Universidade Federal da Bahia - UFBA
Instituto de Matematica - IM
Departamento de Ciência da Computação - DCC
Bacharelado em Sistemas de Informação - BSI

Trabalho de Conclusão de Curso

Desenvolvimento e Avaliação do GuideAutomator Mobile

Afonso Almeida da Silva

Salvador - Bahia

Dezembro de 2018

Desenvolvimento e Avaliação do GuideAutomator Mobile

Afonso Almeida da Silva

Monografia apresentada como trabalho de conclusão de curso para o curso de Bacharelado em Sistemas de Informação do Departamento de Ciência da Computação na Universidade Federal da Bahia.

Orientador: Prof. Dr. Rodrigo Rocha Gomes e Souza.

Salvador - Bahia

Dezembro de 2018

Desenvolvimento e Avaliação do GuideAutomator Mobile

Rodrigo Rocha Gomes e Souza

Monografia apresentada como trabalho de conclusão de curso para o curso de Bacharelado em Sistemas de Informação do Departamento de Ciência da Computação na Universidade Federal da Bahia.

Banca Examinadora:

Prof. Dr. Rodrigo Rocha Gomes e Souza (Orientador)
UFBA

Prof. Dr. Tiago de Oliveira Januario
UFBA

Iury Maia de Almeida
UFBA

À minha família

Agradecimentos

Primeiramente quero agradecer do fundo do meu coração a pessoa mais importante em minha vida, minha mãe Rosineide Santos Almeida, pela sua confiança, investimento e participação durante todo meu processo de formação e também a minha irmã, Caroline Almeida da Silva, por sempre me ajudar e socorrer nos mais diferentes momentos e situações. Agradeço a todos os meus professores e professoras desde a escola até os da graduação e à Universidade Federal da Bahia (UFBA) por todas as oportunidades oferecidas que me foram oferecidas durante o meu período de formação e ao meu orientador Rodrigo Rocha Gomes e Souza pela sua paciência, disponibilidade e orientação durante a execução de todo este trabalho.

"A Matemática não mente. Mente
quem faz mau uso dela".
Albert Einstein

Resumo

O GuideAutomator é uma ferramenta para automatização na geração de documentação de software para usuários finais. Através da entrada de um arquivo de texto escrito em linguagem de marcação, capturas de screenshots são feitas na aplicação a ser documentada produzindo um documento de software como artefato de saída, porém a versão atual do GuideAutomator realiza apenas a documentação de aplicações web. Este trabalho tem como objetivo apresentar uma outra versão do GuideAutomator que funcione em dispositivos móveis, além de uma execução e avaliação de um experimento piloto com a ferramenta.

Palavras-chave: <Documentação>, <mobile>, <usuário>, <GuideAutomator>.

Abstract

GuideAutomator is a tool to automate the generation of software documents for end users. By entering a text file written in markup language, screenshots are made in the application to be documented and a software's document is generated as an output artifact, however a current version of GuideAutomator performs only one page of web applications. This paper aims to present another version of the Mobile Task Automation Guide, as well as an execution and evaluation of a pilot with a tool.

Keywords: <Documentation>, <mobile>, <user>, <GuideAutomator>.

Sumário

Lista de Figuras

Lista de Tabelas

1. Introdução

O uso das tecnologias da Informação está cada vez mais difundido e recorrente na sociedade contemporânea, as pessoas estão cada vez mais conectadas e a cada dia exigem um maior acesso à informação em um tempo de resposta cada vez menor. Ao fazer o uso de uma nova tecnologia é natural que o usuário tenha dúvidas a respeito de como realizar algum tipo de operação ou procedimento e nestes casos eles podem recorrer à documentação da tecnologia que estão usando para saber como proceder. Porém, em muitos casos essa documentação está desatualizada fazendo com que as informações apresentadas no documento para o usuário não estejam de acordo com o estado atual do software [1]. A documentação normalmente é feita e mantida em arquivos binários de ferramentas como o LibreOffice e ou Microsoft Office [2], mas o uso dessas ferramentas mostra-se pouco eficaz para a realização da tarefa de documentar, existem problemas ligados a compatibilidade entre os arquivos ao serem acessados em ambientes multiplataforma como Windows e Linux, dificuldades para realização do controle de versão e junção do material produzido por equipes com diferentes contribuidores, além do esforço necessário para executar os casos de testes, capturar novos screenshots da aplicação e adicioná-los ao documento, soma-se a isso um intervalo cada vez mais curto entre o lançamento das releases. Como resultado, há um cenário no qual torna-se difícil e custoso manter um documento de software que esteja de acordo com o status atual da aplicação em produção.

A fim de se solucionar esses problemas em 2016 foi desenvolvida uma ferramenta chamada GuideAutomator[1], essa é uma ferramenta para automatização de documentação destinada ao usuário final. Um arquivo .txt é usado como artefato de entrada e ao ser processado gera arquivos de saída nos formatos PDF ou HTML. O uso de arquivos de texto permite um acesso simplificado em diferentes plataformas de sistemas operacionais, além do controle de versão da documentação poder ser realizado juntamente com o código fonte da própria aplicação. Capturar screenshots da tela inteira ou de partes dela é uma das funcionalidades mais notáveis do GuideAutomator [3], já que permite que toda a documentação possa ser gerada ou atualizada ao se executar o script de entrada, reduzindo muito o esforço necessário para se manter um documento de software atualizado.

Inicialmente desenvolvido somente para aplicações web, o GuideAutomator [4] não está disponível para automatizar a documentação de outras plataformas como aplicações

[1]GuideAutomator. Disponível em <`www.npmjs.com/package/guide-automator`> . Acesso em: 28 jul. 2018.

móveis ou desktop. Este trabalho tem como objetivo expandir a área de atuação do GuideAutomator, através da criação e avaliação de uma outra versão da ferramenta para smartfones. Dentre as razões para a escolha deste ambiente em detrimento dos demais destacam-se o crescente número de novas aplicações que são desenvolvidas para o ambiente mobile e a crescente quantidade de usuários.

No capítulo seguinte, o funcionamento o GuideAutomator Web é apresentado de modo breve e são listadas diferentes tipos de aplicações móveis com suas respectivas características, além de alguns conceitos de tipos de testes de software que serão usados neste trabalho. No terceiro, diferentes tipos de frameworks de automatização de teste de softwares para dispositivos móveis são listados e comparados, a arquitetura de criação do GuideAutomator Mobile também é explanada. O quarto apresenta o ambiente de execução do experimento piloto, o roteiro do experimento e os resultados observados. E no último são apresentadas as considerações finais e pontos para melhorias futuras do projeto.

2. Fundamentação Teórica

Para se tratar a respeito do desenvolvimento de uma versão do GuideAutomator que esteja disponível para ambientes móveis, primeiramente é preciso entender como o GuideAutomator web funciona e qual a sua estrutura e a partir desses conceitos, apresentar novos frameworks que poderão ser usados para a construção da nova versão do Guide.

Um arquivo de texto é usado como entrada e dentro dele o desenvolvedor deverá especificar todas as instruções necessárias para construção do manual da aplicação a ser documentada, as instruções são escritas usando-se Markdown[1] que é uma linguagem de marcação simples criada por John Gruber que pode ser facilmente convertida para o HTML. Ao processar o arquivo Markdown serão gerados artefatos de saída em HTML ou PDF, assim como ilustrado na Figura 2.1. Algumas características do arquivo de saída são formatações para o texto com diferentes orientações e tamanhos, capturas de telas do sistema, capturas essas que podem ser de toda a tela e ou de partes dela, além de marcações que podem ser feitas nas próprias imagens capturadas pelo GuideAutomator.

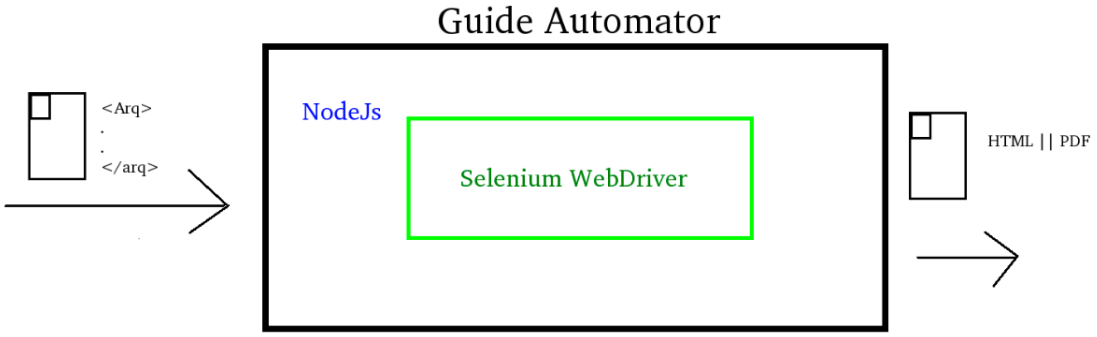


Figura 2.1: Arquitetura do GuideAutomator

Uma vez entendida a forma como o GuideAutomator funciona é preciso tratar das tecnologias usadas para confecção da ferramenta. O Guide foi desenvolvido com Node.js, isso significa que todo o código usado para abrir, interpretar as instruções em Markdown e gerar os artefatos de saída deve ser escrito em JavaScript. Porém para tirar os screenshots da aplicação que está sendo documentada é preciso que o GuideAutomator "use" a aplicação que será documentada e para isso ele faz uso do selenium WebDriver, a Figura 2.1 também ilustra esse procedimento, o GuideAutomator usa o Node.js para ler e

[1]Markdown - Markdown Guide. Disponível em <`markdownguide.org`>. Acesso em: 21 jul. 2018.

interpretar as instruções no arquivo Markdown e passa essas informações para o Selenium WebDriver para que o Selenium percorrerá a aplicação tirando capturas de telas que serão usadas para compor o documento final.

