INSTITUTO FEDERAL GOIANO - CAMPUS CERES
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LETÍCIA SANTOS CAMARGO

A UTILIZAÇÃO DE TESTES AUTOMATIZADOS NO PROCESSO DE GARANTIA DE QUALIDADE DE INTERFACES DE APLICAÇÕES WEB: UM MAPEAMENTO SISTEMÁTICO

CERES - GO
2024

LETÍCIA SANTOS CAMARGO

A UTILIZAÇÃO DE TESTES AUTOMATIZADOS NO PROCESSO DE GARANTIA DE QUALIDADE DE INTERFACES DE APLICAÇÕES WEB: UM MAPEAMENTO SISTEMÁTICO

Trabalho de curso apresentado ao curso de Bacharelado em Sistemas de Informação do Campus Ceres do Instituto Federal Goiano, como requisito parcial para a obtenção do título de Bacharela em Sistemas de Informação, sob orientação do Prof. Me. Adriano Honorato Braga.

CERES - GO
2024

Sistema desenvolvido pelo ICMC/USP
Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas - Instituto Federal Goiano

C581u

Camargo, Letícia Santos
A UTILIZAÇÃO DE TESTES AUTOMATIZADOS NO PROCESSO DE GARANTIA DE QUALIDADE DE INTERFACES DE APLICAÇÕES WEB: UM MAPEAMENTO SISTEMÁTICO / Letícia Santos Camargo; orientadora Adriano Honorato Braga. -- Ceres, 2024.
49 p.

TCC (Graduação em Bacharelado em Sistemas de Informação) -- Instituto Federal Goiano, Campus Ceres, 2024.

1. qualidade de software. 2. interface. 3. GUI. 4. sistemas web. 5. automatização. I. Honorato Braga, Adriano , orient. II. Título.

Responsável: Johnathan Pereira Alves Diniz - Bibliotecário-Documentalista CRB-1 n°2376

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

INSTITUTO FEDERAL
Goiano

Repositório Institucional do IF Goiano - RIIF IF Goiano Sistema Integrado de Bibliotecas

---

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR PRODUÇÕES TÉCNICO-CIENTÍFICAS NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Com base no disposto na Lei Federal nº 9.610/98, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano, a disponibilizar gratuitamente o documento no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

Identificação da Produção Técnico-Científica

[ ] Tese	[ ] Artigo Científico
[ ] Dissertação	[ ] Capítulo de Livro
[ ] Monografia – Especialização	[ ] Livro
[ ] Artigo - Especialização	[ ] Trabalho Apresentado em Evento
[X] TCC - Graduação	[ ] Produção Técnica

Nome Completo do Autor: Leticia Santos Camargo
Matrícula: 2020103202030062
Título do Trabalho: A utilização de testes automatizados no processo de garantia de qualidade de interfaces de aplicações web: um mapeamento sistemático

Restrições de Acesso ao Documento

Documento confidencial: [X] Não [ ] Sim, justifique: _____
Informe a data que poderá ser disponibilizado no RIIF Goiano: 16/08/2024
O documento está sujeito a registro de patente? [ ] Sim [X] Não
O documento pode vir a ser publicado como livro? [ ] Sim [X] Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O(a) referido(a) autor(a) declara que:

1. o documento é seu trabalho original, detém os direitos autorais da produção técnico-científica e não infringe os direitos de qualquer outra pessoa ou entidade;

2. obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

3. cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Ceres, 16 de agosto de 2024.

Leticia Santos Camargo
(Assinado Eletronicamente pelo o Autor e/ou Detentor dos Direitos Autorais)

Ciente e de acordo:

Adriano Honorato Braga
(Assinado Eletronicamente pelo orientador)

Documento assinado eletronicamente por:

- Adriano Honorato Braga, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 17/08/2024 10:46:58.
- Letícia Santos Camargo, 2020103202030062 - Discente, em 17/08/2024 10:46:28.

Este documento foi emitido pelo SUAP em 17/08/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 624043
Código de Autenticação: 6420997f88

INSTITUTO FEDERAL GOIANO
Campus Ceres
Rodovia GO-154, Km 03, SN, Zona Rural, CERES / GO, CEP 76300-000
(62) 3307-7100

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

ATA DE DEFESA DE TRABALHO DE CURSO

Aos 08 dias do mês de agosto do ano de dois mil e vinte e quatro, realizou-se a defesa de Trabalho de Curso da acadêmica Leticia Santos Camargo, do Curso de Bacharelado em Sistemas de Informação, matrícula 2020103202030062, cujo título é "A UTILIZAÇÃO DE TESTES AUTOMATIZADOS NO PROCESSO DE GARANTIA DE QUALIDADE DE INTERFACES DE APLICAÇÕES WEB: UM MAPEAMENTO SISTEMÁTICO". A defesa iniciou-se às 19 horas e 37 minutos, finalizando-se às 20 horas e 45 minutos. A banca examinadora considerou o trabalho APROVADO com média 9,5 no trabalho escrito, média 10,0 no trabalho oral, apresentando assim média aritmética final de 9,8 pontos, estando a estudante APTA para fins de conclusão do Trabalho de Curso.

Após atender às considerações da banca e respeitando o prazo disposto em calendário acadêmico, a estudante deverá fazer a submissão da versão corrigida em formato digital (.pdf) no Repositório Institucional do IF Goiano – RIIF, acompanhado do Termo Ciência e Autorização Eletrônico (TCAE), devidamente assinado pela autora e orientador.

Os integrantes da banca examinadora assinam a presente.

(Assinado Eletronicamente)
Orientador
Adriano Honorato Braga

(Assinado Eletronicamente)
Membro da banca
Rafael Divino Ferreira Feitosa

(Assinado Eletronicamente)
Membro da banca
Thalia Santos de Santana

Documento assinado eletronicamente por:

- Rafael Divino Ferreira Feitosa, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 08/08/2024 21:11:39.
- Thalia Santos de Santana, Thalia Santos de Santana - Membro externo - Instituto Federal Goiano - Campus Ceres (10651417000410), em 08/08/2024 21:09:50.
- Adriano Honorato Braga, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 08/08/2024 21:08:26.

Este documento foi emitido pelo SUAP em 08/08/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 621383
Código de Autenticação: 0d0b169680

À minha família,
ao projeto Meninas Digitais no Cerrado,
e a todos que me apoiaram nesta jornada.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a Deus, minha fonte de força e inspiração. Sua presença em minha vida me ajudou a superar as dificuldades e encontrar o caminho certo para alcançar meus objetivos. A Ele, que jamais me desamparou e nunca me deixou sozinha nesta jornada.

Ao IF, que me proporcionou tantos momentos inesquecíveis e me apresentou a pessoas extraordinárias, das quais me inspiro e me orgulho. Foi graças a esta instituição e às pessoas que dela fazem parte que me tornei quem sou hoje. Não há palavras suficientes para expressar minha gratidão por tudo que vivi e realizei neste lugar tão especial.

À minha amada família, que me apoiou e incentivou diariamente na minha busca por conhecimento. Um agradecimento especial ao meu herói e meu tudo, meu pai Divino, que sempre cuidou de mim, compartilhou minhas dores, nunca duvidou da minha força e me encorajou desde o início a buscar a educação. Ao meu maior exemplo de força e determinação, minha querida irmã Raquel, que sempre fez o possível e o impossível por mim, abrindo caminhos para a realização dos meus sonhos e sendo meu suporte e força, tornando a conquista desta graduação uma realidade.

Aos meus eternos orientadores, Adriano e Ramayane, que me acolheram com tanto carinho e me ensinaram tudo o que sei. Obrigada por transformarem o meu futuro e me permitirem voar tão alto; vocês são e sempre serão minha inspiração. Acredito que nunca conseguirei expressar toda a minha gratidão a vocês. Obrigada, obrigada e obrigada.

À minha amiga Thalia, que é luz, amor e esperança, e que me ensina todos os dias um novo jeito de fazer as coisas. Você é uma fonte constante de inspiração e compaixão. Sua fé em mim me impulsionou a ir além, e você é uma das pessoas mais extraordinárias que conheci. Obrigada por compartilhar tantos saberes e por sempre mostrar e dar o seu melhor, me inspirando a mostrar o meu melhor também.

Não há agradecimentos suficientes ao projeto Meninas Digitais no Cerrado. Sinto-me orgulhosa e honrada por ser fruto e parte dessa iniciativa. Acredito no poder das conexões, e me conectar com tantas mulheres inspiradoras com certeza me trouxe até aqui, graças às oportunidades proporcionadas por esse projeto. Um obrigada a todos os envolvidos, aos coordenadores e as minhas colegas de projeto.

Agradeço a todos os meus professores que trilharam este caminho comigo, aos meus colegas que tornaram essa jornada mais leve e agradável, e às minhas amigas que compartilharam comigo essa escrita e me ajudaram a concluir este trabalho: Laureana e Juliana, vocês são incríveis. Obrigada por tudo. Um agradecimento especial à Maria Isabela, que me ajudou de tantas formas a construir este trabalho, compartilhando noites em claro e preocupações constantes. Obrigada por estar ao meu lado e me apoiar todos os dias.

Enfim, a todos que fizeram parte desta jornada e construíram comigo esse caminho,

sou imensamente grata por todo o apoio, ensinamentos e memórias. Jamais esquecerei tudo o que vivi aqui e levo comigo a certeza de que essa experiência me moldou para sempre. Que esta jornada seja apenas o começo de muitas outras aventuras.

"A qualidade não é um ato, é um hábito. É a soma dos pequenos detalhes que, quando negligenciados, possuem um grande impacto."

- Lisa Crispin

RESUMO

O teste de *software* desempenha um papel fundamental na garantia da qualidade durante o desenvolvimento de um *software*, pois busca identificar falhas por meio da execução do programa. No entanto, os testes manuais tem se mostrado inviáveis em muitos projetos devido ao seu alto custo financeiro e consumo de tempo. Para contornar essa questão, foram desenvolvidas diversas técnicas de automatização de testes, que têm demonstrado resultados significativos na redução dos custos. Aplicações *web* se tornaram uma ferramenta fundamental na sociedade, presentes no dia-a-dia de milhões de pessoas e a dependência do uso desses sistemas tornou indispensável a existência de um processo de garantia de qualidade de *software*. Sistemas *web* utilizam interfaces gráficas como meio de interação humano-computador, em que todos os dados de entrada e saída são geridos por elementos gráficos, tornando essa interação facilitada e otimizada. A garantia de qualidade das interfaces gráficas são consideradas extremamente importantes, já que suas falhas podem impactar toda a funcionalidade do sistema. É possível encontrar diversas ferramentas que automatizam o processo de testagem de interfaces gráficas de sistemas *web*, dessa forma, essa pesquisa realizou um mapeamento sistemático sobre a utilização dos testes automatizados no processo de garantia de qualidade em interfaces *web*. A partir da *string* de busca, foram encontrados um total de 505 trabalhos. Após a aplicação de filtros de seleção, de acordo com os critérios de inclusão e exclusão, foram considerados e analisados 37 estudos relevantes, que permitiram revelar características importantes sobre o uso de tecnologias na automatização de testes de interfaces *web* nos últimos 5 anos. Foi identificado que cerca de 64,8% dos estudos analisados propuseram uma nova ferramenta e/ou técnica de testagem automatizada de testes em interfaces *web*. Dos *frameworks* existentes, o Selenium obteve 62,5% das citações dentro dos estudos comparativos entre ferramentas, sendo assim a ferramenta mais utilizada. Foram reportadas múltiplas limitações, sendo principalmente relatadas o custo orçamentário elevado da implementação dos *frameworks* e ferramentas já existentes no mercado, e nos estudos que propuseram uma nova ferramenta, foram relatadas dificuldades na generalização dos testes para diferentes tipos de plataformas *web*. Portanto, este estudo possibilitou a caracterização da utilização de testes automatizados em interfaces web, identificando ferramentas e técnicas propostas e desenvolvidas nos últimos 5 anos. Evidenciando as múltiplas dificuldades que a automatização de testes GUI ainda enfrenta, impulsionando assim a necessidade de novas pesquisas que abordem soluções eficientes a essa atividade.

Palavras-chave: qualidade de *software*. interface. GUI. sistemas *web*. automatização. testagem.

