

Formación de Testing en .NET

Nafarroako
Gobernua



Gobierno
de Navarra

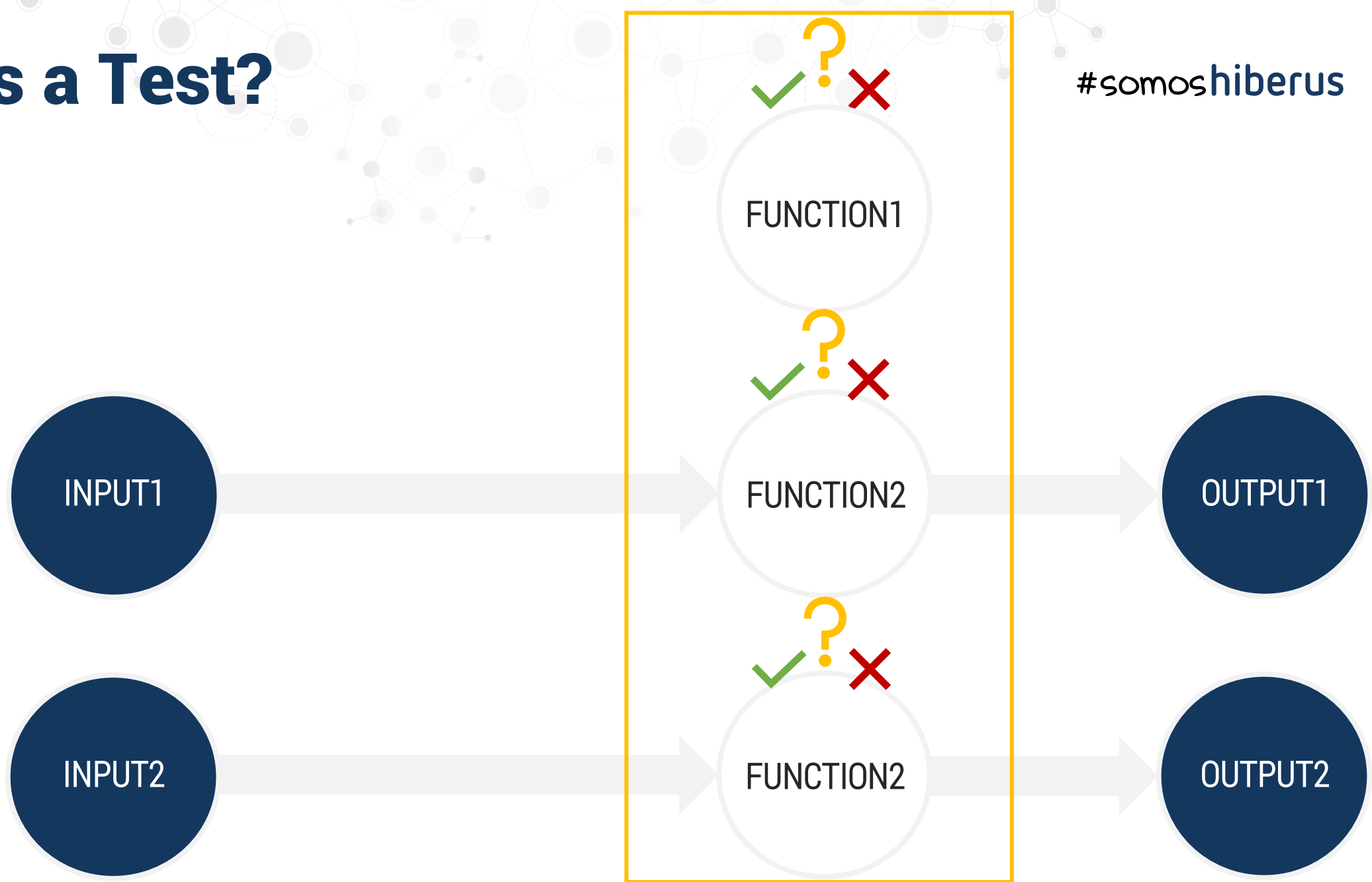
hiberus[©]

La compañía **hiperespecializada**
en las TIC

Tema 2.1: xUnit Avanzado

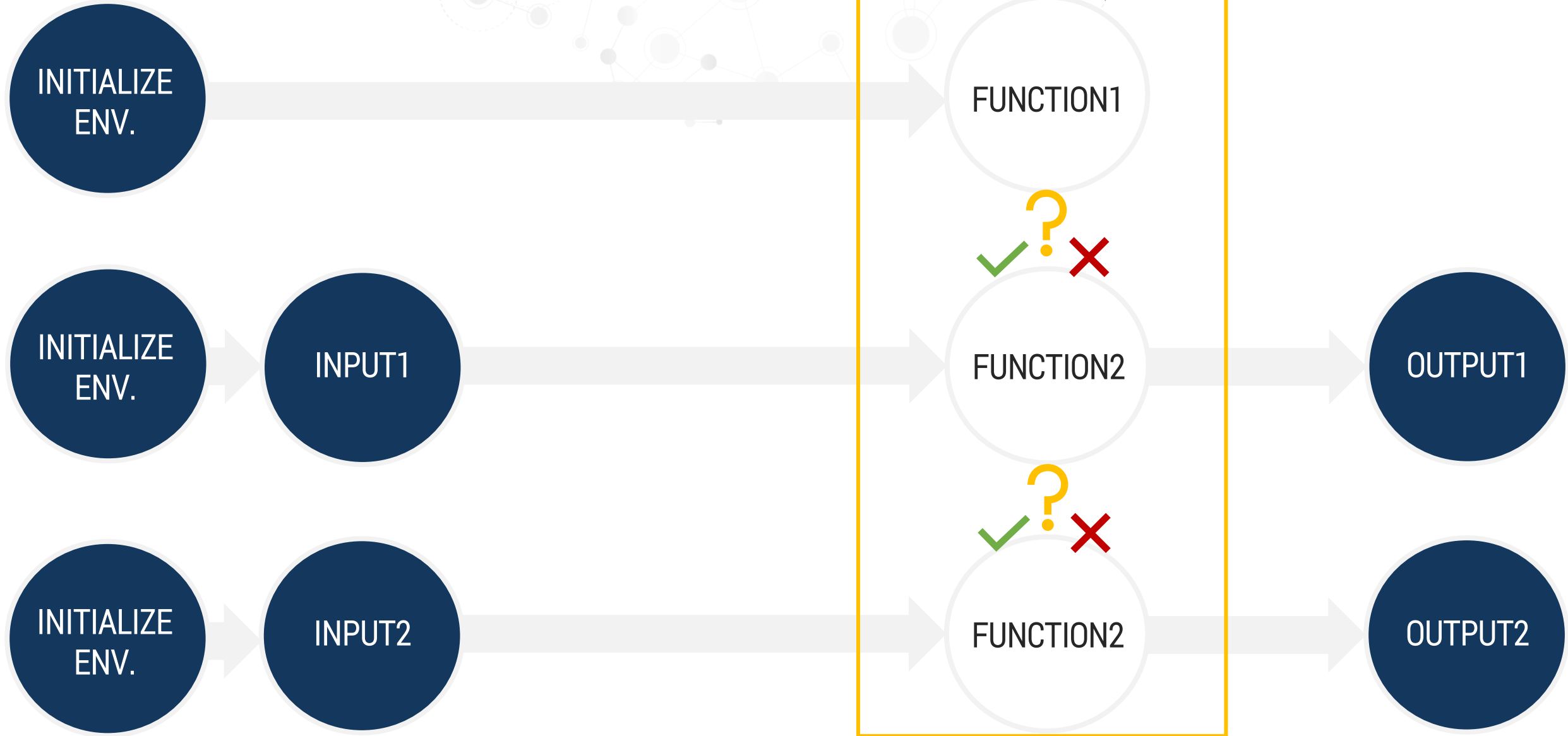
What is a Test?

#somoshiberus



What is a *Real-World* Test?

