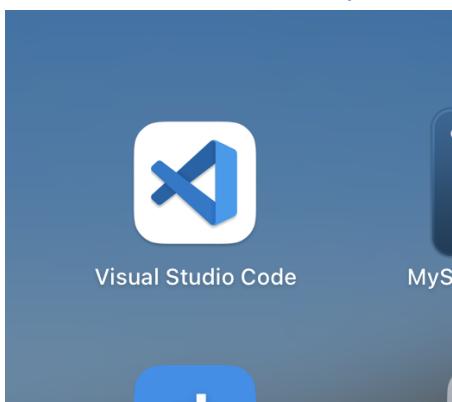


## Trabajo Práctico 9 - Pruebas Unitarias

### 5- Desarrollo de Pruebas Unitarias sobre una aplicación de consola.

5.1 Preparamos el entorno:

- Instalamos VS.Code: <https://code.visualstudio.com/download>

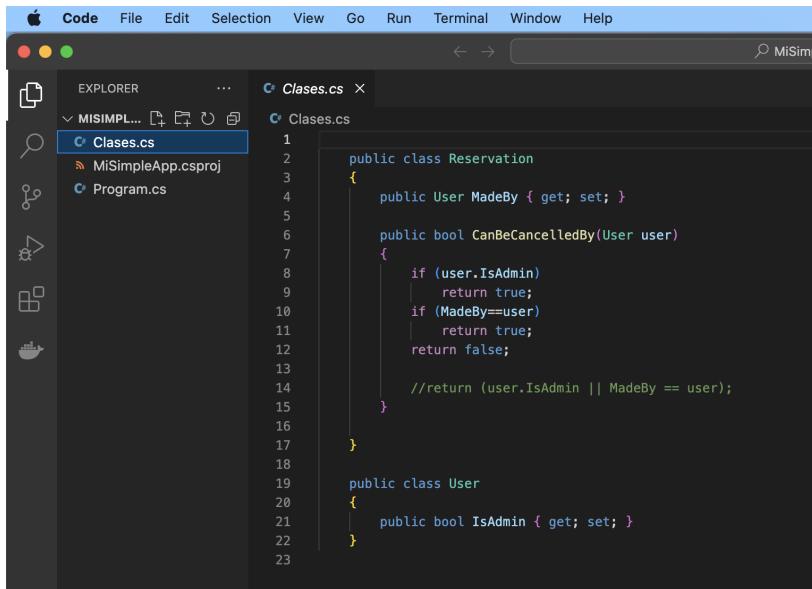


- Desde linea de comandos clonamos el proyecto MiSimpleApp, entramos a la carpeta y abrimos VS.Code

```
Last login: Tue Oct  3 15:47:25 on ttys001
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % git clone https://github.com/ingsoft3ucc/MiSimpleApp.git
Cloning into 'MiSimpleApp'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % cd MiSimpleApp
```

A screenshot of a terminal window on a Mac. The terminal shows the command `git clone https://github.com/ingsoft3ucc/MiSimpleApp.git` being run. The output of the command is displayed, showing the cloning process: it lists 5 objects, counts them as 100% complete, compresses them, and then receives them all at 100% completion. Finally, it changes the directory to `MiSimpleApp`. The terminal has a dark theme with red, yellow, and green status indicators at the top.

- Revisamos el código
- Cerramos VS.Code



```
public class Reservation
{
    public User MadeBy { get; set; }

    public bool CanBeCancelledBy(User user)
    {
        if (user.IsAdmin)
            return true;
        if (MadeBy==user)
            return true;
        return false;
        //return (user.IsAdmin || MadeBy == user);
    }
}

public class User
{
    public bool IsAdmin { get; set; }
}
```

- Nos movemos una carpeta hacia arriba y creamos un nuevo proyecto de pruebas unitarias con NUnit:

```
pedrofernandez@MacBook-Air-de-Pedro MiSimpleApp % cd ..
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % dotnet new nunit -n MiSimpleAppTests
[La plantilla "NUnit 3 Test Project" se creó correctamente.

Procesando acciones posteriores a la creación...
Restaurando /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitari
Determinando los proyectos que se van a restaurar...
Se ha restaurado /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-
Restauración realizada correctamente.

pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias %
```

