do sistema de agendamento do campus, e serve apenas para ilustrar como serão realizados os outros testes na funcionalidade de reserva de espaços do sistema de agendamento do campus, que são os que realmente importam.

4.3.2 Principais métricas consideradas para a realização da análise dos resultados obtidos

Como mencionado anteriormente a ferramenta possui diversas métricas que por sua vez contém vários valores que colaboram para uma análise mais assertiva acerca do processo de testagem, porém torna-se exaustivo e até certo ponto ineficaz analisar tudo que foi produzido pela ferramenta.

Dentre as métricas obtidas, há algumas que se sobressaem e possuem mais peso na análise,utilizando o relatório de sumário, que versa-se de uma tabela onde são expostas algumas métricas de *quality of service* (QOS) ou qualidade de serviço, onde é possível analisar mais detalhadamente o que ocorreu durante os testes de desempenho com os usuários simultâneos realizando as requisições programadas.

Algumas das métricas de *quality of service* utilizadas incidem diretamente sobre a análise de desempenho do sistema, e é extremamente válido abordar sobre o que elas representam no contexto do teste e em que ponto influem sobre o sistema em questão. Outras, porém, nos dão uma ideia geral de alguns acontecimentos pontuais mas que não incidem diretamente no objetivo do trabalho. Abaixo se encontram algumas das métricas utilizadas pela ferramenta que tem influência direta sobre o desempenho do sistema e nos

norteiam com mais exatidão no processo de interpretação do teste. Tais métricas foram consideradas para os testes realizados neste trabalho:

- Vazão: Pode ser definida como um valor que é transmitido em um meio por uma medida de tempo, ex: número de bits que podem ser transmitidos sobre a rede em um dado tempo, sendo expressa neste exemplo em bits/segundo (b/s). Muitas vezes usa-se o termo taxa de transmissão para se referir a vazão. Iremos observar que ao atingir o valor de 250 requisições por segundos, a vazão tende a estabilizar-se em uma porcentagem de 6bits/s, a partir deste valor o sistema começa a entrar em congestionamento no atendimento das requisições dos usuários para agendamento de espaços no sistema, mas a maioria das requisições ainda é processada.

- Porcentagem de erro: Mostra o quanto de erro, requisições não processadas, em relação a quantidade de carga de requisições enviadas foram geradas durante todo o processo. Como dito anteriormente, o inicio da ocorrência de erros se dá no valor de carga de 250 usuários simultâneos, que respectivamente possui uma vazão próxima de 6 bits/segundo, porém durante as simulações o valor máximo suportado pelo sistema foi de 450 usuários simultâneos, onde a vazão é próxima de 11 bits/segundo, sendo que nesta quantidade 50 porcento das requisições já não conseguem ser processadas e o sistema tende a entrar em colapso.

- Média de uma requisição: Exibe quanto tempo decorreu do início da requisição até o seu processamento. Observa-se, neste trabalho que quando esta métrica está elevada, podemos nos atentar com relação a latência no processamento, devido a produção de engarrafamento no tráfego de requisições.

