

# Testing in Visual Studio



Universidad de Almería

Antonio Bordes Giménez

# Índice

1. MS Test
  - a. Clases utilizadas
  - b. Creación de proyecto de pruebas
  - c. Ejecución de tests
2. IntelliTest
  - a. Clases utilizadas
  - b. Generación de pruebas
  - c. Ejecución de tests
3. NUnit 3
  - a. Integración en Visual Studio
  - b. Creación de proyecto de pruebas
  - c. Test parametrizados
4. Cobertura
5. Referencias

- Realizado con C# en Visual Studio 2015
- Integrado con Github
- Repositorio público → <https://github.com/abordes96/testing-vs>

# MS Test

# MS Test

## ❖ Clases utilizadas

```
0 referencias | abordes96, Hace 23 horas | 1 autor, 1 cambio
public class Calculadora
{
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public double Sumar(double num1, double num2)
    {
        return num1 + num2;
    }
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public double restar(double num1, double num2)
    {
        return num1 - num2;
    }
}
```

```
0 referencias | 0 cambios | 0 autores, 0 cambios
public double max(double num1, double num2)
{
    if (num1 > num2) return num1;

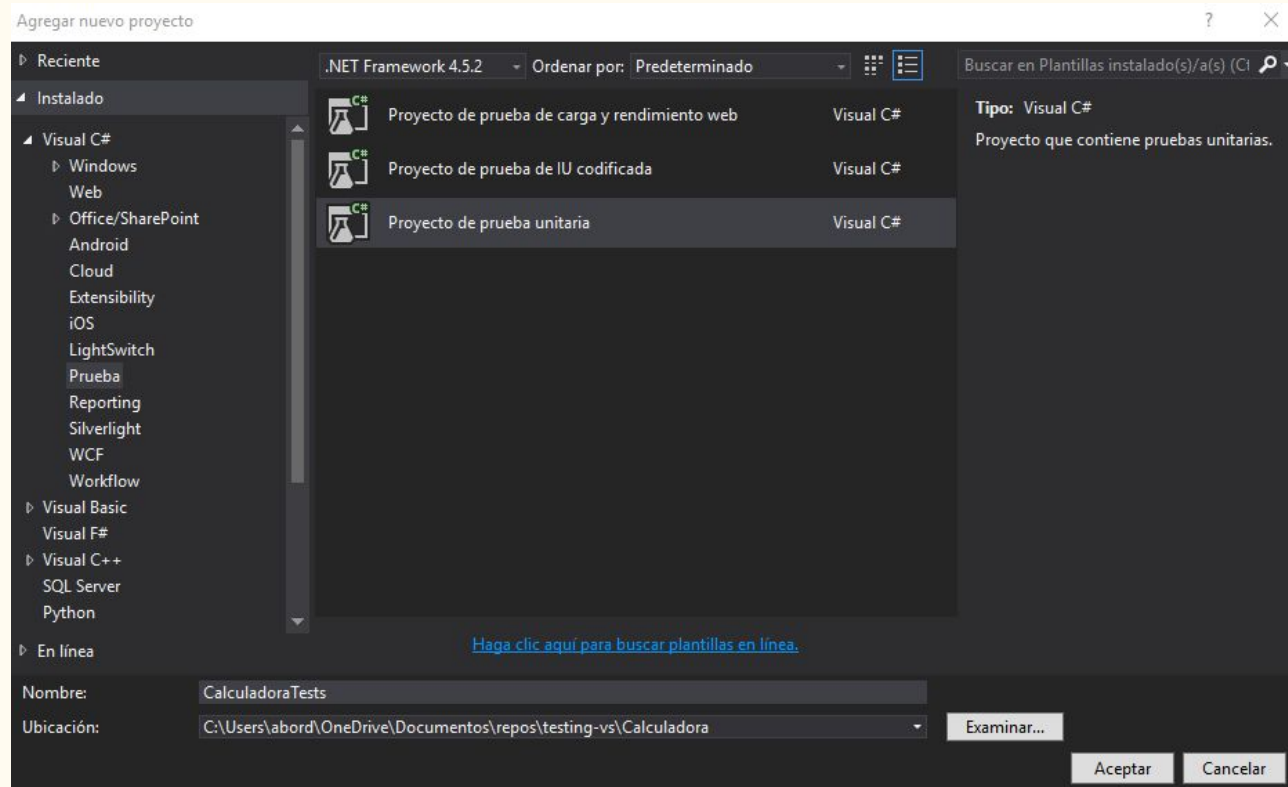
    return num2;
}

0 referencias | 0 cambios | 0 autores, 0 cambios
public double min(double num1, double num2)
{
    if (num1 < num2) return num1;

    return num2;
}
```