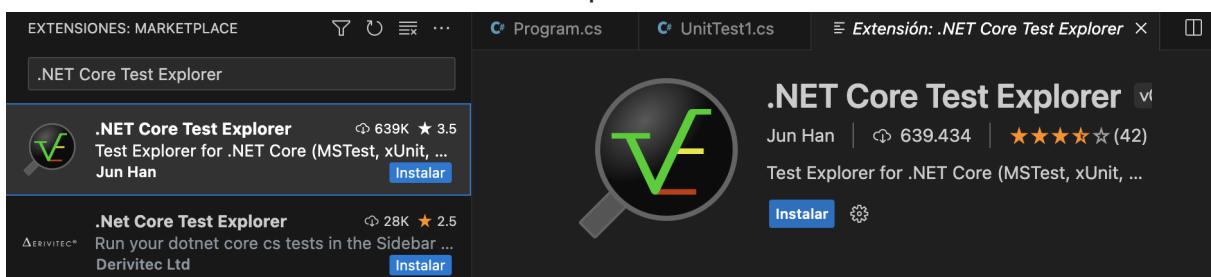
- Entramos a la carpeta del nuevo proyecto y agregamos los paquetes NUnit y NUnit.ConsoleRunner. Luego le agregamos al proyecto de pruebas una referencia al proyecto que vamos a probar. Nos movemos una carpeta hacia arriba y lo vemos en VS.Code

```

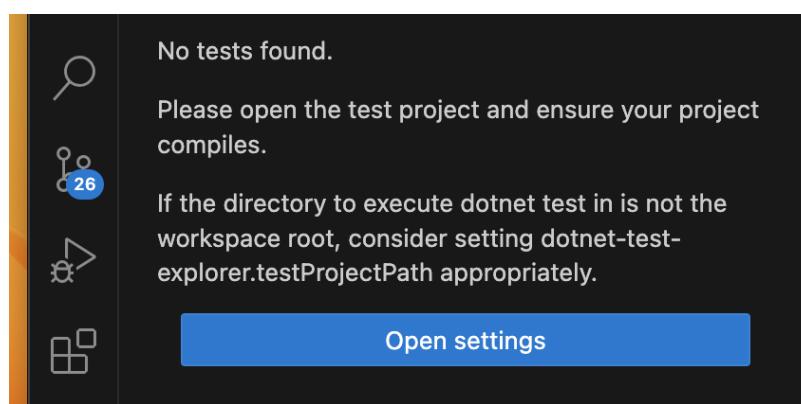
pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % cd MiSimpleAppTests
pedrofernandez@MacBook-Air-de-Pedro MiSimpleAppTests % dotnet add package NUnit
  Determinando los proyectos que se van a restaurar...
    Writing /var/folders/4c/516vr0dj4qsgd87fmw6nj4dr0000gn/T/tmpXV9pGs.tmp
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/local/share/dotnet/sdk'
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/local/share/dotnet/sdk'
[info : Agregando PackageReference para el paquete "NUnit" al proyecto "/Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
info : GET https://api.nuget.org/v3/registration5-gz-semver2/nunit/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/nunit/index.json 1126 ms
info : Restaurando paquetes para /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
info : El paquete "NUnit" es compatible con todos los marcos de trabajo especificados del proyecto "/Users/pedrofernandez/MiSimpleAppTests.csproj".
info : Se actualizó PackageReference para la versión "3.13.3" del paquete "NUnit" en el archivo "/Users/pedrofernandez/MiSimpleAppTests.csproj".
info : El archivo de recursos no ha cambiado, así que se omitirá su escritura. Ruta de acceso: /Users/pedrofernandez/project.assets.json
log : Se ha restaurado /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
pedrofernandez@MacBook-Air-de-Pedro MiSimpleAppTests % dotnet add package NUnit.ConsoleRunner
  Determinando los proyectos que se van a restaurar...
    Writing /var/folders/4c/516vr0dj4qsgd87fmw6nj4dr0000gn/T/tmpcrXA0g.tmp
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/local/share/dotnet/sdk'
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/local/share/dotnet/sdk'
info : Agregando PackageReference para el paquete "NUnit.ConsoleRunner" al proyecto "/Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
info : GET https://api.nuget.org/v3/registration5-gz-semver2/nunit.consolerrunner/index.json
info : OK https://api.nuget.org/v3/registration5-gz-semver2/nunit.consolerrunner/index.json 1184 ms
info : Restaurando paquetes para /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
info : GET https://api.nuget.org/v3-flatcontainer/nunit.consolerrunner/index.json
info : OK https://api.nuget.org/v3-flatcontainer/nunit.consolerrunner/index.json 1026 ms
info : GET https://api.nuget.org/v3-flatcontainer/nunit.consolerrunner/3.16.3/nunit.consolerrunner.3.16.3.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/nunit.consolerrunner/3.16.3/nunit.consolerrunner.3.16.3.nupkg 54
info : Se instaló NUnit.ConsoleRunner 3.16.3 de https://api.nuget.org/v3/index.json con el hash de contenido bhc4f
info : El paquete "NUnit.ConsoleRunner" es compatible con todos los marcos de trabajo especificados del proyecto "/Users/pedrofernandez/MiSimpleAppTests.csproj".
[info : Se agregó PackageReference para la versión "3.16.3" del paquete "NUnit.ConsoleRunner" al archivo "/Users/pedrofernandez/MiSimpleAppTests.csproj".
info : Generación de archivo MSBuild /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
info : Escribiendo el archivo de recursos en el disco. Ruta de acceso: /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
log : Se ha restaurado /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests.csproj".
pedrofernandez@MacBook-Air-de-Pedro MiSimpleAppTests % dotnet add reference ../../MiSimpleApp/MiSimpleApp.csproj
Se ha agregado la referencia "..\MiSimpleApp\MiSimpleApp.csproj" al proyecto.
pedrofernandez@MacBook-Air-de-Pedro MiSimpleAppTests %

```

- Instalamos extensión ".NET Core Test Explorer"



- Nos aparece un ícono de Pruebas, lo seleccionamos, hacemos click en *Open Settings*



- Se abre el archivo settings.json y escribimos el nombre del proyecto de pruebas y el nombre de la dll resultante de la compilación del proyecto de pruebas:

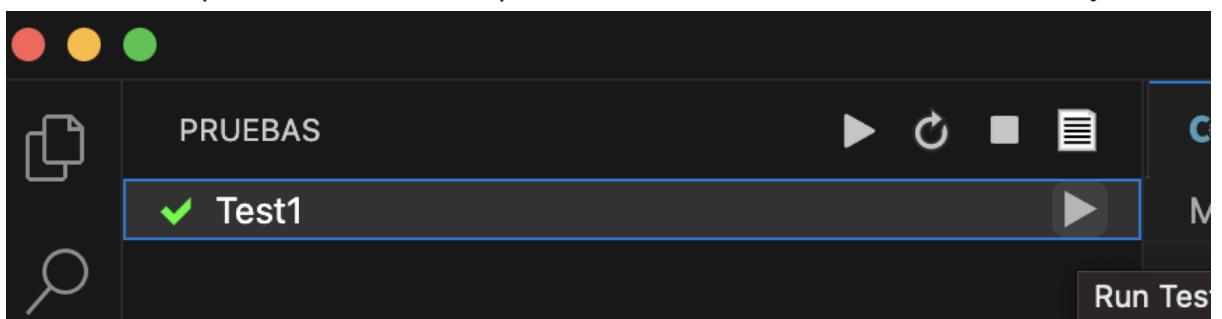


```

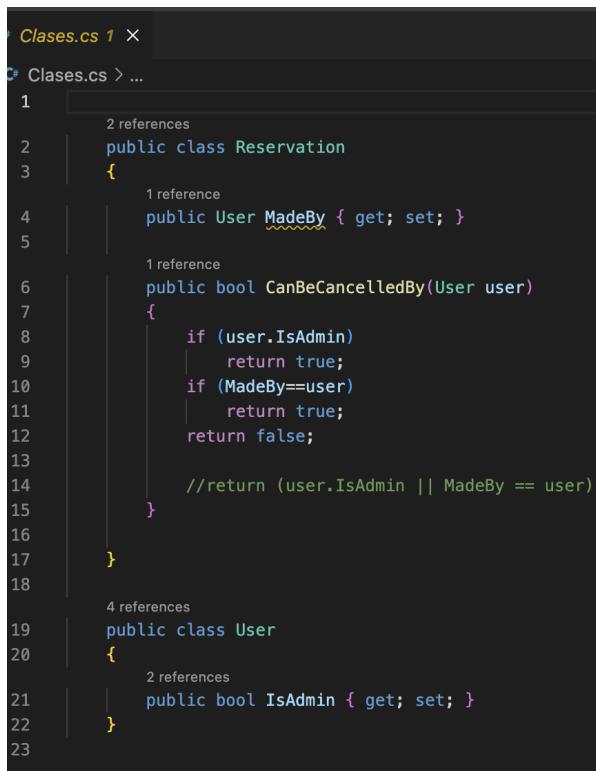
Extension: .NET Core Test Explorer    {} settings.json X
.vscode > {} settings.json > ...
1  {
2      "dotnet-test-explorer.testProjectPath": "MiSimpleAppTests",
3      "dotnet-test-explorer.testFileSuffix": ".Tests.dll"
4  }

