EnGeTec
Encontro de gestão e tecnologia

TÉCNICAS DE WEB SCRAP NA AUTOMAÇÃO DE TESTES PARA IDENTIFICAÇÃO DE ELEMENTOS EM SISTEMAS WEB LEGADOS OU COM AUSÊNCIA DE PADRONIZAÇÃO

Rafael Santana de Almeida[1], Ana Carolina Vieira Batista[1], Lucas Silva do Santos[1], Paola Montini Marinelli[2] Edson Melo de Souza[2]*

[1] Universidade Nove de Julho, Avenida Doutor Adolpho Pinto, 109 – CEP 01156-050
São Paulo – SP – Brasil

2 Universidade Nove de Julho (PPGI), Rua Vergueiro, 235/249 - CEP 01504-001
Liberdade – São Paulo/SP – Brasil
prof.edson.melo@gmail.com * autor correspondente.

RESUMO. Este trabalho tem como objetivo apresentar um sistema de *scripts* para realizar a análise de páginas HTML a fim de identificar características em sistemas legados ou sem padronização, localizando elementos como botões, caixas de texto e marcadores de seleção, por meio da utilização de técnicas de raspagem de dados (*webscrap*). Tal abordagem visa promover a implementação da qualidade de *software* aos sistemas dessa natureza, mediante testes de *software*. A solução foi desenvolvida utilizando-se a linguagem Python com módulos auxiliares para tratamento de elementos HTML, expressões regulares, entre outros. Foram realizados 150 testes em três *websites* distintos, sendo o Mercado Livre e OLX com tecnologias modernas de construção e o da SPTrans com sistema legado e sem especificações do W3C (*World Wide Web Consortium*), para os quais foram obtidos índices de sucesso de 98,56%, 99,00% e 94,00% respectivamente na realização dos testes. Os resultados mostraram que a aplicação desenvolvida se mostrou eficiente para detecção dos padrões e identificação de elementos, fornecendo condições de avaliar as características quanto a arquitetura e tecnologias utilizadas na construção dos sites. Os aspectos observados nos resultados dos testes permitem concluir que a solução e as técnicas de *webscrap* aplicadas cumpriram o propósito de se obter automação de testes em sistemas com histórico de desenvolvimento legado ou que foram criados sem especificações.

Palavras-chave. *Automação de testes. Qualidade de software. Sistemas legados. Sistemas web.*

ABSTRACT. This work aims to present a script system to perform the analysis of HTML pages in order to identify features in legacy or non-standard systems, locating elements such as buttons, text boxes and selection markers, using scraping techniques, known as webscrap. Such an approach aims to promote the implementation of software quality to such systems through software testing. The solution was developed using Python language with auxiliary modules for handling HTML elements, regular expressions, among others. Were performed 150 tests on three different websites, being Mercado Livre and OLX with modern building technologies and SPTrans with legacy system and without specifications of the World Wide Web Consortium (W3C), for which success rates of 98, 56%, 99.00% and 94.00% respectively in the tests. The results showed that the developed application proved to be efficient for pattern detection and element identification, providing conditions to evaluate the characteristics regarding the architecture and technologies used in the construction of the sites. The aspects observed in the test results allow us to conclude that the solution and applied webscrap techniques fulfilled the purpose of obtaining test automation on systems with legacy development history or that were created without specifications.

Keywords. *Test Automation. Software quality. Legacy systems. Web systems.*

1. INTRODUÇÃO

Com o avanço contínuo das novas tecnologias para internet, surge a necessidade de garantir que os sistemas desenvolvidos para esta plataforma funcionem em conformidade com o que foi projetado. De acordo com o tipo de aplicação, o desenvolvimento das soluções demanda grande tempo de especificação, codificação e testes, objetivando agregar valor ao negócio. Neste cenário, a realização dos testes automatizados surgiu como uma ferramenta auxiliar no processo, tornando-se rapidamente imprescindível no processo para oferecer um produto de *software* com qualidade.

Testes automatizados são recursos computacionais realizados por meio de *scripts* (segmentos de códigos com fins específicos) que executam funcionalidades de forma automática. Sua execução possibilita localizar erros em uma aplicação com maior acurácia, possibilitando a realização de correções com rapidez, além de minimizar custos de desenvolvimento e melhorarem a qualidade dos sistemas desenvolvidos.

A criação dos *scripts* para realização de testes depende do tipo de sistema a ser avaliado e, uma vez desenvolvidos, é possível reutiliza-los em novos testes a cada mudança de código do programa em questão, sem a necessidade reescreve-los. Esta característica permite reduzir tempo e custos com mão de obra para realização recorrente de novos testes.

