ESTUDO DE CASO SOBRE TESTES FUNCIONAIS EM UM SISTEMA DE RESERVA DE SALAS

Lívia Beatriz Maia de Lima (UFERSA) E-mail: liviabeatrizmaia7@gmail.com
Geísa Morais Gabriel (UFERSA) E-mail: geisamorais8@gmail.com
Alysson Filgueira Milanez (UFERSA) E-mail: alysson.milanez@ufersa.edu.br
David Candeia Medeiros Maia (IFPB) E-mail: david.maia@ifpb.edu.br

Resumo: Este estudo consiste na realização de testes de software no Sistema de Reserva de Salas (SRS), para a identificação de defeitos no software. Considerando que o sistema planeja melhorar o processo de agendamento de espaços, realizou-se uma pesquisa de cunho quali-quantitativo com a execução de testes de sistema. Obteve-se como resultados a identificação de falhas e erros, destacando inconsistências no quesito de usabilidade e responsividade para dispositivos móveis. A partir das ferramentas Selenium e JUnit, foi possível identificar erros significativos na conexão com o banco de dados; com o JMeter e o Lighthouse, constatou-se lentidão nas respostas com alto volume de carga, além de visualizar os diagnósticos de desempenho do sistema.

Palavras-chave: Automatização de testes, casos de teste, qualidade de software, teste de software.

CASE STUDY ON FUNCTIONAL TESTING IN A ROOM RESERVATION SYSTEM

Abstract: This study consists of conducting software tests on the Room Reservation System (SRS) to identify software defects. Considering that the system aims to improve the space scheduling process, a qualitative-quantitative research was carried out with the execution of system tests. The results obtained identified flaws and errors, highlighting inconsistencies in usability and responsiveness for mobile devices. Using Selenium and JUnit tools, it was possible to identify significant errors in the database connection; with JMeter and Lighthouse, slow responses under high load volume were observed, in addition to visualizing system performance diagnostics.

Keywords: Test automation, test cases, software quality, software testing.

1. Introdução

O Sistema de Reserva de Salas (SRS) é um sistema Web que visa dinamizar e agilizar o processo de realização de reservas dos espaços na universidade, diante de uma interface gráfica. Com o avanço tecnológico, a era digital emergiu no cotidiano das pessoas expressivamente, bem como as aplicações Web. As aplicações Web são definidas como um artefato visual que expõe informações para o usuário interativamente (DELAMARO; MALDONADO; JINO, 2016). Dessa forma, para evitar conflitos e inconsistências em suas ações, a validação das funcionalidades do sistema se torna importante.

Posto isso, a qualidade de software tem ganhado cada vez mais espaço no desenvolvimento dos softwares, sendo um diferencial notório e premissa básica das aplicações Web (PRESSMAN; MAXIM, 2021). A forma mais comum de controle de qualidade é o teste de software, definido como uma operação técnica que identifica uma ou mais características de defeito no software (IEEE, 1990). Dessa forma, o uso de aplicações Web por instituições acadêmicas para resolução de problemas deve ser acompanhado de testes para assegurar a qualidade e o funcionamento correto com respeito aos requisitos e restrições de negócio (PRESSMAN; MAXIM, 2021).

Nessa perspectiva, a experiência do usuário (do inglês, *User Experience* - UX) atua como ponto-chave para um software bem-sucedido, uma vez que corresponde aos sentimentos de satisfação associados às ações do usuário. Logo, com a experiência positiva, o usuário fica mais propício a utilizar novamente o sistema (KRUG, 2018). Dessa forma, surge a ideia da

realização de testes funcionais para identificar e garantir que as funcionalidades atendam aos requisitos especificados diante do comportamento externo do sistema e, a partir disso, sugerir melhorias para a resolução das possíveis inconsistências.

Análogo ao exposto, o Sistema de Reserva de Salas carece da incorporação de testes de software como elementos cruciais em seu desenvolvimento. Essa lacuna impacta na compreensão e funcionamento do sistema, bem como na produtividade e manutenção do software (SOMMERVILLE, 2018), uma vez que o sistema tem importância tanto para docentes quanto para discentes na organização das atividades acadêmicas. A partir disso, a análise se estenderá para o âmbito dos aspectos comportamentais, mediante a realização de testes funcionais e não funcionais. Para tal, essa abordagem é adotada de modo a avaliar as funcionalidades do sistema do ponto de vista do usuário, buscando identificar eventuais erros e falhas que afetam a experiência de uso (PRESSMAN; MAXIM, 2021).

Sob aspectos gerais, este estudo destaca a importância da aplicação de testes de software para a garantia da qualidade do produto, demonstrando que a ausência dessas práticas durante o desenvolvimento do sistema teve impactos significativos. Ao longo do estudo, foi detalhado todo o processo de teste, bem como a escolha de ferramentas e procedimentos, visando disseminar o conhecimento adquirido.

