GUIA BDD CON LA HERREMIANTA SPECFLOW OPERACIONES BASICAS MATEMATICAs EN C# Medellín, abril 2020

Por: Johana Catalina Ríos Torres
Ingeniería Sistema UdeA
Cod 41960845
Calidad de Software

HERRAMIENTAS:

- Visual Studio 2019 *
- Guia de installation SpecFlow y SpecFlow+ Runner

* Para evitar incompatibilidad de versiones y errores en la ejecución de los test, se recomienda utilizar la última versión de Visual Studio 2019. Cuando se vaya a generar el archivo del proyecto .specs (click derecho propiedades) se debe revisar que target framework es, se recomienda el .NETCore3.1

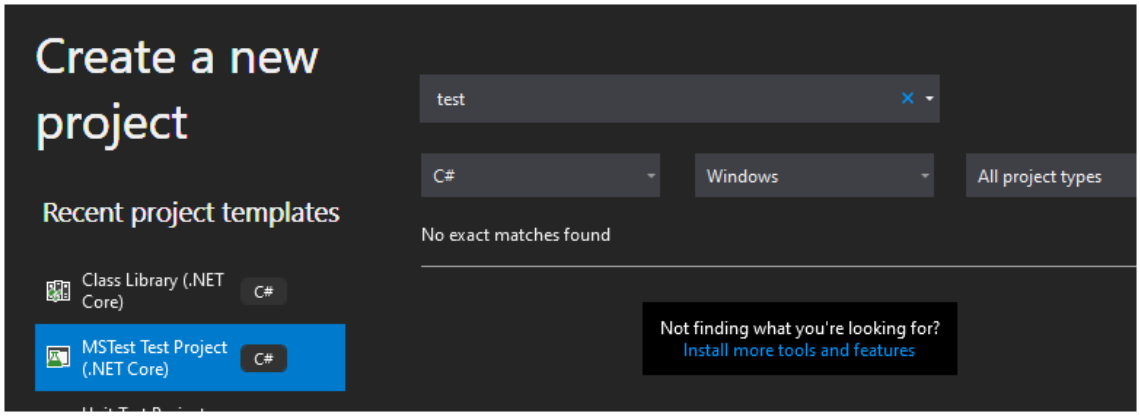
SpecFlow es una herramienta basada en lenguaje Gherkin (Given-When-Then) busca mejorar la especificación de requerimientos, se generan escenarios en palabras de negocio, esto facilita al desarrollador saber cuál es la funcionalidad real en la que se debe trabajar, evitando código que no va ser ejecutado, esta herramienta permite también utilizar la especificación para pasar rápidamente a diseñar los test, de ahí a que en el diseño se genera paralelamente documentación la cual es muy fácil de entender para los "Stakeholders",dado que se realiza en palabras de negocio y no en lenguaje técnico.

En el siguiente ejemplo se va a diseñar una calculadora básica (suma, resta, multiplicación y división.

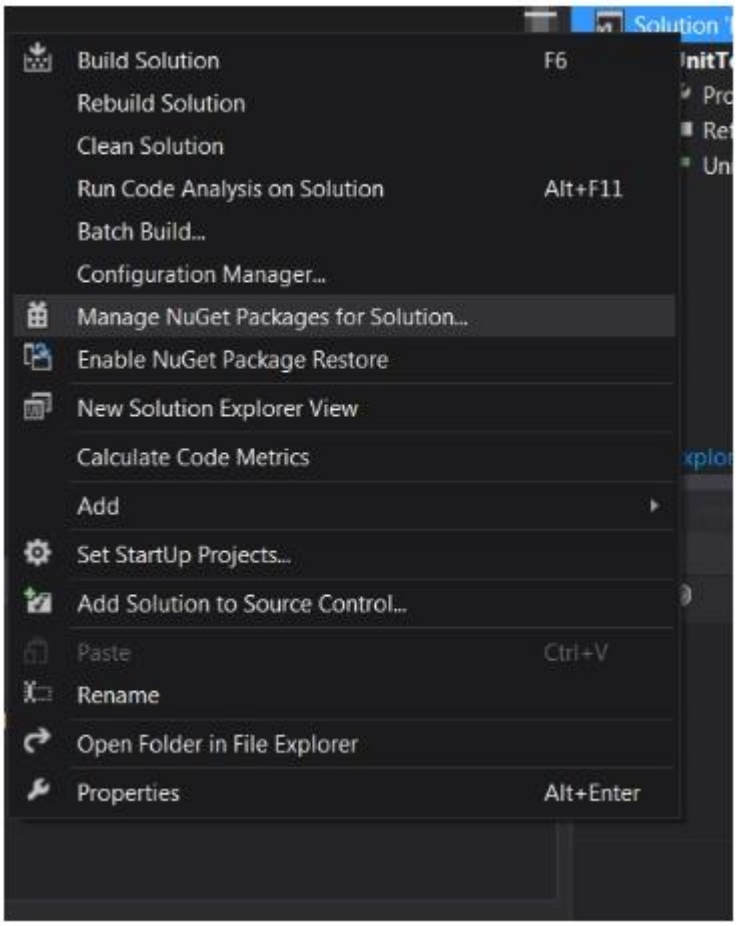
Se tomó como referencia la guia Specflow https://specflow.org/getting-started/#AddingFeature

