Desatando el poder de las *Pruebas Unitarias*

JANIEL BALDEMAR NOH CELIS

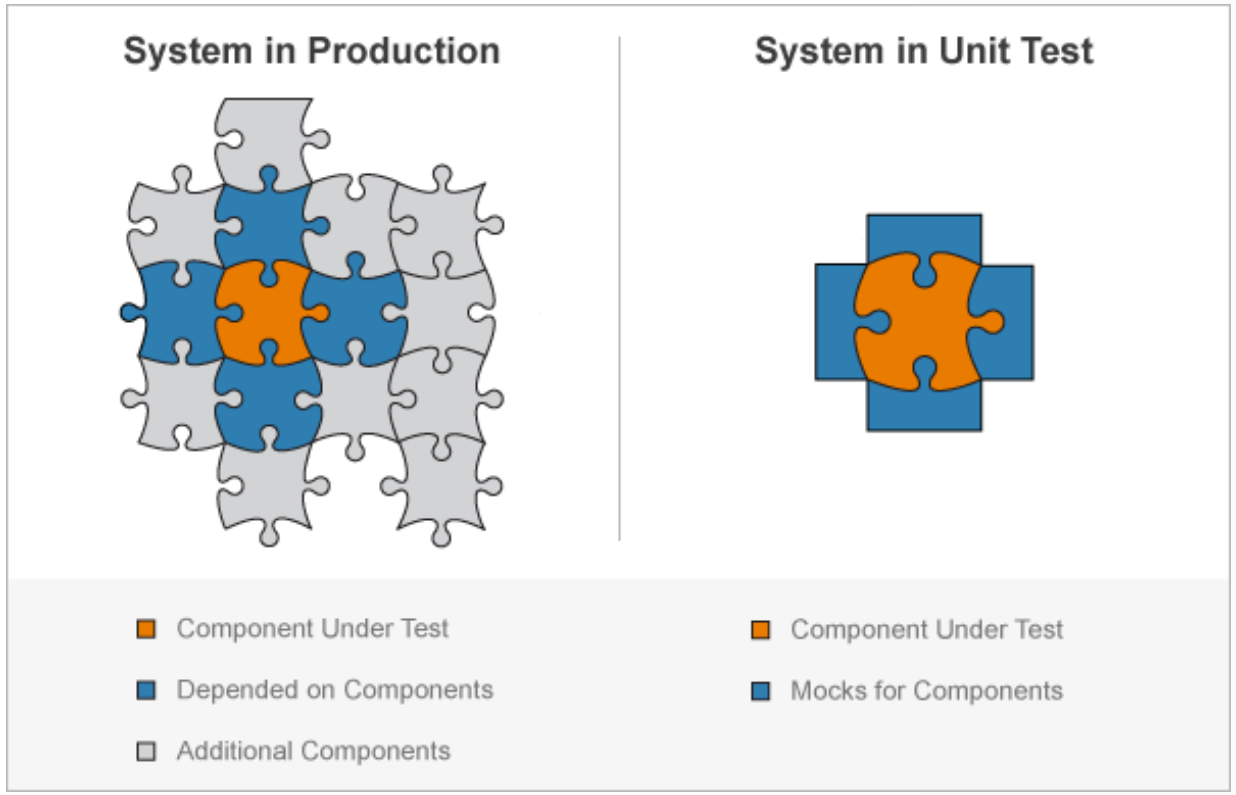
AGOSTO 2023

Orden del día

- Acerca de CONTPAQi®
- Pruebas Unitarias
  - Definición
  - Misión
  - 4 Pilares
  - Cobertura
  - Anatomía
  - Tamaño de las Secciones
  - Nomenclatura
  - Cuadrante de Pruebas
  - Pirámide de Pruebas
  - Uso de Mock
- Mitos
- Reflexión

