

AUTORES: Laura Cavero Loza, Rubén López Babón, Marc Torres Torres

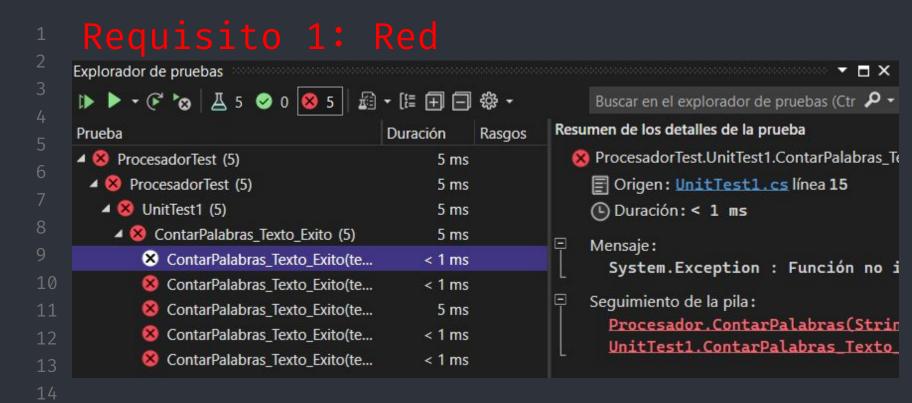
ASIGNATURA: 21783 - Test, Integración y Evolución de software

AÑO ACADÉMICO: 2021 / 2022

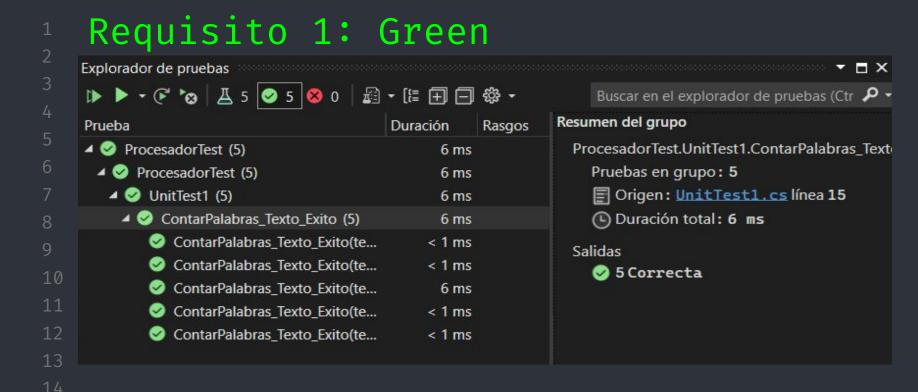
```
Requisito 1: Paso 0
public int ContarPalabras(string texto)
   throw new Exception("Función no implementada");
```

Requisito 1: Red

```
[Theory]
[InlineData("hola que tal", 3)]
[InlineData("hola", 1)]
[InlineData(" ", 0)]
[InlineData("", 0)]
[InlineData("1 } *** \n 😀 ", 5)]
0 referencias
public void ContarPalabras_Texto_Exito(string texto, int resultadoEsperado)
    //Arrange
    var sut = new Procesador();
    //Act
    int resultado = sut.ContarPalabras(texto);
    //Assert
    Assert.Equal(resultado, resultadoEsperado);
```



```
Requisito 1: Green
1 referencia | 5/5 pasando
public int ContarPalabras(string texto)
   return texto.Split(' ', StringSplitOptions.RemoveEmptyEntries).Length;
```



Requisito 2: Red

```
[Theory]
[InlineData("hola que tal", 3)]
[InlineData("hola", 1)]
[InlineData(" ", 0)]
[InlineData("", 0)]
[InlineData("1 } *** \n 😀 ", 5)]
O referencias
public void ContarPalabras_Texto_Exito(string texto, int resultadoEsperado)
    //Arrange
    var sut = new Procesador();
    //Act
    int resultado = sut.ContarPalabras(texto);
    //Assert
    Assert.Equal(resultado, resultadoEsperado);
```