ABSTRACT

Software testing plays a crucial role in ensuring quality during the development of software, as it aims to identify defects through program execution. However, manual testing has proven to be unfeasible for many projects due to its high financial cost and time consumption. To address this issue, various automated testing techniques have been developed, which have shown significant results in reducing costs. Web applications have become an essential tool in society, present in the daily lives of millions of people, and the dependence on these systems has made the existence of a software quality assurance process indispensable. Web systems use graphical interfaces as a means of human-computer interaction, where all input and output data are managed by graphical elements, facilitating and optimizing this interaction. Ensuring the quality of graphical interfaces is considered extremely important, as their failures can impact the entire functionality of the system. Numerous tools that automate the graphical interface testing process for web systems are available. Thus, this research conducted a systematic mapping of the use of automated tests in the quality assurance process for web interfaces. Using the search string, a total of 505 papers were found. After applying selection filters according to inclusion and exclusion criteria, 37 relevant studies were considered and analyzed, revealing important characteristics about the use of technologies in the automation of web interface testing over the past 5 years. It was identified that approximately 64.8% of the analyzed studies proposed a new tool and/or technique for automated web interface testing. Among existing frameworks, Selenium received 62.5% of the citations within comparative studies of tools, making it the most used tool. Multiple limitations were reported, primarily the high budgetary cost of implementing existing frameworks and tools in the market, and in studies proposing new tools, difficulties in generalizing tests for different types of web platforms were reported. Therefore, this study allowed for the characterization of the use of automated testing in web interfaces, identifying tools and techniques proposed and developed in the past 5 years, highlighting the multiple challenges that GUI test automation still faces, thus driving the need for further research to address efficient solutions for this activity.

Keywords: software quality. interface. GUI. web systems. automation. testing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CI	Critério de Inclusão
CE	Critério de Exclusão
E2E	*End-to-end*
GUI	*Graphical User Interface*
IA	Inteligência Artificial
ML	*Machine Learning*
MSL	Mapeamento Sistemático da Literatura
QA	*Quality Assurance*
QP	Questão de Pesquisa

SUMÁRIO

1 INTRODUÇÃO

Em um cenário comum de desenvolvimento de um *software*, o processo se baseia em três fases principais: o estudo do problema, o planejamento da solução e a sua implementação. Após isso, é realizado então uma sequência de testes, em que geralmente a pessoa responsável pelo desenvolvimento da aplicação verificará manualmente, por meio de testes, se a funcionalidade desenvolvida atende aos requisitos solicitados.

Em situações em que são encontrados erros ou falhas, a equipe de desenvolvimento deve corrigi-los e assim retomar a cadeia de testes novamente. Formando assim um ciclo de controle de qualidade em que o sistema somente será liberado quando não se encontrar mais falhas. É comum, em empresas, que antes do *software* ser liberado comercialmente, ele seja submetido a uma bateria de testes pela equipe especializada em qualidade do *software* (Bernardo; Kon, 2008).

Diante do cenário atual, onde a evolução das tecnologias, ferramentas e até mesmo das linguagens de programação tem sido notavelmente rápida, o processo de desenvolvimento de software tornou-se extremamente ágil. Essa aceleração na entrega dos sistemas trouxe diversas vantagens, como a capacidade de lançar produtos mais rapidamente e adaptar-se rapidamente às demandas do mercado.

No entanto, essa mesma rapidez pode impactar diretamente a garantia de qualidade do software. A agilidade no desenvolvimento pode fazer com que a qualidade deixe de ser a principal prioridade, resultando em um comprometimento da confiabilidade do produto final. Enquanto a velocidade de desenvolvimento proporciona vantagens competitivas, ela pode desafiar os métodos tradicionais de garantia de qualidade, exigindo a adoção de novas abordagens e ferramentas para manter altos padrões de qualidade no software(Rollwagen et al., 2020).

O teste de *software* é o processo de execução de um *software* com a intenção de encontrar falhas (Myers; Sandler; Badgett, 2011). Portanto esse processo se torna uma das técnicas mais importantes durante o processo de desenvolvimento de um *software* quando se trata sobre garantia de qualidade. De acordo com Biswal, Nanda e Mohapatra (2010) a qualidade do produto final dependerá diretamente da escala dos testes, uma vez que, quanto maior for a abrangência dos cenários de testes, maior então a cobertura de verificação de qualidade do *software*.

A testagem adequada de um *software* é fundamental para garantir a sua qualidade, e por isso deve ser executado com a maior cobertura possível (Pressman, 2016). De acordo com Bernardo e Kon (2008), alcançar uma boa qualidade em *software* não é uma tarefa simples devido à sua complexidade e aos problemas que podem surgir durante o processo de desenvolvimento. E por isso é totalmente compreensível que nem sempre seja possível que os desenvolvedores ou a equipe de testes consiga fielmente testar todos os viéses das

alterações, já que essa execução manual dos testes pode se tornar extremamente cansativa e repetitiva quando se envolve um grande número casos de testes.

Tais características dos testes manuais acabam tornando-o inviável dentro de diversos projetos de *software*, devido ao seu alto consumo de tempo e custo financeiro que geram atrasos e prejuízos às empresas. Frente a isso, diversas técnicas de automatização de testes de *softwares* vêm sendo desenvolvidas e demonstram resultados significativos quanto a redução de custos e de tempo. (Biswal; Nanda; Mohapatra, 2010).

Dentro de metodologias ágeis o conceito de qualidade do produto é uma prioridade em todos os níveis do projeto e é exercida por todos os envolvidos, pois acredita-se que a prevenção dos erros é mais simples e barata do que a sua correção. Em algumas metodologias específicas recomenda-se o uso de testes automatizados como ferramentas auxiliares para alcançar a qualidade do *software* (Bernardo; Kon, 2008).

Entre os diversos tipos de aplicações existentes, as aplicações *web* se tornaram uma ferramenta fundamental na sociedade atual, presentes no dia-a-dia de milhões de pessoas, usadas por veículos de imprensa, instituições públicas, empresas e milhares de outras vertendes (Tonella; Ricca; Marchetto, 2014). Portanto, com o volume e a dependência do uso desses sistemas, tornou-se indispensável o processo de garantia de qualidade durante o desenvolvimentos desse tipo de *software*. Devido ao impacto na experiência do usuário, a busca por ferramentas e técnicas que possam facilitar o processo de garantia de qualidade está em constante evolução (Eaton; Memon, 2009).

Em aplicações do tipo *web*, o método de interação humano-computador mais utilizado é a GUI, do inglês *Graphical User Interface*. As interfaces gráficas de usuários, são uma parte muito importante de uma aplicação, já que são elas que recebem os eventos de cliques, seleções, digitação de textos e várias outras interações do usuário com o sistema. São responsáveis por manipular as entradas e garantir a apresentação dos resultados na tela. Para os usuários as interfaces gráficas possibilitam uma liberdade de uso, pois podem optar por diferentes tipos de interação, além de facilitar a usabilidade do sistema. Já para os desenvolvedores, o desenvolvimento de interfaces gráficas robustas se torna um grande desafio, já que requerem a criação de artes gráficas, controle de sincronicidade das ações e das múltiplas possibilidades de comandos (Banerjee et al., 2013).

A garantia de qualidade de interfaces gráficas é consideradas extremamente importantes, já que suas falhas podem impactar toda a funcionalidade do sistema. Em consequência da liberdade de uso dada pelas GUI, as inúmeras possibilidades de entrada de dados tornam o processo de testagem extremamente difícil. Uma maneira de se testar uma interface é executar cada funcionalidade separadamente e analisar os resultados diante do esperado, porém essa atividade pode ser extremamente cansativa e custosa quando executada em sistemas robustos (Banerjee et al., 2013).

Com o objetivo de sanar as dificuldades encontradas no processo de garantia da qualidade de sistemas com GUI, diversas ferramentas foram desenvolvidas para auxiliar no

processo de testagem destes sistemas. Podendo ser encontradas no mercado diversas ferramentas e técnicas desenvolvidas para automatização de testes, principalmente para testes de interfaces gráficas (Bernardo; Kon, 2008).

Mesmo diante disto, não foram encontrados pesquisas que apresentassem o estado da arte atual dos estudos relacionados, de forma sistemática, e portanto este estudo busca realizar um mapeamento sistemático da literatura, com o objetivo de apresentar informações quanto ao estado atual de pesquisas relacionadas a automatização de testes em interfaces gráficas do usuário em aplicações *web*.

Sendo assim, a estrutura deste estudo segue apresentando na seção 2, o referencial teórico, a saber: contextualização e conceitos, em seguida na seção 3 a descrição do material e métodos, utilizados diante o protocolo proposto. Na seção 4 são apontados os principais resultados decorrente do mapeamento, tais como: a identificação e listagem das ferramentas utilizadas e/ou desenvolvidas pelos estudos encontrados; indicação dos tipos de testes executados em combinação com os testes de interface e a especificação das limitações reportadas. Por fim, são descritas, na seção 5 deste trabalho, as considerações finais e trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 CONTEXTUALIZAÇÃO

Erros e falhas são um risco constante no desenvolvimento de software. Uma vez que falhas no *software* geram um aumento significativo nos custos do projeto, o planejamento e execução de testes se tornam cruciais na garantia de qualidade deste *software* (Vasconcelos et al., 2006).

A garantia de qualidade (QA, do inglês *Quality Assurance*) possui uma definição ampla, a depender de cada empresa desenvolvedora. Entende-se como uma definição de processos e padrões a se seguir com o objetivo de garantir a qualidade do produto (Sommerville, 2015). No caso de *softwares*, o processo de garantia da qualidade é dada por diversos processos, e um dos mais usuais é o processo de teste de *software*.

Os testes de software são uma oportunidade de encontrar defeitos, erros e/ou falhas antes que o sistema seja entregue aos usuários. De acordo com a terminologia padrão para Engenharia de Software do IEEE – *Institute of Electrical and Electronics Engineers*, o defeito é um ato acometido por um indivíduo, em que é realizado um comando ou instrução incorreta. O erro, portanto, é a manifestação do defeito, como por exemplo um resultado inesperado na execução do software. Já a falha é o comportamento incosistente do sistema dado por um processamento incorreto e em algumas vezes, por um conjunto de erros (IEEE, 1990).

A testagem pode ser feita tanto de forma manual, quanto automatizada. Um ponto importante de se lembrar é que a atividade de testar um *software* não irá garantir por si só a acuracidade do sistema, apenas irá detectar erros e/ou falhas existentes (Vasconcelos et al., 2006). Sendo assim, o processo de testagem permite que sejam encontrados defeitos durante o desenvolvimento de um sistema, podendo assim corrigi-lo antes de chegar ao usuário final.

Com a evolução constante das tecnologias, as aplicações *mobile* e *web* se tornaram presentes diariamente na vida de milhões de pessoas, e é inegável seu valor dentro da sociedade atual, seja conectando pessoas ao redor do mundo ou seja disseminando informações. Ao conquistar um papel tão fundamental, se é questionado então a integridade e a garantia de qualidade desses tipos de sistemas (Kousar et al., 2023a).

Sistemas *web* utilizam o modelo de interface do utilizador que permite a interação dos usuários com os dispositivos digitais por meio de elementos gráficos, conhecido como GUI. Os componentes gráficos presentes na interface são os responsáveis por recolher os dados de entrada e apresentar os dados de saída. Portanto toda a integridade do sistema dependerá diretamente da funcionalidade destes componentes. Assim sendo, a testagem de interfaces de sistemas *web*, se tornou essencial devido a sua alta complexidade e relevância (Fulcini;

Ardito, 2022a).

A testagem de forma automatizada vêm crescendo ao longos dos últimos anos e ganhando espaço nesse processo de garantia de qualidade de *software*. Os desenvolvimentos de *scripts* de testes se tornaram extremamente vantajosos por serem de fácil reprodução, diminuindo assim o esforço humano por meio dos testes manuais e agilizando o processo de testagem do *software* (Bernardo; Kon, 2008).

Uma estrutura de automação de testes, chamados de *Frameworks*[1], é uma plataforma que acopla um conjunto de padrões e estratégias necessárias para execução e controle dos *scripts* de testes (Pinheiro, 2023). Podem ser encontradas diversas ferramentas e técnicas no mercado para o desenvovimento dos *scripts* de automatização dos testes, em que podem ser aplicadas em *softwares desktops*, aplicativos *mobile* e *webs*.

Banerjee et al. (2013) conduziram um mapeamento sistemático sobre a testagem de interfaces gráficas de usuários, onde reuniu 230 artigos da área, publicados entre os anos de 1991 a 2011, trazendo os aspectos principais sobre a testagem GUI durante o período. O estudo conclui que haveria um crescimento notável na área de desenvolvimento de scripts de teste em larga escala, e sugere novas pesquisas e produções sobre a implementação desse tipo de teste em plataformas *web* e *mobile*.

O estudo publicado por Banerjee et al. registrou que os primeiros trabalhos sobre a testagem de interfaces gráficas foram publicados em 1991, pelos os autores S. Yip e D. Robsonm. Trabalhos nos quais foram discutidos sobre os testes funcionais de GUI e novas estratégias de validações (Banerjee et al., 2013).

