

SERENITY BDD con CUCUMBER

5

Implementando pasos y objetos III

(Tiempo ejecución 4 horas)

Recordemos....

Un modelo BDD está compuesto por la siguiente estructura de conexión

Feature: cada línea de un scenario mapea con un método en la clase .definition

Definition: cada método mapea con unos métodos en la clase .steps

Steps: para la construcción de los pasos es necesaria la definición de objetos en .pageobjects

Pageobjects: contiene las clases con la definición de los objetos

Este es nuestro requisito:

Historia de Usuario: Verificar el diligenciamiento de la pantalla “Popup Validation”.

Criterios de Aceptación:

- Verificar diligenciamiento exitoso.
- Verificar mensaje de validación para cada campo

url de prueba: <https://colorlib.com/polygon/metis/login.html>

Pasos para la ejecución de la prueba.

1. Autenticacion en colorlib

- Abrir navegador con la url de prueba
- Ingresar usuario demo
- Ingresar password demo
- Click en botón Sign in
- Verificar la Autenticación (label en home)

2. Ingresar a Funcionalidad Popup Validation

- Clic en Menu “Forms”
- Clic en submenú “Form Validation”
- Verificación : se presenta pantalla de la funcionalidad con título Popup Validation

3. Diligenciar Formulario Popup Validation

- Diligenciar todos los campos del formulario
- Clic en botón Validate

4. Verificar Respuesta Exitosa/Fallida

Objetivo:

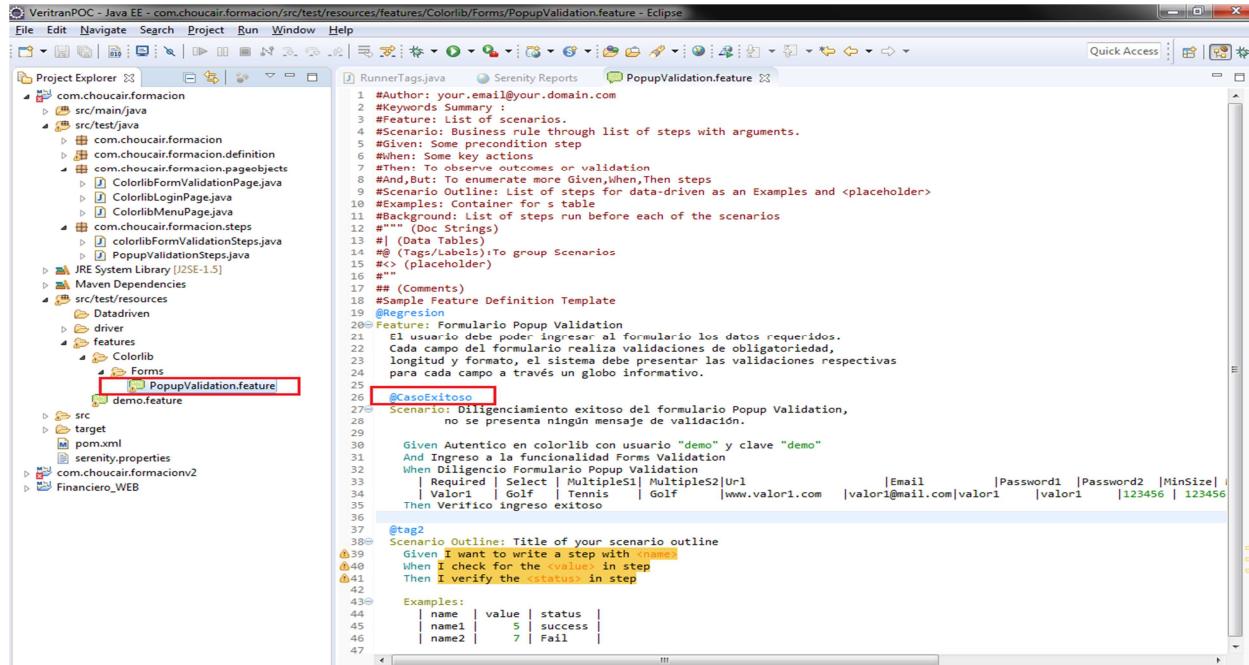
Durante esta sesión vamos a enfocarnos en implementar el código requerido para las acciones 2, 3 y 4.



Objetivo:

Crear el caso alterno para el diligenciamiento del formulario popup validation.

