Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

COMPARATIVO ENTRE FERRAMENTAS DE AUTOMAÇÃO DE TESTES DE SOFTWARE PARA SISTEMAS WEB[1]

COMPARATIVE BETWEEN SOFTWARE TESTING AUTOMATION TOOLS FOR WEB SYSTEMS

André Fernando Rollwagen[2], Joseane Amaral[3], Eduardo Rodrigues[4], Ricardo Vanni Dallasen[5], Maikon Cismoski dos Santos[6], Vanessa Lago Machado[7]

[1] Monografia do Curso de Tecnologia em Sistemas para Internet do IFSUL - Instituto Federal de Educação Ciência e Tecnologia Sul-Rio-Grandense, Campus Passo Fundo
[2] Orientador Professor Me. do Curso de Tecnologia em Sistemas para Internet do IFSUL - Câmpus Passo Fundo.
[3] Professora Dra. do Curso de Tecnologia em Sistemas para Internet do IFSUL - Câmpus Passo Fundo.
[4] Aluno do Curso de Tecnologia em Sistemas para Internet do IFSUL - Câmpus Passo Fundo.
[5] Professor Me. do Curso de Tecnologia em Sistemas para Internet do IFSUL - Câmpus Passo Fundo.
[6] Professor Me. do Curso de Tecnologia em Sistemas para Internet do IFSUL - Câmpus Passo Fundo.
[7] Professora Ma. do Curso de Tecnologia em Sistemas para Internet do IFSUL - Câmpus Passo Fundo.

Resumo

Diferentes plataformas de softwares surgiram com o avanço computacional, como a própria internet, muito utilizada por aplicações WEB, portais e outros tipos de aplicações. Muitos dos sistemas passam por etapas de testes, tanto no desenvolvimento quanto em produção. A automação de testes tem grande relevância ao testar as funcionalidades de um sistema, por meio do uso das ferramentas de teste de software automatizadas pode-se economizar tempo considerável do testador. A presente pesquisa apresenta um estudo comparativo das ferramentas automação de testes de software Badboy, Selenium IDE e Sikuli. Um sistema WEB foi desenvolvido como ambiente de testes, permitindo aplicar as ferramentas e testar as suas funcionalidades. Desta forma, vantagens e limitações das ferramentas selecionadas foram identificadas através de diferentes aspectos analisados, como métodos, tratamento com código, tempos de execuções e documentação.

Abstract

Different software platforms have emerged with computational advancement, such as the Internet itself, widely used by WEB applications, portals and other types of applications. Many of the systems go through testing steps, both in development and in production. Test automation is of great relevance when testing a system's functionality, using automated software testing tools can save considerable tester time. This research presents a comparative study of the software testing automation tools Badboy, Selenium IDE and Sikuli. A WEB system was developed as a testing environment, allowing to apply the tools and test their functionality. In this way, the advantages and limitations of the selected tools were identified through different aspects analyzed, such as methods, treatment with code, execution times and documentation.

Palavras-chave: Teste de software, Automação de testes de software, Ferramentas de teste de software.

Keywords: Software testing, Software testing automation, Software testing tools.

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

1 INTRODUÇÃO

O desenvolvimento de software com a utilização de metodologias ágeis aliadas ao avanço tecnológico vem mostrando eficiência e rapidez no processo de desenvolvimento. A entrega de um software que poderia demorar meses, pode ser finalizada em semanas, dependendo da complexidade do mesmo. Devido ao surgimento de frameworks, bibliotecas prontas e as próprias linguagens de programação que evoluiram com o tempo, tornaram o processo de desenvolvimento mais ágil e simples, assim, é possível entregar mais softwares em menos tempo. Porém com toda essa agilidade, a qualidade de muitos dos programas é posta à prova, além de o sistema ser finalizado com bugs e falhas.

