Testes funcionais de aplicações (mobile) Android com Appium

Professor: Francisco Wagner Costa Aquino

Principais ferramentas utilizadas durante o curso:

Java (JDK/SE 8), Eclipse, Android, Emulador, Appium

JAVA JDK versão 8+

Configurar nas variáveis de ambiente do sistema.
User Variables e System Variables
JAVA_HOME
VALOR DA VARIÁVEL = C:\Program Files\Java\jdk1.8.0_241

System Variables / Path
%JAVA_HOME%\bin;

Android Studio / Instalar SDKs

Android SDK instalação (dentro do android studio)
(ícone sdk manager, no canto superior direito da tela, um cone com uma flecha para baixo, na aba SDK TOOLS)

- Android SDK Build-Tools
- Android Emulator
- Android SDK Tools
- Android SDK Platform-Tools

Configurar as variáveis de ambiente
System Variables
ANDROID_HOME
Valor da variável = C:\Users\caio\AppData\Local\Android\Sdk

Path
%ANDROID_HOME%\tools;
%ANDROID_HOME%\tools\bin;
%ANDROID_HOME%**platform-tools**;

Para testar se foi tudo configurado corretamente, no cmd (command prompt), executar Adb, emulator (no avd specified..) e uiautomatorviewer.

//adicionar o inspetor de elementos do appium nas variáveis.
%ANDROID_HOME%\tools\bin\uiautomatorviewer

Qualquer erro com execução do teste, por não encontrar o device, talvez precise deixá-lo "ativo", para isso basta testar dentro da pasta platform-tools o comando adb devices 2x

Criando Emulador no Android Studio

Novo projeto / template em branco

Clicar em AVD Manager
(ícone no canto superior direito da tela, um celular com um android)

Criar novo device Nexus 5 android versão mais recente.

Device Name Nexus 5 API 28

(como o android studio e o emulador consomem muitos recursos, vamos inicializar o emulador pela linha de comando no cmd)

Listar nomes dos devices criados:
emulator -list-avds
Entrar na pasta da SDK %ANDROID_HOME%\tools ou %ANDROID_HOME%\emulator

Iniciar emulador: emulator @nome do emulador

Download e instalação do Appium

https://github.com/appium/appium-desktop/releases
Fazer download da versão mais recente.

Start Server deve ser 1.7.2

Clicar na lupa para editar um Custom Server, em Capabilites, vamos editar as capacidades. Isso é informar detalhes de qual aparelho vamos testar.

platformName = (Text) Android

deviceName = (qualquer coisa) porém vamos colocar o nome do dispositivo que está conectado.

Para isso no cmd, entrar na pasta caminho cd %ANDROID_HOME%/platform-tools e digitar **adb devices**, aparecerá o emulator-5554. (Com o aparelho sendo apresentado na tela do windows).

*** Qualquer problema com o adb devices, como estado offline

Try **adb kill-server** and then **adb start-server**. Run **adb devices** to make sure your emulator is in the list and you should be set.

automationName = (Text) uiautomator2

- Instalar/arrastar o arquivo (.apk) que deseja realizar o teste dentro do emulador para que seja instalado.
- Entrar em APK info no emulador
- Procurar pela “calculadora”
- Dar um clique longo até aparecer as opções e entrar em detalhes
- Esses dados em activities serão usados no appPackage e appActivity

appPackage = (Text) com.android.calculator2

appActivity = (Text) com.android.calculator2.Calculator

Json Representation

```
{
  "platformName": "Android",
  "deviceName": "emulator-5554",
  "automationName": "uiautomator2",
  "appPackage": "com.android.calculator2",
  "appActivity": "com.android.calculator2.Calculator"
}
```

Se clicar em Start Session, irá iniciar a conexão entre o Appium e o aparelho que está sendo emulado. O Appium irá mostrar a ferramenta de inspeção e ação para realizar o teste.

Você pode selecionar os elementos do aplicativo na tela do emulador via Appium, lá será exibido detalhes desse elemento, como id, xpath.. Etc.

Você poderá realizar uma ação com esse elemento selecionado, por exemplo Tap (clicar), enviar dígitos (send keys), e limpar (clear).

Para gravar uma ação (teste), é preciso:

Clicar no ícone com forma de olho (Start Recording), selecionar o elemento e fazer uma ação (clicar ou digitar). Exemplo 2 + 2 = 4.
Clicar no campo que aparece o resultado 4 e clicar para que o código reconheça aquele campo.

```
MobileElement el3 = (MobileElement)
driver.findElementById("com.android.calculator2:id/digit_2");
el3.click();
MobileElement el4 = (MobileElement) driver.findElementByAccessibilityId("plus");
el4.click();
MobileElement el5 = (MobileElement)
driver.findElementById("com.android.calculator2:id/digit_2");
el5.click();
MobileElement el6 = (MobileElement)
driver.findElementById("com.android.calculator2:id/result");
el6.click();
```

Para aparecer o código completo do teste, precisa apertar no ícone:

(Show/Hide boilerplate Code)

```
import io.appium.java_client.MobileElement;
import io.appium.java_client.android.AndroidDriver;
import junit.framework.TestCase;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import java.net.MalformedURLException;
import java.net.URL;
import org.openqa.selenium.remote.DesiredCapabilities;

import static org.junit.Assert.*;

public class SampleTest {
```

```
        //private AndroidDriver driver;
        private AndroidDriver<MobileElement> aplicativo;

    @Before
    public void setUp() throws MalformedURLException {
        DesiredCapabilities desiredCapabilities = new DesiredCapabilities();
        desiredCapabilities.setCapability("platformName", "Android");
        desiredCapabilities.setCapability("deviceName", "emulator-5554");
        desiredCapabilities.setCapability("automationName", "uiautomator2");
        desiredCapabilities.setCapability("appPackage", "com.android.calculator2");
        desiredCapabilities.setCapability("appActivity", "com.android.calculator2.Calculator");

        URL remoteUrl = new URL("http://127.0.0.1:4723/wd/hub");

        //aplicativo = new AndroidDriver(remoteUrl, desiredCapabilities);
        aplicativo = new AndroidDriver<MobileElement>(remoteUrl, desiredCapabilities)

    }

    @Test
    public void sampleTest() {
        MobileElement el3 = aplicativo.findElementById("com.android.calculator2:id/digit_2");
        el3.click();

        //AccesbilityId é um método especialmente criado para indentificar elementos no mobile.
        MobileElement el4 = aplicativo.findElementByAccessibilityId("plus");
        el4.click();
        MobileElement el5 = aplicativo.findElementById("com.android.calculator2:id/digit_2");
        el5.click();
        MobileElement el6 = aplicativo.findElementById("com.android.calculator2:id/result");
        el6.click();
    }

    @After
    public void tearDown() {
        aplicativo.quit();
    }
}
```

Iniciando um Novo Projeto no Eclipse ou outro framework.