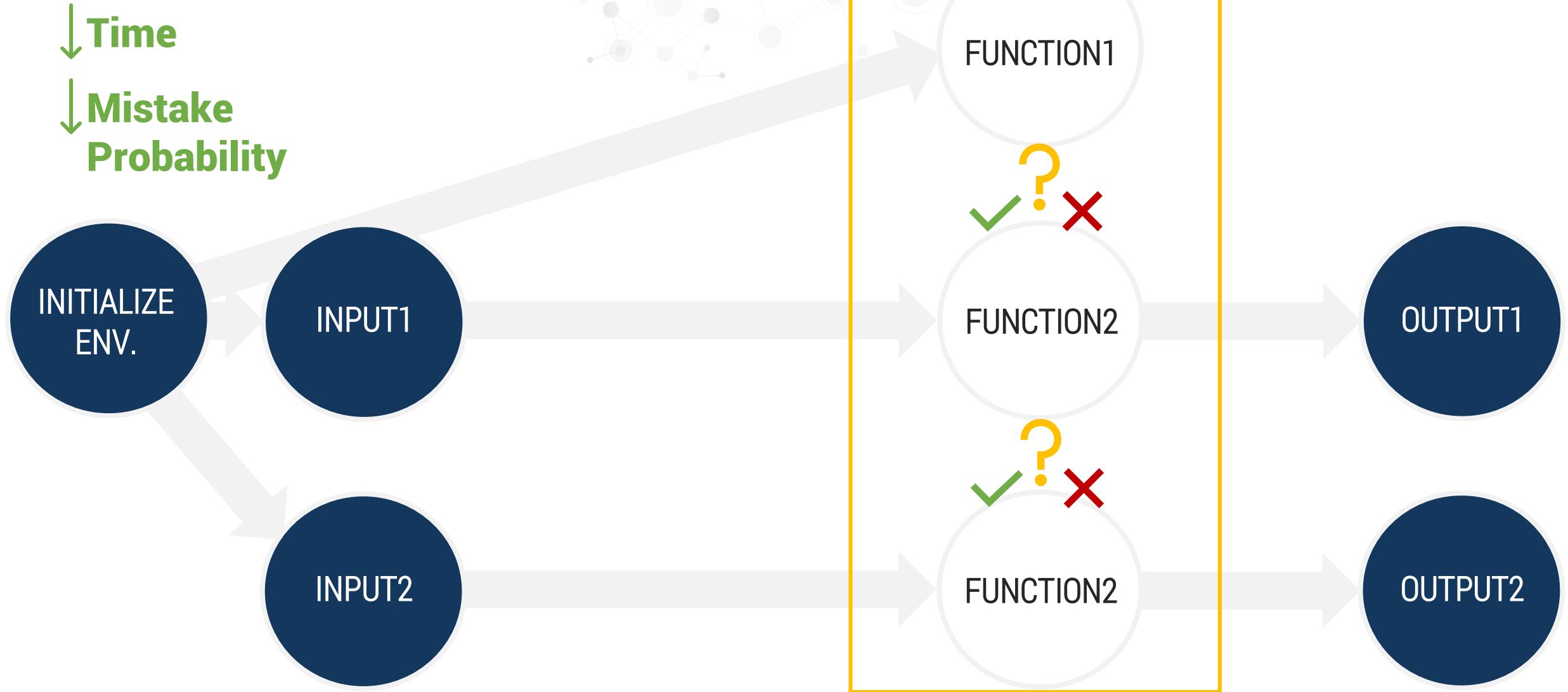
#somoshiberus



What is a *Real-World Test*?

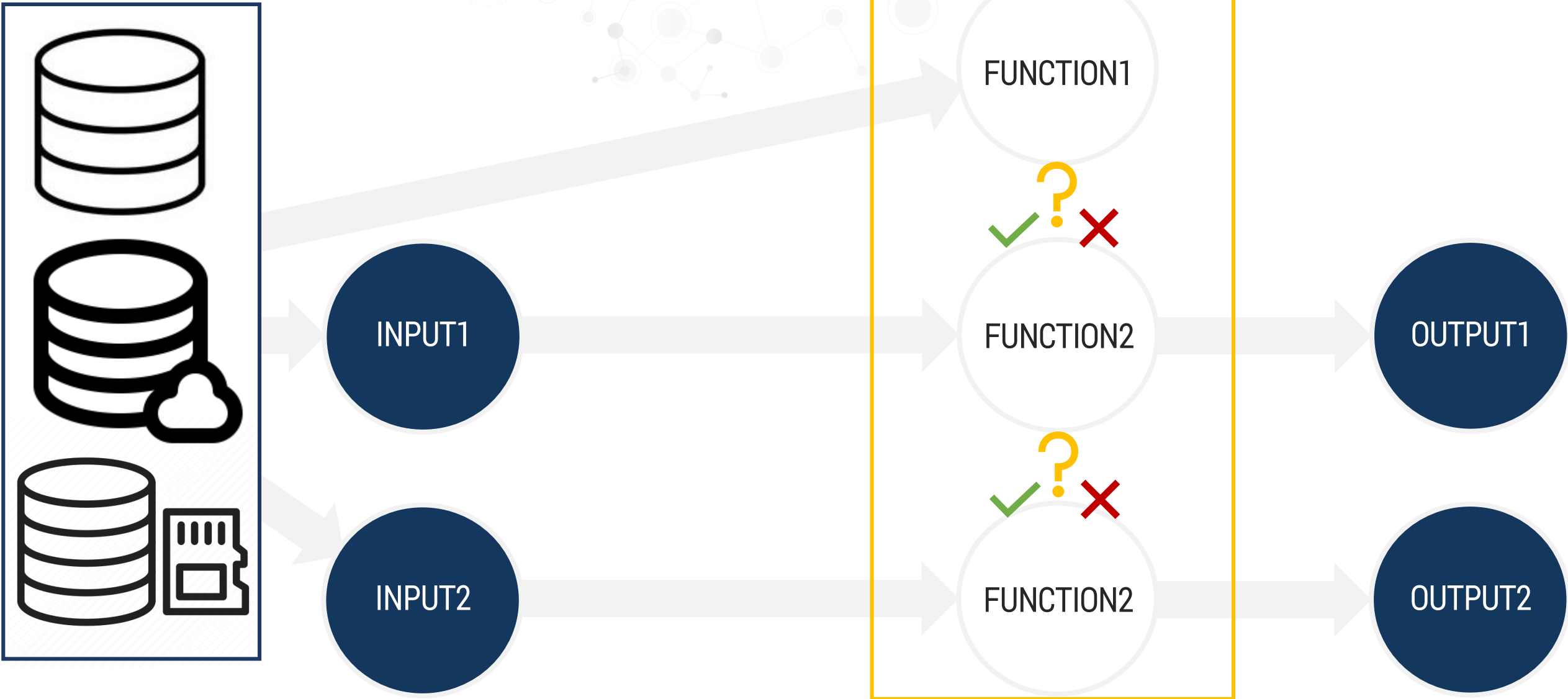
#somoshiberus

↓ Time
↓ Mistake
Probability



What is a *Real-World* Test?

#somoshiberus



What is a *Real-World* Test?

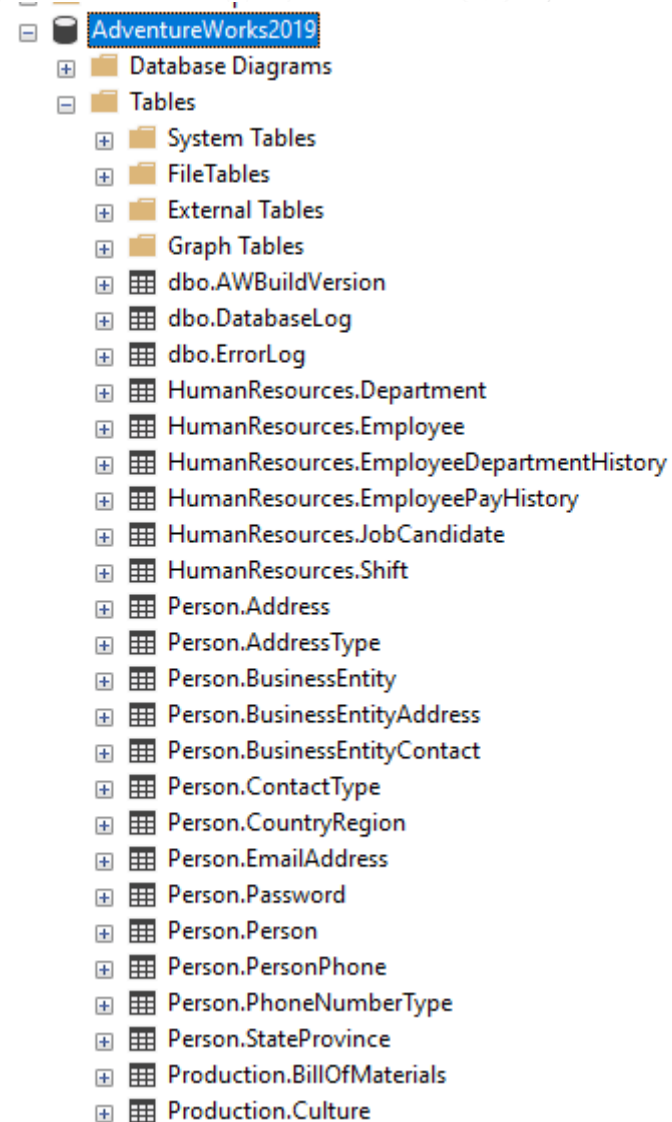
#somoshiberus



AdventureWorks



<https://docs.microsoft.com/es-es/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms>



- **Constructor And Dispose**
Shared setup/clean code without sharing object instances
- **Class Fixtures**
Shared object instances across tests in a single class
- **Collection Fixtures**
Shared object instances across multiple tests classes

Shared Context

Constructor & Dispose

#somoShiberus

```
public class EmployeesConstructorTest : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";

    7 referencias | 0/2 pasando
    public SqlConnection Db { get; private set; }

    0 referencias
    public EmployeesConstructorTest()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
    }
}
```

Shared Context

Constructor & Dispose

#somoshiberus

```
[Fact]
0 referencias
public void Employees_GetAll_CanExecute()
{
    Db.Open();
    SqlCommand command = Db.CreateCommand();
    command.CommandText = @"SELECT *
                            FROM HumanResources.Employee";
    var obj = command.ExecuteScalar();
    Assert.NotNull(obj);
    Db.Close();
}
```

```
1 referencia
public class EmployeesConstructorTest : IDisposable
{
    private const string cn = "Server=localhost;Database=
                              "Integrated Security=SSPI;Tru

7 referencias | 1/2 pasando
public SqlConnection Db { get; private set; }

0 referencias
public EmployeesConstructorTest()
{
    Db = new SqlConnection(cn);
    // ... initialize data in the test database ...
}

0 referencias
public void Dispose()
{
    // ... clean up test data from the database ...
}
```

Entities_GetAll_NonEmpty	000 ms
EmployeesConstructorTest (2)	11,9 s
Employees_GetAll_CanExecute	11,9 s

