FERRAMENTA PARA SUPORTE A AUTOMAÇÃO DE TESTE EM DISPOSITIVOS MÓVEIS

Alisson Schmitz de Medeiros[1], Matheus Leandro Ferreira[2]

Resumo: Com o avanço da tecnologia móvel e a crescente procura por um smartphone, a qualidade de software tem sido cada vez mais exigente para os usuários. A automação de testes na tecnologia móvel, não é algo muito comum no mercado. Nas empresas de desenvolvimento de software, as prioridades atendidas são o custo e o prazo de um projeto. Portanto, o presente trabalho tem como objetivo o desenvolvimento de uma ferramenta para teste em dispositivos móveis possibilitando a melhoria contínua por meio de atividades automatizadas de testes, reduzindo o trabalho repetitivo em tarefas simples. Como resultado, o protótipo conseguiu apresentar ótimo desempenho encontrando problemas com seus scripts em aplicativos, o *Appium* provou ser uma ferramenta poderosa para automatizar testes em dispositivos móveis, possuindo excelentes recursos a serem explorados.

Palavras-chave: Testes de software. Dispositivos móveis. Framework. Qualidade. Automação de testes. Comparação de Imagens.

ABSTRACT: With the advancement of mobile technology and the growing demand for a smartphone, the quality of software has been increasingly demanding for users. Test automation in mobile technology is not very common in the market. In software development companies, the priorities served are the cost and time of a project. Therefore, the present work aims to develop a tool for testing on mobile devices, enabling continuous improvement through automated testing activities, reducing repetitive work in simple tasks. As a result, the prototype was able to present great performance finding problems with its scripts in applications, Appium proved to be a powerful tool to automate tests on mobile devices, having excellent resources to be explored.

[1] Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense(UNESC), Criciúma-SC alissonschmitz2@hotmail.com.
[2] Orientador, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense(UNESC), Criciúma-SC mlf@unesc.net.

Keywords: Software tests. Mobile devices. Framework. Quality. Test automation. Image comparison.

1. INTRODUÇÃO

Atualmente, a tecnologia e o software estão cada vez mais presentes na vida das pessoas e da sociedade. Assim, com a rapidez das evoluções tecnológicas que vem crescendo dia a dia, é perceptível que a qualidade de software passou a ser uma exigência obrigatória para o mercado.

A qualidade é um requisito que deve estar associado a todos os tipos de serviços prestados, principalmente quando se trata do desenvolvimento de um software. Pode-se conceituar qualidade de software como a totalidade das características de uma entidade que lhe confere a capacidade de satisfazer às necessidades explícitas e implícitas (NBR-ISO, 9000:2000). A entidade é o produto, considerado um serviço ou um bem. As necessidades explícitas são fatores relativos à qualidade do processo da construção do produto, percebidos apenas por pessoas que trabalham no seu desenvolvimento. Necessidades Implícitas é quando o cliente relata algum tipo de insatisfação ou dificuldade, uma necessidade abstrata em que não tem certeza do que ele precisa, mas são extremamente importantes para o funcionamento do produto.

Segundo dados do CHAOS Report (2020), 31% dos projetos de software foram entregues dentro do prazo, dentro do orçamento e com funcionalidades completas, 50% foram entregues fora do prazo, fora do orçamento ou com funcionalidades incompletas e 19% foram cancelados, gerando um custo enorme às empresas ao ano em termos de manutenção e redução da produtividade. Um software com defeitos e sem a qualidade necessária, além de gerar gastos com manutenção, compromete a reputação da empresa, reduz a produtividade e ainda resulta na insatisfação dos clientes.

O teste de software é essencial para o desenvolvimento de um produto qualificável no mercado, é a etapa principal para o comprometimento com os requisitos de um sistema. Porém, vale ressaltar, que o processo de testes nem sempre foi priorizado como uma atividade principal no desenvolvimento de uma solução. Em algumas situações, o prazo da entrega do software contava muito, e quando ultrapassado, as empresas desconsideravam a execução das atividades do

teste. Contudo, este comportamento vem se tornando cada vez menos frequente, conforme menciona Sommerville (2011) muitos defeitos passaram a ser encontrados em sistemas já liberados para os clientes, diminuindo a credibilidade do produto no mercado de trabalho, assim como a confiança dos clientes no produto.

No entendimento de Bernardo (2011) o teste de software é uma atividade eficaz para a garantia de qualidade de software, porém a implementação é um grande desafio devido à alta complexidade dos sistemas e às inúmeras dificuldades relacionadas às etapas de desenvolvimento.

Quando se testa o software, o programa é executado várias vezes utilizando dados fictícios a fim de descobrir defeitos antes do uso. Os resultados do teste são verificados à procura de erros, anomalias ou informações sobre os atributos não funcionais do programa (SOMMERVILLE, 2011). As atividades exercidas durante o processo de testes consistem na detecção e correção de erros de software e não conformidades. Este processo busca validar o software diante das especificações garantindo que seus objetivos sejam atendidos, assim como as necessidades dos clientes (PRESSMAN; MAXIM, 2016).