Kousar et al. (2023b), realizou um estudo sistemático com 24 estudos, publicados entre 2013 e 2023 com o objetivo de investigar o atual estado da arte das técnicas e ferramentas utilizadas em uma nova abordagem de padrão de teste de interfaces gráficas em aplicativos *android* e *web*, chamada *"Pattern-Based GUI Testing* (PBGT)".

> "*Pattern-Based GUI Testing* (PBGT) é uma nova abordagem de teste baseada em modelo que visa melhorar a reutilização e, ao mesmo tempo, reduzir a carga de trabalho associado à modelagem de interfaces de usuário e testes de aplicativos."(Kousar et al., 2023b)

A pesquisa realizada não retorna dados específicos sobre a automatização de teste, porém conclui que, de acordo com os estudos levantados, o novo modelo proposto *Pattern-based* de testagem ajudará no aumento da cobertura de teste e na automatização deles (Kousar et al., 2023a).

Com base nas pesquisas iniciais, não foram encontrados na literatura nenhum estudo sistemático, entre janeiro de 2019 e dezembro de 2023, que relacionasse diretamente o uso

[1] É uma ferramenta que reúne um conjunto de bibliotecas, que abordam funcionalidades, e estruturas, para o desenvolvimento de aplicações, a fim de fornecer soluções para um mesmo domínio de problema, permitindo a reutilização do seu código.

de automatização de teste em interfaces de sistemas *web*. Diante a essa lacuna, o presente estudo objetivou responder três principais questões de pesquisa sobre o atual cenário da automatização de teste em interfaces *web*. Realizando portanto um levantamento sobre as ferramentas e técnicas de teste utilizadas durante a automatização de teste de interfaces *web*, nos últimos cinco anos.

Para abordar essas questões de maneira eficaz e proporcionar uma compreensão completa dos métodos empregados, é essencial primeiro estabelecer uma base sólida sobre os conceitos fundamentais de testes de software. Portanto, serão discorridas nos próximos tópicos alguns dos conceitos fundamentais da testagem de software, como os níveis de testes, os tipos de testes e as metodologias empregadas nessa atividade.

2.1.1 Testes de *Software*

O processo de teste contém três principais etapas: geração do caso de teste, execução e avaliação. Em comparação com as duas outras etapas, a geração do caso de teste se torna a mais desafiadora e difícil (Mingsong; Xiaokang; Xuandong, 2006). Portanto é dito como base do teste de *software* a determinação do conjuntos de casos de testes para o *software* a ser validado (Cartaxo et al., 2006).

Um caso de teste é composto por uma entrada, que contém uma condição inicial para que aquela execução do teste aconteça, e os passos a serem executados. Além disso, o caso irá contar com uma saída, que representa os resultados esperados daquela determinada execução, e uma pós-condição de estado final do sistema (Jorgensen, 1995).

2.1.2 Níveis de Teste

Os níveis de testes são metodologias realizadas na verificação dos requisitos definidos de um *software*. De acordo com Pressman (2016), existem 4 principais níveis de teste:

Testes de Unidade

Os testes de unidade, conhecidos popularmente como testes unitários, são responsáveis por testarem apenas pequenas partes do sistema, tendo como alvo pequenos trechos de códigos como funções e objetos. São realizados com o objetivo de provocar falhas, sendo por erros de lógica ou falhas na implementação.

Esse tipo de teste é realizado principalmente por desenvolvedores no momento da codificação, e possuem um papel importante, já que são capazes de encontrar erros e falhas na fase inicial do *software*, facilitando a resolução do problema e diminuindo custos de reparos futuros.

Testes de Integração

Os testes de integração são realizados logo após os testes unitários, pois visam garantir que, após a integração de diferentes fases desenvolvidas em um projeto, o sistema ao todo funcione de acordo com os requisitos planejados inicialmente.

Esse tipo de teste complementa o teste unitário, garantindo assim a funcionalidade completa do *software*, desde pequenas unidades até a interação entre os módulos. Portanto, assim pode-se garantir que ao ser implementadas novas funcionalidades ou que correções sejam inseridas no sistema, o comportamento do sistema não seja afetado.

Testes de Sistema

Após a integração do sistema, em que o *software* e o *hardware* estão completamente integrados, há a ocorrência do teste de sistema. Nesse tipo de teste é verificado se os requisitos projetados para o sistemas estão em plena funcionalidade em ambiente de produção. O teste de sistema é, portanto, uma série de testes diferentes com o propósito de colocar o sistema inteiro à prova.

Testes de Aceitação

Por fim, os testes de aceitação são os realizados com um grupo seleto de clientes, a fim de verificar o comportamento do sistema e aprovar seu funcionamento como o requerido (Rocha, 2001). As validações desse tipo de teste podem ser realizadas de outras maneiras, mas usualmente, o sucesso da validação é resultante do funcionamento do *software* com base no esperado pelo usuário final.

2.1.3 Técnicas de Teste

Dentro dos níveis de testes, podem ser utilizadas diversas técnicas de teste, a depender do objetivo que se busca. Dois principais tipos de testes utilizados são os testes estruturais e testes funcionais.

Testes Estruturais (Caixa-Branca)

Os testes estruturais, conhecidos como testes de caixa-branca, são realizados diretamente sobre o código fonte dos componentes do sistema, avaliando aspectos como fluxo de dados e caminhos lógicos. Esse tipo de teste é utilizado comumente em testes unitários e de integração em que são praticados geralmente no processo de desenvolvimento do código fonte (Pressman, 2016).

Testes Funcionais (Caixa-Preta)

Já os testes funcionais, conhecidos como testes de caixa-preta, não levam em conta aspectos internos do sistema. Nessa técnica, uma entrada é fornecida, o teste é executado e o resultado obtido é comparado a um resultado pré-determinado. Testes funcionais podem ser aplicados em todos os níveis de teste e inclusive podem ser desenvolvidas critérios de teste, abrangendo ainda mais os requisitos do sistema (Pressman, 2016).

Diversas técnicas foram desenvolvidas e estão sendo utilizadas na garantia de qualidade de um *software*, tais como: teste de desempenho, teste de usabilidade, teste de carga, teste de *stress*, teste de confiabilidade, testes de ponta-a-ponta e teste interface gráfica do usuário (GUI). Todos os tipos de testes podem ser combinados entre si, para criar camadas de verificação, e assim expandir a garantia de qualidade do sistema a ser testado (Neto; Claudio, 2007).

Dentre essas diversas técnicas, a testagem de um sistema atrás da interface gráfica têm se tornado indispensáveis, principalmente em aplicações *web*, em que se tem composições de elementos gráficos cada vez mais complexos e robustos, os quais são transmitidas os dados de entrada e saída (Fulcini; Ardito, 2022b).

2.1.4 Testes Manuais

Nos testes manuais o sistema é executado por um testador que fica responsável por inserir os dados ao *software* e analisar as informações de retorno, em caso de falhas o testador reporta ao desenvolvedor (Sommerville, 2015).

A interação exclusivamente humana, traz o benefício de que a pessoa testadora vivi-encie o ambiente de produção e possa definir padrões de teste em tempo real. Por outro lado, os testes manuais são extremamentes lentos e morosos, e estão sujeitos a falhas humanas (Silva, 2021).

2.1.5 Testes Automatizados

A utilização da automação nos testes de *software* é uma abordagem que visa acelerar o processo de testagem e assim desenvolver ferramentas mais eficientes, capazes de assegurar a qualidade de forma abrangente (Izabel, 2014). O desenvolvimento de *scripts* de testes agilizam o processo de encontro de erros e falhas no sistema e por serem automatizados dispensam a necessidade de uma pessoa para executar a suíte de testes (Silva, 2021).

De acordo com o autor Chicanelli et al. (2019), a implementação de *scripts* de testes automatizados no processo de desenvolvimento de *softwares*, apresentaram melhora significativa na segurança e qualidade. Aumentando a abrangência dos casos de testes, reduzindo o esforço humano e facilitando a reexecução dos testes.

Apesar das vantagens oferecidas pelos testes automatizados, como o aumento da produtividade, reutilização de testes, redução de erros e aumento da confiabilidade, é importante destacar que podem apresentar algumas desvantagens. Estão envolvidas nas

desvantagens, os custos na implantação das ferramentas, adaptação de cenários complexos e específicos, geração de dados de teste e os desafios na codificação dos *scripts* de teste. Embora a automação da execução de casos de teste tenha representado um avanço significativo no campo, as atividades de teste de *software* tendem a se tornar mais difíceis e custosas à medida que os sistemas se tornam cada vez mais complexos (Durelli et al., 2019).

3 MATERIAL E MÉTODOS

Para alcançar o objetivo desta pesquisa, o método utilizado foi a realização de um mapeamento sistemático da literatura (MSL) que segundo Kitchenham e Charters (2007), se denomina como um estudo secundário que tem como objetivo identificar e classificar o conteúdo relacionado com um tópico de pesquisa, sendo assim uma investigação ampla envolvendo estudos primários relacionados com o tópico de pesquisa específico, visando identificar as evidências disponíveis sobre esse tópico.

Utilizando-se das diretrizes de mapeamento sistemático, previstos por Kitchenham e Charters, no qual se envolve as fases de planejamento, condução do estudo e publicação dos resultados. Foi então desenvolvido um protocolo de mapeamento. Onde definiu-se as questões de pesquisa (QPs), estratégia de busca e critérios de seleção dos estudos presentes na literatura.

A fase de planejamento e condução do estudo foram realizadas pelo *software Parsifal*[1], que é uma ferramenta on-line, projetada para auxiliar os pesquisadores na condução de revisões sistemáticas da literatura e que segue as diretrizes citados anteriormente, apresentadas no estudo de Kitchenham e Charters (2007).

Devido a inexistência de trabalhos que identifiquem dados sobre as ferramentas existentes e utilizadas pelos estudos atuais, além de identificar quais tipos de testes são comumente realizados nesse tipo de sistemas, definiu-se então as seguintes questões de pesquisa para nortear este mapeamento sistemático.

3.1 QUESTÕES DE PESQUISA

A partir da definição do objetivo principal deste estudo, foram formuladas as principais questões de pesquisa a serem respondidas de acordo com o protocolo estabelecido. Com o foco principal em encontrar evidências e resultados relevantes que respondessem ao objetivo geral. Portanto, foram delimitadas as seguintes QPs:

QP01 - Quais ferramentas têm sido utilizadas e/ou desenvolvidas para automatização de testes em GUI de sistemas *web*?

QP02 - Quais técnicas de teste estão sendo combinadas na automatização de testes em GUI de sistemas *web*?

QP03 - Quais limitações têm sido reportadas na automatização de testes em GUI de sistemas *web*?

[1] https://parsif.al/

3.2 STRING DE BUSCA E BASES DE DADOS

Em seguida, com o objetivo de responder as QP, foram analisadas e selecionadas palavras-chaves que relacionassem os testes automatizados, interfaces gráficas (GUI) e sistemas *web*. Em decorrência dos poucos resultados encontrados, foram listados, da mesma forma, os sinônimos recorrentes encontrados por meio dos testes-piloto. Após os ajustes nos testes-piloto, a *string* de busca final foi determinada, apresentando variações de acordo com o mecanismo de busca considerado. Utilizando-se de uma busca automática, foram considerados as seguintes bases bibliográficas da Ciência da Computação: *IEEE Xplore*[2], *SBC Open Lib*[3], *SciELO*[4] e *Scopus*[5].

A *string* de busca foi aplicada tanto com palavras-chave em Português como em Inglês e foi submetida no dia 30 de maio de 2024, resultanto num total de 505 trabalhos retornados. Após aplicar a *string* de buscas nas bases selecionadas, foi executado um processo de filtragem inicial, com o intuito de selecionar apenas os estudos relevantes. Foram retidos, portanto, já durante a aplicação da *string*, apenas os artigos publicados nos últimos 5 anos completos, ou seja, trabalhos publicados de 2019 a 2023, excluindo os trabalhos que fossem do tipo: livros, capítulos de livros, *reviews*, estudos secundários e terciários, assim como os que não tivessem sido escritos na linguagem Português ou Inglês.

Em resultado a essa primeira triagem, foram selecionados um total 206 trabalhos, 123 retornados da base IEEEXplore, 1 estudo na SBC Open Lib, e 82 na Scopus, não houveram retornos da base de dados Scielo, como demonstrado na Tabela 1.

Tabela 1 – Busca e quantitativo de resultados.

