Análise Comparativa de Ferramentas de Testes Automatizados para Aplicações Web

Giovanni B. S. Duarte[1]

[1]Engenharia de Software – Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, MG – Brasil

939967@sga.pucminas.br

Abstract. *Considering the importance of test automation in agile development environments, this study investigates, through both quantitative and qualitative approaches, the implementation efficiency, technical execution performance, and user perception regarding the Playwright, Selenium, and Cypress tools. A total of 45 Software Engineering students were selected and divided into three groups, each responsible for implementing four standardized test scenarios in JavaScript. The results indicate that Cypress achieved implementation times 30.4% faster than Playwright and 32.4% faster than Selenium. In terms of technical performance, Cypress was 65% more efficient than Selenium and 40% more efficient than Playwright. Participants' perceptions reinforced these differences, identifying Cypress as easier to use compared to the other tools. These findings are expected to support software teams in selecting automation tools that align with their project priorities and constraints.*

Resumo. *Considerando a importância da automação de testes em ambientes de desenvolvimento ágil, este trabalho investiga, de forma quantitativa e qualitativa, a eficiência de implementação, o desempenho técnico de execução e a percepção dos usuários em relação às ferramentas Playwright, Selenium e Cypress. Foram selecionados 45 estudantes de Engenharia de Software, distribuídos em três grupos, responsáveis por implementar quatro cenários de teste padronizados em JavaScript. Como resultados foi indentificado que o Cypress apresentou tempo de implementação 30,4% mais rápido que o Playwright e 32,4% mais rápido que o Selenium. No desempenho técnico, o Cypress foi mais eficiente em 65% que o Selenium e 40% que o Playwright. A percepção dos participantes reforça essas diferenças, apontando o Cypress com maior facilidade de uso que as demais ferramentas. Espera-se que os resultados auxiliem equipes de software na seleção de ferramentas de automação alinhadas às prioridades e restrições de seus projetos.*

Bacharelado em Engenharia de *Software* - PUC Minas
Trabalho de Conclusão de Curso (TCC)

Orientador de conteúdo (TCC I): Joana Souza - joanasouza@pucminas.br
Orientador de conteúdo (TCC I): João Pedro Batisteli - 1019080@pucminas.br
Orientador de conteúdo (TCC I): Leonardo Vilela - leonardocardoso@pucminas.br
Orientador acadêmico (TCC I): Cleiton Tavares - cleitontavares@pucminas.br
Orientador do TCC II: Cleiton Tavares - cleitontavares@pucminas.br

Belo Horizonte, 16 de dezembro de 2025.

1. Introdução

A Garantia da Qualidade de Software (GQS) é essencial no desenvolvimento, sendo os testes fundamentais para verificar consistência, desempenho e conformidade com os requisitos [Aburas 2024]. Com a crescente complexidade das aplicações web e a demanda por entregas ágeis [Santos et al. 2024], a automação de testes tem ganhado destaque por aumentar a eficiência, reduzir custos e acelerar lançamentos [Mosleh et al. 2024]. Ferramentas como Playwright, Selenium e Cypress são amplamente utilizadas nesse contexto. O Selenium, amplamente adotado na indústria, é valorizado por seu suporte a múltiplas linguagens e navegadores, embora exija configurações adicionais e não suporte testes móveis nativamente [Pelivani and Cico 2021, Majeed et al. 2021]. O Cypress se destaca pelos relatórios integrados e foco em testes de interface de aplicações web [Pelivani et al. 2022]. Já o Playwright, diferencia-se pelo suporte multiplataforma e pela execução rápida e confiável dos testes [Apriansah and Desanti 2024]. A escolha entre essas e outras ferramentas disponíveis no mercado não é trivial, dependendo do contexto do projeto, requisitos específicos e perfil técnico da equipe.

Embora essas ferramentas ofereçam benefícios distintos, a seleção da mais adequada para um projeto específico permanece uma decisão complexa [Mosleh et al. 2024]. Estudos destacam que fatores como tipo de teste (unidade, integração, sistema, aceitação), plataforma, linguagem de programação e facilidade de uso são critérios relevantes, mas insuficientes para garantir uma escolha otimizada [Aburas 2024]. A literatura reforça que a ferramenta selecionada pode impactar diretamente os resultados de qualidade [Kavaler et al. 2019]. No entanto, **a escassez de dados empíricos que quantifiquem essa associação em diferentes contextos de uso dificulta que equipes de software tomem decisões embasadas sobre quais ferramentas, considerando suas características e limitações, são mais compatíveis com seus objetivos e restrições específicas.** Isso torna desafiador compreender como a eficiência na implementação e na execução de testes automatizados se relaciona com os resultados de qualidade em projetos de aplicações web.

A relevância deste estudo reside na crescente importância dos testes automatizados para a garantia da qualidade em um cenário de desenvolvimento ágil e de entregas contínuas [Santos et al. 2024], onde a velocidade e a confiabilidade são primordiais. A atividade de testes representa uma parcela significativa do orçamento de projetos de software; relatórios da indústria de 2024 indicaram que as empresas destinam cerca de 25% do orçamento de tecnologia da informação (TI) para testes [Aburas 2024]. A detecção precoce de falhas por meio de testes automatizados contribui diretamente para a redução de custos no processo de desenvolvimento, impactando positivamente a satisfação do usuário [Kavaler et al. 2019]. Portanto, compreender a associação entre ferramentas de teste específicas e os resultados de qualidade é essencial para orientar a escolha de ferramentas que estejam associadas a melhores resultados de qualidade [Mosleh et al. 2024], complementando a análise baseada apenas em características técnicas ou custo.

Diante deste problema, o objetivo geral deste trabalho é **realizar um estudo comparativo entre as ferramentas Playwright, Selenium e Cypress, aplicadas a testes em aplicações web**. A escolha dessas ferramentas reflete a diversidade de abordagens encontradas no desenvolvimento de software, as três ferramentas, são voltados majoritariamente para testes de interface de usuário. Comparar ferramentas frequentemente utilizadas em

ambientes de desenvolvimento reais, permite investigar não apenas sua eficiência individual, mas também o impacto prático dessa escolha. Para alcançar este objetivo geral, os seguintes objetivos específicos foram definidos: (i) Identificar qual ferramenta apresenta maior eficiência de implementação, por meio da análise de métricas objetivas. (ii) Avaliar o desempenho técnico das ferramentas na execução dos testes automatizados em ambientes controlados. (iii) Examinar a percepção dos usuários quanto à facilidade de uso, curva de aprendizado e esforço percebido ao utilizar cada ferramenta.

Com base nos resultados obtidos, este estudo demonstrou diferenças quantitativas claras entre Playwright, Selenium e Cypress nos cenários avaliados. O Cypress apresentou, em média, uma implementação 56% mais rápida que o Selenium e cerca de 49% mais rápida que o Playwright na Tarefa 1, além de exigir 50% menos linhas de código que o Selenium ao longo dos cenários. No desempenho técnico, o Cypress também se destacou ao registrar tempos de execução entre 56% e 72% mais rápidos que o Selenium, mantendo desempenho até 63% superior ao Playwright nas tarefas mais rápidas. Em relação à taxa de sucesso das tarefas, o Selenium apresentou maior dificuldade nas tarefas 3 e 4, concluindo corretamente cerca de 80% menos tarefas que Playwright e Cypress nesses cenários. De forma geral, esses resultados indicam que o Cypress oferece o melhor equilíbrio entre velocidade, concisão e estabilidade, enquanto o Playwright apresentou desempenho consistente, e o Selenium, embora robusto, demandou maior esforço de implementação. Essas evidências empíricas reforçam que a escolha da ferramenta depende do contexto, destacando diferentes *trade-offs* entre produtividade, desempenho e experiência do usuário.

