

Manual de Usuario



- 1 Instalación de
Plugins
- 2 Clonar proyecto
Reubicación de
la demo
- 3 Abrir y
configurar
proyecto
- 4
- 5 Ejecución de
Proyecto

Índice

Introducción

A continuación, se redacta una lista detallada de ejecución de las pruebas automatizadas bajo la herramienta **IntelliJ idea**.

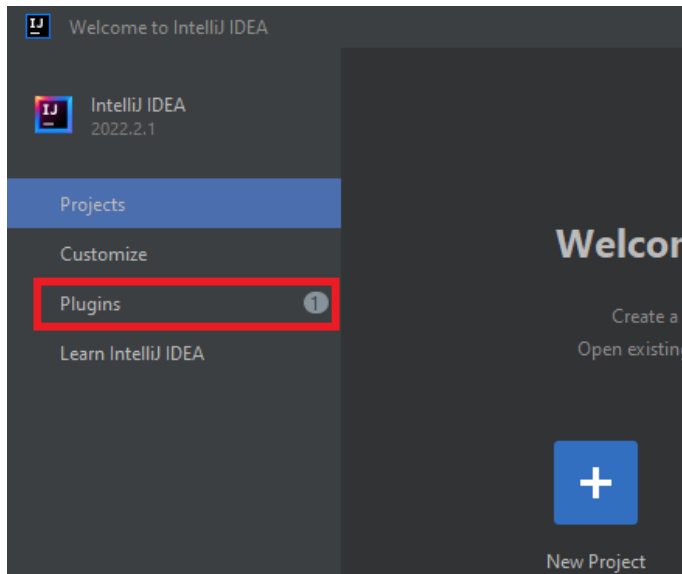
- **IntelliJ idea – Plugins**
- **Git – Clonar proyecto**
- **Ejecutar Script Feature**
- **Ejecutar Script Class**

De acuerdo a cada uno de estos requisitos mostrados, también se comparte los pasos a seguir para poder tenerlos correctamente instalados y verificados.

IntelliJ Idea

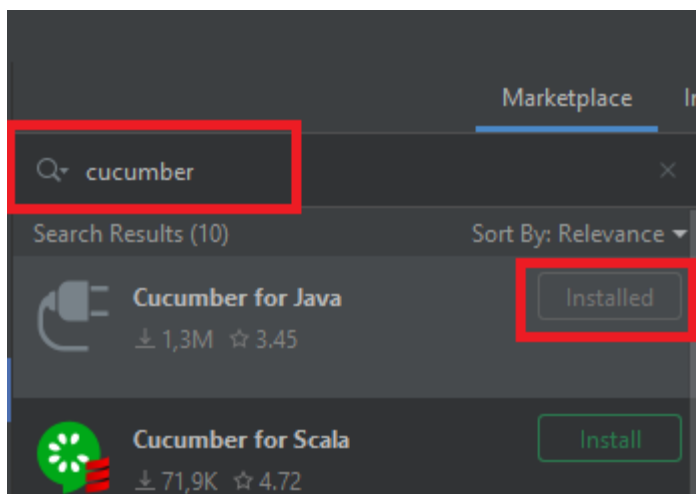
Instalación de Plugins

Para poder inicializar la ejecución del proyecto es necesario instalar algunos complementos para IntelliJ desde la misma herramienta



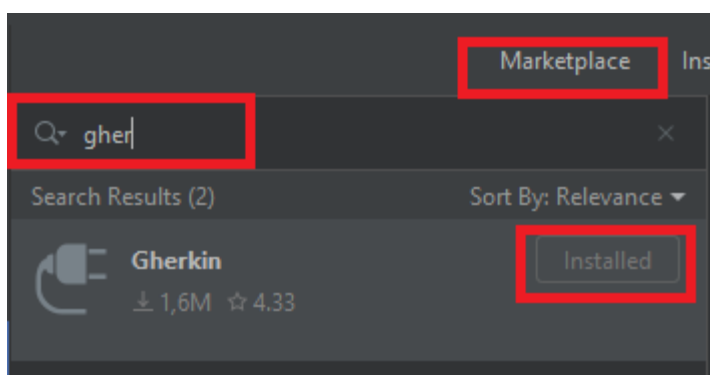
Plugins

Al iniciar el IDE de IntelliJ IDEA tenemos las siguientes opciones donde debemos ir a la sección de plugins.



