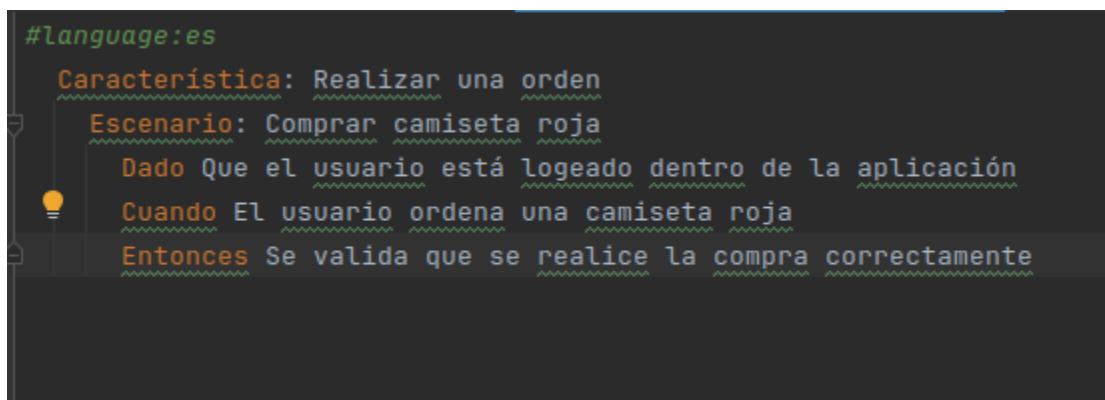
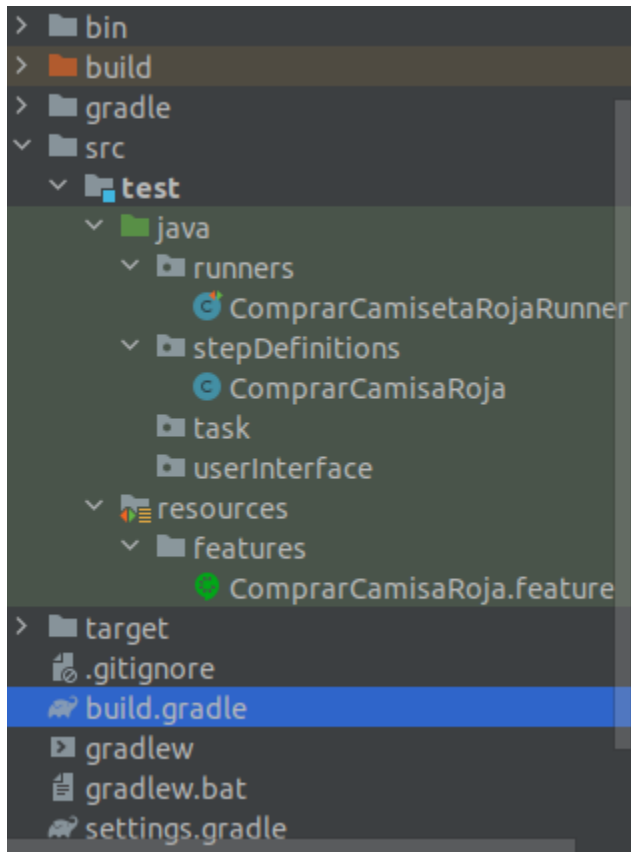