Base	**String de Busca**	**Quantidade**
IEEE Digital Library	*(("All Metadata":GUI OR "All Metadata":"graphical user interface"OR "All Metadata":"front-end"OR "All Metadata": frontend) AND ("All Metadata":testing OR "All Metadata":test OR "All Metadata":teste) AND ("All Metadata":automation OR "All Metadata":automated OR "All Metadata":"automação"OR "All Metadata":"automatizado") AND ("All Metadata":web))*	123
SBC Open Lib	*((gui OR "graphical user interface"OR "front-end"OR frontend) AND (testing OR test OR teste) AND (automation OR automated OR automatizado OR automação) AND (web))*	1
Scielo	*((gui) OR (graphical user interface) OR (front-end) OR (frontend)) AND ((testing) OR (test) OR (teste)) AND ((automation) OR (automated) OR (automação) OR (automatizado)) AND ((web))*	0
Scopus	*TITLE-ABS-KEY((gui OR "graphical user interface"OR "front-end"OR frontend) AND (testing OR test OR teste) AND (automation OR automated OR automacao OR automatizado) AND (web))*	82

[2] https://ieeexplore.ieee.org/
[3] https://sol.sbc.org.br/
[4] https://www.scielo.br/
[5] https://www.scopus.com/

Foram estabelecidos o critério de inclusão (CI) e de exclusão (CE) afim de responder adequadamente às QPs propostas. Com esses critérios, foi possível identificar as ferramentas de automatização de testes desenvolvidas e utilizadas nos trabalhos publicados nos últimos cinco anos.

Os estudos retornados foram selecionados a partir dos CI e CE definidos no protocolo, descritos a seguir:

3.3 CRITÉRIOS DE INCLUSÃO

Com o objetivo de mapear o uso da automação de testes em interfaces web, foi estabelecido um critério de inclusão: somente trabalhos que proponham ou utilizem ferramentas ou técnicas de automação de testes para interfaces web foram considerados.

CI 01 - O estudo propõe ou utiliza alguma ferramenta ou técnica de automatização de testes de interfaces de sistemas *web*.

3.4 CRITÉRIOS DE EXCLUSÃO

Para garantir a consistência e a relevância dos dados analisados, foram rejeitados estudos publicados fora do período de 2019 a 2023, assegurando assim que a análise focasse apenas nas práticas mais recentes. O segundo critério desconsidera artigos de revisão e outros tipos de publicações não primárias, como resumos de conferências, estudos secundários e terciários. Rejeitando também retornos como livros e capítulo de livros.

Entre os idiomas publicados, foram selecionados estudos em inglês, por ser o idioma predominante na publicação de pesquisas científicas e técnicas, e o português com o objetivo de identificar se havia pesquisas relevantes publicadas nacionalmente sobre o tema.

Também foram desconsiderados artigos que não tratavam especificamente de ferramentas ou técnicas de automatização de testes em interfaces web. E para garantir a acessibilidade e a transparência na análise da literatura, foi estabelecido o critério de exclusão para artigos que não estavam disponíveis gratuitamente. Este critério foi adotado para assegurar que todos os estudos incluídos na revisão pudessem ser acessados e avaliados de forma equitativa.

CE 01 - Trabalhos publicados anteriores a 2019 ou posteriores a 2023.

CE 02 - Livros, capítulos de livros, *reviews*, estudos secundários e terciários.

CE 03 - O estudo não está escrito em inglês ou português.

CE 04 - O estudo completo não está disponível gratuitamente.

CE 05 - O estudo não propõe ou utiliza alguma ferramenta ou técnica de automatização de testes de interfaces de sistemas *web*.

3.5 SELEÇÃO DOS ESTUDOS

A somatória de estudos selecionados para leitura e análise foram de 206 trabalhos, e a seleção deles foram dividas em duas sessões principais. A primeira sessão em que se foi realizada uma triagem lendo-se os metadados como título e o resumo, verificando assim se adequariam aos critérios de inclusão ou exclusão. Nessa primeira fase foram aceitos 54 estudos e 122 foram rejeitados. Também foram encontrados 30 trabalhos duplicados que foram desconsiderados do estudo.

Na segunda sessão, foram realizadas leituras completas dos 54 artigos pré-selecionados. Destes, 14 estudos foram rejeitados por não se adequarem ao CI01 e destaca-se que três trabalhos identificados como relevantes, não tiveram seus textos completos disponíveis gratuitamente, o que resultou em sua rejeição. Por fim, resultando assim, um total final de 37 trabalhos aceitos, como apresentado na Figura 1.

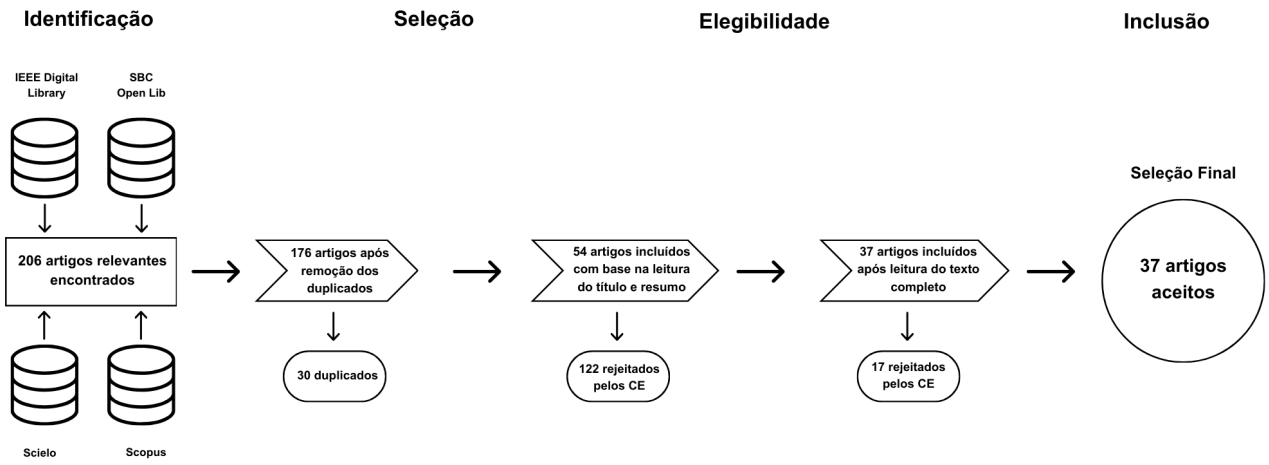

Figura 1 – Fluxo do processo de seleção dos estudos.

A lista completa com os 37 artigos selecionados foram tabulados e estão descritos na Tabela 2.

Tabela 2 – Trabalhos aceitos após critérios do MSL.

Título	**Fonte**	**Referência**
Similarity-based Web Element Localization for Robust Test Automation	*Scopus*	(Nass et al., 2023b)
Web Performance Engineering-Finding Best Data Structure for Automatically Collected HTML Locators	*Scopus*	(Padhy; Nayak; Vaidya, 2023)

Tabela 2 – Trabalhos relevantes analisados neste MSL (*Continuação*).

Título	**Fonte**	**Referência**
Automated Model-Based Test Case Generation for Web User Interfaces (WUI) From Interaction Flow Modeling Language (IFML) Models	*IEEE DL*	(Yousaf et al., 2019)
Automating GUI Response Time Measurements in Mobile and Web Applications	*IEEE DL*	(Quental et al., 2019)
Research on Automation Testing Technology Based on Image Recognition	*IEEE DL*	(Wang; Wu, 2019)
Wait, Wait. No, Tell Me. Analyzing Selenium Configuration Effects on Test Flakiness	*IEEE DL*	(Presler-Marshall et al., 2019)
An industrial study on applications of combinatorial testing in modern web development	*Scopus*	(Ozcan, 2019)
Web automation using selenium web driver python	*Scopus*	(Neethidevan; Chandrasekaran, 2019)
Behavior-driven development (BDD) Cucumber Katalon for Automation GUI testing case CURA and Swag Labs	*IEEE DL*	(Bahaweres et al., 2020)
Artificial intelligence in automated system for web-interfaces visual testing	*Scopus*	(Ivanova et al., 2020)
Test Automation with the Gauge Framework: Experience and Best Practices	*Scopus*	(Garousi et al., 2020)
Automatic Web Testing Using Curiosity-Driven Reinforcement Learning	*IEEE DL*	(Zheng et al., 2021)
Comparative Study of Open Source Automation Testing Tools: Selenium, Katalon Studio & Test Project	*IEEE DL*	(Majeed et al., 2021)
Zero WFuzzer: Target-Oriented Fuzzing for Web Interface of Embedded Devices	*IEEE DL*	(Zheng; Dong; Cheng, 2021)
A Novel approach of GUI Mapping with image based widget detection and classification	*Scopus*	(Jaganeshwari; Djodilatchoumy, 2021)
Image-based approaches for automating GUI testing of interactive web-based applications	*Scopus*	(Macchi et al., 2021)
Model-based testing leveraged for automated web tests	*Scopus*	(Mattiello; Endo, 2022)
STILE: A Tool for Parallel Execution of E2E Web Test Scripts	*Scopus*	(Olianas et al., 2021)
Semi-Automated Behavior-Driven Testing for the Web Front-Ends	*Scopus*	(Ma et al., 2023)

Tabela 2 – Trabalhos aceitos após critérios do MSL (*Continuação*).

Título	**Fonte**	**Referência**
Automatically identifying potential regressions in the layout of responsive web pages	*Scopus*	(Walsh; Kapfhammer; McMinn, 2020)
WebEvo: Taming web application evolution via detecting semantic structure changes	*Scopus*	(Shao, 2021)
An Automated Testing Tool Based On Graphical User Interface With Exploratory Behavioural Analysis	*Scopus*	(Jaganeshwari; Djodilatchoumy, 2022)
Evaluation for Web GUI Automation Testing Tool - Experiment	*Scopus*	(Azzam et al., 2022)
Testing React Single Page Web Application using Automated Testing Tools	*Scopus*	(Hasan et al., 2022)
Test Case Auto-Generation for Web Applications: A Model-Based Approach	*Scopus*	(Othman; Zein, 2022)
Identification and Validation of Web Themes: A DOM-Structure Matching Approach	*Scopus*	(Nguyen et al., 2022)
An empirical study of async wait flakiness in front-end testing	*Scopus*	(Pei et al., 2022)
A Comprehensive Evaluation of Q-Learning Based Automatic Web GUI Testing	*IEEE DL*	(Fan et al., 2023)
An Empirical Study on Web GUI Load Testing and the Hybrid with HTTP Requests	*IEEE DL*	(Lo; Lee, 2023)
An Investigation Into the Efficacy of RANOREX Software Test Automation Tool	*IEEE DL*	(Mallick et al., 2023)
E2E-Loader: A Framework to Support Performance Testing of Web Applications	*IEEE DL*	(Battista et al., 2023)
Robust web element identification for evolving applications by considering visual overlaps	*IEEE DL*	(Nass et al., 2023a)
Browserspot – A Multifunctional Tool For Testing The Front-End Of Websites And Web Applications	*Scopus*	(Binek; Góral, 2023)
Cytestion: Automated GUI Testing for Web Applications	*Scopus*	(Moura et al., 2023)
GUI-Based Software Testing: An Automated Approach Using GPT-4 and Selenium WebDriver	*Scopus*	(Zimmermann; Koziolek, 2023)
Kraken 2.0: A platform-agnostic and cross-device interaction testing tool	*Scopus*	(Ravelo-Méndez; Escobar-Velásquez; Linares-Vásquez, 2023)
Scripted and scriptless GUI testing for web applications: An industrial case	*Scopus*	(Bons et al., 2023)

Após a seleção dos estudos, foram então realizadas as extrações de dados, conduzidos por meio de um formulário com as categorias descritas na Tabela 3. As questões descritas foram formuladas com o objetivo de responder as QP que norteiam esta pesquisa, categorizando assim os artigos selecionados. Cada artigo foi lido completamente e realizado o preenchimento do formuários descrito, utilizando-se da ferramenta Parsifal. Logo após a categorização, os dados extraídos foram exportados para uma planiha eletrônica, na qual foi analisada e manipulada para se gerar as informações descritas na seção 4 deste estudo.

Tabela 3 – Formulário de categorias para extração de dados.

Descrição	**Tipo**	**Valores**
Qual técnica de teste o estudo utiliza?	Seleção Múltipla	Teste de interface (GUI) Teste de Carga Teste de Desempenho Teste *End-to-end* (E2E)
Qual o sistema sob teste (SUT)?	Descritiva	-
É um estudo comparativo entre ferramentas existentes?	*Boolean*	*True* *False*
Qual *framework* existente utiliza?	Descritivo	-
O estudo desenvolveu alguma ferramenta de automatização de testes?	*Boolean*	*True* *False*
Descrição da ferramenta e/ou metodologia proposta	Descritivo	-
Qual técnica e/ou metodologia existentes o estudo utiliza?	Descritivo	-
Utiliza alguma técnica de *machine learning* ou de inteligência artificial?	*Boolean*	*True* *False*
Quais limitações foram reportadas pelo estudo?	Descritiva	-

4 RESULTADOS E DISCUSSÃO

Esta seção apresenta as análises obtidas a partir da extração dos dados de cada um dos 37 estudos selecionados. Tais resultados permitem responder as QPs propostas por esse mapeamento sistemático da literatura.

