

ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

Primeros pasos con el patrón Screenplay y Appium

En la presente guía, daremos nuestros primeros pasos usando el patrón Screenplay con Appium. Una vez cargado nuestro proyecto base en intellij, procederemos a crear la Configuración Inicial, Feature y StepDefinitions.

Contenido

Primeros pasos con el patrón Screenplay y Appium	1
Importar proyecto Base	2
Mi primera prueba usando Screenplay BDD con APPIUM	5
Creación de la feature.	6
Redacción de nuestra feature usando lenguaje Gherkin.	7
Creación del runner y el stepdefinitions.	8

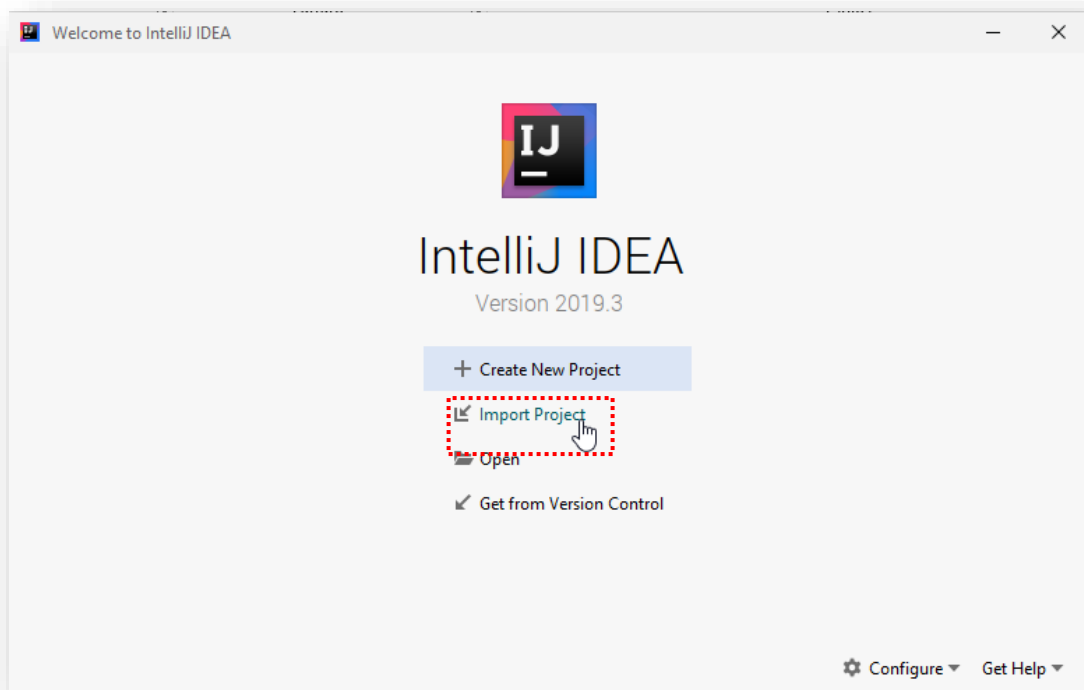


ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

Importar proyecto Base

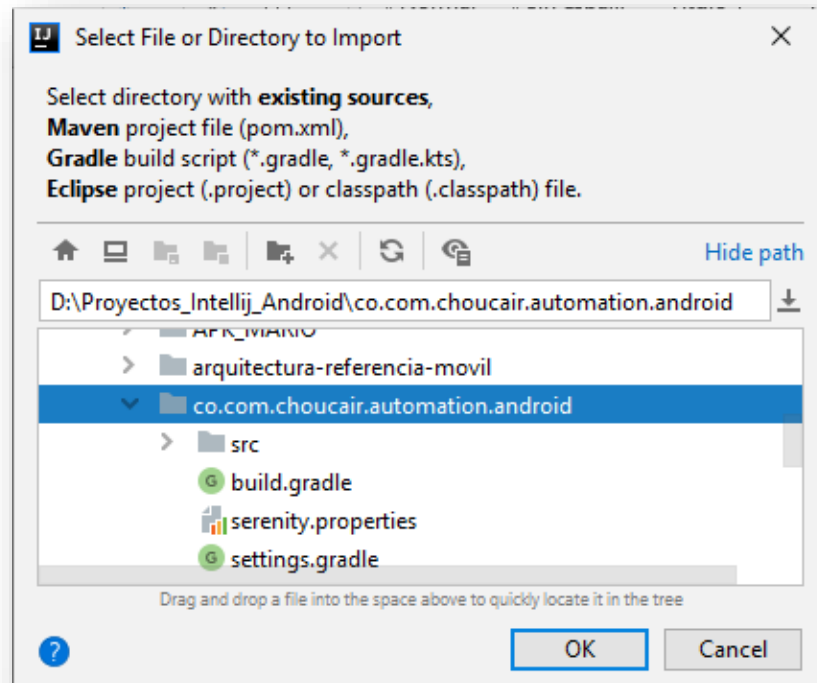
Se cuenta con un proyecto base el cual contiene la estructura del patrón de screenplay para automatización móviles.