Cucumber for Java

Buscamos en la opción Marketplace el plugin **Cucumber for Java** y presionamos en **Install**, después presionamos en **Restart IDE** esta figura donde indica Install



Gherkin

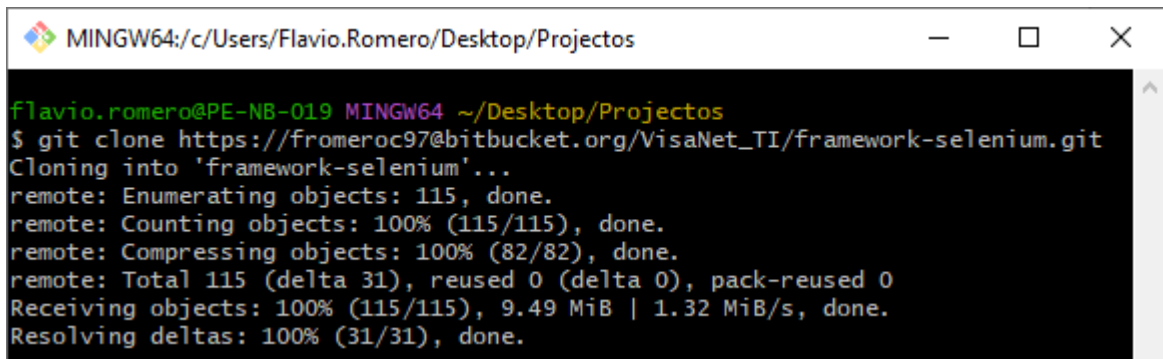
Buscamos en la opción Marketplace el plugin **Gherkin** y presionamos en **Install**, después presionamos en **Restart IDE** esta figura donde indica Install

GIT

Clonación de Proyecto

Para la clonación es necesario tener o saber la ruta donde este alojado el proyecto en el repositorio **Bitbucket**. Para este ejemplo se esta utilizando un repositorio ya existe dentro del repositorio de niubiz.

1. Ubicarnos en la ruta donde se va a clonar el proyecto con el **git bash**
2. Escribimos el siguiente comando **git clone "url_repositorio"**
3. Si el proyecto se encuentra la otra rama que no sea master o main escribir el siguiente comando y obvia el paso 2: **git clone -b my_branch "url_repositorio"**
4. Presionamos **ENTER** y esperamos que termine de clonar el proyecto

A screenshot of a terminal window titled 'MINGW64:/c/Users/Flavio.Romero/Desktop/Proyectos'. The terminal shows the execution of the command 'git clone https://fromeroc97@bitbucket.org/VisaNet_TI/framework-selenium.git'. The output indicates that the repository is being cloned into 'framework-selenium' and shows progress for enumerating, counting, and compressing objects, as well as receiving and resolving deltas. The process completes successfully.

```
MINGW64:/c/Users/Flavio.Romero/Desktop/Proyectos

flavio.romero@PE-NB-019 MINGW64 ~/Desktop/Proyectos
$ git clone https://fromeroc97@bitbucket.org/VisaNet_TI/framework-selenium.git
Cloning into 'framework-selenium'...
remote: Enumerating objects: 115, done.
remote: Counting objects: 100% (115/115), done.
remote: Compressing objects: 100% (82/82), done.
remote: Total 115 (delta 31), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (115/115), 9.49 MiB | 1.32 MiB/s, done.
Resolving deltas: 100% (31/31), done.
```

Xampp

Reubicación del proyecto web

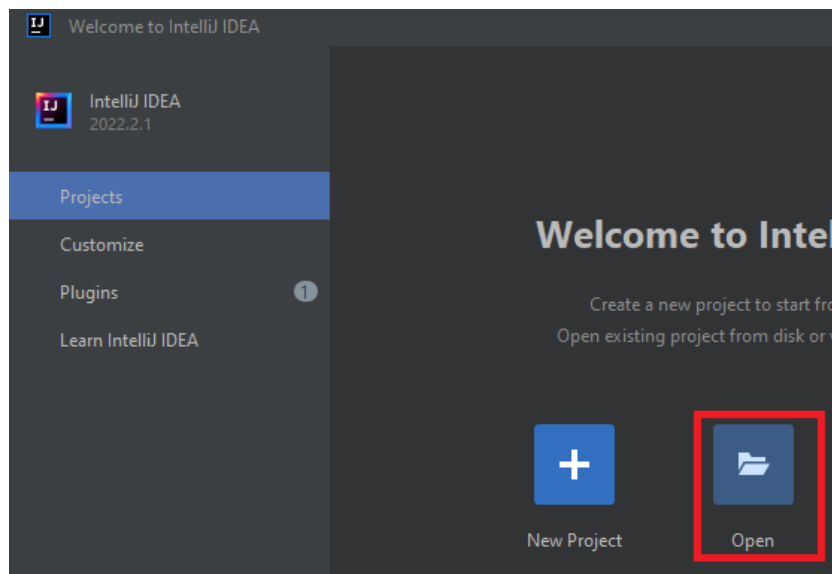
1. Entramos dentro de la carpeta de proyecto y nos vamos a la siguiente ruta:
src\main
2. Copiamos toda la carpeta **web-demo**
3. La movemos a la ruta **C:\xampp\htdocs**

