UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA

LUCAS MOREIRA LOPES

TESTE DE VALIDAÇÃO DE SOFTWARE PARA *SMARTPHONES* ANDROID

Manaus
2022

LUCAS MOREIRA LOPES

TESTE DE VALIDAÇÃO DE SOFTWARE PARA *SMARTPHONES* ANDROID

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas como pré-requisito para a obtenção do título de Engenheiro em Eletrônica.

Orientador: Angilberto Muniz Ferreira Sobrinho.

Manaus
2022

Universidade do Estado do Amazonas – UEAEscola Superior de Tecnologia – EST

Reitor:
André Luiz Nunes Zogahib.

Vice-Reitora:
Katia do Nascimento Couceiro, Dr.

Diretora da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo, Me.

Coordenador do Curso de Engenharia:
Bruno da Gama Monteiro, Me.

Banca Avaliadora composta por: *Data da defesa: 27/10/2022*
Prof. Angilberto Muniz Ferreira Sobrinho Dr. *(Orientador)*
Prof. José Nilson Cordeiro de Oliveira, Me. (Avaliador 1)
Prof. Daniel Guzman Del Rio, Dr. (Avaliador2)

CIP – Catalogação na Publicação

Lopes, Lucas

Desenvolvimento de um sistema teste de *software* do sistema android de smartphones para a acelerar o processo de entrega de novas *releases* no mercado. / Lucas Moreira Lopes; [orientado por] Prof. Angilberto Muniz Ferreira Sobrinho – Manaus: 2022.61 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica).
Universidade do Estado do Amazonas, 2022.

1. Teste de Validação de Software para smartphone android.I Muniz,Angilberto.

LUCAS MOREIRA LOPES

TESTE DE VALIDAÇÃO DE SOFTWARE PARA *SMARTPHONES* ANDROID

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Eletrônica da Escola Superior de Tecnologia da Universidade Estadual do Amazonas, como pré-requisito para a obtenção do título de Engenheiro em Eletrônica.

Nota obtida:_____(_____)

Aprovada em_____/_____/_____.

Área de concentração: Engenharia de *Software*, Sistemas Embarcados.

BANCA EXAMINADORA

_____
Orientador: Angilberto Muniz Ferreira Sobrinho, Dr.

_____
Avaliador: Daniel Guzman Del Rio, Dr.

_____
Avaliador: José Nilson Cordeiro De Oliveira, Me.

Manaus
2022

DEDICATÓRIA

Ao meu Rei Jesus, aos meus pais, a minha noiva e a todos os meus familiares e amigos que direta ou indiretamente me apoiaram nesta caminhada. Meu muito obrigado.

RESUMO

A presente pesquisa tem a finalidade de apresentar o desenvolvimento de um sistema de teste de validação de *software* para *smartphones* androids que poderá ser aplicado em empresas que desenvolvem o sistema android. Este sistema consegue otimizar e acelerar o processo de criação e validação dos sistemas operacionais dos *smartphones* deixando as entregas de novas atualizações e correções mais seguras e rápidas. O sistema será apresentado de uma maneira muito simples e eficiente com uma comunicação que utiliza o Debug Bridge (ADB) que facilita uma variedade de ações dos dispositivos. Primeiramente mostra-se a fundamentação teórica das tecnologias utilizadas no projeto: Android, tipos de teste de *software*, ADB e Python. Em seguida, em métodos e materiais, são apresentadas as etapas de desenvolvimento da pesquisa e os materiais utilizados. Posteriormente, em implementação da pesquisa, haverá a exposição do que foi executado durante a pesquisa. Nesta etapa são apresentados os dados do sistema feitos em Python. Foram feitos testes experimentais e educacionais para demostrar o funcionamento do teste de validação de *software*. Nesta etapa também foram feitos testes de tempo de execução entre o sistema de validação e testes manuais que são os mais utilizados atualmente. Em seguida são mostrados os resultados dos testes de validação e dados da sua eficácia. Finalmente se conclui que os testes de validação de *software* aceleram a entrega de novas atualizações para os clientes, sendo uma maneira viável e segura para as empresas que atuam neste mercado de criações de sistemas operacionais android. O resultado apresentado é que por um programa pode se executar qualquer teste de validação com mais rapidez e eficiência.

Palavras-chaves: validação de *software*, teste, android, ADB, python.

ABSTRACT

This research aims to present the development of a software validation test system for android smartphones that can be applied in companies that develop the android system. This system can optimize and speed up the process of creating and validating smartphone operating systems, making the delivery of new updates and patches safer and faster. The system will be presented in a very simple and efficient way with a communication using the Debug Bridge (ADB) that facilitates a variety of actions on the devices. First the theoretical background of the technologies used in the project is shown: Android, types of software testing, ADB and Python. Next, in methods and materials, the research development steps and the materials used are presented. Later, in research implementation, there will be the exposition of what was executed during the research. In this step the system data made in Python is presented. Experimental and educational tests were performed to demonstrate how software validation testing works. This step also includes runtime tests between the validation system and the manual tests that are most commonly used today. Next the results of the validation tests and data on their effectiveness are shown. Finally it is concluded that software validation tests speed up the delivery of new updates to customers, being a viable and safe way for companies operating in this market of android operating system creations. The result presented is that any validation test can be performed faster and more efficiently with one program.

Keywords: software validation, test, android, ADB, python

LISTA DE FIGURAS

SUMÁRIO

INTRODUÇAO

A migração dos consumidores e dos negócios para ambiente online colocou o *software* em evidência. Com isso, os testes de *software* na qualidade do sistema são de suma importância pelos seguintes motivos: permitem identificar erros durante as etapas de desenvolvimento; garantem a confiança do usuário final e sua satisfação ao utilizar um *software*; permite, ainda, assegurar a qualidade do produto e seu funcionamento correto; e é fundamental para manter a reputação do negócio no setor.

Com isso o crescimento em testes de validação de *software* para smartphone android vem ganhando mais destaque entre as grandes empresas de celulares, fazendo com que os processos de testes de validação de *software* fiquem mais rápidos e eficientes, garantindo que o produto não perca a qualidade. Para isso os testes de automatização vem ganhando mais espaço no mercado em decorrência da sua agilidade e praticidade, facilitando, assim, o trabalho dos desenvolvedores de *software* na correção de *bugs* e consequentemente, gerando melhorias no sistema.

A presente pesquisa expõe o problema da carência de trabalhos voltados para acelerar o processo de validação de *software* que trazem demorar na entrega de atualizações e correções no sistema android. Nesta pesquisa, foi testada a hipótese de que seja possível o desenvolvimento de um sistema que automatize os testes de validação de *software* android para deixa de se utilizar testes manuais. O sistema também consegue exibir o teste de aplicações mostrando se suas funções estão sendo aplicadas conforme sua proposta.

Para teste e validação da hipótese acima, foram implementados os seguintes objetivos. Primeiro fez-se o estudo da programação dos parâmetros dos dispositivos celulares que utilizam o sistema operacional android. Em seguida, a realização das instalações dos *softwares* de comunicação entre o celular e computador. Por fim, foram realizados testes para viabilizar a melhor programação deixando mais clara a ideia proposta.

Após a etapa de testes, foram desenvolvidas algumas programações para simular os casos de teste de validação de *software* para garantir que as funcionalidades das aplicações dos sistemas estão seguras e com qualidade para chegar ao usuário final.

