ANÁLISE DE FERRAMENTAS DE AUTOMAÇÃO DE TESTES BASEADAS EM INTELIGÊNCIA ARTIFICIAL

ANALYSING TEST AUTOMATION TOOLS BASED ON ARTIFICIAL INTELLIGENCE

Danilo de Araújo Leite Gomes[1]
Bruno Jean de Souza Lima[2]
Miguel Estevão Brasil Yanez Marques[3]
Luciano Gonçalves de Carvalho[4]

RESUMO: A integração da Inteligência Artificial (IA) nos processos de automação de testes em desenvolvimento *front-end* tem gerado avanços significativos em eficiência, precisão e cobertura de cenários. Este artigo propõe uma análise de ferramentas de automação de teste baseadas em IA explorando sua aplicabilidade em ambientes complexos e dinâmicos. Ao adaptar-se automaticamente a mudanças de código, essas ferramentas superam limitações dos métodos tradicionais, permitindo uma cobertura mais ampla e identificando problemas com maior agilidade. A revisão e a aplicação de soluções como Applitools, Mabl e Testim buscam demonstrar como a IA transforma o cenário de testes automatizados, oferecendo benefícios específicos para profissionais e organizações que visam qualidade e velocidade no desenvolvimento de *software front-end*.
Palavras-chave: Selenium; Applitools; Test.AI; *Front-End*; Testes automatizados.

ABSTRACT: The integration of Artificial Intelligence (AI) into test automation processes in front-end development has generated significant advances in efficiency, accuracy and scenario coverage. This article proposes an analysis of AI-based test automation tools, exploring their applicability in complex and dynamic environments. By automatically adapting to code changes, these tools overcome the limitations of traditional methods, enabling broader coverage and identifying problems more quickly. The review and the application of solutions such as Applitools, Mabl and Testim seeks to demonstrate how AI transforms the automated testing landscape, offering specific benefits to professionals and organizations that aim for quality and speed in front-end software development.
Keywords: Selenium; Applitools; Test.IA; Front-End; Automated Testing.

1 INTRODUÇÃO

De acordo com Gautam (2024), diante da crescente complexidade dos sistemas e da necessidade de entregas rápidas, os métodos tradicionais de teste manual tornam-se insuficientes. Os testes automatizados oferecem velocidade, precisão e escalabilidade, permitindo que as equipes de desenvolvimento identifiquem e corrijam *bugs* mais rapidamente, melhorem a qualidade do código e reduzam o

Análise e desenvolvimento de Sistemas - Fatec Mogi das Cruzes - danilo.gomes20@fatec.sp.gov.br [1]
Análise e desenvolvimento de Sistemas - Fatec Mogi das Cruzes - bruno.lima89@fatec.sp.gov.br [2]
Análise e desenvolvimento de Sistemas - Fatec Mogi das Cruzes - miguel.marques@fatec.sp.gov.br [3]
Professor - Fatec Mogi das Cruzes– SP - luciano.carvalho@fatec.sp.gov.br

tempo de entrega ao mercado. Além disso, os testes automatizados são essenciais para a integração contínua (CI) e a entrega contínua (CD), práticas fundamentais no desenvolvimento ágil de *software*. Com a evolução da IA, o processo de desenvolvimento de *software* introduziu ferramentas e técnicas que aumentam a produtividade, a precisão e a inovação (Finio; Downie, 2024). Ao simular uma ampla gama de cenários de interação com o usuário de maneira dinâmica, a automação de testes com IA supera as limitações inerentes aos métodos de teste tradicionais, tanto manuais quanto automatizados. Essa evolução dos testes é fundamental para a validação de sistemas cada vez mais complexos.

A introdução de ferramentas de automação de testes impulsionadas por IA oferece recursos avançados que se adaptam às demandas dos sistemas em evolução e a casos de uso mais complexos.

