Formación de Testing en .NET

Nafarroako Gobernua
Gobierno de Navarra

hiberus©

La compañía hiperespecializada en las TIC

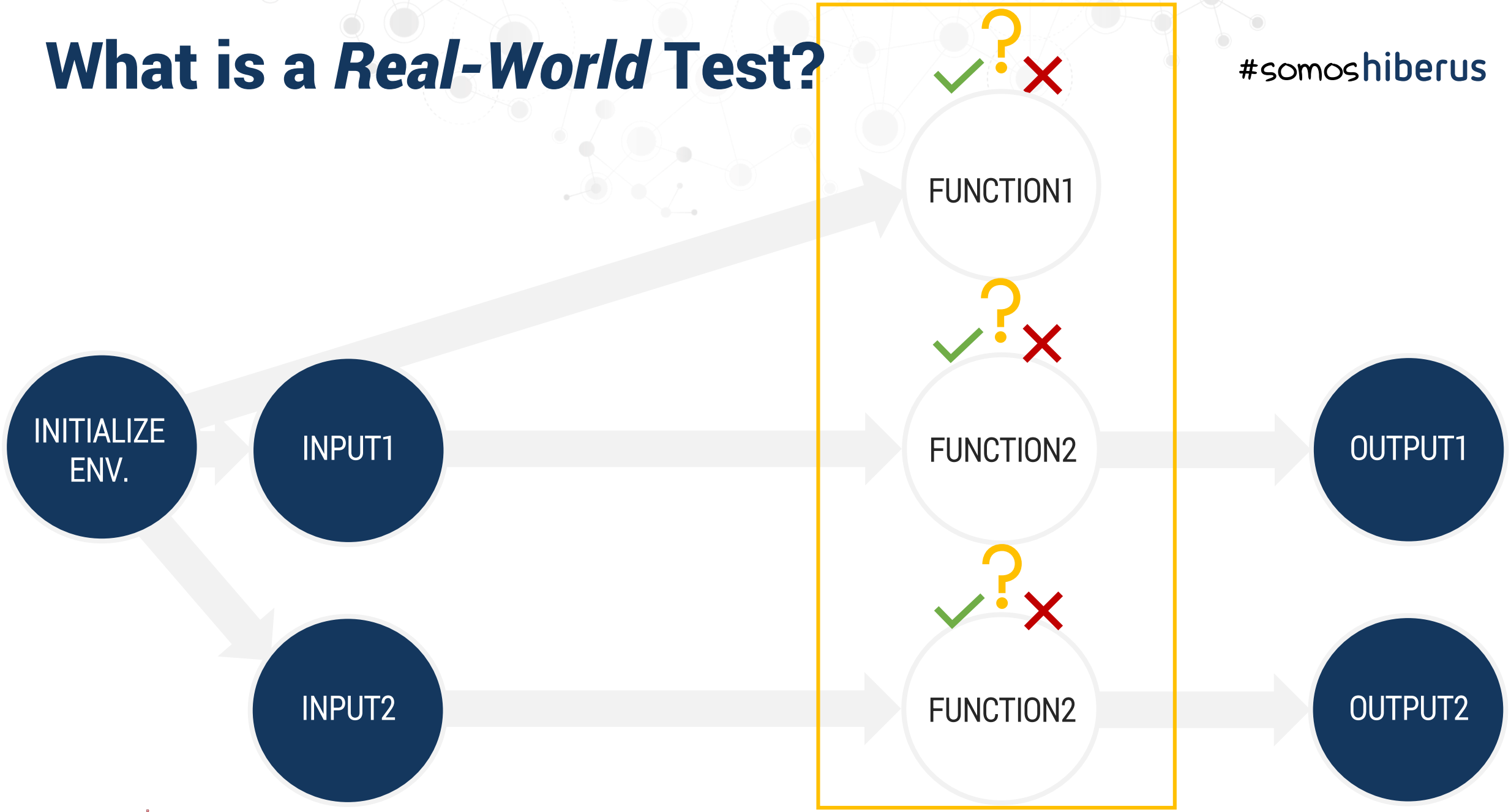
Tema 3.1: Moq Avanzado

Nafarroako Gobernua