Figura 2.2: Exemplo entrada e saída GuideAutomator

A Figura 2.2 [3], em sua esquerda mostra um arquivo de entrada usado pelo GuideAutomator, observe que existem trechos escritos usando markdown e as funções escritas em Node.js ficam entre as tags "<automator>", essas funções foram processadas pelo automatizador de documentação para navegar até o Bing, preencher o campo de busca, realizar uma busca e em seguida realizou a captura de um trecho da tela, a saída da execução do script pode ser visualizada a direta da imagem.

2.1 Aplicações móveis

Durante este trabalho ao se falar a respeito de aplicações móveis refere-se apenas a aplicativos desenvolvidos para Android[2] e iOS[3], esse será considerado como o universo de aplicações móveis que desejamos alcançar com o desenvolvimento do GuideAutomator Mobile. As aplicações feitas para dispositivos móveis podem ser agrupadas em alguns tipos e apresentando diferentes características, os tipos de aplicações móveis e seus respectivos atributos são apresentados a seguir.

2.1.1 Web app

São aplicações desenvolvidas com HTML, CSS e JavaScript que são carregadas via browser. Ao serem executadas acabam abrindo uma aba no navegador e possuem pouca ou nenhuma interação com recursos do smartfone. Esse tipo de aplicativo é muito utilizado para a empresas ou instituições que desejam um aplicativo simples e não precisaram se

[2]Android. Disponível em <`developer.android.com`>. Acesso em: 05 abr. 2018.
[3]iOS. Disponível em <`developer.apple.com/documentation`>. Acesso em: 05 mar. 2018

preocupar com performance ou interagir com recursos mais profundos e sofisticados do dispositivo.

Os progressive web apps, que são um tipo de web app, permitem que mais operações possam ser realizadas pelos web apps como mandar mensagens de notificação e usar alguns recursos do celular como a vibração.

Vantagens de web apps: Fácil de escrever, fácil de serem mantidos, aplicativo multiplataforma - iOS e Android.

Desvantagens de web apps: São mais lentos que os aplicativos nativos, possuem limitações de interação com os recursos do celular, precisam ser executados no browser.

2.1.2 Nativas

São o tipo mais comum de aplicações desenvolvidas para plataformas mobile, elas são desenvolvidas usando-se linguagens e frameworks para a plataforma específica.

Vantagens de aplicações mobile nativas: São aplicações com uma melhor performance se comparadas com as nativas e híbridas, são distribuídos nas lojas de apps da plataforma e normalmente possibilitam maior interação com o usuário.

Desvantagens de aplicações mobile nativas: Maior custo para o desenvolvimento da aplicação, ferramentas de desenvolvimento mais difíceis e complexas, não é a melhor opção para aplicativos pequenos e simples.

2.1.3 Híbridas

São aplicações que são desenvolvidas usando-se a mesma estrutura de um web app, porém eles são encapsulados em um web app que funciona como uma espécie de container para a aplicação. Isso permite que aplicação possa ser instalada no dispositivo, além de permitir uma maior interação com os sensores e recursos do smartphones ao compararmos com os Web Apps.

Vantagens de apps híbridos: Mais simples e fáceis de serem construídos do que aplicações nativas, mais baratos que as aplicações nativas, pode ser portado para múltiplas plataformas, pode acessar recursos do aparelho, como câmera e cartão de memória.

Desvantagens de apps híbridos: Mais lentos que os aplicativos desenvolvidos de modo nativo, menos interativos que os aplicativos nativos, mais caros e complexos que os aplicativos web.

2.2 Testes de Software

Em qualidade de software uma das principais formas de diminuir a quantidade de erros e falhas no software que está sendo produzido é através da realização de testes. Mesmo que este trabalho não tenha um viés ligado diretamente a qualidade de software e testes, os conceitos aqui apresentados a respeito do desenvolvimento de testes serão utilizados na seção seguinte deste trabalho, uma vez que um framework de automatização de testes foi selecionado para o desenvolvimento do novo GuideAutomator.

2.2.1 Tipos de testes

Existem inúmeros tipos e estratégias de testes de software, cada um possuindo diferentes características, mas para não fugir ao escopo deste projeto serão apresentados apenas os tipos de testes, pertinentes a este trabalho:

Testes Automatizados - Um cenário de teste pode ser executado de maneira manual ou automatizada, um teste automatizado normalmente é desenvolvido com o auxilio de um framework de automatização de tarefas e tem como principal vantagem o baixo custo para a execução de novos testes, a geração de testes automatizados normalmente exigem domínio de uma linguagem programação. Um framework muito popular para a realização de testes automatizados de aplicações web é o Selenium WebDriver, que foi usado para automatização das tarefas no GuideAutomator Web.

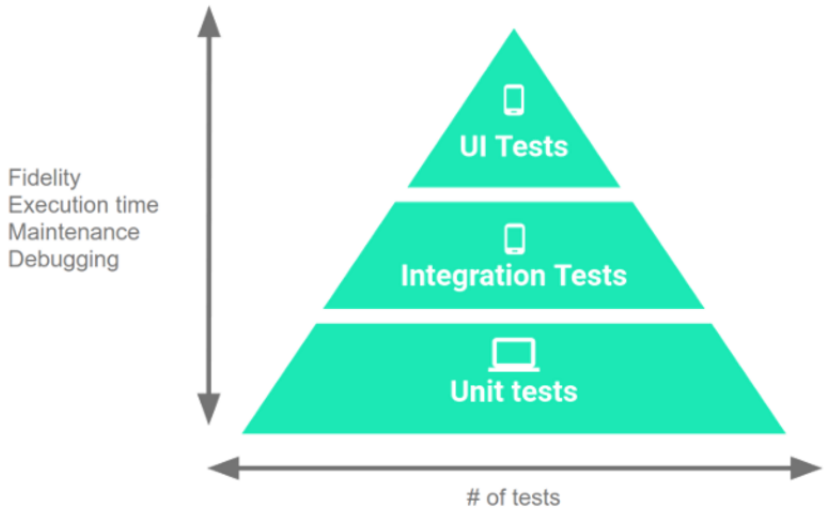


Figura 2.3: Pirâmide de Testes

Testes de Caixa Preta e Caixa Branca - Os testes também podem ser classificados segundo o seu acesso ao código fonte da aplicação a ser testada. Os testes chamados de Caixa Preta são realizados sem que o testador tenha acesso ao código fonte da aplicação, ou seja, ele não sabe como aquela rotina a ser testada foi implementada. Quanto aos

caixa branca são aqueles que testador tem acesso ao código fonte da ferramenta que está testando, normalmente os testes caixa branca são realizados pelo próprio desenvolvedor [5].

Teste a nível do sistema - Os testes também podem ser avaliados em relação a que parte do sistema está sendo testada, os testes unitários testam apenas uma função ou parte especifica do sistema e pode ser observado na base da pirâmide na Figura 2.3 [4], a sua grande quantidade se dá ao fato de que muitos teste unitários precisam ser implementados para se testar uma aplicação, os testes de integração testam um grupo de funcionalidades e componentes e podem ser observados no meio da pirâmide, e os testes de interface ficam no topo da piramide e tem o objetivo de testar a interface com o qual o usuário irá interagir. Para cada nível da aplicação a ser testada o teste poderá ser classificado segundo as definições supracitadas, como automatizado ou manual, como caixa preta ou branca.

[4]iOS. Disponível em <`developer.android.com/training/testing/fundamentals`>. Acesso em: 05 ago. 2018

3. GuideAutomator Mobile

3.1 Framework para Testes Mobile

O Selenium WebDriver[1] é uma ferramenta para automatização de tarefas em browsers, ele permite que tarefas normalmente feitas de modo manual em ambientes web possam ser automatizadas, dentre as diversas funções de automação do Selenium podemos destacar o clique em botões, preenchimento de formulários, acesso a endereços web e rolagem na tela, estas características acabam fazendo com que o Selenium seja amplamente usado para a automatização de testes de interface de aplicações web. Dentro do contexto do GuideAutomator Web, o Selenium funciona como uma espécie de núcleo, assim como ilustrado na Figura 2.1, já que ele é o responsável por realizar a navegação na aplicação e tirar os screenshots que serão usados na composição do documento, ou seja, é preciso que uma ferramenta semelhante ao Selenium WebDriver seja encontrada, porém funcional no ambiente mobile multiplataforma e a também dando suporte à maior quantidade possível de tipos de aplicações mobile.

Nome do FrameWork	**Multiplataforma**	**Mult. Linguagem**	**C. Preta**
Espresso	0	0	0
UI Automator	0	0	0
Robotium	0	0	1
selendroid	0	1	1
Appium	1	1	1
Calabash	1	0	1

Tabela 3.1: Comparação entre frameworks- Parte 1.