Recordemos como tenemos organizada nuestra **feature** hasta el momento:



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for 'VeritanPOC - Java EE - com.choucair.formacion'. It includes packages like 'com.choucair.formacion', 'com.choucair.formacion.definition', 'com.choucair.formacion.pageobjects', and 'com.choucair.formacion.steps'. A file 'PopupValidationSteps.java' is also visible.
- Serenity Editor:** The 'PopupValidation.feature' file is open. The code is as follows:

```

1 #Author: your.email@your.domain.com
2 #Keywords Summary :
3 #Feature: List of scenarios.
4 #Scenario: Business rule through list of steps with arguments.
5 #Given: Some precondition step
6 #When: Some key actions
7 #Then: Some outcomes or validation
8 #And,But: To enumerate more Given,when,Then steps
9 #Scenario Outline: List of steps for data-driven as an Examples and <placeholder>
10 #Examples: Container for a table
11 #Background: List of steps run before each of the scenarios
12 #"" (Doc Strings)
13 #| (Data Tables)
14 #| (Tags/Labels): To group Scenarios
15 #<> (placeholder)
16 #"
17 # (Comments)
18 #Sample Feature Definition Template
19 @Regression
20 Feature: Formulario Popup Validation
21 Given Autentico en colorlib con usuario "demo" y clave "demo"
22 Cada campo del formulario realiza validaciones de obligatoriedad,
23 longitud y formato el sistema debe presentar las validaciones respectivas
24 para cada campo a través un globo informativo.
25
26 @CasoExitoso
27 Scenario: Diligenciamiento exitoso del formulario Popup Validation,
28 no se presenta ningún mensaje de validación.
29
30 Given Autentico en colorlib con usuario "demo" y clave "demo"
31 And Ingreso a la funcionalidad Forms Validation
32 When Diligencio Formulario Popup Validation
33 | Required | Select | MultipleS1| MultipleS2|Url
34 | Valor1 | Golf | Tennis | Golf | www.valor1.com | valor1@mail.com|valor1 | valor1 | 123456 | 123456
35 Then Verifco ingreso exitoso
36
37 @tag2
38 Scenario Outline: Title of your scenario outline
39 Given I want to write a step with <name>
40 When I check for the <value> in step
41 Then I verify the <status> in step
42
43 Examples:
44 | name | value | status |
45 | name1 | 5 | success |
46 | name2 | 7 | Fail |
47

```

Tenemos una **feature** con un **Scenario** identificado con tag **@CasoExitoso** con el cual podemos filtrar nuestra ejecución a través de la clase **RunnerTags**.

Este escenario lo denominamos **@CasoExitoso**, debido a que hace referencia a la verificación del flujo primario de nuestra transacción, ahora debemos crear el caso alterno, que consiste en verificar que el sistema presente un globo informativo de validación al momento de presentarse un error en el ingreso de los datos para cada campo.

Nuestra estrategia de prueba consistirá en combinar los datos de tal forma que sólo se nos presente un globo informativo para cada ejecución que realicemos, de esta forma tendremos un control sobre nuestra prueba.

Si analizamos, el flujo del **@CasoExitoso**, comparte la gran mayoría de los pasos con el alterno, la única variación se presenta en el paso final "Then", por lo que vamos a reutilizar gran parte de nuestra automatización.



Comencemos.....

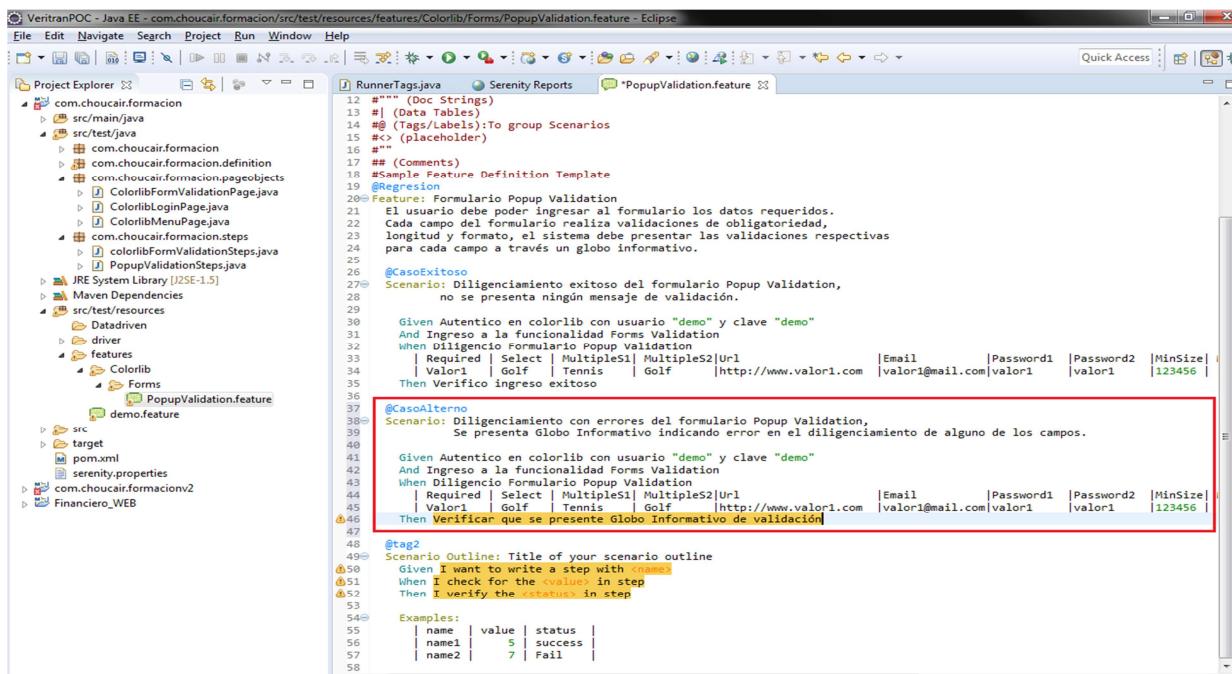
Paso 1, Crear **Scenario** alterno en la **feature**.