Uma prática a ser usada no desenvolvimento de programas é acrescentar etapas de testes de software antes de ser concluído e após sua finalização. Um teste pode, além de agregar qualidade ao produto final, prevenir grandes perdas como: tempo, dinheiro e diversos outros tipos de danos. Os sistemas que são classificados como aplicações WEB, exigem um tipo de teste mais especifico, com inserção de valores repetidamente, os quais são testados usando as métricas de automação de testes. Para facilitar a aplicação destes testes, que exigem do usuário uma atenção redobrada para analisar e testar cada parte do sistema, cujo pode ser extremamente grande e fazer com que a experiência do testador seja exaustiva, se torna muito mais viável a utilização de ferramentas de automação de testes. Tais ferramentas podem auxiliar o testador a criar um teste e replicar seu script a todas as situações semelhantes a que ele deseja testar. Na automação de testes existem muitas ferramentas que auxiliam no processo, porém focadas nos testes como um todo. Essa generalização pode prejudicar um usuário que deseja utilizar uma ferramenta para automatizar seus testes, pela falta de detalhes na documentação das ferramentas para auxiliar na escolha de qual delas pode ser mais adequada em relação a sua necessidade.

As diferentes características presentes nas ferramentas de automação de testes de software, permitem a elaboração de um estudo através de uma análise comparativa, verificando vantagens e limitações que estas podem apresentar ao implementar determinadas rotinas de teste em um sistema.

Os aspectos negativos podem influenciar ao definir qual ferramenta pode ser utilizada pelo testador e os aspectos positivos, quando unificados, podem ser empregados no desenvolvimento de uma aplicação que complete os recursos mais vantajosos identificados em cada teste executado por determinada ferramenta. “Uma ferramenta de teste reduz a intervenção humana nos resultados obtidos, aumentando a qualidade de teste. Influencia diretamente a confiabilidade do software testado” (SOMMERVILLE, 2007, p. 68).

Uma comparação entre ferramentas de teste que tem o mesmo seguimento, como, por exemplo, automação, influencia diretamente na escolha do software para testar uma aplicação. Com uma ferramenta de testes que possui uma descrição que apresenta seus pontos positivos em determinado modelo de testes, ela, se torna mais recomendada em relação a outra que obteve um resultado inferior, agregando qualidade no teste como um todo.

Este trabalho tem por objetivo realizar um comparativo entre ferramentas de testes automatizados de software para o desenvolvimento de aplicações Web.

2 METODOLOGIA

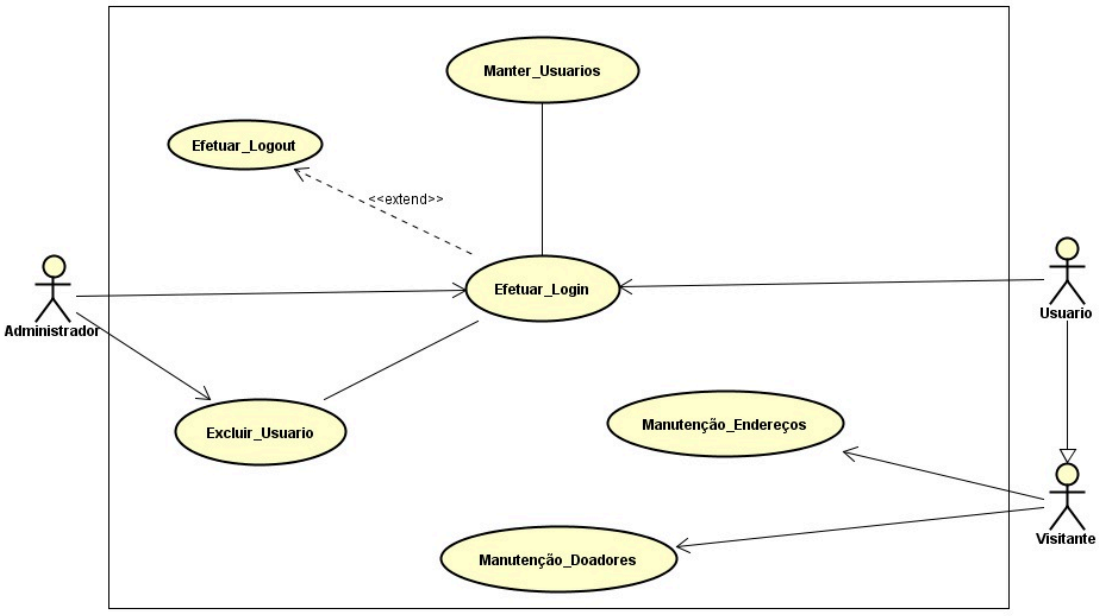
O desenvolvimento metodológico deu-se pela modelagem da plataforma de testes, seguida da implementação do software usado para aplicação dos testes, criação dos casos de teste, execução dos testes e análise comparativa dos resultados para apresentação dos pontos positivos e negativos das ferramentas escolhidas.