As ferramentas de automação de testes baseadas em IA emergiram como uma inovação crucial no setor de Tecnologia da Informação (TI). A popularidade e o potencial dessas soluções para aumentar o retorno sobre o investimento em desenvolvimento de *software* amplificam a necessidade de uma análise crítica sobre suas aplicações práticas (Li et al, 2020). Essas ferramentas atendem à demanda da indústria por maior agilidade e precisão, ambas essenciais em um mercado cada vez mais competitivo e dinâmico.

Este trabalho tem como objetivo conduzir uma análise de ferramentas de automação de testes com IA, no contexto do desenvolvimento de *front-end*, com ênfase nas plataformas Mabl, Applitools e Testim. A seleção dessas ferramentas foi fundamentada em sua usabilidade e acessibilidade, permitindo a execução de testes sem a exigência de conhecimentos avançados em programação. Esse diferencial viabiliza a participação de profissionais não técnicos, incluindo o usuário final, na validação de funcionalidades e na garantia de qualidade das aplicações (Hamill, 2004).

2 METODOLOGIA

O presente trabalho adota uma metodologia fundamentada em pesquisa bibliográfica, abrangendo livros, artigos científicos, monografias e a aplicação de testes prático para validar a eficácia de cada ferramenta. A seleção das produções acadêmicas foi realizada por meio de *strings* de busca elaboradas especificamente

para esse fim, aplicadas em bibliotecas digitais e repositórios reconhecidos, como *IEEE Xplore* e *ACM Digital Library*. Adicionalmente, foram utilizadas pesquisas na plataforma Google Acadêmico.

Os testes automatizados foram conduzidos em um ambiente computacional com as seguintes especificações técnicas:

- Placa mãe: AMD Ryzen A620M-E;
- Processador: Ryzen 8500G;
- Memória RAM: 16 GB DDR5;
- Armazenamento: SSD Kingston 480GB;
- Socket: AM5;
- Fonte de Alimentação: ATX 750W Plus bronze, Maximus.

Os componentes de *hardware* empregados nos testes são suficientes para garantir uma infraestrutura adequada, o que proporciona um processamento rápido e eficiente para ferramentas de teste baseadas em IA minimizando o risco de limitações de *hardware* afetarem o desempenho dos testes. A configuração proposta permite a execução de testes tanto em servidores locais quanto em ambientes de nuvem, conforme necessário.

Para garantir a uniformidade dos testes realizados, utilizamos em todas as ferramentas avaliadas (Applitools, Testim.AI e Mabl), o mesmo ambiente de *sandbox* disponibilizado pela Applitools, que simula um aplicativo bancário e possibilita a validação automatizada de interfaces gráficas, permitindo a verificação do layout e do comportamento da aplicação em condições controladas, sem a necessidade de um ambiente de produção real. Dessa forma, foi possível replicar os cenários de teste de maneira consistente entre as diferentes soluções analisadas. Os testes efetuados se concentraram no fluxo de *login* para avaliar a consistência da interface e a eficácia das capacidades de teste visual da ferramenta. Isto permitiu uma análise focada na capacidade da ferramenta de detectar discrepâncias visuais e assegurar a confiabilidade da interface.

3 REFERENCIAL TEÓRICO

A crescente complexidade dos sistemas de *software* modernos e a pressão para ciclos de lançamento mais rápidos tornaram a automatização dos testes essencial, especialmente no desenvolvimento *front-end*. Os testes automatizados

proporcionam uma abordagem mais eficiente e fiável em comparação com os testes manuais, permitindo uma execução mais rápida e uma maior cobertura dos testes, o que é fundamental para garantir a qualidade do *software* em ambientes ágeis.

