INTERFACE TECNOLÓGICA

VELOCIDADE DO SELENIUM WEBDRIVER & JUNIT

SELENIUM WEBDRIVER & JUNIT SPEED

Rijordy Patrick Moura de Oliveira – rijordy_@hotmail.com
Faculdade de Tecnologia de Taquaritinga – Taquaritinga – São Paulo – Brasil
Marcus Rogério de Oliveira – marcus.oliveira@fatectq.edu.br
Faculdade de Tecnologia de Taquaritinga – Taquaritinga – São Paulo – Brasil

RESUMO

O objetivo deste trabalho é mostrar a diferença na velocidade entre realizar testes com o uso de Teste Manual e com Selenium WebDriver & Junit. Para isso, será feito uma breve introdução ao mundo de qualidade de software, dizendo o porquê de ser tão importante e a dificuldade de se entregar um produto no prazo e com a qualidade desejada. Também será comentado o que é a automação de testes de software, o porquê de ela ser tão necessária nos dias de hoje, e suas vantagens. Para ambos os tipos de teste será utilizada a mesma aplicação, que são dois formulários de cadastro. Para os testes automatizados serão utilizados: a Eclipse IDE para escrever os comandos em Java, o WebDriver, para realizar as operações Web e o Junit, que é necessário para estruturar os testes. Também serão mostrados os resultados obtidos, demonstrando que ao utilizar esse tipo de automação se consegue diminuir bastante os testes em longo prazo.

Palavras-chave: Teste.Qualidade.Velocidade.WebDriver.Junit.

ABSTRACT

The objective of this work is to show the difference in speed between performing tests with the use of Manual Test and with Selenium WebDriver & Junit. For this, a brief introduction will be made to the world of software quality, telling you why it is so important and the difficulty of delivering a product on time and with the desired quality. It will also be commented on what automation of software tests is, why it is so necessary in these days, and its advantages. For both types of test the same application will be used, which are two registration forms. For automated testing will be used: the Eclipse IDE, to write the commands in Java, the WebDriver to perform the Web operations, and the Junit that is required to structure the tests. It will also be shown the results obtained, showing that using this type of automation can greatly reduce the tests in the long term.

Keywords: Test.Quality.Speed.WebDriver.Junit

1 INTRODUÇÃO

Qualidade e tempo são duas incógnitas que sempre batem de frente, pois muitas vezes que se tenta fazer algo com qualidade, o prazo acaba sendo ultrapassado. E quando se tenta fazer algo no prazo, a qualidade nem sempre é das melhores.

O que traz qualidade ao software são os testes, que devem apontar todos os problemas encontrados. Mas até mesmo os testes possuem um problema com o tempo, isso ocorre devido ao fato de ter-se que testar o mesmo caso de teste várias vezes e ao invés de otimizar a função, acaba-se gastando muito tempo com funcionalidades repetidas.

Para ajudar nesse problema vieram os testes automatizados. Eles irão nos ajudar de uma forma mais eficaz, e nos possibilitar uma alternativa de ter ter um sistema que possa ser entregue no prazo e com qualidade.

2 TESTE DE SOFTWARE

De acordo com Glen Myers (*apud* Rios et al., 2006, p. 10) teste de software pode ser definido como processo de executar um programa ou sistema com a intenção de encontrar defeitos.

Qualidade é fundamental em tudo que é feito, e as empresas vêm se preocupando mais e mais com esse fator. Hoje em dia a concorrência é bem vasta, e não é mais preciso apenas ter um diferencial, mas sim ter um diferencial com qualidade. E para empresas de desenvolvimento de software não é diferente.

Segundo Greife (2013), teste de software é a etapa de controle de qualidade, serve para assegurar que o software está contemplando todas as funcionalidades esperadas e que estas estão funcionando corretamente.

Então o que podemos entender como qualidade? Qualidade é fazer o que era para ser feito, de acordo com o que o cliente queria e de forma que o satisfaça. A empresa não deve elaborar um projeto sem o total conhecimento das necessidades e vontades do cliente, pois o que é considerado o melhor para a empresa nem sempre é o melhor para o cliente. Qualidade de software funciona da mesma forma.

Gandara (2012) diz que temos que pensar: se determinado sistema ficar parado, quanto estou perdendo? Por quanto tempo ficará parado? Por que ocorreu? Ocorrerá novamente?

