

Pruebas unitarias con Visual Studio .Net



César Portero Pestaña.
1ºDAW.
Entornos de Desarrollo.

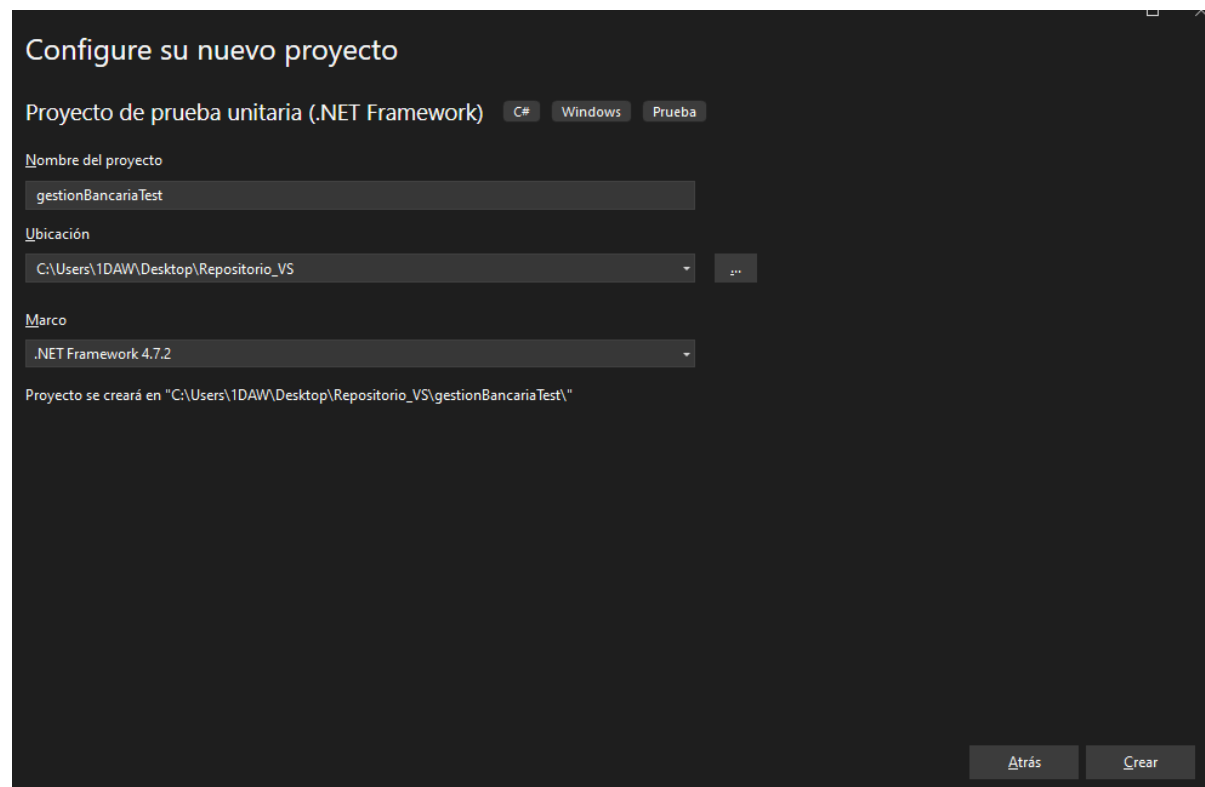
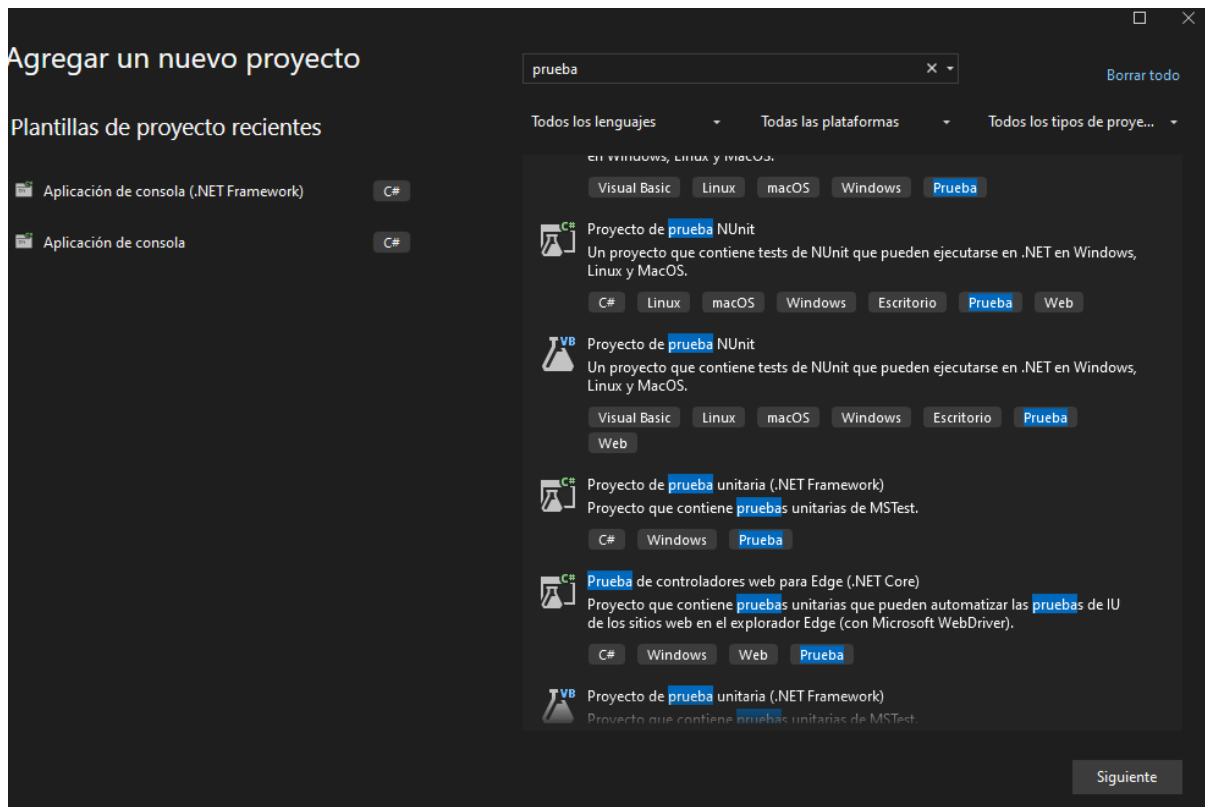
1. Crear un proyecto de prueba unitaria:.....	2
2. Crear la clase de prueba.....	3
3. Crear el primer método de prueba.....	4
4. Compilar y ejecutar la prueba.....	5
• ACTIVIDAD 1:.....	6

