PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Estudo comparativo de ferramentas de automação de testes em ambiente web

Diogo Faria Pereira Chehab

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Engenharia da Computação

Rio de Janeiro, junho de 2023

Diogo Faria Pereira Chehab

Estudo comparativo de ferramentas de automação de testes em ambiente web

Relatório de Projeto Final, apresentado ao programa de Engenharia da Computação da PUC-Rio como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Waldemar Celes

Rio de Janeiro

junho de 2023.

"Sendo engraçado ou não, quando você sorrir tem a chance de ser feliz."

(One Piece - Eiichiro Oda)

Agradecimentos

Aos meus pais, Renata e Murilo Chehab, ao meu irmão, João Pedro Chehab, e meu amor, Catarina Sabbadim, por todo apoio incentivo e apoio que foram necessários para a realização deste trabalho.

Resumo

Chehab, Diogo Faria Pereira. Celes, Waldemar. Estudo comparativo de ferramentas de automação de testes em ambiente web. Rio de Janeiro, 2023. 41 p. Relatório Final de Projeto de Conclusão de Curso – Centro Técnico Científico – CTC, Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro.

Dado o aumento da quantidade de ferramentas de automação disponíveis no mercado, juntamente com a segurança e desempenho melhorando cada vez mais, muitas empresas reconhecem que testes de software são cada vez mais necessários para manter a qualidade do software e o aumento de confiabilidade, sendo aspectos importantes dentro do mercado de tecnologia. Desta forma, o presente trabalho objetiva avaliar as ferramentas de automação de teste de software, buscando realizar um estudo comparativo entre elas a fim de identificar seus pontos de importância a serem considerados durante a escolha de ferramentas a serem implementadas em um projeto. O levantamento de dados ocorreu por meio de pesquisa e experiência do autor ao utilizar as ferramentas. Neste estudo foram utilizadas quatro ferramentas de automação, onde seus desempenhos foram comparados ao serem utilizadas em um ambiente web. Com as ferramentas selecionadas, foi realizada uma análise comparativa, buscando seus pontos de relevância.

Palavras-chave

Teste de software, automação de testes, qualidade de software

Abstract

Chehab, Diogo Faria Pereira. Celes, Waldemar. Comparative study between test automation tools in a web environment. Rio de Janeiro, ano. 41 p. Relatório Final de Projeto de Conclusão de Curso – Centro Técnico Científico – CTC, Departamento de Informática. Pontifícia Universidade Católica do Rio de Janeiro. Pontifícia Universidade Católica do Rio de Janeiro.

Given the increasing amount of automation tools available on the market, along with security and performance improving more and more, many companies recognize that software testing is increasingly necessary to maintain software quality and increase reliability, being important aspects within the technology market. In this way, this work aims to evaluate the software test automation tools, seeking to carry out a comparative study among them to identify their points of importance to be considered during the choice of tools to be implemented in a

project. Data collection occurred through research and the author’s experience when using the tools. This study used four automation tools, and their performances were compared when used in a web environment. With the selected tools, a comparative analysis was carried out, seeking their points of relevance.

Keywords
Software testing, test automation, software quality

Sumário

Lista de Figuras

1. Introdução

Devido a agilidade no desenvolvimento de software, por conta do surgimento de frameworks, bibliotecas prontas e a evolução de linguagens de programação, o processo de desenvolvimento está se tornando cada vez mais rápido e eficiente, possibilitando a entrega das soluções em um menor tempo. Porém, com isso, o sistema pode apresentar bugs e falhas, além da qualidade poder ficar em segundo plano (ROLLWAGEN et al, 2020).

Dentro desse contexto, os testes se tornam cada vez mais essenciais, buscando aumentar a eficiência e qualidade, e diminuindo a taxa de erros durante a execução desses sistemas (PESSOA, 2020). Etapas de teste acrescentadas dentro do processo de desenvolvimento podem agregar, além da qualidade, a prevenção de perdas monetárias, de tempo e outros tipos de dano. Os testes em um projeto, de acordo com Barbosa e Torres (2011), podem tomar cerca de 40% do tempo planejado. Porém, se forem aplicados apenas durante a implantação, podem ocasionar em um aumento de 60% dos custos, mostrando assim sua importância.

A atividade de teste de forma manual pode demandar uma atenção extra do testador, pois ele deve observar e testar cada parte do sistema, e, em casos de testes maiores, isso pode tornar a experiência do testador mais cansativa e pode levar a erros humanos. Com isso, uma opção viável é a implementação da automação de testes, onde pode-se utilizar o apoio de ferramentas de automação que são capazes de tornar o processo mais rápido. Essas ferramentas fazem uso de scripts automatizados, onde o testador cria esses scripts utilizando uma linguagem de programação, podendo replicá-los sempre que houver necessidade (ROLLWAGEN et al, 2020).

Outra vantagem da automação dos testes está relacionada com a diminuição do tempo de execução desses e também com a diminuição do esforço necessário para realiza-los; ainda, testes automatizados podem garantir maior cobertura do código testado. Pontos importantes sobre essas ferramentas estão relacionados como, por exemplo, são capazes de gerar relatórios detalhados sobre os erros e defeitos, se possui histórico de todos os testes executados e se são capazes de importar e exportar conteúdo (ROLLWAGEN et al, 2020).

De acordo com Pessoa (2020), as ferramentas de teste de software auxiliam empresas dentro do mercado tecnológico, onde a automação de testes é capaz de aumentar a produtividade e possivelmente até o lucro dessas empresas. Com isso, as empresas precisam escolher, dentre as ferramentas disponíveis, quais se encaixam melhor no contexto do sistema e do projeto, e que sejam capazes de

cobrir potenciais riscos, defeitos e erros, analisando assim técnica, tempo e recursos disponíveis. Como existem diversas ferramentas, o processo de escolha da ferramenta ideal para automação de testes dentro de um projeto pode ser difícil para o usuário escolher as mais adequadas.

Este trabalho tem como objetivo realizar uma coleta e fazer um estudo comparativo entre as ferramentas de automação de testes para uma aplicação Web, buscando avaliar suas propriedades e funcionalidades, identificando seus pontos positivos e negativos dentro do modelo de testes. Desta forma, espera-se que o processo de escolha de uma ferramenta certa e que se adeque da melhor forma dentro de empresas e projetos seja facilitado, buscando assim agregar qualidade nos testes de software.

2. Situação Atual

Os testes automatizados estão se tornando cada vez mais populares e importantes na indústria de desenvolvimento de software. Atualmente, as ferramentas de testes são amplamente utilizadas em muitas empresas de tecnologia como uma forma de garantir a qualidade do software e reduzir custos em relação aos testes manuais. Isso ocorre porque os testes automatizados podem ser executados repetidamente sem intervenção humana, o que os torna muito mais rápidos e eficientes. Além disso, eles podem ser executados em paralelo em vários ambientes, o que permite testar uma ampla variedade de cenários de teste em pouco tempo.

Com o aumento da adoção de práticas ágeis de desenvolvimento de software, como o desenvolvimento contínuo e a integração contínua, os testes automatizados se tornaram ainda mais importantes. Isso ocorre porque essas práticas exigem que o software seja testado constantemente e de forma rápida para garantir que as mudanças de código não causem problemas inesperados.

Além disso, com o surgimento de ferramentas e frameworks de teste automatizado mais avançados, como o Selenium e Cypress, os testes automatizados se tornaram mais acessíveis e fáceis de serem implementados. Isso permite que desenvolvedores e testadores escrevam e executem seus testes com mais facilidade e eficiência.

No entanto, apesar de todas as vantagens dos testes automatizados, ainda há desafios a serem superados, como a criação de testes robustos e confiáveis, a manutenção de conjuntos de testes grandes e complexos e a integração dos testes automatizados em pipelines de integração e entrega contínuas.

2.1. **Testes de software**

O teste de software é uma técnica que busca encontrar falhas e erros durante a execução de um programa. Ele é essencial dentro do ciclo de um desenvolvimento de software, pois é capaz de avaliar se o software está de acordo com os requisitos atribuídos e se está funcionando corretamente. Para Sommerville (2011), o teste de software possui como objetivo mostrar que um programa está fazendo o que é proposto e descobrir seus defeitos, buscando mostrar ao desenvolvedor e cliente que o software está atendendo aos requisitos. Para Andrade (2013), ele demonstra a confiabilidade e a garantia de qualidade do software.

De acordo com Pessoa (2020), as atividades de um teste de software se dividem em quatro etapas, que são: o planejamento de testes, o projeto dos casos de teste, a execução do teste e a avaliação dos resultados após a execução. Com isso, pode-se garantir que a ferramenta está funcionando conforme as suas especificações. Essas atividades estão incluídas dentro das fases de testes de unidade, integração e sistema. Através dessas atividades, busca-se encontrar o maior número de falhas realizando o menor trabalho possível.

O teste de software é um dos principais auxílios disponívies para a obtenção de qualidade durante a entrega do produto. Existem algumas empresas que não reconhecem a importância desse tipo de teste, considerando os mesmos apenas como atividades a serem aplicadas. Entretanto, os testes devem ser realizados durante todo o processo de desenvolvimento do software, garantindo qualidade no produto e no software que está sendo desenvolvido (PESSOA, 2020).

De acordo com Silva (2022), o teste de software é fundamental no processo de garantia de qualidade, buscando evitar, prevenir e achar falhas durante o processo, utilizando um método de validação e execução controlados através de um programa. As especificações precisam ser revisadas antes do teste ser realizado, pois é preciso obter a segurança de que o software irá funcionar conforme esperado. Além disso, é necessário elaborar planos e procedimentos que sejam eficazes, desenvolvendo o software de forma ordenada com os erros sendo identificados durante o processo.