A estrutura deste artigo está organizada da seguinte forma: a Seção 2 apresenta a Fundamentação Teórica, abordando os principais conceitos e estudos que sustentam esta pesquisa. A Seção 3 discute os Trabalhos Relacionados, destacando pesquisas semelhantes e suas diferenças em relação ao presente estudo. A Seção 4 descreve os Materiais e Métodos, detalhando o ambiente experimental, as métricas e os instrumentos utilizados. A Seção 5 apresenta os Resultados obtidos e discute esses achados à luz dos objetivos definidos. A Seção 6 aborda as Ameaças à Validade do estudo, e a Seção 7 apresenta as Conclusões e Perspectivas Futuras.

2. Fundamentação Teórica

Nesta seção, são apresentados os principais conceitos e teorias relacionados à Garantia da Qualidade de Software (GQS) e teste de sistemas. Esses elementos são fundamentais para a compreensão da proposta deste trabalho, que visa comparar a eficiência dessas ferramentas em cenários práticos de aplicações web.

2.1. Garantia da Qualidade de Software

A Garantia da Qualidade de Software refere-se ao conjunto de atividades sistemáticas implementadas com o objetivo de assegurar que os processos e produtos de software atendam aos padrões de qualidade previamente definidos [Aburas 2024]. Segundo Sommerville (2011), a GQS é um componente essencial do ciclo de vida do software, especialmente em contextos de desenvolvimento ágil, onde há ênfase na entrega contínua e na detecção precoce de defeitos.

Um dos pilares da GQS são os testes de software e eles podem ser classificados em diferentes níveis, como testes de unidade, integração, sistema e aceitação

[Myers et al. 2012]. Com o crescimento da complexidade das aplicações web, a automação dos testes passou a ser uma estratégia amplamente adotada para aumentar a cobertura, reduzir erros humanos e acelerar o *feedback* ao time de desenvolvimento.

2.2. Testes de sistemas

Os testes de sistema, também chamados de testes end-to-end (E2E), validam o comportamento da aplicação como um todo, verificando fluxos completos do ponto de vista do usuário final [Aburas 2024]. Conforme Myers, Sandler e Badgett (2012), esse nível de teste garante que os componentes integrados funcionem corretamente e atendam aos requisitos definidos. Sommerville (2011) reforça que, em sistemas web, a execução de cenários reais é essencial para identificar defeitos que não aparecem em níveis inferiores de teste. No contexto da automação, ferramentas como Selenium, Cypress, Playwright, Katalon Studio e diversas outras, tornam os testes E2E mais eficientes ao permitir a execução repetível de interações reais em navegadores [Pelivani and Cico 2021, Mosleh et al. 2024].

3. Trabalhos Relacionados

Kavaler et al. (2019) tiveram como objetivo investigar como a adoção de ferramentas de garantia de qualidade JavaScript impacta a evolução de projetos open-source no GitHub. Metodologicamente, os autores analisaram eventos de adoção de *linters*, gerenciadores de dependência e ferramentas de cobertura de código utilizando *logs* do Travis CI, *badges* de repositórios e regressão linear de efeitos mistos, modelando os efeitos antes e depois da adoção. Os resultados mostraram que determinadas ferramentas, como *standardJS*, *coveralls* e *david*, estiveram associadas à redução gradativa do número de issues ao longo do tempo, embora apresentassem um aumento imediato logo após a adoção. Em comparação ao presente estudo, os autores abordam ferramentas de QA e sua influência em projetos reais, enquanto este trabalho se concentra em comparar ferramentas de automação de testes em um ambiente controlado, examinando desempenho, produtividade e experiência do usuário.

O estudo de Pelivani e Cico (2021) teve como objetivo comparar o desempenho do Selenium e do Katalon Studio na execução de testes automatizados de interface. A metodologia consistiu na implementação dos mesmos cinco casos de teste (cerca de 120 passos) em máquinas idênticas, permitindo uma comparação direta do tempo de execução. Como resultado, o Selenium apresentou melhor desempenho (1m29s), enquanto o Katalon foi mais lento (3m16s), diferença atribuída à sobrecarga do uso de Groovy. O estudo também destacou que o Katalon oferece relatórios mais completos e acessíveis que o Selenium. Em contraste, o presente trabalho avalia ferramentas diferentes e adota um desenho experimental com participantes, o que permite analisar não apenas desempenho, mas também facilidade de uso, curva de aprendizado e sucesso das tarefas, aspectos não considerados por Pelivani e Cico.

Majeed et al. (2021) buscaram comparar Selenium, Katalon Studio e Test Project para auxiliar equipes de QA na escolha da ferramenta mais adequada. A metodologia envolveu a avaliação das três ferramentas por meio de uma escala de cinco pontos, considerando critérios como gravação de testes, suporte a dados, relatórios, reusabilidade, colaboração e curva de aprendizado. Os resultados indicaram que Katalon Studio e Test

Project superam o Selenium em funcionalidades como gravação e reprodução, relatórios nativos e suporte a testes data-driven, enquanto o Test Project demonstrou o melhor suporte à colaboração remota. enquanto aos autores tem foco funcional e qualitativo, sem execução prática de testes nem análise temporal, ao passo que o presente trabalho realiza uma comparação empírica baseada em implementação real, coleta de métricas e participação de usuários.

O trabalho de Aburas (2024) teve como objetivo comparar 16 ferramentas populares de automação de testes com base em critérios como tipo, fase do ciclo de vida, linguagem suportada e capacidade de gravação e reprodução. A metodologia consistiu na categorização das ferramentas de acordo com esses critérios, buscando identificar tendências e adequação para diferentes contextos de uso. Como resultado, o autor concluiu que não existe uma ferramenta universalmente superior e que a escolha depende do tipo de teste, da plataforma e da linguagem utilizada. Esse estudo se diferencia do presente trabalho por adotar uma abordagem ampla e descritiva, enquanto este realiza um experimento controlado com três ferramentas específicas, analisando métricas quantitativas e percepção dos participantes.

Mosleh et al. (2024) tiveram como objetivo propor um *framework* para avaliar e comparar ferramentas de teste utilizadas em ambientes ágeis. A metodologia baseou-se em uma revisão estruturada da literatura e na aplicação do *framework* a cinco ferramentas comerciais (QTP, TestComplete, RFT, Silk Test e AutoIt), avaliando critérios de design de testes, interfaces e geração de relatórios. Os resultados mostraram que ferramentas comerciais oferecem cobertura mais ampla de funcionalidades, como testes *data-driven* e *record-playback*, enquanto ferramentas como AutoIt apresentam limitações significativas. Diferentemente desse estudo, que é conceitual e voltado à proposição de um *framework*, o presente trabalho aplica um método experimental envolvendo execução prática de tarefas, análise estatística e percepção dos participantes para comparar ferramentas amplamente usadas no mercado atual.

4. Materiais e Métodos

Este estudo adota uma abordagem predominantemente quantitativa, de caráter exploratório, complementada por dados qualitativos referentes à percepção dos participantes. A dimensão quantitativa vem da análise de métricas objetivas, como tempo de implementação, LOC e tempo de execução, utilizadas para comparar o desempenho das ferramentas em um ambiente controlado. O caráter exploratório decorre do objetivo de investigar e identificar padrões e diferenças práticas entre as ferramentas, sem a intenção de testar hipóteses formais. Os dados qualitativos, por sua vez, enriquecem a análise ao fornecer percepções sobre facilidade de uso, curva de aprendizado e dificuldades encontradas, contribuindo para uma interpretação mais contextualizada dos resultados numéricos.