1. Descarga el proyecto base en la ruta del workspace configurado del siguiente link: [Proyecto Base Appium](#).
2. Descomprime el proyecto en la ruta de tu elección.
3. Importar el proyecto.
 - a. Iniciamos abriendo IntelliJ IDEA y damos clic en Import Project

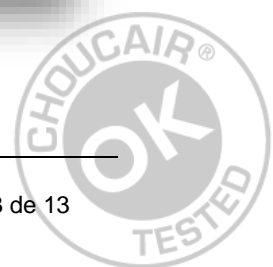
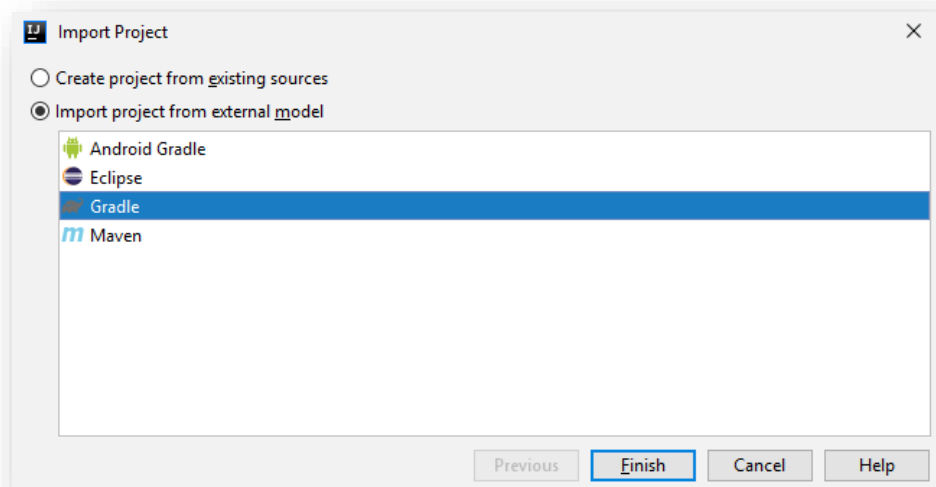


- b. Seleccionamos la ruta donde descomprimos el proyecto y damos clic en OK.

ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

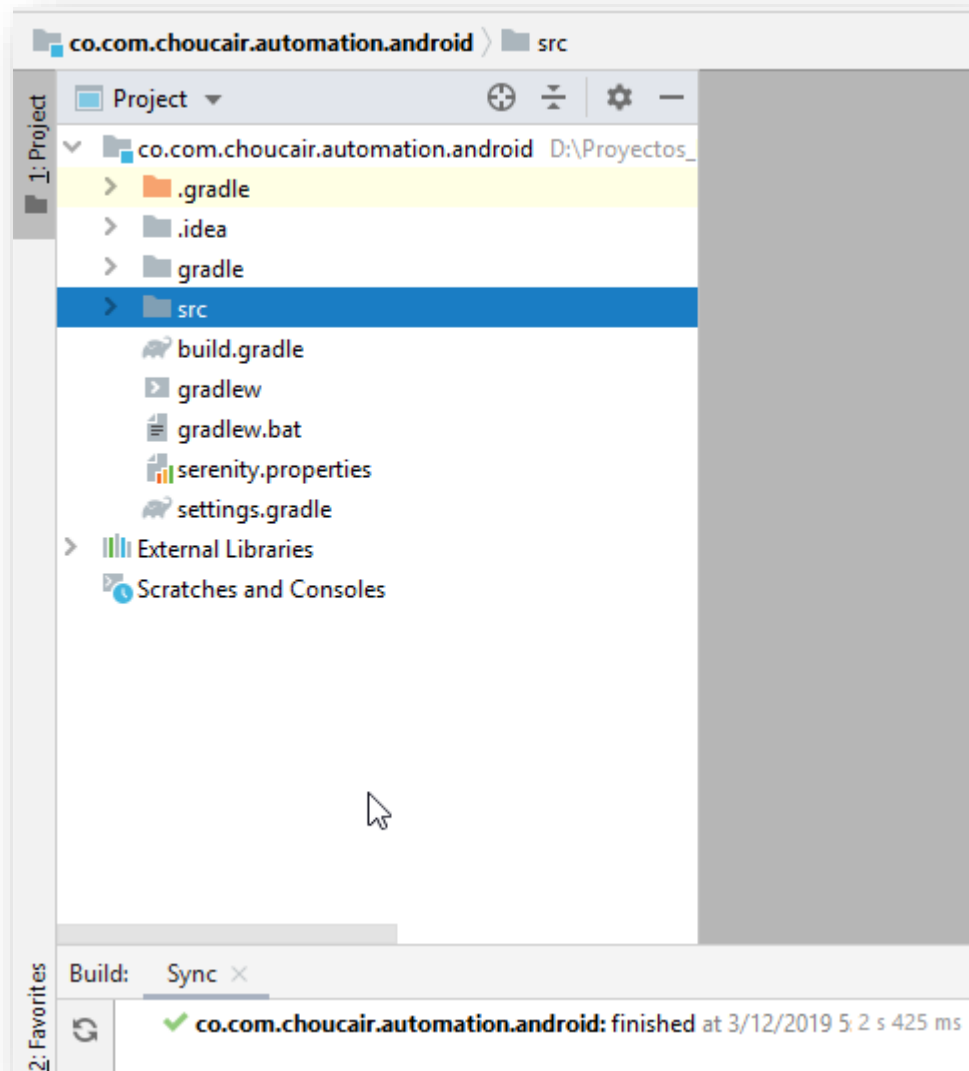


- c. Escoge la opción de importar proyecto y que además sea de tipo Gradle y da clic en Finish.



ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

- d. Este proceso puede durar varios minutos, permita que el proceso finalice por completo. Una vez terminado debe quedar una estructura así:



ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

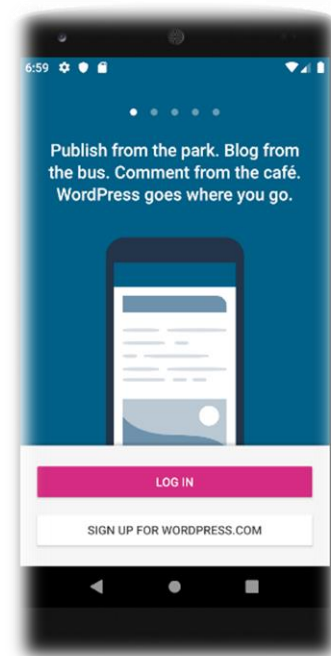
Mi primera prueba usando Screenplay BDD con APPIUM

Historia de Usuario: Verificar el acceso exitoso a la APP Móvil.

- Criterios de Aceptación: Verificar Logueo Exitoso a la APP Móvil.

Pasos para la ejecución de la prueba.

1. Autenticación en la APP WordPress
 - a. Abrir la APP
 - b. Dar clic en el botón LOG IN
 - c. Ingresar “pruebaappappium@gmail.com” en el campo Email Address
 - d. Dar clic en el botón NEXT
 - e. Dar clic en el Link “Enter your password instead”
 - f. Ingresar “pruebaapp99” en el campo Password
 - g. Dar clic en el botón NEXT
 - h. Verificar la Autenticación (Logged in as)



Análisis:

Existen diferentes formas de analizar el flujo de una transacción, para efectos de este taller vamos a agrupar los pasos en acciones concretas y específicas. Dichas acciones serán escritas en lenguaje Gherkin en nuestra feature.

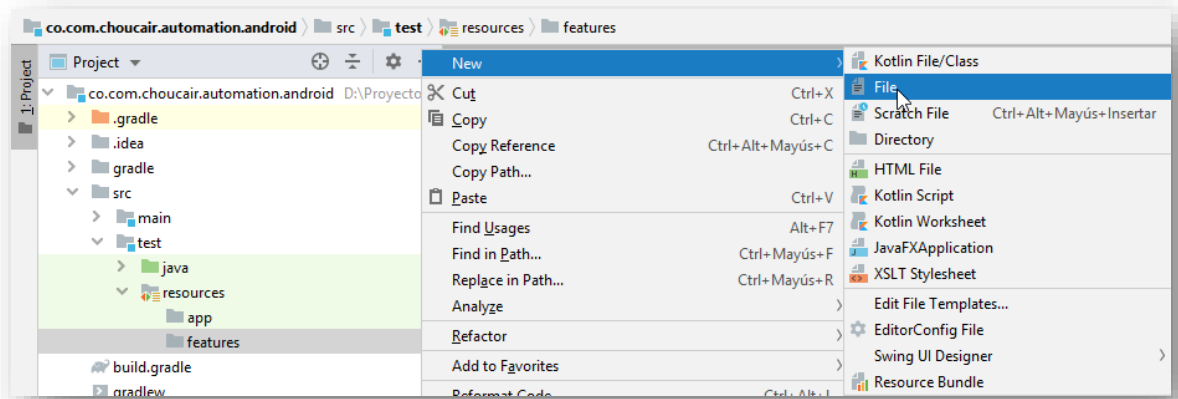
Como siempre en cualquier prueba es importante que determinemos la data requerida para la ejecución de la prueba.

ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

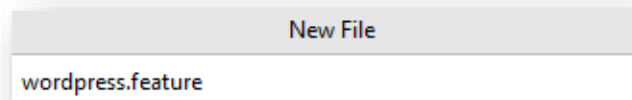
Creación de la feature.

En este punto es necesario volver a intellj para iniciar con la creación de la historia de usuario.

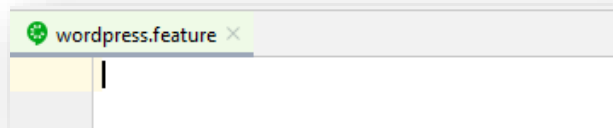
1. Ingresamos a la ruta **src/test/resources/features**
2. Sobre la capeta feature damos clic derecho > New > File



3. Y le ponemos el nombre al archivo para este ejemplo lo llamaremos **wordpress** y como extensión será **.feature** y presionamos la tecla **Enter**.



4. Nos debe generar un file como el de la siguiente imagen en donde redactaremos nuestra historia de usuario



ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

Redacción de nuestra feature usando lenguaje Gherkin.

Importante: Aunque los features se puedan escribir en distintos idiomas con la palabra reservada: `#Language: <idioma>`

Nosotros en **Choucair** usaremos como estándar el idioma nativo de Cucumber que es Inglés, es decir que de aquí en adelante toda nuestra historia de usuario (feature) y nuestra automatización la realizaremos en inglés.

Teniendo en cuenta lo anterior nuestra feature quedaría así:

Feature: Wordpress application automation

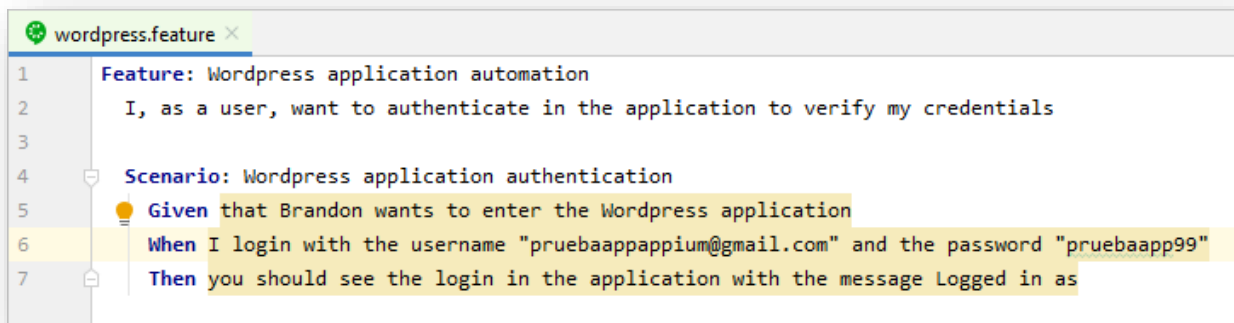
I, as a user, want to authenticate in the application to verify my credentials

Scenario: Wordpress application authentication

Given that Brandon wants to enter the Wordpress application

When I login with the username "pruebaappappium@gmail.com" and the password "pruebaapp99"

Then you should see the login in the application with the message Logged in as



```
wordpress.feature x
1 Feature: Wordpress application automation
2   I, as a user, want to authenticate in the application to verify my credentials
3
4 Scenario: Wordpress application authentication
5   Given that Brandon wants to enter the Wordpress application
6   When I login with the username "pruebaappappium@gmail.com" and the password "pruebaapp99"
7   Then you should see the login in the application with the message Logged in as
```