Shared Context

Constructor & Dispose

#somoshiberus

```
[Fact]
✓ | 0 referencias
public void Employees_GetAll_CanExecute()
{
    Db.Open();
    SqlCommand command = Db.CreateCommand();
    command.CommandText = @"SELECT *
                            FROM HumanResources.Employee";
    var obj = command.ExecuteScalar();
    Db.Close();
}
```

```
[Fact]
❗ | 0 referencias
public void Employees_GetAll_NonEmpty()
{
    Db.Open();
    SqlCommand command = Db.CreateCommand();
    command.CommandText = @"SELECT *
                            FROM HumanResources.Employee";
    var obj = command.ExecuteScalar();
    Assert.NotNull(obj);
    Db.Close();
}
```

Shared Context

Constructor & Dispose

#somoshiberus

Executed once PER METHOD

```
public class EmployeesConstructorTest : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";

    7 referencias | ✔ 2/2 pasando
    public SqlConnection Db { get; private set; }

    0 referencias
    public EmployeesConstructorTest()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
    }
}
```

Shared Context

Constructor & Dispose

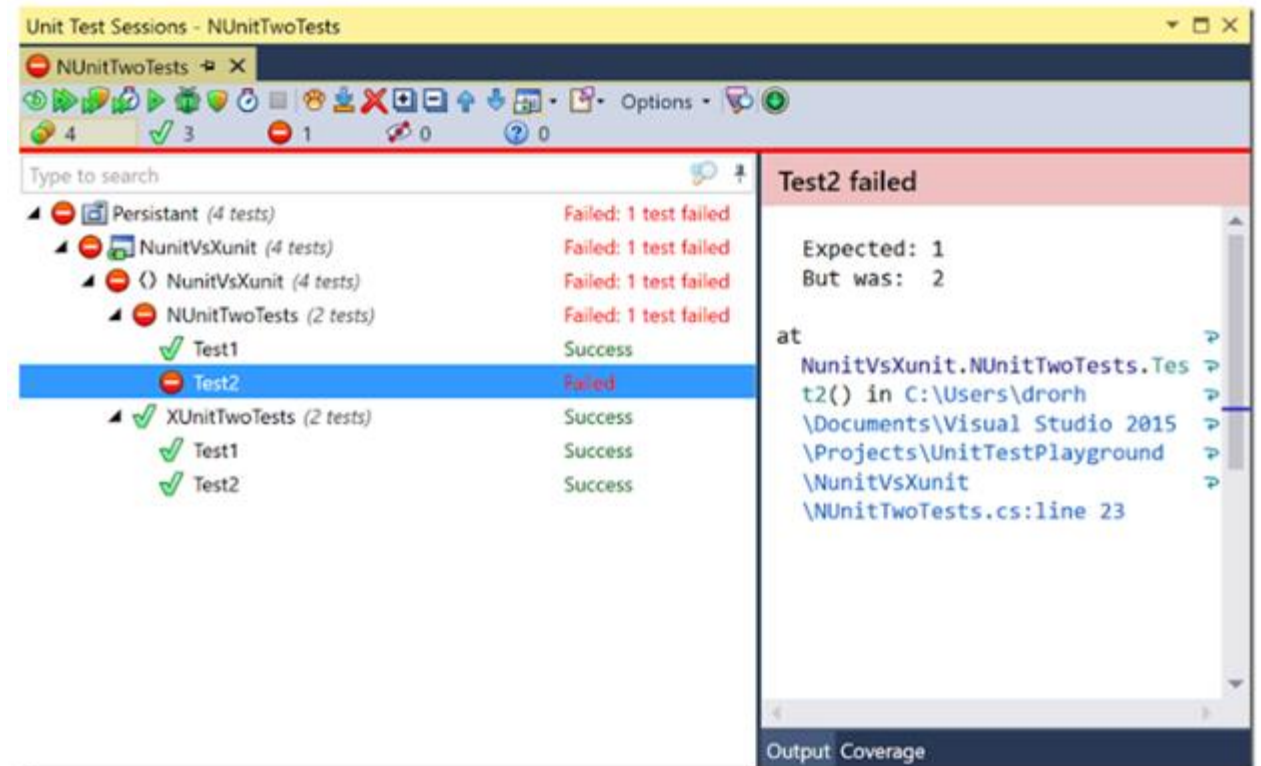
#somoshiberus

Executed once PER METHOD

```
public class XUnitInstanceTest
{
    private int a = 0;

    [Fact]
    0 | 0 referencias
    public void Test1()
    {
        a++;
        Assert.Equal(1, a);
    }

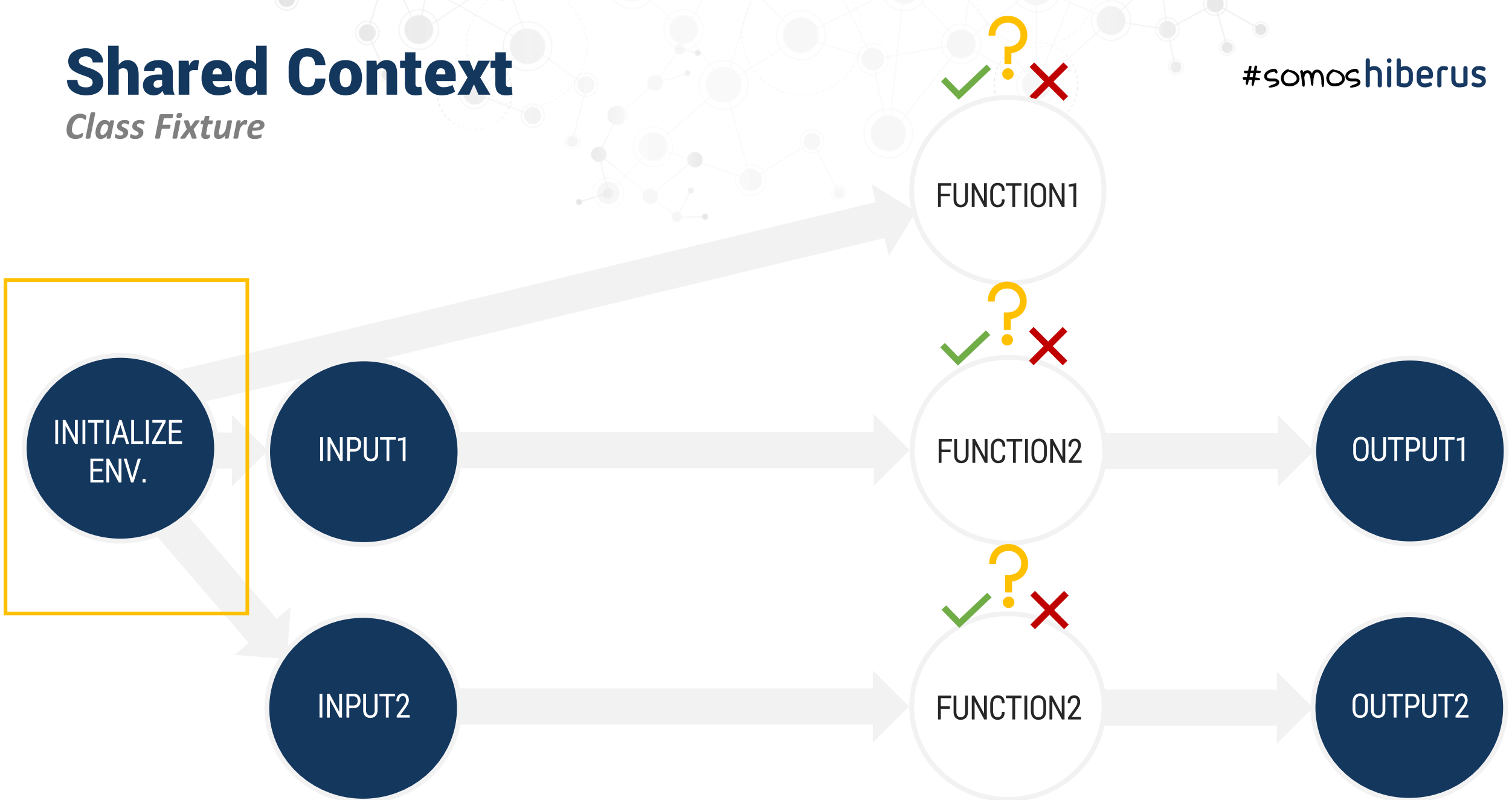
    [Fact]
    0 | 0 referencias
    public void Test2()
    {
        a++;
        Assert.Equal(1, a);
    }
}
```



