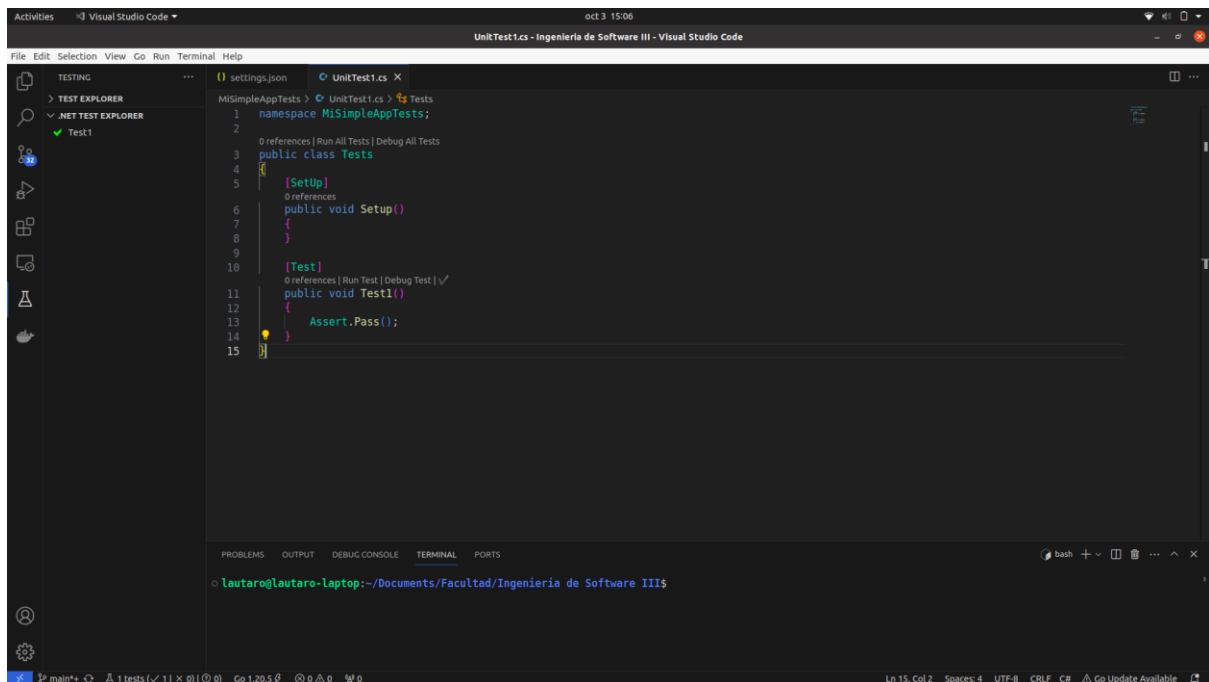


5.1)



Namespace: MiSimpleAppTests es el espacio de nombres en el que se encuentran las clases de prueba.

TestFixture Attribute: Indica que la clase Tests contiene pruebas y debe ser tratada como un conjunto de pruebas por NUnit.

SetUp Attribute: Este método se ejecuta antes de cada prueba. Actualmente está vacío en este código, lo que significa que no realiza ninguna acción específica antes de cada prueba.

Test Methods: Hay tres métodos de prueba en esta clase:

CanBeCancelledBy_AdminCancelling_ReturnsTrue: Esta prueba verifica si un administrador puede cancelar una reserva (CanCancelBy devuelve true si el usuario es administrador). Se crea un usuario administrador, se instancia una reserva y luego se verifica si la reserva puede ser cancelada por el usuario administrador. La prueba pasa si CanCancelBy devuelve true.