Sobre CONTPAQi ®

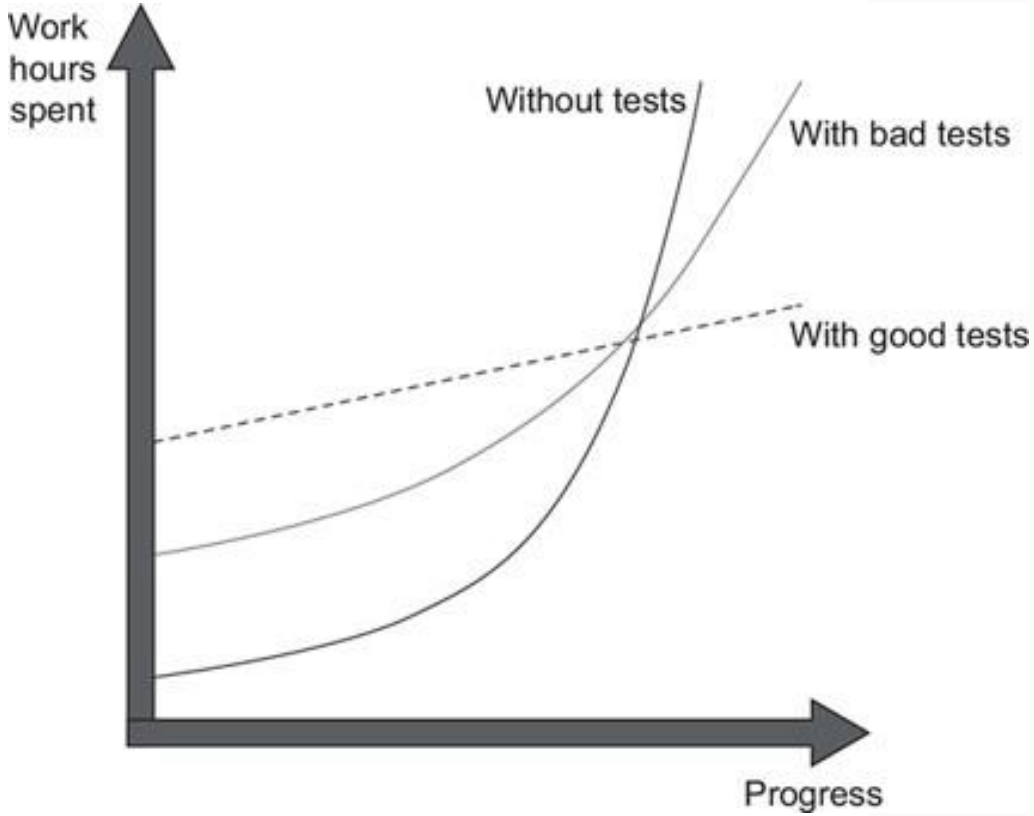
Definición



- Es una parte esencial del desarrollo de software que tiene como objetivo verificar el correcto funcionamiento de unidades de código individualmente, como funciones, métodos o clases, de manera aislada y sin depender de otras partes del programa.

  - Verifica una unidad de código
  - De forma ágil
  - De forma independiente

Misión



- Permitir el crecimiento sostenible del proyecto de software.

4 Pilares

PROTECCIÓN CONTRA REGRESIONES

RESISTENCIA A LA REFACTORIZACIÓN

RÁPIDA RETROALIMENTACIÓN

MANTENIBILIDAD

Cobertura

Cobertura de Código
o
Cobertura de Pruebas

$$= \frac{\text{LOC Ejecutadas}}{\text{LOC Totales}}$$

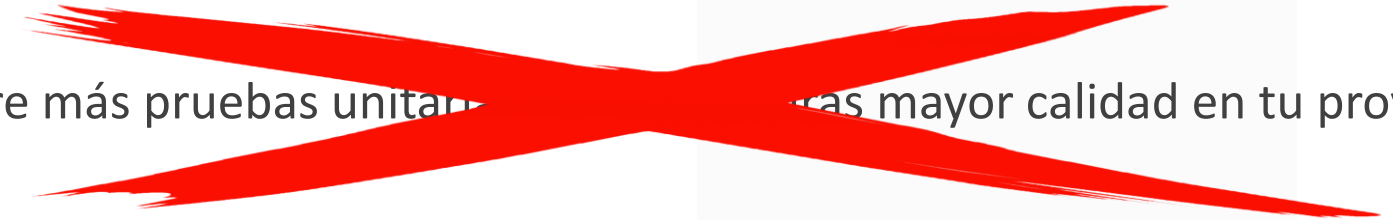
```
public static bool IsStringLong(string input)
{
    return input.Length > 5;
}
```

```
public void Test()
{
    bool result = IsStringLong("abc");
    Assert.Equal(false, result);
}
```