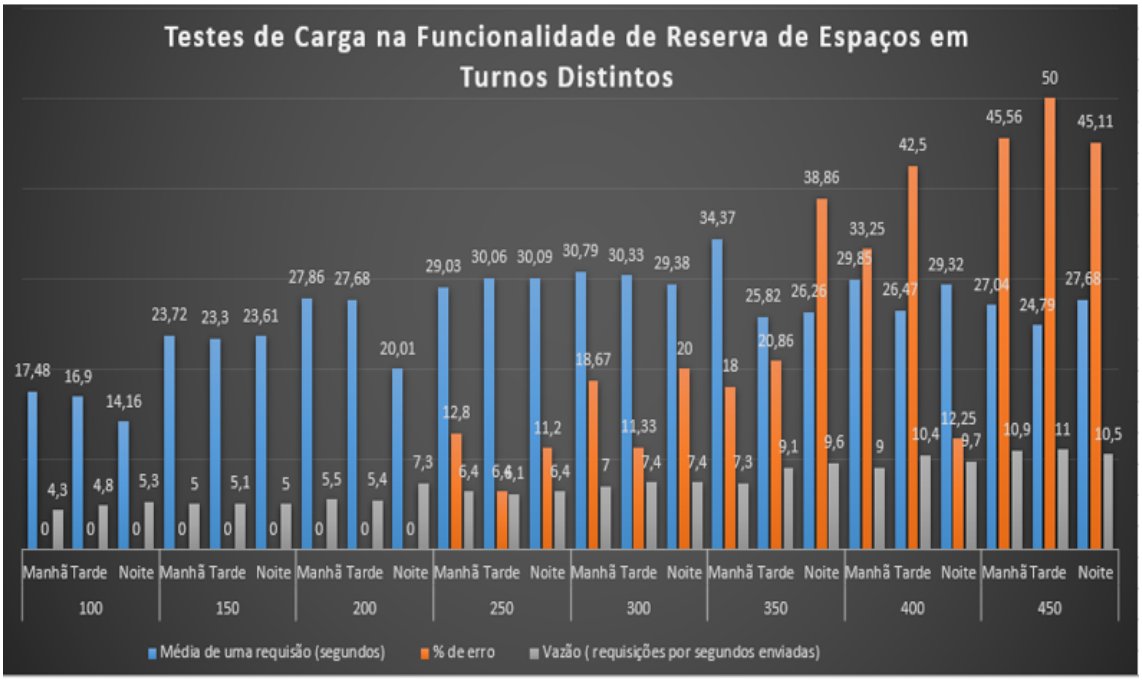
5 Análise de resultados obtidos

Com a configuração da ferramenta realizada, um exemplo prático mostrado e a explicação de quais das várias métricas presentes na ferramenta serão consideradas para a análise; podemos nos voltar agora à realização dos testes de desempenho na funcionalidade de reserva de espaços e a análise dos seus respectivos resultados.

O teste funcionará da seguinte forma: os usuários virtuais, gerados pela ferramenta, irão simular o acesso de pessoas reais, onde irão efetuar requisições HTTP na funcionalidade de reserva de espaços com o objetivo de se descobrir a quantidade de carga suportada e seu desempenho com contínuos acréscimos de carga.

Lembrando que operabilidade da funcionalidade em si não será testada, se realmente funciona ou não, o que irá ser testado é a performance ou desempenho do sistema em relação a uma funcionalidade para um fluxo de carga especificado e construído por meio da aplicação automatizada chamada: Jmeter. A funcionalidade de reserva já se encontra operando normalmente, como o esperado, só o que não se sabe é a sua performance quando exigida por um volume maior de requisições simultâneas. Posto isso e refazendo toda a configuração contida na seção 4, para a criação dos testes, e levando em consideração apenas as métricas escolhidas e também citadas na seção 4, foram realizados os testes no sistema. Com a carga inicial de 100 usuários simultâneos, e adicionando 50 em 50 usuários simultâneos até se atingir o valor de 450 usuários simultâneos. Abaixo podemos ver a figura 5.1 que exibe o gráfico e os resultados obtidos totalizados.