As Tabelas 3.1 e 3.2, apresentam uma lista com frameworks populares utilizados para automatização de testes de interface em ambientes mobile e atributos para avaliação desses frameworks quanto a sua aplicabilidade ao serem usados como ferramenta núcleo para o funcionamento do GuideAutomator Mobile. Cada coluna da tabela foi preenchida com o valor 0 ou 1 para informar se o framework da linha correspondente atende ao atributo em questão; a última coluna score mostra o somatório da pontuação total de cada uma das linhas. A coluna multiplataforma foi preenchida com o valor 1

[1] Selenium. Disponivel em <`www.seleniumhq.org`>. Acesso em 17 Mai 2018.

Nome do FrameWork	C. Branca	Híbrido	Web App	Screenshot	Score
Espresso	1	1	1	0	3
UI Automator	1	0	0	1	2
Robotium	1	1	1	1	5
selendroid	0	1	1	1	5
Appium	0	1	1	1	6
Calabash	0	1	1	0	4

Tabela 3.2: Comparação entre frameworks - Parte 2.

para os frameworks que estão disponíveis tanto para Android quanto para iOS; alguns frameworks, como Selendroid[2] já eram conhecidos por não serem multiplataforma, mas foram selecionadas para análise uma vez que no início da pesquisa, não havia certeza se seria encontrado um framework multiplataforma que pudesse contemplar os requisitos necessários. Caso um framework multiplataforma não satisfizesse os requisitos de modo pleno, poderia ser escolhido um framework que estivesse disponível apenas para documentar aplicações Android, como pode ser observado apenas o Appium[3] e o Calabash[4] funcionam tanto para Android quanto IOS.

A coluna Mult. Linguagem informa se o framework está disponível para mais de uma linguagem de programação, essa é uma informação importante para da flexibilidade ao programador na escolha da linguagem que usará para implementar o Guide Mobile. O Espresso[5], UI Automator[6] e Robotium[7] dão suporte apenas ao uso do Java o que não é uma surpresa já que esses são frameworks pensados exclusivamente para a plataforma Android, enquanto o Calabash apenas está disponivel para Ruby, e o Selendroid e Appium dão suporte a uma grande variedade de linguagens de programação que são: Java, Ruby, Python, PHP, Javascript e C#. Essa similaridade entre as linguagens de programação é natural uma vez que ambos são desenvolvidos sobre o conceito do JSON Wire Protocol.

JSON Wire Protocol é um protocolo de comunicação estabelecido sobre o HTTP que em testes automatizados tem como objetivo estabelecer a comunicação e execução

[2]Selendroid - Test automation for native or hybrid Android apps and the mobile web with Selendroid. Disponível em <`selendroid.io`>. Acesso em: 25 abr. 2018.

[3]Appium - Automation for Apps. Disponível em <`appium.io`>. Acesso em 02 mai. 2018.

[4]Calabash - Automated acceptance testing for mobile apps. Disponível em <`calaba.sh`>. Acesso em: 24 abr. 2018.

[5] Espresso - Espresso to write concise, beautiful, and reliable Android UI tests. Disponível em <`developer.android.com/training/testing/espresso/`>. Acesso em: 30 mar. 2018.

[6]UI Automator. Disponível em <`developer.android.com/training/testing/ui-automator`>. Acesso em: 24 abr. 2018.

[7]Robotium - User scenario Testing for Android. Disponível em <`www.robotium.org`>. Acesso em: 25 abr. 2018.

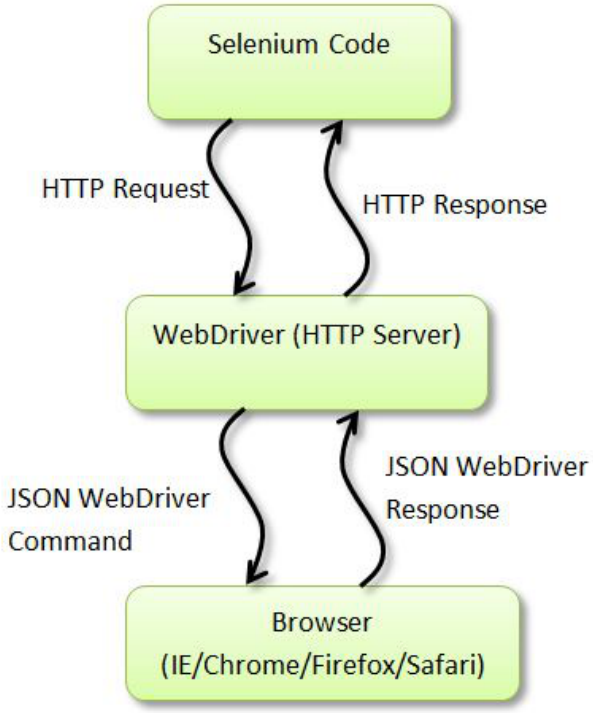


Figura 3.1: Json Wire Protocol

de testes automatizado entre três entidades o cliente com o código de execução da tarefa o WebDriver que funciona como um servidor HTTP ouvindo as requisições do cliente e as encaminham para o ambiente de execução de testes, em aplicações web este cliente normalmente é o browser de execução de testes, no caso de aplicações mobile é um smartfone seja ele real ou emulado, a Figura 3.1 [8] ilustra graficamente o funcionamento deste protocolo.

As colunas C. Branca e C. Preta informam o suporte a que tipo de testes os frameworks suportam, sendo C. Branca a abreviação usada para se referir a testes de Caixa Branca e C. Preta para testes de Caixa Preta. Neste trabalho há preferência pelos frameworks que funcionam com testes de caixa preta, para permitir aos que não tenham trabalhado no desenvolvimento de um aplicativo possam documentá-lo. Neste item ganha um destaque especial o Robotium que funciona tanto para realizar testes caixa preta quanto caixa branca. Em contrapartida temos o Espresso que funciona apenas para testes caixa branca. Todos os demais frameworks examinados trabalham com testes de caixa preta.

Híbrido e web app avaliam a suporte dos frameworks a testar aplicações não nativas, apesar de ambas as colunas serem levadas em consideração na escolha da ferramenta a coluna que se refere a aplicativos híbridos tem uma maior importância sobre a coluna

[8] Json Wire Protocol. Disponível em <`github.com/SeleniumHQ/selenium/wiki/JsonWireProtocol`>. Acesso em 05 ago. 2018.

de aplicativos do tipo web apps, isso se deve à maior quantidade de aplicativos que são desenvolvidos usando-se as tecnologias de empacotamento, usadas para gerar aplicativos híbridos. Apesar de grande parte dos itens avaliados terem se saído bem nesse aspecto notamos uma vantagem do Appium sobre o Calabash nesse ponto, uma vez que apesar de ambos poderem documentar aplicações multiplataforma o Calabash não dá suporte para realização de testes em web app. O atributo Screenshot informa o valor 1 para os frameworks que permitem capturar screenshots da tela da aplicação. Esse é um atributo muito importante já que a captura automática de screenshots da aplicação é um dos principais diferencias do Guide na geração da documentação, todos os frameworks listados para avaliação satisfazem esse requisito, com exceção do Espresso que é o único que não permite a captura de screenshots nativamente.

Ao examinar o score final dos aplicativos o Appium foi a ferramenta com o maior score com o valor 6, seguido pelo Selendroid e Robotium com 5 pontos e depois o Calabash com 4 pontos, apesar das pontuações maiores do Selendroid e Robotium, a decisão final acabou ficando entre o Appium e o Calabash, devido ao fato do peso dos atributos não ser contemplado pelas tabelas 3.1 e 3.2, o atributo multiplataforma tem um peso muito elevado em relação aos demais, assim como o atributo Web App tem um peso pequeno em comparação com os outros. O score final apresentado, não objetiva informar que o framework 1 é melhor que o framework 2, pois isso é algo fora do escopo deste trabalho, o objetivo dela é demonstrar de modo discreto qual seria o melhor framework para ser usado na concepção do GuideAutomator. Devido à maior pontuação do Appium ao seu suporte ao desenvolvimento de aplicações usando-se diferentes linguagens de programação, além de sua grande comunidade de usuários, desenvolvedores e ao seu suporte para criação de testes em ambientes mobile multiplataforma o Appium acabou sendo escolhido como ferramenta chave para o desenvolvimento do GuideAutomator Mobile.

3.2 Jupyter Notebook

O GuideAutomator é uma ferramenta de construção de manuais, ou seja, não é do intuito dos mantedores do projeto que o GuideAutomator seja uma ferramenta que precise ter um auto conhecimento em computação para ser usada, porém a versão atual da ferramenta funciona através de linha de comando, ambiente este que não é muito popular e interativo para usuários convencionais, por isso um dos desafios desse trabalho é tornar o GuideAutomator mais fácil, interativo e agradável para o usuário final e afim de se alcançar este objetivo, o GuideAutomator Mobile foi desenvolvido integrado ao Jupyter Notebook.

Jupyter Notebook [9], anteriormente conhecido como IPython Notebook, é uma poderosa aplicação web que permite a criação de documentos de texto compostos por linguagem de marção Markdown, trechos de código programação e a saída do que foi processado por essas linguagens em um único arquivo de texto. Esses itens são integrados para realizar a composição de um único documento no formato .ipynb, que nada mais é que um arquivo JSON com lista ordenada de entradas e saídas que serão processadas pelo Jupyter e poderão ser exportadas para os mais diversos formatos como HTML, apresentação de slides, Latex, PDF ou Markdown.