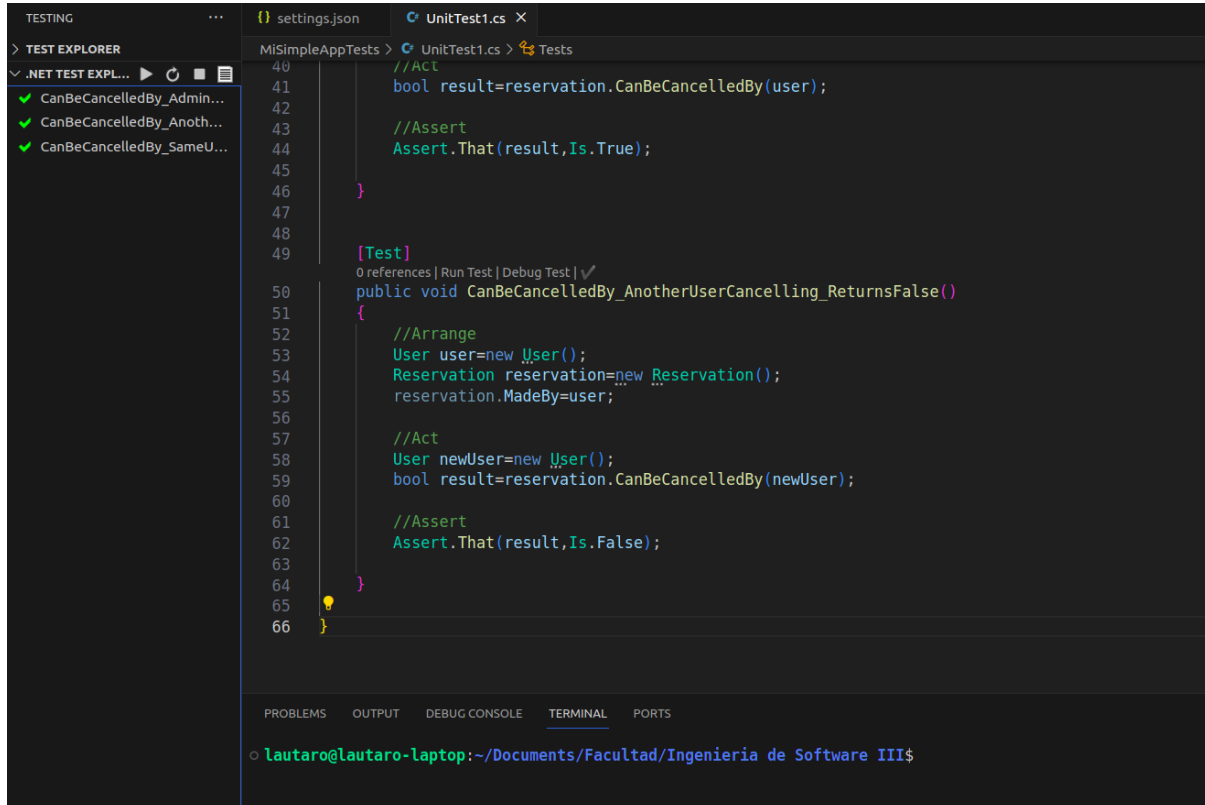
CanBeCancelledBy_SameUserCancelling_ReturnsTrue: Esta prueba verifica si un usuario que ha creado la reserva puede cancelarla (CanCancelBy devuelve true si el usuario es el mismo que creó la reserva). Se crea un usuario, se crea una reserva con ese usuario y luego se verifica si la reserva puede ser cancelada por el mismo usuario. La prueba pasa si CanCancelBy devuelve true.

CanBeCancelledBy_AnotherUserCancelling_ReturnsFalse: Esta prueba verifica si un usuario que no ha creado la reserva no puede cancelarla (CanCancelBy devuelve false si el usuario no es el mismo que creó la reserva). Se crea un usuario, se crea una reserva con

ese usuario y luego se crea otro usuario. Se verifica si este otro usuario puede cancelar la reserva. La prueba pasa si CanCancelBy devuelve false.

Assert: Las aserciones (Assert) se utilizan para verificar si los resultados obtenidos en las pruebas coinciden con los resultados esperados. Si las condiciones en las aserciones son verdaderas, las pruebas pasan; de lo contrario, fallan.