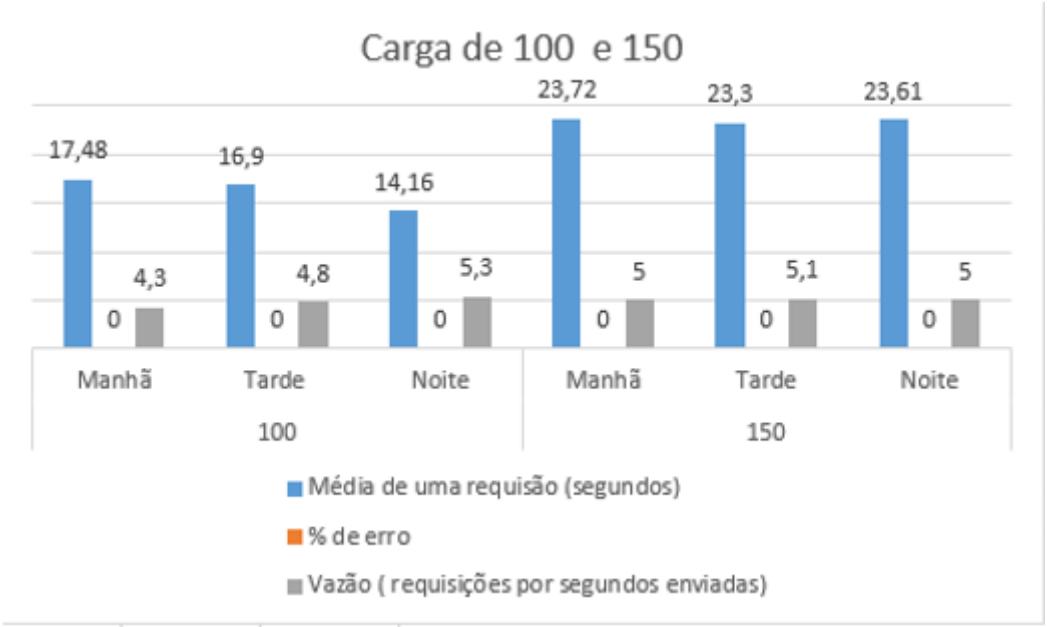
Figura 5.1 – Gráfico exibindo a totalidade dos resultados obtidos



Fonte: Própria (2020)

5.1 Resultados da realização dos testes de desempenho com 100 e 150 usuários simultâneos

Figura 5.2 – Valores encontrados para as cargas de 100 e 150 usuários simultâneos em turnos distintos



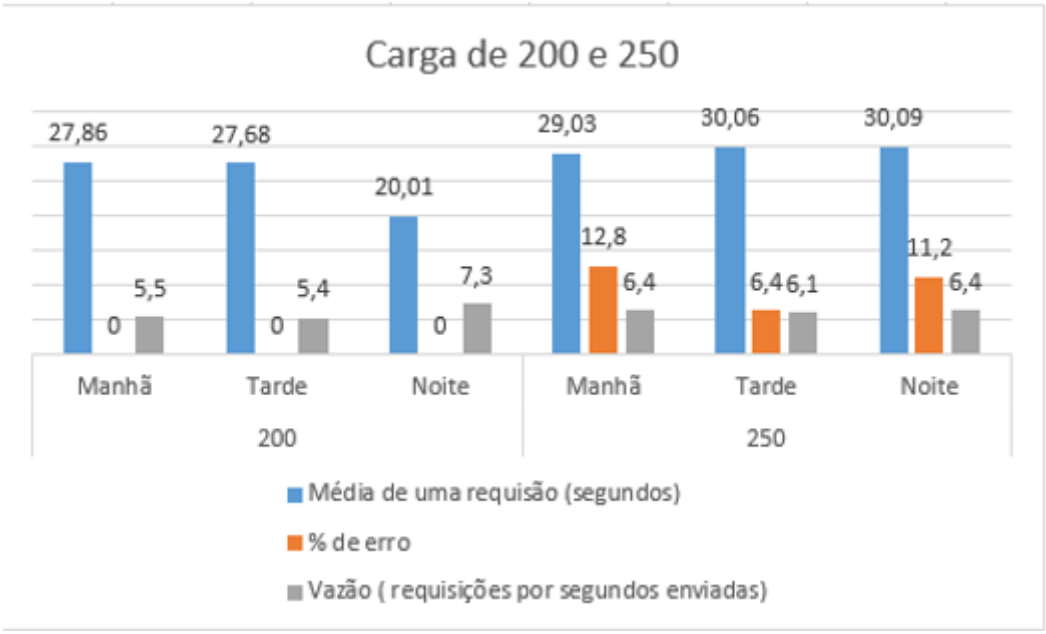
Fonte: Própria (2020)

Para se entender os resultados de forma mais didática é necessário subdividir o gráfico 5.1 em outros menores para assim ser possível comentar o que se decorreu. Iniciou-se o teste de desempenho com a carga de 100, como é exibido na figura 5.2, e posteriormente com carga de 150 usuários realizando os acessos simultâneos via requisições HTTP e com isso verificou-se a inexistência de quaisquer porcentagens de erros, devido, há uma baixa demanda de requisições HTTP em turnos de mesma quantidade de usuários.