Paso 1.1, duplicar el scenario [@CasoExitoso](#).

Paso 2.2, modificar la etiqueta [@CasoExitoso](#), por la etiqueta [@CasoAlterno](#).

Paso 3.3, crear el paso “Then”, “Verificar que se presente Globo Informativo de validación”.

En este punto hacemos una anotación, para el ejemplo vamos a agregar un único scenario de prueba con el que validaremos todos los casos alternos, sin embargo, también es viable, crear un scenario para la verificación de cada uno de los campos.



```

Feature: Formulario Popup Validation
  El usuario debe poder ingresar al formulario los datos requeridos.
  Cada campo del formulario realiza validaciones de obligatoriedad,
  longitud y formato, el sistema debe presentar las validaciones respectivas
  para cada campo a través de un globo informativo.

  @CasoExitoso
  Scenario: Diligenciamiento exitoso del formulario Popup Validation,
  no se presenta ningún mensaje de validación.

  Given Autentico en colorlib con usuario "demo" y clave "demo"
  And Ingreso a la funcionalidad Forms Validation
  When Diligencio Formulario Popup Validation
    | Required | Select | MultipleS1| MultipleS2|Url
    | Valor1 | Golf | Tennis | Golf | http://www.valor1.com |valor1@mail.com|valor1 |valor1 |123456 |
  Then Verifico ingreso exitoso

  @CasoAlterno
  Scenario: Diligenciamiento con errores del formulario Popup Validation,
  Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de los campos.

  Given Autentico en colorlib con usuario "demo" y clave "demo"
  And Ingreso a la funcionalidad Forms Validation
  When Diligencio Formulario Popup Validation
    | Required | Select | MultipleS1| MultipleS2|Url
    | Valor1 | Golf | Tennis | Golf | http://www.valor1.com |valor1@mail.com|valor1 |valor1 |123456 |
  Then Verifico que se presente Globo Informativo de validación

  @tag2
  Scenario Outline: Title of your scenario outline
  Given I want to write a step with Scenario Outline
  When I check for the value in step
  Then I verify the status in step

  Examples:
  | name | value | status |
  | name1 | 5 | success |
  | name2 | 7 | Fail |

```

Paso 3.4, Generar el método a agregar en la definition, para este caso vamos a comentar todos los pasos previos al que deseamos implementar, así:

```

@CasoAlterno
Scenario: Diligenciamiento con errores del formulario Popup Validation,
Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de los campos.

# Given Autentico en colorlib con usuario "demo" y clave "demo"
# And Ingreso a la funcionalidad Forms Validation
# When Diligencio Formulario Popup Validation
#   | Required | Select | MultipleS1| MultipleS2|Url
#   | Valor1 | Golf | Tennis | Golf | http://www.valor1.com |valor1@mail.com|valor1 |valor1 |123456 |
Then Verificar que se presente Globo Informativo de validación

```

Esto es temporal mientras realizamos la ejecución.

Paso 3.5, ejecutar la clase RunnerTags, modificando el campo tags por la etiqueta [@CasoAlterno](#).



```

package com.choucair.formacion;
import org.junit.runner.RunWith;
import cucumber.api.junit.CucumberWithSerenity;
import cucumber.api.CucumberOptions;

@RunWith(CucumberWithSerenity.class)
@CucumberOptions (features = "src/test/resources/features/", tags = "@SmokeTest")
@CucumberOptions (features = "src/test/resources/features/Colorlib/Forms/PopupValidation.feature", tags = "@CasoAlterno")
public class RunnerTags {
}

```



Paso 3.6, Consulte la pestaña “Console”, en la cual cucumber nos propone el método a implementar.

