



# Selenium

# **Documentación Del Conocimiento Adquirido De Selenium**

Autor: Ronny De Los Santos Peralta

Fecha: 15 / 10 / 2024

## Contenido

Introducción .....	4
Lenguaje utilizado .....	5
Manejador de dependencias.....	5
WebDriver.....	6
Interacción con Selenium .....	7
- <i>Ruta driver</i> .....	7
- <i>Crear sesión</i> .....	7
- <i>Navegando</i> .....	8
- <i>Maximizar ventana</i> .....	8
- <i>Selectores</i> .....	8
- <i>Seleccionando elemento</i> .....	10
- <i>Acción sobre el elemento</i> .....	11
➤ Introducir valor a una caja de texto.....	11
➤ Hacer click .....	12
➤ Obtener texto .....	13
➤ Tiempos .....	14
Referencias .....	16

## Introducción

En el presente documento estaré mostrando detalladamente mis conocimientos adquiridos sobre Selenium un entorno de pruebas que se utiliza para comprobar si el software que se está desarrollando funciona correctamente. Esta herramienta permite: grabar, editar y depurar casos de pruebas que se pueden automatizar.

Estos conocimientos fueron adquiridos través del curso “Curso de Selenium WebDriver Gratuito”, en la última página encontrara la referencia al curso.

## Lenguaje utilizado

Es necesario anticipar que Java fue el lenguaje de programación que utilicé para hacer los programas de automatización utilizando Selenium para establecer la comunicación e interacción con el navegador web Chrome.

### IDE (Entorno de Desarrollo Integrado)

Java hasta la actualidad ha sido utilizado en tres IDE:

- NetBeans
- Eclipse
- IntelliJ IDEA

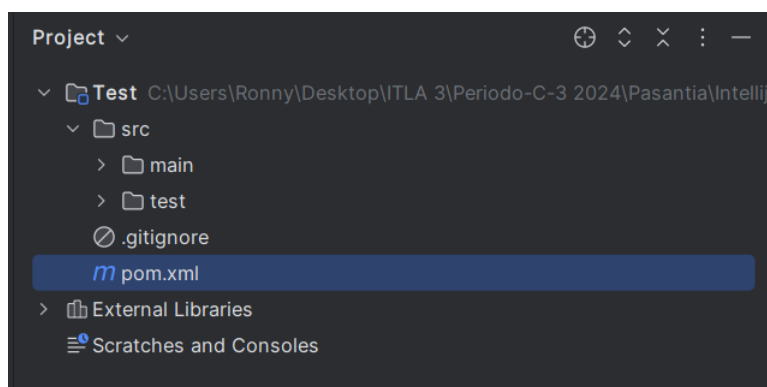
El IDE que yo utilice fue IntelliJ IDEA.

## Manejador de dependencias

Java utiliza Maven y Gradle para manejar las dependencias. En mis pruebas con Selenium yo trabaje directamente con Maven.

Maven es un gestor de dependencias (o manejador de librerías) para proyectos Java, además de manejar dependencia, es una herramienta de automatización de proyectos que también ayuda a gestionar el ciclo de vida del desarrollo de software.

En la carpeta raíz del proyecto encontramos un archivo llamado pom.xml perteneciente a Maven, en este archivo es donde se declaran las dependencias y Maven las descarga automáticamente.



```
m pom.xml (Test) x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
5   <modelVersion>4.0.0</modelVersion>
6
7   <groupId>org.example</groupId>
8   <artifactId>Test</artifactId>
9   <version>1.0-SNAPSHOT</version>
10
11   <properties>
12     <maven.compiler.source>21</maven.compiler.source>
13     <maven.compiler.target>21</maven.compiler.target>
14     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
15   </properties>
16
17   <dependencies>
18
19     <!-- Dependecia de Selenium -->
20     <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
21     <dependency>
22       <groupId>org.seleniumhq.selenium</groupId>
23       <artifactId>selenium-java</artifactId>
24       <version>4.25.0</version>
25     </dependency>
26
27   </dependencies>
28
29 </project>
30
31
```