```

- Guardamos el archivo, volvemos hacer click en el ícono de pruebas y ahora sí tenemos la posibilidad de correr pruebas haciendo click en el botón de *Play*



- Revisamos el código del proyecto MiSimpleApp:
  - Clases.cs:



```

Clases.cs 1 ×
Clases.cs > ...
1
2     2 references
3     public class Reservation
4     {
5         1 reference
6         public User MadeBy { get; set; }
7
8         1 reference
9         public bool CanBeCancelledBy(User user)
10        {
11            if (user.IsAdmin)
12                return true;
13            if (MadeBy==user)
14                return true;
15            return false;
16
17        }
18
19     4 references
20     public class User
21     {
22         2 references
23         public bool IsAdmin { get; set; }

```

- Program.cs:

```
C# Program.cs X

C# Program.cs > ...

1  using System;
2  ...
3  class Program
4  {
5      ...
6      static void Main()
7      {
8          Reservation reservation = new Reservation();
9          User user=new User();
10         user.IsAdmin=true;
11         bool result=reservation.CanBeCancelledBy(user);
12         Console.WriteLine(result);
13     }
14 }
```

## 5.2: Creamos nuestros Tests:

- Dado que el método *CanBeCancelledBy* de la clase *Reservation* tiene una lógica con 3 caminos posibles, debemos probar esos 3 caminos:
- Modificamos nuestro archivo *UnitTest1.cs* del proyecto *MiSimpleAppTests*

## 5.3 Explicamos detalladamente los tests en el entregable

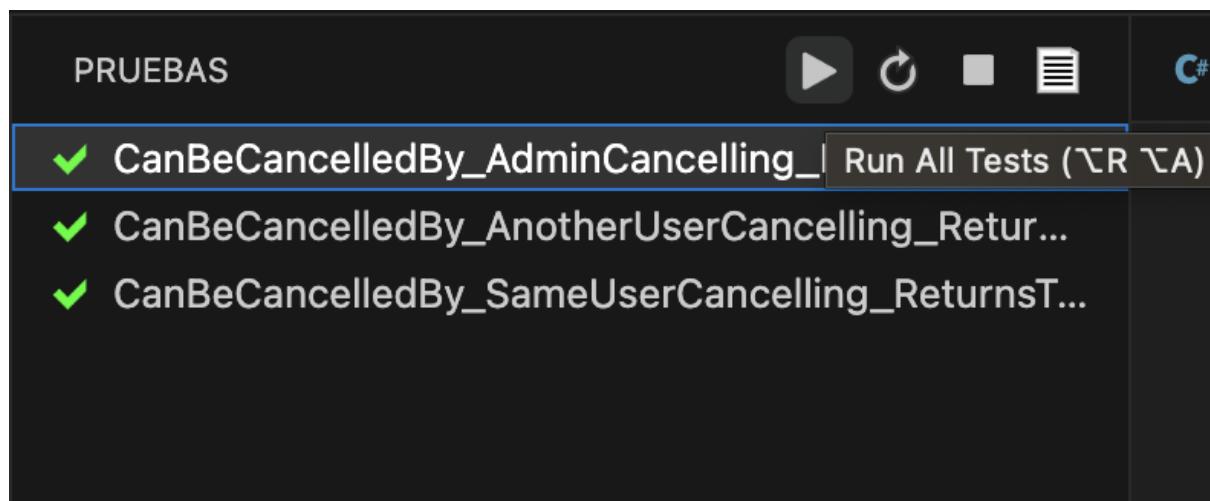
The screenshot shows the Visual Studio interface. On the left, the 'EXPLORER' pane is open, showing a project structure with 'MISIMPLEAPPTESTS' expanded, containing 'bin', 'obj', 'GlobalUsings.cs', 'MiSimpleAppTests.cs...', and 'UnitTest1.cs'. The 'UnitTest1.cs' file is selected. On the right, the 'UnitTest1.cs X' editor window is displayed, showing the following C# code:

```
C# UnitTest1.cs X

C# UnitTest1.cs > ...

1  namespace MiSimpleAppTests;
2  ...
3  [TestFixture]
4  public class Tests
5  {
6      [SetUp]
7      public void Setup()
8      {
9      }
10
11     [Test]
12     public void CanBeCancelledBy_Admin Cancelling_ReturnsTrue()
13     {
14         //Arrange
15
16         User user=new User();
17         user.IsAdmin=true;
18         Reservation reservation=new Reservation();
19
20         //Act
21
22         bool result=reservation.CanBeCancelledBy(user);
23
24         //Assert
25
26         //Assert.IsTrue(result);
27         Assert.That(result,Is.True);
28     }
29 }
```