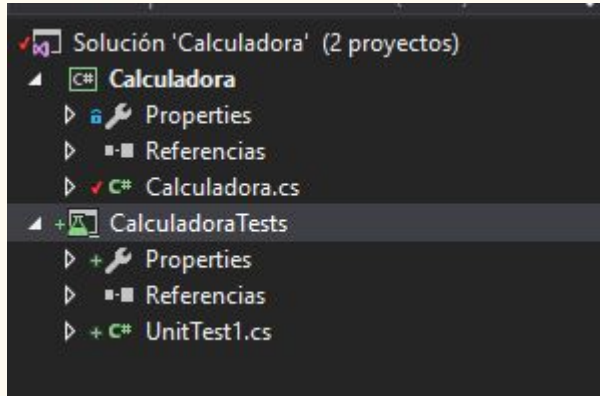
# MS Test

## ❖ Creación de proyecto de pruebas

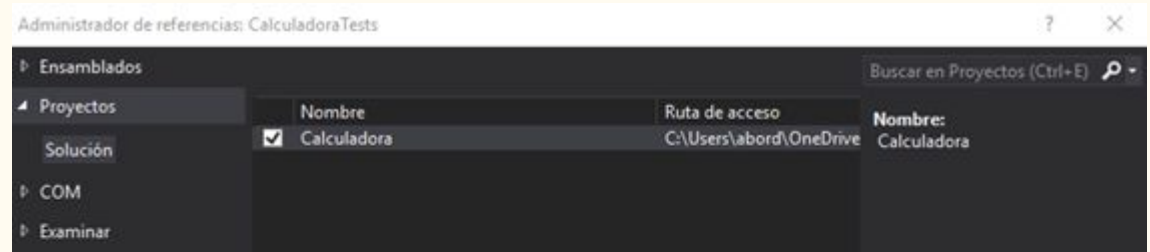


# MS Test

- Agregar referencias



Proyecto generado



Referencia al proyecto que contiene los métodos

# MS Test

- Test generado

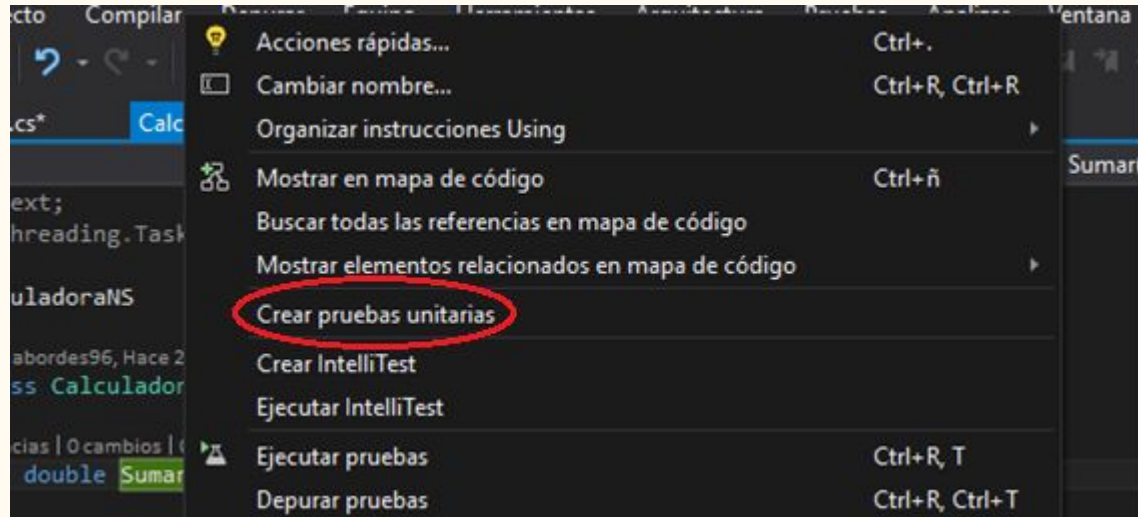
```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CalculadoraTests
{
    [TestClass]
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public class OperacionesElementalesTest
    {
        [TestMethod]
        0 referencias | 0 cambios | 0 autores, 0 cambios
        public void TestMethod1()
        {
        }
    }
}
```



# MS Test

- Generación de plantillas de prueba



# MS Test

Crear pruebas unitarias

Marco de prueba: MSTest [Obtener extensiones adicionales](#)

Proyecto de prueba: CalculadoraTests

Formato de prueba para proyecto de prueba: [Project]Tests

Espacio de nombres: [Namespace].Tests

Archivo de salida: OperacionesElementalesTest.cs

Formato de nombre para clase de prueba: [Class]Tests

Formato de nombre para método de prueba: [Method]Test

Código para el método de prueba: Error de aserción

Aceptar Cancelar

# MS Test

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

using CalculadoraNS;

namespace CalculadoraNS.Tests
{
    [TestClass]
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public class OperacionesElementalesTest
    {
        [TestMethod]
        0 referencias | 0 cambios | 0 autores, 0 cambios
        public void SumarTest()
        {
            Assert.Fail();
        }
    }
}
```

Método de prueba generado

# MS Test

- Clase Assert

AreEqual(Object,  
Object, String)

AreNotEqual(Object  
, Object, String)

IsTrue(Boolean,  
String)

IsFalse(Boolean,  
String)

IsNull(Object,  
String)

IsNotNull(Object,  
String)

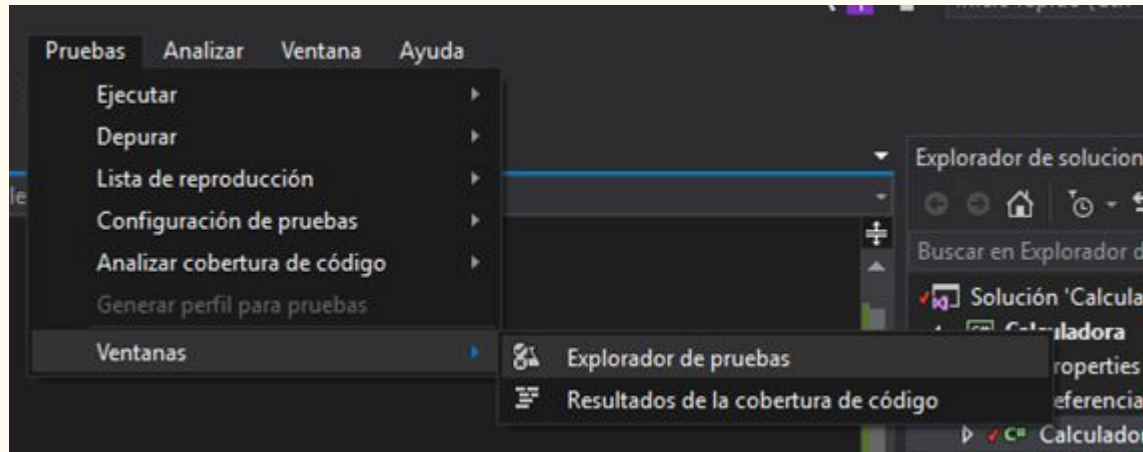
Fail(String)

```
[TestClass()]
0 referencias | 0 cambios | 0 autores, 0 cambios
public class OperacionesElementalesTest
{
    [TestMethod()]
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public void SumarTest()
    {
        var calculadora = new Calculadora();
        var resultado = calculadora.Sumar(1, 2);
        Assert.AreEqual(3, resultado);
    }

    [TestMethod()]
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public void restarTest()
    {
        var calculadora = new Calculadora();
        var resultado = calculadora.restar(1, 2);
        Assert.AreEqual(-1, resultado);
    }
}
```

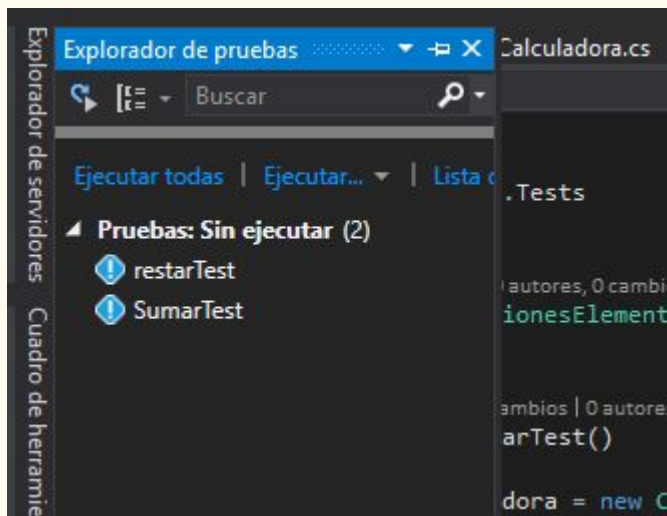
# MS Test

## ❖ Ejecución de tests

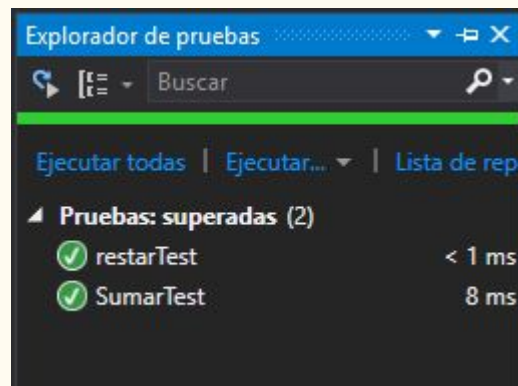


# MS Test

- Estados de un test



Test sin ejecutar

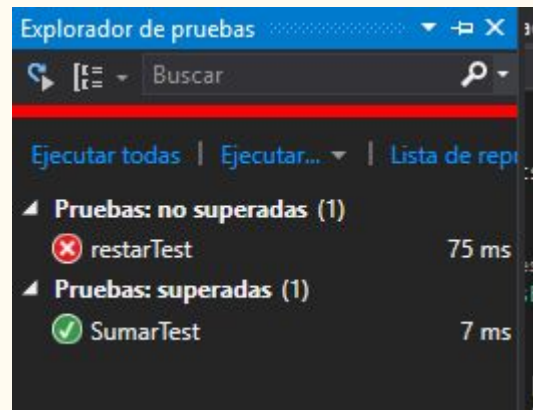


Test correcto

# MS Test

