**Automatización de Testing**

**Objetivo**: el objetivo de este documento radica en mostrar los pasos a seguir para realizar la automatización del Testing en el Proyecto Municipio VPSA.

**Pasos a seguir**

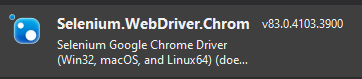
Para las pruebas de Testing Automatizado se utilizará la herramienta Selenium Web Driver, que es bastante versátil y tiene soporte para visual Studio 2019.

El test se realizará sobre el Frontend. Se realizará sobre el login de usuario Relacionado con la US MA-66

Las pruebas de Backend se realizan de manera manual, no automatizada por falta de tiempo.

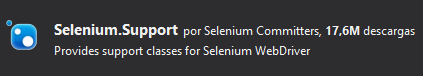
Pasos para el desarrollo del test:

1. Debe crearse un proyecto de consola. Se utilizó el IDE Visual Studio 2019.
2. Buscar el paquete en el gestor Nuget de Microsoft y se procede a instalar los siguientes paquetes uno para el navegador de preferido. Se utilizó Chrome pero puede utilizarle internet explorer , Firefox , opera.



Selenium WEB driver with Chrome

1. Una vez realizada la instalación en el proyecto, se agrega el de soporte para poder automatizar elementos como por ejemplo combos.



Recomendación de desarrollo, si se utiliza este entorno estático generalmente se utiliza el “Id” de cada elemento del sitio que requiera testing, botones, checkbox, textbox, etc.

En caso de cambios de nombres o de elementos en el front, deben reconfigurarse los nombres de la variables a utilizar. Para evitar esto hay una alternativa de uso del mismo framework Page object, que detallaremos posteriormente.

Código de prueba realizada, Login y Carga de Trabajo en Denuncia Existente.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

using OpenQA.Selenium.IE;

using OpenQA.Selenium.Interactions;

using OpenQA.Selenium.Support.UI;

namespace SELENIUMTEST

{

class Program

{

static void Main(string[] args)

{

IWebDriver webDriver = new ChromeDriver();

webDriver.Navigate().GoToUrl("https://vpsamvc.azurewebsites.net");

webDriver.Navigate().GoToUrl("https://vpsamvc.azurewebsites.net/Account/Login");

//login insert user

webDriver.FindElement(By.Id("Legajo")).SendKeys("202003");

//login insert Pass

webDriver.FindElement(By.Id("Password")).SendKeys("Vpsa$11");

//login Click

webDriver.FindElement(By.ClassName("btn")).Submit();

//Carga de Trabajos en Denuncia.

webDriver.Navigate().GoToUrl("https://vpsamvc.azurewebsites.net/Denuncias/Details/1");

//Agregar Trabajo

webDriver.FindElement(By.ClassName("btn-info")).Click();

//Carga de texto de Trabajo

Thread.Sleep(2000);

webDriver.FindElement(By.TagName("textarea")).SendKeys("Testing Automatizado Carga de Trabajo de Prueba");

//Empleado

SelectElement selector = new SelectElement(webDriver.FindElement(By.Name("UserId")));

selector.SelectByIndex(3);

//"Estado Denuncia"

SelectElement selector2 = new SelectElement(webDriver.FindElement(By.Name("EstadoDenunciaId")));

selector2.SelectByIndex(2);

// Guardar "SaveButton"

webDriver.FindElement(By.Id("SaveButton")).Click();

}

}

}

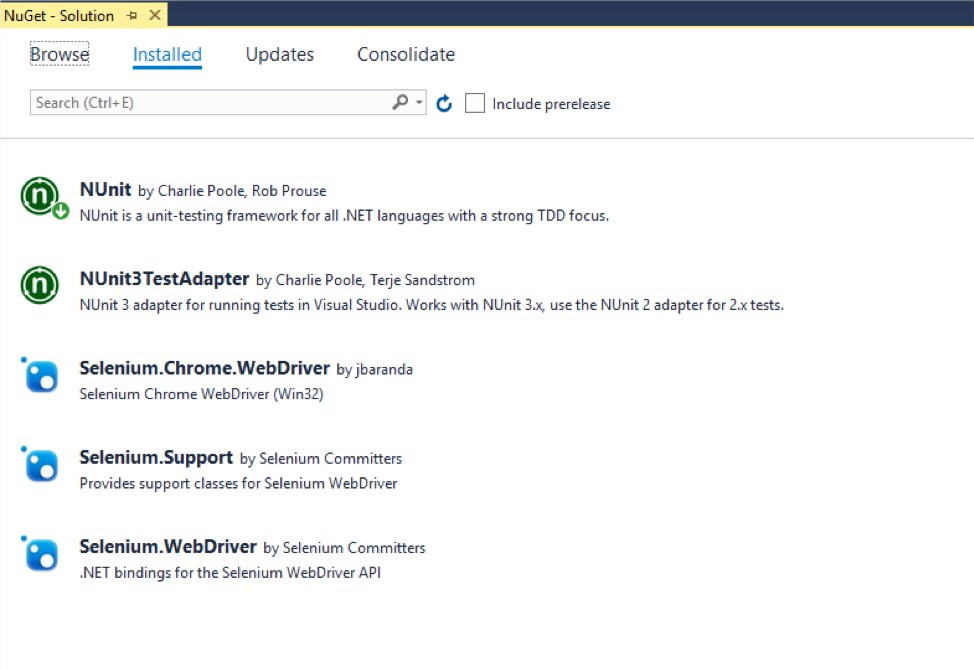
Se recomienda para implementación posterior el uso de Page Object es un patrón de diseño que se utiliza en pruebas automatizadas para evitar código duplicado y mejorar el mantenimiento de las mismas.

Un punto importante a tener en cuenta cuando automatizamos es que la UI (interfaz de usuario) puede sufrir cambios. Si nuestras pruebas manipulan directamente los elementos de la página, serán frágiles y requerirán alto mantenimiento.

Cuando diseñamos una prueba automatizada para un sitio web, debemos consultar sus elementos para hacer clic en enlaces/botones y determinar qué se muestra a continuación. El patrón Page Object encapsula el comportamiento de las páginas, o una parte de ella, con una API específica de la aplicación, lo que nos permite escribir tests y manipular elementos de página sin tener que lidiar con el HTML.

En resumen, el patrón Page Object se usa para hacer pruebas funcionales automatizadas. La idea es modelar las páginas y sus comportamientos para lograr pruebas claras de escribir, entender y mantener.

Como ejemplo, vamos a modelar un escenario sencillo utilizando WebDriver y .NET.



Escenario

* Acceder a la página de Login
* Introducir Username
* Introducir Password
* Presionar Login
* Verificar que el usuario se loguea correctamente

Ejemplo

[Test]

public void LoginPage\_SignsTheUserIn()  
  
{  
  
            var loginPage = new LoginPage();  
  
            loginPage.SignIn("maria", "123456");  
  
            var homePage = new HomePage();  
  
            Assert.IsTrue(homePage.IsCurrent());  
  
}  
  
LoginPage.cs  
  
[Css("input[name='username']")]  
  
private IElement usernameInput;  
  
private IElement passwordInput;  
  
private IElement signButton;  
  
this.usernameInput = Browser.GetElementByCss("");  
  
public void SignIn(string username, string password)  
  
{  
  
            this.usernameInput.SetValue(username);                       
  
            this.passwordInput.SetValue(password);  
  
            this.signInButton.Click();  
  
}  
  
HomePage.cs  
  
public bool IsCurrent()  
  
{  
  
            return this.bannerDiv.IsVisible;  
]