CAMPUS JABOATÃO — IFPE INSTITUTO FEDERAL DE PERNAMBUCO

Pós-Graduação em Gestão e Qualidade em Tecnologia da Informação e Comunicação

ADRIANO FÉLIX SOARES

COMPARATIVO ENTRE FERRAMENTAS DE TESTES DE SOFTWARE E2E EM SISTEMAS WEB

JABOATÃO DOS GUARARAPES

2022

ADRIANO FÉLIX SOARES

COMPARATIVO ENTRE FERRAMENTAS DE TESTES DE SOFTWARE E2E EM SISTEMAS WEB

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do diploma de Pós-Graduação Lato Sensu em Gestão e Qualidade em Tecnologia da Informação e Comunicação do IFPE Campus Jaboatão dos Guararapes, sob orientação o professor Francisco do Nascimento Junior.

JABOATÃO DOS GUARARAPES

2022

FICHA CATALOGRÁFICA

S676c

Soares, Adriano Félix.
Comparativo entre ferramentas de testes de software E2E em sistemas web/ Adriano Félix Soares; orientação de Francisco do Nascimento Júnior. Jaboatão dos Guararapes, 2022.

26f.; il.
Trabalho de Conclusão de Curso (Especialização em Gestão e Qualidade em Tecnologia da Informação e Comunicação) – IFPE - Campus Jaboatão dos Guararapes.
Inclui Referências.

1. Tecnologia da Informação e Comunicação 2. Software - Testes 3. Programação (Computadores) - Gerência I.Nascimento Júnior, Francisco do. II. IFPE. III. Título

CDD 004.21

Catalogação na fonte elaborada pelo Bibliotecário Thiago Leite Amaro da Silva - CRB4/ 2.021

COMPARATIVO ENTRE FERRAMENTAS DE TESTES DE SOFTWARE E2E EM SISTEMAS WEB

Trabalho aprovado. Jaboatão dos Guararapes, 06 de Outubro de 2022

_____

Professor Francisco do Nascimento Júnior

_____

Prof. Josino Rodrigues Neto

_____

_____

Havana Diogo Alves de Andradre

Jaboatão dos Guararapes

2022

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

AGRADECIMENTOS

A Deus, por ter me dado força e saúde mesmo em uma rotina de trabalho intensa, proporcionou-me energias para realizar essa pesquisa.

Ao meu orientador, professor Francisco, pela paciência e dedicação, me mostrando os melhores caminhos a serem tomados para conclusão deste trabalho.

Aos amigos, em especial os da área de T.I que pude contar para sanar algumas dúvidas.

Resumo

O presente artigo aborda o estudo das ferramentas de automação de testes Selenium e Cucumber, há como objetivo distinguir através de um estudo documental suas aplicações, e transparecer as vantagens e desvantagens na utilização de cada ferramenta. A motivação do trabalho é demonstrar a evolução das ferramentas de testes de software em nível de teste de ponta a ponta, *End-to-End (E2E)*, avaliando a solução Cucumber, apresentando quais são os benefícios reais de fazer uso desta ferramenta. Neste trabalho serão abordados os problemas existentes na automação de testes ponta a ponta, e os recursos disponíveis nas ferramentas. Dessa forma, objetiva-se justificar a utilização do Cucumber como uma melhor alternativa no processo de teste de software.

Palavras-chave: Cucumber, Testes ponta a ponta, Automação de Testes, Selenium

Abstract

This article addresses the study of Selenium and Cucumber test automation tools, with the objective of distinguishing their applications through a documentary study, and showing the advantages and disadvantages of using each tool. The motivation of the work is to demonstrate the evolution of software testing tools at the end-to-end, End-to-End (E2E) level, evaluating the Cucumber solution, presenting the real benefits of making use of this tool. In this work, the existing problems in end-to-end test automation will be addressed, as well as the resources available in the tools. Thus, the objective is to justify the use of Cucumber as a better alternative in software testing.

Keywords: Cucumber, End-to-End Testing, Test Automation, Selenium

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

ÍNDICE DE FIGURAS

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

ÍNDICE DE TABELAS

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

SUMÁRIO

1. Introdução e Motivação

Na atualidade os softwares estão cada vez mais presentes na sociedade, sendo utilizado de forma diária, responsáveis por automatizar diversas atividades do dia a dia e a dar suporte a diversas organizações de diferentes ramificações. Cada vez mais há uma grande preocupação em relação a qualidade dos serviços oferecidos por esses softwares, com isso surge uma maior preocupação em entregar os softwares com a melhor qualidade possível e para tal é utilizado ferramentas de teste de software.