```
[TestMethod()]
✓ | 0 referencias | 0 cambios | 0 autores, 0 cambios
public void restarTest()
{
    var calculadora = new Calculadora();
    var resultado = calculadora.restar(1, 2);
    Assert.AreEqual(1, resultado);
}
```

Error en un Assert



Test erróneo

# MS Test

- Se puede visualizar en el código de los test el resultado de la última ejecución de las pruebas

```
{  
    [TestMethod()]  
    ✓ | 0 referencias | 0 cambios | 0 autores, 0 cambios  
    public void SumarTest()  
    {  
        var calculadora = new Calculadora();  
        var resultado = calculadora.Sumar(1, 2);  
        Assert.AreEqual(3, resultado);  
    }  
  
    [TestMethod()]  
    ✗ | 0 referencias | 0 cambios | 0 autores, 0 cambios  
    public void restarTest()  
    {  
        var calculadora = new Calculadora();  
        var resultado = calculadora.restar(1, 2);  
        Assert.AreEqual(1, resultado);  
    }  
}
```



# IntelliTest

# Intellitest

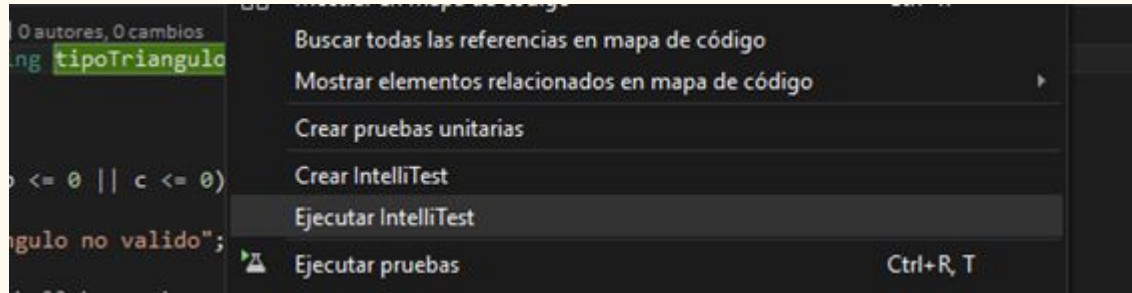
## ❖ Clases utilizadas

```
public static String tipoTriangulo(int a, int b, int c)
{
    String tipo;

    if(a <= 0 || b <= 0 || c <= 0)
    {
        tipo = "Triangulo no valido";
    }
    else if (a == b && b == c)
    {
        tipo = "Equilatero";
    }
    else if (b == c || a == b || c == a)
    {
        tipo = "Isosceles";
    }
    else {
        tipo = "Escaleno";
    }
    return tipo;
}
```

# IntelliTest

## ❖ Generación de pruebas



# Intellitest

- Analiza las posibles variantes del código y genera una ejecución parametrizada

Resultados de exploración de IntelliTest - stopped

Calculadora.tipoTriangulo(Int32, Ir ▶ Ejecutar 0 advertencias

7 0 15/15 bloques, 0/0 aserciones, 7 ejecuciones

	▲	a	b	c	resultado	Resumen/excepción	Mensaje d
✓	1	0	0	0	"Triangulo no valido"		
✓	2	1	0	0	"Triangulo no valido"		
✓	3	1	1	0	"Triangulo no valido"		
✓	4	1	1	1	"Equilatero"		
✓	5	3	3	2	"Isosceles"		
✓	6	2	3	1	"Escaleno"		
✓	7	3	2	2	"Isosceles"		

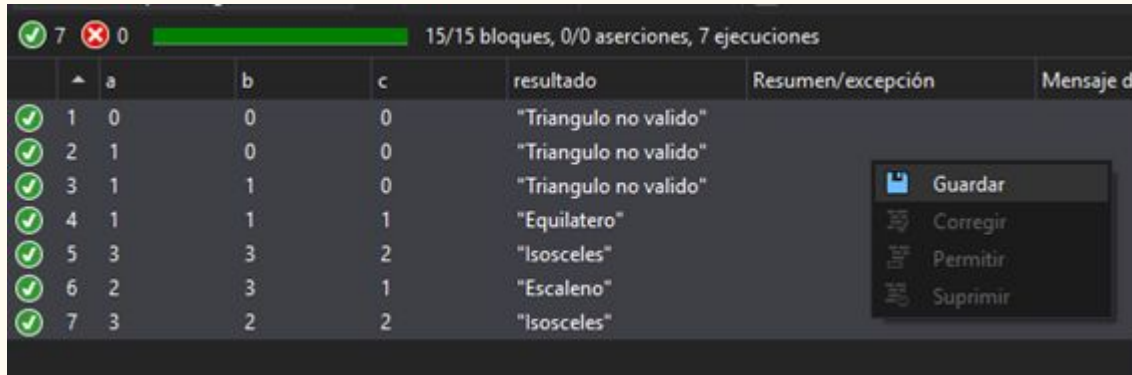
**Detalles**

```
[TestMethod]
[PexGeneratedBy(typeof(CalculadoraTest))]
public void tipoTriangulo280()
{
    string s;
    s = this.tipoTriangulo(0, 0, 0);
}
```

▶ Seguimiento de la pila





# Intellitest

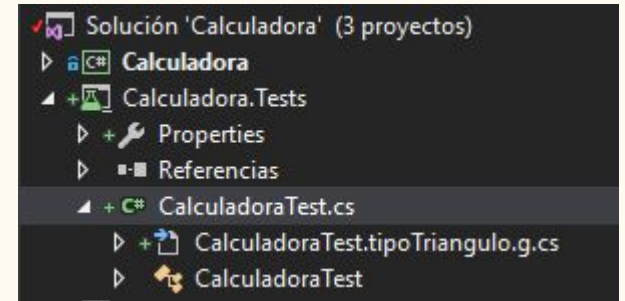
- Al guardar la salida de las ejecuciones se genera un proyecto de pruebas



7 0 15/15 bloques, 0/0 aserciones, 7 ejecuciones

	^	a	b	c	resultado	Resumen/excepción	Mensaje d
✓	1	0	0	0	"Triangulo no valido"		
✓	2	1	0	0	"Triangulo no valido"		
✓	3	1	1	0	"Triangulo no valido"		
✓	4	1	1	1	"Equilatero"		
✓	5	3	3	2	"Isosceles"		
✓	6	2	3	1	"Escaleno"		
✓	7	3	2	2	"Isosceles"		

 Guardar  
 Corregir  
 Permitir  
 Suprimir



# Intellitest

- Una de las clases del proyecto generado contiene las ejecuciones parametrizadas realizadas

```
namespace CalculadoraNS.Tests
{
    8 referencias | 0 cambios | 0 autores, 0 cambios
    public partial class CalculadoraTest
    {

        [TestMethod]
        [PexGeneratedBy(typeof(CalculadoraTest))]
        0 referencias | 0 cambios | 0 autores, 0 cambios
        public void tipoTriangulo280()
        {
            string s;
            s = this.tipoTriangulo(0, 0, 0);
        }

