Manual de instalación y configuración de entorno (Básico)
Automatización de Pruebas

Nombre: Luis Ortuzar, Nicolás Endress
Fecha: 06/09/2023

Índice

Introducción

En este manual de automatización de pruebas veremos la instalación paso a paso de cada aplicativo, la configuración del entorno de pruebas para poder configurarla de manera correcta. Se nos presentaran imágenes de cada paso que hagamos para que no existan muchas complicaciones al momento de realizar cada paso.

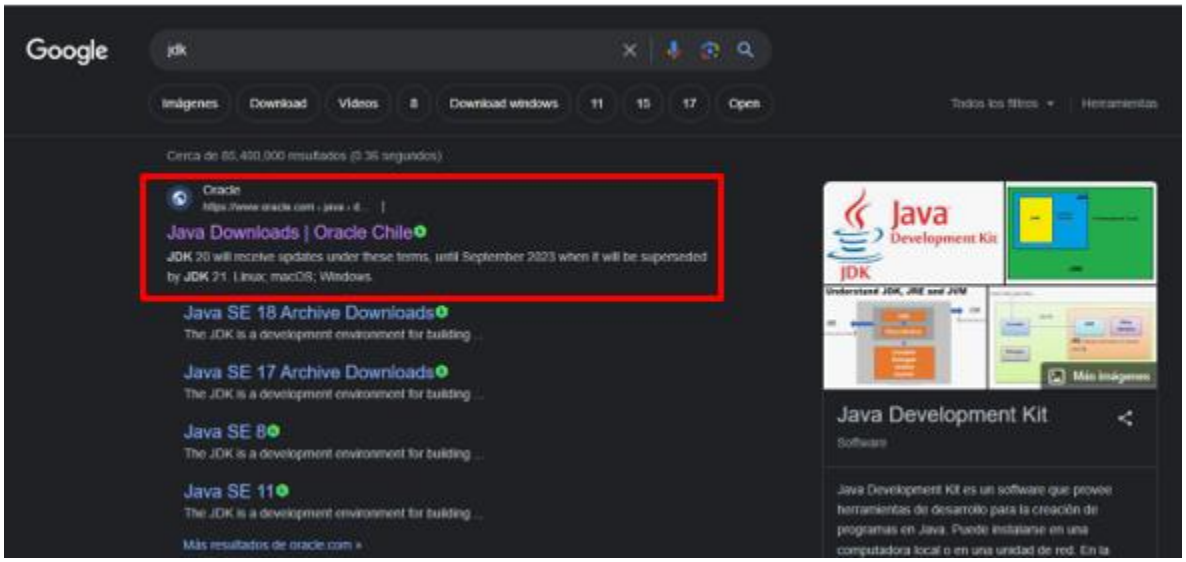
Se dejarán las dependencias necesarias para el funcionamiento correcto del entorno de automatización.

Daremos a conocer los localizadores y para qué sirve cada uno, de donde se sacan y como utilizarlos en nuestras pruebas automatizadas.

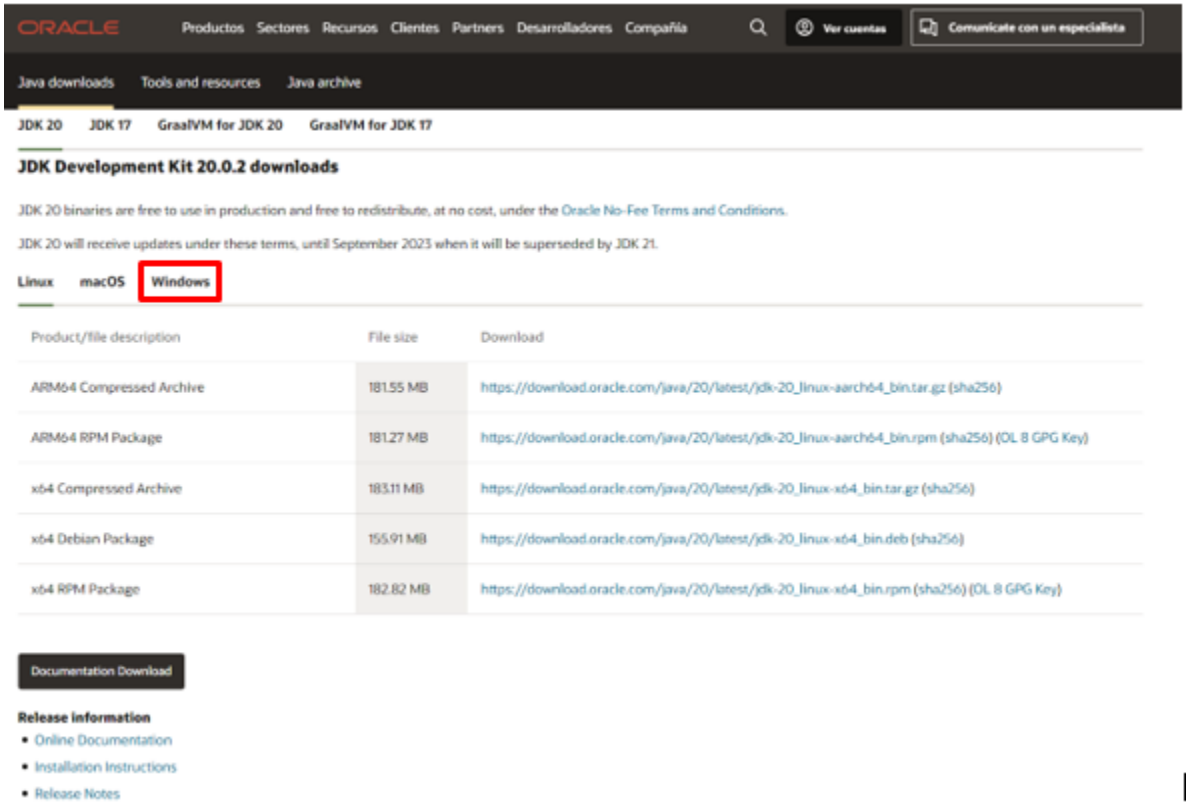
Instalación de JDK

Procedemos a buscar y descargar la jdk o Java Development Kit (Kit de Desarrollo de Java en español).

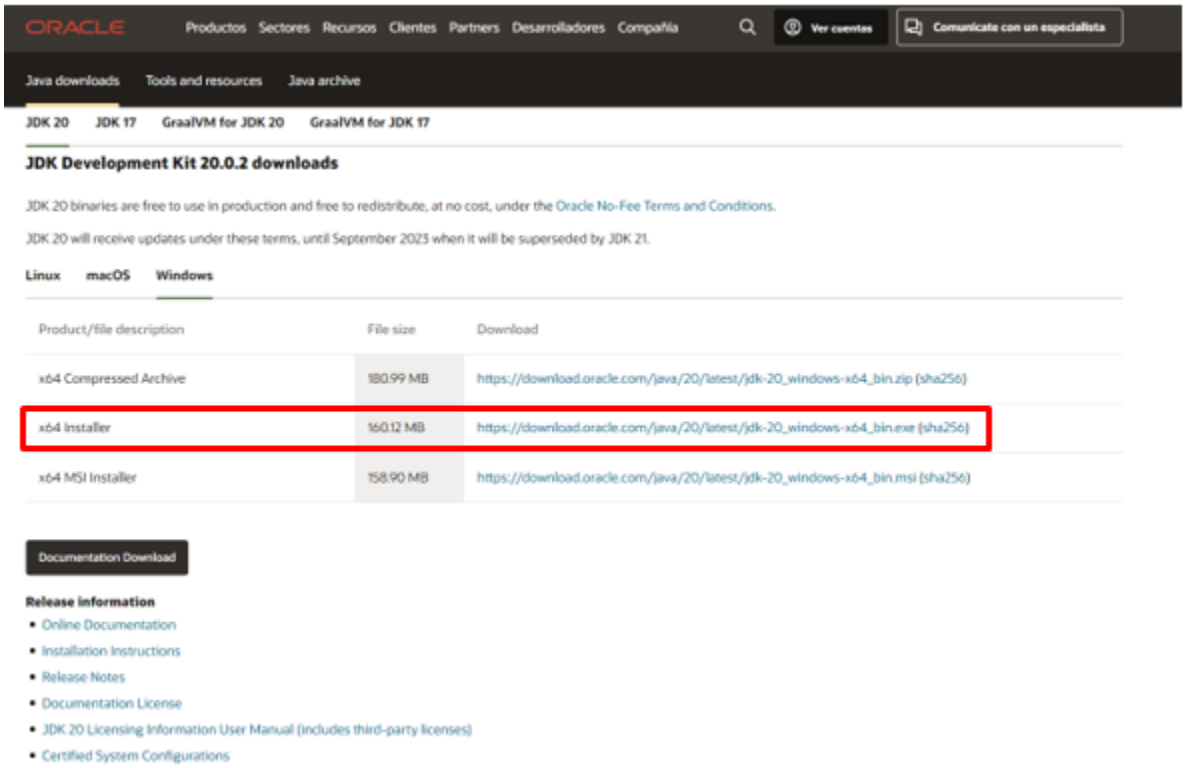
Presionamos el primer link que nos aparece.



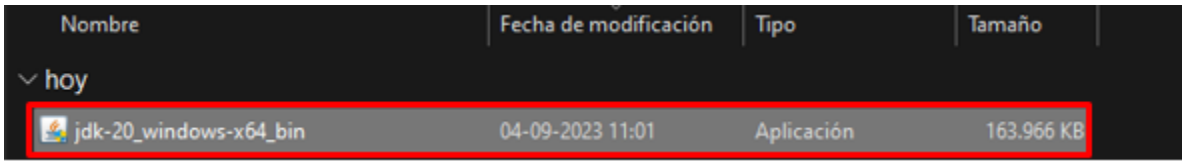
Luego buscamos la plataforma en la que trabajaremos en este caso “Windows”



Elegimos el instalador que nosotros queramos. aquí elegimos el instalador directo.



Hacemos doble clic en el instalador.

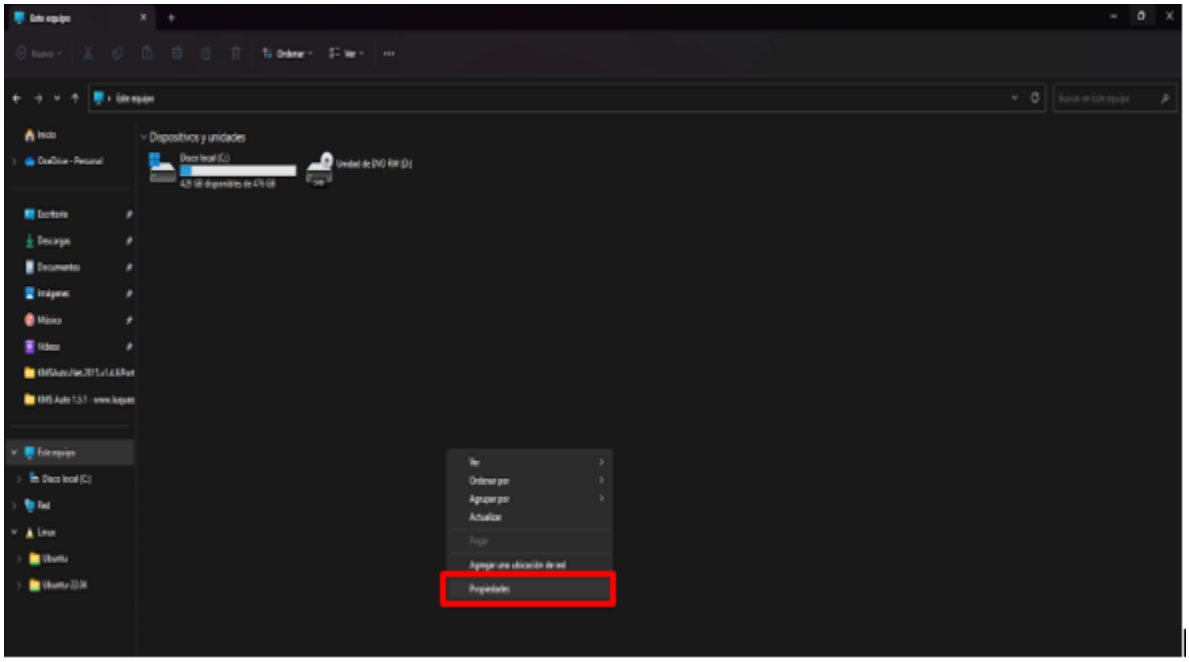


Nos aparecerá la siguiente ventana y presionamos el botón “Siguiente”.

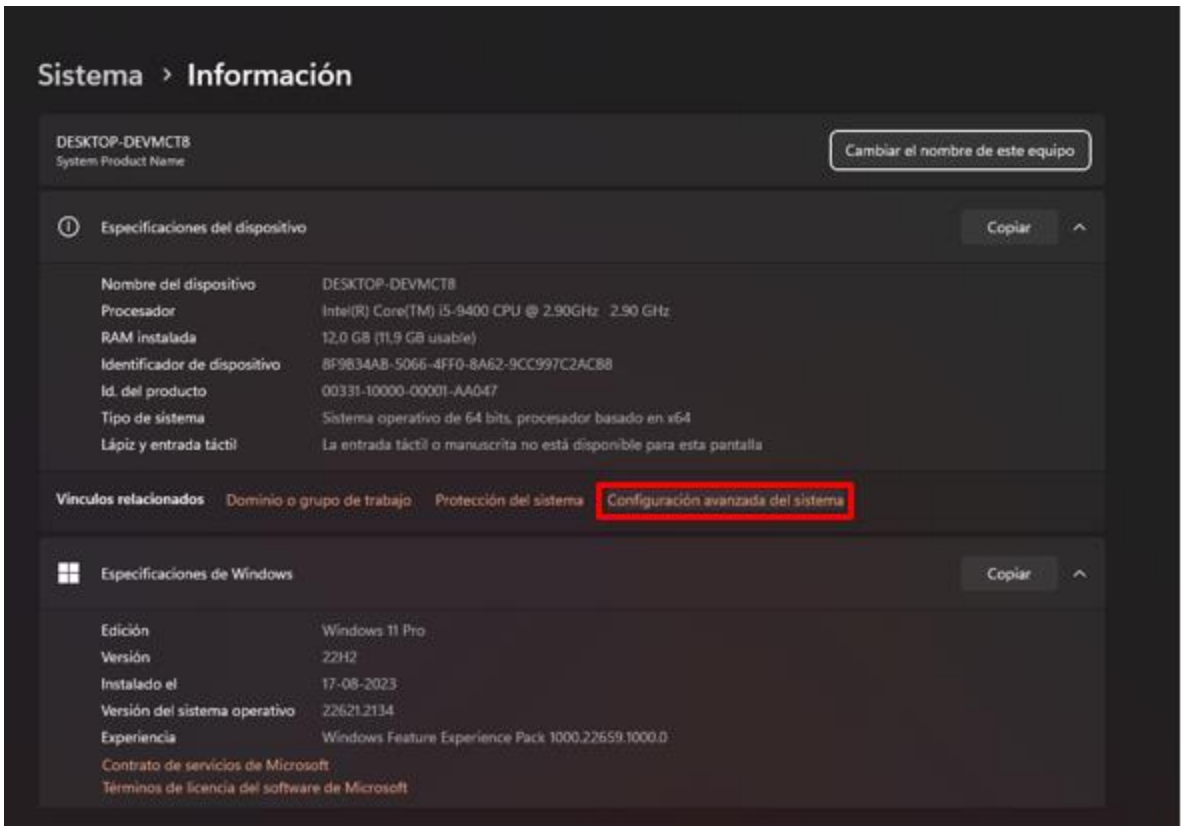
Luego nos dará a conocer la ruta de instalación podemos cambiarla o dejarla por defecto. Presionamos el botón “Siguiente”.

Luego nos aparecerá la siguiente ventana con el mensaje de que la instalación fue satisfactoria. Presionamos el botón “Cerrar”.

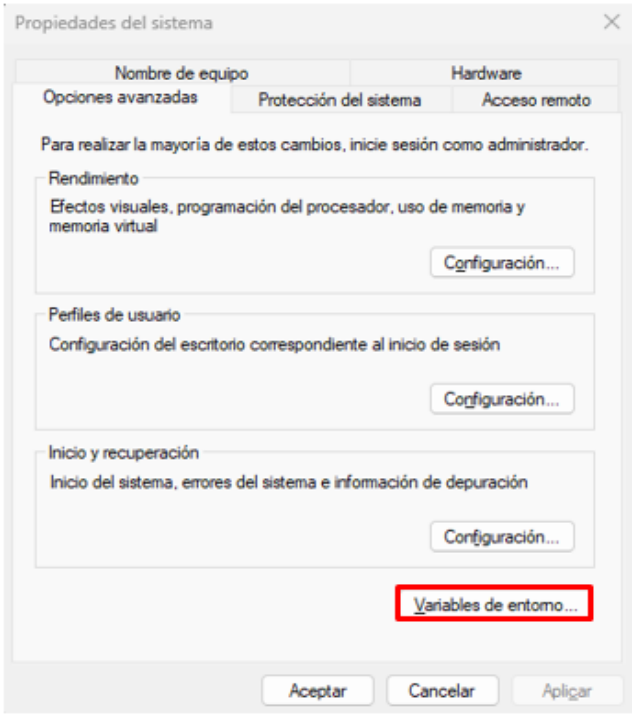
Luego en equipo hacemos clic derecho en “Propiedades”.



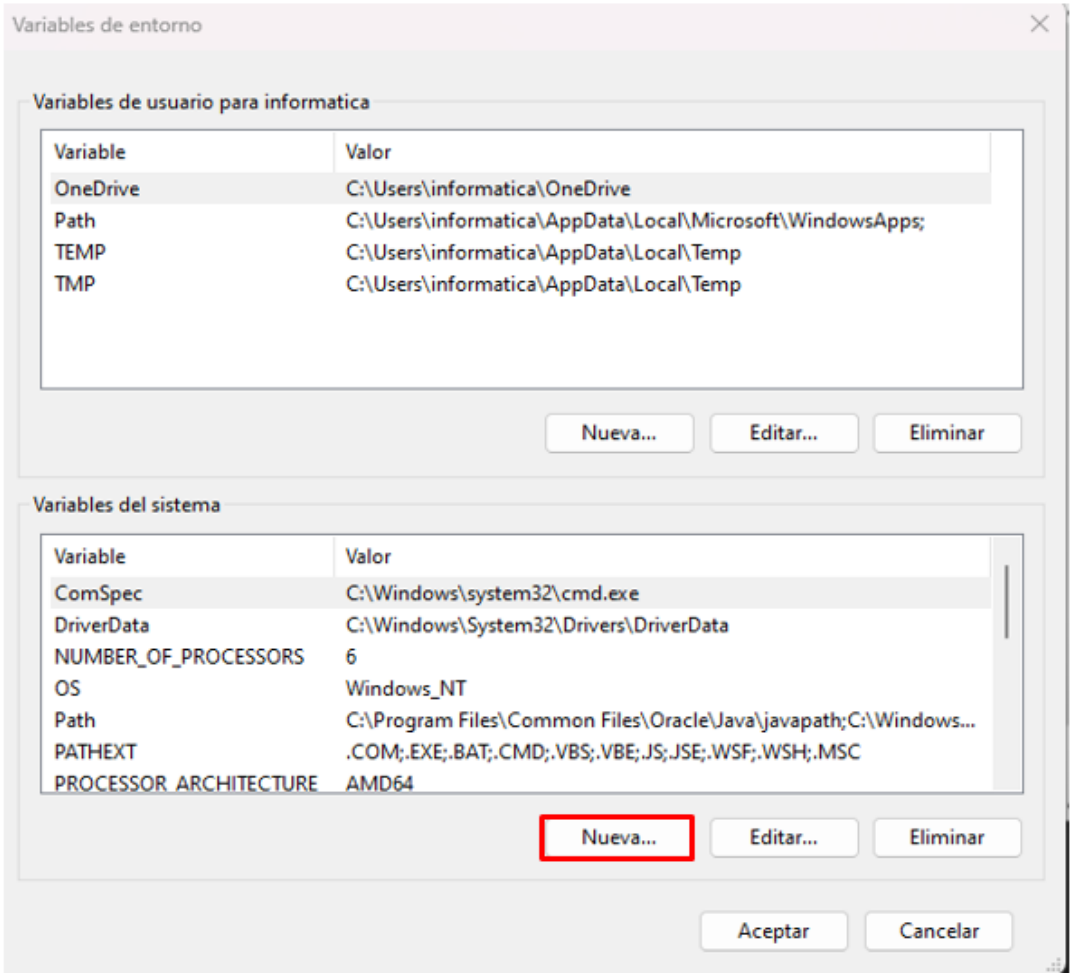
Nos adentramos en “configuración avanzada del sistema”.



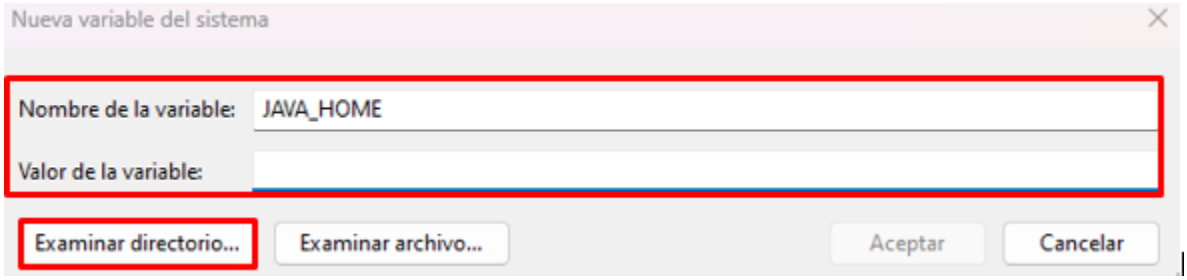
Luego presionamos el botón “Variable de entorno”.



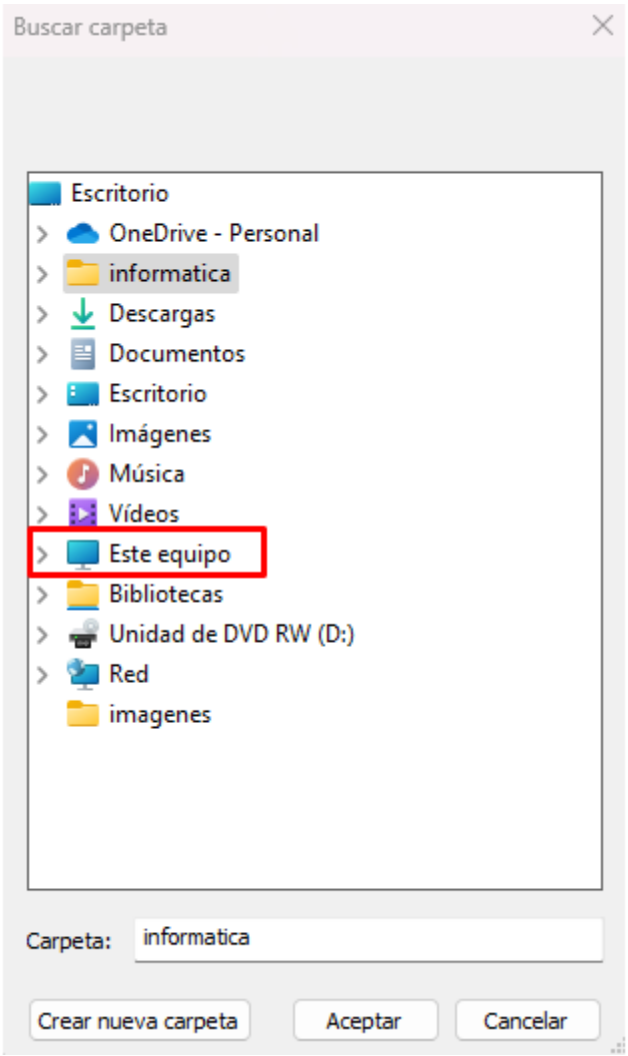
Deberemos crear una nueva variable del sistema



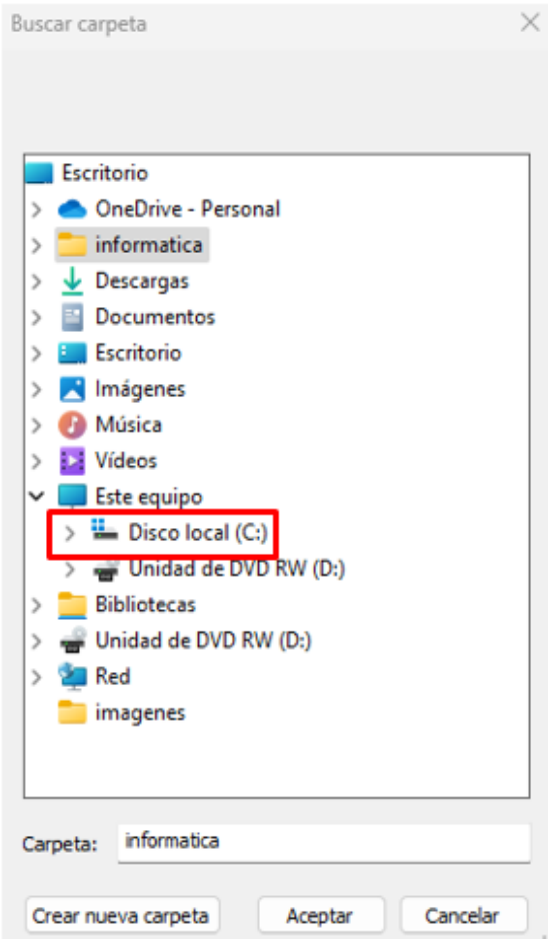
La variable de ambiente se deberá llamar “JAVA_HOME” y el valor es la ruta donde instalamos la “JDK”. al presionar el botón “Examinar directorio”.