Sendo assim, a automação do teste de software é algo extremamente importante para o mercado, por aumentar a produtividade e atingir um tempo menor que em geral é repetitivo no ambiente de teste. Porém, os testes manuais não podem ser eliminados em hipótese alguma, sendo que os mesmos devem ser focados naquilo que é muito caro de se automatizar (MOLINARI, 2010).

Levando em consideração o que foi abordado, propõe-se por meio deste trabalho empregar os conceitos de engenharia de software na construção de um protótipo de uma ferramenta desktop para criação de testes automatizados para aplicações mobile Android. Busca-se minimizar falhas em aplicativos e mostrar como os casos de testes podem ser uma etapa fundamental para a construção de um produto. Ainda, espera-se que a tecnologia desenvolvida, ofereça um suporte não apenas aos programadores, mas a toda empresa de tecnologia que busca agregar valor ao seu produto utilizando conceitos e técnicas da engenharia de software.

Os objetivos específicos consistem em: pesquisar algoritmos de testes para soluções mobile; diminuir o esforço gasto com testes repetitivos simples em testes manuais em aplicativos móveis; implementar algoritmos por meio de um *framework* de automação de testes; prototipar um aplicativo móvel em *Android* para demonstrar os resultados; implementar métodos de comparações para os resultados

dos testes; desenvolver um software para a criação de casos de testes automatizados; analisar os resultados dos casos de testes criados por meio desta pesquisa.

2. TRABALHOS CORRELATOS

Para o desenvolvimento deste artigo foi necessário pesquisar trabalhos semelhantes sobre o assunto proposto, adquirindo um maior conhecimento em relação ao tema.

O trabalho de conclusão de curso desenvolvido por Marcelo Dehon Batista de Prá no ano de 2012 na Universidade do Extremo Sul Catarinense (UNESC) para obtenção do grau de bacharelado em ciência da computação tem como objetivo desenvolver uma ferramenta para criação e manutenção de scripts de automação de testes funcionais de software, com base nas técnicas Data-Driven e Keyword-Driven (PRÁ, 2012).

A estrutura do software baseia-se sempre na visão de dois usuários distintos, onde primeiramente existe o usuário responsável pela criação e manutenção dos scripts de automação de testes, ou seja, o que detém o conhecimento técnico referente aos processos de automação criando casos com ambas as técnicas. Com uma visão um pouco diferente existe o usuário que irá criar os casos de testes com base em seu conhecimento da lógica de negócio do sistema a ser testado utilizando somente a técnica Data-Driven (PRÁ, 2012).

Nesse TCC apresentado, foi chegado à conclusão que inicialmente até se desenvolver o script, a janela e criar os casos de testes, o tempo de desenvolvimento pode ser maior do que se processo fosse testado de forma manual. O grande ganho da automação de testes está nos testes regressivos, onde após um caso a ser desenvolvido, este pode ser re-executado diversas vezes em várias versões do software a ser testado, onde a partir de uma determinada execução começará a ser mais lucrativa a automação de testes do que os testes manuais. Vale lembrar também que os testes manuais são mais sujeitos a falhas do que os automatizados (PRÁ, 2012).

O artigo desenvolvido por Eduardo Corrêa de Sá e Rodrigo Silva no ano de 2016 para obtenção do título de Bacharel em Sistemas de Informação na Universidade do Sul de Santa Catarina (UNISUL) tem como objetivo apresentar uma alternativa para uma melhoria na qualidade do software desenvolvido em diversos

modelos de *smartphones*, utilizando técnicas de elaboração e execução de scripts de teste automatizados emulando diferentes modelos de smartphones através de serviços na nuvem (SÁ; SILVA, 2016).

Buscando melhorias da qualidade do software desenvolvido, executando em diversos modelos de *smartphones* presentes no mercado, este artigo teve como objetivo apresentar ferramentas e implementar um projeto piloto, com o intuito de testar um aplicativo para dispositivo móvel utilizando serviços de emulação de dispositivos na nuvem, para execução dos scripts de teste automatizados (SÁ; SILVA, 2016).

Utilizaram o *Appium* para implementação dos scripts de teste que foi de grande ajuda para realização do trabalho, tornando possível a comunicação com os dispositivos móveis e a realização de ações conforme os casos apresentados. Juntamente com o Selenium, responsável pela configuração do script de automação de testes, interage por meio dos drivers instanciados, como por exemplo o *AndroidDriver* e *IOSDrive* demonstrou-se simples e eficaz, cumprindo bem todas as suas funções (SÁ; SILVA, 2016).

