

1. Pruebas unitarias MSTest.....	2
1.2. Ejemplo de prueba unitaria utilizando MSTest en C#.....	3
2. Pruebas unitarias NUnit.....	7
2.2. Ejemplo de prueba unitaria utilizando NUnit en C#.....	8

## 1. Pruebas unitarias MSTest

En el contexto de pruebas unitarias en C#, la clase **Assert** se utiliza para realizar afirmaciones o aserciones que verifican el comportamiento esperado de tu código. Aquí hay algunos métodos comunes de la clase **Assert** en el marco de pruebas unitarias MSTest, aunque ten en cuenta que la disponibilidad y la sintaxis pueden variar según el marco de pruebas que estés utilizando:

1. **AreEqual**: Comprueba si dos valores son iguales.

```
Assert.AreEqual(expected, actual);
```

2. **AreNotEqual**: Comprueba si dos valores no son iguales.

```
Assert.AreNotEqual(notExpected, actual);
```

3. **AreSame**: Comprueba si dos referencias apuntan al mismo objeto.

```
Assert.AreSame(expected, actual);
```

4. **AreNotSame**: Comprueba si dos referencias no apuntan al mismo objeto.

```
Assert.AreNotSame(notExpected, actual);
```

5. **IsTrue**: Comprueba si la expresión dada es verdadera.

```
Assert.IsTrue(expression);
```

6. **IsFalse**: Comprueba si la expresión dada es falsa.

```
Assert.IsFalse(expression);
```

7. **IsNull**: Comprueba si el valor proporcionado es nulo.

```
Assert.IsNull(value);
```

8. **IsNotNull**: Comprueba si el valor proporcionado no es nulo.

```
Assert.IsNotNull(value);
```

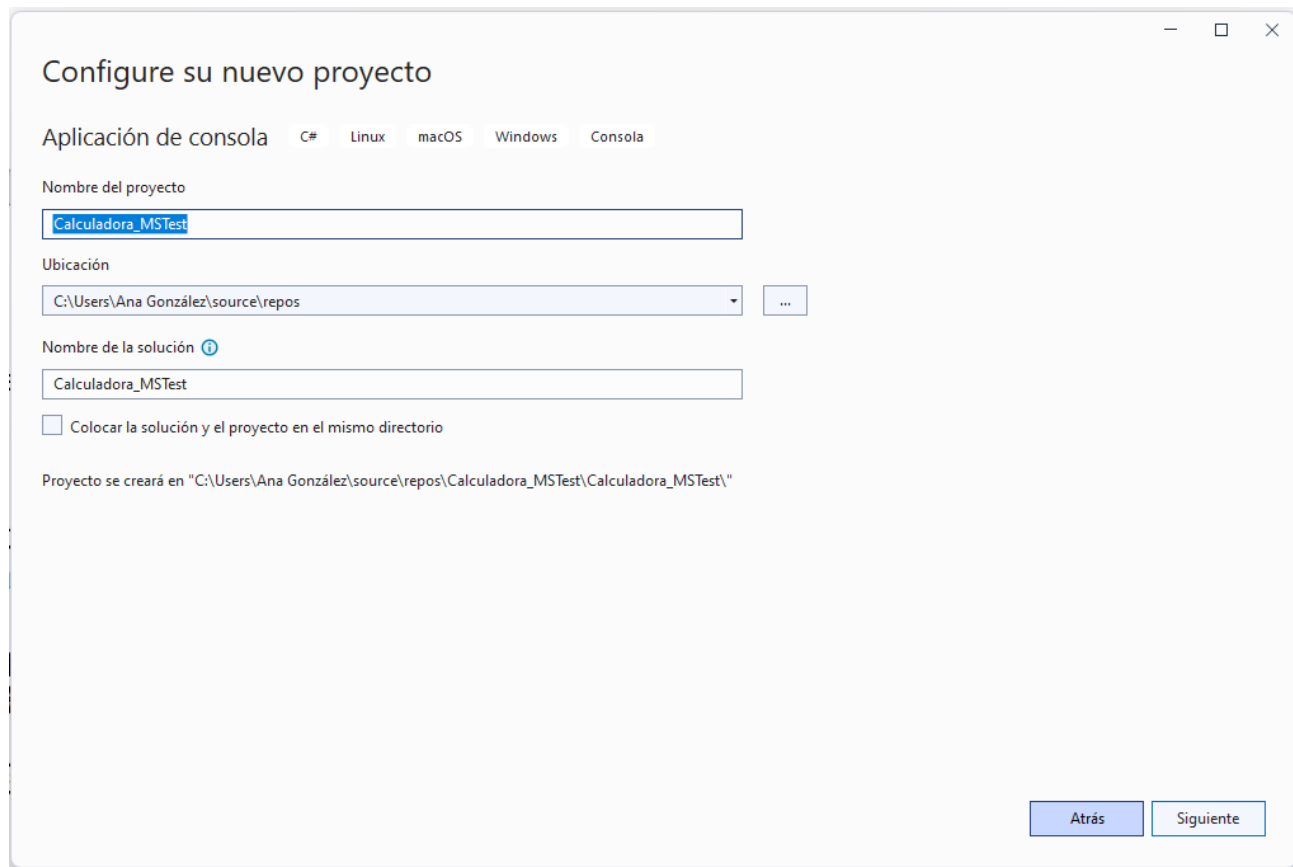
9. **Fail**: Indica una falla en la prueba.

```
Assert.Fail("Mensaje de error");
```

