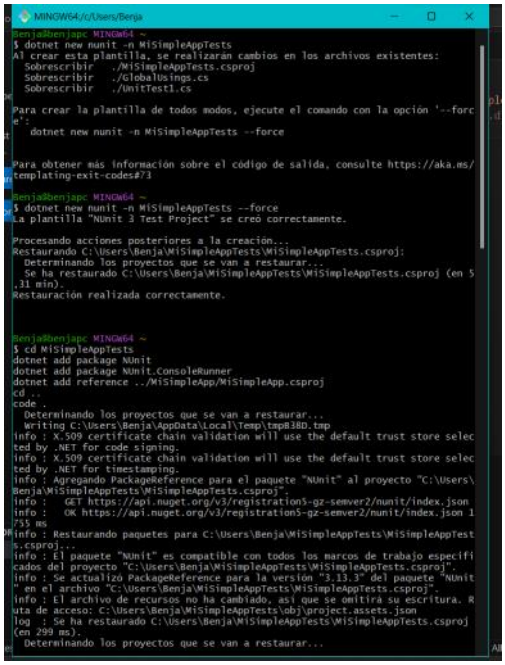
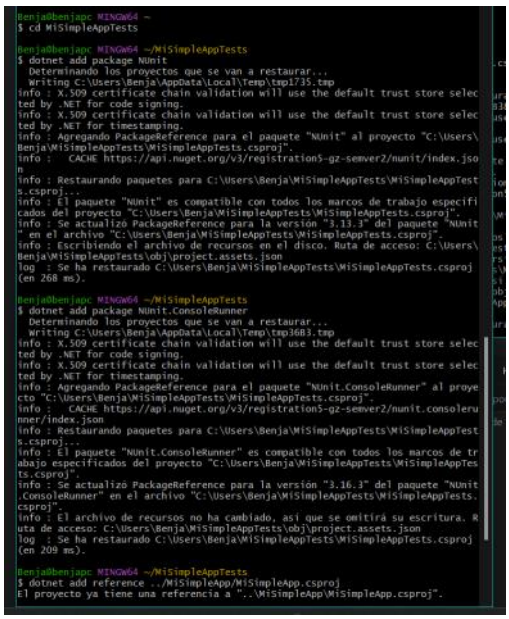
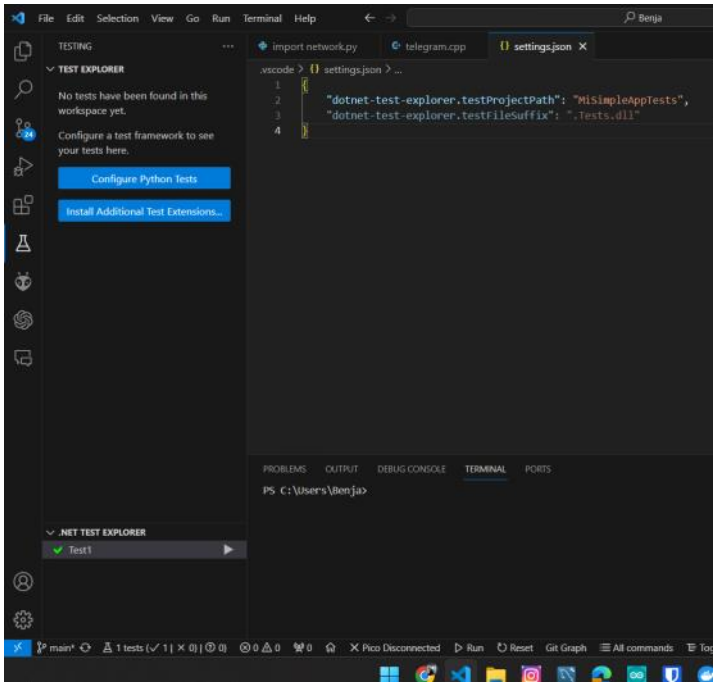
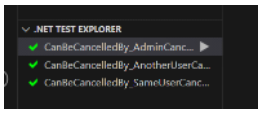
PRACTICO 9