4.1. Metodologia utilizada

A metodologia foi estruturada a partir de um conjunto de 5 etapas sequenciais, definidas para garantir padronização, reprodutibilidade e correspondência direta com os objetivos específicos do trabalho, demontrado na Figura 1.

Etapa 1. Seleção e capacitação dos participantes. O experimento foi conduzido a partir da seleção de 45 estudantes do curso bacharelado em Engenharia de Software

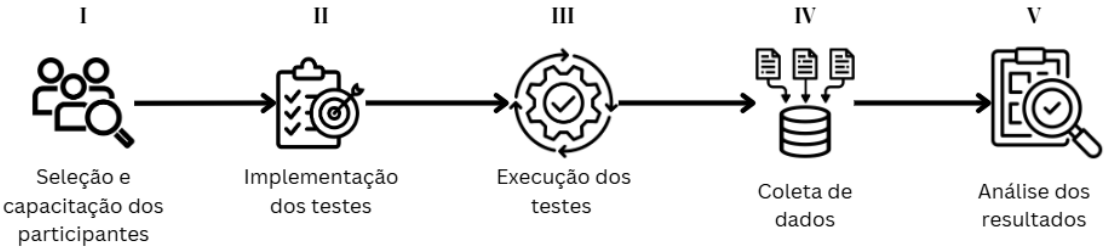

Figura 1. Metodologia

da PUC Minas, distribuídos em três grupos, cada qual responsável pela utilização de uma das ferramentas: 1) Playwright, que é robusta para o desenvolvimento de programas automatizados [Apriansah and Desanti 2024]; 2) Selenium, incluído como um ponto de referência amplamente utilizado na indústria, apesar de exigir configurações adicionais [Aburas 2024]; 3) Cypress, justificado por sua arquitetura inovadora, execução rápida e confiável, e recursos de depuração aprimorados [Pelivani and Cico 2021].

Para garantir padronização, foi feita uma apresentação expositiva de aproximadamente 15 minutos contendo instruções de instalação, apresentação da aplicação web utilizada nos testes, explicação dos cenários, critérios de medição e links de apoio. Os slides e links necessários foram disponibilizados para todos os alunos. Os participantes utilizaram os computadores do laboratório providos pela PUC Minas como ambientes de desenvolvimento, seguindo instruções de configuração unificadas.

Etapa 2. Implementação dos testes. Após a aula expositiva os alunos foram convidados a participar do experimento no qual tiveram que preencher um questionário (demonstrado no Apêndice A) contendo 4 seções. Iinicialmente eles preencheram o termo de consentimento, depois foi feita a coleta do perfil dos participantes. Na terceira etapa eles começaram a implementação de quatro cenários e por fim a coleta das percepções da ferramenta que utilizaram.

Para a implementação dos testes a aplicação selecionada para os experimentos foi "The Internet[1]", da Sauce Labs, devido à sua estabilidade. Sua variedade de componentes interativos permitiu criar cenários padronizados e reprodutíveis, foram esses: login com sucesso, login com falha, adição e remoção de elementos e checkbox interativo. Além da coleta dos arquivos de código dos cenários, cada participante registrou tempo de início e termino de desenvolvimento e o número de linhas de código (LOC) geradas para cada tarefa, fornecendo os dados necessários para avaliar a eficiência de implementação. Para manter a comparabilidade entre grupos, a linguagem de programação padronizada foi JavaScript em todas as ferramentas.

Na última parte do questionário que está com sua estrutura demonstrada no Apêndice A, foi aplicada uma seção para coletar informações relacionadas à percepção de uso das ferramentas. Com isso foram registrados aspectos como facilidade de instalação, curva de aprendizado, clareza da sintaxe, dificuldades encontradas, esforço percebido e experiência geral ao implementar os testes, também foram coletados comentários adicionais dos participantes sobre esses assuntos. Esses dados complementam a análise objetiva, permitindo examinar a facilidade de uso sob a perspectiva do usuário.

[1]https://the-internet.herokuapp.com/

Etapa 3. Execução dos testes. Após a implementação pelos participantes, as suítes de testes geradas foram executadas em um ambiente único e controlado, evitando interferências causadas por diferenças de hardware ou sistema operacional dos participantes. O ambiente utilizado possuía as seguintes especificações: processador Intel® Core™ i7-10700T, 32 GB de RAM e sistema operacional Windows 11 Home (versão 24H2). Nessa etapa, ao executar cada código 5 vezes, fazendo uma média entre elas, foram coletadas as métricas de tempo de execução e estabilidade (sucesso ou falha) dos testes, permitindo avaliar o desempenho técnico de cada ferramenta.

Etapa 4 e 5. Coleta e análise de dados. Os dados foram organizados em planilhas e analisados por meio de estatística descritiva, incluindo média e desvio padrão, além da construção de gráficos e diagramas que permitiram visualizar diferenças de desempenho entre as ferramentas. Os dados qualitativos foram categorizados e avaliados mediante análise simples, de modo a identificar percepções recorrentes que contribuíssem para interpretar os resultados quantitativos. Esse conjunto integrado de procedimentos assegura aderência à metodologia proposta e sustenta a análise comparativa entre Playwright, Selenium e Cypress.

4.2. Métricas

As métricas utilizadas neste estudo foram cuidadosamente definidas para refletir os objetivos centrais da pesquisa, que envolvem avaliar a eficiência de ferramentas de teste automatizado em aplicações web, com foco na qualidade do processo e na adequação prática em ambientes de desenvolvimento reais. Dado o contexto de desenvolvimento ágil e de entregas contínuas, tornou-se fundamental mensurar não apenas aspectos técnicos e objetivos, mas também percepções subjetivas dos usuários, construindo uma análise mais completa e realista sobre o desempenho das ferramentas avaliadas.

Para mensurar a eficiência de implementação, foram adotadas métricas que quantificam diretamente o esforço e o tempo necessários para criar os testes automatizados. O tempo/esforço de implementação, medido em minutos, corresponde ao tempo total que cada participante levou para desenvolver um mesmo conjunto de cenários funcionais com cada ferramenta. Essa métrica busca capturar não apenas a produtividade proporcionada por cada ferramenta, mas também sua curva de aprendizado, reconhecendo que familiaridade prévia com a linguagem ou com a abordagem de automação pode influenciar significativamente esse tempo. Essa avaliação é essencial, considerando que ferramentas mais rápidas de implementar reduzem o tempo de desenvolvimento e os custos associados.

O número de linhas de código, obtido por contadores de linhas (*Count Lines of Code* (cloc)), representa a quantidade de código escrita para implementar os testes. Embora essa métrica não reflita diretamente o esforço humano, ela serve como um importante indicativo da concisão e expressividade de cada linguagem ou *framework*. Um número reduzido de linhas pode estar associado a maior clareza, menor complexidade e menor propensão a erros, o que é particularmente relevante em projetos que lidam com grandes volumes de testes e demandam fácil manutenção ao longo do tempo.

No que se refere à eficiência de execução, foram utilizadas duas métricas principais. O tempo de execução corresponde à duração necessária para rodar a suíte de testes completa em ambiente controlado e padronizado. Essa métrica está diretamente relacionada à agilidade no ciclo de *feedback*, sendo especialmente importante em contextos de

integração e entrega contínuas, onde execuções mais rápidas contribuem para identificar falhas e validar funcionalidades com maior frequência.

Complementando essas métricas objetivas, foram coletadas informações subjetivas sobre a experiência dos participantes por meio de um questionário de avaliação aplicado ao final de cada ciclo de testes. Os aspectos avaliados incluíram a facilidade de uso, o esforço percebido e as principais dificuldades encontradas pelos participantes durante a implementação e execução dos testes. Esses dados fornecem uma visão mais holística e contextualizada do impacto prático de cada ferramenta, contribuindo para uma análise mais completa que vai além dos números e considera a experiência real dos usuários no processo de automação de testes.