## 5.4 Ejecutamos los tests:

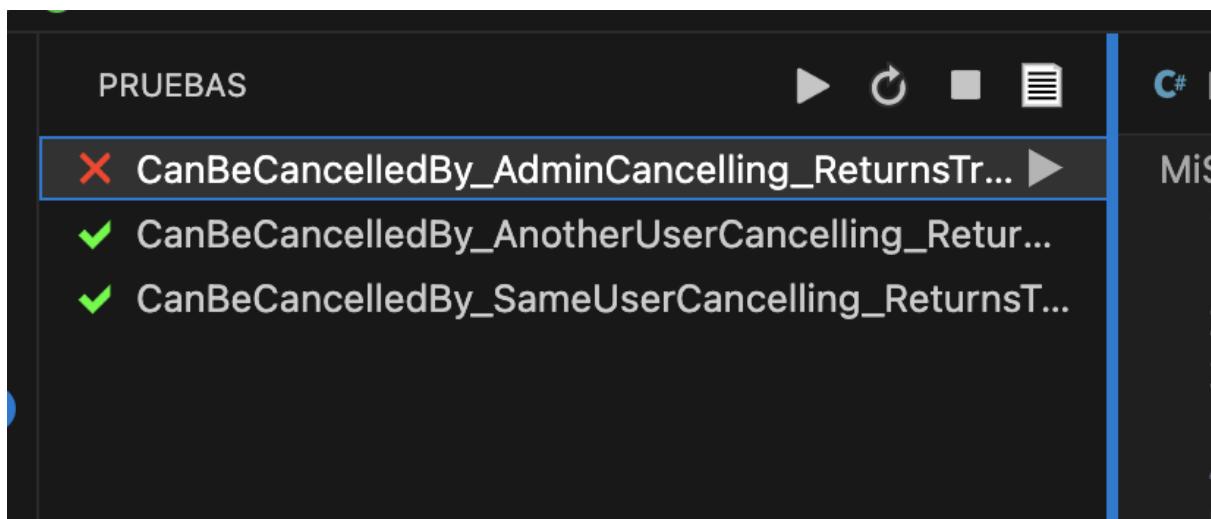


- Modificamos la lógica de nuestro código bajo prueba haciendo que devuelva false la linea 9 (recordar guardar el archivo):

```
Clases.cs 68
C# Program.cs C# Clases.cs 1 × C# UnitTest1.cs
MiSimpleApp > C# Clases.cs > 📁 Reservation > ⚙️ CanBeCancelledBy
1
2     8 referencias
3         public class Reservation
4             {
5                 3 referencias
6                     public User MadeBy { get; set; }
7
8                         4 referencias
9                             bool Reservation.CanBeCancelledBy(User user)
10
11                         public bool CanBeCancelledBy(User user)
12                             {
13                                 if (user.IsAdmin)
14                                     return true;
```

```
C# Program.cs C# Clases.cs 1 ● C# UnitTest1.cs
MiSimpleApp > C# Clases.cs > 📁 Reservation > ⚙️ CanBeCancelledBy
1
2     8 referencias
3         public class Reservation
4             {
5                 3 referencias
6                     public User MadeBy { get; set; }
7
8                         4 referencias
9                             public bool CanBeCancelledBy(User user)
10
11                             if (user.IsAdmin)
12                                 return false;
```

- Volvemos a ejecutar los tests:



- Dejamos la línea 9 de nuestro código bajo prueba como estaba originalmente:

The screenshot shows the Visual Studio code editor with the file "Clases.cs" open. The code defines a class "Reservation" with a method "CanBeCancelledBy". The line "return true;" is highlighted with a yellow oval icon, indicating it is under test. The code editor interface includes tabs for "Program.cs", "Clases.cs", and "UnitTest1.cs", and a status bar at the bottom.

```

1     8 referencias
2     public class Reservation
3     {
4         3 referencias
5         public User MadeBy { get; set; }
6         4 referencias
7         public bool CanBeCancelledBy(User user)
8         {
9             if (user.IsAdmin)
    |         return true;

```

- Realizamos una refactorización de nuestro código y guardamos el archivo:

C# Program.cs    C# Clases.cs 1 ● C# UnitTest1.cs 1

MiSimpleApp > C# Clases.cs > Reservation

```

1
2     8 referencias
3     public class Reservation
4     {
5         3 referencias
6         public User MadeBy { get; set; }
7
8         4 referencias
9         public bool CanBeCancelledBy(User user)
10        {
11            // if (user.IsAdmin)
12            //     return true;
13            // if (MadeBy==user)
14            //     return true;
15            // return false;
16
17        }
18
19        12 referencias
20        public class User
21        {
22            3 referencias
23            public bool IsAdmin { get; set; }

```

- Por último ejecutamos nuestros tests desde la linea de comandos. Nos posicionamos en el directorio de nuestro proyecto de pruebas y ejecutamos el comando dotnet test

```

Last login: Wed Oct  4 12:32:59 on ttys001
[pedrofernandez@MacBook-Air-de-Pedro MiSimpleAppTests % dotnet test
Determinando los proyectos que se van a restaurar...
Todos los proyectos están actualizados para la restauración.
MiSimpleApp -> /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleApp/bin/Debug
MiSimpleAppTests -> /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests
Serie de pruebas para /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiSimpleAppTests
Herramienta de línea de comandos de ejecución de pruebas de Microsoft(R), versión 17.7.0-preview-23364-03+bc17bb9693cfc4778ded5
Copyright (c) Microsoft Corporation. Todos los derechos reservados.

Iniciando la ejecución de pruebas, espere...
1 archivos de prueba en total coincidieron con el patrón especificado.

Correctas! - Con error:    0, Superado:    3, OMITIDO:    0, Total:    3, Duración: 4 ms - MiSimpleAppTests.dll (net7.0)
pedrofernandez@MacBook-Air-de-Pedro MiSimpleAppTests %

```

## 6- Desarrollo de Pruebas Unitarias sobre una WebAPI:

### 6.1 Preparamos el entorno:

- Clonamos nuestro repo SimpleWebAPI

```
Last login: Wed Oct  4 12:33:12 on ttys000
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % git clone https://github.com/ingsoft3ucc/SimpleWebAPI.git
Cloning into 'SimpleWebAPI'...
remote: Enumerating objects: 150, done.
remote: Counting objects: 100% (150/150), done.
remote: Compressing objects: 100% (115/115), done.
remote: Total 150 (delta 60), reused 33 (delta 9), pack-reused 0
Receiving objects: 100% (150/150), 28.41 KiB | 1.09 MiB/s, done.
Resolving deltas: 100% (60/60), done.
pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias %
```

- Creamos proyecto de pruebas

```
Resolving deltas: 100% (60/60), done.
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % dotnet new nunit -n SimpleWebAPITests
La plantilla "NUnit 3 Test Project" se creó correctamente.

Procesando acciones posteriores a la creación...
Restaurando /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/SimpleWebAPITests/SimpleWebAPITests.csproj:
  Determinando los proyectos que se van a restaurar...
    Se ha restaurado /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/SimpleWebAPITests/SimpleWebAPITests.csproj (en 102 ms).
  Restauración realizada correctamente.

pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias %
```

- Entramos a la carpeta del nuevo proyecto y le agregamos al proyecto de pruebas una referencia al proyecto que vamos a probar. Nos movemos una carpeta hacia arriba y lo vemos en VS.Code