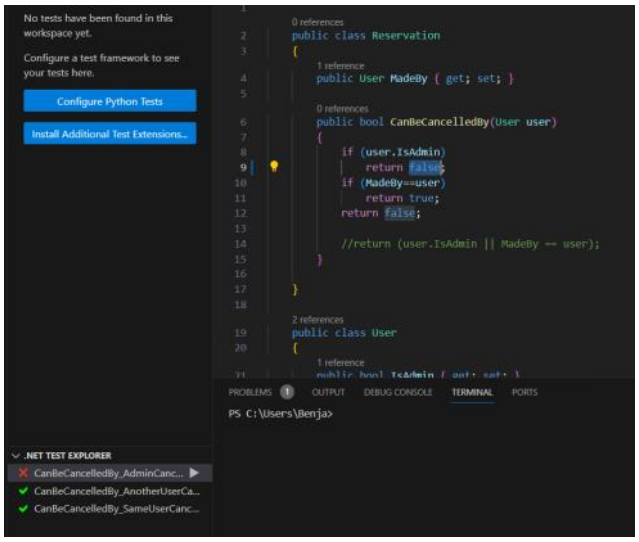
martes, 3 de octubre de 2023 15:19

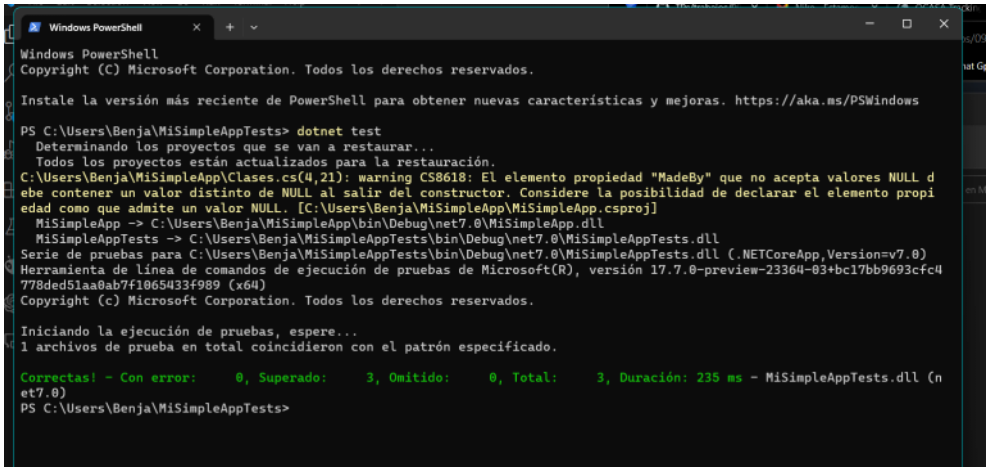


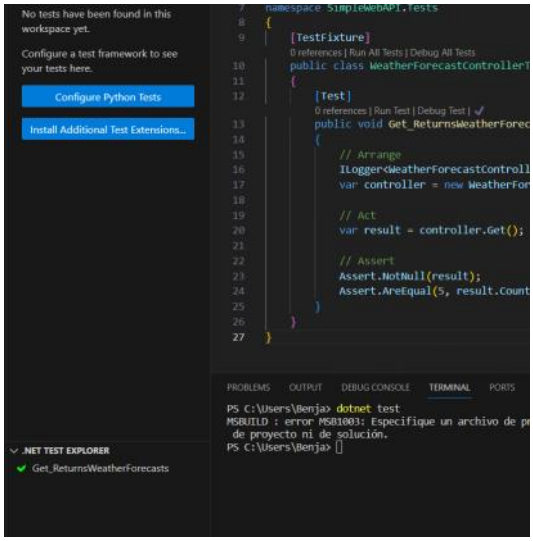


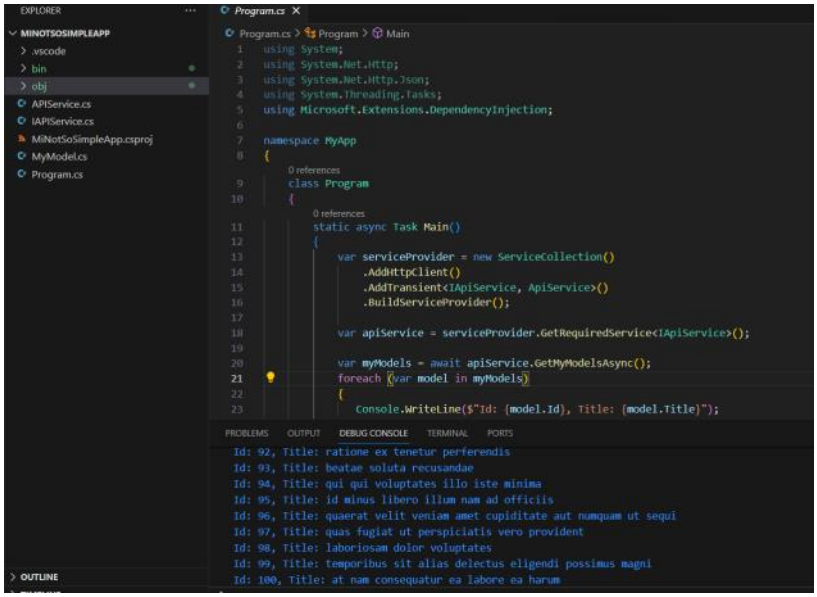

File Edit Selection View Go Run Terminal Help
TESTING
TEST EXPLORER
No tests have been found in this workspace yet.
Configure a test framework to see your tests here.
Configure Python Tests
Install Additional Test Extensions...
