Vantagens e Dificuldades de se Testar Serviços Web com Selenium Web Driver: Um Relato de Experiência

Wellyson Dannilo Soares de Sousa[1]

Departamento de Ciências Exatas – Universidade Federal da Paraíba (UFPB)
Campus IV – Rua da Mangueira, S/N – Companhia de Tecido Rio Tinto CEP 58297-000 – Rio Tinto – PB - Brasil

{wellyson.dannilo@dcx.ufpb.br}

Abstract. *Software Testing Automation is gaining space inside the developers' tasks. Test Automation makes sure to find unconformities in a faster and efficient way. Selenium Tests were prepared and executed for the web service SisAlfa, which is a system developed to support alphabetization. It was possible to observe the advantages and limitations of its use in the automation of the system. The objective of this work is to share these observed aspects in this experience to encourage and give paths to automation to be possible in different projects, mostly involving Information Technology and Communication graduation students. It was observed in this analysis that automation has advantages in scenarios where it is necessary to execute the same tests repeatedly. On the other hand, the effort needed to develop the tests' scripts couldn't have advantages in scenarios where manual tests are simple or running the same test repeatedly is not necessary.*

Resumo. *A automação de teste de software vem ganhando um grande espaço dentro das atividades de um desenvolvedor. A automação de testes torna mais eficiente e rápido o encontro de inconformidades em um software. Foram realizados testes utilizando o Selenium WebDriver no serviço Web SisAlfa, um serviço criado para apoiar a alfabetização, e foi possível observar vantagens e limitações de sua utilização na automatização dos testes deste sistema. O objetivo deste trabalho é compartilhar estes aspectos observados nessa experiência de modo a incentivar e dar caminhos para a automatização em diferentes projetos, principalmente envolvendo alunos de graduação na área de Tecnologia da Informação e Comunicação. Observou-se no caso analisado que a utilização da automação é vantajosa nos cenários onde é necessário executar repetidas vezes o mesmo teste. Por outro lado, o esforço exigido para desenvolvimento dos scripts de teste pode não ser vantajoso em cenários onde os testes manuais sejam simples ou onde não seja necessário realizar tantas repetições da execução dos mesmos.*

[1] Trabalho de conclusão de curso, sob orientação da professora Ayla Débora Dantas de Souza Rebouças submetido ao Curso de Bacharelado de Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

1 Introdução

O teste de software é uma das principais atividades realizadas para melhorar a qualidade de um produto ou serviço em desenvolvimento. Seu principal objetivo é revelar a presença de erros o mais cedo possível no desenvolvimento de software, buscando minimizar o custo da correção dos mesmos [Pressman 2011]**.** Muitos recursos são gastos durante o desenvolvimento de um software, e em alguns casos, a qualidade do software não agrada os clientes pelo não atendimento dos requisitos.

Controlar a qualidade de sistemas de software é um grande desafio devido à alta complexidade dos produtos e às inúmeras dificuldades relacionadas ao processo de desenvolvimento, que envolvem questões humanas e técnicas. A ideia é que os sistemas de software devem não só fazer corretamente o que o cliente precisa, mas também fazê-lo de forma segura, eficiente e de maneira a permitir uma fácil manutenção e evolução desses sistemas.

A abordagem manual de testes leva à ocorrência de muitos problemas e em muitas situações deve ser evitada. Métodos ágeis de desenvolvimento de software e práticas recomendadas de programação sugerem o uso de testes automatizados, que são uma técnica voltada principalmente para a melhoria da qualidade dos sistemas de software [Rakitin 2001]**.**

A execução manual de um caso de teste pode ser rápida e efetiva, mas a execução e repetição de um vasto conjunto de testes manualmente é uma tarefa muito dispendiosa e cansativa. É possível que os testadores não verifiquem novamente todos os casos a cada mudança do código, o que pode fazer com que produtos de software possam chegar aos usuários com erros.

A automação dos testes de software tem sido vista como a principal medida para melhorar a eficiência dessa atividade, e várias soluções têm sido propostas para esta finalidade. A automação do teste consiste em repassar para o computador tarefas de teste de software que seriam realizadas manualmente, sendo feitas geralmente por meio do uso de ferramentas de automação de teste [Molinari 2008]. Quando executada corretamente, a automação de teste é uma das melhores formas de reduzir o tempo de teste no ciclo de vida do software, diminuindo o custo e aumentando a produtividade do desenvolvimento de software como um todo, além de, consequentemente, aumentar a qualidade do produto final.

A grande vantagem dos testes automatizados é que podem ser repetidos a qualquer momento, seja ao acrescentar uma nova funcionalidade ou ao realizar alguma modificação em uma funcionalidade específica. Sua implementação reduz os esforços e o tempo gasto na execução dos testes, garantindo que não ocorrerá falha humana nessa execução [Bernardo;Kon 2008].

Além disso, como os casos para verificação são descritos através de um código interpretado por um computador, é possível criar situações de testes bem mais elaboradas e complexas do que as realizadas manualmente, possibilitando qualquer combinação de comandos e operações. A magnitude dos testes pode também facilmente ser alterada. Por exemplo, simular vários usuários acessando um sistema ou inserir milhares de registros em uma base de dados, o que seria extremamente dispendioso através de testes manuais. Todas estas características ajudam a solucionar os problemas encontrados nos testes manuais, diminuindo a quantidade de erros e aumentando a qualidade do software. Como é relativamente fácil executar todos os testes a qualquer momento, mudanças no sistema podem ser feitas com segurança, o que aumenta a vida útil do produto.

Outros motivos para se utilizar a automação são os ciclos de feedback mais rápidos. Problemas de qualidade, configuração e implantação podem ser detectados antecipadamente

e tratados de modo a reduzir totalmente (ou quase) o impacto no usuário final. Uma das principais características dos testes automatizados é o determinismo. Com certeza, isso exige um cuidado por parte de quem automatiza. Pois, se os testes são aplicados em ambientes sem gerência de estado, essa característica se perde e as execuções podem se tornar instáveis [Azevedo Fraga 2018]. Contudo, assumindo um ambiente devidamente controlado e adequado, podemos dizer que os testes sempre são executados da mesma maneira. E isso os torna naturalmente confiáveis, ainda mais se comparados a procedimentos manuais.

