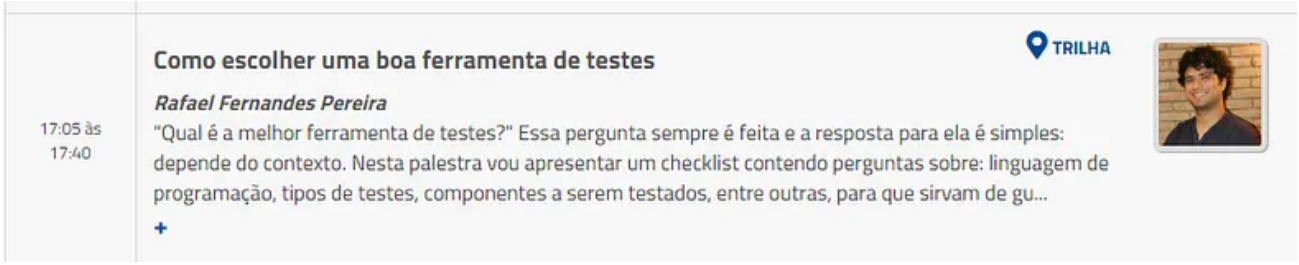
Published in Dev to QA

Rafael Fernandes Pereira Follow
Feb 7, 2021 · 13 min read

Save

Como escolher a melhor ferramenta de automação de testes

Atualização: bom, esse texto virou uma apresentação no TDC Innovation em março de 2021, na trilha de testes! A apresentação não ficou idêntica ao texto original, mudei algumas coisas e adicionei mais informações, além de criar um checklist para ajudar na escolha da melhor ferramenta de automações de testes. Então o texto abaixo é o que foi apresentado no TDC, porém mantive toda a essência do texto original. Segue o link, para mostrar que é verdade esse bilhete https://thedevconf.com/tdc/2021/innovation/trilha-testes



Olha a palestra desse artigo 😍

. . .

Foto de suntorn somtong no Pexels

> *Qual é a melhor ferramenta para automatizar testes?*

Cypress ou Selenium IDE? Robot Framework ou Appium? Este tipo de pergunta está sempre em debate, gerando discussões em fóruns, comunidades, entre influencers, vídeos de youtube, episódios de podcasts, etc. Mas qual a resposta? é bem simples: **depende do contexto**. Essa pergunta é o mesmo que perguntar o que é melhor? um martelo ou um serrote?

Martelo vs Serrote: quem ganha?

A resposta é: vai depender do que você quer fazer: pregar um quadro na parede, serrar uma mesa, tirar um prego da cadeira, etc.

Então quero demonstrar o que devemos considerar ao escolher ferramentas de automações de testes, para que elas sejam úteis dentro do nosso contexto.

Mas antes...

O que é automação de testes?

Uma automação de teste é quando fazemos o uso de software para controlar a execução do teste de software. Ou seja, ao invés de executar manualmente um teste, como logar no app do seu projeto, adicionar um item no carrinho de compras, ir ao carrinho de compras, remover o item e validar se o carrinho está vazio poderia ser feito através de um software, automatizando este processo.

Quais são as ferramentas de automação de testes?

Existem diversas ferramentas de automação de testes, para diversas linguagens, para diversos cenários, como:

- Selenium IDE para testes em sites
- Appium para testes de aplicativos mobile
- Rest Assured para testes de api, para Java
- Cypress que faz vários tipos de testes, em Javascript
- Visual Studio Test Professional e TestComplete, como ferramentas pagas
- Robot framework para Python, Watir para Ruby, Karate para Java e Protractor para Angular.

Por que devemos automatizar os testes?

A automação pode nos economizar tempo, mas também gastar mais tempo... Ela pode trazer benefícios e problemas.

A automação vai nos ajudar quando:

- Precisamos temos uma grande repetição de testes a cada entrega realizada
- Queremos reduzir o tempo de execução de testes de regressão. É bem comum ouvirmos por aí que “times ágeis” trabalham com uma sprint de atraso por conta de testes. Mas isso é para outro post.

- Queremos também mais rapidez ao procurar erros.
- Precisamos simular uma grande quantidade de usuários na aplicação, seja para performance, seja para testar comportamento. No caso do comportamento, imagine a situação: um e-commerce em que você precisa testar o comportamento quando um produto acaba o estoque e há vários usuários tentando comprar esse produto.
- Entre outros benefícios.

Mas não há só benefícios, há problemas também! Vamos ter problemas quando:

- Automatizamos funcionalidades que são alteradas constantemente.
- A manutenção e criação de novos testes de automação são demorados.
- A maioria dos, ou todos os, testes estão concentrados em testes ponta a ponta.