IntelliJ Idea

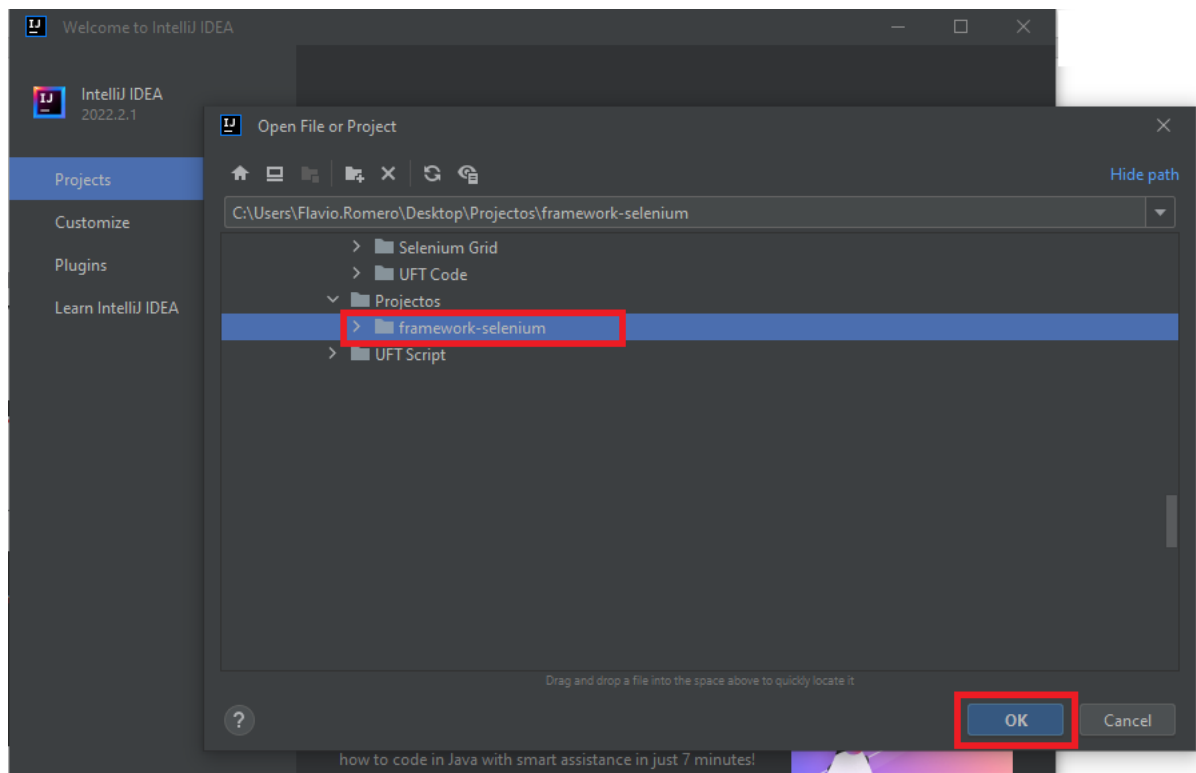
Abrir y configurar proyecto

Para poder inicializar este paso es necesario tener ya clonado el proyecto de vuestro repositorio de proyectos.

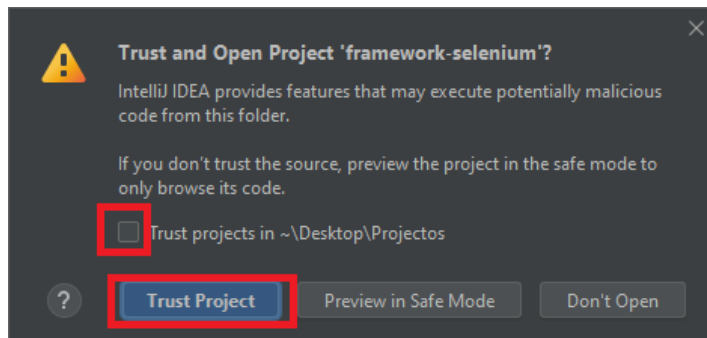
1. Abrimos el IntelliJ IDEA y seleccionamos la opción Open



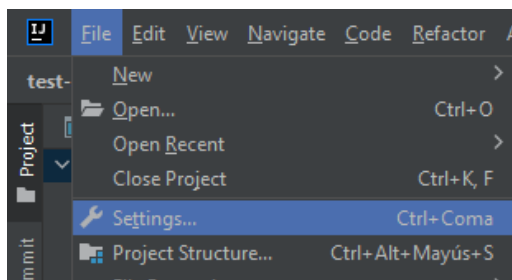
2. Se visualizará una ventana con título “Open File or Project”, aquí buscamos donde hemos alojado el proyecto en su propio equipo y damos OK para continuar.