É fato que o teste automatizado garante segurança e agilidade. Isso porque ele permite a execução e, principalmente, a repetição de procedimentos de teste em menos tempo, quando comparado ao teste manual de software. Dessa forma, ele otimiza o tempo e aumenta a velocidade e a confiabilidade na avaliação das aplicações.

Parece óbvio que o teste automático é melhor que o manual, entretanto, nem sempre se decide automatizar os testes pois não é um processo tão simples. Afinal, um dos principais desafios da automação de testes é o bom desenvolvimento, a manutenção e a atualização dos scripts de teste. Também vale lembrar que até mesmo a automação depende da interferência humana. Assim sendo, certamente testes automatizados não excluem a necessidade de um teste manual de software. Inclusive, a iniciativa de automação pode ser iniciada a partir de um processo de teste manual, estabelecido e maduro. Quando desatualizados ou incorretos, também podem falhar, perdendo a eficácia.

É preciso destacar que os testes, sejam eles do tipo manual ou automatizado, são essenciais e devem compor o planejamento do projeto, passando por todas as etapas até a finalização e a entrega total do software. Essa prática tem como intuito a descoberta de defeitos já nas fases iniciais do ciclo de desenvolvimento, tornando mais fácil a correção. Então ao comparar se é melhor automatizar ou não os testes, é importante saber que essa resposta depende muito do contexto de desenvolvimento analisado.

Este trabalho consiste em apresentar um relato de experiência visando comparar o uso da automação de testes para realizar testes funcionais na interface de um sistema Web utilizado para alfabetização, com testes feitos manualmente apresentando vantagens e dificuldades do processo de automação de forma a guiar projetos que optem pela automatização de seus testes e incentivar a automatização, principalmente entre alunos de graduação. O framework escolhido para realização da automação de testes do presente estudo foi o Selenium WebDriver em Java, o qual vem se apresentado como uma ferramenta bastante utilizada no mercado e que tem sido considerada importante para execução de testes automatizados [Devmedia 2017]. O sistema escolhido para ser testado foi o SisAlfa, que é um *Web Service* que pode ser utilizado para apoiar o desenvolvimento de diferentes jogos e exercícios digitais para alfabetização.

O objetivo geral deste trabalho é aplicar o framework Selenium sobre um serviço Web colaborativo chamado SisAlfa que é utilizado para apoiar a alfabetização, apresentando as vantagens e dificuldades do processo de automação em comparação com o uso de testes manuais de forma a orientar estudantes de graduação de cursos de TI iniciantes na área de qualidade de software no processo de automação compartilhando aspectos observados nessa experiência de modo a incentivar e dar caminhos para a automatização em diferentes projetos.

Para atingir o objetivo geral do trabalho, foram definidos os seguintes objetivos específicos:

- A criação da modelagem dos casos testes e execução de testes manuais e automáticos para o serviço SisAlfa, que não apresentava documentação de

seus requisitos, foi feito uma engenharia reversa para criação dos cenários de testes.

- Identificação de vantagens da automatização e dificuldades desse processo no sistema SisAlfa.

2 Referencial Teórico

Com a crescente expansão da utilização de sistemas informatizados na maioria das atividades das pessoas, a qualidade não está sendo considerada mais diferencial dos produtos e vem se tornado uma exigência. Constituintes do processo de validação de software e parte fundamental de qualquer processo de qualidade de software, os testes devem ser realizados para que se obtenha um produto confiável.

Conforme Pressman (2011) define, Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente. Testes visam mostrar que um programa faz o que se propõe a fazer e para descobrir os defeitos do programa antes de seu uso.

Seguindo essa mesma linha, Bartié (2002) define que: "Teste é um processo sistemático e planejado que tem por finalidade única a identificação de erros". Outro objetivo dos testes de software é o de verificar alguns atributos do sistema, como, por exemplo, a sua confiabilidade. Pode-se ainda dizer que o teste de software tem três propósitos: verificar, validar e localizar falhas na aplicação [Bentley; Bank 2005].

Repetir um trabalho várias vezes, como escrever a mesma entrada de dados diversas vezes, é muito entediante quando feito manualmente e pode levar pessoas a cometerem alguns erros. Além do fato de que ao executar testes manualmente, tende-se a repetir os mesmos testes ocasionando o que chamam de 'Paradoxo do Pesticida', onde testes executados repetidamente irão localizar cada vez menos problemas. A documentação necessária cresce em proporção ao número de testes realizados. Sendo assim, quanto mais testes, maiores serão os gastos [Hendrickson 2006].

Com testes de software automatizados, profissionais criam cenários de teste, desenvolvem códigos que executam estes testes e fazem a análise, se preocupando somente com a manutenção dos mesmos. Nos testes manuais, os profissionais executam toda a tarefa. Segundo Bartié (2002), "os testes automatizados são definidos como a utilização de ferramentas de testes que possibilitem simular usuários ou atividades humanas de forma a não requerer procedimentos manuais no processo de execução dos testes".

Neste contexto, testes manuais de software se mostram muito exaustivos. Isso porque são necessários muitos testes de regressão, que têm papel crucial na garantia da qualidade do produto final, pois eles são realizados para identificar os efeitos negativos de mudanças feitas no software, envolvendo desde a criação de novos defeitos até redução da confiabilidade, desempenho ou funcionalidade do produto [Pressman 2005]. Os testes automatizados geralmente demandam conhecimento de programação para serem implementados e utilizados. Como a interação humana é muito pequena, geralmente é possível executar uma suíte de testes automatizados muitas vezes (uma suíte de testes é uma coleção de casos de teste destinados a testar um programa para verificar um determinado comportamento). Além disso, a execução dos testes é pouco sujeita à falha humana já que a maior parte da atividade é executada por computadores. Os testes automatizados podem simular situações extremamente complexas por serem executados por testes automatizados.