Vamos a crear un proyecto de pruebas en Visual Studio Comunity 2019:

- *Click en File, New, Project*
- Seleccionamos la opción "MSTest Test Project (.NET Core)" como nombre de Proyecto "MyProject.Specs"

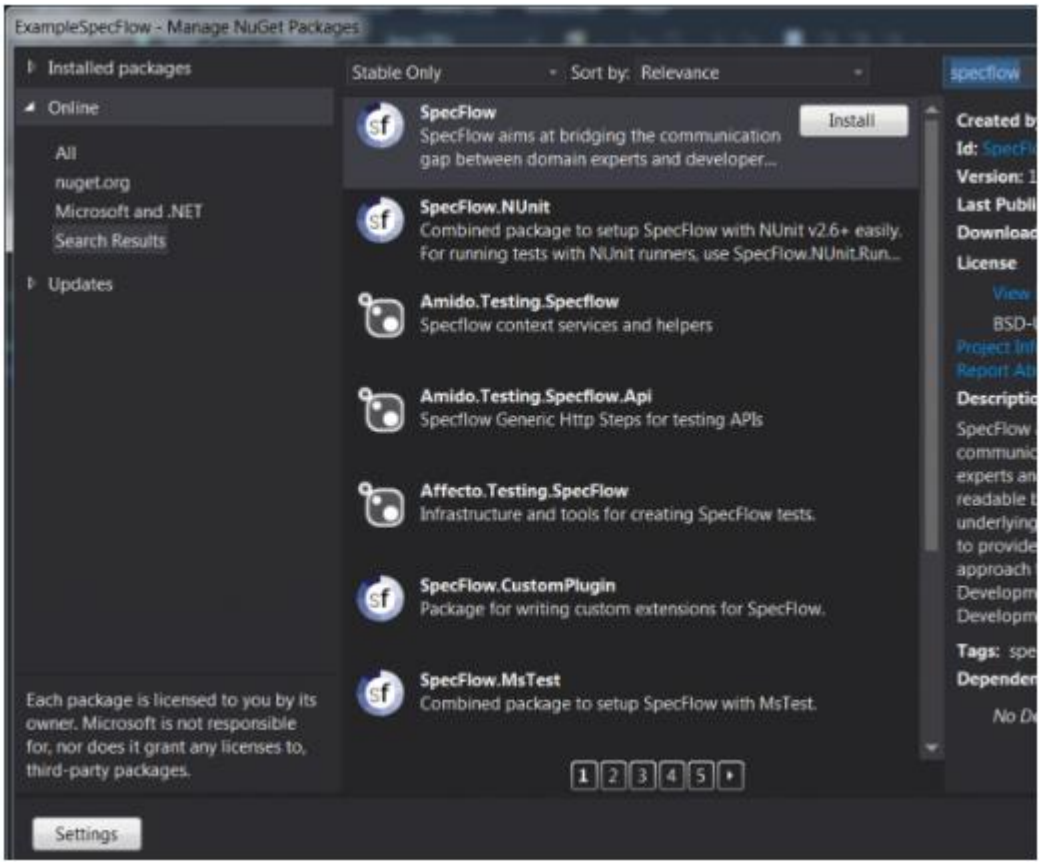


- Eliminamos el archivo *UnitTest1.cs* no es requerido
- clic derecho en la Solución “Myproject’ y elegimos la opcion *ManageNuget Packages*