Maven Project / Create a simple project

Artifact
Group Id = br.com.caioandrian.appium
Artifact Id = webdriver-java-appium-mobile

Projeto criado.

Caso necessário, alterar a Java Build Path para Java SE versão 8+ em propriedades do projeto.

Adicionar **dependências** no pom.xml

```
<!-- https://search.maven.org/artifact/io.appium/java-client/7.3.0/jar -->
<dependency>
	<groupId>io.appium</groupId>
	<artifactId>java-client</artifactId>
	<version>7.3.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
	<groupId>junit</groupId>
	<artifactId>junit</artifactId>
	<version>4.13</version>
	<scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
	<groupId>org.seleniumhq.selenium</groupId>
	<artifactId>selenium-java</artifactId>
	<version>3.141.59</version>
</dependency>

<dependency>
	<groupId>org.apache.commons</groupId>
	<artifactId>commons-io</artifactId>
	<version>1.3.2</version>
```

```
        <scope>test</scope>
</dependency>

<!-- https://mvnrepository.com/artifact/org.easetech/easytest-core -->
<dependency>
        <groupId>org.easetech</groupId>
        <artifactId>easytest-core</artifactId>
        <version>1.4.0</version>
</dependency>
```

Criar a classe de teste no projeto, podendo seguir o exemplo do código gerado pelo Appium, com as capacidades, @before, @test, e @after.

- **Para executar o teste é preciso inicializar o Appium Server e o Emulador criado no android studio!!**

Como fazer Testes com um Celular Físico (Real):

Ativar a opção de DESENVOLVEDOR no aparelho do emulador (em configurações/about emulator/clicar várias vezes no build number).
Manter ativado as opções de desenvolvedor.
Ativar a opção Depuração USB, caso não esteja ativado.

Campo de Treinamento Appium (CTAppium.apk)

Como instalar uma apk no emulador diretamente pelo Appium?

Colocar a apk dentro do projeto, na pasta resources.
No projeto, vamos alterar as linhas appPackage e AppActivity.

```
desiredCapabilities.setCapability(MobileCapabilityType.APP,
"C:\Users\caio\IdeaProjects\webdriver-java-appium-mobile\src\test\resources\CTAppium_1_0.apk”;
```

- No caso se da instalação em celulares reais, é preciso habilitar a opção permitir apk de fontes desconhecidas.

Inspecionando Elementos na tela do aparelho mobile

No cmd, vamos iniciar o **uiautomatorviewer.**
(ele é mais leve do que criar uma sessão dentro do Appium!!!)

Adicionar no Path das variáveis de ambiente do sistema:
%ANDROID_HOME%\tools\uiautomatorviewer

Passo 1: Abrir o aplicativo no emulador
Passo 2: Tirar uma screenshot da tela naquele momento (segundo ícone).

- Para cada tela que for analisar é preciso tirar uma nova screenshot, após ter entrado nessa tela via emulador!!!

!!! Se estiver travando na hora de rodar o teste, desativar o plugin Klotin pode ajudar. No intellij o atalho é Ctrl + Alt + S.

```
//Adicionar uma espera implícita pode ajudar antes de acessar o aplicativo.
aplicativo.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

Diferença entre o xpath do Selenium e no Mobile

No Selenium fica **By.xpath.(“//tag[@atributo='valor']”)**
No Mobile fica **By.xpath(“//class[@atributo='']”)**, exemplo:
//android.widget.TextView[@text='Formulário']

**** Atenção para o detalhe, no mobile usa-se a Classe e não a tag do elemento!**

Pegando o Id do Elemento no mobile

Podemos pegar o id do elemento mobile, através do **MobileBy.AccessibilityId() e passando o valor do elemento com content-desc**!!

Exemplo:
MobileElement campoNome = aplicativo.findElement(**MobileBy.AccessibilityId**("nome"));

Navegação pela árvore de elementos com xpath

aplicativo.findElement(**By.xpath("//android.widget.Spinner/android.widget.TextView")**);

```
//classe do primeiro elemento / classe do segundo elemento
```

Campos Switch(Ativo ou Inativo) e CheckBox

```
//status do checked deve vir como desmarcado
assertTrue(check.getAttribute("checked").equals(“false”));

//status do switch deve vir como marcado
assertTrue(elementoSwitch.getAttribute("checked").equals(“true”));
```

Organização do Código com @Before, @After

```
@Before
public void setUp() throws MalformedURLException {
        aplicativo = DriverFactory.getDriver();

        //tudo que será executado repetidamente nos testes
}

@After
public void tearDown() {
        DriverFactory.killDriver();
}
```

Criando apenas uma instância do driver / Driver Centralizado

(na pacote Suporte - classe DriverFactory)

```
public static AndroidDriver<MobileElement> getDriver()  {
      if (aplicativo == null){
         createDriver();
      }

      return aplicativo;
}

private static void createDriver() {
      DesiredCapabilities desiredCapabilities = new DesiredCapabilities();
      desiredCapabilities.setCapability("platformName", "Android");
```

```
        desiredCapabilities.setCapability("deviceName", "emulator-5554");
        desiredCapabilities.setCapability("automationName", "uiautomator2");


       //instalando o arquivo diretamente no emulador usando o Appium
        desiredCapabilities.setCapability(MobileCapabilityType.APP,
"C:\\Users\\caio\\IdeaProjects\\webdriver-java-appium-mobile\\src\\test\\resources\\CTAppium_1
_0.apk");

        try {
            URL remoteUrl = new URL("http://127.0.0.1:4723/wd/hub");
            aplicativo = new AndroidDriver(remoteUrl, desiredCapabilities);
        }catch (MalformedURLException e){
            e.printStackTrace();
        }

        //espera implicita
        aplicativo.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
}

public static void killDriver(){
        if(aplicativo != null){
            aplicativo.quit();
            aplicativo = null;
        }
}
```

Criando a classe DSL (depois será a nossa BasePage)

Classe no pacote **Suporte** para criar **métodos genéricos**

Vantagens!!!
Facilita uma possível manutenção em testes que utilizam o mesmo campo, botão, ou elemento na tela.

