FUNDAÇÃO HERMÍNIO OMETTO

TRABALHO DE CONCLUSÃO DE CURSO (TCC)

Sistemas de Informação / **2025**

AUTOMAÇÃO DE TESTES PARA DISPOSITIVOS MÓVEIS

BARRETA, Caio Eduardo da Silva[1]
PAULA, Icaro Vinícius de[2]
PERUCCI, Camilo Cesar[3]

Centro Universitário Hermínio Ometto – FHO, Araras – SP, Brasil

Resumo

A crescente diversidade de dispositivos Android e a pressão por entregas rápidas e com alta qualidade tornam o processo de testes uma etapa essencial no desenvolvimento de software móvel. Este trabalho tem como foco a análise comparativa entre testes manuais e testes automatizados, utilizando dados reais de execução em um ambiente corporativo. A automação será analisada sob duas abordagens: com o uso de um framework proprietário de desenvolvimento interno, aqui denominado FORCEZ, e com a aplicação da ferramenta open source Appium. A motivação para este estudo surgiu da necessidade de avaliar, de forma objetiva, os ganhos proporcionados pela automação frente à abordagem manual. O objetivo principal é evidenciar o impacto da automação na produtividade e na confiabilidade dos testes em aplicações Android, além de verificar a viabilidade técnica de soluções de código aberto no contexto corporativo. A metodologia adotada é caracterizada como um trabalho aplicado, realizado a partir da coleta e análise de dados reais de testes já executados pela equipe de QA da empresa parceira. Esses dados foram organizados e comparados com base em métricas previamente definidas, relacionadas a tempo de execução, quantidade de erros identificados e carga de trabalho envolvida. Espera-se, com isso, demonstrar uma redução significativa no tempo de validação e um aumento na eficiência da detecção de falhas, oferecendo dados concretos que apoiem decisões sobre a adoção ou ampliação da automação de testes em projetos Android.

Palavras-chave: Android, Produtividade, Qualidade de Software, Frameworks.

[1] FHO|UNIARARAS. Aluno do Curso de Sistemas de Informação, Caio Eduardo da Silva Barreta, caio.eduardo.03@alunos.fho.edu.br
[2] FHO|UNIARARAS. Aluno do Curso de Sistemas de Informação, Icaro Vinicius de Paula, icaro.vpaula@alunos.fho.edu.br
[3] FHO|UNIARARAS. Professor do Curso de Sistemas de Informação, Camilo César Perucci, camiloperucci@fho.edu.br

1 Introdução

1.1 Contextualização

Num cenário em que dispositivos móveis Android variam em hardware (tamanhos de tela, processadores, memória) e *software* (múltiplas versões de sistema e customizações de fabricantes), garantir qualidade e estabilidade tornou-se um desafio central. Aqui, entendemos qualidade como o atendimento correto aos requisitos e à experiência do usuário, e estabilidade como a capacidade do aplicativo de manter suas funcionalidades sem falhas críticas (*crashes*, bloqueios ou consumo excessivo de recursos), mesmo após atualizações de código ou mudanças de plataforma.
Para assegurar esses atributos, aplicam-se diferentes tipos de teste: testes unitários (validação de métodos isolados), testes de integração (verificação das interações entre módulos), testes funcionais ou de caixa-preta (confirmação do comportamento conforme especificado), testes de compatibilidade (execução em múltiplos dispositivos e versões) e testes de desempenho (avaliação sob carga). A automação de testes se torna indispensável nesse contexto, pois proporciona escalabilidade (execução massiva e repetitiva sem fadiga humana), consistência (mesma sequência de passos em cada execução) e rapidez (feedback imediato ao time de desenvolvimento), reduzindo custos de retrabalho e aumentando a confiabilidade do produto.

1.2 Tema de Pesquisa

O estudo investiga a eficácia da automação de testes no desenvolvimento Android, comparando-a com testes manuais. O foco é demonstrar as vantagens da automação em termos de economia de tempo, precisão na detecção de falhas e impacto na produtividade da equipe de desenvolvimento.

1.3 Motivações e Justificativas

O rápido crescimento do ecossistema Android e a pressão por ciclos de entrega cada vez mais curtos e confiáveis motivaram este estudo. Na empresa parceira, embora já se utilize um framework proprietário de automação (referido neste trabalho como FORCEZ), a equipe de QA ainda dedicava grande parte de sua carga horária a testes manuais repetitivos, além de manter e atualizar os scripts de automação. Isso gerava gargalos no cronograma de releases, limitava a cobertura de cenários críticos e prolongava o tempo necessário para identificar falhas. Com base nesse contexto, o presente trabalho propõe quantificar, de forma objetiva, quanto tempo as automações (FORCEZ e uma solução open source, Appium) conseguem poupar em relação ao teste manual e se essas ferramentas capturam um número maior de defeitos. Os resultados oferecerão subsídios para orientar decisões sobre investimento em automação e otimização dos processos de QA em desenvolvimento móvel.