Com isso, existem dois tipos de qualidade que devem ser verificadas, de acordo com Bartié (2002). O primeiro tipo é a qualidade dos processos, que é verificada quando testes são aplicados nos documentos gerados durante cada fase do processo de desenvolvimento, considerando que se apresentarem um número muito alto de defeitos é necessário revisá-lo e reconstruí-lo, para assim ser possível reavaliar a qualidade do processo. Esses são conhecidos como testes

de verificação. O segundo tipo de testes, que são os testes de validação, é relacionado à qualidade dos produtos de software, que pode ser garantida através de várias aplicações de teste em várias fases do desenvolvimento. Com esses testes é possível validar as estruturas internas e conexões com requisitos, verificando todas as interfaces de comunicação entre cada componente que faz parte do software (PESSOA, 2020).

2.1.1. Qualidade de software

A qualidade de software se refere a garantir a qualidade de um produto durante o processo de desenvolvimento do mesmo, de acordo com Silva (2022), onde procura-se verificar se o software está atendendo aos requisitos dos envolvidos, definidos a partir de qual tecnologia deve ser utilizada, características específicas e necessidades da organização. Com isso, o processo é focado nos artefatos e etapas, garantindo a conformidade e diminuindo a quantidade de defeitos.

Para Silva (2022), os testes verificam e validam os artefatos do software, prevenindo e eliminando defeitos, utilizando um extenso planejamento garantindo a qualidade da entrega total ou parcial. Isso é feito seguindo técnicas, padrões, modelos e procedimentos que buscam um software de qualidade e de constante otimização.

De acordo com Silva (2022), as atividades de teste dentro da qualidade de software possuem o objetivo de encontrar erros e buscam o controle de qualidade, onde os testes precisam ser planejados e conduzidos corretamente. Após isso, os erros e defeitos são entendidos e eliminados.

2.1.2. Técnicas de Teste

Como os testes de software possuem o objetivo de avaliar todos os aspectos de software, de acordo com Silva (2022), diferentes técnicas precisam ser aplicadas no processo de avaliação. Além disso, essas técnicas auxiliam na descoberta de erros relacionados a funcionamento, desempenho e comportamento.

1. Testes funcionais

O teste funcional envolve a avaliação de funções que um sistema é responsável por executar, considerando o comportamento do software. Com isso, técnicas caixa-preta podem ser utilizadas para verificar a funcionalidade do componente ou do sistema. Sua eficácia pode ser avaliada pela cobertura funcional, sendo uma medida expressa como porcentagem do tipo de elemento que devem ser cobertos pelos testes (ISTQB, 2018).

Para o ISTQB (2018), os tipos de teste se dividem em: teste funcional, teste não funcional, teste caixa-branca e testes relacionados à mudança. Esses testes são de extrema importância durante um planejamento de testes, e se diferem pelo nível e como os testes serão aplicados

Os requisitos para um teste funcional, de acordo com o ISTQB (2018), podem ser produtos de trabalho, como: especificações de requisitos, negócios, épicos, histórias de usuários, casos de uso ou especificações funcionais. Além disso, devem ser realizados em todos os níveis de teste, mudando somente seu objetivo.

Os testes funcionais não consideram a estrutura de controle, de acordo com Pressman e Maxim (2016), buscando encontrar erros nos seguintes tópicos:

- Erros de interface;
- Erros relacionados ao comportamento e de desempenho;
- Erros relacionados à inicialização e término;
- Funções ausentes ou erradas.
- Erros na estrutura dos dados ou erros durante o acesso a base de dados externas.

De acordo com Silva (2022), essa técnica possui como vantagem o fato de que o testador não precisa possuir conhecimentos relacionados à complexidade associadas à parte interna do software ou às tecnologias empregadas.

2. Testes não-funcionais

De acordo com o ISTQB (2018), os testes não funcionais são responsáveis pela avaliação das características de sistemas e/ou softwares, verificando a usabilidade, performance ou segurança. Além disso, de acordo com Wiest et al. (2021), os testes não funcionais podem ser aplicados em todos os níveis de teste, testando as qualidades técnicas do sistema, como testes de carga e segurança. É de extrema importância que os defeitos sejam descobertos no início do projeto para que não impactem seu sucesso durante a finalização.

A eficácia dos testes não-funcionais pode ser avaliada por meio da métrica de cobertura não funcional, que é expressa como uma porcentagem dos elementos não funcionais que foram testados completamente (ISTQB, 2018).

De acordo com o exposto pelo blog Vericode (2022), os testes não funcionais identificam se um sistema está de acordo com os requisitos necessários para tornar a experiência de uso do cliente mais agradável, procurando erros, ameaças e gargalos no processo.

Durante a execução dos testes não-funcionais, é necessário que o testador possua habilidades ou conhecimentos relacionados ao projeto ou a tecnologia

utilizada, pois isso pode facilitar durante a descoberta de erros e defeitos (ISTQB, 2018).

3. Testes caixa-branca

O teste caixa-branca se refere a realização de testes na estrutura interna ou implementação do sistema, incluindo código, arquitetura, fluxos de trabalho e dados dentro do sistema. Para implementação dos testes, o testador deve possuir habilidades e conhecimentos, buscando utilizar esses conhecimentos para interpretar corretamente os resultados (ISTQB, 2018).

Este tipo de teste, de acordo com Silva (2022), possui como foco testar o código-fonte do sistema. É necessário que o testador possua conhecimento das tecnologias que foram utilizadas durante o desenvolvimento do software, utilizando assim parâmetros de entradas e alcançando os resultados esperados.

A cobertura é medida através da cobertura estrutural, sendo expressa como uma porcentagem do elemento que irá ser coberto. Para o ISTQB (2018), durante o teste de integração de componentes, o teste caixa-branca pode se basear na arquitetura do sistema.

Para Pressman e Maxim (2016), casos de testes podem ser criados utilizando os parâmetros de:

- Caminhos independentes do módulo;
- Decisões lógicas;
- Ciclos;
- Estrutura dos dados internos.

4. Testes relacionados à mudança

Os testes de mudança estão relacionados a testes realizados quando há alteração dentro de um sistema, seja essa mudança com o objetivo de corrigir um defeito, ou para implementar uma funcionalidade nova. Com isso, o ambiente deve ser testado para que não haja consequências no resto do sistema (ISTQB, 2018). Os testes de mudança são divididos em:

- Testes de confirmação, que possui como objetivo identificar se o defeito foi corrigido com sucesso;
- Testes de regressão, que têm como intuito verificar se, após uma correção, não foram introduzidos defeitos em outras funcionalidades.

Ambos os testes devem ser realizados em todos os níveis de teste, principalmente dentro de um ciclo de vida de desenvolvimento do software, ou quando há alterações ou novos recursos (ISTQB, 2018).

2.1.3. Níveis de Teste

Os níveis de teste são uma abordagem feita para verificar se o software está atendendo aos requisitos estabelecidos, encontrar defeitos e aumentar a qualidade. Neste tópico, iremos abordar os quatro níveis de teste, de acordo com Silva (2022), que são: teste de unidade, teste de integração, teste de sistema e teste de aceitação.

1. Teste de unidade

Os testes de unidade, também conhecidos como testes unitários, de acordo com Berto (2019), possui como objetivo testar a menor unidade possível do projeto, ou seja, esses testes possuem como alvo os métodos dos objetos ou pequenos trechos de código, onde o teste tem como intuito provocar falhas, que podem ser ocasionadas tantos por erro de lógica como por erro de implementação.

Os desenvolvedores realizam esses testes durante a codificação do projeto, e utilizam como regra que para toda funcionalidade ou modificação do código implementados, os testes de unidade devem ser executados. Dentro dessa etapa deve existir certo nível de integração para que, para qualquer conteúdo que seja desenvolvido, os testes unitários devem ser executados.

De acordo com Silva (2022), o teste de unidade é responsável por testar componentes considerados mais simples dentro de um sistema, onde esses componentes são analisados pelo desenvolvedor com o objetivo de identificar erros e falhas.

A partir disso, pode-se observar que o teste de unidade é importante para o desenvolvimento do software pois, ao verificar cada unidade de código, o teste é capaz de identificar erros na primeira fase, facilitando o isolamento e resolução do problema. Gerando assim, um aumento da confiança na qualidade de software.

2. Teste de integração

Após os testes unitários, são realizados os testes de integração. De acordo com Berto (2019), os testes se complementam, enquanto os testes unitários buscam falhas na menor unidade do projeto, os de integração visam identificar possíveis falhas na comunicação entre módulos do produto, buscando também garantir que ao unir partes de uma aplicação, elas funcionem de forma planejada como um todo.

O teste de integração desempenha um papel crucial na validação da interação entre diferentes módulos de um sistema. Ele é responsável por garantir que todas as etapas de um programa funcionem corretamente quando estão integradas, além de testar possíveis erros de interface, onde esses testes

possuem como objetivo desenvolver a estrutura do programa a partir dos componentes que foram testados em testes de unidade (SILVA, 2022).

Um exemplo, citado por Fernandes (2018), é o teste de requisições HTTP. Durante esse teste, é verificado o resultado da requisição, o formato de dados e validações.

Com isso, podemos notar que os testes de integração possuem um papel fundamental nas futuras manutenções do sistema, pois, conforme novas funcionalidades vão sendo criadas (ou alterações em funcionalidades já existentes), com os testes, pode-se garantir que o comportamento de outras partes do sistema não vai ser afetado.

3. Teste de sistema

Para Pessoa (2020), o teste de sistema é frequentemente considerado um dos estágios mais complexos e desafiadores de se operacionalizar dentro de um processo de desenvolvimento de software. Isto porque, ao contrário dos outros testes, o teste de sistema busca validar a solução como um todo. É necessário que, para planejar esse tipo de teste, haja um entendimento completo dos requisitos não funcionais de um software.

Dentro desse tipo de teste, o foco está em avaliar aspectos como desempenho, segurança, disponibilidade e instalação do sistema, por conta de possuir uma infraestrutura mais complexa de hardware. Esses testes buscam também garantir que a solução seja capaz de funcionar de forma adequada em diferentes condições e consiga atender aos requisitos estabelecidos (PESSOA, 2020).