Estos son solo algunos ejemplos de los métodos disponibles en la clase **Assert**. Dependiendo del marco de pruebas que estés utilizando (MSTest, NUnit, xUnit, etc.), la sintaxis y los métodos pueden variar, pero la idea general es proporcionar una forma de verificar condiciones y generar resultados de prueba.

## 1.2. Ejemplo de prueba unitaria utilizando MSTest en C#.

Supongamos que tenemos una clase Calculadora con un método Sumar que quieres probar:



Configure su nuevo proyecto

Aplicación de consola C# Linux macOS Windows Consola

Nombre del proyecto

Calculadora\_MSTest

Ubicación

C:\Users\Ana González\source\repos

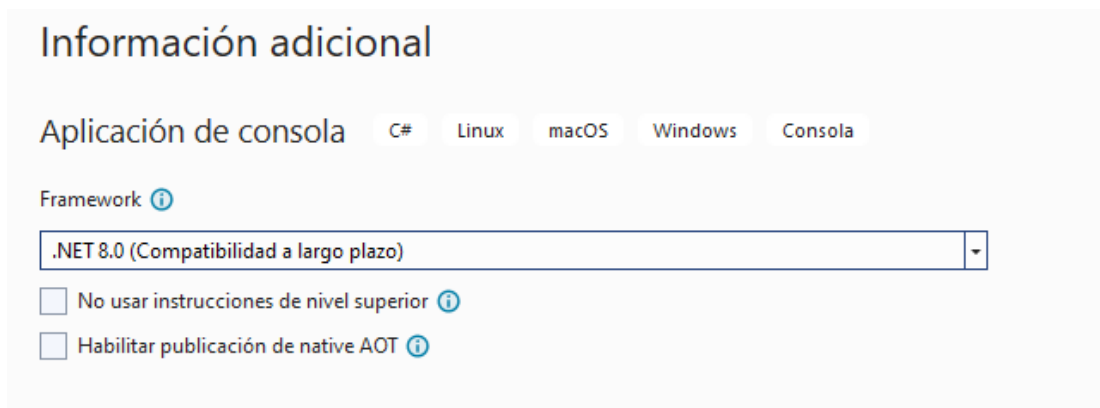
Nombre de la solución ⓘ

Calculadora\_MSTest

☐ Colocar la solución y el proyecto en el mismo directorio

Proyecto se creará en "C:\Users\Ana González\source\repos\Calculadora\_MSTest\Calculadora\_MSTest\"