> "Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais" [Pnressman, 2016].

Devido ao crescimento da quantidade de ferramentas de automação de testes e a sua crescente evolução, não é muito trivial avaliar com clareza quais ferramentas devem ser utilizadas em projetos de software. Cada ferramenta tem uma nuance diferente e servem melhores a alguns propósitos do que a outros. Sendo assim é necessário muita pesquisa comparativa para que seja visto qual ferramenta se encaixa melhor em determinados contextos. (SOMMERVILLE, 2011)

Durante o desenvolvimento de programas são acrescentadas etapas de teste de softwares antes de ser concluído e após sua finalização de modo a verificar se está de acordo com o que foi requisitado por parte do cliente. Um teste pode, além de agregar qualidade ao produto final, prevenir grandes perdas como: tempo, dinheiro e diversos outros tipos de danos. Os sistemas que são classificados como aplicações WEB, exigem um tipo de teste mais específico, com inserção de valores repetidamente, os quais são testados usando as métricas de automação de testes.

O presente trabalho fará uma comparação entre ferramentas de testes de ponta a ponta de diferentes gerações, para que se entenda quais são os benefícios adquiridos com a evolução das mesmas. Assim espera-se contribuir para a análise das ferramentas disponíveis para automação de testes de ponta a ponta.

1.1. Justificativa

Para garantir a qualidade satisfatória de um software é necessário realizar testes no mesmo e para otimizar o tempo, é de suma importância o uso de teste de software automatizado, sendo importante escolher a ferramenta que mais se adequa ao projeto que está sendo desenvolvido. Através do estudo das ferramentas Cucumber e o Selenium o leitor deste artigo poderá ter informações necessárias relativas a aplicação das ferramentas abordadas.

1.2. Objetivo Geral

Este trabalho tem como objetivo analisar e comparar as ferramentas de automação de testes de ponta a ponta, mostrando suas diferenças e vantagens de utilização. As ferramentas escolhidas são de gerações diferentes e se propõem a resolver o mesmo problema de formas diferentes. De forma que esse trabalho também é dedicado a mostrar a evolução das ferramentas em estudo, Cucumber e Selenium fazendo um comparativo entre ambas assim como analisar o comportamento das ferramentas quando utilizadas em forma conjunta.

1.3. Objetivos Específicos

- Abordar o processo de testes e como a automação de testes funciona;
- Definir as características de ferramentas de automação de testes de ponta a ponta.
- Descrever as características de cada ferramenta de teste e
- Realizar uma análise quanto a configuração e funcionamento das ferramentas de teste: Cucumber e Selenium

1.4. Metodologia

A metodologia de pesquisa que será utilizada para elaboração deste trabalho será: exploratória e descritiva, sob o método hipotético-dedutivo, levando em consideração uma abordagem qualitativa. Para este trabalho foi realizado uma pesquisa com base em procedimentos bibliográficos e documentais. Com isso serão abordados

outros trabalhos científicos, documentação oficial das ferramentas, artigos e livros. Para que como resultado se obtenha uma comparação entre as duas principais ferramentas propostas para estudo que é o Selenium e o Cucumber, elucidando vantagens e desvantagens de cada uma. Contribuindo assim para futuros trabalhos que venham utilizar as ferramentas estudadas, bem como contribuir para que sejam feitas escolhas mais assertivas no uso das ferramentas de automação de testes apresentadas.

Foi adotado uma análise documental das ferramentas em estudo, dando maior foco nas características que são primordiais para as ferramentas de automação de testes. Diante dessa análise será feita uma comparação entre as ferramentas, que ressaltará suas características e contextos de utilização. Para efeitos de análise, neste estudo serão consideradas somente as características que são crucias para a montagem de um bom projeto de automação de testes.

2. Qualidade de Software

A qualidade de software é definida como um campo de estudo e prática que descreve os atributos desejáveis dos produtos de software. Há duas abordagens principais para a qualidade de software: gerenciamento de defeitos e atributos de qualidade. (SOMMERVILLE, 2011, p. 457) atribui em sua obra “a avaliação da qualidade de software é um processo subjetivo, em que a equipe de gerenciamento de qualidade precisa usar seu julgamento para decidir se foi alcançado um nível aceitável de qualidade”.

Uma boa definição da qualidade de software mais simplista foi dada por (PRESSMAN; MAXIM, 2016, p. 360) em sua obra.