De acordo com Silva (2022), um teste de sistema tem como objetivo principal comparar as características e funcionalidades especificadas durante a fase de projeto com as características reais encontradas no programa após a integração de todos os módulos. Durante esse estágio, o sistema é testado como um todo, simulando o ambiente real de operação para verificar se ele atende aos requisitos definidos, e se os componentes são compatíveis, interagem corretamente e a transferência de dados está ocorrendo no momento certo.

De acordo com Berto (2019), esses tipos de testes utilizam o conceito de teste caixa preta, ou seja, considera que o teste utiliza apenas um conjunto de entradas de dados e verifica no final se o resultado está conforme o esperado, utilizando dados iguais ou próximos da realidade.

4. Teste de aceitação

De acordo com Berto (2019), os testes de aceitação são conduzidos por um grupo seleto de pessoas, geralmente incluindo gerentes do projeto ou até mesmo

os clientes. Os testes são realizados quando o sistema já está pronto, e possui como objetivo verificar se todos os requisitos estão sendo atendidos. Quando não há erros, o produto já está pronto para lançamento para todos os usuários.

Este teste possui como objetivo verificar que o sistema não apresenta erros ou desvio dos requisitos, sendo importantes para validar a qualidade do sistema antes de sua implantação definitiva, garantindo assim confiança para os clientes e para os desenvolvedores (SILVA, 2022).

2.1.4. Teste manual

Nos testes manuais, de acordo com Silva (2021), o testador executa o sistema, inserindo dados ao software e verificando a resposta do mesmo, de forma manual, comparando o resultado obtido com o resultado esperado, reportando defeitos caso haja algum. Para Carvalho (2022), o teste feito de forma manual possui um alto custo financeiro, pois possuem o padrão de ser mais demorados e com total interação humana. Porém, o ponto positivo é que o testador é capaz de vivenciar as condições correspondentes ao ambiente de produção.

De acordo com Pessoa (2020), os testes de software se relacionam com aspectos da psicologia humana, pois o testador deve ter uma visão adequada e estratégica, pois muitas vezes as atitudes humanas possuem mais relevância que o próprio processo.

Dentro do processo de teste manual, de acordo com Silva (2022), o testador compara o resultado do teste manual com o esperado, reportando as falhas, com suas informações, ao desenvolvedor, para que o código seja corrigido e o teste volte a se repetir, em busca de maior qualidade. Com isso, dentro da abordagem automatizada, esse processo é realizado de forma mais rápida, pois um testador, com conhecimento de linguagens de programação, codifica o teste em uma ferramenta de automação.

Com o aumento de ambientes a serem testados e o custo da execução de testes manuais, para Chicanelli et al (2019), esse processo se torna difícil. De acordo com o autor, ao realizar a implementação de testes automatizados, existem relatos de ganhos como redução no esforço, aumento na quantidade de ambientes testados, facilidade na repetição dos testes, e com isso, maior segurança e qualidade.

Porém, apesar de existirem vários benefícios, o autor Lourenço (2022) menciona que a extinção dos testes manuais é difícil de acontecer, pois com eles, a usabilidade, acessibilidade e responsividade são averiguadas. A automação dos testes manuais contribui para que tarefas repetitivas sejam diminuídas, auxiliando na economia do tempo e redução de custos, principalmente no caso de testes de

regressão, que são caros e difíceis de serem realizados de forma manual, pois custos com tempo e esforço são altos.

2.1.5. Teste automatizado

No início, de acordo com Rios e Moreira (2006), os dados e simulação dos testes eram gerados de forma manual. Com o aumento de sistemas Web, os processos de testes ficaram mais complexos, ocasionando em:

- Aumento do tempo esperado para realização dos testes;
- Aumento dos custos dos testes;
- Diminuição na qualidade dos sistemas que foram desenvolvidos.

Dentro desse contexto, de acordo com Lourenço (2022), as ferramentas de automação surgiram como uma forma de auxiliar nessas dificuldades. A automação é feita utilizando scripts, que realizam as atividades manuais e verifica se os resultados obtidos estão de acordo com o esperado, sendo utilizado como uma forma de agregar valor ao teste.

O processo de execução de testes em um teste automatizado, de acordo com Silva (2021), é realizado através de ferramentas específicas de automação, onde o testador desenvolve o script utilizando uma linguagem de programação, necessitando que o mesmo tenha um bom conhecimento em relação a computação, fazendo assim com que o teste seja executado de forma mais rápida.

Com isso, de acordo com Carvalho (2022), é mais rápido descobrir se o sistema está com o desempenho desejado, além de não necessitar de uma pessoa para executar o teste de forma manual. Possui como vantagem o aumento da produtividade, reuso, diminuição de erros e aumento de confiabilidade, e também simular a execução de múltiplos usuários em um sistema, porém, como desvantagem, pode ter um alto custo de implantação, com cenários complexos e mais específicos, além de poder exigir um maior conhecimento no desenvolvimento de scripts de teste.

No mercado existe uma vasta gama de ferramentas disponíveis, cada qual com suas próprias características e feitas para tipos específicos de teste. Com isso, é essencial avaliar suas vantagens e desvantagens antes de implementá-las em um projeto.

Apesar de seus benefícios, como melhora na eficácia do teste, agregação de qualidade, maior segurança e diminuição no tempo, de acordo com Silva e Dallilo (2019), durante a implementação das ferramentas, é necessário possuir uma equipe qualificada e com conhecimento em linguagens de programação. Além disso, existem desvantagens como altos custos iniciais de implementação e possível dificuldade no entendimento dos casos de teste ou da lógica de

programação. Por isso, é necessário que, além de uma ferramenta adequada, a equipe seja capacitada para se beneficiar das automações de teste.

Devido a este ponto, os testes manuais não devem ser extintos, pois possui abordagem diferente da automatizada, mas ainda sim importante. É responsabilidade da equipe de responsáveis definir quais casos de teste serão automatizados, para que assim todas as funcionalidades sejam cobertas.

2.2. Ferramentas de automação

Durante o passar dos anos e com o aumento da utilização de testes em ambientes de desenvolvimento no mercado, ferramentas de automação começaram a ser desenvolvidas buscando atender esse mercado e fornecendo ainda uma melhoria nos processos (como controle, gerenciamento e execução de atividades) relacionados aos testes de software (PESSOA, 2020).

Conforme é citado no ISTQB (2018), essas ferramentas são denominadas ferramentas de automação, que funcionam utilizando um conjunto de instruções escritas em uma linguagem de programação, considerando a ordem das entradas e como estão preenchidas, e valores utilizados em parâmetros de entrada e saída esperados.

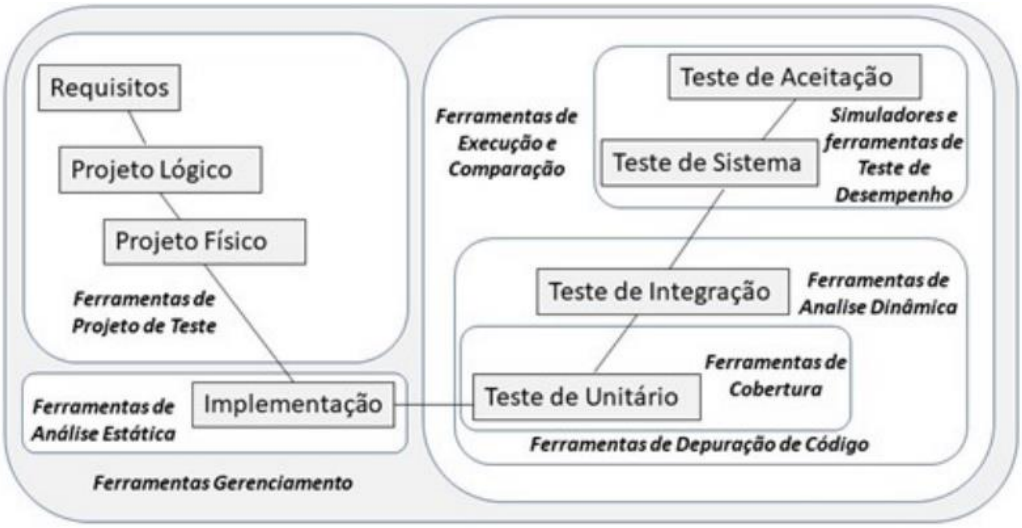
Pessoa (2020) complementa que as ferramentas possibilitam simular usuários e atividades humanas, buscando assim não utilizar passos manuais enquanto o teste de software é executado, exigindo assim que o testador responsável por automatizar o teste tenha os conhecimentos necessários dentro da área de tecnologia.

De acordo com Rollwagen (2020), uma ferramenta de automação de teste fornece suporte ao teste de software, buscando maior agilidade no processo e aumento em relação aos ganhos de tempo.

Existem muitas ferramentas para automação disponibilizadas no mercado, para Pessoa (2020), que podem ser utilizadas em várias fases dentro da implementação de um software, atuando desde a fase de testes unitários (buscando testar cada unidade do sistema) até a fase dos testes funcionais (testando todas as funcionalidades do sistema).

Como forma de melhorar o processo de teste, existem vários tipos de ferramentas disponíveis para auxílio durante o desenvolvimento de software, conforme pode ser observado na Figura 1:

Figura 1 - Tipos de Ferramentas



Fonte: Dustin (2003)

Essas ferramentas, de acordo com Pessoa (2020), são

- Ferramentas de Projeto de Testes: orientadas para gerar dados de teste a partir de uma especificação;
- Ferramentas de Projeto Lógico: utilizadas para geração de casos de testes;
- Ferramentas de Projeto Físico: utilizadas como manipulação ou geração de dados de teste;
- Ferramentas de Gerenciamento: permite existir uma relação entre os testes que estão sendo executados;
- Ferramentas de Análise Estática: utilizadas para identificar defeitos no código fonte;
- Ferramentas de Cobertura: utilizadas como forma de obter qual a porcentagem da aplicação que os testes estão cobrindo;
- Ferramentas de Depuração de Código: não são consideradas diretamente ferramentas de teste, sendo utilizadas somente como apoio para depurar o código;
- Ferramentas de Análise Dinâmica: utilizadas para avaliação do comportamento durante a execução do software;
- Simuladores: responsáveis por simular condições que testam partes do sistema;
- Ferramentas de Teste de Desempenho: utilizadas para avaliar o tempo de resposta do software;

- Ferramentas de Execução e Comparação: ferramenta responsável pela automação dos testes e comparação de saídas obtidas com saídas esperadas.