2. Fundamentação Teórica

Nesta seção, são apresentados os conceitos necessários para o entendimento do presente trabalho. Dessa maneira, na seção 2.1 é apresentada a definição de sistemas Web; então, na seção 2.2 conceitua-se qualidade de software; já na seção 2.3 são apresentados tópicos referentes ao teste de software; posteriormente, na seção 2.4 é definida a experiência do usuário e os conceitos relacionados às métricas de usabilidade; por fim, a seção 2.5 informa as ferramentas e tecnologias utilizadas nos testes.

2.1. Sistemas Web

Um sistema Web corresponde a uma aplicação composta por uma base de dados e elementos visuais hospedados na Internet (DELAMARO; MALDONADO; JINO, 2016). Sabendo disso, os sistemas são acessados por meio de um navegador de rede, que possibilita as interações com usuários (PRESSMAN; MAXIM, 2021) utilizando a arquitetura cliente/servidor.

Por sua vez, a arquitetura cliente/servidor define o cliente como responsável pelas requisições feitas ao servidor por meio de interações e o servidor é encarregado por responder às solicitações diante das requisições (KUROSE; ROSS, 2021). Logo, em aplicações Web, a comunicação cliente/servidor é estabelecida entre os clientes, representados pelos navegadores Web, e o servidor, sendo este os programas que hospedam as páginas e sites.

Assim, dada a crescente evolução tecnológica, os sistemas Web necessitam se manter em constante evolução conforme a necessidade do cliente. Para isso, são necessárias a concordância nas regras de qualidade de software para atender às demandas de uso.

2.2. Qualidade de Software

O conceito de qualidade de software não possui uma definição precisa; para isso, neste estudo, determina-se qualidade como o grau no qual o sistema atende aos requisitos especificados (VAZQUEZ; SIMÕES, 2016). Por conseguinte, a qualidade de software é fundamental para a sustentabilidade do sistema Web e sua avaliação diz respeito ao valor mensurável recebido e transferido pelo software em uso (GONÇALVEZ *et al.*, 2019).

A qualidade ligada aos sistemas Web está direcionada à parte de Interação Humano-Computador (IHC) a qual se preocupa com os elementos visuais, funcionalidades

atendidas e clareza nas necessidades dos usuários para a completude das tarefas (BARBOSA; SILVA, 2010). A obtenção de qualidade do sistema está ligada à aplicação de técnicas e atividades operacionais, sendo uma das atividades mais importantes o teste de software (SOMMERVILLE, 2018). Diante disso, a utilização de testes em seus artefatos previne e garante a conformidade na verificação e validação do sistema, bem como a confiabilidade para com os usuários (VALENTE, 2020).

2.3. Teste de Software

A definição de teste de software corresponde ao conjunto de técnicas que validam e verificam o funcionamento adequado do sistema (PRESSMAN; MAXIM, 2021). Com os testes é possível identificar erros, bem como assegurar que os requisitos de software estão conforme especificados pelo cliente (SOMMERVILLE, 2018). No entanto, as atividades de teste requerem tempo e organização, o que, em muitos casos, conduzem a um negligenciamento destas atividades. No processo de testes, além do conhecimento exigido, deve-se ter planejamento, projeto, execução e acompanhamento, como também o registro documentado das observações (RIOS; MOREIRA, 2006).

Vale salientar que os testes não indicam a ausência dos erros, mas sim a detecção deles (VALENTE, 2020). Alinhado a isto, a realização de bons casos de teste é definida pela quantidade diversa de erros ainda não identificados no sistema. Portanto, o plano de teste deve ser elaborado de maneira a ser rígido, mas também flexível para se adequar às manutenções sucessivas do sistema sob teste (PRESSMAN; MAXIM, 2021).

O caso de teste é um artefato que define um conjunto de entradas e saídas para uma ação específica do sistema por meio de passos sequenciais (SOMMERVILLE, 2018). Nesse contexto, tem-se o oráculo, termo utilizado para o mecanismo que determina as saídas esperadas no sistema em qualquer ação (DELAMARO; MALDONADO; JINO, 2016); de forma simplificada, dispõe ao testador uma resposta indicando se o teste foi realizado com sucesso ou fracasso. Com base nisso, o processo de teste conduzido é sistemático, deve empregar técnicas, incluindo aspectos funcionais e estruturais (DELAMARO; MALDONADO; JINO, 2016).

Em teste de software existem duas principais vertentes: caixa branca e caixa preta (PRESSMAN; MAXIM, 2021; DELAMARO; MALDONADO; JINO, 2016). Essas técnicas apresentam visões diferentes do mesmo software que podem ser usadas em conjunto ou não, mas que se complementam. Nos testes de caixa branca ou estruturais, o testador tem acesso à estrutura interna do sistema. A avaliação ocorre por meio de unidades/componentes específicos do sistema, acarretando avaliações mais precisas (DELAMARO; MALDONADO; JINO, 2016). Já nos testes de caixa preta ou funcionais, o testador não se preocupa com os elementos do código, mas sim com o funcionamento do sistema, uma vez que não possui acesso à estrutura interna do programa. Dessa forma, busca garantir que os requisitos do produto sejam atendidos (DELAMARO; MALDONADO; JINO, 2016).