Em um aspecto geral, em relação os trabalhos aceitos por ano de publicação (Figura 2), foi possível observar que nos últimos anos houve um aumento significativo quanto ao número de publicações. Coerente assim, como já previsto por Banerjee et al. (2013), a automatização de interfaces, e em específico de aplicações *web* em larga escala, se tornam uma tendência e ganha cada vez mais visibilidade e investimento.

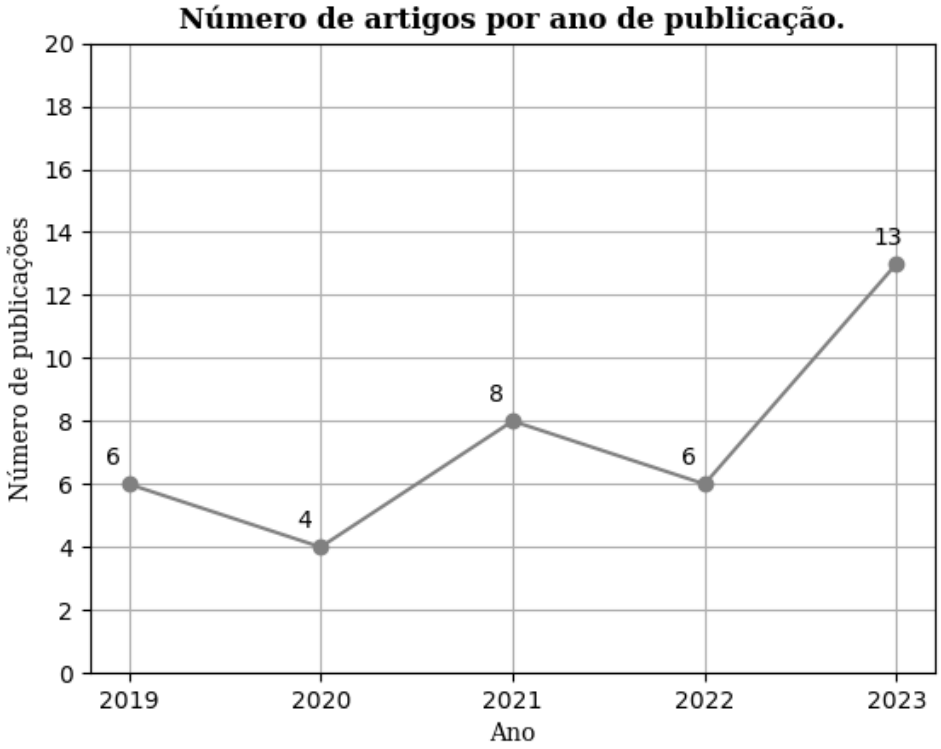

Figura 2 – Número de artigos por ano de publicação.

4.1 QUESTÃO DE PESQUISA 01

QP01 - Quais ferramentas têm sido utilizadas e/ou desenvolvidas para automatização de testes em GUI de sistemas *web*?

Dentre os estudos analisados, 25 artigos (64,8%) propuseram uma nova ferramenta e/ou metodologia de automatização de teste, como demonstrado na Figura 3. As ferramentas e técnicas desenvolvidas estão descritas na Tabela 4.

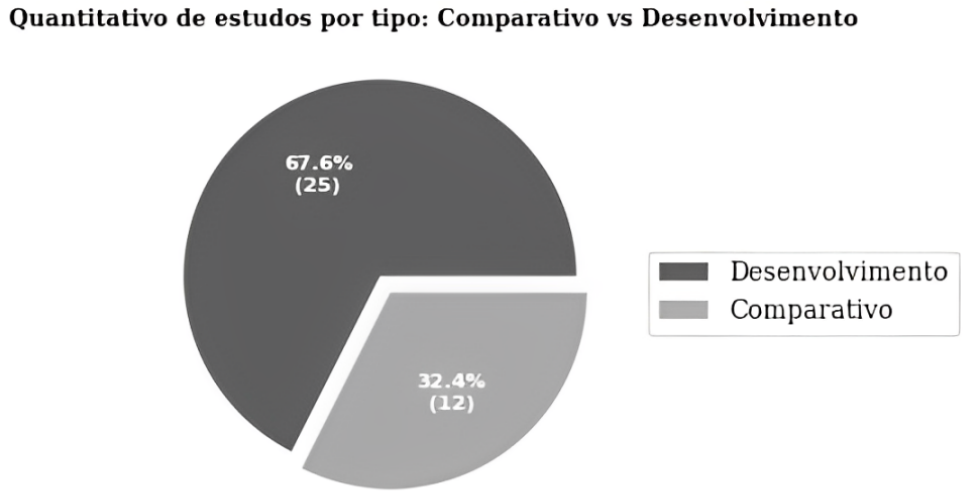

Figura 3 – Quantitativo de estudos por tipo: Comparativo vs Desenvolvimento.

Foi observado que na maioria dos estudos que relataram uma nova metodologia, também propuseram uma ferramenta que implementasse a técnica desenvolvida. As ferramentas desenvolvidas pelos estudos, foram majoritariamente codificadas em linguagem *Java* e *JavaScript*, como pode ser visto pelo gráfico presente na Figura 4.

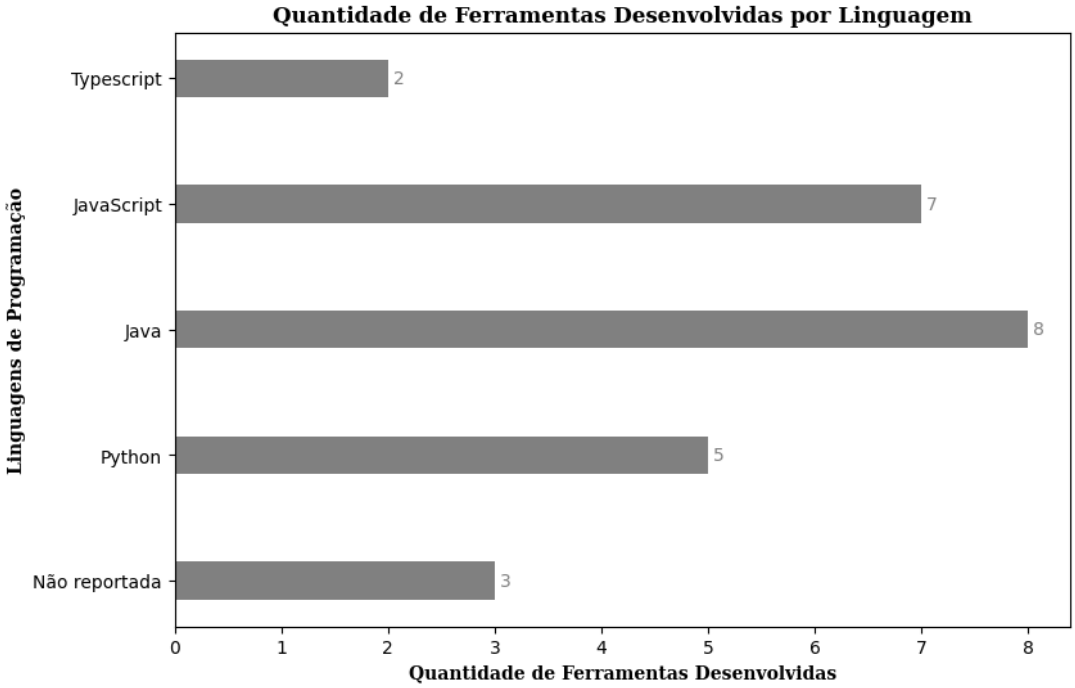

Figura 4 – Quantidade de ferramentas desenvolvidas por linguagem.

Tabela 4 – Relação de ferramentas e metodologias propostas.

Artigo	**Ferramenta/Metodologia**
Automating GUI Response Time Measurements in Mobile and Web Applications	Propôs uma nova metodologia que permite medir os tempos de resposta dos cenários de teste definidos pelos *designers* para aplicações *web* e *mobile* com pequenas modificações nos códigos de automação de testes.
A Comprehensive Evaluation of Q-Learning Based Automatic Web GUI Testing	Propôs um *framework* genérico de automação de testes de interfaces *web* baseado em QL, *Q-learning* é um modelo livre de *machine learning* bastante adotado na testagem GUI.
E2E-Loader: A Framework to Support Performance Testing of Web Applications	Este estudo introduz o E2E-Loader, uma técnica automatizada para projetar testes de desempenho em aplicações *web*. A ferramenta cria testes de desempenho a partir de casos de teste funcional *end-to-end*, permitindo que esses casos sejam executados antes que os usuários necessitem interagir com o sistema.
An Empirical Study on Web GUI Load Testing and the Hybrid with HTTP Requests	Foi desenvolvido uma técnica utilizando o *plugin* JMeter para possibilitar a realização de testes de carga em interfaces *front-end*.
Automated Model-Based Test Case Generation for Web User Interfaces (WUI) From Interaction Flow Modeling Language (IFML) Models	Propuseram uma nova técnica e uma ferramenta para gerar automaticamente casos de teste da interface de sistemas *web*, utilizando modelos da Linguagem de Modelagem de Fluxo de Interação (IFML).
Automatic Web Testing Using Curiosity-Driven Reinforcement Learning	Propõe um *framework* chamado WebExplor, que utiliza aprendizado por reforço orientado pela curiosidade para gerar sequências de ações (casos de teste) de alta qualidade.
Robust web element identification for evolving applications by considering visual overlaps	Este artigo propõe o conceito inovador de Nós Visualmente Sobrepostos (VON), que reduz a fragilidade utilizando o fenômeno de que elementos visuais da *web* (observados pelo usuário) são construídos a partir de múltiplos elementos *web* no *Document Object Model (DOM)* que se sobrepõem visualmente.
Zero WFuzzer: Target-Oriented Fuzzing for Web Interface of Embedded Devices	Este artigo propõe um *framework* chamado ZeroWFuzzer que combina a interface *web* de dispositivos embarcados com o binário do *firmware*, gerando automaticamente casos de teste através da interface *web* e utilizando instrumentação binária para orientar os testes de *fuzzing*.
Artificial intelligence in automated system for web-interfaces visual testing	Neste artigo, os autores propuseram criar um sistema que será integrado em conjuntos de testes automatizados funcionais, desempenhando a função de monitorar e analisar mudanças visuais na interface gráfica de aplicativos *web*.

Tabela 4 – Relação de ferramentas e metodologias propostas (*Continuação*).

Artigo	**Ferramenta/Metodologia**
Web Performance Engineering-Finding Best Data Structure for Automatically Collected HTML Locators	Foi desenvolvido uma técnica que ajuda a identificar a melhor estrutura de dados e detecção automática de localizadores de elementos *web*.
Semi-Automated Behavior-Driven Testing for the Web Front-Ends	O estudo propõe combinar os conceitos de *Behavior-Driven Development* e a ferramenta Cucumber, com a gravação automatizada de testes (*Selenium*) e os conceitos de testes de aceitação para propor uma nova técnica chamada "Testes BDD baseados em Gravador para *front-end* (RBTF)".
Browserspot – A Multifunctional Tool For Testing The Front-End Of Websites And Web Applications	Foi desenvolvida a ferramenta BrowserSpot para automatizar a validação de páginas de *websites*, garantindo sua visibilidade adequada em diferentes dispositivos e resoluções.
Automatically identifying potential regressions in the layout of responsive web pages	O trabalho propõe a técnica REDECHECK (*Responsive Design Checker*) para detectar regressões em páginas *web* responsivas. A ferramenta extrai a responsividade dos *layouts* de duas versões da página e compara essas informações, alertando os desenvolvedores sobre quaisquer diferenças nos *layouts* para investigação detalhada.
Similarity-based Web Element Localization for Robust Test Automation	Este artigo propõe e avalia uma nova abordagem chamada localização de elementos da baseada em similaridade, chamada Similo, que aproveita informações de vários parâmetros de localização de elementos da *web* para identificar um alvo elemento usando uma pontuação de similaridade ponderada.
A Novel approach of GUI Mapping with image based widget detection and classification	Desenvolveram uma técnica para melhorar a testagem automatizada de interfaces *web*, usando técnicas de *machine learning* para detecção e classificação de *widgets* GUI a partir de capturas de tela.
GUI-Based Software Testing: An Automated Approach Using GPT-4 and Selenium WebDriver	Propõe um método para testagem GUI em aplicações *web* que automatiza em grande escala o processo de testagem integrando a linguagem avançada baseada em GPT-4 com o *framework* Selenium.
Kraken 2.0: A platform-agnostic and cross-device interaction testing tool	Desenvolvido a Kraken 2.0, uma ferramenta que permite testers a escrever, executar e validar cenários de teste envolvendo a interação de duas ou mais *web* ou dispositivos *mobile*.