Project Explorer

File Edit Source Refactor Navigate Search Project Run Window Help

RunnerTags.java Serenity Reports PopupValidation.feature

```
1 package com.choucair.formacion;
2
3 import org.junit.runner.RunWith;
4
5 @RunWith(CucumberWithSerenity.class)
6 // @CucumberOptions(features = "src/test/resources/features/", tags = "@SmokeTest")
7 // @CucumberOptions(features = "src/test/resources/features/Colorlib/Forms/PopupValidation.feature", tags = "@CasoAlternativo")
8 public class RunnerTags {
9
10 }
```

JRE System Library [J2SE-1.8]

Maven Dependencies

src/test/resources

- Datadriven
- drivers
- features
 - Colorlib
 - Forms
 - PopupValidation.feature

src

pom.xml

serenity.properties

com.choucair.formacion2

Financiero_WEB

Markers Properties Servers Data Source Explorer Snippets Console JUnit Terminal Debug Expressions

```
<terminated> RunnerTags (2) [JUnit] C:\Program Files\Java\jdk1.8.0_144\bin\javaw.exe (25/01/2018, 5:15:46 p.m.)
1175 [main] INFO net.serenitybdd.core.Serenity -
```

TEST PENDING

TEST PENDING: Diligenciamiento con errores del formulario Popup Validation.

```
1796 [pool-1-thread-1] INFO net.thucydides.core.reports.json.JSONTestOutcomeReporter - Generating JSON report for Diligenciamiento
2638 [pool-1-thread-1] INFO net.thucydides.core.reports.html.HtmlAcceptanceTestReporter - GENERATING HTML REPORT FOR Formulario Pop
```

1 Scenarios (0@33m1 undefined@0m)
2 Steps (0@33m1 undefined@0m)
0m8,000s

You can implement missing steps with the snippets below:

```
@Then("^Verificar que se presente Globo Informativo de validación$")
public void validar_que_se_presente_Globo_Informativo_de_validación() throws Throwable {
    // Write code here that turns the phrase above into concrete actions
    throw new PendingException();}
```

Picked up JAVA_TOOL_OPTIONS: -agentlib:jvmhook
Picked up _JAVA_OPTIONS: -Xrunkvmhook -Xbootclasspath/a:"C:\Program Files (x86)\HP\Unified Functional Testing\bin\java_shared\class

Nota: el objetivo de comentar los pasos “#”, era para que el sistema no ejecutar la automatización, sino sólo el paso nuevo, por lo que debemos retirar los comentarios nuevamente en el Scenario.

Paso 2, Implementar paso en la **definition**.

Paso 2.1, Copiar el método propuesto por cucumber en la clase “*PopupValidationDefinition*” del paquete “*com.choucair.formacion.definition*”.

Paso 2.2, limpiar el método.

Paso 2.3, agregar el llamado al paso “`verificar_ingreso_datos_formulario_con_errores()`” de la clase “`colorlibFormValidationSteps`” que habíamos dejado implementado en la guía pasada:

```
③ @Then("^Verifico ingreso exitoso$")
public void verifico_ingreso_exitoso() {
    colorlibFormValidationSteps.verificar_ingreso_datos_formulario_exitoso();
}

④ @Then("^Verificar que se presente Globo Informativo de validación$")
public void verificar_que_se_presente_Globo_Informativo_de_validación() {
    colorlibFormValidationSteps.verificar_ingreso_datos_formulario_con_erroros();
}
```

Paso 3, Consultar los steps.

Para verificar la implementación de los pasos, de clic sobre el método y luego F3 para ir a la clase "**colorlibFormValidationSteps**", como vemos ya se encontraba implementado el llamado al método "**form_con_errores**" en la clase "**colorlibFormValidationPage**".

```
    @Step
    public void verificar_ingreso_datos_formulario_con_errores() {
        colorlibFormValidationPage.form_con_errores();
    }
```



Paso 4, Consultar el pageobject.