Então antes de ver qual a ferramenta, analise se é necessário uma automação dos testes.

Lembre-se: a automação **PRECISA AGREGAR VALOR AO PROJETO!** Se a automação não agrega valor ao projeto, será tempo e dinheiro jogado fora, onde poderia ser direcionado para outros pontos de qualidade.

Bom, finalizado as explicações, vamos ao que interessa.

. . .

O que preciso considerar ao escolher uma ferramenta de automação de testes?

Uma vez decidido que vai ter testes automatizados, então como escolher uma ferramenta que realmente agregue valor, que faça diferença dentro do

projeto e da empresa? Ao invés de respostas prontas, esse artigo vai trazer mais perguntas a serem feitas e, com a resposta do seu contexto, irá dar um norte a escolha da ferramenta ideal para seu contexto.

1) Qual é a stack do meu projeto?

Por mais que tenhamos nossas preferências, escolher uma stack diferente do projeto em que você está atuando pode acarretar em diversos problemas, como:

- Na hora de alguma dificuldade técnica, não vai ter ajuda dos desenvolvedores de como resolver ou como encontrar a solução. Então pensa na situação: um time que só mexe com Java e você escolhe uma ferramenta feita em Ruby, vai ser muito difícil de ter ajuda quando houver problemas.
- Falando em desenvolvedores, se eles irão ajudar a automatizar os testes, a curva de aprendizado é alta demais, pois vai precisar ensinar os conceitos da linguagem, ao invés de explicar apenas como funciona a ferramenta.
- Além disso, a empresa pode ter uma cultura de desenvolvimento voltada a uma certa stack. Eu trabalhei em uma empresa onde a linguagem de programação era PHP, pois o dono gostava da versatilidade, etc. E era muito difícil levar propostas de outras linguagens. Então, se houver isso, vai ser mais difícil levar uma ferramenta que é em uma stack diferente.

2) Quais são os componentes a serem testados?

O que será testado? Um aplicativo web? Banco de dados? API's? Batch? Saber qual componente serão feitos os testes será importante para não usar um martelo onde é necessário uma chave de fenda.

Por exemplo: se os testes são em API Rest, não há motivos para eu usar um Selenium IDE. Ou o Appium para testar banco de dados. As ferramentas não foram feitas para isso.

3) O quão fácil é a escrita dos testes? E a massa de dados, é facilmente inserida na ferramenta?

Não adianta ter uma boa ferramenta se ela for complexa e difícil de se escrever os testes, ou que a massa de dados seja bem difícil de ser inserida. Tudo que é mais complexo leva mais tempo para ser feito. E, por experiência, se leva mais tempo as chances da ferramenta não ser mais usada é alta.

4) A ferramenta possui bons relatórios, ou a opção de inclusão de relatórios?

Uma ferramenta que só informa que nem todos os testes foram sucesso pode ser motivos de desclassificação, pois as pessoas que vão analisar e corrigir o problema vão querer são informações mais precisas e ricas sobre o teste que deu erro, como massa usada, que tipo de teste foi usado, qual foi o critério de aceite, ou caso de teste, que deu problema, etc.

Quanto mais informações as pessoas que forem corrigirem os problemas tiverem, melhor.

5) A ferramenta tem integração com DevOps?

Imagina a situação:

> *É sábado, você está em sua casa, fazendo stories sobre um pão de queijo e café fresquinho que acabou de comer, seu chefe liga pedindo para você executar todos os testes automatizados que estão em seu computador, pois o time está trabalhando ao final de semana e precisa realizar uma entrega urgente.*

Isso não iria acontecer se a automação estivesse na esteira de DevOps, pois no processo de build do projeto, iria rodar os testes e você poderia continuar com seus stories.

Então considerar a integração com a esteira de DevOps da sua empresa ajuda a não ter uma dependência com você ou com quem tiver o projeto de automação.

6) Há possibilidade de customização dentro da ferramenta?

Cada projeto, cada empresa, por mais que os ramos sejam iguais, tem suas particularidades. E essas particularidades podem trazer a necessidade de customizações. Então pegando os três itens anteriores, tendo a possibilidade de customização você pode:

- Criar a facilidade na escrita do teste.
- Criar um plugin ou biblioteca para inserir a massa de dados.
- Alterar os relatórios que têm na ferramenta.
- Desenvolver uma solução para o DevOps da sua empresa.

A customização será importante para os casos particulares dentro de seu projeto.

7) A ferramenta está homologada dentro da sua empresa?

O que não falta hoje em dia são pessoas tentando invadir as empresas e tirar proveito. E muitas empresas têm investido fortemente na área de segurança cibernética. Com isso, não se pode sair usando qualquer software por aí sem a validação da área de segurança. E a análise de segurança não é algo simples que é feita do dia para noite: é um processo demorado.