> “No sentido mais geral, a qualidade de software pode ser definida como: uma gestão de qualidade efetiva aplicada de modo a criar um produto útil que forneça valor mensurável para aqueles que o produzem e para aqueles que o utilizam.”

Um software é produzido de acordo com as especificações estabelecidas pelo cliente no início, logo podemos crer que para que seja considerado de qualidade é preciso que ele entregue valor ao cliente que o utilizará de maneira que cumpra com seu propósito e também seja confiável.

3. Testes de Softwares

O Teste de Software é um método para verificar se o produto de software real atende aos requisitos esperados e para garantir que o produto de software esteja livre de defeitos. Envolve a execução de componentes de software/sistema usando ferramentas manuais ou automatizadas para avaliar uma ou mais propriedades de interesse. O objetivo deste etapa é identificar erros, lacunas ou requisitos ausentes em contraste com os requisitos reais.

Para discutir sobre os testes de software é importante primeiro conceituá-lo., segundo um conceito trazido por (PRESSMAN; MAXIM, 2016), “O teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente”. O conceito se estende um pouco mais em outros trabalhos, mas de forma resumida, é um conjunto de atividades do processo de desenvolvimento de software, que requer planejamento e execução sistemática

3.1. Processo de Teste

Pautado pelas considerações em relação ao planejamento e à execução das atividades de teste de software, este pode ser estruturado em fases sequenciais (tabela 1). Nesta abordagem (Whittaker, 2000), cada fase trata de problemas relacionados e o resultado de seu processamento serve como base para a fase seguinte.

1: Modelagem do Ambiente de Software
2: Seleção dos Cenários de Teste
3: Execução e Avaliação dos Cenários de Teste
4: Medição do Progresso de Teste

Tabela 1 - Processo de teste
Fonte: Próprio autor

O primeiro passo é a modelagem do ambiente de software (tabela 1). Nesta fase, é preciso identificar e simular as interfaces que o sistema de software utiliza bem como enumerar as possíveis entradas de dados aplicadas a estas interfaces. Cada ambiente de aplicação pode resultar num número significativo de interações a testar. Um modelo é utilizado para descrever estas interações. O segundo passo é a seleção dos cenários de teste. Os parâmetros normalmente empregados na definição dos casos de teste são a cobertura do código e a cobertura do domínio dos dados de entrada. Mas essas coberturas analisadas de forma isolada não são suficientes. A sequência de execução de comandos de código, a sequencia de aplicação de entradas e o ambiente de teste (configurações de hardware e software) também influenciam na formação dos cenários de teste, (Whittaker, 2000).

O terceiro passo corresponde à execução e avaliação dos cenários de teste (tabela 1). A etapa inicial desta fase consiste em converter os cenários de teste numa forma executável que simula as ações dos usuários. A aplicação dos cenários pode ser manual ou automatizada. O último passo equivale à capacidade de medição do progresso de teste (tabela 1). (Whittaker, 2000).

3.2. Caso de teste

Um caso de teste pode ser definido como um documento que descreve as etapas a serem seguidas para realização de um teste. Desde os dados de entrada, a atividade em execução e o resultado de saída. É utilizado para observar se o resultado de saída é realmente o esperado. Em caso positivo o teste foi realizado com êxito (INTHURN, 2001).

> "Técnicas de teste de software consistem em elaborar casos de teste capazes de varrer o software a procura de erros ainda não constatados. As atividades de testes não visam culpar alguém por falhas ou, ainda, testar o software com finalidade destrutiva. Os testes são realizados com o objetivo de encontrar as falhas e, posteriormente, corrigi-las, garantindo assim a qualidade do mesmo" (PRESSMAN, 1995)

O caso de teste serve para verificar se o sistema segue as definições impostas no projeto e se o mesmo foi devidamente construído. Segundo Bartié (2002), o caso de

teste é o documento que registra todo o planejamento dos testes, nele devem ser abordados os seguintes itens:

- identificação das condições de testes;
- identificação do que será testado;
- definição de cada caso de teste identificado;
- detalhamento das classes de dados de entrada;
- detalhamento da saída gerada;
- responsáveis pela atividade de teste;
- definição de como será realizada a bateria de testes;
- cronograma das atividades.