Shared Context

Class Fixture

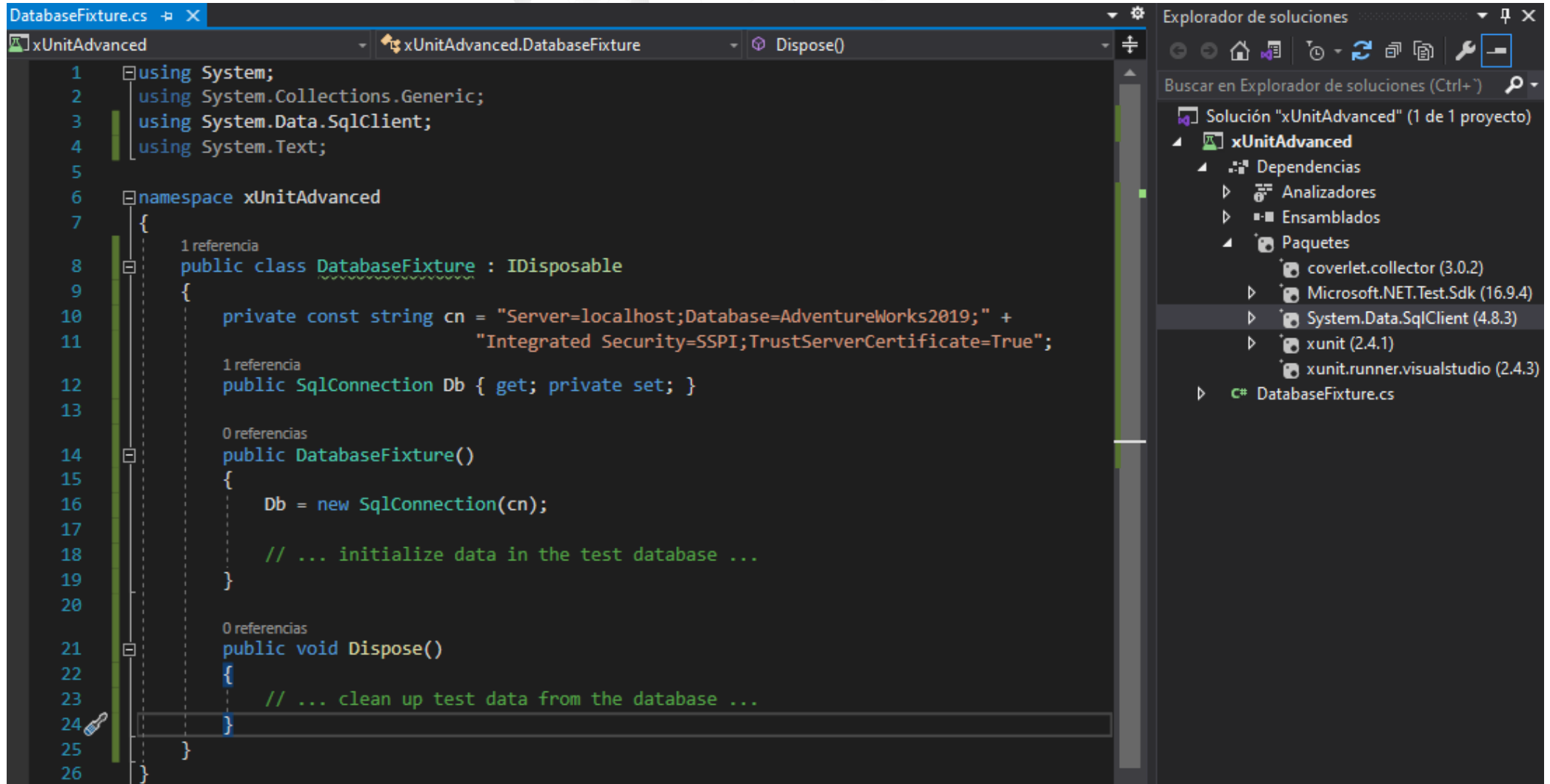
#somoshiiberus



Shared Context

Class Fixture

#somoshiberus



```
1  using System;
2  using System.Collections.Generic;
3  using System.Data.SqlClient;
4  using System.Text;
5
6  namespace xUnitAdvanced
7  {
8      1 referencia
9      public class DatabaseFixture : IDisposable
10     {
11         private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
12             "Integrated Security=SSPI;TrustServerCertificate=True";
13
14         1 referencia
15         public SqlConnection Db { get; private set; }
16
17         0 referencias
18         public DatabaseFixture()
19         {
20             Db = new SqlConnection(cn);
21
22             // ... initialize data in the test database ...
23         }
24
25         0 referencias
26         public void Dispose()
27         {
28             // ... clean up test data from the database ...
29         }
30     }
31 }
```

Explorador de soluciones

Buscar en Explorador de soluciones (Ctrl+)

Solución "xUnitAdvanced" (1 de 1 proyecto)

- xUnitAdvanced
 - Dependencias
 - Analizadores
 - Ensamblados
 - Paquetes
 - coverlet.collector (3.0.2)
 - Microsoft.NET.Test.Sdk (16.9.4)
 - System.Data.SqlClient (4.8.3)
 - xunit (2.4.1)
 - xunit.runner.visualstudio (2.4.3)
 - C# DatabaseFixture.cs

Shared Context

Class Fixture

#somoshiberus

```
1 referencia
public class EmployeesTest : IClassFixture<DatabaseFixture>
{
    private DatabaseFixture dbFixture;

    0 referencias
    public EmployeesTest(DatabaseFixture databaseFixture)
    {
        this.dbFixture = databaseFixture;
    }
}
```

Explorador de pruebas

1 1 0

Prueba	Duración
✓ xUnitAdvanced (1)	1,1 s
✓ xUnitAdvanced (1)	1,1 s
✓ EmployeesTest (1)	1,1 s
✓ Employees_GetAll_CanExecute	1,1 s

[Fact]

✓ | 0 referencias

```
public void Employees_GetAll_CanExecute()
{
    dbFixture.Db.Open();
    SqlCommand command = dbFixture.Db.CreateCommand();
    command.CommandText = @"SELECT *
                            FROM HumanResources.Employee";

    var obj = command.ExecuteScalar();
    Assert.NotNull(obj);
    dbFixture.Db.Close();
}
```


Shared Context

Class Fixture - DRY!

#somoshiberus

```
public class DBUtils
{
    1 referencia | 1/1 pasando
    public static object ExecuteScalar(SqlConnection db,
                                       string sqlQuery)
    {
        SqlCommand command = db.CreateCommand();
        command.CommandText = sqlQuery;

        return command.ExecuteScalar();
    }
}
```

```
public class EmployeesTest : IClassFixture<DatabaseFixture>
{
    private DatabaseFixture dbFixture;

    0 referencias
    public EmployeesTest(DatabaseFixture databaseFixture)
    {
        this.dbFixture = databaseFixture;
    }

    private const string tableName = "HumanResources.Employee";
    private const string getAllQuery = @"SELECT *
                                       FROM " + tableName;

    [Fact]
    0 referencias
    public void Employees_GetAll_CanExecute()
    {
        dbFixture.Db.Open();

        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        Assert.NotNull(obj);
        dbFixture.Db.Close();
    }
}
```

Shared Context

Class Fixture – Smaller tests

#somoshiberus

```
[Fact]
✓ | 0 referencias
public void Employees_GetAll_CanExecute()
{
    dbFixture.Db.Open();

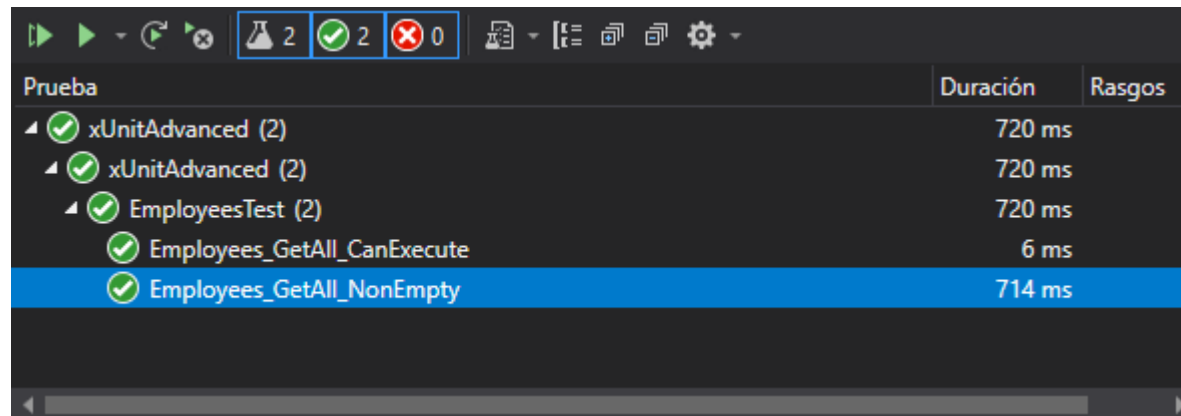
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    dbFixture.Db.Close();
}
```

```
[Fact]
ⓘ | 0 referencias
public void Employees_GetAll_NonEmpty()
{
    dbFixture.Db.Open();

    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    Assert.NotNull(obj);
    dbFixture.Db.Close();
}
```



Prueba	Duración	Rasgos
✓ xUnitAdvanced (2)	720 ms	
✓ xUnitAdvanced (2)	720 ms	
✓ EmployeesTest (2)	720 ms	
✓ Employees_GetAll_CanExecute	6 ms	
✓ Employees_GetAll_NonEmpty	714 ms	

Shared Context

Class Fixture

#somoshiberus

```
[Fact]
0 referencias
public void Employees_GetAll_CanExecute()
{
    dbFixture.Db.Open();

    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQue

    dbFixture.Db.Close();
}
```

Prueba

- ✗ xUnitAdvanced (4)
- ✗ xUnitAdvanced (4)
 - ⓘ DepartmentsTest (2)
 - ✗ EmployeesTest (2)
 - ✗ Employees_GetAll_CanExecute
 - ✓ Employees_GetAll_NonEmpty

Resumen de los detalles de la prueba

Employees_GetAll_CanExecute

línea 26

Exception : El nombre de objeto 'HumanResources.Employees' no es válido.