Devido a quantidade de ferramentas com diferentes objetivos, é importante saber utilizá-las conforme seu intuito e assim obter ganhos de tempo e menor custo durante a execução dos testes de software.

Dentro do contexto de desenvolvimento de sistemas, diversas ferramentas podem ser utilizadas para automação dos testes, como as ferramentas: Selenium WebDriver, Cypress, Unified Functional Testing e Robot Framework. Cada uma possui suas particularidades para utilização, e serão expostas nos tópicos a seguir:

2.2.1. Selenium

O Selenium é um framework utilizado para testes de aplicação web de forma automatizada, utilizando um conjunto de ferramentas e bibliotecas para seu funcionamento. Ele é capaz de atender diferentes tipos de necessidade, utilizando operações flexíveis e comparando resultados esperados e obtidos. Pode ser programado utilizando as linguagens Java, CSharp, Python, Ruby, Php, Perl e JavaScript, e pode ser utilizado em vários navegadores (CONTRI, 2019).

De acordo com Contri (2019), ele possui como vantagens a facilidade para execução de testes de regressão, fornece um rápido feedback dos casos de teste, além de possuir um relatório de defeitos personalizado que permite a localização de defeitos que não foram encontrados em testes manuais. Possui também uma documentação disciplinada e suporte para metodologias ágeis. Para Silva (2023), as vantagens, além de ser open source, são relacionadas a ele ser compatível com várias linguagens de programação, e suportar qualquer sistema operacional ou plataforma.

As desvantagens, de acordo com Silva (2023), são algumas limitações em casos de testes mais complexos, também sendo complicado para testar imagens, pois necessita integrar o Selenium com outras ferramentas, como o Sikuli. Além disso, para relatório nativo, é preciso integrar a ferramenta com o TestNG ou JUnit.

2.2.2. Cypress

O Cypress é um framework de automação de testes com o objetivo de testar aplicações *end to end*, utilizando o Javascript para escrever os scripts de teste. Ele executa o teste em conjunto com a execução da aplicação, respondendo a eventos no momento da execução. É utilizado em sistemas ou sites *front-end*, com frameworks reativos, mas também pode ser utilizado em ferramentas mais antigas (VIEIRA, 2021).

Como vantagens, é capaz de interagir com o tráfego da rede, manipulando dados. A aprendizagem para utilização do mesmo é rápida e fácil. A ferramenta permite a depuração dos testes, e possui fácil instalação e configuração (CORRÊA e SILVA, 2021). De acordo com Khan (2021), ele também considera a visibilidade dos elementos ao executar o teste. Ou seja, se estiver testando um comportamento de um botão (visível ou oculto, por exemplo), e ele não estiver nessa situação, o teste falhará, algo que outros frameworks não levam em consideração. A ferramenta também é capaz de tirar capturas de tela e vídeos do teste durante sua execução.

O framework possui como desvantagem o fato de não poder testar múltiplas guias ou janelas no mesmo momento. Além disso, os testes podem ser escritos somente em JavaScript (CORRÊA e SILVA, 2021). Para Khan (2021), a desvantagem está em não permite visitar dois domínios no mesmo teste, como por exemplo, acessar o google.com e o linkedin.com.

2.2.3. Robot Framework (com SeleniumLibrary)

O Robot framework é uma ferramenta de automação de testes que possui como objetivo realizar a criação de testes de aceitação e comportamento dentro de sistemas, onde sua automação é orientada a palavra-chave (FONSECA, 2020)

O framework possui como vantagem ser *open source*, de acordo com Monteiro (2020), além de uma linguagem simples baseada em palavras-chave, e possui fácil instalação. Os relatórios são gerados de forma automática, e disponibiliza saídas em formato XML. A ferramenta fornece também recursos que facilitam na automação de testes, e uma documentação com detalhes sobre uso e exemplos, além de disponibilizar bibliotecas com o objetivo de facilitar a customização de novas bibliotecas de teste. É um framework flexível, podendo ser usado diferentes tipos de tecnologias e interfaces, e necessita de pouca habilidade de programação para utilizá-lo, possuindo fácil utilização.

Uma das bibliotecas mais utilizadas do Robot Framework é SeleniumLibrary, que é indispensável para realizar testes automatizados em ambientes Web.

O Robot Framework possui regras rígidas de indentação, e seus relatórios em HTML são complicados de personalizar. A depuração de erros é considerada difícil, e é uma ferramenta considerada difícil de se manter (WASHINGTON, 2021). Além disso, de acordo com site Test Automation Tools (2023), os casos de testes complexos podem possuir certa dificuldade para se escrever utilizando a abordagem orientada por palavras-chave.

2.2.4. Unified Functional Testing (anteriormente conhecido como QTP)

O Unified Functional Testing (UFT), anteriormente conhecido como QuickTest Professional (QTP), é uma ferramenta de automação de testes que fornece a capacidade de realizar testes funcionais automatizados e teste de regressão, utilizando a linguagem Visual Basic Scripting Edition, de acordo com o site Prime Control (2018). Ele pode ser utilizado em ambientes web, mobile e desktop.

Possui como vantagem a automação de Smoke Testing e Regressão, além de poder dar suporte ao Record e Play Suporte para plataformas como Oracle, SAP e WebForms (CORRÊA e SILVA, 2021). Ele também necessita de habilidades básicas de programação, de acordo com Umar e Zhanfang (2019), facilitando no momento de criação e execução de testes. A ferramenta possui alta integração com o HP ALM (ferramenta para gestão de testes). Também possui Smart Object Recognition, um mecanismo para tratamento de erros e documentação automática (PRIME CONTROL, 2018).

Como desvantagem, ele utiliza somente o VBScript como forma de escrita, além de possuir suporte somente para o Windows, além de necessitar de hardware com forte capacidade para executá-lo. Possui também um alto custo financeiro (CORRÊA e SILVA, 2021). Além do alto custo, de acordo com Umar e Zhanfang (2019), as atualizações e módulos adicionais também são caros.

2.3. Critérios de avaliação a serem utilizados

Durante o processo de desenvolvimento de software, a seleção da ferramenta ideal desempenha um papel importante dentro de um projeto. Com a variedade de ferramentas disponíveis no mercado, critérios devem ser estabelecidos como forma de auxiliar na escolha da ferramenta que se adeque melhor às necessidades específicas do projeto. Com isso, esse tópico visa explorar quais critérios são importantes para escolher uma ferramenta e garantir maior qualidade e confiabilidade dentro do desenvolvimento de software.

2.3.1. Recursos

Serão avaliados quaisquer recursos e funcionalidades oferecidos por cada ferramenta para apoiar a automação de testes. Esse critério pode ser de extrema importância para se determinar se a ferramenta é adequada para o projeto. Um exemplo é os relatórios de teste, que são bastante comuns entre os sistemas de automação, e são bastante variados entre elas.

Outro exemplo é a gravação das ações do usuário com a interface, podendo transformá-las em código e reproduzi-las nos testes.

2.3.2. Facilidade de Uso

Será avaliada a facilidade com que uma determinada ferramenta pode ser compreendida, configurada, operada e mantida pelos usuários. Outro ponto que deve ser levado em consideração é se a documentação é clara e abrangente.

Esse critério desempenha um papel crucial na eficiência e produtividade das equipes de testes automatizados, pois quanto mais intuitiva e amigável for a interface da ferramenta, mais fácil será para os usuários começarem a usá-la e se habituarem com suas funcionalidades. Além disso, uma ferramenta de teste automatizado fácil de usar requer menos tempo de treinamento, permitindo que novos testadores entrem em ação mais rapidamente.

2.3.3. Flexibilidade

Será avaliada a capacidade das ferramentas se adaptarem a diferentes cenários de teste e requisitos específicos de um projeto. Além disso, também deve ser levado em consideração a facilidade de personalização e configuração para atender as diferentes necessidades da empresa.

Além disso, será avaliada a capacidade de uma ferramenta de se conectar e interagir com outras ferramentas e tecnologias utilizadas no processo de desenvolvimento de software. Essa integração é importante para permitir uma automação eficiente e suave dos testes em todo o ciclo de vida do desenvolvimento de software.

2.3.4 Escalabilidade

Será avaliada a capacidade de uma ferramenta de teste se adaptar e lidar com um aumento na complexidade dos sistemas, volumes de dados e demandas de desempenho à medida que o projeto cresce.

Um aspecto importante é o desempenho, pois a ferramenta deve ser capaz de executar rapidamente os testes quando há uma quantidade grande deles ou massa de dados. Também deve ser levado em consideração a facilidade de manutenção deles, mesmo quando o projeto é complexo e possui muitos testes.

2.3.5. Custo-benefício

Será avaliado o custo associado para poder utilizar a ferramenta e seus recursos levando em conta os seus benefícios, utilizando os demais critérios que foram usados nas avaliações neste projeto.

2.3.6. Suporte Técnico

Será avaliado a disponibilidade e qualidade do suporte oferecido pelos fornecedores das ferramentas de testes automatizados. É um fator importante a ser considerado ao escolher uma ferramenta, pois o suporte adequado pode ser crucial para resolver problemas técnicos, receber orientações e garantir o bom funcionamento da ferramenta em uso.

Alguns exemplos de aspectos importantes são os canais disponíveis para obter suporte técnico, documentação abrangente, tutoriais, guias de usuário, comunidade de usuários e atualizações com manutenções e correções de erros.

3. Objetivos

O presente trabalho possui como objetivo geral realizar a comparação e avaliação de ferramentas de automação, buscando identificar suas características, funcionalidades, vantagens e desvantagens, com o propósito de auxiliar no processo de seleção da ferramenta mais adequada para um determinado contexto de automação, teste e sistema.