Devido ao crescente surgimento de novas tecnologias, muitos sistemas tornam-se legados rapidamente, além de problemas com especificações que impactam, principalmente, no momento da realização de integrações e melhoria do desempenho. Portanto, a realização de testes automatizados é uma solução para minimizar tais problemas, possibilitando aos desenvolvedores aprimorar os novos sistemas, bem como recuperar informações de sistemas legados.

1.1 APRESENTAÇÃO DO POBLEMA

Considerando-se o contexto apresentado, elaborou-se a seguinte questão de pesquisa: A identificação de objetos não nomeados no DOM (*Document Object Model*), do HTML, pode ser solucionada com a aplicação de técnicas de Webscrap (raspagem de dados) para facilitar o processo de realização de testes em sistemas legados ou com ausência de padronização?

Para responder a esta questão de pesquisa, este trabalho tem como objetivo apresentar um sistema de scripts para realizar a análise de páginas HTML (*Hypertext Markup Language*, ou em português, Linguagem de Marcação de Hipertexto), a fim de identificar características de elementos como botões, caixas de texto e marcadores de seleção, por meio da utilização de técnicas de raspagem de dados.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 QUALIDADE DE SOFTWARE

O papel dos *softwares* no processo de produção da informação é receber, processar e disseminar

informações importantes para o gerenciamento e controle das atividades de uma organização. Um dos principais objetivos de um sistema de informação é auxiliar seus usuários em atividades, processos e tomada de decisão. No contexto de desenvolvimento de *software*, qualidade pode ser entendida como um conjunto de características a serem satisfeitas em um determinado grau, de modo que o produto de *software* atenda às necessidades explícitas e implícitas de seus usuários (ENTRINGER; DA SILVA FERREIRA, 2019).

O *software* é feito com base na padronização do desenvolvimento e processos, a fim de garantir maior qualidade dos sistemas. Os modelos aplicados na garantia da qualidade atuam principalmente no processo, mas o objetivo principal é garantir um produto que atenda às expectativas do cliente (GUERRA; COLOMBO, 2009). Diversos modelos de qualidade de *software* vêm sendo propostos ao longo dos últimos anos. Esses modelos têm sido fortemente adotados por organizações em todo o mundo (TONINI; CARVALHO; SPINOLA, 2008).

FIGURA 1: NORMAS ISO PARA VOLTADAS PARA QUALIDADE DE SOFTWARE.

Nome	Descrição
ABNT NBR ISO 9001:2015	Esta Norma especifica requisitos para um sistema de gestão da qualidade quando uma organização: a) necessita demonstrar sua capacidade para prover consistentemente produtos e serviços que atendam aos requisitos do cliente e aos requisitos estatutários e regulamentares aplicáveis, e b) visa aumentar a satisfação do cliente por meio da aplicação eficaz do sistema
ISO/IEC 25001:2007	Esta Norma especifica requisitos e recomendações para uma organização responsável por implementar e gerenciar a especificação de requisitos de qualidade do produto de software e pelas atividades de avaliação da qualidade de software.
ISO/IEC 15504-4:2004	Faz parte da norma ABNT 15504 e traz orientações de como utilizar o processo de avaliação conforme contexto de um programa de melhoria
ISO/IEC 25062:2011	Norma que especifica um relatório de medidas obtidas em teste de usabilidade e qualidade, conforme eficiência, eficácia e satisfação dentro de um contexto especificado

FONTE: ELABORAÇÃO PRÓPRIA.

Além das normas ISO, existem outros modelos como o CMMI (*Capability Maturity Model Integration*), mantido pelo *Software Engineering Institute* (SEI), nos Estados Unidos. Este instituto tem contribuído com o fortalecimento da área de qualidade de *software*, através da definição de modelos internacionais focados no processo de *software*, assim como na publicação de relatórios técnicos e artigos relacionados ao assunto (TONINI; CARVALHO; SPINOLA, 2008).

O CMMI foi desenvolvido pela SEI, em cooperação com a indústria de *software* e o governo americano para construir um *framework* que consolidasse e integrasse outros modelos já desenvolvidos pelo SEI. Este modelo tem sua origem mediante a evolução de outros três modelos que, segundo Agrawal e Chari, 2007, foram: SW-CMM (*Capability Maturity Model for Software*), o SE-CMM: EIA 731 (*System Engineering Capability Maturity Model*) e o IPD-CMM (*Integrated Product Development Capability Maturity Model*). Sua arquitetura é composta basicamente pela

definição de um conjunto de áreas de processos, apresentadas de duas formas distintas, uma como um modelo de estágio e uma como um modelo contínuo. Portanto, a utilização das normas ISO em conjunto como o CMMI permite que haja controle e mantenimento para alcançar a qualidade de um *software* desejada.