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

2.1 Modelagem da Plataforma de Testes

Para modelagem da plataforma de testes foram criados esquemas lógicos de uma plataforma de testes, que segundo Sommerville (2011), é conceituada como um modelo independente de plataforma, em princípio, gerando um programa de trabalho para servir de teste.
Neste contexto elencaram-se os seguintes requisitos funcionais: cadastrar usuários, excluir usuários, alterar usuários, listar registros, efetuar login, efetuar log out, busca registros, filtrar registros, imprimir, download. Já os requisitos não funcionais envolveram aplicação para WEB, persistir registros em um banco de dados e exibí-los na tela do sistema, e controle de permissões.
O diagrama de casos de uso representante da plataforma de teste do sistema WEB desenvolvido apresentado na Figura 1, tem o objetivo de documentar o que o sistema propõe ao ponto de vista do usuário. "Ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema, porém, não nos aprofundamos em detalhes técnicos que dizem como o sistema faz" (RIBEIRO, 2016, p.1).

Figura 1 - Diagrama de caso de uso da Plataforma de Teste



Fonte: do autor, 2018.

Para plataforma de testes desenvolveu-se uma aplicação WEB tendo como função um portal de notícias e cadastro de usuários. Este apresenta interações com recursos de textos, imagens, vídeos, download, segurança de informação e interação com API. Segundo Faria (2015), aplicação WEB é um serviço independente de hardware, ou seja, está se referindo a sites ou sistemas onde parte do desenvolvimento fica hospedado em um servidor na nuvem, e o cliente basicamente não necessita ter instalado em seu dispositivo para utilizá-las, além de um navegador (browser).

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

2.2 Implementação da Plataforma de Testes

Para implementação da ferramenta utilizou-se a IDE Netbeans em ambiente Windows (NETBEANS, 2018), Java foi a linguagem utilizada para o desenvolvimento da camada de modelo do sistema, que dentro das especificações JavaEE, foi implantado o componente JavaServerFaces (JSF) que foi usado para desenvolver a parte Web (ORACLE, 2018). Toda a estrutura, como menu, campos de entrada de texto, hyperlinks, imagens e botões são gerados pelo framework PrimeFaces (PRIME, 2018). A partir desta estrutura, somada à estilização CSS (CASTRO, 2013) e ao Java JSF, torna-se possível apresentar um site organizado e com boa usabilidade. A linguagem XHTML foi usada para a criação de scripts do lado do servidor (PROFFITT, 2001). O controle dos registros é permitido totalmente ao Sistema Gerenciador de banco de dados PostgreSQL (OBE, 2017), que por sua vez está hospedado na máquina do autor do trabalho.

2.3 Ferramentas de Teste de Software

Uma ferramenta que dá suporte ao teste de software se torna útil quando, visando uma maior agilidade nas atividades do processo de teste, a sua utilização de suporte ao teste pode contribuir para consideráveis ganhos de tempo (ELIZA, 2013).