Gobierno de Navarra

www.hiberus.com

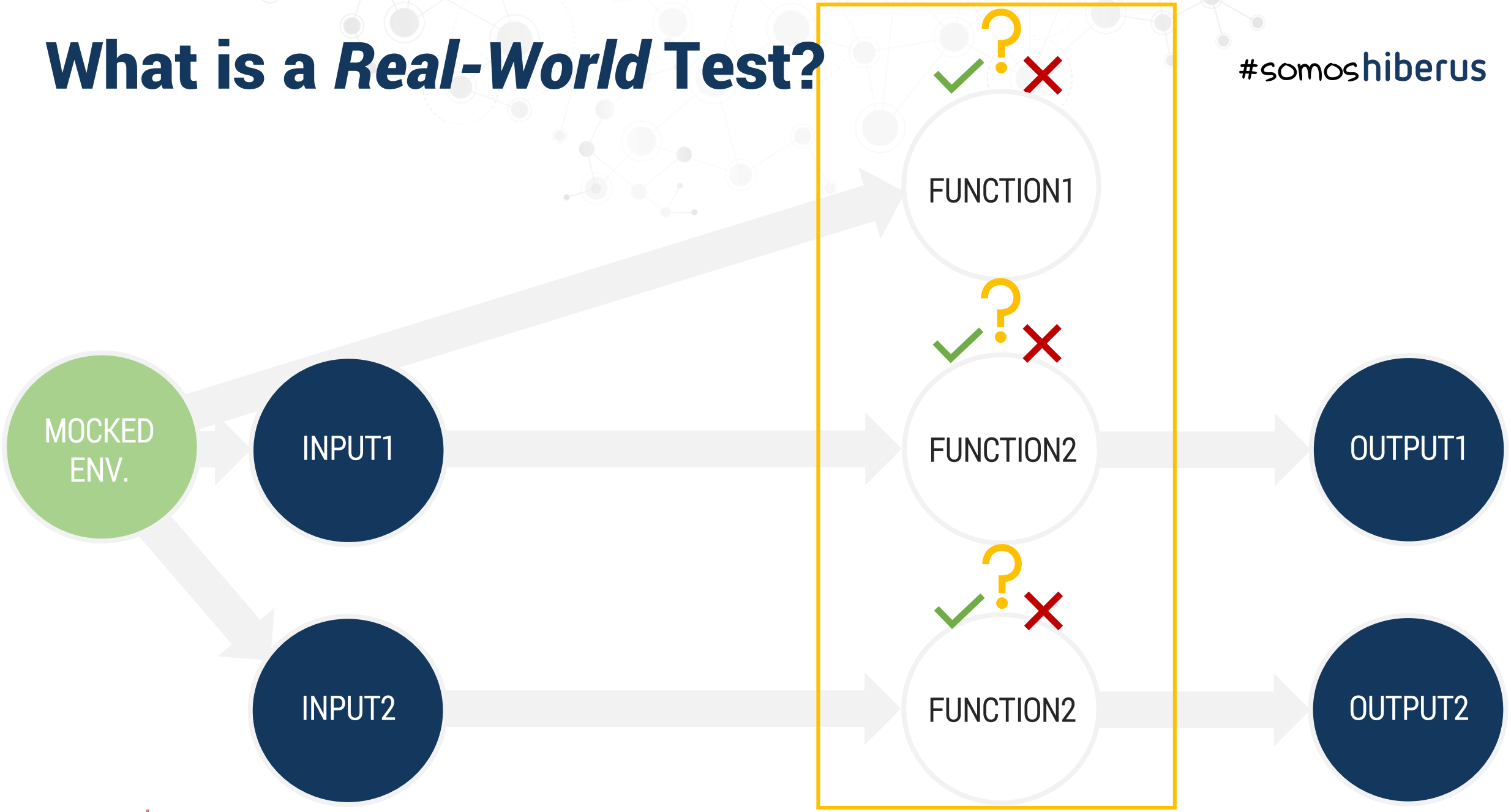
What is a *Real-World* Test?

#somoshiberus



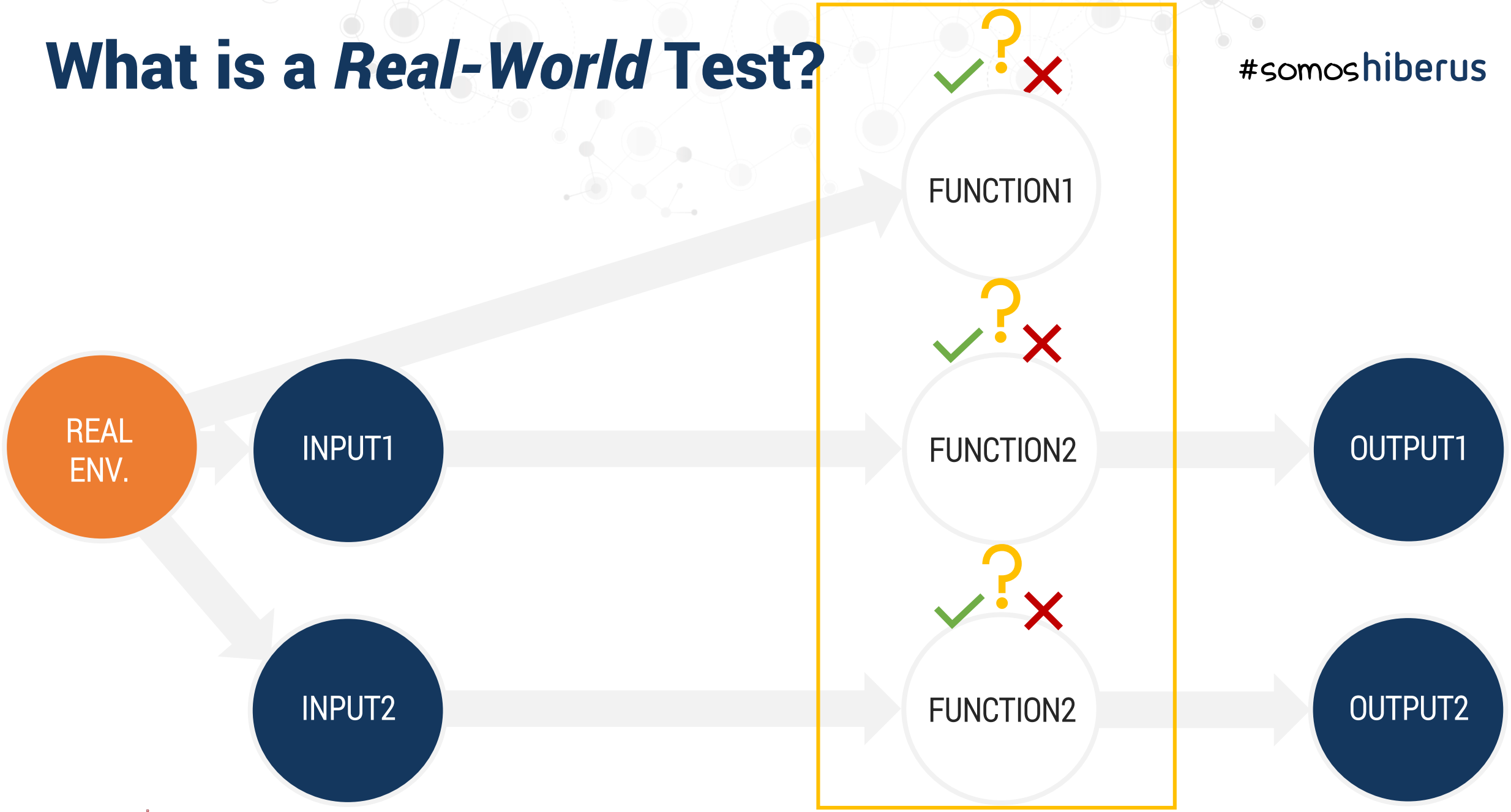
What is a *Real-World* Test?