A figura 1 mostra um caso de teste realizado em *Test Driven Development* (TDD) .ou em português Desenvolvimento guiado por testes

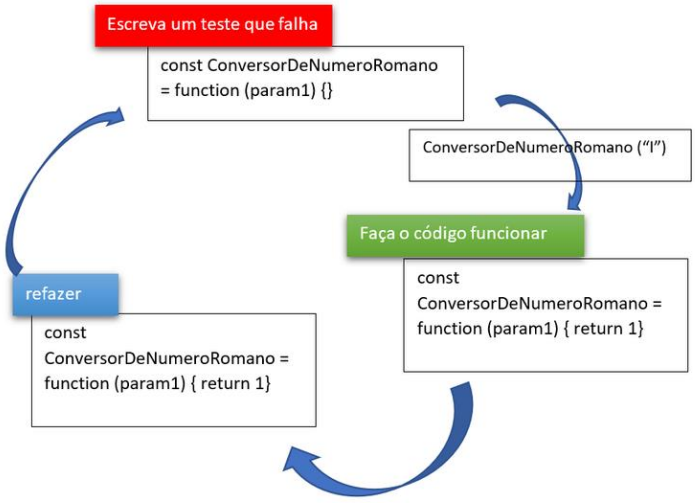

Figura 1 - Caso de teste

4. Tipos de Testes

Classificação dos testes em função do estágio do ciclo de vida do produto:

4.1. Teste de componente

Esse tipo de teste procura defeitos e verifica o funcionamento do software (ex: módulos, programas, objetos, classes, etc.) que são testáveis separadamente. Pode ser feito isolado do resto do sistema, dependendo do contexto do ciclo de desenvolvimento e do sistema. Há a possibilidade de ser utilizado simulador, pode incluir teste de funcionalidade e características específicas não-funcionais.

4.2. Teste de integração

É um tipo de teste de software no qual as diferentes unidades, módulos ou componentes de um aplicativo de software são testados como uma entidade combinada. No entanto, esses módulos podem ser codificados por diferentes programadores. O objetivo do teste de integração é testar as interfaces entre os módulos e expor quaisquer defeitos que possam surgir quando esses componentes são integrados e precisam interagir entre si.

4.3. Teste de sistema

Trata o comportamento de todo do sistema ou produto definido pelo escopo de um projeto ou programa de desenvolvimento. Envolve os requisitos funcionais e não-funcionais, no teste de sistema, o ambiente de teste deve ser o mais semelhante possível ao ambiente de produção, a fim de maximizar a identificação de falhas específicas de ambiente.

4.4. Teste de aceite

Um teste de aceitação é uma descrição formal do comportamento de um produto de software, geralmente expresso como um exemplo ou cenário de uso. Várias notações e abordagens diferentes foram propostas para tais exemplos ou cenários. Em muitos casos, o objetivo é que seja possível automatizar a execução de tais testes por uma ferramenta de software, seja ad hoc para a equipe de desenvolvimento ou pronta para uso. Semelhante a um teste de unidade, um teste de aceitação geralmente tem um resultado binário, aprovado ou reprovado. Uma falha sugere, mas não prova, a presença de um defeito no produto.

4.5. Teste de manutenção.

O teste de manutenção também é conhecido como teste de software pós-lançamento. Este é um tipo de teste de software que ocorre quando o software foi lançado em produção e quaisquer alterações foram feitas para corrigir bugs ou adicionar novos recursos ao sistema existente.

Classificação dos testes em função do objetivo do teste:

4.6. Teste Funcional

No teste funcional, a seleção dos cenários de teste se baseia nas características reveladas pela especificação do sistema em teste ou pelo ambiente operacional. A estrutura do código fonte ou as estruturas de dados internas (não expostas) não servem como referencial para elaboração dos testes. A ideia principal de um teste funcional é ignorar os mecanismos internos de um sistema ou de um componente e focar somente nas saídas geradas em resposta às entradas selecionadas e às condições de execução do software (Gao et al., 2003). Desta forma, mesmo nos casos em que o código fonte não esteja disponível, é possível testar um software e verificar se ele atende aos requisitos do usuário.

4.7. Teste não funcionais

Não funcional é um tipo de teste de software executado para verificar os requisitos não funcionais do aplicativo. Ele verifica se o comportamento do sistema está de acordo com o requisito ou não. Ele testa todos os aspectos que não são testados no teste funcional. O teste não funcional é definido como um tipo de teste de software para verificar os aspectos não funcionais de um aplicativo de software. Ele é projetado para testar a prontidão de um sistema de acordo com parâmetros não funcionais que nunca

são tratados pelo teste funcional. O teste não funcional é tão importante quanto o teste funcional.