```
public void escrever(By by, String texto){
        DriverFactory.getDriver()
            .findElement(MobileBy.AccessibilityId("nome")).sendKeys(texto);
}

public String obterTexto(By by){
            return DriverFactory.getDriver().findElement(by).getText();
```

```
}

public void selecionarCombo(By by, String texto){
	DriverFactory.getDriver().findElement(by).click();
	clicarOpcaoDoCombo(texto);
}

public void clicar(By by){
	DriverFactory.getDriver().findElement(by).click();
}

public void clicarPorTexto(String valor){
	clicar(By.xpath("//*[@text='"+ valor +"']"));
}

public boolean isCheckMarcado(By by){
	return DriverFactory.getDriver().findElement(by).getAttribute("checked").equals("true");
}
```

Na classe de teste:

```
	private DSL dsl = new DSL();

	//Escrever nome
	dsl.escrever(MobileBy.AccessibilityId("nome"), "Caio");

	//validar nome escrito
	assertEquals("Caio", dsl.obterTexto(MobileBy.AccessibilityId("nome")));

	//clicar e selecionar a opção do Combo
	dsl.selecionarCombo(MobileBy.AccessibilityId("console"), "Nintendo Switch");

	//validar a opcao selecionada
	String texto =
dsl.obterTexto(By.xpath("//android.widget.Spinner/android.widget.TextView"));
	assertEquals("Nintendo Switch", texto);

	//status do checked deve vir como desmarcado
	assertFalse(dsl.isCheckMarcado(By.className("android.widget.CheckBox")));

	//status do switch deve vir como marcado
	assertTrue(dsl.isCheckMarcado(MobileBy.AccessibilityId("switch")));
```

```
        //clicar nos elementos
        dsl.clicar(By.className("android.widget.CheckBox"));
        dsl.clicar(MobileBy.AccessibilityId("switch"));

        //verificar estados alterados
        assertTrue(dsl.isCheckMarcado(By.className("android.widget.CheckBox")));
        assertFalse(dsl.isCheckMarcado(MobileBy.AccessibilityId("switch")));

        //preencher campos
        dsl.escrever(MobileBy.AccessibilityId("nome"), "Caio");
        dsl.clicar(MobileBy.AccessibilityId("console"));
        dsl.clicar(By.xpath("//android.widget.CheckedTextView[@text='PS4']"));

        //salvar
        dsl.clicar(By.xpath("//android.widget.TextView[@text='SALVAR']"));

        //validar campos
        String nome = dsl.obterTexto(By.xpath("//android.widget.TextView[@text='Nome: Caio']"));
        assertEquals("Nome: Caio", nome);

        String console = dsl.obterTexto(By.xpath("//android.widget.TextView[@text='Console: ps4']"));
        assertEquals("Console: ps4", console);
```

Trabalhando com Page Object

Resolver problemas de manutenção no teste, por exemplo:
Um novo elemento switch entrou na mesma página ou formulário, por padrão pegamos a classe (css) do elemento, e na época só havia um 1 elemento switch!

Insert Test	**Form Page**	**Form**
	List Page	**List**

!!! Classes Page não devem ter assert.
!!! Classes Test nao devem ter .. ?

Pacote page

Classe MenuPage.java

```
	private DSL dsl = new DSL();

	public void acessarFormulario(){
		dsl.clicarPorTexto("Formulário");
	}
```

Classe FormularioPage.java

```
	private DSL dsl = new DSL();

	public void escreverNome(String nome){
		dsl.escrever(MobileBy.AccessibilityId("nome"), nome);
	}

	public String obterNome(){
		return dsl.obterTexto(MobileBy.AccessibilityId("nome"));
	}



	public String obterNomeCadastrado(){
		return dsl.obterTexto(By.xpath("//android.widget.TextView[starts-with(@text,
	'Nome:')]"));
	}

	Etc…….
```

Pacote tests

```
private MenuPage menu = new MenuPage();
private FormularioPage formularioPage = new FormularioPage();

//Selecionar a opcao formulario
menu.acessarFormulario();

//preencher campos
formularioPage.escreverNome("Caio");
formularioPage.selecionarCombo("PS4");

//salvar
formularioPage.salvar();

//validar campos
```

```
assertEquals("Nome: Caio", formularioPage.obterNomeCadastrado());
assertEquals("Console: ps4", formularioPage.obterConsoleCadastrado());
assertTrue(formularioPage.obterCheckBoxCadastrado().endsWith("Desabilitado"));
assertTrue(formularioPage.obterSwitchCadastrado().endsWith("On"));
```

Herdando Comportamentos

Vamos excluir a classe DSL, pois criamos uma nova classe chama **BasePage (no pacote page)** com os métodos que serão usados mais de uma vez, e são genéricos a todas as páginas.

Agora a classe **FormulárioPage** irá extender **(extends)** a **BasePage.**

Na classe **FormulárioTest** vamos extender a **BaseTest.**

Motivo: Em todos os testes, iremos encerrar o driver, então apenas estendemos a classe base para não precisar repetir essa parte do código na classe de teste.