Como objetivos específicos, o trabalho se propõe a:

- Identificar as ferramentas de automação de teste de software;
- Aplicar as ferramentas em um cenário real, utilizando ambiente web para testes, buscando entender como se comportam e como é sua curva de aprendizado;
- Identificar as vantagens e limitações das ferramentas de automação;
- Apresentar um resultado comparativo entre as ferramentas de automação em um caso de teste, com base na avaliação de cada ferramenta considerando seus recursos, facilidade de uso, flexibilidade, escalabilidade, custo-benefício e suporte técnico;
- Compreender a importância dos testes de software de forma automatizada;
- Fornecer uma visão simplificada de ferramentas de software, facilitando o processo de escolha de ferramentas para testadores, desenvolvedores e pessoas dentro da área de desenvolvimento de software, buscando atender da melhor maneira as diferentes necessidades empresariais.

4. Atividades realizadas

Em relação aos estudos preliminares, ferramentas como Unified Functional Testing (UFT), e as linguagens Python e VBscript foram mais fáceis de se adequar, devido ao conhecimento prévio do aluno sobre esses recursos.

Sobre os estudos que foram realizados e aprofundamento dos conhecimentos, foi necessário estudar sobre as ferramentas: Selenium, Cypress, Robot Framework. Além disso, em relação às linguagens, foi necessário aprender as linguagens Javascript e Robot Framework Language, possibilitando a ampliação das habilidades do aluno nesses aspectos.

No processo de desenvolvimento, as atividades foram separadas em fases, onde a primeira fase foi a realização dos testes de ambientes, que foi dividido em:

1. Escolha do ambiente de teste;
2. O passo a passo foi testado no ambiente de teste, buscando entender se o mesmo estava funcionando conforme esperado;
3. Escolha das ferramentas de automação para utilizar durante o estudo comparativo;
4. Instalação das ferramentas de automação;
5. Aprendizado sobre como funcionam as ferramentas de automação
6. Aprendizado sobre as linguagens de programação a serem utilizadas dentro das ferramentas;
7. Verificação se as ferramentas estão compatíveis com o ambiente de testes.

Após isso, os resultados foram comparados buscando entender como cada ferramenta se comportava e buscando pontos de comparação, onde as etapas foram divididas da seguinte forma:

1. Definição dos critérios de avaliação das ferramentas;
2. Realização do teste manual;
3. Realização dos testes automatizados com todas as ferramentas;
4. Comparação da duração de cada teste;
5. Análise comparativa sobre as ferramentas, destacando seus pontos positivos e negativos e separando-os de acordo com os critérios de avaliação.

5. Projeto e especificação do sistema

Os testes no ambiente web foram testados no site de compras da Amazon Brasil, e o processo foi a realização de compras de uma garrafa térmica, onde o teste verifica se o login está sendo realizado com sucesso, se a barra de pesquisas está funcionando corretamente e se o produto estava realmente sendo inserido no carrinho de compras. Além disso, também foi testado a forma de pagamento e a inserção de endereço de entrega.

As ferramentas utilizadas para realização dos testes automatizados foram as ferramentas de automação: Selenium, Cypress, Unified Functional Testing, Robot Framework, além de ferramentas de apoio, como: PyCharm, Visual Studio Code e Google Chrome.

As linguagens utilizadas nas ferramentas de automação foram Python (no Selenium), Javascript (no Cypress), VBscript (no Unified Functional Testing) e Robot Framework Language(no Robot Framework usando Selenium e Appium).

O presente trabalho possibilita enxergar as diferenças, além de benefícios e desvantagens, de utilizar as ferramentas de automação para automação dos testes de software, buscando a melhor opção com o melhor custo benefício. Ao fim do estudo, a pesquisa deve servir como um apoio para testadores, desenvolvedores e pessoas dentro da área de informática, possibilitando assim, escolher qual ferramenta se encaixa melhor nos objetivos e no contexto dos testes de software.

6. Implementação e avaliação

Neste tópico será demonstrado como foi realizado o planejamento dos testes e suas respectivas execuções.

6.1. Planejamento e execução de testes funcionais

Durante o planejamento dos testes manuais e automatizados, primeiramente foram realizados testes nas ferramentas de automação, buscando verificar:

- Como a ferramenta funciona e qual tipo de linguagem deve ser utilizada;
- Instalação de plug-ins e extensões, quando necessário;
- Como funciona a integração com o navegador Google Chrome;
- Se estavam funcionando conforme o esperado.

Após esses passos, o ambiente de teste web foi testado, com o intuito de verificar:

- Se os ambientes apresentaram algum tipo de instabilidade;
- Se os dados estavam funcionando corretamente;
- Se as ferramentas de teste conseguiam identificar bem todos os campos e objetos.

Com isso, ferramentas e ambientes de testes estavam funcionando corretamente, possibilitando uma maior confiança que os testes seriam executados com qualidade.

6.2. Planejamento e execução de outros testes

Neste item foram abordadas as execuções dos testes de software, de forma manual e automatizada.

6.2.1. Testes manuais

Durante os testes manuais, foram verificados vários pontos do website como forma de validação, seguindo um passo a passo padronizado tanto para os testes

manuais quanto para os testes automatizados, e os passos foram divididos da seguinte forma:

Passo 1: O usuário deve acessar o sistema da Amazon Brasil através do link https://www.amazon.com.br/ref=ap_frn_logo;

Resultado esperado do passo 1: O sistema deve apresentar a tela inicial da Amazon.

Passo 2: O usuário deve clicar em “Olá, faça seu faça seu login Contas e Listas”;

Resultado esperado do passo 2: O sistema deve apresentar a tela de login.

Passo 3: O usuário deve preencher o campo “Número do celular ou e-mail” com seu e-mail de login e clicar no botão “Continuar”;

Resultado esperado do passo 3: O sistema deve apresentar o campo para inserir a senha.

Passo 4: O usuário deve preencher o campo “senha” com sua senha e clicar no botão “Fazer login”.

Resultado esperado do passo 4: O sistema deve apresentar a tela inicial da Amazon com o login do usuário.

Passo 5: O usuário deve preencher a barra de pesquisa “Pesquisa Amazon.com.br” com a palavra “garrafa” e clicar no ícone de lupa.

Resultado esperado do passo 5: O sistema deve apresentar os ítens resultados da pesquisa.

Passo 6: O usuário deve clicar em um dos itens apresentados.

Resultado esperado do passo 6: O sistema deve abrir a página do item selecionado.

Passo 7: O usuário deve clicar no botão “Comprar agora”.

Resultado esperado do passo 7: O sistema deve apresentar a página “Finalizar a compra”.

Passo 8: O usuário deve verificar se o endereço apresentado em “Endereço de entrega” está correto e clicar em “Alterar” em “Método de pagamento”.

Resultado esperado do passo 81: O sistema deve exibir as diferentes formas de pagamento.

Passo 9: O usuário deve escolher a opção de boleto e clicar em “Usar este método de pagamento”.

Resultado esperado do passo 9: O sistema deve deixar selecionada a opção de boleto.

Passo 10: O usuário deve clicar em “Finalizar pedido”. (não utilizado nos testes automatizados)

Resultado esperado do passo 10: O sistema deve apresentar o boleto para o pagamento. (não utilizado nos testes automatizados)

Todos os passos foram executados com sucesso, concluindo assim que o teste foi realizado com sucesso, sem nenhuma ocorrência de falhas, e sua duração total foi de dois minutos e vinte segundos.

Todas as etapas foram realizadas nos cenários automatizados, com exceção do passo 10, que para evitar a criação de inúmeros boletos, não foi executado.

6.2.2. Testes automatizados

Foram executados testes automatizados no ambiente de teste web. Para os testes web, foram utilizadas as seguintes ferramentas: Selenium WebDriver, Cypress, UFT e Robot Framework. Além disso, pode-se contar com o suporte adicional do Appium e Selenium no contexto do Robot Framework.

Os testes automatizados de cada ferramenta foram criados a partir do teste manual detalhado no item 6.2.1

6.2.2.1 Resultados

Durante o processo de comparação, primeiramente foi realizado um estudo de tempos avaliando o tempo de execução das ferramentas Selenium WebDriver, Cypress, Unified Functional Testing e Robot Framework com o ambiente de teste, onde cada teste foi executado dez vezes. Todos os valores vieram arredondados, com exceção do Selenium, que foi necessário arredondar para melhor comparação. Pode-se observar os resultados desse estudo na figura abaixo:

Figura 2 - Estudo de tempos (em segundos) das ferramentas de automação

	Selenium	Cypress	UFT	Robot Framework
Teste 1	25	25	42	20
Teste 2	23	26	41	19
Teste 3	23	26	40	18
Teste 4	22	26	40	19
Teste 5	23	26	39	18
Teste 6	22	25	38	19
Teste 7	21	24	38	19
Teste 8	21	27	39	19
Teste 9	21	26	44	20
Teste 10	22	28	37	18
Total	22	26	40	19

Fonte: Autoria própria (2023)

Pode-se observar que a ferramenta Robot Framework apresentou um melhor desempenho em um ambiente web se comparada às outras ferramentas

testadas, com o valor de 19 segundos de duração, sendo a ferramenta mais rápida.

Com resultados medianos, ficaram as ferramentas Selenium Webdriver e Cypress, com respectivamente, 22 e 26 segundos de duração.

Como pior desempenho em sua relação com o ambiente de teste web, o Unified Functional Testing obteve o valor de 40 segundos de duração.

Além disso, em comparação com o teste manual, todas as ferramentas obtiveram um melhor resultado, onde o manual teve um minuto e quarenta segundos a mais de diferença se comparado com a ferramenta de menor desempenho.