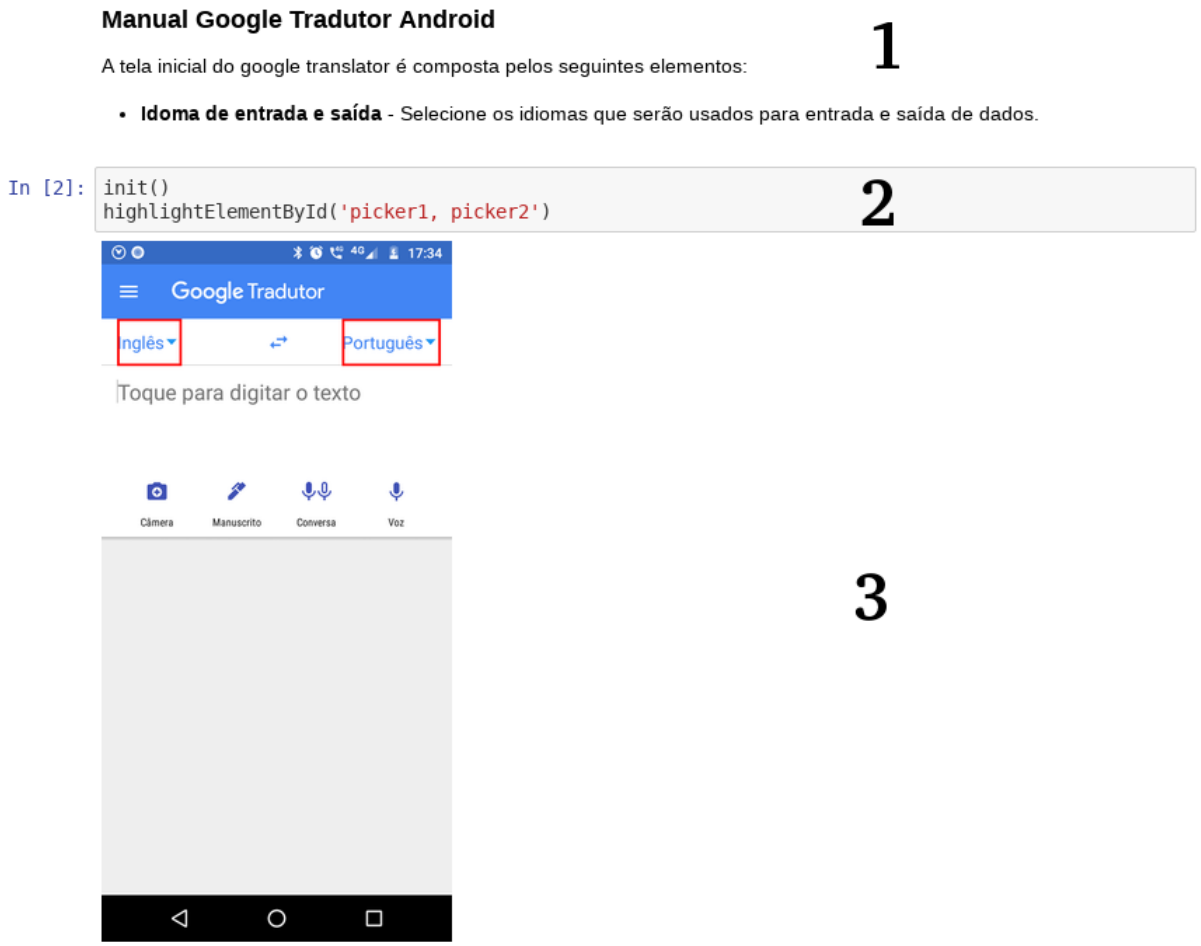


Figura 3.2: Jupyter Notebook exemplo

A fim de permitir a criação de arquivos de texto tão poderosos o Jupyter Notebook é estruturado em duas partes principais os campos de entrada de texto e os de visualização, os campos de entrada podem assumir um dos dois tipos comportamento um deles é como RichText, o que possibilita com o texto possa ser escrito usando-se a linguagem de marcação Markdown, um exemplo de texto escrito usando markdown que foi processado pelo Jupyter notebook pode ser visto na imagem 3.2 no trecho correspondente ao item

[9]Jupyter Notebook. Disponível em <`jupyter.org`>. Acesso em 17 Set. 2018.

"1"da imagem, um campo de texto como live code também permite com que seja inserido código em linguagem de programação, este campo de texto pode ser visto destacado em cinza na imagem 3.2 e marcado com o número "2". Embora o Jupyter Notebook tenha sido projetado e pensando para se trabalhar com a linguagem Python, a versão corrente do Jupyter notebook trabalha com mais de 40 tipos de linguagem de programação, incluindo Python, JavaScript e R. O campo de visualização mostram a saída do que foi processado pelas linguagens de programação e este pode ser visto na imagem 3.2, marcado pelo número "3".

3.3 Arquitetura GuideAutomator Mobile

O GuideAutomator Mobile assim como sua versão Web é um projeto open source, ou seja, seu código fonte está disponível para todos que desejarem olhar e até mesmo contribuírem com correções ou melhorias [6]. O repositório utilizado é o GitHub e está acessível em <https://github.com/aside-ufba/guide-automator-mobile>.

Embora o GuideAutomator Web tenha sido desenvolvido com Node.js[10], este não é um requisito para o desenvolvimento deste trabalho, uma vez que o Appium, ferramenta core para o desenvolvimento do GuideAutomator Mobile, e o Jupyter notebook permitem trabalhar com diversas linguagens de programação. Durante a execução deste trabalho e desenvolvimento do GuideAutomator Mobile, houveram dificuldades para desenvolver o Jupyter Notebook usando o Node.js devido as diferentes implementações de suas bibliotecas com características síncronas e assíncronas. Como o Jupyter Notebook possui uma grande afinidade com o python e está linguagem tem uma rápida curva de aprendizado e grande popularidade, o GuideAutomator Mobile foi desenvolvido usando python.

Requisito	**Tecnologia Selecionada**
Linguagem de Programação:	Python
FrameWork para automatização de Testes:	Appium
Entradas e saídas:	Jupyter Notebook

Tabela 3.3: Tecnologias GuideAutomator Mobile.

A Tabela 3.3 apresenta de modo resumido as tecnologias que foram usadas para o desenvolvimento do GuideAutomator Mobile, enquanto a Figura 3.3 apresenta a arquitetura proposta para integração destas tecnologias no GuideAutomator Mobile. O Jupyter Notebook funciona como interface para o usuário final fazendo com que toda a

[10]Node.js. Disponível em <nodejs.org>. Acesso em: 20 jul. 2018.

interação seja realizada através dele, um arquivo `.ipynb` é usado como entrada e dentro deste estão dispostos tanto trechos escritos com Markdown, como trechos de chamadas de funções em Python do GuideAutomator, como pode ser observado na imagem 3.2. Ele também é responsável por gerar o arquivo de saída, que pode ter diversos outros formatos além das saídas em HTML e PDF comumente geradas pelas versões anteriores. A API do GuideAutomator permite que os comandos executados dentro do Jupyter sejam enviados para serem processados no Appium e consequentemente no smartfone de destino, a saída gerada pelo comando faz o caminho inverso, vindo do Appium e passando pelo GuideAutomator até chegar ao Jupyter Notebook e ser usada para compor o manual do usuário.

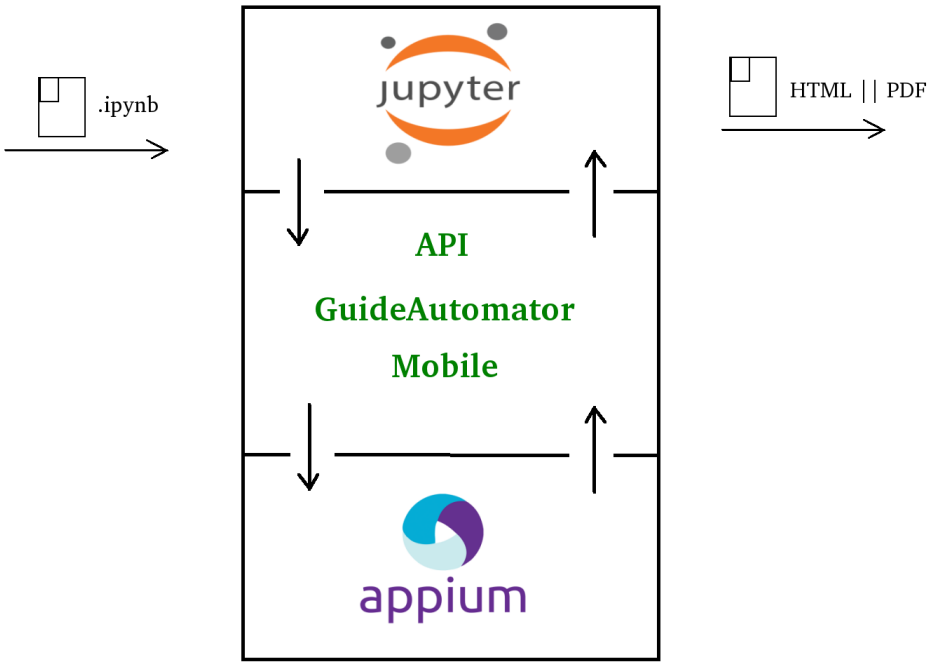


Figura 3.3: Arquitetura do GuideAutomator Mobile

3.4 Comandos do GuideAutomator Mobile

As tabelas a seguir apresentam e descrevem as funções implementadas que podem ser usadas no GuideAutomator Mobile. Muitas das funções apresentadas executam a mesma ação, porém esperando parâmetros diferentes, os parâmetros para as ações que interagem com a tela esperam os seguintes parâmetros: Id identificador único de um elemento na tela. Acessibility Id identificador único de um elemento na tela, normalmente escrito usando-se uma linguagem mais próxima do usuário o que pode incluir uso de espaços e acentos neste identificador. O xpath também tem um objetivo de identificar um

único elemento em tela, mas diferentes dos demais seu valor é formado pelo caminho do macro elemento da tela passando por todos os componentes no qual o elemento selecionado está inserido até chegar no elemento desejado, essa característica acaba tornando o xpath uma string longa e facilmente suscetível a mudanças caso o layout da tela venha a ser alterado.