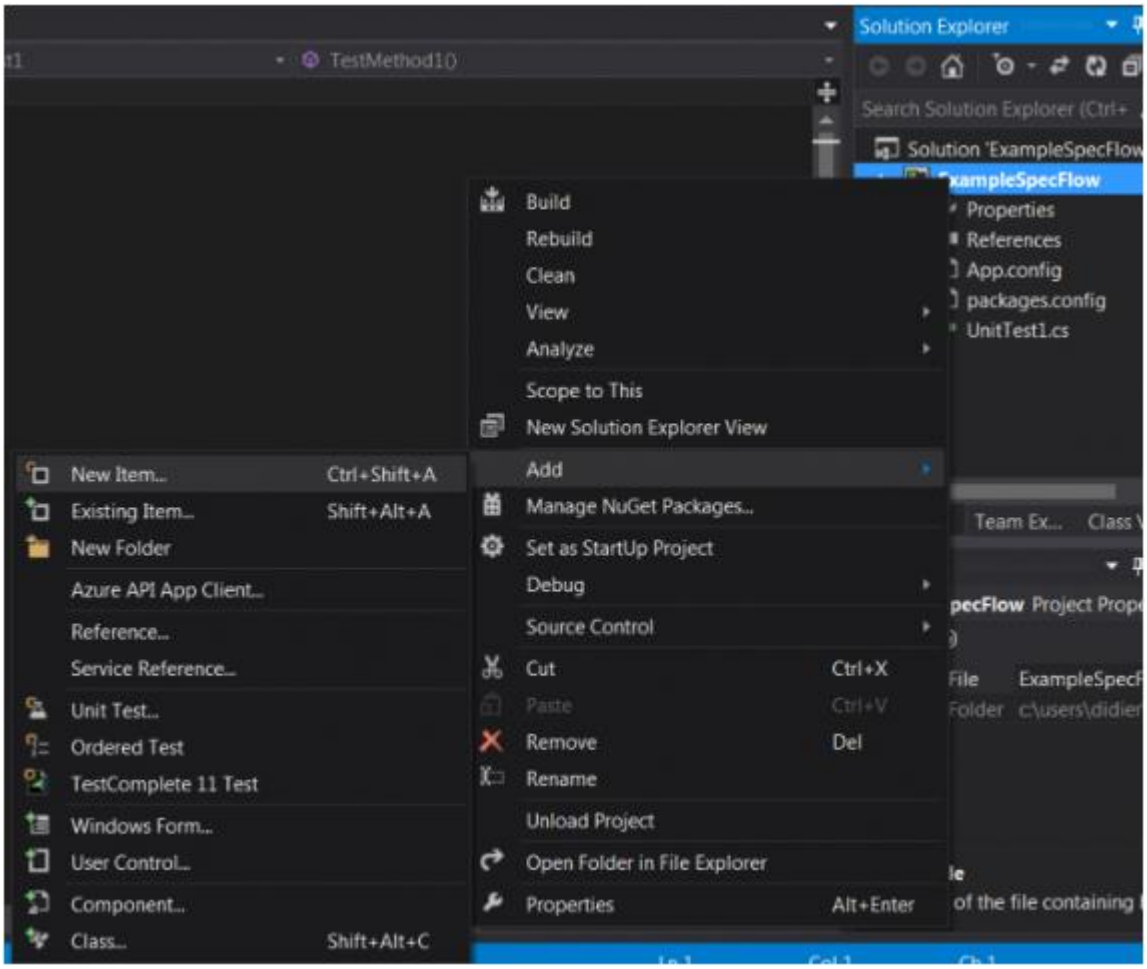


En este punto vamos a Seleccionar los siguientes Packages:

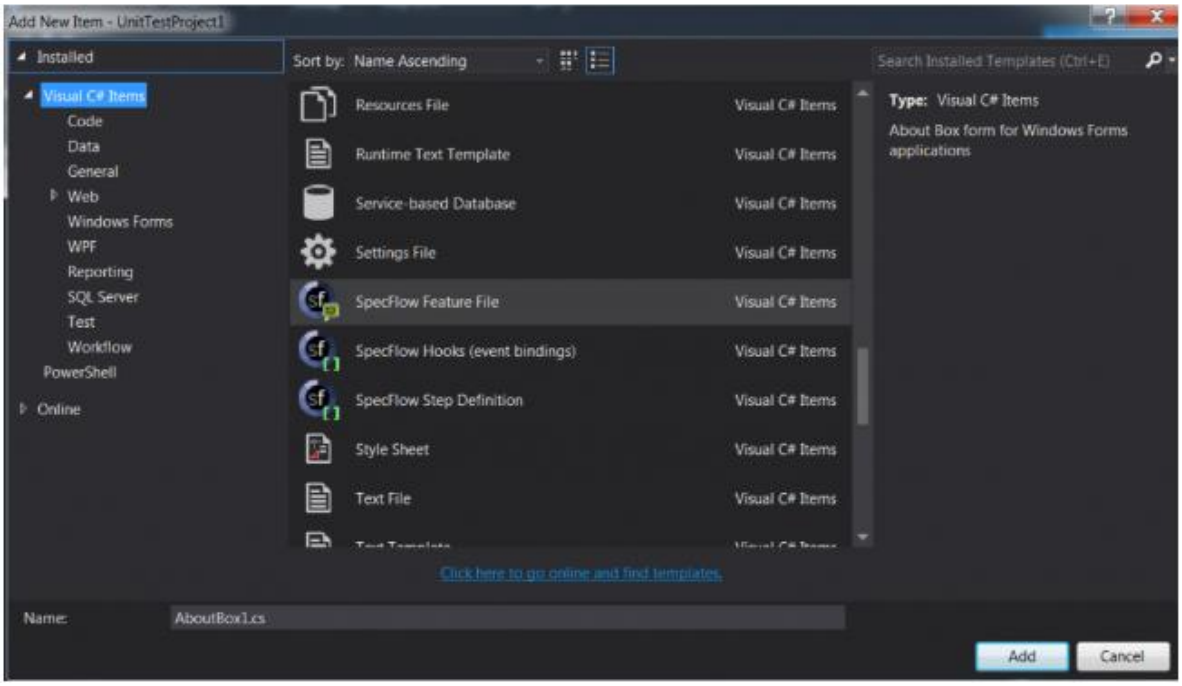
- SpecFlow
- SpecFlow.Tools.MsBuild.Generation
- SpecRun.SpecFlow
- SpecFlow Runner
- Verifica si tienes el paquete de Microsoft .NET Test SDK, sino se descarga.