5. Resultados

Nesta seção, apresentam-se os resultados obtidos pelas análises dos dados coletados nos experimentos comparativos entre as ferramentas Playwright, Selenium e Cypress.

5.1. Análise do *Background* dos participantes

Ferramenta	N° Participantes	Sexo			Período na universidade				
		Masculino	Feminino	Prefiro Não Informar	4°	5°	6°	7°	8°
Cypress	13	9	3	1	0	0	6	6	1
Playwright	16	14	2	0	16	0	0	0	0
Selenium	14	13	1	0	0	0	14	0	0
Total	43	36	6	1	16	0	20	6	1

Tabela 1. Perfil dos participantes

O experimento foi conduzido com um pequeno grupo de participantes, A amostra inicial é composta por 45 participantes (83% masculino, como mostra a Tabela 1) das 45 respostas enviadas, foram coletadas 43 respostas válidas que abrangem as três ferramentas em estudo. Os dois participantes foram excluidos porque os arquivos enviados pelos participantes, que eram para ser os códigos das tarefas, estavam vazios. Todos os participantes estão cursando Engenharia de Software, variando do 4º ao 8º período. Como demonstrado na Tabela 1 é possível perceber que os participantes que fizeram parte do grupo da ferramenta Playwright, em geral, tinham menos experiência que os outros participantes, sendo todos eles estudantes do 4° período de Engenharia de Software, já os outros participantes variam do 6° ao 8° período.

Ferramenta	N° Participantes	Nível de experiencia prévia com automação de testes?				Nível de conhecimento com a ferramenta				
		1 - Nenhuma	2 - Básica	3 - Intermediária	4 - Avançada	1 - Muito Baixo	2 - Baixo	3 - Intermediário	4 - Alto	5 - Muito Alto
Cypress	13	2	7	3	1	11	1	0	1	0
Playwright	16	8	6	2	0	16	0	0	0	0
Selenium	14	4	5	4	1	11	2	1	0	0
Total	43	14	18	9	2	38	3	1	1	0

Tabela 2. Experiencia prévia dos participantes de cada ferramenta

A Tabela 2 apresenta a distribuição dos participantes por ferramenta e nível de experiência prévia em automação de testes. Observa-se que a maior parte dos participantes (74%) possuí experiência prévia classificada como Nenhuma ou Básica, principalmente no grupo Playwright. A familiaridade anterior com a ferramenta de teste específica é muito baixa, tendo como resposta um nível muito baixo por 88% dos 43 participantes. Apenas 2 no grupo Cypress e 3 no grupo Selenium já haviam utilizado a ferramenta anteriormente.

5.2. Relação da Ferramenta Utilizada com os Resultados do Experimento

Esta subseção estabelece a relação entre a ferramenta utilizada e o desempenho dos participantes nas métricas de eficiência (Tempo de Implementação, Concisão) e Percepção das ferramentas (Facilidade de Uso, Esforço Percebido).

5.2.1. Eficiência de implementação

Para avaliar a eficiência da implementação, foi considerado o tempo de desenvolvimento dedicado a cada tarefa pelos participantes. No entanto, os dados de três deles foram desconsiderados devido a indícios de uso de inteligência artificial nos códigos. Essas suspeitas surgiram pela presença de caracteres não padronizados, como emojis, e pela inclusão de funcionalidades não solicitadas. Assim, a análise final foi realizada com base nos resultados de 10 participantes do grupo Cypress.

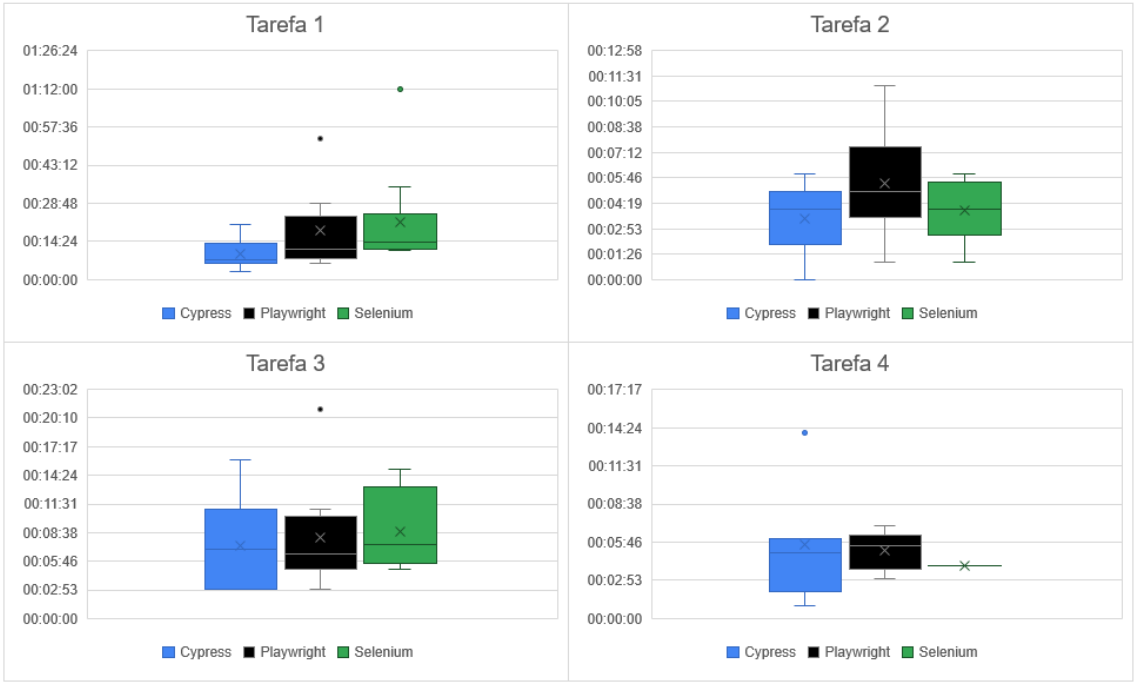

Figura 2. Tempo de Implementação de cada tarefa (min)

Observando Figura 2 é possível perceber que para a Tarefa 1 (“login com sucesso”) os participantes do grupo Cypress apresentaram, em média, o menor tempo (9,5 minutos), seguidos por Playwright (média de 18,5 minutos) e Selenium (média de 21,8 minutos). Para as tarefas posteriores, como a Tarefa 4 (“checkbox”), o tempo médio gasto diminuiu em todas as ferramentas, sendo o Selenium o que apresentou o menor tempo médio (4 minutos) em comparação com Playwright (média de 5,1 minutos) e Cypress (média de 5,5 minutos).

Como é possível ver na Figura 3, a contagem de linhas de código (CLOC) indica que o Selenium exigiu uma sintaxe mais prolixa em comparação com as demais ferramentas. Para a Tarefa 1, o CLOC médio do Selenium (aproximadamente 24 linhas),

com um participante em particular fazendo diversas asserções durante seu código da primeira tarefa, terminando com 64 linhas de código, foi significativamente maior do que o CLOC médio do Cypress (12 linhas) ou do Playwright (9 linhas) . Na Tarefa 3 (“adição e remoção de elementos”), o grupo Selenium apresentou, em média, 20 linhas, enquanto Cypress e Playwright ficaram entre 14 e 11 linhas, respectivamente. Essa diferença no CLOC corrobora a percepção de um participante do Selenium de que a sintaxe é “muito verbosa”e que é necessário “escrever muito para realizar uma ação”.