Com o passar dos anos, a IA vem reformulando a maneira como o *software* é testado, reduzindo o esforço manual e automatizando atividades como a geração de casos de teste e a análise de resultados, melhorando, assim, a eficiência dos testes (Tao; Gao; Wang, 2019). Com o desenvolvimento de técnicas de Inteligência Artificial (IA), a automação de testes evoluiu para abranger sistemas capazes de simular interações humanas e adaptar-se a mudanças, o que resolve as limitações dos métodos tradicionais, tais como:

- Tempo prolongado na criação e manutenção de testes, especialmente quando há frequentes mudanças no *software*;
- Baixa flexibilidade, dificultando a adaptação a interfaces dinâmicas e *layouts* que se modificam constantemente;
- Alta suscetibilidade a falsos positivos e falsos negativos, tornando o diagnóstico de falhas mais complexo;
- Dependência de *scripts* rígidos, que exigem intervenção manual para ajustes sempre que há pequenas alterações no ambiente de execução;
- Cobertura limitada, com dificuldade para testar cenários complexos ou imprevistos que ocorrem no uso real.

Essas limitações são amplamente discutidas na literatura especializada. Segundo Garousi et al. (2024), ferramentas tradicionais de automação de testes apresentam dificuldades em se adaptar a mudanças frequentes no *software*, resultando em manutenção intensiva dos *scripts* de teste. Além disso, a falta de adaptabilidade e a rigidez dos *scripts* são apontadas como fatores que contribuem para a geração de falsos positivos e negativos, comprometendo a eficácia dos testes.

De acordo com Borges e Lima (2024), a IA permite a geração automática de casos de teste com base no comportamento do sistema em tempo real, aumentando a cobertura e a relevância dos testes. Além disso, a IA contribui para a detecção proativa de defeitos em estágios iniciais do desenvolvimento, otimizando o processo de testes e reduzindo o tempo necessário para a identificação de falhas.

Ferramentas de automação de testes baseadas em IA diferenciam-se das abordagens convencionais por utilizarem algoritmos capazes de aprender e adaptar-

se com base nos dados coletados durante a execução dos testes. Segundo Felderer e Ramler (2021), a IA aumenta a precisão e a adaptabilidade dos testes automatizados, pois permite que os sistemas se ajustem automaticamente a mudanças de código, acelerando o processo de identificação de falhas. Essa adaptabilidade é especialmente promissora para o desenvolvimento de *software* de *front-end*, onde as alterações de código são frequentes e exigem ferramentas flexíveis e eficientes.

Com a utilização da IA na automação de testes é possível aumentar a cobertura dos cenários de teste, já que as ferramentas tradicionais, como o Selenium[1] (ferramenta de automação de testes para aplicações *web*, que permite simular interações de usuários em navegadores), dependem de *scripts* rígidos, o que limita a adaptação a novas condições e exige constante manutenção de código (Tosun; Bener; Kale, 2010).

Além da flexibilidade e adaptabilidade, a automação de testes com IA contribui para a qualidade do *software*, identificando possíveis falhas de maneira mais rápida e precisa. Ferramentas como Mabl, Testim e Applitools são amplamente recomendadas por sua capacidade de impulsionar a produtividade, automatizar tarefas complexas e aumentar a estabilidade dos testes, reduzindo o esforço de manutenção e o tempo de desenvolvimento (Lewczuk, 2024). Esse tipo de tecnologia é especialmente relevante em aplicações *front-end*, onde a experiência visual do usuário é fundamental e qualquer inconsistência pode comprometer a usabilidade do *software*.

Por fim, a IA na automação de testes tem se mostrado um investimento que proporciona retorno positivo ao longo do ciclo de vida do *software*. Conforme destacado por Li et al (2020), a adoção de IA na automação de testes representa um aumento na eficiência e no retorno sobre o investimento, pois a agilidade e a precisão dos testes reduzem os custos associados à manutenção e à correção de erros em estágios avançados de desenvolvimento.