```
public class BaseTest {
	@After
	public void tearDown() {
		DriverFactory.killDriver();
	}
}
```

Resetando a Aplicação (.resetApp() e @AfterClass)

Vamos configurar nossa classe de teste para que **utilize apenas uma sessão do driver** para cada novo teste. Dessa forma todos os testes de formulário (uma classe de teste) será executado na mesma sessão e encerrando a sessão após finalizar os testes dessa classe com a anotação **@AfterClass**.

***** Porém é recomendado que a cada novo teste seja totalmente independente da anterior, para não haver possibilidades de interferência no processo de teste.**

*** resolver problema de **updating indices no intellij**
delete 'caches' folder in user/.IntelliJIdea13/system/

No classe **BaseTest**

```
@AfterClass
public static void finalizarClasseDeTeste() {
	DriverFactory.killDriver();
}

@After
public void tearDown() {
	//DriverFactory.killDriver();
	DriverFactory.getDriver().resetApp();
}
```

Tirando screenshot do teste

Na classe **BaseTest**

```
@Rule
public TestName testName = new TestName();

public void gerarScreenshot(){
	String dataHora = Generator.dataHoraParaArquivo();

	String screenshotArquivo = "screenshots/"
		+  testName.getMethodName() + "_" + dataHora +".png";

	Screenshot.tirar(DriverFactory.getDriver(), screenshotArquivo);
}
```

Nova classe chamada Generator no pacote suporte

```
public static String dataHoraParaArquivo(){
	Timestamp ts = new Timestamp(System.currentTimeMillis());
	return new SimpleDateFormat("ddMMyyyyHHmmss").format(ts); //20200124194200
}
```

Nova classe chamada Screenshot no pacote suporte

```
public static void tirar(WebDriver navegador, String arquivo){
	File screenshot = ((TakesScreenshot) navegador).getScreenshotAs(OutputType.FILE);
	try {
```

```
            FileUtils.copyFile(screenshot, new File(arquivo));
        } catch (Exception e){
            System.out.println("Houveram problemas ao copiar o arquivo para a pasta " +
e.getMessage());
        }
}
```

Espera Implícita x Espera Explícita (thread.sleep)

Na espera implícita, dizemos o **tempo máximo que sistema pode esperar até que se encontre algum elemento que queremos na tela**.

```
//esperar no máximo 10 segundos.
DriverFactory.getDriver().manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

A **espera explícita** **não é recomendado**, pois pode atrasar os testes que estamos fazendo.

```
try {
        Thread.sleep(tempo);
}catch (InterruptedException e) {
        e.printStackTrace();
}
```

Utilizando o WebDriverWait

Esse é o **método mais recomendado** para se esperar elementos na tela, seja mobile ou do navegador.

```
WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(), 10);

wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//*[@text = 'Nome:
Caio']")));
```

Elemento Splash

Aquele elemento que aparece por um breve período na tela e depois o mesmo desaparece.

Verificar se o elemento splash apareceu na tela:

```
public boolean existeElementoPorTexto(String texto){
```

```
        List<MobileElement> elementos =
DriverFactory.getDriver().findElements(By.xpath("//*[@text= '"+ texto +"']"));

            return elementos.size() > 0;
}
```

Aguardando o elemento splash sumir da tela:

```
WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(), 10);
wait.until(ExpectedConditions.invisibilityOfElementLocated(By.xpath("//*[@text='Splash!']")));
```

Elemento Alert

Obtendo o valor de cada texto pelo resource-id

Exemplo:

```
            obterTexto(By.id("android:id/alertTitle"));
            obterTexto(By.id("android:id/message"));
```

Elemento Tabs (Abas)

```
public boolean isEnabled(By by){
            return DriverFactory.getDriver().findElement(by).getAttribute("enabled").equals("true");
}

public boolean aba1IsEnabled(){
            return isEnabled(By.xpath("//android.widget.TextView[@text = 'ABA 1']"));
}

//verificar que esta na aba 2
assertTrue(abasPage.aba1IsEnabled());

public String obterTextoConteudoDaAba(){
            return
obterTexto(By.xpath("//android.support.v4.view.ViewPager//android.widget.TextView"));
}

//verificar que mensagem na aba 2
assertEquals("Este é o conteúdo da Aba 2", abasPage.obterTextoConteudoDaAba());
```

Elemento Accordion

```
public String obterTextoDescricaoOpcao1(){
      return obterTexto(
By.xpath( "//*[@text='Opção 1']/../../following-sibling::android.view.ViewGroup
//android.widget.TextView" ));
   }
```

Explicando o xpath usado:
Uma barra (/) pode navegar apenas um elemento, **enquanto duas barras (//)** pode navegar mais elementos de uma vez só.

/../../ irá voltar dois elementos, é igual aos comandos no cmd.

/following-sibling::android.view.ViewGroup irá procurar o próximo elemento da classe ViewGrupo, que esteja no mesmo nível (“na mesma camada”) dele.

```
//espera explicita de 1seg pois o accordion possui uma animacao para abrir o texto
//thread.sleep()..
esperar(1000);

//verificar se o texto apresentado é igual o esperado
assertEquals("Esta é a descrição da opção 1", pagina.obterTextoDescricaoOpcao1());
```

***** Tomar cuidado com Accordion, e/ou qualquer outros elementos que usam animações, pois eles podem quebrar o código de testes. Usar esperas explícitas.**

Elemento DatePicker

```
formularioPage.clicarPorTexto("01/01/2000");
formularioPage.clicarPorTexto("20");

//formularioPage.clicarPorTexto("OK");
formularioPage.clicar(By.id("android:id/button1"));

assertTrue(formularioPage.existeElementoPorTexto("20/2/2000"));

public boolean existeElementoPorTexto(String texto){
      List<MobileElement> elementos =
DriverFactory.getDriver().findElements(By.xpath("//*[@text= '"+ texto +"']"));
      return elementos.size() > 0;
}
```

Elemento TimePicker

Através do uiautomatorviewer, é possível verificar que para o **timepicker todo elemento na tela possui um content-desc,** ou seja um ID único que podemos usar o AcessibilityId criado especialmente para Mobile.

```
//clicar no texto com a data
formularioPage.clicarPorTexto("06:00");

//selecionar horas
formularioPage.clicar(MobileBy.AccessibilityId("22"));

 //selecionar minutos
 formularioPage.clicar(MobileBy.AccessibilityId("30"));


//clicar em OK
formularioPage.clicarPorTexto("OK");

//validar se o horario foi alterado
assertTrue(formularioPage.existeElementoPorTexto("22:30"));
```