        [TestMethod]
        [PexGeneratedBy(typeof(CalculadoraTest))]
        0 referencias | 0 cambios | 0 autores, 0 cambios
        public void tipoTriangulo727()
        {
            string s;
            s = this.tipoTriangulo(1, 0, 0);
        }
    }
}
```

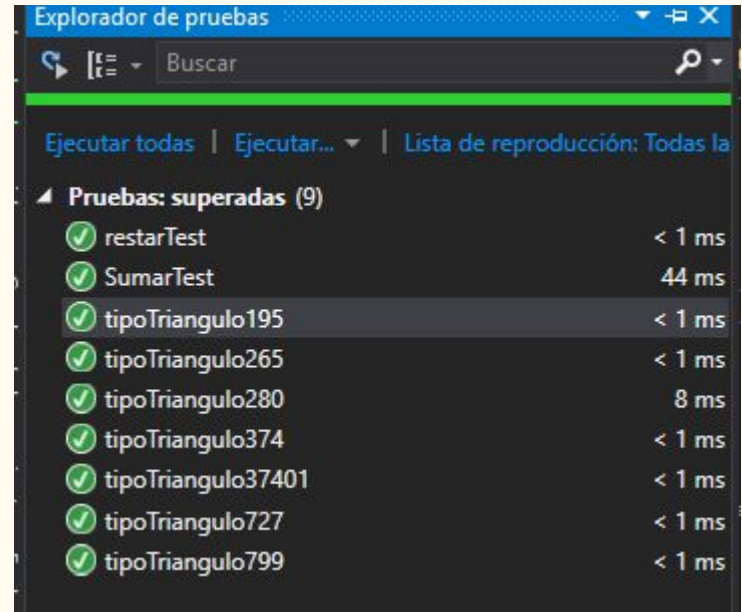
# Intellitest

- Se pueden añadir Assert para realizar las comprobaciones de la funcionalidad

```
}  
  
[TestMethod]  
[PexGeneratedBy(typeof(CalculadoraTest))]  
✓ | 0 referencias | 0 cambios | 0 autores, 0 cambios  
public void tipoTriangulo374()  
{  
    string s;  
    s = this.tipoTriangulo(1, 1, 1);  
    Assert.AreEqual(s, "Equilatero");  
}
```

# Intellitest

- Se puede ejecutar con el explorador de pruebas como cualquier otro test

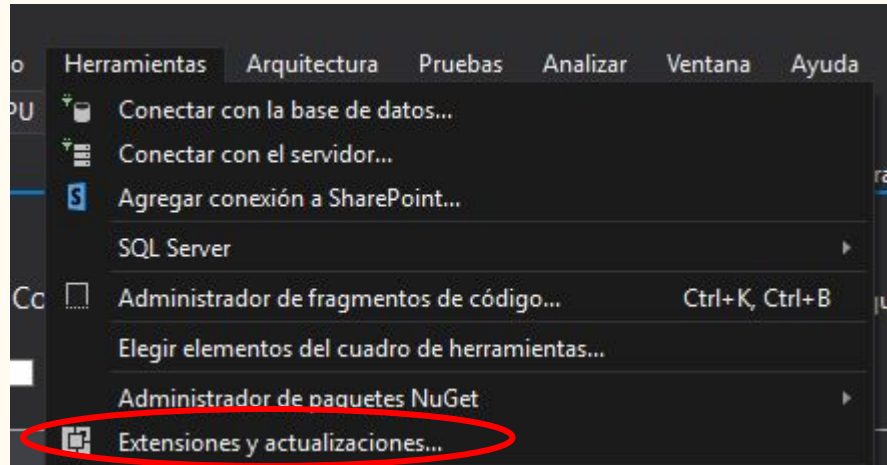




# NUnit 3

# NUnit 3

## ❖ Integración en Visual Studio



# NUnit 3

The screenshot shows the 'Extensões y actualizaciones' (Extensions and Updates) window in Visual Studio. The search term 'nunit' is entered in the search bar. The results are sorted by 'Relevancia' (Relevance). The top result is 'NUnit 3 Test Adapter' by Charlie Poole, which is highlighted. To its right, a preview pane shows details for the 'nunit' extension, including its version (3.7.0.0), download count (250927), and a 5-star rating. Below the details, a test results window is visible, showing a list of tests: 'Failed Tests (2)' (TestCaseFails(31,11,99) and TestCaseFails\_Result(31,11)), 'Skipped Tests (2)' (TestCaseIsIgnored\_Assert(31,11) and TestCaseIsIgnored\_Property(31,11)), and 'Passed Tests (2)' (TestCaseSucceeds(0,5,5)).

Extensões y actualizaciones

Ordenar por: Relevancia

En línea

- Galería de Visual Studio
  - Controles
  - Herramientas
  - Plantillas
  - Resultados de la búsqueda
  - Galería de ejemplos
- Actualizaciones (5)

**NUnit VS Templates**  
Provides Visual Studio project and item templates for NUnit 3 along with code snippets.

**NUnit 3 Test Adapter**  
NUnit 3 adapter for running tests in Visual Studio. Works with NUnit 3.x, use the NUnit 2 adapter for 2.x tests. [Descargar](#)

**NUnit 2 Test Adapter**  
NUnit 2 adapter for running tests in Visual Studio 2012 and newer. Works with NUnit 2.x, for 3.x tests use the NUnit 3 adapter.

**Test Generator NUnit extension**  
Test Generator, NUnit extensions for Visual Studio 2015. Creates Unit tests and Intelitests with both NUnit 2.6.4 and NUnit 3 fra...

**NUnit Project Template**  
Creates NUnit Test Project and Item Template

**NUnit Test Project Template**  
A project that contains nunit tests.

**AttachTo-Next**

1

Cambiar la configuración de las extensiones y las actualizaciones

Cerrar

nunit

Creado por: Charlie Poole  
Versión: 3.7.0.0  
Descargas: 250927  
Clasificación: ★★★★★ (10 Votos)  
[Más información](#)  
[Extensión de informes en Microsoft](#)

Run All | Run... | Playlist: All Te

**Failed Tests (2)**

- ✗ TestCaseFails(31,11,99)
- ✗ TestCaseFails\_Result(31,11)

**Skipped Tests (2)**

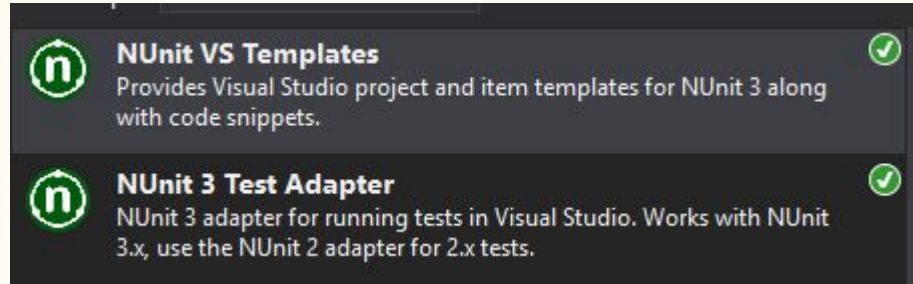
- ⚠ TestCaseIsIgnored\_Assert(31,11)
- ⚠ TestCaseIsIgnored\_Property(31,11)

**Passed Tests (2)**

- ✓ TestCaseSucceeds(0,5,5)

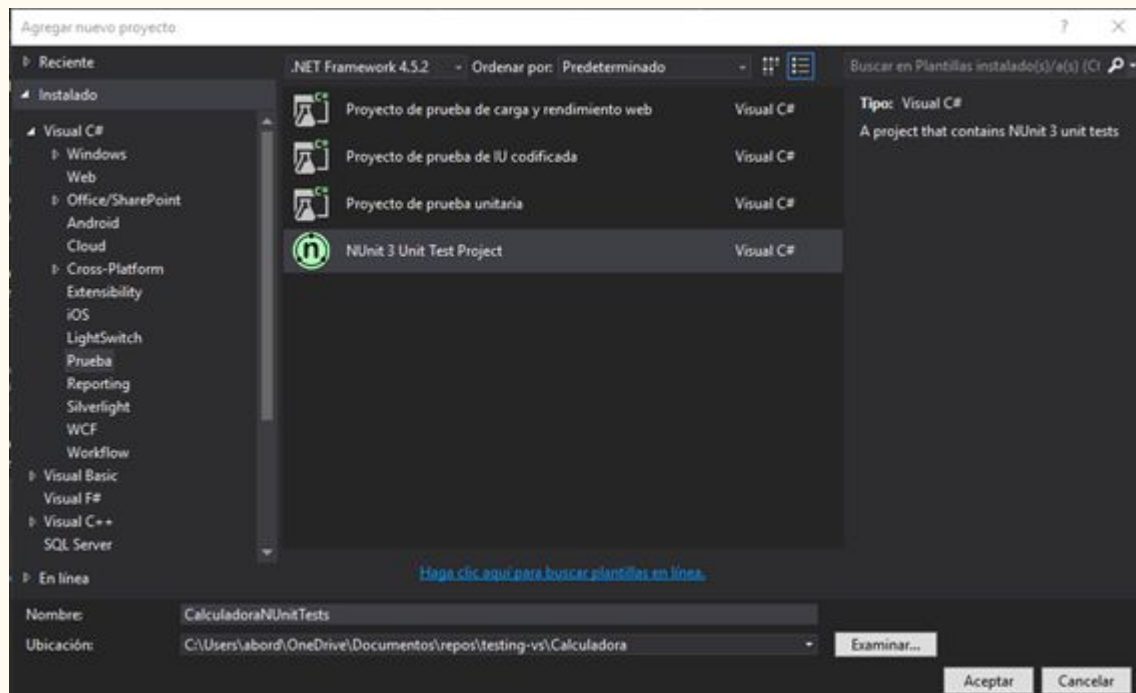
# NUnit 3

- Descargar *Test Adapter* y *Templates*



# NUnit 3

## ❖ Creación de proyecto de pruebas



# NUnit 3

- Comprobar que el framework del proyecto generado sea compatible con el del proyecto a probar

Aplicación

Configuración: N/D Plataforma: N/D

Nombre del ensamblado: CalculadoraNUnitTests

Espacio de nombres predeterminado: CalculadoraNUnitTests

Marco de trabajo de destino: .NET Framework 4.6

Tipo de salida: Biblioteca de clases

Objeto de inicio: (Sin establecer)

Información de ensamblado...

Recursos

Especifique cómo se administrarán los recursos de la aplicación:

☒ Icono y manifiesto

Los manifiestos determinan la configuración específica de una aplicación. Para incrustar un manifiesto personalizado, primero agréguelo al proyecto y luego selecciónelo de la lista de abajo.

# NUnit 3

- TestFixture construirá una instancia separada por cada conjunto de argumentos