1.4 Objetivos

1.4.1 Gerais

Avaliar o impacto da automação de testes, através dos seus resultados em contraste com os resultados de testes manuais, comparando eficiência e cobertura, com ênfase nos benefícios da automação.

1.4.2 Específicos

- Identificar e selecionar casos de teste manuais e automatizados que representem funcionalidades críticas do sistema.
- Definir e aplicar métricas comparativas, como tempo de execução, quantidade de falhas detectadas e esforço humano envolvido.
- Analisar os resultados dos testes automatizados e manuais, considerando critérios de sucesso e falha previamente estabelecidos.
- Avaliar a eficiência da automação em diferentes dispositivos e cenários.
- Consolidar os dados obtidos em relatórios que evidenciem os ganhos em produtividade, cobertura e qualidade dos testes.

2 Revisão Bibliográfica

2.1 Fundamentação Teórica e Técnica

Evolução dos Dispositivos Móveis e Sistemas Operacionais

Os dispositivos móveis passaram por uma evolução significativa desde sua criação, tornando-se ferramentas essenciais no cotidiano das pessoas. Inicialmente, os celulares eram utilizados apenas para chamadas e mensagens de texto, mas, com o avanço da tecnologia, passaram a incorporar funcionalidades avançadas, como acesso à internet, execução de aplicativos e integração com serviços em nuvem. Esse crescimento foi impulsionado pelo avanço dos sistemas operacionais móveis, que possibilitaram a criação de um ecossistema dinâmico e inovador para o desenvolvimento de aplicações (Simas, 2019).

Atualmente, dois sistemas operacionais dominam o mercado de dispositivos móveis: Android e iOS. O Android, desenvolvido pelo *Google* e lançado em 2008, tornou-se líder no setor devido à sua flexibilidade, código aberto e ampla compatibilidade com diferentes hardwares. Já o iOS, lançado pela Apple em 2007, se destaca pela sua estabilidade, segurança e controle rigoroso sobre os aplicativos disponibilizados na App Store

(Tanembaum, 2008). A popularização desses sistemas trouxe novos desafios para o desenvolvimento de *software*, especialmente no que diz respeito à qualidade e segurança das aplicações.

Importância do Teste de Software

Os testes de *software* desempenham um papel essencial na garantia da qualidade e estabilidade dos sistemas, assegurando que eles atendam aos requisitos funcionais e operacionais esperados. Esse processo evita que falhas e inconsistências cheguem ao usuário final, reduzindo riscos e garantindo uma experiência confiável (Soares et al., 2022).

Segundo Sommerville (2011, p. 144), o processo de teste tem dois objetivos principais:

- Verificação de conformidade: certificar que o *software* atenda às especificações definidas pelo desenvolvedor e pelo cliente. Para sistemas personalizados, são aplicados testes específicos para cada funcionalidade, enquanto *softwares* genéricos passam por validações mais amplas.

- Identificação e correção de falhas: detectar erros de processamento, interações indevidas com outros sistemas e falhas que possam comprometer a integridade dos dados, garantindo que o sistema funcione conforme esperado.

Dessa forma, os testes não apenas garantem a qualidade do produto final, mas também contribuem para a otimização do processo de desenvolvimento, evitando retrabalho e reduzindo custos associados a falhas encontradas em estágios mais avançados do ciclo de vida do *software*.

Principais Tipos de Teste

Os testes de *software* envolvem diferentes abordagens para validar aspectos fundamentais, como desempenho, segurança e usabilidade. A seguir, são apresentados alguns dos principais tipos de testes e suas características.

Teste Estrutural (Caixa-Branca)

O teste estrutural, também chamado de teste de caixa-branca, analisa diretamente o código-fonte do *software*. Ele avalia se todas as condições, laços e estruturas lógicas funcionam conforme esperado. Dependendo da complexidade e da tecnologia utilizada, podem ser aplicados critérios variados para validar esses aspectos. O testador tem acesso ao código e pode criar cenários que garantam a cobertura de diversas possibilidades, verificando todas as ramificações da lógica implementada (Pressman, 2005).