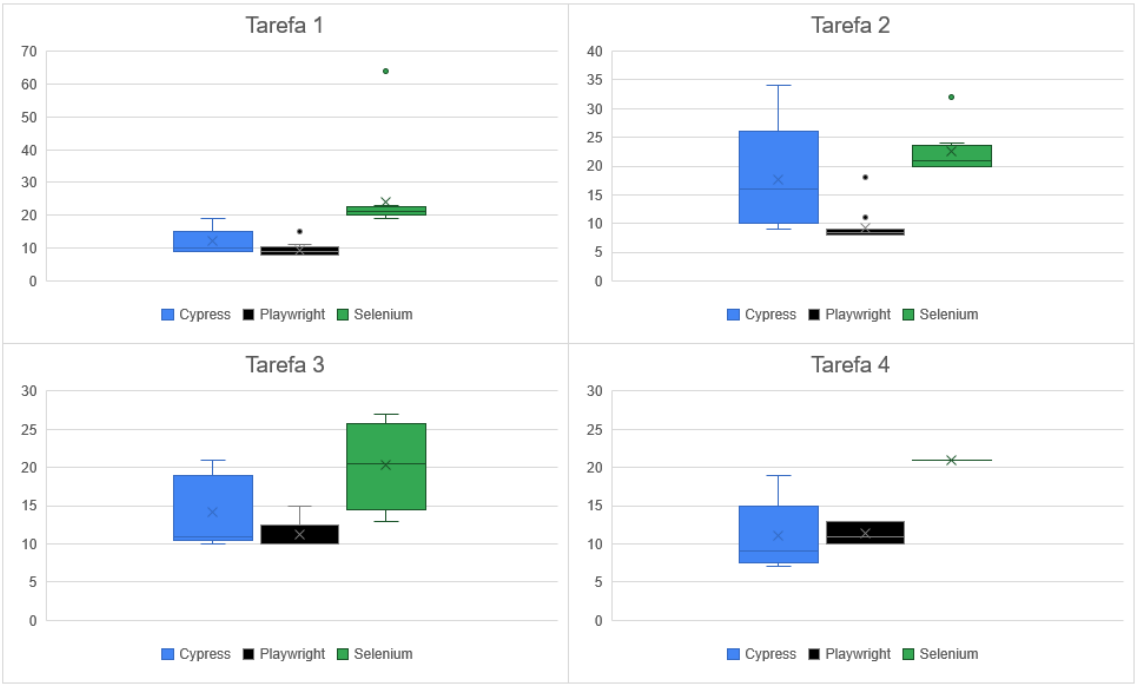

Figura 3. CLOC de cada tarefa

5.2.2. Desempenho técnico

A análise do desempenho técnico é fundamental porque o tempo de execução afeta diretamente a velocidade do ciclo de feedback e a eficiência de processos de integração contínua, especialmente em ambientes ágeis que dependem de execuções rápidas e frequentes. Considerando essa importância, a Figura 4 mostra um contraste claro entre as ferramentas: o Selenium apresentou os maiores tempos, variando entre 2,1 e 3,9 segundos; o Playwright obteve desempenho intermediário, entre 1,3 e 3 segundos; e o Cypress foi o mais eficiente, mantendo tempos consistentemente abaixo de 2 segundos, chegando a ser mais de 70% mais rápido que o Selenium em algumas tarefas.

Ao acabar o tempo disponível para que os alunos realizassem todos os cenários, alguns não conseguiram concluir todas as tarefas. A Figura 5 demonstra exatamente isso. A análise do grupo Selenium evidencia desafios na conclusão de tarefas mais complexas, como “adição e remoção de elementos” (Tarefa 3) e “checkbox” (Tarefa 4). Os participantes desse grupo frequentemente não conseguiram finalizar as últimas tarefas, citando falta de tempo ou dificuldades técnicas, como não conseguir contar ou selecionar os elementos. Um participante do Selenium, ao se autoavaliar na Tarefa 4, atribuiu nota 5 (máxima

dificuldade) e comentou: “Acabei agarrando muito na configuração do ambiente e tentando entender/finalizar a primeira tarefa. As demais ficaram de escanteio.”Também é possível perceber que, nas tarefas 3 e 4, no gráfico de número de pessoas que concluíram corretamente, os grupos Playwright e Cypress apresentaram os mesmos valores.

Figura 4. Tempo de execução de cada tarefa (seg)

A disparidade entre conclusão e conclusão correta indica que, mesmo quando os participantes entregavam uma solução, esta frequentemente não atendia plenamente aos requisitos funcionais, especialmente nas tarefas iniciais, nas quais ainda estavam se familiarizando com a ferramenta. Essa diferença foi mais acentuada no grupo Cypress, principalmente porque, conforme discutido na análise de eficiência de implementação na seção 5.2.1, três participantes tiveram seus códigos desconsiderados devido a suspeitas de uso de inteligência artificial, pelos motivos já explicados. Por isso, as tarefas desses três participantes não foram contabilizadas como concluídas corretamente

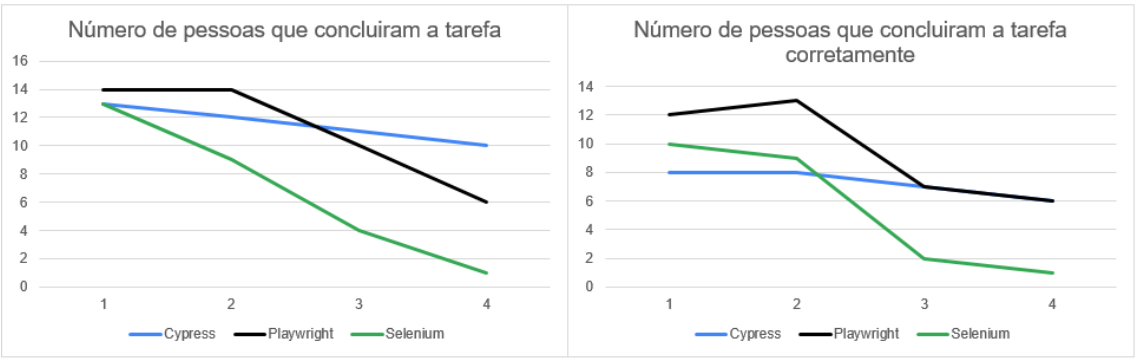

Figura 5. Quantidade de participantes que concluíram cada tarefa

5.2.3. Percepção dos usuários

A percepção dos participantes complementa os resultados quantitativos ao mostrar como cada ferramenta foi vivenciada na implementação. Elementos subjetivos, como dificuldade, clareza da sintaxe e curva de aprendizado, ajudam a explicar variações de desempenho e revelam a usabilidade prática em cenários reais. Esses fatores influenciam diretamente o esforço de adoção e manutenção das soluções de automação de testes, impactando produtividade e qualidade ao longo do tempo. Considerando que a nota 1 representa maior facilidade e a nota 5 representa maior dificuldade, a Figura 6 demonstra que o grupo do Playwright apresentou maior dificuldade em todas as tarefas. Já Selenium e Cypress, por sua vez, mostraram níveis de dificuldade similares, com as notas dos participantes se mantendo próximas nas primeiras atividades. Observa-se que a dificuldade percebida aumentou nas tarefas finais, acompanhando a maior complexidade dessas atividades.