Atrás Siguiente



Información adicional

Aplicación de consola C# Linux macOS Windows Consola

Framework ⓘ

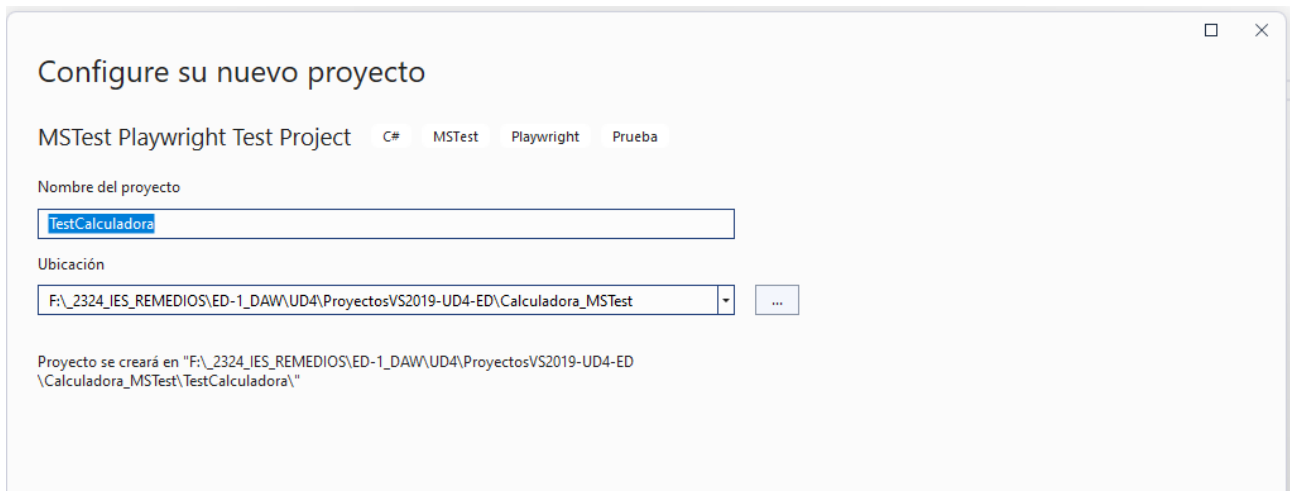
.NET 8.0 (Compatibilidad a largo plazo)

☐ No usar instrucciones de nivel superior ⓘ

☐ Habilitar publicación de native AOT ⓘ

```
// Calculadora.cs
public class Calculadora
{
    public int Sumar(int a, int b)
    {
        return a + b;
    }
}
```

Pruebas unitarias para esta clase utilizando MSTest. Crea un archivo llamado TestCalculadora.cs:



```
using Microsoft.VisualStudio.TestTools.UnitTesting;
```

```
[TestClass]
public class TestCalculadora
{
    [TestMethod]
    public void TestSumarPositivos()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();

        // Act
        int resultado = calculadora.Sumar(3, 5);

        // Assert
        Assert.AreEqual(8, resultado);
    }

    [TestMethod]
    public void TestSumarNegativos()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();

        // Act
        int resultado = calculadora.Sumar(-2, -4);

        // Assert
        Assert.AreEqual(-6, resultado);
    }
}
```

```

[TestMethod]
public void TestSumarMezclados()
{
    // Arrange
    Calculadora calculadora = new Calculadora();

    // Act
    int resultado = calculadora.Sumar(10, -7);

    // Assert
    Assert.AreEqual(3, resultado);
}
}