100%

Cobertura

- “Entre más pruebas unitarias mayor calidad en tu proyecto”



- “Las métricas de cobertura son un buen indicador negativo pero un mal indicador positivo”

- La cobertura es un **INDICADOR**, no es un **OBJETIVO**

Anatomía

- El más común es el patrón **AAA**.

```
public class CalculatorTests
{
    [Fact]
    public void Sum_of_two_numbers()
    {
        // Arrange
        double first = 10;
        double second = 20;
        var calculator = new Calculator();

        // Act
        double result = calculator.Sum(first, second);

        // Assert
        Assert.Equal(30, result);
    }
}
```

Arrange: Inicializa y Prepara

Act: Ejecuta el método o la acción

Assert: Fase de verificación

¿Qué tamaño debe tener cada sección?

```
[Fact]
public void Purchase_succeeds_when_enough_inventory()
{
    // Arrange
    var store = new Store();
    store.AddInventory(Product.Shampoo, 10);
    var customer = new Customer();

    // Act
    bool success = customer.Purchase(store, Product.Shampoo, 5);
    store.RemoveInventory(success, Product.Shampoo, 5);

    // Assert
    Assert.True(success);
    Assert.Equal(5, store.GetInventory(Product.Shampoo));
}
```

- **Arrange** es usualmente la sección más larga.
- **Act**, usualmente es una línea de código.
- **Assert**, una línea por resultado de la prueba, pero si se tiene más de una consecuencia, entonces se deben de validar todas.

Nomenclatura

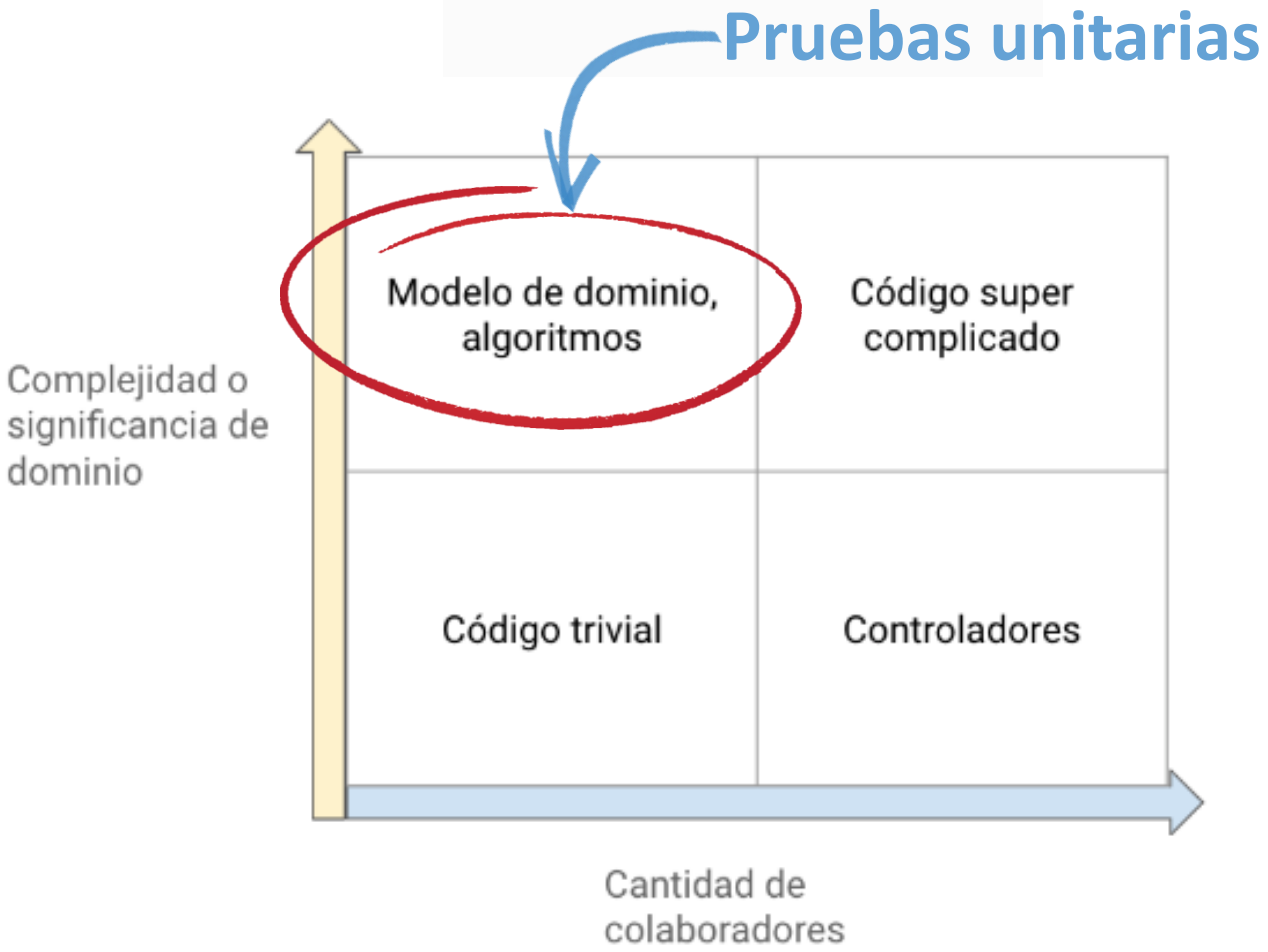
- Nombrado convencional:

```
[MethodUnderTest]_[Scenario]_[ExpectedResult]
```