Teste Funcional (Caixa-Preta)

O teste funcional, conhecido como teste de caixa-preta, foca na validação das funcionalidades sem considerar a estrutura interna do código. Nessa abordagem, são fornecidos dados de entrada, e os resultados obtidos são comparados com os resultados esperados. O teste é bem-sucedido se a saída for compatível com a expectativa. Esse método pode ser aplicado em diferentes níveis, desde funções e métodos individuais até sistemas completos (Pressman, 2005).

Teste de Desempenho

Esse tipo de teste verifica como o *software* se comporta sob condições de alta carga e uso intenso. Ele avalia, por exemplo, a capacidade do sistema de suportar múltiplos usuários simultâneos ou medir o tempo de resposta da aplicação em um ambiente web. O objetivo é identificar gargalos que possam comprometer a experiência do usuário e garantir que o *software* atenda a padrões aceitáveis de desempenho (Myers, 2012, p.126).

Teste de Compatibilidade

Dado o grande número de dispositivos, sistemas operacionais e navegadores disponíveis, o teste de compatibilidade tem a função de verificar se o *software* funciona corretamente em diferentes ambientes. Esse tipo de teste avalia a adaptação do sistema a diversas configurações e garante que ele se comporte conforme esperado em diferentes plataformas (Myers, 2012).

Teste de Aceitação

Esse teste é a etapa final antes do lançamento do *software* para os usuários. Ele tem o propósito de verificar se todas as funcionalidades foram implementadas corretamente e se o sistema está estável. Além disso, avalia se o *software* segue as regras de negócio estabelecidas nos requisitos do projeto, garantindo que ele esteja pronto para ser utilizado no ambiente real (Gaidargi, 2021).

Teste de Segurança

A segurança da informação é uma preocupação crescente, especialmente com o aumento da regulamentação sobre proteção de dados, como a Lei Geral de Proteção de Dados (LGPD). O teste de segurança avalia os mecanismos do *software* para proteger informações sensíveis, garantindo que ele esteja preparado para prevenir acessos não autorizados e vulnerabilidades. Esses testes podem abranger desde a integridade do armazenamento de dados até a proteção contra ataques externos (Maxim & Pressman, 2016).

Teste de Usabilidade

O teste de usabilidade avalia a experiência do usuário ao interagir com o *software*, simulando cenários reais de uso. Segundo Rubin (1994), esse teste mede tanto a eficiência do sistema quanto a satisfação do usuário. Nielsen (1993) destaca que a usabilidade deve ser analisada com base em diferentes perfis de usuários e suas necessidades específicas.

Mili e Tchier (2015) sugerem cinco aspectos essenciais para avaliar a usabilidade de um *software*:

- Facilidade de Uso: mede o quão intuitivo e acessível o sistema é para o usuário.
- Facilidade de Aprendizado: analisa o tempo e o esforço necessários para um usuário compreender e operar o *software*.
- Customização: avalia o nível de personalização permitido pelo sistema para atender às preferências do usuário.
- Calibrabilidade: verifica se o usuário pode ajustar parâmetros operacionais do *software*.
- Interoperabilidade: avalia a compatibilidade do *software* com outros sistemas e aplicações.

Uma das vantagens desse teste é a detecção de problemas na interface e na experiência do usuário que dificilmente seriam identificados por testes automatizados.

Principais funcionalidades a serem testadas

Com base na análise dos repositórios oficiais do ecossistema Android (documentações do Google, da Samsung, e da Motorola) é possível entender que as funcionalidades cruciais e imprescindíveis desdobram-se em usabilidade, segurança, personalização, compatibilidade e atualizações.
As atualizações de segurança são fundamentais para manter o aparelho protegido contra brechas perigosas, como a CVE-2022-20210, que podia travar o dispositivo explorando o *firmware* dos chips, ou falhas em bibliotecas de mídia que permitiam invasões remotas. Cada atualização corrige essas falhas antes que um invasor possa aproveitá-las, garantindo que o sistema continue estável e seguro. Para afirmar que tal funcionalidade opera como o esperado, é preciso simular ataques de negação de serviço logo após aplicá-los, conferindo se o aparelho resiste a tentativas de sobrecarga, e também verificar se o download e instalação das atualizações ocorrem somente por conexões *Wi-Fi*, evitando o gasto desnecessário de dados móveis do usuário.
Personalizar a interface do celular é importante para aumentar a produtividade e deixar o uso do aparelho de acordo com as preferências do usuário. Na *Play Store*, existem vários *launchers* e *widgets* que permitem organizar atalhos, informações e funções conforme desejado, facilitando o acesso no dia a dia. A Samsung, por exemplo, vai além e oferece temas prontos e a *One UI*, que permite mudar cores, ícones e o *layout* da tela para combinar com o estilo de cada usuário. Mas, para que toda essa personalização funcione bem, é necessário verificar se os *widgets* continuam funcionando em diferentes tamanhos de tela e versões do Android.
É importante que o Android funcione bem em diferentes marcas de celular, como Samsung e Motorola, mesmo que cada uma tenha seu próprio tipo de hardware e personalização. Para ajudar nisso, existe o Samsung *Remote Test Lab*, uma ferramenta que permite testar aplicativos em vários modelos *Galaxy* sem precisar ter esses aparelhos