Para verificar la implementación de los pasos, de clic sobre el método y luego F3 para ir a la clase "*ColorlibFormValidationPage*", como vemos ya se encontraba implementado el llamado al método "*form_con_erroros*" en la clase "*colorlibFormValidationPage*".

```
public void form_con_erroros() {
    assertThat(globoInformativo.isCurrentlyVisible(), is(true));
}
```

Resumen:



```

@CasoAlterno
Scenario: Diligenciamiento con errores del formulario Popup Validation,
Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de los campos.

# Given Autentico en colorlib con usuario "demo" y clave "demo"
# And Ingreso a la funcionalidad Forms Validation
# When Diligencio Formulario Popup Validation
# | Required | Select | MultipleS1| MultipleS2|Url          |Email           |Password1 |Password2 |MinSize
# | Valor1  | Golf   | Tennis   | Golf    |http://www.valor1.com |valor1@mail.com|valor1     |valor1     |123456
Then Verificar que se presente Globo Informativo de validación

com.choucair.formacion.definition
  PopupValidationDefinition.java
    @Then("^Verificar que se presente Globo Informativo de validación$")
    public void verificar_que_se_presente_Globo_Informativo_de_validación() {
        colorlibFormValidationSteps.verificar_ingreso_datos_formulario_con_erroros();
    }

com.choucair.formacion.steps
  colorlibFormValidationSteps.java
    @Step
    public void verificar_ingreso_datos_formulario_con_erroros() {
        colorlibFormValidationPage.form_con_erroros();
    }

com.choucair.formacion.pageobjects
  ColorlibFormValidationPage.java
    public void form_con_erroros() {
        assertThat(globoInformativo.isCurrentlyVisible(), is(true));
    }

```

Paso 5, Ejecutar y verificar evidencias.

Paso 5.1, Antes de ejecutar completa la historia, recuerda quitar los comentarios en la feature. (#).

Paso 5.2, modificar la data de prueba.

La tabla de datos que tenemos adjunta es una copia del escenario exitoso, es por esto que vamos a ajustar los datos para obtener el resultado esperado, es decir que falle por un campo y se presente el globo informativo.

Para el ejemplo iniciemos probando el campo Required, borremos el contenido del primer campo del datatable.

```

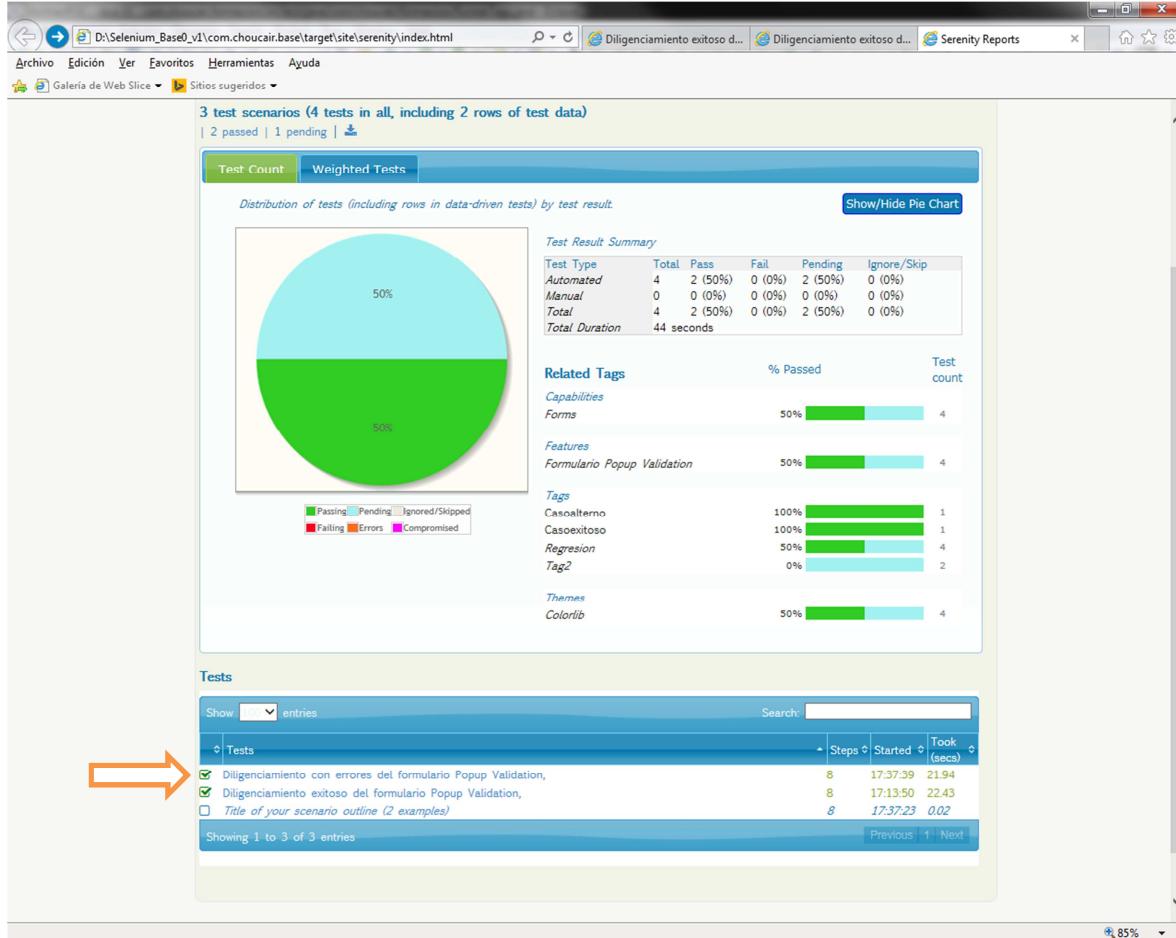
@CasoAlterno
Scenario: Diligenciamiento con errores del formulario Popup Validation,
Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de los campos.

Given Autentico en colorlib con usuario "demo" y clave "demo"
And Ingreso a la funcionalidad Forms Validation
When Diligencio Formulario Popup Validation
| Required | Select | MultipleS1| MultipleS2|Url          |Email           |Password1 |Password2 |MinSize|
|  | Golf   | Tennis   | Golf    |http://www.valor1.com |valor1@mail.com|valor1     |valor1     |123456 |
Then Verificar que se presente Globo Informativo de validación