#somoshiberus



What is a *Real-World* Test?

#somoshiberus



Moq

Default Behaviours


```
public interface IFoo
{
    3 referencias
    int GetCount();
}
```

```
public enum MockBehavior
{
    Strict = 0,
    Loose = 1,
    Default = 1
}
```

```
[Fact]
 0 referencias
public void Test1()
{
    var mockDefault = new Mock<IFoo>();
    var mockStrict = new Mock<IFoo>(MockBehavior.Strict);
    var mockLoose = new Mock<IFoo>(MockBehavior.Loose);

    Assert.Throws<MockException>(() => mockStrict.Object.GetCount());
    Assert.Equal(0, mockDefault.Object.GetCount());
    Assert.Equal(0, mockLoose.Object.GetCount());
}
```

Moq

Default Behaviours


```
public interface IParentFoo
{
    0 referencias
    IFoo getFoo();
    0 referencias
    string getError();
}
```

```
[Fact]
0 referencias
public void Test2()
{
    var mockStrict = new Mock<IParentFoo>(MockBehavior.Strict);
    var mockLoose = new Mock<IParentFoo>(MockBehavior.Loose);

    Assert.Throws<MockException>(() => mockStrict.Object.getFoo());
    Assert.Throws<MockException>(() => mockStrict.Object.getError());
    var foo = mockLoose.Object.getFoo();
}
```

```
public enum DefaultValue
{
    Empty = 0,
    Mock = 1,
    Custom = 2
}
```

```
var mockWDefaultMock = new Mock<IParentFoo>() { DefaultValue = DefaultValue.Mock };
var fooMock = mockWDefaultMock.Object.getFoo();
Assert.NotN
```

Moq

Default Behaviours


```
var mockWDefaultCustom = new Mock<IParentFoo>() { DefaultValue = DefaultValue.Custom };
var fooCustom = mockWDefaultCustom.Object.getFoo();
Assert.NotNull(fooCustom);
```

Excepción no controlada por el usuario

System.ArgumentOutOfRangeException: 'Specified argument was out of the range of valid values. '

```
public interface IParentFoo
{
    0 referencias
    IFoo getFoo();
    0 referencias
    string getError();
}
```

```
class MyEmptyDefaultValueProvider : LookupOrFallbackDefaultValueProvider
{
    1 referencia | 1/1 pasando
    public MyEmptyDefaultValueProvider()
    {
        base.Register(typeof(string), (type, mock) => "Empty string");
        base.Register(typeof(List<>), (type, mock) => Activator.CreateInstance(type));
        base.Register(typeof(IFoo), (type, mock) => {
            Mock<IFoo> fooM = new Mock<IFoo>();
            fooM.Setup(m => m.GetCount()).Returns(10);
            return fooM.Object;
        });
    }
}
```

```
public enum DefaultValue
{
    Empty = 0,
    Mock = 1,
    Custom = 2
}
```

```
var mockWDefaultCustom = new Mock<IParentFoo>() {
    DefaultValueProvider = new MyEmptyDefaultValueProvider()
};
var fooCustom = mockWDefaultCustom.Object.getFoo();
Assert.NotNull(fooCustom);
Assert.Equal(10, fooCustom.GetCount());
```

Moq

Protected members


```
public class FooClass{

    protected virtual int DemoMethod(string fooParam) { return 0; }
}
```

```
[Fact]
0 referencias
public void Test3()
{
    var mock = new Mock<FooClass>();
    mock.Setup(m => m.)
}
```

Equals
GetHashCode
GetType
ToString

```
[Fact]
0 referencias
public void Test3()
{
    var mock = new Mock<FooClass>();
    mock.Protected().Setup(m => m.)
```

using Moq.Protected;

Introducir la variable local de 'mock.Protected().Setup(m => m.)'

CS1061 "Mock<MoqAdvancedDemo.FooClass>" no contiene una definición para "Protected" ni un método de extensión accesible "Protected" que acepte un primer argumento del tipo "Mock<MoqAdva...

```
using Moq;
using Moq.Protected;
using System;
```

Moq

Protected members


```
[Fact]
0 referencias
public void Test3()
{
    var mock = new Mock<FooClass>();
    mock.Protected()
        .Setup<int>("DemoMethod", ItExpr.IsAny<string>())
        .Returns(1234);

    var obj = mock.Object;
    Assert.Equal(1234, obj.GetDemoMethod());
}
```

```
1 referencia
public class FooClass
{
    1 referencia
    public int GetDemoMethod() => DemoMethod("asdf");
    1 referencia
    protected virtual int DemoMethod(string fooParam) { return 0; }
}
```

```
public abstract class FooClass
{
    1 referencia
    public int GetDemoMethod() => DemoMethod("asdf");
    1 referencia
    protected abstract int DemoMethod(string fooParam);
}
```