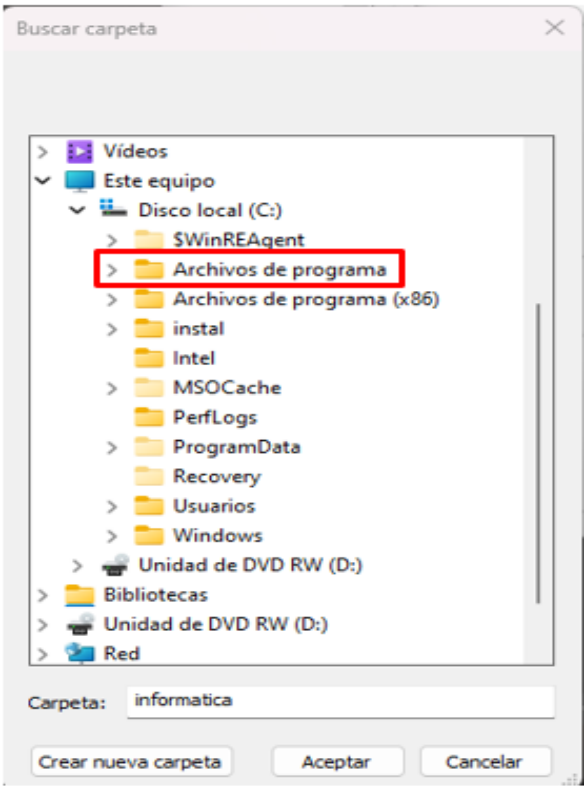
Nos adentraremos en “Este equipo”.



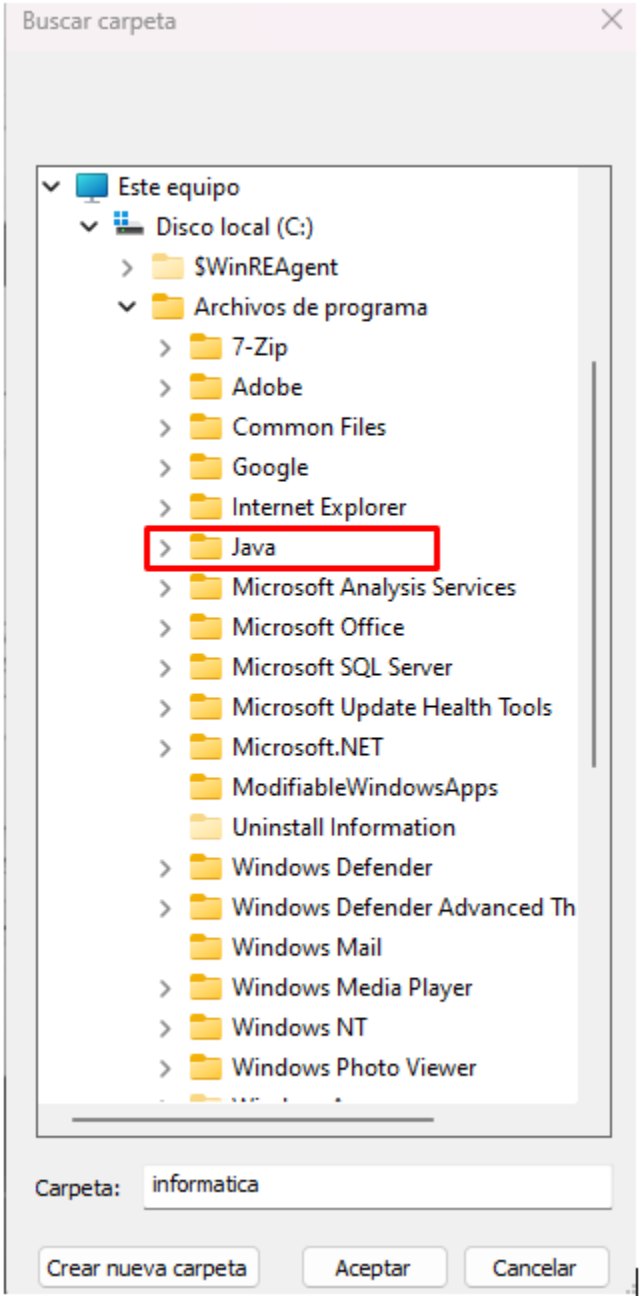
Luego en “Disco local (C:)”.



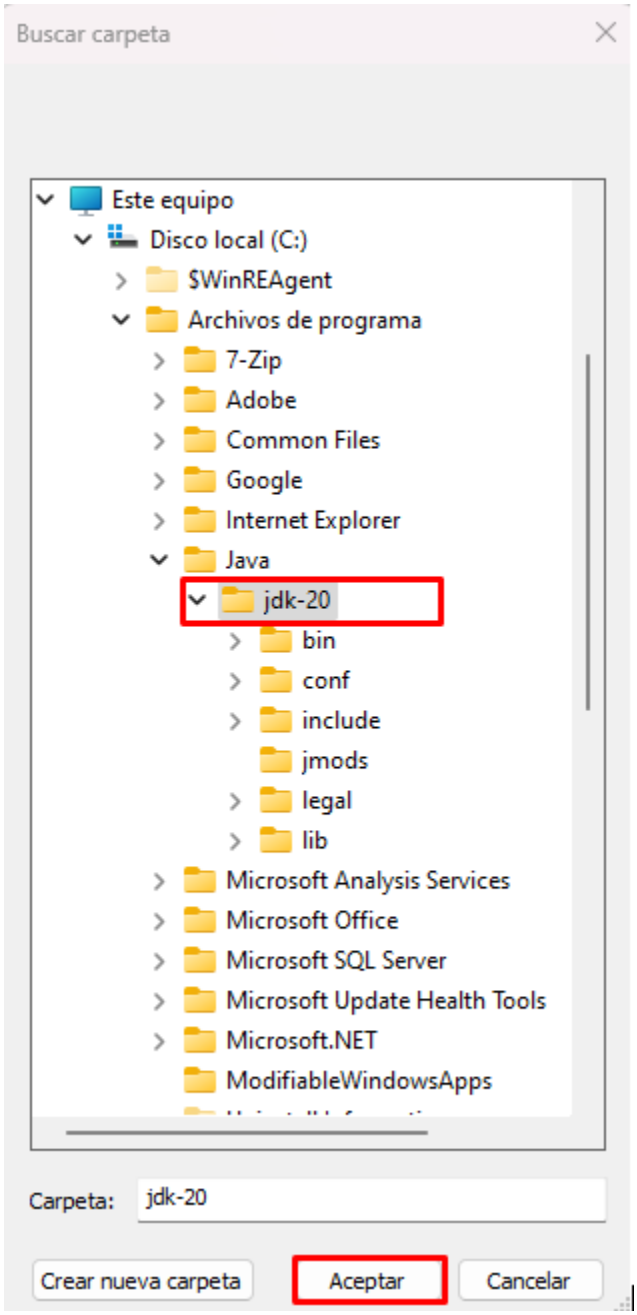
Presionamos el directorio “Archivo de programa”.



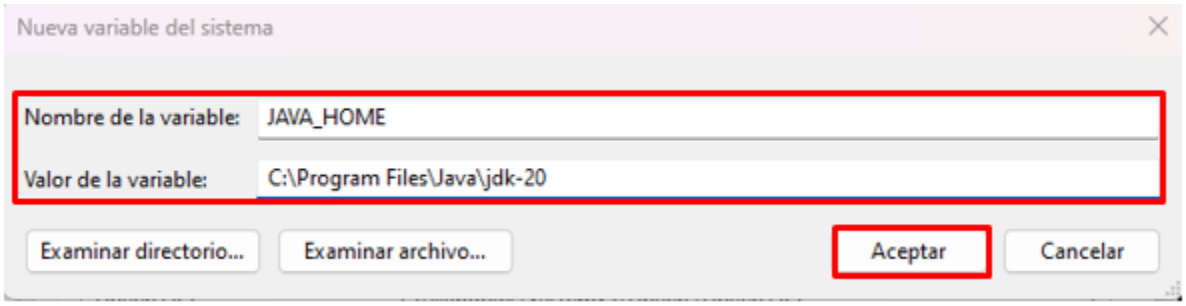
Luego en el directorio “Java”.



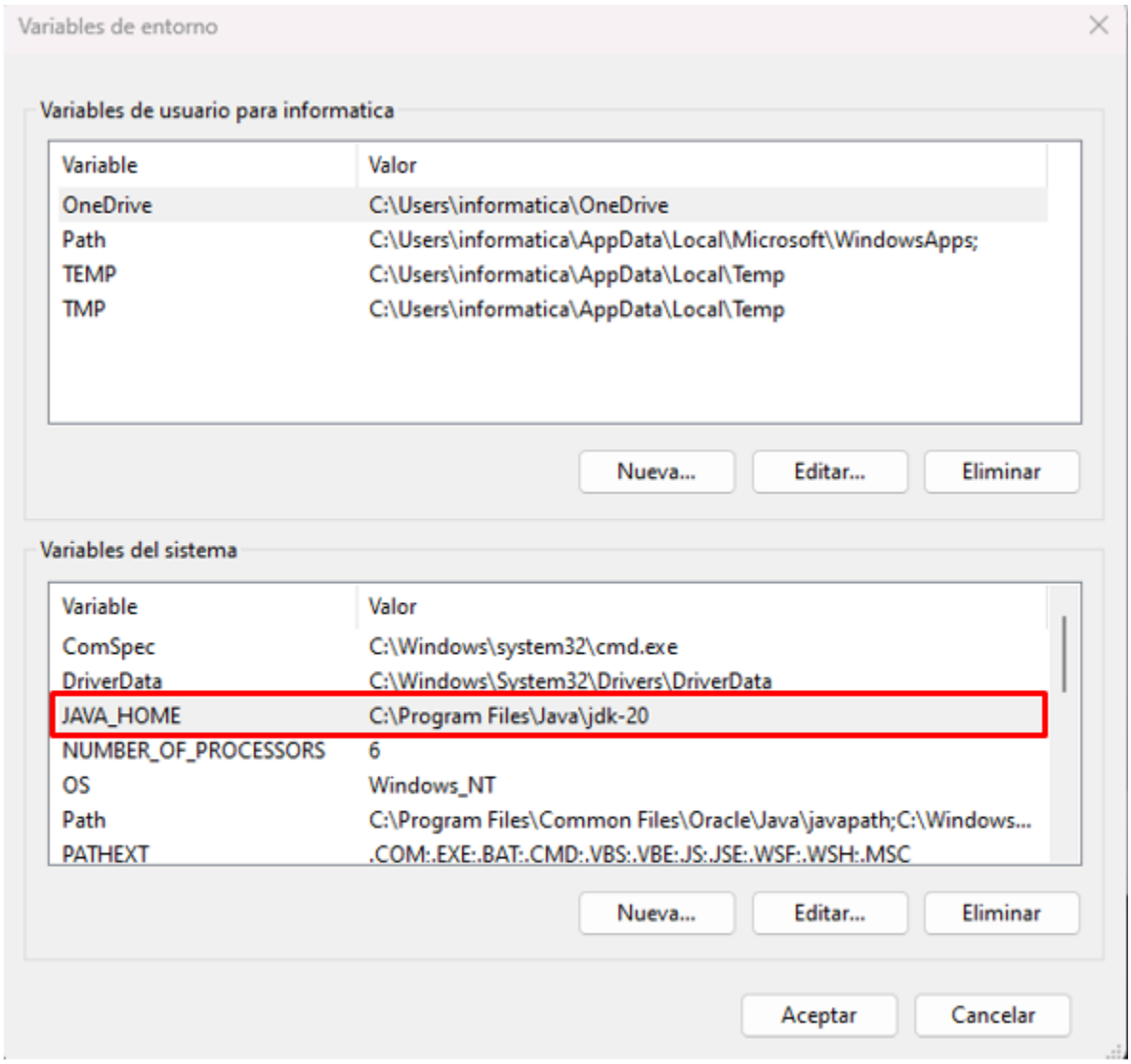
Y por último en el directorio “jdk-20” esto dependiendo de la versión y presionamos el botón “Aceptar”.



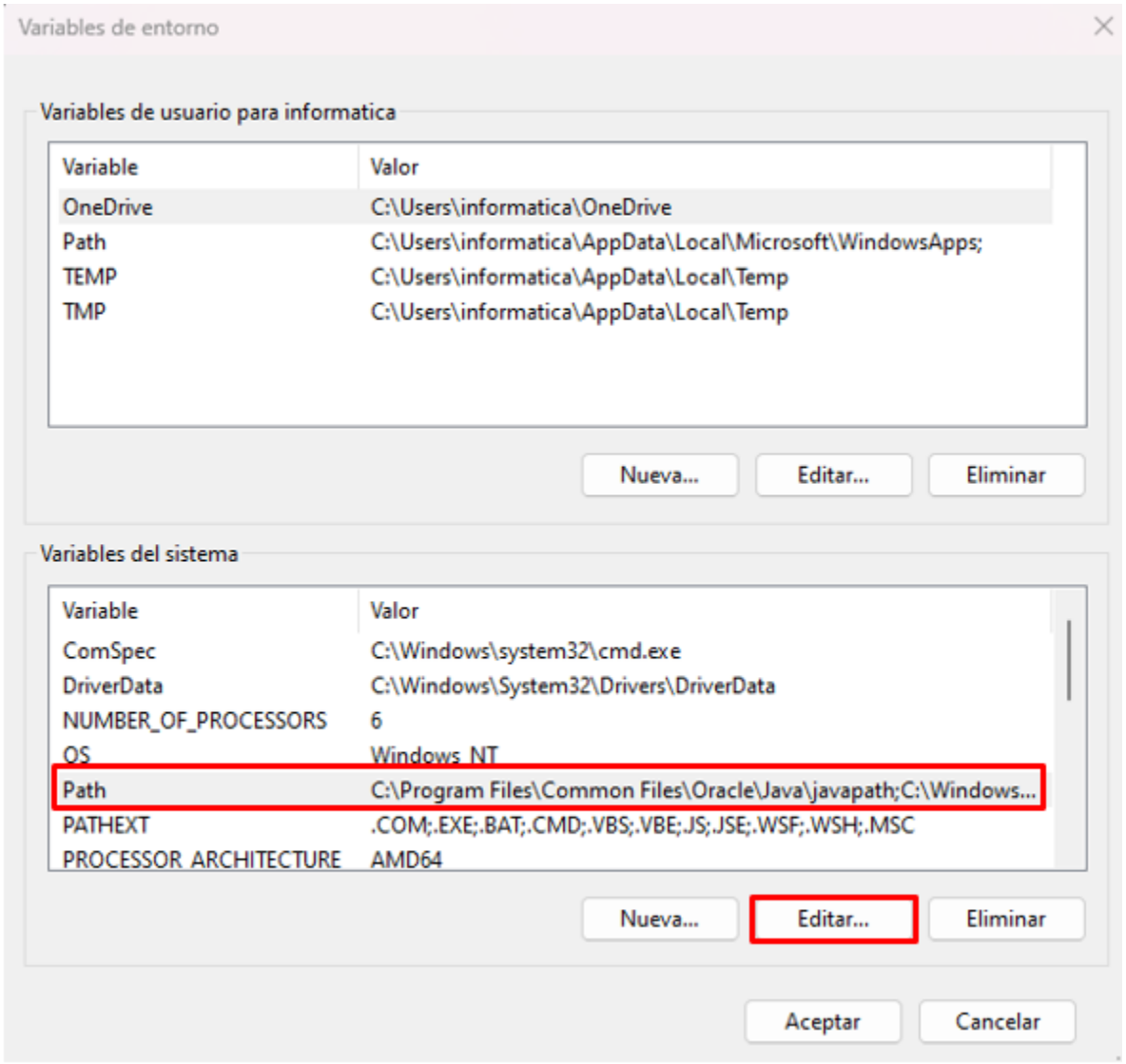
Una vez ingresado el “Valor de la variable” presionamos el botón “Aceptar”



Ahí se encontrará creada la “Variable del sistema”.

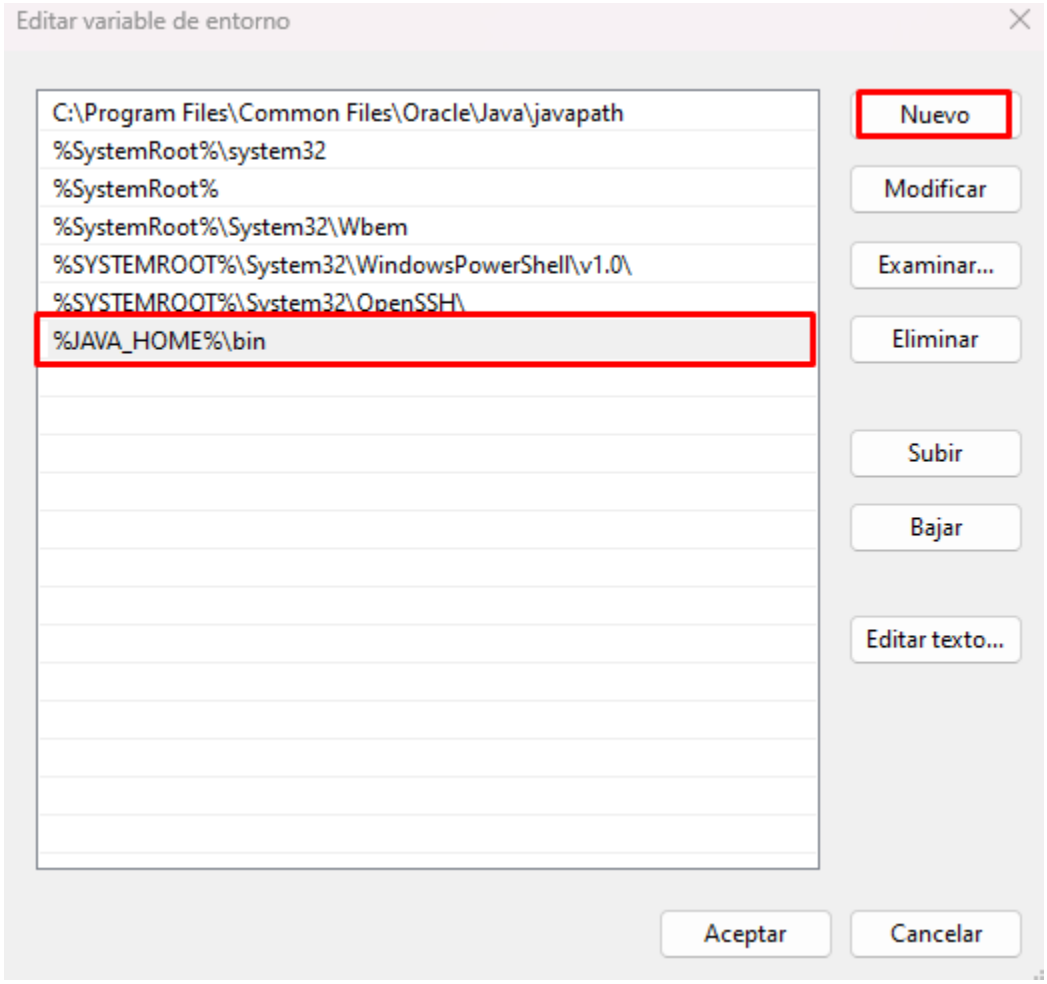


Luego procederemos a editar el “Path” en variables del sistema.



Creamos una nueva variable en la ruta o “Path” presionando el botón “Nuevo”

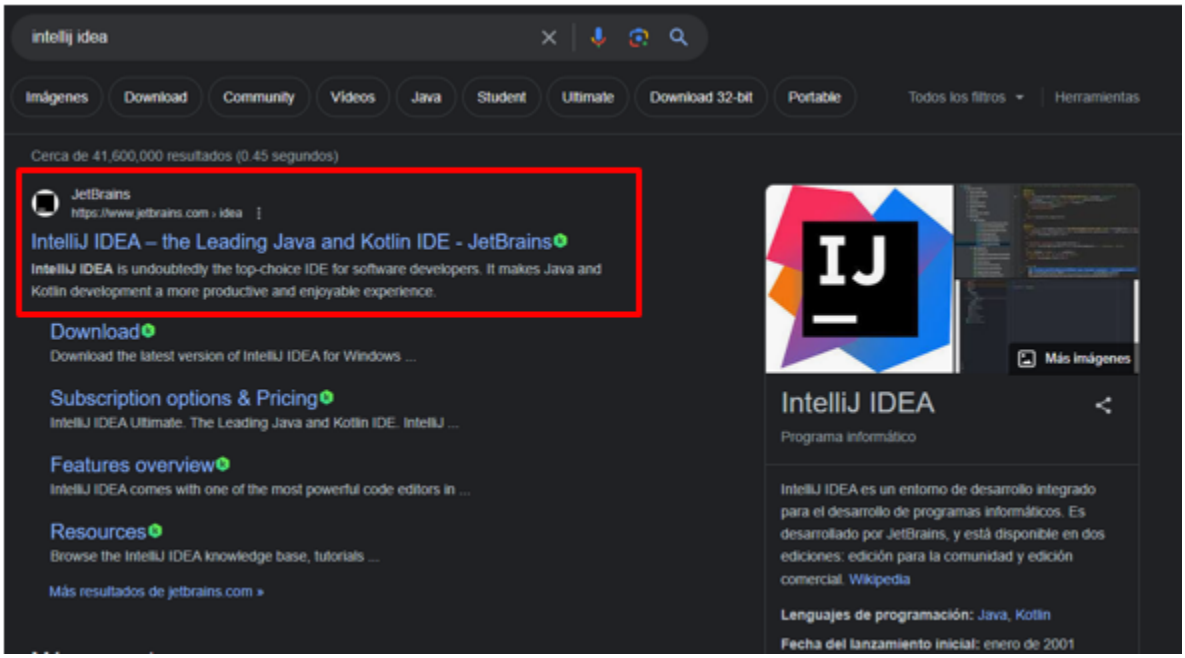
ahí escribiremos lo que se encuentra en el recuadro rojo.



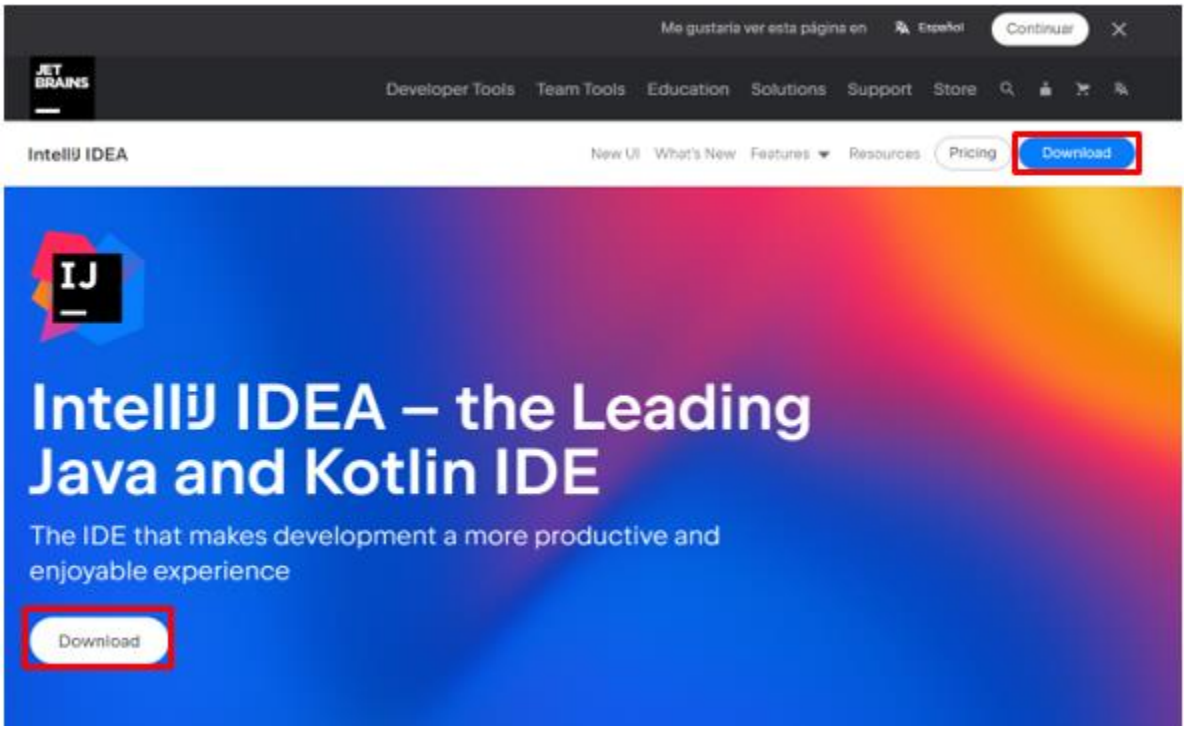
instalador de intellij idea

Procedemos a escribir en el buscador de nuestra preferencia “intellij idea”.