Nos testes de caixa preta, há os testes funcionais e não funcionais, sendo testes funcionais aqueles que visam descobrir inconsistências na validação das funcionalidades descritas no documento de requisitos (DELAMARO; MALDONADO; JINO, 2016). Em contrapartida, os testes não funcionais são aqueles que avaliam os comportamentos e limitações do sistema, mostrando “como” o software opera. A mesclagem das estratégias de caixa branca e preta, define o teste de caixa cinza (PRESSMAN; MAXIM, 2021), em que o testador tem conhecimento intermediário do sistema e realiza suas ações na parte lógica e funcional; tal estratégia híbrida, surgiu inicialmente para aprimorar os testes de sistemas Web, como o SRS.

Na execução dos testes de software, existe uma hierarquia que assume uma visão incremental,

que se inicia pelas unidades isoladas do programa, seguindo a etapa de integração e, por fim, culminando com a avaliação completa do sistema (HIRAMA, 2011). O teste de unidade é direcionado à menor parte do programa, sendo verificadas as funcionalidades individualmente (PRESSMAN; MAXIM, 2021). Em seguida, o teste de integração visa assegurar a interligação das funcionalidades, uma vez que é realizado à medida que o programa é criado (SOMMERVILLE, 2018). Por fim, os testes de sistema avaliam o software na verificação dos requisitos e funcionalidades no sistema completo, demonstrando o nível mais próximo do usuário final, na tentativa de identificar erros na interface (DELAMARO; MALDONADO; JINO, 2016).

Dada a complexidade dos projetos, é fundamental que o sistema seja testado minuciosamente. Assim, o método empregado para a realização dos testes de software pode ocorrer de forma manual ou automatizada (GONÇALVEZ *et al.,* 2019). A aplicação do método depende de cada objetivo e etapa do projeto, sendo escolhida a que apresenta o melhor nível de qualidade e obtenção de resultados. No processo de teste manual, o sistema é executado por um testador, que realiza as operações de teste com entradas e saídas de resultados, para comparar as respostas esperadas do sistema com as respostas reais (SOMMERVILLE, 2018). Já em testes automatizados, o testador usa softwares para controlar e executar os testes por meio de *scripts* de linguagem de programação e utilizando ferramentas de auxílio para desenvolver, gerenciar e/ou analisar os testes (SOMMERVILLE, 2018).

2.4. Experiência do Usuário

A experiência do usuário (do inglês, *User Experience* - *UX*) diz respeito a uma área fundamental no desenvolvimento de produtos digitais, pois se concentra na compreensão das percepções, emoções e interações dos usuários com o sistema (TEIXEIRA, 2014). No âmbito da experiência do usuário, a usabilidade desempenha um papel de destaque. O termo usabilidade condiz com a facilidade de utilização e clareza na estruturação das funcionalidades e conteúdos de interação do sistema proposto (TEIXEIRA, 2014). Portanto, sistemas que demandam um tempo maior em suas interações têm menor probabilidade de serem adotados pelos usuários (KRUG, 2018).

Com intuito de medir a usabilidade do sistema, são aplicados alguns métodos avaliativos para a descoberta de potenciais problemas, os quais se dividem em três principais vertentes: inspeção, observação e investigação (BARBOSA; SILVA, 2010). A vertente de inspeção é caracterizada por analisar os problemas na usabilidade do produto diante de regras e diretrizes de conhecimentos de especialistas, um exemplo é a avaliação heurística (ROGERS; SHARP; PREECE, 2013), destacando-se a heurística de Nielsen (MEDEIROS et al., 2020). Métodos que permitem a participação de pessoas estão interligados à segunda vertente (observação), sendo exemplos os testes de usabilidade (ROGERS; SHARP; PREECE, 2013). Por fim, em relação à terceira vertente (investigação) utilizam-se escalas de usabilidade que fornecem uma visão geral das avaliações (ROGERS; SHARP; PREECE, 2013).

2.5. Ferramentas e tecnologias

A crescente complexidade dos sistemas torna necessário o uso de ferramentas de apoio ao teste, para otimizar o trabalho. Dessa forma, as tecnologias escolhidas foram baseadas na utilização já existente do SRS e no domínio da ferramenta para atingir o objetivo do estudo.

A utilização do *framework* Selenium em aplicações Web permite validar as funcionalidades por meio da criação de testes funcionais (PERES, 2016), em que os testes rodam diretamente em navegadores Web. O Selenium IDE é um *plug-in* específico da série do Selenium, cujo propósito é ser uma solução simples e rápida sem ter contato direto com a linguagem de programação, lidando somente com os comandos na ferramenta. Em conjunto, o *framework*

open-source JUnit permite a criação da estrutura de testes automatizados na linguagem Java. A utilização do JUnit refere-se à criação e execução de casos de testes, dessa forma, possibilita verificar se os resultados são iguais aos esperados (cujo oráculo é criado manualmente), fornecendo uma resposta imediata.

JMeter é uma ferramenta de código aberto criada pela *Apache Software Foundation* cujo intuito é realizar testes de desempenho, carga e estresse em aplicações Web. A sua utilização permite analisar o desempenho da aplicação mediante simulações de usuários. Já o *Google Lighthouse* é uma ferramenta automatizada que visa medir a qualidade de uma página Web, sendo executada em extensão do Google ou por linha de comando. Além disso, a ferramenta gera um relatório com as dimensões e alertas que afetam a experiência do usuário, no quesito de desempenho, acessibilidade, responsividade e afins.