[1] https://www.selenium.dev/pt-br/documentation/

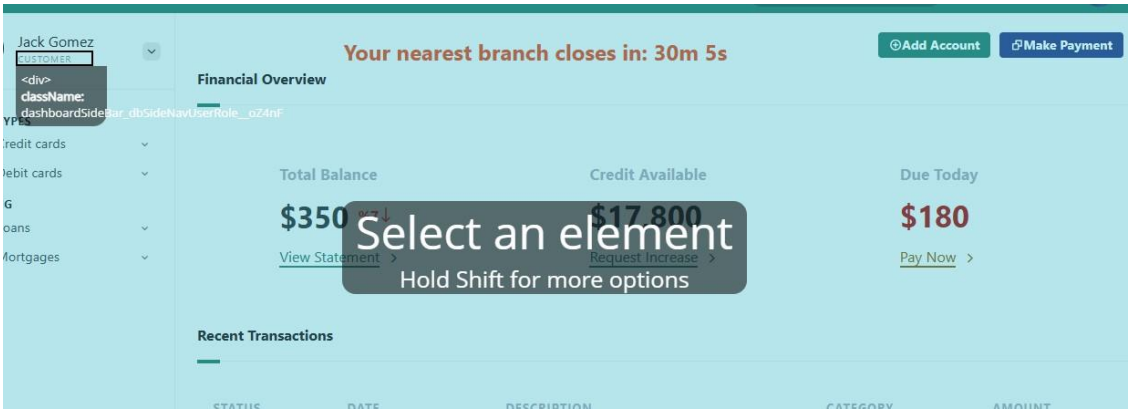
4 RESULTADOS E DISCUSSÃO

Os resultados obtidos serão discutidos em termos de funcionalidades, usabilidade, performance e integração, com destaque para pontos fortes e fracos, visando aprimorar a qualidade de aplicações *front-end*.

A Mabl aplica aprendizado de máquina para aumentar a eficiência e a eficácia dos testes, permitindo que as equipes de desenvolvimento testem aplicações *web* de maneira contínua e com pouca necessidade de codificação intensiva. Ela oferece um conjunto robusto de funcionalidades, incluindo testes funcionais, testes de regressão e testes de carga, além de recursos de análise visual, detecção de anomalias e alertas automáticos para falhas. Suas integrações com ferramentas de desenvolvimento como Jenkins, Jira e Slack, facilitam a comunicação e o acompanhamento dos testes nos fluxos de trabalho das equipes. Em termos de usabilidade, a Mabl se destaca pela facilidade de uso e *feedback* visual. Sua interface permite configurar o fluxo dos testes de forma individual ou geral e, ao selecionar os itens a serem testados, é possível visualizar as seleções em um painel à direita, onde ajustes podem ser feitos antes da execução. A Figura 1 mostra a usabilidade da Mabl.

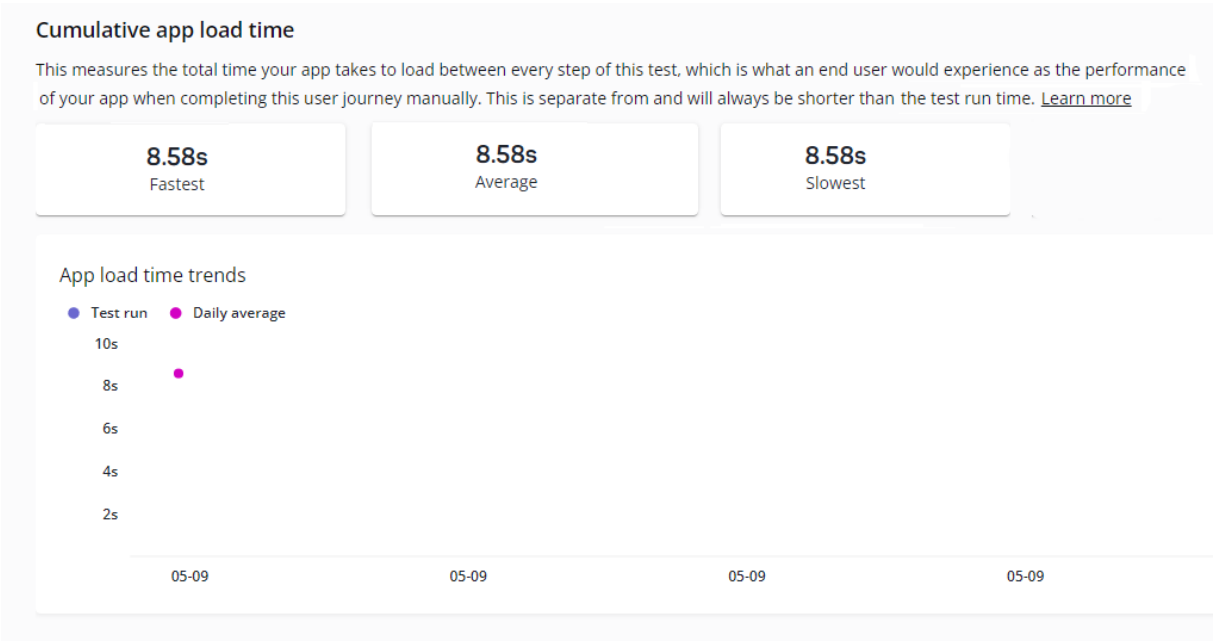
A performance da Mabl é um dos seus pontos fortes, com execução rápida e otimizada de recursos, o que melhora a eficiência e qualidade dos projetos. A ferramenta possibilita a análise de métricas de performance, identificando gargalos e proporcionando *insights* sobre a estabilidade do sistema, como ilustrado na Figura 2. Isso permite que as equipes otimizem cada componente de forma contínua.