- Si es la primera vez que estas abriendo el proyecto es posible que te figure la siguiente venta. Solo necesitamos activar el siguiente checkbox y presionar Trust Project.



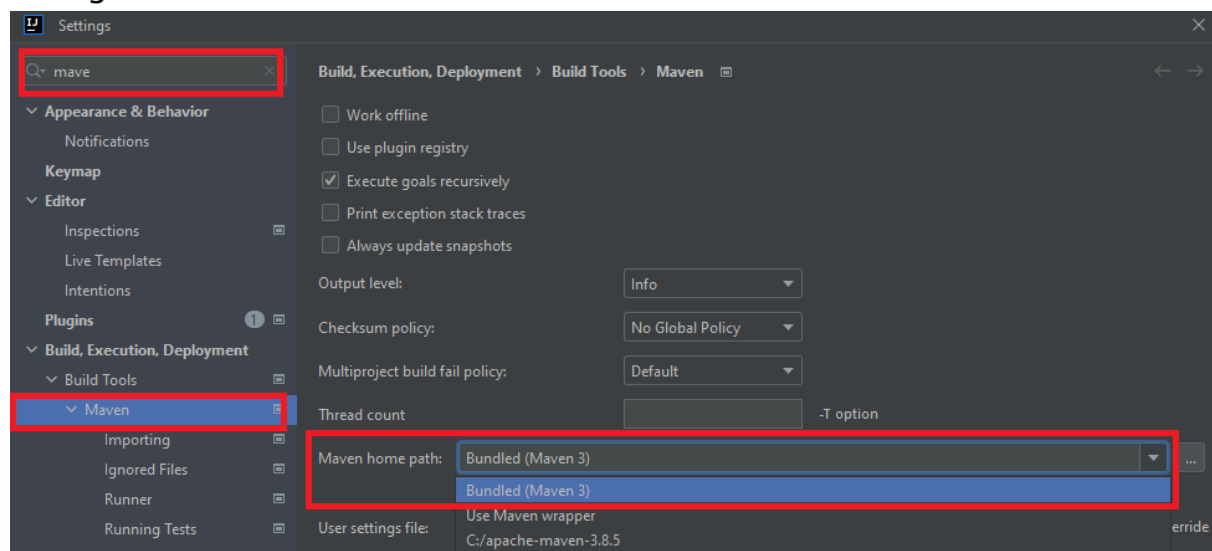
- Una vez cargado el proyecto en el IDE hay dos formas de ejecutarlo por feature o por clase, pero antes de todo ello necesitamos configurar el Maven, JDK y bajar las dependencias del proyecto



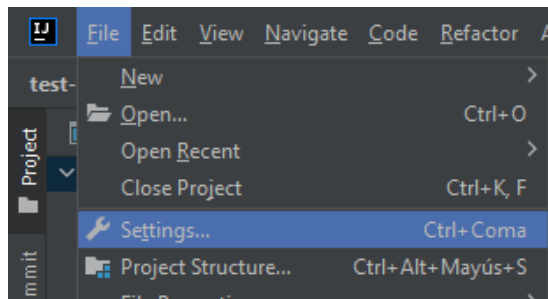
Settings

Buscamos la opción de **Settings** e ingresamos.

- En ajustes buscamos en Maven, **Build Tools/Maven**. Seleccionamos el **Maven home path**: Bundled (Maven3) o el Maven previamente descargado en su equipo y configurado.



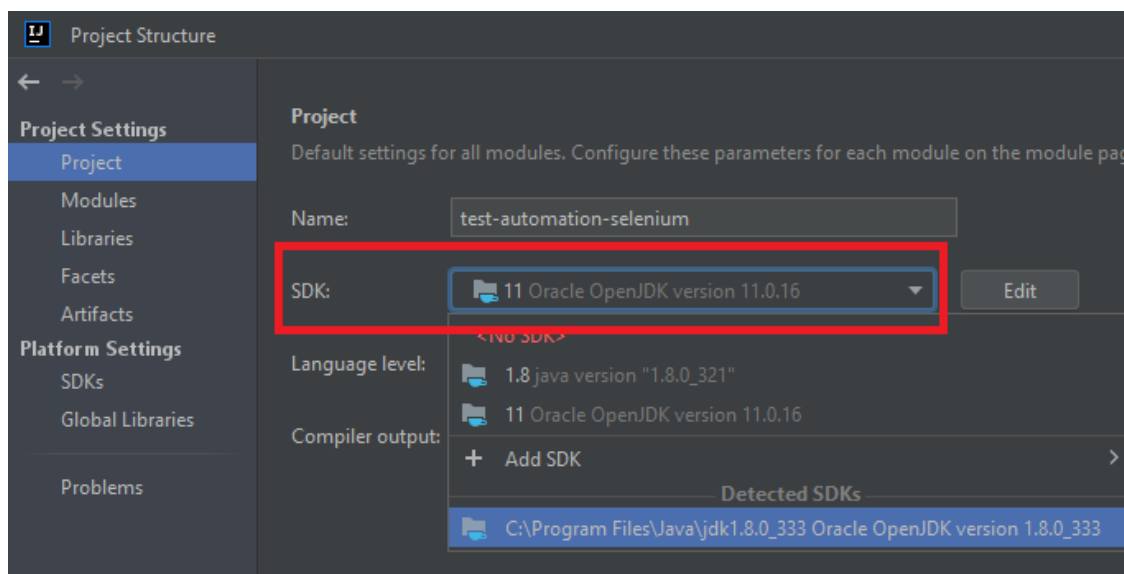
6. Ahora vamos a configurar y validar que tenga el JDK 11 el proyecto