Requisito 2: Red

```
[Theory]
[InlineData("hola que tal.", 3)]
[InlineData("Pepe juega. Al lol.", 2)]
[InlineData(".", 0)]
[InlineData("Pepe juega mucho. Al lol", 2)]
[InlineData("Pepe juega mucho", 3)]

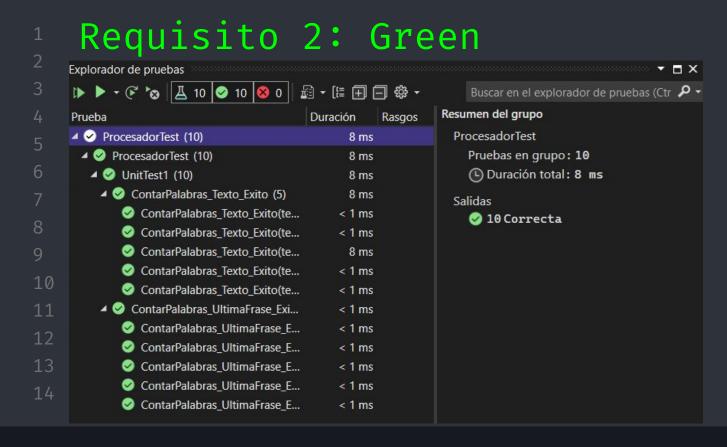
    O referencias

public void ContarPalabras_UltimaFrase_Exito(string texto, int resultadoEsperado)
   //Arrange
   var sut = new Procesador();
   //Act
   int resultado = sut.ContarPalabras(texto);
    //Assert
   Assert.Equal(resultado, resultadoEsperado);
```



Requisito 2: Green

```
2 referencias | 0 10/10 pasando
public int ContarPalabras(string texto)
    if (texto.Split(' ', StringSplitOptions.RemoveEmptyEntries).Length == 0) return 0;
    int num = 0;
    string[] frases = texto.Split('.', StringSplitOptions.RemoveEmptyEntries);
    for (int i = 0; i < frases.Length; i++)
        num = frases[i].Split(' ', StringSplitOptions.RemoveEmptyEntries).Length;
    return num;
```



```
Requisito 2: Refactor
```

```
2 referencias | 0 10/10 pasando
public int ContarPalabras(string texto)
    string[] frases = texto.Split('.', StringSplitOptions.RemoveEmptyEntries);
    if (frases.Length == 0) return 0;
    return frases[frases.Length-1].Split(' ', StringSplitOptions.RemoveEmptyEntries).Length;
```

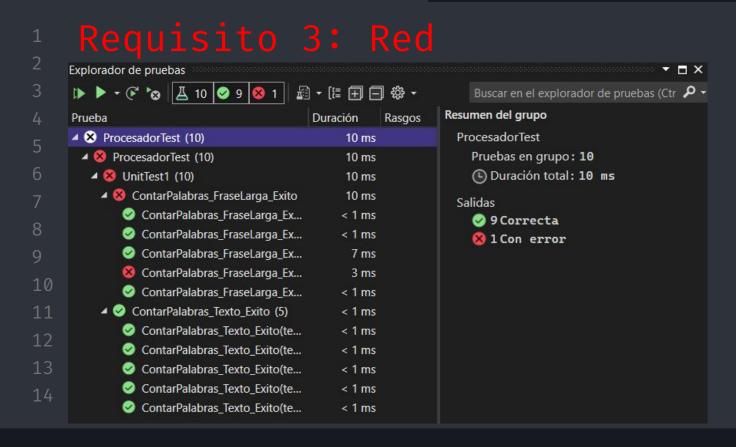


Requisito 3: Red

```
[Theory]
[InlineData("hola que tal", 3)]
[InlineData("hola", 1)]
[InlineData(" ", 0)]
[InlineData("", 0)]
[InlineData("1 } *** \n 😀 ", 5)]
0 | 0 referencias
public void ContarPalabras_Texto_Exito(string texto, int resultadoEsperado)
    //Arrange
    var sut = new Procesador();
    //Act
    int resultado = sut.ContarPalabras(texto);
    //Assert
    Assert.Equal(resultado, resultadoEsperado);
```

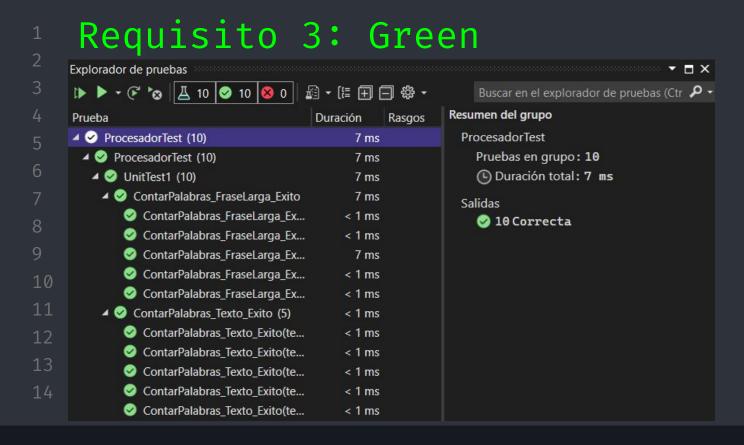
Requisito 3: Red

```
[Theory]
[InlineData("hola que tal.", 3)]
[InlineData("Pepe juega. Al lol.", 2)]
[InlineData(".", 0)]
[InlineData("Pepe juega mucho. Al lol", 3)]
[InlineData("Pepe juega mucho", 3)]
0 | 0 referencias
public void ContarPalabras_FraseLarga_Exito(string texto, int resultadoEsperado)
   //Arrange
    var sut = new Procesador();
   //Act
    int resultado = sut.ContarPalabras(texto);
   //Assert
    Assert.Equal(resultado, resultadoEsperado);
```