2.2 TESTE DE SOFTWARE

O teste de *software* é um processo que tem como objetivo principal encontrar problemas de programação ou de uso nos componentes do *software* desenvolvido ou em desenvolvimento, evitando que esses problemas possam ser corrigidos antes de ser colocado em ambiente de produção. Este processo é feito a partir de especificações realizadas e posteriormente disponibilizadas com base em um levantamento de requisitos realizado diretamente com o cliente, onde há um detalhamento de funcionalidades que o sistema deve conter.

Segundo Viegas (2017), Os testes de *software* podem ser classificados em:

- **Teste de Unidade** – Seu objetivo é validar pequenas partes do código do software com base em suas possíveis entradas e saídas especificadas.
- **Teste Funcional** - Tem como objetivo validar funcionalidades, requerimentos, regras de negócio descritas na documentação.
- **Teste Estrutural** – Seu objetivo é apurar o comportamento do sistema com foco no código fonte sob perspectivas de condições, fluxo de dados, ciclos e caminhos lógicos.
- **Teste de Performance** - Submete o software a simulações intensas com o objetivo de validar o seu comportamento, analisando se o tempo de resposta é o esperado para o momento de utilização da aplicação.
- **Testes de Segurança** - Visa garantir a segurança da aplicação das mais diversas formas, testando papéis, perfis e permissões para navegar no sistema garantindo que os acessos estão compatíveis com o que foi idealizado.
- **Teste de Integração** - Seu objetivo é realizar a junção de módulos do sistema e testa-los verificando o comportamento em conjunto.
- **Teste de Instalação e configuração** - Seu objetivo é verificar o comportamento do sistema em configurações distintas e nas configurações especificadas de ambiente.
- **Teste de Integridade** - Visa garantir que os componentes envolvidos irão manter-se íntegros mesmo com um alto volume de dados.
- **Teste de Usabilidade** - Tem por objetivo verificar a compreensibilidade que o sistema

possui em sua manipulação pela ótica do usuário.

- **Teste de Regressão** - Consiste na aplicação de novos testes em componentes já analisados após modificações do sistema, ou implantações novas, averiguando se que o sistema continua consistente com o que foi idealizado.

- **Teste Automatizado** - Realizado por um script de códigos escritos posteriormente para um determinado sistema, que executam testes automáticos em aplicações.

- **Teste Exploratório** - Consiste na exploração do *software* enquanto simultaneamente, realiza a criação e execução dos testes.

Cada um dos tipos descritos pode ser utilizado isoladamente ou em conjunto, a fim de atingir o objetivo para uma análise específica.

2.3 AUTOMAÇÃO DE TESTES

A automação de testes é parte fundamental no processo de desenvolvimento de *software* em qualquer empresa que trabalha com processos padronizados para criar seus produtos, sejam eles empresariais ou de lazer para usuários finais (SERNA; MARTÍNEZ; TAMAYO, 2019).

Estando alinhada com as boas práticas de desenvolvimento de *software*, utiliza o CMMI e a ISO como modelos para garantir a qualidade dos produtos entregues, além da aplicação de outras práticas como testes de maturidade e orientações de projeto (JAMMALAMADAKA; RAMAKRISHNA, 2016).

Existem estudos relatando a dificuldade de se automatizar sistemas legados e com códigos não padronizados, gerando obstáculos para se obter a qualidade de *software* quando comparados os números de Erros *versus* a Cobertura de Teste (WILD, 2003).

A automação de testes não visa somente realizar este processo através de *scripts* confeccionados com base em casos de testes, previamente desenhados e moldados, mas também realizá-los de forma inteligente com cada vez mais ênfase nas experiências dos usuários.

Assim, segundo Costa (2006), “[...] a automação traz a possibilidade de tornar o teste mais rápido e efetivo em encontrar erros, desde que se tenha atenção quanto ao que automatizar e como deve ser feita esta automação.”. A grande vantagem desta abordagem é que todos os casos de teste podem ser facilmente e rapidamente repetidos a qualquer ocasião com mínimo de esforço.

Identificar o elemento a ser testado é um passo essencial para a realização da confecção do *script* de automação de testes. Contudo, com sistemas legados onde elementos não são nomeados e não há padronização no código, a realização de testes automatizados eficientes é uma tarefa árdua (COSTA, 2018).