Para o versionamento e hospedagem dos testes, foram escolhidas as ferramentas Git e GitHub, respectivamente. O Git, ferramenta de controle de versão distribuída, permite que vários usuários tenham ramificações locais independentes e o GitHub permite a disponibilidade em somente um local para que diferentes usuários contribuam para o mesmo projeto, facilitando assim a comunicação (AQUILES; FERREIRA, 2014).

3. Método da pesquisa

Este estudo executa e analisa testes funcionais no SRS de uma universidade. Nesse sentido, foi realizada uma abordagem qualiquanti (COELHO, 2019), utilizando o método quantitativo na demonstração dos dados analisados dos testes; e no que diz respeito ao qualitativo, refere-se à criação de artefatos para o uso do software. Assim, caracteriza-se o estudo de cunho descritivo e exploratório, sob a natureza de pesquisa aplicada (COELHO, 2019).

A primeira etapa foi o levantamento bibliográfico e estudo direcionado ao arcabouço do sistema. Após a análise dos recursos do SRS, foi elaborado um relatório com o roteiro vinculado à estruturação e realização dos testes funcionais e não-funcionais.

O planejamento dos testes deste estudo teve caráter descritivo, em que foi estabelecido o roteiro e o gerenciamento dos testes. A criação dos casos e cenários de teste necessita ser o mais próximo possível da realidade do sistema, caso contrário, torna-se desperdício de tempo e de validação (GONÇALVEZ *et al.*, 2019). Nesse sentido, é realizado o seguinte processo para identificar e registrar os defeitos: cria-se um cenário, realiza-se um teste e valida-se a saída.

Após o planejamento, para a execução dos testes foi preciso configurar o ambiente para uso de ferramentas de apoio ao teste. As ferramentas utilizadas foram o Selenium, para automatizar os testes de navegador Web em conjunto com o *framework* JUnit para a verificação das saídas esperadas e geradas através dos *Asserts*. A combinação dessas ferramentas proporciona aos testes precisão quanto à cobertura e execução automatizada dos casos de testes. Em seguida, foram realizados testes funcionais e não funcionais no sistema com *JMeter* e *Lighthouse*.

4. Resultados

Nesta seção, apresentam-se os resultados obtidos no presente trabalho. Serão mostrados os resultados dos testes de sistema na seção 4.1, dos testes exploratórios na seção 4.2, dos testes de responsividade na seção 4.3, e dos testes de desempenho na seção 4.4.

4.1. Testes de Sistema

Antes dos testes funcionais, fez-se uma análise para identificar casos de testes, dados de testes e condições dos testes. Para isso, utilizou-se o particionamento em classes por equivalência,

juntamente com a análise de valor limite (PRESSMAN; MAXIM, 2021). Assim, foi possível identificar formatos e valores válidos e inválidos dos atributos, bem como a ausência de limites/fronteiras em seus atributos.

Ao realizar os testes do sistema nos requisitos funcionais, foram definidos o âmbito da configuração e as condições para os casos de teste. Inicialmente, foram realizados testes de caso de sucesso, em que todas as entradas e ações são executadas conforme esperado, levando aos resultados desejados. Em seguida, são conduzidos os testes de caso de fracasso, visando identificar situações em que o sistema não se comporta conforme o esperado, expondo suas falhas potenciais. Foram realizados 228 casos de testes nos requisitos funcionais do sistema, tendo sua cobertura de funcionalidade 100% atendida, conforme a documentação do sistema. A Tabela 1 mostra o quantitativo filtrado por requisitos de cada caso de teste realizado.