Molinari (2003) cita que a aplicação de testes automatizados “permitirá aumentar a profundidade e abrangência dos casos de testes envolvidos” em que profissionais criam cenários de teste, desenvolvem códigos que executam estes testes e fazem a análise, se preocupando somente com a supervisão dos mesmos, ao contrário dos testes manuais, onde profissionais executam toda a tarefa.

Uma ferramenta bastante utilizada para automação de testes é o Selenium Webdriver, é um framework open source (código aberto), que é utilizado na automatização de testes funcionais em aplicações web [selenium 2017], e que foi desenvolvido por Jason Huggins em 2004. Heinen (2014) descreve o Selenium WebDriver como sendo um conjunto de bibliotecas que são utilizadas em várias linguagens de programação e que servem para integrar os scripts criados com a tela do sistema. Heinen (2014) também afirma que, diferente do IDE, o WebDriver possibilita a interação com banco de dados, criar *Page Objects* (elementos de um padrão de projeto que auxilia na organização de testes funcionais) nos permite criar um repositório de objetos com elementos contidos numa página Web. Sob este modelo, para cada página, deve haver uma classe correspondente, abstraindo uma página HTML (ou parte) em uma API específica da aplicação, permitindo você manipular os elementos de página [Nelson Sousa 2014].

3 Trabalhos Relacionados

Nesta seção é realizada a análise dos trabalhos relacionados com base em buscas nas bibliotecas digitais da ACM e IEEE usando como chaves de busca "*Software testing*", "Selenium" "*API Testing*", comentando suas similaridades com este trabalho e diferenças.

Romani (2018) aborda por meio de vivências e pesquisas uma introdução para aplicação de automação de testes utilizando o framework Selenium WebDriver e o TestNG. Com a evolução dos modelos de gerenciamento de projetos, as práticas de métodos ágeis (Scrum) com seus ciclos curtos de desenvolvimento e seus times enxutos, impactam significativamente no modo como os testes de software hoje são feitos [Romani 2018]. Por exemplo, executar um teste de regressão após uma mudança no código de maneira manual acaba sendo oneroso, gastando muito tempo do analista de teste e gerando um custo a mais para a empresa. Com isso, houve um maior incentivo pelo uso dos testes automatizados. No final, os profissionais interessados na área de teste terão uma base sobre como aplicar as ferramentas para aumentar a confiabilidade e ganharão agilidade na execução de casos de teste.

Parejo, Troya e Ruiz-Cortés (2016) relatam como as APIs Web estão progressivamente se tornando a base da integração de software, e como sua validação está ficando cada vez mais crítica. Dentro deste cenário, a rápida localização de bugs é de extrema importância para aumentar a qualidade de produtos internos e aplicativos de terceiros. Os autores dão uma alternativa quando a saída esperada de uma execução de teste é complexa ou desconhecida. Em vez de verificar uma única saída, o teste verifica se múltiplas execuções do programa cumprem certas propriedades.

Esse trabalho se inspira no trabalho de Romani (2018) e busca apresentar os benefícios encontrados com a automação de testes, mas também trazendo as possíveis dificuldades encontradas com o uso dessa prática. Também é relacionado ao de Parejo, Troya e Ruiz-Cortés (2016), por envolver o teste de uma API WEB, como é o caso do SisAlfa.

4. Metodologia

Para o presente trabalho será utilizado um estudo de caso, que consiste no estudo profundo e exaustivo de um ou poucos objetos, de maneira que permita seu amplo e detalhado conhecimento [Gil 2002]. No entanto, o estudo apresentado aqui é apenas um estudo inicial e exploratório e que terá como foco relatar as principais experiências obtidas. Para estudar as vantagens e dificuldades de utilização do Selenium foi realizada a aplicação da automação dos testes no serviço SisAlfa. Como parte deste estudo de caso inicial, serão apresentadas as técnicas utilizadas para realização dos testes.

Inicialmente foram planejados os casos de testes a utilizar na avaliação e que estão documentados na modelagem dos testes apresentada no Apêndice A. Umas das técnicas escolhidas para criação dos cenários de teste foi a técnica de partição de equivalência, que consiste em identificar as entradas e então dividi-las em possíveis valores. Outra técnica utilizada foi a de análise do valor limite, que é usada para validar se os limites de uma determinada partição foram atendidos da forma esperada.

Os casos de testes criados no plano foram executados de duas formas: inicialmente os testes foram realizados manualmente, e seu tempo de execução foi cronometrado. Em um segundo momento foram desenvolvidos *scripts* de teste para os mesmos cenários descritos no plano de testes manual e foram medidos os tempos de execução, além de observar vantagens e desafios ao realizar a automatização e exemplos que podem ser úteis principalmente para alunos de graduação iniciantes na área de qualidade de software.

5. O Sistema SisAlfa

O SisAlfa é um *Web Service* que pode ser utilizado para apoiar o desenvolvimento de diferentes jogos e exercícios digitais para alfabetização. Ele permite a construção colaborativa de uma base de dados multimídia de palavras agrupadas em contextos e às quais estão associados vídeos, áudios e imagens [Matheus Nicolas da Silva; Robson Alves 2017].

O sistema em sua versão atual apresenta uma interface Web que acessa sua API REST que permite o cadastro e a consulta de desafios e contextos que serão utilizados[2]. O serviço Web permite a construção de uma base de dados colaborativa dinâmica, fornece e adquire desafios (ou palavras) com base em contextos (ou temas). Os desafios armazenados podem estar relacionados a um ou mais contextos, e para cada contexto ou desafio pode estar associada a URL de um vídeo, som e imagem relacionado, para tornar a experiência mais lúdica e interessante ao usuário, conforme ilustra a Figura 3.