```

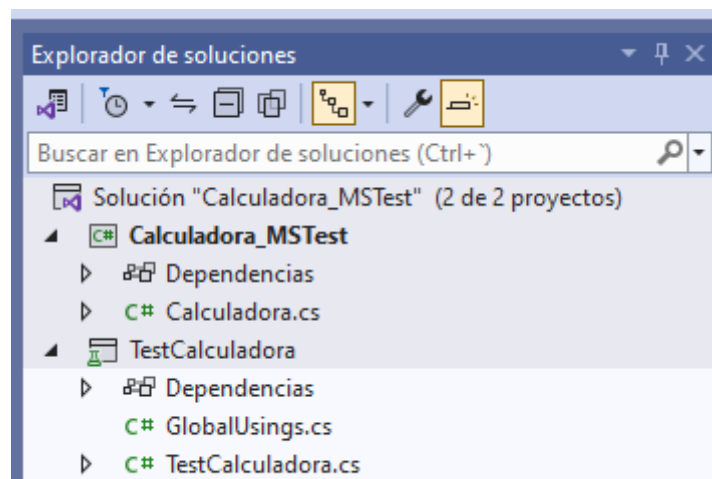
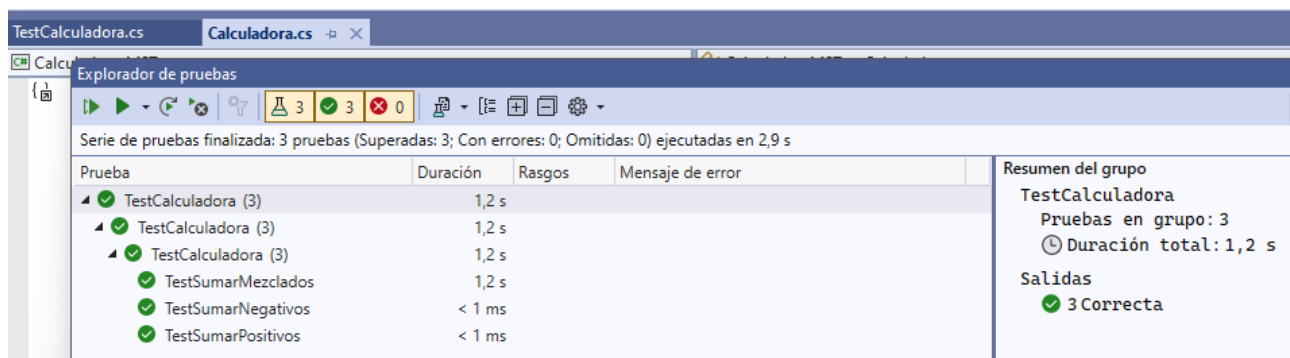
En este ejemplo:

Se ha creado una clase llamada TestCalculadora.

Cada método dentro de la clase TestCalculadora representa una prueba individual y está decorado con [TestMethod].

Dentro de cada método, se realiza la organización (Arrange), la acción (Act), y se utiliza la aserción (Assert) para verificar que el método Sumar de la clase Calculadora produce los resultados esperados.

Puedes ejecutar estas pruebas utilizando el explorador de pruebas de Visual Studio o mediante la línea de comandos utilizando la herramienta vstest.console.exe.



```

namespace CalculadoraMSTest
{
    6 referencias
    public class Calculadora
    {
        3 referencias | 3/3 pasando
        public int Sumar(int a, int b)
        {
            return a + b;
        }

        0 referencias
        public static void Main()
        {
            Console.WriteLine("Hello World: suma a y b!");
        }
    }
}

```

```

using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.IO;
using System;
using CalculadoraMSTest;

namespace TestCalculadora
{
    [TestClass]
    0 referencias
    public class TestCalculadora
    {

```

## 2. Pruebas unitarias NUnit

En NUnit, otro popular marco de pruebas unitarias para C#, también se utiliza la clase `Assert` para realizar afirmaciones o aserciones. Aquí hay algunos métodos comunes de la clase `Assert` en NUnit:

1. **AreEqual:** Comprueba si dos valores son iguales.  
`Assert.AreEqual(expected, actual);`
2. **AreNotEqual:** Comprueba si dos valores no son iguales.  
`Assert.AreNotEqual(notExpected, actual);`
3. **AreSame:** Comprueba si dos referencias apuntan al mismo objeto.  
`Assert.AreSame(expected, actual);`
4. **AreNotSame:** Comprueba si dos referencias no apuntan al mismo objeto.  
`Assert.AreNotSame(notExpected, actual);`
5. **IsTrue:** Comprueba si la expresión dada es verdadera.  
`Assert.IsTrue(expression);`
6. **IsFalse:** Comprueba si la expresión dada es falsa.  
`Assert.IsFalse(expression);`
7. **IsNull:** Comprueba si el valor proporcionado es nulo.  
`Assert.IsNull(value);`
8. **IsNotNull:** Comprueba si el valor proporcionado no es nulo.  
`Assert.IsNotNull(value);`
9. **Fail:** Indica una falla en la prueba.  
`Assert.Fail("Mensaje de error");`