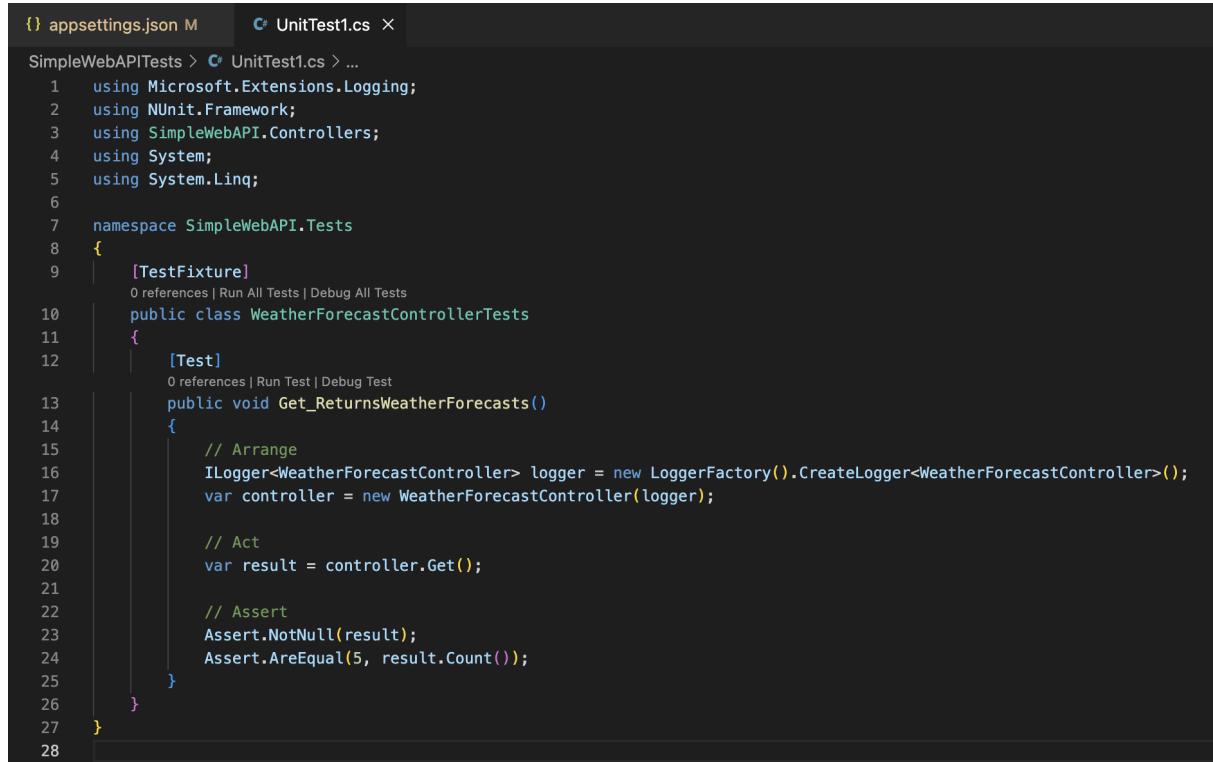
```
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % cd SimpleWebAPITests ]
[pedrofernandez@MacBook-Air-de-Pedro SimpleWebAPITests % dotnet add reference ..\SimpleWebAPI\SimpleWebAPI\SimpleWebAPI.csproj
  Se ha agregado la referencia "..\SimpleWebAPI\SimpleWebAPI\SimpleWebAPI.csproj" al proyecto.
[pedrofernandez@MacBook-Air-de-Pedro SimpleWebAPITests % cd ..
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % code .
zsh: command not found: code
pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias %
```

- Se abre el archivo settings.json y escribimos el nombre del proyecto de pruebas y el nombre de la dll resultante de la compilación del proyecto de pruebas:

```
{} appsettings.json M X
SimpleWebAPI > SimpleWebAPI > {} appsettings.json > ...
1  {
2   "dotnet-test-explorer.testProjectPath": "SimpleWebAPITests",
3   "dotnet-test-explorer.testFileSuffix": ".Tests.dll"
4 }
```

## 6.2 : Creamos Tests

- Reemplazamos el código de *UnitTest1.cs* por:



```
{ appsettings.json M C# UnitTest1.cs X
SimpleWebAPITests > C# UnitTest1.cs > ...
1  using Microsoft.Extensions.Logging;
2  using NUnit.Framework;
3  using SimpleWebAPI.Controllers;
4  using System;
5  using System.Linq;
6
7  namespace SimpleWebAPI.Tests
8  {
9      [TestFixture]
10     0 references | Run All Tests | Debug All Tests
11     public class WeatherForecastControllerTests
12     {
13         [Test]
14         0 references | Run Test | Debug Test
15         public void Get_ReturnsWeatherForecasts()
16         {
17             // Arrange
18             ILogger<WeatherForecastController> logger = new LoggerFactory().CreateLogger<WeatherForecastController>();
19             var controller = new WeatherForecastController(logger);
20
21             // Act
22             var result = controller.Get();
23
24             // Assert
25             Assert.NotNull(result);
26             Assert.AreEqual(5, result.Count());
27         }
28     }
}
```

## 6.3 Explicamos en el entregable que hace nuestro test

El test unitario escrito en C# utiliza el framework de pruebas NUnit. Lo que hace es verificar el comportamiento del método Get en la clase WeatherForecastController en la aplicación web API simple.

Basicamente, verifica que el método Get del controlador WeatherForecastController devuelve una lista de pronósticos del tiempo que no es nula y que tiene exactamente 5 elementos. Este es un ejemplo para asegurarse que la aplicación siga funcionando correctamente a medida que se realizan cambios en el código.

## 6.4 Ejecutamos el test:

```
UnitTest1.cs - 08_02
PRUEBAS
Get_ReturnsWeatherForecasts
PITests > UnitTest1.cs > WeatherForecastControllerTests > Get_ReturnsWeatherForecasts
namespace SimpleWebAPI.Tests
{
    [TestFixture]
    public class WeatherForecastControllerTests
    {
        [Test]
        public void Get_ReturnsWeatherForecasts()
        {
            // Arrange
            ILogger<WeatherForecastController> logger = new Logger();
            var controller = new WeatherForecastController(logger);

            // Act
            var result = controller.Get();

            // Assert
            Assert.NotNull(result);
            Assert.AreEqual(5, result.Count());
        }
    }
}

PROBLEMAS SALIDA ... Test Explorer (Test runner) Microsoft (R) Test Execution Command Line Tool Version 17.6.3 (x64)
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.
Results File: /var/folders/8s/vfs50kh14r377stwqyrcnd6r0000gn/T/test-explorer-PqAv95/0.trx

Passed! - Failed: 0, Passed: 1, Skipped: 0, Total: 1,
Duration: 49 ms - SimpleWebAPITests.dll (net7.0)
```

## 7- Familiarizarse con algunos conceptos de Mock

SOLO TEÓRICO...

## 8- Utilizando Moq

### *Intro*

Moq es un marco de pruebas y simulación (mocking framework) para el lenguaje de programación C# en el entorno de desarrollo de .NET. Permite a los desarrolladores crear objetos simulados, llamados "mocks" o "stubs", para simular el comportamiento de componentes del sistema durante las pruebas unitarias.

Algunas de las características y ventajas clave de Moq incluyen:

Sintaxis Fluent: Moq utiliza una sintaxis fluent y expresiva que facilita la creación y configuración de objetos simulados. Esto hace que las pruebas sean más legibles y mantenibles.

**Generación Dinámica:** Moq genera objetos simulados en tiempo de ejecución, lo que significa que no es necesario escribir clases separadas para implementar mocks. Esto ahorra tiempo y reduce la complejidad del código de prueba.