Requisito 3: Green

```
2 referencias | 9 10/10 pasando
public int ContarPalabras(string texto)
    if (texto.Split(' ', StringSplitOptions.RemoveEmptyEntries).Length == 0) return 0;
    int num = 0, aux = 0;
    string[] frases = texto.Split('.', StringSplitOptions.RemoveEmptyEntries);
    for (int i = 0; i < frases.Length; i++)
        aux = frases[i].Split(' ', StringSplitOptions.RemoveEmptyEntries).Length;
        if (aux > num) num = aux;
    return num;
```



```
Requisito 4: Red
```

```
[Theory]
[InlineData("hola que tal", 3)]
[InlineData("hola", 1)]
//[InlineData(" ", 0)]
//[InlineData("", 0)]
[InlineData("1 } *** \n 😀 ", 5)]
0 | 0 referencias
public void ContarPalabras_Texto_Exito(string texto, int resultadoEsperado)
    //Arrange
    var sut = new Procesador();
   //Act
    int resultado = sut.ContarPalabras(texto);
    //Assert
    Assert.Equal(resultado, resultadoEsperado);
```

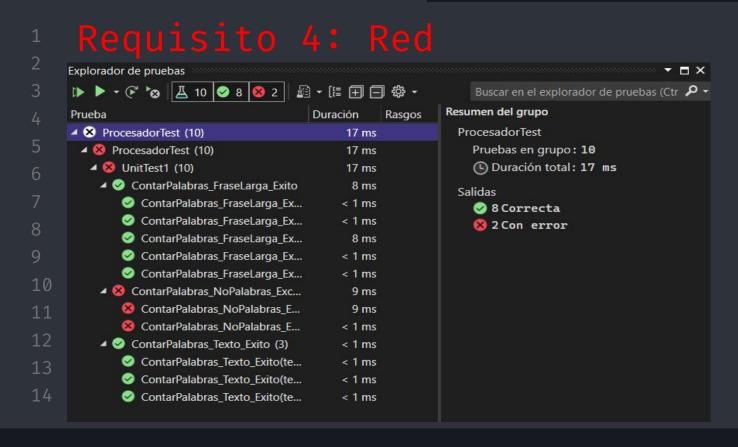
Requisito 4: Red

```
[Theory]
[InlineData("hola que tal.", 3)]
[InlineData("Pepe juega. Al lol.", 2)]
[InlineData(".", 0)]
[InlineData("Pepe juega mucho. Al lol", 3)]
[InlineData("Pepe juega mucho", 3)]
0 | 0 referencias
public void ContarPalabras_FraseLarga_Exito(string texto, int resultadoEsperado)
   //Arrange
    var sut = new Procesador();
   //Act
    int resultado = sut.ContarPalabras(texto);
   //Assert
    Assert.Equal(resultado, resultadoEsperado);
```

```
[Theory]
[InlineData(" ")]
[InlineData("")]

    O referencias

public void ContarPalabras_NoPalabras_Exception(string texto)
    //Arrange
    var sut = new Procesador();
   //Act
    var exception = Assert.Throws<Exception>(() => sut.ContarPalabras(texto));
    //Assert
    Assert.Equal("No existe ninguna palabra", exception.Message);
```



Requisito 4: Green

```
3 referencias | ● 10/10 pasando
public int ContarPalabras(string texto)
    if (texto.Split(' ', StringSplitOptions.RemoveEmptyEntries).Length == 0)
        throw new Exception("No existe ninguna palabra");
    int num = 0, aux = 0;
    string[] frases = texto.Split('.', StringSplitOptions.RemoveEmptyEntries);
    for (int i = 0; i < frases.Length; i++)
        aux = frases[i].Split(' ', StringSplitOptions.RemoveEmptyEntries).Length;
        if (aux > num) num = aux;
    return num;
```

