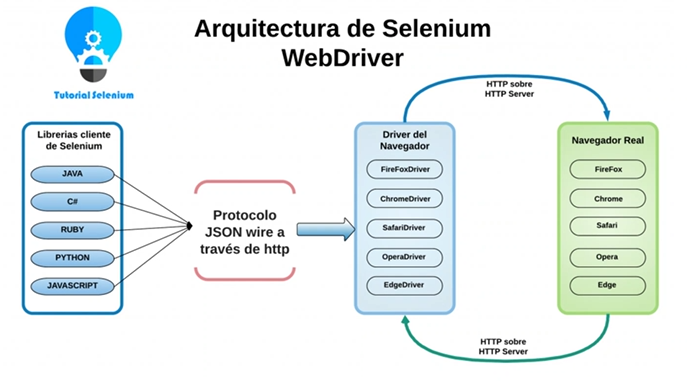
## Instalación IDE



La página web tiene **WebElements**

Al ser WebElements nosotros podemos ejecutar acciones sobre estos elementos

POM - **Page Object Model**

Forma de organizar nuestro proyecto

**Locators**

Se puede buscar por id (lo más recomendable, si tiene ID significa que es un elemento único)

Como segunda alternativa podríamos buscar por name

**Instalación Visual Studio Code y cosas varias para que funcione el entorno**

code.visualstudio.com

Darle todo siguiente, siguiente, siguiente

Se instalan extensiones yendo a la solapa de Extensiones:

*Extensión C#*

*Nuget Package Manager*

File 🡪 Open Folder 🡪 New Folder y la abrimos

Nos tiene que aparecer el nombre de nuestra carpeta y vacía

https://dotnet.microsoft.com/en-us/download/dotnet/6.0

descargar e instalar esto (SDK .NET de Windows)

Crear proyecto para utilizar C# y Selenium  
  
Ctrl+Ñ 🡪 se abre la consola

Vamos a crear nuestro proyecto ingresando este comando en la consola

Dotnet new nunit

*The template "NUnit 3 Test Project" was created successfully.*

*Processing post-creation actions...*

*Running 'dotnet restore' on C:\Users\fernando.e.nieto\Desktop\Acc\Selemiun C++\Training Automation Basic\Training Automation Basic.csproj...*

*Determining projects to restore...*

*Restored C:\Users\fernando.e.nieto\Desktop\Acc\Selemiun C++\Training Automation Basic\Training Automation Basic.csproj (in 25.5 sec).*

*Restore succeeded.*

Instalar Selenium

View🡪 Command pallet  
nuget package add package

Selenium webdriver

4.3.0

dotnet restore en la terminal siempre después de instalar cosas

Vamos de nuevo a view🡪 comnand pallet  
package

Webdriver manager

2.14.1

En .csproj se va viendo lo que vamos instalando

View command pallet  
testplatform.ObjectModel

17.2.0

Luego de instalaciones

Archivo UnitTest1.cs

Tiene Etiqueta SetUp

Se va a definir la inicialización del WebDriverManager

Es un Driver que actúa sobre el Chrome

[SetUp]

public void Setup()

{

new DriverManager().SetUpDriver(new ChromeConfig());

}

Y otra que se llama Test

Si a una clase que creamos le agregamos la etiqueta Test se va a tratar como un test

Podemos tener N tests con la etiqueta Test

Posicionándonos sobre un método que no esté en verde

Ctrl+.  
y agregamos el método si no lo reconoce

## Advanced

### Clase 1 - Selectores o locators - xPath

Selectores flaky – no son buenos selectores, estoy ubicándolos de manera no eficiente

XML Path Language – Lenguaje para acceder al DOM

\* DOM es una abreviatura de **Document Object Model**. En español podríamos traducirlo por Modelo de Objeto de Documento, nos hemos referido al DOM habitualmente con el nombre de jerarquía de objetos del navegador, porque realmente es una estructura jerárquica donde existen varios objetos y unos dependen de otros.