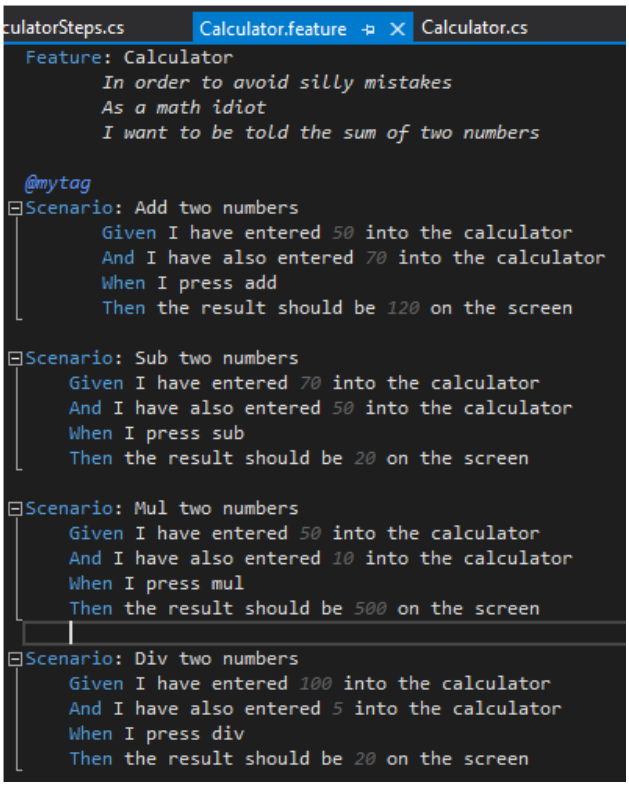


Una vez instalados los paquetes vamos a hacer clic derecho sobre el proyecto de pruebas y seleccionamos la opción *add*, *New Item* y buscamos *SpecFlow Feature File.*



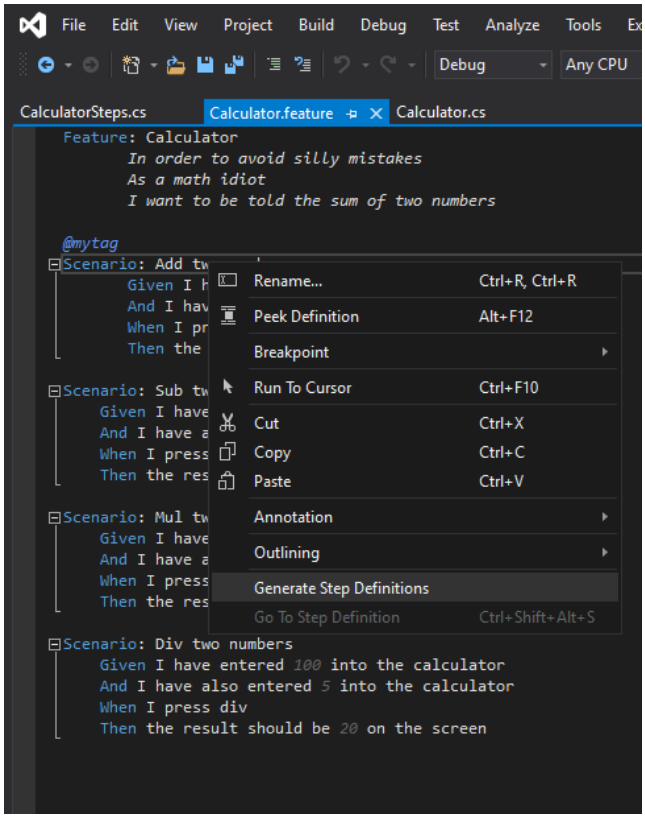
- Se le da el nombre de “Calculator.feature”