César Portero Pestaña.

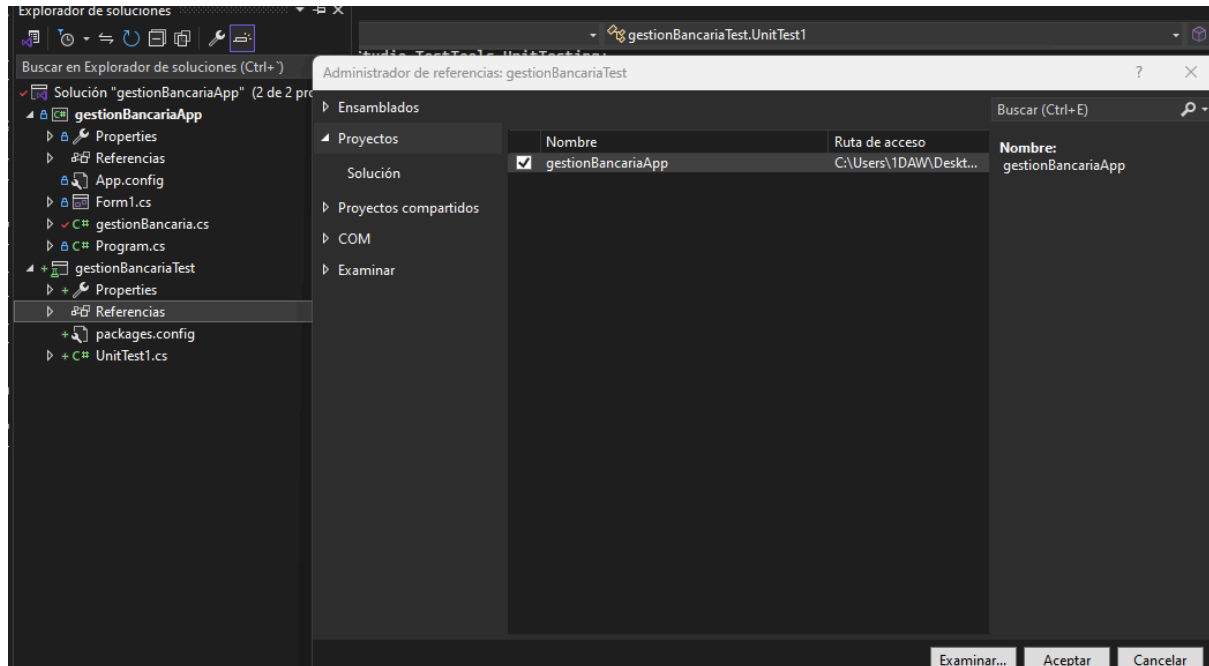
1ºDAW.

Entornos de Desarrollo.

1. Crear un proyecto de prueba unitaria:



César Portero Pestaña.
1ºDAW.
Entornos de Desarrollo.



2. Crear la clase de prueba

Le hemos cambiado el nombre al caso de prueba al indicado en el código.

Después de aceptar el cambio de nombre aparece el código con el nombre actualizado.

```
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  using System;
3
4  namespace gestionBancariaTest
5  {
6      [TestClass]
7      public class gestionBancariaTests
8      {
9          [TestMethod]
10         public void TestMethod1()
11         {
12         }
13     }
14 }
15
```

- Agregar una instrucción using al proyecto en pruebas

```
1  using Microsoft.VisualStudio.TestTools.UnitTesting;
2  {
3      using gestionBancariaApp;
4      using System;
5  }
6
7  namespace gestionBancariaTest
8  {
9      [TestClass]
10     0 referencias
11     public class gestionBancariaTests
12     {
13         [TestMethod]
14         0 referencias
15         public void TestMethod1()
16         {
17         }
18     }
19 }
```

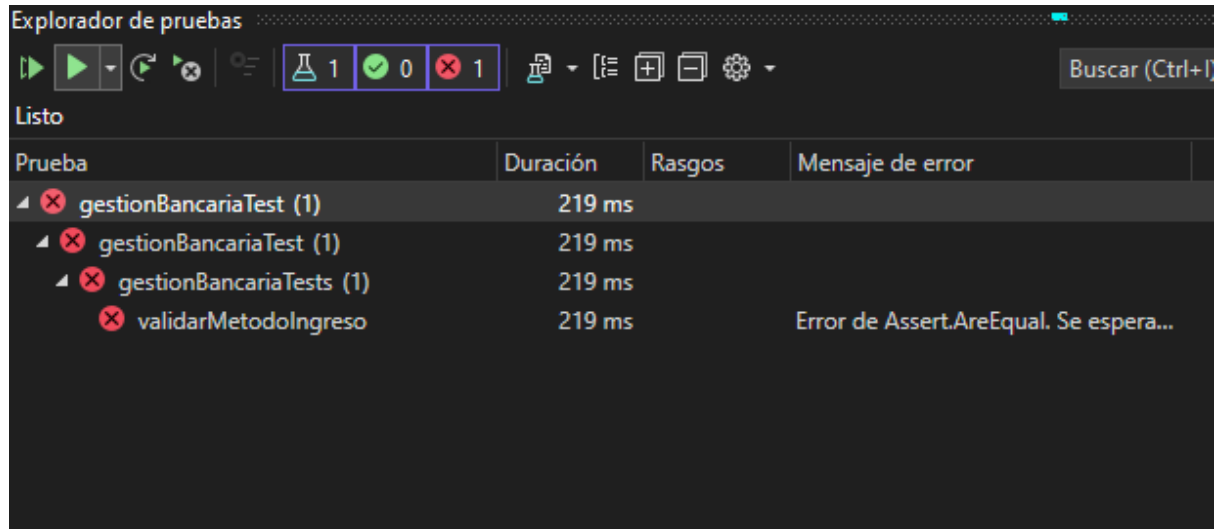
3. Crear el primer método de prueba.