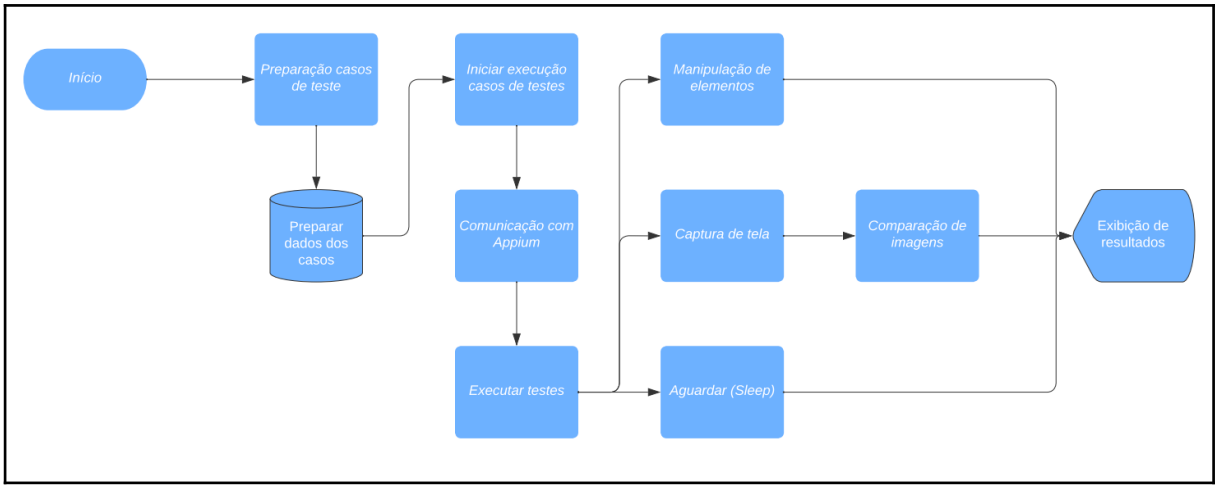
Por fim, os objetivos definidos foram atingidos com êxito. A utilização da automação de teste para dispositivos móveis na nuvem, na rotina de testes e desenvolvimento de uma aplicação, se mostrou uma alternativa viável para auxiliar testadores e interessados, nas tarefas de execuções de testes (SÁ; SILVA, 2016).

3. MATERIAIS E MÉTODOS

Esta pesquisa caracteriza-se por ser aplicada e de base tecnológica, no qual a finalidade é criar uma ferramenta para suporte a automação de testes em dispositivos móveis. Para atingir os objetivos criou-se casos de testes capazes de navegar em elementos de um aplicativo, realizando comparações de imagens e demonstrando seus resultados para o analista de testes.

Para um melhor entendimento da aplicação, pode ser observado na figura 1, o fluxograma completo da ferramenta proposta. Após a preparação dos casos de testes, faz-se necessário a comunicação com o software *Appium*, no qual realiza a instalação do aplicativo configurado, a interação com os elementos no caso de teste e realiza a captura de tela para exibição dos resultados.

Figura 1: Fluxograma completo da ferramenta



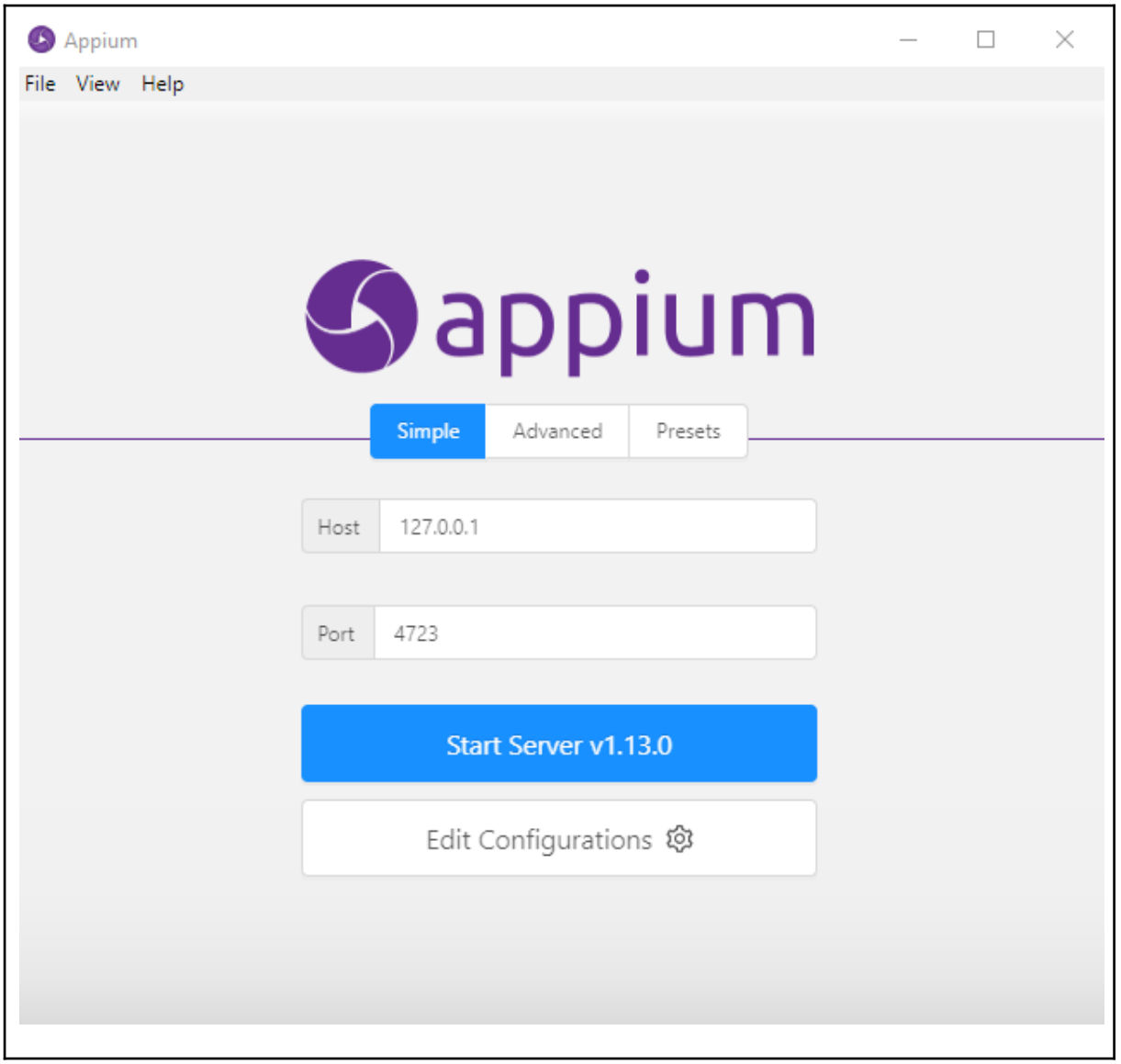
Fonte: Do Autor.