Tabela 1 - Quantitativo de Casos de Teste

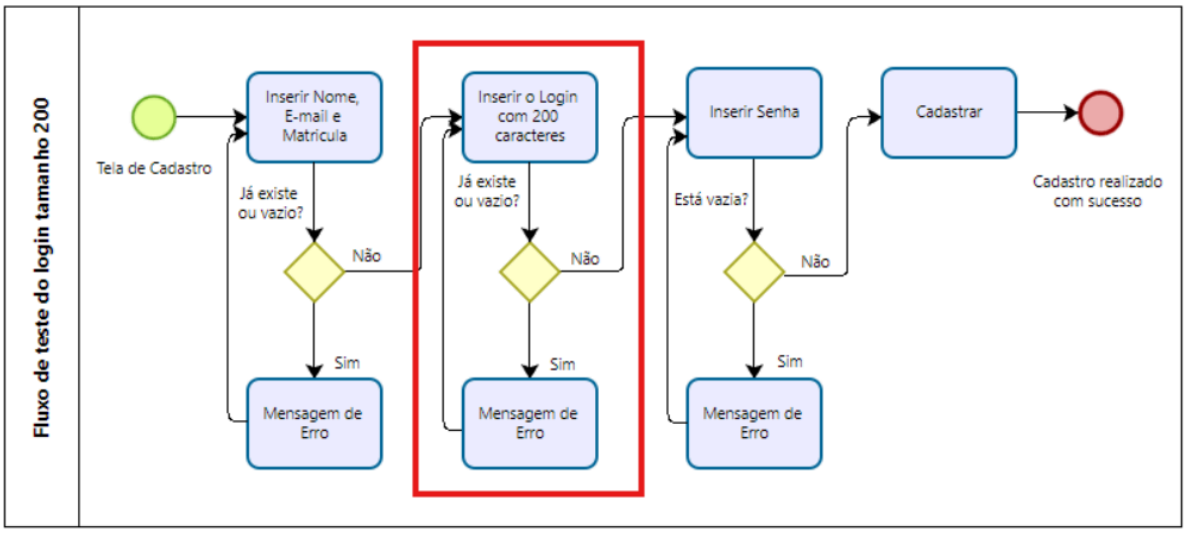
Requisito	Quantidade	Requisito	Quantidade
RF001 - Área de Cadastro	6	RF010 - Listar reservas	1
RF002 - Cadastrar usuário	38	RF011 - Listar reservas por data	1
RF003 - Login	7	RF012 - Visualizar calendário por sala	1
RF004 - Recuperar Senha	5	RF013 - Gerenciar usuários	18
RF005 - Alteração de dados	14	RF014 - Gerenciar espaços	72
RF006 - Cadastrar Reserva	23	RF015 - Cadastrar várias reservas	30
RF007 - Excluir Reserva	4	RF016 - Excluir várias reservas	4
RF008 - Visualizar detalhes da reserva	1	RF017 - Listar várias reservas	1
RF009 - Notificação de confirmação	1	RF018 - Listar reservas em lote	1
TOTAL			**228**

Fonte: Dados da pesquisa (2025)

Para a automatização dos testes, utilizou-se o *Selenium*, visando simular as ações realizadas pelo usuário na interface gráfica, em conjunto com a ferramenta JUnit, que permite validar as ações por meio de anotações *@After*, *@Before* e *@Test*. Dado os casos de testes realizados, a maioria dos erros identificados foi relacionada à interligação entre o banco de dados e a interface. As falhas ocorrem ao lidar com os limites mínimos e máximos dos atributos numéricos, como capacidade e matrícula, além de limites quantitativos de caracteres aos atributos como nome e login, que não atendem às restrições estabelecidas no banco de dados.

Para exemplificar um dos resultados, é ilustrado na Figura 1 o fluxo de teste no qual o usuário insere um login com 200 caracteres no ambiente de cadastro no sistema. Na interface, as validações abrangem somente campos já cadastrados ou em branco. Contudo, apesar do cadastro na interface ter sido bem-sucedido, ele não é registrado no banco de dados devido ao limite de 100 caracteres, estabelecido para o atributo.

Figura 1 – Fluxo de teste do atributo login com tamanho 200



Fonte: Arquivo dos autores (2025)

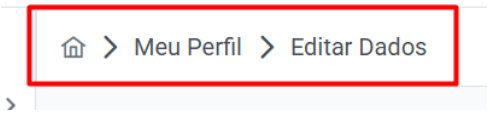
Logo, esta situação evidencia uma falha entre a integração dos componentes revelando a necessidade de manutenção no software para garantir um funcionamento correto e adequação funcional. Dessa forma, os testes de sistemas coincidem com os da inspeção no particionamento em classes por equivalência e análise de valor limite, onde o sistema não tem limites impostos.

4.2. Testes Exploratórios

A escolha da abordagem de testes exploratórios se justifica pela identificação flexível e dinâmica dos erros. A definição de ser exploratório se dá pelo motivo de não seguir um *script* definido de caso de teste, sendo embasado na *expertise* do testador. Na análise realizada, foram identificados 13 problemas que violam os estudos de usabilidade, este trabalho discorre sobre os três problemas de maior relevância.

A primeira heurística de Nielsen (MEDEIROS et al., 2020) trata da visibilidade do estado do sistema, em que o software deve indicar ao usuário o que está acontecendo; porém, o SRS viola essa regra. Na Figura 2, a barra de menu, ao ser clicada não exibe nenhuma mudança ou redirecionamento, causando confusão ao usuário. Esta ação é considerada de alta severidade devido ao fato de ser recorrente em todas as telas que apresentam essa navegação, o que pode gerar múltiplas requisições e sobrecarregar o sistema, uma vez que é um elemento clicável.