Após o estudo de tempos, as ferramentas de automação foram comparadas utilizando os critérios de avaliação definidos e relatados nos tópicos a seguir:

6.2.2.1.1. Selenium

Uma das ferramentas disponíveis pela Selenium é o Selenium IDE, de acordo com o site Selenium IDE (2019), que é uma extensão para navegador disponível para Google Chrome e Mozilla Firefox, que permite o usuário gravar suas ações, transformando-as em código, além de editar, depurar, adicionar pontos de interrupção e pausas nos testes. Em geral, é uma opção mais adequada para testes simples. Outra funcionalidade do Selenium IDE é possibilitar a execução de testes em paralelo com qualquer combinação de navegadores e sistemas operacionais.

Em relação à questão do relatório, é necessário a integração com fontes externas como o JUnit e TestNG para que seja possível a visualização do mesmo (BHARATI, 2023).

O Selenium é de fácil utilização e possui uma curva de aprendizagem baixa, possibilitando os novos colaboradores de serem mais produtivos mais rapidamente. Além do mais, possui um site para apoio de dúvidas e fóruns, facilitando o aprendizado de quem está começando. A ferramenta também tem uma vasta documentação, grupo de usuários, blog, tutoriais e exemplos em cada linguagem suportada. Ele é compatível com várias linguagens, como por exemplo Java, JavaScript, C# e Python, além de também ter compatibilidade com diversos navegadores, como o Google Chrome, Mozilla Firefox, Safari e Edge (SELENIUM, 2023).

Em relação à sua escalabilidade, para o site Medium (2015), seus testes foram o segundo mais rápido dentre as ferramentas comparadas. É possível executar testes de regressão utilizando frameworks de testes, como o NUnit. Além de que, é possível paralelizar execuções em diferentes máquinas, com várias

combinações de navegadores (em diferentes versões) e sistemas operacionais utilizando o Selenium Grid, que simplifica a gestão de testes em grande quantidade (SELENIUM, 2023).

De acordo com o site Browser Stack (2023), o Selenium é uma ferramenta *open source*, e possui diversos serviços terceirizados comerciais de suporte, indicados por eles, e atualizações de sistemas frequentes com correções de bugs.

6.2.2.1.2. Cypress

De acordo com os testes realizados, pode-se observar que a ferramenta Cypress permite acesso a capturas de tela e vídeos em seu relatório, além da funcionalidade *time-travel* que permite ver o comportamento do programa em cada passo-a-passo da execução (CYPRESS, 2023).

O Cypress apresentou uma fácil utilização, com uma interface de usuário bastante amigável, uma curva de aprendizado média em relação às demais ferramentas testadas, e por fim, uma fácil escrita de testes.

Para ajudar novos testadores, seu site possui uma documentação detalhada e em seu canal no YouTube existem múltiplos vídeos expondo palestras, aulas e tópicos para tirar dúvidas. Além de tudo isso, o Cypress disponibiliza cursos online cujo tema vai da instalação do software até conceitos avançados.

Uma limitação que pode ser observada é que os códigos só podem ser escritos em JavaScript. Também pode-se observar que não é possível testar várias guias ao mesmo tempo. Os navegadores que são possíveis de serem utilizados ao se utilizar a ferramenta são Google Chrome, Mozilla Firefox e Edge.

Em relação à sua escalabilidade, seus testes foram o terceiro mais rápido dentre as quatro ferramentas comparadas. É possível executar os testes em paralelo em diferentes máquinas usando o Cypress Cloud.

O Cypress é open-source e possui quatro versões diferentes: Free, Team, Business e Enterprise. Essa primeira é grátis e o preço mensal ou anual vai aumentando respectivamente, assim como seus benefícios.

Os planos de suporte variam de acordo com a versão da ferramenta que foi contratada, sendo que os três que não são gratuitos possuem SLA (Acordo de Nível de Serviço) com um tempo mínimo para o atendimento de um eventual ticket de suporte. Também possui atualizações de sistemas frequentes com correções de bugs.

6.2.2.1.3. UFT (Unified Functional Testing)

O UFT (Unified Functional Testing) permite gravar a interação do usuário com o ambiente, para transformá-la em código e posteriormente repeti-la durante o teste. Essa funcionalidade é chamada pelos desenvolvedores de Record and

Playback. Outro recurso disponível é o relatório que está integrado no software e pode ser configurado para exibir a captura de tela de cada passo a passo.

O UFT não apresentou uma interface de fácil uso e sua curva de aprendizagem é alta em relação às outras ferramentas usadas. A documentação apresentada é escassa e seu site de apoio é extremamente confuso. Por muitas vezes, é difícil encontrar apoio para a ajuda com erros e dúvidas e exemplos de aplicação de códigos. A comunidade apresenta poucos fóruns e vídeos para ajudar novos usuários. Por essas razões, pode ser necessário um certo tempo para utilizar-se da ferramenta.

Os testes desenvolvidos pela ferramenta só podem ser escritos em VBScript e podem ser executados nos seguintes navegadores: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge e Internet Explorer, sendo necessária a instalação do plug-in do UFT. Além do ambiente Web tratado no projeto, a ferramenta também é compatível com outras tecnologias como SAP, Java e Oracle.

Em relação à sua escalabilidade, seus testes foram os mais lentos dentre as quatro ferramentas comparadas no ambiente Web. Porém, ela é capaz de realizar regressão e possui integração com o ALM/Quality Center que ajuda a controlar as atividades de gerenciamento de qualidade de software.

O UFT é uma ferramenta que não possui open-source, mas possui uma licença de teste de 30 dias para experimentá-la. Em relação ao suporte técnico, eles possuem telefones e um plano de Suporte Premium.

6.2.2.1.4. Robot Framework (Selenium)

A ferramenta Robot Framework apresenta um relatório simples em formato HTML, ao final de cada execução, e a possibilidade de capturar a tela quando for necessário. Pode-se utilizar diversas bibliotecas externas de acordo com suas necessidades, como por exemplo a “SeleniumLibrary”, necessária para criar e executar testes em ambiente Web.

É uma ferramenta de fácil utilização, com uma curva de aprendizagem extremamente fácil e rápida. Para novos usuários, seus fóruns e sites são muito úteis como apoio para tirar dúvidas e resolver problemas. Porém, sua documentação não é muito detalhada e não possui muitos exemplos.

Ela permite a automação nos seguintes navegadores: Mozilla Firefox, Google Chrome, Internet Explorer, Edge e Safari. Só pode ser escrito na linguagem Robot Framework Language.

Em relação à sua escalabilidade, seus testes foram os mais rápidos dentre as quatro ferramentas comparadas no ambiente Web, permite realizar testes de

regressão e com a ajuda da ferramenta Pabot é possível executar testes de forma paralela.

Robot Framework é open-source, mas existe a opção de pagar uma tarifa anual para se tornar membro e conseguir alguns benefícios.

Se for necessário suporte técnico, possui apenas um email para contato, mas possui atualizações com frequência para correções de erros.

6.2.3. Critérios de Avaliação

A partir do exposto nos tópicos anteriores, foi criada uma tabela relacionando as ferramentas de automação com os critérios avaliativos definidos no presente trabalho, buscando assim facilitar a visualização sobre os pontos relevantes sobre cada ferramenta.

Figura 3 – Critérios de avaliação - parte 1

	Recurso	Facilidade de uso	Flexibilidade
Selenium	- Utilizando a ferramenta Selenium IDE, permite a gravação das ações do usuário, transformando isso em código; - A Selenium IDE possibilita editar, depurar, adicionar pontos de interrupção e pausas nos testes; - Utilizando recursos que podem ser integrados, o Selenium fornece um relatório detalhado da execução do teste.	- Fácil utilização, curva de aprendizem baixa; - Possui site para apoio de dúvidas e fóruns; - Tem vasta documentação, grupo de usuários, blog e tutoriais, além de exemplos em cada linguagem.	- Compatível com várias linguagens, como por exemplo, Java, JavaScript, C# e Python; - Compatível também com vários navegadores, como Google Chrome, Mozilla Firefox, Edge e Safari.
Cypress	-Permite acesso a capturas de tela, relatórios e vídeos do teste após execução; - Possue a funcionalidade time-travel, permitindo ver o comportamento do programa em cada passo-a-passo da execução.	- Fácil utilização, curva de aprendizado média; - Fácil escrita de testes; - Disponibiliza cursos online; - Documentação detalhada e canal no YouTube focado em ajudar os usuários, com vídeos expondo palestras, aulas e tópicos para tirar dúvidas.	- Pode ser escrito somente em Javascript, não pode testar várias guias ao mesmo tempo; - Permite automatizar somente nos navegadores: Google Chrome, Mozilla Firefox e Edge.
UFT	-Permite acesso a capturas de tela e relatórios do teste após a execução; - Permite gravar as ações do usuário transformando em código.	- A aplicação não possui uma interface de fácil uso; - Curva de aprendizagem alta; - Poucos fóruns e vídeos para ajudar novos usuários; - Falta de exemplos para aplicação de códigos;	- Pode ser escrito somente em VBScript; - Permite execução em navegadores com Google Chrome, Mozilla Firefox, Safari e Internet explorer; - Precisando apenas de plug-in instalado nesses navegadores; - Permite automação em diferentes ferramentas também, como o SAP, Java e Oracle.
Robot Framework	- Possui relatório de execução dos testes; - Pode-se utilizar diversas bibliotecas externas de acordo com as suas necessidades.	- Fácil utilização da ferramenta, com curva de aprendizagem fácil e rápida; - Existe site para apoio para tirar dúvidas, além de fóruns; - Possui documentação, porém não é muito detalhada e possui poucos exemplos.	- Assim como mostrado no critério recursos, pode-se utilizar diversas bibliotecas externas de acordo com as suas necessidades; - Só pode ser escrito na linguagem Robot Framework Language; - Permite automação nos navegadores Mozilla Firefox, Google Chrome, Internet Explorer, Edge e Safari.