Na preparação e execução dos casos de teste, foram utilizadas ferramentas do software Android Studio, como o *UI Automator Viewer* e o emulador respectivamente. A execução dos casos de teste pode ser realizada por um emulador ou dispositivo físico Android.

3.1 CONFIGURAÇÃO DO APPIUM

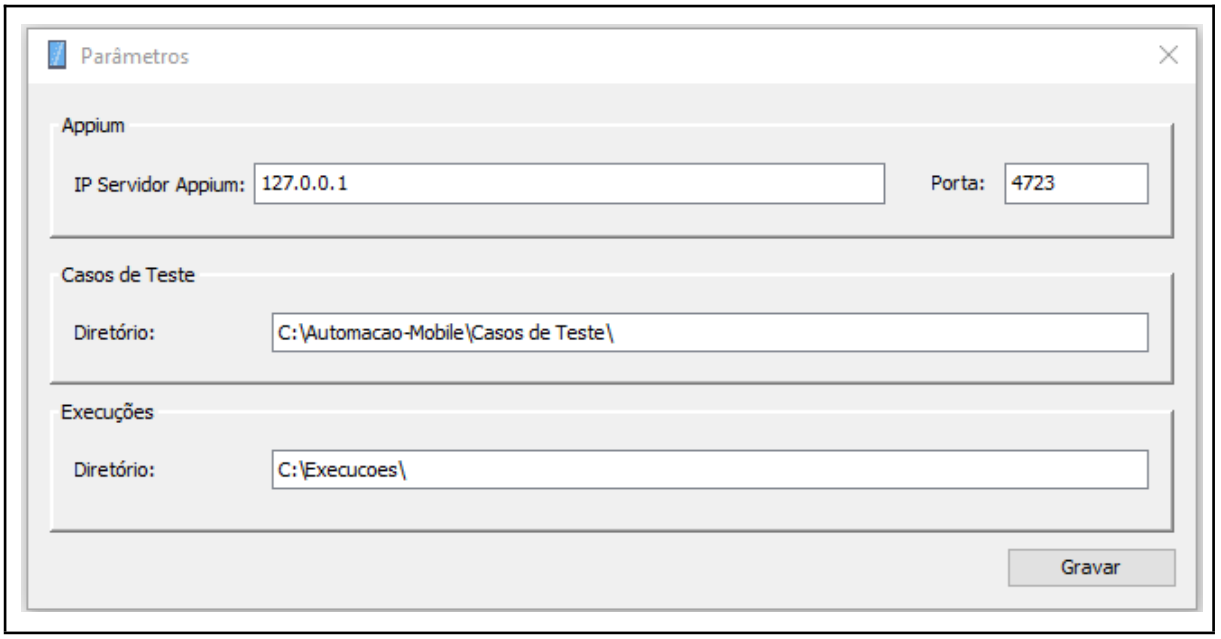
Após a instalação da aplicação Appium Desktop na máquina, deve-se fazer a sua configuração. Conforme exemplo da figura 2, o endereço ip deve ser adicionado no campo *Host* junto com sua respectiva porta. Vale ressaltar que essa pré-definição de parâmetros deve ser a mesma indicada na ferramenta de automação desenvolvida, localizado em parâmetros, quadrante Appium.

Figura 2: Configuração do Appium



Fonte: Do Autor.