4.8. Teste estrutural

Teste estrutural é um tipo de teste de software que utiliza o design interno do software para teste ou em outras palavras o teste de software que é realizado pela equipe que conhece a fase de desenvolvimento do software, é conhecido como teste estrutural.

O teste estrutural está basicamente relacionado ao design interno e à implementação do software, ou seja, envolve os membros da equipe de desenvolvimento na equipe de teste. Ele basicamente testa diferentes aspectos do software de acordo com seus tipos. O teste estrutural é exatamente o oposto do teste comportamental.

4.9. Teste de regressão

O teste de regressão é uma técnica do teste de software que consiste na aplicação de versões mais recente do software, para garantir que não surgiram novos defeitos em componentes já analisados. De acordo com (BASTOS et al., 2007) a realização de testes regressivos envolve re-testar segmentos já testados após a realização de uma mudança em uma outra parte do software os testes de regressão visam garantir a integridade do software após a realização de medicações.

4.10. Testes de Ponta a Ponta

Teste de ponta a ponta em inglês conhecido como *end-to-end testing* é o processo de avaliação das funções do software por meio da revisão de todo o fluxo de trabalho do aplicativo do início ao fim. Testes de ponta a ponta bem-sucedidos imitam como o software opera na vida real, executando cenários comuns de usuário e identificando quaisquer erros ou outros defeitos. Essa técnica geralmente é uma das etapas finais para testar aplicativos de software porque combina todos os elementos de codificação individuais e funções de software em um único teste. (LIMA, 2020)

5. Ferramentas de Teste de Software

5.1. Selenium

É um conjunto de ferramentas de código aberto multiplataforma usado para automatizar teste nos navegadores web, suporta diversas linguagens de programação tais como: Ruby, Java, NodeJS, PHP, Perl, Python e C#. Suporta também diversos navegadores Web.

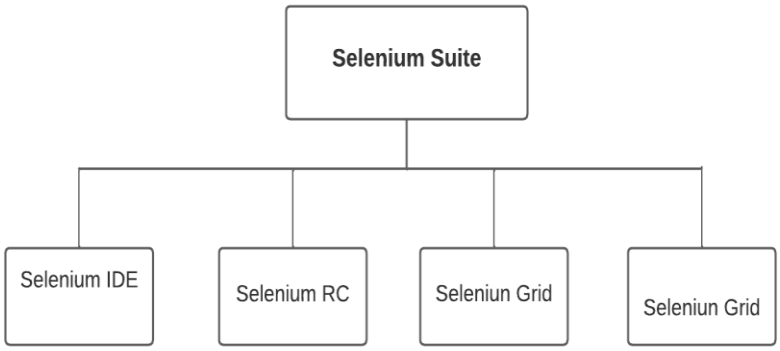

Figura 2 - Ecossistema do Selenium

Fonte: próprio autor

5.1.1. Selenium IDE

O Selenium IDE permite que um usuário ou desenvolvedor de casos de teste crie os casos de teste e suítes de teste e os edite posteriormente de acordo com seus requisitos. O ambiente de desenvolvimento também oferece a capacidade de converter casos de teste para diferentes linguagens de programação, o que facilita para o usuário e não impõe a necessidade de conhecer uma linguagem de programação específica.

5.1.2. Selenium WebDriver

A ferramenta Selenium WebDriver é usada para automatizar o teste de aplicações Web para verificar se está de acordo com os requisitos estabelecidos no tocante ao funcionamento da aplicação. Ele suporta muitos navegadores, como Firefox, Chrome, IE e Safari. No entanto, usando o Selenium WebDriver, podemos automatizar os testes

apenas para aplicativos da web. Ele não se qualifica para aplicativos baseados em janela.

5.1.3. Selenium Grid

Selenium Grid é uma ferramenta de teste que nos permite executar testes em diferentes máquinas em diferentes navegadores. Faz parte do Selenium Suite, pode-se conectar a ele com o Selenium Remote especificando o navegador, a versão do navegador e o sistema operacional desejado por meio dos recursos do Selenium Remote.

5.2. Cucumber

Cucumber é uma ferramenta de teste que suporta o BDD (Behavior Driven Development). Oferece uma maneira de escrever testes de uma forma mais compreensível de modo que as pessoas possam entender, independentemente de seu conhecimento técnico. No BDD, os usuários (analistas de negócios, proprietários de produtos) primeiro escrevem cenários ou testes de aceitação que descrevem o comportamento do sistema da perspectiva do cliente, para revisão e aprovação pelos proprietários do produto antes que os desenvolvedores escrevam seus códigos. O framework Cucumber usa a linguagem de programação Ruby.