em mãos. Isso ajuda a verificar se a interface e as funções do app funcionam corretamente em cada modelo. Ao mesmo tempo, a Motorola oferece o *app* "*Device Help*", que serve para testar partes importantes do celular, como sensores de movimento, GPS e o estado da bateria. Assim, dá para descobrir se há problemas de *hardware* ou se o *app* está gastando muita energia. Além disso, o Android *Studio* — que é a ferramenta oficial para desenvolver *apps* Android — permite testar os aplicativos em emuladores (simuladores de celulares) e também em aparelhos reais. Com isso, dá para analisar se o *app* funciona bem em diferentes tamanhos de tela, tipos de processador e versões do Android, garantindo bom desempenho e acessibilidade.
Manter os aplicativos atualizados automaticamente é importante para a segurança do celular. *Apps* desatualizados podem ter falhas que facilitam ataques. O Android, por meio da *Play Store*, cuida dessas atualizações de forma automática e normalmente só usa o *Wi-Fi*, para não gastar o plano de internet do usuário. Para garantir que isso funcione corretamente, é necessário simular situações de internet fraca ou ruim. Assim, é possível verificar se o sistema realmente espera por uma conexão *Wi-Fi* antes de atualizar, evitando custos inesperados.

2.2 Trabalhos Relacionados

Diversos estudos acadêmicos abordam a importância da automação de testes no contexto do desenvolvimento de *software*, especialmente para aplicações móveis, com foco em desempenho, confiabilidade e qualidade dos produtos entregues.

O trabalho de Stabel (2019) realizou uma análise prática da aplicação de ferramentas de automação em dispositivos reais, comparando seus resultados com testes manuais. A autora concluiu que, apesar das diferenças entre as ferramentas analisadas (como Selenium e Katalon), a automação contribui para a linearidade e reprodutibilidade dos testes. Mesmo em uma aplicação de pequena escala, já foi possível identificar ganhos de desempenho e economia de tempo, o que evidencia seu potencial em ambientes maiores e mais complexos.

Cunha (2021) propôs uma análise comparativa entre ferramentas de automação voltadas para testes funcionais em aplicativos móveis. A pesquisa utilizou o Appium, XCTest e Espresso, avaliando suas configurações, limitações e desempenho. Apesar de apontar que o Appium possui dependências complexas, o estudo demonstrou que a ferramenta foi capaz de executar todas as interações planejadas, apresentando resultados mais completos quando comparada às demais. A pesquisa também destaca a importância de preparar adequadamente a infraestrutura para execução de testes em larga escala e sugere a adoção de pipelines de integração contínua para aprimorar o processo.

Oliveira et al. (2024) analisaram as principais ferramentas de automação para dispositivos móveis, destacando as vantagens do Robot Framework e do Appium. O estudo escolheu essas ferramentas com base na simplicidade de uso, extensibilidade e suporte a múltiplas plataformas. Segundo os autores, a combinação dessas soluções proporciona uma base robusta para garantir a qualidade de aplicativos móveis, principalmente em ambientes com alta rotatividade de mudanças.

Freitas (2022), por sua vez, reforça a relevância da automação no desenvolvimento de aplicativos Android, diante da necessidade crescente de qualidade frente ao aumento do uso de dispositivos móveis. O autor apresenta o Robot Framework como alternativa viável para a automação de testes, discutindo sua estrutura, aplicação prática e benefícios em relação à confiabilidade dos testes. O trabalho também realiza uma revisão sistemática sobre abordagens de automação na plataforma Android, contribuindo com uma visão mais ampla sobre o tema.

3 Metodologia

A metodologia deste estudo baseia-se na coleta, análise e comparação de dados provenientes da execução de testes manuais e automatizados em aparelhos Android. Serão priorizados os testes que abordam funcionalidades consideradas críticas e de alto impacto para o sistema, permitindo uma avaliação mais precisa da eficácia da automação em cenários relevantes. Posteriormente, os dados serão analisados, filtrados e exibidos em formato de gráficos, facilitando a visualização e o entendimento do impacto na produtividade.