2.3.1 BadBoy

A ferramenta de teste de software BadBoy é um programa que auxilia na implementação de testes automatizados de software, por esse motivo foi selecionada, com a finalidade de ser avaliada pelo autor do presente trabalho. O BadBoy utiliza do método de automação Record and Play, que consiste em ações dos botões de gravação: Play, Stop e Pause. As ações do método se baseiam respectivamente em: Início da gravação das ações da tela, parada de gravação e pausa na gravação. Além de ser muito útil para testes de validação a ferramenta Badboy atribui o código fonte a tags HTML (LIMA, 2014).

2.3.2 Sikuli

A ferramenta baseia-se em reconhecimento de imagem para realizar ações na tela do computador do usuário, como clicar, mover o mouse e digitar, entre outras ações, podendo assim ser usada para testar software independente da interface utilizada. O propósito do Sikuli é poder testar qualquer aplicação que apresenta uma interface com o usuário, é baseado na linguagem Jython (Java + Python). A linguagem Python pode ser utilizada para a criação da biblioteca, uma vez que apresenta uma linha de aprendizagem extensa e pode ser integrada ao ambiente. O foco do programa é o teste automatizado de software, comparada com as demais ferramentas do mesmo seguimento, o Sikuli possui singularidades consideráveis, as quais, tornam a ferramenta uma das únicas disponíveis que trabalha com o conceito de passos de imagem. Seus planos de teste são criados em sequências de screenshots (imagens referentes a captura de tela), para que quando for salvo o script de teste, seguir a ordem em que as imagens foram selecionadas.

2.3.3 Selenium

A ferramenta de teste de software Selenium IDE é baseada em métodos funcionais de teste, que são derivados de gravações na tela. A grande diferença do Selenium IDE com as demais ferramentas, é

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

a sua instalação, pois ele é instalado como um plugin no navegador, diferente das outras ferramentas que são instaladas nas máquinas. Uma das grandes vantagens do Selenium é a possibilidade da realização de testes em qualquer navegador que tenha suporte ao JavaScript. A esse respeito, Nogueira (2014, p.1) declara: "Selenium IDE (Integrated Development Environment) é uma ferramenta utilizada para o desenvolvimento de scripts de teste com o Selenium através de um plugin a partir do Firefox 6". O que torna o desenvolvimento dos scripts mais eficientes pelo método Record and Replay (Gravação e Execução) é a mobilidade que a ferramenta possui, por ser acoplada a um navegador (COSTA, 2014).

2.4 Casos de Teste de Software

Todo teste de software é uma projeção de possíveis falhas que o sistema possa apresentar após sua implementação, o mesmo busca revelar falhas que antes não eram vistas. Antes de uma aplicação de testes, tanto automatizados quanto manuais, o testador deve analisar os possíveis caminhos que deverão ser percorridos na execução dos testes. Segundo Sommerville (2011) a documentação destes testes é classificada como casos de teste, que permite aos testadores simular todas as rotas que deverão ser seguidas e implantadas para atingir o objetivo final do teste.
Os casos de teste podem ser escritos em forma de documento texto, onde o testador fará as classificações dos passos a serem seguidos, declarando os valores e nomes de variáveis que possivelmente serão transferidas no teste. Como exemplo de casos de teste, a tabela 1 apresenta casos de teste CRUD usuários.

Tabela 1 - Caso de teste CRUD usuário

Objetivo do Teste:	Garantir que os registros do sistema possam ser persistidos, alterados e excluídos com sucesso.
Pré-Condições:	Ser Usuário e Administrador do Sistema.
Prioridade:	Alta
Tipo de Execução:	Automatizado
Passo 1:	Logar no sistema. Resultados esperados: O login deve ser efetuado com sucesso.
Passo 2:	Clicar em "Cadastros" , "usuários". Resultados Esperados: As listagens de usuários com os botões de edição devem aparecer.
Passo 3:	Clicar em "+ Novo". Resultados Esperados: O formulário de cadastros deve ser aberto ao usuário.
Passo 4:	Clicar em "Alterar". Resultados Esperados: O formulário de edição de cadastros deve ser aberto.
Passo 5:	Clicar em "Excluir". Resultados Esperados: O usuário deve ser excluído da base de dados.