**Configuración de Comportamiento:** Puedes configurar cómo debe comportarse un objeto simulado cuando se llama a sus métodos o propiedades. Esto incluye especificar los valores de retorno, establecer acciones personalizadas y verificar si se han llamado métodos específicos.

**Verificación de Llamadas:** Moq permite verificar si se han llamado los métodos simulados y cuántas veces se han llamado. Esto es útil para asegurarse de que el código bajo prueba interactúa correctamente con sus dependencias simuladas.

**Soporte para Pruebas Parametrizadas:** Moq es compatible con pruebas parametrizadas, lo que significa que puedes ejecutar la misma prueba con múltiples conjuntos de datos o escenarios, cambiando la configuración de los mocks según sea necesario.

**Integración con Marcos de Pruebas:** Moq se integra bien con marcos de pruebas populares como NUnit y xUnit.NET, lo que facilita la incorporación de mocks en tus pruebas unitarias.

**Ligero y de Código Abierto:** Moq es una biblioteca de código abierto y liviana que no agrega una sobrecarga significativa a tu proyecto.

En resumen, Moq es una herramienta valiosa para escribir pruebas unitarias efectivas en C#. Permite a los desarrolladores crear mocks de manera rápida y sencilla para simular el comportamiento de las dependencias y componentes externos, lo que facilita la prueba aislada de unidades de código y la identificación de problemas en el código durante el desarrollo.

## 8.1 Preparamos el entorno.

- Clonamos una app de consola en .NET Core que hace uso de un servicio externo (una llamada a una API Rest) y la abrimos en VS.Code

```
[pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias % git clone https://git
hub.com/ingsoft3ucc/MiNotSoSimpleApp.git
Cloning into 'MiNotSoSimpleApp'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 22 (delta 6), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (22/22), 5.19 KiB | 1.73 MiB/s, done.
Resolving deltas: 100% (6/6), done.
pedrofernandez@MacBook-Air-de-Pedro 09-pruebas-unitarias %
```

- Ejecutamos la app y vemos como nos devuelve 100 items desde la API:

The screenshot shows a code editor window with the title bar "MiNotSoSimpleApp". The main area displays the content of `Program.cs`:

```
C# Program.cs X
C# Program.cs > ...
1  using System;
2  using System.Net.Http;
3  using System.Net.Http.Json;
4  using System.Threading.Tasks;
5  using Microsoft.Extensions.DependencyInjection;
6
7  namespace MyApp
8  {
9      0 referencias
10     class Program
11     {
12         0 referencias
13         static async Task Main()
14         {
15             var serviceProvider = new ServiceCollection()
16                 .AddHttpClient()
17                 .AddTransient<IApiService, ApiService>()
18                 .BuildServiceProvider();
19
20             var apiService = serviceProvider.GetRequiredService<IApiService>();
21
22             var myModels = await apiService.GetMyModelsAsync();
23             foreach (var model in myModels)
24             {
25                 Console.WriteLine($"Id: {model.Id}, Title: {model.Title}");
26             }
27         }
28     }
29 }
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The DEBUG CONSOLE tab is selected, showing the output of the application's execution:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Id: 90, Title: ad iusto omnis odit dolor voluptatibus
Id: 91, Title: aut amet sed
Id: 92, Title: ratione ex tenetur perferendis
Id: 93, Title: beatae soluta recusandae
Id: 94, Title: qui qui voluptates illo iste minima
Id: 95, Title: id minus libero illum nam ad officiis
Id: 96, Title: quaerat velit veniam amet cupiditate aut numquam ut sequi
Id: 97, Title: quas fugiat ut perspiciatis vero provident
Id: 98, Title: laboriosam dolor voluptates
Id: 99, Title: temporibus sit alias delectus eligendi possimus magni
Id: 100, Title: at nam consequatur ea labore ea harum
El programa "[88618] MiNotSoSimpleApp.dll" terminó con el código 0 (0x0).
```

- Analizamos el código
- Identificamos el servicio externo y su interfaz a mockear. Incluir en la entrega una explicación del código

El código en `Program.cs` es una aplicación hecha en C# que se conecta a un servicio externo a través de HTTP para obtener datos. Utiliza la inyección de dependencias para configurar y acceder a un servicio llamado `IApiService`. Para probar este código, se necesita crear una versión simulada de `IApiService` que devuelva datos ficticios en lugar de hacer solicitudes reales a un servicio externo.

El servicio externo a "mockear" en el código es la interfaz `IApiService`, que se

utiliza para realizar solicitudes HTTP y obtener datos externos.

- Cerramos VS.Code y creamos el proyecto de NUnit:

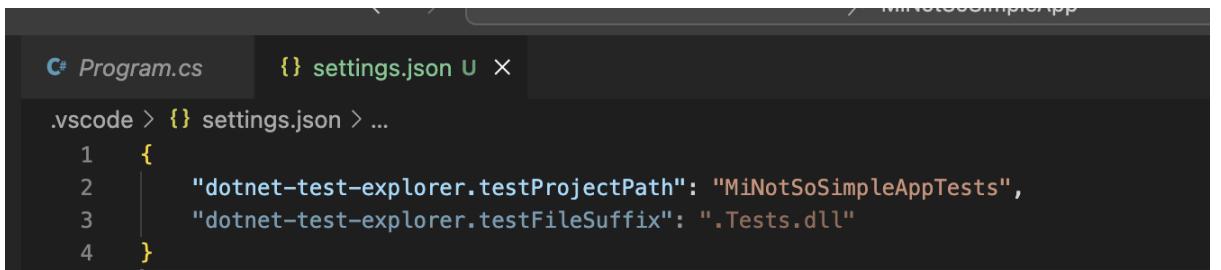
```
pedrofernandez@MacBook-Air-de-Pedro:~/pruebas-unitarias% dotnet new nunit -n MiNotSoSimpleAppTests
La plantilla "NUnit 3 Test Project" se creó correctamente.

Procesando acciones posteriores a la creación...
Restaurando /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj:
| Determinando los proyectos que se van a restaurar...
| Se ha restaurado /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj (en 102 ms).
Restauración realizada correctamente.

[redacted]

pedrofernandez@MacBook-Air-de-Pedro:~/pruebas-unitarias% cd MiNotSoSimpleAppTests
pedrofernandez@MacBook-Air-de-Pedro:~/pruebas-unitarias/MiNotSoSimpleAppTests% dotnet add reference ..\MiNotSoSimpleApp\MiNotSoSimpleApp.csproj
Se ha agregado la referencia "..\MiNotSoSimpleApp\MiNotSoSimpleApp.csproj" al proyecto.
pedrofernandez@MacBook-Air-de-Pedro:~/pruebas-unitarias/MiNotSoSimpleAppTests% dotnet add package NUnit
Determinando los proyectos que se van a restaurar...
Writing /var/folders/4c/516vr0d4qsgd87mwb6nj4dr/0000gn/T/tmp7MaI5.tmp
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/local/share/dotnet/sdk/7.0.400/trustedroots/codesignctl.pem'.
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/local/share/dotnet/sdk/7.0.400/trustedroots/timestampcctl.pem'.
info : Agregando PackageReference para el paquete "NUnit" al proyecto "/Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTest.sproj".
info : GET https://api.nuget.org/v3/registration5-gz-server?nunit/3.13.3/index.json
info : 000ms
info : Restaurando /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj...
info : El paquete "NUnit" es compatible con todos los marcos de trabajo especificados del proyecto "/Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj".
info : Se actualizó PackageReference para la versión "3.13.3" del paquete "NUnit" en el archivo "/Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj".
info : Generación de archivo MsBuild /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/obj/MiNotSoSimpleAppTests.csproj.nuget.g.targets.
info : Escribiendo el archivo de recursos en el disco. Ruta de acceso: /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/obj/project.assets.json
log : Se ha restaurado /Users/pedrofernandez/Desktop/Facu Pedro/Ing de Software III/ing-sw-3/09-pruebas-unitarias/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj (en 69 ms).
Determinando los proyectos que se van a restaurar...
Writing /var/folders/4c/516vr0d4qsgd87mwb6nj4dr/0000gn/T/tmpP0BzrV.tmp
```