```
using CalculadoraNS;

namespace CalculadoraNUnitTests
{
    [TestFixture(1, 2, 3)]
    [TestFixture(2, 2, 4)]
    1 referencia | abordes96, Hace 2 días | 1 autor, 1 cambio
    class CalculadoraText
    {
        private readonly int _num1;
        private readonly int _num2;
        private readonly int _expected;

        0 referencias | abordes96, Hace 2 días | 1 autor, 1 cambio
        public CalculadoraText(int num1, int num2, int expected)
        {
            _num1 = num1;
            _num2 = num2;
            _expected = expected;
        }

        [Test]
        0 referencias | abordes96, Hace 2 días | 1 autor, 1 cambio
        public void Sumar()
        {
            var calculadora = new Calculadora();
            var resultado = calculadora.Sumar(_num1, _num2);
            Assert.AreEqual(_expected, resultado);
        }
    }
}
```

# NUnit 3



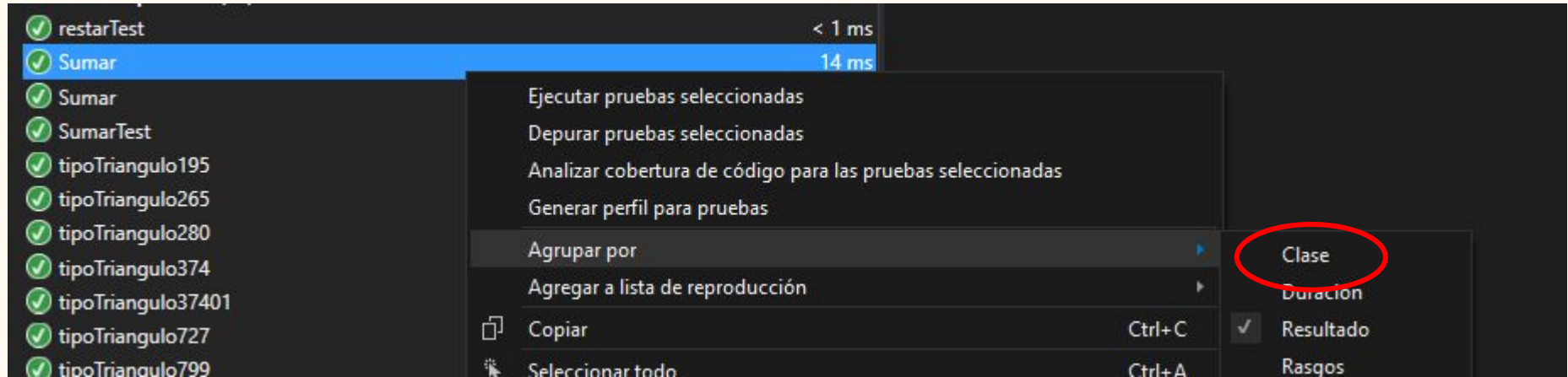
A screenshot of the NUnit test runner interface. It shows a list of 11 tests, all of which have passed, indicated by green checkmarks. The tests are: restarTest, Sumar, Sumar, SumarTest, tipoTriangulo195, and tipoTriangulo265. To the right of each test name is the execution time. The times are: < 1 ms, 14 ms, < 1 ms, 7 ms, < 1 ms, and < 1 ms. The header of the list is 'Pruebas: superadas (11)'.