Tabela 4 – Relação de ferramentas e metodologias propostas (*Continuação*).

Artigo	**Ferramenta/Metodologia**
An Automated Testing Tool Based On Graphical User Interface With Exploratory Behavioural Analysis	Esse estudo propõe uma metodologia e uma ferramenta inovadora para detecção de *widgets*, classificação de *widgets* e cobertura de testes usando conceitos de *machine learning* e processamento de imagens.
Image-based approaches for automating GUI testing of interactive web-based applications	Este artigo propõe uma nova ferramenta baseada em imagens que combina técnicas atuais com novas abordagens. Estas aproveitam algoritmos de *Machine Learning* e Visão Computacional para analisar capturas de tela da interface para validação.
STILE: A Tool for Parallel Execution of E2E Web Test Scripts	Propôs STILE, uma ferramenta para a execução paralela de *scripts* de teste *web* que garante o cumprimento de todos os cronogramas de execução com as dependências entre os *scripts* de teste envolvidos
Cytestion: Automated GUI Testing for Web Applications	O estudo propõe o Cytestion, uma ferramenta que explora automaticamente a interface gráfica de aplicação *web*, detectando erros que causam falhas visíveis, como travamentos, erros e comportamentos inesperados.
Model-based testing leveraged for automated web tests	Desenvolvida uma ferramenta protótipo denominada MoLeWe (*Model-based testing Leveraged for Web tests*), na qual apresenta uma abordagem que utiliza testes existentes para inferir um modelo, o qual é então utilizado por testadores para incorporar novos testes, com o código de teste sendo gerado posteriormente.
WebEvo: Taming web application evolution via detecting semantic structure changes	WebEvo é uma ferramenta desenvolvida para automatizar o monitoramento *web*, o qual detecta mudanças na estrutura semântica dos elementos do DOM.
Test Case Auto-Generation for Web Applications: A Model-Based Approach	Um *framework*, denominado MAJD, uma nova abordagem baseada em modelo que gera automaticamente casos de teste para ferramenta de teste Selenium, utilizando recursos visuais específicos de domínio Linguagem (DSVL) e linguagem textual específica de domínio (DSTL).
Identification and Validation of Web Themes: A DOM-Structure Matching Approach	Esta pesquisa propõe um sistema para permitir a identificação automática de modelos de temas da *web* em páginas da *web* existentes, e para examinar a conformidade de novas páginas da *web*, com base no modelo de tema da *web* identificado.

Entre os 12 estudos (32,4%), catalogados como comparativos, obteve-se a evidência de

que 7 deles, cerca de 58,3%, relataram um estudo comparativo apenas entre ferramentas já existentes. Outros 4 estudos produziram uma comparação apenas de técnicas e apenas um estudo realizou ambas comparações, entre técnica e ferramenta, como ilustrado na Figura 5.

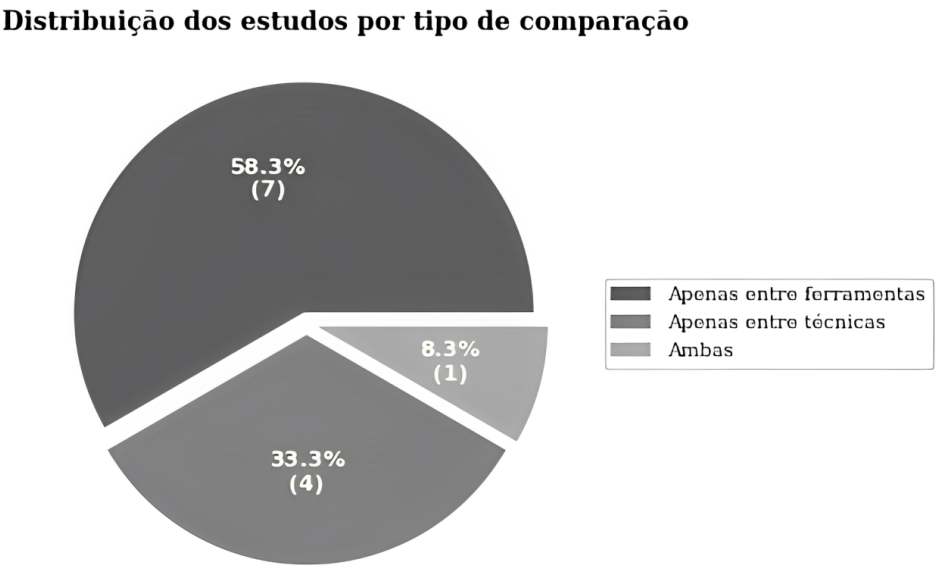

Figura 5 – Distribuição dos estudos por tipo de comparação

Foram encontrados 8 estudos comparativos entre *frameworks* existentes no mercado, o Selenium[1] foi a ferramenta com mais relatos entre os estudos, sendo analisado em 5 deles (62,5%). Outros *frameworks* citados foram: Ranorex[2], Cypress[3], Jest[4], Enzyme[5], Gauge[6] e TestComplete[7].

Foram reportados estudos comparativos entre técnicas e/ou metodologias de testagem, na qual eram propostas uma aprimoração em ferramentas já existentes ou desenvolvidas novas técnicas de automatização de testes. Bahaweres et al. (2020) por exemplo, conduziu um estudo no qual propõe a aplicação do método *"Behavior-driven development (BDD)"* na automação de testagem de interfaces *web* com o *framework* Cucumber[8].

Já Wang e Wu (2019), desenvolveram uma nova técnica de automatização de testes baseados em reconhecimento de imagem. O reconhecimento de imagem consegue verificar a visibilidade dos elementos da página e as posições relativas entre eles e assim pode

[1] https://www.selenium.dev/
[2] https://www.ranorex.com/
[3] https://www.cypress.io/
[4] https://jestjs.io/pt-BR/
[5] https://enzymejs.github.io/enzyme/
[6] https://gauge.org/index.html
[7] https://smartbear.com/product/?product=TestComplete
[8] https://cucumber.io/

permitir os testes em diferentes sistemas operacionais. O método proposto de testes baseados em testes de reconhecimento de imagem mostraram vantagens únicas na testagem do *software*, que podem efetivamente verificar a interface gráfica, melhorar o desempenho dos testadores e a eficiência do trabalho, economizando custos e acelerando a interação de desenvolvimento e versão de *software*.

No estudo empírico desenvolvido por Pei et al. (2022), foram realizados investigações quanto as causas e os impactos do das falhas dos *"async wait"*, estruturas as quais, simplificam a codificação de funções assíncronas e permite a escrita de forma semelhante ao código síncrono em *scripts* de testes *front-end*. Os resultados preliminares sugerem que os testes que dependem de um tempo explícito para esperar para sincronizar os testes, tendem a criar mais instabilidade do que sincronizar em o status dos elementos DOM.

Neethidevan e Chandrasekaran (2019) analisaram a utilização do *framework* Selenium WebDriver, implementando os casos de testes automatizados em linguagem *Python*, realizando comparação sobre a técnica de testagem em diferentes tipos de *browsers*.

4.2 QUESTÃO DE PESQUISA 02

QP02 - Quais técnicas de teste estão sendo combinadas na automatização de testes em GUI de sistemas *web*?

O método de testagem de ponta-a-ponta, conhecido como E2E, foi utilizado em praticamente todos os estudos. Tal caracteristica é justificável ao se considerar que a interface gráfica de um sistema está intrinsecamente ligada a todos os seus processos. Como descrito por Banerjee et al. (2013) e Fulcini e Ardito (2022a) a testagem de GUI em sistemas *web* é uma tarefa complexa e difícil, pelo fato de serem responsáveis por recolher os dados de entrada e apresentar todos os dados de saída, suas falhas e erros impactam toda a funcionalidade do sistema.

De acordo com Leotta et al. (2016), os testes E2E dependem diretamente das alterações de interface, e essa dependência pode gerar essa convergência entre testes E2E e GUI, como visto nos estudos. Formando assim uma suíte de testes combinatoriais extremamente eficazes, que garantem as funcionalidades do componentes da página, juntamente com suas funcionalidades dentro do processo do sistema.

Foram encontrados 4 estudos que relatavam ferramentas de automatização de testes utilizando-se de recursos de geração de casos de testes. Othman e Zein (2022); Battista et al. (2023); Yousaf et al. (2019) e Zheng et al. (2021) propuseram uma nova técnica/ferramenta para gerar automaticamente casos de teste da interface de sistemas *web* a partir dos modelos. Tais propostas demonstram pesquisas em desenvolvimento de automação na geração dos casos de testes estão sendo realizadas com maior frequência dentro da área.

Lo e Lee (2023) desenvolveram uma técnica na qual permite-se que sejam realizados testes de cargas nas interfaces *web*, possibilitando assim encontrar erros e falhas de *delay*

nesses sistemas. O estudo experimental dessa técnica, também relatado pelos autores, indicaram que testes de carga de *front-end* expuseram *delays* que não foram capturados nos testes apenas no *back-end*, como comumente é realizado.

Portanto, é possível observar que a testagem GUI consegue ser combinada com demais tipos de teste, realizando assim os chamados testes combinatoriais, que são utilizados para aumentar a abrangência dos testes, e assim validar e verificar maiores partes do sistema.

4.3 QUESTÃO DE PESQUISA 03

QP03 - Quais limitações têm sido reportadas na automatização de testes em GUI de sistemas *web?*

As limitações reportadas nos estudos analisados foram tabuladas e correlacionadas entre si, formulando então, a seguinte Tabela 5. Dentre os estudos analisados, 8 deles não reportaram quaisquer limitações e portanto não foram analisados neste tópico.

Quanto as limitações dos *frameworks* existentes, Mallick et al. (2023) reportaram dificuldades quanto ao uso do Ranorex, revelando uma baixa integralidade com outras bibliotecas ou aplicativos adicionais e um custo orçamentário alto. Já Quental et al. (2019) acusou a necessidade do Selenium WebDriver possuir uma integração com o sistema operacional, possibilitando assim cenários de casos de teste onde existem a necessidade de realizar carregamento de arquivos presentes no sistema. Presler-Marshall et al. (2019) demonstrou em seu estudo que o *hardware* tem um impacto significativo no tempo de execução e na confiabilidade de um teste suíte desenvolvida em Selenium, e que um ambiente de entrega contínua mais rápido representa um impacto significativo na execução dos testes.

Sobre o *framework* Gauge, o estudo de Garousi et al. (2020) descreve que enquanto o Gauge oferece um ambiente poderoso para desenvolver *scripts* de teste em linguagem natural, a sua flexibilidade pode se tornar uma desvantagem se não forem estabelecidos padrões e práticas adequadas para o *design* sistemático e a manutenção dos *scripts*. As dificuldades destacadas indicam a importância de uma abordagem disciplinada e estruturada no desenvolvimento de suítes de testes automatizados utilizando a ferramenta.

Relatos sobre o TestComplete, como o de Azzam et al. (2022), informaram o custo orçamentário alto e que ao usar uma de suas ferramentas adicionais, a geração de caso por gravação, um objeto ou a espera dele, pode não ser reconhecida e portanto é necessário um ajuste manual destes casos.

Tabela 5 – Relação de limitações e/ou dificuldades reportadas.