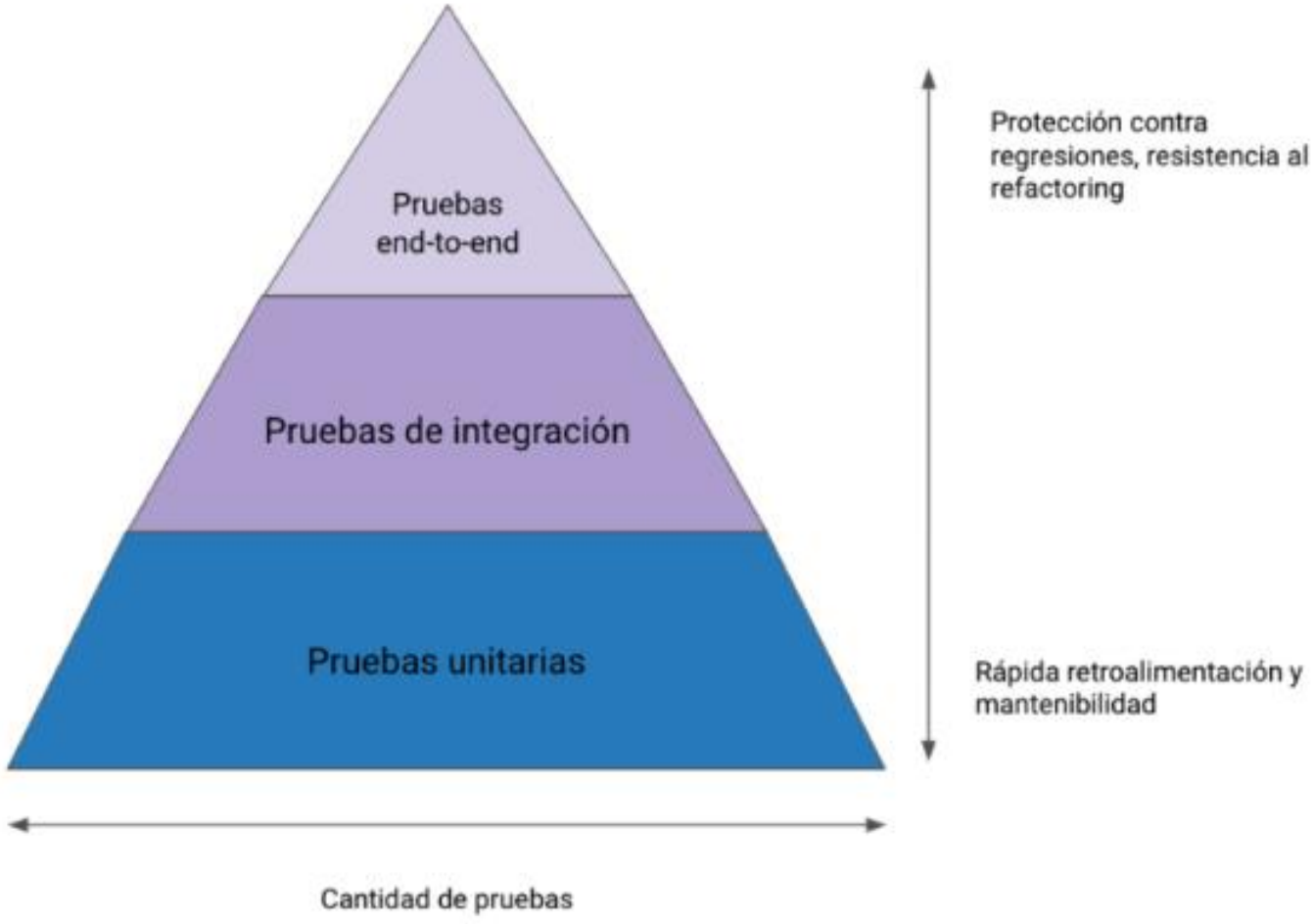
- Ejemplo:

```
public class CalculatorTests
{
    [Fact]
    public void Sum_TwoNumbers_ReturnsSum()
    {
        double first = 10;
        double second = 20;
        var sut = new Calculator();

        double result = sut.Sum(first, second);

        Assert.Equal(30, result);
    }
}
```