2.1 Ciclo PDCA

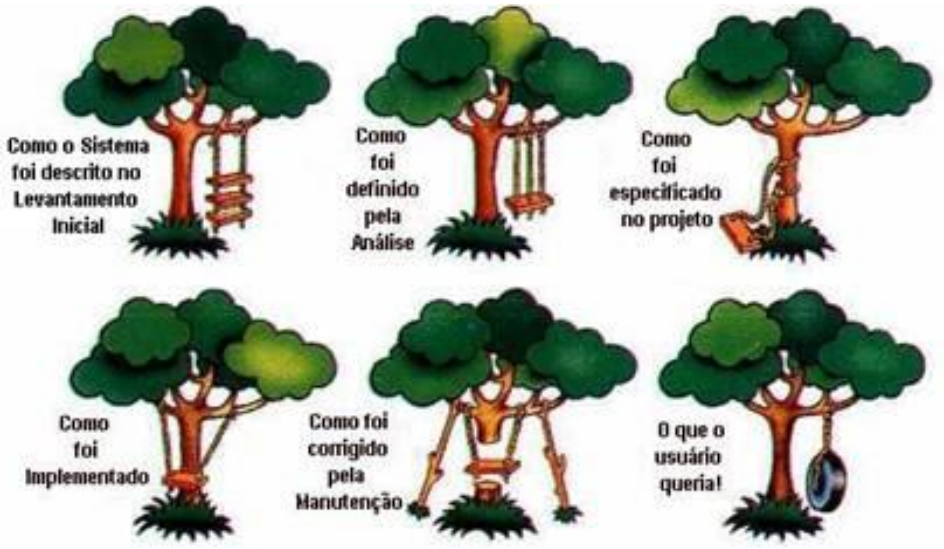
É difícil falar sobre qualidade sem falar sobre o ciclo PDCA (Planejar, Fazer, Verificar e Agir) que vem do inglês Plan, Do, Check e Act. Ele fala que para um produto ter a máxima qualidade, deve passar por esse ciclo várias vezes.

De acordo com Camargo (2017), a primeira coisa a se fazer é Planejar, ou seja, pensar e estruturar tudo o que deve ser feito. Após o planejamento estar pronto é a hora de Fazer, que nada mais é que realizar tudo o que foi planejado. Muitos acham que é simplesmente planejar e fazer, ou apenas fazer, mas as etapas que se seguem também são muito importantes, são elas que garantem o maior nível de qualidade.

Muitas empresas focam em Planejar e Fazer, mas se esquecem das fases de Verificar e Agir, sendo essas, muitas vezes, as mais importantes. Na fase de Verificar será visto se há algum problema, e o que deve ser feito para melhorá-lo. E por fim, esses problemas devem ser solucionados na fase de Agir.

Segundo Justen (2015), podemos separar a qualidade em três tipos de perfis: cliente, empresa e desenvolvedor. A comunicação entre eles é muito importante, contudo na maioria das vezes acaba não sendo muito feliz. O que pode acontecer é o cliente pedir A, o analista de requisitos escrever B e o desenvolvedor fazer C. Veja a figura 1.

Figura 1: Ciclo do desenvolvimento de um software



Fonte: CESAR (2015)

Para sanar esses problemas as empresas de software estão cada vez mais em busca de formar equipes de testadores de softwares, cujo maior objetivo é minimizar os defeitos que possam ocorrer em seus produtos.

2.2 Introdução ao teste de software

Testar significa avaliar as características ou qualidades de algo, e provar que aquilo funciona ou não. Dessa forma, pode-se dizer que ao testar estamos pondo à prova de que aquilo está funcionando de maneira correta, ou seja, tem qualidade. Assim sendo, podemos entender que qualidade e teste estão diretamente ligados, pois os testes vieram da necessidade de qualidade.

De acordo com Bill Hetzel (*apud* Rios et al., 2006, p. 10) teste de software também pode ser definido como qualquer atividade que a partir da avaliação de um atributo ou capacidade de um programa ou sistema seja possível determinar se ele alcança os resultados desejados.

O teste de software muitas vezes não é muito bem visto por ser uma atividade destrutiva e não construtiva, isto é, seu principal objetivo é mostrar e apontar defeitos e falhas no sistema. Mas isso tudo é uma má interpretação. O teste de software trabalha com qualidade e seu foco

sempre foi e sempre será entregar um sistema com qualidade e que vá suprir as necessidades do cliente.