Esta pesquisa se justifica, pelo fato de que os problemas com as correções de *bugs* e atualizações para os smartphones que utilizam o sistema operacional android tem uma

dificuldade muito grande na entrega de novas versões, devido ao grande número de etapas que validação de *software* manual. Sendo assim de suma importância a automatização dos testes de *software* para que os usuários possam receber de maneira mais rápida e com qualidade as atualizações e assim gerando mais credibilidade pra empresa daquele *smartphone*. Complementarmente, pelo fato dos dispositivos android serem uma tecnologia bastante utilizada, o estudo deste sistema de validação poderá servir como base para outras aplicações.

Realizou-se uma revisão teórica que contempla os assuntos de testes de *software* utilizando Python, o funcionamento do Debug Bridge (ADB), o protocolo de comunicação *ADB* e tópicos sobre alguns parâmetros importantes da comunicação ADB. Também foram abordados os *softwares* empregados para programação do sistema android, sistema operacional utilizado nesta pesquisa.

Este trabalho está dividido nos seguintes capítulos: Referencial teórico, Métodos e Materiais, implementação do projeto e resultados obtidos.

No primeiro capítulo é abordada uma revisão dos assuntos referentes aos dispositivos e *softwares* utilizados e conceitos do protocolo de comunicação ADB, além dos tipos de testes. No segundo capítulo são mostrados os métodos utilizados para se obter o objetivo final. No terceiro capítulo, implementação do projeto, é abordada a execução dos passos citados no capítulo de métodos. Mostra-se todos os testes executados para se obter os dados e as melhorias com a automatização de testes de validação de *software*. Também é apresentada a construção de uma programação e a criação de uma interface para a simulação e visualização dos testes no smartphone. Os programas elaborados durante a implementação do projeto, são mostrados nos Apêndices.

No quarto capítulo, mostram-se os resultados dos experimentos realizados na implementação, e fazem-se análises baseadas no referencial teórico e no conhecimento adquirido durante todo o processo de pesquisa e implementação do projeto.

E por fim é apresentada a conclusão que aborda a análise do comportamento dos testes de validação de *software* para smartphone android e a discussão acerca da utilização destes testes para a aplicação em desenvolvimento de *software*.

1 REFERENCIAL TÉORICO

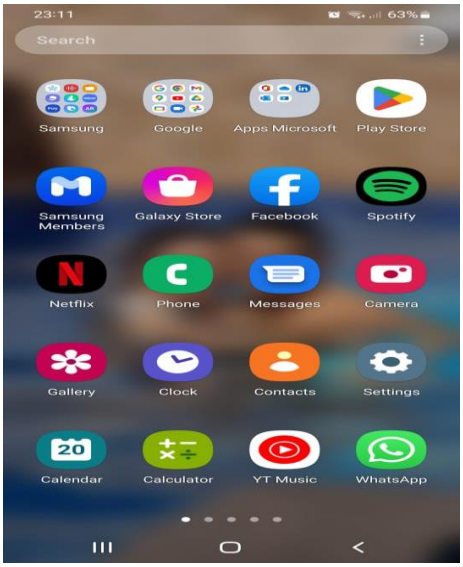
1.1 ANDROID

Android é considerada uma plataforma de *software*, líder na área de *smartphones* sendo suportada por milhares de dispositivos de diversos fabricantes disponíveis no mercado hoje, como, por exemplo: Samsung, Motorola, Xiaomi, entre outros (MITCHELL; TIAN; WANG, 2014). Ela foi desenvolvida com o propósito de oferecer aos desenvolvedores a possibilidade de produzir aplicações móveis que possam aproveitar vários recursos de um aparelho portátil, como realizar chamadas telefônicas, enviar mensagens de texto ou usar a câmera, entre outras aplicações embarcadas (PEREIRA; DA SILVA, 2003). O sistema Android foi criada pela Google e pela Open Handset Alliance e está dentro de milhões de telefones celulares e outros dispositivos móveis (BURNETTE, 2009).

Trata-se de plataforma de código aberto (*open source*) para a criação de sistemas móveis apresentando um sistema operacional Linux padrão, um ambiente de execução personalizado, um arcabouço (*framework*) para aplicações e um conjunto de aplicativos do usuário (MAJI et al., 2012). Por se basear no Linux, apresenta um modelo de *driver* poderoso, recursos de segurança, gerenciamento de processos e memória, auxílio à gerência de rede e *drivers* para um vasto grupo de dispositivos.

O ambiente de execução (*runtime*) é composto por bibliotecas gerais e pelo Dalvik, uma máquina virtual baseada em registro e otimizada para rodar considerando os requisitos de memória escassos. O arcabouço de aplicações provê aos desenvolvedores APIs (*Application Programming Interfaces*) a criação de aplicativos de usuário (popularmente conhecidos como *apps*) como mostra a (figura 1).

Figura 1: Application Programming Interfaces.



Fonte: O próprio autor

Para este projeto a forma mais viável e adequada para se colocar a ideia de teste de validação de *software* foi o uso do android utilizado pela Samsung desenvolvido pelo Google.

1.2 TESTE DE SOFTWARE

Segundo Crespo et. al (2009), testar um s*oftware* é executá-lo de maneira controlada como intuito de analisar se este se comporta conforme o especificado. Com isso, é possível perceber que se trata de uma atividade importante para analisar se o *software* desenvolvido cumpre as especificações e requisitos dos usuários. Para identificação de possíveis *bugs*, caso existam, que podem provocar falhas de *software*, é recomendada a realização de testes de maneira sistemática e minuciosa, visto que, além disso, os testes permitem fornecer evidências de confiabilidade e melhorias do software ou simplesmente a falta destes.

De acordo com Função (2011), teste de *software* pode ser classificado como um conjunto de ações que objetivam a validação ou atribuição de qualidade ao *software* adotando uma metodologia, conhecendo melhor o processo, adotando o conteúdo de engenharia de *software*, e recorrendo a alguns modelos de engenharia.

De maneira geral, o teste é um grupo de processos onde a execução será feita em cadeia, com o propósito de qualificar, minimizar *bugs*, por meio da apuração, mapeamento e análise rigorosa, e que tem a finalidade de garantir a confiabilidade, integridade no desenvolvimento/modificação de um *software*. O teste tem como atribuição garantir a qualidade do *software* e diminuir determinadas falhas críticas em produção.

1.3 TIPOS DE TESTES

De acordo com Bourque e Fairley (2014), o teste de *software* é normalmente realizado em diferentes níveis ao longo dos processos de desenvolvimento e manutenção. Ou seja, o alvo do teste pode variar: um único módulo, um grupo de tais módulos (relacionados por fim, o uso, comportamento ou estrutura), ou um sistema inteiro. Três grandes estágios de testes podem ser conceitualmente distintos, ou seja, Unidade, Integração e Sistema. Nenhum modelo de processo está implícito, nem são de qualquer uma destas três fases tidos como tendo uma maior importância do que os outros dois. Na Figura 2 são mostrados alguns tipos de testes segundo o seu alvo.