É válido ressaltar, que nas duas simulações o tempo médio que uma requisição leva para ser realizada não chega a ser altamente discrepante entre as cargas. É claro que com o acréscimo de mais 50 usuários simultâneos e os mesmos realizando as requisições HTTP no mesmo tempo, esse tempo médio de uma requisição para a simulação com a carga de 150 usuários simultâneos tende a aumentar como exemplificado no gráfico 4.7, assim como ambas as taxas de vazão (requisições por segundo enviadas) que variam para mais e para menos em alguns momentos para as duas cargas (100 e 150), mas tendem a se estabilizar e se manter constantes e no valor 5 bits/segundo.

5.2 Resultados da realização dos testes de desempenho com 200 e 250 usuários simultâneos

Figura 5.3 – Valores encontrados para as cargas de 200 e 250 usuários simultâneos em turnos distintos



Fonte: Própria (2020)

Com a intensificação do teste de desempenho e agora com simulações que contam com cargas de 200 e 250 usuários realizando os acessos simultâneos via requisições HTTP,

percebemos os primeiros sinais de erro, como é exibido na figura 5.3, sobretudo quando imputamos a carga de 200 que é quando notamos que o tempo médio de uma requisição já se encontra elevado, embora o software fique lento ele processa as requisições HTTP que lhe são redirecionadas.

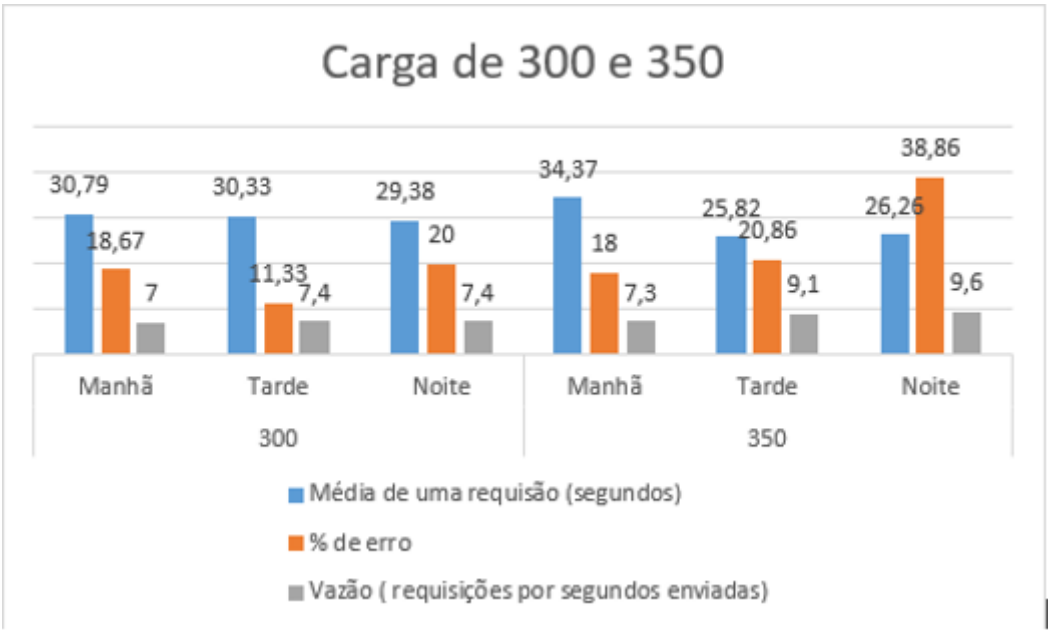
O panorama muda com a carga de 250, onde verifica-se que o tempo médio de uma requisição está bem mais elevado se comparado com a carga de 200, e ainda mais se considerado com as simulações anteriores (100 e 150). É nessa quantia de usuários, realizando os acessos simultâneos através de requisições HTTP, que começam a surgir os primeiros erros.