É muito importante os testes de software passarem por vários ciclos, e isso deve ser feito porque é bastante comum aparecer um erro no sistema oriundo da correção de outro erro, ou o testador não ter previsto certo cenário.

O grande problema dos testes de software é o tempo, quando começa a ser gasto mais tempo realizando testes antigos do que novos. Desse modo, a mão de obra que poderia estar testando novas funcionalidades, estará testando funcionalidades que já foram testadas várias vezes, e o pessoal de teste não conseguirá acompanhar as demandas. E isso gera um grande problema, porque o sistema pode acabar sendo entregue com defeitos por não ter sido testado. Para suprir esse problema apareceram os testes automatizados.

2.3 Automação de Teste de Software

Hoje em dia é bem comum o testador ter que testar a mesma funcionalidade várias vezes, onde em cada uma delas há uma boa chance dele cometer um erro. Para sanar este e outros problemas foram criadas ferramentas para realizar testes automatizados. Esse tipo de teste irá diminuir o envolvimento do ser humano com atividades repetitivas, estas, que o levam a cometer erros “bobos”.

Segundo Teixeira (2016), entregar um aplicativo com inteligência de testes embutida no código aporta valor principalmente em testes regressivos e de aceite.

A automação, por mais que em uma crescente expansão, ainda é bastante imatura. Muitas empresas a vê e a trata de maneira errada, não sabem dar o devido valor ao que é necessário para utilizá-la, onde e quando deve ser utilizada, e de que maneira deve ser implementada nos processos do software.

Muitas vezes as empresas compram ferramentas de testes sem saber ao certo o que fazer com elas, por não terem maturidade o suficiente para lidar com testes. Elas não entendem que a automação de teste deve funcionar como uma parte do processo do desenvolvimento do software e não algo que simplesmente irá solucionar todos os problemas.

Para uma automação funcionar é necessário que o sistema esteja funcionando. Para isso, é necessário que tenha sido feito vários testes manuais, deixando o sistema viável para utilizá-

la. Uma empresa madura que consegue impor os testes na fase do desenvolvimento dos seus projetos como uma etapa essencial, terá a necessidade de utilizar automação naturalmente.

Algo a ser levado em conta ao se falar de automação é dar prioridade a testes críticos. É claro que ao final seria bom todo o sistema estar automatizado, mas isso leva tempo e custo. Para contornar isso, devem-se escolher inicialmente os testes que devem ter maior cuidado e atenção.

2.4 Vantagens de Testes Automatizados

Podem-se observar várias vantagens de se utilizar automação de testes, como:

- Ganho de velocidade: Realização dos testes muito mais rápido que o ser humano.

- Diminuição dos custos: Os testadores podem trabalhar em outras atividades pelo simples fato da automação ser mais rápida. Isso porque não estarão gastando horas, dias, semanas em um teste que foi automatizado.

- Nível de profundidade: Os testes automatizados conseguem testar com um nível de profundidade bem maior que o ser humano.

- Diminuição de riscos: Ainda comparando humano e máquina, os testes automatizados conseguem realizar operações repetitivas tranquilamente, onde que para o ser humano seria estressante, diminuindo os riscos de isso afetar os testes.

- Horário: Podem-se deixar os testes automatizados sendo executados fora do expediente. Poupando recursos que são utilizados em horário normal de trabalho.

2.5 Selenium Webdriver & Junit

O Selenium Webdriver é uma ferramenta com uma grande gama de funcionalidades, mas sendo restrito a sistemas Web. Podem-se automatizar HTML, CSS e até XPATH. Ele é feito por linhas de código, e o Selenium atua em várias linguagens de programação, como Java, C#, Ruby, entre outros. Algo bastante interessante sobre ele é que ele pode trabalhar por máquina virtual, ou seja, tem como realizar o mesmo teste em várias configurações diferentes.

O Selenium WebDriver consegue realizar várias funcionalidades de navegação, como acessar links, preencher formulários, colocar imagens, fazer downloads, localizar elementos, entre muitas outras ações possíveis.