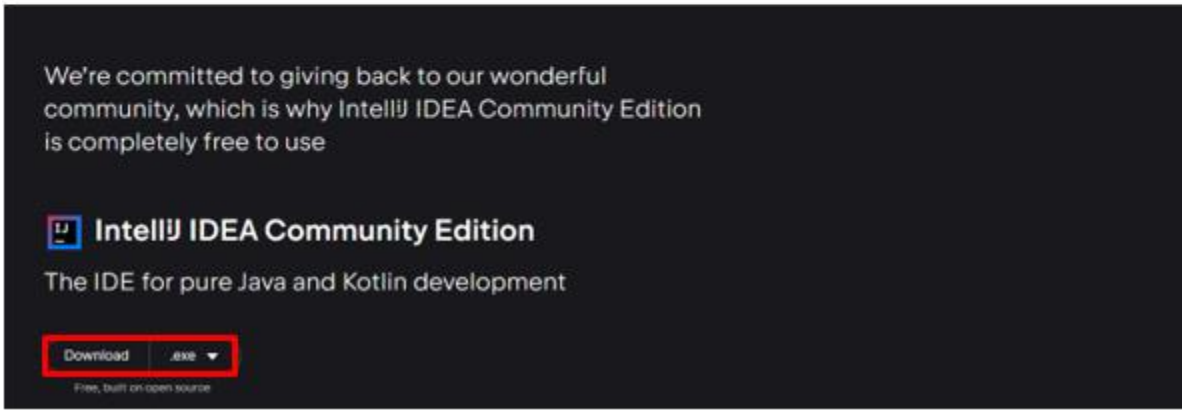
Clickeamos en el primer link que nos aparece.



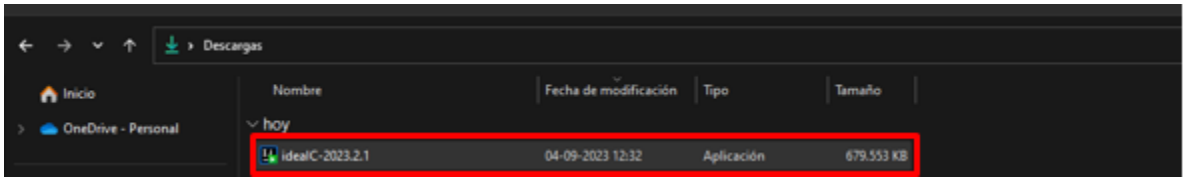
Presionamos el botón “Download” que se encuentra en la parte superior derecha o la parte inferior izquierda.



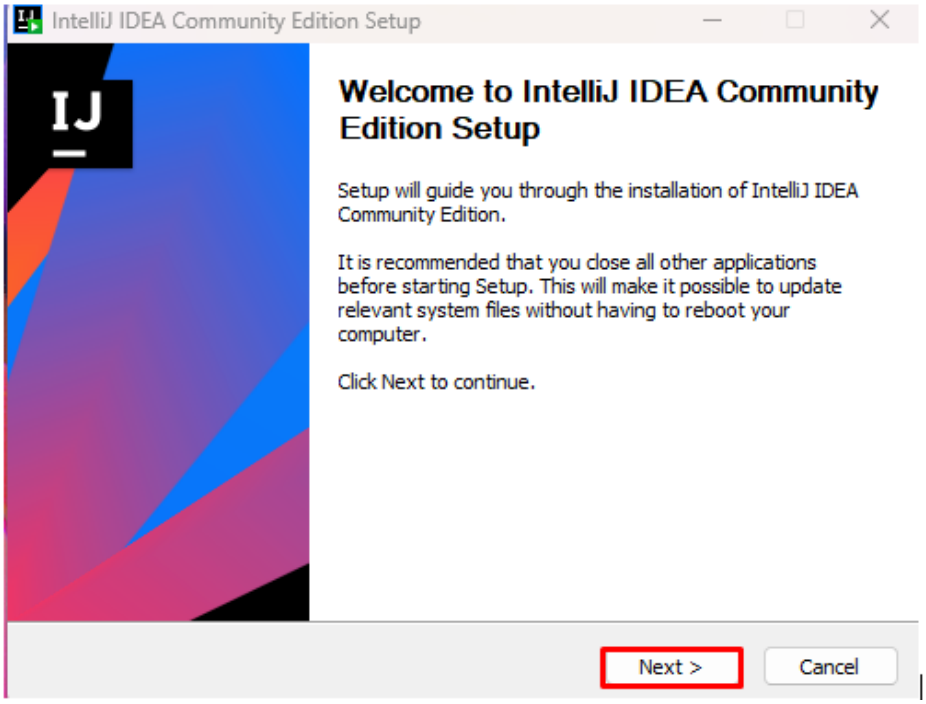
Se nos abrirá la siguiente ventana bajaremos un poco y descargamos la versión “Community” si tenemos la versión de paga “Ultimate” descargamos esa. la que sea de nuestra preferencia



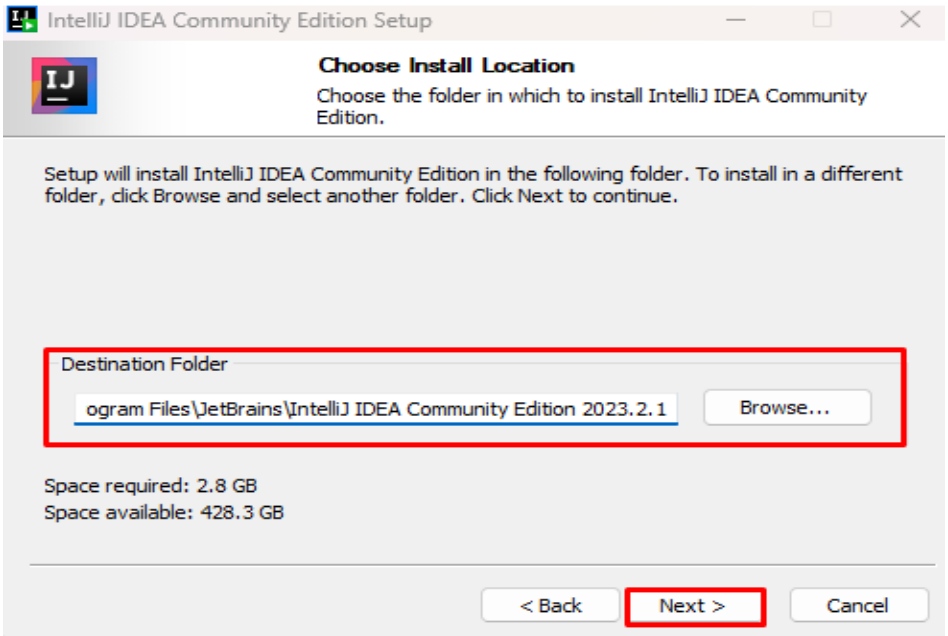
Una vez descargado presionamos doble clic en el instalador.



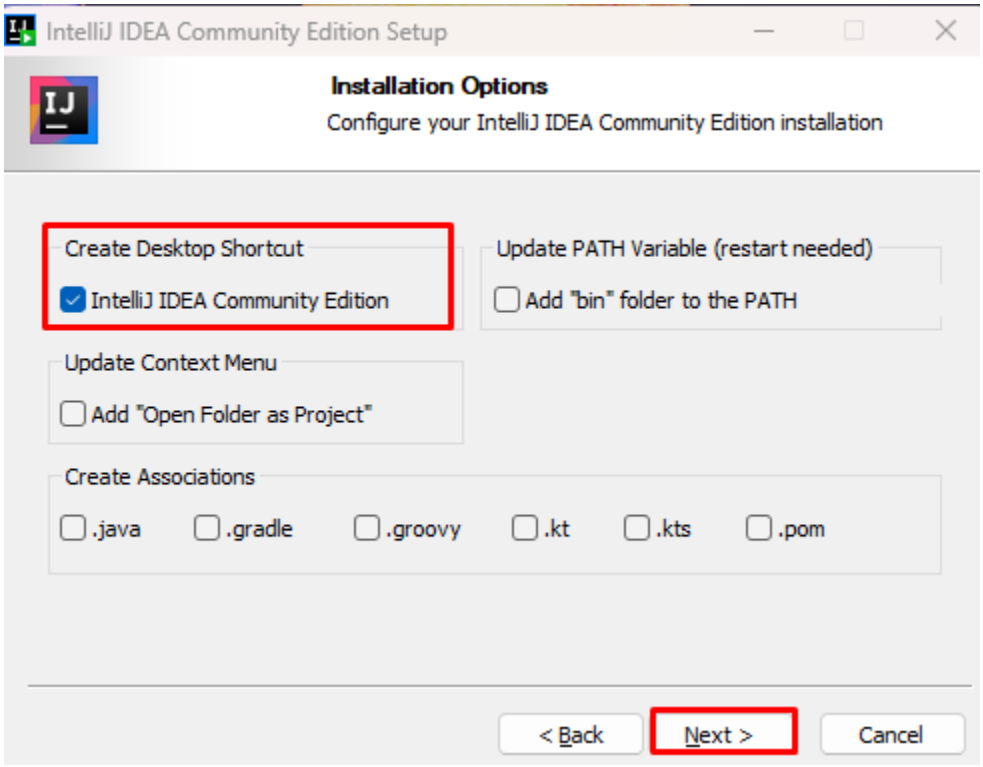
Nos aparecerá la siguiente ventana. presionamos “Siguiente”.



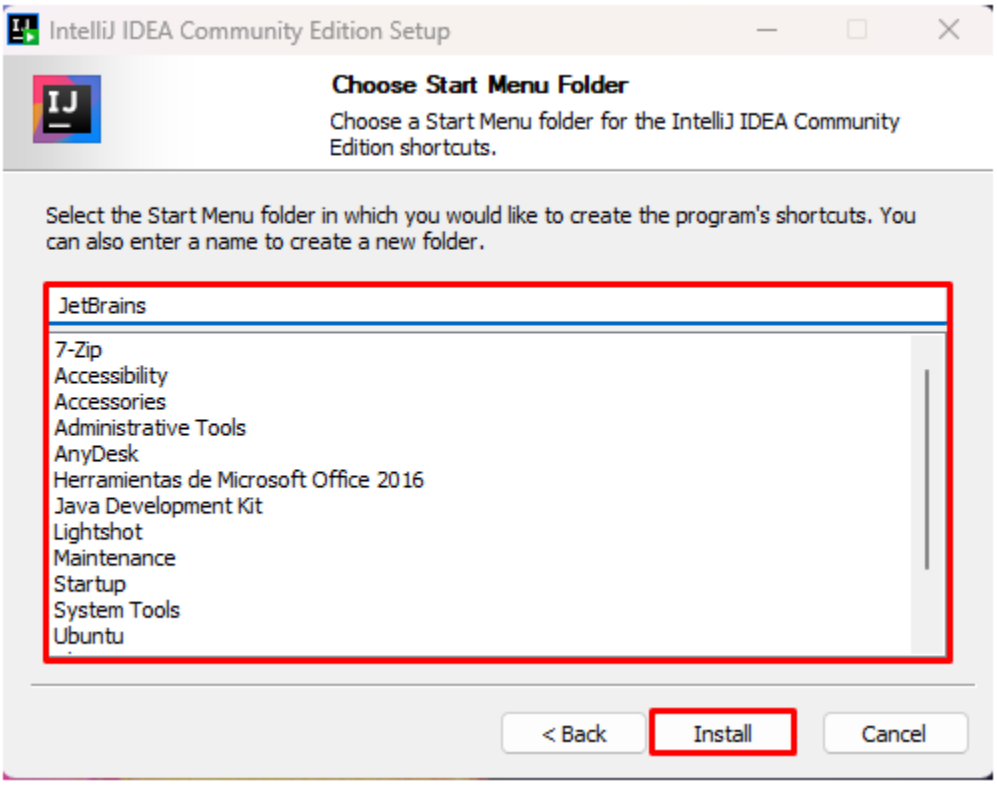
Luego nos aparecerá la ruta donde se instalará la aplicación. si nos parece bien presionamos “Siguiente”.



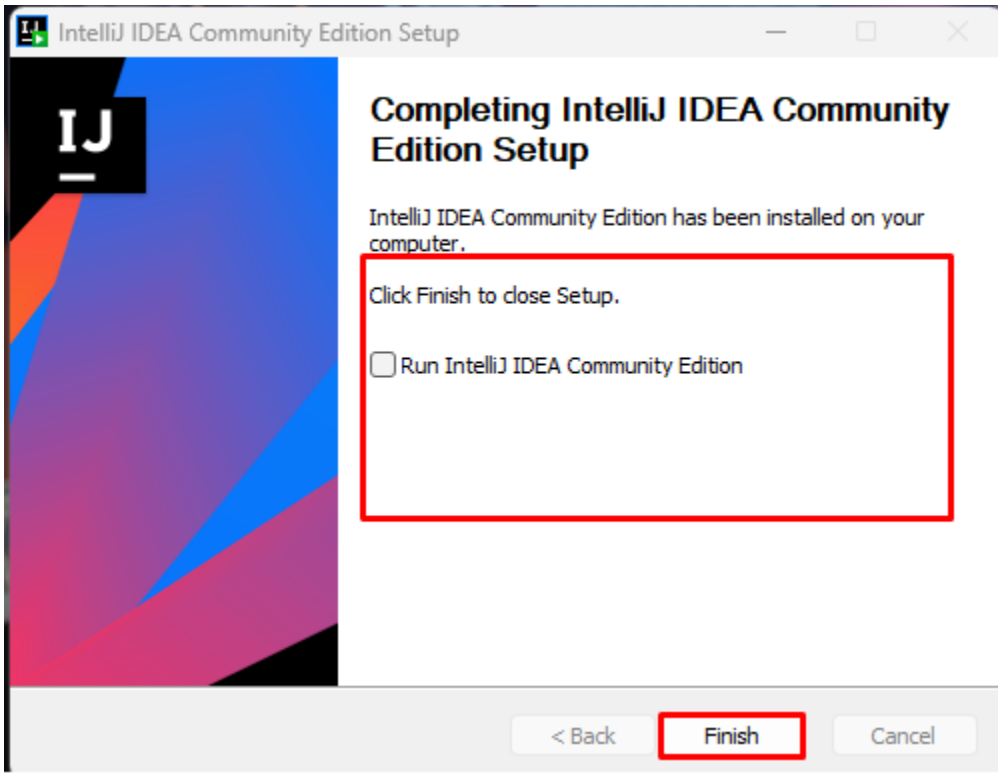
En la siguiente ventana nos aparecerán diferentes opciones. si deseamos las marcamos todas o lo dejamos tal cual. Luego presionaremos el botón “Siguiente”.



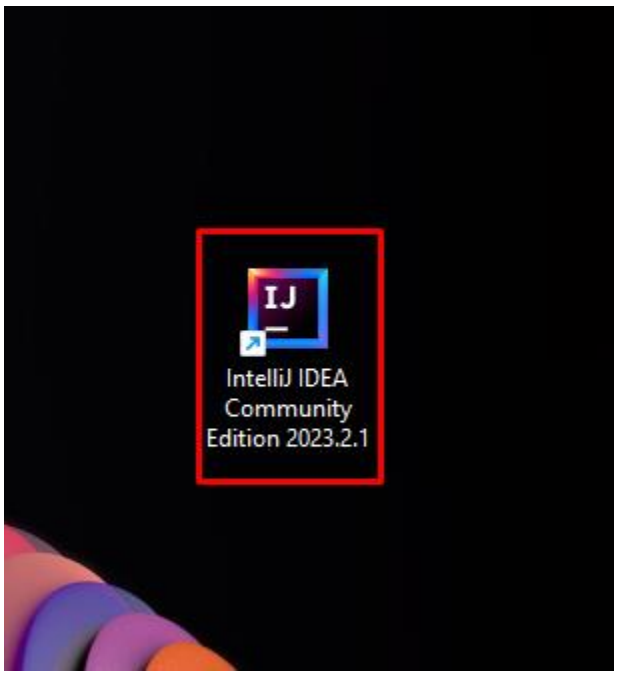
Para finalizar presionamos el botón “Instalar”.



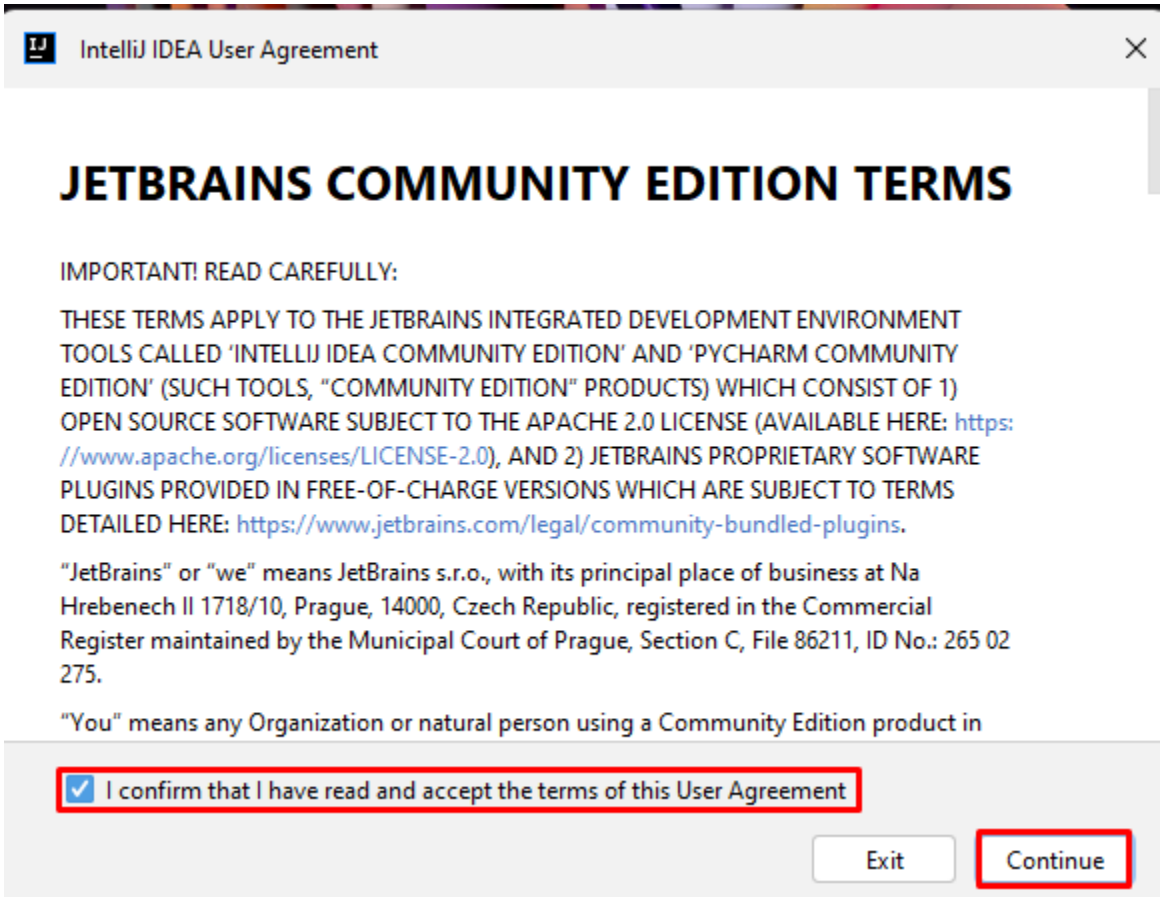
Al terminar nos aparecerá la siguiente ventana donde si queremos iniciar el programa o bien después. A gusto del consumidor.



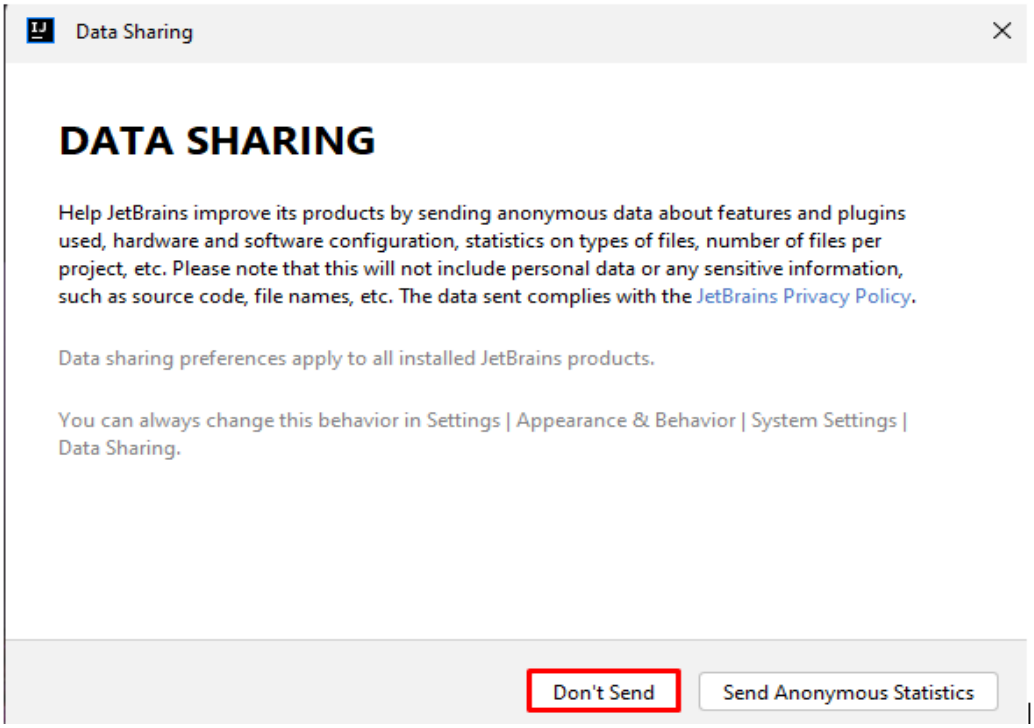
Una vez terminada la instalación nos aparecerá el programa en el escritorio o inicio.



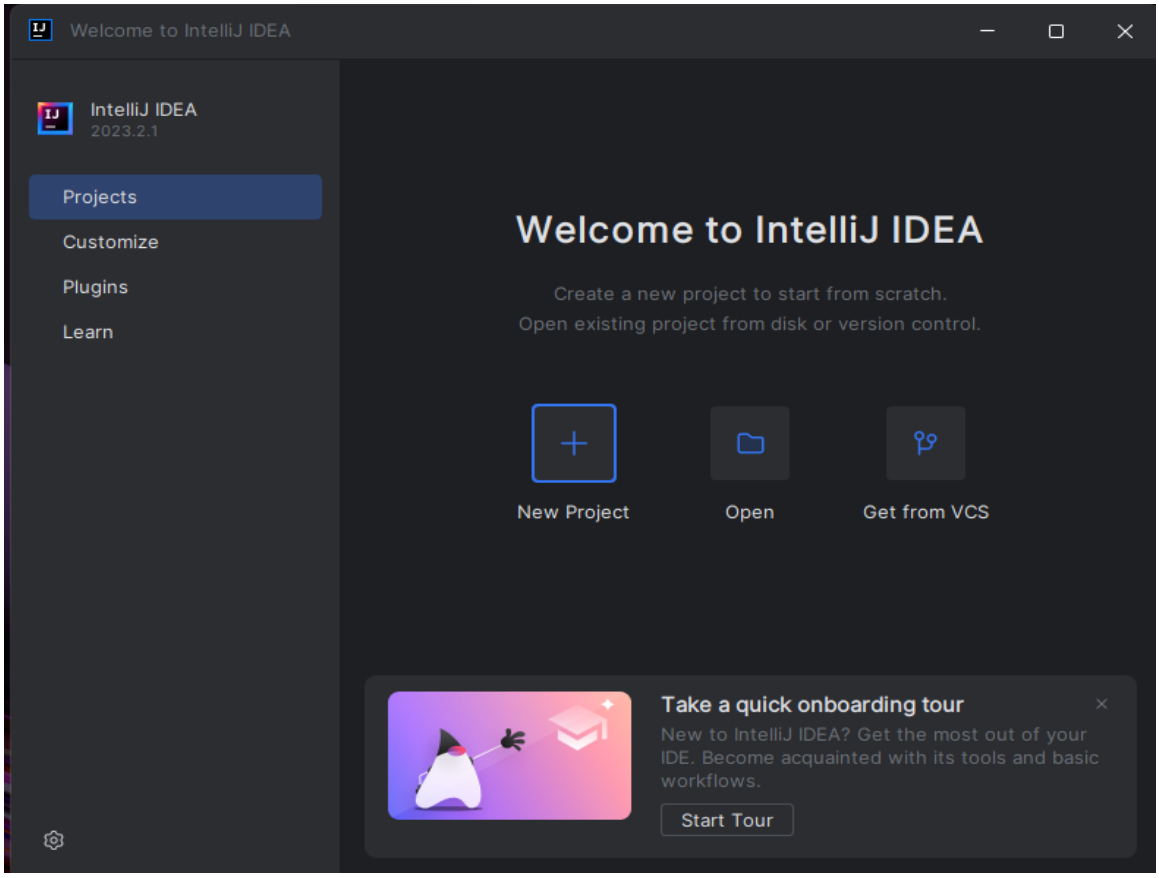
Al hacer clic se abrirá una ventana con los términos del programa. para utilizarlos tendremos que confirmar los términos y luego presionar el botón “Continuar”.