Figura 2: Tipos de teste de acordo com o seu alvo

Fonte: (SILVA, 2014, p. 20)

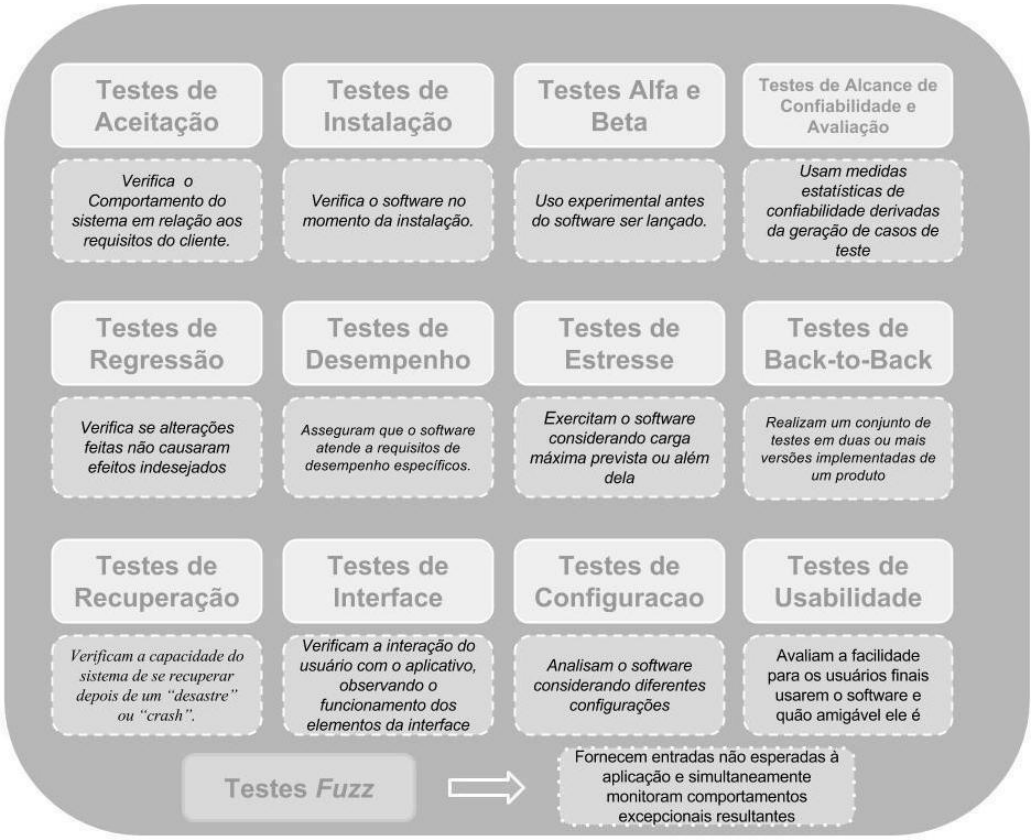
Segundo Bourque e Fairley (2014), o teste é realizado para um determinado objetivo, o qual está apresentado relativamente de forma explícita, e com diferentes graus de precisão. Estando o objetivo expresso em termos quantitativos precisos, torna-se possível o controle a ser estabelecido sobre o processo de teste.

O teste pode ser destinado a verificar propriedades diferentes. Os casos de testes podem ser projetados para verificar se as especificações funcionais são corretamente implementadas, sendo variadamente referido na literatura como testes de conformidade, teste de correção ou teste funcional. No entanto, várias outras propriedades não-funcionais podem ser testadas, incluindo o desempenho, confiabilidade e usabilidade, entre muitos outros.

Outros objetivos importantes para os testes incluem (mas não estão limitados a) medição de confiabilidade, avaliação de usabilidade e aceitação, porque seria feita de diferentes abordagens. Na Figura 3 é mostrada uma breve descrição sobre tipos de teste conforme os objetivos, considerando os tipos previstos no Guide to the Software Engineering Body of Knowledge, conhecido pela sigla SWEBOK (BOURQUE; FAIRLEY, 2014) acrescidos apenas dos Testes Fuzz (ilustrados na parte inferior da figura). As fontes para as definições utilizadas nesta figura foram baseadas no próprio SWEBOK e nos trabalhos de Delamaro; Maldonado;

Jino (2007), Molinari (2003), Bastos et al. (2007) e Ye et al. (2013).

Figura 3: Tipos de teste de acordo com o seu objetivo.



Fonte: (SILVA, 2014, p. 21)

1.4 PYTHON

Python é uma linguagem de programação de alto nível ou *High Level Language*, dinâmica, interpretada, modular, multiplataforma e orientada a objetos, uma forma específica de organizar softwares onde, a grosso modo, os procedimentos estão submetidos a classes, o que possibilita maior controle e estabilidade de códigos para projetos de grandes proporções.

Por ser uma linguagem de sintaxe relativamente simples e de fácil compreensão, ganhou popularidade entre profissionais da indústria tecnológica que não são especificamente programadores, como engenheiro, matemáticos, cientistas de dados, pesquisadores e outros.

1.5 ANDROID DEBUG BRIDGE (ADB)

O Android Debug Bridge (ADB) é uma ferramenta de linha de comando versátil que permite a comunicação com um dispositivo. O comando ADB facilita uma variedade de ações do dispositivo, como instalar e depurar aplicativos e fornece acesso a um Shell Unix que pode

ser usado para executar diversos comandos em um dispositivo. Ele é um sistema cliente-servidor com três componentes:

Um cliente, que envia comandos. O cliente é executado no computador de desenvolvimento. Você pode emitir um comando ADB para invocar o cliente de um terminal de linha de comando.

2 MÉTODOS E MATERIAIS

Neste capítulo será mostrado como ocorreram os testes de validação de *software* desde as pesquisas e as instalações do *software* Python até as comunicações feitas com o *smartphone* e os parâmetros de testes. O sistema consiste em fazer testes de validação das aplicações do sistema android através de *script* de testes criados em programação em Python com a finalidade de fazer simulações dos usuários do sistema operacional android. O sistema foi desenvolvido nas seguintes etapas:

Na primeira etapa foram realizadas pesquisas nas áreas de programação, dando enfase na programação Python, para melhor entendimento e uso das bibliotecas em Python e suas variadas plataformas de ambiente de testes (IDE). Foi acessado o site do fabricante para download da biblioteca e instalação na IDE do Python.

Na segunda etapa foram feitos testes de envio de dados para verificar o comportamento do ADB instalado no computador utilizado para fazer os testes de validação. Para realizar esses testes foi feito instalações de bibliotecas com os comandos específicos das áreas do LCD para receber os comandos corretos. No desenvolvimento desta fase do projeto foi utilizado as ferramentas PyCharm e ADB.

Na terceira fase foram escolhidos os equipamentos para serem usados no processo de elaboração dos testes de validação de *software*, sendo utilizados os de mais fácil acesso educacional. Foi usado um Notebook da marcar LENOVO modelo g400s touch intel Celeron dua core 8gb ram 500gb SSD de 14 polegadas, um celular da marcar Samsung modelo SM-A105M (Galaxy A10) de 32gb e um cabo USB original Samsung. Esses equipamentos não tiveram nenhum custo já que possuía todos.

3 IMPLEMENTAÇÃO DO PROJETO

Neste capítulo será apresentado a realização do projeto em questão. Foi desenvolvido a programação no ambiente Python IDE, instalações do módulo virtual e os casos de teste e seus *scrips* de teste.