Cuadrante de Pruebas



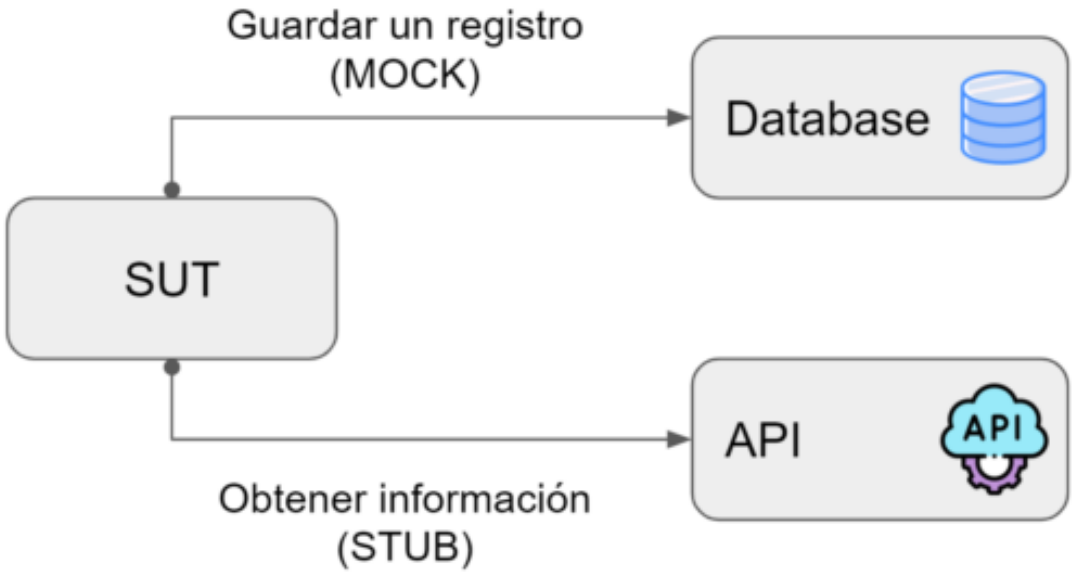
Pirámide de Pruebas



Uso de Mock

MOCKS

- Mock
- Spy



STUBS

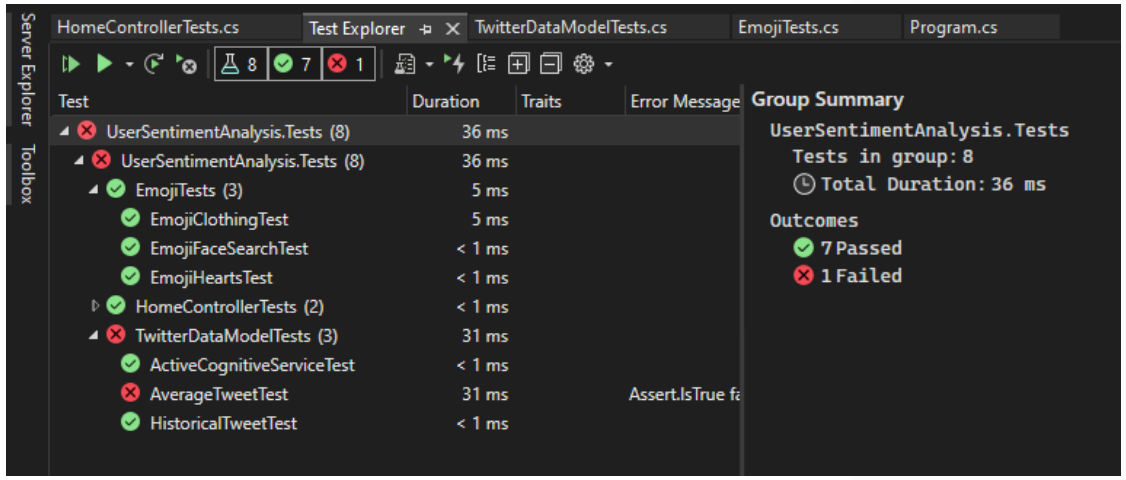
- Dummy
- Stub
- Fake

Problemas con Moq

- Recopilando información de correo electrónico.
- SponsorLink es de Código cerrado.

Mitos

- Ya estamos haciendo pruebas unitarias.



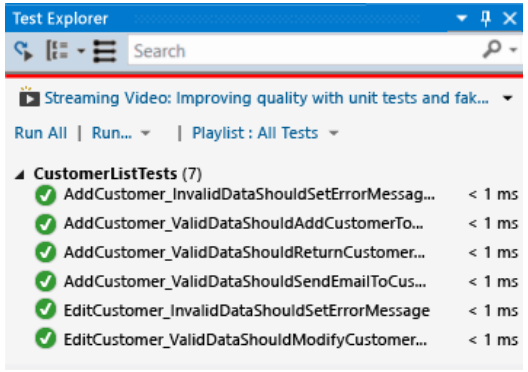