Fonte: Autoria própria (2023)

Figura 4 – Critérios de avaliação - parte 2

	Escalabilidade	Custo-benefício	Suporte técnico
Selenium	- Utilizando o Selenium IDE, é possível executar testes em paralelo; - Permite regressão utilizando o framework de testes, como o NUnit, e seus testes foram o segundo mais rápido das ferramentas comparadas; - Com o Selenium Grid, é possível paralelizar execuções em diferentes máquinas remotas.	- Open source.	- Possui diversos serviços terceirizados comercias de suporte; - Possui atualizações de sistema com correções de erros.
Cypress	- Testes automatizados com duração média; - É possível paralelizar testes utilizando o Cypress Cloud.	- Open source; - Possui quatro tipos de versão que podem ser contratadas: Free, Team, Business e Enterprise.	- Possui quatro tipos diferentes de planos de suporte, um pra cada versão; - Os três que não são gratuitos possuem SLA (Acordo de Nível de Serviço) com um tempo mínimo para o atendimento de ticket de suporte;
UFT	- Baseado nos testes realizados, o UFT é capaz de executar teste de regressão; - Possui integração com o ALM/Quality Center, que ajuda a controlar as atividades; - É uma ferramenta mais lenta e os testes tiveram um maior tempo de execução.	- Pago, com 30 dias de avaliação.	- Possui telefones para suporte no Brasil; - Possui um plano de Suporte Premium.
Robot Framework	- Permite realizar regressão, com rápida execução dos testes automatizados; - Com a ferramenta Pabot, pode-se executar testes de forma paralela.	- Open source; - Pode pagar uma tarifa anual para se tornar membro e conseguir benefícios.	- Possui atualizações de sistema com correções de erros; - Possui apenas um email para contato.

Fonte: Autoria própria (2023)

6.3. Comentários sobre a implementação

Em um primeiro momento, estava planejado realizar testes e ambiente mobile com as ferramentas Robot Framework, com a biblioteca AppiumLibrary, e UFT Mobile. Porém, depois de requisitar acesso gratuito à essa segunda, não foi possível obter nenhum tipo de resposta da empresa Micro Focus. Causando, então, a não realização desses testes, pois seria impossível realizar alguma comparação.

7. Considerações finais

O desenvolvimento de software requer muita organização, conhecimento técnico e atenção. Por conta disso, é inevitável que haja falhas durante o processo. Por este motivo, é importante considerar a opção de utilizar atributos que facilitem esse processo, tornando-o mais econômico, prático e gastando um menor tempo, como é o caso de testes automatizados, que auxiliam na identificação de erros e defeitos e contribuem para o aumento da qualidade de software.

Após a coleta de dados e a realização dos testes e comparação entre as diferentes ferramentas, tornou-se possível identificar os pontos positivos e negativos de cada uma delas. Com base nos critérios de avaliação, foi possível tomar conclusões sobre as ferramentas de forma mais embasada, sendo fundamental para garantir um desenvolvimento mais eficiente.

O Robot Framework obteve o melhor resultado de desempenho em comparação com as outras ferramentas. Além disso, como pontos positivos, possui um relatório HTML integrado, e também integração com bibliotecas com diversas linguagens, facilitando assim a automação. Apesar de ser uma ferramenta *open source*, para maiores benefícios existe um plano pago, sendo possível aumentar possivelmente o custo de um projeto.

O Selenium permite a utilização de várias linguagens de programação. Para testes web simples, o Selenium possui uma extensão onde o usuário pode gravar suas ações e transformá-las em código, além de ter a opção também de executar testes em paralelo, facilitando, por exemplo, em testes de regressão. Para relatório, precisa integrar com outras fontes, sendo um processo relativamente mais complicado que em comparação com outras ferramentas, onde o relatório já vem integrado.

O Cypress possui como ponto positivo o fato de realizar capturas de tela e gravações do teste, além de poder voltar no código automatizado para observar como foi feito o passo na tela do ambiente web. É permitido somente executar com a linguagem Javascript, o que em comparação com as outras ferramentas é uma desvantagem, pois elas aceitam o uso de outras linguagens. Porém, o Cypress possui uma curva de aprendizagem rápida, compensando o fato de só permitir o uso de uma linguagem.

Assim como o Selenium, o UFT também possui a opção de gravar as ações do testador, sendo mais simples para casos onde o testador não possui tanta experiência com linguagem de programação, e sendo ideal também para casos de testes mais simples. Com a duração de 40 segundos de duração, por mais que

ainda seja melhor que um teste manual no quesito duração, pode-se observar que o UFT não é a melhor opção ao se realizar testes em ambientes web. Além disso, por não ser uma ferramenta *open source*, demandaria um alto custo para implementação e também um maior tempo de aprendizado, visto que sua curva de aprendizagem é alta. Assim como o Cypress, o UFT também permite somente a linguagem Javascript. Como ponto positivo, possui a geração de relatórios de forma integrada no sistema.

Tendo em vista os pontos acima, cabe aos responsáveis por um projeto analisar qual ferramenta é ideal para ser utilizada, podendo utilizar como critérios pontos como: Rapidez, onde o Robot Framework se mostrou o melhor. Porém, para casos onde se precisa de uma ferramenta totalmente gratuita, ferramentas como o Cypress e Selenium se mostraram melhores, por exemplo.

Apesar disso, como resultado das comparações, a que se apresentou de forma mais satisfatória para o presente trabalho foi o Robot Framework, por sua aprendizagem ser mais fácil, ter um comportamento mais rápido e ter ferramentas como exibição de relatórios integrada, além de aceitar diferentes bibliotecas, facilitando a implementação de um projeto. Porém, como pode-se observar, todas as ferramentas analisadas apresentaram pontos positivos e importantes para se considerar, pois cada uma apresenta características que podem se adequar melhor com o desenvolvimento em questão.

8. Referências bibliográficas

ANDRADE, J. et al. **An architectural model for software testing lesson learned systems**. Information and Software Technology, v. 55, n. 1, p. 18-34, 2013.

BARBOSA, F. B. e TORRES, I. V. **O Teste de Software no Mercado de Trabalho**. Revista Tecnologias em Projeção, v. 2, n. 1, p. 49-52, 2011.

BARTIÉ, A. **Garantia da Qualidade de Software**: Adquirindo Maturidade Organizacional. Rio de Janeiro: Elsevier, 2002 – 13ª Reimpressão.

BAUMGARTNER, Cristiano. **Conheça 5 benefícios dos testes automatizados de software**. 2021. Disponível em: https://testingcompany.com.br/blog/conheca-5-beneficios-dos-testes-automatizados-de-software. Acesso em: 02 mai. 2023.

BERTO, Rafael Figueiró. **Técnicas de teste e simulação automatizada de software voltadas a sistemas embarcados**. Tese (monografia) - Curso de Engenharia de Controle e Automação, Departamento de Automação e Sistemas, Centro Tecnológico, Universidade Federal de Santa Catarina, 2019. Disponível em:

https://repositorio.ufsc.br/bitstream/handle/123456789/200133/PFC%20Rafael%

20Figueir%c3%b3%20Berto_2019-1.pdf?sequence=1&isAllowed=y. Acesso em: 02 mai. 2023.

BHARATI, Neha. **Top 5 Selenium Reporting Tools**. 2023. Disponível em: https://www.browserstack.com/guide/top-selenium-reporting-tools. Acesso em: 03 mai. 2023.

BROWSER STACK. **Selenium**. 2023. Disponível em: https://www.browserstack.com/selenium#:~:text=Selenium%20is%20an%20open-source,%2C%20and%20C%23%2C%20among%20others. Acesso em: 03 mai. 2023.

CARVALHO, Kilvia da Silva. **Avaliação de ferramentas de geração de casos de teste de software**: uma análise comparativa. Trabalho de Conclusão de Curso - Monografia (Curso de Bacharelado em Engenharia da Computação) - Instituto Federal da Paraíba, 2022.

CHICANELLI, Rachel et al. **Aspectos sociais, humanos e econômicos da utilização de testes automatizados no desenvolvimento de sistemas**. Disponível em: https://www.iiisci.org/journal/PDV/risci/pdfs/CA234CS19.pdf. Acesso em: 30 abr. 2023.

CONTRI, Maria Eduarda. **Selenium como uma ferramenta para testes de software**. 2019. Disponível em: https://medium.com/@dudacontri65/selenium-como-uma-ferramenta-para-testes-de-software-22541584b960. Acesso em: 25 abr. 2023.

CORRÊA, Jonas Santos; SILVA, Welington Tierry. **Elaboração de um mínimo processo viável para automação de testes de interface em fábricas de software**. 41 p. Monografia - Curso de Engenharia de Computação Universidade Evangélica de Goiás - UniEVANGÉLICA, Anápolis, 2021. Disponível em: http://repositorio.aee.edu.br/bitstream/aee/19683/2/Entrega%2005%20%288%29.pdf. Acesso em: 26 abr. 2023.

CYPRESS. **Cypress**. 2023. Disponível em: https://www.cypress.io/app/#flake_resistance. Acesso em: 27 abr. 2023.

CYPRESS. **Cypress Suport**. 2023. Disponível em: https://www.cypress.io/support/. Acesso em: 28 abr. 2023.

CYPRESS. **Parallelization**. 2023. Disponível em: https://docs.cypress.io/guides/guides/parallelization. Acesso em: 28 abr. 2023.

CYPRESS. **Pricing**. 2023. Disponível em: https://www.cypress.io/pricing/. Acesso em: 28 abr. 2023.

CYPRESS. **Real World Testing with Cypress.** 2023. Disponível em: https://learn.cypress.io/#courses. Acesso em: 27 abr. 2023.

DUSTIN., E. **Effective Software Testing: 50 Specific Ways to Improve Your Testing**. Addison-Wesley Professional; 1st edition, 2003.

FERNANDES, Mateus. **Teste de Unidade e Teste de Integração:** O que são?. 2018. Disponível em: https://medium.com/@mateus1198/teste-de-unidade-e-teste-de-integra%C3%A7%C3%A3o-o-que-s%C3%A3o-de58d7a3d3d2. Acesso em: 30 abr. 2023.