Serão apresentados os seguintes tópicos neste capítulo:

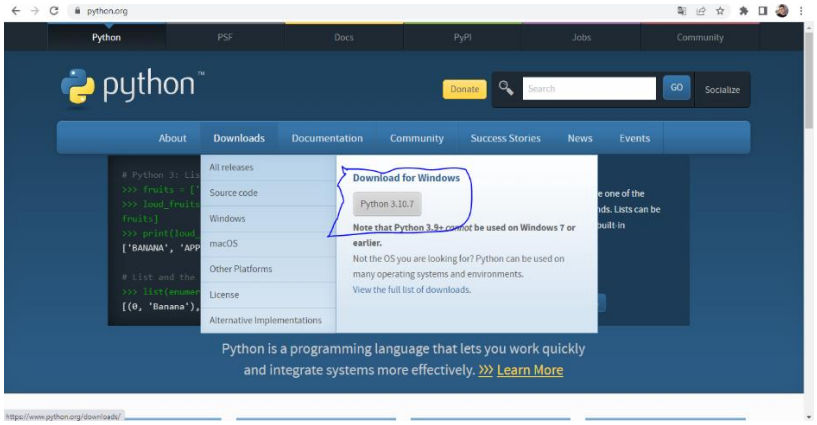
- Instalação das bibliotecas no Payton IDE;
- Instalação do modulo virtual;
- Os casos de teste e seus *scrips* de teste;

3.1 INSTALAÇÃO DAS BIBLIOTECAS NO PYTHON IDE

Para desenvolver esse projeto foram instaladas algumas bibliotecas específicas que serão usadas para as validações dos testes de *software.*

No princípio foi necessário baixar Python para máquina com o Windows como seu sistema operacional, foi preciso baixar o instalador que no site oficial Python com mostra a figura 4.

Figura 4: Instalador no site oficial Python.



Fonte: Próprio autor.

Em seguida foi verificado o *download* para máquina com sistema de 64bit que é o sistema do Notebook utilizado no projeto. Depois que o instalador esteve disponível, foi clicado duas vezes nele para iniciar o assistente de instalação do Python. O processo de instalação e bem simples:

1. Marque a opção “*Add Python to PATH*.”

2. Em seguida clique em *“Intall Now*.”

3. Será mostrado uma tela de “*Setup Progress*” aguarde enquanto o instalador completa o processo de instalação.

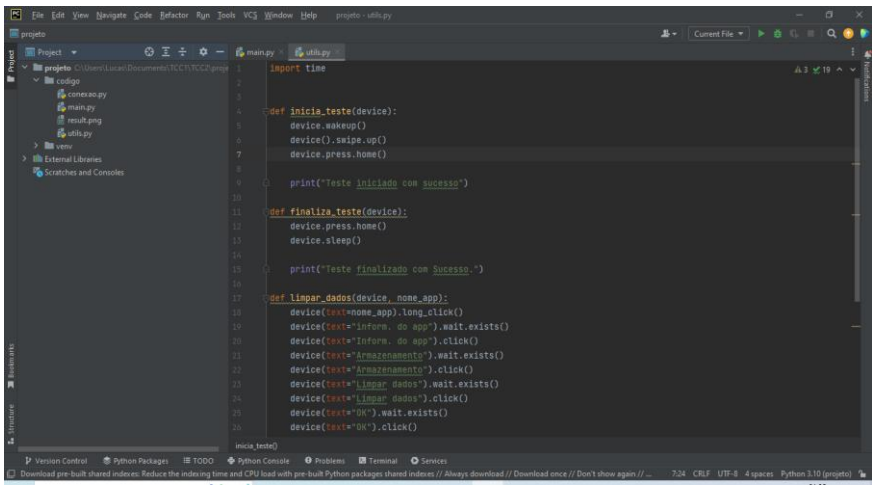
4. Se tudo ocorrer bem, a próxima tela será mostrada. Clique em “CLOSE”.

Depois para verificar se a instalação do Python foi bem-sucedida, pesquise no menu iniciar por “cmd” em seguida foi clicado duas vezes assim abrindo. Digite o seguinte comando “*python –version”*. Este comando retornará a versão do Python que está instalada na máquina.

3.2 INSTALAÇAO DO AMBIENTE VIRTUAL

Para iniciar a criação dos testes usando Python foi preciso fazer a instalação do módulo virtual que serve basicamente para executa os testes de validação do *software* que são os seus arquivos de testes, uma única classe de teste, método ou todos os testes em uma única pasta. É possível também observar os resultados no executor de testes gráficos com estáticas de execução como mostra a figura 5.

Figura 5: IDE PYCHARM.

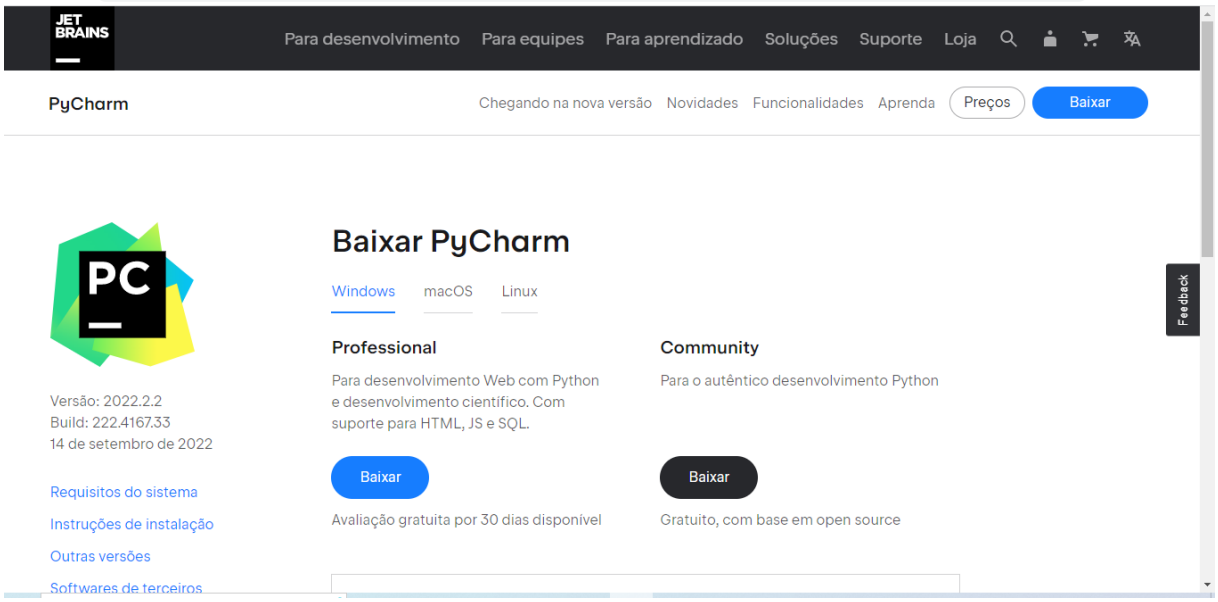


Fonte: Próprio Autor.

Existem alguns tipos de ambientes virtuais pra execução de *scripts* em Python. Nesse projeto foi escolhido o PyCharm, além de uma interface muito limpa e personalizável, é a ideal para aqueles que estão dando os primeiros passos com Python.

O primeiro passo para instalação do PyCharm foi realizar o *download* do programa, feito no site JetBrains como mostra a figura 6.