Estos métodos son similares a los de MSTest, pero es importante destacar que pueden haber algunas diferencias en la sintaxis y en los nombres de los métodos entre los distintos marcos de pruebas unitarias. En general, los marcos de pruebas unitarias proporcionan funcionalidades similares para realizar aserciones y verificar el comportamiento esperado del código bajo prueba.

## 2.2. Ejemplo de prueba unitaria utilizando NUnit en C#.

Supongamos que tenemos una clase Calculadora con un método Sumar que quieres probar:

The image shows the 'Configure your new project' dialog in Visual Studio. The 'Aplicación de consola' (Console Application) template is selected. The project name is 'Calculadora\_NUnit'. The location is 'F:\\_2324\_IES\_REMEDIOS\ED-1\_DAW\UD4\ProyectosVS2019-UD4-ED'. The solution is 'Crear nueva solución' (Create new solution). The solution name is 'Calculadora\_NUnit'. There is an unchecked checkbox for 'Colocar la solución y el proyecto en el mismo directorio' (Place the solution and the project in the same directory). The project will be created in 'F:\\_2324\_IES\_REMEDIOS\ED-1\_DAW\UD4\ProyectosVS2019-UD4-ED\Calculadora\_NUnit\Calculadora\_NUnit\'. Below this, the 'Información adicional' (Additional Information) section shows the 'Framework' set to '.NET 8.0 (Compatibilidad a largo plazo)' (Long-term compatibility). There are two unchecked checkboxes: 'No usar instrucciones de nivel superior' (Do not use top-level statements) and 'Habilitar publicación de native AOT' (Enable native AOT publishing).

Configure su nuevo proyecto

Aplicación de consola C# Linux macOS Windows Consola

Nombre del proyecto

Calculadora\_NUnit

Ubicación

F:\\_2324\_IES\_REMEDIOS\ED-1\_DAW\UD4\ProyectosVS2019-UD4-ED

Solución

Crear nueva solución

Nombre de la solución ⓘ

Calculadora\_NUnit

☐ Colocar la solución y el proyecto en el mismo directorio

Proyecto se creará en "F:\\_2324\_IES\_REMEDIOS\ED-1\_DAW\UD4\ProyectosVS2019-UD4-ED\Calculadora\_NUnit\Calculadora\_NUnit\"

Información adicional

Aplicación de consola C# Linux macOS Windows Consola

Framework ⓘ

.NET 8.0 (Compatibilidad a largo plazo)

☐ No usar instrucciones de nivel superior ⓘ

☐ Habilitar publicación de native AOT ⓘ

```
// Calculadora.cs
public class Calculadora
{
    public int Sumar(int a, int b)
    {
        return a + b;
    }
}
```



Pruebas unitarias para esta clase utilizando NUnit. Crea un archivo llamado TestCalculadora.cs:

Configure su nuevo proyecto

Proyecto de prueba NUnit   C#   Linux   macOS   Windows   Escritorio   Prueba   Web

Nombre del proyecto

TestCalculadoraNUnit

Ubicación

F:\\_2324\_IES\_REMEDIOS\ED-1\_DAW\UD4\ProyectosVS2019-UD4-ED\Calculadora\_NUnit   ...