Os testes automáticos e manuais foram feitos com foco nas funcionalidades de Cadastro de Usuário, Login, Contexto e Desafio. As Figuras 1, 2 e 3, apresentadas a seguir, ilustram as telas que serão utilizadas para realização dos testes.

[2] http://github.com/MatheusCrispim/literacy_api

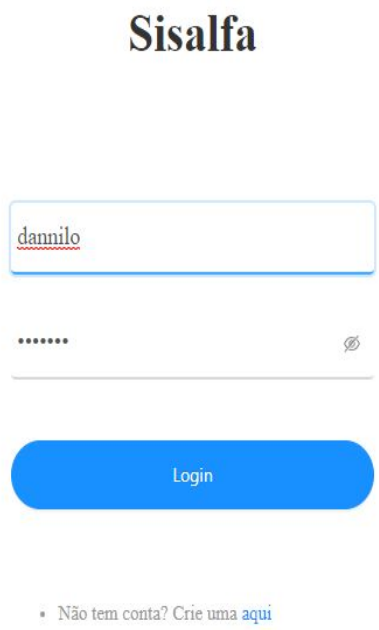

Figura 1 - Tela Login

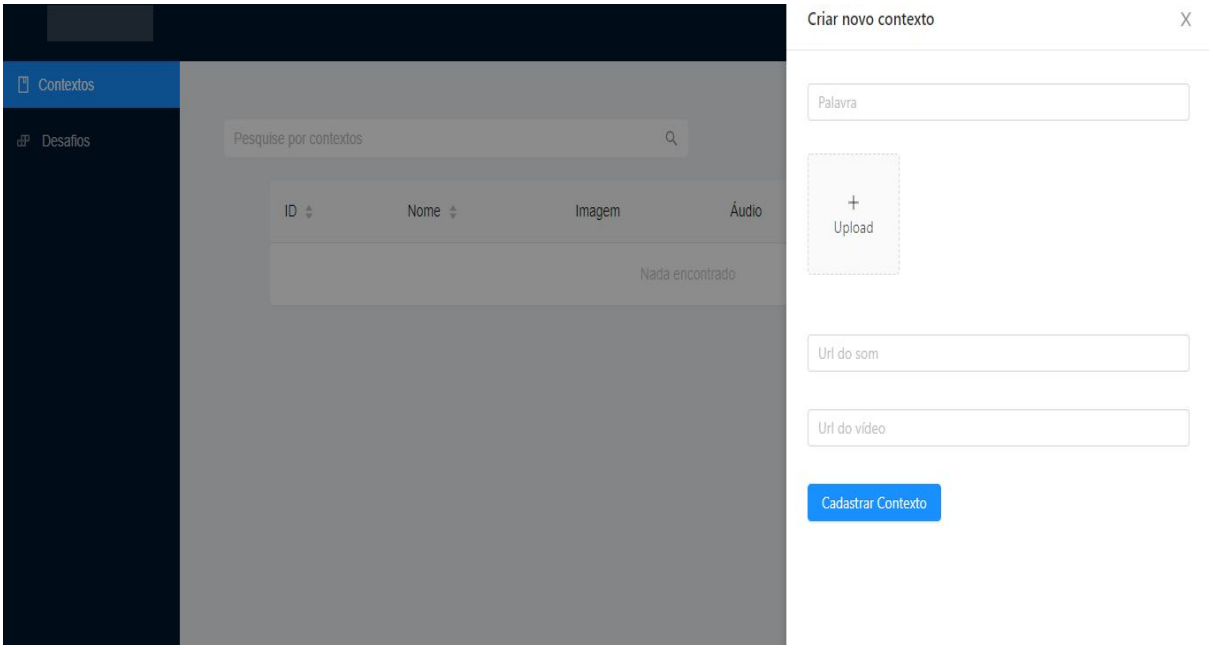

Figura 2 - Tela Criação Contexto

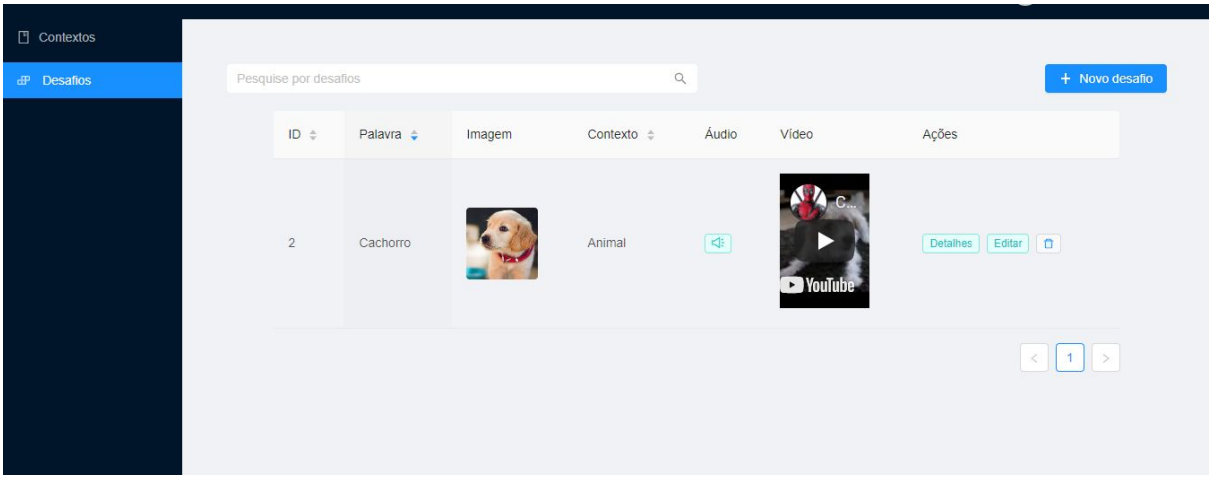

Figura 3 - Tela de Desafios

6 Resultados

Esta seção apresenta os resultados obtidos com a implementação e execução dos testes automatizados sobre o software SisAlfa.