Tenha isso em mente ao escolher uma ferramenta: se já houverem ferramentas homologadas dentro da empresa, veja se compensa usar. Se realmente não compensa, veja com o time de segurança qual é o processo e qual o tempo médio que leva uma nova homologação. Se o tempo não for um problema para suas automações, vá em frente. Porém, dependendo do tempo e da burocracia, é melhor ficar com as que já foram homologadas.

8) A empresa está disposta a pagar por alguma ferramenta?

Ferramentas pagas tem suporte do fornecedor. Pode ser que essa ferramenta tenha integração com ferramentas de gestão (jira, por exemplo), suporte caso haja algum problema técnico, adaptação ao contexto em que você está inserido, relatórios personalizados, etc. Então, se houver a oportunidade, analise e levante essa possibilidade.

Checklist!

Com as dúvidas anteriores, juntei todos, separei alguns e criei um checklist para auxiliar na decisão, para comparar ferramentas.

Esse checklist é separado por: dúvidas, que foram extraídas dos tópicos anteriores, o peso de cada pergunta e as ferramentas a serem comparadas.

Ao final temos a contabilização de itens positivos e negativos, soma do total de positivos e negativos e, ao final, o resultado das somas.

Para explicar melhor o funcionamento desse checklists, vamos imaginar o seguinte projeto:

- A empresa é do ramo financeiro
- O time é de backend e o projeto é focado em APIs Rest.
- Há também um batch que faz atualizações no banco de dados
- A stack é Java e Spring boot como linguagem de programação, Oracle como banco de dados e tudo está na cloud, dentro da AWS.
- A massa de dados é externa, via outra API.

E o que iremos analisar é a viabilidade entre as ferramentas Cypress, Robot Framework ou RestAssured. Essa análise é por conta que eu tenho experiência com as três ferramentas.

Dúvidas

As dúvida foram divididas em:

- A mesma stack do projeto? A stack pode ser substituída por linguagem, esteira, banco de dados, etc.
- Teste app mobile?
- Teste website?
- Teste API?
- Teste desktop?
- Teste banco de dados?
- Teste batch?
- Fácil escrita de testes?
- Inserção de massa de dados?
- Relatórios?
- Integração com DevOps? Aqui pode ser: A ferramenta possui integração com qualquer devops ou com a esteira da empresa.
- Possibilidade de customização?
- Homologação na empresa?
- Ferramenta paga?

Peso

Na coluna de peso coloquei de 0 a 5. Zero para quando a pergunta não fizer sentido para seu contexto. Por exemplo: em um time que precisa de uma ferramenta para API's, a ferramenta não precisa testar app mobile. Depois de 1 a 5, para os casos da pergunta se encaixar no contexto. Pegando o

exemplo anterior: O time precisa que a ferramenta faça testes em APIs.

No projeto de exemplo, precisamos que:

- Precisa que seja a mesma stack do projeto, principalmente na questão da linguagem, pois os desenvolvedores irão ajudar na automação dos testes. Então o peso é 5.
- O projeto é de APIs e possui um batch, porém o batch não é tão importante pois a sua ação, que é atualizar cadastros, pode ser testado via API. Peso 5 para testes de API e 2 para testes de batch.
- Como os dev's irão ajudar a automatizar, a ferramenta deve ter a possibilidade de customização, pois se precisar de alguma coisa nós podemos personalizar da forma que precisarmos. Peso 5
- A escrita dos testes deve ser fácil, assim como deve ter a inserção de massa de dados. Mas com a customização, se não for fácil, iremos criar. Peso 3.

- Relatórios, para esse momento, são pouco importantes. Peso 1.
- A integração com devops, que é AWS, é importante, pois ninguém quer ficar executando a cada build do time todo processo. Peso 4.
- A homologação dentro da empresa é importante para esse momento e não teremos tempo para homologar outra ferramenta. Sofremos ataques demais no ano passado e uma nova homologação tende a levar cerca de 3 meses. Peso 5.
- Não queremos ferramentas pagas pois somos uma empresa do ramo financeiro, logo o lema é: “economizar no que puder”. Peso 0.
- Não precisamos que a ferramenta faça os testes de app mobile, website e desktop. Peso 0.
- E, por último, os testes de banco de dados podem ser feitos diretamente pelas API’s. Peso 0.

Aqui uma dica para fazer as pontuações: chame outras pessoas para ajudarem nos pesos, como dev’s e analistas de negócios, ou PO.

Ferramentas

Na coluna de ferramentas, onde você vai colocar o nome da ferramenta, ou framework, ou biblioteca, teŕa apenas dois valores: Sim e Não. Sim representando os positivos, Não os negativos.