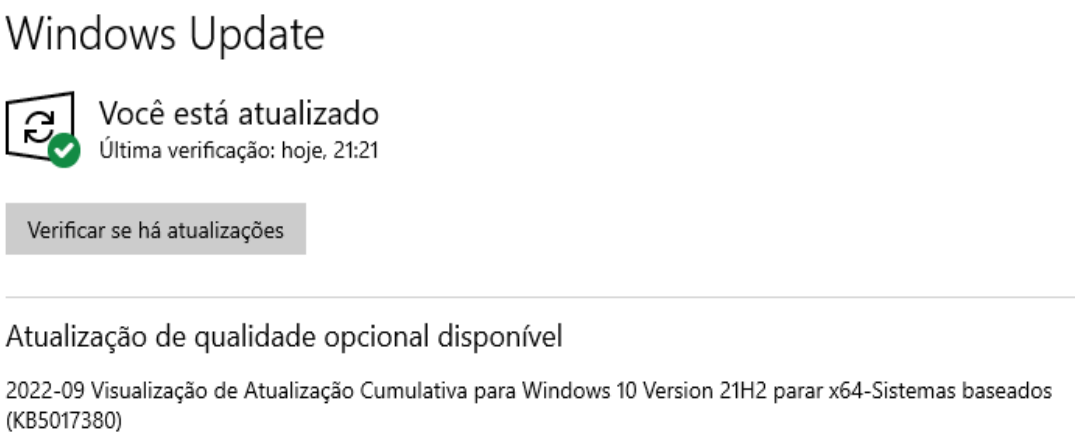
Figura 6: Site Da JetBrains para Download.



Fonte: Próprio Autor.

Nesta página da figura 6 foi escolhida a opção baixar PyCharm para Windows que é o sistema operacional da máquina utilizada nesse projeto de validação de teste de *software* como mostra a figura 7.

Figura 7: Sistema operacional Windows.



Fonte: Próprio Autor.

Foi escolhida a versão gratuita que está em negrito "**Baixar"** como pode se ver na figura 6. Em seguida foi instalado o programa, ficando assim faltando a penas a instalação do

ADB.

O próximo passo para deixar o ambiente pronto para a criação dos testes de validação de *software* foi instalar o ADB, uma tarefa muito simples:

1. Instalando o Android Studio e a SDK do Android.

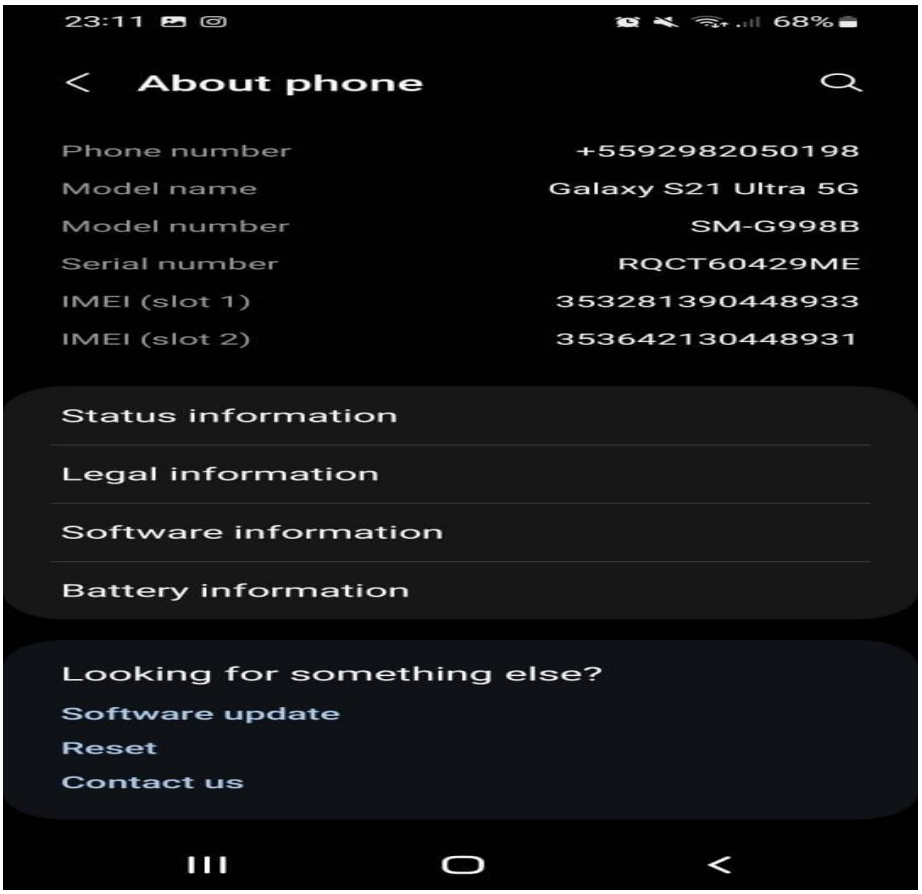
A primeira coisa a ser feita para ter acesso ao ADB do Android foi a instalação do SDK (Software *Development Kit, "Kit de Desenvolvimento de Software"*, em inglês). Para isso, foi preciso baixar e instalar o Android Studio, uma plataforma utilizada por desenvolvedores para criar aplicativos para Android. Esse programa já possui a SDK do Android e o ADB do Android embutidos na instalação.

Foi feito o *download* do instalador e assim que o processo de instalação foi concluído foi iniciado o Android Studio e sendo seguidas as instruções de configurações para que o programa realize as instalações de pacote de ferramentas de plataforma que contem o ADB do Android.

2. Habilitando a depuração USB no Smartphone.

Para usar o ADB a android no *Smartphone* escolhido pro projeto foi necessário habilitar um recurso chamado "Depuração USB" para isso foi preciso entrar nas configurações do aparelho na opção "*About phone*" como mostra a figura 8.

Figura 8: *About phone*.



Fonte: Próprio Autor.

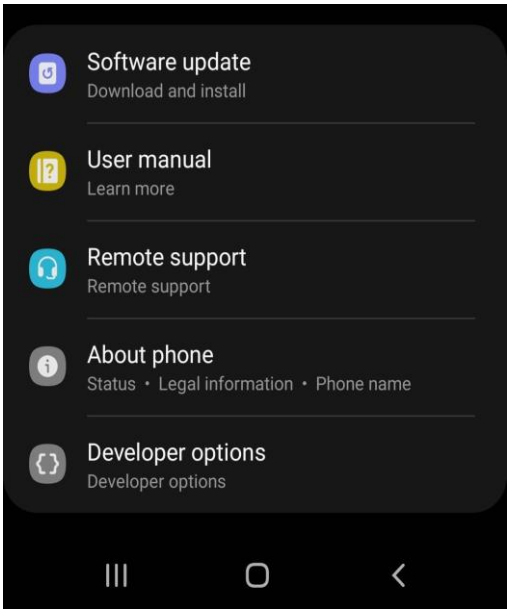
Depois foi preciso ir em *"Software information"* como é possível visualizar na figura 8. Apos isso foi preciso clicar 4x vezes em "*Bluid Number*" como mostra a figura 9, para acionar "Developer Options" como mostra a figura 10.

Figura 9: *"Software information"*

Fonte: Próprio Autor.

