

## Evidencia SpecFlow – AddProduct

The screenshot shows the SpecFlow Explorer interface. On the left, a tree view shows the test structure: 'SpecFlowPharmacyGo (1)' with a duration of 77 ms, 'SpecFlowPharmacyGo.Features (1)' with 77 ms, 'AddingAProductFeature (1)' with 77 ms, and 'AddingAValidProductEntry' with 77 ms and a tag 'tag1'. The right pane shows the test results for 'AddingAValidProductEntry'. It indicates that one or more step definitions are not implemented yet. The standard output shows a table of product data:

Code	Name	Description	Price	myList
1234	Shampoo-Bella	Para el Cabello	50.50	[Shampoo-Bella]
1255	Crema para manos	Tersa y Suave	150	[Shampoo-Bella, Crema para Manos]

### Feature: Adding a Product

Pharmacy employees can enter product information with the following data:

Code (unique within the pharmacy).

Name.

Description.

Price.

All the data is mandatory.

@tag1

#### Scenario: Adding a Valid Product Entry

Given that I input the <code>,  
And that I input the <name>,  
And that I input the <description>,  
And that I input the <price> into the product registration system  
When I press the "Add to List" button  
Then the result <result> displayed on the screen should show the product added to the myList <myList>:

Code	Name	Description	Price	myList
1234	Shampoo-Bella	Para el Cabello	50.50	[Shampoo-Bella]
1255	Crema para manos	Tersa y Suave	150	[Shampoo-Bella, Crema para Manos]

```
namespace SpecFlowPharmacyGo.StepDefinitions
```

```
{  
    [Binding]  
    public class AddProductStepDefinitions  
    {  
        private readonly Product _product = new Product();  
  
        [Given(@"A non-empty product name '([^']*),'")]  
        public void GivenANon_EmptyProductName(string name)  
        {  
            _product.Name = name;  
        }  
  
        [Given(@"a non-empty product description '([^']*),'")]  
        public void GivenANon_EmptyProductDescription(string description)  
        {  
            _product.Description = description;  
        }  
;  
    }  
  
    [Given(@"a non-empty product price (.*)'")]  
    public void GivenANon_EmptyProductPrice(Decimal precio)
```

```

        {
            _product.Precio = precio;
        }
    }
}

```

Se va construyendo la clase a medida que se agregan las propiedades y se completa el BDD en Gherky

```

using PharmaGo.Domain.Entities;
using System;
using TechTalk.SpecFlow;

namespace SpecFlowPharmacyGo.StepDefinitions
{
    [Binding]
    public class AddProductStepDefinitions    {

        private Product _product = new Product();
        private List<Product> myProducts = new List<Product>();

        [Given(@"A non-empty product name '([^']*),'")]
        public void GivenANon_EmptyProductName(string name)
        {
            _product.Name = name;
        }

        [Given(@"a non-empty product description '([^']*),'")]
        public void GivenANon_EmptyProductDescription(string
description)
        {
            _product.Description = description;
        }

        [Given(@"a non-empty product price (.*)'")]
        public void GivenANon_EmptyProductPrice(Decimal precio)
        {
            _product.Precio = precio;
        }

        [When(@"I select the action to add")]
        public void WhenISelectTheActionToAdd()
        {
            throw new PendingStepException();
        }

        [Then(@"the product '([^']*)' es agregado a myproducts")]
        public void ThenTheProductEsAgregadoAMyproducts(string shampoo)
        {
            string productName = "shampoo";
            string productDescription = "ideal for hair";
            double productPrice = 50.50;

            myProducts.Add(new Product(productName, productDescription,
productPrice));
        }
    }
}

```

}

```
main PharmacGo.Domain.Entities.Product Product(string productName, string productDescription, double productPrice)

namespace PharmaGo.Domain.Entities
{
    6 referencias
    public class Product
    {
        private string productName;
        private string productDescription;
        private double productPrice;

        2 referencias
        public Product(string productName, string productDescription, double productPrice)
        {
            this.productName = productName;
            this.productDescription = productDescription;
            this.productPrice = productPrice;
        }

        1 referencia
        public string Name { get; set; }
        1 referencia
        public string Description { get; set; }
        1 referencia
        public decimal Precio { get; set; }
    }
}
```

### Escenario 1: Ingreso de producto válido

**Dados:** un código de producto no vacío, un nombre de producto no vacío, una descripción de producto no vacía y un precio de producto no vacío

**Cuando:** selecciono la acción de agregar a mi lista de productos

**Entonces:** Verifica que el código no esté repetido y se agrega el producto a mi lista de productos existente.

The screenshot shows a test runner interface with a table of test results and a detailed view of a specific test.