De modo a auxiliar o desenvolvedor a ter mais agilidade e qualidade no código é necessário uma ferramenta de edição apropriada, tais como: Aptana, RubyMine e o Katalon Studio, todos eles suportam o Cucumber.

6. Comparações entre as ferramentas

Cucumber e Selenium são duas ferramentas open source, Cucumber pode ser integrado com Selenium para realizar testes de softwares. Ambas ferramentas funcionam de forma similar e podem trabalhar juntas para automação de testes.

Ponto de diferença	Selenium	Cucumber
Framework de Automação	**Sim**	**Não**
Tipo de Ferramenta	**Automação de Navegador**	**BDD**
Escrito em	**Java**	**Ruby**
Script writing	**Linguagem de Programação**	**Gherkin**
Ferramenta de teste de automação	**Funcional e Performance**	**Não**
legibilidade	**apenas técnico**	**todos**

Tabela 1 – Cucumber vs Selenium

Fonte: Próprio Autor

Selenium é um software para automação de teste de softwares, já o Cucumber é responsável por ler as especificações executáveis escritas em texto simples e converte em alguma linguagem de programação. Cucumber é uma ferramenta para BDD, não é possível automatizar um navegador apenas com Cucumber para tal é necessário o uso do Cucumber com o Selenium. Selenium foi escrito na linguagem Java e Cucumber em Ruby.

Para elaborar os script em Selenium é utilizada a linguagem Java, já os scripts em Cucumber é preciso escrever na linguagem Gherkin. O Gherkin é um dos elementos principais quando se trata de BDD em automação. Sua função é padronizar a forma de descrever especificações de cenários, baseado na regra de negócio. Uma vez escrito em Gherkin é possível gerar um código correspondente em diversas linguagens, tais como: Java, Python entre outras.

Legibilidade, apenas pessoas com conhecimento técnico podem entender Selenium no entanto Cucumber torna-se mais fácil a compreensão possibilitando que

qualquer pessoa consiga entender, seja o Tester de testes manuais ou gerente de projeto. A performance do script em Selenium é bastante rápida, no entanto se estiver integrado com Cucumber o tempo de execução será maior fazendo com que o script seja mais lento. Com selenium é possível realizar diversos tipos de teste de software, com Cucumber é possível basicamente implementar o BDD.

Ponto de diferença	Selenium	Cucumber
Instalação	Difícil: Requer conhecimento técnico	Fácil: Simples instalação
Confiabilidade	Maior confiabilidade	há uma menor confiabilidade em relação ao Selenium
Línguagem de Programação	Phyton, Java, C3, JavaScript Ruby, PHP, Perl, Kotlin	Python, Java, .Net, JavaScript, Ruby, PHP, Perl Groovy, Clojure, Gosu, Lua, Jython, C++, Tcl
Erro de sintaxe	Destacado imediatamente	não destaca o erro
Revisão de Script	Técnico	Qualquer
Declarações condicionais	é possível realizar	Não é possível realizar

Tabela 2 - Análise Cucumber vs Selenium

Fonte: Próprio Autor

No tocante a instalação o Selenium requer um pouco mais de conhecimentos técnico sendo mais trabalhosa a instalação comparando-a com a do Cucumber, a confiabilidade no Selenium acaba sendo mais forte pois há uma ampla comunidade que dá um suporte a ferramenta, Cucumber acaba tendo uma comunidade bem menor. No que se refere a linguagem de programação, ambas ferramentas suportam uma grande variedade de linguagens, não havendo muita diferença de uma para outra. Quanto ao tempo de documentação ao usar o Selenium é preciso escrever os casos de teste de forma manual e através deles é automatizado os casos de teste em uma língua de programação, utilizando o Cucumber esse processo é realizado de forma mais rápida utilizando menos tempo comparado ao Selenium, pois é possível escrevê-los diretamente na linguagem Gherkin.

Erros na digitação de código de programação ocorrem com uma certa frequência no dia a dia de quem trabalha com programação, ferramentas que indicam erro se sintaxe acabam facilitando muito, neste aspecto Selenium é mais interessante pois a ferramenta verifica e exibe erro de sintaxe, os plugins do Cucumber não são tão eficientes nesse sentido. Para que se possa compreender os Scripts que são gerados através do Selenium é necessário um conhecimento técnico de linguagem de programação e da ferramenta em uso, neste aspecto o Cucumber acaba tendo uma vantagem pois a linguagem Gherkin pode ser facilmente compreendida por todos envolvidos no projeto.