Pruebas: superadas (11)	
✓ restarTest	< 1 ms
✓ Sumar	14 ms
✓ Sumar	< 1 ms
✓ SumarTest	7 ms
✓ tipoTriangulo195	< 1 ms
✓ tipoTriangulo265	< 1 ms

Resultado al ejecutar el test parametrizado

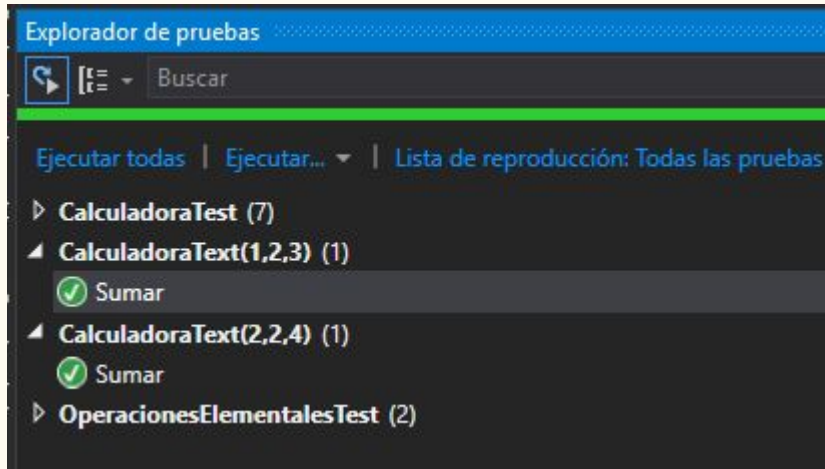


# NUnit 3

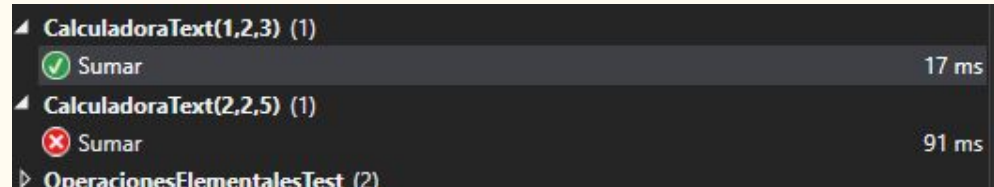


# NUnit 3

- Al agrupar por clase se pueden observar los parámetros de cada prueba parametrizada



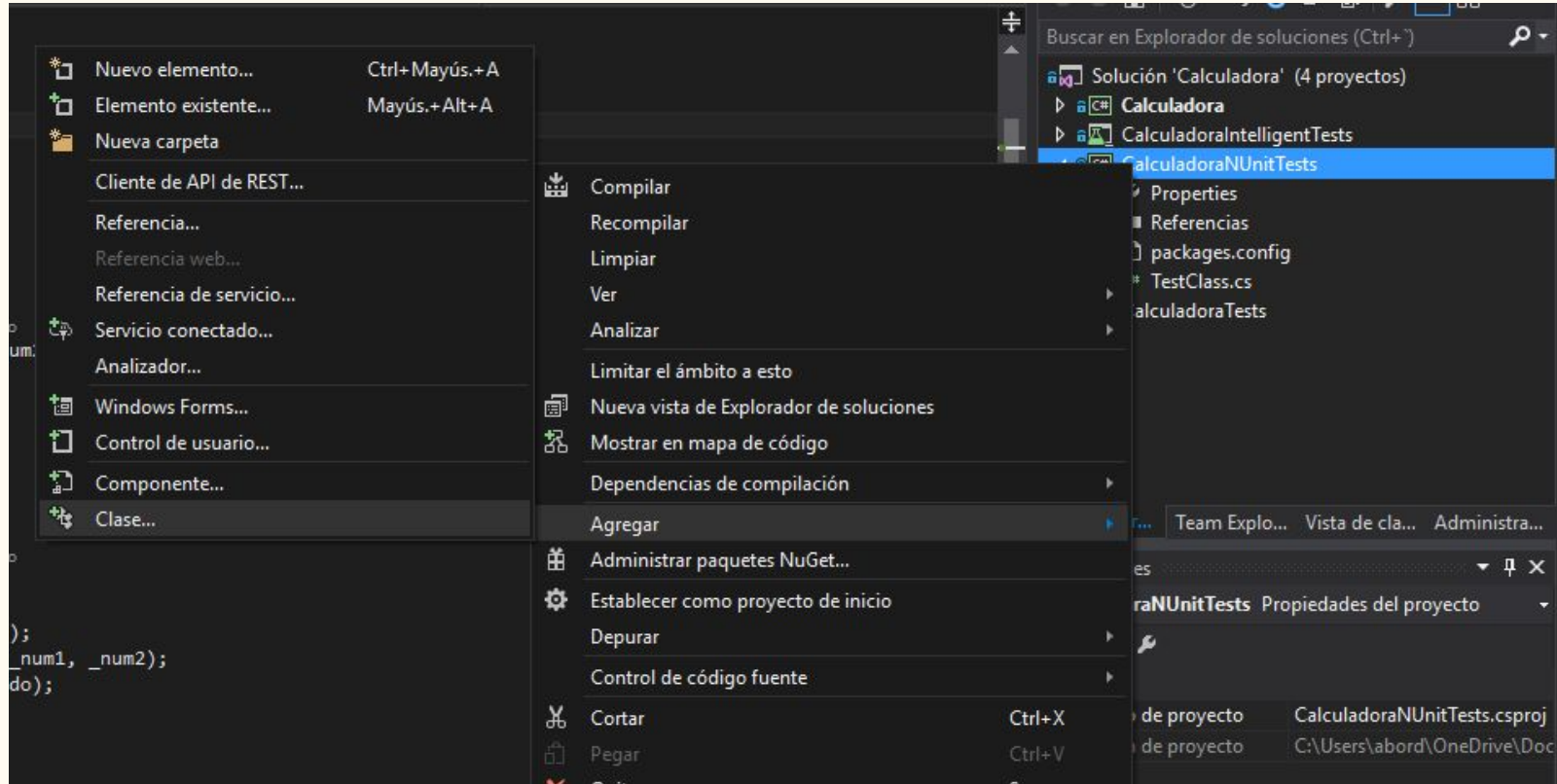
Prueba sin fallos



Prueba errónea

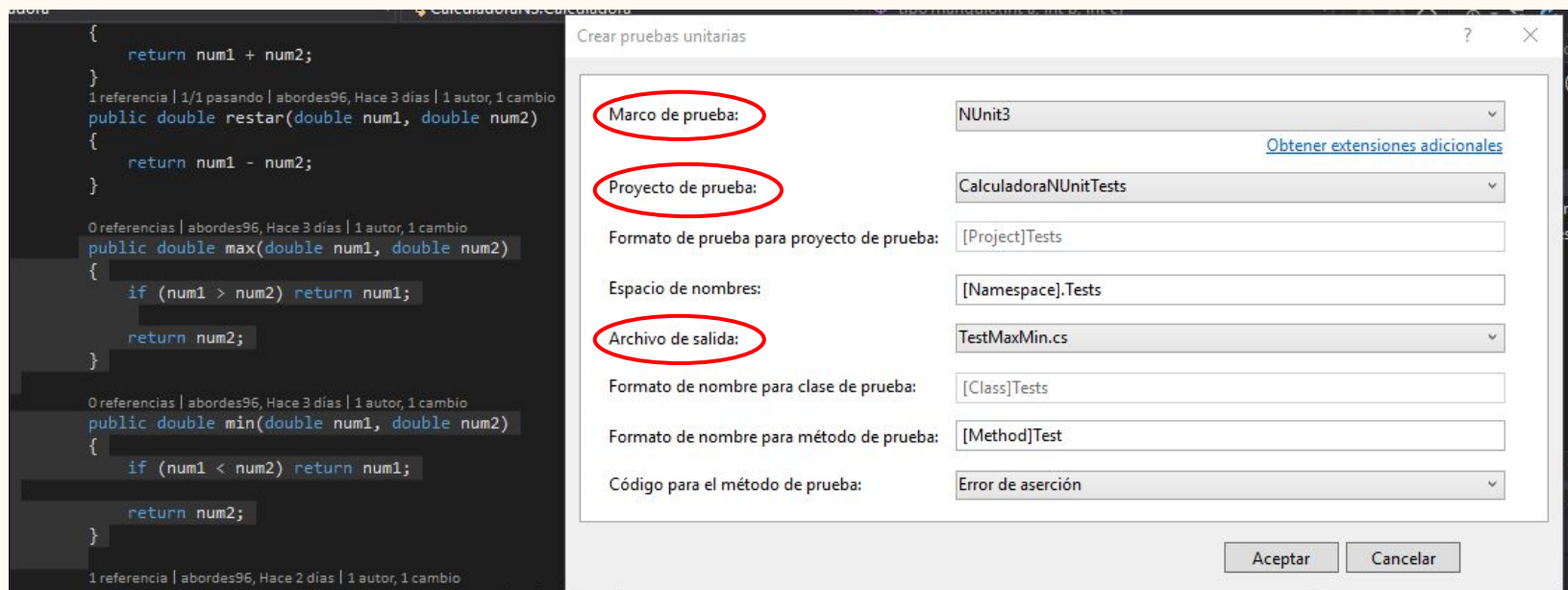
# NUnit 3

- Creación de una nueva clase de pruebas



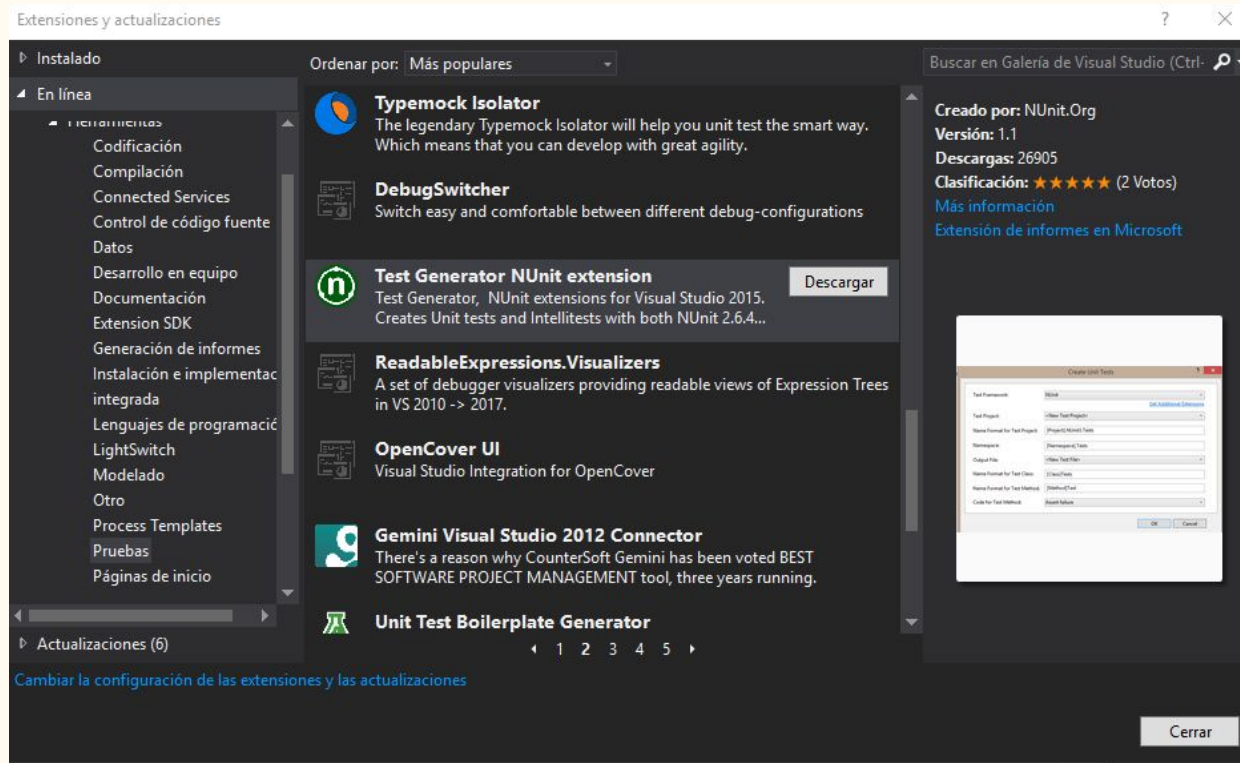
# NUnit 3

- Generación de la plantilla de las pruebas



# NUnit 3

- Descargar la extensión  
*Test Generator NUnit*



# NUnit 3

## ❖ Tests parametrizados

- Los tests son parametrizados mediante el paso de los valores de prueba directamente en el argumento de cada test.
- *Sequential* sirve para que al ejecutar los test se realicen únicamente en el orden de escritura de las parejas de argumentos, sin hacer combinaciones extras.