3.1 Coleta de Dados dos Casos de Teste

Serão coletados a partir de relatórios fornecidos pelo time de QA, para cada execução (manual, FORCEZ e Appium), as seguintes métricas:

- Tempos de execução: duração total de cada caso de teste.
- Quantidade de defeitos encontrados: avaliação da eficácia na identificação de falhas em cada abordagem.

3.2 Automações utilizadas

Ferramentas de Automação

Com o aumento da complexidade dos *softwares*, surgiram diversas ferramentas de automação voltadas à otimização dos testes. Essas ferramentas variam conforme o tipo de aplicação e os requisitos específicos de cada projeto. Entre as mais utilizadas estão Selenium, Cypress, Playwright, WebDriver, Robot Framework e Appium, que permitem a automação de testes funcionais, de interface e de integração, contribuindo significativamente para a validação eficiente de aplicações.

3.2.1 APPIUM

Framework de código aberto, voltado à automação de testes de interfaces gráficas (UI) em diversas plataformas de aplicativos, o Appium permite a criação de *scripts* de teste em qualquer linguagem de programação que ofereça um cliente WebDriver (Java, Python, Ruby, C#, JavaScript etc.), além de ser amplamente adotado por organizações que precisam de soluções de automação de testes móveis confiáveis e escaláveis.

3.2.2 FORCEZ (Framework Proprietário de Automação)

Para efeito deste trabalho, a ferramenta de automação utilizada internamente pela empresa será referida como FORCEZ, a fim de preservar informações confidenciais sobre sua estrutura e funcionamento. Trata-se de um framework proprietário desenvolvido pela própria equipe técnica da organização, com o objetivo de automatizar testes funcionais e facilitar a execução e monitoramento de casos de teste em dispositivos Android. Sua execução ocorre em um servidor dedicado, configurado com Ubuntu 20.04, Python 3.10 e Java 17, atendendo aos requisitos de ambientes de integração contínua (CI/CD).

O FORCEZ é mantido pela própria equipe de QA da empresa e conta com um conjunto de bibliotecas e plugins especializados que possibilitam funcionalidades como geração de relatórios, análise de memória, captura de vídeo, envio de e-mails, integração com repositórios e execução paralela de casos de teste. As dependências utilizadas são, em sua maioria, bibliotecas Python de uso geral, como *pandas, pytest, requests, openpyxl, matplotlib* e outras voltadas à automação e análise de dados.

Por se tratar de uma solução interna, informações técnicas detalhadas — como arquitetura, estrutura de código, componentes internos e integrações específicas — não serão abordadas neste trabalho, em respeito a cláusulas de confidencialidade firmadas com a empresa parceira.

Comparativo entre Appium e FORCEZ

Embora o foco principal deste trabalho esteja na comparação entre testes manuais e testes automatizados, optou-se por incluir duas abordagens distintas de automação para enriquecer a análise: o uso do FORCEZ, um framework proprietário adotado internamente pela empresa, e o Appium, uma ferramenta open source amplamente utilizada no mercado.

A escolha do Appium se justifica por sua compatibilidade com dispositivos Android, sua flexibilidade de integração e, principalmente, pelo fato de ser uma solução de código aberto. Isso permite sua adoção sem restrições de licenciamento ou acesso a código-fonte, o que facilita a replicação dos experimentos em outros contextos. A utilização do Appium neste estudo tem como objetivo demonstrar que, mesmo sem depender de soluções proprietárias, é possível obter ganhos relevantes em eficiência, desde que a automação seja bem implementada.

A comparação entre Appium e FORCEZ é apresentada como um complemento à análise central do trabalho, que visa mensurar os benefícios da automação frente ao processo manual. Para isso, são observadas métricas como tempo de execução, estabilidade dos testes, esforço de manutenção e taxa de falhas identificadas.

No caso do FORCEZ, por ser uma ferramenta interna e de código fechado, os dados foram obtidos por meio de resultados práticos fornecidos pela empresa, observações diretas e relatos de profissionais envolvidos em sua aplicação. A intenção não é avaliar o FORCEZ isoladamente, mas sim utilizá-lo como referência real de uma automação corporativa em uso, contrastando seus resultados tanto com os testes manuais quanto com o Appium.

Com isso, o trabalho busca oferecer uma visão prática sobre os impactos da automação em ambientes reais, reforçando que, mesmo com recursos open source, é possível alcançar melhorias significativas na qualidade, produtividade e confiabilidade do processo de testes.