Figura 1 – Visualização da usabilidade do Mabl



Fonte: Mabl, 2024

Figura 2 – Performance geral da Mabl na realização dos testes



Fonte: Mabl, 2024

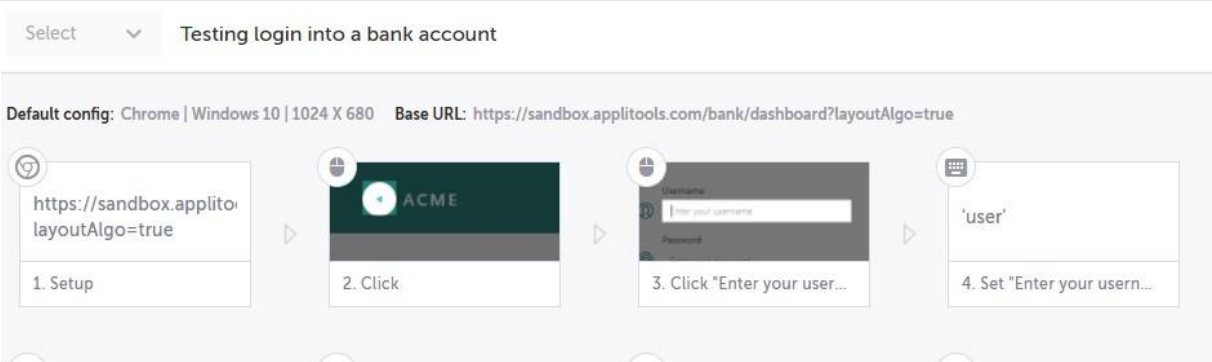
O Testim AI, que foi projetada para simplificar e otimizar o processo de criação, manutenção e execução de testes de *software*, utiliza aprendizado de máquina para identificar dinamicamente elementos da interface e se adaptar a mudanças, reduzindo quebras de teste devido a alterações no código ou na interface do usuário. Em termos de funcionalidades, ela oferece a criação automática de testes com base no comportamento da interface do usuário, eliminando a necessidade de *scripts* complexos, além de possuir também análise inteligente, detectando mudanças na interface e adaptando os testes conforme o *software* evolui. A plataforma permite validar a consistência visual e identificar falhas não detectadas por testes tradicionais.

Ao considerar a usabilidade, o Testim AI foi projetado para ser intuitivo e de fácil navegação, mesmo para usuários com pouca experiência em testes automatizados. Sua interface gráfica permite que os usuários criem, executem e gerenciem testes de forma simples, utilizando recursos como gravação de interações e edição visual dos testes, como demonstrado na Figura 3.