Tabela 3.4: Comandos de básicos GuideAutomator Mobile

Comando	**Descrição**
init()	Inicializa uma nova sessão no smartfone destino usando o Appium.
clickById(id)	Clica em um elemento na tela do através do seletor "id".
clickByAccessibilityId (accessibilityId)	Clica em um elemento na tela do através do seletor "accessibilityId".
clickByXPath(xpath)	Clica em um elemento na tela do através do seletor "xpath".
sendKeysById(id, text)	Encontra um campo de texto através do seletor "id" e o preenche através com o valor da parâmetro "text".
sendKeysByAccessibilityId (accessibilityId, text)	Encontra um campo de texto através do seletor "accessibilityId" e o preenche através com o valor da parâmetro "text".
getTextById(id)	Recupera o texto de um label através do seu atributo "id".
getTextByAccessibilityId (accessibilityId)	Recupera o texto de um label através do seu atributo "accessibilityId".
getTextByXpath(xpath)	Recupera o texto de um label através do seu atributo "xpath".
scrolltoElementById(id)	Desliza a tela até o elemento desejado através do seu atributo "id".
scrolltoElementByAccessibilityId (accessibilityId)	Desliza a tela até o elemento desejado através do seu atributo "accessibilityId".
scrolltoElementByXpath(xpath)	Desliza a tela até o elemento desejado através do seu atributo "xpath".
quit()	Finaliza a sessão do GuideAutomator com o Appium.

Observe que na tabela a seguir é apresentado o comando tapBy, este comando tem a mesma finalidade do clickBy apresentando anteriormente neste mesmo capitulo, porém alguns elementos nos smartfones não respondem ao comando clickBy já que este comando equivale a um click realizado pelo mouse, enquanto o comando tab realiza uma interação com o elemento semelhante a um toque realizado pela mão humana, logo para garantir que o usuário do GuideAutomator Mobile consiga interagir com o elemento desejado as duas implementações foram feitas na ferramenta.

Tabela 3.5: Comandos de toque GuideAutomator Mobile

Comando	**Descrição**
tapById(id, pCount=1)	Realiza a operação de toque em um elemento da tela pelo seu atributo "id", o parâmetro "pCount"informa a quantidade de toques que serão realizados no elemento.
tapByAccessibilityId (accessibilityId)	Realiza a operação de toque em um elemento da tela pelo seu atributo "accessibilityId", o parâmetro "pCount"informa a quantidade de toques que serão realizados no elemento.
tapByAccessibilityXpath (xpath, pCount=1)	Realiza a operação de toque em um elemento da tela pelo seu atributo "xpath", o parâmetro "pCount"informa a quantidade de toques que serão realizados no elemento.
pressById(id)	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "id".
pressByAccessibilityId (accessibilityId)	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "accessibilityId".
pressByXpath(xpath)	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "xpath".
longPressById(id, pDuration = 1)	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "accessibilityId", o parâmetro "pDuration"permite definir o tempo que o elemento será pressionado.

longPressByAccessibilityId (accessibilityId, pDuration = 1):	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "accessibilityId", o parâmetro "pDuration"define o tempo que o elemento será pressionado.
longPressByXpath(xpath, pDuration = 1)	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "xpath", o parâmetro "pDuration"define o tempo que o elemento será pressionado.
longPressByXpath (xpath, pDuration=1)	Realiza a operação de pressionar em um elemento da tela pelo seu atributo "xpath", o parâmetro "pDuration"define o tempo que o elemento será pressionado.
moveToDirectionById (id, idDirection)	Pressiona e arrasta o elemento de informado através do parâmetro ”id” em direção do elemento informado através do parâmetro ”idDirection”, através do atributo ”id” de ambos os elementos.
moveToDirectionByAccessibilityId (accessibilityId, accessibilityIdDirection)	Pressiona e arrasta o elemento de informado através do parâmetro ”accessibilityId” em direção do elemento informado através do parâmetro ”accessibilityIdDirection”, através do atributo ”accessibility id” de ambos os elementos.
moveToDirectionByXpath (xpath, xpathDirection)	Pressiona e arrasta o elemento de informado através do parâmetro ”xpath” em direção do elemento informado através do parâmetro ”xpathDirection”, através do atributo ”xpath” de ambos os elementos.

Tabela 3.6: Comandos captura de tela GuideAutomator Mobile

Comando	**Descrição**
takeScreenshot()	Tira um screenshot de toda a tela do celular.
takeScreenshotElementById(id)	Tira um screenshot de uma parte da dela, através do atributo "id"do elemento informado.

takeScreenshotElementAccessibilityId (accessibilityId)	Tira um screenshot de uma parte da tela, através do atributo "accessibilityId"do elemento informado.
takeScreenshotElementXpath(xpath)	Tira um screenshot de uma parte da dela, através do atributo "xpath"do elemento informado.
highlightElementById(id, rectangleWidth = 5)	Tira um screenshot de toda a tela e destaca o elemento informado através do parâmetro "id", alterando o valor do parâmetro "rectangleWidth"é possível alterar a espessura da linha que destaca o elemento.
highlightElementByAccessibilityId (accessibilityId, rectangleWidth = 5)	Tira um screenshot de toda a tela e destaca o elemento informado através do parâmetro "accessibilityId", alterando o valor do parâmetro "rectangleWidth"é possível alterar a espessura da linha que destaca o elemento.
highlightElementByXpath(xpath, rectangleWidth = 5)	Tira um screenshot de toda a tela e destaca o elemento informado através do parâmetro do parâmetro "xpath", alterando o valor do parâmetro "rectangleWidth"é possível alterar a espessura da linha que destaca o elemento.

Tabela 3.7: Demais comandos GuideAutomator Mobile

Comando	**Descrição**
setLandscapeOrientation()	Altera a orientação do celular para modo panorama.
setPortraitOrientation()	Altera a orientação do celular para modo retrato.
sleep(time-steep)	Faz com que a execução dos testes aguarde o valor de "time-steep"em segundos.

4. Experimento Piloto

A fim de avaliar a praticidade e capacidade do GuideAutomator Mobile em gerar manuais para os usuários foi executado um experimento piloto controlado, no qual um estudante universitário foi convidado a gerar um manual de uma aplicação mobile usando dois métodos, primeiro usando ferramentas convencionais como editores de texto e imagem e o segundo usando o GuideAutomator Mobile, para ambos os cenários o critério de avaliação do experimento foi o tempo levado pelo usuário fazer o manual do software e o nível de dificuldade para confeccionar o manual, valor este que foi informado pelo próprio participante durante o Ato Cinco do experimento, para mais informações a respeito de cada ato de execução do experimento ver o tópico 4.2 desde trabalho.

O aplicativo a ser documentado durante a execução do experimento foi o Google Translate. Por ser uma ferramenta que trabalha com uso de formulários e sua popularidade foram um dos fatores chaves para sua escolha deste aplicativo, tendo em vista que o conhecimento prévio do usuário a respeito do aplicativo facilitou a execução do experimento. Além disso, o aplicativo também está disponível para IOS, embora este experimento tenha sido realizado apenas para dispositivos Android, a disponibilidade da aplicação também para IOS torna mais fácil, caso em um experimento futuro se deseje comparar os resultados obtidos com o GuideAutomator Mobile com o IOS e Android.