Moq

Generic Types


```
public interface IFoo
{
    5 referencias | 1/2 pasando
    int GetCount();
    0 referencias
    bool IsValid<T>(T arg);
}
```

```
[Fact]
0 referencias
public void TestGenericTypes()
{
    var mock = new Mock<IFoo>();
    mock.Setup(m => m.IsValid(It.IsAny<string>())).Returns(true);
    Assert.True(mock.Object.IsValid(""));
```

```
mock.Setup(m => m.IsValid(It.IsAny<It.IsValueType>())).Returns(true);
Assert.True(mock.Object.IsValid(0));
mock.Setup(m => m.IsValid(It.IsAny<It.IsAnyType>())).Returns(true);
Assert.True(mock.Object.IsValid(0));
Assert.True(mock.Object.IsValid(""));
mock.Setup(m => m.IsValid(It.IsAny<It.IsSubtype<object>>())).Returns(true);
Assert.True(mock.Object.IsValid(0.0m));
```

class System.Object
Supports all classes in the .NET class

Moq

Mock Repository


```
public enum MockBehavior
{
    Strict = 0,
    Loose = 1,
    Default = 1
}
```

```
public enum DefaultValue
{
    Empty = 0,
    Mock = 1,
    Custom = 2
}
```

```
var mockWDefaultCustom = new Mock<IParentFoo>()
{
    DefaultValueProvider = new MyEmptyDefaultValueProvider()
};
```

```
[Fact]
0 referencias
public void TestMockRepository()
{
    var mockRepository = new MockRepository(MockBehavior.Default);

    var foo = mockRepository.Create<IFoo>(MockBehavior.Default);
    foo.Setup(m => m.GetCount()).Returns(1);

    var fooObj = mockRepository.Of<IFoo>().FirstOrDefault();
    Assert.NotEqual(foo.Object, fooObj);

    var foo2 = mockRepository.OneOf<IParentFoo>();
    Mock.Get(foo2).Setup(m => m.getError()).Returns("asdf");
}
```

Moq

Summing up

#somoshiberus

```
namespace Ssid.Ejemplo.Bll.UnitTests.Utils
{
    5 referencias
    public class DataBaseManagerMock : IDataBaseManager
    {
        private readonly IDataBaseManager _dbManager;

        2 referencias
        public DataBaseManagerMock()
        {
            this._dbManager = Mock.Of<IDataBaseManager>();
        }

        2 referencias | 0/2 pasando
        public void ConfigureReadStoredProcedure(string storedProcedureName, IDataReader dataReader)
        {
            var dbCommand = Mock.Of<DbCommand>();

            Mock.Get(_dbManager)
                .Setup(x => x.GetStoredProcedureCommand(storedProcedureName))
                .Returns(dbCommand);

            Mock.Get(_dbManager)
            .Setup(x => x.ExecuteReader(dbCommand)).Returns(dataReader);
        }

        1 referencia | 0/1 pasando
        public void ConfigureWriteStoredProcedure(string storedProcedureName, string outputParamName, object outputParamValue)
        {
            var dbCommand = Mock.Of<DbCommand>();

            Mock.Get(_dbManager)
                .Setup(x => x.GetStoredProcedureCommand(storedProcedureName))
                .Returns(dbCommand);

            Mock.Get(_dbManager)
                .Setup(x => x.GetParameterValue(dbCommand, outputParamName))
                .Returns(outputParamValue);
        }
```

Moq

Summing up

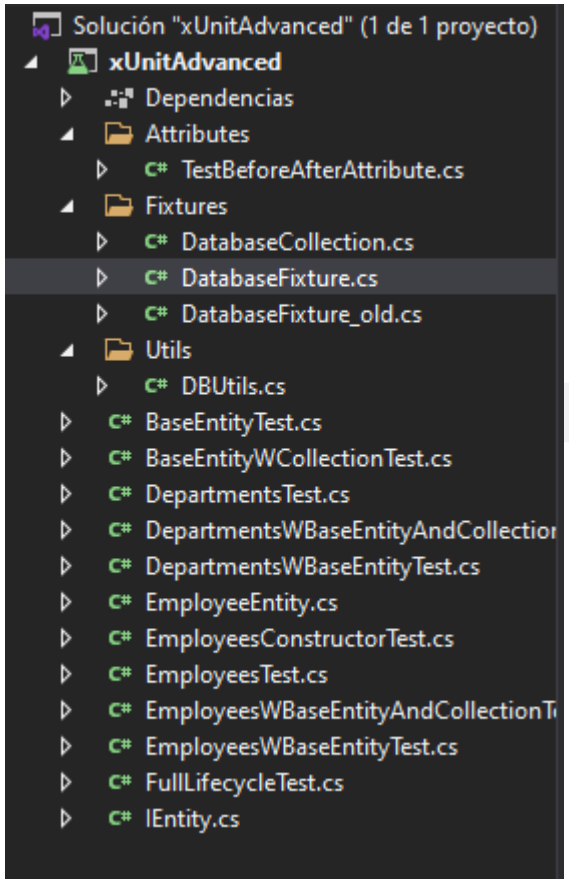
https://github.com/Moq/moq4/wiki/Quickstart