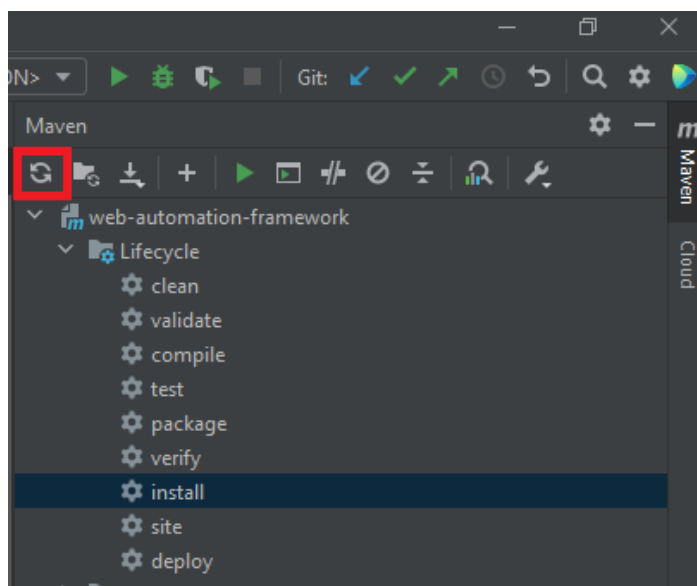
Project Structure

Buscamos la opción de **Project Structure** e ingresamos.

El SDK debe estar el 11 en caso contrario que no haya detectado automáticamente puede agregar su JDK en la opción **ADD SDK**



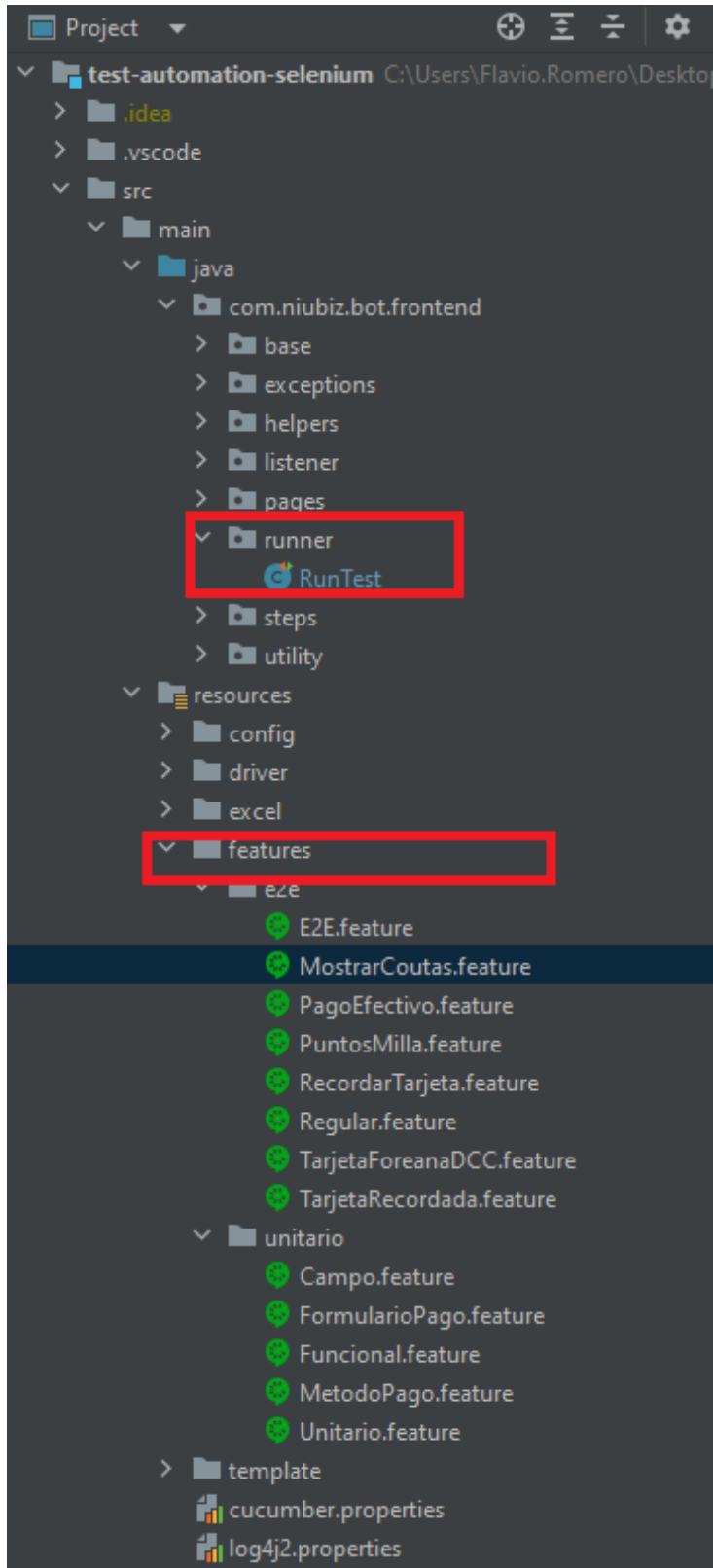
7. Antes de continuar recuerda haber configurado el Maven como variable de entorno o utilizar el propio Maven que viene en el IDE. Ahora vamos a instalar las dependencias