4.1 Ambiente de Execução do Experimento

Descrição das configurações de hardware e software do computador usado para realizar o experimento:

4.1.1 Descrição do Hardware

- **Máquina** - Dell Inspiron 7520;
- **Processador** - Intel Core i5-3230M;
- **Memória** - Card 1 = Tamanho: 4GB, Velocidade: 1600 MHz, Tipo: DDR3; Card 2 = Tamanho: 4GB, Velocidade: 1600 MHz, Tipo: DDR3;
- **Disco Rígido** - SDA: 1000.2GB , 500 RPM; SDB: SSD, size: 32.0GB;
- **Chip Gráfico** - AMD/ATI Chelsea LP [Radeon HD 7730M];

4.1.2 Softwares usados para realização do experimento através da metologia tradicional

- **Sistema Operacional** - Debian GNU/Linux 9 (stretch);
- Gnome 3.22.2;
- Appium Desktop 1.5.0;
- Libre Office 5.2.7.2;
- KolourPaint 16.08.3;
- Ferramenta de captura de tela do Sistema Operacional;

4.1.3 Softwares usados para realização do experimento usando GuideAutomator Mobile

- **Sistema Operacional** - Debian GNU/Linux 9 (stretch);
- Gnome 3.22.2;
- Appium Desktop 1.5.0;
- GuideAutomator Mobile 1.0.0;
- Jupyter Notebook 4.4.0;

4.1.4 Descrição do smartphone usado no experimento

- **Aparelho** - Moto G6 Play;
- **Sistema Operacional** - Android 8.0;
- **Aplicativo a ser documento** - Google Translate;

4.2 Roteiro do Experimento Piloto

O experimento consiste no desenvolvimento de um manual da ultima versão estável do Google Translate disponível para Android. O mesmo manual deverá ser desenvolvido de duas maneiras distintas pelo usuário: através dos métodos convencionais e através do uso do GuideAutomator Mobile. O tempo que o usuário levar para fazer cada manual foi medido. Ao final do experimento o participante foi convidado a preencher um formulário

no qual avaliou o experimento realizado; a avaliação tem como objetivo identificar pontos que podem ser melhorados durante o experimento de avaliação.

Atenção em nenhum momento o usuário teve que iniciar nenhuma das ferramentas para execução do experimento, todas elas já foram dadas executando em memória para o participante. A realização do experimento foi divida em partes aqui referidas como "atos". A seguir há uma descrição do que foi feito em cada ato:

Primeiro Ato: No primeiro ato o participante é recebido, com uma breve descrição do proposito do experimento e do que é o GuideAutomator. É importante ressaltar nesta etapa para o participante que ele não está sendo avaliado naquele momento e sim o GuideAutomator. Em seguida, o mesmo é convidado a ler e assinar o termo de comprometimento para participação no experimento, que se encontra no apêndice A deste trabalho.

Segundo Ato: Agora o participante é convidado a responder o formulário de perfil do usuário, que se encontra no apêndice B deste trabalho, nele o participante terá a oportunidade de falar um pouco a respeito de si mesmo e de suas experiências na área de Tecnologia da Informação, assim como das tecnologias pertinentes para a geração de manuais.

Terceiro Ato: Breve treinamento com duração de cerca de 20 minutos falando a respeito do uso do uso do Sistema Operacional usado para conduzir o experimento, além das tecnologias como Libre Office, Kolour Paint, ferramenta de print do Sistema Operacional e Appium Desktop, para mais informações a respeito do ambiente de execução do experimento favor ver a Seção 4.1, "Ambiente de execução do experimento". Nesta etapa também foi entregue ao usuário um manual já pronto e impresso que deveria ser desenvolvido pelo próprio usuário, este manual serviu como meta ou tarefa que deveria ser desenvolvida pelo participante. O usuário tem total liberdade para perguntar a qualquer momento e quantas vezes desejar caso tenha dúvida ou dificuldade para realizar uma tarefa. O tempo de construção do manual foi medido pelo pesquisado, além de notas foram tomadas das dificuldades e comentários realizados pelo usuário.

É importante ressaltar que o Appium Desktop foi usado para realizar a documentação em ambos os métodos tanto no tradicional quanto na sua versão automatizada com o GuideAutomator Mobile. O Appium foi utilizado em ambos os métodos porque objetivo do experimento é comparar a forma como o usuário documenta uma aplicação usando a forma tradicional e o GuideAutomator, como o GuideAutomator Mobile só pode ser usado juntamente com o Appium, para se capturar os seletores geradores através da ferramenta e usa-los nas funções de interação com software o Appium também acabou sendo usando na forma tradicional para realizar os screenshots que foram incorporados ao documento binário de texto gerado pelo Libre Office.

Quarto Ato: Neste momento o usuário recebeu um novo treinamento com uma duração semelhante a do primeiro explicando o funcionamento do GuideAutomator e como usar o Jupyter Notebook, depois disso ele foi convidado a mais uma vez realizar o mesmo procedimento do ato três, porém usando a nova metodologia.

Quinto Ato: No quinto e último ato o participante é convidado a preencher um novo formulário, no qual ele pode relatar suas impressões e considerações a respeito tanto da ferramenta quanto da realização do experimento.

4.3 Resultados do Experimento

O usuário que participou do experimento é uma estudante graduanda em Sistemas de Informação na Universidade Federal da Bahia com experiencia profissional na área de TI, mas sem nenhuma experiencia na confecção de manuais para usuários, com um conhecimento intermediário em editores de texto como o LibreOffice e Microsoft Word, e conhecimento básico em editores de imagem e front-end.

	Método Tradicional	**GuideAutomator Mobile**
Tempo	31 minutos e 49 segundos	40 minutos e 13 segundos
Dificuldade (1 fácil a 5 difícil)	2	2

Tabela 4.1: Comparação Resultados.

Durante a realização do experimento foi medido o tempo de confecção do manual para ambos os casos, tanto para a confecção do manual usando GuideAutomator, quanto sem usar a ferramenta. Observou-se que o usuário levou um tempo de 31 minutos e 49 segundos para criação do manual sem usar a ferramenta, enquanto usando o GuideAutomator o usuário levou cerca de 40 minutos e 13 segundos, como pode ser observado na tabela 4.1. É importante ressaltar que o estudante realizou primeiramente o desenvolvimento do manual usando-se as ferramentas tradicionais. Isso foi feito de modo proposital para mesmo usando-se ferramentas tradicionais o usuário ainda teria que se adaptar ao ambiente da máquina usada no experimento, ao uso do Appium e à própria tarefa de conceber um manual, apenas na segunda vez que o usuário teve contato com o GuideAutomator e o Jupyter notebook. Embora isso tivesse o objetivo de gerar menos estresse e pressão sobre o participante é importante ressaltar que caso a ordem de execução dos experimentos fosse invertida a diferença entre o tempo de execução possivelmente seria ainda maior dando uma vantagem ainda mais larga para o tempo de desenvolvimento usando-se as ferramentas tradicionais, isso se deve ao fato de que ao realizar a documentação usando-se o GuideAutomator Mobile apenas no segundo experimento o usuário

já estava ciente do que deveria ser documentado, além de está já ter usado o Sistema Operacional e o Appium Desktop na tarefa anterior. O participante também relatou ter a mesma dificuldade ao realizar as tarefas usando as duas formas, esse é um excelente sinal o que mostra que o GuideAutomator Mobile é uma ferramenta tão simples e fácil de usar quanto editores de texto e imagens normalmente usados pelos usuários menos experientes, essa resposta dada pelo participante possivelmente se dá pelo pelo ambiente de inspeção proporcionado pelo Appium Desktop e também pelo ambiente integrado para se desenvolver o manual do Jupyter noteboook.

Porém, mesmo com um menor tempo para realizar a tarefa, foi observado pelo pesquisador que o participante teve problemas para realizar tarefas ligadas a edição de imagem com a mesma qualidade, o usuário teve problemas para definir se os screenshots inseridos no documento de texto tinham o mesmo tamanho ao serem redimensionados, também houveram dificuldades para marcar elementos contidos na imagem capturada do aplicativo com a mesma precisão em todos os casos, o que acabou comprometendo parcialmente a qualidade do relatório gerado.

Já usando o GuideAutomator Mobile, houve um momento no qual o usuário acabou removendo um bloco de código usando a ferramenta de recortar do juptyter notebook de modo acidental e a ação não pôde ser desfeita. Este foi um momento no qual o usuário claramente demostrou insatisfação em relação ao comportamento da solução e que também foi citado pelo mesmo usuário em sua avaliação do experimento. Outro ponto de queixa por parte do usuário foi em relação à duração da sessão através do uso do Appium durante a realização do experimento. Em alguns momentos o Appium acabou tendo sua sessão com o dispositivo móvel interrompida o que acabou gerando intervenções na execução da tarefa para que o pesquisado pode-se reconectar os dispositivos.

5. Conclusão

A expansão da área de atuação do GuideAutomator para aplicações além do Web foi um grande desafio, já que a produção de um automatizador de documentação para smartfones sejam eles Android ou IOS ao início deste trabalho não passava de uma hipótese, mas através de muito trabalho e pesquisa o seu desenvolvimento pôde ser completado, experimentado e avaliado. Não apenas isso, mas o uso do Jupyter Notebook como ambiente para desenvolvimento de scripts do manual permitiu que a versão para Mobile do GuideAutomator deixasse de ser uma ferramenta de linha de comando e passasse a ter uma poderosa e agradável interface gráfica e ainda mantendo os padrões usados definidos pelas versões anteiores do GuideAutmator como uso de Rich Text através de MarkDown e chamadas a funções de automatização de tarefas.