```
namespace CalculadoraNS.Tests
{
    [TestFixture()]
    0 referencias | 0 cambios | 0 autores, 0 cambios
    public class TestMaxMin
    {

        [Test()]
        [Sequential]
        0 referencias | 0 cambios | 0 autores, 0 cambios
        public void maxTest([Values(1, 2, 3)] int num1, [Values(4, 5)] int num2, [Values(4, 5, 3)] int resultado)
        {
            var calculadora = new Calculadora();
            Assert.AreEqual(resultado, calculadora.max(num1, num2));
        }

        [Test()]
        [Sequential]
        0 referencias | 0 cambios | 0 autores, 0 cambios
        public void minTest([Values(1, 2, 3)] int num1, [Values(4, 5)] int num2, [Values(1, 2, null)] int resultado)
        {
            var calculadora = new Calculadora();
            Assert.AreEqual(resultado, calculadora.min(num1, num2));
        }
    }
}
```

# NUnit 3

- Resultados al ejecutar los tests en el explorador de pruebas



A screenshot of the NUnit test runner interface. It shows a tree view with a folder icon and the text 'TestMaxMin (6)'. Below this, there are six test entries, each preceded by a green checkmark icon. The test names are: 'maxTest(1,4,4)', 'maxTest(2,5,5)', 'maxTest(3,null,3)', 'minTest(1,4,1)', 'minTest(2,5,2)', and 'minTest(3,null,null)'.

```
TestMaxMin (6)  
  ✓ maxTest(1,4,4)  
  ✓ maxTest(2,5,5)  
  ✓ maxTest(3,null,3)  
  ✓ minTest(1,4,1)  
  ✓ minTest(2,5,2)  
  ✓ minTest(3,null,null)
```

# NUnit 3

- Por cada *TestCase* se crea una instancia con los valores pasados como parámetros, además se puede especificar cual es el resultado esperado y un nombre para la prueba.