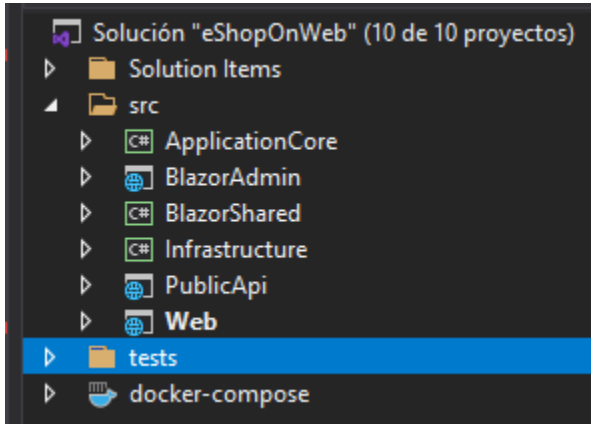
Real-World Test

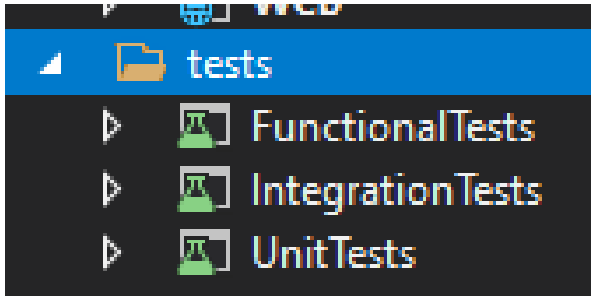
N-Layer Arquitecture

#somoshiberus

https://github.com/dotnet-architecture/eShopOnWeb







Real-World Test

Unit, Integration & Functional Tests

- **Unit Testing**
  Test that individual modules **issolated**
  are working as expected

- **Integration Testing**
  Test that the interaction of differentes modules **working together**
  are working as expected

- **Functional Testing**
  Test that **specific functionalities** of our system
  are working as expected

Real-World Test

Unit, Integration & Functional Tests

Example: Functioning Mobile Phone

Unit Testing

- Battery Capacity is the expected
- Battery Life is the expected
- Processor is working at the expected speed
- Screen can display info.
- Screen can detect single tap
- Screen can detect multi-touch
- Camera is functional

Integration Testing

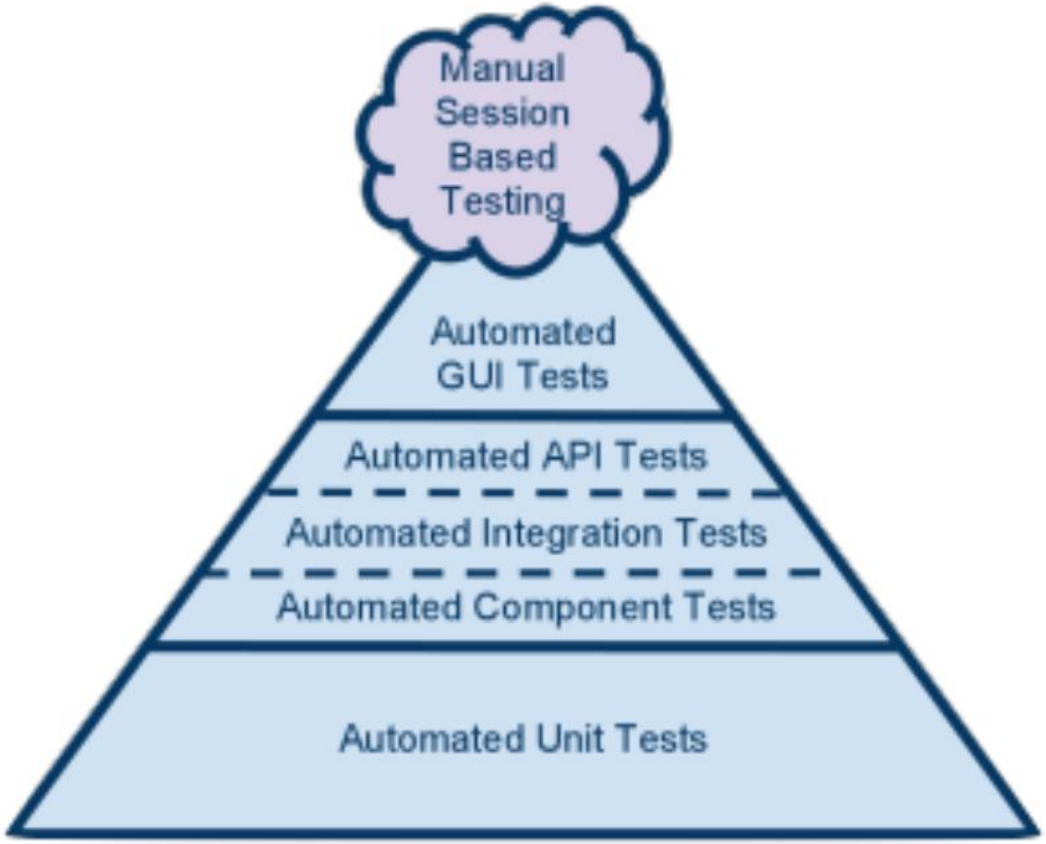
- Processor can initiate the integrated Operating System
- Screen is powered correctly through battery
- SIM is detected correctly by the Operating System
- Camera can store the taken picture in the phone memory

Functional Testing

- The phone can install an app that uses filters over pictures taken with the camera
- The phone can send messages through WhatsApp via WiFi
- The phone can receive security updates from the manufacturer via WiFi and update its software