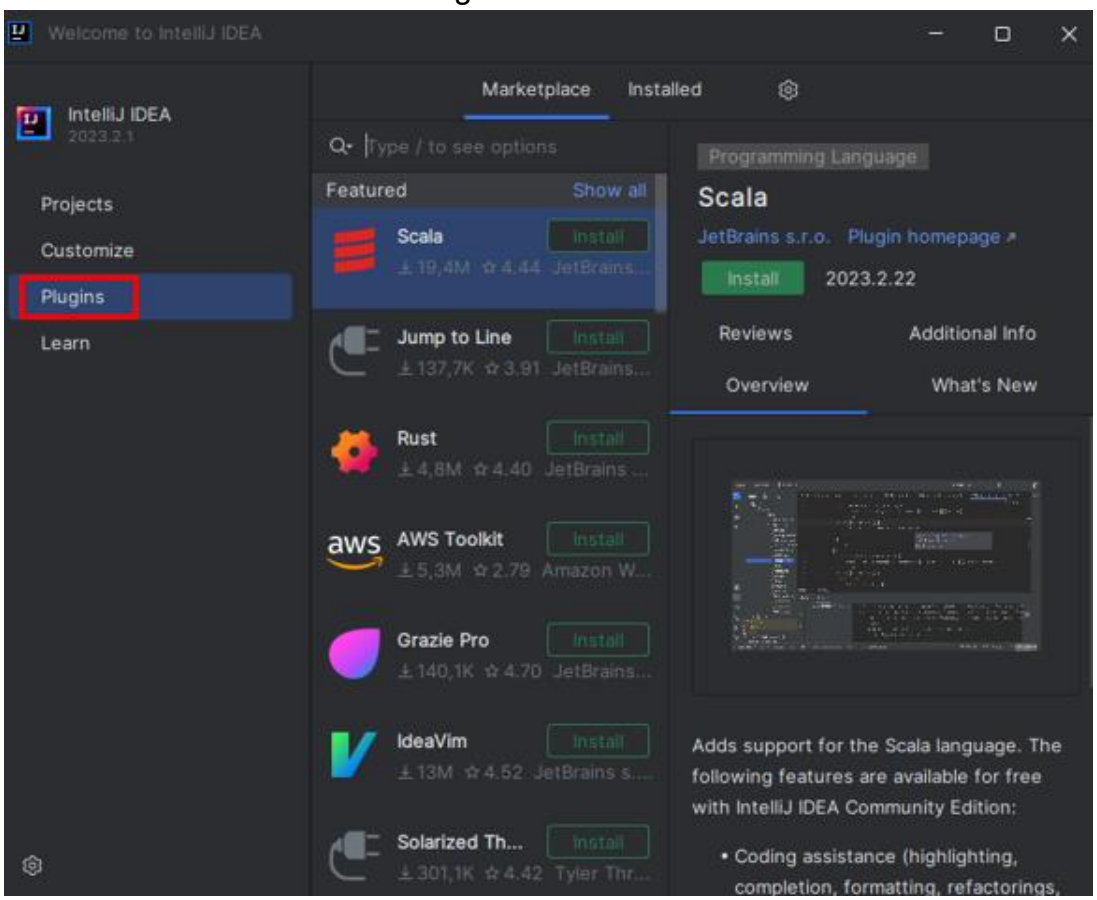
La siguiente ventana nos consultará si deseamos compartir los datos de uso del programa de forma anónima. ahí a gusto del consumidor. en este caso pondremos que no.



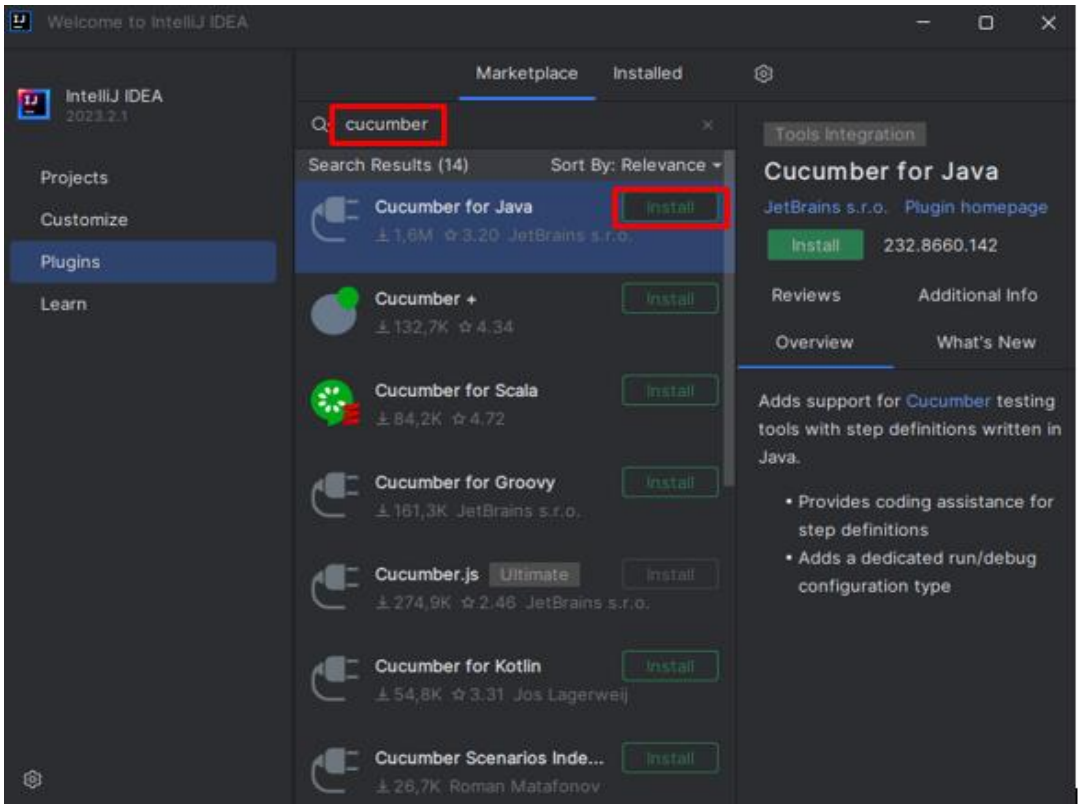
Nos aparecerá la siguiente ventana donde podremos empezar a configurar nuestro proyecto.



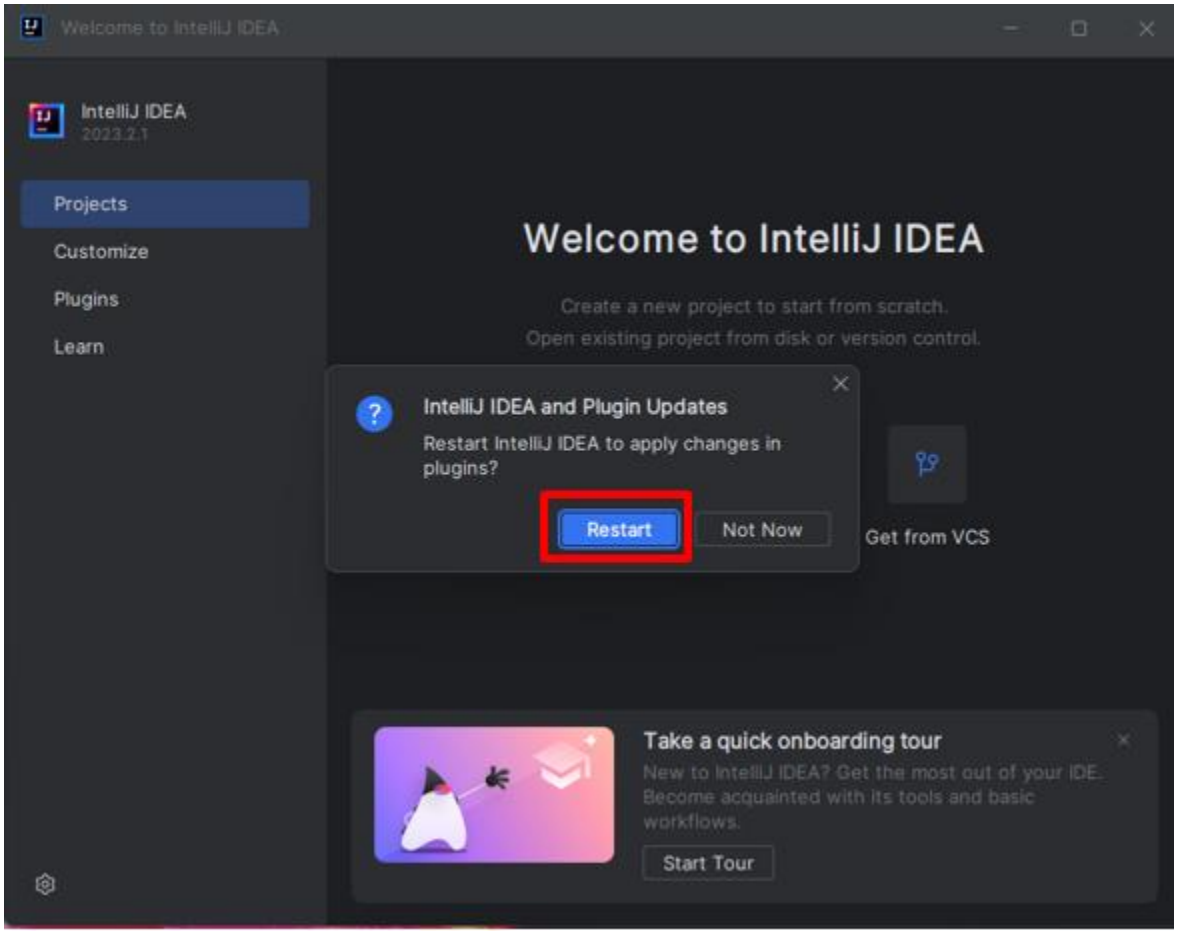
Seleccionamos el botón de “Plugins”.



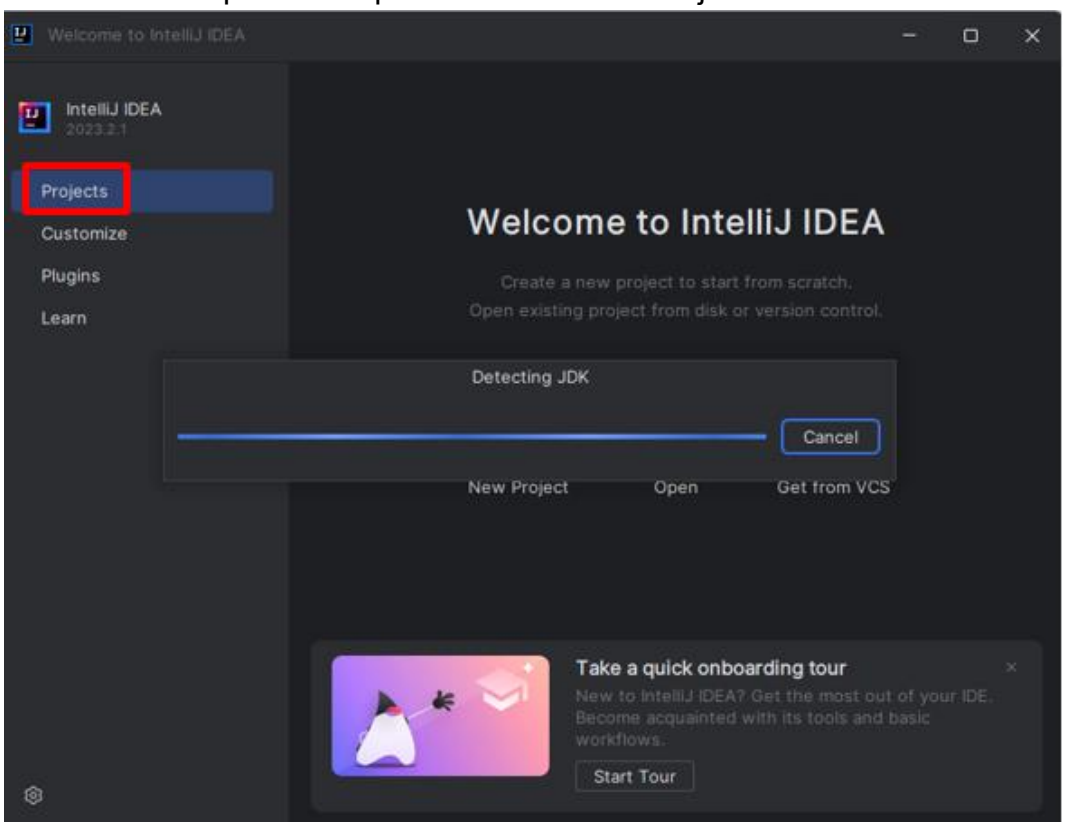
Se procede a escribir cucumber e instalamos “Cucumber for Java”.



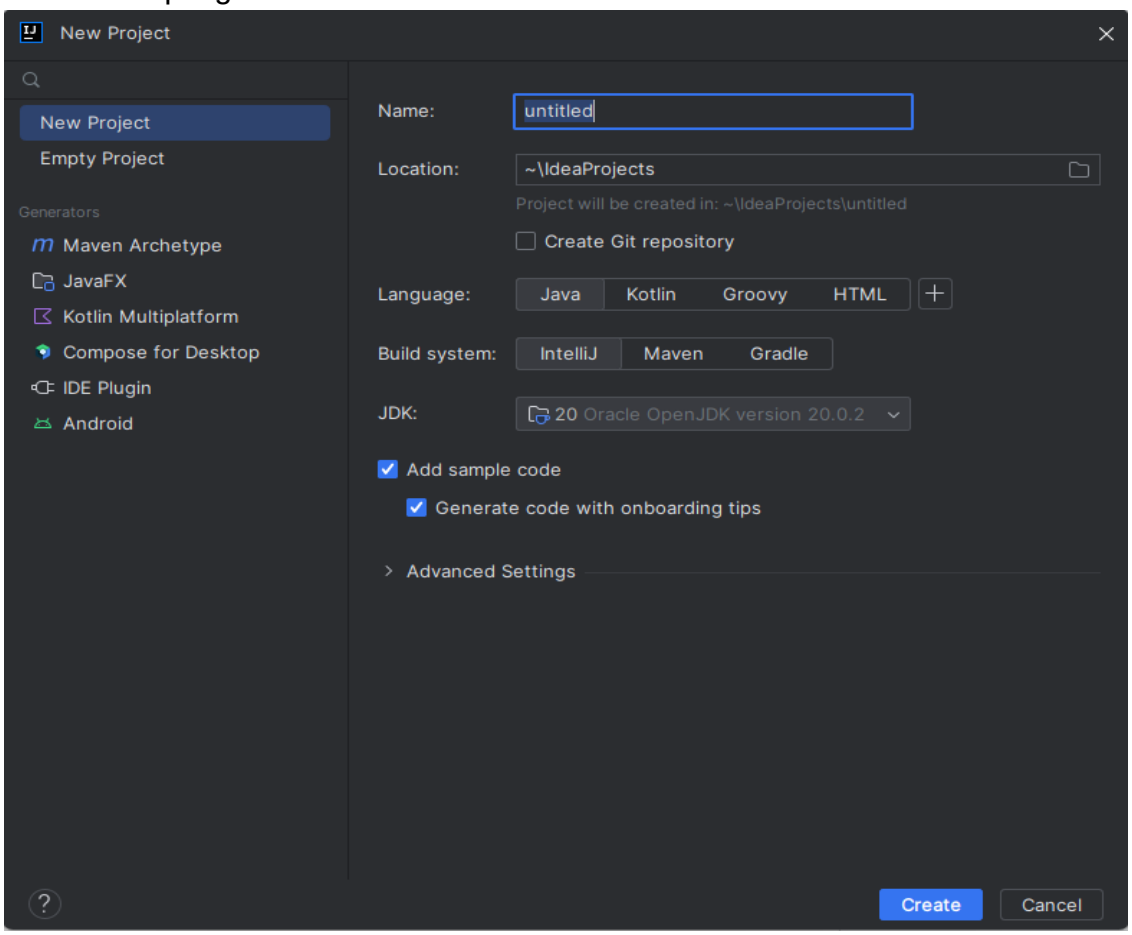
Seleccionamos el botón “Restart” para que se apliquen los cambios.



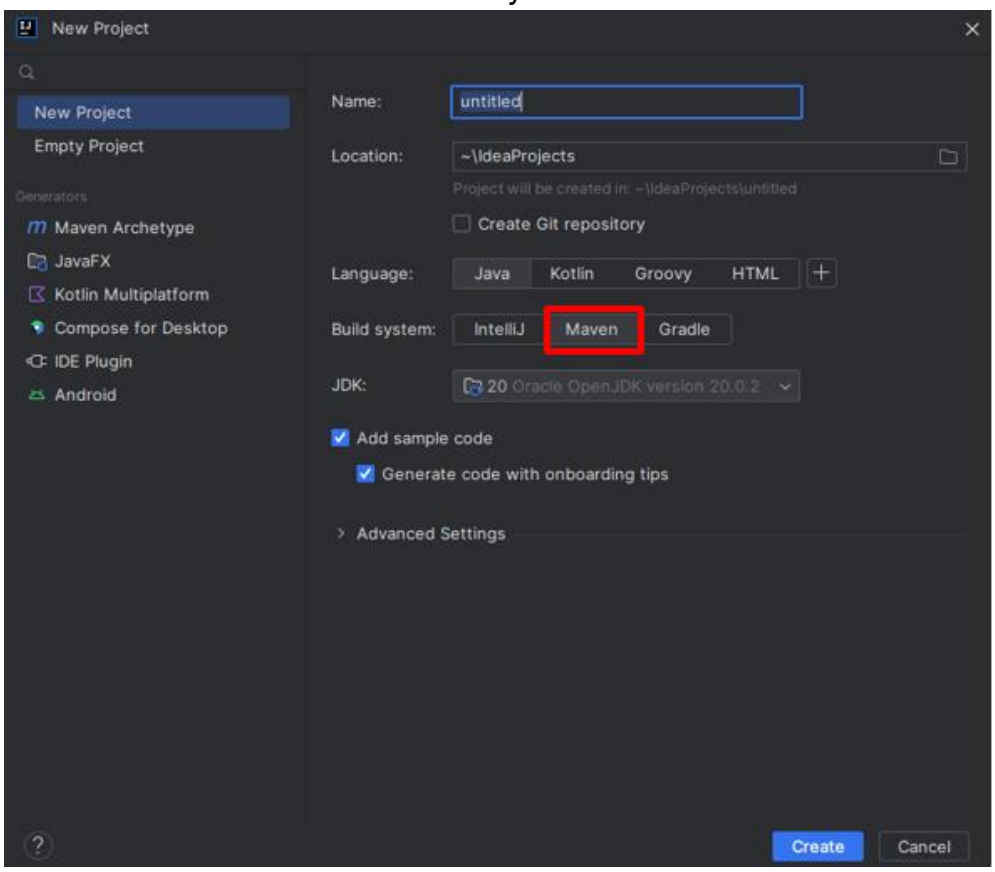
Al reiniciar la aplicación apretamos el botón “Projects”.



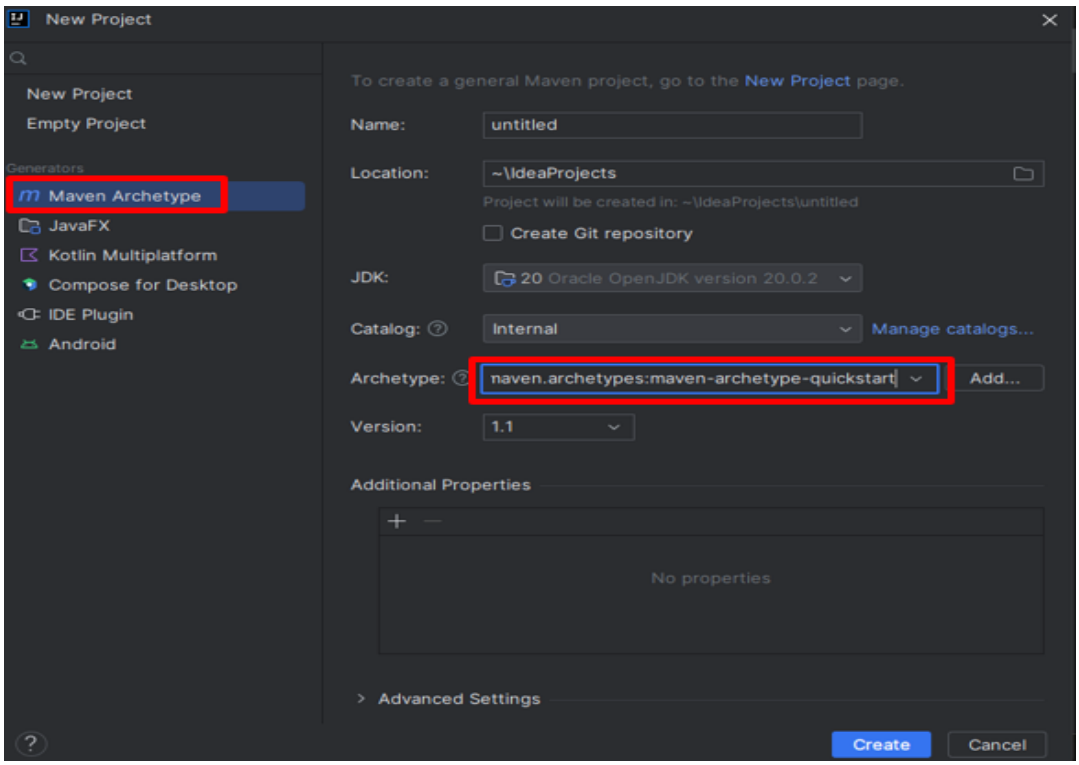
Se nos despliega esta ventana



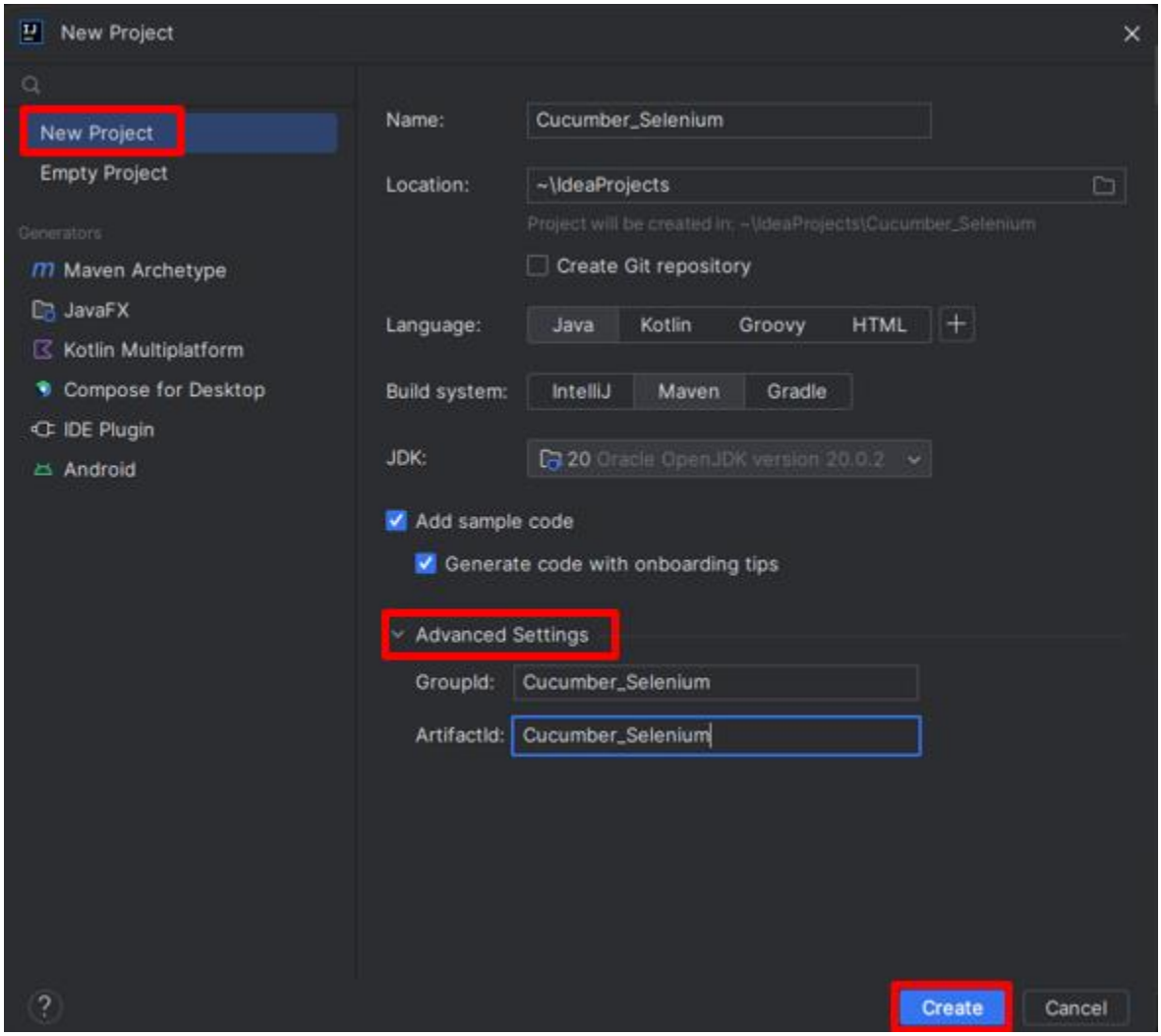
Seleccionamos “Maven” en “Build system”.