import network.py
telegram.cpp
settings.json
.vscode > {} settings.json > ...
{
"dotnet-test-explorer.testProjectPath": "MiSimpleAppTests",
"dotnet-test-explorer.testFileSuffix": ".Tests.dll"
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Benja>
.NET TEST EXPLORER
Test1

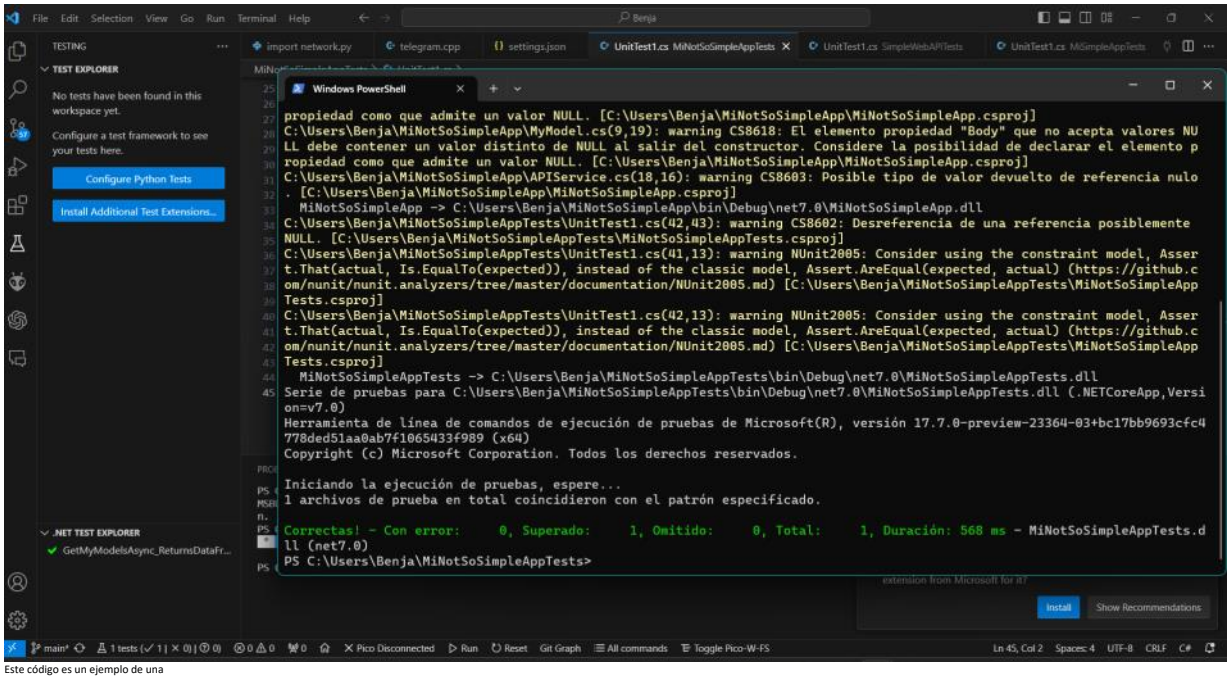

.NET TEST EXPLORER
CanBeCancelledBy_AdminCanc...
CanBeCancelledBy_AnotherUserCa...
CanBeCancelledBy_SameUserCanc...