5.4)



```
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66

//Act
bool result=reservation.CanBeCancelledBy(user);

//Assert
Assert.That(result,Is.True);
}

[Test]
0 references | Run Test | Debug Test | ✓
public void CanBeCancelledBy_AnotherUserCancelling_ReturnsFalse()
{
    //Arrange
    User user=new User();
    Reservation reservation=new Reservation();
    reservation.MadeBy=user;

    //Act
    User newUser=new User();
    bool result=reservation.CanBeCancelledBy(newUser);

    //Assert
    Assert.That(result,Is.False);
}
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

o lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III\$

TESTING

... settings.json Clases.cs 1, M X UnitTest1.cs 1

TEST EXPLORER

.NET TEST EXPLORER

CanBeCancelledBy_Admin...
CanBeCancelledBy_Anoth...
CanBeCancelledBy_SameU...

MiSimpleApp > Clases.cs > Reservation > CanBeCancelledBy

1
2 8 references
3 public class Reservation
4 {
5 3 references
6 public User MadeBy { get; set; }
7
8 4 references
9 public bool CanBeCancelledBy(User user)
10 {
11 if (user.IsAdmin)
12 return false;
13 if (MadeBy==user)
14 return true;
15 return false;
16
17 //return (user.IsAdmin || MadeBy == user);
18 }
19 }
20
21 12 references
22 public class User
23 {
24 3 references
25 public bool IsAdmin { get; set; }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS

o lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III\$

• lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/MiSimpleAppTests\$ dotnet test
Determining projects to restore...
All projects are up-to-date for restore.
MiSimpleApp -> /home/lautaro/Documents/Facultad/Ingenieria de Software III/MiSimpleApp/bin/Debug/net7.0/MiSimpleApp.dll
MiSimpleAppTests -> /home/lautaro/Documents/Facultad/Ingenieria de Software III/MiSimpleAppTests/bin/Debug/net7.0/MiSimpleAppTests.dll
Test run for /home/lautaro/Documents/Facultad/Ingenieria de Software III/MiSimpleAppTests/bin/Debug/net7.0/MiSimpleAppTests.dll (.NETCoreApp,Version=v7.0)
Microsoft (R) Test Execution Command Line Tool Version 17.7.1 (x64)
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: 3, Skipped: 0, Total: 3, Duration: 8 ms - MiSimpleAppTests.dll (net7.0)

```
using Microsoft.Extensions.Logging;  
using NUnit.Framework;  
using SimpleWebAPI.Controllers;  
using System;  
using System.Linq;
```

```
namespace SimpleWebAPI.Tests  
{  
    [TestFixture]
```

```

public class WeatherForecastControllerTests
{
    [Test]
    public void Get_ReturnsWeatherForecasts()
    {
        // Arrange
        ILogger<WeatherForecastController> logger = new
        LoggerFactory().CreateLogger<WeatherForecastController>();
        var controller = new WeatherForecastController(logger);

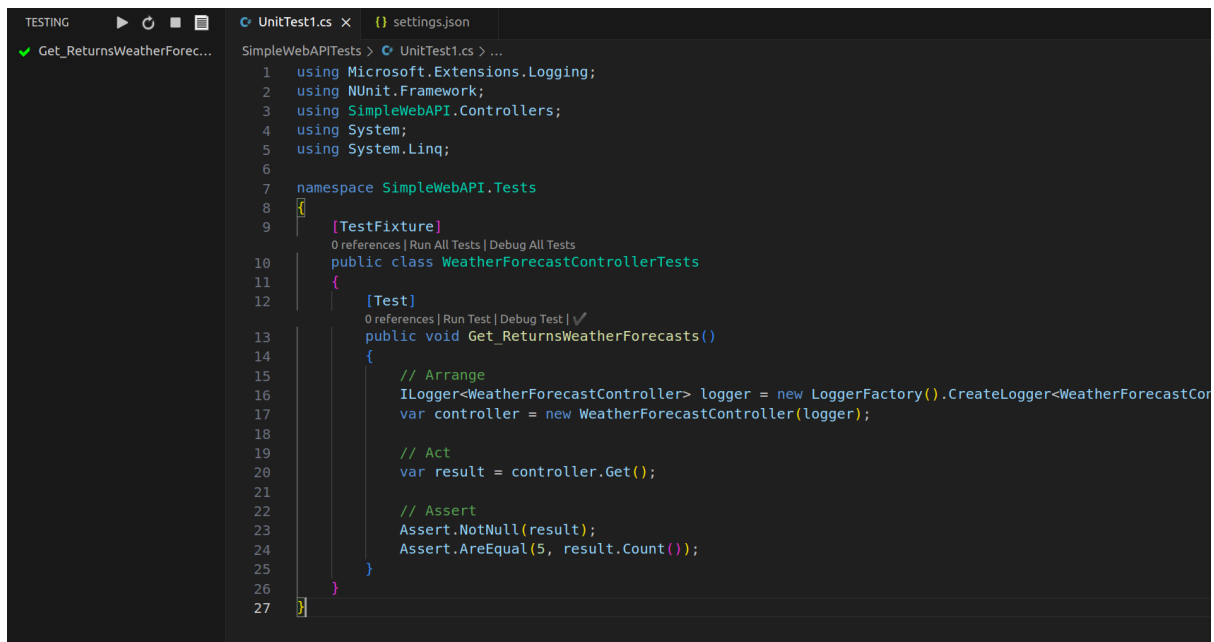
        // Act
        var result = controller.Get();