6.1 Modelagem dos Testes

O documento com a modelagem dos testes contém basicamente todos os cenários de testes que serão feitos com seu passo a passo. Como o sistema não tinha um documento com requisitos, foram feitos inicialmente testes exploratórios para a documentação dos casos de teste. Teste exploratório é uma atividade iterativa e empírica de exploração que exige idas e vindas num processo de investigação contínuo onde a intuição, a criatividade e a experiência do testador são indispensáveis para garantir a eficiência do teste [Lazaro Santos 2018]. Os casos de teste se encontram junto com os *scripts* de teste no github[3] e no Apêndice A. Um exemplo de um caso de teste usado é este que está ilustrado pela Figura 4:

[3] https://github.com/wellysondannilo/tcc

	Funcionalidade/ Componente	Titulo	Passo-a-passo para execução	Resultado esperado	Status
2	Cadastro de Contexto	Cadastro de Contexto Com campo Url Som vazio	1- Efetuar login; 2- Na home do usuário, clicar no menu "Contexto"-> Novo Contexto"; 3- Preencher dados válidos para a novo Conxtexto, mas deixar o campo "Url Som" com o caractere espaço (barra de espaço).	1- Usuário logar com sucesso e visualizar a sua home. 2- Deve ser aberta uma pop-up contendo um formulário para cadastro de uma novo contexto; 3- O sistema não deve permitir o cadastro do contexto. Deve ser exibida uma mensagem de erro informando que este campo é obrigatório.	NOK

Figura 4 - Exemplo de um cenário de testes.

6.2 Criação dos Scripts de Teste

Para criação dos scripts foi utilizado o Eclipse[4], o Selenium Webdriver[5], o JUnit[6] e o Google Chrome, que permite por meio do driver do Selenium a execução do sistema no Google Chrome de forma automatizada. Ao escrever o script no Eclipse, usamos algumas anotações sobre os métodos para que o JUnit os entenda como testes que devem ser executados. As anotações são um recurso usado para que marcações em classes, campos e métodos possam ser tratadas pelas bibliotecas, compilador e ferramentas de desenvolvimento em tempo de execução ou compilação. Os elementos são localizados através do identificador (ID) caso tenha sido declarado, pelo nome da classe, css ou por alguma tag. E caso não exista um identificador, o caminho para definição dos elementos é montado através de um xpath.

É utilizado o padrão *Page Object* para criação das *pages Web*. *Page Object* é um padrão de projeto com a proposta de criar objetos para cada página web. Ele permite criar repositórios de objetos com elementos da página *Web*. Assim, para cada página, deve haver uma classe correspondente [Medium 2017]. Um exemplo de uso do padrão é a classe CadastroPage ilustrada na Figura 5.

Os Scripts criados estão disponíveis no *Github*. Um destes scripts está ilustrado pela Figura 5. Esse script mostra os passos e dados que serão utilizados para um cadastro de usuário.

[4] https://www.eclipse.org/documentation/
[5] https://www.selenium.dev/documentation/en/
[6] https://junit.org/junit5/docs/current/user-guide/

LoginTeste.java | LoginPage.java | CadastroPag... | CadastroUsu... | BaseTest.java | MenuPag

```
16
17      public void setNome(String Nome) {
18          escrever(By.xpath("//input[@id='firstName']"), Nome);
19      }
20
21      public void setSobrenome(String Sobrenome) {
22          escrever(By.xpath("//input[@id='lastName']"), Sobrenome);
23      }
24
25      public void setUsername(String username) {
26          escrever(By.xpath("//input[@id='username']"),username);
27      }
28
29      public void setEmail(String Email) {
30          escrever(By.xpath("//input[@id='email']"),Email);
31      }
32
33      public void setSenha(String Senha) {
34          escrever(By.xpath("//input[@id='password']"),Senha);
35      }
36
37      public void setCadastrar() {
38          clicarBotao(By.xpath(".//button[@class='ant-btn form btnSignup ant-btn-primary']"));
39
40      }
41      public String obterMensagemErro() {
42          return obterTexto(By.xpath("//div[text()='Usuário já existente']"));
43
```

Figura 5: Exemplo de uma page.

6.3 Vantagens

Nessa seção serão relatadas as vantagens encontradas com a utilização do Selenium WebDriver, a partir dos testes realizados anteriormente. Tais resultados podem contribuir na tomada de decisão a respeito da realização ou não de automação de testes.

6.4. Reutilizar os Testes durante o desenvolvimento.

A maior vantagem encontrada com a utilização do Selenium WebDriver para a automação dos testes é que o usuário tem a possibilidade de reutilizar os testes desenvolvidos sempre que necessário, sendo uma grande vantagem para realização dos testes de regressão. Os testes de regressão são testes que visam testar, após a implementação de uma mudança no software, segmentos implementados que já foram testados anteriormente [Bastos 2012].

6.4.1 Vantagem no tempo de execução.

Também com base nos dados obtidos após a automação, como esperado, podemos afirmar que a automação de testes faz com que o teste a ser realizado seja executado mais rápido que uma pessoa executando manualmente. Neste estudo de caso foi executado este pequeno fluxo de teste para análise, mas levando em conta a automação de teste em um sistema de grande proporção, vê-se que a economia de tempo com seus testes de regressão, ou a localização de bugs, seria ainda maior. Tal economia pode ser observada a longo prazo, e beneficia a equipe como um todo. Em testes futuros será mais rápido e simples rodar o script de automação de teste, do que ter que entender o sistema para refazer todos os testes de forma manual.

6.4.2 Ciclos de feedback mais rápidos

A Automação de Testes, combinada com outras automações no processo de Desenvolvimento de Software, promove ciclos de *feedback* mais rápidos. Problemas de qualidade, configuração e implantação podem ser detectados antecipadamente e tratados de modo a reduzir totalmente (ou quase totalmente) o impacto no usuário final.