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

Após a criação dos casos de teste com exemplificado na tabela 1, os testes foram executados de forma automatizada utilizando as ferramentas para testes de software automatizados. Posteriormente, os resultados foram analizados, sendo criado quadros comparativos sobre essas ferramentas.

3 RESULTADOS E DISCUSSÃO

Os casos de testes foram executados de forma automatizada após o desenvolvimento da plataforma de testes, para execução dos testes foram utilizadas as três ferramentas citadas no presente trabalho, seus resultados foram separados em quatro seções descritas neste capítulo. A tabela 2 apresenta uma legenda para os resultados dos testes.

Tabela 2: Legenda para os resultados dos testes

SUCCESS	Teste concluído sem erros
OK	Teste concluído, mas com algum tipo de erro
FAILED	Teste não concluído

3.1 Funcionalidades

Para testar as funcionalidades os casos de teste foram baseados nos seguintes casos de uso: Buscar CEP, Listar Doadores, Efetuar Login e Efetuar Loggout, assim, foram aplicados os testes especificados no caso de teste erro login, sucesso login, teste loggout e listar doadores. A tabela 3 apresenta os resultados das ferramentas em relação aos casos de testes aplicados.

Tabela 3. Legendas dos Testes de Funcionalidades

	ErroLogin	**SucessoLogin**	**TesteLogout**	**ListarDoadores**
BadBoy	SUCCESS	SUCCESS	SUCCESS	FAILED
Selenium	SUCCESS	SUCCESS	SUCCESS	SUCCESS
Sikuli	OK	OK	SUCCESS	OK

Após a aplicação dos casos de teste, com base na tabela 3, pode-se perceber que apenas a ferramenta Selenium IDE concluiu todos os casos de teste sem nem uma falha. A ferramenta Sikuli finalizou todos os casos de teste, porém com algum tipo de erro, cujos foram resolvidos utilizando de suas próprias funcionalidades. A ferramenta BadBoy obteve sucesso nos casos de teste ErroLogin, SucessoLogin e Teste Loggout, porém falhou ao automatizar o caso ListarDoadores.

3.2 Tratamento com o Código

Nesta seção são avaliados os atributos gráficos de cada ferramenta, referente ao tratamento de um bloco automatizado manipulado diretamente no código. As ferramentas Selenium IDE e Badboy, possuem esse recurso localizado no canto superior esquerdo da tela, onde ficam seus "Tests Suits", na ferramenta Sikuli o mesmo se encontra na parte central. Nas três ferramentas o script que é gerado ao longo dos testes vai sendo replicado por ordem de criação. Para alterar determinada parte do script basta expandí-lo, então, cada uma das ferramentas gera o seu próprio tratamento de caso. Baseados nos casos de teste citados na tabela 4 podem ser observados os resultados obtidos.

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

Tabela 4. Legendas dos Testes de Tratamento de Código

	OrdenarRegistros	BuscarNome	Imprimir
BadBoy	SUCCESS	SUCCESS	FAILED
Selenium	SUCCESS	SUCCESS	SUCCESS
Sekuli	SUCCESS	SUCCESS	FAILED

Após a aplicação dos casos de teste, como pode ser visto na tabela 4, as três ferramentas concluíram sem nem uma falha os casos de teste Ordenar Registros e Buscar Nome. A ferramenta Selenium IDE concluiu com êxito todos os casos de teste declarados nesta sessão. As ferramentas BadBoy e Sikuli falharam na execução do caso Imprimir. Neste caso de teste, nas ferramentas Sikuli e BadBoy, não foi disponibilizado um código de tratamento para funcionalidades JSF que realizam a conexão com dispositivos externos, cujos implementam funções de downloads programadas na aplicação.

