

Automação de Testes para Aplicações Mobile na DEPAE

Léo Roberto Knetig

Projetista de Teste

Silton Menezes Sommer

Projetista de Teste



Objetivos

- Agilidade nos testes funcionais em dispositivos móveis
- Melhorar qualidade dos produtos
- Implementar automação de testes para o sistema SINESP Cidadão disponível nas plataformas Android, iOS e Windows Phone



Testes Mobile - Execução do Teste na Prática



Testes Mobile - Como fizemos isso?

- Android SDK
- Appium
- Robot Framework
- Mapear Elementos
- Links

Testes Mobile - Pré-Requisitos de Execução

- Máquina virtual Android com uso do AVD Manager(ferramenta do Android SDK)
- Servidor do Appium executando
- Executar o script de teste propriamente dito



Testes Mobile - Android SDK

Necessário para executar o emulador do Android

- Android Studio
- SDK Manager
- AVD Manager
- UIAutomatorViewer



Testes Mobile - Appium

- Ferramenta de código aberto para automatizar aplicações Nativas, web para mobile e híbridas em plataformas iOS e Android
- Appium é "multiplataforma": permite que você escreva testes contra várias plataformas (iOS e Android), usando a mesma API = Reutilização de código entre iOS e Android (mesmo teste em plataformas diferentes)

Execução de testes – Necessário ter instalado:

- Appium Server
- Appium Client



Testes Mobile - Appium

Pré-Requisitos para execução dos testes (conforme site Appium.io):

iOS

- Mac OSX 10.7+
- XCode 4.5+ w/ Command Line Tools

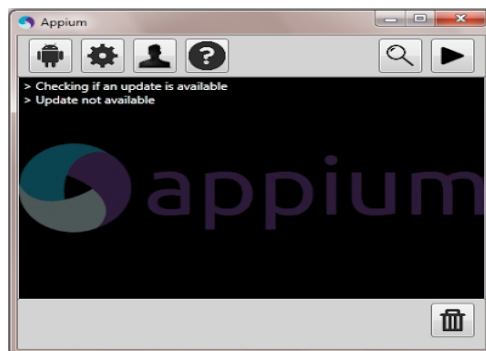
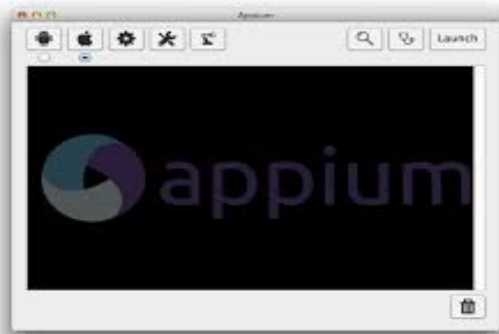
Android

- Mac OSX 10.7+ or Windows 7+ or Linux
- Android SDK ≥ 16 (SDK < 16 in Selendroid mode)

Testes Mobile - Appium

Appium Server

- Possui versões para Windows, Mac e Linux
- No Windows e Mac – Instalador e Interface Gráfica para download disponível no site do Appium