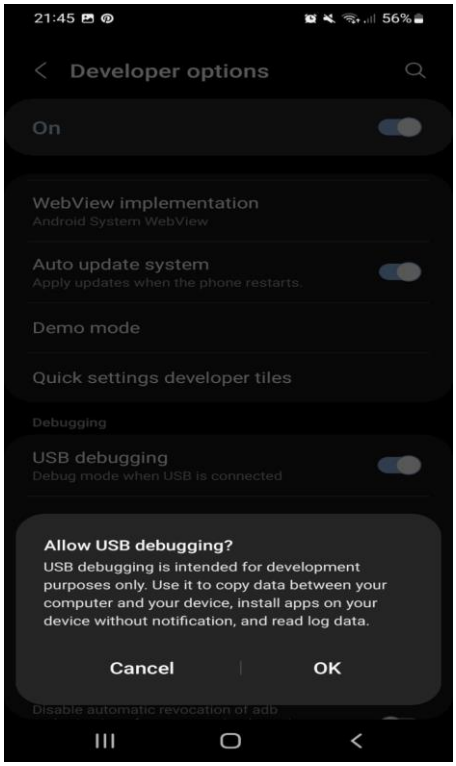
Após ativas essas opções descritas, voltamos a página principal das configurações onde vai se observar a *"Developer Options"* figura 10; onde foi clicado para entra nas opções a onde foi acionado a *"USB Debugging"* figura 11, assim ativando a comunicação entre a máquina do projeto e o smartphone de teste.

Figura 10: *"Developer Options"*



Fonte: Próprio Auto

Figura 11: *"USB Debugging"*



Fonte: Próprio Autor

Agora que todas as comunicações foram feitas, a segunda parte da implementação do projeto está finalizada.

3.3 OS CASOS DE TESTE E SEUS *SCRIPS* DE TESTE

Na terceira parte da implementação do projeto foi iniciado a criação dos casos de testes e seus *scripts*. Os casos de testes foram baseados em funcionalidades das aplicações do sistema operacional do *smartphone android* no qual o usuário entende que são simples, mas muitas vezes apresentam pequenos *bugs* que acabam prejudicando a credibilidade da marca. Veja a lista dos 5 casos de testes escolhidos para esse projeto:

1. Limpar dados do aplicativo que tem com objetivo liberar espaço, corrigir algum erro ou eliminar informações, o aplicativo escolhido foi o *"GALAXY STORE".*

2. Criar contato que tem a função simples de armazenar o número escolhido pelo usuário.

3. Excluir um contato que tem o objetivo de eliminar aquele número que não é mais do interesse do usuário.

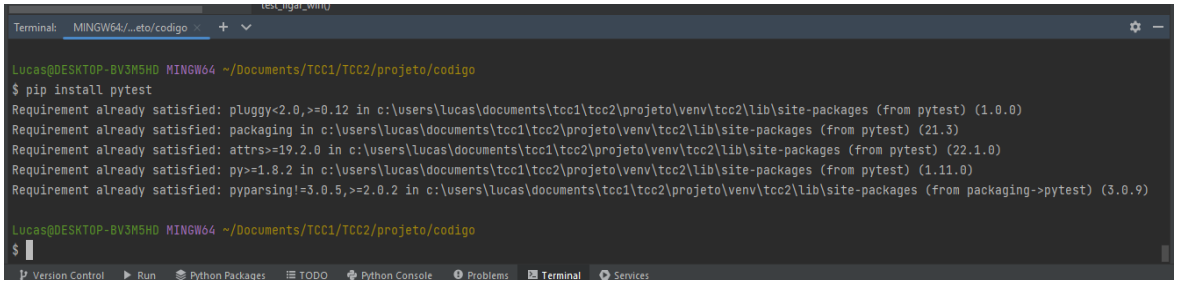
Observação: Foi criado um *script* de "procurar contato" para verificar se os resultados dos casos de teste 2 e 3 foram executados de maneira correta.

4. Desligar o botão *wi-fi* para verificar se foi interrompida a conexão.

5. Ligar o botão *wi-fi* afim de verificar as redes disponíveis.

Depois dos casos de testes criados chegou a hora de criar os *scripts* de testes no ambiente virtual PyCharm que teve como primeiro passo a execução do download de duas *framework* que são elas *UI Automator* que é uma *framework* de testes funcionais entre UI entre aplicativos e do sistema e a *Pytest* que é uma *framework* que tem objetivo te permitir escrever testes pequenos de maneira fácil, e assim possível escalar o seu uso para testes funcionais que e o caso da validação de *software* para smartphone android. Essas estruturas são instaladas pelo gerenciador de pacotes do Python. Como mostra a figura 12.

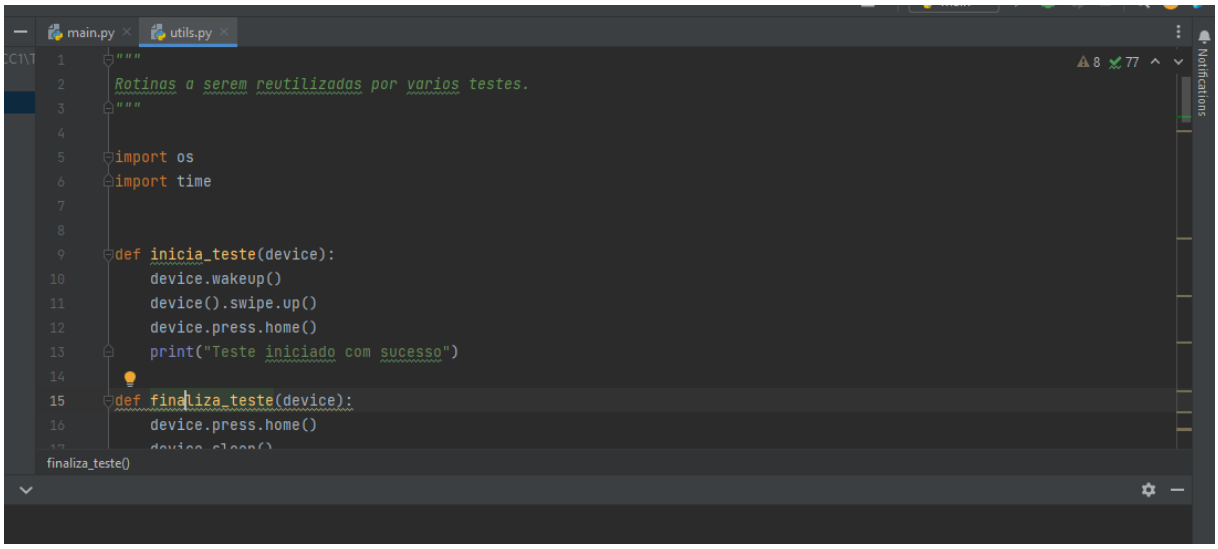
Figura 12: “Gerenciador de pacotes Python”.



Fonte: Próprio Autor.

Em seguida foram criadas funções de testes para serem reutilizadas, reduzindo o tamanho do corpo do *script*, essas funções receberam o nome de “Utils” como mostra a figura 13.