- Abrimos VS.Code y configuramos settings:
- Seleccionamos el icono de Test y OpenSettings Se abre el archivo settings.json y escribimos el nombre del proyecto de pruebas y el nombre de la dll resultante de la compilación del proyecto de pruebas:
- Guardamos el archivo.



## 8.2 Escribimos Test:

8.3 Incluir en la entrega una explicación del código, centrado en explicar cómo se reemplaza el servicio real por el mock, que líneas de nuestro código de test se encargan de hacerlo y cómo funciona el mecanismo.

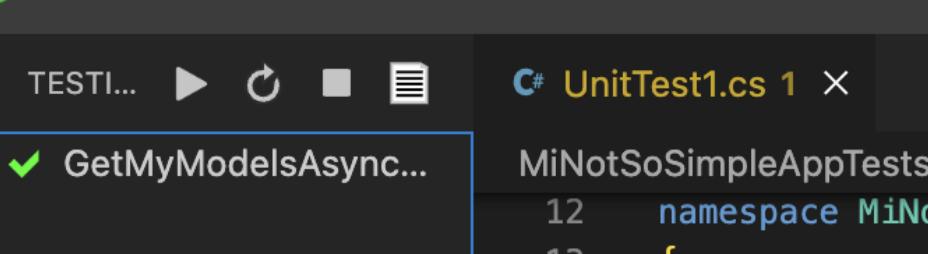
En este código de prueba unitaria, se utiliza Moq para simular el comportamiento de un servicio HTTP real al probar el método GetMyModelsAsync de la clase ApiService.

Primero, se crea un objeto mockHttpMessageHandler que simula las respuestas HTTP, configurándolo para que devuelva una respuesta simulada en lugar de hacer una solicitud real.

Luego, se configura el contenedor de servicios para que utilice este objeto simulado en lugar del servicio HTTP real al instanciar IApiService. Así, cuando se ejecuta el

caso de prueba, el método GetMyModelsAsync utiliza el servicio simulado, permitiendo verificar el comportamiento de ApiService sin depender de llamadas HTTP reales. Esto se utiliza para confirmar que la lógica de ApiService funciona correctamente con la respuesta simulada.

8.4 Ejecutamos Test en VSCode y en linea de consola.



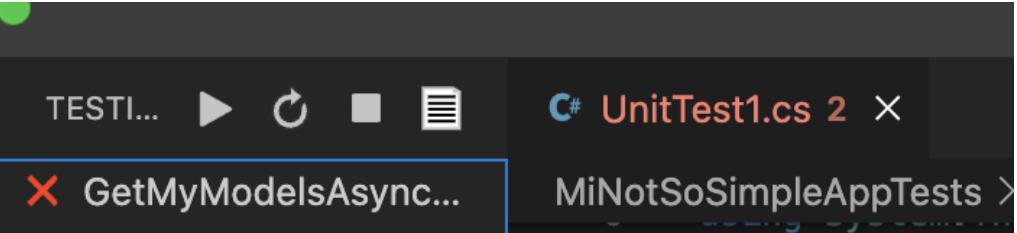
The screenshot shows the VS Code interface with the Test Explorer open. A single test, 'GetMyModelsAsync...', is listed and marked with a green checkmark, indicating it has passed. The status bar at the bottom displays the command line output of the test execution:

```
Microsoft (R) Test Execution Command Line Tool Version 17.0.0 Preview 25504-03-BC17B85053C1C4770dcd1880007f1003453f989 (arm64)
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed! - Failed: 0, Passed: 1, Skipped: 0, Total: 1, Duration: 68 ms - MiNotSoSimpleAppTests.dll (net7.0)
o pedrofernandez@MacBook-Air-de-Pedro: MiNotSoSimpleAppTests %
```

8.5 Hacerlo fallar, arreglarlo y volverlo a correr.



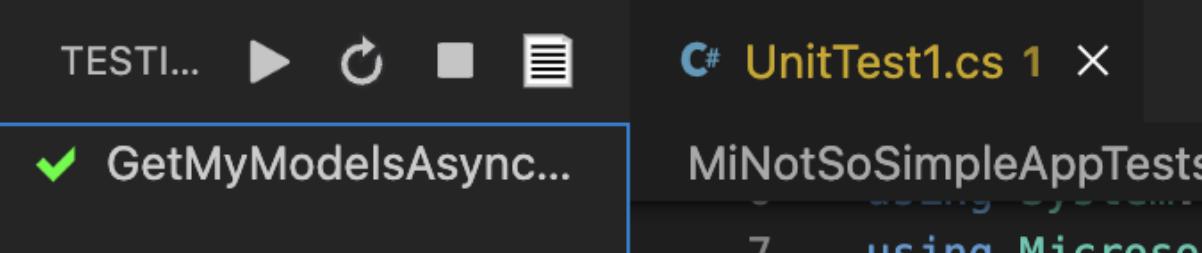
The screenshot shows the VS Code interface with the Test Explorer open. The same test, 'GetMyModelsAsync...', is now marked with a red X, indicating it has failed. The status bar at the bottom shows the command line output:

```
Microsoft (R) Test Execution Command Line Tool Version 17.0.0 Preview 25504-03-BC17B85053C1C4770dcd1880007f1003453f989 (arm64)
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Failed! - Failed: 1, Passed: 0, Skipped: 0, Total: 1, Duration: 68 ms - MiNotSoSimpleAppTests.dll (net7.0)
x pedrofernandez@MacBook-Air-de-Pedro: MiNotSoSimpleAppTests %
```

El caso de prueba espera que el título del primer elemento en result sea "Test Title", pero la respuesta simulada ahora contiene "Test Title Incorrecto". Cuando se ejecute el caso de prueba, fallará.