No tests have been found in this workspace yet.
Configure a test framework to see your tests here.
Configure Python Tests
Install Additional Test Extensions...
0 references
public class Reservation
{
1 reference
public User MadeBy { get; set; }
0 references
public bool CanBeCancelledBy(User user)
{
if (user.IsAdmin)
return false;
if (MadeBy==user)
return true;
return false;
//return (user.IsAdmin || MadeBy == user);
}
}
2 references
public class User
{
1 reference
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Benja>
.NET TEST EXPLORER
CanBeCancelledBy_AdminCanc...
CanBeCancelledBy_AnotherUserCa...
CanBeCancelledBy_SameUserCanc...


Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows
PS C:\Users\Benja\MiSimpleAppTests> dotnet test
Determinando los proyectos que se van a restaurar...
Todos los proyectos están actualizados para la restauración.
C:\Users\Benja\MiSimpleApp\Clases.cs(4,21): warning CS8618: El elemento propiedad "MadeBy" que no acepta valores NULL debe contener un valor distinto de NULL al salir del constructor. Considere la posibilidad de declarar el elemento propiedad como que admite un valor NULL. [C:\Users\Benja\MiSimpleApp\MiSimpleApp.csproj]
MiSimpleApp -> C:\Users\Benja\MiSimpleApp\bin\Debug\net7.0\MiSimpleApp.dll
MiSimpleAppTests -> C:\Users\Benja\MiSimpleAppTests\bin\Debug\net7.0\MiSimpleAppTests.dll
Serie de pruebas para C:\Users\Benja\MiSimpleAppTests\bin\Debug\net7.0\MiSimpleAppTests.dll (.NETCoreApp,Version=v7.0)
Herramienta de línea de comandos de ejecución de pruebas de Microsoft(R), versión 17.7.0-preview-23364-03+bc17bb9693cfc4778ded51aa0ab7f1065433f989 (x64)
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: 235 ms - MiSimpleAppTests.dll (net7.0)
PS C:\Users\Benja\MiSimpleAppTests>







Este código es un ejemplo de una prueba unitaria utilizando NUnit y Moq en el contexto de un servicio de API. Aquí hay una explicación línea por línea:

1. **`ServiceCollection` y `ServiceProvider`:**

```
var serviceCollection = new ServiceCollection();
var serviceProvider = serviceCollection.BuildServiceProvider();
```

Estas líneas crean una colección de servicios (`ServiceCollection`) y luego construyen un proveedor de servicios (`ServiceProvider`) a partir de esa

colección. Los servicios son utilizados para configurar y construir objetos necesarios para la prueba.

2. **Configuración del Mock del Cliente HTTP:**

```
  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);
```

Aquí, se crea un objeto `HttpResponseMessage` simulado que contiene datos de respuesta simulados. Luego, se configura un mock de `HttpMessageHandler` utilizando Moq. Este mock se encarga de interceptar las llamadas a `SendAsync` del cliente HTTP y devolver la respuesta simulada en lugar de realizar una solicitud real.

3. **Configuración del Servicio y la Inyección de Dependencias:**

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

Se configura el servicio `IApiService` para que utilice una instancia de `HttpClient` que utiliza el `HttpMessageHandler` mock. Esto asegura que, cuando se realice una solicitud HTTP a través del servicio, se utilizará el mock en lugar de hacer una solicitud real.

4. **Creación del Objeto a Probar:**

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

Se obtiene una instancia del servicio que se va a probar, que en este caso es `IApiService`.

5. **Act:**

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

Se llama al método que se está probando, en este caso, `GetMyModelsAsync` del servicio.

6. **Assert:**

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

Se realizan afirmaciones para verificar que el resultado de la llamada al servicio es el esperado. En este caso, se verifica que el resultado no sea nulo, que haya un elemento en la lista y que el título del primer elemento sea "Test Title".

En resumen, el código utiliza Moq para simular las respuestas del cliente HTTP, permitiendo que las pruebas se centren en el comportamiento del servicio sin depender de la comunicación real a través de la red.