2.4 WEBSCRAP

Também conhecido como Raspagem de Dados, o *webscrap* é uma técnica de extração de dados empregada na coleta dados de sites na internet. Essa técnica possibilita a captura de conteúdo em formato HTML por meio das *tags* utilizadas na construção das páginas, possibilitando a padronização da informação ou exportação para outros formatos (CALDERÓN; GUTIÉRREZ, 2006).

Encontram-se disponíveis diversas ferramentas ou recursos das linguagens de programação que permitem a realização da extração dos dados. Uma das linguagens mais utilizadas para esta prática é o Python por meio do módulo *requests* e da biblioteca *Beautifulsoup4*, esta última apresentando funções que permitem realizar tarefas de extração de dados de forma simplificada, baixo custo computacional e com alto desempenho (CHATTERJEE, 2019).

O termo *webscrap* está fortemente relacionado ao conceito de Ciências de Dados ou *Data Science*, uma vez que utiliza a *web* como fonte de extração de dados e, por tal motivo, é de relevante importância a discussão sobre sua utilização e das técnicas utilizadas para sua prática.

2.5 TECNOLOGIAS WEB

Os sistemas e aplicações com base na *web* produzem uma complexa matriz de conteúdos e funcionalidades para uma ampla população de usuários (FERREIRA FILHO e FERREIRA, 2009). Tais sistemas evoluíram para ferramentas computacionais sofisticadas que oferecem, além das funções básicas, a integração com bancos de dados corporativos e aplicações de negócios (PRESSMAN, 2011; VIEIRA; NUNES, 2012). Para construção de sistemas desta natureza são utilizadas tecnologias como o HTML, CSS e Javascript.

2.5.1 HTML

O HTML (*Hyper Text Markup Language*) ou Linguagem de Marcação de Hipertexto é uma linguagem utilizada na produção de páginas Web, tendo como função definir uma padronização de modo que a informação seja interpretada por qualquer computador (BRANNAN, 2009). A interpretação de tais páginas é dada por meio da utilização de programas mais conhecidos como navegadores.

Um documento HTML é formado por etiquetas ou *tags*, as quais são os comandos da linguagem e que têm a função de marcar e formatar o texto para exibição no navegador (HARRIS, 2009). Quando solicitado, o documento HTML é enviado pelo servidor para o computador solicitante, interpretado e apresentado na tela. Uma etiqueta é formada por comandos, atributos e valores. O comando é o nome da etiqueta propriamente dito, os atributos são os modificadores deste comando e os valores são os dados afetados pelo comando. Para o seu correto funcionamento em concordância com os padrões do W3C, as etiquetas devem estar envolvidas entre os símbolos menor “<” e barra maior “/>”, os quais indicam o início e o término da mesma, existindo ainda algumas etiquetas com

sintaxe diferenciada (DE SOUZA, 2013).

As *tags* do HTML permitem a interação do usuário com uma página, as quais permitem a inserção de controles como caixas de texto (*input*), botões de opção (*Radio-Button* e *Check-Box*), botões de ação (*Button*), entre outros. Neste contexto, a utilização de forma correta permite que os elementos sejam interpretados pelos navegadores e que o usuário faça uso das suas funcionalidades, direcionando-o para a função estabelecida para aquele elemento.

2.5.2 CSS

O CSS (*Cascading Style Sheets* ou Folhas de Estilo em Cascata) é uma linguagem que permite aplicar formatos de estilos no HTML, de forma a personalizar a apresentação dos conteúdos gerados por meio das *tags* HTML. A sua utilização permite, por exemplo, alterar o estilo de uma letra, animação de elementos na tela, configuração de margens, entre outros recursos (BRANNAN, 2009).

Apesar de melhorar a qualidade estética das páginas da *web*, a utilização incorreta causa erros no momento da renderização, ou seja, o processamento dos códigos pelo navegador, provocando desconfiguração da página ou não permitindo o acesso ao elemento desejado. Portanto, estes recursos devem ser usados com prudência, fornecendo uma apresentação estética de qualidade, mas permitindo o funcionamento correto dos elementos HTML inseridos nas páginas.

2.5.3 JAVASCRIPT

O Javascript é uma linguagem de programação utilizada em conjunto com o HTML, a fim de proporcionar dinâmica na utilização dos elementos implementados em uma página (CHAFFER, J.; SWEDBERG, 2009).

Uma das aplicações desta linguagem mais presentes em páginas HTML é a realização da validação dos dados digitados nos campos de entrada ou seleção de objetos. Essa prática proporciona dinamismo nas páginas, mas, em contrapartida, pode gerar erros devido à incompatibilidade de versão dos navegadores ou até mesmo ausência do interpretador da linguagem nos mesmos.