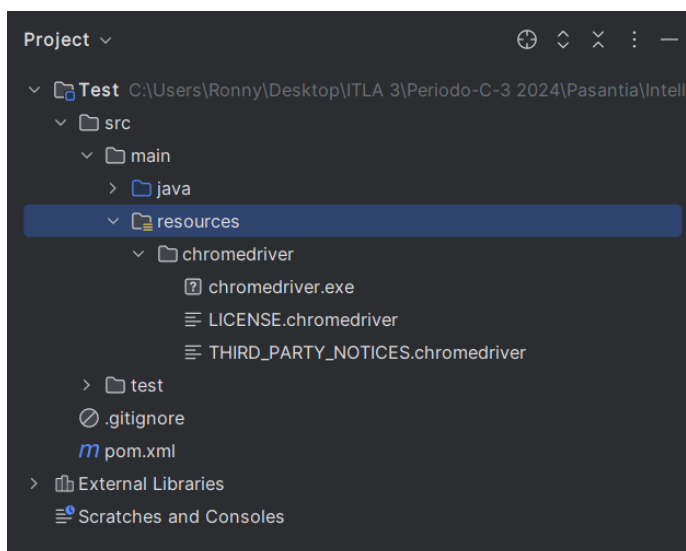
Dependecia de Selenium

## WebDriver

Para que Selenium pueda comunicarse con un navegador, es necesario descargar el driver correspondiente para ese navegador.

En mi caso yo trabajé Google Chrome.

El driver se coloca en la carpeta *resources*.



## Interacción con Selenium

### - **Ruta driver**

Cuando vamos a trabajar con Selenium, debemos agregar la ruta del driver en las propiedades del sistema de Java.

```
// Agregar ruta del driver a la propiedad del sistema java  
System.setProperty("webdriver.chrome.driver", "src\\main\\resources\\chromedriver");
```

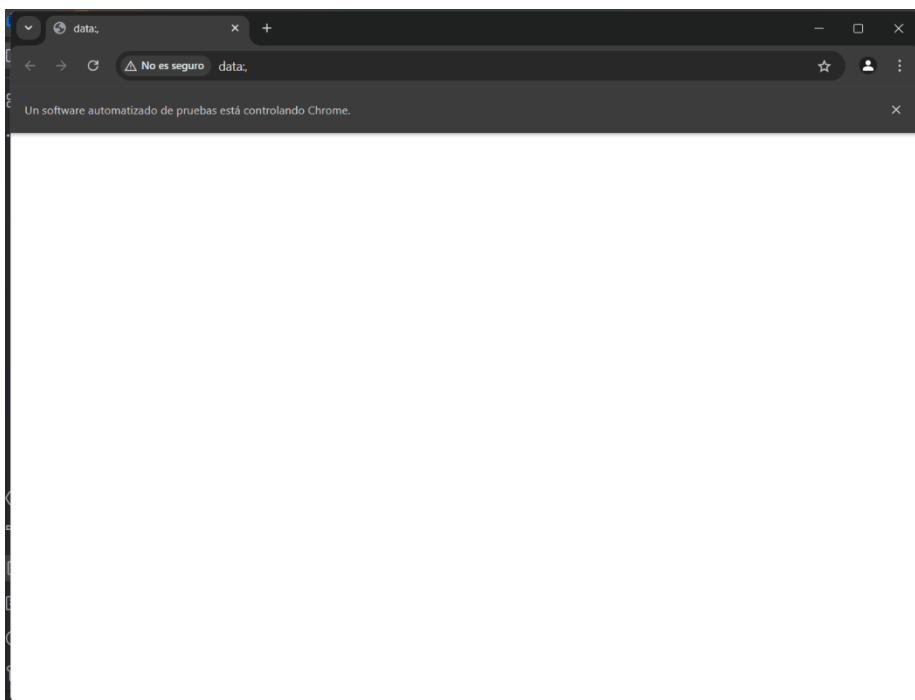
**Nombre con el que Selenium identifica la ruta**

### - **Crear sesión**

Para crear una sesión de navegador, se utiliza una variable de tipo WebDriver y se crea una instancia de ChromeDriver.

```
// Crear una sesión del navegador  
WebDriver driver = new ChromeDriver();
```

Al crear una sesión y correr el programa de abre el navegador de la siguiente manera.



## - **Navegando**

Para navegar a una página web, utilizamos el método `get()` que proporciona la interfaz `WebDriver`.