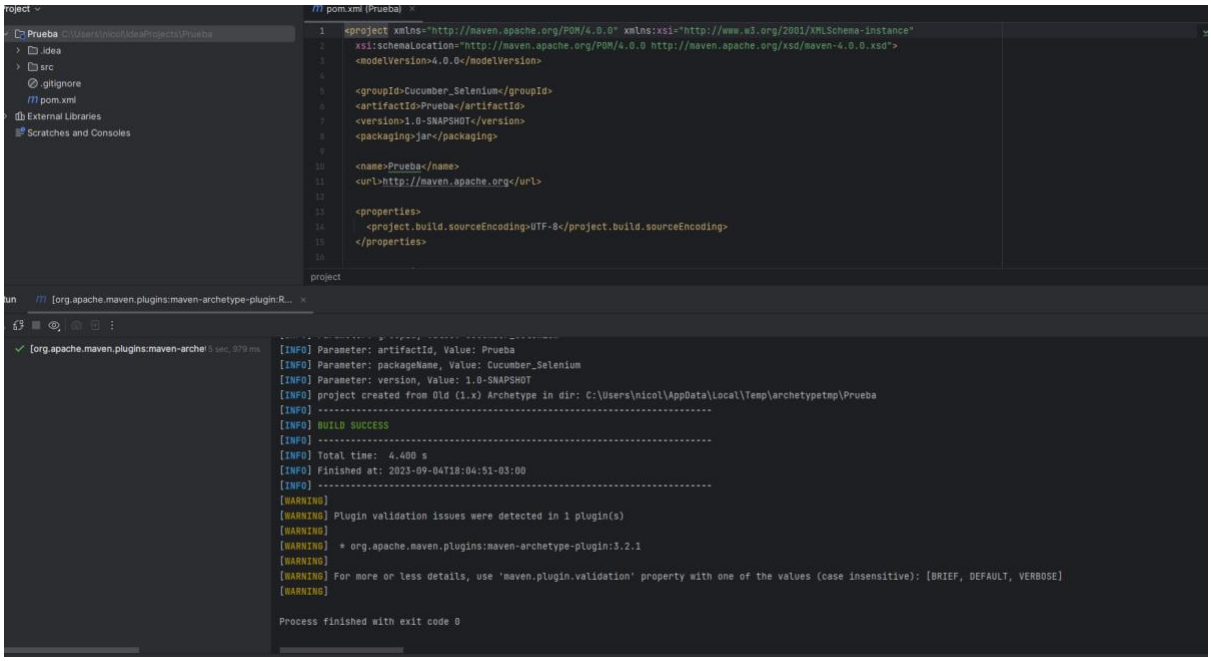


Presionamos el botón “Maven Archetype” y en “Archetype” debemos seleccionar el que termina en “quickstart”.

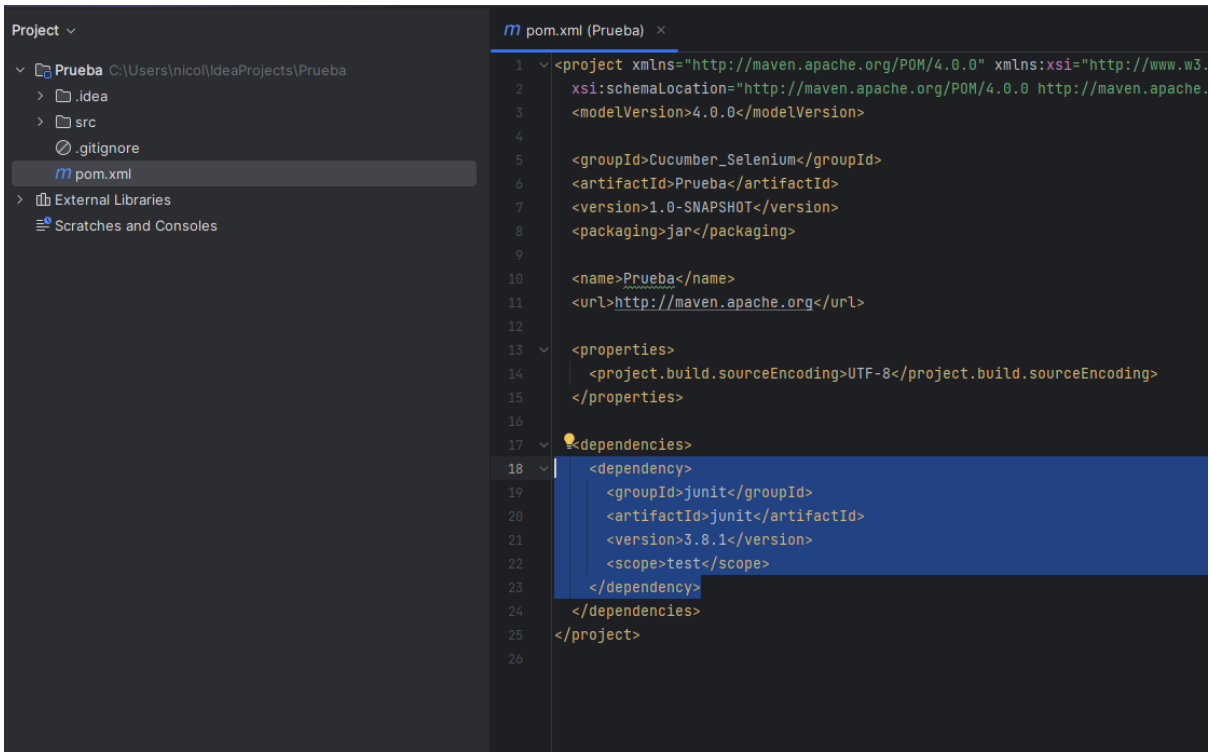


Luego volvemos a “New Project” y apretamos en “Advanced Settings” y escribimos en las dos casillas “Cucumber_Selenium” o el nombre que deseemos y luego apretamos el botón “Create”.





Se borran las dependencias que estaban y se reemplazan por las de abajo



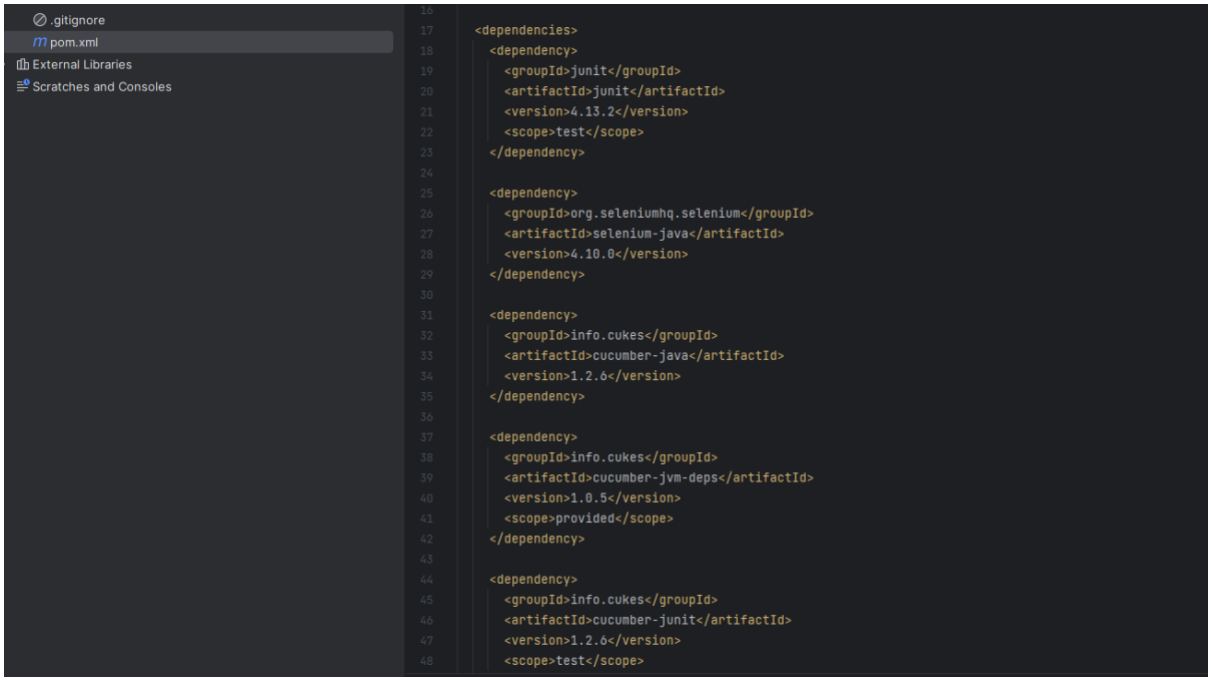
Dependencias (pom.xml)

dependencias necesarias para que funcione todo (Se modifican a medida que se van actualizando)

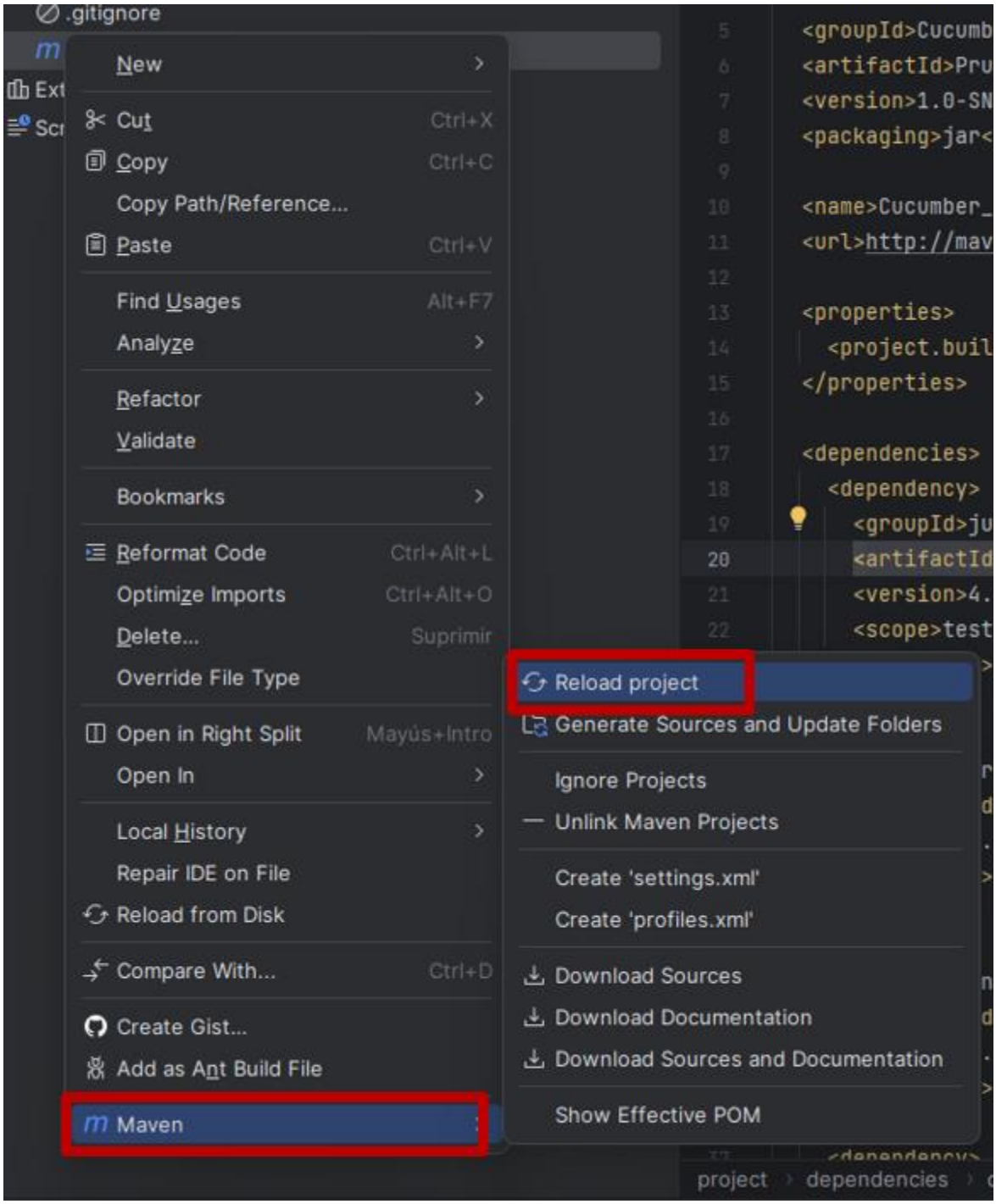
```
<dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>4.13.2</version>
     <scope>test</scope>
   </dependency>

   <dependency>
     <groupId>org.seleniumhq.selenium</groupId>
     <artifactId>selenium-java</artifactId>
     <version>4.10.0</version>
   </dependency>

   <dependency>
     <groupId>info.cukes</groupId>
     <artifactId>cucumber-java</artifactId>
     <version>1.2.6</version>
   </dependency>

   <dependency>
     <groupId>info.cukes</groupId>
     <artifactId>cucumber-jvm-deps</artifactId>
     <version>1.0.5</version>
     <scope>provided</scope>
   </dependency>

   <dependency>
     <groupId>info.cukes</groupId>
     <artifactId>cucumber-junit</artifactId>
     <version>1.2.6</version>
     <scope>test</scope>
   </dependency>

   <dependency>
     <groupId>com.vimalselvam</groupId>
     <artifactId>cucumber-extentsreport</artifactId>
     <version>3.1.1</version>
   </dependency>
```

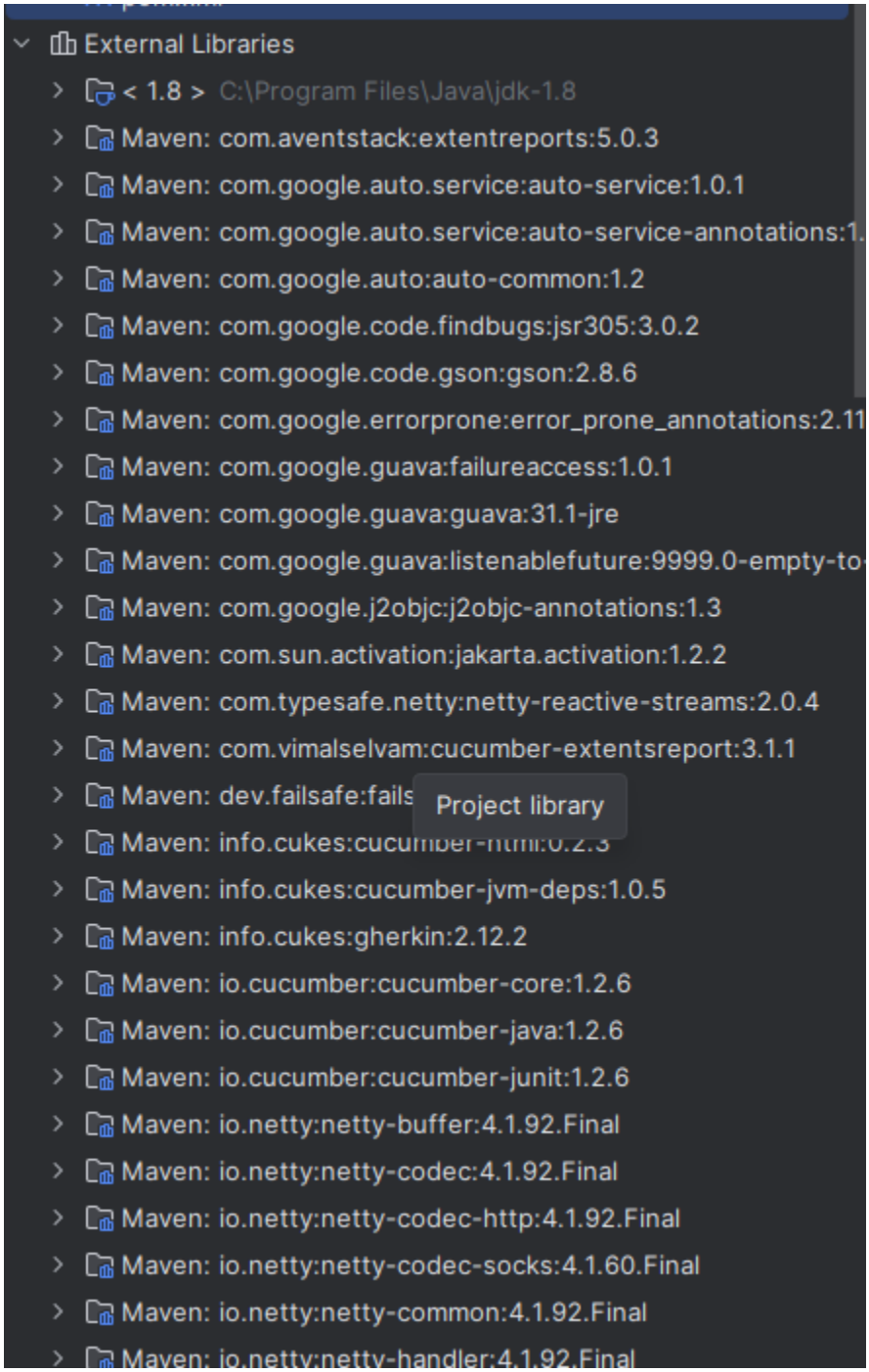
```
<dependency>
  <groupId>com.aventstack</groupId>
  <artifactId>extentreports</artifactId>
  <version>5.0.3</version>
</dependency>

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>2.0.7</version>
</dependency>
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>2.0.7</version>
</dependency>
```

Agregadas las dependencias en el archivo “pom.xml” se integrará todo lo necesario.

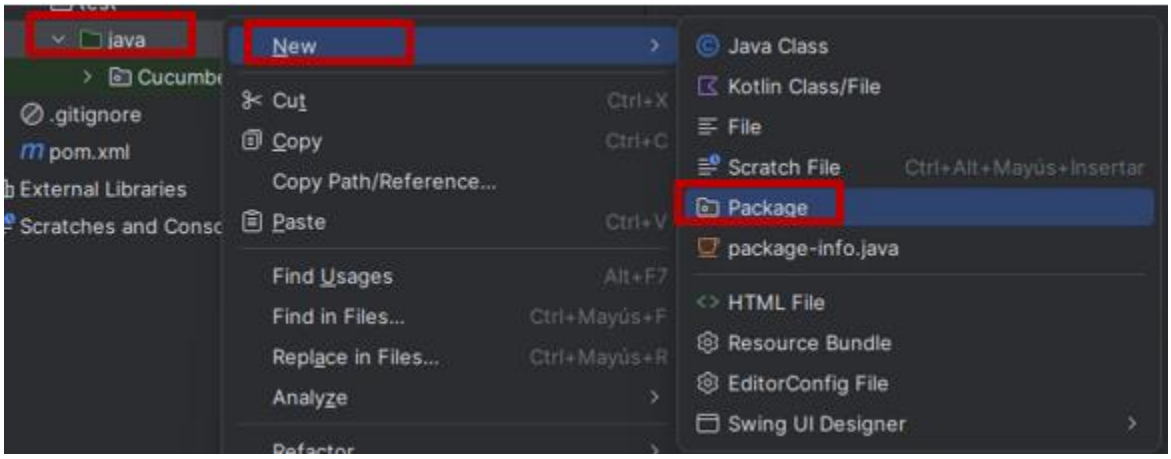


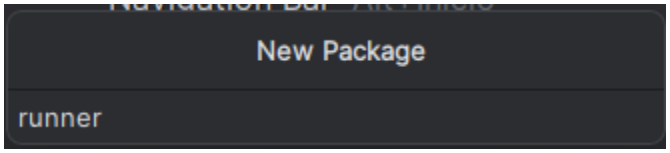
Hacemos clic derecho en el archivo “pom.xml” nos vamos a lo último donde dice “Maven”
y seleccionamos la opción “Reload project”. Una vez hecho esto se integrarán todos los archivos maven que son requeridos.

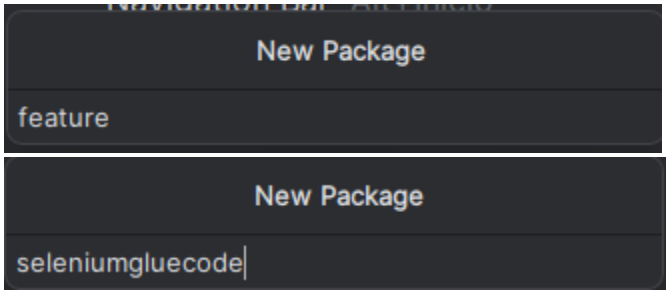



External Libraries
< 1.8 > C:\Program Files\Java\jdk-1.8
Maven: com.aventstack:extentreports:5.0.3
Maven: com.google.auto.service:auto-service:1.0.1
Maven: com.google.auto.service:auto-service-annotations:1.
Maven: com.google.auto:auto-common:1.2
Maven: com.google.code.findbugs:jsr305:3.0.2
Maven: com.google.code.gson:gson:2.8.6
Maven: com.google.errorprone:error_prone_annotations:2.11
Maven: com.google.guava:failureaccess:1.0.1
Maven: com.google.guava:guava:31.1-jre
Maven: com.google.guava:listenablefuture:9999.0-empty-to
Maven: com.google.j2objc:j2objc-annotations:1.3
Maven: com.sun.activation:jakarta.activation:1.2.2
Maven: com.typesafe.netty:netty-reactive-streams:2.0.4
Maven: com.vimalselvam:cucumber-extentsreport:3.1.1
Maven: dev.failsafe:fails
Project library
Maven: info.cukes:cucumber-jvm-deps:1.0.5
Maven: info.cukes:gherkin:2.12.2
Maven: io.cucumber:cucumber-core:1.2.6
Maven: io.cucumber:cucumber-java:1.2.6
Maven: io.cucumber:cucumber-junit:1.2.6
Maven: io.netty:netty-buffer:4.1.92.Final
Maven: io.netty:netty-codec:4.1.92.Final
Maven: io.netty:netty-codec-http:4.1.92.Final
Maven: io.netty:netty-codec-socks:4.1.60.Final
Maven: io.netty:netty-common:4.1.92.Final

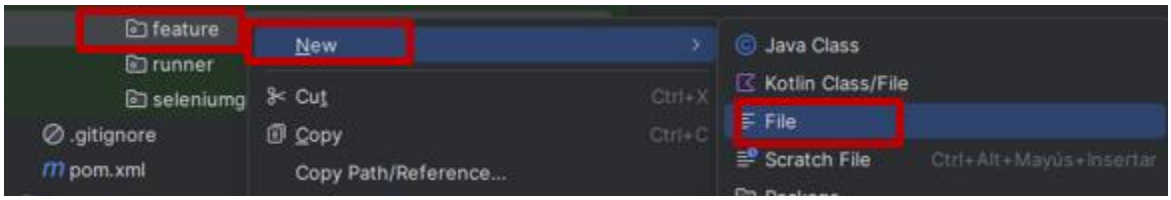
Luego en la ruta del “src/java” haremos clic derecho en “Java” para agregar un “package” con los siguientes nombres que se muestran en las imágenes. uno por uno. “feature, runner, selenium” Enter a cada uno para agregarlos.