- Linux : necessário possuir pacotes node.js e npm e instalar via linha de comando: `$ npm install -g appium` `$ appium`
- Interface para linux somente em linha de comando

```
93368216015@serpro-1563032:~$ appium
[Appium] Welcome to Appium v1.5.3
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```



Testes Mobile - Appium

Appium Client

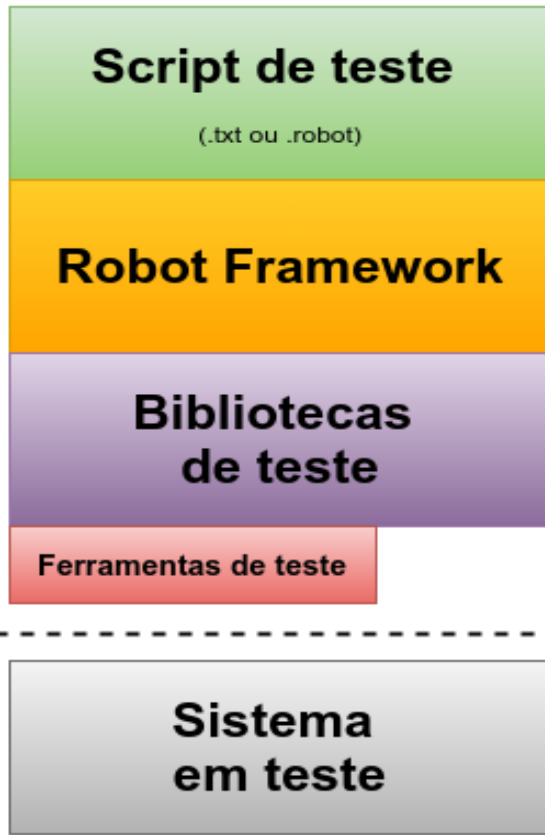
- Ruby
- Python
- Java
- JavaScript
- PHP
- C#
- **RobotFramework**

Testes Mobile - Robot Framework

- É um framework de automação genérico
- Pode ser usado com aplicativos web, móvel , desktop, SSH, FTP
- Pode ser executado com o Python ou com Java
- A sintaxe fácil e orientado a palavras-chave.
- Permite criação de palavras chaves reutilizáveis
- Fornece relatórios no formato HTML
- Arquitetura modular



Testes Mobile - Robot Framework



Palavra Chave

Elemento localizador

Exemplos:

- Input Text id=name Joao da Silva
- Click Button Salvar

Testes Mobile - Robot Framework - Algumas Bibliotecas

- bibliotecasrobotframework-faker 3.0.0 - Geração de palavras aleatória
- **AppiumLibrary - Teste IOS e Android**
- AutoITLibrary - Teste aplicações Windows
- Database Library - Teste de database
- ImageHorizon - Teste de reconhecimento de imagem
- Swing Library - Teste de interface local e remota em Swing
- Selenium2library - Teste Web via Selenium2
- SSHLibrary - Execução de comandos SSH
- FTP Library - Execução de comandos FTP



Robot Framework - AppiumLibrary - Algumas Keywords

Open Application

Input Text <identificador>

Click Button

Sleep <tempo>

Press Keycode <keycode>

Title Should Be <texto>

Shake

Close Application

Lock

Click Element <identificador>

Set Network Connection Status

Reset Application

Go To Url <URL>

Hide Keyboard

Wait Until Page contains <texto>

zoom <locator><percentual>

Close All Applications

Captura Page Screenshot



Testes Mobile - Robot Framework - Script .robot

*** Settings

Importar bibliotecas

Importa arquivos de variáveis

*** Variables

Criar variáveis

*** Test Cases

Criar casos de teste com palavras chaves existentes available keywords.

*** Keywords

Criar palavras chave personalizadas



Testes Mobile - Robot Framework - Exemplo

*** Settings ***

Library AppiumLibrary

*** Test Cases ***

Teste inicio

#Para Android

Open Application http://localhost:4723/wd/hub platformName=Android
platformVersion=4.2.2 deviceName=localhost:5555 app=\$
{CURDIR}/teste/TesteApp.apk appPackage=teste.demo
appActivity=MainActivity

#Para iOS

Open Application http://localhost:4723/wd/hub alias=Myapp1
platformName=iOS platformVersion=9.0.1 deviceName='iPhone' app=sua
aplicacao.app



Testes Mobile - Robot Framework - Exemplo

*** Settings ***

Library AppiumLibrary

*** Test Cases ***

Teste inicio

Open Application remote_url=http://localhost:4723/wd/hub platformName=Android
platformVersion=5.1.1 deviceName=http://localhost:5554
app=C:/APK/Sinesp_Cidadao_3110_1710.apk

Wait Until Page Contains Element id=br.gov.sinesp.cidadao.android:id/botoes timeout=30

Pesquisar Veiculo Encontrado

Input Text id=br.gov.sinesp.cidadao.android:id/txPlacaLetra IQF

Input Text id=br.gov.sinesp.cidadao.android:id/txPlacaNumero 0820

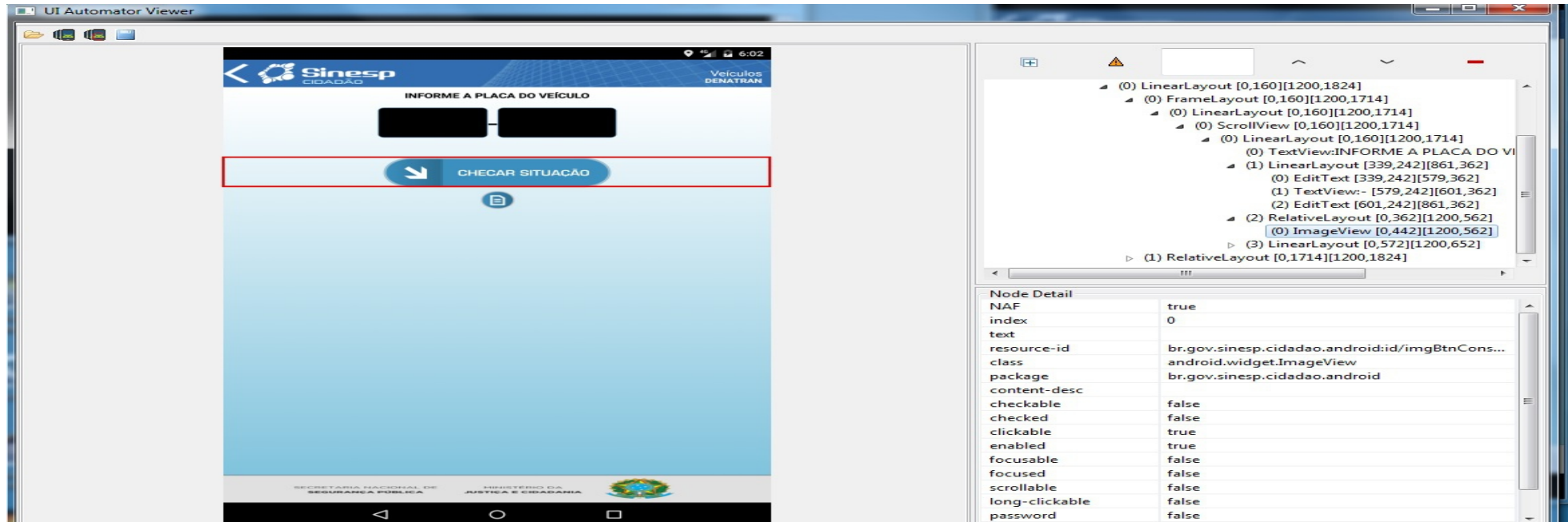
Click element id=br.gov.sinesp.cidadao.android:id/imgBtnConsultar

wait until page contains element id=br.gov.sinesp.cidadao.android:id/imgBtnSituacaoLegal

Click element id=br.gov.sinesp.cidadao.android:id/btnNovaConsulta

Testes Mobile - Mapear Elementos ?

- Dúvida comum ao automatizar sistemas não web
- No caso do Android: UIAutomatorViewer
- Para o iOS a ferramenta equivalente chama-se Accessibility Inspector



The screenshot displays the UIAutomatorViewer application. On the left, a mobile app interface is shown with a red box highlighting a button labeled "CHECAR SITUÇÃO". The app interface includes a header with "Sinesp CIDADÃO" and "Veículos DENATRAN", a form for "INFORME A PLACA DO VEÍCULO", and a footer with "SECRETARIA NACIONAL DE SEGURANÇA PÚBLICA" and "MINISTÉRIO DA JUSTIÇA E CIDADANIA".

On the right, the UI hierarchy tree is visible, showing the following structure:

```
(0) LinearLayout [0,160][1200,1824]
├── (0) FrameLayout [0,160][1200,1714]
│   ├── (0) LinearLayout [0,160][1200,1714]
│   │   └── (0) ScrollView [0,160][1200,1714]
│   │       └── (0) LinearLayout [0,160][1200,1714]
│   │           ├── (0) TextView:INFORME A PLACA DO VI
│   │           ├── (1) LinearLayout [339,242][861,362]
│   │           │   ├── (0) EditText [339,242][579,362]
│   │           │   ├── (1) TextView:- [579,242][601,362]
│   │           │   └── (2) EditText [601,242][861,362]
│   │           └── (2) RelativeLayout [0,362][1200,562]
│   │               └── (0) ImageView [0,442][1200,562]
│   └── (3) LinearLayout [0,572][1200,652]
└── (1) RelativeLayout [0,1714][1200,1824]
```

Below the tree, the Node Detail section shows the following properties for the selected node:

NAF	true
index	0
text	
resource-id	br.gov.sinesp.cidadao.android:id/imgBtnCons...
class	android.widget.ImageView
package	br.gov.sinesp.cidadao.android
content-desc	
checkable	false
checked	false
clickable	true
enabled	true
focusable	false
focused	false
scrollable	false
long-clickable	false
password	false



Testes Mobile - Próximos Passos

- Integração contínua com Jenkins (em tratativas)
- Realização dos Testes em dispositivos com IOS
- Avaliar Testes em dispositivos com Windows Mobile



Links

- <http://voce.serpro/robot-framework/blog>
- <http://voce.serpro/robot-framework/blog/introducao-ao-uso-do-robot-para-aplicacoes-mobileandroid-e-ios-com-uso-do-appium>
- <http://appium.io/>
- <http://robotframework.org/>
- <http://jollychang.github.io/robotframework-appiumlibrary/doc/AppiumLibrary.html>
- <https://developer.android.com/studio/index.html?hl=pt-br>
- <https://developer.apple.com/xcode/>



Dúvidas?

Agradecemos pela atenção.

Léo Knetig

leo.knetig@serpro.gov.br

• #51 1226

Silton Sommer

silton.sommer@serpro.gov.br

#51 1183