3.3 Quantificação e Análise Comparativa

A análise quantitativa dos dados buscará mensurar a eficiência das automações realizadas através do Appium e do FORCEZ, em relação aos testes manuais, com ênfase nos seguintes pontos:

- **Economia de tempo:** comparação do tempo total de execução dos testes manuais e automatizados, destacando a redução de horas de trabalho com a automação.
    - Tempo total por suíte: soma dos tempos de todos os casos em cada abordagem.
    - Estatísticas de tempo individual: média, mediana e desvio-padrão dos tempos por caso de teste.
    - Redução percentual: cálculo de % de ganho de velocidade (por ex., “Appium foi X% mais rápido que manual”).

- Eficiência na detecção de falhas: comparação da profundidade e precisão na identificação de falhas, evidenciando a eficácia da automação.
    - Total de defeitos encontrados: número absoluto de falhas reportadas em cada abordagem.
    - Média de defeitos por caso: total de defeitos ÷ número de casos executados.
    - Classificação de severidade: proporção de defeitos críticos versus leves detectados por cada método (para ver qual abordagem acha mais cenários críticos).

3.4 Documentação dos Resultados

Os resultados serão documentados para estabelecer uma linha de base de eficiência, permitindo a comparação objetiva entre os testes manuais e automatizados. A documentação incluirá as métricas de sucesso e falha para cada caso de teste, garantindo uma análise precisa, bem como os benefícios da automação, com ênfase na economia de tempo e aumento da produtividade.

3.5 Avaliação do Impacto na Qualidade e Produtividade

A análise dos dados coletados permitirá avaliar os impactos da automação na qualidade e produtividade do desenvolvimento Android. Os principais pontos de avaliação serão:

- Taxa de falhas: comparação da quantidade de falhas detectadas, com foco na maior eficiência dos testes automatizados, especialmente em cenários complexos e repetitivos.
- Economia de tempo: quantificação do tempo poupado com a automação, destacando a redução do tempo de execução e os ganhos de eficiência.

Esses resultados preliminares fornecerão uma base para conclusões mais detalhadas, demonstrando o potencial da automação para melhorar a eficiência do processo de testes e a produtividade da equipe de desenvolvimento.

4 Resultados

Com base nos dados coletados até o momento, observou-se uma diferença relevante no tempo de execução entre os testes manuais e os testes automatizados. A automação foi aplicada a um conjunto de casos de teste relacionados à funcionalidade de alteração de imagem de fundo e bloqueio de tela no Android (*wallpaper*). Nessas execuções, foi possível registrar que cada **teste automatizado** levou, em média, cerca de **2 minutos** para ser concluído.

Figura 1 – Testes Automatizados de *Wallpaper*

Fonte: FORCEZ, 2025

Em paralelo, por meio de conversas com a equipe de QA responsável pelos testes manuais, foi relatado que, em um expediente de **8 horas**, são realizados aproximadamente **40 casos de teste**. Esse número considera o tempo necessário para preparar o ambiente, executar os passos de cada caso e registrar os resultados manualmente.

A partir dessas informações preliminares, pode-se estimar que a automação permite executar um volume consideravelmente maior de testes no mesmo intervalo de tempo, o que indica um potencial ganho de produtividade. Contudo, essa estimativa se baseia em um cenário específico e ainda requer validação em outros contextos e funcionalidades para garantir sua generalização.

Outros dados, como taxa de falhas detectadas e esforço de manutenção dos testes automatizados, ainda estão sendo levantados e organizados para análises futuras. Até o

momento, os resultados indicam que a automação tende a reduzir a dependência de fatores humanos e apresenta maior escalabilidade, especialmente em cenários repetitivos e críticos do sistema Android.

6 Considerações Finais

Até o momento, os dados coletados apontam indícios de que a automação de testes pode trazer ganhos relevantes em produtividade e qualidade no desenvolvimento de software Android. As primeiras comparações entre a execução manual e os testes automatizados realizados com o framework proprietário FORCEZ indicam redução de tempo, maior cobertura e menor suscetibilidade a falhas causadas por fatores humanos. No entanto, trata-se de uma análise parcial, baseada em um conjunto específico de testes e observações iniciais.

Atualmente, estamos em fase de organização e estruturação dos dados provenientes dos testes manuais e automatizados com o FORCEZ. Os próximos passos do estudo incluem a execução e análise dos mesmos casos de teste utilizando o Appium, ferramenta open source adotada neste trabalho como referência comparativa adicional. A partir dessa nova etapa, será possível ampliar a análise com base em três abordagens — teste manual, automação com ferramenta proprietária e automação com ferramenta open source — permitindo uma avaliação mais completa dos impactos da automação em diferentes contextos e cenários.