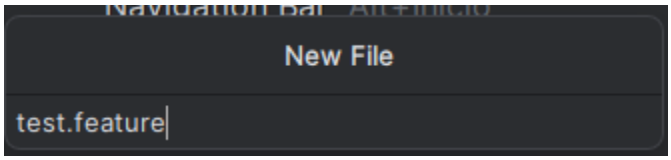




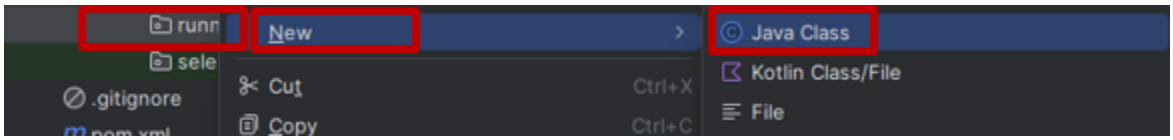


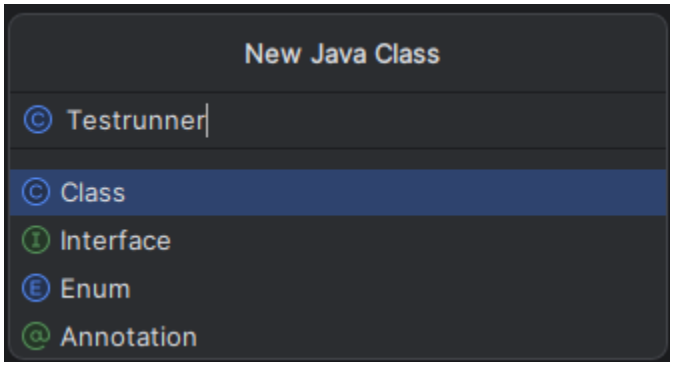
luego en el package “features” crearemos un File con el nombre “test.feature”. aquí crearemos nuestros escenarios con lenguaje natural.



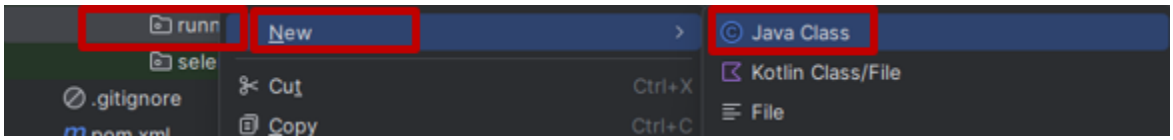


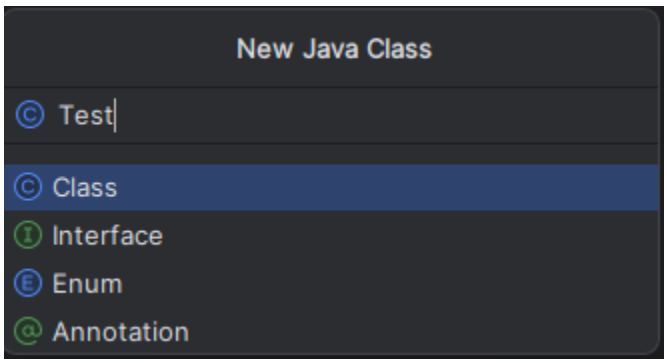
Luego en el package “runner” crearemos nuestro motor haciendo clic derecho en New y agregando un “Java Class” con el nombre “Testrunner”



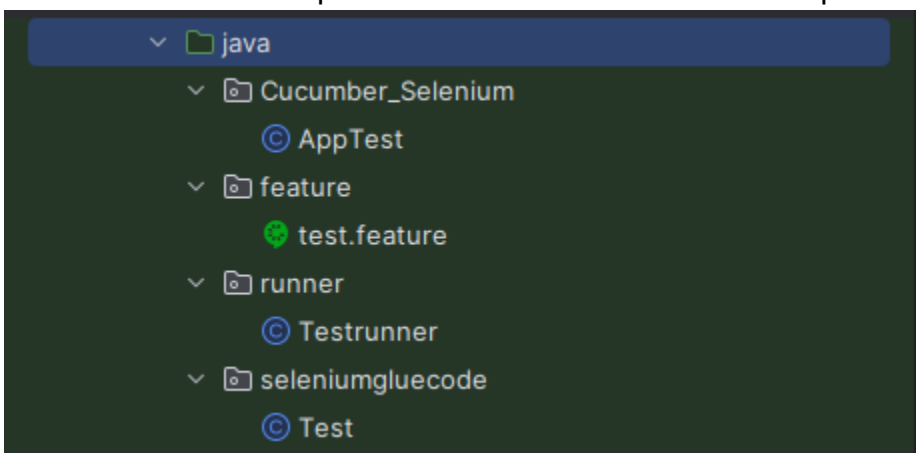


Por último, en el package “seleniumgluecode” agregaremos la clase “Test” donde configuraremos la prueba de automatización para que pueda funcionar de forma correcta.

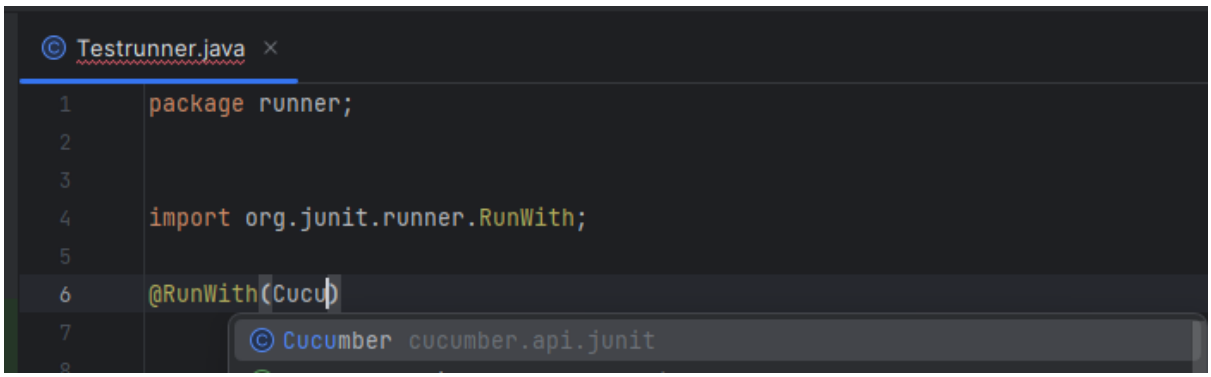


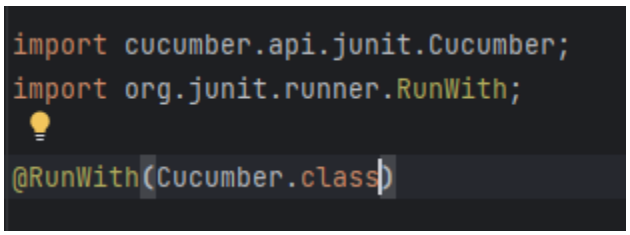


Así es como debería quedarnos una vez hecho todos los pasos anteriores.



Nos dirigimos a la clases “Testrunner” y procedemos a escribir @RunWith(Cucumber.class)

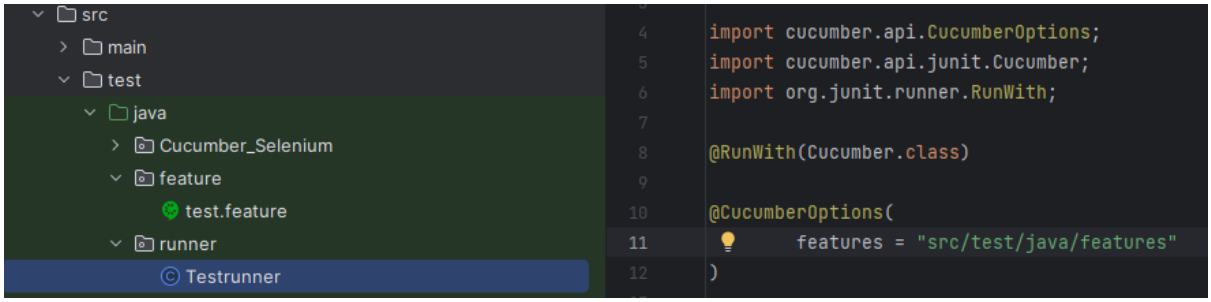




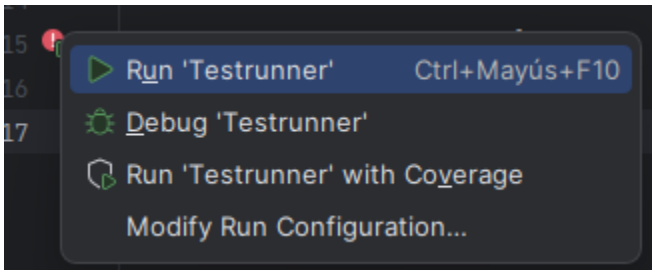
Luego procedemos a escribir la ruta donde se encuentra nuestro lenguaje natural para que lo pueda reconocer. (@Cucumber.class{

features = “src/test/java/features”

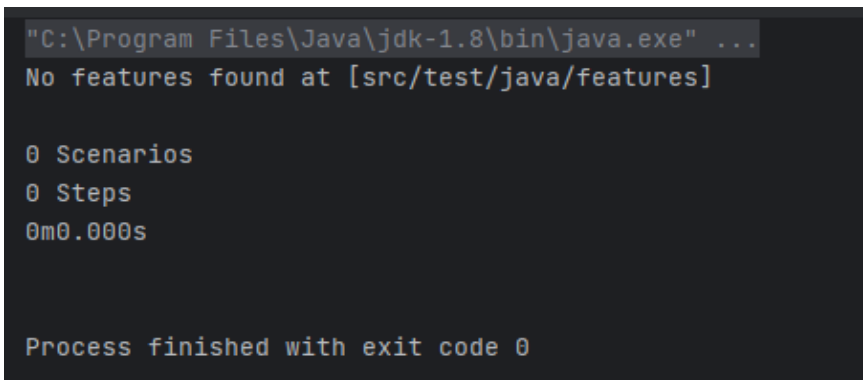
}

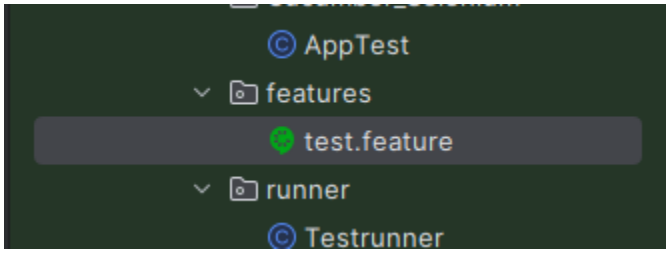


Una vez hecho esto procedemos a correr la clase “Testrunner” con f5 o bien en la parte superior nos debería dar la opción.



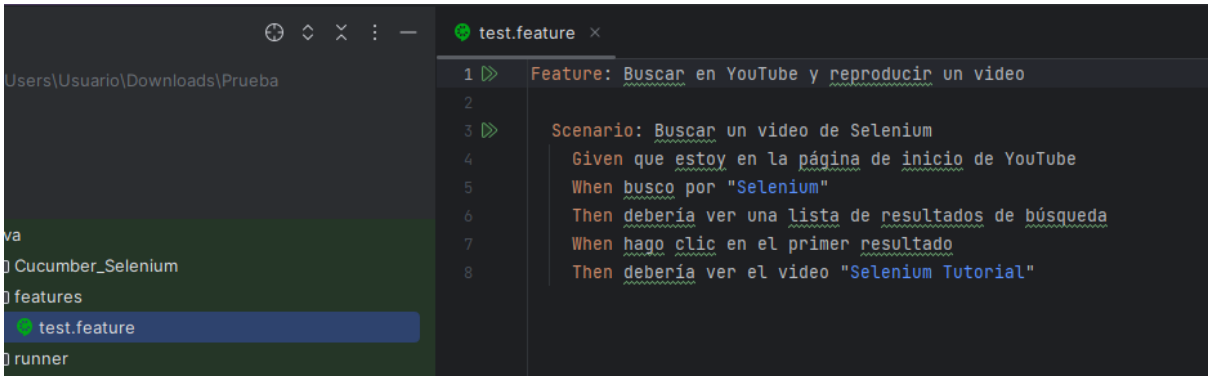
Al terminar de correr la clase nos saldrá como respuesta esto. es correcto ya que aun no creamos los escenarios de prueba.



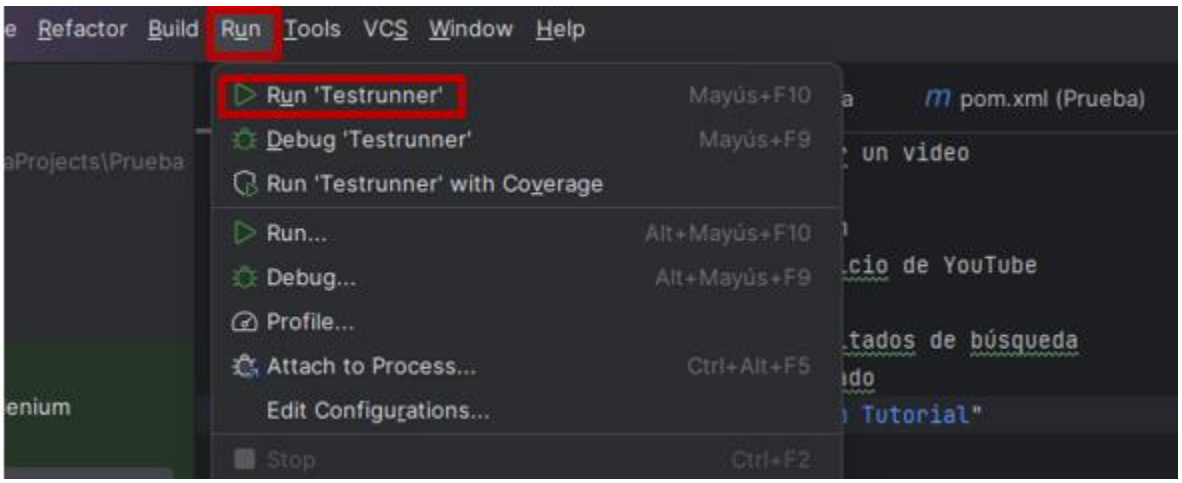


Estando en el File “test.feature” crearemos nuestro escenario en lenguaje natural.

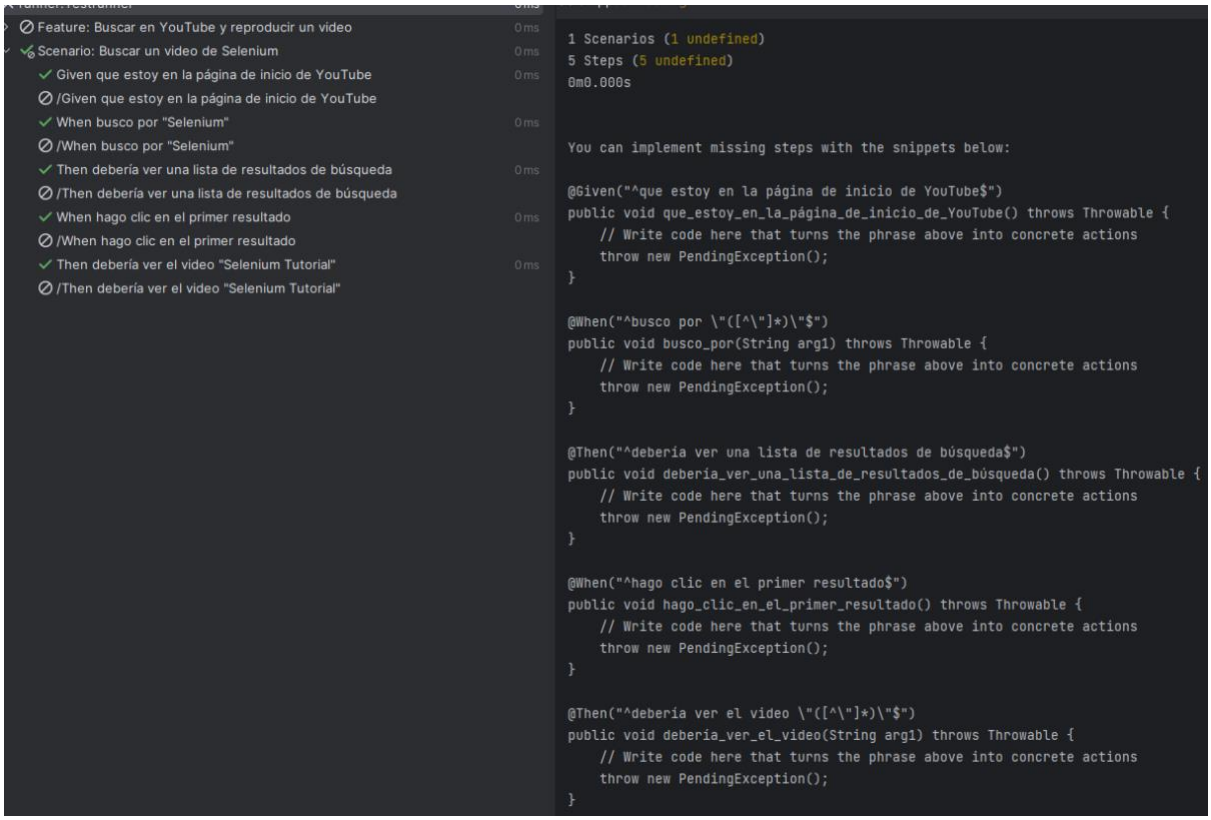
- Feature: Aquí se da el contexto del escenario que se quiere crear.
- Scenario: Una breve descripción de lo que se hará
- Given: Desde donde comenzaré mi plan de automatización la página inicial.
- When: Describo lo que quiero buscar o hacer.
- Then: Lo que debería suceder si hice algo anteriormente.



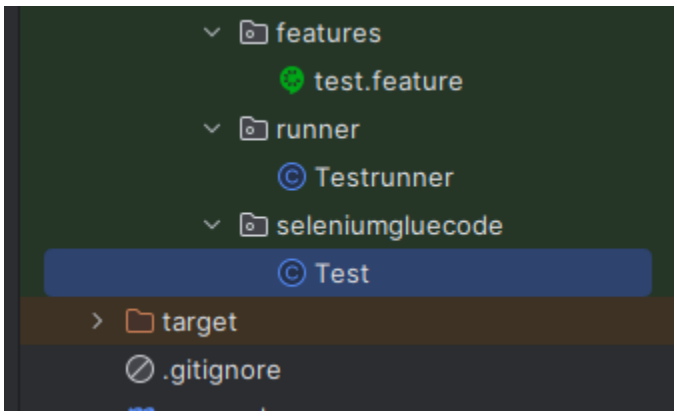
Una vez hecho el paso anterior procedemos a correr el “Testrunner”



Al terminar de correr el test nos arrojará abajo que fueron ignoradas las pruebas y más abajo nos saldrá lo siguiente si seguimos todos los pasos. Copiaremos todo lo que se encuentra abajo marcado.



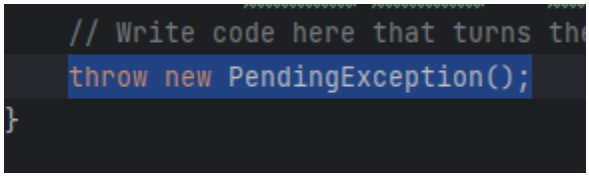
Una vez copiado lo anterior nos dirigiremos a la clase “Test” y lo pegamos.



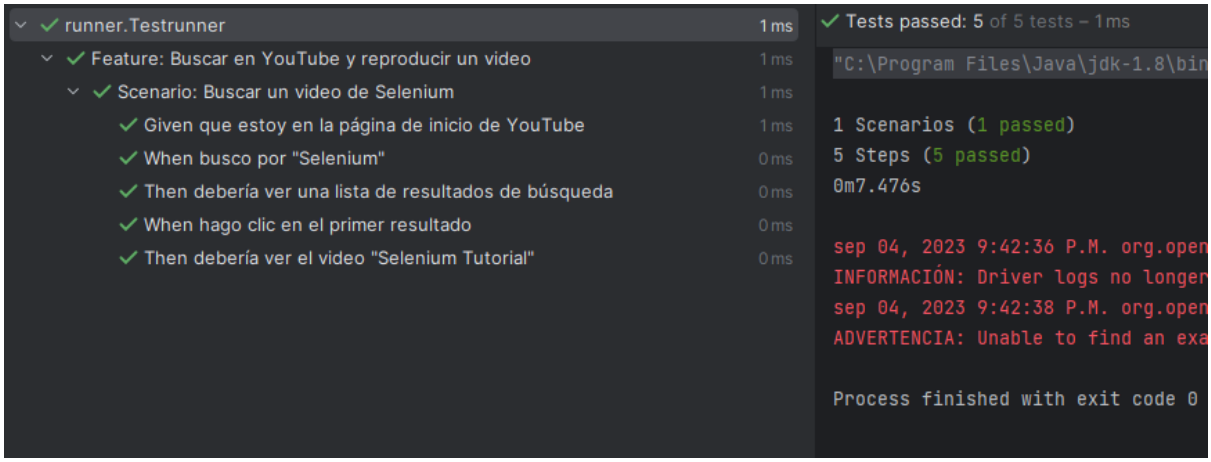
Lo copiaremos debajo de “public class Test {“. Los Given, When, Then se tendrán que agregar con la tecla “alt + enter” encima de cada uno.

```
public class Test {

    @Given("^que estoy en la página de inicio de YouTube$")
    public void que_estoy_en_la_página_de_inicio_de_YouTube() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @When("^busco por \"([^\"]*)\"$")
    public void busco_por(String arg1) throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Then("^debería ver una lista de resultados de búsqueda$")
    public void debería_ver_una_lista_de_resultados_de_búsqueda() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @When("^hago clic en el primer resultado$")
    public void hago_clic_en_el_primer_resultado() throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }

    @Then("^debería ver el video \"([^\"]*)\"$")
    public void debería_ver_el_video(String arg1) throws Throwable {
        // Write code here that turns the phrase above into concrete actions
        throw new PendingException();
    }
```

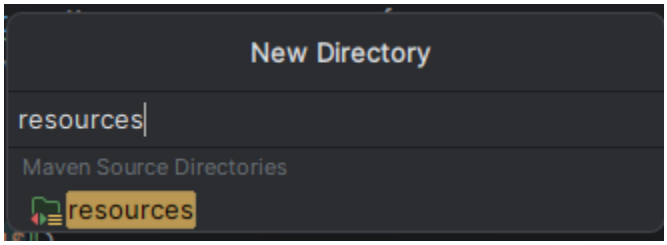
Procedemos a eliminar lo siguiente.

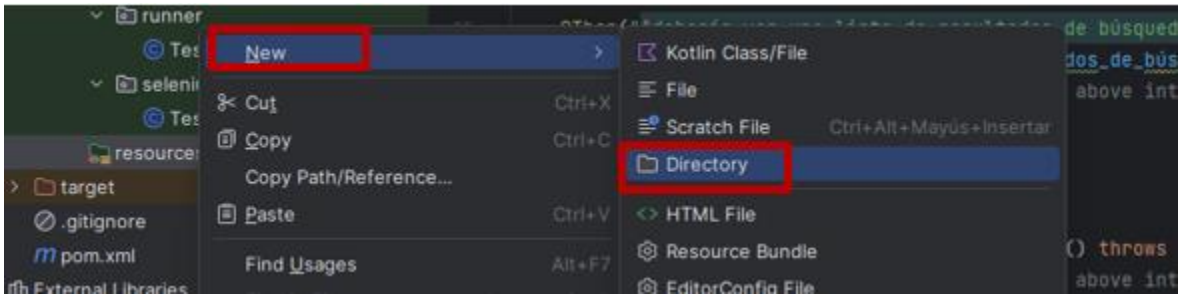


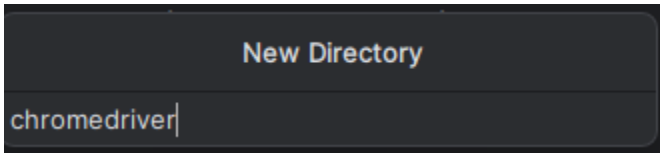
Luego procedemos a correr nuevamente nuestra prueba y debería arrojarnos este resultado.