Figura 3: Configuração dos parâmetros do protótipo

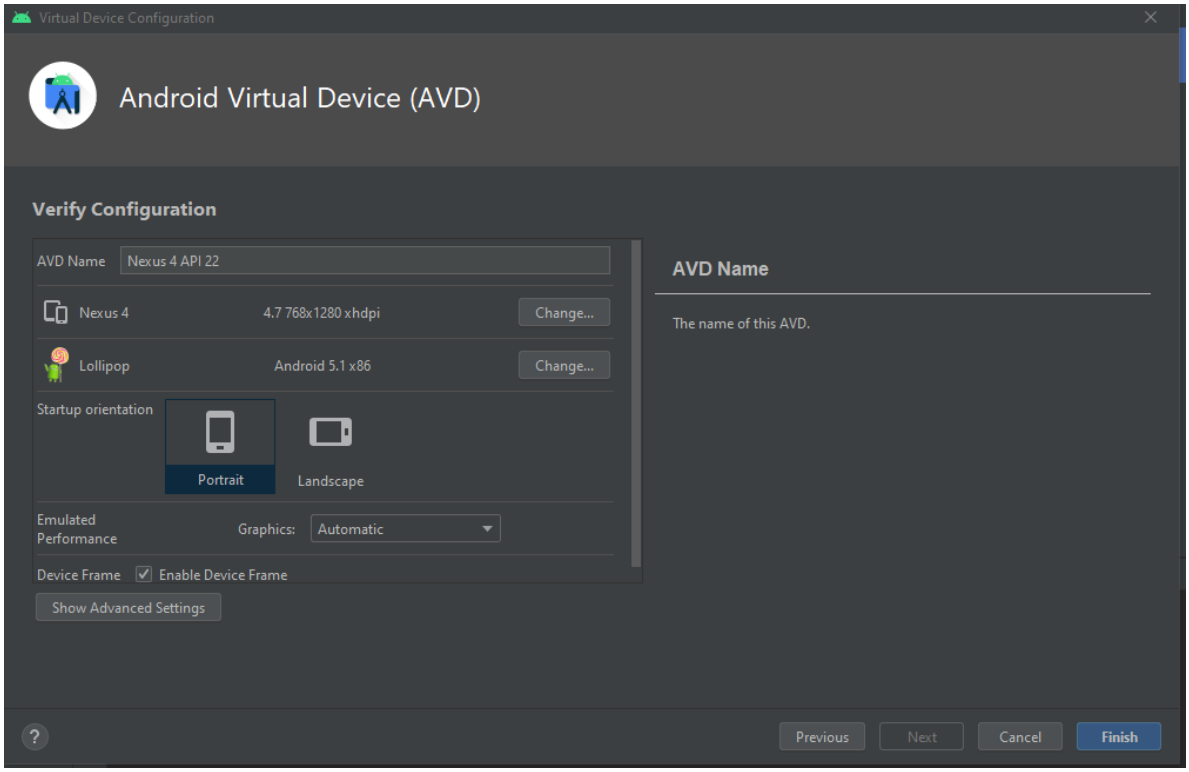


Fonte: Do Autor.

3.2 EMULADOR ANDROID STUDIO

No software Android Studio, em “Tools/Device Manager”, na opção “Create Device”, deve ser criado o emulador para execução dos casos. As configurações mínimas para o dispositivo são versão 5.0 do Android ou superior conforme exemplo da Figura 4.