> “Cucumber é principalmente uma ferramenta de colaboração com a intenção de criar um entendimento comum entre todos os membros da equipe. No Cucumber as funcionalidades devem ser escritas antes do desenvolvimento. Quando se trabalha com os exemplos de escrita do BDD, testes de regressão são uma consequência, e não a atividade principal.” (STENBERG, 2015).

(TORRES , 2020) argumenta que o Gherkin usa um conjunto de palavras-chave para dar significado às especificações junto ao Cucumber desta forma torna-se mais simples a laboração dos scripts, na imagem 2 o autor mostra cenário de teste usando o Cucumber como ” (TORRES, 2020)

```
1  Cenário: Confirmar se o usuário que acessou é o mesmo logado
2    Dado que o usuário tem acesso ao sistema
3    Quando os dados são informados
4    Então o nome do usuário exibido é "Wagner Silva Torres"
```

Figura 3 - Caso de teste com Gherkin
Fonte: (TORRES , 2020)

Uma vez utilizando a linguagem Gherkin, linguagem esse usada no Cucumber não há a possibilidade de elaborar declarações condicionais (if/else) sendo assim uma desvantagem em relação ao Selenium que por sua vez é possível devido a utilização direta de uma linguagem de programação ao criar os casos de teste automatizados.

No artigo publicado no Salão do Conhecimento, Evento: XXV Jornada de Pesquisa 2020, os autores, (ROLLWAGEN) André Fernando, (AMARAL) Joseane, Eduardo Rodrigue e (DALLASEN) Ricardo Vanni apresentam um estudo comparativo das ferramentas automação de testes de software Badboy, Selenium IDE e Sikuli. Os autores concluem que:

O Selenium IDE foi a ferramenta com o melhor desempenho nos testes de tempo e funcionalidades. Sem dúvida além de ser descrita como uma IDE, a ferramenta é

recomendada para qualquer automação de teste, ela se destaca em relação ao BadBoy (É uma ferramenta de teste de software projetada para ajudá-lo a testar e desenvolver sites on-line complexos e dinâmicos) e Sikuli (ferramenta que utiliza seu computador, teclado, mouse e monitor para executar automaticamente qualquer tarefa que um ser humano pode realizar no computador em seu dia-a-dia) por conter inúmeras opções de tratamento de HTML e JavaScript, o que é fundamental em testes de aplicações WEB

6.1. Execução em Sistemas Operacionais

Atualmente diversos sistemas operacionais são utilizados nos computadores, sendo assim é de suma importância saber se a ferramenta que será utilizada suporta o sistema operacional usado para execução. Tendo em vista que atualmente há uma ampla variedade de sistemas operacionais e distribuições diferentes sendo utilizadas é de suma importância os testes feitos possam ser executados pela ferramenta em uma boa variedade de sistemas operacionais. Os sistemas operacionais que serão considerados neste estudo serão: Microsoft Windows, Apple macOS (ou macOS), Android OS (ou Android), Apple iOS (ou iOS) e Linux.

Tendo em vista que o Cucumber foi implementado em Ruby isso implica dizer que qualquer plataforma que possua suporte a linguagem Ruby dará suporte também ao Cucumber.

	Windows	Linux	macOS	Android	iOS
Selenium WebDriver	SIM	SIM	SIM	SIM	SIM
Cucumber	SIM	SIM	SIM	NÃO	NÃO

Tabela 3 - Plataformas
Fonte: Próprio Autor

A ferramenta Selenium é suportada por todas as plataformas apresentadas na tabela 3, apresentando assim uma vantagem em relação ao Cucumber que não suporta os sistemas operacionais mobile Adroid e iOs.

6.2. Execução em Navegadores

	Chrome	Edge	FireFox	Safari	Opera
Selenium WebDriver	SIM	SIM	SIM	SIM	SIM
Cucumber	SIM	SIM	SIM	NÃO	SIM

Tabela 4 - Comparação de uso entre os principais navegadores.
Fonte: Próprio Autor

Em relação a essa característica vemos por meio da Tabela 4 que as ferramentas não possuem tantas diferenças dentro desse contexto, podem ser executadas em diversos navegadores, um destaque apenas no Selenium que pode ser executado no Safari e o Cucumber não.