Proyecto se creará en "F:\\_2324\_IES\_REMEDIOS\ED-1\_DAW\UD4\ProyectosVS2019-UD4-ED\Calculadora\_NUnit\TestCalculadoraNUnit\"

```
using NUnit.Framework;
[TestFixture]
public class TestCalculadora
{
    [Test]
    public void TestSumarPositivos()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();
        // Act
        int resultado = calculadora.Sumar(3, 5);
        // Assert
        Assert.AreEqual(8, resultado);
    }
    [Test]
    public void TestSumarNegativos()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();
        // Act
        int resultado = calculadora.Sumar(-2, -4);
        // Assert
        Assert.AreEqual(-6, resultado);
    }
    [Test]
    public void TestSumarMezclados()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();
        // Act
        int resultado = calculadora.Sumar(10, -7);
        // Assert
        Assert.AreEqual(3, resultado);
    }
}
```

Aquí tienes un ejemplo de prueba unitaria utilizando NUnit en C#. Supongamos que tienes la misma clase Calculadora con el método Sumar que deseas probar:

```
csharp
```

```
// Calculadora.cs
public class Calculadora
{
    public int Sumar(int a, int b)
    {
        return a + b;
    }
}
```

Ahora, puedes escribir pruebas unitarias para esta clase utilizando NUnit. Crea un archivo llamado TestCalculadora.cs:

```
csharp
```

```
using NUnit.Framework;

[TestFixture]
public class TestCalculadora
{
    [Test]
    public void TestSumarPositivos()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();

        // Act
        int resultado = calculadora.Sumar(3, 5);

        // Assert
        Assert.AreEqual(8, resultado);
    }

    [Test]
    public void TestSumarNegativos()
    {
        // Arrange
        Calculadora calculadora = new Calculadora();
```

```

// Act
int resultado = calculadora.Sumar(-2, -4);

// Assert
Assert.AreEqual(-6, resultado);
}

[Test]
public void TestSumarMezclados()
{
    // Arrange
    Calculadora calculadora = new Calculadora();

    // Act
    int resultado = calculadora.Sumar(10, -7);

    // Assert
    Assert.AreEqual(3, resultado);
}
}

```

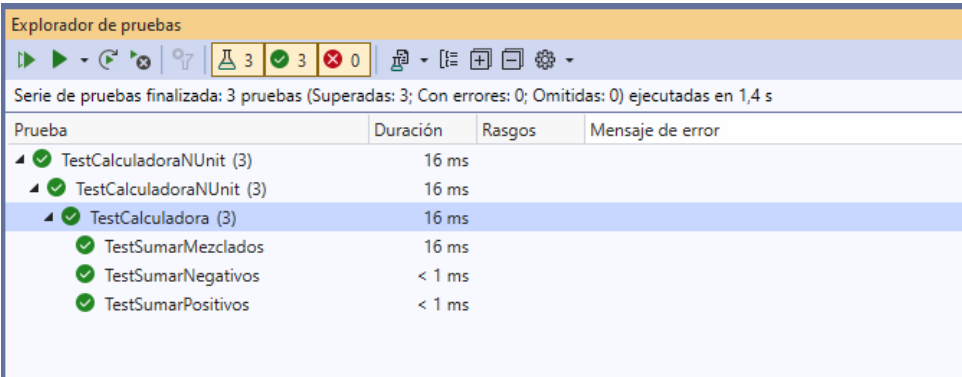
En este ejemplo:

Se ha creado una clase llamada TestCalculadora.

Cada método dentro de la clase TestCalculadora representa una prueba individual y está decorado con [Test].

Dentro de cada método, se realiza la organización (Arrange), la acción (Act), y se utiliza la aserción (Assert) para verificar que el método Sumar de la clase Calculadora produce los resultados esperados.

Puedes ejecutar estas pruebas utilizando el explorador de pruebas de NUnit, o mediante la línea de comandos utilizando NUnit Console Runner u otras herramientas compatibles con NUnit.



Explorador de pruebas

Serie de pruebas finalizada: 3 pruebas (Superadas: 3; Con errores: 0; Omitidas: 0) ejecutadas en 1,4 s

Prueba	Duración	Rasgos	Mensaje de error
TestCalculadoraNUnit (3)	16 ms		
TestCalculadoraNUnit (3)	16 ms		
TestCalculadora (3)	16 ms		
TestSumarMezclados	16 ms		
TestSumarNegativos	< 1 ms		
TestSumarPositivos	< 1 ms		