Figura 4: Configuração Emulador Android Studio

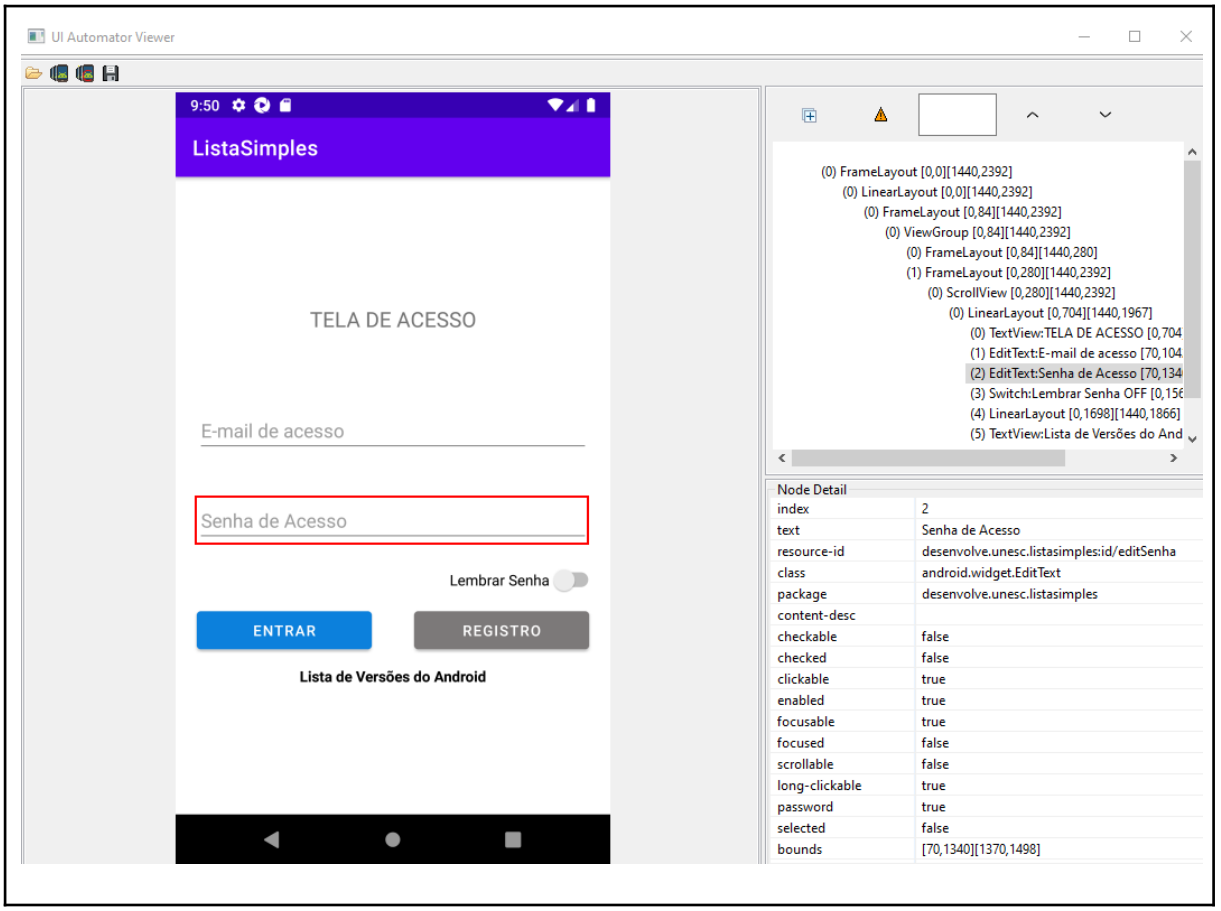


Fonte: Do Autor.

3.3 UI AUTOMATOR VIEWER

UI Automator Viewer é uma ferramenta do Android Studio que possibilita a análise dos componentes do aplicativo Android. A ferramenta oferece uma interface conveniente para verificar e analisar os componentes exibidos em um dispositivo Android. Utilizado para inspecionar a hierarquia de layouts e ver as propriedades de componentes visíveis no primeiro plano do dispositivo (ANDROID, 2022).

Figura 5: UI Automator Viewer



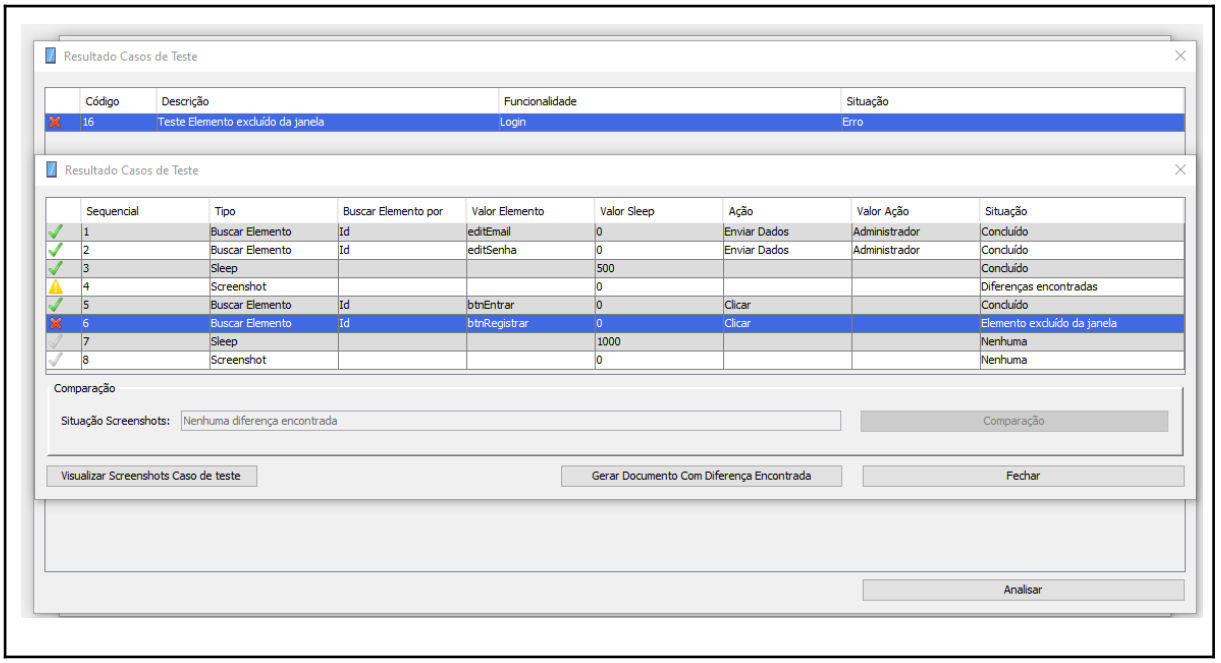
Fonte: Do Autor.

Como pode ser observado na Figura 5 UI Automator Viewer, podemos utilizar o componente “Senha de Acesso” e identificar seu ID, para posteriormente manipular o elemento na ferramenta.

3.4 COMPARAÇÃO DE IMAGENS E RESULTADOS

Ao executar os casos de testes, a ferramenta demonstra os resultados de cada item obtendo sucesso, advertência ou erro. Em cada item é demonstrado o ícone e a situação após a execução conforme observamos na figura 6.

