Um estudo comparativo de características das ferramentas de automação de teste end-to-end: Cypress vs QA Wolf vs TestCafé

Rebeca M. de C. T. Figueiredo[1], Raiane Eunice S. Fernandes[1], Raquel M. de C. T. Figueiredo [1], Clênia Melo Araujo [1], Jislane S. S. de Menezes[1]

[1]Coordenadoria de Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de Sergipe (IFS)
Aracaju - Sergipe - Brasil

{rebeca.figueiredo043, raiane.fernandes069}@academico.ifs.edu.br

raquelmctf@dcomp.ufs.br,{clenia.araujo,jislane.menezes}@ifs.edu.br

Abstract. *Faced with the need for robust and high quality software, testing tools were created to help detect failures and meet the required requirements. A comparison of the features between End-to-End testing tools, Cypress, TestCafé and QA Wolf, analyzed documentation, test environment preparation and comparison between their applications to determine the strengths and weaknesses of the tools. The results show that the easiest tool to use is QAWolf, while TestCafe requires solid knowledge of programming and testing domain, and finally, Cypress has more features but requires advanced knowledge of testing and programming.*

Resumo. *Diante da necessidade de softwares robustos e de alta qualidade, ferramentas de teste foram criadas para ajudar a detectar falhas e atender aos requisitos exigidos. Um comparativo das características entre as ferramentas de teste End-to-End, Cypress, TestCafé e QA Wolf, analisou documentação, preparação do ambiente de teste e comparação entre suas aplicações para determinar os pontos fortes e fracos das ferramentas. Os resultados mostram que a ferramenta mais fácil de usar é o QAWolf, enquanto o TestCafe requer sólidos conhecimentos de programação e domínio de testes e, por fim, Cypress tem mais recursos, mas requer um conhecimento avançado de testes e programação.*

1. Introdução

A fim de aumentar a confiabilidade dos sistemas, a Engenharia de *Software* surgiu para sistematizar o processo de desenvolvimento, de modo que a qualidade das aplicações pudesse ser replicável, independentemente do contexto e domínio de aplicação [Sommerville 2011]. Este fato surgiu após a crise do *software* da década de 1960, quando haviam diversos problemas enfrentados no processo de desenvolvimento, implementação e manutenção do *software*, acarretando em entregas com baixa qualidade [Maffeo and da Silva 1992].

Para garantir e melhorar a qualidade dos produtos, foram desenvolvidas ferramentas de automação de testes. Segundo [de Sousa Veloso et al. 2010], a escolha de uma ferramenta de automação de testes inapropriada desperdiça tempo no cronograma

de testes montado durante o planejamento da lista de tarefas. Esse planejamento leva em consideração o desenvolvimento das tarefas, as tarefas de interrupção, o tempo que o testador leva para validar as tarefas e o tempo de execução previsto no cronograma, antes de fazer a implantação do projeto em produção.

A existência de diversas ferramentas de automação faz com que seja de extrema importância entender seus benefícios e suas desvantagens. Dependendo da ferramenta a ser utilizada, a construção do teste automatizado pode ser mais produtivo quando comparado a outras ferramentas, especialmente quando são testadas aplicações mais complexas.

Assim, com o intuito de experimentar ferramentas recentes, lançadas na última década, este artigo tem como objetivo realizar um comparativo entre as características de ferramentas de teste *Cypress, TestCafé* e *QA Wolf*, usadas em testes *end to end* para sistemas *web*.

2. Ferramentas de Automação para Teste *End-to-End*

O processo de teste geralmente envolve uma combinação de testes manuais e automatizados. No teste manual, um testador executa o programa com alguns dados de teste e compara os resultados com suas expectativas; ele anota e reporta as discrepâncias aos desenvolvedores do programa. Em testes automatizados, os testes são codificados em um programa que é executado cada vez que o sistema em desenvolvimento é testado [Sommerville 2011].

Dessa forma, as ferramentas automatizadas são capazes de simplificar o desenvolvimento de testes executáveis, além de auxiliar a descobrir rapidamente se o sistema está funcionando conforme o esperado e, por serem automatizadas, podem ser executados sem intervenção humana.