//div[contains(text(),”parte del texto’”)] 🡪 esto es mejor que utilizar sólo el text(), xq sino por un carácter o espacio por ahí no encuentra el elemento

//div[@id=”sarasa”]/child::\* 🡪 ahí me traería todos los hijos de ese div sarasa

//div[@id=”sarasa”]/child::div/child::div[@id=”sarasa2”]

Y con parent en vez de child, va para arriba y se puede combinar como el ejemplo de arriba

### Clase 2

Xpath

Css selector es muy útil y rápido para id o clases, solamente jerarquía de igual o para abajo

Como si puede hacer el xpath

~

Pagina de ayuda para CSS Selector

CSS Selectors Cheatsheet - FrontEnd30

[CSS Selectors Cheatsheet - FrontEnd30](https://frontend30.com/css-selectors-cheatsheet/)

Pagina de ayuda para XPath

Xpath cheatsheet (devhints.io)

[Xpath cheatsheet (devhints.io)](https://devhints.io/xpath)

### Clase 3

[B36183-ARG-FY22-08.22.2022 (microsoft.com)](https://forms.microsoft.com/Pages/ResponsePage.aspx?id=OT154DkJbUmxKRmO3ZFv6-snmf1l9QxCtVa9PiTP2V1UME44WkRLOU81SzNQUU5SWFBNOUNMTE1PRyQlQCN0PWcu)

[Selenium Training - Advanced - Accenture - Media Exchange](https://mediaexchange.accenture.com/playlist/dedicated/1_78a5ok6a/1_9ma3ro90)

### Estructura POM de mi proyecto de pruebas

Creamos carpeta Core (va a contener las bases)

Creamos carpeta Pages (métodos de páginas y selectores)

Creamos carpeta Tests (conteniendo las pruebas propiamente dichas)

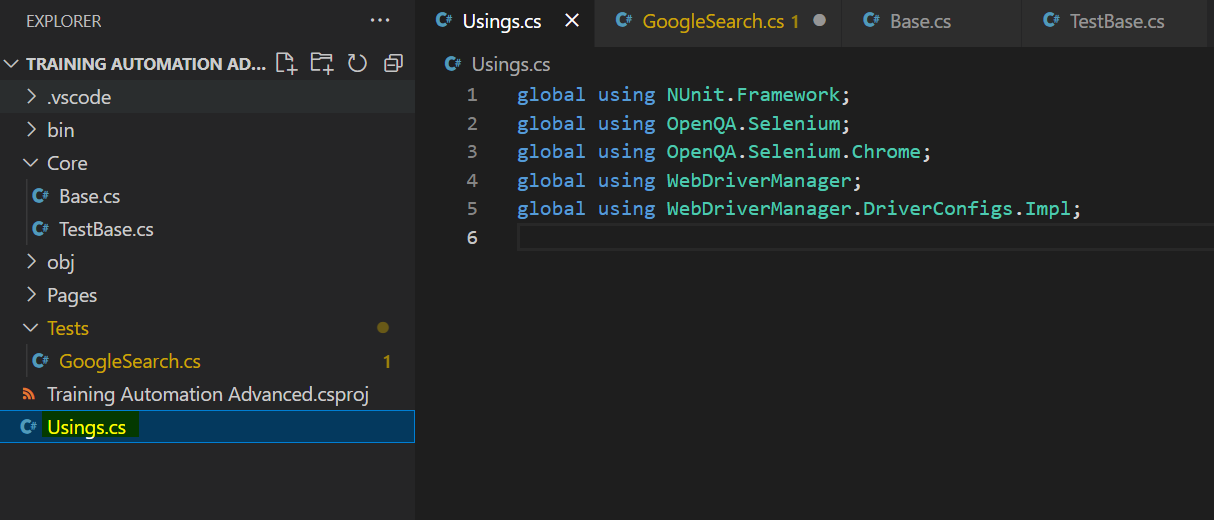
Creamos clases Base.cs y TestBase.cs dentro de la carpeta Core

Base 🡪 Funcionalidades de Selenium y driver, o customizar métodos de Selenium

Usings en archivo aparte Usings.cs,

Todos los usings que se vayan necesitando se van declarando como global en este archivo

Archive que va a contener las dependencias en modo global de esta manera



En carpeta TestBase también se evita utilizar el driver (IMPORTANTE y es donde se van a crear e instanciar los objetos pagina), en Base solamente es en la clase en donde se debería poder interactuar con el driver

Dentro de la estructura Pages, se crean más carpetas (una por cada página a testear)

Y dentro de cada carpeta de cada Página, se crean 3 archivos

[CualquierNombre1].Map.cs 🡪 Selectores de una página específica con devolución de IWebElement

[CualquierNombre1].Asserter.cs 🡪 se utilizan los asserts que utilizan los elentos localizados en Map, que a su vez estos asserts se van a utilizar en la clase siguiente

[CualquierNombre1].cs 🡪 Se implentan las acciones que vamos a hacer en la página

Por ejemplo primer método que se me ocurre de cualquier página, ir a esa página

Public void GoToGoogle(){} //por ejemplo el primer método, si fuera para la pagina Home de Google.