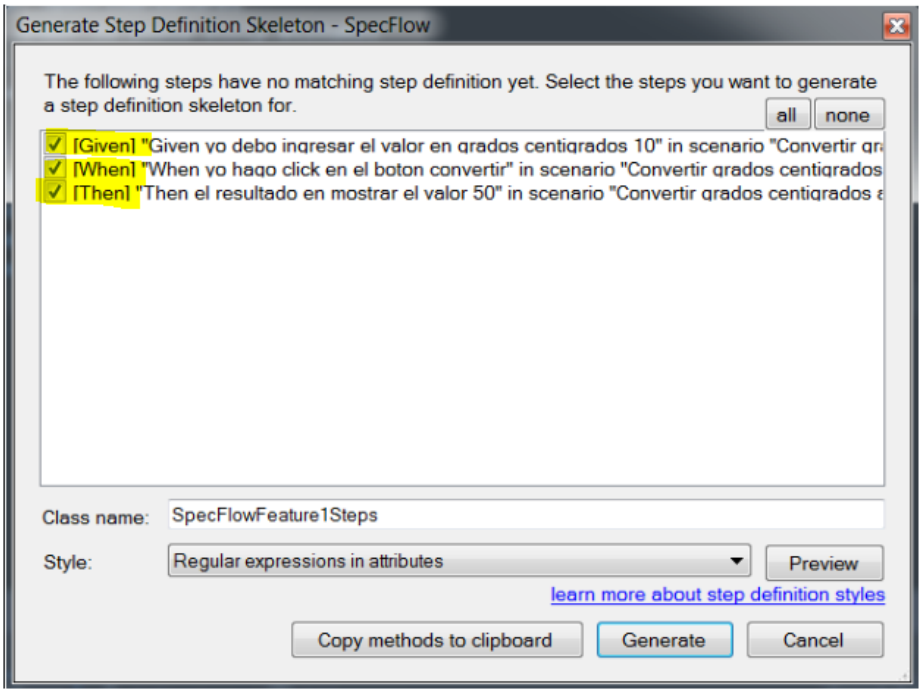
- Se ha generado un archivo como el siguiente, el cual incluye la estructura básica:

- Agregamos la palabra ‘also’ en cada una de los escenarios ejm. “And I have also entered 70 into….” En este archivo feature se agregan a las demás operaciones matemáticas.