Figura 2 – Erro fluxo de navegação



Fonte: Arquivo dos autores (2025)

Outro erro encontrado fere a heurística **Minimizar a Carga de Memória do Usuário**: o sistema não informa ao usuário que a sessão expirou, o que pode causar perda de trabalho, justificando classificação de severidade alta. A heurística de **Ajuda e Documentação** é violada, com baixa severidade, uma vez que, na interface do SRS não há espaço dedicado à disponibilidade de manuais ou documentação para o usuário. De forma geral, pode-se identificar que o sistema indica fragilidades no quesito de comunicação do estado do sistema

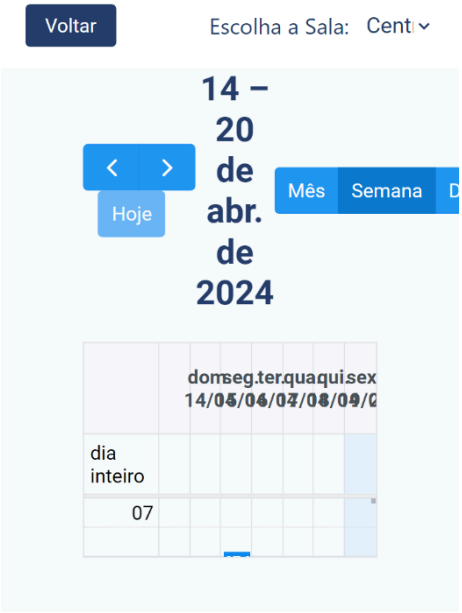
e no suporte.

4.3. Testes de Responsividade

A responsividade é o fator de qualidade no qual a tela se adapta conforme o dispositivo e o contexto de uso (ZEMEL, 2015). Dessa forma, para analisar esses testes no sistema, foram determinadas as seguintes resoluções padrão para *desktop* (1920x1080 *pixels*), *tablet* (768x1280 *pixels*) e *mobile* (375x780 *pixels*). Utilizou-se a extensão *DevTools* do Google Chrome para a captura das imagens.

Ao realizar os testes nas telas e interações do sistema, foi possível observar que os principais erros na interface, ocorrem com a mudança de telas maiores para telas menores. Para exemplificar essa questão, na Figura 3, observa-se a tela de calendário, que não se adapta adequadamente aos tamanhos de telas menores como *mobile*, existindo assim uma sobreposição e perda de legibilidade dos conteúdos na tela. Esse problema viola a heurística de Nielsen de visibilidade do estado do sistema ocorrendo em 6 das 15 páginas interativas do sistema.

Figura 3 – Tela com erro de responsividade em plataforma *mobile*



Fonte: Arquivo dos autores (2025)

4.4. Testes de Desempenho

Os testes de desempenho visam identificar falhas na garantia da qualidade de um produto no âmbito de aspectos de *performance* que expõem o sistema a cenários específicos. Assim, foi utilizada a ferramenta JMeter na realização de teste de carga e estresse. Os testes foram projetados para avaliar a capacidade de estabilidade do sistema sob cargas simuladas.

Desse modo, foram reunidos os quantitativos de carga conforme documento obtido junto à universidade, tendo as informações datadas no semestre 2023.2 (abril de 2024). Posto isto, os valores coletados são referentes aos usuários ativos da instituição, sendo: 1322 discentes; 79 docentes e 39 técnicos administrativos. Dessa forma, foram criados três cenários de testes de carga: carga mínima utilizando 360 usuários, carga média com 720 usuários e carga máxima com 1440 usuários.

As funcionalidades escolhidas para os testes de carga foram: Login, Cadastro e Recuperar Senha. A justificativa se dá por serem as únicas a não apresentar problemas com a

autenticação e com o *framework* PrimeFaces nas requisições AJAX, uma vez que, ferramentas de simulação frequentemente falham em coletar dados de páginas autenticadas, resultando em dados inconsistentes para a pesquisa (KIRAN; MOHAPATRA; SWAMY, 2015). Dentre as métricas obtidas nos relatórios gerados, algumas têm maior relevância para a análise, tendo em vista a Qualidade de Serviço (do inglês, *Quality of Service*). Dessa forma, serão consideradas as métricas de tempo de resposta (mínimo, máximo e médio), vazão e porcentagem de erro. Os testes foram iniciados pela carga de 360 usuários. Logo, identificou-se a ausência de erros nas três funcionalidades, devido à baixa demanda de requisição HTTP nessa leva de testes, demonstrado na Tabela 2.

Tabela 2 - Avaliação de desempenho com carga de 360 usuários

	Carga 360 usuários				
Funcionalidade	**Min.**	**Max.**	**Médio**	**Vazão**	**% de Erro**
Login	558	3414	2023	104,95	0,00%
Cadastro	104	762	379	228,86	0,00%
Recuperar Senha	106	1230	278	160,07	0,00%

Fonte: Dados da pesquisa (2025)

Na avaliação do tempo de resposta e vazão das páginas, observa-se na página de Login uma maior discrepância com as demais. Enquanto sua média de respostas foi superior às demais páginas, a sua vazão foi a mais baixa, isso demonstra um desempenho considerado variável e em certos pontos instável. Diante do segundo cenário apresentado (Tabela 3), o sistema apresenta os primeiros sinais de erros, principalmente na página de Login, com uma porcentagem de quase 50% de erros. Na página de Cadastro, houve uma taxa de erro de cerca de 17% e a página de Recuperar Senha se mostrou estável com 0,00% de erro.