Limitações	**Artigo**
Custo orçamentário e técnico elevado	Mallick et al. (2023), Zheng et al. (2021), Azzam et al. (2022), Zheng, Dong e Cheng (2021)
Pouca integração com bibliotecas adicionais	Mallick et al. (2023)
Não possui integração com o sistema operacional, impossibilitando interações com *upload* de arquivos	Quental et al. (2019)
O treinamento de modelos abrangentes de Q-learning são custosos e longos	Fan et al. (2023)
Dificuldade em generalizar para diferentes tipos de aplicações *web*	Fan et al. (2023), Zheng et al. (2021), Walsh, Kapfhammer e McMinn (2020), Zimmermann e Koziolek (2023), Mattiello e Endo (2022)
Não há suporte para análise de resultados e desempenho	Battista et al. (2023)
Abrangência de elementos limitada	Yousaf et al. (2019)
Testes abstratos limitando a utilidade imediata	Yousaf et al. (2019)
A configuração de *hardware* causou uma limitação na execução	Presler-Marshall et al. (2019)
Método de aprendizado por reforço é limitado pela complexidade das ações de um teste *web*	Zheng et al. (2021)
Sensível à escolha de hiperparâmetros	Zheng et al. (2021)
Limitados apenas para abordagens de multi-localizadores	Nass et al. (2023a)
Variação das informações necessárias para localizar um elemento *web*	Nass et al. (2023a)
Limitação na abrangência de adaptação em diferentes arquiteturas	Zheng, Dong e Cheng (2021)
Dificuldade na escolha das técnicas de inteligência artificial	Ivanova et al. (2020)
Há limitações quanto a acuracidade podendo haver erros de sintaxe	Ma et al. (2023), Othman e Zein (2022), Jaganeshwari e Djodilatchoumy (2022)

Tabela 5 – Relação de limitações e/ou dificuldades reportadas (*Continuação*).

Limitações	**Artigo**
Falta de parâmetros de medida de eficiência dos testes	Bons et al. (2023)
Dificuldades na implementação da ferramenta no time	Bons et al. (2023)
Dependência de acesso à internet durante o tempo de execução dos testes	Binek e Góral (2023)
Divergências sobre os conceitos de classificação dos *layouts*	Walsh, Kapfhammer e McMinn (2020)
A inexistência de padrões e práticas bem estabelecidas e estruturada podem afetar o desempenho da ferramenta	Garousi et al. (2020)
Desafios relacionados a definição de valores limites	Nass et al. (2023b)
Latências dos sistemas *web* podem gerar inconsistências nas buscas da ferramenta na página	Nass et al. (2023b)
Dificuldade ao ser aplicado em diferentes plataformas	Nass et al. (2023b)
Custo elevado de tempo na interpretação do modelo de teste combinatorial	Ozcan (2019)
Não possuem criptografia das informações sensíveis	Ravelo-Méndez, Escobar-Velásquez e Linares-Vásquez (2023)
Dependência de capturas de telas	Macchi et al. (2021)
Alto custo computacional de manutenção	Macchi et al. (2021), Moura et al. (2023)
Ambientes com recursos limitados podem sofrer diminuição na eficiência	Olianas et al. (2021)
Determinação de número ideal de computação é complexa	Olianas et al. (2021)
Ferrramenta não suporta testes com estruturas de repetição e condicionais	Mattiello e Endo (2022)
Dependência da conservação e estruturação devida das nomenclaturas dos arquivos do sistema	Shao (2021)
Depêndencia de uma árvore base	Olianas et al. (2021) Nguyen et al. (2022)
Dependência de um padrão específico	Mattiello e Endo (2022)
Ferrramenta não suporta testes com estruturas de repetição e condicionais	Mattiello e Endo (2022)

Entre as tecnologias usadas e reportadas pelos estudos, é notório a utilização de técnicas baseadas em inteligência artificial (IA), em especial, a técnica de *Machine Learning* (ML). A IA é uma ciência que busca criar máquinas capazes de aprender com dados, reconhecer padrões e tomar decisões de forma autônoma. O ML é uma metodologia que permite aos computadores realizar tarefas específicas de forma inteligente, aprendendo com exemplos e modelos, sem precisarem de uma codificação direta (Society, 2017).

Cerca de 16,2% dos estudos analisados relataram utilizar alguma metodologia ou técnica de IA/ML, identificados somente a partir de 2020, indicando assim uma possível tendência de pesquisas nessa área (vide Figura 6).

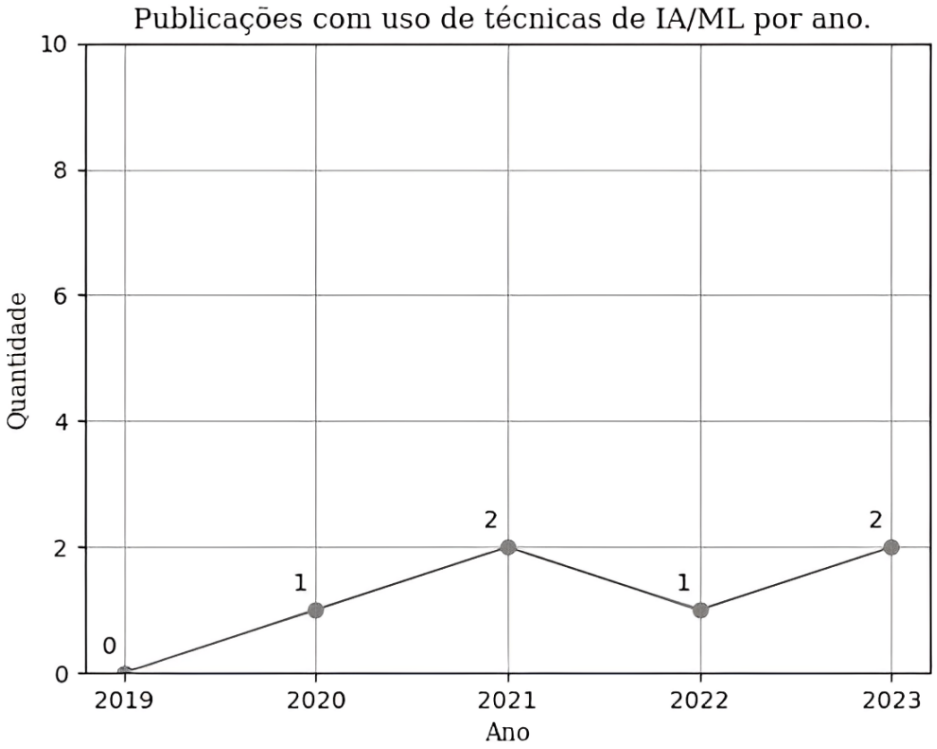

Figura 6 – Publicações com uso de técnicas de IA/ML em testagem GUI por ano.

Dos estudos que utilizaram alguma técnica de ML, a principal dificuldade reportada foi sobre a generalização do método, e dos modelos gerados, para diferentes tipos de aplicações *web*. O custo computacional foi comumente acusado, devido ao treinamento de modelos de aprendizagem serem extremamente demorados e exaustivos computacionalmente.

Diante as múltiplas limitações encontradas, nota-se que a utilização de testes automatizados em interfaces *web* ainda enfrenta muitos desafios e obstáculos. Dando espaço para que novas pesquisas e investigações explorem diferentes abordagens e propostas de solução para os problemas em questão.

Para aumentar a confiabilidade dos resultados e das análises, adotou-se um procedimento de revisão dupla. A revisão foi conduzida pela autora do trabalho e pelo seu

orientador. Ambos participaram ativamente da verificação dos dados e da interpretação dos resultados. A abordagem de dupla revisão foi crucial para validar a consistência dos resultados e garantir a integridade da análise.

5 CONSIDERAÇÕES FINAIS

Este trabalho realizou um mapeamento sistemático da literatura sobre a utilização dos testes automatizados no processo de garantia de qualidade em sistemas que possuam interfaces *web*. Tendo como objetivo responder sobre três principais QPs: i) mapear e descrever as principais as principais ferramentas utilizadas e/ou desenvolvidas para a automatizações de testes de interfaces de sistemas *web*; ii) identificar quais técnicas de teste estão sendo combinadas na automatização de testes em GUI de sistemas *web* e iii) encontrar as principais limitações reportadas pelos autores.

Foram analisados 37 estudos primários, os quais foram categorizados e analisados diante o formulário de extração de dados (vide Tabela 3). Com base nos estudos selecionados, foram identificadas as ferramentas e metodologias desenvolvidas e utilizadas na automação de interfaces *web*. Possibilitando assim uma tabulação ds dados e uma compactação das informações de cada estudo (vide Tabela 4), respondendo assim a **QP01**. Cerca de 64,8% dos estudos analisados desenvolveram uma ferrramenta e/ou metodologia de automatização de testes em interfaces *web*. Entre esses estudos observou-se que a maior parte dos estudos que propunham uma técnica de testagem também realizaram o desenvolvimento de uma ferramenta que exercesse tal metodologia. Por fim, foram mapeados as principais linguagens de programação usadas nas implementações reportadas, sendo elas, respectivamente, *Java, JavaScript, Python e TypeScript*.

Em resposta a **QP02**, a testagem E2E foi a técnica com o maior número de relatados, estando presente em praticamente todos os estudos encontrados. Essa presença significativa se baseia na dependência dos testes E2E com as interações na interface, fazendo com que as ferramentas desempenhem testes combinatoriais entre a testagem GUI e E2E, aproveitando assim dos métodos de testagem. Outras técnicas igualmente foram reportadas, como o teste de carga e a geração de casos de teste, demonstrando assim como os testes em interface gráfica podem conter combinações, e assim garantir uma maior abrangência do sistema.

Para responder a **QP03**, foram extraídos dos estudos as limitações relatadas pelos autores, em análise, observou-se algumas convergência entre as dificuldades reportadas. A listagem das limitações foram tabuladas na Tabela 5 e pôde-se concluir que custos orçamentários, computacionais e técnicos foram relatados com mais frêquencia. Notou-se além disto, que os estudos comparativos de ferramentas existentes como Selenium, Gauge e TestComplete reportaram dificuldades quanto ao uso do *framework*, tais como: tempo de execução dependente de configurações do *hardware*, baixa integrabilidade com outras aplicações e bibliotecas e o custo orçamentário alto em suas implementações.

A complexidade crescente das interfaces *web* e os desafios inerentes à automação de seus testes impulsionam a necessidade de pesquisas que explorem novas abordagens e tec-

nologias, visando superar as limitações atuais e oferecer soluções mais robustas e eficientes. Portanto os desafios atuais abrem caminho para o desenvolvimento de novas ferramentas que possam transformar o processo de testagem, garantindo que a qualidade de interfaces *web* seja cada vez mais segura e confiável.

Os resultados presentes neste mapeamento sistemático permitem caracterizar a utilização de testes automatizados em interfaces *web*. Consolidando uma visão sistematizada sobre as ferramentas e técnicas propostas e desenvolvidas nos últimos 5 anos, bem como suas limitações e dificuldades. Como trabalhos futuros, pretende-se abranger a área de pesquisa e investigar sobre a utilização de técnicas de *machine learning* e inteligência articial no desenvolvimento de ferramentas de testagem automatizada para sistemas *web*, revelado neste estudo como uma tendência.

REFERÊNCIAS

AZZAM, F. et al. Evaluation for web gui automation testing tool - experiment. **ISMSIT 2022 - 6th International Symposium on Multidisciplinary Studies and Innovative Technologies, Proceedings**, p. 549 – 554, 2022. Cited by: 1. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85142772212&doi=10.1109%2fISMSIT56059.2022.9932773&partnerID=40&md5=d712d21f58677478555d2863bc804524.

BAHAWERES, R. B. et al. Behavior-driven development (bdd) cucumber katalon for automation gui testing case cura and swag labs. In: **2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)**. [S.l.: s.n.], 2020. p. 87–92.

BANERJEE, I. et al. Graphical user interface (gui) testing: Systematic mapping and repository. **Information and Software Technology**, v. 55, n. 10, p. 1679–1694, 2013. ISSN 0950-5849. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950584913000669.

BATTISTA, E. et al. E2e-loader: A framework to support performance testing of web applications. In: **2023 IEEE Conference on Software Testing, Verification and Validation (ICST)**. [S.l.: s.n.], 2023. p. 351–361.

BERNARDO, P. C.; KON, F. A importância dos testes automatizados. **Engenharia de Software Magazine**, v. 1, n. 3, p. 54–57, 2008.

BINEK, S.; GóRAL, J. Browserspot – a multifunctional tool for testing the front-end of websites and web applications; [browserspot – multifunkcjonalne narzĘdzie do testowania front-endu stron internetowych oraz aplikacji sieciowych]. **Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska**, v. 13, n. 4, p. 61 – 65, 2023. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85180526970&doi=10.35784%2fiapgos.5374&partnerID=40&md5=0c18ad8f4f90359f39732d215bed7883.

BISWAL, B. N.; NANDA, P.; MOHAPATRA, D. P. A novel approach for optimized test case generation using activity and collaboration diagram. **International Journal of Computer Applications**, Citeseer, v. 1, n. 14, p. 67–71, 2010.

BONS, A. et al. Scripted and scriptless gui testing for web applications: An industrial case. **Information and Software Technology**, v. 158, 2023. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85148538920&doi=10.1016%2fj.infsof.2023.107172&partnerID=40&md5=e71cefd4034df596773dd4f634482d89.

CARTAXO, E. G. et al. Geração de casos de teste funcional para aplicações de celulares. **Universidade Federal de Campina Grande**, 2006.