Os resultados finais pretendem oferecer uma visão prática sobre a viabilidade e os benefícios da automação, considerando diferentes tecnologias, níveis de esforço humano e desempenho geral na execução de testes em aplicações Android.

Referências Bibliográficas

ALOTAIBI, Ashwaq A.; QURESHI, Rizwan J.. **Novel Framework for automation testing of mobile applications using Appium**. International Journal of Modern Education and Computer Science, v. 9, n. 2, p. 34-40, 2017. Disponível em: https://www.mecs-press.org/ijmecs/ijmecs-v9-n2/IJMECS-V9-N2-4.pdf. Acesso em: 02 nov. 2024. DOI: 10.5815/ijmecs.2017.02.04.

STABEL, Ana Caroline Torres; PERUCCI, Camilo César. **Análise de uso de testes automatizados para plataformas android**. 2019. 19 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro Universitário Hermínio Ometto – Uniararas, Araras, 2019. Cap. 1.

BAUMGARTNER, Cristiano. **Conheça 5 benefícios dos testes automatizados de software**. Testing Company, 2021. Disponível em: https://testingcompany.com.br/blog/conheca-5-beneficios-dos-testes-automatizadosde-software. Acesso em 08 de novembro de 2024.

CAMPOS, Ana Gabriela de Abreu. **Proposta para implantação de automação de testes de software usando behavior driven development (BDD) - Estudo de caso**. 2024. 186 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Federal de Uberlândia, Uberlândia, 2024. Disponível em: https://repositorio.ufu.br/bitstream/123456789/41987/1/PropostaImplantacaoAutomacao.pdf. Acesso em: 08 nov. 2024.

CEZERINO, A.; NASCIMENTO, F. P. **Utilização da técnica de desenvolvimento orientado por comportamento (BDD) no levantamento de requisitos**. *Revista Interdisciplinar Científica Aplicada*, 10(3), 40-51, 2016.

CROZATTI, Fernando. **Automação dos testes de validação em uma plataforma de comércio eletrônico**. 2021. 73 f. Dissertação (Mestrado) - Curso de Mestrado em Gestão de Redes de Telecomunicações, Pontifícia Universidade Católica de Campinas, Campinas, 2021. Disponível em: https://repositorio.sis.puc-campinas.edu.br/bitstream/handle/123456789/16476/ceatec_ppggrt_me_Fernando_C.pdf?sequence=1&isAllowed=y. Acesso em: 06 nov. 2024.

CUNHA, Levy Marcelo Goulart. **Automação de testes em aplicativos móveis**. 2021. 62 f. Monografia (Especialização) - Curso de Especialização em Engenharia de Software e Inovação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2021. Disponível em: https://lume.ufrgs.br/bitstream/handle/10183/231554/001133309.pdf?sequence=1&isAllowed=y. Acesso em: 08 nov. 2024.

DUVAL, Paul M. ***Continuous Integration: Improving Software Quality and Reducing Risk*. 7a edição**. Boston: Addison-Wesley Professional, 2007.

ERGÖREN, B. **How continuous mobile development can benefit from test automation**. InfoQ, 2024. Disponível em: https://www.infoq.com/news/2024/02/continuous-mobile-development/. Acesso em: 8 nov. 2024.

FÉLIX, Rafael (org.). **Teste de software.** São Paulo: Pearson, 2016. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 01 maio 2024.

FONSECA, Maria Adriana Neto. **Desenvolvimento de testes automatizados para backend**. 2021. 88 f. Tese (Doutorado) - Curso de Mestrado em Engenharia Informática, Universidade Nova de Lisboa, Lisboa, 2021. Disponível em: https://run.unl.pt/bitstream/10362/120492/1/Fonseca_2021.pdf. Acesso em: 06 nov. 2024.

FREITAS, Nayane Araujo Machado de. **Testes automatizados para desenvolvimento mobile em Android.** 2022. Monografia (Graduação em Ciência da Computação) - Universidade Federal de Alagoas, Arapiraca, Alagoas.

GAIDARGI, Juliana. **Tudo sobre teste de aceitação.** Infonova, 2021. Disponível em: https://www.infonova.com.br/tutoriais/tudo-sobre-teste-de-aceitacao/. Acesso em: 14 de abr. de 2023.