Figura 6: Resultados de Caso de Teste

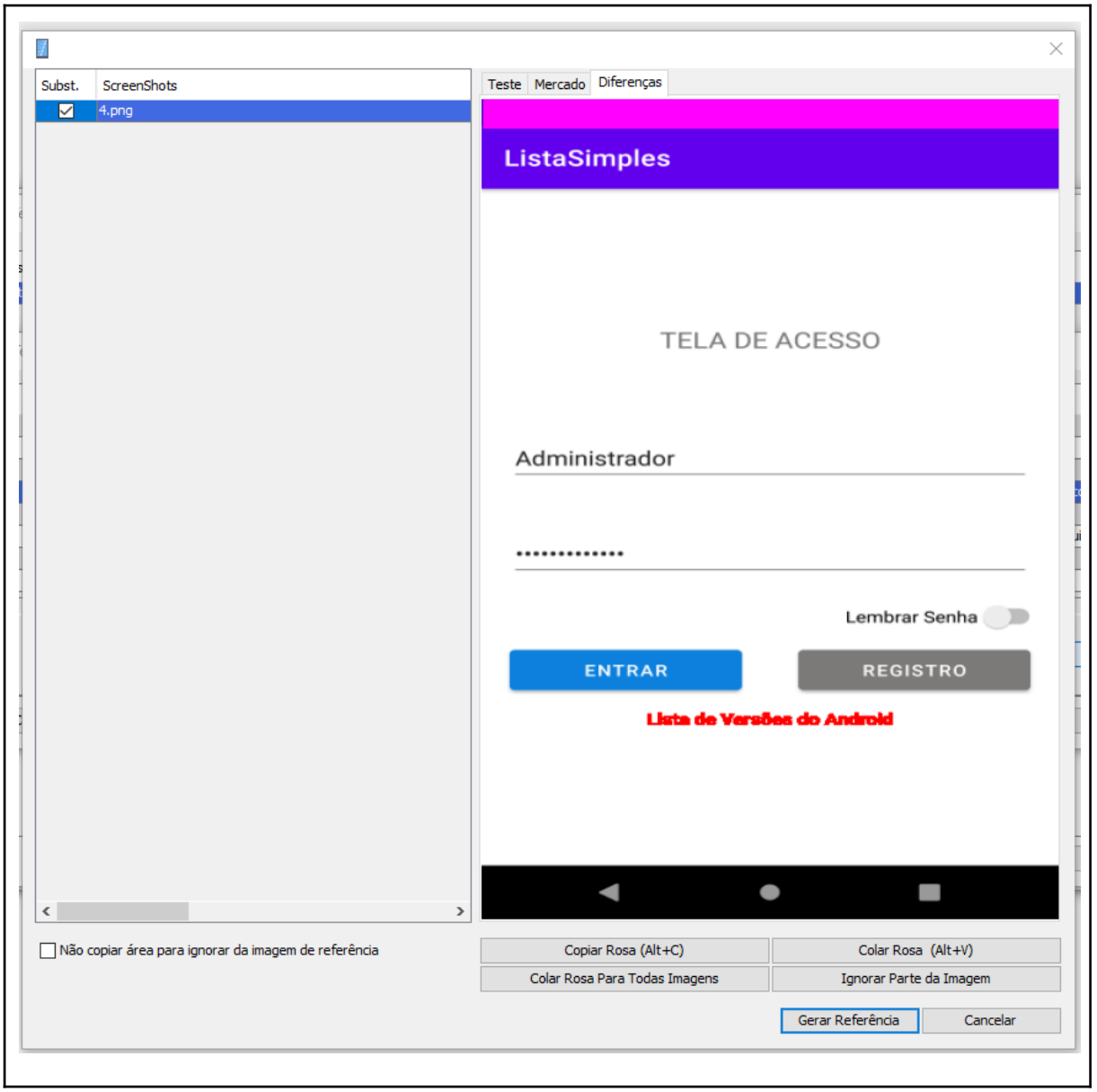


Fonte: Do Autor.

Ainda na figura 6, esse caso de teste possui um item com o tipo “Screenshot”, no qual depois de executado realiza uma comparação verificando se possui alguma imagem que foi gerada referência. No exemplo, temos uma que já foi gerada em outra execução, porém demonstrou diferença na comparação ao executarmos com outra compilação de nosso aplicativo.

A comparação de imagens realizada na ferramenta, é feita verificando o RGB pixel a pixel da imagem, comparando as duas imagens, sendo, a primeira armazenada no caso de teste e a imagem gerada a partir da execução atual. No exemplo demonstrado, obtemos uma diferença e ao clicar em "Comprar", podemos verificar as duas imagens comparadas e a diferença conforme figura 7.

Figura 7: Janela de Comparação de Imagens



Fonte: Do Autor.

A diferença é pintada de vermelho caso selecionada a guia “Diferença”. A ferramenta tem a opção de ignorar uma parte para não compararmos durante a execução, como por exemplo o horário presente na barra superior. Assim podemos gerar uma nova referência desse caso de testes para posteriores testes.