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

```

- Primero se importan los namespaces necesarios para la ejecución del programa. "using Microsoft.Extensions.Logging;"
- Se define una clase llamada weatherForecastController, la cual se la etiqueta con [TestFixture] lo cual indica que contiene pruebas unitarias.
- Define un método de prueba llamado Get_ReturnsWeatherForecasts. Esta función realiza la prueba principal.
- Arrange
 "ILogger<WeatherForecastController> logger = new
 LoggerFactory().CreateLogger<WeatherForecastController>();
 var controller = new WeatherForecastController(logger);"
 Esta parte del código se configuran las condiciones previas para la prueba (Arrange). Se crea una instancia de WeatherForecastController pasando un objeto ILogger como argumento al constructor.
- Luego, se realiza la acción que se va a probar (Act). En este caso, se llama al método Get() del controlador.
- Finalmente, se verifica el resultado de la acción (Assert). Se asegura de que el resultado no sea nulo y que la cantidad de elementos en el resultado sea igual a 5. Estas afirmaciones son las que determinan si la prueba pasa o falla.

6.4)



The screenshot shows the Visual Studio IDE with a unit test file named `UnitTest1.cs` open. The test is for the `WeatherForecastController` class. The code is as follows:

```
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     public class WeatherForecastControllerTests
11     {
12         [Test]
13         public void Get_ReturnsWeatherForecasts()
14         {
15             // Arrange
16             ILogger<WeatherForecastController> logger = new LoggerFactory().CreateLogger<WeatherForecastController>();
17             var controller = new WeatherForecastController(logger);
18
19             // Act
20             var result = controller.Get();
21
22             // Assert
23             Assert.NotNull(result);
24             Assert.AreEqual(5, result.Count());
25         }
26     }
27 }
```

8.1)



The screenshot shows the Visual Studio IDE with a `Program` class open. The code is as follows:

```
class Program
{
    0 references
    static async Task Main()
    {
        var serviceProvider = new ServiceCollection()
            .AddHttpClient()
            .AddTransient<IApiService, ApiService>()
            .BuildServiceProvider();

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

Se define la clase “Program” que contiene el método “Main()”, que es el punto de entrada de la aplicación.

Se crea un contenedor de servicios utilizando “ServiceCollection”.

Se agrega un cliente HTTP utilizando el método “AddHttpClient()”. Esto permite la creación de instancias de “HttpClient” con configuraciones predeterminadas.

Se agrega la implementación de la interfaz “IApiService” (“ApiService”) como un servicio transitorio. Esto significa que se crea una nueva instancia de “ApiService” cada vez que se solicita.

```
public interface IApiService
{
    2 references
    Task<IEnumerable<Post>> GetMyModelsAsync();
}
```

La interfaz IApiService define un contrato que cualquier clase puede implementar. En este caso, la interfaz tiene un método llamado GetMyModelsAsync(), que es un método asíncrono que devuelve una tarea (Task) que se espera que produzca una colección de objetos Post o una clase que implementa la interfaz IEnumerable<Post>.

La interfaz IApiService proporciona una abstracción para cualquier servicio que quiera obtener una colección de objetos Post de alguna fuente, sin especificar cómo se obtienen estos datos. Esto permite la flexibilidad en la implementación y facilita el intercambio de diferentes servicios de API sin cambiar el código que los utiliza, siempre y cuando implementen la interfaz.