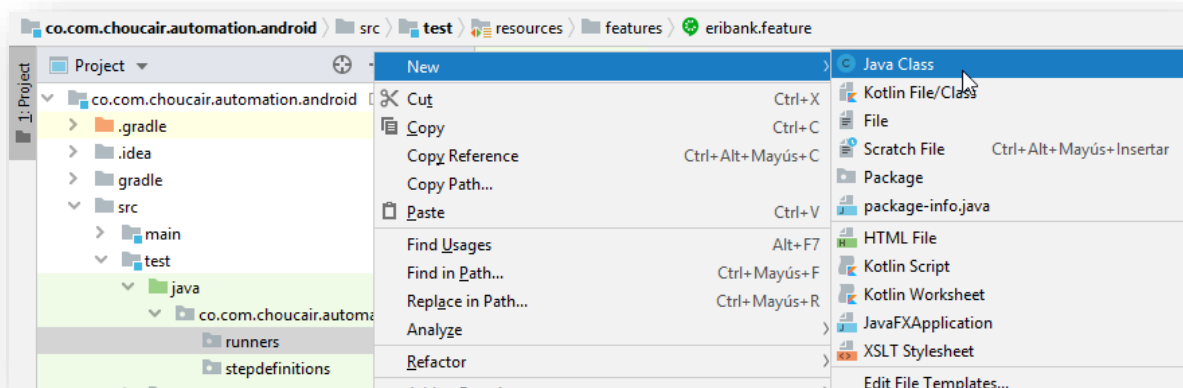


ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

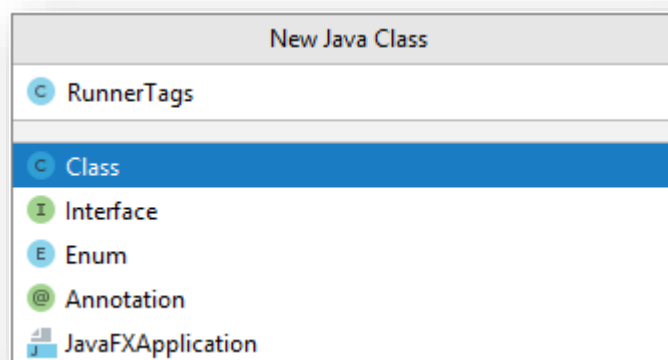
Creación del runner y el stepdefinitions.

El runner será donde se iniciará la ejecución de nuestra prueba, para ello vamos a crear la clase a la cual llamaremos RunnerTags.

1. Ingresamos a la ruta **src/test/java/ co.com.choucair.automation.android/runners**
2. Sobre la carpeta runners presionamos clic derecho > New > Java Class

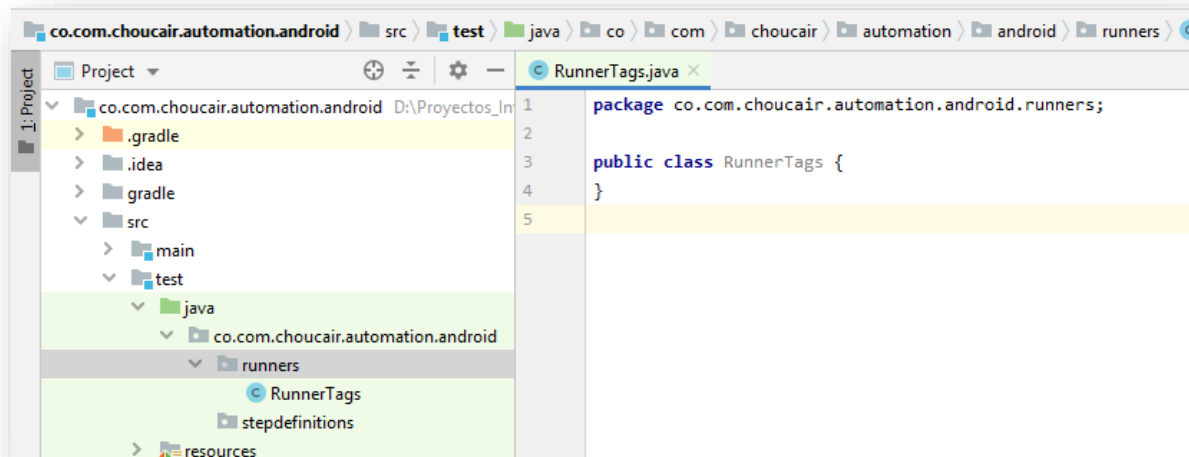


3. Creamos la clase llamada RunnerTags



ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

4. Y nos debe quedar algo así:



5. Escribimos lo siguiente en nuestro runner.

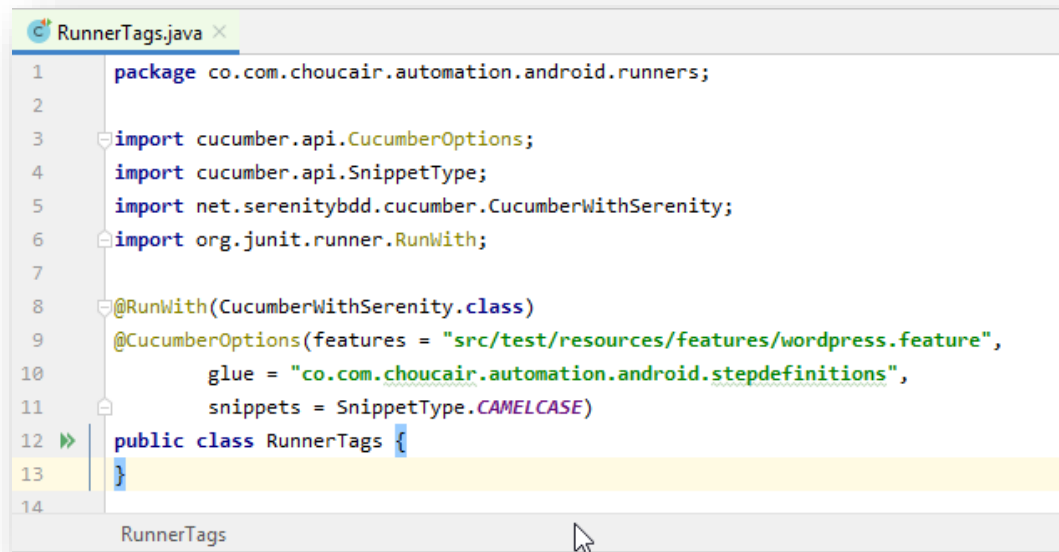
```
package co.com.choucair.automation.android.runners;

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

@RunWith(CucumberWithSerenity.class)
@CucumberOptions(features = "src/test/resources/features/wordpress.feature",
    glue = "co.com.choucair.automation.android.stepdefinitions",
    snippets = SnippetType.CAMELCASE)
public class RunnerTags {
}
```