O teste *End-to-End* (E2E) conhecido também como teste do sistema, teste de interface ou teste de ponta-a-ponta, pode ser definido como uma série de testes diferentes que exercitam totalmente o sistema, cujo objetivo principal é verificar se os elementos integrados desempenham plenamente as funções atribuídas a eles pelo sistema [Pressman 2009]. Em razão disso, o teste *End-to-End* destina-se a fornecer uma visão geral da funcionalidade do sistema, auxiliar na tomada de decisão e pode ser usado para verificar a sua conformidade com as normas estabelecidas. Sendo, portanto, considerado um tipo avançado de teste.

As ferramentas de automação de teste E2E escolhidas para usar neste trabalho foram: *Cypress, QA Wolf e TestCafé*. Todas elas possuem versão gratuita e são desenvolvidas para *JavaScript* e *TypeScript*.

2.1. *TestCafé*

Ferramenta de testes E2E, *web* e *mobile, open source*, sob licença *MIT (Massachusetts Institute of Technology)*, desenvolvida pelo *DevExpress*, baseada em *Node.js*. Ela utiliza as linguagens de programação *JavaScript e TypeScript* com versões gratuita e paga, mantidas pela comunidade do *StackOverflow*. Seu sistema é *open-source* desde o final do ano de 2016. Neste trabalho foi utilizada a versão 1.20.1.

2.2. *QA Wolf*

É uma ferramenta de automação de testes E2E para *web, open source*, sob a licença *Apache*, desenvolvida para pessoas sem conhecimento em automação de testes e para profissionais da área de testes. Baseia-se no *framework Playwright* da *Microsoft*, sem necessidade de instalação e configuração com versão gratuita e versão paga [Qa Wolf (org.) 2022]. Foi fundada em 2019 por Jon Perl, Laura Cressman e Scott Wilson. Para as análises realizadas nesta pesquisa foi utilizada a versão 8.11.0.

2.3. *Cypress*

O *Cypress app* é a ferramenta de automação de testes E2E que será abordada neste estudo. Utiliza *JavaScript* como linguagem de programação e para a criação dos *scripts* possui um diferencial, a utilização da sintaxe "cy." antes de cada comando, aceitando comandos do *Node*, permitindo a execução de testes no modo *headless*, executando testes diretamente do navegador [Cypress (org.) 2022] . Para a sintaxe da estrutura dos testes suporta apenas o *framework Mocha*, e para as *assertions* apenas a biblioteca *Chai*.

O *Cypress* apresenta interface própria que exibe exatamente o que está acontecendo durante a execução dos testes. Seu lançamento inicial foi em setembro de 2017. Para a realização das análises foi utilizada a versão 10.8.0.

3. Metodologia

[Wazlawick 2010] argumenta que a caracterização clássica das formas de pesquisa consiste em sua classificação como experimental ou não-experimental. Sendo que, a pesquisa não-experimental consiste no estudo dos aspectos sem a intervenção sistemática do pesquisador e a pesquisa experimental que o pesquisador provocará sistematicamente alterações no ambiente de forma a observar se cada intervenção produz os resultados esperados.

Ao considerar estas definições em relação ao intento desta pesquisa, a mesma pode ser classificada como uma pesquisa não-experimental, cuja abordagem utilizada foi a análise documental das ferramentas de testes automatizados, por meio de pesquisa bibliográfica, obtendo-se dados comparativos entre os tipos de ferramentas de teste em questão.

Para realizar a comparação entre as ferramentas de automação de teste foram definidas três etapas: definição da *IDE* (*Integrated Development Interface*) a ser usada, a linguagem de programação e navegadores. Em seguida, a definição dos critérios de comparação para a análise dos pontos positivos e negativos de cada ferramenta. E por fim a instalação e configuração das ferramentas.

3.1. Definição de Ferramentas

Para preparar o ambiente de teste foi utilizado um computador com Sistema operacional *Windows 10*, portando um processador Intel(R) Core(TM) i7-7500U e RAM instalada de 8,00 GB.

A interface de desenvolvimento escolhida foi o *Visual Studio Code*, um editor de código-fonte independente que executa em sistemas operacionais *Windows*, *macOS* e *Linux* [Visual Sutdio Code (org.) 2022].

A linguagem de programação escolhida foi o *JavaScript*, para utilizar essa tecnologia é preciso ter o *Node.js* instalado. Isso porque de acordo com o site oficial [Node.Js (org.) 2022], ela é uma ferramenta de execução *JavaScript* assíncrona orientada a eventos, projetada para criar aplicativos de rede escaláveis.