CHICANELLI, R. et al. Aspectos sociais, humanos e econômicos da utilização de testes automatizados no desenvolvimento de sistemas. **Memorias de la Décima Octava Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2019)**, 07 2019.

DURELLI, V. H. et al. Machine learning applied to software testing: A systematic mapping study. **IEEE Transactions on Reliability**, IEEE, v. 68, n. 3, p. 1189–1212, 2019.

EATON, C.; MEMON, A. M. Chapter 5 advances in web testing. In: **Computer Performance Issues**. Elsevier, 2009, (Advances in Computers, v. 75). p. 281–306. Disponível em: https://www.sciencedirect.com/science/article/pii/S006524580800805X.

FAN, Y. et al. A comprehensive evaluation of q-learning based automatic web gui testing. In: **2023 10th International Conference on Dependable Systems and Their Applications (DSA)**. [S.l.: s.n.], 2023. p. 12–23.

FULCINI, T.; ARDITO, L. Gamified exploratory gui testing of web applications: a preliminary evaluation. In: **2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. [S.l.: s.n.], 2022. p. 215–222.

FULCINI, T.; ARDITO, L. Gamified exploratory gui testing of web applications: a preliminary evaluation. In: **2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. [S.l.: s.n.], 2022. p. 215–222.

GAROUSI, V. et al. Test automation with the gauge framework: Experience and best practices. **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**, v. 12250 LNCS, p. 458 – 470, 2020. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85093108433&doi=10.1007%2f978-3-030-58802-1_33&partnerID=40&md5=0f34242014b896fdd60b26437fc4a3ba.

HASAN, M. M. et al. Testing react single page web application using automated testing tools. **International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE - Proceedings**, p. 469 – 476, 2022. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85140997560&doi=10.5220%2f0011077900003176&partnerID=40&md5=3502a1662d5f58dd713b371e90d2ed89.

IEEE. Ieee standard glossary of software engineering terminology. **IEEE Std 610.12-1990**, p. 1–84, 1990.

IVANOVA, K. et al. Artificial intelligence in automated system for web-interfaces visual testing. **CEUR Workshop Proceedings**, v. 2604, p. 1019 – 1031, 2020. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85085180679&partnerID=40&md5=97c800d6b45ff1b2639a569fed522d26.

IZABEL, L. R. P. **Testes automatizados no processo de desenvolvimento de softwares**. Dissertação (Mestrado) — Repositório da Politécnica UFRJ, 2014. Disponível em: http://monografias.poli.ufrj.br/monografias/monopoli10012548.pdf.

JAGANESHWARI, K.; DJODILATCHOUMY, S. A novel approach of gui mapping with image based widget detection and classification. In: **2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)**. [S.l.: s.n.], 2021. p. 342–346.

JAGANESHWARI, K.; DJODILATCHOUMY, S. An automated testing tool based on graphical user interface with exploratory behavioural analysis. **Journal of Theoretical and Applied Information Technology**, Little Lion Scientific, v. 100, n. 22, November 2022. ISSN 1992-8645. © 2022 Little Lion Scientific. Disponível em: http://www.jatit.org.

JORGENSEN, P. C. **Software Testing–a Craftsman Approach**. [S.l.]: CRC Press, 1995.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing systematic literature reviews in software engineering – version 2.3. Keele/Staffs-UK and Durham-UK, 2007.

KOUSAR, A. et al. A systematic review on pattern-based gui testing of android and web apps: State-of-the-art, taxonomy, challenges and future directions. In: **2023 25th International Multitopic Conference (INMIC)**. [S.l.: s.n.], 2023. p. 1–7.

KOUSAR, A. et al. A systematic review on pattern-based gui testing of android and web apps: State-of-the-art, taxonomy, challenges and future directions. In: **2023 25th International Multitopic Conference (INMIC)**. [S.l.: s.n.], 2023. p. 1–7.

LEOTTA, M. et al. Chapter five - approaches and tools for automated end-to-end web testing. In: MEMON, A. (Ed.). **Advances in Computers**. Elsevier, 2016. v. 101, p. 193–237. Disponível em: https://www.sciencedirect.com/science/article/pii/S0065245815000686.

LO, J. N.; LEE, S.-J. An empirical study on web gui load testing and the hybrid with http requests. In: **2023 IEEE International Conference on e-Business Engineering (ICEBE)**. [S.l.: s.n.], 2023. p. 217–224.

MA, S.-P. et al. Semi-automated behavior-driven testing for the web front-ends. **Proceedings - 2023 IEEE International Conference on e-Business Engineering, ICEBE 2023**, p. 225 – 230, 2023.

MACCHI, F. et al. Image-based approaches for automating gui testing of interactive web-based applications. In: **2021 28th Conference of Open Innovations Association (FRUCT)**. [S.l.: s.n.], 2021. p. 278–285.

MAJEED, B. et al. Comparative study of open source automation testing tools: Selenium, katalon studio test project. In: **2021 International Conference on Innovative Computing (ICIC)**. [S.l.: s.n.], 2021. p. 1–6.

MALLICK, S. R. et al. An investigation into the efficacy of ranorex software test automation tool. In: **2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)**. [S.l.: s.n.], 2023. p. 1–5.

MATTIELLO, G. R.; ENDO, A. T. Model-based testing leveraged for automated web tests. **Software Quality Journal**, v. 30, n. 3, p. 621 – 649, 2022. Cited by: 1. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85120068446&doi=10.1007%2fs11219-021-09575-w&partnerID=40&md5=b5694b47577ca52b73c2982a79f826eb.

MINGSONG, C.; XIAOKANG, Q.; XUANDONG, L. Automatic test case generation for uml activity diagrams. In: **Proceedings of the 2006 international workshop on Automation of software test**. [S.l.: s.n.], 2006. p. 2–8.

MOURA, T. S. D. et al. Cytestion: Automated gui testing for web applications. In: **XXXVII Brazilian Symposium on Software Engineering**. [s.n.], 2023. p. 388 – 397. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85174548293&doi=10.1145%2f3613372.3613408&partnerID=40&md5=1a3e81991049220f1f31ac2e517fabe3.

MYERS, G.; SANDLER, C.; BADGETT, T. **The Art of Software Testing**. Wiley, 2011. (ITPro collection). ISBN 9781118133156. Disponível em: https://books.google.com.br/books?id=GjyEFPkMCwcC.

NASS, M. et al. Robust web element identification for evolving applications by considering visual overlaps. In: **2023 IEEE Conference on Software Testing, Verification and Validation (ICST)**. [S.l.: s.n.], 2023. p. 258–268.

NASS, M. et al. Similarity-based web element localization for robust test automation. **ACM Transactions on Software Engineering and Methodology**, v. 32, n. 3, 2023. Cited by: 6; All Open Access, Bronze Open Access. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85159145393&doi=10.1145%2f3571855&partnerID=40&md5=3e20c6f16e9c69dfbea6e814f5ff11d4.

NEETHIDEVAN, V.; CHANDRASEKARAN, G. Web automation using selenium webdriver python. **International Journal of Recent Technology and Engineering (IJRTE)**, Blue Eyes Intelligence Engineering, v. 7, n. 6S, March 2019. ISSN 2277-3878. Retrieval Number: F03700376S19/19©BEIESP & Sciences Publication.

NETO, A.; CLAUDIO, D. Introdução a teste de software. **Engenharia de Software Magazine**, v. 1, p. 22, 2007.

NGUYEN, B.-A. et al. Identification and validation of web themes: A dom-structure matching approach. **Journal of Information Science & Engineering**, v. 38, n. 2, 2022.

OLIANAS, D. et al. Stile: a tool for parallel execution of e2e web test scripts. In: **2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)**. [S.l.: s.n.], 2021. p. 460–465.

OTHMAN, R.; ZEIN, S. Test case auto-generation for web applications: A model-based approach. In: **2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)**. [S.l.: s.n.], 2022. p. 18–25.

OZCAN, M. An industrial study on applications of combinatorial testing in modern web development. In: **2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)**. [S.l.: s.n.], 2019. p. 210–213.

PADHY, P.; NAYAK, P.; VAIDYA, S. Web performance engineering-finding best data structure for automatically collected html locators. **ICCSC 2023 - Proceedings of the 2nd International Conference on Computational Systems and Communication**, 2023.

PEI, Y. et al. An empirical study of async wait flakiness in front-end testing. In: **BENEVOL**. [S.l.: s.n.], 2022.

PINHEIRO, P. **Tipos de frameworks de automação de testes**. [S.l.], 2023. Disponível em: https://www.linkedin.com/pulse/tipos-de-frameworks-automa%C3%A7%C3%A3o-testes-paulo-pinheiro.

PRESLER-MARSHALL, K. et al. Wait, wait. no, tell me. analyzing selenium configuration effects on test flakiness. In: **2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)**. [S.l.: s.n.], 2019. p. 7–13.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. [S.l.]: Palgrave macmillan, 2016.

QUENTAL, N. C. et al. Automating gui response time measurements in mobile and web applications. In: **2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)**. [S.l.: s.n.], 2019. p. 35–41.

RAVELO-MéNDEZ, W.; ESCOBAR-VELáSQUEZ, C.; LINARES-VáSQUEZ, M. Kraken 2.0: A platform-agnostic and cross-device interaction testing tool. **Science of Computer Programming**, v. 225, 2023. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85141924461&doi=10.1016%2fj.scico.2022.102897&partnerID=40&md5=f64c07013499458249d84c9787b97a6f.

ROCHA, A. R. C. da. **Qualidade de software: teoria e prática**. [S.l.]: Prentice Hall, 2001.

ROLLWAGEN, A. F. et al. Comparativo entre ferramentas de automação de testes de software para sistemas web. **Salão do Conhecimento – Unijuí**, Rio Grande do Sul, Brasil, outubro 2020. Disponível em: https://publicacoeseventos.unijui.edu.br/index.php/salaoconhecimento/article/view/18489/17223.

SHAO, F. Webevo: Taming web application evolution via semantic structure change detection. In: **2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)**. [S.l.: s.n.], 2021. p. 141–143.

SILVA, G. Tipos de testes: quais os principais e por que utilizá-los? **Alura**, 2021. Disponível em: https://www.alura.com.br/artigos/tipos-de-testes-principais-por-que-utiliza-los.

SOCIETY, R. **Machine learning: the power and promise of computers that learn by example**. 2017. Disponível em: https://royalsociety.org/~/media/policy/projects/machine-learning/publications/machine-learning-report.pdf.

SOMMERVILLE, I. **Software Engineering**. [S.l.]: Pearson, 2015.

TONELLA, P.; RICCA, F.; MARCHETTO, A. Chapter 1 - recent advances in web testing. In: MEMON, A. (Ed.). **Advances in Computers**. Elsevier, 2014, (Advances in Computers, v. 93). p. 1–51. Disponível em: https://www.sciencedirect.com/science/article/pii/B9780128001622000014.

VASCONCELOS, A. M. L. d. et al. **Introdução à Engenharia de Software e à Qualidade de Software**. Curso de pós-graduação "lato sensu" (especialização) à distância – melhoria de processo de software. Lavras: UFLA/FAEPE, 2006. 157 p.

WALSH, T. A.; KAPFHAMMER, G. M.; MCMINN, P. Automatically identifying potential regressions in the layout of responsive web pages. **Software Testing Verification and Reliability**, v. 30, n. 6, 2020. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088874856&doi=10.1002%2fstvr.1748&partnerID=40&md5=4970d2e3e3f8d58bfe7ce3fd0d360072.

WANG, J.; WU, J. Research on automation testing technology based on image recognition. In: **2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)**. [S.l.: s.n.], 2019. p. 40–42.

YOUSAF, N. et al. Automated model-based test case generation for web user interfaces (wui) from interaction flow modeling language (ifml) models. **IEEE Access**, v. 7, p. 67331–67354, 2019.

ZHENG, Y.; DONG, W.; CHENG, H. Zero wfuzzer: Target-oriented fuzzing for web interface of embedded devices. In: **2021 IEEE 4th International Conference on Computer and Communication Engineering Technology (CCET)**. [S.l.: s.n.], 2021. p. 194–198.

ZHENG, Y. et al. Automatic web testing using curiosity-driven reinforcement learning. In: **2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)**. [S.l.: s.n.], 2021. p. 423–435.

ZIMMERMANN, D.; KOZIOLEK, A. Gui-based software testing: An automated approach using gpt-4 and selenium webdriver. In: **38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW),**. [s.n.], 2023. p. 171 – 174. Cited by: 1. Disponível em: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85178510464&doi=10.1109%2fASEW60602.2023.00028&partnerID=40&md5=5905b7e1c08650949819f9e27ae8cd39.