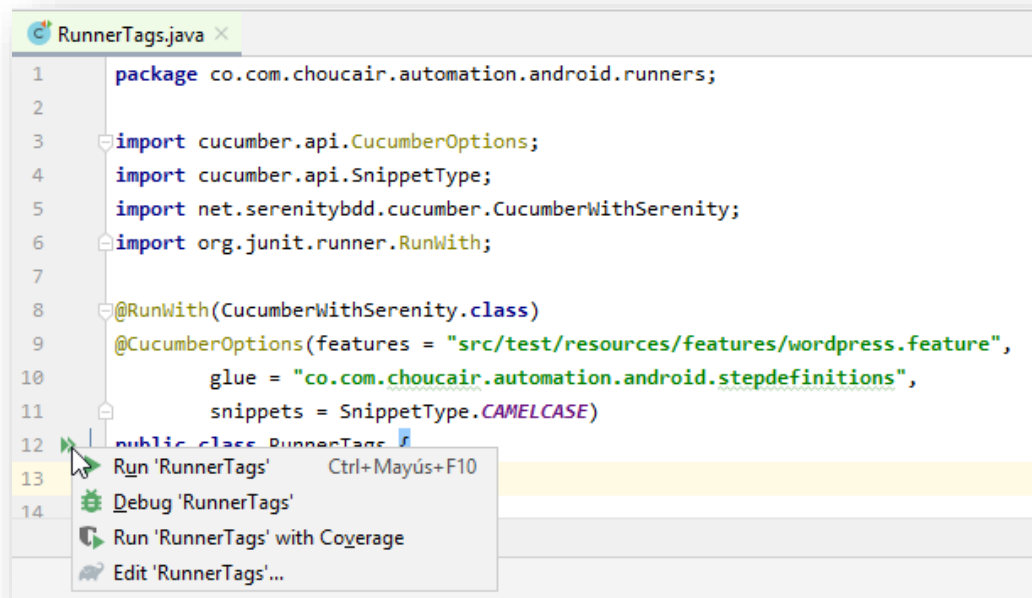
ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

6. Y nuestra clase RunnerTags debe verse así:



```
1 package co.com.choucair.automation.android.runners;
2
3 import cucumber.api.CucumberOptions;
4 import cucumber.api.SnippetType;
5 import net.serenitybdd.cucumber.CucumberWithSerenity;
6 import org.junit.runner.RunWith;
7
8 @RunWith(CucumberWithSerenity.class)
9 @CucumberOptions(features = "src/test/resources/features/wordpress.feature",
10                 glue = "co.com.choucair.automation.android.stepdefinitions",
11                 snippets = SnippetType.CAMEL_CASE)
12 public class RunnerTags {
13 }
14
```