```
[TestCase(3, 4, ExpectedResult = 3, TestName = "min_3_4_should_be_3")]  
[TestCase(2, 4, ExpectedResult = 2)]  
[TestCase(5, 14, ExpectedResult = 2, TestName = "min_5_14_should_be_5")]  
0 referencias | 0 cambios | 0 autores, 0 cambios  
public double minTestCases(int num1, int num2)  
{  
    var calculadora = new Calculadora();  
    return calculadora.min(num1, num2);  
}
```



# NUnit 3

- Resultados de la ejecución de los tests

The screenshot displays the NUnit test runner interface. At the top, a tree view shows the test suite 'TestMaxMin (0)' with a list of tests. The test 'min\_5\_14\_should\_be\_5' is marked with a red 'X' and is circled in red. Below it, several other tests are marked with green checkmarks and are also circled in red. The bottom section provides a detailed view of the failed test 'min\_5\_14\_should\_be\_5', showing the origin as 'TestMaxMin.cs línea 35' and the error message: 'Pruebas no superadas - min\_5\_14\_should\_be\_5' with 'Mensaje: Expected: 2' and 'But was: 5.0d'. The execution time for this test is 17 ms.

Test Name	Result	Time
min_5_14_should_be_5	Failed	17 ms
maxTest(1,4,4)	Passed	31 ms
maxTest(2,5,5)	Passed	< 1 ms
maxTest(3,null,3)	Passed	< 1 ms
min_3_4_should_be_3	Passed	< 1 ms
minTest(1,4,1)	Passed	< 1 ms
minTest(2,5,2)	Passed	< 1 ms
minTest(3,null,null)	Passed	< 1 ms
minTestCases(2,4)	Passed	< 1 ms

**min\_5\_14\_should\_be\_5**  
Origen: [TestMaxMin.cs línea 35](#)  
Pruebas no superadas - min\_5\_14\_should\_be\_5  
Mensaje: Expected: 2  
But was: 5.0d  
Tiempo transcurrido: 17 ms

# Cobertura

# Cobertura

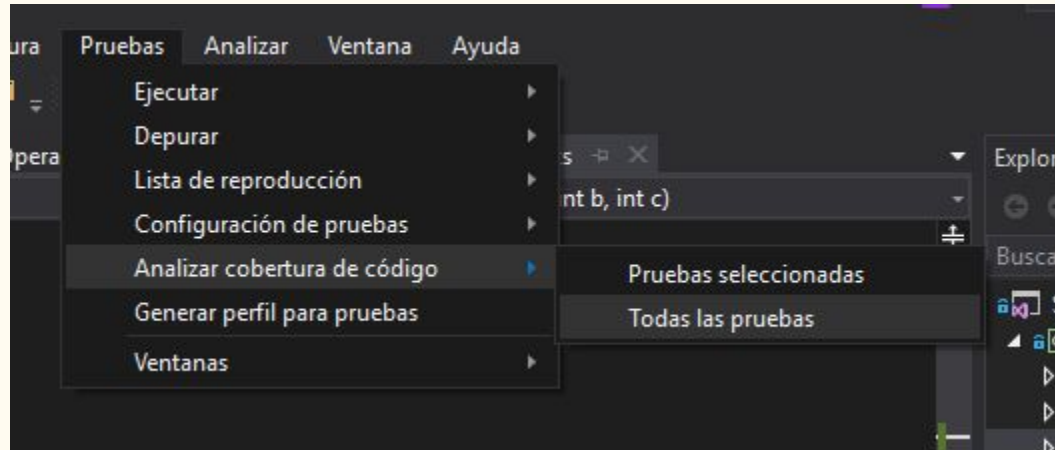
- Método y prueba creados

```
1 referencia | 1/1 pasando | 0 cambios | 0 autores, 0 cambios  
public int metodoSinCoberturaCompleta(int i)  
{  
    if (i > 0) return 1;  
    else return -1;  
}
```

```
[TestMethod()]  
✔ | 0 referencias | 0 cambios | 0 autores, 0 cambios  
public void metodoSinCoberturaCompletaTest()  
{  
    var calculadora = new Calculadora();  
    var resultado = calculadora.metodoSinCoberturaCompleta(10);  
    Assert.AreEqual(1, resultado);  
}
```

# Cobertura

- Ejecución del análisis de la cobertura



# Cobertura

- Análisis de la cobertura generado

Jerarquía	No cubiertos (bloques)	No cubiertos (% de bloques)	Cubiertos (bloques)	Cubiertos (% de bloques)
abord_LAPTOP-5O6618A6 2017...	1	1,19 %	83	98,81 %
└─ calculadora.dll	1	2,78 %	35	97,22 %
└─ { } CalculadoraNS	1	2,78 %	35	97,22 %
└─ └─ Calculadora	1	2,78 %	35	97,22 %
└─ └─ └─ Sumar(double, d...	0	0,00 %	2	100,00 %
└─ └─ └─ max(double, dou...	0	0,00 %	4	100,00 %
└─ └─ └─ metodoSinCober...	1	25,00 %	3	75,00 %
└─ └─ └─ min(double, dou...	0	0,00 %	4	100,00 %
└─ └─ └─ restar(double, do...	0	0,00 %	2	100,00 %
└─ └─ └─ tipoTriangulo(int,...	0	0,00 %	20	100,00 %
└─ calculadoraintellitests.dll	0	0,00 %	18	100,00 %
└─ calculadoranunittests.dll	0	0,00 %	18	100,00 %
└─ calculadoratests.dll	0	0,00 %	12	100,00 %

# Cobertura

- Si se selecciona la opción de mostrar color, aparecerán en azul las líneas de código probadas y en blanco las líneas sin probar.



```
1 referencia | 1/1 pasando | 0 cambios | 0 autores, 0 cambios  
public int metodoSinCoberturaCompleta(int i)  
{  
    if (i > 0) return 1;  
    else return -1;  
}
```

# Referencias

Extensión de github para visual studio: <https://visualstudio.github.com/>

Tutorial de integración de github con visual studio:

<https://blogs.msdn.microsoft.com/esmsdn/2015/05/25/integracin-de-github-con-visual-studio-2015/>

Creación de tests de unidad:

<https://www.visualstudio.com/en-us/docs/test/developer-testing/getting-started/getting-started-with-developer-testing>

Tutorial de testing: <https://msdn.microsoft.com/es-es/library/ms182532.aspx>

IntelliTest: <https://msdn.microsoft.com/es-es/library/dn823749.aspx>

Introducción a NUnit: <http://panicoenlaxbox.blogspot.com.es/2014/02/introduccion-nunit-i.html>

Atributos de NUnit <http://panicoenlaxbox.blogspot.com.es/2014/02/introduccion-nunit-ii-atributos-basicos.html>

Ciclo de vida de los tests en NUnit [http://panicoenlaxbox.blogspot.com.es/2014/02/introduccion-nunit-iii-ciclo-de-vida-de 21.html](http://panicoenlaxbox.blogspot.com.es/2014/02/introduccion-nunit-iii-ciclo-de-vida-de-21.html)

Tests parametrizados en NUnit <http://panicoenlaxbox.blogspot.com.es/2014/04/introduccion-nunit-iv-tests.html>

# Referencias

Tipos de clases Assert: <https://msdn.microsoft.com/es-es/library/ms182530.aspx>

Clase Assert: <https://msdn.microsoft.com/es-es/library/microsoft.visualstudio.testtools.unittesting.assert.aspx>

Añadir marcos de prueba: <https://msdn.microsoft.com/library/hh598952.aspx>

Test Fixture: <https://www.nunit.org/index.php?p=testFixture&r=2.5>

Test Case: <http://nunit.org/?p=testCase&r=2.5>

Cobertura: <https://msdn.microsoft.com/es-es/library/dd537628.aspx>

Repositorio público con el código utilizado y la presentación: <https://github.com/abordes96/testing-vs>