Con el objeto creado anteriormente que contiene la sesión, llamamos el método `get()` y le pasamos la url.

```
// Navegar
driver.get("https://www.google.com/");
```

## - **Maximizar ventana**

Podemos maximizar la ventana del navegador de la siguiente manera.

```
// Maximizar ventana
driver.manage().window().maximize();
```

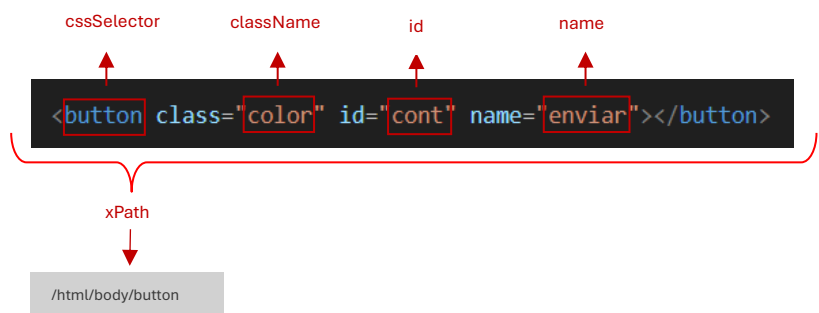
## - **Selectores**

En Selenium, utilizamos selectores css y xPath para ubicar los elementos de la página con los que queremos interactuar.

Entre otros tenemos los más usados y los que yo utilizo:

- id
- name
- className
- cssSelector
- xPath

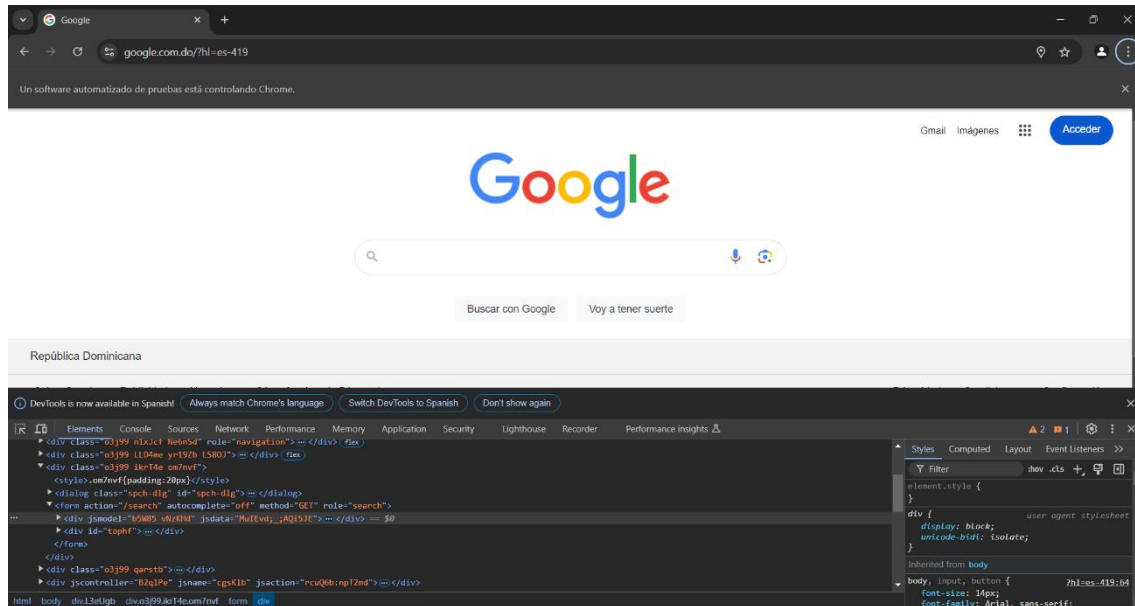
### Ejemplos de los selectores




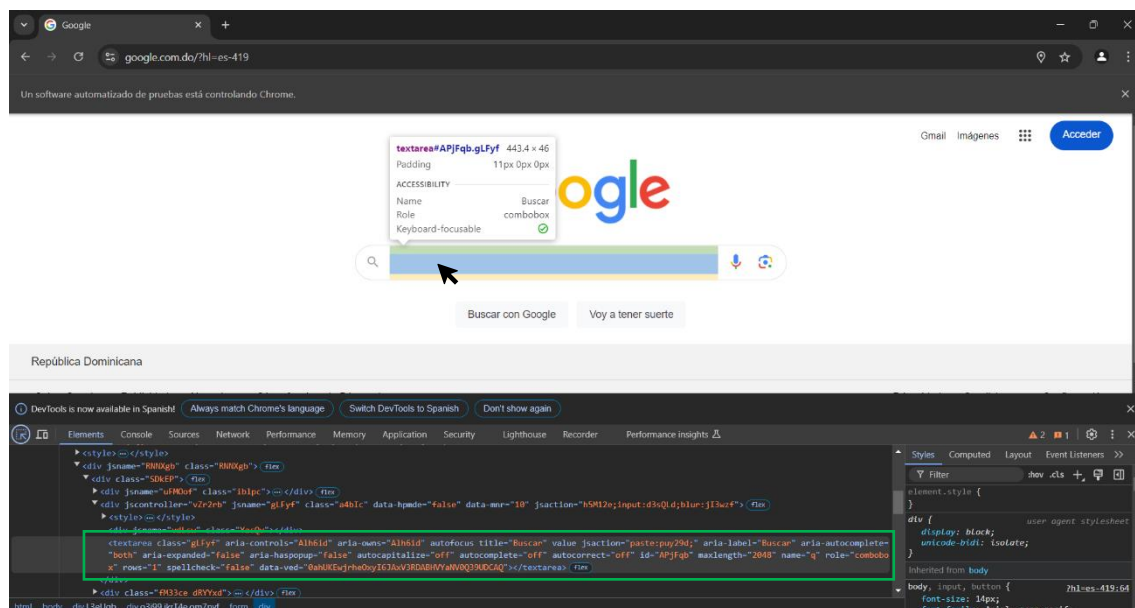


Los navegadores nos brindan una manera más fácil de obtener los selectores que tener que escribirlos a mano para poder seleccionar el elemento del DOM.

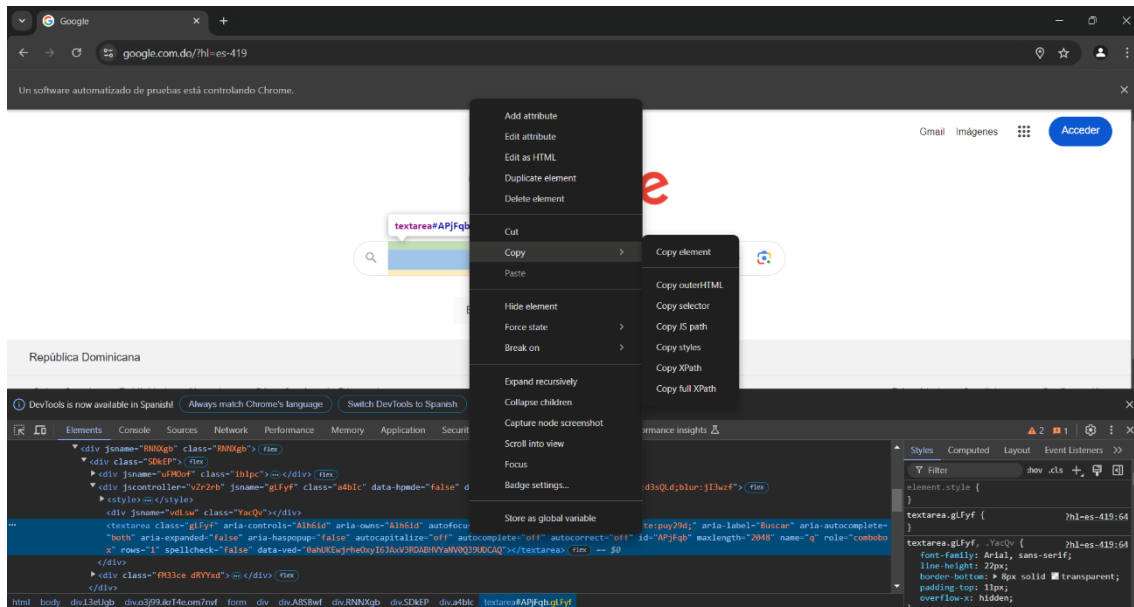
Al dar click derecho he inspeccionar se abre la herramienta de desarrollo:



Al darle click en  podemos seleccionar el elemento que queremos, en este caso el input de navegación:



Luego podemos hacer click derecho sobre el elemento en la herramienta de desarrollo, presionar copiar y luego elegir el sector de preferencia, en mi caso XPath:



**xPath:** `//*[@id="APjFqb"]`

## - **Seleccionando elemento**

Para seleccionar un elemento del DOM vemos utilizar una variable del tipo WebElement para almacenar el elemento en tiempo real, este es el código:



## - ***Acción sobre el elemento***

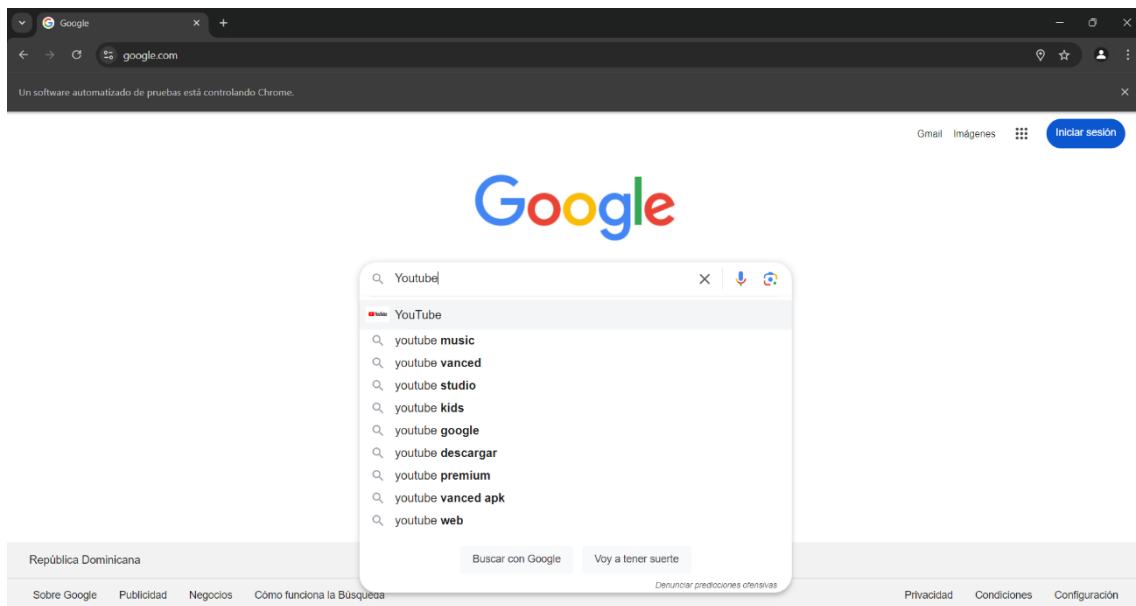
Podemos hacer diferentes tipos de acción sobre el elemento, a continuación, muestro las que utilizo más:

### ➤ Introducir valor a una caja de texto

Con la función `sendKeys()` podemos pasar texto al DOM:

```
// Elemento del DOM
WebElement element = driver.findElement(By.xpath(xpathExpression: "//*[@id=\"APjFqb\"]"));
element.sendKeys(...keysToSend: "Youtube");
```