La clase Post tiene la siguiente forma:

```
public class Post
{
    0 references
    public int UserId { get; set; }
    1 reference
    public int Id { get; set; }
    1 reference
    public string Title { get; set; }
    0 references
    public string Body { get; set; }
}
```

- Se construye el proveedor de servicios utilizando `BuildServiceProvider()`.

```

var myModels = await apiService.GetMyModelsAsync();
foreach (var model in myModels)
{
    Console.WriteLine($"Id: {model.Id}, Title: {model.Title}");
}
}

```

Se obtiene una instancia del servicio `IApiService` del contenedor de servicios.
Se llama al método `GetMyModelsAsync()` del servicio para obtener una colección de objetos `Post`.
Se itera sobre la colección de objetos `Post` y se imprime el `Id` y el `Title` de cada objeto en la consola.

Este programa crea un servicio HTTP, obtiene datos de una API utilizando el servicio `IApiService` y muestra los identificadores (`Id`) y títulos (`Title`) de los objetos `Post` en la consola.

El código está configurado para hacer una solicitud GET a la URL ["https://jsonplaceholder.typicode.com/posts/"](https://jsonplaceholder.typicode.com/posts/).

```

var response = await _httpClient.GetAsync("https://jsonplaceholder.typicode.com/posts/");
response.EnsureSuccessStatusCode();

```

Si accedemos al link podemos ver que nos devuelve objetos JSON de esa forma:

```

0:
  userId: 1
  id: 1
  title: "sunt aut facere repellat provident occaecati excepturi optio reprehenderit"
  body: "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"

```

Como vemos la clase `Post` tiene los atributos necesarios, por lo tanto la interfaz a mockear es `IApiService`

```

public interface IApiService
{
    Task<IEnumerable<Post>> GetMyModelsAsync();
}