Nogueira (s.d) diz que antes disso tudo é necessário ter um navegador web, o que é um ponto forte do WebDriver, pois ele suporta vários (Mozzila Firefox, Google Chrome, Safari, Opera, entre outros). Nogueira (s.d) continua dizendo que também é necessário criar uma instância da classe WebDriver, que é utilizada para realizar todas as operações supracitadas. Assim, é possível afirmar que a função do WebDriver é realizar todos os procedimentos do navegador.

É necessário agora algo que faça, organize e monitore os testes. Para tal, pode ser utilizado o Junit. O Junit trabalha com anotações, e essas anotações irão informar se tal método deve ser executado ou não, qual classe deve ser executada antes e qual deve ser executada depois, também informa se aquela parte do teste deve ser ignorada ou até mesmo se a classe testada é uma suíte de teste. A suíte de teste é uma classe que vai chamar e executar as outras classes, e ela vão saber se caso o caso de teste precisa que o anterior seja bem sucedido para dar certo.

3 ESTUDO DE CASO

Nessa parte será detalhada como foi desenvolvido o estudo de caso que tem como apresentar uma comparação entre o teste manual e o teste automatizado, utilizando o Selenium WebDriver & Junit. Também será mostrada a diferença de tempo entre a utilização das duas abordagens. Um testador, com cinco meses de experiência na área esteve envolvido na implementação e execução de todos os casos de testes deste Estudo de Caso.

3.1 Aplicação Utilizada

Neste estudo de caso foi utilizada uma aplicação web chamada Empresa que realiza cadastros de Funcionários e Clientes. O sistema se encontra em: https://github.com/Rijordy/meuCrud.
Para o cadastro de Funcionários serão necessários os campos Nome, Sexo, Estado Civil, Telefone, Celular e Cargo. Já para o cadastro de Clientes deverão existir os campos Nome, Sexo, Estado Civil, Telefone, Celular e E-mail.
Para os testes automatizados serem aceitos, alguns critérios devem ser supridos:

- Todos os campos do formulário de cadastro devem ser obrigatórios,

- A aplicação deve ser executada em algum desses navegadores: Google Chrome, Mozilla Firefox, Internet Explorer, Safari e Opera.

3.2 Realização dos Testes

Os testes serão realizados de duas formas distintas. Na primeira ele será feito manualmente por cinco usuários, com certo nível de conhecimento de desenvolvimento e testes. Na segunda será realizado pelo WebDriver & Junit, onde seus casos de teste serão executados cinco vezes para se obter uma comparação aos testes manuais.

Não haverá a necessidade dos dados serem salvos, o objetivo é preencher os formulários corretamente e observar a diferença de velocidade em que isso é feito em cada tipo de teste.
Para os Testes Manuais, foi instruído aos cinco usuários o link que deveriam acessar e o que deveriam fazer.

Já para os Testes com o WebDriver foram criados os scripts do teste e logo após eles foram executados, conforme as figuras 2 e 3.

Figura 2: Script do Cadastrar Cliente

```java
import static org.junit.Assert.*;

public class CadastrarCliente {

    private ChromeDriver driver;

    @Test
    public void test() {
        System.setProperty("webdriver.chrome.driver", "C:/Program Files (x86)/Treinamento/Selenium/chromedriver.exe");
        driver = new ChromeDriver();

        driver.get("file:///E:/TCC/xampp/htdocs/Cadastro/cadastrarCliente.php");

        WebElement nome = driver.findElement(By.name("nome"));
        nome.sendKeys("Teste do Cliente");

        Select sexo = new Select (driver.findElement(By.name("sexo")));
        sexo.selectByValue("masculino");

        Select civil = new Select (driver.findElement(By.name("civil")));
        civil.selectByValue("solteiro");

        WebElement telefone = driver.findElement(By.name("telefone"));
        telefone.sendKeys("1632041154");

        WebElement celular = driver.findElement(By.name("celular"));
        celular.sendKeys("16994547865");

        WebElement email = driver.findElement(By.name("email"));
        email.sendKeys("rijordy@teste.com");

    }
}
```