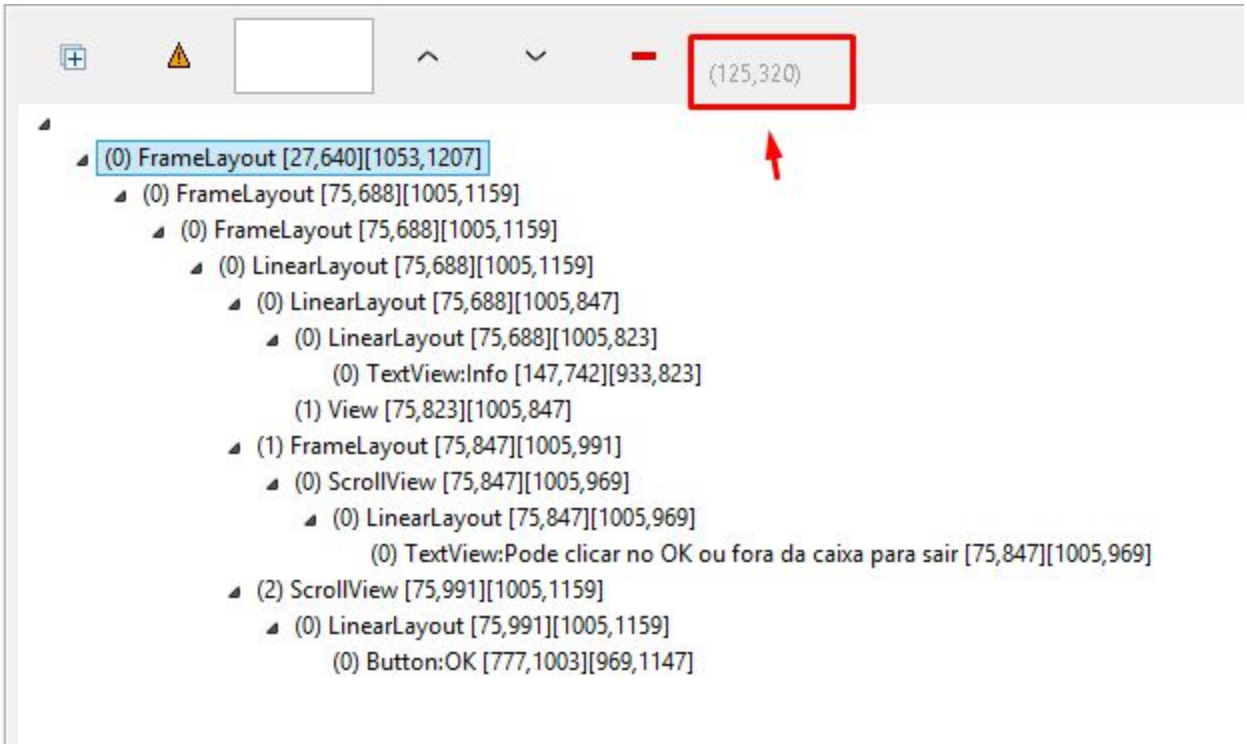
Como clicar em uma coordenada específica

Ação de toque, independentemente se possui elemento clicável na tela ou não.

Exemplos utilizados, alertas com fundo bloqueado (alerta só fecha se clicar em um local específico) e altera com fundo liberado (pode fechar clicando em qualquer lugar).

No uiautomatorviewer , é possível ver a coordenada que estamos usando ao arrastar o mouse na tela do dispositivo. * Apenas quando o uiautomatorviewer estiver com a tela maximizada**

Exemplo de coordenada na foto: 125,320.



Na classe **BasePage** no pacote page.

```
public void tap(int x, int y){
        new TouchAction<>(DriverFactory.getDriver()).tap(PointOption.point(x,
y)).perform();
}
```

Na classe **AlertPage**

```
public void clicarForaDaCaixa(){
      tap(100, 150);
}
```

```
//clicar fora da caixa nas coordenadas 110,150
esperar(1000);
pagina.clicarForaDaCaixa();

//verificar que a mensagem nao existe mais
assertFalse(pagina.existeElementoPorTexto("Pode clicar no OK ou fora da caixa para sair"));
```

Elemento SeekBar

Ação de toque, interagir com a barra/linha.

Na classe **FormularioPage**

```
public void clicarNoSeekBar(double posicao){
	MobileElement seek =
DriverFactory.getDriver().findElement(MobileBy.AccessibilityId("slid"));

	//diferenca entre espaco do elemento e a tela
	int delta = 50;

	int xInicial = seek.getLocation().x + delta;

	int x = (int) (xInicial + (seek.getSize().width-2*delta)*posicao);

	int y = (seek.getLocation().y) + (seek.getSize().height/2);

	//o X é a coordenada de onde se inicial o elemento na tela - o delta.
	//o Y é a largura e altura total do elemento.

	tap(x, y);
}
```

Na classe **FormularioTest**

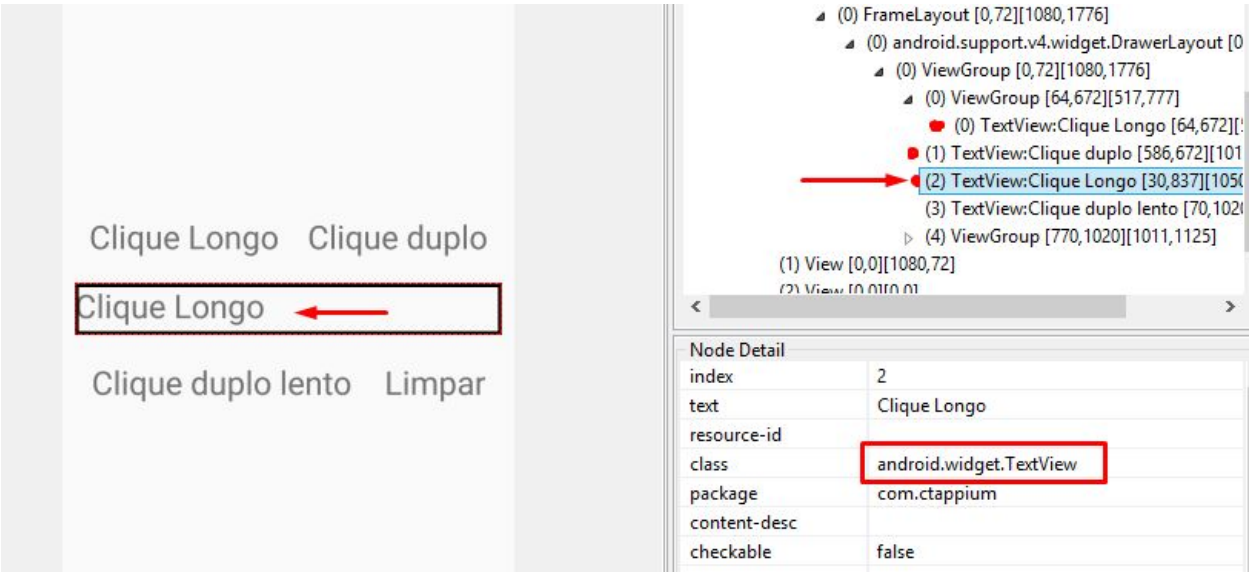
```
//clicar no seekbar, em porcentagem de onde deseja clicar.
pagina.clicarNoSeekBar(0.86);

//validar valor do seekBar
assertEquals(“Slider: 86”, pagina.obterSliderCadastrado());
```