FONSECA, Matheus Fernandes Samuel Tomkelski. **AUTOMAÇÃO DE TESTES DE SOFTWARE**: ESTUDO DE CASO DA EMPRESA SOFTPLAN. 70f. Trabalho de Conclusão de Curso (Bacharel em Sistema de Informação) - Universidade do Sul de Santa Catarina, 2020. Disponível em: https://repositorio.animaeducacao.com.br/bitstream/ANIMA/10927/1/AUTOMA%C3%87%C3%83O%20DE%20TESTES%20DE%20SOFTWARE-ESTUDO%20DE%20CASO%20DA%20EMPRESA%20SOFTPLAN-TCC_MATHEUS%20FERNANDES-SAMUEL%20TOMKELSKI%20FONSECA.pdf. Acesso em: 26 abr. 2023.

ISTQB - International Software Testing Qualifications Board. **Foundation Level Syllabus** – BSQTB, 2023. Disponível em: https://bcr.bstqb.org.br/docs/syllabus_ctfl_3.1.1br.pdf. Acesso em: 25 abr. 2023.

ITERIS. **Três vantagens do investimento em testes automatizados**. 2021. Disponível em: https://www.iteris.com.br/blog/tres-vantagens-investimento-testes-automatizados/#:~:text=A%20utiliza%C3%A7%C3%A3o%20de%20testes%20automatizados,a%20experi%C3%AAncia%20de%20seus%20usu%C3%A1rios. Acesso em: 02 mai. 2023.

LOGAP. **O que é teste automatizado**: importância e como começar a utilizá-los no dia a dia. 2022. Disponível em: https://logap.com.br/blog/o-que-e-teste-automatizado/. Acesso em: 02 mai. 2023.

LOURENÇO, Rony de Sena. **Automação de Testes para um Sistema de E-commerce**. 52f. Monografia (graduação) - Curso de Engenharia de Computação, Centro de Tecnologia, Universidade Federal do Rio Grande do Norte, 2022. Disponível em: https://repositorio.ufrn.br/bitstream/123456789/50670/1/TCC_Rony.pdf. Acesso em: 30 abr. 2023.

KHAN, Shahnawaz. **Advantages and Disadvantages of Cypress (End to End Testing Tool) before choosing it as your Testing Automation Tool**. 2021. Disponível em: https://skakarh.medium.com/advantages-and-disadvantages-of-cypress-end-to-end-testing-tool-before-choosing-it-as-your-347b6436dec8. Acesso em: 26 abr. 2023.

MEDIUM. **Testes de regressão de UI utilizando Selenium Webdriver**. 2015. Disponível em: https://medium.com/netcoders/testes-de-regress%C3%A3o-de-ui-utilizando-selenium-webdriver-8db40b6347af. Acesso em: 03 mai. 2023.

MICROFOCUS. **ALM/Quality Center.** 2023.Disponível em: **https://www.microfocus.com/pt-br/products/alm-quality-center/overview**. Acesso em: 04 mai. 2023

MICROFOCUS. **Micro Focus Premium Support.** 2023.Disponível em: **https://www.microfocus.com/pt-br/services/premium-support**. Acesso em: 04 mai. 2023.

MICROFOCUS. **UFT One.** 2023.Disponível em: **https://www.microfocus.com/media/data-sheet/uft-one-ds.pdf**. Acesso em: 04 mai. 2023.

MICROFOCUS. **Unified Functional Testing (UFT) Tests.** 2023.Disponível em: **https://www.microfocus.com/documentation/silk-central/200/en/silkcentral-help-en/GUID-531809BA-688F-41D5-BDB2-FCE786A284CE.html**. Acesso em: 04 mai. 2023.

MONTEIRO, Michele. **Descomplicando a automatização de testes com Robot Framework**. 2020. Disponível em: https://medium.com/riachuelo/descomplicando-a-automatiza%C3%A7%C3%A3o-de-testes-com-robot-framework-af793f590ef1. Acesso em: 26 abr. 2023.

PESSOA, Clinton Hudson Moreira. **AVALIAÇÃO DE FERRAMENTAS DE AUTOMAÇÃO DE TESTE: UMA ANÁLISE DOCUMENTAL E COMPARATIVA**. 74f. Monografia (Bacharel) - Curso de Engenharia de Software, Instituto de Ciências Exatas e Tecnologia, Universidade Federal do Amazonas, 2020. Disponível em: https://riu.ufam.edu.br/bitstream/prefix/5831/2/TCC_ClintonPessoa.pdf. Acesso em: 15 mai. 2023.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**, uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

PRIME CONTROL. **10 ferramentas de automação de testes mais usadas**. 2018. Disponível em: https://www.primecontrol.com.br/10-ferramentas-de-automacao-de-testes-mais-usadas/. Acesso em: 26 abr. 2023.

RIOS, Emerson; MOREIRA FILHO, Trayahú. **Teste de software**. 2. ed. rev. e ampl. Rio de Janeiro: Alta books, 2006. 222 p. ISBN: 9788576081289.

ROBOTFRAMEWORK. **Foundation.** 2022.Disponível em: **https://robotframework.org/foundation**. Acesso em: 03 mai. 2023

ROBOTFRAMEWORK. **Resources.** 2022.Disponível em: **https://robotframework.org/?tab=libraries#resources**. Acesso em: 04 mai. 2023

ROLLWAGEN, André Fernando et al. **COMPARATIVO ENTRE FERRAMENTAS DE AUTOMAÇÃO DE TESTES DE SOFTWARE PARA SISTEMAS WEB**. Salão do Conhecimento – Unijuí, Rio Grande do Sul, 20 a 23 de outubro de 2020. Disponível em: https://publicacoeseventos.unijui.edu.br/index.php/salaoconhecimento/article/view/18489/17223. Acesso em: 20 mai. 2023.

SELENIUM. **Grid**. 2023. Disponível em: https://www.selenium.dev/documentation/grid/. Acesso em: 03 mai. 2023.

SELENIUM. **The Selenium Browser Automation Project**. 2023. Disponível em: https://www.selenium.dev/documentation/. Acesso em: 03 mai. 2023.

SELENIUM IDE. **Selenium IDE**. 2019. Disponível em: https://www.selenium.dev/selenium-ide/. Acesso em: 03 mai. 2023.

Silva, Aylma Benício da. **Estratégia na qualidade de software**: um comparativo sobre as ferramentas gerenciais: Jira x ALM. Trabalho de Conclusão de Curso (Especialização em Gestão e Qualidade em Tecnologia da Informação e Comunicação) - Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco - campus Jaboatão dos Guararapes, 2022. Disponível em: https://repositorio.ifpe.edu.br/xmlui/bitstream/handle/123456789/817/Tcc%20Aylma%20Ben%C3%ADcio-%20incluso%20ficha%20catalogr%C3%A1fica.pdf?sequence=1. Acesso em: 15 mai. 2023.

SILVA, Gizele. **O que é Selenium?**. 2023. Disponível em: https://coodesh.com/blog/dicionario/o-que-e-selenium/. Acesso em: 25 abr. 2023.

SILVA, Lisiane Gonçalves da. **Analise das atividades de teste de software quanto a sua aplicabilidade e importância em empresas privadas**. Monografia (Monografia em ciências exatas e da terra). – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – Nova Cruz/RN, 2022. Disponível em: https://memoria.ifrn.edu.br/bitstream/handle/1044/2210/TCC-Final-%20Lisiane%20Gon%C3%A7alves%20da%20Silva.pdf?sequence=1&isAllowed=y. Acesso em: 15 mai. 2023.

SILVA, Mario Luis Moreira da. DALLILO, Felipe Diniz. **Uma visão geral sobre automação de testes**. Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 04, Ed. 12, Vol. 01, pp. 117-130. Dezembro de 2019. ISSN: 2448-0959. Disponível em:

https://www.nucleodoconhecimento.com.br/tecnologia/automacao-de-testes. Acesso em: 09 mai. 2023.

SILVA, Maxwell Anjo da. **Desenvolvimento de testes automatizados e testes manuais para um sistema web**: uma análise comparativa. Monografia (Monografia em Ciência da computação) – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, 2021.

SOMMERVILLE, Ian et al. **Engenharia de software**.[Sl]. Pearson Education, v. 19, p. 23, 2011

TEST AUTOMATION TOOLS. **Pros and Cons of Robot Framework**. 2023. Disponível em: https://testautomationtools.dev/pros-and-cons-of-robot-framework-general-overview/#advantagesDisadvantages. Acesso em: 26 abr. 2023.

UMAR, Mubarak Albarka; ZHANFANG, Chen. **A Study of Automated Software Testing**: Automation Tools and Frameworks. International Journal of Computer Science Engineering (IJCSE), 2019. Disponível em: https://www.researchgate.net/publication/338282426_A_Study_of_Automated_Software_Testing_Automation_Tools_and_Frameworks. Acesso em: 26 abr. 2023.

VERICODE. **Testes não funcionais:** o que são e como podem ajudar a melhorar as aplicações?. 2022. Disponível em:**https://blog.vericode.com.br/testes-nao-funcionais-o-que-sao-e-como-podem-ajudar-a-melhorar-as-aplicacoes/.** Acesso em: 04 mai. 2023

VIEIRA, Josimar de Souza. **AVALIAÇÃO E COMPARAÇÃO ENTRE FRAMEWORKS DE DESENVOLVIMENTO DE TESTES**. Monografia (Bacharel em Ciência da Computação) - Curso de Ciências da Computação, Universidade do Sul de Santa Catarina – UNISUL, 2021. Disponível em: https://repositorio.animaeducacao.com.br/handle/ANIMA/19950. Acesso em: 26 abr. 2023.

WASHINGTON, Vanita. **AUTOMATED TESTING WITH ROBOT FRAMEWORK – HERE ARE THE THINGS YOU NEED TO KNOW**. 2021. Disponível em: https://www.topcoder.com/thrive/articles/automated-testing-with-robot-framework-here-are-the-things-you-need-to-know. Acesso em: 26 abr. 2023.

WIEST, Ieda Rosana Kolling et al. **BENEFÍCIOS DA AUTOMAÇÃO DE TESTES COM O USO DA FERRAMENTA KATALON STUDIO**. Salão do Conhecimento – Unijuí, Rio Grande do Sul, 26 a 29 de outubro de 2020.