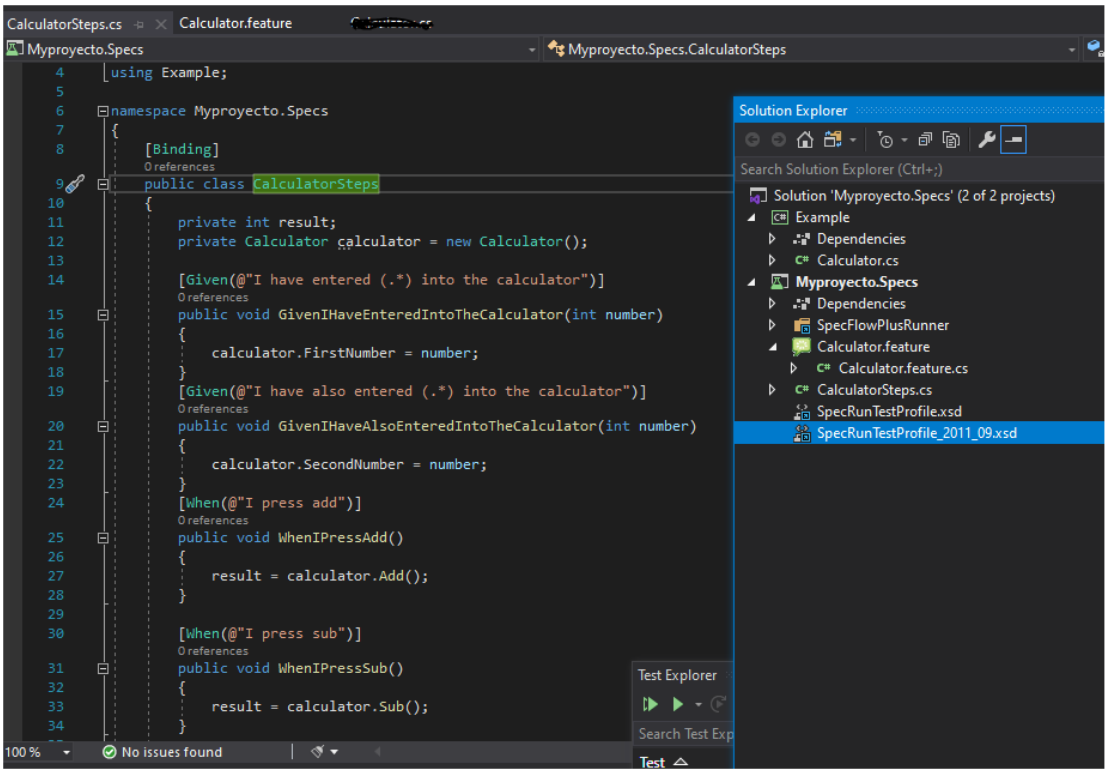


- Vamos a hacer clic derecho sobre los escenarios y elegimos la opción *Generate Step Definitions y damos* el nombre de “CalculatorSteps”.

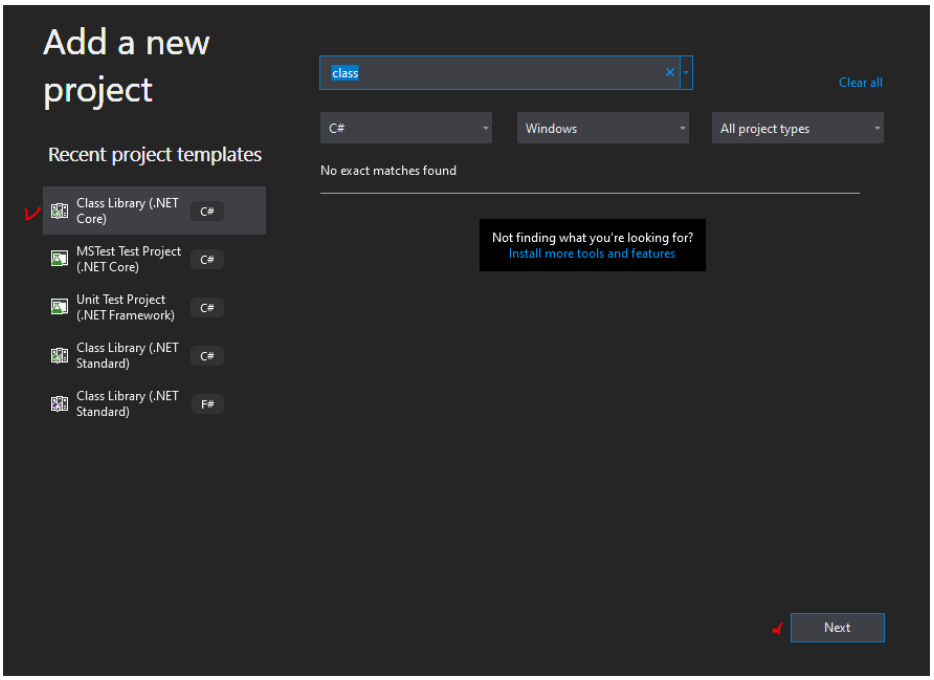
- Revisamos que cada una de las lineas del lenguaje Gerkins este chuleadas de cada escenario.



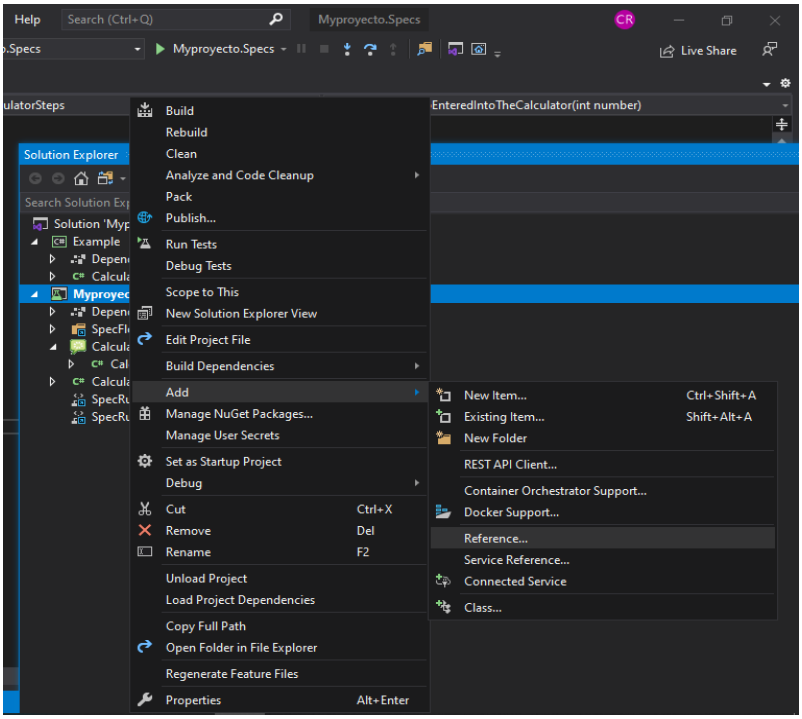
- Se ha generado el archivo de pasos *SpecFlowFeature1Steps* el cual genero automáticamente el código para implementar los pasos que fueron definidos en palabras de negocio o lenguaje natural.