O navegador utilizado é o navegador padrão da própria ferramenta, utilizando *Cypress Chrome* e *QA Wolf Chromium*. O *TestCafé* oferece suporte a vários navegadores, portanto, se os usuários quiserem usar um navegador diferente, eles precisam fazer o *download* e instalá-lo. Este estudo usa o navegador *Google Chrome*.

3.2. Critérios de Análise

Como critérios de análise foram definidos os seguintes pontos:

1. Navegadores utilizados pela ferramenta;
2. Grau de dificuldade na preparação do ambiente de teste;
3. Disponibilidade de documentação oficial;
4. Presença de recursos que facilitem a escrita de testes.

Para cada critério foram definidas pontuações. Se a ferramenta utiliza até 2 navegadores foi considerada pontuação 1, para 3 e 4 navegadores, pontuação 2 e acima de 4 pontuação 3. Os níveis de dificuldade foram alinhados em 3 categorias, baixa (3), média (2) e alta (1). Quando associado a quantidade de comandos, até 2 comandos se considera nível de dificuldade baixo (3), para 3 e 4 comandos, nível médio e acima de 4 nível alto. Quanto a disponibilidade de documentação e presença de recursos específicos na ferramenta, que adimitem resposta binária Sim/Não, foram pontuados com valores em 3 e 0, respectivamente.

3.3. Instalação e Configuração das Ferramentas

Para configurar o ambiente, levou-se em consideração algumas similaridades e poucas particularidades descritas a seguir:

1. Precisou-se criar um diretório para o projeto;

2. Em seguida, adicionou-se o diretório do projeto no *VS Code*;

3. No terminal do *VS Code* digitou-se os comandos para iniciar a instalação de cada ferramenta; A Tabela 1 apresenta os comandos executados para esta tarefa. A quantidade de comandos executada foi considerada uma métrica quantitativa para análise do critério instalação.

Tabela 1. Comandos executados durante instalação.

	TestCafé	QA Wolf	Cypress
Comandos para a instalação das ferramentas	npm install npm init -y npm i -save-dev testcafe npm install -g testcafe	npm install npm init -y npm install qawolf	npm install npm init -y npm install cypress

Fonte: Elaborado pelos autores.

Para executar o teste nos navegadores foi preciso configurar o *script*. Ao contrário de outras ferramentas, o *QA Wolf* não exigiu essa configuração. O *TestCafé* exigia o nome da ferramenta, o navegador e o caminho para o arquivo de teste criado pelo usuário. O

Cypress, por outro lado, precisava apenas executar um comando que abria uma *interface* de usuário que permite selecionar o tipo de teste e o navegador.

No *QA Wolf*, não é necessário adicionar o endereço da *web* do site de teste ao comando, mas foi escrito um comando que criou automaticamente um arquivo de teste no navegador *Chromium*. Para o *TestCafé*, foi criado um arquivo para os códigos de teste da *IDE* e foi necessário executar um comando para abrir o navegador selecionado no *script*, mas apenas os navegadores pré-instalados.

4. Análise e Resultados

Para a análise comparativa entre as ferramentas de automação escolhidas levou-se em consideração os seguintes pontos:

1. Execução em navegadores: validar o comportamento da ferramenta em navegadores.

2. Preparação do ambiente de testes: validar se as ferramentas possuem dependências com ferramentas de terceiros e avaliar a complexidade da instalação. Esta etapa foi descrita na seção anterior em que apresenta o processo de instalação e configuração.

3. Documentação das ferramentas: é de extrema importância que a documentação de uma ferramenta seja escrita de forma clara e com todos os passos necessários para instalação e utilização bem sucedidas.

4. Recursos de escrita: outro fator que pode ajudar na popularidade da ferramenta são os recursos que facilitam a escrita dos códigos e ajudam o testador na manutenção e na visualização dos erros ocorridos.

4.1. Execução em navegadores

Nesse sentido, a superioridade do *TestCafé* sobre as outras ferramentas é significativa, permitindo 6 opções de navegadores. O uso de navegadores no *TestCafé* requer sua instalação para executar testes *web* e *mobile*. O *QA Wolf* executa testes da *web* com apenas uma opção de navegador, e o *Cypress* oferece ao usuário três opções que executam apenas testes de navegador.