Elemento Clique Longo

```
public void fazerCliqueLongo(){
	MobileElement mobileElemento= DriverFactory.getDriver()
		.findElement(By.xpath("//*[@text = 'Clique Longo']"));
```

```
        new TouchAction<>(DriverFactory.getDriver())
                .longPress(ElementOption.element(mobileElemento)).perform();
}

public String obterTextoCampo(){
        //usamos "(.....)" para procurar todos os elementos que possuem essa classe
        //deixando todas ocorrencias em mesmo nivel/camada e retornando a terceira
ocorrencia.

        return DriverFactory.getDriver().findElement(
                By.xpath("(//android.widget.TextView)[3] ").getText();
}
```

Como só temos uma classe para identificar o campo que queremos obter o texto, e essa classe aparece em vários elementos.



Elemento Clique Duplo

```
@Test
public void deveInteragirComCliqueDuplo(){
        new BasePage().clicarPorTexto("Clique duplo");
        new BasePage().clicarPorTexto("Clique duplo");

        //verificar o texto abaixo
        assertEquals("Duplo Clique", pagina.obterTextoCampo());
```

}

Elemento Scroll - Como arrastar a tela para cima ou para baixo.

Devemos arrastar no cenário do aplicativo utilizando as dimensões (largura e altura) da tela.

*** **Não é recomendado pegar os valores fixos da tela**, mesmo que esteja inspecionando o aplicativo através do uiautomatorviewer, pois dependendo do aparelho podemos ter tela maiores ou menores. Portanto pegamos uma largura central dinamicamente!

Lembrete:
Em aplicativos móveis nos deslizamos a tela para baixo ao arrastar para cima.

Na classe **OpcaoEscondidaTest**

```
//vamos arrasatar  a tela de 90% (final da tela) ate 10% (o comeco da tela)
menu.scrollDown(0.9, 0.1);

//clicar menu Opcao bem escondida
menu.acessarOpcaoBemEscondida();
```

Na classe **BasePage**

```
public void scrollDown(double inicio, double fim){

        //Esperamos no maximo 10 segundos ate que a tela esteja carregada.
        WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(), 10);
        wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//*[@text =
'Formulário']")));

        System.out.println("Começando");

        //dimensões da tela do aparelho
        Dimension size = DriverFactory.getDriver().manage().window().getSize();

        int x = size.width/2;

        //clicamos no final
        int start_y = (int) (size.height * inicio);
```

```
        //clicamos no inicio da tela
        int end_y = (int) (size.height * fim);

        //arrastar a tela do meio para a parte superior da tela.
        //segurar um pouco a tela
        //mover para o fim da tela
        //soltar a tela.
        new TouchAction<>(DriverFactory.getDriver()).press(PointOption.point(x, start_y))
        .waitAction(WaitOptions.waitOptions(Duration.ofMillis(500)))
        .moveTo(PointOption.point(x, end_y))
                .release().perform();
}

//tela subindo!!
public void scrollDown(){
        //são porcentagens da tela
        scroll(0.9, 0.1);
}

//tela descendo / atualizar!!
public void scrollUp(){
        //são porcentagens da tela
        scroll(0.1, 0.9);
}
```

Método Adicional:

Fazer o scroll a partir do menu superior.

```
        WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(), 10);

        wait.until(ExpectedConditions.presenceOfElementLocated(By.className(nomedaclasse
        )));

        MobileElement elemento = DriverFactory.getDriver()
        .findElement(By.className(nomedaclasse));

        //comecar a arrastar a tela depois do elemento definido
        int delta = elemento.getSize().height;

        //size = dimensões da tela do dispositivo
        int start_y = (int) (size.height * inicio) + delta;
```

Elemento Swipe - Arrastar para esquerda ou para direita

Vamos utilizar a mesma ideia do Scroll, porém ao invés de usarmos a altura, vamos trabalhar com a **largura da tela**.

Na classe **BasePage**

```
public void swipe(double inicio, double fim){
        Dimension size = DriverFactory.getDriver().manage().window().getSize();

        //no meio da tela verticalmente
        int y = size.height/2;

        int start_x = (int) (size.width * inicio);
        int end_x = (int) (size.width * fim);

        new TouchAction<>(DriverFactory.getDriver()).press(PointOption.point(start_x, y))
                .waitAction(WaitOptions.waitOptions(Duration.ofMillis(500)))
                .moveTo(PointOption.point(end_x,y))
            .   release().perform();
}

public void swipeRight(){
        //são porcentagens da tela
        swipe(0.9,0.1);
}

public void swipeLeft(){
        //são porcentagens da tela
        swipe(0.1,0.9);
}
```

Na classe **SwipeTest**

!!!! atenção para os símbolos utilizados em dispositivos mobiles, podem ser diferentes do que usamos como o sinal de maior (> / ›) e/ou menor (< / ‹).

```
// swipe para o lado esquerdo
menu.swipe(0.1, 0.9);
```

```
//clicar botão da esquerda
wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//*[@text = '‹']")));
menu.clicarPorTexto("‹");

//verificar texto da proxima tela
assertTrue(menu.existeElementoPorTexto("a esquerda"));
```

Aula de Desafio - Swipe com Elementos!!!