Muitos pontos ainda podem ser avaliados no GuideAutomator Mobile que acabaram ficando de fora do escopo do experimento piloto realizado neste trabalho, o participante executou os testes apenas em um dispositivo Android e não foram feitos testes em um dispositivo IOS, a fim de verificar se os mesmos problemas e dificuldades encontrados no Android também serão encontrados ao usar a ferramenta em um smartfone IOS. O comportamento e opiniões do participante são importantes para a avaliação do experimento e consequentemente do próprio GuideAutomator, é apropriado a realização de um experimento com a participação de um voluntário que tenha conhecimento avançado na área de requisitos ou negócios para prover um feedback enriquecedor a respeito do funcionamento do GuideAutomator. Além disso, o escopo do experimento se manteve apenas na criação de um novo documento de software, porém um dos pontos mais fortes de GuideAutomator é em relação ao baixo esforço para se manter o documento de software atualizado e este importante tópico ficou de fora da realização do experimento realizado neste trabalho.

Dentre os pontos de melhoria no funcionamento do GuideAutomator Mobile detectados no experimento piloto com Android, observa-se a necessidade de se realizar uma nova conexão com o Appium toda vez que um manual é processado o que acaba elevando muito o tempo de processamento da ferramenta e gerando um período de inatividade do usuário enquanto aguarda a tarefa ser concluída, possivelmente este pode ser corrigido alterando-se as configurações do Appium para que ele possa manter uma sessão mais longas com o smartfone que estiver rodando os testes. A ausência de uma opção para desfazer e refazer operações a nível de blocos de entrada de texto no Jupyter Notebook,

também é um ponto de melhoria neste projeto, assim como foi percebido durante a realização do experimento, embora este problema só possa ser corrigido pela própria equipe de desenvolvimento do Jupyter Notebook este ainda é um ponto de melhoria na interface gráfica que foi escolhida.

Com o desenvolvimento e conclusão deste trabalho duas versões do GuideAutomator passam a existir o já conhecido GuideAutomator Web e o recém desenvolvido GuideAutomator Mobile, embora o desenvolvimento do GuideAutomator Mobile tenha sido feito com base em seu antecessor cada ferramenta tem suas características e semelhanças, por exemplo, GuideAutomator Web é um software em linha de comando desenvolvido em Node.Js, enquanto o GuideAutomator Mobile tem uma interface gráfica e foi desenvolvido em python, o custo para manter ambas as ferramentas acaba sendo elevado, tendo em vista que o esforço e conhecimento técnico necessário acaba sendo duplicado. Logo, o maior e principal desafio a frente é a realização de uma integração entre as duas soluções. Um possível solução para esse problema é a utilização do Appium como ponto de integração, o Appium realiza a automatização em diferentes dispositivos para cada um deles um diferente framework de automatização de tarefas é usado, por exemplo, para aplicações nativas em aparelhos Android é usado o UIAutomator, já pra para browsers sejam Android ou IOS é usado o Selenium WebDriver, o Selenium é mesma solução para automatização de tarefas usado no GuideAutomator Web, logo existe a possibilidade do Appium ser usado para automatizar tarefas na máquina na qual o GuideAutomator Mobile está executando, caso essa hipótese seja verdadeira, as demais funções existentes no Guide Automator Web podem ser migradas para Python e integradas ou GuideAutomator Mobile, gerando uma versão unificada das duas ferramentas.

Referências

[1] Michel dos Santos Soares. Metodologias Ágeis extreme programming e scrum para o desenvolvimento de software. *Revista Eletrônica de Sistemas de Informação*, 2004.

[2] Todd Waits and Joseph Yankel. Continuous system and user documentation integration. In *2014 IEEE International Professional Communication Conference (IPCC)*, pages 1–5. IEEE, 2014.

[3] Rodrigo Souza and Allan Oliveira. Guideautomator: continuous delivery of end user documentation. In *Proceedings of the 39th International Conference on Software Engineering: New Ideas and Emerging Results Track*, pages 31–34. IEEE Press, 2017.

[4] Allan Oliveira. Guideautomator: Automated user manual generation with markdown. *UFBA*, Abril 2016.

[5] Jerry Gao; H.-S. J. Tsao. *Testing and Quality Assurance for Component-based Software.* Artech House, 2003.

[6] Richard Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman.* Boston: GNU Press, Free Software Foundation, 2002.

[7] *ABNT.* Associação Brasileira de Normas Técnicas, 2005.

[8] Philip. Hodgson. Tips for writing user manuals. 2007.

[9] J. WAITS, T.; YANKEL. *Continuous system and user documentation integration.* IEEE., IEEE International Professional Communication Conference (IPCC)., 2014.

[10] Welbert Serra. *Avaliação experimental e melhoria do GuideAutomator, uma ferramenta para criação de manuais de usuário.* 2017.

[11] Wen Yin Ting Hasimah Hj Mohamed Muhammad Rafie Hj Mohd Arshad Ng Moon Hui, Liu Ban Chieng. Cross-platform mobile applications for android and ios. *IEEE*, Junho 2016.

A. Termo de Consentimento

"Avaliação Experimental do Guide-Automator Mobile para elaboração de documentação para usuário final"

Concordo em participar dos estudos não invasivos os quais serão conduzidos pelo Professor Rodrigo Rocha e pelo aluno de Graduação da UFBA Afonso Almeida da Silva, como parte da atividade de Término de Curso, realizada na UFBA. Esse estudo visa a comparar a eficiência do Guide-Automator sobre a elaboração de manuais para usuário final utilizando métodos tradicionais.

PROCEDIMENTO

Eu entendo que, uma vez o experimento terminado, os trabalhos que desenvolvi serão estudados visando a entender a eficiência do Guide-Automator na elaboração de manuais. Os pesquisadores conduzirão o estudo consistindo da coleta, análise e relato dos dados das atividades desenvolvidas. Eu entendo que não tenho obrigação alguma em contribuir com informação sobre meu desempenho na atividade, e que posso solicitar a retirada de meus resultados do experimento a qualquer momento e sem qualquer penalidade ou prejuízo. Eu entendo também que quando os dados forem coletados e analisados, meu nome será removido dos dados e que este não será utilizado em nenhum momento durante a análise ou quando os resultados forem apresentados.

CONFIDENCIALIDADE

Toda informação coletada neste estudo é confidencial, e meu nome não será identificado em momento algum. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto não terminar o estudo, bem como manter sigilo dos documentos apresentados e que fazem parte do experimento, e respeitar as regras declaradas no escopo do referido experimento.

BENEFÍCIOS e LIBERDADE DE DESISTÊNCIA

Eu entendo que os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado visando atender os requisitos do experimento, independentemente de participar ou não desse estudo, mas que os pesquisadores esperam aprender mais sobre quão eficiente é a utilização de tecnologias de software e os benefícios trazidos por esses estudos para o contexto da Engenharia de Software. Eu entendo que sou livre para realizar perguntas a qualquer momento ou solicitar que qualquer informação relacionada à minha pessoa não seja incluída no estudo. Eu entendo que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento de técnicas e processos para a Engenharia de Software.

Data: _____

Nome (em letra de forma): _____

Assinatura: _____

B. Perfil do Participante

Perfil do Participante

Este formulário tem como objetivo descrever o participante do experimento de avaliação do Guide Automator Mobile para realizar a produção de manuais para usuários finais em relação aos métodos tradicionais de produção de manuais.

* Required

1. **Informe o seu Nome** *

2. **Informe o seu E-mail** *

3. **Informe sua formação acadêmica**
   Mark only one oval.
   - ◯ Graduação em andamento
   - ◯ Graduação Completa
   - ◯ Pós-Graduação em andamento
   - ◯ Other: _____

4. **Curso de graduação em andamento ou concluído**
   Mark only one oval.
   - ◯ Ciência da Computação
   - ◯ Sistemas de Informação
   - ◯ Engenharia da Computação
   - ◯ Licenciatura em Computação
   - ◯ Other: _____

5. **Experiência Profissional**
   Mark only one oval.
   - ◯ Atuo ou já atuei na área de TI
   - ◯ Não Atuou profissionalmente na área de TI

6. **Qual é sua experiência com manuais de sistemas?**
   Mark only one oval.
   - ◯ Nunca escrevi um manual de usuário em um sistema
   - ◯ Já escrevi um manual de usuário para conhecimento próprio
   - ◯ Ja escrevi um manual de usuário para o âmbito acadêmico
   - ◯ Já escrevi um manual de usuário para o âmbito empresarial

7. **Qual é o seu grau conhecimento com Rich Text Editor (Microsoft Office e Libre Office Writer)?**
   Mark only one oval.

   - Nenhum conhecimento
   - Conhecimento básico
   - Conhecimento regular
   - Conhecimento avançado

8. **Qual é o seu grau de conhecimento com editores de Imagem?**
   Mark only one oval.

   - Nenhum conhecimento
   - Conhecimento básico
   - Conhecimento regular
   - Conhecimento avançado

9. **Qual é o seu grau de conhecimento com desenvolvimento front-end?**
   Mark only one oval.

   - Nenhum conhecimento
   - Conhecimento básico
   - Conhecimento regular
   - Conhecimento avançado

C. Avaliação Experimento

Avaliação Experimento

Questionário criado com a finalidade de permitir ao participante relatar suas experiências e avaliar o experimento realizado.

* Required

1. **Informe o seu nome** *

2. **Quais os problemas ou desvantagens você pode identificar no Guide Automator Mobile durante a realização do experimento?**

3. **Você enfrentou algum tipo de problema durante a realização do experimento? Se sim, quais?**

4. **Como você classifica o nível de dificuldade para realizar o manual do usuário usando o Appium e métodos tradicionais? (Sendo 1 muito fácil e 5 muito difícil).**
   Mark only one oval.