6.5 Dificuldades identificadas na automatização de testes com o Selenium Webdriver

Nessa seção serão relatadas as dificuldades e maneiras encontradas para contorná-las durante a utilização do Selenium WebDriver a partir dos testes realizados anteriormente.

6.5.1 Necessidade de conhecimentos em programação.

Uma das dificuldades encontradas durante a utilização do Selenium WebDriver foi a necessidade de um conhecimento de programação prévio. Ele utiliza a linguagem de programação Java, e há uma certa necessidade de adaptação dos comandos e objetos utilizados. Há também a necessidade da utilização de outros softwares, plugins e bibliotecas externas.

Para trabalhar com automação é necessário saber programar, porém não é necessário ser um expert em programação. Os requisitos mínimos para o profissional de automação é conhecer alguma linguagem de programação e ter o conhecimento de orientação a objetos, controle de condições, controle de loops e tratamento de exceções. Sendo assim, alguém que deseje trabalhar na área ou iniciar a automação de algum projeto deve fortalecer seus fundamentos nesses aspectos.

6.5.2 Elementos sem valores de identificação

Outro problema encontrado foi a necessidade de mapear uma série de elementos pertencentes à interface da aplicação. Em algumas aplicações Web, como foi o caso na aplicação testada neste trabalho, os elementos nem sempre possuem classes únicas ou valores de identificação. Sendo assim, é necessário um mapeamento manual para capturá-los e identificá-los unicamente, sendo necessário o uso de Xpath para localizar os elementos na tela.O Xpath é o caminho do elemento dentro do *HTML*

Mas caso possua conhecimento na estrutura de como é criada uma página *Web*, conhecimentos em *HTML e css,* provavelmente conseguirá mapear o caminho sem maiores dificuldades. Caso não tenha esses conhecimentos mínimos, é importante buscar fortalecê-los para utilizar automação de testes em seus projetos.

LoginTeste.java | LoginPage.java

```
7
8  public class LoginPage extends BasePage {
9
10
11     public void acessarTelaInicio() {
12         DriverFactory.getDriver().get("http://localhost:3000/login");
13         }
14
15     public void setUsername(String username) {
16         escrever(By.xpath("//input[@id='username']"),username);
17     }
18
19     public void setPassword(String password) {
20         escrever(By.xpath("//input[@id='password']"), password);
21     }
22
23     public void setEntrar() {
24         clicarBotao(By.xpath(".//button[@class='ant-btn loginBtn ant-btn-primary']"));
25
26     }
27
28     public String obterMensagemErro() {
29         return obterTexto(By.xpath("//div[text()='Credenciais do inválidas']"));
30
31     }
32
33     public String obterMensagemErroPasswordVazio() {
34         return obterTexto(By.xpath("//div[text()='Por favor, insira a senha!']"));
35
```

Figura 6: Exemplo do uso de xpath.

6.5.3 Necessidade de Mão de Obra Qualificada e Falta de Estudo antes de adotar qualquer ferramenta

Outro grande fator de dificuldade na automatização dos testes é a necessidade de qualificação das equipes, que devem ter um grande grau de conhecimento das ferramentas e do sistema ou aplicação. Para a melhor construção e utilização da automatização, diversas vezes pode ser difícil encontrar profissionais qualificados e que entendam da aplicação ou sistema. Às vezes a pessoa conhece o sistema ou aplicação, mas não conhece o processo de automatização ou vice versa, sendo assim necessário treinamento para uma melhor qualidade e benefícios futuros. No caso do estudo aqui apresentado com o SisAlfa, foi necessário conhecer bem o

sistema sendo testado antes de iniciar a automação dos testes, principalmente levando em consideração que seus requisitos não estavam documentados.

Muitos acreditam que apenas adquirir uma ferramenta de automação é suficiente para resolver os problemas no processo de desenvolvimento. Mas não é bem assim. Embora ferramentas sejam imprescindíveis para a automação, é indispensável realizar análises e estudos prévios antes de adotá-las para verificar qual se adapta melhor e atende às necessidades do projeto. Além disso, é preciso considerar também os custos de aquisição, infraestrutura necessária para instalação, equipe e capacitação necessárias para utilizá-las. Para esse estudo em particular, foram analisadas diferentes ferramentas e escolheu-se o Selenium. A ideia inicial era usar RestAssured que é um framework que permite testar serviços RESTfull de um jeito prático sendo possível criar chamadas HTTP, como se estivesse acessando a API. Suporta os métodos POST, GET, PUT, DELETE, OPTIONS, PATCH e HEAD e pode ser usado para validar e verificar a resposta dessas solicitações[Rafael Amaral 2019], mas a Api do sistema possuía alguns problemas, com isso foi decidido fazer os testes na interface Web.

7 Lições Aprendidas

Com a experiência prática da Automação de Testes, foi possível observar, que é necessário um estudo prévio antes da adoção de uma dessas práticas para escolha adequada do tipo de teste e ferramentas a serem utilizadas e se realmente é vantajosa e necessária a automação.

Para configuração do ambiente e criação dos *scripts* neste estudo tudo funcionou bem, e sem grandes problemas tendo em vista a experiência e conhecimento prévio do autor deste trabalho nas tecnologias.

No caso do sistema SisAlfa, pelo fato de ter poucas funcionalidades e não ser complexo, talvez não sejam perceptíveis grandes ganhos com a automação. Mas um ganho a ser observado é o reaproveitamento das *pages* para as outras versões posteriores do projeto por conterem elementos idênticos e o fato de dar maior garantia de controle de qualidade caso muitas mudanças venham a acontecer em seu código.

Um comparativo que ajuda enxergar diferenças entre o teste manual de software e o teste automatizado de software pode ser visto no Quadro, produzido a partir da experiência com o estudo aqui apresentado e com base em um treinamento disponibilizado na empresa Indra Company.