En el lado derecho del IDE se encuentra un botón llamado Maven al presionar se mostrará la ventana donde debemos dar clic en el icono resaltado como de la imagen para que baje las dependencias.

IntelliJ Idea

Ejecución por Feature o RunTest Class



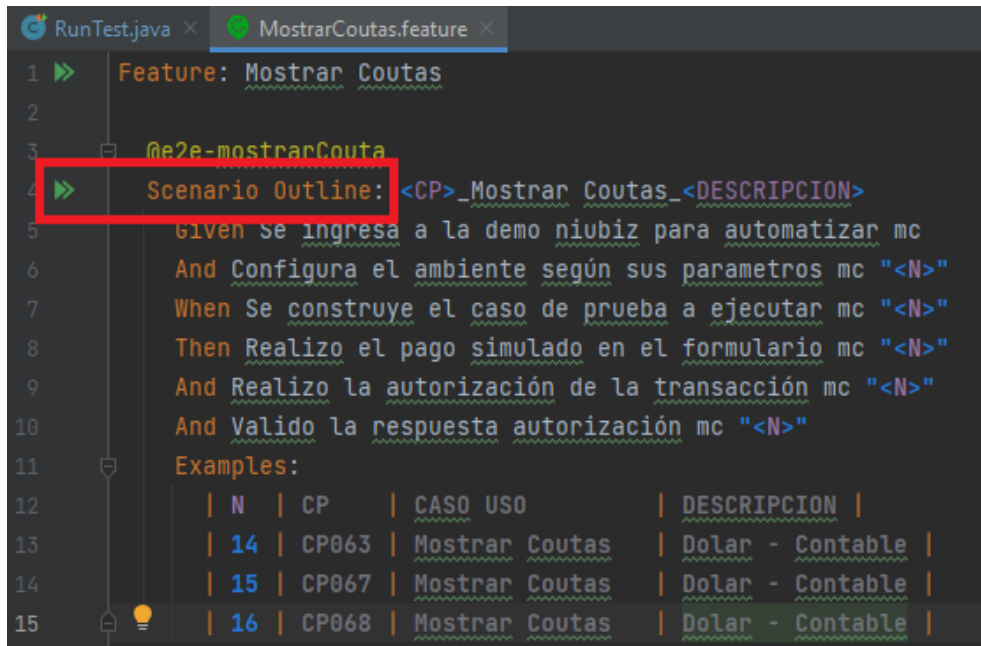
Para ubicar los feature hay que ubicarnos la carpeta **resources/features**. Se encontrar categorizados los feature para e2e como unitario de los diferentes escenarios realizados.

Cada feature tiene un tag por diferentes escenarios.