- Vamos a crear un primer caso de prueba siguiendo estos pasos:
 1. Agregue una instrucción using gestionBancariaApp; al archivo gestionBancariaTest.cs. (Está en el código de la captura de arriba)
 2. Agregue el siguiente método a esa clase gestionBancariaTest.cs

```
9  [TestClass]
10  0 referencias
11  public class gestionBancariaTests
12  {
13      [TestMethod]
14      0 referencias
15      public void validarMetodoIngreso(){
16
17          double saldoInicial = 1000;
18          double ingreso = 500;
19          double saldoActual = 0;
20          double saldoEsperado = 1500;
21
22          gestionBancaria cuenta = new gestionBancaria(saldoInicial);
23
24          cuenta.realizarIngreso(ingreso);
25
26          saldoActual = cuenta.obtenerSaldo();
27
28          Assert.AreEqual(saldoEsperado, saldoActual, 0.001, "El saldo de la cuenta no es correcto");
29      }
30  }
```

4. Compilar y ejecutar la prueba.

- Para compilar y ejecutar la prueba.



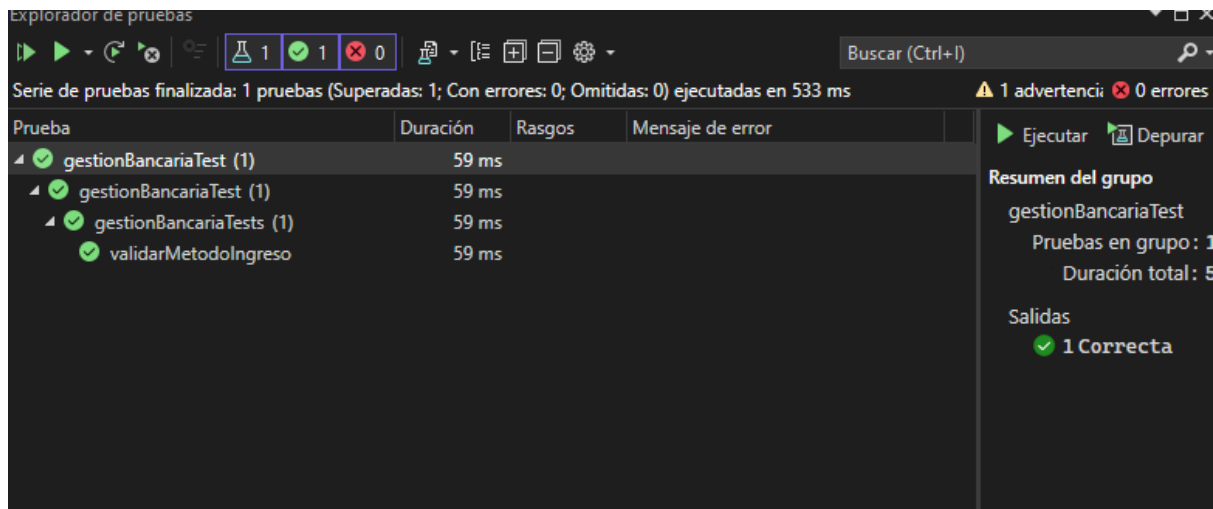
Al ejecutar la prueba da error ya que no se recibe el resultado esperado por la prueba planteada ya que nuestro código tiene un error.