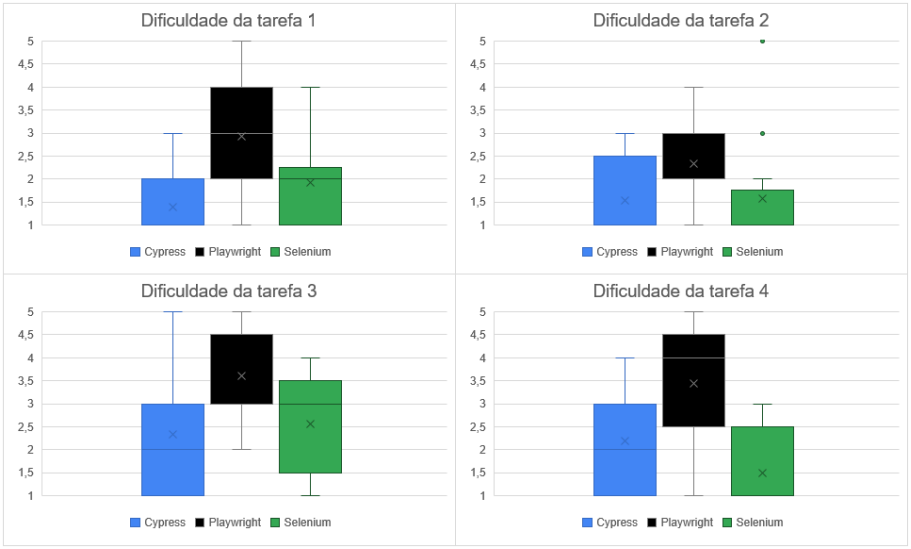

Figura 6. Dificuldade da tarefa por ferramenta

Selenium foi percebida como a ferramenta mais intuitiva e fácil de instalar (nota 1), como visto na Figura 7. Um participante do Selenium descreveu a instalação da ferramenta como “Tutorial muito didático”. No grupo Cypress e Playwright, a percepção foi mista. Enquanto alguns a consideraram fácil, outros classificaram a utilização com notas altas (4 ou 5). A facilidade de implementação dos testes do Playwright foi percebida como intermediária, o que é maior que as ferramentas Cypress e Selenium, nas quais a média dos participantes acharam fácil de implementar os testes com as ferramentas, apesar disso, houve alguns participantes que acharam complicado, uma dificuldade citada no grupo do Selenium foi: “Muito difícil, pois nunca trabalhei com testes antes”. Outro no grupo Cypress disse: “Os testes são relativamentes simples de serem implementados mas manipular multiplos elementos quando não há uma distinção de ID é um pouco difícil”.

A percepção do Selenium sobre a curva de aprendizado também foi similar ao Cypress, enquanto alguns a classificaram como tranquila, outros a comentaram “se eu tivesse mais tempo para aprender provavelmente teria evoluído mais rápido”, já a percepção do grupo Playwright foi em média a maior nota. Os participantes que utilizaram Cypress e Selenium classificou o esforço total percebido como baixo pra moderado (notas 2 e 3).

Um participante do Selenium resumiu a experiência como “Muito simples. Mas querendo ou não há um esforço mínimo para aprender uma nova tecnologia”. Já a nota de esforço do Playwright está um pouco mais alto que as outras ferramentas, com a média das notas sendo 3,3 de 5.

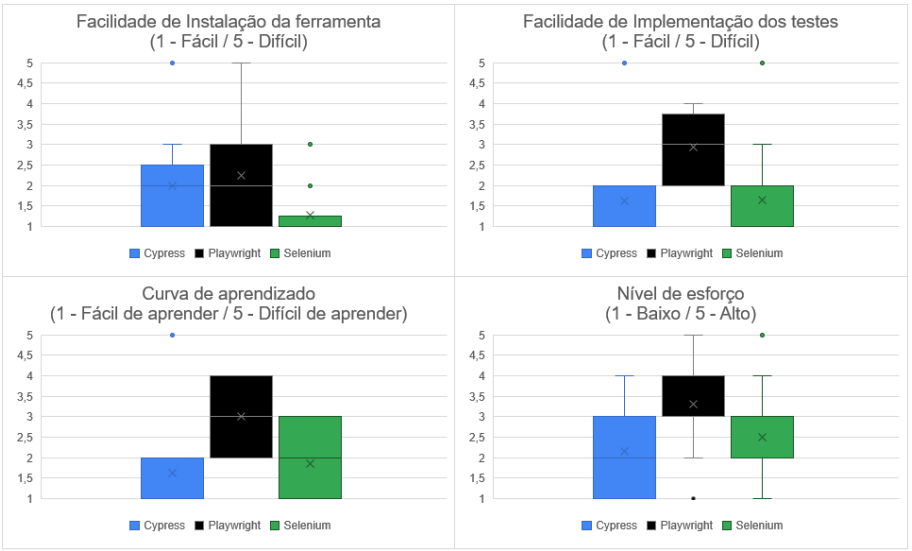

Figura 7. Facilidade de instalação, Uso, Curva de aprendizado e Esforço gasto

5.3. Sumarização e discussão dos resultados

Esta pesquisa atingiu seu objetivo geral de realizar um estudo comparativo entre Playwright, Selenium e Cypress por meio da análise de dados empíricos coletados da implementação de cenários de teste em uma aplicação web. Os resultados, alinhados aos objetivos específicos, permitiram uma avaliação da eficiência das ferramentas com base em critérios pré-definidos.

De forma consolidada, os dados revelam que, em termos de eficiência de implementação (tempo/esforço de desenvolvimento e linhas de código) (Figuras 2 e 3), o Cypress apresentou vantagens notáveis em relação ao Selenium e ao Playwright. A ferramenta permitiu uma implementação mais rápida que ambas e produziu código menos verboso em comparação ao Selenium, o que indica uma curva de aprendizado mais suave e maior produtividade inicial. Quanto à eficiência de execução, os resultados se mostraram estáveis, o Cypress obteve os melhores tempos, enquanto o Playwright e o Selenium apresentaram desempenho consistente, sendo o Selenium a ferramenta mais lenta nesse aspecto.

Na análise da experiência, Selenium foi percebido como mais simples de instalar e configurar, o Cypress se sobressaiu quanto à facilidade de uso e curva de aprendizado. Em contrapartida, o Playwright foi associado a maior dificuldade inicial devido à baixa familiaridade dos participantes, demonstrado na tabela 2. Selenium, apesar de intuitivo para alguns, foi percebido como mais trabalhoso e verboso, tendo um destaque negativo na conclusão das tarefas, com uma taxa de sucesso inferior para tarefas complexas. Esses achados indicam que, de modo geral, Cypress oferece o melhor equilíbrio entre desempenho técnico e experiência do usuário, enquanto Playwright surge como uma alternativa

promissora e o Selenium se mantém robusto, porém menos produtivo em contextos introdutórios de automação.

Como as tarefas avaliadas neste estudo apresentam níveis distintos de complexidade e características operacionais, optou-se por não realizar uma análise agregada única dos resultados. Ainda assim, ao considerar o conjunto das tarefas, a Figura 8 revela tendências gerais que reforçam as discussões apresentadas, especialmente no que se refere às diferenças de desempenho e eficiência entre as ferramentas. Observa-se que o Cypress apresentou tempo de implementação 30% menor que o Playwright e 32% menor que o Selenium, além de um tempo de execução 40% mais eficiente que o Playwright e 65% mais eficiente que o Selenium. Além disso, como discutido anteriormente, o Selenium mostrou-se significativamente mais verboso, com uma média de 60% mais linhas de código que o Cypress e 113% a mais que o Playwright.