   - 1
   - 2
   - 3
   - 4
   - 5

5. **Como você classifica o nível de dificuldade para realizar o manual do usuário usando o Appium com GuideAutomator? (Sendo 1 muito fácil e 5 muito difícil).**
   Mark only one oval.

   - 1
   - 2
   - 3
   - 4
   - 5

6. **Você possui alguma crítica ou sugestão?**

Powered by Google Forms

D. Manual App Google Translate

In [1]:

```
from guide_mobile import *
```

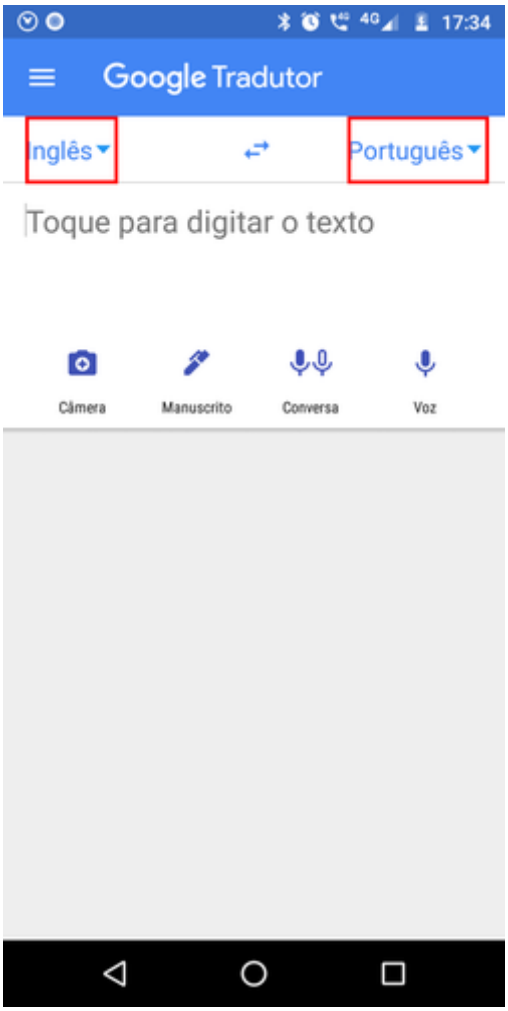
Manual Google Tradutor Android

A tela inicial do google translator é composta pelos seguintes elementos:

- **Idoma de entrada e saída** - Selecione os idiomas que serão usados para entrada e saída de dados.

In [2]:

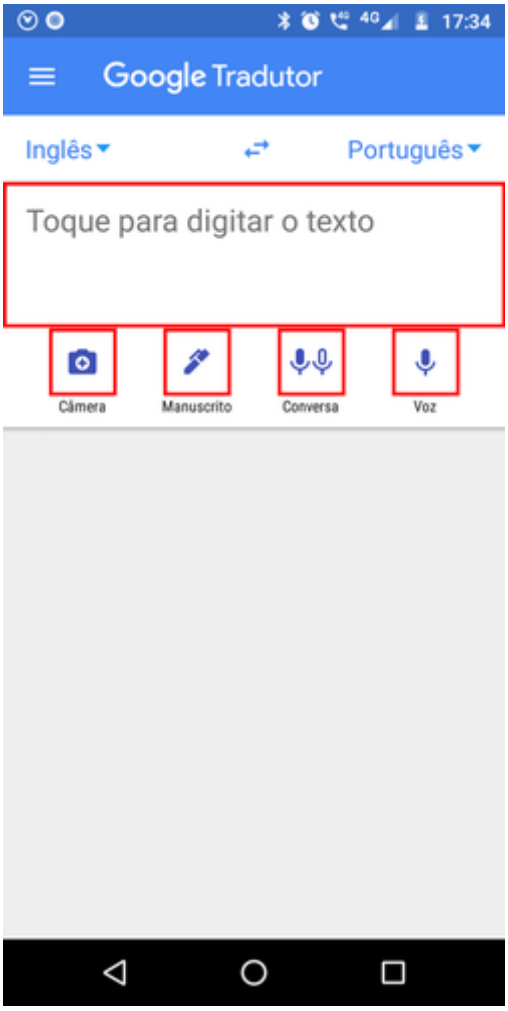
```
init()
highlightElementById('picker1, picker2')
```



- **Métodos de entrada** - Informe a frase a ser traduzida usando o teclado do seu celular ou um dos métodos destacados na imagem a seguir.

In [3]:

```
highlightElementById('lyt_home, btn_camera_icon, btn_handwriting_icon, btn_speech_icon, btn_dictation_icon')
```



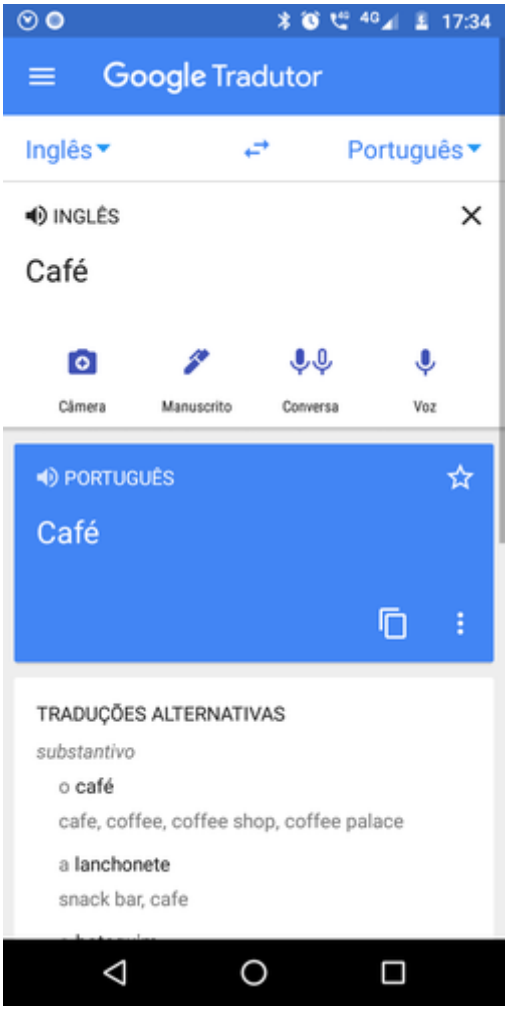
In [4]:

```
clickById('touch_to_type_text')
sleep(2)
clickById('edit_input')
sendKeysById('edit_input','Café')
sleep(2)
clickById('result_selector')
```

- **Saída de dados** - Veja a sua tradução no quadro inferior.

In [5]:

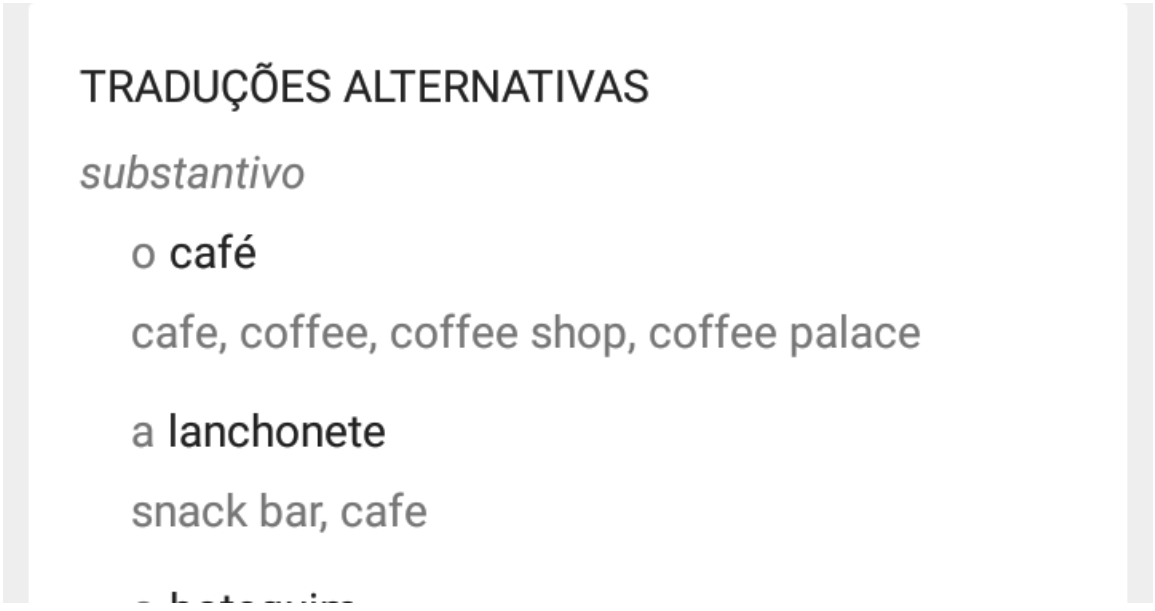
```
sleep(1)
takeScreenshot()
```



- **Traduções Alternativas** - Veja outras traduções para sua expressão no painel infeior da tela.

In [6]:

```
takeScreenshotElementById('dictionary_container')
```



- **Recursos Adicionais** - Você também pode acessar o menu da aplicação para ter acesso a mais recursos.

In [7]:

```
highlightElementByAccessibilityId('Abrir gaveta de navegação')
```