Para poder darle vida a nuestra prueba de automatización empezaremos a configurar la carpeta donde se encontrará la aplicación “Chromedriver” haremos clic derecho en la carpeta test y agregaremos un “New Directory” con el nombre “resources” y dentro de ella haciendo clic derecho haremos otro “New Directory” con el nombre “chromedriver”.

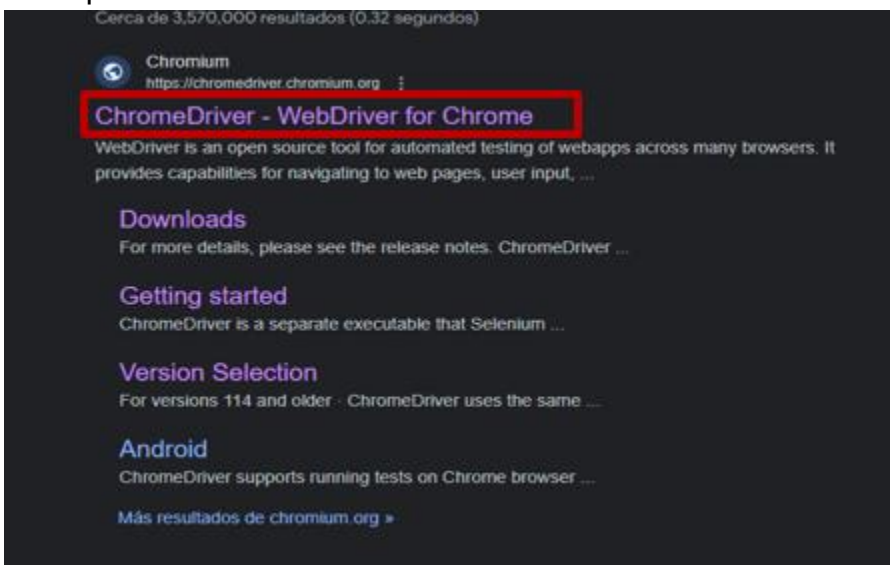




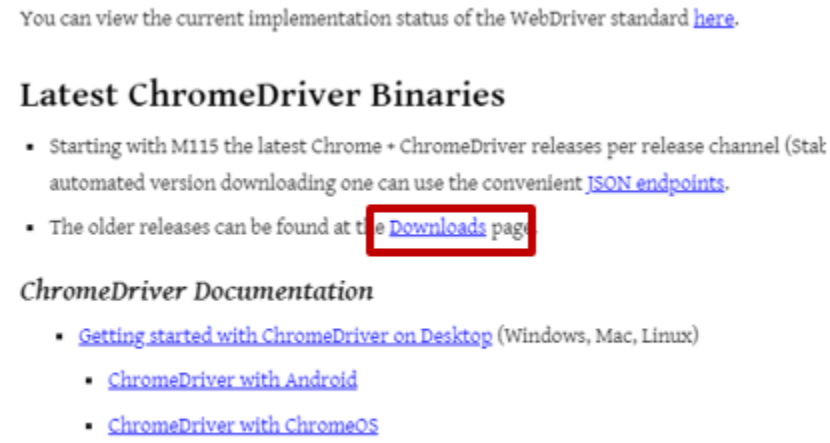


Abriremos nuestro navegador de preferencia y buscaremos “chromedriver”.

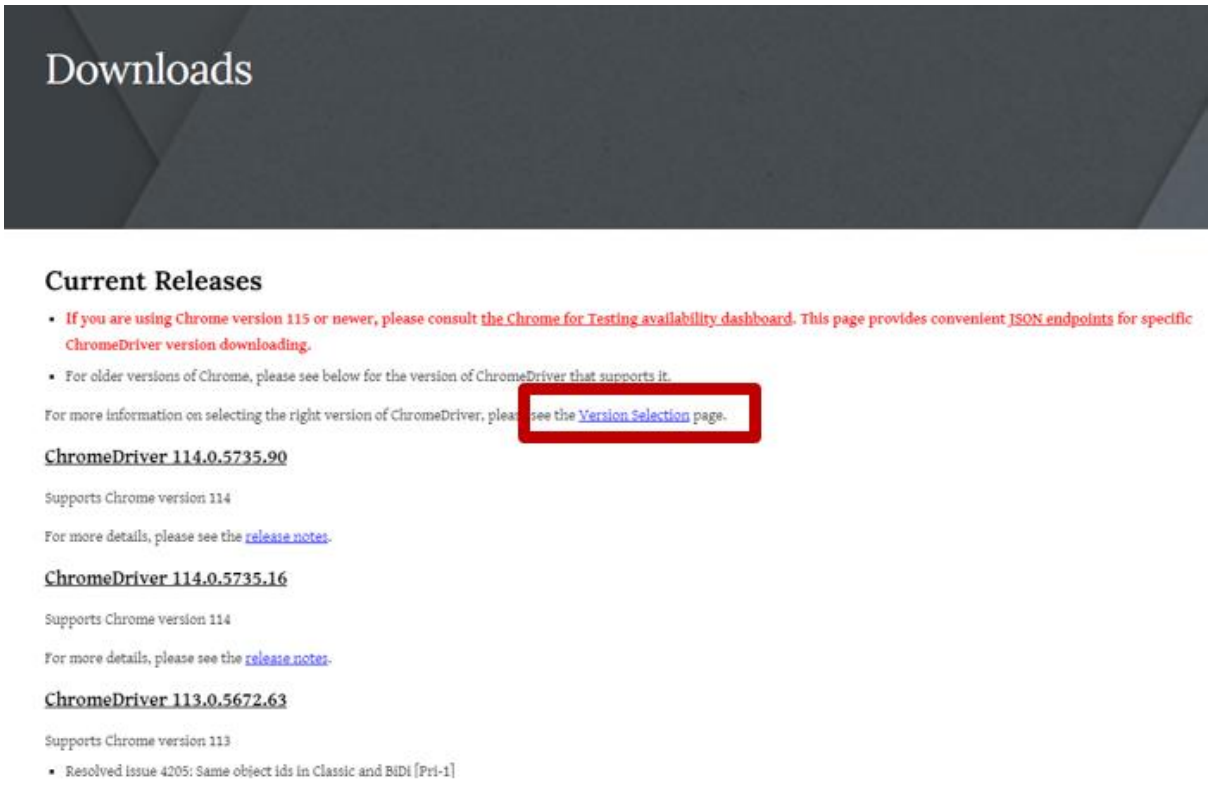
En el primer link nos meteremos.



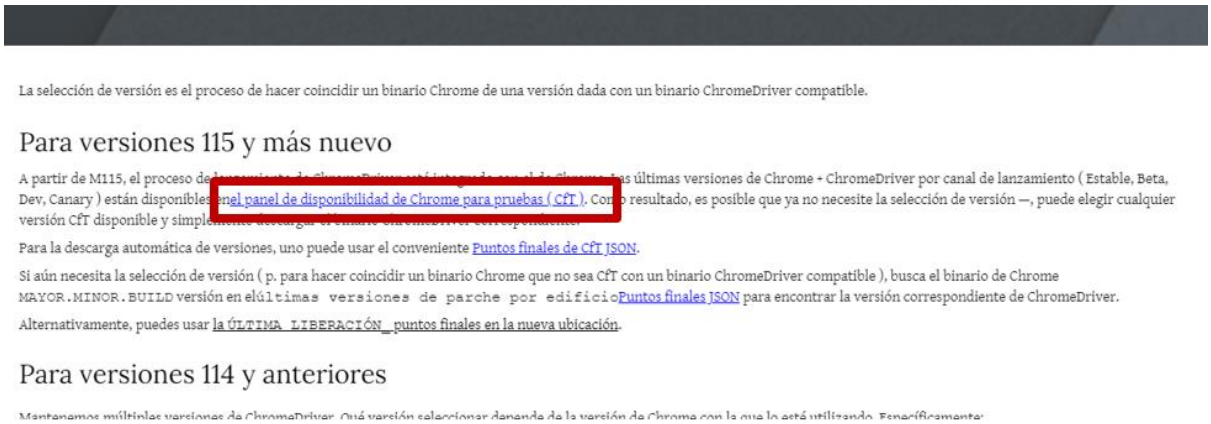
Haremos clic en la opción que dice “Download” esta nos redirigirá a la página de descarga



Si nuestro chrome tiene una superior a las que se encuentran ahí procederemos a presionar el botón “Version Selection”



Y presionaremos el primer link donde se encuentran versiones superiores o igual a las 115.



La selección de versión es el proceso de hacer coincidir un binario Chrome de una versión dada con un binario ChromeDriver compatible.

Para versiones 115 y más nuevo

A partir de M115, el proceso de [illegible] las últimas versiones de Chrome + ChromeDriver por canal de lanzamiento (Estable, Beta, Dev, Canary) están disponibles en el panel de disponibilidad de Chrome para pruebas (CfT). Como resultado, es posible que ya no necesite la selección de versión —, puede elegir cualquier versión CfT disponible y simplemente descargar el binario ChromeDriver correspondiente.

Para la descarga automática de versiones, uno puede usar el conveniente Puntos finales de CfT JSON.

Si aún necesita la selección de versión (p. para hacer coincidir un binario Chrome que no sea CfT con un binario ChromeDriver compatible), busca el binario de Chrome `MAYOR.MINOR.BUILD` versión en el`últimas versiones de parche por edificio`Puntos finales JSON para encontrar la versión correspondiente de ChromeDriver.

Alternativamente, puedes usar la `ÚLTIMA_LIBERACIÓN_` puntos finales en la nueva ubicación.

Para versiones 114 y anteriores

Mantenemos múltiples versiones de ChromeDriver. Qué versión seleccionar depende de la versión de Chrome con la que lo esté utilizando. Específicamente:

Y buscaremos la versión de nuestro google chrome en este caso es la 116.

Chrome for Testing availability

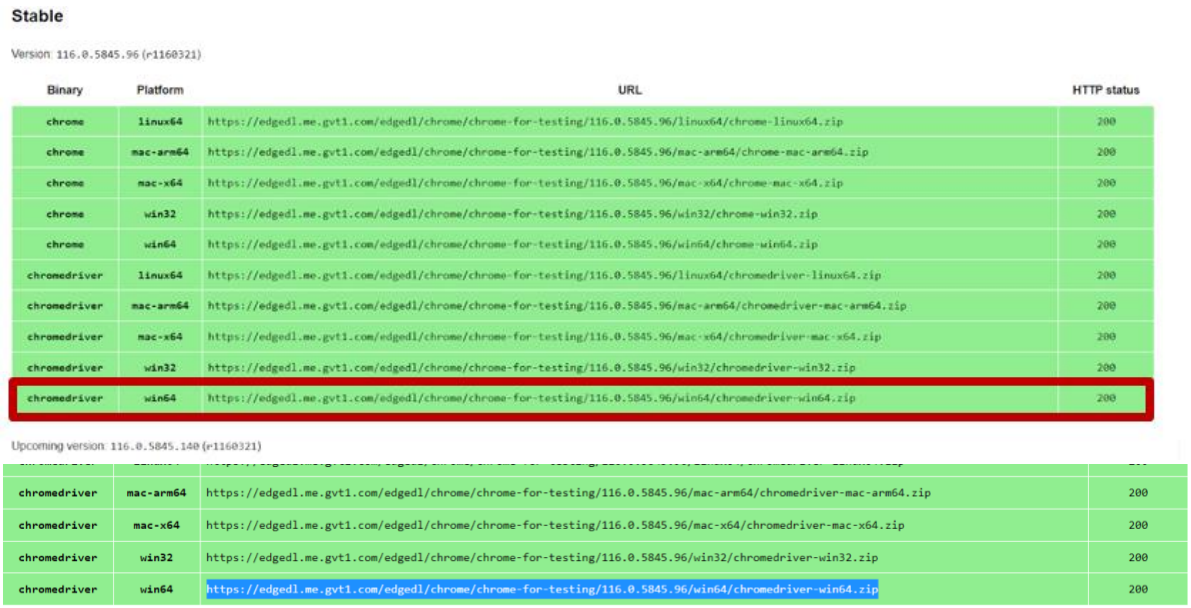
This page lists the latest available cross-platform Chrome for Testing versions and assets per Chrome release channel.

Consult our JSON API endpoints if you're looking to build automated scripts based on Chrome for Testing release data.

Last updated @ 2023-09-04T22:08:03.767Z

Channel	Version	Revision	Status
Stable	116.0.5845.96	r1160321	✓
Stable (upcoming)	116.0.5845.140	r1160321	✗
Beta	117.0.5938.35	r1181205	✓
Dev	118.0.5979.0	r1189757	✓
Canary	118.0.5982.0	r1190646	✓
Canary (upcoming)	118.0.5990.0	r1191994	✗

Elegida la opción procedemos a elegir nuestra plataforma para descargar en este caso windows 11 64 bit.



Stable

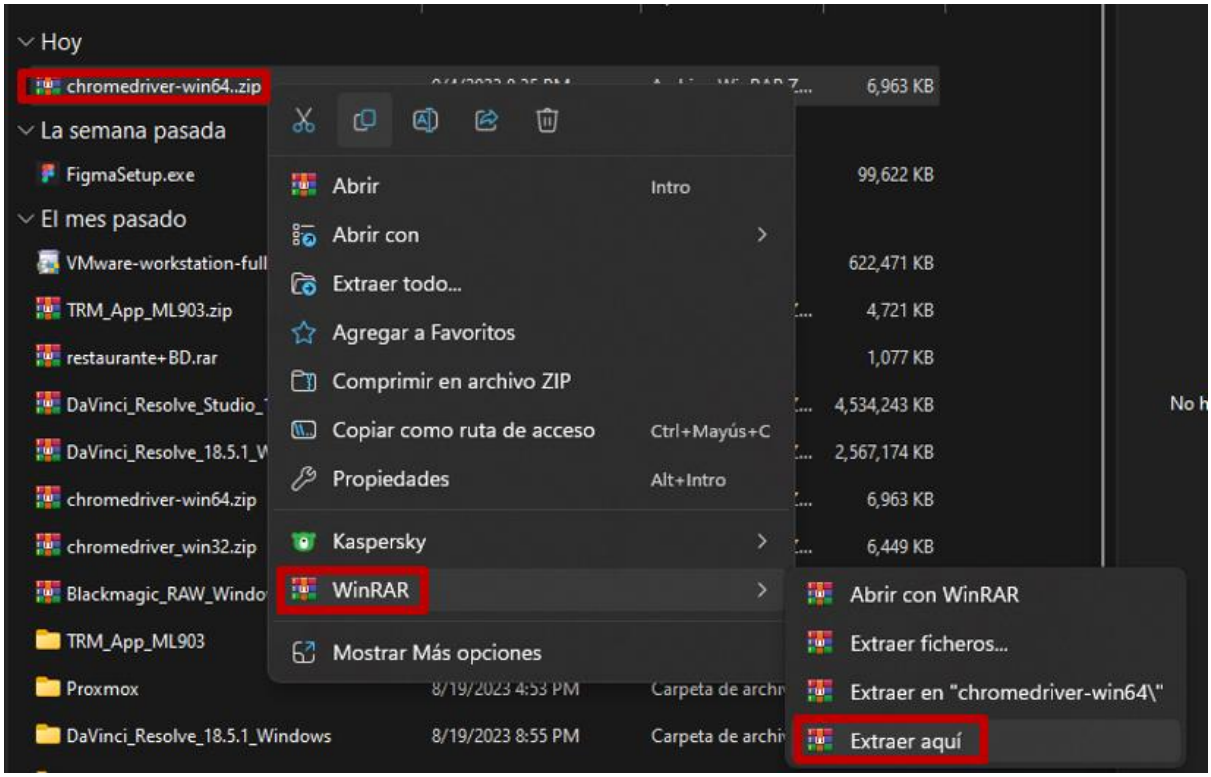
Version: 116.0.5845.96 (r1160321)

Binary	Platform	URL	HTTP status
chrome	linux64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/linux64/chrome-linux64.zip	200
chrome	mac-arm64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/mac-arm64/chrome-mac-arm64.zip	200
chrome	mac-x64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/mac-x64/chrome-mac-x64.zip	200
chrome	win32	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/win32/chrome-win32.zip	200
chrome	win64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/win64/chrome-win64.zip	200
chromedriver	linux64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/linux64/chromedriver-linux64.zip	200
chromedriver	mac-arm64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/win32/chromedriver-win32.zip	200
chromedriver	win64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/win64/chromedriver-win64.zip	200

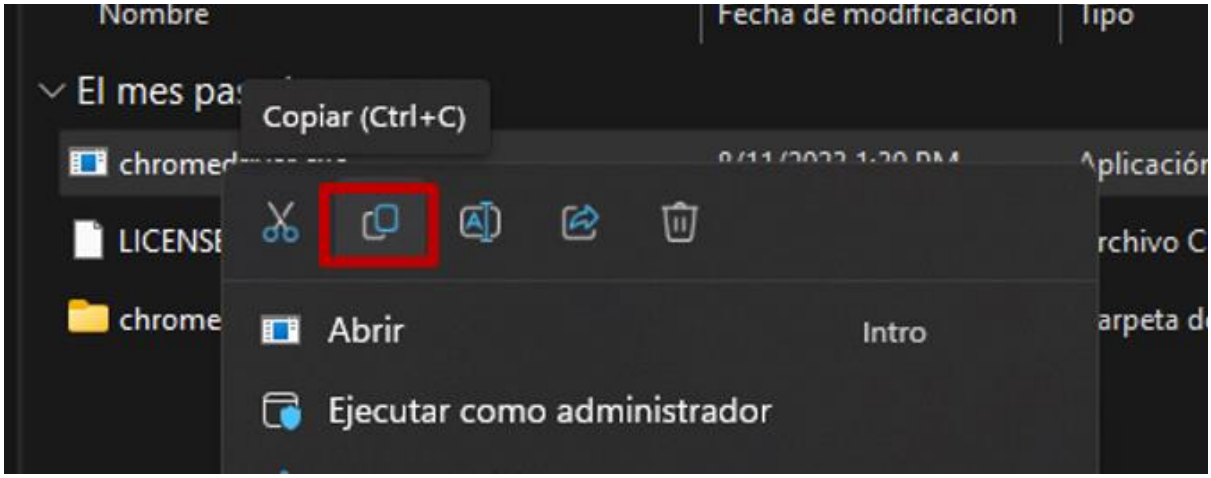
Upcoming version: 116.0.5845.140 (r1160321)

chromedriver	mac-arm64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/mac-arm64/chromedriver-mac-arm64.zip	200
chromedriver	mac-x64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/mac-x64/chromedriver-mac-x64.zip	200
chromedriver	win32	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/win32/chromedriver-win32.zip	200
chromedriver	win64	https://edgedl.me.gvt1.com/edgedl/chrome/chrome-for-testing/116.0.5845.96/win64/chromedriver-win64.zip	200

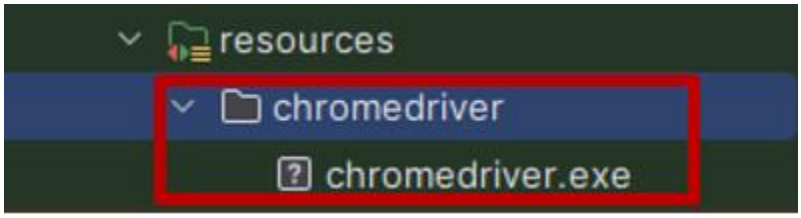
Una vez descargado procedemos a descomprimir el archivo.



Procedemos a copiar el aplicativo “chromerdriver” y pegarlo en esta direccion “C:\Users\Usuario\IdeaProjects” en la mayoría de los casos debería ser esa la dirección para pegar el aplicativo.



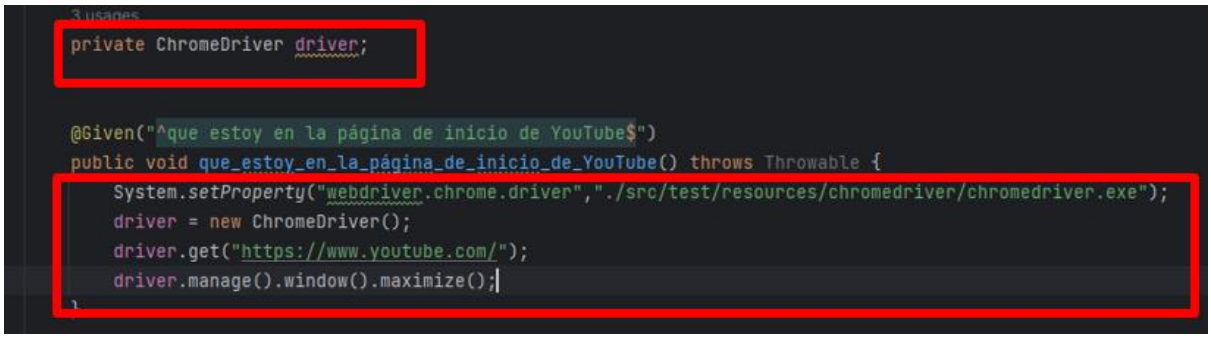
Una vez pegado el aplicativo deberá actualizarse la carpeta que creamos anteriormente con el aplicativo agregado.



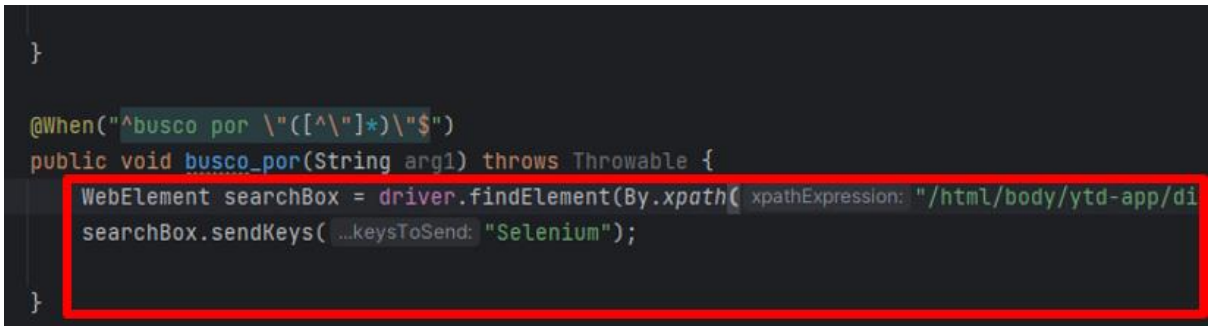
Para configurar el primer paso escribiremos lo siguiente.

- private ChromeDriver driver;

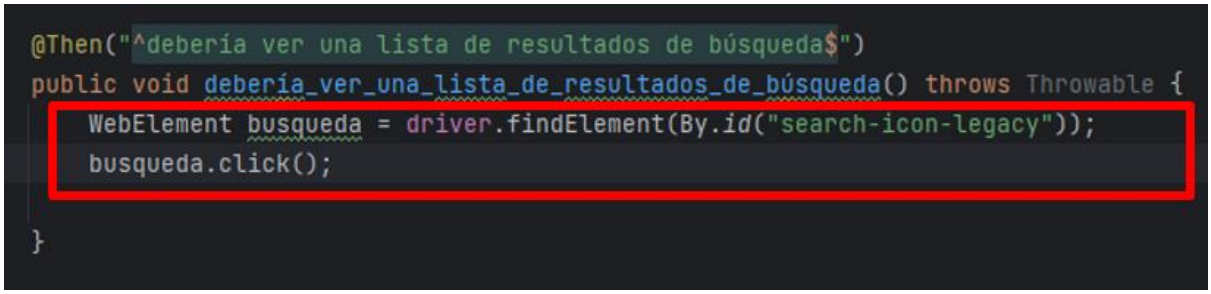
por debajo de “public void” escribiremos lo siguiente. Esto es para hacer funcionar el Chromedriver e iniciar la página que queremos realizar las pruebas.



En el segundo paso donde se desea buscar cierta palabra en el buscador de youtube. se hará la configuración con el siguiente localizador. Los localizadores deben ser únicos para que se puedan realizar las pruebas de automatización correctamente



Aquí se utiliza un localizador del botón de búsqueda de youtube en este caso se utilizó el de la “id”.