Tabela 3 - Avaliação de desempenho com carga de 720 usuários

	Carga 720 usuários				
Funcionalidade	**Min.**	**Max.**	**Médio**	**Vazão**	**% de Erro**
Login	8871	10573	9465	67,96	44,72%
Cadastro	575	3531	2093	184,23	17,64%
Recuperar Senha	103	3926	964	147,48	0,00%

Fonte: Dados da pesquisa (2025)

Durante a realização dos testes de carga média, identificou-se uma queda significativa na vazão das funcionalidades, correspondente ao aumento no tempo de resposta da aplicação, conforme ilustrado na Tabela 3. A página que ainda não exibe alarmes quanto ao desempenho é a de Recuperar Senha, com os valores seguindo um padrão simulados na primeira carga, além de manter a porcentagem de erro zerada.

Por fim, foi executada a última leva de testes com o número total de usuários ativos da instituição, lidando com o possível estresse do sistema, demonstrado na Tabela 4. Com a intensificação dos testes, observa-se na página de Login a porcentagem de erro de 56,25% que indica uma alta taxa de erros ou falhas durante a realização do teste diante dessa carga.

Tabela 4 - Avaliação de desempenho com carga de 1440 usuários

Funcionalidade	Carga 1440 usuários				
	Min.	Max.	Médio	Vazão	% de Erro
Login	5927	16998	9990	84,44	56,25%
Cadastro	104	5471	1365	177,25	0,00%
Recuperar Senha	113	5846	1378	160,92	0,00%

Fonte: Dados da pesquisa (2025)

Ao examinar este cenário de teste, para a página de Login, é possível notar que as solicitações HTTP apresentam variações nos tempos de resposta, variando de 5.927 milissegundos a 16.998 milissegundos. Isto indica uma inconsistência no tempo de autenticação do usuário devido à demora na resposta, justificada pela vazão de 84,44. Vale destacar que a página de Cadastro apresentou zero na taxa de erros em cenários de alta carga, causada pela sobrecarga do sistema, que deixa de aceitar novas requisições. Em termos de carga média, a porcentagem foi de cerca de 17% devido ao sistema não recusar totalmente as solicitações e aceitar as que forem possíveis. A análise das páginas de Cadastro e Recuperar Senha revela variações semelhantes, com capacidades moderadas de lidar com altas cargas.

Para a análise de páginas com autenticação de usuário, a ferramenta JMeter exibe uma limitação por não permitir a realização dessas requisições, sendo inviável o envio de credenciais de usuário. Dessa forma, para mitigar esse problema, foi utilizada a ferramenta do Google Lighthouse que permite a realização de testes de desempenho, porém com uma nova interpretação. Com o Google Lighthouse é possível gerar relatórios que contém os aspectos de qualidade da página, bem como recomendações de diagnóstico de pontos identificados. A ferramenta fornece índices de qualidade (desempenho, acessibilidade, melhores práticas, *search engine optimization* (SEO) e *progressive Web app* (PWA)). Para esse estudo, só foram utilizadas as métricas de desempenho, acessibilidade, melhores práticas e SEO. A Figura 4 traz um exemplo das métricas calculadas para a página “Meu Perfil” do SRS.

Figura 4 - Métricas da ferramenta Lighthouse na página “Meu Perfil”

Fonte: Arquivo dos autores (2025)

Com o Lighthouse, foi possível observar uma qualidade de páginas mediana com base nas métricas coletadas nos quesitos de desempenho e boas práticas para *desktop*, como também de acessibilidade para *mobile*, dado o intervalo de valores para cada métrica: 0 sendo qualidade ruim e 100 sendo de boa qualidade. Dessa forma, os dois aspectos relevantes para a análise são demonstrados na Tabela 5. Os relatórios indicam que os problemas mais frequentes e graves identificados pela ferramenta foram o uso de APIs (do inglês, *Application Programming Interface*) obsoletas e a exibição de imagens com proporções inadequadas.

A fim de sintetizar a apresentação dos resultados, as páginas foram avaliadas e divididas em cinco módulos, de acordo com a natureza das funcionalidades: Reservas, Salas, Usuários, Consultas e Autenticação.

Tabela 5 - Desempenho das páginas de desktop e mobile - Lighthouse

Páginas	Desktop		Mobile	
	Desempenho	Acessibilidade	Desempenho	Acessibilidade
Reservas (Home, Nova Reserva, Reserva em Lote, Minhas Reservas, Todas as Reservas, Listar Lotes)	82	92	57	88
Consulta (Consulta, Calendário)	75	89	52	89
Salas (Nova Sala, Listar Salas)	85	91	58	88
Usuários (Usuários, Meu Perfil)	85	91	57	88
Autenticação (Ativar Conta, Login, Cadastro, Recuperar Senha)	87	95	58	95
MÉDIA DAS MÉDIAS	83	92	56	90

Fonte: Dados da pesquisa (2025)

Sendo assim, foram identificados mais de 400 pontos de melhorias em ambas plataformas (*desktop* e *mobile*). Dos pontos identificados, cerca de 230 foram categorizados como críticos, sendo que, aproximadamente, 66% estão relacionados ao aspecto de desempenho do sistema. Ademais, a média das páginas *mobile* em relação à métrica de desempenho é notável, apresentando um resultado médio de 56, o que é visto como uma média baixa, em linha com os resultados dos testes responsivos sobre a baixa usabilidade em dispositivos móveis.