Exception exception, Boolean breakConnection, Action`1 wrapCloseInAction)

Shared Context

Class Fixture

#somoshiberus

Only executed once

```
8 public class DatabaseFixture : IDisposable
9 {
10     private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
11                               "Integrated Security=SSPI;TrustServerCertificate=True";
12     public SqlConnection Db { get; private set; }
13
14     public DatabaseFixture()
15     {
16         Db = new SqlConnection(cn);
17
18         // ... initialize data in the test database ...
19     }
20
21     public void Dispose()
22     {
23         // ... clean up test data from the database ...
24     }
25 }
```

Shared Context

Class Fixture

#somoshiberus

```
public class DepartmentsTest : IClassFixture<DatabaseFixture>
{
    private DatabaseFixture dbFixture;

    0 referencias
    public DepartmentsTest(DatabaseFixture databaseFixture)
    {
        this.dbFixture = databaseFixture;
    }
}
```

```

  xUnitAdvanced (2) 1,1 s
  xUnitAdvanced (2) 1,1 s
    DepartmentsTest (1) 409 ms
      Departments_GetAll_CanExecute 409 ms
    EmployeesTest (1) 682 ms
      Employees_GetAll_CanExecute 682 ms

private const string tableName = "HumanResources.Department";
private const string getAllQuery = @"SELECT *
                                   FROM " + tableName;

[Fact]
0 referencias
public void Departments_GetAll_CanExecute()
{
    dbFixture.Db.Open();

    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    Assert.NotNull(obj);
    dbFixture.Db.Close();
}
```

Shared Context

Class Fixture

#somoshiberus

Only executed once PER CLASS

```
8 public class DatabaseFixture : IDisposable
9 {
10     private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
11                               "Integrated Security=SSPI;TrustServerCertificate=True";
12     public SqlConnection Db { get; private set; }
13
14     public DatabaseFixture()
15     {
16         Db = new SqlConnection(cn);
17
18         // ... initialize data in the test database ...
19     }
20
21     public void Dispose()
22     {
23         // ... clean up test data from the database ...
24     }
25 }
```

Shared Context

Class Fixture - DRY!

#somoshiberus

```
public class EmployeesTest : IClassFixture<DatabaseFixture>
{
    private DatabaseFixture dbFixture;

    0 referencias
    public EmployeesTest(DatabaseFixture databaseFixture)
    {
        this.dbFixture = databaseFixture;
    }

    private const string tableName = "HumanResources.Employees";
    private const string getAllQuery = @"SELECT *
                                        FROM " + tableName;

    [Fact]
    0 referencias
    public void Employees_GetAll_CanExecute()
    {
        dbFixture.Db.Open();

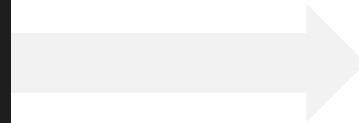
        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        dbFixture.Db.Close();
    }

    [Fact]
    0 referencias
    public void Employees_GetAll_NonEmpty()
    {
        dbFixture.Db.Open();

        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        Assert.NotNull(obj);
        dbFixture.Db.Close();
    }
}
```



```
public class DepartmentsTest : IClassFixture<DatabaseFixture>
{
    private DatabaseFixture dbFixture;

    0 referencias
    public DepartmentsTest(DatabaseFixture databaseFixture)
    {
        this.dbFixture = databaseFixture;
    }

    private const string tableName = "HumanResources.Department";
    private const string getAllQuery = @"SELECT *
                                        FROM " + tableName;

    [Fact]
    0 referencias
    public void Employees_GetAll_CanExecute()
    {
        dbFixture.Db.Open();

        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        dbFixture.Db.Close();
    }

    [Fact]
    0 referencias
    public void Employees_GetAll_NonEmpty()
    {
        dbFixture.Db.Open();

        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        Assert.NotNull(obj);
        dbFixture.Db.Close();
    }
}
```



Shared Context

Class Fixture - DRY!

#somoshiberus

```
public abstract class BaseEntityTest : IClassFixture<DatabaseFixture>
{
    protected DatabaseFixture dbFixture;

    1 referencia
    public BaseEntityTest(DatabaseFixture databaseFixture)
    {
        this.dbFixture = databaseFixture;
    }

    2 referencias
    protected abstract string tableName { get; }
    2 referencias
    private string getAllQuery => @"SELECT *
                                   FROM " + tableName;

    [Fact]
    0 referencias
    public void Entities_GetAll_CanExecute()
    {
        dbFixture.Db.Open();

        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        dbFixture.Db.Close();
    }

    [Fact]
    0 referencias
    public void Entities_GetAll_NonEmpty()
    {
        dbFixture.Db.Open();

        var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

        Assert.NotNull(obj);
        dbFixture.Db.Close();
    }
}
```

```
public class EmployeesWBaseEntityTest : BaseEntityTest, IClassFixture<DatabaseFixture>
{
    0 referencias
    public EmployeesWBaseEntityTest(DatabaseFixture databaseFixture) : base(databaseFixture) { }

    2 referencias
    protected override string tableName => "HumanResources.Employee";
}
```

Explorador de pruebas

6 4 0 2

Prueba

- Employees_GetAll_NonEmpty
- EmployeesWBaseEntityTest (2)
 - Entities_GetAll_CanExecute
 - Entities_GetAll_NonEmpty

EmployeesWBaseEntityTest

Pruebas en grupo: 2

Duración total: 623 ms

Salidas

2 Correcta

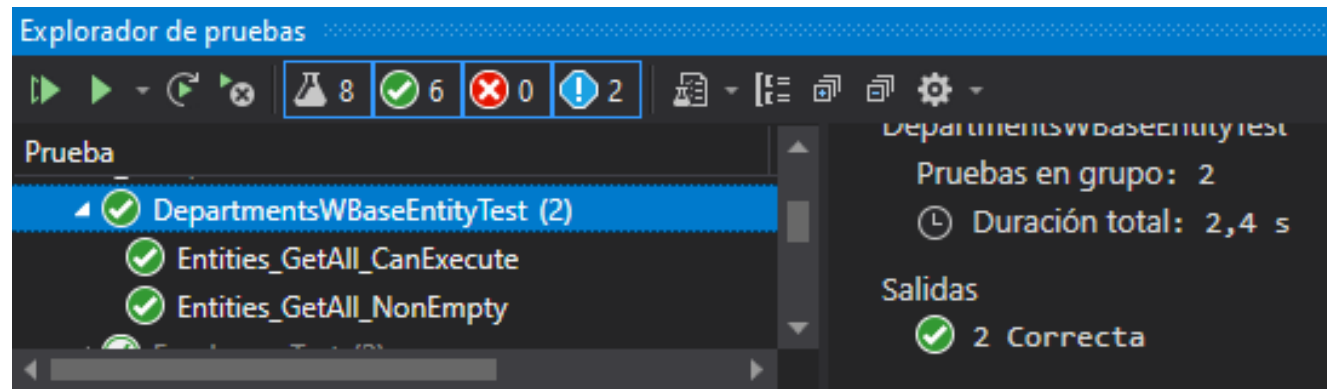
Shared Context

Class Fixture - DRY!

#somoshiberus

```
public class DepartmentsWBaseEntityTest : BaseEntityTest, IClassFixture<DatabaseFixture>
{
    0 referencias
    public DepartmentsWBaseEntityTest(DatabaseFixture databaseFixture) : base(databaseFixture) { }

    2 referencias
    protected override string tableName => "HumanResources.Department";
}
```



Shared Context

Class Fixture - Optimize Resources

#somoshiberus

```
[Fact]
0 referencias
public void Entities_GetAll_CanExecute()
{
    dbFixture.Db.Open();

    Assert.True(false);
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    dbFixture.Db.Close();
}
```

Explorador de pruebas

8 5 1 2

Prueba	Duración	Rasgos
DepartmentsTest (2)		
DepartmentsWBaseEntityTest (2)	2,4 s	
EmployeesTest (2)	1,4 s	
EmployeesWBaseEntityTest (2)	3 s	
Entities_GetAll_CanExecute	20 ms	
Entities_GetAll_NonEmpty	3 s	

Shared Context

Class Fixture - Optimize Resources

#somoshiberus

```
8 public class DatabaseFixture : IDisposable
9 {
10     private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
11                               "Integrated Security=SSPI;TrustServerCertificate=True";
12     public SqlConnection Db { get; private set; }
13
14     public DatabaseFixture()
15     {
16         Db = new SqlConnection(cn);
17
18         // ... initialize data in the test database ...
19     }
20
21     public void Dispose()
22     {
23         // ... clean up test data from the database ...
24     }
25 }
```

Shared Context

Class Fixture - Optimize Resources

#somoshiberus

```
public class DatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";

    20 referencias | ✔ 2/4 pasando
    public SqlConnection Db { get; private set; }

    0 referencias
    public DatabaseFixture()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
        Db.Close();
        Db.Dispose();
    }
}
```

Shared Context

Class Fixture - Optimize Resources

#somoshiberus

```
[Fact]
0 referencias
public void Entities_GetAll_CanExecute()
{
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);
}

[Fact]
0 referencias
public void Entities_GetAll_NonEmpty()
{
    var obj = DBUtils.ExecuteScalar(dbFixture.Db, getAllQuery);

    Assert.NotNull(obj);
}
```

✓ EmployeesWBaseEntityTest (2)	1,5 s
✓ Entities_GetAll_CanExecute	14 ms
✓ Entities_GetAll_NonEmpty	1,5 s

