Dice que lo primero que he de hacer es ejecutar .... 2

El segundo paso es to add the test project to your solution .... 5

En tercer lugar, add a reference from the test project -> API project .... 7

To verify that everything is set up correctly, corremos dotnet build from the solution root ('backend' folder) .... 10

At this point, your test project has everything it needs to start unit testing the CRUD endpoints of ReservasController. .... 19

## Tests unitarios de my ASP.NET Core Web API backend con xUnit

Según chatgpt, no necesito instalarme nada nuevo para hacer estos tests.

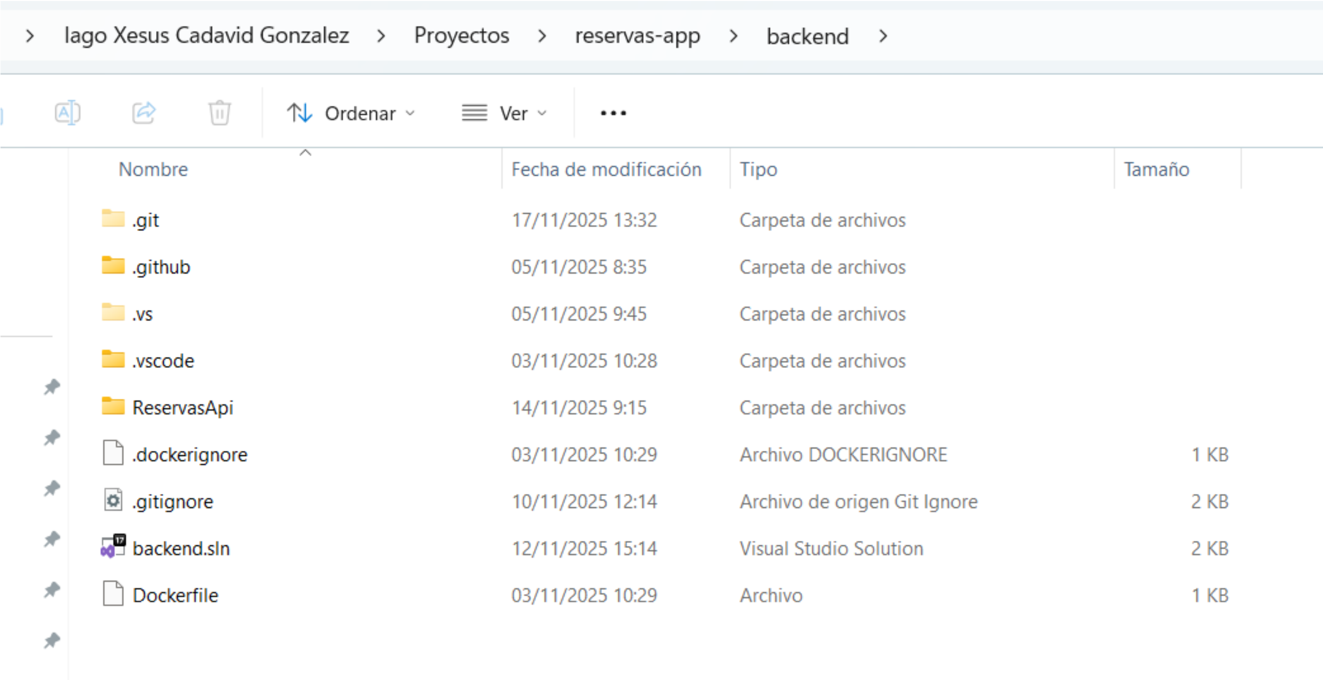
Dice que lo primero que he de hacer es ejecutar:

```
dotnet new xunit –n MyBookingApi.Tests
```

That is “the **first step** to start adding unit tests to your project”.

Le pregunté “is the test project created based on my existing (backend) project or not?” y chatgpt me respondió: No, the xUnit test project is NOT created from your existing project. It (the xUnit test project) is a separate, empty project that you connect to your main project afterward.

“Running **the command** in the wrong folder creates the test project in the wrong place”. You should run it from the root folder of your solution, the same directory where your .sln file is located.