O Javascript pode ser estendido por meio de *frameworks*, conjunto de instruções prontas que facilitam a implementação de funcionalidades. Um exemplo largamente utilizado é a biblioteca JQuery, que promove muitos recursos (BIBEAULT; KATZ, 2010; RICHARDS et al., 2010), destacando-se: orientação a objetos; adição de efeitos visuais e animações; acesso e manipulação de objetos HTML; recuperação de informações no servidor sem a necessidade de recarregamento da página; disponibiliza interatividade e alteração de conteúdos dinamicamente. Assim como o Javascript em sua forma pura, a utilização da JQuery possui os mesmos impactos quando aplicada de forma incorreta.

2.6 LINGUAGEM PYTHON

O Python é uma linguagem de programação versátil, permitindo a construção de aplicações que vão desde sistemas para internet, até a criação de jogos digitais. Suas características permitem que uma aplicação possa ser desenvolvida com o paradigma da orientação a objetos ou de forma estruturada (BORGES, 2014).

De acordo com o *Data Science Institute* (2018), a partir do ano de 2014 o Python vem crescendo e ganhando adeptos de forma assustadora, principalmente por ser a linguagem mais utilizada no mundo do *Data Science*. Neste cenário, a mineração de dados faz uso constante das técnicas de *webscrap* para extração de dados. Portanto, neste trabalho a linguagem adotada foi a Python, devido suas características e aplicabilidade dentro da questão a ser desenvolvida no problema de pesquisa.

3. MATERIAIS E MÉTODOS

De acordo com Rampazzo (2005), pode-se dizer que a pesquisa científica é um procedimento que se utiliza de processos para a solução de problemas, contribuindo para o avanço em qualquer área do conhecimento. Do ponto de vista da pesquisa aplicada, Ciribelli (2003, p.54) descreve-a como sendo "[...] aquela que aplica os informes obtidos pela Pesquisa Pura. Equipara-se, portanto, à Tecnologia na atualidade, elemento indispensável para aumentar as tarefas desenvolvidas pelo homem.".

Uma vez que este trabalho propõe o desenvolvimento de um aplicativo para fins práticos de utilização, utilizou-se o método de Pesquisa Exploratória, uma vez que seu objetivo é aumentar o número de informações sobre determinado conteúdo, a fim de se trazer mais conhecimento do assunto em pauta (BASTOS, 2009).

O *framework*, denominado Web-Test, foi construído utilizando a linguagem de programação Python para implementação das técnicas de *webscrap*. Os pacotes (*Unittest, Bs4, Sys, Csv, Pandas, Numpy e Inspect*), foram encapsulados nas classes principais para execução dos *scripts*, os quais foram implementados utilizando os métodos do *framework*, além da utilização dos pacotes do *Selenium*, para interação com os objetos no DOM. Os métodos da classe Web-Test foram criados para facilitar o desenvolvimento dos *scripts* de testes automatizados de maneira mais simples para o desenvolvedor de testes.

O algoritmo é divido em três fases, onde a primeira realiza a busca que representa a fase de identificação das principais características dos elementos pesquisados, para que seja possível a captura de propriedades significativas e relevantes para realização dos testes funcionais, Figura 2.

FIGURA 2. PROPRIEDADES DE UM TESTE.

```
# confirma a gravação da inclusão do pedido de venda.
oHelper.SetButton("Salvar")

# após a gravação, o Protheus volta para a tela de inclusão, então eu cancelo essa tela para voltar para o mbrowse.
oHelper.SetButton("Cancelar")

# na tela do mbrowse, seleciono o indice e pesquiso o pedido de venda incluido acima.
oHelper.PesqBrowse(True, 'Filial+numero',"D MG 01 SEL102")
```

FONTE: ELABORAÇÃO PRÓPRIA (2018)

A segunda fase, Figura 3, consiste em identificar os padrões encontrados durante o processamento de busca e realizar o armazenamento das informações em um modelo de dados estruturado. Este procedimento permite acessar tais informações durante a próxima execução e tornar a busca dos elementos mais precisa e performática em relação à execução anterior. Algumas variações podem ocorrer entre determinados sites devido alterações nas características dos elementos utilizados e, para esses casos, o *framework* é preparado para identificar os diferentes padrões de elementos HTML (*tags*), capturar e armazena-los.