- Corregir el error.

```
public void realizarIngreso(double cantidad)
{
    if (cantidad < 0)
    {
        mostrarError(ERR_CANTIDAD_INDICADA_NEGATIVA);
    }
    else
    {
        if (cantidad > 0)
        {
            saldo += cantidad; //Esto es un error, resta el saldo en lugar de sumarlo
        }
    }
}
```

Corregimos el error del código que hacía que al ingresar se restará al valor del saldo en lugar de sumar.

- Vuelva a ejecutar la prueba



Una vez corregido, volvemos a lanzar las pruebas y obtenemos el resultado esperado.

● ACTIVIDAD 1:

1. Clases de equivalencia:

a. realizarIngreso

- Cantidad válida: Cantidad mayor que 0.
- Cantidad negativa: Cantidad menor que 0.

Caso	Entrada	Resultado esperado
1	100.0	Incrementa el saldo correctamente.
2	-50.0	Genera una excepción.

b. realizarReintegro

- Cantidad válida y saldo suficiente: Cantidad mayor que 0 y menor o igual al saldo.
- Cantidad negativa: Cantidad menor que 0.
- Saldo insuficiente: Cantidad mayor al saldo disponible.

Caso	Entrada	Saldo Inicial	Resultado esperado
1	50.0	100.0	Incrementa el saldo correctamente.
2	-30.0	100.0	Genera una excepción.
3	100.0	100.0	Genera una excepción.

2. Código implementado:

```
[TestMethod]
| 0 referencias
public void realizarIngreso_Valido()
{
    gestionBancaria cuenta = new gestionBancaria(100.0);
    cuenta.realizarIngreso(50.0);
    Assert.AreEqual(150.0, cuenta.obtenerSaldo());
}

[TestMethod]
| 0 referencias
public void realizarIngreso_CantidadNegativa()
{
    gestionBancaria cuenta = new gestionBancaria(100.0);
    cuenta.realizarIngreso(-50.0);
}

[TestMethod]
| 0 referencias
public void realizarReintegro_SaldoInsuficiente()
{
    gestionBancaria cuenta = new gestionBancaria(100.0);
    cuenta.realizarReintegro(150.0);
}
```


César Portero Pestaña.
1ºDAW.
Entornos de Desarrollo.

Resultado de las pruebas:

The screenshot shows the 'Explorador de pruebas' (Test Explorer) window. At the top, it indicates 'Serie de pruebas finalizada: 4 pruebas (Superadas: 4; Con errores: 0; Omitidas: 0) ejecutadas en 5,9 s'. Below this is a table with columns: 'Prueba', 'Duración', 'Rasgos', and 'Mensaje de error'. The table lists the following tests:

Prueba	Duración	Rasgos	Mensaje de error
gestionBancariaTest (4)	5,4 s		
gestionBancariaTest (4)	5,4 s		
gestionBancariaTests (4)	5,4 s		
realizarIngreso_CantidadNegati...	4,1 s		
realizarIngreso_Valido	1 ms		
realizarReintegro_SaldoInsuficie...	1,2 s		
validarMetodoIngreso	1 ms		

On the right side, the 'Resumen del grupo' (Group Summary) shows 'gestionBancariaTest' with 'Pruebas en grupo: 4' and 'Duración total: 5,9 s'. Below this, the 'Salidas' (Outputs) section shows '4 Correcta' (4 Correct).