4.2. Documentação das ferramentas

Cada ferramenta possui um site oficial em inglês e traduzido para outros idiomas, com informações de como instalar, executar e construir testes, explicações sobre *plugins*, ferramentas e outras linguagens de programação, etc. No entanto, a documentação do *QA Wolf* acaba por remeter o leitor a outros sites para uma explicação geral de determinados tópicos abordados nos textos.

4.3. Recursos de escrita

Os recursos analisados para ajudar a escrever testes são: *interface* de execução de teste, *hot reload*, *time travel*, seletor de elemento por meio de sua *interface* do usuário e clique para conversão de código.

A interface de execução de testes é uma opção adicional para o usuário, pois fornece informações sobre testes concluídos e pendentes. O *Hot Reload* é um recurso exclusivo do *Cypress* que permite ao usuário editar e executar códigos simultaneamente; *time*

travel é um recurso encontrado apenas no *Cypress* que permite ao usuário visualizar a execução do teste em uma interface ”antes e depois”. Esse recurso torna mais fácil para o usuário entender o código durante a depuração. Seletores de itens através da própria interface da ferramenta permite que o usuário obtenha os seletores de forma mais rápida. A conversão de clique para código é um recurso exclusivo do *QA Wolf* que permite que usuários com um nível de conhecimento de automação de teste usem a ferramenta e automatizem o teste. Já a versão gratuita do *TestCafé* apresenta uma desvantagem em relação às outras duas ferramentas de automação, pois é necessário criar o arquivo de teste, escrever o código de teste e somente após sua finalização que é possível executá-lo.

Tabela 2. Comparativo das Ferramentas.

	TestCafé	QA Wolf	Cypress
Nível de dificuldade para configurar o ambiente	Médio	Baixo	Baixo
	(2)	(3)	(3)
Quantidade de comandos para executar o navegador	1 comando	2 comandos	1 comando
(Nível de dificuldade)	Baixo	Médio	Baixo
	(3)	(2)	(3)
Opções de navegadores *web*	6	1	3
	(3)	(1)	(2)
Opções de navegadores versão *mobile*	2	0	0
	(1)	(0)	(0)
Documentação	SIM	SIM	SIM
	(3)	(3)	(3)
Interface própria	NÃO	NÃO	SIM
	(0)	(0)	(3)
Hot reload	NÃO	NÃO	SIM
	(0)	(0)	(3)
Time travel	NÃO	NÃO	SIM
	(0)	(0)	(3)
Seletor de elementos via *interface* própria	NÃO	NÃO	SIM
	(0)	(0)	(3)
Conversão de *click* em código	NÃO	SIM	NÃO
	(0)	(3)	(0)
TOTAL DE PONTOS	12	12	23

Fonte: Elaborado pelos autores.

4.4. Discussões

À luz do que foi exposto acima, entende-se que:

Cypress - Apresenta uma instalação demorada por possuir mais etapas na preparação do ambiente. O código de teste deve ser escrito na *IDE* usando seletores e *assertions* - comandos usados em ambiente de desenvolvimento que testam se o código está de acordo com as regras. A execução do teste é monitorada em uma interface do usuário que fornece resultado de teste, tempo de execução, seletor e todos os outros comportamentos comuns do navegador. Além de *logs* com informações diversas que auxiliam o usuário na identificação de erros de execução. Possui ótimos recursos de escrita e documentação oficial completa.

QA Wolf - Possui um *layout* mais simples, com menos etapas e oferece duas formas de escrever código na *IDE*: I) o usuário pode escrever código com seletores e *assertions* ou II) interagir com um único navegador que tenha as mesmas funções de um navegador normal. Não possui interface com o usuário, portanto, o *log* de teste é exibido na *IDE* com poucas informações sobre os testes aprovados e o tempo de execução. Possui poucos recursos escritos e a documentação oficial não ajuda tanto o usuário, pois

é preciso consultar outras documentações/tutoriais para um entendimento mais completo da funcionalidade da ferramenta.