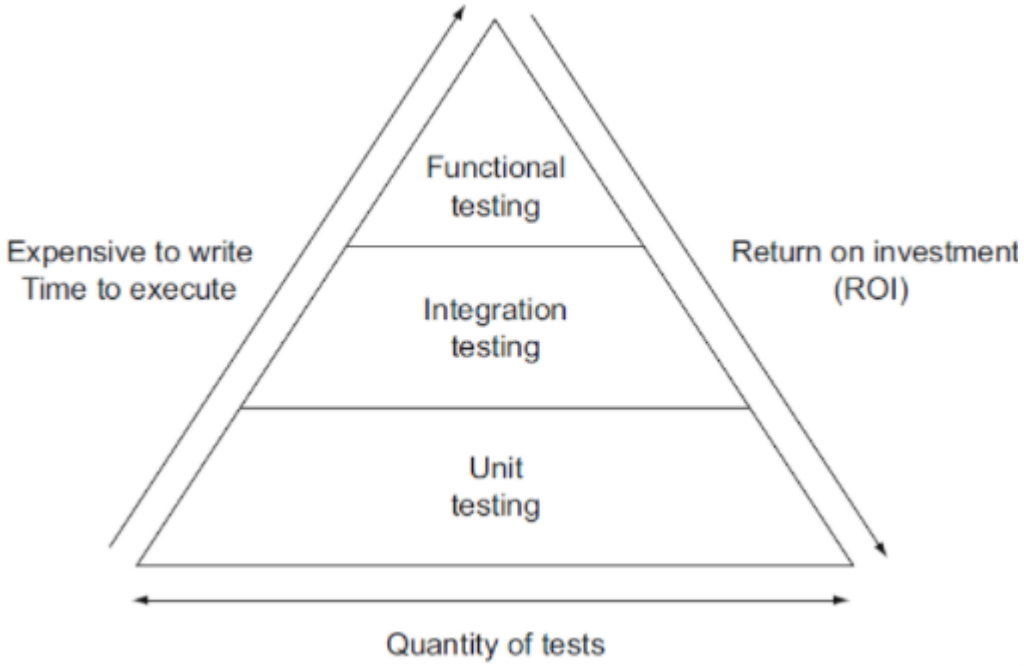
Real-World Test

Cohn's Pyramid



Real-World Test

Unit, Integration & Functional Tests

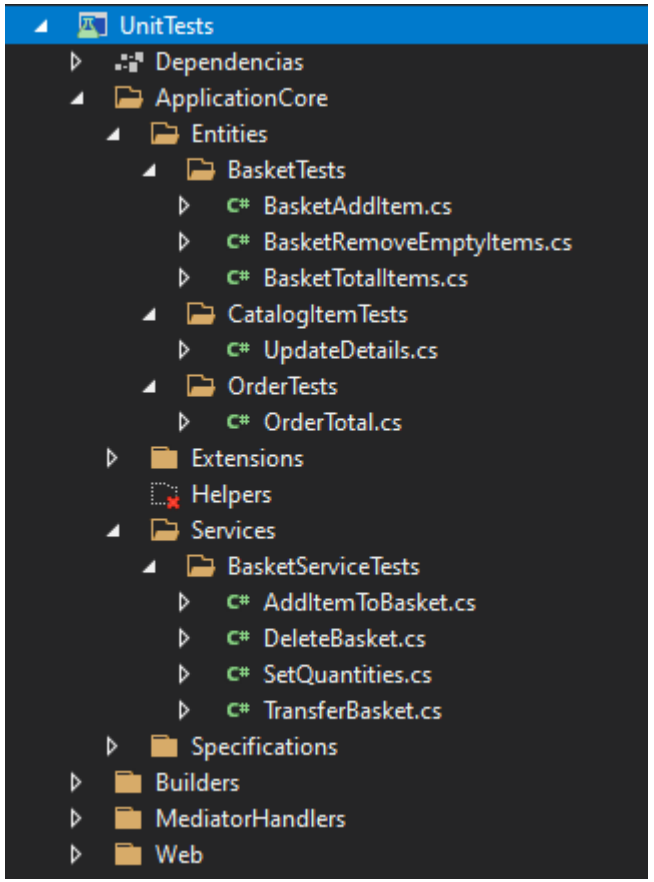


Real-World Test

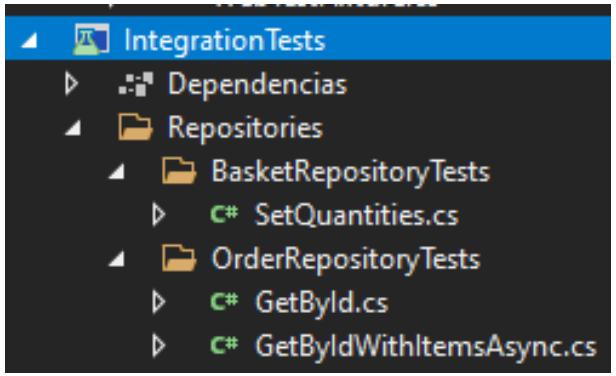
Unit, Integration & Functional Tests

#somoshiberus

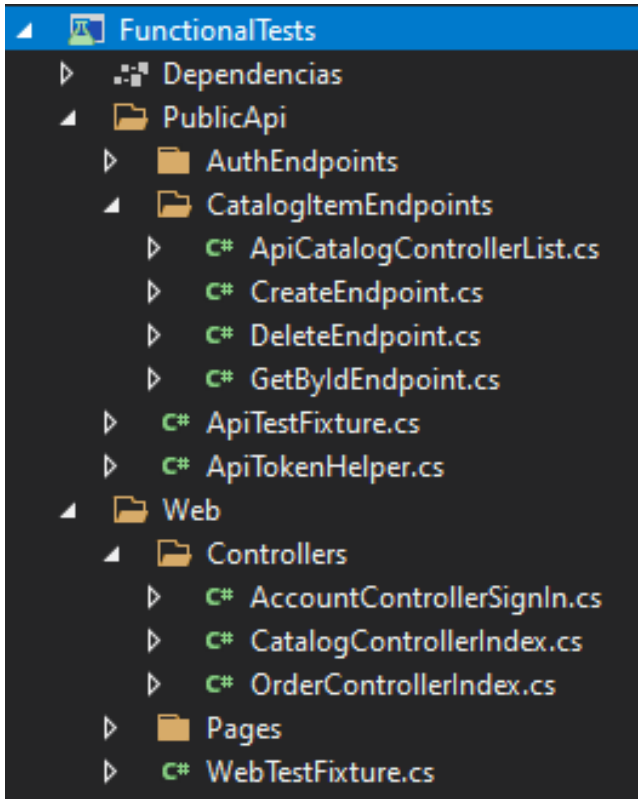
Unit Testing



Integration Testing



Functional Testing



Real-World Test

Unit, Integration & Functional Tests

Unit Test

```
namespace Microsoft.eShopWeb.UnitTests.ApplicationCore.Services.BasketServiceTests

0 referencias
public class SetQuantities
{
    private readonly int _invalidId = -1;
    private readonly Mock<IRepository<Basket>> _mockBasketRepo = new();

    [Fact]
    0 referencias
    public async Task ThrowsGivenInvalidBasketId()
    {
        var basketService = new BasketService(_mockBasketRepo.Object, null);

        await Assert.ThrowsAsync<BasketNotFoundException>(async () =>
            await basketService.SetQuantities(_invalidId, new System.Collections.G
    }
```

Real-World Test

Unit, Integration & Functional Tests

#somoshiberus

Integration Test

```
namespace Microsoft.eShopWeb.IntegrationTests.Repositories.BasketRepositoryTests

1 referencia
public class SetQuantities
{
    private readonly CatalogContext _catalogContext;
    private readonly EfRepository<Basket> _basketRepository;
    private readonly BasketBuilder BasketBuilder = new BasketBuilder();

    0 referencias
    public SetQuantities()
    {
        var dbOptions = new DbContextOptionsBuilder<CatalogContext>()
            .UseInMemoryDatabase(databaseName: "TestCatalog")
            .Options;
        _catalogContext = new CatalogContext(dbOptions);
        _basketRepository = new EfRepository<Basket>(_catalogContext);
    }
```

```
[Fact]
0 referencias
public async Task RemoveEmptyQuantities()
{
    var basket = BasketBuilder.WithOneBasketItem();
    var basketService = new BasketService(_basketRepository, null);
    await _basketRepository.AddAsync(basket);
    _catalogContext.SaveChanges();

    await basketService.SetQuantities(BasketBuilder.BasketId, new Di