FIGURA 2. CÓDIGO PARA IDENTIFICAÇÃO DE PADRÕES.

```
def SetSenha(self, senha):
        """
        Preenchimento da senha do sistema
        """
        try:
            Id = self.SetScrap('Senha', 'div', 'tget')
            if self.consolelog:
                print('SetUsr ID: %s' %Id)
            field = self.driver.find_element_by_id(Id)
            field.click()
            field.send_keys(senha)
```

FONTE: ELABORAÇÃO PRÓPRIA (2018)

Por fim, são realizadas as ações com base nas características capturadas dos elementos, que representam a interatividade do usuário e sistema, realizando então os testes funcionais automatizados, as ações podem variar de acordo com o elemento e seus respectivos eventos, por exemplo, um *input (<input></input>)*, irá reproduzir ações de digitação de através de teclado, um *button (<button></ button>)*, irá reproduzir ações de cliques de *mouse*. Na Figura 3 é mostrado o resultado esperado para execução.

FIGURA 3. CÓDIGO PARA IDENTIFICAÇÃO DE PADRÕES.

```
oHelper.UTCheckResult("aItens","C6_PRODUTO", "COM000000000000000000000000110")
oHelper.UTCheckResult("aItens","C6_QTDVEN","1.000,00")
oHelper.UTCheckResult("aItens","C6_PRCVEN","990,00")
oHelper.UTCheckResult("aItens","C6_TES","501")
oHelper.UTCheckResult("aItens","C6_DESCONT","1,00")
```

FONTE: ELABORAÇÃO PRÓPRIA (2018)

Os resultados esperados das ações são validados através dos métodos de teste do pacote *Selenium*, nas quais as validações são feitas de acordo com os valores enviados pelo usuário, ou, em alguns casos, podem ser alteradas de acordo com a regra de negócio do site como os *inputs*, ações de cliques e preenchimento de campos em tela. Ao final dos testes os resultados são capturados e armazenados nos arquivos de *log* que são gerados durante o processo e que podem ser utilizados como parâmetro para gestão de testes posteriormente.

A metodologia utilizada fornece flexibilidade e confiança nos testes de qualidade, pois ao se identificar características de elementos, a inclusão de novos elementos em determinadas interfaces ou a mudança de sua localização tornam-se transparentes para a ferramenta, uma vez feita à identificação de um padrão de características, sua identificação se torna variável em função do contexto atual.

4. RESULTADOS E DISCUSSÃO

O *framework* de testes *Selenium* foi utilizado para a execução dos *scripts* de testes automatizados. Os testes foram executados em páginas HTML de sites públicos e de sistemas ERP no qual seu legado abrange páginas HTML, que foram convertidas de telas criadas utilizando a linguagem de programação *Python* para o modelo de aplicação *Web Browser*.

Foram realizados 150 testes de automação em três endereços diferentes da *web* para realização das análises de desempenho, qualidade de teste e comportamento do algoritmo nas pesquisas dos elementos, mostrados na Figura 4.

FIGURA 4. COMPARATIVO SOBRE A EXECUÇÃO DO ALGORITMO.

Site	Componentes HTML	Resposta (ms)	Testes	Sucesso	Falha	Taxa de Acerto
Mercado Livre	Button	0,2	150	146	4	97,33%
	Input	0,5	150	147	3	98,00%
	Link	0,1	150	150	0	100,00%
	Combo-Box	0,3	150	147	3	98,00%
	Radio-Button	0,1	150	147	3	98,00%
	Check-Box	0,1	150	150	0	100,00%
OLX	Button	0,1	150	149	1	99,33%
	Input	0,7	150	144	6	96,00%
	Link	0,1	150	148	2	98,67%
	Combo-Box	0,2	150	150	0	100,00%
	Radio-Button	0,1	150	150	0	100,00%
	Check-Box	0,1	150	150	0	100,00%
SPTrans	Button	0,1	150	142	8	94,67%
	Input	0,2	150	148	2	98,67%
	Link	0,1	150	127	23	84,67%
	Combo-Box	0,2	150	143	7	95,33%
	Radio-Button	0,2	150	147	3	98,00%
	Check-Box	0,1	150	139	11	92,67%

FONTE: ELABORAÇÃO PRÓPRIA (2018)

Os números mostram que a metodologia utilizada se mostrou eficiente para detecção dos padrões e identificação de elementos, apresentando em média, um índice de sucesso nas detecções e acionamento e/ou preenchimento dos elementos com 98,56% para o Mercado Livre, 99,00% para o OLX e, por fim, de 94,00% para a SPTrans.

A relevância dos elementos testados e localizados nos endereços, Figura 4, mostra que há dificuldades para o usuário quando se trata, principalmente, da localização ou acesso aos *links*, dificultando a navegação ou prosseguimento da mesma.