Fonte: Elaborado pelo autor

Figura 1: Script do Cadastrar Funcionário

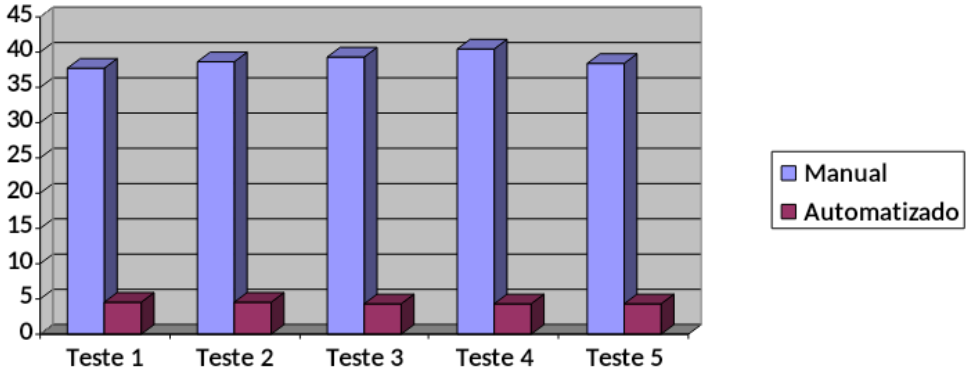
```java
import static org.junit.Assert.*;

public class CadastrarFuncionario {

    private ChromeDriver driver;

    @Test
    public void test() {
        System.setProperty("webdriver.chrome.driver", "C:/Program Files (x86)/Treinamento/Selenium/chromedriver.exe");
        driver = new ChromeDriver();

        driver.get("file:///E:/TCC/xampp/htdocs/Cadastro/cadastrarFuncionario.php");

        WebElement nome = driver.findElement(By.name("nome"));
        nome.sendKeys("Teste do Funcionario");

        Select sexo = new Select (driver.findElement(By.name("sexo")));
        sexo.selectByValue("feminino");

        Select civil = new Select (driver.findElement(By.name("civil")));
        civil.selectByValue("casado");

        WebElement telefone = driver.findElement(By.name("telefone"));
        telefone.sendKeys("1632041154");

        WebElement celular = driver.findElement(By.name("celular"));
        celular.sendKeys("16994547865");

        Select cargo = new Select (driver.findElement(By.name("cargo")));
        cargo.selectByValue("gerente");
    }
}
```

Fonte: Elaborado pelo autor

4 RESULTADOS E DISCUSSÃO

No teste da funcionalidade Cadastrar Funcionário, obtiveram-se os seguintes dados:

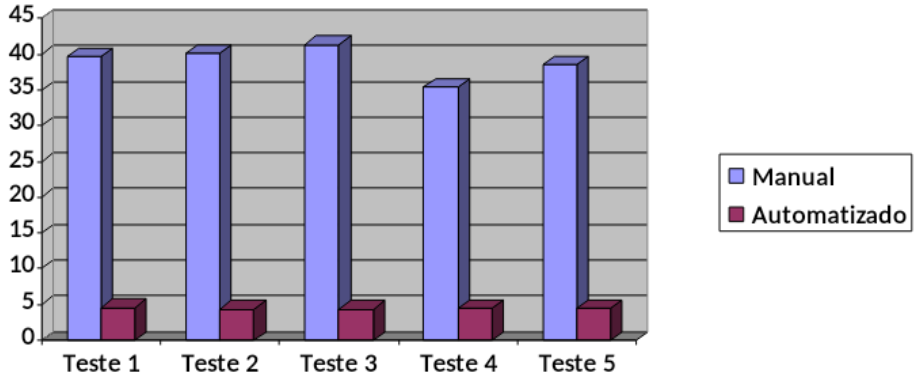
Gráfico 1: Cadastrar Funcionários



Fonte: Elaborado pelo autor

Analisando o gráfico dos dados obtidos, pode-se observar que enquanto os testes manuais levaram em média 38,89 segundos para preencher todos os campos do formulário de cadastro de funcionário, os testes automatizados levaram apenas uma média de 4,38 segundos.

Gráfico 2: Cadastrar Clientes



Fonte: Elaborado pelo autor

Analisando o gráfico dos dados obtidos, pode-se observar que enquanto os testes manuais levaram em média 39,03 segundos para preencher todos os campos do formulário de cadastro de funcionário, os testes automatizados levaram apenas uma média de 4,41 segundos.

Com esses dados, consegue-se afirmar que o teste automatizado com o Selenium WebDriver junto ao Junit faz o teste ser realizado quase 10 vezes mais rápido. Agora, ao somar todo o tempo utilizado para a execução dos testes manuais, chega-se em um valor de 77, 92 segundos, enquanto os testes automatizados levaram dez minutos para seus scripts serem feitos e apenas 8,79 segundos para os casos de testes serem feitos.