The screenshot shows the VS Code interface with the Test Explorer open. The test 'GetMyModelsAsync...' is now marked with a green checkmark again, indicating it has been successfully run and corrected. The status bar at the bottom shows the command line output:

```
Microsoft (R) Test Execution Command Line Tool Version 17.0.0 Preview 25504-03-BC17B85053C1C4770dcd1880007f1003453f989 (arm64)
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...
A total of 1 test files matched the specified pattern.

Passed! - Failed: 0, Passed: 1, Skipped: 0, Total: 1, Duration: 68 ms - MiNotSoSimpleAppTests.dll (net7.0)
✓ pedrofernandez@MacBook-Air-de-Pedro: MiNotSoSimpleAppTests %
```

corregido...

8.6 Modificar el mock para que devuelva una colección de Posts y en un nuevo test verificar que la cantidad devuelta por el mock coincide con la esperada en el nuevo test.



```
var mockResponse = new HttpResponseMessage(HttpStatusCode.OK)
{
    Content = new StringContent("[{"UserId": 1, "Id": 1, "Title": "Test Title 1", "Body": "Test Body 1"}, {"UserId": 2, "Id": 2, "Title": "Test Title 2", "Body": "Test Body 2"}]");
}
```

Se modifica el código del caso de prueba existente para que el mock devuelva una colección de Posts.

```

public async Task GetMyModelsAsync_ReturnsCorrectNumberOfItems()
{
    // Arrange
    var serviceCollection = new ServiceCollection();
    var mockResponse = new HttpResponseMessage(HttpStatusCode.OK)
    {
        Content = new StringContent("[{ \"UserId\": 1, \"Id\": 1, \"Title\": \"Test Title 1\", \"Body\": \"Test Body 1\" }, { \"UserId\": 2,
    };

    var mockHttpMessageHandler = new Mock<HttpMessageHandler>();
    mockHttpMessageHandler.Protected()
        .Setup<Task<HttpResponseMessage>>["SendAsync", ItExpr.IsAny<HttpRequestMessage>(), ItExpr.IsAny< CancellationToken>()]
        .ReturnsAsync(mockResponse);

    serviceCollection.AddTransient<IApiService>(_ => new ApiService(new HttpClient(mockHttpMessageHandler.Object)));
    var serviceProvider = serviceCollection.BuildServiceProvider();

    var apiService = serviceProvider.GetRequiredService<IApiService>();

    // Act
    var result = await apiService.GetMyModelsAsync();

    // Assert
    Assert.IsNotNull(result);
    Assert.AreEqual(2, result.Count()); // Verifica que se devuelvan 2 elementos
}

```

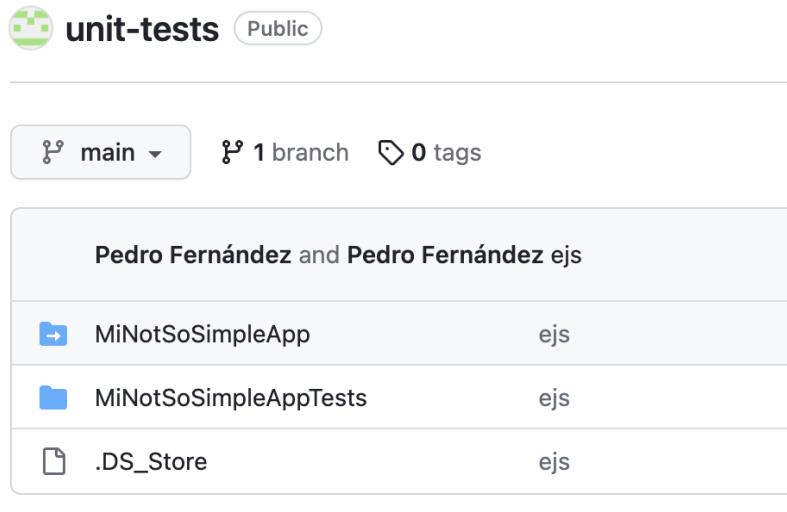
Se creó un nuevo caso de prueba llamado para verificar que la cantidad de elementos devueltos por el mock coincide con la cantidad esperada.

En este caso de prueba, se espera que result contenga 2 elementos, lo que verifica que el mock devuelva correctamente una colección de Posts.

En el nuevo caso de prueba, se utiliza `Assert.AreEqual(2, result.Count())` para verificar que la cantidad de elementos en result sea igual a 2, lo que confirma que el mock devuelve correctamente una colección con la cantidad esperada de elementos.

## 9- Capturar los unit tests como parte del proceso de CI/CD

- Subir el proyecto a GITHUB



- Configurar una GitHubAction que haga el build, corra el test y si funcionó, haga el deploy en una imagen que se suba a Docker.

**Workflow file for this run**  
.github/workflows/buildd.yml at 620e073

```

1 name: CI/CD
2
3 on:
4   push:
5     branches:
6       - main
7
8 jobs:
9   build:
10    runs-on: ubuntu-latest
11
12   steps:
13     - name: Checkout repository
14       uses: actions/checkout@v2
15
16     - name: Setup .NET Core
17       uses: actions/setup-dotnet@v1
18       with:
19         dotnet-version: 3.1.x
20
21     - name: Build and test
22       run: |
23         dotnet build
24         dotnet test --logger trx --results-directory ./test-results
25
26     - name: Docker login
27       run: echo ${{ secrets.DOCKER_PASSWORD }} | docker login -u ${{ secrets.DOCKER_USERNAME }} --password-stdin
28

```

**CI/CD**

**CI/CD**

Management

Caches

**1 workflow run**

● **Create buildd.yml**  
CI/CD #1: Commit 620e073 pushed by pedrofernandezmz

The screenshot shows the Docker Hub interface. At the top, there's a blue header bar with the Docker Hub logo, a search bar, and navigation links for Explore, Repositories, Organizations, and Help. Below the header, there are dropdown menus for 'piterfmz' and 'Search by repository name'. A large blue button labeled 'Create repository' is visible. In the main content area, a repository card for 'piterfmz / unit-tests' is shown. The card indicates it's a public repository with 0 stars, 0 forks, and was last pushed a few seconds ago. The repository name is also displayed above the card.

- Incluir link al repo y print de corridas correctas y falidas.  
link repo: <https://github.com/pedrofernandezmz/unit-tests.git>