3.3 Tempos de Execução

Nessa seção é avaliado o desempenho de cada ferramenta ao aplicar os casos de teste CRUD Usuários e Manutenção de Endereços. Foram realizados comparativos entre os tempos obtidos em cada ferramenta ao realizarem os testes citados anteriormente. Também são comparados os tamanhos de cada arquivo de teste, quando o mesmo é finalizado. O teste ao ser salvo por uma ferramenta se transforma em um arquivo. Cada ferramenta testada neste trabalho gera um arquivo de teste de acordo com suas funcionalidades. Os tamanhos de arquivos de teste variam de cada ferramenta, pois o mesmo teste pode ter sido acrescentado passos adicionais, em relação a outro programa. Um arquivo de teste ao ser inicializado pela ferramenta que lhe construiu, contempla todas as funcionalidades salvas anteriormente no seu desenvolvimento. Todas as ferramentas testadas no presente trabalho possuem características diferentes ao criarem um arquivo de teste, porém elas têm em comum um "log" de cronometragem, referente ao tempo de execução de um teste, através desta funcionalidade tornou-se possível compará-los entre si e descrever conclusões sobre os resultados obtidos, como mostram as tabelas 5,6 e 7.

Tabela 5: Legendas da Ferramenta BadBoy sobre Tempos de Execução

BadBoy	**CRUD Usuários**	**Manutenção Endereços**
Status	OK	SUCCESS
Tempo	Inserir: 3902 (ms) Alterar: 4020 (ms) Excluir: 1502 (ms)	1133 (ms)
Tamanho	Inserir: 598,00 KB Alterar: 688,00 KB Excluir: 201,00 KB	116,00 KB

Evento: XXV Jornada de Pesquisa
ODS: 4 - Educação de qualidade

Na ferramenta BadBoy, ao executar o caso "CRUD USUÁRIOS", a ferramenta precisou carregar a página inteira antes de aplicar a sua automação. Isso influenciou diretamente na agilidade do teste. A ferramenta Selenium finalizou com sucesso os casos de teste, além de obter os melhores tempos no caso CRUD, como pode ser visto na tabela 6, e também produziu arquivos mais leves que as demais ferramentas.

Tabela 6: Legendas da Ferramenta Selenium IDE sobre Tempos de Execução

Selenium	**CRUD Usuários**	**Manutenção Endereços**
Status	SUCCESS	SUCCESS
Tempo	Inserir:1900 (ms) Alterar:2124 (ms) Excluir: 603 (ms)	533 (ms)
Tamanho	Inserir:3,00 KB Alterar:3,91 KB Excluir: 1,10KB	1,00 KB

A ferramenta Sikuli, ao executar o caso de teste "MANUTENÇÃO ENDEREÇOS", apresentou erro na funcionalidade "dragDrop", o segundo parâmetro não reconheceu o atributo como uma imagem. A solução foi editar o método persistindo um campo de imagem diferente.

Tabela 7 - Legendas da Ferramenta Sikuli sobre Tempos de Execução

Sikuli	**CRUD Usuários**	**Manutenção Endereços**
Status	SUCCESS	OK
Tempo	Inserir:3121 (ms) Alterar:3500 (ms) Excluir: 1102 (ms)	1578 (ms)
Tamanho	Inserir:9,72 KB Alterar:7,64 KB Excluir: 15,7 KB	3,28 KB

3.4 Documentação

Nesta etapa são avaliadas as documentações de cada uma das ferramentas analisadas no presente trabalho. Todas as ferramentas testadas possuem uma aba no seu ambiente gráfico que referencia os documentos de auxílio ao usuário. A ferramenta BadBoy apresenta um pequeno sistema para exibir a sua documentação, com uma interface gráfica para gerenciar os documentos, a qual inicia com um manual de instrução das funções básicas, finalizando a seção com uma introdução sobre automação de testes. As demais sessões de tutoriais da ferramenta BadBoy, são manuais de métodos, testes de

carga e tratamento de scripts. Toda a documentação da ferramenta, disponibilizada dentro da mesma, é escrita em inglês.