Na classe **BasePage**

```
public void swipeElement(MobileElement element, double inicio, double fim){
        //Esperamos no maximo 10 segundos ate que a tela esteja carregada.
        WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(), 10);
        wait.until(ExpectedConditions.visibilityOf(element));

        //vamos utilizar o tamanho apenas do elemento para interagir com ele
        Dimension ElementSize = element.getSize();

        //devemos utilizar a localização do elemento na tela
        int y = element.getLocation().y + (ElementSize.height / 2);

        int start_x = (int) (ElementSize.width * inicio);
        int end_x = (int) (ElementSize.width * fim);

        new TouchAction<>(DriverFactory.getDriver()).press(PointOption.point(start_x, y))
                .waitAction(WaitOptions.waitOptions(Duration.ofMillis(500)))
                .moveTo(PointOption.point(end_x,y))
                .release().perform();
}
```

Na classe **SwipeListPage**

```
public void swipeElementLeft(String opcao){
        //vamos subir um nível e pegar a div que possui esse texto
        swipeElement(DriverFactory.getDriver().findElement(By.xpath("//*[@text='"+ opcao
+"']/.")), 0.1, 0.9);
}

public void swipeElementRight(String opcao){
        //vamos subir um nível e pegar a div que possui esse texto
```

```
        swipeElement(DriverFactory.getDriver().findElement(By.xpath("//*[@text='"+ opcao
+"']/..")), 0.9, 0.1);
    }

public void clicarBotaoMais(){
        MobileElement botao =
DriverFactory.getDriver().findElement(By.xpath("//*[@text='(+)']/.."));
        new TouchAction<>(DriverFactory.getDriver())
                .tap(TapOptions.tapOptions()
                .withElement(ElementOption.element(botao, -50, 0)))
                .perform();
}
```

Na classe **SwipeTest**

```
//opcao 1 para direita
pagina.swipeElementRight("Opção 1");

//clicar na opt 1+
pagina.clicarBotaoMais();

//verificar que a opt1 ficou com + na frente
assertTrue(menu.existeElementoPorTexto("Opção 1 (+)"));

//resetar a opcao 6
pagina.swipeElementLeft("Opção 6 (+)");

//verificar que a opt4 ficou com - na frente
assertTrue(menu.existeElementoPorTexto("Opção 6"));
```

Elemento Drag and Drop

Vamos arrastar um objeto da lista para alterar a posição em que ele se encontra.

Na classe **DragAndDropPage**

```
public void arrastar(String origem, String destino){
        WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(), 10);

        wait.until(ExpectedConditions.visibilityOf(DriverFactory.getDriver().findElement(By.xpath
        ("//*[@text='"+ origem +"']"))));
```

```
        MobileElement inicio = DriverFactory.getDriver().findElement(By.xpath("//*[@text='"+
origem +"']"));
        MobileElement fim = DriverFactory.getDriver().findElement(By.xpath("//*[@text='"+
destino +"']"));

        new TouchAction<>(DriverFactory.getDriver())
        .longPress(ElementOption.element(inicio))
        .moveTo(ElementOption.element(fim))
        .release()
        .perform();

    }

    public String[] obterLista(){
        List<MobileElement> elements = DriverFactory.getDriver()
        .findElements(By.xpath("//android.widget.TextView"));

        String[] retorno = new String[elements.size()];

        for (int i = 0; i < elements.size(); i++) {
                retorno[i] = elements.get(i).getText();
                //System.out.println("\"" + retorno[i] + "\", ");
        }

        return retorno;
    }
```

Na classe **DragAndDropTest**

```
private final String[] estadoInicial = new String[]{
        "Esta",
        "é uma lista",
        "Drag em Drop!",
        "Faça um clique longo,",
        "e arraste para",
        "qualquer local desejado."};

private final String[] estadoIntermediario = new String[]{
        "é uma lista",
        "Drag em Drop!",
        "Faça um clique longo,",
```

```
        "e arraste para",
        "Esta",
        "qualquer local desejado."};

//verificar estado inicial
esperar(1000);
assertArrayEquals(estadoInicial, pagina.obterLista());

//arrastar "Esta" para "e arraste para"
pagina.arrastar("Esta", "e arraste para");

//verificar estado intermediario
esperar(1000);
assertArrayEquals(estadoIntermediario, pagina.obterLista());
```

Como funciona as Aplicações Híbridas?

São **aplicações Web (Webview)** encapsuladas com aplicações nativas.

***** Detalhe importante que a aplicação web encapsulada, normalmente não vai apresentar a cadeia de elementos ao inspecionar a tela pelo uiautomatorviewer.**

Exemplo de uma aplicação web (formulário) utilizado na aula:
http://seubarriga.wcaquino.me:4001

Precisamos trocar de contexto, vamos passar instruções para nosso teste, dizendo que agora vamos trabalhar com uma aplicação web, ou seja vamos utilizar o navegador chrome além da aplicação mobile.
Na classe **WebViewPage**

```
public void entrarContextoWeb(){

        WebDriverWait wait = new WebDriverWait(DriverFactory.getDriver(),300);
        By webView = By.className("android.webkit.WebView");
        wait.until(ExpectedConditions.visibilityOfElementLocated(webView));

        Set<String> contextHandles = DriverFactory.getDriver().getContextHandles();

        System.out.println("quantidade de contexts = "+ availableContexts.size());

        for (String valor : contextHandles){
```

```
                System.out.println(valor);

                if(context.contains("WEBVIEW_com")){
                        System.out.println("Novo Contexto: " + context);
                        //WEBVIEW_com.ctappium

                        DriverFactory.getDriver().context(context);
                        break;
                }
        }
}

public void sairContextoWeb() {
        DriverFactory.getDriver().context("NATIVE_APP");
}
```