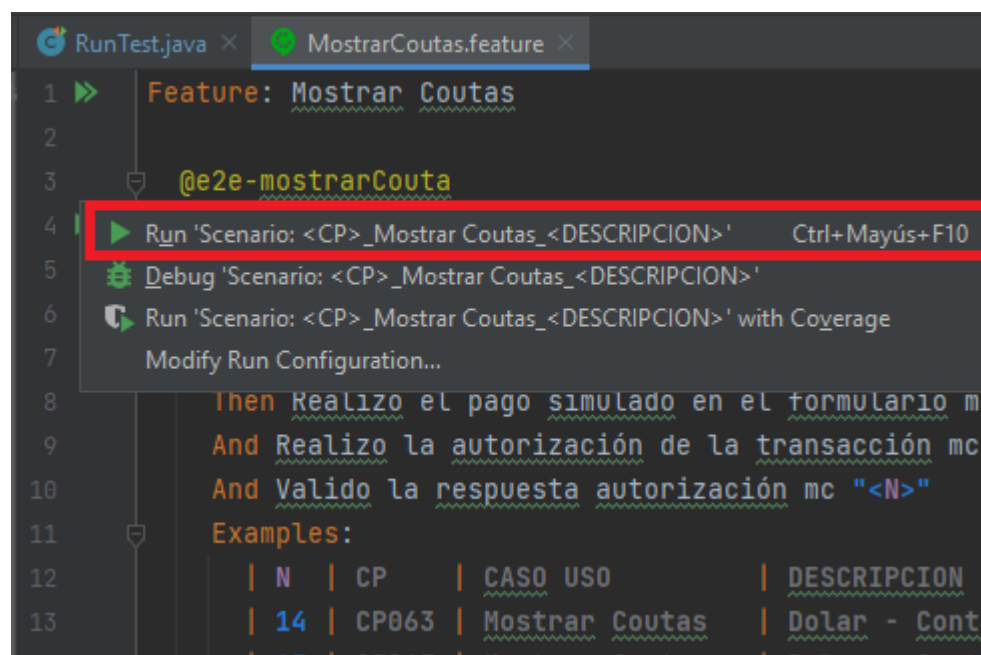
- **E2E.feature:** Es capaz de ejecutar diferentes tipos de escenarios
- **MostrarCouta.feature:** Ejecuta escenario Mostrar Coutas
- **PagoEfectivo.feature:** Ejecuta escenario pago efectivo
- **PuntoMillas.feature:** Ejecuta escenario de puntos bbva y millas ibk
- **RecordarTarjeta.feature:** Ejecuta escenario de recordar tarjeta y recordar tarjeta + ocultar datos
- **Regular.feature:** Ejecuta escenario regular
- **TarjetaForeanaDCC.feature:** Ejecuta escenario de Tarjeta Foreana y DCC
- **TarjetaRecordada.feature:** Ejecuta escenario de tarjeta recordada.

Para los escenarios Ocultar Monto, Ocultar Cerrar son parámetros configurables desde la data.

Para ejecutar desde los archivos feature hay que seleccionar el icono que se encuentra en el lado izquierdo del escenario al dar click se presentara 3 opciones y seleccionar el primero. Esto realizara la ejecución del escenario.