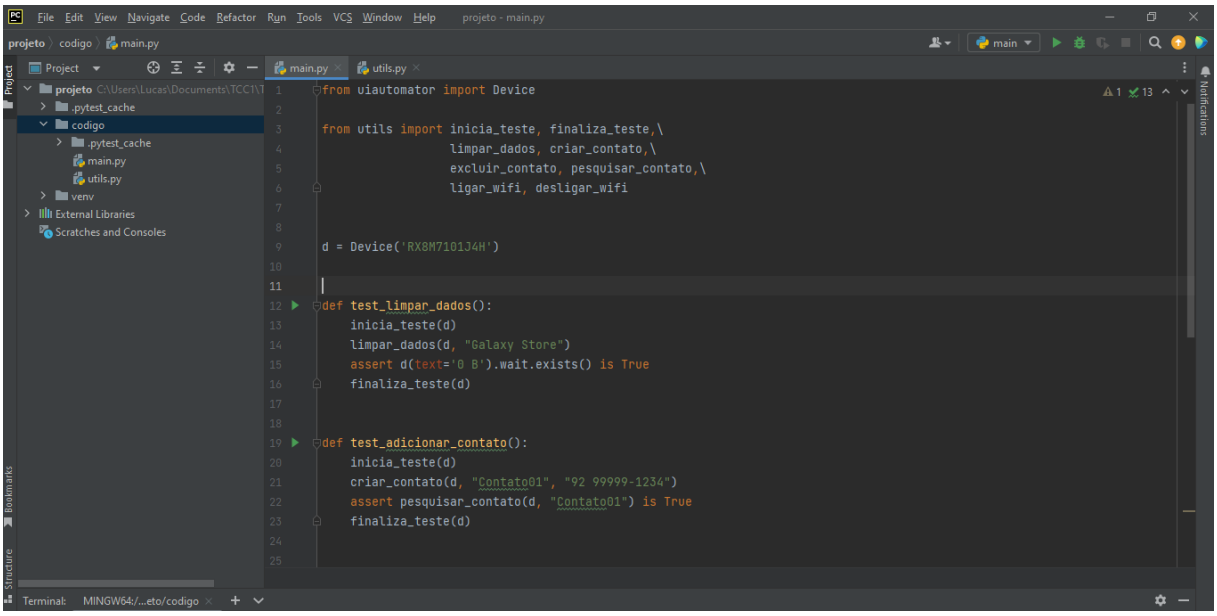
Figura 13: “utils.py”



Fonte: Próprio Autor.

Com a criação da biblioteca “utils.py” a criação dos scripts dos 5 casos testes ficam mais resumidas utilizando as funções criadas na mesma. Foram usados conhecimentos na *features* do sistema android da ONE UI 2.0 android 10 da Samsung que facilitaram na construção da biblioteca *“main.py”* como mostra a figura 14.

Figura 14: *"main.py"*.



Fonte: Próprio Autor.

4. RESULTADOS OBTIDOS

Neste capítulo será mostrado um tipo de resultado no qual foi focada a agilidade do processo de automatização, fazendo uma comparação dos resultados que hoje a maioria das empresas que fazem as validações de *software* em *smartphone android* usam os testes manuais feitos por desenvolvedores de teste.

4.1 COMPARAÇÃO ENTRE TESTES MANUAIS E AUTOMATIZADOS

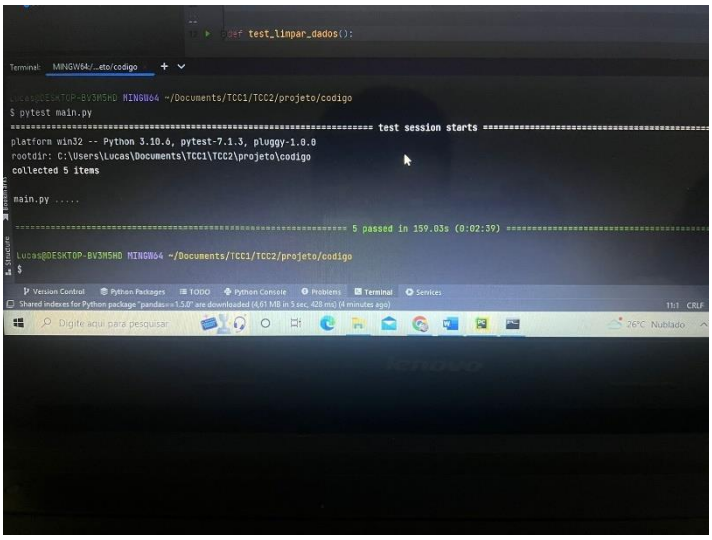
No teste de comparação feito no dia 2 de setembro de 2022, 5 casos de testes sendo feitos por uma pessoa com conhecimento em teste, onde foram usados o mesmo celular utilizado na proposta do projeto e um cronômetro de celular para verificar o tempo de execução do teste manual como mostra a figura 15.

Figura 15: “Teste executado de maneira manual”.

Fonte: Próprio Autor.

Depois disso foi executado o teste de validação de *software* para *smartphone android* que teve um resultado de tempo mais expressivo e com a mesma qualidade como mostra a figura 16, apresentada a seguir:

Figura 16: “Teste executado pelo teste de validação”.



Fonte: Próprio Autor.

O teste manual executando os 5 (cinco) casos levou um tempo de 03min40s, já o teste de validação de *software* para *smartphone* android levou um tempo de 02min03s para fazer o mesmo procedimento.

CONCLUSÃO

No desenvolvimento da presente pesquisa foram conduzidas revisões dos assuntos de testes de *software* seus tipos e suas aplicações na criação e atualizações de novas *releases* e da tecnologia de testes automatizados utilizando Python e suas configurações e programações. Foram abordados, em seguida, os métodos que melhor se adequariam nas validações de teste de *software* para smartphones android. Também consta uma apresentação sobre o *software*Python e sua IDE, utilizados no processo de teste de validação de *software*.

No teste de validação automatizado utilizando Python fico claro que isso aumenta o processo de *softwares* validados para mercado e novas atualizações com as correções necessárias para o usuário. Tendo em vista que Python proporciona uma melhor visualização dos seus resultados foram satisfatórios, tento em vista que se trada de uma pesquisa científica. Porém, existem em numeras possibilidades de melhora a implementação e métodos do projeto, com um investimento financeiro maior e acesso a tecnologias particulares.

Na criação dos *scripts* de testes e nos casos de teste observou-se que os testes de validação de *software* para smartphone android tem um resultado mais expressivo, lucrativo e satisfatório com a sua automatização acelerando um processo de criação de *software* de um mês para uma semana.

Com base nos resultados obtidos, verificou-se que, por meio da utilização da tecnologia da automatização de validação de *software* para *smartphone* android utilizando Python é possível solucionar a problemática apresentada neste trabalho, que foi, o processo de validação de *software* de smartphones androids para mercado e suas atualizações que levam muitos dias usando os testes manuais já usando a presente proposta reduzira drasticamente o tempo da validação do sistema operacional.

REFERÊNCIAS BIBLIOGRÁFICAS

BASTOS, Anderson; RIOS, Emerson; CRISTALLI, Ricardo; MOREIRA, Trayahú. **Base de conhecimento em teste de software**. Martins Fontes, 2007.

BOURQUE, P.; FAIRLEY, R. E. **Guide to the software engineering body of knowledge.** 3ª ed. Piscataway: IEEE Computer Society Products and Service, 2014. Disponível em: www.swebok.org.

BURNETTE, Ed. Hello, **Android: introducing Google's mobile development platform**. Pragmatic Bookshelf, 2009.

CRESPO, Adalberto Nobiato *et al*. **Teste de Software no Desenvolvimento Colaborativo.** 2009.

DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. **Introdução ao teste de software**. Elsevier, 2007.