Na nossa análise temos:

Coloquei no gist pois é mais fácil visualizar lá em leitores de tela.

Total de positivos e negativos

Simplesmente é a contabilização de itens marcados como Sim e Não, para

serem os totais de positivos e negativos (nessa ordem).

Se formos apenas considerar os totais, o Cypress estaria ganhando. Mas só isso não basta, pois o peso não faria sentido.

Soma dos totais e resultado das somas

Aqui mostra as somas dos pesos, divididos em positivos e negativos. Para nosso exemplos, vamos ter:

E, ao final, temos o resultado da somas, que é a subtração entre a soma dos

positivos pela soma dos negativos:

> *Resultado das somas = Soma do total de positivos — Soma do total de negativos*

E, assim, temos:

Bem, se formos considerar apenas a última linha, teríamos o RestAssured como o ganhador de tudo. Mas olhando para as linhas de totais, o Cypress ganha e na pontuação, se formos analisar, apenas perde pontos por não ser da mesma linguagem de programação.

Aqui, novamente, entra a análise: será que não vale a pena conversar com o time para eles aprenderem Javascript e, consequentemente, Cypress? Ou será que eu consigo dar todo suporte necessário caso escolha o Cypress. Mais uma vez, depende do contexto. Então vá e converse com o time sobre esses pontos.

. . .

Esse checklist está disponível no meu github, nesse link, que também estará no final.. Ele é um template, então adeque ao seu contexto! Se precisar adicionar mais perguntas, alterar o peso, etc, está mais do que aberto. Que ele possa servir não só de ferramenta para usar no dia-a-dia, mas uma inspiração para resolver algum conflito de escolhas dentro do seu projeto.

. . .

Conclusão

Não existe bala de prata. Não existe algo único que irá resolver seus problemas. Cada ferramenta tem seu contexto, sua área de atuação, seu propósito e quais problemas ela resolve. Saiba o contexto, pois assim a ferramenta irá trazer valor ao projeto.

> *Nunca vi alguém defendendo chave de fenda para colocar pregos na parede.*

. . .

Erratas

- No artigo original, eu tinha escrito sobre os testes na pirâmide de testes, como uma pergunta a ser considerada no seu contexto. Porém, depois de ler e reler várias vezes o artigo e a apresentação, não estava fazendo sentido essa pergunta, pois somente ela dá um artigo. Em breve irei escrever sobre a distribuição dos testes nas camadas da pirâmide. Ou

batata de testes.

- Além disso, no final do artigo original haviam os vídeos do Akita e o que inspirou a conclusão do artigo. Segue o original:

. . .

[...]

Para finalizar, algumas reflexões:

Recomendo assistir a série de vídeos do Fabio Akita sobre "*Sua linguagem de programação NÃO É Especial*"

No começo do primeiro vídeo ele diz:

> *[...] linguagens de programação não são filosofias de vida. Linguagens de programação são meras ferramentas. Ferramentas foram feitas para servir seus mestres, e não o contrário. Linguagens de programação não merecem lealdade e definitivamente não são religiões. Eu não sou definido pela minha linguagem de programação favorita. Você não é sua linguagem de programação.*

Eu concordo plenamente com isso e vou além: ferramentas e frameworks também não são especiais. Na citação acima podemos trocar por ferramentas ou frameworks.

Não existe bala de prata. Não existe algo único que irá resolver seus problemas. Cada ferramenta tem seu contexto, sua área de atuação, seu propósito e quais problemas ela resolve.

Nunca vi alguém defendendo chave de fenda para colocar pregos na parede.

[...]

. . .

Links que ajudaram a escrever esse artigo e a apresentação

O link para o download do checklist está aqui também.

E algo que faltou tanto no artigo original foram as fontes que eu procurei

para escrever. Seguem abaixo:

- https://pt.wikipedia.org/wiki/Automa%C3%A7%C3%A3o_de_teste
- https://dzone.com/articles/how-to-select-the-best-automation-testing-tool#:~:text=The%20automation%20tool%20must%20support,browser%20testing%20of%20web%20applications.&text=Technical%20support%20and%20assistance%20are,technical%20support%20for%20your%20tool
- https://www.saviantconsulting.com/blog/4-steps-select-test-automation-tool.aspx
- https://www.softwaretestinghelp.com/automation-testing-tutorial-4/
- https://www.katalon.com/resources-center/blog/automation-testing-tool-strategy/
- https://testsigma.com/blog/10-points-to-help-you-choose-the-right-test-automation-tool/
- https://www.guru99.com/testing-automation-why-right-tools-are-necessary-for-testing-success.html

Test Automation Testing