Na classe **WebViewTest**

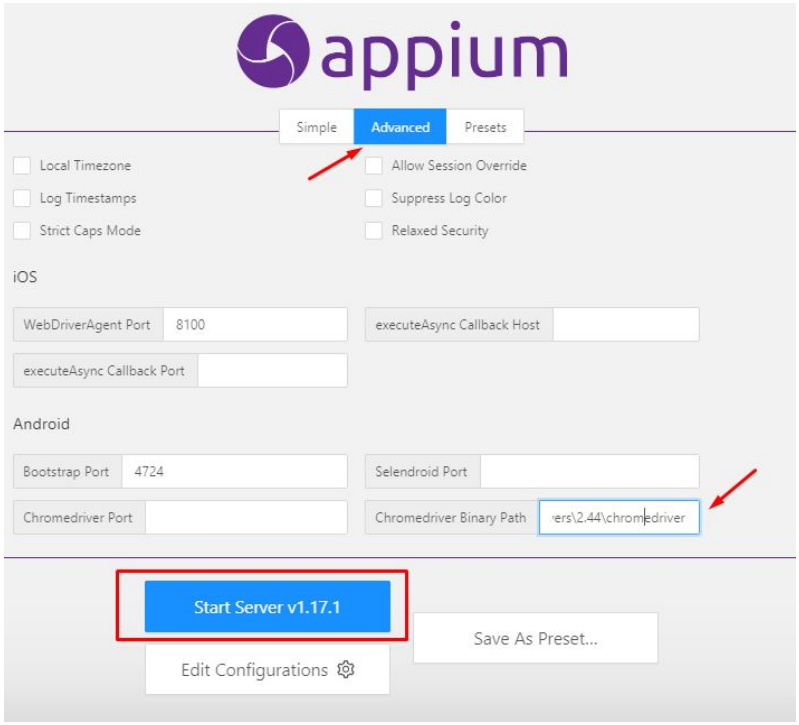
```
//acessar o menu seu barriga hibrido
menu.acessarSBHibrido();

//3seg
esperar(3000);
pagina.entrarContextoWeb();

@After
public void tearDown(){
        //precisamos adicionar um método a ser executado após o teste para que ele saia do
        contexto Web e volte para o contexto nativo, dessa forma ele continuará executando
        todo o código que criamos antes, inclusive no BaseTest, como @after e @afterClass.
        Caso contrário ele não irá encerrar o teste mobile.

        pagina.sairContextoWeb();
}
```

***** Importante! É provável que a versão do chrome no seu dispositivo (emulador) seja mais antiga do que a atual. Portanto será necessário fazer o download do chromedriver para aquela versão.**

https://chromedriver.chromium.org/downloads
https://chromedriver.storage.googleapis.com/index.html (para versões mais antigas)

Configurar a versão do chrome no Appium Launcher!

Na aba avançado, ChromeDriver Binary Path …. C://chromedriver.exe (no windows)



Resolvendo problema com uiautomatorviewer

Mensagem de erro: Remote Object don't exist

Executar esses comandos no cmd na pasta platform-tools:
adb kill-server
adb start-server

Validar Inclusão de Movimentação

```
@Test
public void deveIncluirEValidarMovimentacao(){
        pagina.movimentacoes();
```

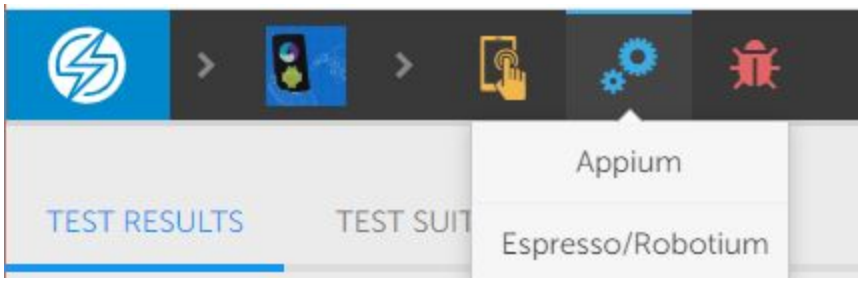
```
        assertTrue(pagina.existeElementoPorTexto("Receita"));

        pagina.alterarStatus();
        assertTrue(pagina.isCheckMarcado(By.xpath("//android.widget.Switch")));

        pagina.selecionarDataMovimentacao();
        pagina.selecionarDataPagamento();
        pagina.salvar();
        assertTrue(pagina.existeElementoPorTexto("Descrição é um campo obrigatório"));

        pagina.escrever(By.xpath("//*[@text='Descrição']"), "prêmio da loteria");
        pagina.salvar();
        assertTrue(pagina.existeElementoPorTexto("Interessado é um campo obrigatório"));

        pagina.escrever(By.xpath("//*[@text='Interessado']"), "Eu mesmo");
        pagina.salvar();
        assertTrue(pagina.existeElementoPorTexto("Valor é um campo obrigatório"));

        pagina.escrever(By.xpath("//*[@text='Valor']"), "2300,50");
        pagina.salvar();
        assertTrue(pagina.existeElementoPorTexto("Conta é um campo obrigatório"));

        pagina.selecionarContaMovimentacao("Conta Saldo 2020");
        pagina.salvar();
        assertTrue(pagina.existeElementoPorTexto("Movimentação cadastrada com sucesso"));

        pagina.home();

        esperar(1000);
        float saldoAntigo = pagina.obterSaldoDaConta("Conta Saldo 2020");

        //atualizar saldos
        esperar(1000);
        pagina.scrollUpAfterElement("android.widget.HorizontalScrollView");

        //validar saldo atual
        esperar(3000);
        assertEquals(saldoAntigo + 2300.50, pagina.obterSaldoDaConta("Conta Saldo 2020"),
0.01);
}
```

Como Executar na Nuvem

Acessar: https://app.testobject.com/
app.testobject.com/#/caioandrian/
(ferramenta paga)

Logar, Novo Android or IOS App, Uploading da APK no site.
Configurações padrão.

Live Testing = aparelhos disponíveis para executar.

Appium 

Get Start

No DriveFactory do nosso teste, vamos configurar para acessar o dispositivo real na nuvem.

```
private static void createTestObjectDriver() {
        DesiredCapabilities desiredCapabilities = new DesiredCapabilities();
        desiredCapabilities.setCapability("platformName", "Android");

        //trocar o xxxx pelo código fornecido na sua conta no site
        desiredCapabilities.setCapability("testobject_api_key", "xxxxxxxx");

        //para garantir mandamos a mesma versão do appium que utilizamos
        desiredCapabilities.setCapability("appiumVersion", "1.17.1");

        desiredCapabilities.setCapability("automationName", "uiautomator2");
        desiredCapabilities.setCapability("fullContextList ", "true");
        desiredCapabilities.setCapability("setWebContentsDebuggingEnabled ", "true");

        try {
                //enviamos as definicioes antes para o Appium, agora para a nuvem.
                URL remoteUrl = new URL("https://us1-manual.app.testobject.com/wd/hub");
```

```
            aplicativo = new AndroidDriver(remoteUrl, desiredCapabilities);
        }catch (MalformedURLException e){
            e.printStackTrace();
        }

        //espera implicita
        aplicativo.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
}

public static AndroidDriver<MobileElement> getDriver()  {
        if (aplicativo == null){
            //createDriver();
            createTestObjectDriver();
        }

        return aplicativo;
}
```

No site **https://app.testobject.com/** em Dashboard, **após rodar um teste, atualizar a página.**