A ferramenta Selenium IDE conta com uma documentação atrelada ao seu próprio site, porém é referenciada na aba "ajuda". Uma interface gráfica construída em HTML é alimentada com manuais de usabilidade da ferramenta Selenium IDE. A documentação da ferramenta Selenium ainda oferece manuais para a criação de casos de teste, alertas de popUp, tratamento de variáveis, erros de sintaxe e até mesmo depurador de bugs de código interno da aplicação. Toda a documentação oferecida pela ferramenta Selenium IDE está escrita em inglês.

A documentação da ferramenta Sikuli é escrita no seu próprio Blog, porém em sua interface gráfica é disponibilizada através da aba "ajuda" o respectivo caminho (link) para o Blog. Esta documentação baseia-se em categorias de produtividade, compilação de testes e sugestões de uso da ferramenta em jogos. Todos os itens descritos no manual da ferramenta Sikuli estão disponibilizados através da aba "guia completo sikuli", onde são apresentados links para opiniões de desenvolvedores, documentação de versões anteriores e métodos de conexões para a linguagem Java. Toda a documentação da ferramenta Sikuli está escrita em inglês.

4 CONSIDERAÇÕES FINAIS

Atualmente é cada vez mais raro que uma aplicação web de grande porte seja desenvolvida sem ser testada, tanto pelas métricas manuais quanto automatizadas. O teste em geral permite ao desenvolvedor entregar um produto mais confiável e qualificado ao cliente. Porém quanto maior uma aplicação, mais tempo será necessário para ser testada. A aplicação de testes automatizados se torna muito importante nesse quesito. Uma vez que com o auxílio de uma ferramenta para automatizar um teste, o script gerado é salvo e disponibilizado para todas as ocorrências, iguais ou semelhantes que necessitam dos mesmos parâmetros em outro teste.

Assim, o presente trabalho teve como objetivo o estudo comparativo entre as ferramentas de automação de testes de software Badboy, Sikuli e Selenium IDE. A partir de um sistema WEB proposto como ambiente de testes, foi possível identificar aspectos relevantes em cada uma das ferramentas. Vale ressaltar que todas as ferramentas comparadas nesse estudo cumpriram com o objetivo de implementar os casos de testes no sistema WEB desenvolvido.

O BadBoy mesmo possuindo um navegador próprio, o que teoricamente deveria solucionar muitos bugs referentes a automação em sistemas WEB, mostrou-se menos eficaz em testes de performance, comparado as outras ferramentas. Diferente do Sikuli e Selenium IDE, ele disponibiliza seus comandos exibidos por ícones, o que dificultou muito a depuração do script e a visualização de erros do teste. Além do tempo de espera de carregamento da página do seu próprio navegador interferir diretamente na gravação do teste. O teste após finalizado pela ferramenta BadBoy se transforma em um arquivo com a extensão record, isso acabou gerando arquivos muito maiores em comparação as demais ferramentas. Porém a documentação da ferramenta BadBoy demonstrou ser a mais completa em relação aos manuais de usabilidade oferecidos aos seus usuários.

O Sikuli mesmo contando com uma interface discreta comparado as outras ferramentas, mostrou ser uma excelente escolha para aplicação de automações de pequeno porte. Forneceu todos os recursos necessários para a implantação dos casos de teste envolvendo funcionalidades, tratamento de código e tempos de execução, porém, apresentou erros ao tratar componentes que estavam ocultos na tela. O reconhecimento de imagem disponibilizado na automação da ferramenta Sikuli, concluiu todos os casos de testes referenciados no presente trabalho, mesmo quando a automação não reconheceu determinado componente do formulario foi possível tratá-lo diretamente no script do teste.

O Selenium IDE foi a ferramenta com o melhor desempenho nos testes de tempo, funcionalidades

e tratamento com o código. Sem dúvida além de ser descrita como uma IDE, a ferramenta é recomendada para qualquer automação de teste, ela se destaca em relação ao BadBoy e Sikuli por conter enumeras opções de tratamento de HTML e JavaScript, o que é fundamental em testes de aplicações WEB. Além de conter uma interface simples e intuitiva o Selenium IDE gerou os arquivos de testes em códigos HTML, que ao serem abertos pela ferramenta são traduzidos como um teste. A ferramenta Selenium, por ser atrelada a um navegador, conseguiu reconhecer todos os elementos JSF implementados na plataforma de teste do presente trabalho.