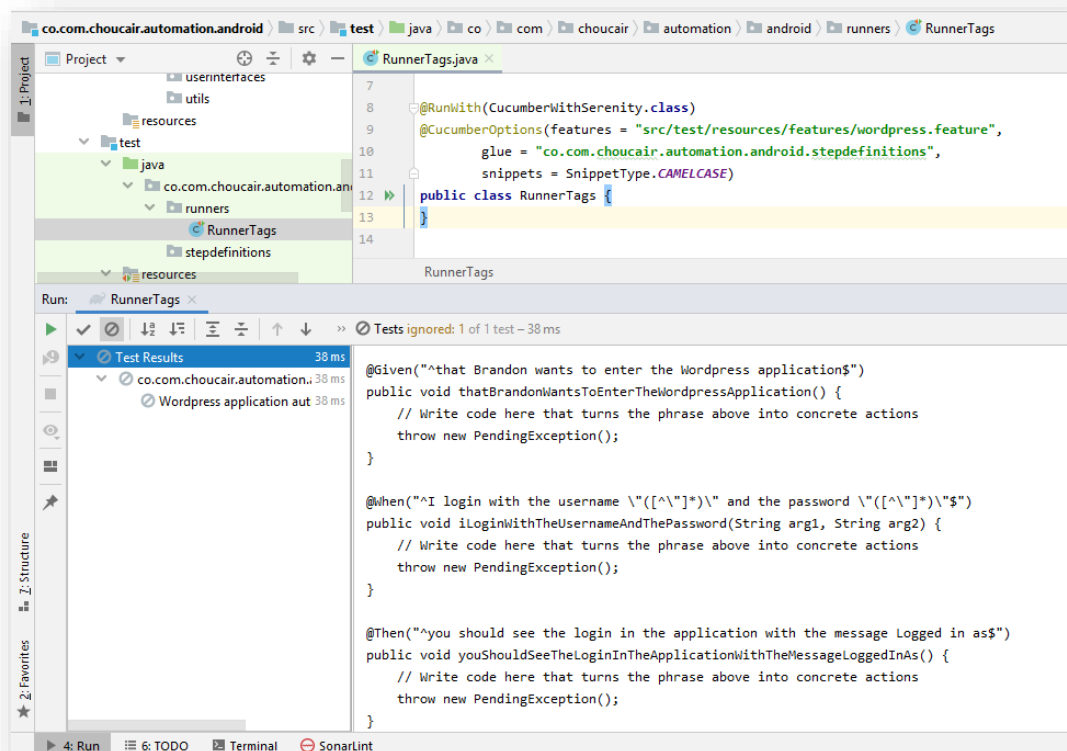
7. Para ejecutar la clase RunnerTags, clic derecho sobre el runner > Run 'RunnerTags'.



```
1 package co.com.choucair.automation.android.runners;
2
3 import cucumber.api.CucumberOptions;
4 import cucumber.api.SnippetType;
5 import net.serenitybdd.cucumber.CucumberWithSerenity;
6 import org.junit.runner.RunWith;
7
8 @RunWith(CucumberWithSerenity.class)
9 @CucumberOptions(features = "src/test/resources/features/wordpress.feature",
10                 glue = "co.com.choucair.automation.android.stepdefinitions",
11                 snippets = SnippetType.CAMEL_CASE)
12 public class RunnerTags {
13 }
14
```

ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

8. Revisamos la ejecución en la parte Run y tendremos los métodos recomendados por Cucumber para implementar en nuestra próxima clase StepDefinitions.



9. Copiamos los métodos recomendados por cucumber.

```
@Given("^that Brandon wants to enter the Wordpress application$")
public void thatBrandonWantsToEnterTheWordpressApplication() {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

@When("^I login with the username \"([^\"]*)\" and the password \"([^\"]*)\"$")
public void iLoginWithTheUsernameAndThePassword(String arg1, String arg2) {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}

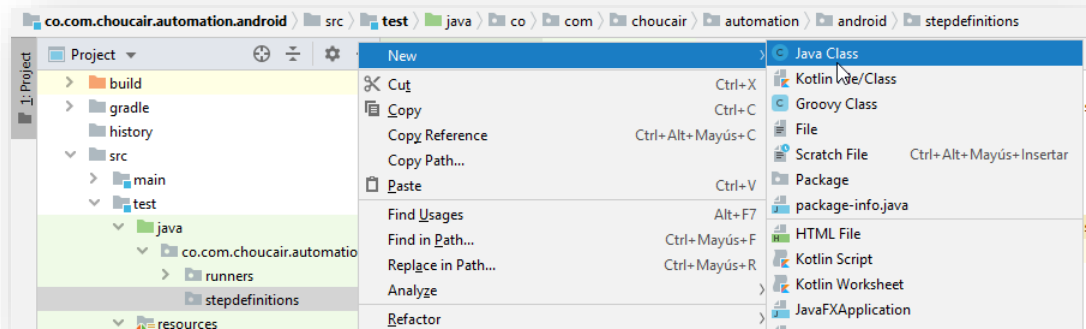
@Then("^you should see the login in the application with the message Logged in as$")
public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs() {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();
}
```



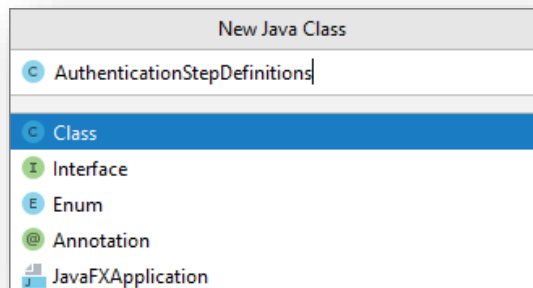
ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

10. Crea una clase en el paquete **stepdefinitions** donde definiremos los métodos generados por cucumber, por ejemplo, “**AuthenticationStepDefinitions**”.

a. Clic derecho sobre el paquete **stepdefinitions** > **New** > **Java Class**.



b. Asignamos el nombre deseado para nuestra clase stepdefinitions.