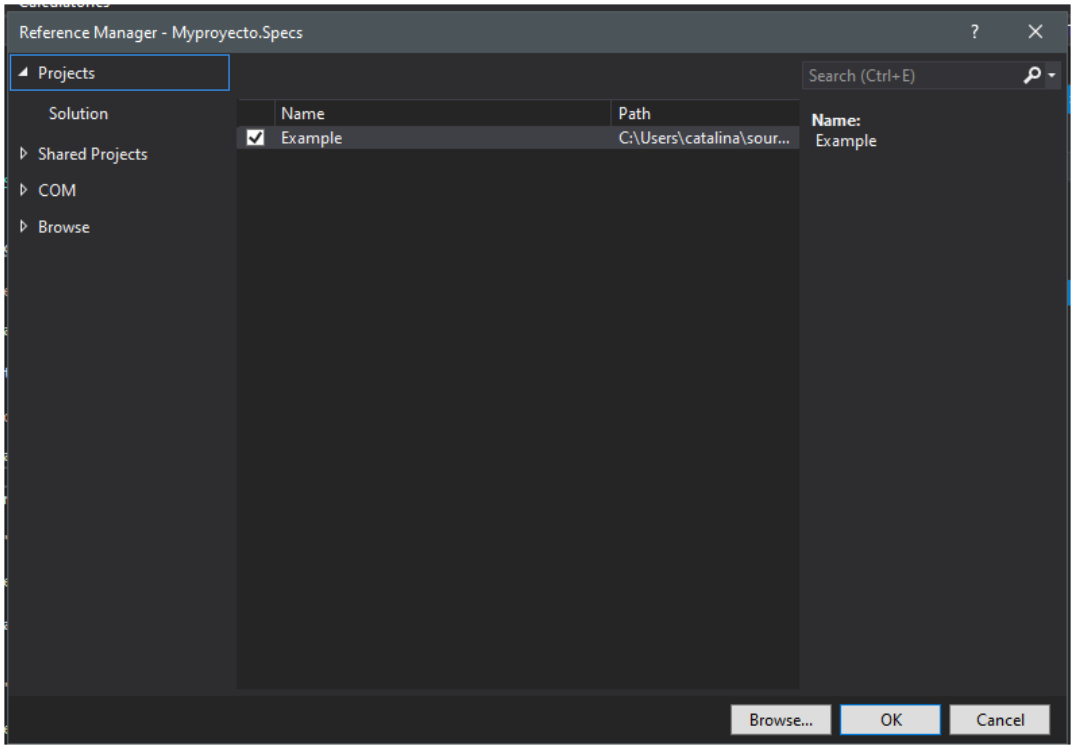
- Para implementar los métodos que se marcan como pendientes en el archivo de pasos anterior vamos a hacer clic derecho sobre la solución y elegimos la opción Add, New project, C# y seleccionamos un proyecto de tipo Class Library(.NET Core), click en next y le asignamos un nombre “Example” y click en Create.



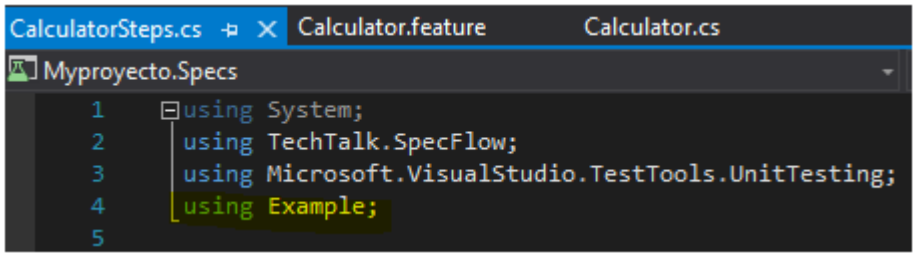
El siguiente paso es asociar el proyecto de pruebas unitarias con el nuevo proyecto que acabamos de configurar, para ello seleccionamos en proyecto de Pruebas, clic derecho y seleccionamos la opción *Add, Reference,* en este se genera una clase por default



- En este paso elegimos el menú Projects y seleccionamos el que acabamos de Crear “Example” y damos ok.