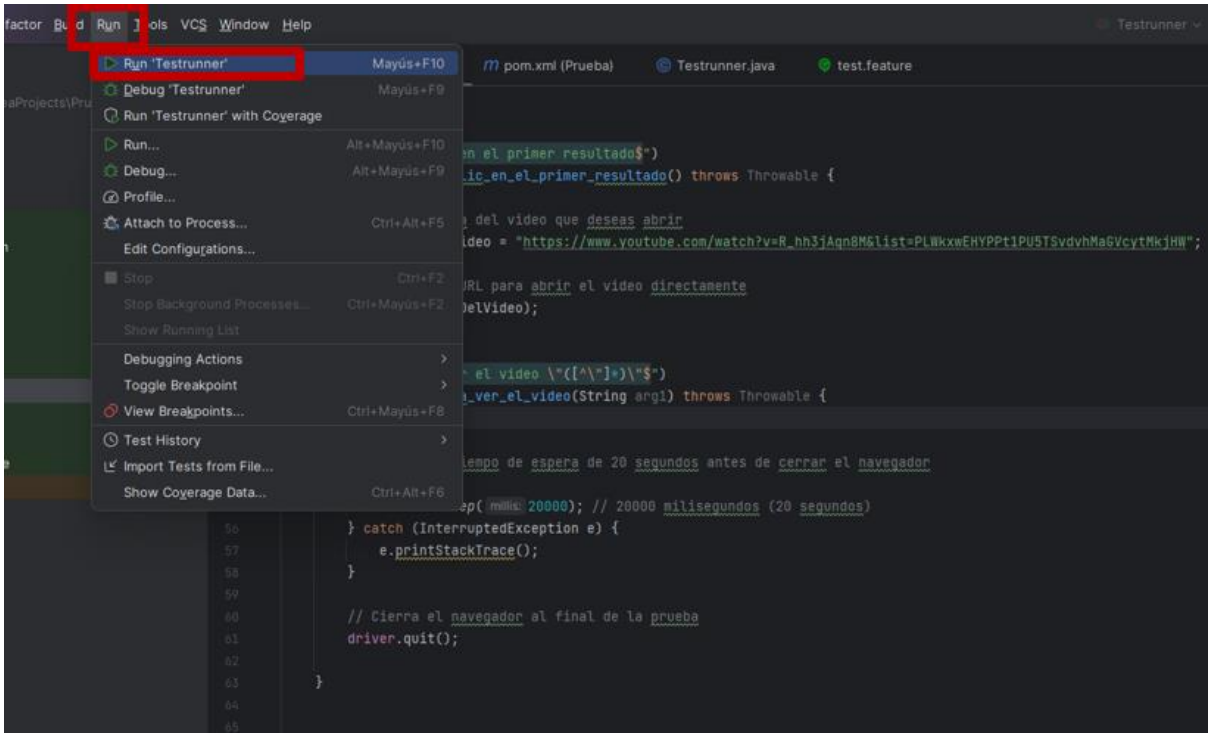
En este caso no se encontró un Localizador único, se tuvo que utilizar la dirección del video para que funcionara. la idea es no hacer esto ya que en una prueba real la idea es verificar que la aplicación web funcione de manera correcta.

El último escenario de pruebas daremos un par de segundos si así lo deseamos para que se mantenga o bien se cierre inmediato. en este caso se dieron 20 segundos y luego para terminar la prueba se cierra el navegador. si no ponemos el último código se mantendrá la prueba abierta hasta que nosotros cerremos el navegador.

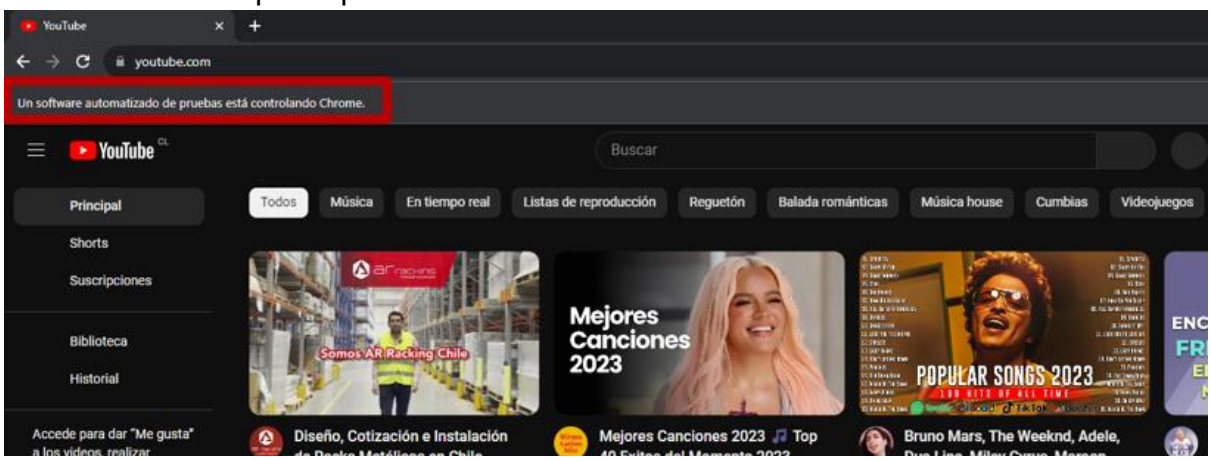
```java
@Then("^debería ver el video \"([^\"]*)\"$")
public void debería_ver_el_video(String arg1) throws Throwable {

    // Agrega un tiempo de espera de 20 segundos antes de cerrar el navegador
    try {
        Thread.sleep( millis: 20000); // 20000 milisegundos (20 segundos)
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    // Cierra el navegador al final de la prueba
    driver.quit();

}
```

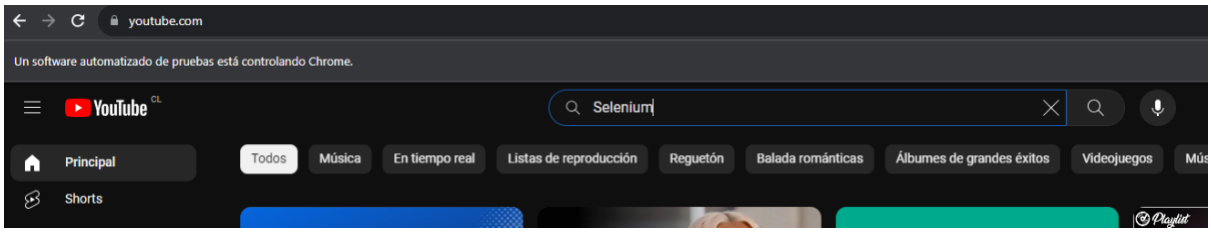
Inicio de Pruebas de Automatización

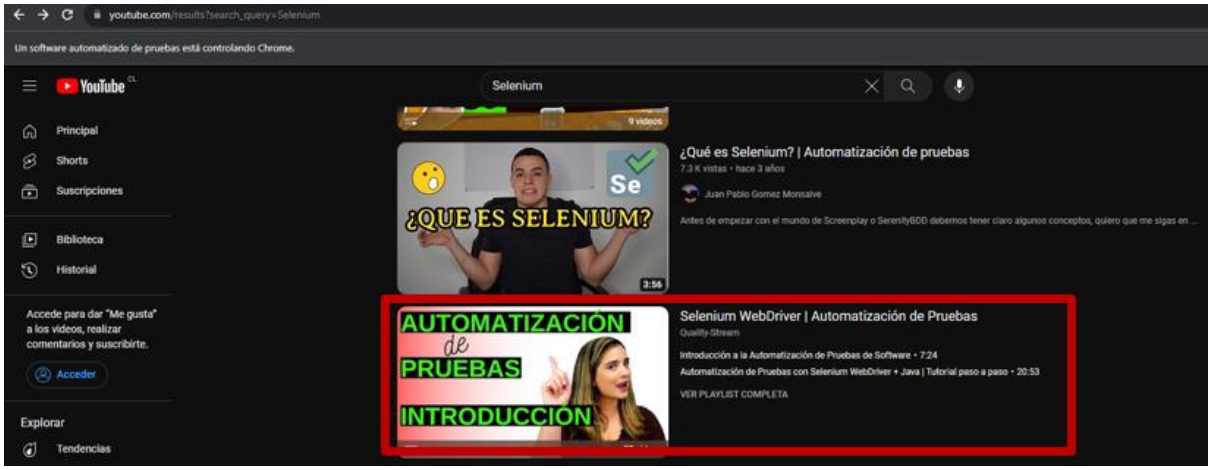
Por último, para probar que todo funciona correremos la prueba.

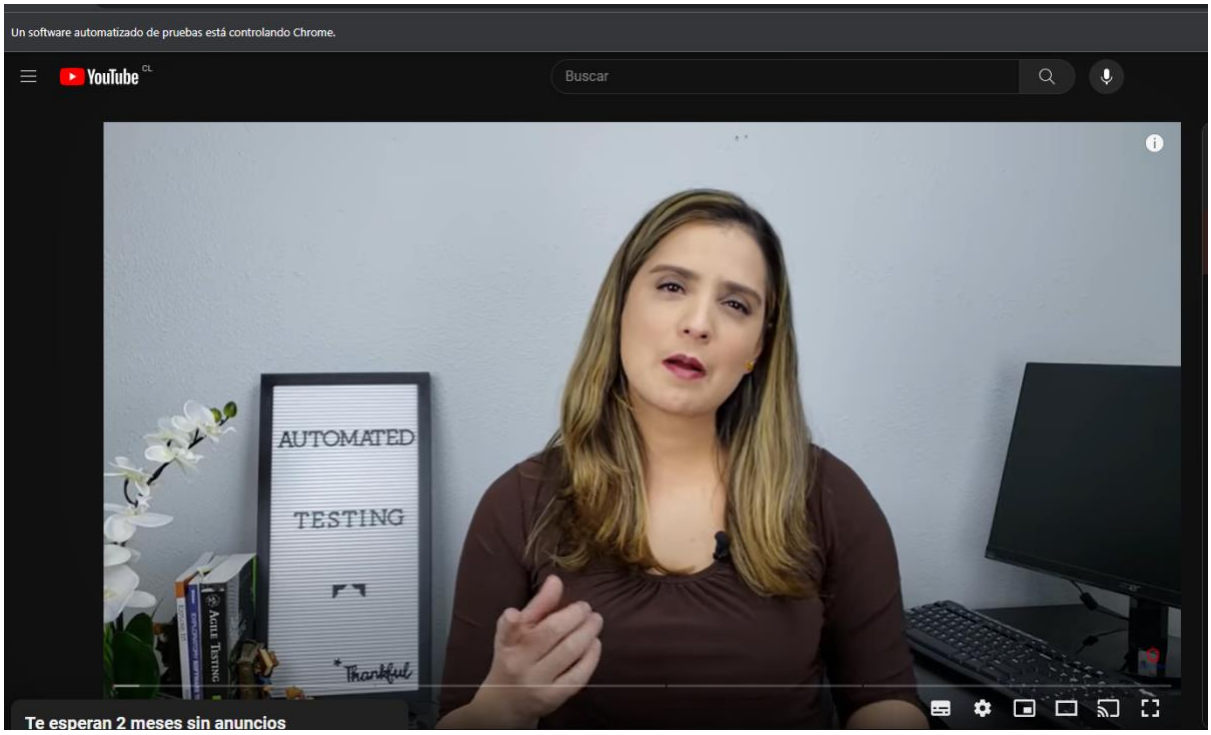


Visualización de que la prueba está en curso.

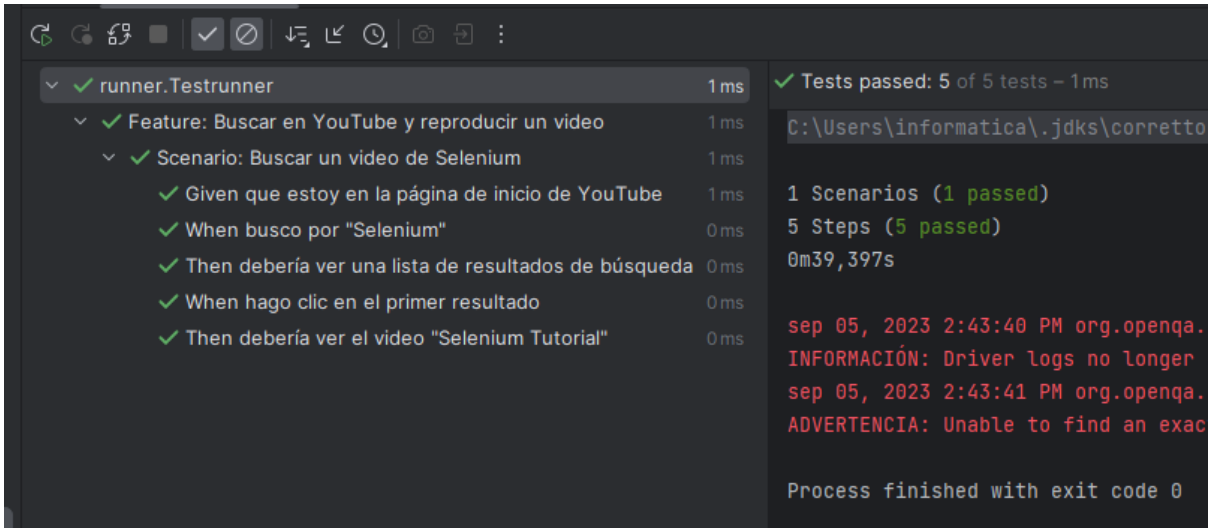








Al terminar la prueba se cerrará el navegador y nos dará como resultado que el escenario de pruebas fue realizado con éxito.

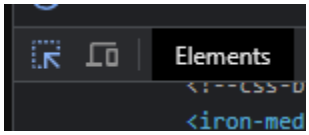


Localizadores

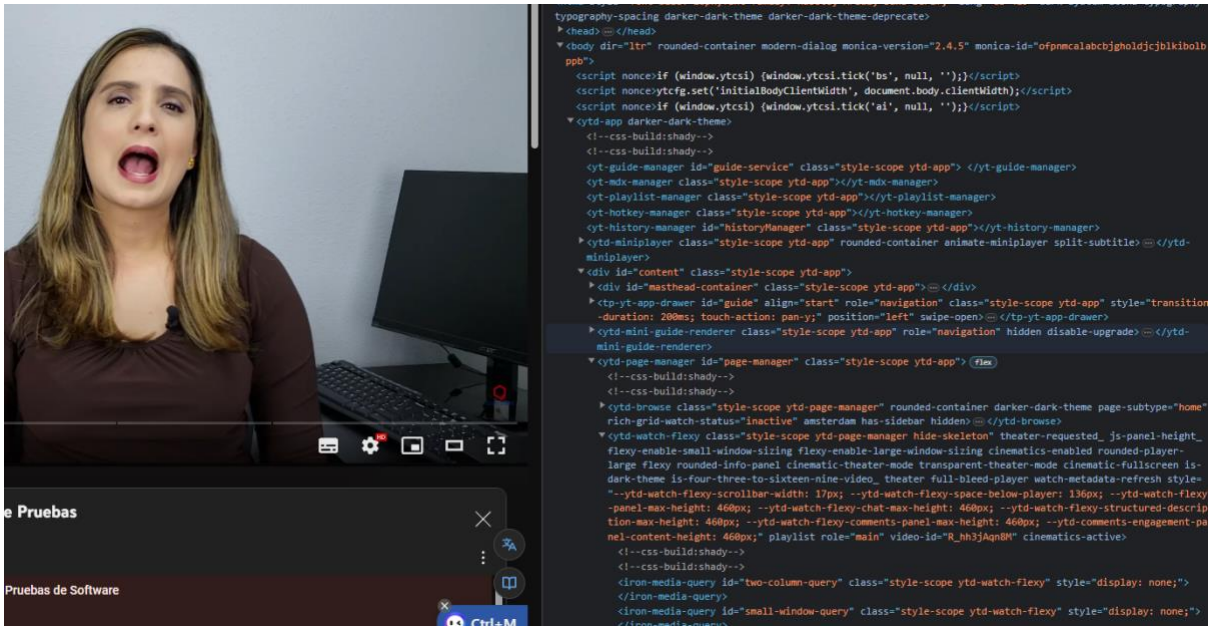
- By.id("id_del_elemento"): Localiza un elemento por su atributo "id".
- By.name("nombre_del_elemento"): Localiza un elemento por su atributo "name".
- By.className("nombre_de_clase"): Localiza un elemento por su clase CSS.
- By.cssSelector("selector_css"): Localiza un elemento utilizando un selector CSS.
- By.linkText("texto_del_enlace"): Localiza un enlace por el texto exacto del enlace.
- By.partialLinkText("parte_del_texto_del_enlace"): Localiza un enlace por una parte del texto del enlace.
- By.tagName("nombre_de_la_etiqueta"): Localiza elementos por el nombre de la etiqueta HTML.
- By.xpath("expresion_xpath"): Localiza elementos utilizando una expresión XPath personalizada.

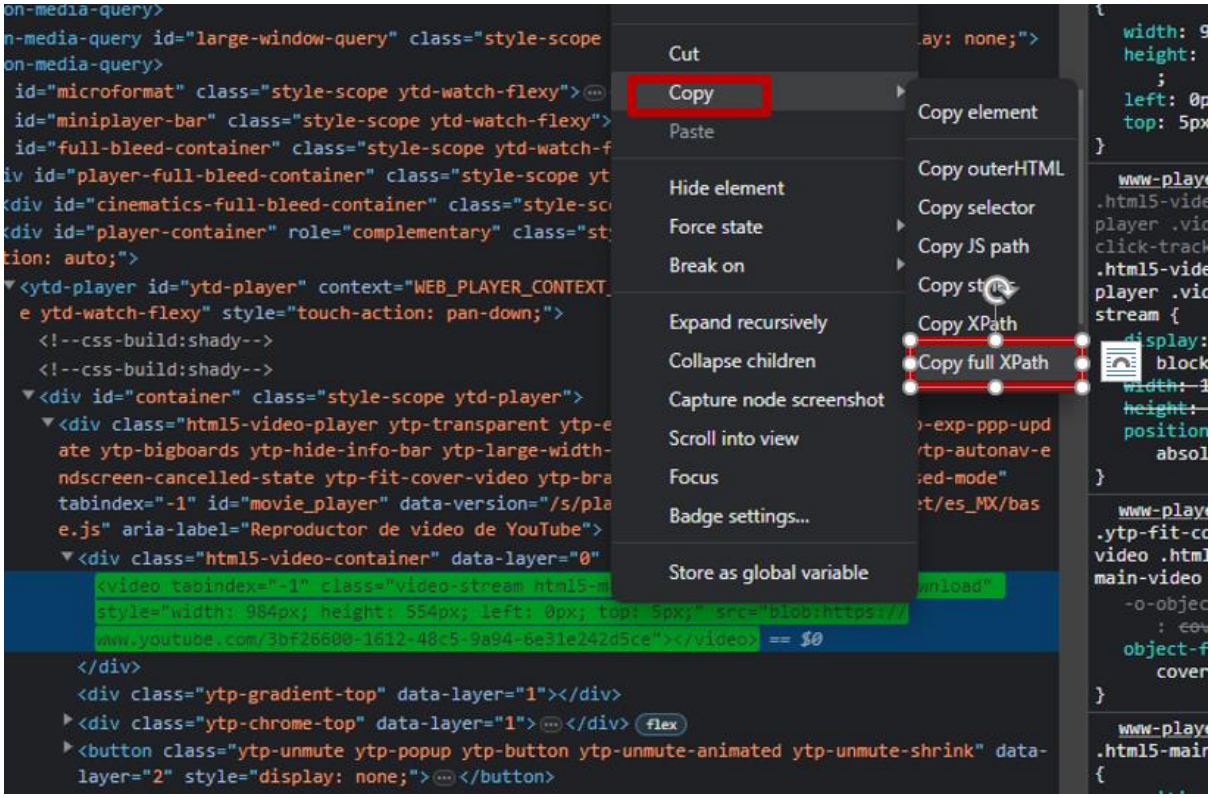
Buscar Localizadores

Para buscar los localizadores de la página hacemos clic derecho, inspeccionar y nos saldrá los códigos de la página. hacemos clic en la parte superior en esta como flechita.

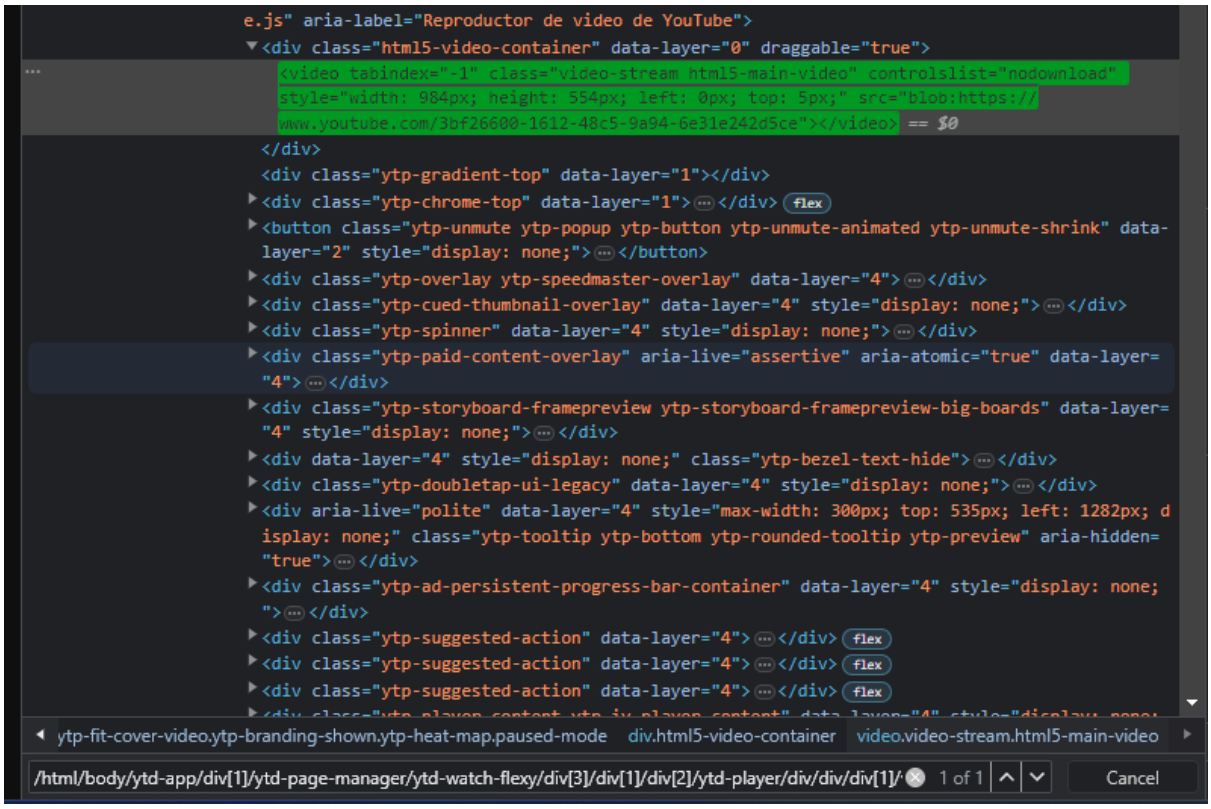


Luego de eso presionamos lo que queremos inspeccionar, en este caso el video





Para comprobar que el Localizador es único presionamos en el teclado “control + f” y colocamos el localizador copiado ahí en la parte inferior que se muestra. nos dirá si es único o hay más elementos iguales. Si es único podemos probar utilizándolo. dependiendo del Localizador que elijamos deberemos usar uno u otro.



Conclusión:

En resumen, el manual proporciona una guía detallada para configurar y realizar pruebas de automatización de Selenium con Cucumber en Java. A continuación, se presenta una breve conclusión de los pasos clave:

Instalación de JDK : El manual comienza con la instalación de la Java Development Kit (JDK) en Windows, que es esencial para ejecutar pruebas de automatización en Java.

Instalación de IntelliJ IDEA: Aquí se describe cómo descargar e instalar IntelliJ IDEA, un entorno de desarrollo integrado (IDE) popular para escribir código Java y automatizar pruebas.

Configuración de Proyecto: Se explica cómo configurar un proyecto Maven en IntelliJ IDEA y se proporcionan las dependencias necesarias en el archivo "pom.xml".

Configuración de localizadores: Se muestra cómo identificar y utilizar localizadores (selectores) para interactuar con elementos de la página web, incluyendo ejemplos de localizadores comunes.

Creación de Escenarios BDD: Se describe cómo crear escenarios de pruebas en lenguaje natural utilizando la notación Gherkin, que incluye palabras clave como "Given", "When" y "Then".

Configuración de Chromedriver: Se explica cómo descargar y configurar el ChromeDriver para la automatización de pruebas en el navegador Google Chrome.

Ejecución de Pruebas: Se muestra cómo ejecutar pruebas de automatización en IntelliJ IDEA utilizando el runner de Cucumber.

Verificación de resultados: Se proporciona un ejemplo de cómo verificar los resultados de las pruebas y se cierra el navegador al final de la prueba.

Localizadores: Se enumeran una variedad de métodos de localización de elementos en una página web, incluyendo "id", "name", "className", "cssSelector", "linkText", "partialLinkText", "tagName" y "xpath. "

El manual es una guía útil para configurar un entorno de pruebas de automatización en Java utilizando Selenium y Cucumber en IntelliJ IDEA. Proporciona instrucciones detalladas y ejemplos que pueden ayudar a los usuarios a comenzar con la automatización de pruebas web.