# Informe Automatizacion Mobile

## Paso a paso

1. Se configura el proyecto de Java con screenplay y cucumber.



```
public class ComprarCamisaRoja {  
    @Before  
    public void setUp(){  
        OnStage.setTheStage(new OnlineCast());  
        theActorCalled( requiredActor: "kevin");  
    }  
  
    @Dado("Que el usuario está logueado dentro de la aplicación")  
    public void queElUsuarioEstáLogeadoDentroDeLaAplicación() {  
        // Write code here that turns the phrase above into concrete actions  
        throw new io.cucumber.java.PendingException();  
    }  
  
    @Cuando("El usuario ordena una camiseta roja")  
    public void eUsuarioOrdenaUnaCamisetaRoja() {  
        // Write code here that turns the phrase above into concrete actions  
        throw new io.cucumber.java.PendingException();  
    }  
  
    @Entonces("Se valida que se realice la compra correctamente")  
    public void seValidaQueSeRealiceLaCompraCorrectamente() {  
        // Write code here that turns the phrase above into concrete actions  
        throw new io.cucumber.java.PendingException();  
    }  
}
```

```

package runners;

import io.cucumber.junit.CucumberOptions;
import net.serenitybdd.cucumber.CucumberWithSerenity;
import org.junit.runner.RunWith;

import static io.cucumber.junit.CucumberOptions.SnippetType.CAMELCASE;

@RunWith(CucumberWithSerenity.class)
@CucumberOptions(
    features = "src/test/resources/features/ComprarCamisaRoja.feature",
    glue = {"stepDefinitions"},
    snippets = CAMELCASE
)

public class ComprarCamisetaRojaRunner {
}

```

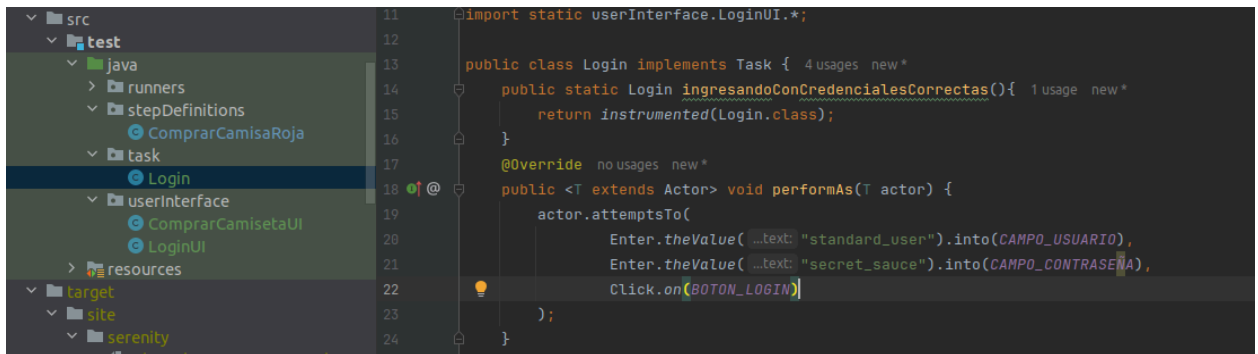
2. Se configura el archivo serenity.properties para establecer las capabilities

```

#Device Capabilities
appium.platformName=Android
appium.deviceName=ZT322HR9BM
appium.platformVersion=12.0
appium.app=/home/kevin/banistmo/automatizacionMobileBanistmo/sauceLabsApp271.apk
appium.appPackage=com.swaglabsmobileapp
appium.appActivity=com.swaglabsmobileapp.MainActivity
#Server Capabilities
webdriver.driver=appium
appium.hub=http://127.0.0.1:4723/wd/hub
#Generic Capabilities
appium.autoGrantPermissions=true
appium.autoAcceptAlerts=true
appium.deviceOrientation=portrait
appium.captureScreenshots=true
serenity.take.screenshots=AFTER_EACH_STEP
appium.sessionName=Android automation test session
appium.newCommandTimeout = 120000