De acordo com Kacheru (2024), o Testim utiliza localizadores inteligentes e testes auto adaptáveis, permitindo que os testes se ajustem automaticamente a alterações na interface do usuário, minimizando a necessidade de manutenção manual. Além disso, o Testim tem a capacidade de integrar-se facilmente com ferramentas populares de CI/CD, como Jenkins, GitLab e CircleCI, tornando-o

acessível tanto para equipes pequenas quanto grandes. Com relação à performance, o Testim AI se destaca pela execução rápida e precisa de testes em larga escala. A IA adapta os testes a mudanças na interface, minimizando o tempo necessário para identificar falhas. Com execução paralela de testes, é possível cobrir múltiplos cenários simultaneamente, o que melhora a cobertura e qualidade do *software*.

Figura 3 – Steps visuais do fluxo de testes



Fonte: Testim, 2024

A ferramenta Applitools é especialmente útil para validar componentes gráficos e de *design* em diferentes resoluções e dispositivos, garantindo uma experiência visual consistente para os usuários, independentemente do dispositivo utilizado. Essa abordagem faz do Applitools uma solução confiável para garantir a integridade de UI/UX ao longo do desenvolvimento de *software*.

Considerando o aspecto de funcionalidade, o Applitools permite comparações visuais de capturas de tela, suporte para testes em múltiplos dispositivos e navegadores, e o uso de *baseline* para comparação entre versões. Suas integrações com diversas linguagens e *frameworks* de testes populares, aliadas ao teste em ambiente paralelo, tornam a ferramenta completa para testes de interface. Sua interface intuitiva facilita a criação e revisão de testes visuais. Seu sistema de detecção de alterações reduz a necessidade de ajustes manuais, permitindo que usuários revisem rapidamente as diferenças visuais para decidir se representam erros ou mudanças esperadas. A ferramenta permite ignorar dados variáveis para evitar falsos positivos, conforme mostrado na Figura 4.

Figura 4 – Partes selecionadas para destarte nos testes

Fonte: Applitools, 2024

Em termos de performance, o Applitools, com algoritmos avançados de IA, é possível identificar mudanças visuais de maneira precisa e eficiente, mesmo em grandes volumes de dados. Os relatórios detalhados ajudam a identificar melhorias na interface, garantindo uma experiência de alta qualidade para o usuário final.

Os testes automatizados tradicionais geralmente utilizam *scripts* codificados manualmente para verificar funcionalidades específicas, o que pode ser eficaz em garantir que as funcionalidades básicas permaneçam consistentes. Contudo, esses testes exigem manutenção intensiva à medida que a interface e o código evoluem. Alterações visuais e de *design* muitas vezes passam despercebidas ou geram falsos positivos, dificultando a detecção de falhas reais sem comprometer a precisão dos testes.

Por outro lado, os testes com IA, realizados por meio de ferramentas como o Mabl, Testim AI e Applitools, utilizam aprendizado de máquina para se adaptar a mudanças de interface do usuário e *design*, reduzindo o tempo e esforço de manutenção. Eles oferecem detecção de anomalias, comparação visual avançada e suporte a múltiplos dispositivos, com execução paralela de testes que proporciona agilidade sem perder a precisão. A IA permite detectar alterações em tempo real, minimizando a ocorrência de falsos positivos e permitindo que equipes de desenvolvimento mantenham a qualidade visual e funcional de suas aplicações com maior eficiência e menor intervenção manual.

5 CONSIDERAÇÕES FINAIS

A aplicação de Inteligência Artificial (IA) em ferramentas de automação de testes para o desenvolvimento *front-end* é altamente benéfica para aumentar a precisão, eficiência e abrangência dos testes, especialmente em ambientes ágeis. A análise das ferramentas evidenciou como a IA supera limitações de abordagens

tradicionais, oferecendo flexibilidade e adaptabilidade valiosas para detecção de falhas e manutenção de sistemas em constante evolução.