É importante ressaltar, que mesmo com porcentagens de erros presentes, o sistema ainda consegue manter sua operacionalidade, processando grande partes das requisições HTTP providas pelos usuários simultâneos. Analisando a vazão da carga de 250, nota-se que com o aumento da carga, a vazão também aumenta e gira em torno de 6 bits/segundo, o que nos instiga a idealizar uma correlação diretamente proporcional entre o aumento da vazão e a incidência de erros.

Se o tráfego simultâneo aumenta na rede, devido às requisições, logicamente o desempenho do sistema tende a ser mais cobrado com o intuito de manter operante o serviço, com isso o surgimento de erros na rede é bem mais provável.

5.3 Resultados da realização dos testes de desempenho com 300 e 350 usuários simultâneos

Figura 5.4 – Valores encontrados para as cargas de 300 e 350 usuários simultâneos em turnos distintos



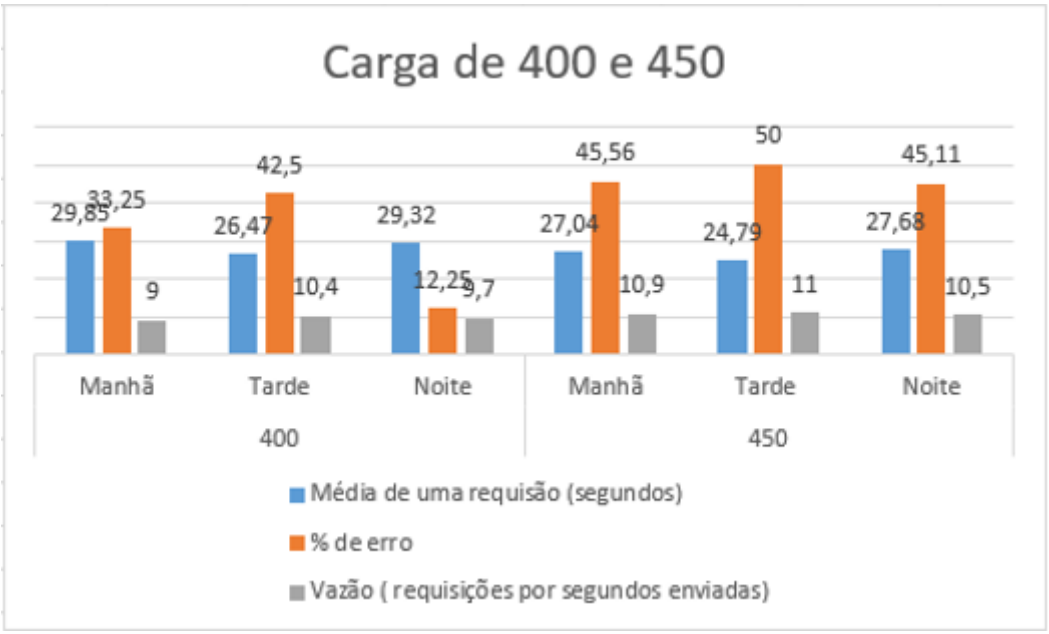
Fonte: Própria (2020)

Com valores de carga de 300 e 350 usuários realizando os acessos simultâneos via requisições HTTP, notamos que a vazão chega a 7 e consegue alcançar o patamar de 9, como é exemplificado na figura 5.4.