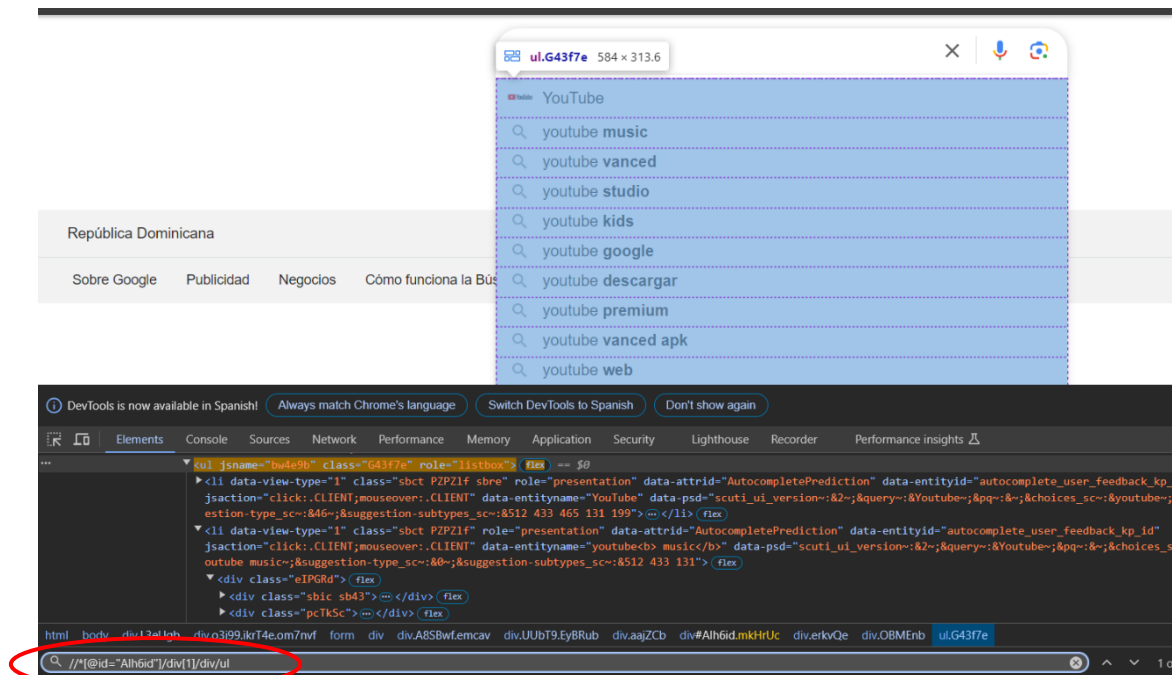
Al ejecutar:



Al introducir el texto me aparecerán sugerencias de búsquedas, en este caso podría seleccionarlas todas con el método `findElements()`:

```
List<WebElement> elements = driver.findElements(By.xpath(xpathExpression: "//*[@id=\"Ah6id\"]/div[1]/div/ul/li[4]"));
```

Variable que almacena una lista de elementos

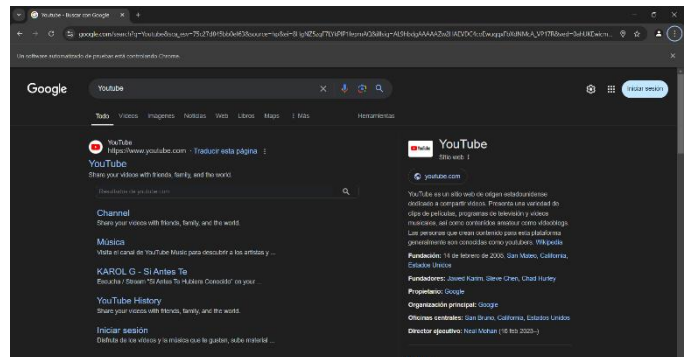
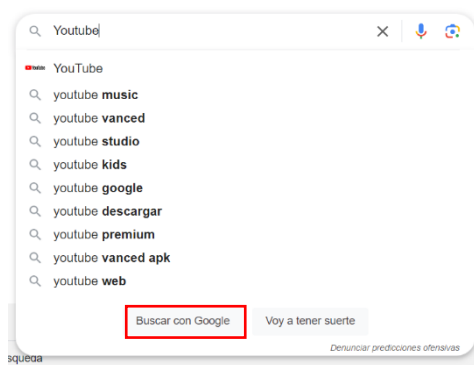


## ➤ Hacer click

Podemos seleccionar un elemento y hacer click con el función click():

```
// Click en el botón
WebElement elementClick = driver.findElement(By.xpath( "//*[@id="Alh6id"]/div[1]/div/ul"));
elementClick.click();
```

En estas líneas de código hago click en el botón.