```



Paso 5.3, ir a la clase RunnerTags y ejecutar
 Paso 5.4, generar y verificar evidencias



D:\Selenium_Base0_v1\com.choucair.base\target\site\serenity\formulario-popup-validation_diligenciamiento_exitoso_d... | Diligenciamiento exitoso d... | Diligenciamiento exitoso d... | Diligenciamiento con errores d...

Archivo Edición Ver Favoritos Herramientas Ayuda

Galería de Web Slice Sítios sugeridos

Serenity BDD

Proyecto base para Automatización

Home > Colorib > Forms > Formulario Popup Validation > Diligenciamiento con errores del formulario Popup Validation,

Overall Test Results Requirements Themes Capabilities Features

Report generated 25-01-2018 17:37

Formulario Popup Validation

El usuario debe poder ingresar al formulario los datos requeridos. Cada campo del formulario realiza validaciones de obligatoriedad, longitud y formato, el sistema debe presentar las validaciones respectivas para cada campo a través un globo informativo.

Diligenciamiento Con Errores Del Formulario Popup Validation,

Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de los campos.

Steps	Screenshot	Outcome	Duration
Given Autentico en colorib con usuario "demo" y clave "demo"		SUCCESS	11.84s
And Ingreso a la funcionalidad Forms Validation		SUCCESS	2.08s
When Diligencio Formulario Popup Validation		SUCCESS	6.9s
Then Verificar que se presente Globo Informativo de validación			
Verificar ingreso datos formulario con errores			

file:///D:/Selenium_Base0_v1\com.choucair.base\target\site\serenity\formulario-popup-validation_diligenciamiento-con-errores-del-formulario-popup-validation_comma_screenshots.html#screenshots?Screenshot=6



Ejecutar varias iteraciones del mismo Escenario.

Hasta el momento tenemos nuestro caso alterno ejecutando de forma correcta, así que vamos a agregar más iteraciones al mismo escenario, es decir una combinatoria de datos que nos ayude a verificar el mismo caso de prueba, así que dupliquemos la primera línea de datos en nuestro datatable y adecuemos la data para que falle en segunda instancia por el campo “Select”, para esto ingresamos el valor “Choose a sport”.

```

@CasoAlterno
Scenario: Diligenciamiento con errores del formulario Popup Validation,
Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de los campos.

Given Autentico en colorlib con usuario "demo" y clave "demo"
And Ingreso a la funcionalidad Forms Validation
When Diligencio Formulario Popup Validation
| Required | Select | MultipleS1| MultipleS2|Url          |Email           |Password1 |Password2 |MinSize| Ma
|          | Golf  | Tennis | Golf      |http://www.valor1.com |valor1@mail.com|valor1    |valor1    |123456 | 12
| Valor1  | Choose a sport | Tennis | Golf      |http://www.valor1.com |valor1@mail.com|valor1    |valor1    |1
Then Verificar que se presente Globo Informativo de validación
  
```

Listo ahora vamos a modificar el método “diligencio_Formulario_Popup_Validation” para que ejecute el paso de diligenciar el formulario de forma iterativa acorde a la cantidad de filas del dataTable, para esto agregaremos un for como el de la imagen:

<pre> com.choucair.formacion.definition > PopupValidationDefinition.java </pre>	<pre> @When("^Diligencio Formulario Popup Validation\$") public void diligencio_Formulario_Popup_Validation(DataTable dtDatosForm) { List<List<String>> data = dtDatosForm.raw(); for(int i=1; i<data.size(); i++){ colorlibFormValidationSteps.diligenciar_popup_datos_tabla(data, i); try { Thread.sleep(5000); }catch(InterruptedException e){} } } </pre>
--	--

Analicemos el código:

Se crea un objeto “data” tipo “List<>” que contendrá toda la matriz.