    Assert.Equal(0, basket.Items.Count);
}
```

Real-World Test

Unit, Integration & Functional Tests

#somoshiberus

Functional Test

```
[Collection("Sequential")]
1 referencia
public class ApiCatalogControllerList : IClassFixture<TestApiApplication>
{
    0 referencias
    public ApiCatalogControllerList(TestApiApplication factory)
    {
        Client = factory.CreateClient();
    }

    3 referencias
    public HttpClient Client { get; }

    [Fact]
    0 referencias
    public async Task ReturnsFirst10CatalogItems()
    {
        var response = await Client.GetAsync("/api/catalog-items?pageSize=10");
        response.EnsureSuccessStatusCode();
        var stringResponse = await response.Content.ReadAsStringAsync();
        var model = stringResponse.FromJson<CatalogIndexViewModel>();

        Assert.Equal(10, model.CatalogItems.Count());
    }
```

```
namespace Microsoft.eShopWeb.FunctionalTests.Web;

10 referencias
public class TestApplication : WebApplicationFactory<IBasketViewModelService>
{
    private readonly string _environment = "Development";

    0 referencias
    protected override IHost CreateHost(IHostBuilder builder)
    {
        builder.UseEnvironment(_environment);

        // Add mock/test services to the builder here
        builder.ConfigureServices(services =>
        {
            services.AddScoped(sp =>
            {
                // Replace SQLite with in-memory database for tests
                return new DbContextOptionsBuilder<CatalogContext>()
                .UseInMemoryDatabase("InMemoryDbForTesting")
                .UseApplicationServiceProvider(sp)
                .Options;
            });
            services.AddScoped(sp =>
            {
                // Replace SQLite with in-memory database for tests
                return new DbContextOptionsBuilder<AppIdentityDbContext>()
                .UseInMemoryDatabase("Identity")
                .UseApplicationServiceProvider(sp)
                .Options;
            });
        });

        return base.CreateHost(builder);
    }
```