Cada ferramenta, de acordo com seus próprios métodos, realizaram todas as automações descritas nos casos de testes, porém, cada uma possui uma área de teste mais recomendada sobre seu uso. A ferramenta de teste Sikuli apresar de conter apenas o método de reconhecimento de imagens, é recomendada para automações em sistemas de pequeno porte, além da sua velocidade de automação e reconhecimento gráfico. A ferramenta BadBoy, já possui um método ultrapassado de reconhecimento de HTML, pois a WEB convencional é desenvolvida com mais atributos agregados ao HTML, isso quebra as automações do Badboy, que é recomendado para páginas específicas que contenham testes pequenos de inserção de dados. Na ferramenta Selenium IDE notou-se uma grande variedade de recomendações do seu uso, em sistemas de pequeno e médio porte, a sua automação poderosa contém todos os recursos necessários para suprir um sistema de médio porte.

Como planejamentos para trabalhos futuros, pode-se estender o estudo da ferramenta Sikuli, desenvolvendo uma interface acoplada a um navegador, onde suas funcionalidades iniciais estejam presentes e acrescentadas a um módulo de tratamento para linguagem HTML e Javascript, usando como base os testes de código da ferramenta Selenium IDE identificados neste trabalho.

REFERÊNCIAS

CASTRO, E. HYSLOP, B. HTML5 E CSS3. Guia Prático e Visual. Tradução da 7a Ed. Alta Books. 2013.

COSTA, G. SIKULI – O QUE É ? COMO UTILIZAR ?. 2014. Disponível em: <https://guilerme18.wordpress.com/2014/02/13/sikuli-o-que-e-como-utilizar/>. Acesso em: 04 outubro. 2017.

ELIZA, Renata. Ferramentas de suporte ao Teste de Software. 2013. Disponível em: <http://www.devmedia.com.br/ferramentas-de-suporte-ao-teste-de-software/28642> Acesso em: 22 agosto. 2018.

FARIA, Thiago. Java EE 7 com JSF, PrimeFaces e CDI. 2.ed. São Paulo: Editora USP, 2015.

LIMA, J. Automação de testes com Badboy. 2014. Disponível em:<http://www.qualister.com.br/blog/automacao-de-testes-com-badboy-parte-1-introducao>. Acesso em: 20 outubro. 2017.

NETBEANS. Apache NetBeans IDE. Disponível em: <https://netbeans.org>. Pesquisado em: 20 de abril. 2018.

NOGUEIRA, E. Introdução ao Selenium IDE. 2014. Disponível em: <http://www.qualister.com.br/blog/introducao-ao-selenium-ide> Acesso em: 04 outubro. 2017.

OBE, R. O.; Hsu. L. S. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open

Source Database. eBook Kindle. 2017.

ORACLE. Java EE at a Glance. Disponível em: <https://www.oracle.com/br/java/technologies/java-ee-glance.html>. Pesquisado em: 10 de abril. 2018.

PRESSMAN, Roger S. Engenharia de Software - Uma Abordagem Profissional. 7.ed. Amgh Editora, 2011.

PRIME. Leading Provider of Open Source UI Component Libraries. Disponível em: <https://www.primefaces.org/>. Pesquisado em: 02 de maio. 2018.

PROFFITT, B. Xhtml - Desenvolvimento Na Web. Pearson - Education. 2001.

RIBEIRO, Leandro. O que é UML e Diagramas de Casos de Uso. 2016. Disponível em:< https://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 25 de abril. 2018.

SOMMERVILLE, Ian. Engenharia de software. 9.ed. São Paulo: Editora PEB, 2011.

Parecer CEUA: 3.069.588