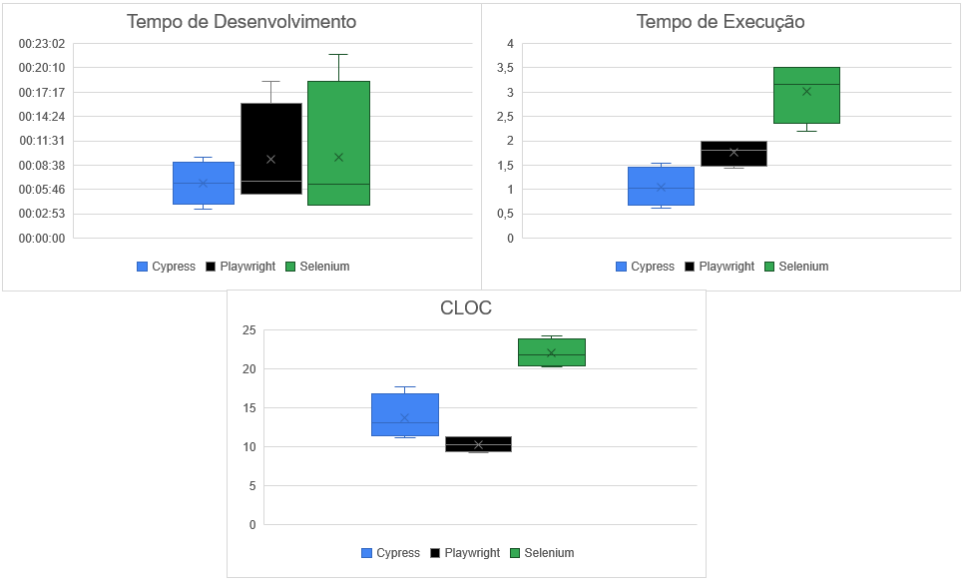

Figura 8. Tempo de Implementação, Execução e CLOC agregados

6. Ameaças à validade

Esta seção apresenta as principais ameaças à validade do estudo e as estratégias de mitigação adotadas, categorizadas como internas, externas, de construção e de conclusão [Wohlin et al. 2012].

As ameaças internas e externas à validade deste estudo incluem fatores relacionados tanto ao processo de coleta quanto ao perfil dos participantes. Uma possível limitação surge da chance de os participantes não responderem corretamente ao questionário, seja por interpretações incompletas das instruções ou por dificuldades no uso das ferramentas. Para reduzir esse risco, foi realizada uma apresentação padronizada com orientações detalhadas sobre instalação, utilização das ferramentas, funcionamento da aplicação web, exemplo de código e demonstração prática do preenchimento do questionário; além disso, dúvidas foram esclarecidas durante a execução do experimento e as respostas foram posteriormente avaliadas quanto à consistência antes de serem incluídas na análise. Outra ameaça relevante ocorre devido à seleção dos participantes por conveniência, já que todos

eram estudantes de Engenharia de Software da PUC Minas e cada turma foi vinculada a uma ferramenta específica, o que gerou diferenças de experiência entre os grupos e pode ter influenciado os resultados, limitando a generalização dos achados para outros contextos e perfis profissionais.

As ameaças relacionadas à construção e à conclusão deste estudo envolvem principalmente limitações na aplicação e na qualidade do instrumento de coleta de dados. Como o questionário foi aplicado em grupo, não foi possível garantir que todos os participantes seguiram fielmente as instruções apresentadas, embora tenha havido supervisão durante toda a execução para reduzir erros de preenchimento. Além disso, existe o risco de que o questionário não tenha capturado de maneira totalmente adequada as informações pretendidas ou que algumas perguntas possam não ter sido suficientemente claras, o que poderia resultar em respostas incompletas, interpretações divergentes ou registros incorretos de métricas como tempo e número de linhas de código. Para minimizar essas limitações, foi realizado um estudo piloto com sete participantes, permitindo avaliar a clareza das instruções, identificar ambiguidades e ajustar a estrutura do questionário antes de sua aplicação definitiva. Esse procedimento buscou assegurar que o instrumento refletisse corretamente os construtos analisados e fosse compreendido de forma consistente pelos participantes.

7. Conclusão

Este trabalho atingiu seu objetivo central de comparar as ferramentas Playwright, Selenium e Cypress no contexto da automação de testes em aplicações web. Para alcançá-lo, adotou-se uma metodologia baseada na implementação prática de cenários padronizados e na coleta estruturada de métricas quantitativas e percepções qualitativas. A execução do experimento por diferentes grupos de participantes, associada a um ambiente de execução controlado, permitiu avaliar de forma sistemática a eficiência das ferramentas em termos de tempo de desenvolvimento, concisão do código, desempenho de execução e experiência de uso.

Os resultados obtidos reforçam que não há uma ferramenta universalmente superior, mas sim soluções que se destacam em aspectos específicos conforme o contexto de aplicação. O Cypress apresentou melhor equilíbrio geral entre velocidade e facilidade de uso, enquanto o Playwright mostrou desempenho consistente e boa expressividade do código. Já o Selenium, apesar de robusto e amplamente adotado na indústria, demandou maior esforço de implementação e maior tempo de execução, especialmente para participantes com pouca familiaridade prévia. Além disso, fatores como curva de aprendizado, clareza da sintaxe, configuração inicial do ambiente e suporte nativo influenciaram diretamente o desempenho observado e a percepção dos usuários durante o experimento.

Este estudo também contribui para a área de Engenharia de Software ao apresentar evidências empíricas obtidas a partir da comparação prática e padronizada de ferramentas de automação de testes amplamente utilizadas. A análise de métricas de implementação, desempenho de execução, taxa de sucesso e percepção de uso fornece subsídios objetivos para compreender como a escolha da ferramenta impacta a produtividade e a experiência de usuários em contextos de adoção inicial. Como trabalhos futuros, recomenda-se ampliar o número de participantes, incluir perfis com diferentes níveis de experiência e avaliar essas ferramentas em outros ecossistemas de programação.

Pacote de Replicação

O pacote de replicação deste trabalho encontra-se disponível em:

`https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2025-1-tcci-0393100-pes-giovanni-bogliolo`

Referências

Aburas, A. (2024). Choosing the right automated software testing tools. In *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, pages 31–35.

Apriansah, M. R. and Desanti, R. I. (2024). Efficiency comparison of employee management automation using selenium and playwright: A case study of technology consulting company. In *2024 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 419–425.

Kavaler, D., Trockman, A., Vasilescu, B., and Filkov, V. (2019). Tool choice matters: Javascript quality assurance tools and usage outcomes in github projects. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 476–487.

Majeed, B., Toor, S. K., Majeed, K., and Chaudhary, M. N. A. (2021). Comparative study of open source automation testing tools: Selenium, katalon studio & test project. In *2021 International Conference on Innovative Computing (ICIC)*, pages 1–6.

Mosleh, M. A., Ameen Al-Khulaidi, N., Gumaei, A. H., Alsabry, A., and Musleh, A. A. A. (2024). Classification and evaluation framework of automated testing tools for agile software: Technical review. In *2024 4th International Conference on Emerging Smart Technologies and Applications (eSmarTA)*, pages 1–12.

Myers, G. J., Sandler, C., and Badgett, T. (2012). *The Art of Software Testing*. John Wiley & Sons, Inc., Hoboken, New Jersey, third edition.

Pelivani, E., Besimi, A., and Cico, B. (2022). A comparative study of ui testing framework. In *2022 11th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–5.

Pelivani, E. and Cico, B. (2021). A comparative study of automation testing tools for web applications. In *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, pages 1–6.

Santos, R., Santos, I., Magalhaes, C., and de Souza Santos, R. (2024). Are we testing or being tested? exploring the practical applications of large language models in software testing. In *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 353–360.

Sommerville, I. (2011). *Engenharia de Software*. Pearson Prentice Hall, São Paulo, 9ª ed. edition.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer, Berlin, Heidelberg, 2nd edition.