Dimensão	**Testes Manuais**	**Testes Automatizados**
Confiabilidade	Propenso a erros humanos	Menos suscetíveis a erros, tornando-o mais confiável
Custo	Depende do perfil dos testadores do time.	Custo mais alto, visto que dependendo da ferramenta será necessário adquirir licenças ou ter um hardware específico.
Tempo de Execução	Varia de profissional para profissional. Além do teste não ser reaproveitado	Menos tempo para execução. São reutilizáveis e podem ser executados em

		diferentes máquinas.
Usabilidade	Facilmente validado e verificado pelo testador.	Não permite julgar estética ou usabilidade da aplicação.
Colaboração	Somente o testador tem visibilidade do progresso e resultados do seu teste.	Outros testadores podem acessar e visualizar os progressos e resultados do teste.
Indicado Para	Testes exploratórios, usabilidade.	Testes de regressão, carga e performance.

Fonte Quadro 1: Treinamento Indra

Mesmo que vários dos aspectos apontados na tabela estejam disponíveis na literatura, o presente estudo visa auxiliar desenvolvedores iniciantes, principalmente alunos de graduação, a adotarem a automação de testes. Mesmo com tais vantagens, em algumas empresas se percebe um certo nível de resistência para a automação de testes, principalmente devido ao esforço inicial do processo e por demandar às vezes uma mudança de cultura na organização. [Lima, Dantas e Vasconcelos 2012]. Nesse sentido, foram identificadas as seguintes lições ao realizar a automação de testes do SisAlfa:

- É necessário saber programar, porém não é necessário ser um expert em programação. Os requisitos mínimos para o profissional de automação é conhecer alguma linguagem de programação e ter o conhecimento de orientação a objetos e de *HTML e css;*
- Se não há requisitos, planeje os testes lembrando que o documento mais atualizado é sempre o próprio sistema sendo testado, embora se deva lembrar que ele pode ter defeitos;
- Para contornar a dificuldade de elementos sem valores de identificação, busque conhecimento sobre a estrutura de como é criada uma página *Web* e com conhecimentos em *HTML e css* provavelmente conseguirá mapear o caminho sem maiores dificuldades;
- Antes de adotar qualquer ferramenta para automação de testes é necessário ter um conhecimento prévio das ferramentas e do sistema a ser testado e é indispensável realizar análises e estudos prévios antes de adotá-las para verificar qual se adapta melhor e atende às necessidades do projeto.

8 Conclusão e Trabalhos futuros

A automação de testes pode garantir uma maior agilidade na execução de testes em grande escala, principalmente para testes que são repetitivos e testes que devem ser executados com uma certa frequência. Um ponto a ser ressaltado é que em etapas posteriores do desenvolvimento do projeto, os custos com as falhas e defeitos se tornam mais caros para serem resolvidos, pois envolvem correções de várias etapas do projeto.

É promissora a utilização da automação para realizar o teste de regressão, onde se tem a necessidade de realizar toda a bateria de testes a cada nova alteração no sistema.

O custo da criação dos *scripts* de teste para a execução automatizada geralmente é maior do que o custo dos testes realizados de forma manual, justamente por causa do tempo gasto com a implementação de tais scripts. Por este motivo, é necessário que este custo adicional da implementação seja compensado com os benefícios gerados pela automatização.

Testes automatizados não excluem a necessidade de um teste manual de software, onde questões como usabilidade podem ser verificadas.

De forma geral, é recomendada a utilização da ferramenta Selenium WebDriver para a automatização de tarefas repetitivas, projetos longos e que são atualizados com frequência. Nessas situações, a realização dos testes de forma manual se torna uma tarefa cansativa, onde para cada alteração do sistema se faz necessária a execução de todos os testes já realizados.

Um estudo é o que deve ser feito antes de se escolher automatizar ou não, para saber se a automação irá valer a pena, avaliando se quem irá trabalhar está preparado e caso não esteja, considerar o custo e tempo gastos para aprenderem ferramentas e processos de testes, além de fortalecerem sua base em aspectos básicos de programação.

Acreditamos que este trabalho traz como contribuição a partilha de experiência com um estudo de caso inicial, mas são necessários trabalhos futuros visando investigar a importância da automação dos testes do SisAlfa para a evolução do projeto e também observar junto a alunos de graduação se as lições aqui compartilhadas realmente são úteis para a formação de alunos de graduação em cursos de TI.

Referências

GAEA. **Os Desafios da Automatização de testes.** Disponível em: <https://conteudo.gaea.com.br/desafios-da-automatizacao-de-testes>. Acesso em: 04 jun. 2019b.

SILVA, P. C. B; ALVES, T. S; BRUNO, E. A. **Automação de testes funcionais: testes funcionais automatizados de software**. Revista de Ciências Exatas e Tecnologia, v. 6, n. 6, p. 113-133, 2011.

BARTIÉ, Alexandre. **Garantia da qualidade de software: adquirindo maturidade organizacional.** p. 49-50, 1 set. 2002.

BERNARDO, P. C; KON, F. **A Importância dos Testes Automatizados. Artigo publicado na Engenharia de Software Magazine**. 2008. Disponível em: https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf. Acesso em: 13. Nov. 2019.

S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés. 2017. **Metamorphic Testing of RESTful Web APIs.** ***IEEE Transactions on Software Engineering*** **PP**, 99 (2017), 1--1

S. K. Chakrabarti and P. Kumar, "**Test-the-REST: An Approach to Testing RESTful Web-Services," 2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, Athens**, 2009, pp. 302-308, doi: 10.1109/ComputationWorld.2009.116.

RODRIGUEZ, Alex. **Restful web services: The basics. IBM Developer Works**, v. 33, p. 18, 2008.

Pressman, Roger S. (2011), E**ngenharia de software: Uma abordagem profissional**, AMGH, 7ª edição.

YACKEL, Ryan. **A automação de testes de software atinge a maturidade.** Administradores, 16 agosto 2018. Disponível em: . Acesso em: 04 jun. 2019

ROMANINI, 112I ,**SELENIUM WEB DRIVER NA EVOLUÇÃO DOS TESTES MANUAIS**. 2018. Disponíve:https://revista.fatectq.edu.br/index.php/interfacetecnologica/article/view/627/416. Acesso em: 08 jun. 2018.