● ACTIVIDAD 2:

- Código modificado:

```
2 referencias | 1/1 pasando
public void realizarReintegro(double cantidad)
{
    if (cantidad <= 0)
    {
        throw new ArgumentOutOfRangeException("La cantidad indicada es negativa.");
    }
    else if (cantidad > saldo)
    {
        throw new ArgumentOutOfRangeException("Saldo insuficiente para realizar el reintegro.");
    }
    else
    {
        saldo -= cantidad;
    }
}

4 referencias | 3/3 pasando
public void realizarIngreso(double cantidad)
{
    if (cantidad < 0)
    {
        throw new ArgumentOutOfRangeException("La cantidad indicada es negativa.");
    }
    else
    {
        saldo += cantidad; // Corrección del error intencionado
    }
}
```

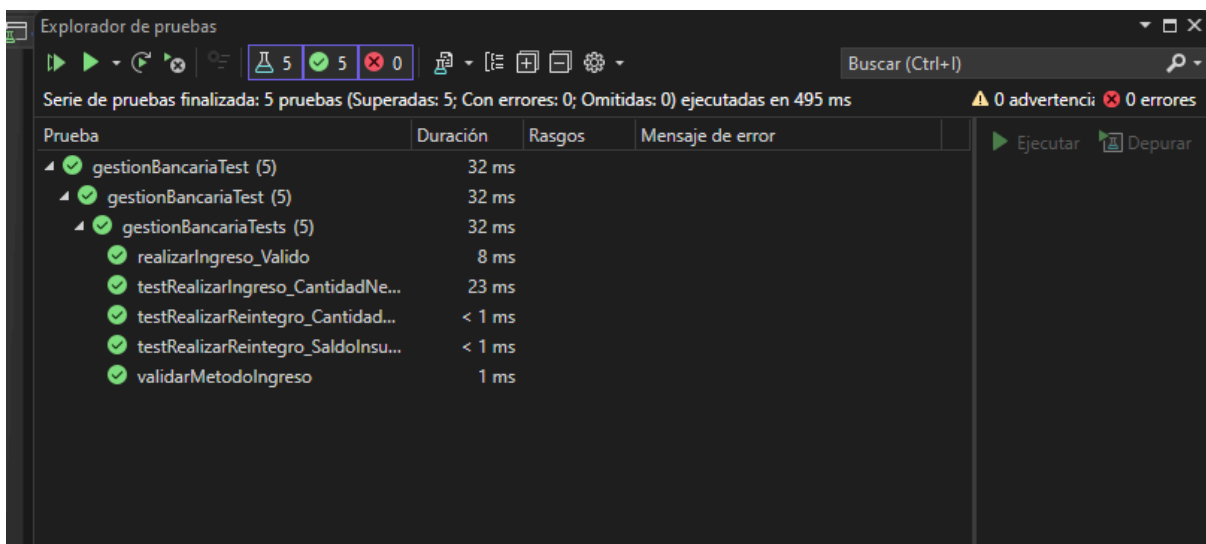
- Pruebas modificadas:

```
[TestMethod]
[ExpectedException(typeof(ArgumentOutOfRangeException))]
public void testRealizarIngreso_CantidadNegativa()
{
    gestionBancaria cuenta = new gestionBancaria(100.0);
    cuenta.realizarIngreso(-50.0);
}

[TestMethod]
[ExpectedException(typeof(ArgumentOutOfRangeException))]
public void testRealizarReintegro_CantidadNegativa()
{
    gestionBancaria cuenta = new gestionBancaria(100.0);
    cuenta.realizarReintegro(-30.0);
}

[TestMethod]
[ExpectedException(typeof(ArgumentOutOfRangeException))]
public void testRealizarReintegro_SaldoInsuficiente()
{
    gestionBancaria cuenta = new gestionBancaria(100.0);
    cuenta.realizarReintegro(150.0);
}
```

- Pruebas finalizadas:



Prueba	Duración	Rasgos	Mensaje de error
gestionBancariaTest (5)	32 ms		
gestionBancariaTest (5)	32 ms		
gestionBancariaTests (5)	32 ms		
realizarIngreso_Valido	8 ms		
testRealizarIngreso_CantidadNe...	23 ms		
testRealizarReintegro_Cantidad...	< 1 ms		
testRealizarReintegro_SaldoInsu...	< 1 ms		
validarMetodoIngreso	1 ms		