FUNÇÃO, Cleomar Dias. **Teste de Software Desmembramento do Desenvolvimento**. Revista Eduf@ tima, v.2, n.1, 2011.

HU, Gang *et al*. **Efficiently, effectively detectind mobile app bugs with AppDoctor**. In: Proceeds of the Ninth European Conference on Computer Systems. ACM, 2014. p. 18.

KHOMH, Foutse; YUAN, Hao; ZOU, Ying. **Adapting linux for mobile platforms: An empirical study of android.** In: Software Maintenance (ICSM), 2012 28th IEEE InternacionalConference on. IEEE, 2012. P. 629-632.

MAJI, Amiya Kumar *et al*. **An empirical study pf the robustness of inter-component communication in Android**. In: Dependable Systems and Netwoks (DSN), 2012 42nd Annual IEEE/IFIP Internacional Conference on**.** IEEE, 2012. P.1-12

MITCHELL, Michael; TIAN, Guanyu; WANG, Zhi. Systematic audit of third-party android phones. In: **Proceedings of the 4th ACM conference on Data and application security and privacy**. ACM, 2014. P. 175-186.

MOLINARI, Leonardo. **Testes de software: produzindo sistemas melhores e mais confiáveis: qualidade de software: soluções, técnicas e métodos**. Érica, 2003.

PEREIRA, Lucio CamiloOliva; DA SILVA, Michel Lourenço. **Android paradesenvolvedores**. Brasport, 2009.

ROVEDA, Ugo. **O que é python, para que serve e por que aprender?** KENZIE. 2020. Disponível em: https://kenzie.com.br/blog/o-que-e-python/

SILVA, Lilian Thaís Barros da. **Teste de software para aplicativos Android: um mapeamento sistemático da literatura**, 2014. Disponível em: https://repositorio.ufpb.br/jspui/handle/123456789/17057

YE, Hui *et al.* DroidFuzzer: Fuzzing the Android Apps with Intent-Filter Tag. In: **Proceedings of International Conference on Advances in Mobile Computing & Multimedia**. ACM, 2013. p. 6

APÊNDICE A – CÓDIGO DA BIBLIOTECA UTILS.PY

```
"""

Rotinas a serem reutilizadas por varios testes.

"""


import os

import time




def inicia_teste(device):

    device.wakeup()

    device().swipe.up()

    device.press.home()

    print("Teste iniciado com sucesso")


def finaliza_teste(device):

    device.press.home()

    device.sleep()

    print("Teste finalizado com Sucesso.")


def encontrar_aplicativo(device, aplicativo, paginas):
```

```
    device.press.home()
    device().swipe.up()
    for i in range(paginas-1):
        device().swipe.left()
        if device(text=aplicativo):
            print(f"Aplicativo {aplicativo} encontrado!")
            return True
    for i in range(paginas-1):
        device().swipe.right()
        if device(text=aplicativo):
            print(f"Aplicativo {aplicativo} encontrado!!")
            return True
    print(f"Aplicativo {aplicativo} não encontrado!")
    return False



def abrir_aplicativo(device, aplicativo):
    device().swipe.up()
    if encontrar_aplicativo(device, aplicativo, 2):
        device(text=aplicativo).click()
        print(f"Aplicativo {aplicativo} aberto!")
```

```
def limpar_dados(device, aplicativo):
    if encontrar_aplicativo(device, aplicativo, 2):
        device(text=aplicativo).long_click()
        device(text="inform. do app").wait.exists()
        device(text="Inform. do app").click()
        device(text="Armazenamento").wait.exists()
        device(text="Armazenamento").click()
        device(text="Limpar dados").wait.exists()
        device(text="Limpar dados").click()
        device(text="OK").wait.exists()
        device(text="OK").click()
        print(f"Dados de {aplicativo} limpos com sucesso!")


def tirar_foto(device):
    abrir_aplicativo(device, "Câmera")
    device(text="Tirar foto").wait.exists()
    device(text="Tirar foto").click()
    time.sleep(2)
    print("Foto capturada com sucesso")
```

```
def criar_contato(device, nome, numero):
    abrir_aplicativo(device, "Contatos")
    device(description="Novo contato").wait.exists()
    device(description="Novo contato").click()
    device(text="Nome").click()
    os.system(f"adb shell input text '{nome}'")
    device(text="Telefone").click()
    os.system(f"adb shell input text '{numero}'")
    device(text="Salvar").click()
    time.sleep(2)
    device.press.back()

def pesquisar_contato(device, contato):
    abrir_aplicativo(device, "Contatos")
    device(description="Pesquisar").wait.exists()
    device(description="Pesquisar").click()
    os.system(f"adb shell input text '{contato}'")
    nenhum = device(text="Nenhum resultado encontrado")
    if not nenhum:
```

```
            print(f"Contato {contato} encontrado.")
            device.press.back()
            device.press.back()
            return True
        else:
            device.press.back()
            device.press.back()
            return False

def excluir_contato(device, contato):
    abrir_aplicativo(device, "Contatos")
    device(description="Pesquisar").wait.exists()
    device(description="Pesquisar").click()
    os.system(f"adb shell input text '{contato}'")
    device(description=contato).wait.exists()
    device(description=contato).click()
    device(description="Mais opções").wait.exists()
    device(description="Mais opções").click()
    device(text="Excluir").wait.exists()
    device(text="Excluir").click()
    device(text="Mover").wait.exists()
```

```
    device(text="Mover").click()
    device.press.back()
    device.press.back()
    print(f"Contato {contato} excluido com sucesso!")

def desligar_wifi(device):
    device.open.quick_settings()
    device(text="Wi-Fi").wait.exists()
    device(text="Wi-Fi").click()
    device(text="Wi-Fi, Ativado").wait.exists()
    device(text="Wi-Fi, Ativado").click()

def ligar_wifi(device):
    device.open.quick_settings()
    device(text="Wi-Fi").wait.exists()
    device(text="Wi-Fi").click()
    device(text="Wi-Fi, Desativado").wait.exists()
    device(text="Wi-Fi, Desativado").click()
```

APÊNDICE B – CÓDIGO DO SCRIPT DE TESTES MAIN.PY

```
from uiautomator import Device


from utils import inicia_teste, finaliza_teste,\
                 limpar_dados, criar_contato,\
                 excluir_contato, pesquisar_contato,\
                 ligar_wifi, desligar_wifi



d = Device('RX8M7101J4H')



def test_limpar_dados():
    inicia_teste(d)
    limpar_dados(d, "Galaxy Store")
    assert d(text='0 B').wait.exists() is True
    finaliza_teste(d)



def test_adicionar_contato():
    inicia_teste(d)
```

```
    criar_contato(d, "Contato01", "92 99999-1234")
    assert pesquisar_contato(d, "Contato01") is True
    finaliza_teste(d)


def test_excluir_contato():
    inicia_teste(d)
    excluir_contato(d, "Contato01")
    assert pesquisar_contato(d, "Contato01") is False
    finaliza_teste(d)


def test_desligar_wifi():
    inicia_teste(d)
    desligar_wifi(d)
    assert d(text="Wi-Fi, Desativado").wait.exists()
    finaliza_teste(d)


def test_ligar_wifi():
    inicia_teste(d)
```

```
ligar_wifi(d)

assert d(text="Wi-Fi, Ativado").wait.exists()

finaliza_teste(d)
```