DEV MEDIA. I**ntrodução aos testes funcionais automatizados com JUnit e Selenium WebDriver**. Disponível em: <https://www.devmedia.com.br/introducao-aos-testes-funcionais-automatizados-com-junit-e-selenium-webdriver/28037>. Acesso em: 25 abr. 2019

SILVA, Mateus Nicolas da., ALVES Robson. e REBOUÇAS, Ayla Dantas. 2017. **SisAlfa: Um Serviço Colaborativo para apoiar a criação de Sistemas para Alfabetização**. em: VI Congresso Brasileiro de Informática na Educação (CBIE).

Apêndice A:

Tìtulo	**Passo-a-passo para execução**	**Resultado esperado**
Efetuar Login Sucesso	1-O site estar aberto e possuir credenciais válidas(Login:admin, Password:admin	Usuário logar com sucesso e visualizar a sua home.
Efetuar Login Sem Cadastro no sistema	1-O site deve estar aberto e deve possuir um usuário válido. 2-Inserir o Login Corretamente e a Senha Inválida.	'Login não é efetuado e é exibida uma mensagem informando que os dados são Inválidos.
Efetuar Login Com Username valido e password invalido.	1-O Site estar aberto na página de login . 2- Inserir dados para login que não estejam cadastrados no sistema.	Login não é efetuado e é exibida uma mensagem informando que os dados informados são inválidas
Efetuar login Informando apenas o Username.	1- O Site está aberto na página de login. 2- Preencher o campo usuário e deixar o campo senha vazio	Login não é efetuado e é exibida uma mensagem informando que o campo password deve ser preenchido
Efetuar login Informando apenas o Password.	1- O Site está aberto na página de login. 2- Preencher o campo senha e deixar o campo usuário vazio	Login não é efetuado e é exibida uma mensagem informando que o campo username é obrigatório.
Efetuar login com campos Username e Password vazios	1- O Site está aberto na página de login. 2- Não preencher os campos username e password	Login não é efetuado e é exibida uma mensagem informando que os Campos Username e password são obrigatórios.
Efetuar login com o campo Username Maiúsculo	1- O Site está aberto na página de login. 2- Preencher o campo username com o caracteres maiúsculos	Login não é efetuado e é exibida uma mensagem informando que as credenciais são inválidas
Efetuar login com o campo Password Maiúsculo	1- O Site está aberto na página de login. 2- Preencher o campo	1- O sistema não deve permitir o login do usuário. 2- Deve ser exibida uma

	password com o caracteres maiúsculos	Mensagem informando que os dados são inválidos.
Cadastro Usuário Sucesso	1- O Site estar aberto na página De Cadastro de Usuario. 2- Preencher todos os campos com dados válidos	O usuário será cadastrado com sucesso.
Cadastro Usuario Invalido	1- O Site está aberto na página De Cadastro de Usuário. 2- Preencher o campo username com caracteres especiais e o resto corretamente.	Deve exibir uma mensagem informando que os caracteres são inválidos.
Cadastro Campos Vazio	1- O Site está aberto na página De Cadastro de Usuário. 2- Preencher o campo username com caracteres especiais e o resto corretamente.	Deve exibir uma mensagem informando que os campos são obrigatórios.
Cadastro Existente	1- O Site está aberto na página De Cadastro de Usuário. 2- Preencher os campos com dados já existentes.	Deve exibir uma mensagem informando que o usuário já existe.
Cadastrar contexto Sucesso	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Contexto>Novo Contexto>Incluir e preencher Todos os campos com dados válidos.	Deve exibir uma mensagem informando que o tamanho da imagem não é permitido
Cadastrar contexto Com uma foto de tamanho maior que o permitido	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Contexto>Novo Contexto>Incluir e preencher Todos os campos com dados	Deve exibir uma mensagem informando que os campos são obrigatórios

	válidos mas não inserir uma foto .	
Cadastrar contexto Sem Enviar uma Foto	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Contexto>Novo Contexto>Incluir e preencher Todos os campos com dados válidos mas enviar uma URL que não seja de um vídeo .	Deve exibir uma mensagem informando que os campos URL vídeo está incorreto.
Cadastrar contexto Enviando uma url que não seja de um vídeo	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Contexto>Novo Contexto>Incluir e preencher Todos os campos com "barra Espaço".	Deve exibir uma mensagem informando que os campos são obrigatórios
Cadastrar Contexto Sem preencher nem um campo	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Desafio>Novo Desafio>Incluir e preencher Todos os campos com dados válidos.	Deve exibir uma mensagem informando que o cliente foi cadastrado com sucesso
Cadastrar desafio com sucesso	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Desafio>Novo Desafio>Incluir e preencher Todos os campos com dados válidos mas não preencher o campo contexto.	Deve exibir uma mensagem informando que o campo contexto é obrigatório.
Cadastrar desafio sem escolher um contexto	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Desafio>Novo	Deve exibir uma mensagem informando que os campos são obrigatórios

	Desafio>Incluir e preencher Todos os campos com dados válidos mas não preencher os campos de URL de som e vídeo.	
Cadastrar desafios Sem Enviar uma url de som e vídeo	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Desafio>Editar Desafio>tentar editar os campos para vazio Utilizando "Barra Espaço".	Deve exibir uma mensagem informando que a edição não é possível e os campos são obrigatórios
Preencher a edição do desafio com campos em branco	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Desafio>Editar Desafio>Editar todos os campos mas deixar o campo URL Áudio vazio.	Deve exibir uma mensagem informando que a edição não é possível e os campos são obrigatórios
Editar Desafio deixando o campo áudio em branco	1- O Site está aberto na página de login. 2- Preencher os campos username e senha válidos. 3- Navegar até Desafio>Editar Desafio>Editar todos os campos mas deixar o campo URL Áudio vazio.	Deve exibir uma mensagem informando que a edição não é possível e os campos são obrigatórios