A consequência dessa chegada ao patamar de 9 há um aumento expressivo na porcentagem de erro, devido a correlação diretamente proporcional entre o aumento da vazão e a incidência de erros que já fora citada anteriormente. O sistema simplesmente perde a capacidade de controle fluxo e seu rendimento é severamente comprometido aparecendo os primeiros gargalos advindos do acúmulo de requisições pelo caminho, que não conseguem prosseguir devido ao grande congestionamento gerado pelas requisições não processadas.

5.4 Resultados da realização dos testes de desempenho com 400 e 450 usuários simultâneos

Figura 5.5 – Valores encontrados para as cargas de 400 e 450 usuários simultâneos em turnos distintos



Fonte: Própria (2020)

Com valores de carga de 400 e 450 usuários realizando os acessos simultâneos via requisições HTTP, ambas as simulações apontam para o colapso total do sistema, os erros disparam em virtude do crescimento da vazão, como é exibido na figura 5.5.

É neste ponto que temos o limite de stress suportado pela aplicação, ou seja, com 250 acessos simultâneos temos o inicio da ocorrência dos erros, o que implica em algum leve atraso relativo ao tempo médio de uma requisição, mas com as requisições sendo atendidas. Com o valor de carga 450 acessos simultâneos temos o máximo de requisições suportadas.

É visível que com porcentagens de erros tão elevadas, como as apresentadas, os gargalos formados no trafego do servidor da aplicação simplesmente impedem as requisições, tornando-as impulsiveis de serem realizadas. As poucas que ainda conseguem superar o trafego engarrafado, tem um tempo médio extremamente elevado, o que em uma situação real geraria inúmeros transtornos aos usuários.

6 Conclusão

Neste trabalho foi apresentado um problema real enfrentado pelo campus da UFPA de Castanhal, sua proposta de solução perpassa pela construção de um sistema que organiza melhor o agendamento dos espaços do campus.

É de suma importância, que se elabore testes que contribuam com o correto funcionamento do sistema em questão, para isso recorremos ao estudo de tópicos de engenharia de software que colaboram para a fundamentação da solução.

Como o software está em desenvolvimento e poucos testes foram realizados até o momento, utilizou-se o universo que compreende os testes de caixa preta, e para dimensionar os resultados no trabalho, optou-se por priorizar o teste de desempenho. Para isso foi apresentada e utilizada a ferramenta *open source* Jmeter, e a partir de seus resultados concretos vimos os limites de carga suportados pela aplicação, em desenvolvimento, e sobretudo o ponto inicial de stress que gera os erros e mal funcionamentos da aplicação. Viabilizando assim as ações cabíveis aos utilizadores do sistema no trato com o programa.

Notamos que os objetivos gerais: adição de novos testes específicos e contribuição para o ganho de qualidade; juntamente com os objetivos específicos: criar testes relacionados ao desempenho da aplicação, inspecionar e diagnosticar a funcionalidade de reserva de espaços do sistema de agendamento do campus, geração de documentação para monitoramento e manutenção, colaborar para a solução do problema de reserva de espaços e apontar problemas presentes no sistema por meio dos testes automatizados; abordados no início foram satisfeitos no decorrer do trabalho.

Vale destacar que os conteúdos abordados neste trabalho são vistos em geral em especializações e em disciplinas de programas de pós graduação, dessa forma, a construção deste trabalho exigiu um estudo além das disciplinas ministradas para um curso de graduação. Claro que para o sistema ser posto em pleno funcionamento são necessários outros testes além do teste de desempenho realizado. Portanto, espera-se que este trabalho tenha contribuído para a resolução do problema e sirva de material de estudo e documentação sólida para o sistema.

Levando em consideração que o pico de maior demanda de requisições para agendamento de espaços no campus castanhal se dá durante a realização de eventos, e que nestas ocasiões possuímos cerca de 10 a 20 requisições de reserva de algum espaço do campus, e como o valor inicial de carga é de 100 usuários simultâneos, e o sistema funciona normalmente. Podemos dizer que o sistema no geral se porta bem, visto que, mesmo quando exigido a valores muito superiores do que diariamente é ou em ocasiões especiais, como eventos, o sistema ainda funciona bem e tende a apresentar as primeiras falhas com

valores de carga extremamente elevados, de 250 requisições simultâneas em diante.