Apêndice

A. Estrutura do Questionário

1. Termo de Consentimento Livre e Esclarecido

- Após ler os Termos de Consentimento Livre e Esclarecido (TCLE) declaro que [Seleção única]
  - Concordo em participar da pesquisa

2. Perfil do Participante:

- Digite seu nome completo. [Resposta Curta]
- Qual curso você esta cursando? [Resposta Curta]
- Em qual período você esta atualmente? Esta regular? [Resposta Curta]
- Sexo: [Seleção única]
  - Feminino
  - Masculino
  - Prefiro Não Informar
- Dia de Nascimento [Data]
- Qual o seu nível de experiência prévia com automação de testes? [Seleção única]
  - 1 - Nenhuma
  - 2 - Básica
  - 3 - Intermediária
  - 4 - Avançada
- Você irá utilizar qual ferramenta para fazer os cenários? (Orientada no slide): [Seleção única]
  - Cypress
  - Selenium
  - Playwright
- Você já havia utilizado esta ferramenta anteriormente? [Seleção única]
  - Sim
  - Não
- Qual o nível de conhecimento com essa ferramenta? [Escala Likert 1-5]
  - 1 - Muito Baixo
  - 2 - Baixo
  - 3 - Intermediário
  - 4 - Elevado
  - 5 - Muito Elevado
- Você já possui experiência com outras ferramentas de testes? Se sim, quais? [Resposta Curta]

3 A. Execução da Tarefa 1:

Para cada cenário abaixo (4 cenários totais), registre o horário de inicio e horário de termino de cada tarefa e faça o upload do código produzido. Cenário 1 – Login com Sucesso (Link da página de Login) // (Link da página inicial)

Neste teste você deve acessar a página de login, inserir credenciais válidas e confirmar que o sistema exibe a mensagem ”You logged into a secure area!”, indicando que o login foi concluído com êxito.

- Horário de início (HH:MM) [Horário]
- Anexe o arquivo do código de login com sucesso desenvolvido (Colocar arquivo do código em uma pasta compactada [.zip]) [Arquivo]
- Horário de início (HH:MM) [Horário]
- Você conseguiu concluir essa tarefa? [Seleção única]
  - Sim
  - Não
- Qual o nível de dificuldade dessa tarefa [Escala Likert 1-5]
  - 1 - Muito fácil
  - 2 - Fácil
  - 3 - Intermediário
  - 4 - Difícil
  - 5 - Muito difícil
- Faça um comentário sobre a dificuldade desse cenário (Opcional) [Texto de Resposta]

3 B. Execução da Tarefa 2:

Para cada cenário abaixo (4 cenários totais), registre o horário de inicio e horário de termino de cada tarefa e faça o upload do código produzido. Cenário 2 – Login com Falha (Link da página de Login) // (Link da página inicial)

Neste teste você deve acessar a página de login, inserir as credenciais inválidas e validar se a mensagem ”Your username is invalid!”ou ”Your password is invalid!”é exibida, garantindo que o sistema bloqueia corretamente logins não autorizados.

- Horário de início (HH:MM) [Horário]
- Anexe o arquivo do código de login com sucesso desenvolvido (Colocar arquivo do código em uma pasta compactada [.zip]) [Arquivo]
- Horário de início (HH:MM) [Horário]
- Você conseguiu concluir essa tarefa? [Seleção única]
  - Sim
  - Não
- Qual o nível de dificuldade dessa tarefa [Escala Likert 1-5]
  - 1 - Muito fácil
  - 2 - Fácil
  - 3 - Intermediário
  - 4 - Difícil
  - 5 - Muito difícil
- Faça um comentário sobre a dificuldade desse cenário (Opcional) [Texto de Resposta]

3 C. Execução da Tarefa 3:

Para cada cenário abaixo (4 cenários totais), registre o horário de inicio e horário de termino de cada tarefa e faça o upload do código produzido. Cenário 3 – Adição e Remoção de Elementos (Link da página de adição e remoção de elementos) // (Link da página inicial)

Neste teste você deve acessar a página de Add/Remove Elements, adicionar dois botões Delete, verificar se ambos aparecem, remover um deles e validar se apenas 1 botão restante está disponível, confirmando a funcionalidade de adicionar e excluir elementos.

- Horário de início (HH:MM) [Horário]
- Anexe o arquivo do código de login com sucesso desenvolvido (Colocar arquivo do código em uma pasta compactada [.zip]) [Arquivo]
- Horário de início (HH:MM) [Horário]
- Você conseguiu concluir essa tarefa? [Seleção única]
  - Sim
  - Não
- Qual o nível de dificuldade dessa tarefa [Escala Likert 1-5]
  - 1 - Muito fácil
  - 2 - Fácil
  - 3 - Intermediário
  - 4 - Difícil
  - 5 - Muito difícil
- Faça um comentário sobre a dificuldade desse cenário (Opcional) [Texto de Resposta]

3 D. Execução da Tarefa 4:

Para cada cenário abaixo (4 cenários totais), registre o horário de inicio e horário de termino de cada tarefa e faça o upload do código produzido. Cenário 4 – Checkbox Interativo Link da página de checkboxes // (Link da página inicial)

Neste teste você deve acessar a página de Checkboxes, marcar o primeiro checkbox e validar se ele foi selecionado corretamente. Em seguida, desmarcar o segundo checkbox e verificar se ele foi desmarcado, confirmando que os checkboxes respondem às interações do usuário.

- Horário de início (HH:MM) [Horário]
- Anexe o arquivo do código de login com sucesso desenvolvido (Colocar arquivo do código em uma pasta compactada [.zip]) [Arquivo]
- Horário de início (HH:MM) [Horário]
- Você conseguiu concluir essa tarefa? [Seleção única]
  - Sim
  - Não
- Qual o nível de dificuldade dessa tarefa [Escala Likert 1-5]
  - 1 - Muito fácil
  - 2 - Fácil
  - 3 - Intermediário
  - 4 - Difícil
  - 5 - Muito difícil
- Faça um comentário sobre a dificuldade desse cenário (Opcional) [Texto de Resposta]

4. Percepção e Experiência:

- Considerando a configuração do ambiente quão fácil foi a instalação da ferramenta? [Escala Likert 1-5]
  - 1 - Muito fácil de instalar
  - 2 - Fácil de instalar
  - 3 - Intermediário
  - 4 - Difícil de instalar

  - 5 - Muito difícil de instalar
- Faça um comentário adicional sobre sua configuração do ambiente (Opcional) [Texto de Resposta]
- Considerando sua experiência na implementação e configuração dos testes, quão fácil foi utilizar esta ferramenta? [Escala Likert 1-5]
  - 1 - Muito fácil
  - 2 - Facíl
  - 3 - Intermediário
  - 4 - Difícil
  - 5 - Muito difícil
- Faça um comentário adicional sobre sua experiencia na implementação e configuração dos testes (Opcional) [Texto de Resposta]
- Qual foi sua percepção sobre a curva de aprendizado necessária para se tornar produtivo com esta ferramenta? [Escala Likert 1-5]
  - 1 - Muito fácil de Aprender
  - 2 - Facíl de Aprender
  - 3 - Intermediário
  - 4 - Difícil de Aprender
  - 5 - Muito difícil de Aprender
- Faça um comentário adicional sobre sua percepção sobre a curva de aprendizado. (Opcional) [Texto de Resposta]
- Qual foi o nível de esforço total percebido para desenvolver e configurar os cenários de teste com esta ferramenta? [Escala Likert 1-5]
  - 1 - Muito Baixo
  - 2 - Baixo
  - 3 - Intermediário
  - 4 - Alto
  - 5 - Muito Alto
- Faça um comentário adicional sobre seu nível de esforço para desenvolver e configurar os cenários (Opcional) [Texto de Resposta]
- Quais foram as principais dificuldades técnicas ou conceituais que você encontrou? [Texto de Resposta]
- Comente um pouco sobre os pontos fortes da ferramenta e qualquer sugestão ou observação adicional que tenha. [Texto de Resposta]
- Comente um pouco sobre os pontos fracos da ferramenta e qualquer sugestão ou observação adicional que tenha. [Texto de Resposta]