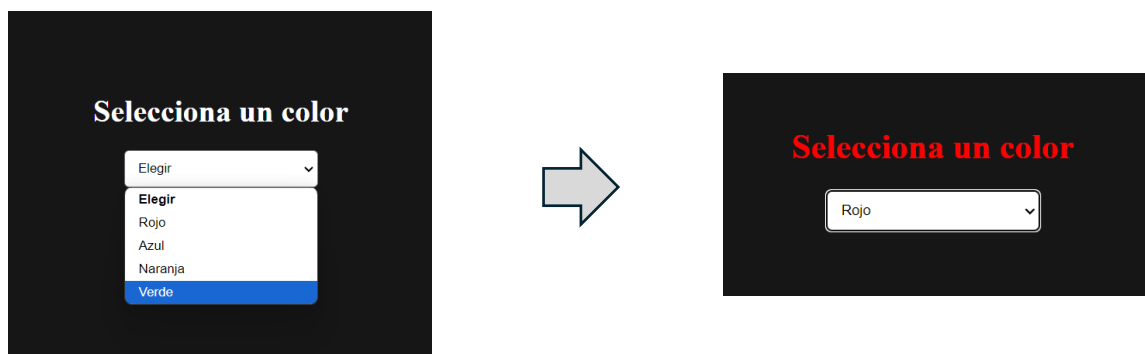
## ➤ Obtener texto

Podemos obtener el texto de un elemento con la función `getText()`:

```
// Obtener texto
String texto = element.getText();
```

Un ejemplo más práctico donde podemos usar esta función, es cuando queremos seleccionar una opción dinámicamente.

En la siguiente imagen se muestra una etiqueta `select` con varias opciones, cuando se seleccione un color el texto se pondrá de ese mismo color seleccionado.



Aquí te muestro el código para este ejercicio:

```
1 driver.get("http://127.0.0.1:5500/index.html");

// Hacer click en la etiqueta select
2 WebElement multiselect = driver.findElement(By.xpath(xpathExpression: "//*[@id=\"colorSelector\"]"));
3 multiselect.click();

// Lista de los resultados
4 List<WebElement> listaOpcion = driver.findElements(By.xpath(xpathExpression: "//*[@id=\"colorSelector\"]/option"));

5 for (int i = 0; i < listaOpcion.size(); i++){
6     if (listaOpcion.get(i).getText().toLowerCase().equals("naranja")){

// Hacer click en la opción
7         listaOpcion.get(i).click();

// Imprimir texto en consola
8         System.out.println(listaOpcion.get(i).getText());
    }
}
```

- En la línea 4 selecciona todas las opciones.
- En la línea 5 recorre la lista.
- En la línea 6 con la función `getText()` trae el texto del elemento índice `i` y lo convierte a minúscula para luego compara con `naranja`.
- En la línea 7 si entra le da click a la opción correspondiente.
- En la línea 8 imprime en consola el texto que tiene el elemento en el índice `i`.

## ➤ Tiempos

Muchas veces, mediante el código seleccionamos un elemento, pero según el tiempo que tarde la pagina en cargar o en aparecer el elemento puede haber una excepción.

Existen tres tipos de tiempos, ***Implicit Wait*** (Espera Implícita), ***Explicit Wait*** (Espera Explícita) y ***Fluent Wait*** (Espera Fluida).

Hasta ahora domino el Implicit Wait.

### Implicit Wait:

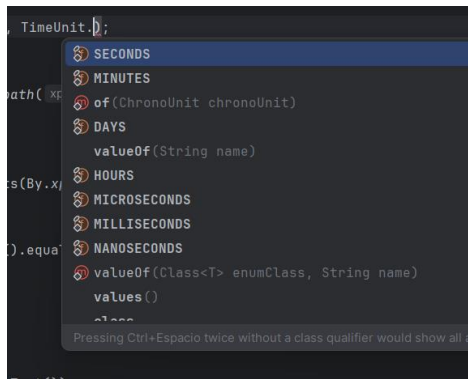
Es una configuración en la que se le indica al driver que espere un tiempo determinado antes de lanzar una excepción si no encuentra un elemento en la página.

Codigo:

```
driver.manage().timeouts().implicitlyWait( time: 3, TimeUnit.SECONDS);
```

`3` indica el tiempo de espera y `TimeUnit.SECONDS` la medida del tiempo, en este caso, segundos.

Otras opciones:



## Referencias

*YouTube*. (s.f.). Obtenido de <https://youtube.com/playlist?list=PLLi9hZ7Ces--QbPWeiBKRBzEOT2ta6Ffg&si=PW4DhF8AysB4CuyR>