Analisando os resultados obtidos, pode-se observar que o teste manual acabou sendo mais rápido, mas a grande diferença está em longo prazo, por dois motivos:

- O primeiro motivo é a velocidade que se ganha com testes futuros. Utilizando o WebDriver após os scripts serem escritos o testador precisará apenas executar os comandos, e as operações desejadas serão realizadas em instantes.
- E a segunda grande vantagem é que o usuário não fica perdido. É bastante comum precisar testar uma funcionalidade que ninguém conhece ou sabe como ela funciona. Se for necessário realizá-la manualmente o usuário gastará bastante tempo tentando entendê-la, já com o WebDriver ele não teria essa dificuldade.

Também se consegue destacar que neste trabalho havia apenas duas funcionalidades de cadastro com poucos campos, mas ao imaginar um sistema com dezenas de funcionalidades e que devem ser testadas várias e repetidas vezes, o gasto de tempo na criação dos scripts iria aumentar consideravelmente. Por isso é muito importante saber quais casos de teste devem ser automatizados primeiro.

5 CONSIDERAÇÕES FINAIS

É muito importante ter meios de se conseguir entregar um software com qualidade e no prazo certo para todas as empresas, incluindo as de desenvolvimento de softwares.

Dessa forma podem ser utilizados testes automatizados, que irão realizar operações repetitivas onde um ser humano poderia cometer erros.

Algo a se pensar é o que aconteceria se algo desse errado na execução dos testes automatizados, ou se o sistema não estivesse funcionando corretamente antes de iniciar esse tipo de teste.

Dessa forma conclui-se que os testes automatizados são muito importantes para entregar projetos no prazo e com qualidade. Entretanto para isso funcionar o sistema já teve ter passado por testes manuais, e necessita de um profissional que saiba lidar com um erro na execução do teste.

REFERÊNCIAS

CAMARGO, Renata Freitas de. **Ciclo PDCA: do conceito à aplicação do famoso Plan Do Check Act (tudo sobre Ciclo de Deming).** 2017. Disponível em: <https://www.treasy.com.br/blog/ciclo-pdca>. Acesso em: 22 ago. 2017.

CESAR, Luan. Engenharia de Software - Introdução a Teste de Software. 2015. Disponível em: <https://pt.linkedin.com/pulse/engenharia-de-software-introdução-teste-luan-cesar>. Acesso em: 22 ago. 2017.

GANDARA, Fernando. **Qualidade e Teste em software**. 2012. Disponível em: <https://books.google.com.br/books?id=L8_5-wgj4n0C&printsec=frontcover&dq=testes+de+software&hl=pt-BR&sa=X&ved=0ahUKEwiw6eKBsejVAhUMkpAKHSXEAIQQ6wEIKDAA#v=onepage&q&f=false>. Acesso em 12 Ago. 2017.

GREIF, Caterine. **O que é Teste de Software e qual a sua importância?**. 2013. Disponível em: <https://www.kinghost.com.br/blog/2013/10/o-que-e-teste-de-software-e-qual-a-sua-importancia/> Acesso em: 21 de Agosto de 2017.

JUSTEN, Willian. **Entendendo Testes de Software.** 2015. Disponível em: < https://willianjusten.com.br/entendendo-testes-de-software/#intro> Acesso em: 21 de Agosto de 2017.

NOGUEIRA, Elias. **BLOG > SELENIUM WEBDRIVER - PARTE 1**. Disponível em: < http://www.qualister.com.br/blog/selenium-webdriver---parte-1> Acesso em: 24 de Agosto de 2017.

RIOS et al. **Teste de Software.** Disponível em: <https://books.google.com.br/books?id=I2a2BAAAQBAJ&printsec=frontcover&dq=testes+de+software&hl=pt-BR&sa=X&ved=0ahUKEwiw6eKBsejVAhUMkpAKHSXEAIQQ6AEIMTAC#v=onepage&q&f;=false>. Acesso em: 23 ago. 2017.

TEIXEIRA, Tiago. **5 ferramentas de automação de testes para projetos digitais**. 2016. Disponível em: <http://convergecom.com.br/tiinside/services/14/10/2016/5-ferramentas-de-automacao-de-testes-para-projetos-digitais/> Acesso em: 24 de Agosto de 2017