- Una vez referenciados los dos proyectos desde el proyecto de pruebas abrimos el archivo “CalculatorSteps” y hacemos una referencia al proyecto Example:



En este paso vamos a modificar el código generado en el archivo de pasos del proyecto de pruebas, siempre tratando de escribir las variables en términos de negocio:

Final CalculatorSteps.cs Code

```
namespace Myproject.Specs
{
    [Binding]
    public class CalculatorSteps
    {
        private int result;
```

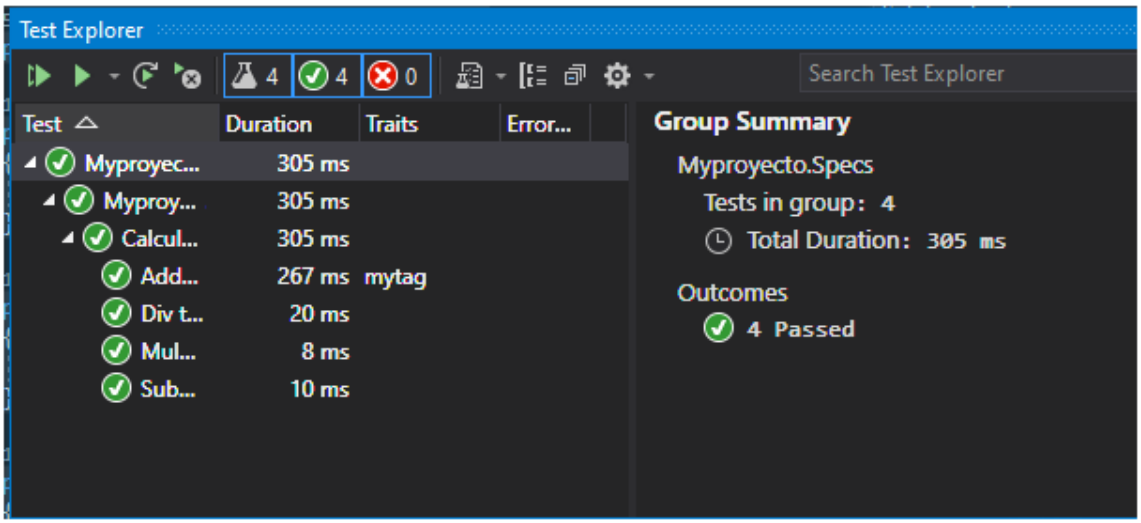
```
        private Calculator calculator = new Calculator();

        [Given(@"I have entered (.*) into the calculator")]
        public void GivenIHaveEnteredIntoTheCalculator(int number)
        {
            calculator.FirstNumber = number;
        }
        [Given(@"I have also entered (.*) into the calculator")]
        public void GivenIHaveAlsoEnteredIntoTheCalculator(int number)
        {
            calculator.SecondNumber = number;
        }
        [When(@"I press add")]
        public void WhenIPressAdd()
        {
            result = calculator.Add();
        }

        [When(@"I press sub")]
        public void WhenIPressSub()
        {
            result = calculator.Sub();
        }

        [When(@"I press mul")]
        public void WhenIPressMul()
        {
            result = calculator.Mul();
        }
        [When(@"I press div")]
        public void WhenIPressDiv()
        {
            result = calculator.Div();
        }

        [Then(@"the result should be (.*) on the screen")]
        public void ThenTheResultShouldBeOnTheScreen(int expectedResult)
        {
            Assert.AreEqual(expectedResult, result);
        }
    }

}
```

Final Calculator.cs Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Example
{
    public class Calculator
    {
        public int FirstNumber { get; set; }
```

```
        public int SecondNumber { get; set; }

        public int Add()
        {
            return FirstNumber + SecondNumber;
        }

        public int Sub()
        {
            return FirstNumber - SecondNumber;
        }

        public int Mul()
        {
            return FirstNumber * SecondNumber;
        }

        public int Div()
        {
            return FirstNumber / SecondNumber;
        }

    }
}
```

Seleccionamos en el menú de visual Studio la opción *Test, y Test Explorer*, en esta ventana elegimos la opción de la parte superior "*Run All*":



Como podemos ver el resumen de la ejecución, el caso de prueba fue exitoso, así podemos ver como esta herramienta permite iniciar procesos de pruebas tan pronto se tiene la especificación y como aporta también al desarrollador una forma de iniciar a trabajar ágilmente sobre lo verdaderamente importante de la funcionalidad necesaria.