Se crea un ciclo que recorrerá el objeto hasta el tamaño de la matriz

Se enviar por parámetros al steps la data y el nro de la fila a leer (i)

Se agrega un delay (Thread.sleep()); para hacer una espera entre ejecuciones, el tiempo está dado en milisegundos.

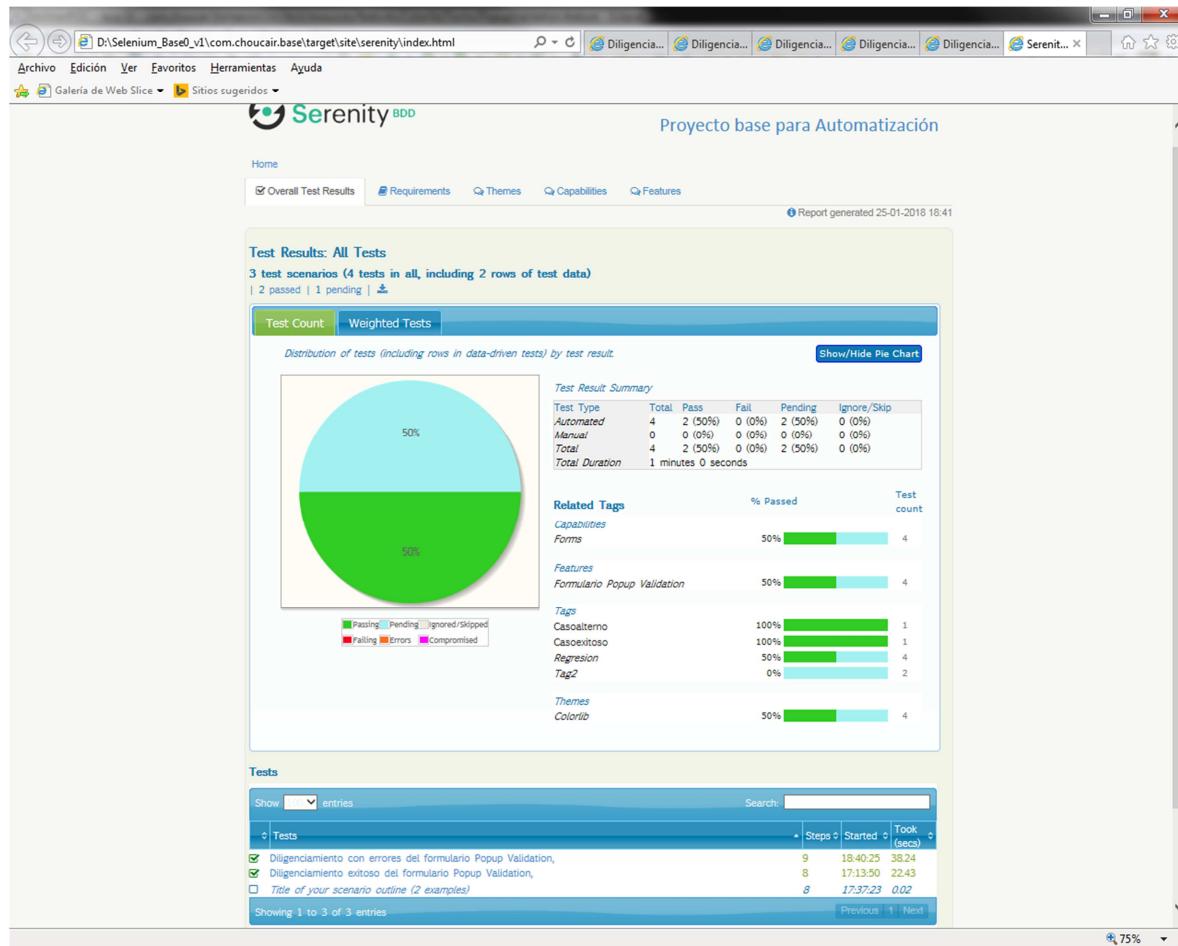


Grabar y ejecutar nuevamente:

El script debe ejecutar de forma iterativa el paso de diligenciamiento del formulario, para cada fila del datatable adjunto en la feature.

Revisar evidencia:

Como se observa en el reporte de evidencia, se generaron dos iteraciones para el mismo paso de diligenciamiento.