- El equipo de pruebas es responsable de mantener la calidad.

- No me da tiempo de hacer test.

Mitos

- Los test son opcionales.

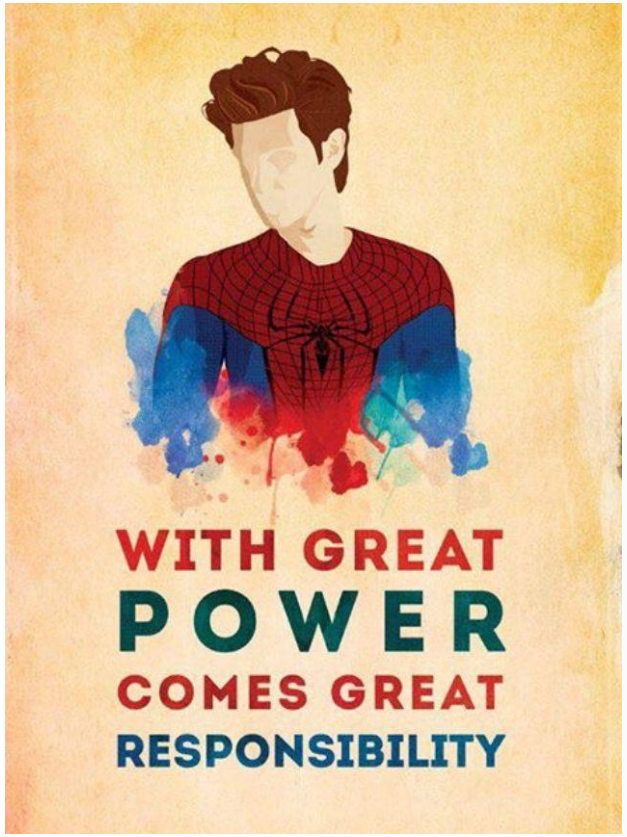
- Todo tiene que estar testeado.



- El software testeado está libre de bugs.

Reflexión

«UN GRAN PODER CONLLEVA UNA GRAN RESPONSABILIDAD»



«QUE LA FUERZA TE ACOMPAÑE»

Gracias

JANIEL BALDEMAR NOH CELIS

JANIEL BALDEMAR NOH CELIS