Shared Context

Class Fixture – Loading Entities

#somoshiberus

```
select * from HumanResources.Employee
```

	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate	MaritalStatus	Gender	HireDate
1	295847284	adventure-works\ken0	NULL	NULL	Chief Executive Officer	1969-01-29	S	M	2009-01-14
2	245797967	adventure-works\term0	0x58	1	Vice President of Engineering	1971-08-01	S	F	2008-01-31
3	509647174	adventure-works\roberto0	0x5AC0	2	Engineering Manager	1974-11-12	M	M	2007-11-11
4	112457891	adventure-works\rob0	0x5AD6	3	Senior Tool Designer	1974-12-23	S	M	2007-12-05
5	695256908	adventure-works\gail0	0x5ADA	3	Design Engineer	1952-09-27	M	F	2008-01-06
6	998320692	adventure-works\jossef0	0x5ADE	3	Design Engineer	1959-03-11	M	M	2008-01-24
7	134969118	adventure-works\dylan0	0x5AE1	3	Research and Development Manager	1987-02-24	M	M	2009-02-08
8	811994146	adventure-works\diane1	0x5AE158	4	Research and Development Engineer	1986-06-05	S	F	2008-12-29
9	658797903	adventure-works\gigi0	0x5AE168	4	Research and Development Engineer	1979-01-21	M	F	2009-01-16
10	879342154	adventure-works\michael6	0x5AE178	4	Research and Development Manager	1984-11-30	M	M	2009-05-03
11	974026903	adventure-works\ovidiu0	0x5AE3	3	Senior Tool Designer	1978-01-17	S	M	2010-12-05
12	480168528	adventure-works\thierry0	0x5AE358	4	Tool Designer	1959-07-29	M	M	2007-12-11
13	486228782	adventure-works\janice0	0x5AE368	4	Tool Designer	1989-05-28	M	F	2010-12-23
14	42487730	adventure-works\michael8	0x5AE5	3	Senior Design Engineer	1979-06-16	S	M	2010-12-30
15	56920285	adventure-works\sharon0	0x5AE7	3	Design Engineer	1961-05-02	M	F	2011-01-18
16	24756624	adventure-works\david0	0x68	1	Marketing Manager	1975-03-19	S	M	2007-12-20
17	253022876	adventure-works\kevin0	0x6AC0	2	Marketing Assistant	1987-05-03	S	M	2007-01-26
18	222969461	adventure-works\john5	0x6B40	2	Marketing Specialist	1978-03-06	S	M	2011-02-07
19	52541318	adventure-works\mary2	0x6BC0	2	Marketing Assistant	1978-01-29	S	F	2011-02-14
20	323403273	adventure-works\wanida0	0x6C20	2	Marketing Assistant	1975-03-17	M	F	2011-01-07
21	243322160	adventure-works\terry0	0x6C60	2	Marketing Specialist	1986-02-04	M	M	2009-03-02
22	95958330	adventure-works\sariya0	0x6CA0	2	Marketing Specialist	1987-05-21	S	M	2008-12-12

Shared Context

Class Fixture – Loading Entities

#somoshiberus

```
public interface IEntity<T>
{
    2 referencias
    T GetEntity(IDataReader reader);
}
```

```
public class EmployeeEntity : IEntity<EmployeeEntity>
{
    2 referencias
    public string LoginID { get; set; }
    2 referencias
    public DateTime BirthDate { get; set; }
    2 referencias
    public char Gender { get; set; }
    2 referencias
    public char MaritalStatus { get; set; }
}
```

```
public EmployeeEntity GetEntity(IDataReader reader)
{
    return new EmployeeEntity()
    {
        LoginID = (string)reader[nameof(LoginID)],
        BirthDate = DateTime.Parse(reader[nameof(BirthDate)].ToString()),
        Gender = (reader[nameof(Gender)].ToString()).FirstOrDefault(),
        MaritalStatus = (reader[nameof(MaritalStatus)].ToString()).FirstOrDefault()
    };
}
```

Shared Context

Class Fixture – Loading Entities

#somoshiberus

```
public static List<T> ExecuteReaderToList<T>(SqlConnection db,
                                             string sqlQuery)
    where T : IEntity<T>, new()
{
    List<T> result = new List<T>();
    SqlCommand command = db.CreateCommand();
    command.CommandText = sqlQuery;

    var objT = new T();
    using (var dataReader = command.ExecuteReader())
    {
        while (dataReader.Read())
        {
            result.Add(objT.GetEntity(dataReader));
        }
    }

    return result;
}
```


Shared Context

Class Fixture

#somoshiberus

```
private readonly char[] ValidGenders = { 'M', 'F' };

[Fact]
✓ | 0 referencias
public void Employees_CheckGenders_OnlyMFValues()
{
    var listEmployees = DBUtils.ExecuteReaderToList<EmployeeEntity>(
        dbFixture.Db, getAllQuery
    );
    Assert.All(listEmployees, e =>
        Assert.Contains(e.Gender, ValidGenders)
    );
}
```

✓ EmployeesWBaseEntityTest (3)	1,6 s
✓ Employees_CheckGenders_OnlyMFValues	152 ms

Check Constraints

Selected Check Constraint:

- CK_Employee_BirthDate
- CK_Employee_Gender**
- CK_Employee_HireDate
- CK_Employee_MaritalStatus
- CK_Employee_SickLeaveHours
- CK_Employee_VacationHours

Editing properties for existing check constraint.

✓ (General)	
Expression	(upper([Gender])='F' OR upper([Gender])='M')
✓ Identity	
(Name)	CK_Employee_Gender
Description	Check constraint [Gender]='f' OR [Gender]='m' OR [Gender]=''
✓ Table Designer	
Check Existing Data On Creat	Yes
Enforce For INSERTs And UPD	Yes
Enforce For Replication	Yes

Add Delete Close

Shared Context

Class Fixture

#somoshiberus

```
private readonly DateTime MIN_BIRTH_DATE = new DateTime(1930, 1, 1);

[Fact]
✓ | 0 referencias
public void Employees_CheckBirthDates_YoungerThan1930()
{
    var listEmployees = DBUtils.ExecuteReaderToList<EmployeeEntity>(
        dbFixture.Db, getAllQuery
    );
    Assert.All(listEmployees, e =>
        Assert.True(e.BirthDate >= MIN_BIRTH_DATE)
    );
}
```

✓ EmployeesWBaseEntityTest (3)	1,6 s
✓ Employees_CheckGenders_OnlyMFValues	152 ms

Check Constraints

Selected Check Constraint:

- CK_Employee_BirthDate
- CK_Employee_Gender
- CK_Employee_HireDate
- CK_Employee_MaritalStatus
- CK_Employee_SickLeaveHours
- CK_Employee_VacationHours

Editing properties for existing check constraint.

✓ (General)	
Expression	([BirthDate]>='1930-01-01' AND [BirthDate]<
✓ Identity	
(Name)	CK_Employee_BirthDate
Description	Check constraint [BirthDate] >= '1930-01-01'
✓ Table Designer	
Check Existing Data On Creat	Yes
Enforce For INSERTs And UPD	Yes
Enforce For Replication	Yes

Add Delete Close

Shared Context

Class Fixture - Optimizing Resources

#somoshiberus

```
[Fact]
✓ | 0 referencias
public void Employees_CheckGenders_OnlyMFValues()
{
    var listEmployees = DBUtils.ExecuteReaderToList<EmployeeEntity>(
        dbFixture.Db, getAllQuery
    );
    Assert.All(listEmployees, e =>
        Assert.Contains(e.Gender, ValidGenders)
    );
}

private readonly DateTime MIN_BIRTH_DATE = new DateTime(1930, 1, 1);

[Fact]
✓ | 0 referencias
public void Employees_CheckBirthDates_YoungerThan1930()
{
    var listEmployees = DBUtils.ExecuteReaderToList<EmployeeEntity>(
        dbFixture.Db, getAllQuery
    );
    Assert.All(listEmployees, e =>
        Assert.True(e.BirthDate >= MIN_BIRTH_DATE)
    );
}
```

Shared Context

Class Fixture – Optimizing Resources (+DRY)

#somoshiberus

```
private readonly List<EmployeeEntity> listEmployees;  
0 referencias  
public EmployeesWBaseEntityTest(DatabaseFixture databaseFixture) :  
    base(databaseFixture) {  
    listEmployees = DBUtils.ExecuteReaderToList<EmployeeEntity>(  
        dbFixture.Db, getAllQuery  
    );  
}
```

```
private readonly char[] ValidGenders = { 'M', 'F' };
```

```
[Fact]
```

```
✓ | 0 referencias
```

```
public void Employees_CheckGenders_OnlyMFValues()
```

```
{  
    Assert.All(listEmployees, e =>  
        Assert.Contains(e.Gender, ValidGenders)  
    );  
}
```

```
private readonly DateTime MIN_BIRTH_DATE = new DateTime(1930, 1, 1);
```

```
[Fact]
```

```
✓ | 0 referencias
```

```
public void Employees_CheckBirthDates_YoungerThan1930()
```

```
{  
    Assert.All(listEmployees, e =>  
        Assert.True(e.BirthDate >= MIN_BIRTH_DATE)  
    );  
}
```

Shared Context

ICollectionFixture – Global Scope

#somoshiberus

```
[CollectionDefinition("DB")]
0 referencias
public class DatabaseCollection : ICollectionFixture<DatabaseFixture>
{
    // This class has no code, and is never created. Its purpose is simply
    // to be the place to apply [CollectionDefinition] and all the
    // ICollectionFixture<> interfaces.
}
```

```
[Collection("DB")]
1 referencia
public class DepartmentsWBaseEntityAndCollectionTest : BaseEntityWCollectionTest
{
}
```

```
[Collection("DB")]
1 referencia
public class EmployeesWBaseEntityAndCollectionTest : BaseEntityWCollectionTest
{
}
```