6.1 Trabalhos futuros

Como foi mencionado, no decorrer do trabalho, apenas o que se propôs a testar foi o desempenho de uma funcionalidade de reserva de espaços presente em um sistema de agendamento do campus de Castanhal, que se encontra em desenvolvimento, para um fluxo de carga especificado. As informações que foram utilizadas para se realizar este trabalho, se encontram principalmente na área de engenharia de software, tal área é extremamente abundante, ou seja, vai além do que fora mostrado no decorrer trabalho e para se utilizar o sistema plenamente, é necessário abrir margem para outros trabalhos futuros como:

- desenvolver módulos com recursos de acessibilidade web para o sistema de agendamento de espaços do campus da UFPA de Castanhal.
- elaboração e realização de testes de segurança no sistema agendamento de espaços do campus da UFPA de Castanhal.

Referências

BETRYBE. 2020. <https://blog.betrybe.com/desenvolvimento-web/aplicacoes-web/>. Acessado em 04/07/2021. Citado na página 24.

BRAGA, F. B. et al. Testes de software: Importância, diferentes tipos e técnicas de testes. In: *FACET-Sistemas de Informação*. [S.l.: s.n.], 2013. Citado 2 vezes nas páginas 25 e 27.

CRUZ, F. *Scrum e Agile em Projetos (2a. edição): guia completo*. [S.l.]: Brasport, 2018. Citado na página 23.

DFILITTO. 2019. <https://dfilitto.com.br/noticias/teste-de-software-norma-iso-9126/>. Acessado em 14/04/2020. Citado 2 vezes nas páginas 21 e 22.

DIEGOMACEDO. 2018. <https://www.diegomacedo.com.br/entendendo-as-aplicacoes-web/>. Acessado em 03/10/2020. Citado na página 24.

FARIAS, E. M. B. et al. Avaliação de desempenho de um website utilizando apache jmeter: Um estudo de caso do website institucional da universidade federal do oeste do pará. 2016. Citado 4 vezes nas páginas 11, 18, 19 e 20.

FERRARI, L. Ferramentas de teste para aplicações web: Um estudo de caso. 2008. Citado 4 vezes nas páginas 11, 18, 19 e 20.

FUKUMORI, A. T.; SANTOS, I. G. d.; MORRO, T. M. A importância da atividade de teste no desenvolvimento de software. 2008. Citado na página 20.

GONÇALVES, T. G. *Componentes de Processo para Análise de Desempenho de Processos de Software*. Tese (Doutorado) — MSc Dissertation, Universidade Federal do Rio de Janeiro, 2014. Citado na página 20.

KOSCIANSKI, A.; SOARES, M. dos S. *Qualidade de Software-2ª Edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. [S.l.]: Novatec Editora, 2007. Citado na página 21.

PFLEEGER, S. L. *Engenharia de software: teoria e prática*. [S.l.]: Prentice Hall, 2004. Citado 2 vezes nas páginas 21 e 34.

PONTES, M. B. Introdução a testes de software. *Engenharia de Software Magazine, ano*, v. 1, 2009. Citado 2 vezes nas páginas 17 e 22.

PRESSMAN, R. *Engenharia de software: uma abordagem profissional. 6th*. [S.l.]: Porto Alegre: Bookman, 2009. Citado 2 vezes nas páginas 23 e 28.

PRIMECONTROL. 2019. <https://www.primecontrol.com.br/o-que-sao-cenarios-scripts-e-casos-de-teste/>. Acessado em 11/07/2021. Citado na página 30.

PRODANOV, C. C.; FREITAS, E. C. de. *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. [S.l.]: Editora Feevale, 2013. Citado na página 29.

TIESPECIALISTAS. 2015. <https://www.tiespecialistas.com.br/teste-de-desempenho-de-software/>. Acessado em 04/07/2021. Citado na página 28.