```

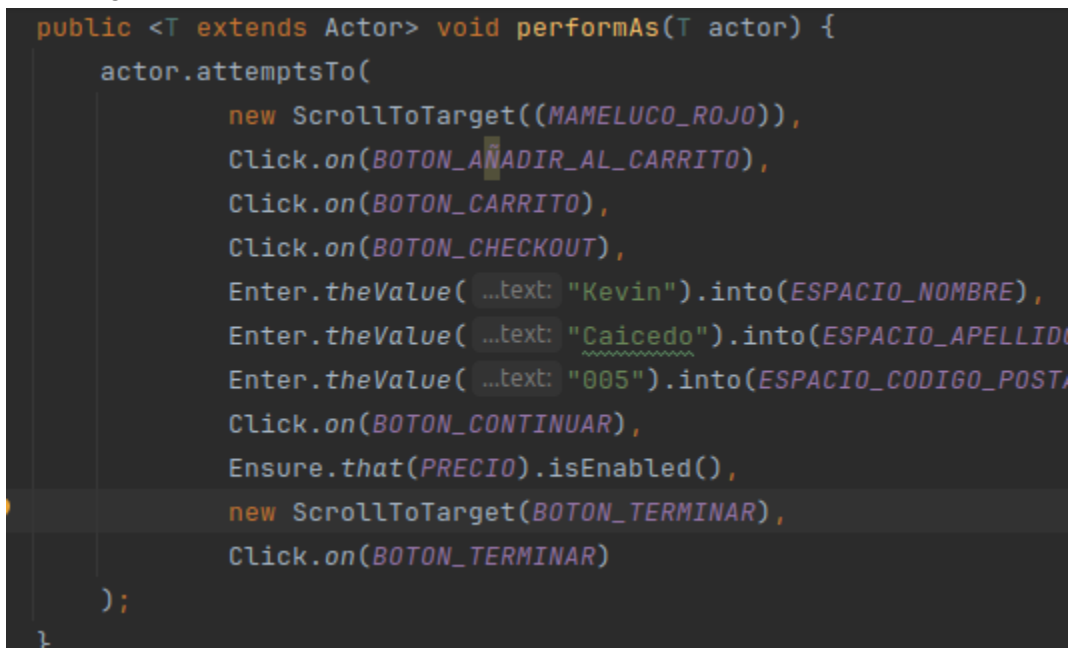
3. Se configura el login



The screenshot shows an IDE with a project structure on the left and the Login class code on the right. The project structure includes a 'test' directory with 'java' subdirectories: 'runners', 'stepDefinitions', 'task', and 'userInterface'. The 'task' directory contains 'Login'. The 'userInterface' directory contains 'ComprarCamisetaUI' and 'LoginUI'. The 'Login' class code is as follows:

```
11 import static userInterface.LoginUI.*;
12
13 public class Login implements Task {
14     public static Login ingresandoConCredencialesCorrectas(){
15         return instrumented(Login.class);
16     }
17
18     @Override
19     public <T extends Actor> void performAs(T actor) {
20         actor.attemptsTo(
21             Enter.theValue("standard_user").into(CAMPO_USUARIO),
22             Enter.theValue("secret_sauce").into(CAMPO_CONTRASEÑA),
23             Click.on(BOTON_LOGIN)
24         );
25     }
26 }
```

4. Se configura el escenario completo



The screenshot shows the 'performAs' method code in the Login class, which is part of the 'task' directory. The code is as follows:

```
public <T extends Actor> void performAs(T actor) {
    actor.attemptsTo(
        new ScrollToTarget((MAMELUCO_ROJO)),
        Click.on(BOTON_AÑADIR_AL_CARRITO),
        Click.on(BOTON_CARRITO),
        Click.on(BOTON_CHECKOUT),
        Enter.theValue("Kevin").into(ESPACIO_NOMBRE),
        Enter.theValue("Caicedo").into(ESPACIO_APELLIDO),
        Enter.theValue("005").into(ESPACIO_CODIGO_POSTAL),
        Click.on(BOTON_CONTINUAR),
        Ensure.that(PRECIO).isEnabled(),
        new ScrollToTarget(BOTON_TERMINAR),
        Click.on(BOTON_TERMINAR)
    );
}
```

5. Se realiza el assertion



The screenshot shows the assertion code in the Login class, which is part of the 'task' directory. The code is as follows:

```
@Entonces("Se valida que se realice la compra correctamente")
public void seValidaQueSeRealiceLaCompraCorrectamente() {
    theActorInTheSpotlight().should(eventually(seeThat(the(TEXTO_COMPLETADO), isCurrentlyVisible())));
}
```

## Repositorio

<https://github.com/kevincaicedo95/AutomatizacionMobile.git>

## Lista comandos Github

git init  
git add .

```
git commit -m "configurando proyecto"
git remote add origin https://github.com/kevincaicedo95/AutomatizacionMobile.git
git branch -M main
git push -u origin main
git add .
git commit -m "configurando capabilities en serenety.properties"
git push -u origin main
git add .
git commit -m "subiendo escenario completo e informe"
git push -u origin main
```