Shared Context

ICollectionFixture – Global Scope

#somoshiberus

Executed once GLOBALLY

Prueba	Duración	Rasgos
✗ xUnitAdvanced (18)	1,4 s	
✗ xUnitAdvanced (18)	1,4 s	
✓ DepartmentsTest (2)	184 ms	
✓ DepartmentsWBaseEntityAndCollectionTest (2)	45 ms	
✓ DepartmentsWBaseEntityTest (2)		Ejecutar
✓ Entities_GetAll_CanExecute		Depurar
✓ Entities_GetAll_NonEmpty		Asociar a caso de prueba
✓ EmployeesConstructorTest (2)		Agregar a lista de reproducción
✓ EmployeesTest (2)		
✗ EmployeesWBaseEntityAndCollectionTest (4)		Expandir
✗ Employees_CheckBirthDates_YoungerThan1930		Contraer
✗ Employees_CheckGenders_OnlyMFValues		Abrir registro de prueba
✗ Entities_GetAll_CanExecute		Ir a la prueba
✗ Entities_GetAll_NonEmpty		
✓ EmployeesWBaseEntityTest (4)	235 ms	

```
9 public class DatabaseFixture : IDisposable
10 {
11     private const string cn = "Server=localhost;Database=
12         "Integrated Security=SSPI;Tru
13     10 referencias
14     public SqlConnection Db { get; private set; }
15     0 referencias
16     public DatabaseFixture()
17     {
18         Db = new SqlConnection(cn);
19         // ... initialize data in the test database ...
20         Db.Open();
21     }
```

```
...public interface IAsyncLifetime
{
    //
    // Resumen:
    //     Called when an object is no longer needed. Called just before System.IDisposable.Dispose
    //     if the class also implements that.
    Task DisposeAsync();
    //
    // Resumen:
    //     Called immediately after the class has been created, before it is used.
    Task InitializeAsync();
}
```

Lifecycle

IAsyncLifetime

#somoshiberus

```
public class FullLifecycleTest : IAsyncLifetime
{
    0 referencias
    public FullLifecycleTest()
    {
        var a = 0;
    }

    0 referencias
    public async Task DisposeAsync()
    {
        var a = 0;
    }

    0 referencias
    public async Task InitializeAsync()
    {
        var a = 0;
    }

    [Fact]
    0 referencias
    public void Nop_Nop_ShouldBeTrue()
    {
        Assert.True(true);
    }
}
```

```
public class FullLifecycleTest : IAsyncLifetime
{
    0 referencias
    public FullLifecycleTest()
    {
        var a = 0;
    }
}
```

```
0 referencias
public async Task InitializeAsync()
{
    var a = 0; ≤ 1 ms transcurridos
}
```

```
0 referencias
public async Task InitializeAsync()
{
    await Task.Delay(10000);
}
```


Lifecycle

Interceptors

#somoshiberus

```
public class TestBeforeAfterAttribute : BeforeAfterTestAttribute
{
    0 referencias
    public override void Before(MethodInfo methodUnderTest)
    {
        base.Before(methodUnderTest);
    }
    0 referencias
    public override void After(MethodInfo methodUnderTest)
    {
        base.After(methodUnderTest);
    }
}
```

```
0 referencias
public override void Before(MethodInfo methodUnderTest)
{
    base.Before(methodUnderTest); ≤ 10.010 ms transcurridos
}
```

```
[Fact]
[TestBeforeAfter]
✓ | 0 referencias
public void Nop_Nop_ShouldBeTrue()
{
    Assert.True(true);
}
```

```
public void Nop_Nop_ShouldBeTrue()
{
    Assert.True(true); ≤ 84 ms transcurr
}
```

- Constructor
- `IAsyncLifetime.InitializeAsync`
- `BeforeAfterTestAttribute.Before` (en orden)
- Ejecución de la prueba
- `BeforeAfterTestAttribute.After` (en orden inverso)
- `IAsyncLifetime.DisposeAsync`
- Dispose

*NOTA: En caso de que exista un **CollectionFixture**, su inicialización se ejecutará ANTES que el inicio de todas las pruebas, y se destruirá al acabar todas ellas, que seguirán este flujo en el mismo orden.*

```
0 referencias
public DatabaseFixture()
{
    Db = new SqlConnection(cn);

    // ... initialize data in the test database ...

    Db.Open();
}
```

Live Testing

Real-Time feedback

#somoshiberus

Visual Studio 2019
Enterprise



<https://docs.microsoft.com/es-es/visualstudio/test/live-unit-testing?view=vs-2022>

```
Class1.cs* X UnitTest1.cs
StringLibrary UtilityLibraries.StringLibrary
0 references
public static class StringLibrary
{
2 references
public static bool StartsWithUpper(this string s)
{
    if (String.IsNullOrEmpty(s))
        return false;

    return Char.IsUpper(s[0]);
}

0 references
public static bool StartsWithLower(this string s)
{
    if (String.IsNullOrEmpty(s))
        return false;
}

Test
X TestHasEmbeddedSpaces
Run All | Debug All

24 {
25 foreach (var ch in s.Trim())
26 {
27     if (ch == ' ')
28         return true;
29 }
30 return false;
31 }
32 }
33 }
```

```
Class1.cs* X UnitTest1.cs*
StringLibrary UtilityLibraries.StringLibrary
2 references
public static class StringLibrary
{
3 references
public static bool StartsWithUpper(this string s)
{
    if (String.IsNullOrEmpty(s))
        return false;

    return Char.IsUpper(s[0]);
}

3 references
public static bool StartsWithLower(this string s)
{
    if (String.IsNullOrEmpty(s))
        return false;

    return Char.IsLower(s[0]);
}

1 reference
public static bool HasEmbeddedSpaces(this string s)
{
    foreach (var ch in s.Trim())
    {
        if (ch == ' ' || ch == '\n')
            return true;
    }
    return false;
}
```

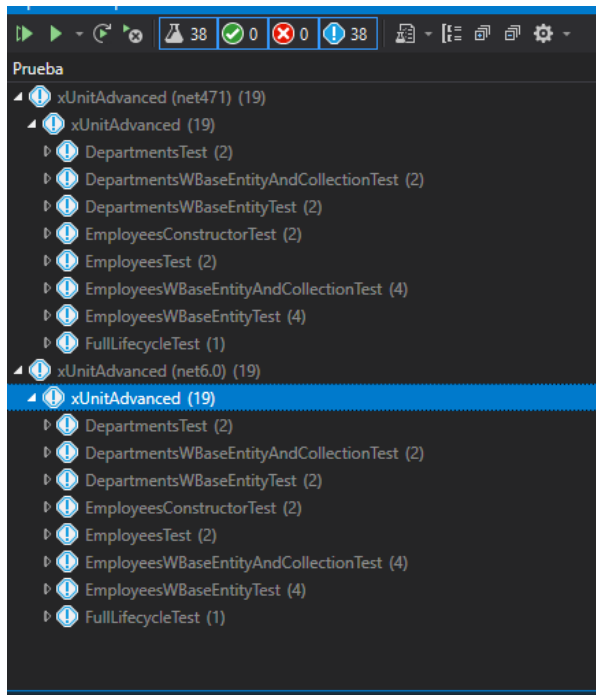
Multiple frameworks

Testing multi-target

#somoshiberus

```
xUnitAdvanced.csproj  ▸ x TestBeforeAfterAttribute.cs FullLifecycleTest.cs BaseEntityWCollection
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFrameworks>net471;net6.0</TargetFrameworks>

    <IsPackable>>false</IsPackable>
  </PropertyGroup>
```



✓ xUnitAdvanced (net471) (19)	1,8 s
✓ xUnitAdvanced (19)	1,8 s
✓ DepartmentsTest (2)	199 ms
✓ DepartmentsWBaseEntityAndCollectionTest (2)	45 ms
✓ DepartmentsWBaseEntityTest (2)	48 ms
✓ EmployeesConstructorTest (2)	573 ms
✓ EmployeesTest (2)	406 ms
✓ EmployeesWBaseEntityAndCollectionTest (4)	228 ms
✓ EmployeesWBaseEntityTest (4)	279 ms
✓ FullLifecycleTest (1)	1 ms
✓ xUnitAdvanced (net6.0) (19)	449 ms
✓ xUnitAdvanced (19)	449 ms
✓ DepartmentsTest (2)	118 ms
✓ DepartmentsWBaseEntityAndCollectionTest (2)	23 ms
✓ DepartmentsWBaseEntityTest (2)	19 ms
✓ EmployeesConstructorTest (2)	120 ms
✓ EmployeesTest (2)	119 ms
✓ EmployeesWBaseEntityAndCollectionTest (4)	13 ms
✓ EmployeesWBaseEntityTest (4)	37 ms
✓ FullLifecycleTest (1)	< 1 ms

Real-World Test

Putting all together

#somoshiberus

```
select * from HumanResources.department
```

	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2008-04-30 00:00:00.000
2	2	Tool Design	Research and Development	2008-04-30 00:00:00.000
3	3	Sales	Sales and Marketing	2008-04-30 00:00:00.000
4	4	Marketing	Sales and Marketing	2008-04-30 00:00:00.000
5	5	Purchasing	Inventory Management	2008-04-30 00:00:00.000
6	6	Research and Development	Research and Development	2008-04-30 00:00:00.000
7	7	Production	Manufacturing	2008-04-30 00:00:00.000
8	8	Production Control	Manufacturing	2008-04-30 00:00:00.000
9	9	Human Resources	Executive General and Administration	2008-04-30 00:00:00.000
10	10	Finance	Executive General and Administration	2008-04-30 00:00:00.000
11	11	Information Services	Executive General and Administration	2008-04-30 00:00:00.000
12	12	Document Control	Quality Assurance	2008-04-30 00:00:00.000
13	13	Quality Assurance	Quality Assurance	2008-04-30 00:00:00.000
14	14	Facilities and Maintenance	Executive General and Administration	2008-04-30 00:00:00.000
15	15	Shipping and Receiving	Inventory Management	2008-04-30 00:00:00.000
16	16	Executive	Executive General and Administration	2008-04-30 00:00:00.000