3.5 BANCO DE DADOS MYSQL

O banco de dados desta pesquisa foi desenvolvido e estruturado no MySQL na sua versão 8.0.30 e controlado pela interface MySQL Workbench versão 8.0 CE.

4 RESULTADOS E DISCUSSÃO

O resultado deste trabalho, propôs criar a ferramenta a fim de executar testes feitos pelos analistas, compreendida na área de engenharia de software, por meio de testes automatizados em dispositivos móveis, sendo este, um dos principais objetivos da pesquisa.

No desenvolvimento da aplicação, a técnica determinada foi a comparação de imagens que são realizadas durante e após a execução. A análise dos resultados está visualmente disponível ao final de cada item, em comparação de imagens e resultados (3.5), apresentando o comportamento esperado com maior agilidade.

Levando em conta o método de comparação utilizado, cada emulador ou dispositivo deve conter a sua própria referência. Visto que cada um possui sua particularidade. Observou-se que o mesmo caso de teste pode ser utilizado em dispositivo diferente, mas que ainda deve-se pensar em replicação de casos para outras versões/dispositivos, pois a ferramenta ainda não está preparada para isso.

No estudo de Eduardo e Rodrigo (2016) utilizou-se a execução em nuvem de casos de testes em scripts, com integração semelhante ao presente trabalho (utilizando a ferramenta *Appium)*, no qual foram automatizados para demonstrar seus resultados com a tecnologia *TestNG*. Em ambos os estudos possuem objetivos parecidos, que oferecem alternativa para pesquisas posteriores complementares.

Appium (2020) ressalta que a ferramenta possui a característica de ser *cross-platform*, tal tecnologia permite desenvolver scripts de automação para as duas plataformas Android e iOS utilizando a mesma API. A aplicação deste trabalho utilizou apenas scripts para a plataforma Android, no qual focamos durante a pesquisa.

Ressalta-se que, com os estudos encontrados e com o trabalho implementado, o *Appium* demonstrou ser um *framework* muito eficiente para testes em dispositivos móveis.

A ferramenta desenvolvida tem a capacidade de atender os objetivos apresentados, conseguiu diminuir o esforço gasto em testes repetitivos simples e demonstrou os resultados de forma clara. Para trabalhos futuros, espera-se criar com mais detalhes a geração do relatório com os dados dos casos de testes e diferenças para análises posteriores com o desenvolvedor do aplicativo.

5 CONCLUSÃO

A presente pesquisa teve como objetivo o desenvolvimento de uma ferramenta para suporte a automação de teste em dispositivos móveis utilizando os conceitos de engenharia de software, melhorando o processo de desenvolvimento de software em dispositivos móveis por meio de testes automatizados.

Conclui-se que é possível diminuir testes repetitivos simples utilizando a ferramenta, com a possibilidade de melhorar processos e aumentar a cobertura de testes de um aplicativo. A ferramenta executa vários casos de teste de uma determinada funcionalidade em apenas uma execução.

Os resultados obtidos processando casos de testes criados pelos usuários têm um feedback benéfico para a análise ao finalizar. Pensando em como demonstrar de forma clara o final do algoritmo de teste criado pelo testador.

A solução obteve um resultado satisfatório e embora automação de testes em dispositivos móveis não seja muito comum em empresas, espera-se que a pesquisa seja aprofundada em uma área não muito utilizada na engenharia de software.

Para trabalhos futuros sugere-se: realizar estudos para replicação de casos de teste reutilizando em emuladores/dispositivos diferentes; criação de baterias de teste com execuções completas de casos de teste; realizar estudo para reaproveitamento de scripts para IOS.

REFERÊNCIAS

APPIUM. Site oficial do Appium. Disponível em: <http://appium.io/>. Acesso em: 27 out. 2022.

ANDROID, Desenvolvedores. **UI Automator**. Disponível em: <https://developer.android.com/training/testing/ui-automator?hl=pt-br>. Acesso em: 27 out. 2022.

BERNARDO, Paulo Cheque; KON, Fabio. **A Importância dos Testes Automatizados**. Artigo publicado na Engenharia de Software Magazine, 2008. Disponível em: <https://www.ime.usp.br/~kon/papers/EngSoftMagazine-IntroducaoTestes.pdf>. Acesso em: 01 nov. 2022.

PRÁ, Marcelo Dehon Batista de. **O USO DAS TÉCNICAS DATA-DRIVEN E KEYWORD-DRIVEN NO PROCESSO DE DESENVOLVIMENTO DOS SCRIPTS DE AUTOMAÇÃO DE TESTES FUNCIONAIS DE SOFTWARE**. 2012. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - UNESC. Criciúma.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: McGraw-Hill, 2011.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: uma abordagem profissional**. 8. ed. Porto Alegre: McGraw-Hill, 2016.

SÁ, Eduardo Corrêa de; SILVA, Rodrigo. **AUTOMAÇÃO DE TESTE PARA DISPOSITIVOS MÓVEIS E EXECUÇÃO DOS SCRIPTS DE TESTE AUTOMATIZADOS NA NUVEM**. 2016. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - UNISUL. Florianópolis.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo: Pearson, 2011.