Os algoritmos de busca foram precisos em todos os casos de localização para inclusão de dados, realizando a simulação de um teste manual feito de forma automatizada, onde as execuções foram realizadas com sucesso em 100% dos casos no Mercado Livre, 96% no OLX e 98,67% na SpTrans, mostrando que há problemas como validação de campos ou preenchimento automático de valores.

No site SPTrans o número de acesso aos *links* teve um índice de 84,67% de sucesso em relação ao Mercado Livre com 100% e ao OLX com 98,67%. Esses números indicam que a navegação no site da SPTrans prejudica o usuário, não permitindo o acesso aos dados ou informações solicitadas. O mesmo ocorre no site OLX, entretanto, com menor impacto.

Os demais elementos *Button*, *Combo-Box*, *Check-Box* e *Radio-Button* obtiveram valores médios de 97,78% sobre a localização e acesso, contabilizando 98,33% par o Mercado Livre, 99,83% para o OLX e 97,78% para a SPTrans. Esses números evidenciam que há problemas relacionados as tecnologias utilizadas na construção dos elementos como o uso de CSS e/ou Javascript, podendo ocorrer por incompatibilidade com o navegador ou desatualização da versão em uso do mesmo.

Os índices de sucesso de localização ou acesso aos elementos que ficaram abaixo dos 100% indicam que há necessidade de correções ou reformulação das estruturas de programação nas páginas para adequação aos padrões do W3C (*World Wide Web Consortium*) e adequação às normas de qualidade de *software*.

As falhas encontradas mostram que a ausência de tais práticas prejudicam a qualidade das páginas *web*, uma vez que pode ocorrer interrupção da navegação ou falha no acesso à informação solicitada por falta ou não funcionamento dos recursos disponibilizados.

5. CONCLUSÃO

Considerando os aspectos observados nos resultados dos testes, é possível concluir que a solução e as técnicas de *webscrap* aplicadas cumpriram o propósito de se realizar testes automatizados em sistemas com histórico de desenvolvimento legado ou que foram criados de maneiras inadequadas quando se trata de automação de testes.

A precisão na busca dos elementos nos navegadores foi suficiente para garantir a integridade e

qualidade dos *scripts* de testes, tanto no quesito acurácia, quanto no tempo de resposta, conforme mostram os números da Figura 4.

Os resultados evidenciam que a metodologia é aplicável para garantir a qualidade de um sistema desenvolvido para o ambiente *web*, facilitando o processo de melhoria e entrega de um produto de *software*, alinhado com as normas e modelos de qualidade de *software*.

Para trabalhos futuros é proposto o aprimoramento dos algoritmos de busca de características de elementos para novos objetos que possam surgir em páginas de sites ou aplicações *web*. Tal solução pode ser implementada com o uso de técnicas de aprendizagem de máquina acessadas pelas funcionalidades do *framework* Web-Test.

AGRADECIMENTOS

Os autores agradecem a Universidade Nove de Julho/SP pela utilização dos laboratórios para a construção do aplicativo e demais tarefas pertinentes ao projeto.

REFERÊNCIAS

AGRAWAL, M.; CHARI, k. **Software Effort, Quality, and Cycle Time**: A Study of CMM Level 5 Projects. IEEE Transactions on Software Engineering, Vol.33, N.3, Março,2007.

BASTOS, R. L. **Ciências humanas e complexidades:** Projetos, métodos e técnicas de pesquisa. O caos, a nova ciência. 2a edição. ed. Rio de Janeiro: e-Papers, 2009.

BIBEAULT, B.; KATZ, Y. ***jQuery in Action***. 2 ed. Greenwich, CT: Manning Publications Co., 2008. p. 475. ISBN 1-935182-32-3.

BORGES, L.E. **Python para desenvolvedores: aborda Python 3.3**. Novatec Editora. 2014.

BRANNAN, J. A. **Briliant HTML & CSS**. 1 ed. Edinburgh, UK: Pearson Education, 2009, p. 304. ISBN: 978-0-273-72152-9.

CALDERÓN, J. J.; GUTIÉRREZ, M. A. A. Web-scrap: A tool for accelerated and optimized navigation and web search. **In**: 12TH AMERICAS CONFERENCE ON INFORMATION SYSTEMS, *AMCIS 2006 Proceedings* (2006): 515. 2006.

CHAFFER, J.; SWEDBERG, K. ***Learning jQuery 1.3 - Better Interaction Design and Web Development with Simple JavaScript Techniques***. Birmingham, UK: Packt Publishing Ltd., 2009. p. 421. ISBN 978-1-847196-70-5.