Real-World Test

Putting all together

#somoshiberus

```
public class DepartmentsWBaseEntityTest : BaseEntityTest, IClassFixture<DatabaseFixture>
{
    0 referencias
    public DepartmentsWBaseEntityTest(DatabaseFixture databaseFixture) : base(databaseFixture) { }

    3 referencias
    protected override string tableName => "HumanResources.Department";

    1 referencia
    protected int insertDepartment(string name, string groupName, DateTime modifiedDate) {
        string sqlQuery = $"INSERT INTO {tableName} (Name, GroupName, ModifiedDate)
            VALUES (@Name, @GroupName, @ModifiedDate)";
        List<SqlParameter> param = new List<SqlParameter>();
        param.Add(new SqlParameter("@Name", name));
        param.Add(new SqlParameter("@GroupName", groupName));
        param.Add(new SqlParameter("@ModifiedDate", modifiedDate));

        return DBUtils.ExecuteNonQuery(dbFixture.Db, sqlQuery, param.ToArray());
    }
}
```


Real-World Test

Putting all together

#somoshiberus

```
1 referencia
public static int ExecuteNonQuery(SqlConnection db,
    string sqlQuery,
    SqlParameter[] param)
{
    SqlCommand command = db.CreateCommand();
    command.CommandText = sqlQuery;
    command.Parameters.AddRange(param);

    return command.ExecuteNonQuery();
}
```

```
[Fact]
✓ | 0 referencias
public void InsertDepartment_NewEntity_ShouldSucceed()
{
    int result = insertDepartment("TEST", "TEST GROUP", DateTime.Now);
    Assert.Equal(1, result);
}
```

- ✓ DepartmentsWBaseEntityTest (3)
- ✓ Entities_GetAll_CanExecute
- ✓ Entities_GetAll_NonEmpty
- ✓ InsertDepartment_NewEntity_ShouldSucceed

select * from HumanResources.department				
L %				
Results Messages				
	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2008-04-30 00:00:00.000
2	2	Tool Design	Research and Development	2008-04-30 00:00:00.000
3	3	Sales	Sales and Marketing	2008-04-30 00:00:00.000
4	4	Marketing	Sales and Marketing	2008-04-30 00:00:00.000
5	5	Purchasing	Inventory Management	2008-04-30 00:00:00.000
6	6	Research and Development	Research and Development	2008-04-30 00:00:00.000
7	7	Production	Manufacturing	2008-04-30 00:00:00.000
8	8	Production Control	Manufacturing	2008-04-30 00:00:00.000
9	9	Human Resources	Executive General and Administration	2008-04-30 00:00:00.000
10	10	Finance	Executive General and Administration	2008-04-30 00:00:00.000
11	11	Information Services	Executive General and Administration	2008-04-30 00:00:00.000
12	12	Document Control	Quality Assurance	2008-04-30 00:00:00.000
13	13	Quality Assurance	Quality Assurance	2008-04-30 00:00:00.000
14	14	Facilities and Maintenance	Executive General and Administration	2008-04-30 00:00:00.000
15	15	Shipping and Receiving	Inventory Management	2008-04-30 00:00:00.000
16	16	Executive	Executive General and Administration	2008-04-30 00:00:00.000
17	17	TEST	TEST GROUP	2022-02-24 07:13:08.593



Real-World Test

Putting all together

#somoshiberus

```
protected int insertDepartment(string name, string groupName, DateTime modifiedDate) {
    string sqlQuery = $"INSERT INTO {tableName} (Name, GroupName, ModifiedDate)
        VALUES (@Name, @GroupName, @ModifiedDate);
        SELECT @@IDENTITY";
    List<SqlParameter> param = new List<SqlParameter>();
    param.Add(new SqlParameter("@Name", name));
    param.Add(new SqlParameter("@GroupName", groupName));
    param.Add(new SqlParameter("@ModifiedDate", modifiedDate));

    return Convert.ToInt32(DBUtils.ExecuteScalar(dbFixture.Db, sqlQuery, param.ToArray()));
}
```

```
public static object ExecuteScalar(SqlConnection db,
    string sqlQuery,
    SqlParameter[] param = null)
{
    SqlCommand command = db.CreateCommand();
    command.CommandText = sqlQuery;
    if (param != null) command.Parameters.AddRange(param);

    return command.ExecuteScalar();
}
```

```
[Fact]
✓ | 0 referencias
public void InsertDepartment_NewEntity_ShouldSucceed()
{
    int idResult = insertDepartment("TEST", "TEST GROUP", DateTime.Now);
    bool inserted = idResult > 0;
    Assert.True(inserted);
}
```


Real-World Test

Putting all together

#somoshiberus

0 referencias

```
protected int deleteDepartment(int ID)
{
    string sqlQuery = $"DELETE FROM {tableName} WHERE DepartmentID = @ID";
    List<SqlParameter> param = new List<SqlParameter>();
    param.Add(new SqlParameter("@ID", ID));

    return DBUtils.ExecuteNonQuery(dbFixture.Db, sqlQuery, param.ToArray());
}
```

[Fact]

✓ | 0 referencias

```
public void InsertDepartment_NewEntity_ShouldSucceed()
{
    int idResult = insertDepartment("TEST", "TEST GROUP", DateTime.Now);
    bool inserted = idResult > 0;
    Assert.True(inserted);

    //Clean-up
    Assert.Equal(1, deleteDepartment(idResult));
}
```

Real-World Test

Putting all together

#somoshiberus

[Fact]

✓ | 0 referencias

```
public void InsertDepartment_NewEntityWTransaction_ShouldSucceed()
{
    SqlTransaction tran = dbFixture.Db.BeginTransaction();
    int idResult = insertDepartment("TEST", "TEST GROUP", DateTime.Now, tran);
    bool inserted = idResult > 0;
    Assert.True(inserted);

    tran.Rollback();
}
```

2 referencias | 0/2 pasando

```
protected int insertDepartment(string name, string groupName, DateTime modifiedDate, SqlTransaction tran = null) {
    string sqlQuery = $"INSERT INTO {tableName} (Name, GroupName, ModifiedDate)
        VALUES (@Name, @GroupName, @ModifiedDate);
        SELECT @@IDENTITY";
    List<SqlParameter> param = new List<SqlParameter>();
    param.Add(new SqlParameter("@Name", name));
    param.Add(new SqlParameter("@GroupName", groupName));
    param.Add(new SqlParameter("@ModifiedDate", modifiedDate));

    return Convert.ToInt32(DBUtils.ExecuteScalar(dbFixture.Db, sqlQuery, param.ToArray(), tran));
}
```

Real-World Test

Putting all together

#somoshiberus

```
public static object ExecuteScalar(SqlConnection db,
                                   string sqlQuery,
                                   SqlParameter[] param = null,
                                   SqlTransaction tran = null)
{
    SqlCommand command = db.CreateCommand();
    command.CommandText = sqlQuery;
    if (param != null) command.Parameters.AddRange(param);
    if (tran != null) command.Transaction = tran;

    return command.ExecuteScalar();
}
```

select * from HumanResources.department

	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2008-04-30 00:00:00.000
2	2	Tool Design	Research and Development	2008-04-30 00:00:00.000
3	3	Sales	Sales and Marketing	2008-04-30 00:00:00.000
4	4	Marketing	Sales and Marketing	2008-04-30 00:00:00.000
5	5	Purchasing	Inventory Management	2008-04-30 00:00:00.000
6	6	Research and Development	Research and Development	2008-04-30 00:00:00.000
7	7	Production	Manufacturing	2008-04-30 00:00:00.000
8	8	Production Control	Manufacturing	2008-04-30 00:00:00.000
9	9	Human Resources	Executive General and Administration	2008-04-30 00:00:00.000
10	10	Finance	Executive General and Administration	2008-04-30 00:00:00.000
11	11	Information Services	Executive General and Administration	2008-04-30 00:00:00.000
12	12	Document Control	Quality Assurance	2008-04-30 00:00:00.000
13	13	Quality Assurance	Quality Assurance	2008-04-30 00:00:00.000
14	14	Facilities and Maintenance	Executive General and Administration	2008-04-30 00:00:00.000
15	15	Shipping and Receiving	Inventory Management	2008-04-30 00:00:00.000
16	16	Executive	Executive General and Administration	2008-04-30 00:00:00.000

✓ DepartmentsWBaseEntityTest (4)	387 ms
✓ Entities_GetAll_CanExecute	2 ms
✓ Entities_GetAll_NonEmpty	46 ms
✓ InsertDepartment_NewEntity_ShouldSucceed	199 ms
✓ InsertDepartment_NewEntityWTransaction_ShouldSucceed	140 ms

Real-World Test

Putting all together

#somoshiberus

```
public class DatabaseFixture : IDisposable
{
    private const string cn = "Server=localhost;Database=AdventureWorks2019;" +
                              "Integrated Security=SSPI;TrustServerCertificate=True";

    14 referencias | 1/1 pasando
    public SqlConnection Db { get; private set; }
    2 referencias
    public SqlTransaction SqlTransaction { get; private set; }

    0 referencias
    public DatabaseFixture()
    {
        Db = new SqlConnection(cn);

        // ... initialize data in the test database ...
        Db.Open();
        SqlTransaction = Db.BeginTransaction();
    }

    0 referencias
    public void Dispose()
    {
        // ... clean up test data from the database ...
        SqlTransaction.Rollback();
        Db.Close();
        Db.Dispose();
    }
}
```

A complex network diagram in the background, consisting of numerous grey circles of varying sizes connected by thin grey lines. Some circles are highlighted with dashed outlines. The overall theme is technology and connectivity.

hiberus[©] TECNOLOGIA

La compañía **hiperespecializada** en las TIC

www.hiberus.com