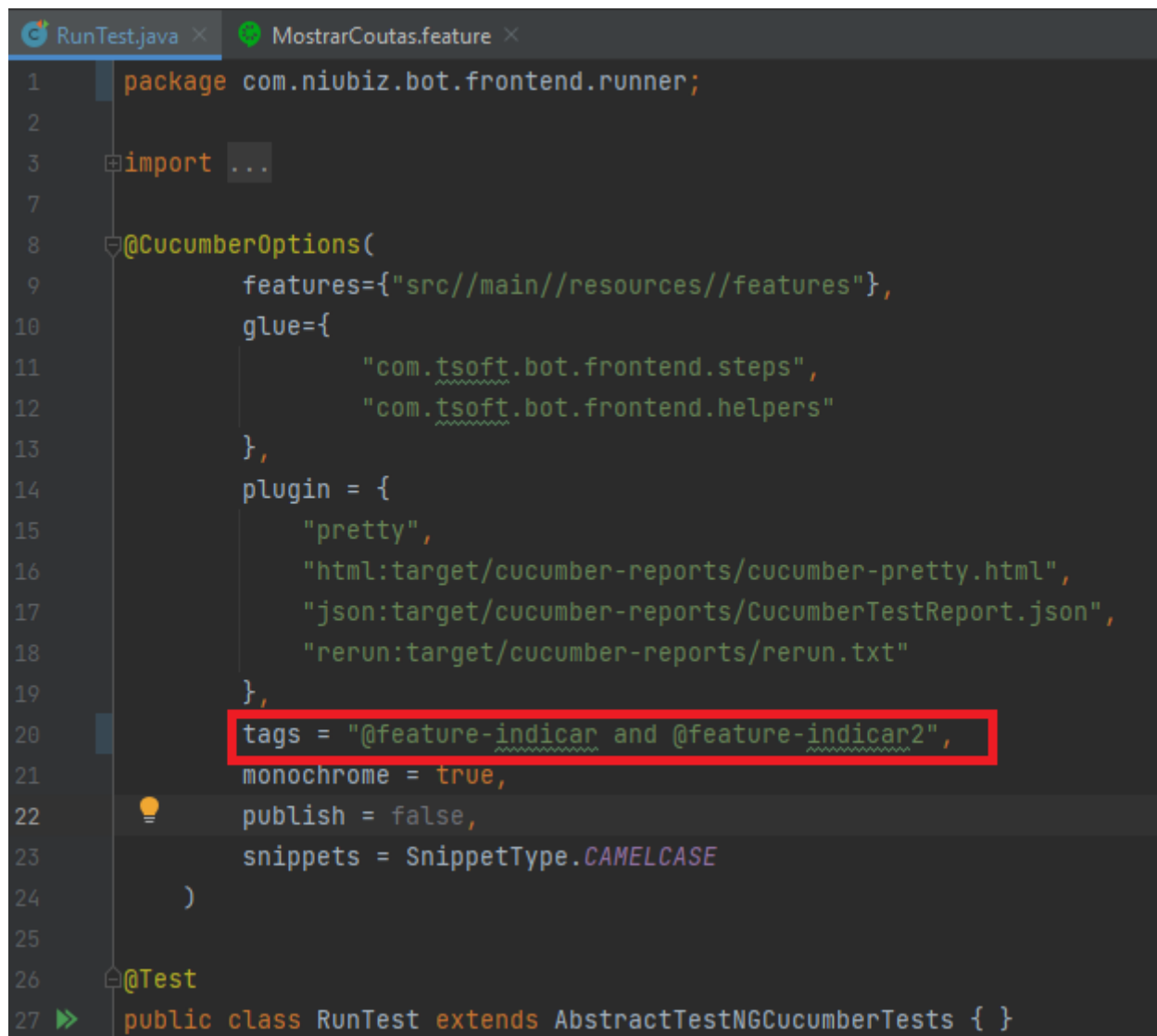
```
1 >> Feature: Mostrar Cutas
2
3 @e2e-mostrarCouta
4 >> Scenario Outline: <CP>_Mostrar Cutas_<DESCRIPCION>
5     Given Se ingresa a la demo niubiz para automatizar mc
6     And Configura el ambiente según sus parametros mc "<N>"
7     When Se construye el caso de prueba a ejecutar mc "<N>"
8     Then Realizo el pago simulado en el formulario mc "<N>"
9     And Realizo la autorización de la transacción mc "<N>"
10    And Valido la respuesta autorización mc "<N>"
11    Examples:
12        | N | CP | CASO USO | DESCRIPCION |
13        | 14 | CP063 | Mostrar Cutas | Dolar - Contable |
14        | 15 | CP067 | Mostrar Cutas | Dolar - Contable |
15        | 16 | CP068 | Mostrar Cutas | Dolar - Contable |
```



```
1 >> Feature: Mostrar Cutas
2
3 @e2e-mostrarCouta
4 ▶ Run 'Scenario: <CP>_Mostrar Cutas_<DESCRIPCION>' Ctrl+Mayús+F10
5 🐛 Debug 'Scenario: <CP>_Mostrar Cutas_<DESCRIPCION>'
6 🏃 Run 'Scenario: <CP>_Mostrar Cutas_<DESCRIPCION>' with Coverage
7 ⚙️ Modify Run Configuration...
8     Then Realizo el pago simulado en el formulario mc
9     And Realizo la autorización de la transacción mc
10    And Valido la respuesta autorización mc "<N>"
11    Examples:
12        | N | CP | CASO USO | DESCRIPCION |
13        | 14 | CP063 | Mostrar Cutas | Dolar - Contable |
14        | 15 | CP067 | Mostrar Cutas | Dolar - Contable |
```

Para ejecutar desde el archivo RunTest.Class debemos obtener los tags de los feature a ejecutar esto nos puede apoyar a ejecutar todos los feature o realizar una ejecución parcial.

Indicar los tags a ejecutar como se indica en la imagen, después presionar en el icono al lado izquierdo de la palabra **public class** para poder ejecutar los diferentes escenarios.



```
1 package com.niubiz.bot.frontend.runner;
2
3 import ...
4
5
6
7
8 @CucumberOptions(
9     features={"src//main//resources//features"},
10     glue={
11         "com.tsoft.bot.frontend.steps",
12         "com.tsoft.bot.frontend.helpers"
13     },
14     plugin = {
15         "pretty",
16         "html:target/cucumber-reports/cucumber-pretty.html",
17         "json:target/cucumber-reports/CucumberTestReport.json",
18         "rerun:target/cucumber-reports/rerun.txt"
19     },
20     tags = "@feature-indicar and @feature-indicar2",
21     monochrome = true,
22     publish = false,
23     snippets = SnippetType.CAMELCASE
24 )
25
26 @Test
27 public class RunTest extends AbstractTestNGCucumberTests { }
```