KIPAR, Dominik. **Test automation for mobile hybrid applications: Using the example of the BILD app for Android and iOS**. 2014. 89 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software) – Turku University of Applied Sciences, Turku, 2014. Disponível em: https://www.theseus.fi/bitstream/handle/10024/76815/Test%20automation%20for%20mobile%20hybrid%20applications.pdf?sequence=1. Acesso em: 08 nov. 2024.

LIN, Jun-Wei; SALEHNAMADI, Navid; MALEK, Sam. **Test automation in open-source Android apps: a large-scale empirical study**. 2020. Disponível em: https://seal.ics.uci.edu/publications/2020_ASE_Empirical.pdf. Acesso em: 8 nov. 2024.

LOURENÇO, Rony de Sena. **Automação de testes para um sistema de e-commerce**. 2022. 58 f. TCC (Graduação) - Curso de Engenharia de Computação, Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2022. Disponível em: https://repositorio.ufrn.br/bitstream/123456789/50670/1/TCC_Rony.pdf. Acesso em: 08 nov. 2024.

MYERS, G.; SANDLER, C.; BADGETT, T. **The art of Software Testing. 3rd ed**. Hoboken: John Wiley & Sons, 2012.

MASCHIETTO, Luís G.; RODRIGUES, Thiago N.; BIANCO, Clicéres M D.; et al. **Processos de Desenvolvimento de Software.** [Grupo A, 2020. E-book. ISBN 9786556900520.

MAXIM, B.R e PRESSMAN, R. S. **Engenharia de Software – Uma Abordagem Profissional.** 8a. ed. Porto Alegre: AMGH Editora Ltda, 2016.

MILÍ, A.; TCHIER, F. **Software Testing: Concepts and Operations.** Ed. Wiley, 2015.

MYERS, G.; SANDLER, C.; BADGETT, T. **The Art of Software Testing.** 3rd ed. Hoboken: John Wiley & Sons, 2012.

NETO, Arilo. **Introdução a teste de software - Engenharia de Software Magazine**, v. 1, p. 22, 2007.

NIELSEN, J.; LYNGBAEK, U. (1990). **Two field studies of hypermedia usability.** In R. McAleese e C. Green (eds), Hypertext: State of the Art. Oxford: Intellect, 64-72.

OLIVEIRA, Camila Gabriel; GONÇALVES, Luciano Bibiano; PERUCCI, Camilo César. **Automação de testes em dispositivos móveis**. 2024. 24 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro Universitário Hermínio Ometto – Fho, Araras, 2024. Cap. 1.

POLO, Rodrigo Cantú. **Validação e teste de software.** 1. ed. São Paulo: Contentus, 2020. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 01 abr. 2024.

PRESSMAN, Roger S. **Engenharia de Software: Uma Abordagem Profissional.** 7a Edição, McGraw-Hill. 780 p., 2011.

RUBIN, J. (1994). **Handbook of Usability Testing.** New York: John Wiley and Sons.

SÁ, E. C. de; Silva, R. (2016). **Automação de teste para dispositivos móveis e execução dos scripts de teste automatizados na nuvem.** Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade do Sul de Santa Catarina, Florianópolis, Santa Catarina, Brasil.

SEVERINO, A. J. **Metodologia do Trabalho Científico.** São Paulo: Cortez, 2000. 279p.

SIMAS, Victor L.; BORGES, Olimar T.; COUTO, Júlia M C.; et al. **Desenvolvimento para dispositivos móveis - Volume 2**. Editora: Grupo A, 2019. E-book. ISBN 9788595029774.

SOARES, Larissa Fernandes et al. **Aprendendo sobre a significância dos testes de protótipo para a garantia da qualidade na engenharia de produtos tecnológicos. Open Science Research III**, Guarujá,

v. 5, n. 219, p. 2984-3004, 2022. Editora Científica Digital. http://dx.doi.org/10.37885/220308105. Disponível em: https://downloads.editoracientifica.com.br/articles/220308105.pdf. Acesso em: 09 nov. 2024.

SOMMERVILLE, I. **Engenharia de software 9**. ed. São Paulo: Pearson Prentice Hall,2011.

TANENBAUM, A. S. (2008). **Sistemas Operacionais Modernos.** (pp. 1-15). Pearson Prentice Hall.

VONTELL, Aaron Richard. **Bility: Automated accessibility testing for mobile applications**. 2019. 142 f. Dissertação (Mestrado em Engenharia Elétrica e Ciência da Computação) – Massachusetts Institute of Technology, Cambridge, 2019. Disponível em: https://dspace.mit.edu/handle/1721.1/121685. Acesso em: 08 nov. 2024.