Prueba	Duración	Rasgos	Mensaje de error
PharmaGo.Test (293)	5,9 s		
SpecFlowPharmacyGo (1)	21 ms		
SpecFlowPharmacyGo.Features (1)	21 ms		
AddingAProductFeature (1)	21 ms		
AddingAValidProductEntry	21 ms	tag1	

**Resumen de los detalles de la prueba**

AddingAValidProductEntry  
Origen: AddProduct.feature línea 12  
Duración: 21 ms

Salida estándar:

```
Given that I input the code 1234
-> done: AddingAProductStepDefinitions.GivenThatIInputTheCode(1234) (0,0s)
And that I input the name 'Shampoo-Bella'
-> done: AddingAProductStepDefinitions.GivenThatIInputTheName("Shampoo-Bella") (0,0s)
And that I input the description 'Para el Cabello',
-> done: AddingAProductStepDefinitions.GivenThatIInputTheDescription("Para el Cabello") (0,0s)
And that I input the price 50.50,
-> done: AddingAProductStepDefinitions.GivenThatIInputThePrice(50,5) (0,0s)
When I press the "Add" button
-> done: AddingAProductStepDefinitions.WhenIPressTheButton("Add") (0,0s)
Then the result 'Su producto se agregó de forma correcta'
-> done: AddingAProductStepDefinitions.ThenTheResult("Su producto se ag...") (0,0s)
```

**Compilación correcta**

Prueba	Duración	Rasgos	Mensaje de error
PharmaGo.Test (293)	5,9 s		
SpecFlowPharmacyGo (1)	32 ms		
SpecFlowPharmacyGo.Features (1)	32 ms		
AddingAProductFeature (1)	32 ms		
AddingAValidProductEntry	32 ms	tag1	

**Resumen del grupo**

SpecFlowPharmacyGo.Features  
Pruebas en grupo: 1  
Duración total: 32 ms

Salidas  
1 Correcta

### \*\*Escenario 2: Ingreso de producto con código vacío\*\*

**\*\*Datos\*\*:** un código de producto vacío, un nombre de producto no vacío, una descripción de producto no vacía y un precio de producto no vacío

**\*\*Cuando\*\* selecciono la acción de agregar a mi lista de productos**

**\*\*Entonces\*\***, se despliega un mensaje de error: 'El código del producto no puede ser vacío' y no se agrega el producto a mi lista

```
@tag1
Scenario: Adding a Valid Product Entry
  Given that I input the code 1234
  And that I input the name 'Shampoo-Bella'
  And that I input the description 'Para el Cabello',
  And that I input the price 50.50,
  When I press the "Add" button
  Then the result 'Su producto se agregó de forma correcta'

#
#| Code      | Name           | Description      | Price  | myList
#|-----|-----|-----|-----|-----
#| 1234      | Shampoo-Bella | Para el Cabello | 50.50  | [Shampoo-Bella]
#| 1255      | Crema para manos | Tersa y Suave   | 150    | [Shampoo-Bella,Crema para Manos]

@tag2
Scenario: Adding a Invalid Product with code vacío Entry
  Given that I input the code ''
  And that I input the name 'Shampoo-Bella'
  And that I input the description 'Para el Cabello',
  And that I input the price 50.50,
  When I press the "Add" button
  Then the result 'Su producto se agregó de forma correcta'
```

Explorador de pruebas			
Serie de pruebas finalizada: 2 pruebas (Superadas: 1; Con errores: 1; Omitidas: 0) ejecutadas en 4,8 s			
Prueba	Duración	Rasgos	Mensaje de error
PharmaGo.Test (293)	5,9 s		
SpecFlowPharmacyGo (1)	72 ms		
SpecFlowPharmacyGo.Features (1)	72 ms		
AddingAProductFeature (1)	72 ms		
AddingAValidProductEntry	72 ms	tag1	

**\*\*Escenario 3: Ingreso de producto con nombre vacío\*\***

**\*\*Datos\*\***: un código de producto no vacío, un nombre de producto vacío, una descripción de producto no vacía y un precio de producto no vacío

**\*\*Cuando\*\*** selecciono la acción de agregar a mi lista de productos

**\*\*Entonces\*\***, se despliega un mensaje de error: 'El nombre del producto no puede ser vacío' y no se agrega el producto a mi lista.