7. CONSIDERAÇÕES FINAIS

Adotar a automação de testes agrega diversas vantagens, reduz os gargalos e aumenta a eficiência, entender qual ferramenta de automação utilizar é de suma importância, as ferramentas estudadas no presente artigo possuem possuem aplicações distintas e muitas diferenças dentro dos critérios analisados. O Selenium possui uma aplicação mais ampla podendo ser executado em diferentes ambientes, sendo utilizado em uma maior variedade de sistemas operacionais e navegadores, inclusive podendo ser aplicado em automação de testes em aparelhos móveis. Já o Cucumber para realizar testes em dispositivos móveis requer um intermediador, podendo-se usar o Calabash que permite a execução dos testes escritos em Cucumber em dispositivos móveis e emuladores/simuladores, tanto para Android quanto iOS.

O Selenium atualmente é umas das ferramentas mais conhecidas para realização de teste de software devido a uma maior liberdade na utilização das linguagens de programação e tecnologias que podem ser utilizadas para compor o projeto de automação. Em contrapartida o Selenium também possui desvantagens em outros critérios como o de configuração do projeto por ter uma configuração mais complexa. Também possui menos recursos para auxiliar na escrita de testes e para identificação de problemas nos relatórios de teste.

O Cucumber permite de forma simplificada que os envolvidos no projeto possam participar das especificações para os testes do sistema através da linguagem Gherkin, sendo uma vantagem frente ao Selenium que requer uma linguagem de programação.

Para realizar teste automatizado ponto a ponto com Cucumber é preciso de uma outra ferramenta de teste de software que possa interagir com o mesmo, pois o Cucumber não consegue automatizar o navegador Web. Contudo, através das pesquisas abordando suas vantagens e desvantagens, foi verificado que o uso da ferramenta Cucumber, apesar de suas limitações a ferramenta consegue fornecer recursos bem mais atrativos que compensam seus pontos fracos, sendo mais interessante pela simplicidade na elaboração dos casos de teste e melhor compreensão por parte de todos envolvidos no projeto.

Cypress é um framework de testes, de código aberto e de fácil configuração que pode ser usado para realizar os seguintes testes: End-to-End, Interface de Usuário, Pl's, componentes, unidade. Como planejamentos para trabalhos futuros, pode-se estender o estudo da ferramenta Cucumber, fazendo um comparativo com a ferramenta Cypress tendo em vista que Cypress também é utilizado para realizar testes ponto a ponto, usando como base as características e vantagens da ferramenta Cucumber abordadas neste trabalho.

8. REFERÊNCIAS

BASTOS, A. et al**. Base de Conhecimento em teste de software**. 2a edição. ed. [S.l.]: São Paulo: Editora Martins, 2007. Citado 2 vezes nas páginas 15 e 24.

INTHURN, Cândida. **Qualidade & teste de software**. [S.l.]: Florianópolis, SC : Visual Books, [C2001]., 2001.

LIMA, Fábio - ACERVO LIMA, 2020. Disponível em:< https://acervolima.com/diferencas-entre-testes-funcionais-e-nao-funcionais//> .Acesso em: 10 de setembro de 2022.

3 PRESSMAN, R. **Software Engineering: A Practitioner's Approach**, sétima edição, 7, Mc Graw Hill, 2016.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software-8a Edição**. [S.l.]: McGraw Hill Brasil, 2016.

SOMMERVILLE, I. **Engenharia de Software**, 9a. [S.l.]: Pearson Education do Brasil, 2011.

SELENIUMHQ. **Selenium Documentation - Browser Automation**. [S.l.], 2019. Disponível em:<https://cucumber.io/docs/guides/browser-automation/> .Acesso em: 10 de setembro de 2022.

STENBERG, Jan.Info - **A ferramenta de BDD Cucumber não é uma ferramenta de teste,** 2015**.**. Disponível https://www.infoq.com/br/news/2015/07/bdd-cucumber-testing/. Acesso em: 05 de setembro de 2022.

Sem autor. **Gherkin Reference.** © 2019 Disponível (https://cucumber.io/docs/gherkin/reference>. Acesso em: 05 de setembro de 2022.

TORRES, Wagner Silva. Revista TSPI online - **Gherkin: o dia em que entendi que estava escrevendo errado.** 2020. Disponível em:<https://medium.com/revista-tspi/gherkin-o-dia-em-que-entendi-que-estava-escrevendo-errado-220a84520819>. Acesso em: 02 de setembro de 2022.