Real-World Test

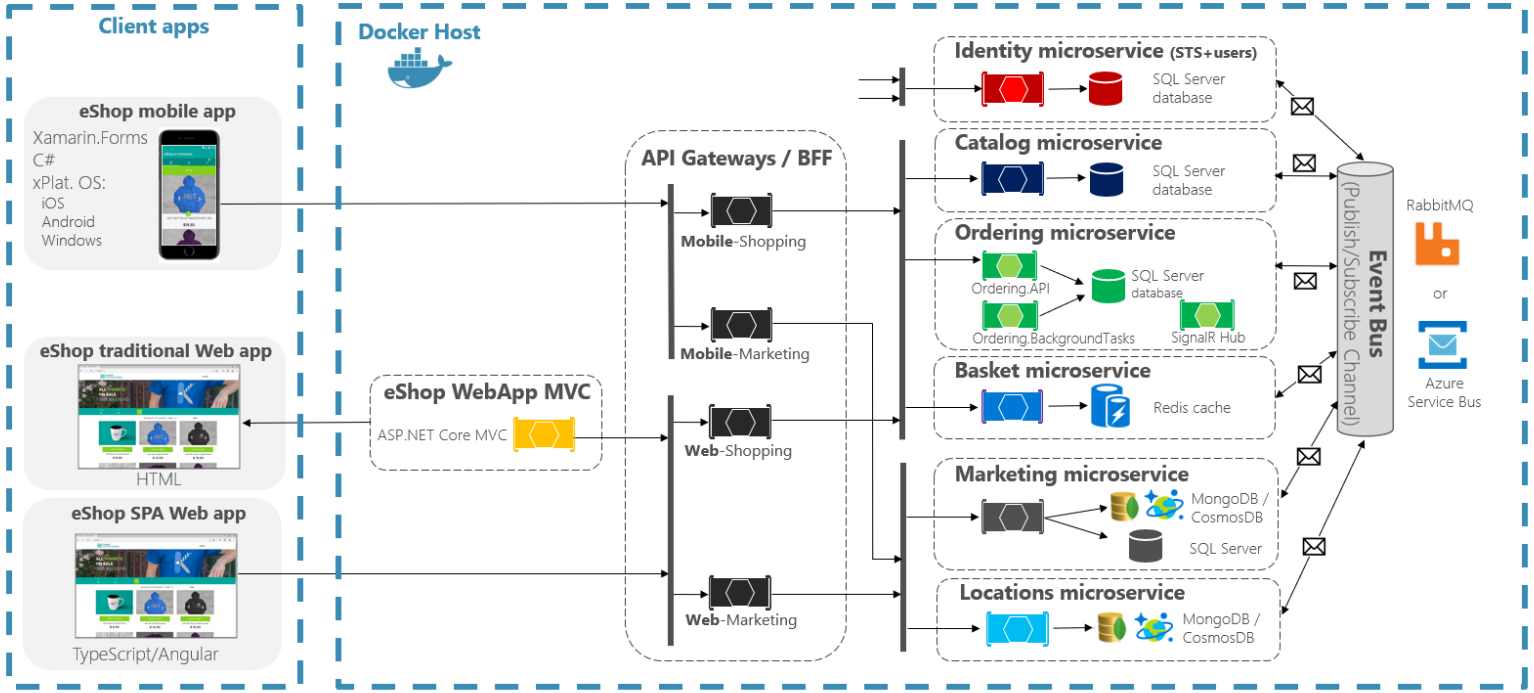
Unit, Integration & Functional Tests

#somoshiberus

https://github.com/dotnet-architecture/eShopOnContainers

eShopOnContainers reference application

(Development environment architecture)

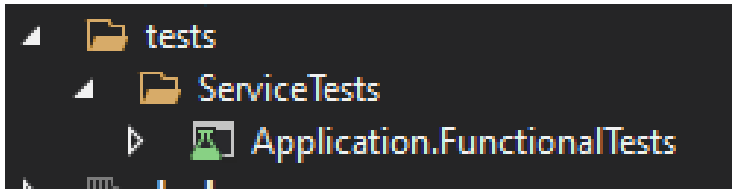


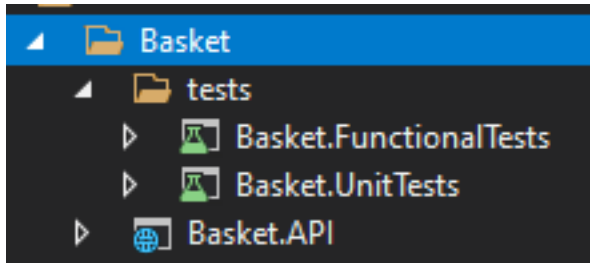
Real-World Test

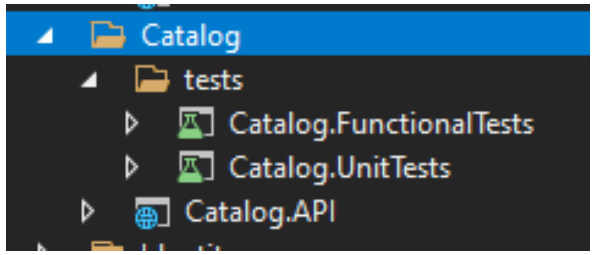
Unit, Integration & Functional Tests

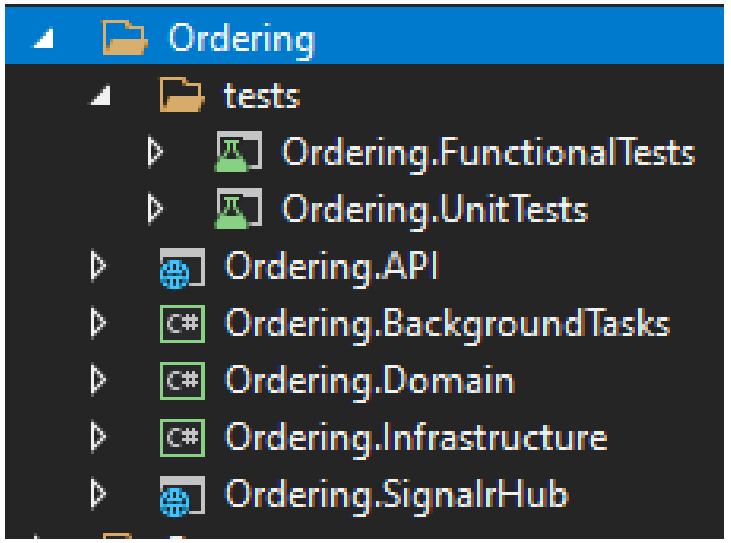
#somoshiberus

https://github.com/dotnet-architecture/eShopOnContainers









Real-World Test

Unit, Integration & Functional Tests

hiberus© TECNOLOGIA

La compañía hiperespecializada en las TIC

www.hiberus.com