**\*\*Escenario 4: Ingreso de producto con descripción vacío\*\***

**\*\*Datos\*\***: un código de producto no vacío, un nombre de producto vacío, una descripción de producto vacía y un precio de producto no vacío

**\*\*Cuando\*\*** selecciono la acción de agregar a mi lista de productos

**\*\*Entonces\*\***, se despliega un mensaje de error: 'La descripción del producto no puede ser vacío' y no se agrega el producto a mi lista.

**\*\*Escenario 5: Ingreso de producto con precio vacío\*\***

**\*\*Datos\*\***: un código de producto no vacío, un nombre de producto vacío, una descripción de producto no vacía y un precio de producto vacío

**\*\*Cuando\*\*** selecciono la acción de agregar a mi lista de productos

**\*\*Entonces\*\***, se despliega un mensaje de error: 'El precio del producto no puede ser vacío' y no se agrega el producto a mi lista.

**\*\*Escenario 6: Ingreso de producto con precio vacío\*\***

**\*\*Datos\*\***: un código de producto no vacío, un nombre de producto vacío, una descripción de producto no vacía y un precio de producto vacío

**\*\*Cuando\*\*** selecciono la acción de agregar a mi lista de productos

**\*\*Entonces\*\***, se despliega un mensaje de error: 'El precio del producto no puede ser vacío' y no se agrega el producto a mi lista.

## Feature

**Feature:** Adding a Product

*Pharmacy employees can enter product information with the following data:*

*Code (unique within the pharmacy).*

*Name.*

*Description.*

*Price.*

*All the data is mandatory.*

@tag1

**Scenario:** Adding a Valid Product Entry

**Given** that I input the code 1234

**And** that I input the name 'Shampoo-Bella'

**And** that I input the description 'Para el Cabello',

**And** that I input the price 50.50,

**When** I press the "Add" button

**Then** the result 'Su producto se agregó de forma correcta'

#	#  Code	Name	Description	Price	
myList			Result		

#	Code	Name	Description	Price	Message
#  1234		Shampoo-Bella	Para el Cabello	50.50	
[Shampoo-Bella]					'Your product has been added to the list'
#  1255		Crema para manos	Tersa y Suave	150	
[Shampoo-Bella, Crema para Manos]					'Your product has been added to the list'

@tag2

Scenario: Adding a InValid Product with code empty

Given that I input the code ''

And that I input the name 'Shampoo-Bella'

And that I input the description 'Para el Cabello',

And that I input the price 50.50,

When I press the "Add" button

Then the result "El código de producto no puede ser vacío"

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore.Update;
using Microsoft.VisualBasic;
using PharmaGo.Domain.Entities;
using System;
using System.ComponentModel.Design;
using TechTalk.SpecFlow;
using TechTalk.SpecFlow.CommonModels;

namespace SpecFlowPharmacyGo.StepDefinitions
{
    [Binding]
    public class AddingAProductStepDefinitions
    {
        private Product _product = new Product();
        public string? OkObjectResult { get; set; }

        [Given(@"that I input the code (.*)")]
        public void GivenThatIInputTheCode(int code)
        {
            _product.Code = code;
        }

        [Given(@"that I input the code is empty (.*)")]
        public void GivenThatIInputTheCodeEmpty(int code)
        {
            _product.Code = code;
        }

        [Then(@"the result '([^\']*)*'")]
        public void ThenTheResult(string result)
        {
            OkObjectResult = result;
        }

        [Given(@"that I input the name '([^\']*)*'")]
        public void GivenThatIInputTheName(string name)
        {
            _product.Name = name;
        }
    }
}
```

```

[Given(@"that I input the description '([^']*),'")]
public void GivenThatIInputTheDescription(string description)
{
    _product.Description = description;
}

[Given(@"that I input the price (.*)")]
public void GivenThatIInputThePrice(double price)
{
    _product.Price = price;
}

[Then(@"the result ""El código de producto no puede ser vacío'")]
public void ThenTheResultElCodigoDeProductoNoPuedeSerVacio()
{
    string expectedMessage = "El código de producto no puede ser
vacío";
    if (OkObjectResult != expectedMessage)
    {
        throw new Exception($"El mensaje esperado es
'{expectedMessage}' pero se obtuvo '{OkObjectResult}'");
    }
}

[When(@"I press the ""([^\"]*)"" button")]
public void WhenIPressTheButton(string add)
{
    OkObjectResult = add;
}
}
}

```