Os resultados obtidos destacam a importância de selecionar ferramentas de automação com IA que atendam aos requisitos específicos de cada projeto, considerando o tipo de teste necessário e o contexto de desenvolvimento. A IA aplicada à automação de testes se mostra, assim, uma estratégia eficaz para equipes que buscam aprimorar a qualidade e a performance de suas aplicações.

Dessa forma, este artigo contribui para o entendimento do papel da IA na automação de testes, abrindo caminhos para novas investigações sobre ferramentas de automação baseadas em IA aplicadas a *softwares* reais em ambientes produtivos.

Essa abordagem pode fornecer resultados mais robustos e consistentes, permitindo avaliar o comportamento das soluções de IA diante de desafios práticos, como mudanças inesperadas na interface, fluxo de navegação não linear e dados dinâmicos. Além disso, para trabalhos futuros, a realização de testes com sistemas reais pode contribuir para mensurar com maior precisão o impacto da IA na redução de retrabalho, no tempo de execução dos testes e na efetividade da detecção de falhas, fortalecendo ainda mais a aplicabilidade dessas ferramentas no cenário industrial.

REFERÊNCIAS BIBLIOGRÁFICAS

BORGES, A.; P. LIMA, R. S. **Inteligência Artificial aplicada à qualidade de software: uma revisão sistemática.** *Revista de Tecnologias e Inovação*, v. 12, n. 1, p. 55–68, 2024. Disponível em: https://revistaft.com.br/inteligencia-artificial-aplicada-a-qualidade-de-software. Acesso em: 5 maio 2025.

FELDERER, M.; RAMLER, R. **Quality assurance for AI-based systems: overview and challenges.** 2021. Disponível em: https://arxiv.org/abs/2102.05351. Acesso em: 07 nov. 2024.

FINIO, Matthew; DOWNIE, Amanda. **IA no desenvolvimento de software.** IBM, [s.d.]. Disponível em: https://www.ibm.com/br-pt/think/topics/ai-in-software-development. Acesso em: 22 mar. 2025.

GAROUSI, V.; FELDERER, M.; MALEK, D. **A systematic literature review and survey of industrial practices in software test automation.** *Journal of Systems and Software*, 2024. Disponível em: https://arxiv.org/abs/2409.00411. Acesso em: 5 maio 2025.

GAUTAM, Abhishek. **The importance of automated testing in modern software development.** Alea IT Solutions, 2024. Disponível em: https://www.aleaitsolutions.com/the-importance-of-automated-testing-in-modern-software-development/. Acesso em: 7 maio 2025.

HAMILL, Paul. **Unit test frameworks: tools for high-quality software development.** Sebastopol: O'Reilly Media, 2004.

KACHERU, Goutham. **AI-powered test automation frameworks: choosing the right tools.** 2024. Disponível em: https://www.researchgate.net/publication/387883704_AI-POWERED_TEST_AUTOMATION_FRAMEWORKS_CHOOSING_THE_RIGHT_TOOLS. Acesso em: 7 maio 2025.

LEWCZUK, Krzysztof. **Revolutionising software testing with AI tools.** 2024. Disponível em: https://www.digiterre.com/2024/10/30/revolutionising-software-testing-with-ai-tools/. Acesso em: 07 nov. 2024.

LI, J. J. *et al.* **Advances in test automation for software with special focus on artificial intelligence and machine learning.** *Software Quality Journal*, 2020. Disponível em: https://link.springer.com/article/10.1007/s11219-019-09472-3. Acesso em 28 set 2024.

TAO, C.; GAO, J.; WANG, T. **Testing and quality validation for AI software: perspectives, issues, and practices.** [S.l.], [s.d.]. Disponível em: https://ieeexplore.ieee.org/document/8811507/. Acesso em: 27 set. 2024.

TOSUN, A.; BENER, A.; KALE, R. **AI-based software defect predictors: applications and benefits in a case study.** *Proceedings of the AAAI Conference on Artificial Intelligence*, 2010. Disponível em: https://cdn.aaai.org/ojs/18807/18807-13-22524-1-10-20210930.pdf. Acesso em: 15 nov. 2024.