El usuario debe poder ingresar al formulario los datos requeridos. Cada campo del formulario realiza validaciones de obligatoriedad, longitud y formato, el sistema debe presentar las validaciones respectivas para cada campo a través un globo informativo.

Diligenciamiento Con Errores Del Formulario Popup Validation,

Se presenta Globo Informativo indicando error en el diligenciamiento de alguno de Los campos.

Steps	Screenshot	Outcome	Duration																						
Given Autentico en colorlib con usuario "demo" y clave "demo"		SUCCESS	17.39s																						
And Ingreso a la funcionalidad Forms Validation		SUCCESS	2.27s																						
When Diligencio Formulario Popup Validation	<table border="1"> <thead> <tr> <th>Required</th> <th>Select</th> <th>MultipleS1</th> <th>MultipleS2</th> <th>Url</th> <th>Email</th> <th>Password1</th> </tr> </thead> <tbody> <tr> <td></td> <td></td> <td>Golf</td> <td>Tennis</td> <td>Golf</td> <td>http://www.valor1.com</td> <td>valor1@mail.com</td> <td>valor1</td> </tr> <tr> <td>Valor1</td> <td>Choose a sport</td> <td>Tennis</td> <td>Golf</td> <td>http://www.valor1.com</td> <td>valor1@mail.com</td> <td>valor1</td> </tr> </tbody> </table> <p>Diligenciar popup datos tabla: [[Required, Select, MultipleS1, MultipleS2, Url, Email, Password1, Password2, MinSize, MaxSize, Number, IP, Date, DateEarlier], [Golf, Tennis, Golf, http://www.valor1.com, valor1@mail.com, valor1, 123456, 123456, -99.99, 200.200.54, 2018-01-22, 2012/09/12], [Valor1, Choose a sport, Tennis, Golf, http://www.valor1.com, valor1@mail.com, valor1, valor1, 123456, 123456, -99.99, 200.200.54, 2018-01-22, 2012/09/12], [Valor1, Choose a sport, Tennis, Golf, http://www.valor1.com, valor1@mail.com, valor1, valor1, 123456, 123456, -99.99, 200.200.54, 2018-01-22, 2012/09/12], 1]</p> <p>Diligenciar popup datos tabla: [[Required, Select, MultipleS1, MultipleS2, Url, Email, Password1, Password2, MinSize, MaxSize, Number, IP, Date, DateEarlier], [Golf, Tennis, Golf, http://www.valor1.com, valor1@mail.com, valor1, valor1, 123456, 123456, -99.99, 200.200.54, 2018-01-22, 2012/09/12], [Valor1, Choose a sport, Tennis, Golf, http://www.valor1.com, valor1@mail.com, valor1, valor1, 123456, 123456, -99.99, 200.200.54, 2018-01-22, 2012/09/12], [Valor1, Choose a sport, Tennis, Golf, http://www.valor1.com, valor1@mail.com, valor1, valor1, 123456, 123456, -99.99, 200.200.54, 2018-01-22, 2012/09/12], 2]</p>			Required	Select	MultipleS1	MultipleS2	Url	Email	Password1			Golf	Tennis	Golf	http://www.valor1.com	valor1@mail.com	valor1	Valor1	Choose a sport	Tennis	Golf	http://www.valor1.com	valor1@mail.com	valor1
Required	Select	MultipleS1	MultipleS2	Url	Email	Password1																			
		Golf	Tennis	Golf	http://www.valor1.com	valor1@mail.com	valor1																		
Valor1	Choose a sport	Tennis	Golf	http://www.valor1.com	valor1@mail.com	valor1																			
Then Verificar que se presente Globo Informativo de validación		SUCCESS	0.75s																						

Serenity BDD version 1.51

75%



Hasta el momento nuestro RunnerTags nos ha permitido ejecutar tanto el `@CasoExitoso` y el `@CasoAlterno` de forma individual, si deseamos ejecutar toda la prueba, podemos usar el tag `@Regresion` que se encuentra al inicio de la feature.

Nota: borremos de la feature el Scenario de ejemplo que aún no se ha implementado.

El RunnerTags, quedaría así:

```
1 package com.choucair.formacion;
2
3 import org.junit.runner.RunWith;
4
5 @RunWith(CucumberWithSerenity.class)
6 // @CucumberOptions (features = "src/test/resources/features/", tags = "@SmokeTest")
7 // @CucumberOptions (features = "src/test/resources/features/Colorlib/Forms/PopupValidation.feature", tags = "@Regression")
8 public class RunnerTags {
9
10 }
```

Ejecute y observe el resultado.