```

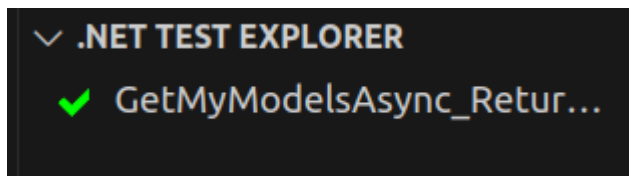
Creamos el proyecto Nunit:

```

lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests$ dotnet add package NUnit3TestAdapter
Determining projects to restore...
Writing /tmp/impedit1.tmp
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/share/dotnet/sdk/7.0.402/trustedroots/codesignctl.pem'.
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/share/dotnet/sdk/7.0.402/trustedroots/timestampctl.pem'.
info : Adding PackageReference for package 'NUnit3TestAdapter' into project '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : GET https://api.nuget.org/v3/registrations-gz-sonserver/nunit3testadapter/index.json
info : OK https://api.nuget.org/v3/registrations-gz-sonserver/nunit3testadapter/index.json 926ms
info : Restoring packages for /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj...
info : GET https://api.nuget.org/v3-flatcontainer/nunit3testadapter/index.json
info : OK https://api.nuget.org/v3-flatcontainer/nunit3testadapter/index.json 908ms
info : GET https://api.nuget.org/v3-flatcontainer/nunit3testadapter/4.5.0/nunit3testadapter.4.5.0.nupkg
info : OK https://api.nuget.org/v3-flatcontainer/nunit3testadapter/4.5.0/nunit3testadapter.4.5.0.nupkg 786ms
info : Installed NUnit3TestAdapter 4.5.0 from https://api.nuget.org/v3/index.json with content hash s83pqlE9B12F49PF3dFRFvSuFQYrATj6Bc3XXJ15/MRGvKLnrgRlqTjdShX+AdFUCCU/4Xex58aduFs6A==.
info : Package 'NUnit3TestAdapter' is compatible with all the specified frameworks in project '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : PackageReference for package 'NUnit3TestAdapter' version '4.5.0' updated in file '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : Generating msbuild file /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/obj/MNotSoSimpleAppTests.csproj.nuget.g.props.
info : Writing assets file to disk. Path: /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/obj/project.assets.json
log : Restored /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj (in 1.83 sec).
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests$ dotnet add package Microsoft.Extensions.Http
Determining projects to restore...
Writing /tmp/imp5Hw7t.tmp
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/share/dotnet/sdk/7.0.402/trustedroots/codesignctl.pem'.
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/share/dotnet/sdk/7.0.402/trustedroots/timestampctl.pem'.
info : Adding PackageReference for package 'Microsoft.Extensions.Http' into project '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : GET https://api.nuget.org/v3/registrations-gz-sonserver/microsoft.extensions.http/index.json
info : OK https://api.nuget.org/v3/registrations-gz-sonserver/microsoft.extensions.http/index.json 929ms
info : Restoring packages for /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj...
info : Package 'Microsoft.Extensions.Http' is compatible with all the specified frameworks in project '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : PackageReference for package 'Microsoft.Extensions.Http' version '7.0.0' added to file '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : Writing assets file to disk. Path: /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/obj/project.assets.json
log : Restored /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj (in 85 ms).
lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests$ dotnet add package Microsoft.Extensions.DependencyInjection
Determining projects to restore...
Writing /tmp/imp9fCt1r.tmp
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/share/dotnet/sdk/7.0.402/trustedroots/codesignctl.pem'.
info : X.509 certificate chain validation will use the fallback certificate bundle at '/usr/share/dotnet/sdk/7.0.402/trustedroots/timestampctl.pem'.
info : Adding PackageReference for package 'Microsoft.Extensions.DependencyInjection' into project '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : GET https://api.nuget.org/v3/registrations-gz-sonserver/microsoft.extensions.dependencyinjection/index.json
info : OK https://api.nuget.org/v3/registrations-gz-sonserver/microsoft.extensions.dependencyinjection/index.json 909ms
info : Restoring packages for /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj...
info : Package 'Microsoft.Extensions.DependencyInjection' is compatible with all the specified frameworks in project '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : PackageReference for package 'Microsoft.Extensions.DependencyInjection' version '7.0.0' added to file '/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj'.
info : Writing assets file to disk. Path: /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/obj/project.assets.json
log : Restored /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MNotSoSimpleAppTests/MNotSoSimpleAppTests.csproj (in 92 ms).

```

8.2)



8.3)

```

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

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

```

Esta parte del código corresponde a la configuración del Mock del servicio HTTP. Se crea un objeto HttpResponseMessage simulando una respuesta HTTP exitosa con un cuerpo JSON que representa un objeto Post. Se crea un objeto mock de HttpMessageHandler utilizando Moq. Se configura para que simule el método SendAsync de HttpMessageHandler, que es el método que se usa

internamente para enviar solicitudes HTTP. Cuando se llama a este método en la prueba, devolverá el HttpResponseMessage preconfigurado.

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

var serviceProvider =
serviceCollection.BuildServiceProvider();

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

Esta parte corresponde a la configuración del contenedor de dependencias. Se configura el contenedor de dependencias (serviceCollection) para registrar una instancia de ApiService que utiliza un HttpClient con el HttpMessageHandler mock como dependencia. Esto asegura que, cuando ApiService haga una solicitud HTTP, utilizará el mock en lugar de realizar una solicitud real.

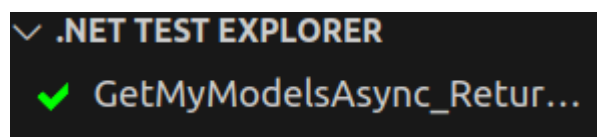
```
var result = await apiService.GetMyModelsAsync();
```

Esta es la ejecución de la prueba.

```
Assert.IsNotNull(result);
Assert.AreEqual(1, result.Count());
Assert.AreEqual("Test Title", result.FirstOrDefault().Title);
```

y por último compara los resultados.