5. Conclusões

Este estudo, avaliou por meio de testes de sistema o Sistema de Reserva de Salas de uma universidade para contribuir com a identificação dos erros e falhas diante das análises. No quesito de testes funcionais, 100% das funcionalidades do sistema foram testadas, com realização de 228 casos de testes manuais. Os erros mais relevantes nessa etapa foram identificados na integração entre componentes, banco de dados e sistema. Com os testes exploratórios que foram conduzidos considerando as heurísticas de Nielsen, a que teve a maior recorrência de problemas foi a de “visibilidade do estado do sistema”, visto a não responsividade do sistema em plataformas com telas menores (*mobile*). A análise demonstrou que o SRS não exibe uma taxa positiva de responsividade, tendo, em sua maioria, telas com problemas na legibilidade das informações.

No quesito de testes não funcionais, identificou-se falhas na operabilidade do SRS perante cargas elevadas. Destaca-se, a página de Login que emitiu maior discrepância em seu tempo de respostas e porcentagem de erro em relação às demais páginas testadas. Na ferramenta Lighthouse a página que teve maior destaque foi “visualizar calendário de salas”, com taxa de 67 para *desktop* e 48 para *mobile* no quesito de desempenho, considerada como nota baixa.

Este trabalho traz a importância da utilização de estratégias de testes em um sistema Web, no aspecto funcional e comportamental. Dessa forma, as contribuições deste trabalho visam identificar os problemas de manutenção e pontos de falha, de modo a proporcionar um estudo de caso de ferramentas e achados em um sistema Web diante de testes funcionais. Este estudo apresenta limitações que devem ser consideradas na simulação de teste de desempenho: as páginas autenticadas não tiveram script de teste por questões de arquiteturas antigas utilizadas no sistema. Portanto, visando a continuidade da pesquisa, pretende-se avaliar o SRS na implantação de testes de segurança diante do gerenciamento de riscos e proteção das informações vinculadas ao software.

Referências

AQUILES, A.; FERREIRA, R. *Controlando versões com Git e Github.* Casa do Código, 2014.

BARBOSA, S.; SILVA, B. *Interação Humano-Computador.* Elsevier Brasil, 2010.

COELHO, B. *Tipos de pesquisa: abordagem, natureza, objetivos e procedimentos.* 2019. Disponível em: https://blog.mettzer.com/tipos-de-pesquisa/. Acesso em: 12 maio 2026.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. *Introdução ao Teste de Software.* Elsevier, 2016.

GONÇALVEZ, P. F.; BARRETO, J. S.; ZENKER, A. M.; FAGUNDES, R. D. R.; ROCHA, B. C.; BIRFELD, K.; TEIXEIRA, M. R. W. *Testes de Software e Gerência de Configuração.* Grupo A, 2019.

HIRAMA, K. *Engenharia de Software: Qualidade e produtividade com tecnologia.* Grupo GEN, 2011.

IEEE STANDARDS ASSOCIATION. *IEEE Standard Glossary of Software Engineering Terminology.* IEEE Std. New York, NY, p. 1 – 84, 1990. DOI: https://doi.org/10.1109/IEEESTD.1990.101064.

KIRAN, S.; MOHAPATRA, A.; SWAMY, R. Experiences in performance testing of web applications with unified authentication platform using jmeter. In: IEEE. International Symposium on Technology Management and Emerging Technologies (ISTMET). [S.l.], 2015. p. 74–78. DOI: https://doi.org/10.1109/ISTMET.2015.7359004.

KRUG, S. *Não me faça pensar: Uma abordagem de bom senso à usabilidade na web.* 2. ed. Alta Books, 2018.

KUROSE, J. F.; ROSS, K. W. *Redes de computadores e internet: uma abordagem top-down.* São Paulo: Pearson, Porto Alegre: Bookman, 2021.

MEDEIROS, S.R.S.; RABELO, H.; MEDEIROS, A.; LIMA, C.; JÚNIOR, H.; MARTINS, C. A.; *Proposta de Avaliação Heurística de uma Plataforma Web para a Difusão do Pensamento Computacional no Ensino Fundamental*. *In*: Anais do Congresso sobre Tecnologias na Educação (CTRL+E). Porto Alegre: Sociedade Brasileira de Computação, p. 395-404, 2020 DOI: https://doi.org/10.5753/ctrle.2020.11417.

PERES, H. *Automatizando Testes de Software Com Selenium.* Simplíssimo, 2016.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software.* Grupo A, 2021.

ROGERS, Y.; SHARP, H.; PREECE, J. *Design de Interação: Além da Interação Humano-Computador.* 3. ed. Porto Alegre: Bookman, 2013.

SOMMERVILLE, I. *Engenharia de software, 10th.* Pearson Education do Brasil, 2018.

TEIXEIRA, F. *Introdução e boas práticas em UX Design.* Casa do Código, 2014.

VALENTE, M. T. *Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade.* Editora: Independente, 2020.

VAZQUEZ, C.; SIMÕES, G. *Engenharia de Requisitos: software orientado ao negócio.* Brasport, 2016.

ZEMEL, T. *Web Design Responsivo: páginas adaptáveis para todos os dispositivos.* Editora Casa do Código, 2015.