CHATTERJEE, P. et al. Exploratory Study of Slack Q&A Chats as a Mining Source for Software Engineering Tools. **2019 IEEE/ACM 16th International Conference on Mining Software Repositories (msr)**, [s.l.], p.490-501, may. 2019. IEEE. http://dx.doi.org/10.1109/msr.2019.00075.

CIRIBELLI, M. C. **Como elaborar uma dissertação de mestrado através da pesquisa científica**. Rio de Janeiro: 7Letras, 2003.

COSTA, Fernanda Guimarães. **Problemas na Manutenção de Sistemas Legados:** Um Estudo de Caso. 2018.

34 f. Monografia (Especialização) - Curso de Qualidade de Software, Vale do Rio dos Sinos, São Leopoldo, 2018.

COSTA, Mozart Guerra. ***Estratégia de automação em testes: requisitos, arquitetura e acompanhamento de sua implantação***. 2006. 116f. Dissertação (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação, Campinas, SP. Disponível em: http://www.repositorio.unicamp.br/handle/REPOSIP/276289. Acesso em: 6 ago. 2018.

DATA SCIENCE INSTITUTE. O Incrível Crescimento da Linguagem Python. Disponível em: http://datascienceacademy.com.br/blog/o-incrivel-crescimento-da-linguagem-python/. Acesso em: 20 out. 2019.

DE SOUZA, Edson Melo. ***Desenvolvimento de um Sistema de Apoio à Decisão e Operacional para a Otimização dos Parâmetros de Corte em Usinagem (COPPISYS)***. 2013. 149 f. Dissertação (Mestrado em Engenharia de Produção) – Programa de Mestrado em Engenharia de Produção, Universidade Nove de Julho, São Paulo, 2013.

ENTRINGER, T. C; DA SILVA FERREIRA, A. Software quality evaluation: a bibliometric analysis and future perspectives. **Independent Journal of Management & Production**, v. 10, n. 5, p. 1499-1515, 2019.

FERREIRA FILHO, O. F.; FERREIRA, M. A. G. V. Serviços Semânticos: uma Abordagem RESTful. In: IADIS Conferência Ibero-Americana WWW/Internet - CIAWI 2009, 2009, Madrid, Spain. ***Annals...*** Madrid, Spain, 2009. p. 35-46.

GUERRA, A.; COLOMBO, RMT. **Qualidade de produto de software**. Ministério da Ciência e Tecnologia, 2009. Disponível em: http://www.mct.gov.br/upd_blob/0203/203505.pdf. Acesso em: 26 out. 2019.

HARRIS, B. A. *Core HTML*. ***Image Rochester NY***, p. 1-6, 2009.

JAMMALAMADAKA, K.; RAMAKRISHNA, V. A model to quantify and improve software test automation. **International Journal of Control Theory and Applications**, [s. l.], v. 9, n. 34, p. 273–282, 2016.

PRESSMAN, R. S. **Engenharia de Software - Uma Abordagem Profissional**. 7. ed. Porto Alegre: Bookman, 2011. p. 780.

RAMPAZZO, L. **Metodologia Científica Para alunos de graduação e pós-graduação**. 3ª. ed. São Paulo: Edições Loyola, 2005.

RICHARDS, G.; LEBRESNE, S.; BURG, B.; VITEK, J. *An analysis of the dynamic behavior of JavaScript programs.* ***2010 ACM SIGPLAN conference on Programming language design and implementation PLDI 10***, v. 45, n. 6, p. 1, 2010.

SERNA, E. M.; MARTÍNEZ, R. M.; TAMAYO, P. O. A review of reality of software test automation | Una revisión a la realidad de la automatización de las pruebas del software. **Computacion y Sistemas**, [s. l.], v. 23, n. 1, p. 169–183, 2019.

TONINI, A. C.; CARVALHO, M. M.; SPINOLA, M. M. Contribuição dos modelos de qualidade e maturidade na melhoria dos processos de software. **Produção**, v. 18, n. 2, p. 275-286, 2008.

VIEGAS, Júlio. **Teste de Software: Introdução, Conceitos Básicos e Tipos de Testes.** 2017. Disponível em: https://blog.onedaytesting.com.br/teste-de-software/. Acesso em: 20 set. 2018.

VIEIRA, F. J. R.; NUNES, M. A. S. N. DICA: Sistema de Recomendação de Objetos de Aprendizagem Baseado em Conteúdo. **Scientia Plena**. v. 8, p. 1-10, 2012.