8.4)



```
• lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests$ dotnet test
Determining projects to restore...
All projects are up-to-date for restore.
MiNotSoSimpleApp -> /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleApp/bin/Debug/net7.0/MiNotSoSimpleApp.dll
MiNotSoSimpleAppTests -> /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/bin/Debug/net7.0/MiNotSoSimpleAppTests.dll
Test run for /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/bin/Debug/net7.0/MiNotSoSimpleAppTests.dll (.NETCoreApp,Version=v7.0)
Microsoft (R) Test Execution Command Line Tool Version 17.7.1 (x64)
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: 92 ms - MiNotSoSimpleAppTests.dll (net7.0)
```

8.5)

Lo hacemos fallar:

```
Assert.IsNotNull(result);
Assert.AreEqual(2, result.Count());
Assert.AreEqual("Test Title", result.FirstOrDefault().Title);
```

```
Starting test execution, please wait...
A total of 1 test files matched the specified pattern.
Failed GetMyModelsAsync_ReturnsDataFromHttpClient [131 ms]
Error Message:
    Expected: 2
    But was: 1

Stack Trace:
    at MiNotSoSimpleAppTests.ApiServiceTests.GetMyModelsAsync_ReturnsDataFromHttpClient() in /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/UnitTest1.cs:line 41
    at NUnit.Framework.Internal.TaskAwaitAdapter.GenericAdapter`1.GetResult()
    at NUnit.Framework.Internal.AsyncToSyncAdapter.Await(Func`1 invoke)
    at NUnit.Framework.Internal.Commands.TestMethodCommand.RunTestMethod(TestExecutionContext context)
    at NUnit.Framework.Internal.Commands.Execute(TestExecutionContext context)
    at NUnit.Framework.Internal.Execution.SimpleWorkItem.<>c__DisplayClass4_0.<PerformWork>b__0()
    at NUnit.Framework.Internal.ContextUtils.<>c__DisplayClass1_0`1.<DoIsolated>b__0(Object _)

1) at MiNotSoSimpleAppTests.ApiServiceTests.GetMyModelsAsync_ReturnsDataFromHttpClient() in /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/UnitTest1.cs:line 41
   at System.Runtime.CompilerServices.AsyncMethodBuilderCore.Start[TStateMachine](TStateMachine& stateMachine)
   at MiNotSoSimpleAppTests.ApiServiceTests.GetMyModelsAsync_ReturnsDataFromHttpClient()

Failed! - Failed: 1, Passed: 0, Skipped: 0, Total: 1, Duration: 131 ms - MiNotSoSimpleAppTests.dll (net7.0)
```

8.6)

modificamos el código para que devuelva una colección de Post:

```
{
    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\" }, " +
        "{ \"UserId\": 3, \"Id\": 3, \"Title\": \"Test Title 3\", \"Body\": \"Test Body 3\" }]")
};
```

y también modificamos los test para que pase, en este caso devuelve 3 objetos y el título del primero es “Test Title 1”.

```
Assert.IsNotNull(result);
Assert.AreEqual(3, result.Count());
Assert.AreEqual("Test Title 1", result.FirstOrDefault().Title);
```

Ejecutamos:

```
• lautaro@lautaro-laptop:~/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests$ dotnet test
  Determining projects to restore...
  All projects are up-to-date for restore.
  MiNotSoSimpleApp -> /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleApp/bin/Debug/net7.0/MiNotSoSimpleApp.dll
/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/UnitTest1.cs(44,45): warning CS8602: Dereference of a possibly null reference. [/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj]
/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/UnitTest1.cs(43,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md) [/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj]
/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/UnitTest1.cs(44,13): warning NUnit2005: Consider using the constraint model, Assert.That(actual, Is.EqualTo(expected)), instead of the classic model, Assert.AreEqual(expected, actual) (https://github.com/nunit/nunit.analyzers/tree/master/documentation/NUnit2005.md) [/home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/MiNotSoSimpleAppTests.csproj]
  MiNotSoSimpleAppTests -> /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/bin/Debug/net7.0/MiNotSoSimpleAppTests.dll
Test run for /home/lautaro/Documents/Facultad/Ingenieria de Software III/TP9/MiNotSoSimpleAppTests/bin/Debug/net7.0/MiNotSoSimpleAppTests.dll (.NETCoreApp,Version=v7.0)
Microsoft (R) Test Execution Command Line Tool Version 17.7.1 (x64)
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: 94 ms - MiNotSoSimpleAppTests.dll (net7.0)
```