TestCafé - Dispõe de uma configuração um pouco mais simples que o *Cypress*, mas requer que o usuário crie um arquivo de teste a partir do código gerado na *IDE* usando seletores e *assertions* e algumas ferramentas de *script*. Não possui *interface* com o usuário, portanto, um *log* é exibido no terminal da *IDE* para informar o usuário sobre a aprovação dos testes e o tempo de execução; outra desvantagem observada é falta de *hot reload*. Suas vantagens estão na documentação oficial é completa e rica em detalhes, além de ser a única a fornecer testes na versão *mobile*.

De acordo com o comparativo apresentado na Tabela 2, percebe-se que a ferramenta *Cypress* possui mais pontos positivos quando comparada às demais. Sua pontuação elevada se deve aos recursos oferecidos pela sua interface própria.

5. Ameaças à Validade

Devido às características deste tipo de pesquisa, esse trabalho está sujeito a ameaças à validade. Ao considerar a validade interna, percebe-se que este trabalho não utiliza um modelo de avaliação validado para ferramentas de teste, no entanto, a definição das características propostas na análise foi realizada por uma das autoras que é especialista na área de testes.

Quanto à validade externa, uma ameaça relacionada à generalização dos resultados está relacionada ao tamanho da amostra usado na avaliação. A análise utilizou apenas uma amostra, o que estatisticamente não é considerada significante, e o projeto do experimento pode ter sido induzido pelos resultados esperados. Contudo, a discussão dos resultados apresenta uma percepção inicial quando considerada a pouca documentação disponível, especialmente sobre a ferramenta *QA Wolf*. Por fim, não foram realizados testes estatísticos para análise dos dados, devido a limitação de uma única amostra.

6. Conclusão

Motivado por experimentar ferramentas de teste lançadas na última década, este trabalho se propôs a utilizar programas da automação de teste *End-to-End* a fim de compará-los a respeito de suas diferentes características. Deste modo, três ferramentas foram escolhidas e avaliadas para este propósito. Os resultados demonstraram que *Cypress* é a ferramenta que possui mais recursos, como documentação ampla, *logs* claros e interface própria; apesar disto, as outras duas ferramentas também demonstraram suas exclusividades como a facilidade de instalação do *QA Wolf* e a opção de testes na versão *mobile* do *TestCafé*.

Para trabalhos futuros, pode-se realizar uma avaliação com um maior número de respondentes a fim de aplicar testes estatísticos durante a análise. E criar uma API para um sistema *web*, onde será testada a eficiência de cada ferramenta de automação de forma experimental, bem como a criação de cenários e casos de teste.

Referências

[Cypress (org.) 2022] Cypress (org.) (2022). JavaScript End to End Testing Framework. Disponível em: `https://www.cypress.io`. Acesso em: 11 julho 2022.

[de Sousa Veloso et al. 2010] de Sousa Veloso, J., dos Santos Neto, P. d. A., de Sousa Santos, I., and de Sousa Britto, R. (2010). Avaliação de ferramentas de apoio ao teste de sistemas de informação. *iSys-Brazilian Journal of Information Systems*, 3(1).

[Maffeo and da Silva 1992] Maffeo, B. and da Silva, S. B. (1992). *Engenharia de softwere e especificação de sistemas: soluções para quem necessita da informação para agir*.

[Node.Js (org.) 2022] Node.Js (org.) (2022). Node.js is an open-source, cross-platform JavaScript runtime environment. Disponível em: `https://nodejs.org/en/download`. Acesso em: 11 julho 2022.

[Pressman 2009] Pressman, R. S. (2009). *Engenharia de Software-7*. Amgh Editora.

[Qa Wolf (org.) 2022] Qa Wolf (org.) (2022). Qa Wolf: Zero-effort automated QA for startups. Disponível em: `https://www.qawolf.com`. Acesso em: 11 julho 2022.

[Sommerville 2011] Sommerville, I. (2011). Software engineering 9th edition. *ISBN-10*, 137035152:18.

[Visual Sutdio Code (org.) 2022] Visual Sutdio Code (org.) (2022). Download visual studio code - mac, linux, windows. Disponível em: `https://code.visualstudio.com/`. Acesso em: 11 julho 2022.

[Wazlawick 2010] Wazlawick, R. S. (2010). Uma reflexão sobre a pesquisa em ciência da computação à luz da classificação das ciências e do método científico. *Revista de Sistemas de Informação da FSMA*, 6:3–10.