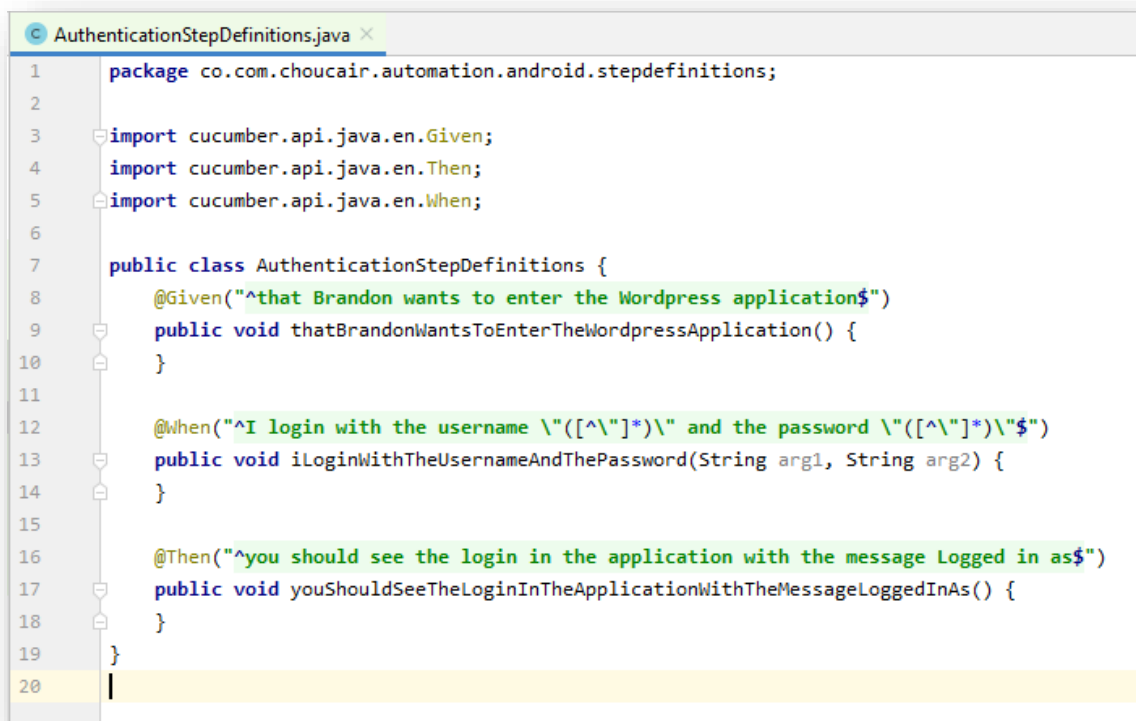
c. Verificamos que nos crea la clase como se ve en la siguiente imagen.

A screenshot of the IDE showing the content of the newly created file 'AuthenticationStepDefinitions.java'. The code is as follows:

```
1 package co.com.choucair.automation.android.stepdefinitions;
2
3 public class AuthenticationStepDefinitions {
4 }
5
```

ScreenPlay +BDD+Appium+Android 2 - Configuración Inicial, Feature y StepDefinitons

11. Agregar los métodos propuestos a la clase de definiciones creada. Y nos debe quedar así después de pegar los métodos, importar las etiquetas de cucumber y eliminar las excepciones:



```
1 package co.com.choucair.automation.android.stepdefinitions;
2
3 import cucumber.api.java.en.Given;
4 import cucumber.api.java.en.Then;
5 import cucumber.api.java.en.When;
6
7 public class AuthenticationStepDefinitions {
8     @Given("^that Brandon wants to enter the Wordpress application$")
9     public void thatBrandonWantsToEnterTheWordpressApplication() {
10    }
11
12     @When("^I login with the username \"([^\"]*)\" and the password \"([^\"]*)\"$")
13     public void iLoginWithTheUsernameAndThePassword(String arg1, String arg2) {
14    }
15
16     @Then("^you should see the login in the application with the message Logged in as$")
17     public void youShouldSeeTheLoginInTheApplicationWithTheMessageLoggedInAs() {
18    }
19 }
20
```

Como se observa en la imagen anterior, cada línea Gherkin en la feature se relaciona con una anotación en la clase AuthenticationStepDefinitions.

Observe como en siguiente línea

```
@When("^I login with the username \"([^\"]*)\" and the password \"([^\"]*)\"$")
```

los parámetros son interpretados a través de expresiones.

Nos vemos en la próxima Guía...