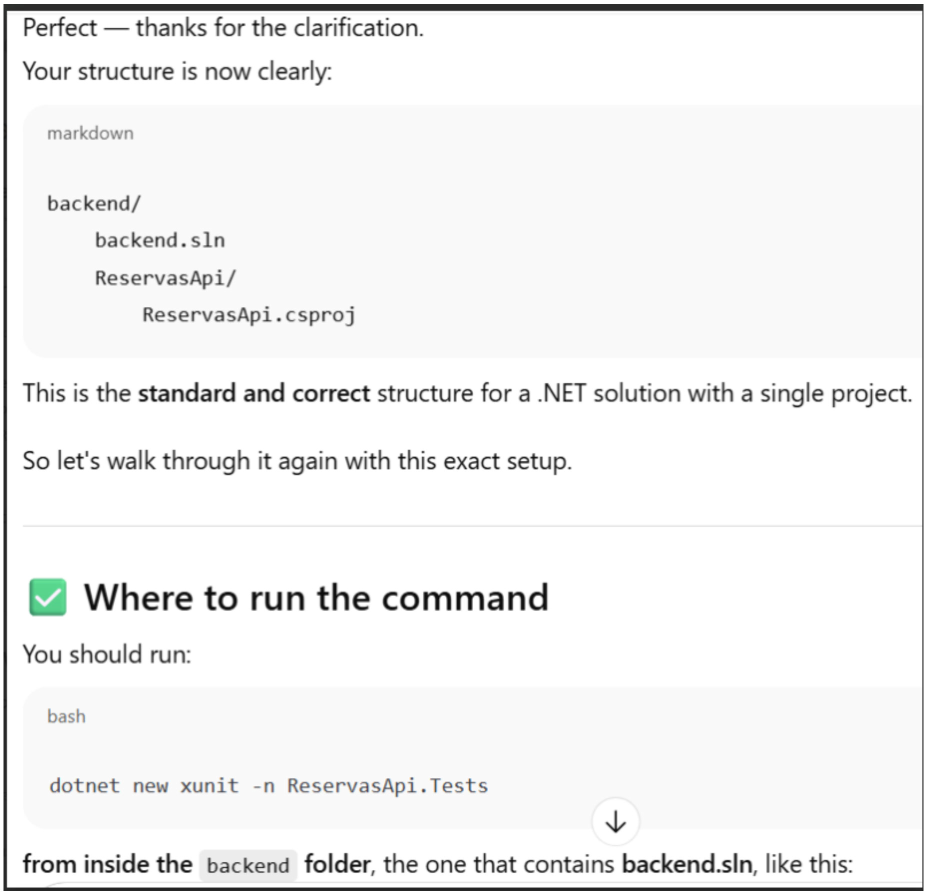


Iago Xesus Cadavid Gonzalez > Proyectos > reservas-app > backend >
Ordenar
Ver

| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|---|---|---|
| .git | 17/11/2025 13:32 | Carpeta de archivos | |
| .github | 05/11/2025 8:35 | Carpeta de archivos | |
| .vs | 05/11/2025 9:45 | Carpeta de archivos | |
| .vscode | 03/11/2025 10:28 | Carpeta de archivos | |
| ReservasApi | 14/11/2025 9:15 | Carpeta de archivos | |
| .dockerignore | 03/11/2025 10:29 | Archivo DOCKERIGNORE | 1 KB |
| .gitignore | 10/11/2025 12:14 | Archivo de origen Git Ignore | 2 KB |
| backend.sln | 12/11/2025 15:14 | Visual Studio Solution | 2 KB |
| Dockerfile | 03/11/2025 10:29 | Archivo | 1 KB |

“That (backend folder es la raíz de mi backend y es padre de backend.sln) means your *solution root* is the backend folder, the one that contains backend.sln”

PÁGINA SIGUIENTE



Perfect — thanks for the clarification.

Your structure is now clearly:

```markdown
backend/
    backend.sln
    ReservasApi/
        ReservasApi.csproj
```

This is the **standard and correct** structure for a .NET solution with a single project.

So let's walk through it again with this exact setup.

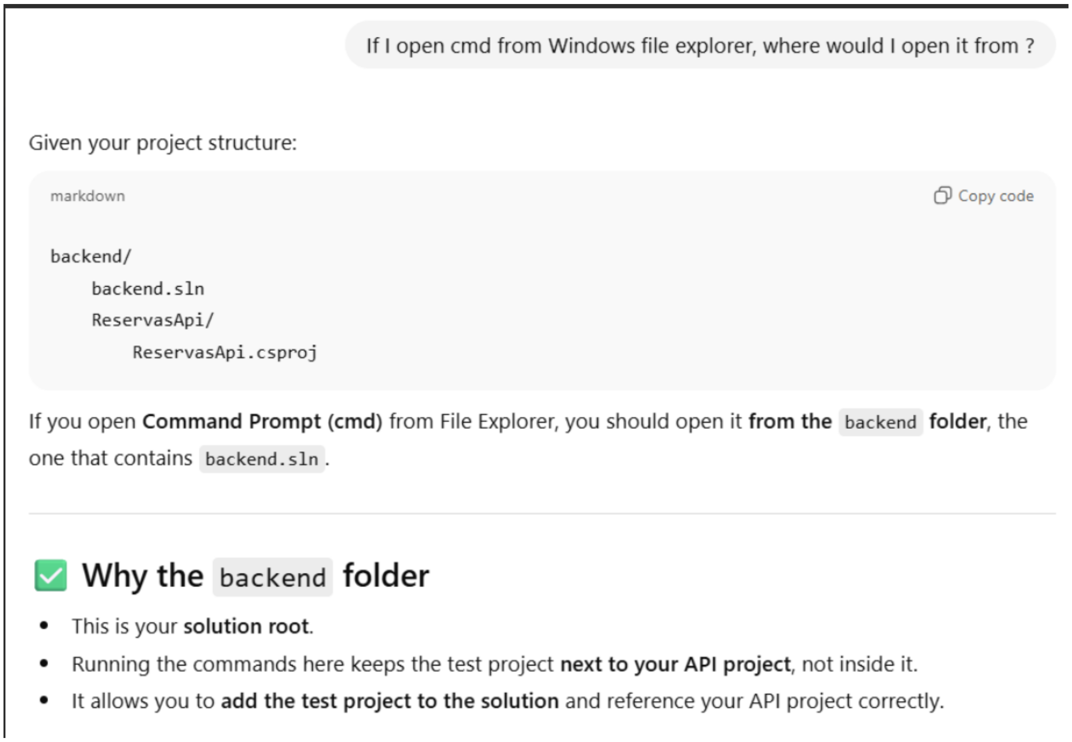
---

## ✅ Where to run the command

You should run:

```bash
dotnet new xunit -n ReservasApi.Tests
```

**from inside the `backend` folder**, the one that contains **backend.sln**, like this:



If I open cmd from Windows file explorer, where would I open it from ?

Given your project structure:

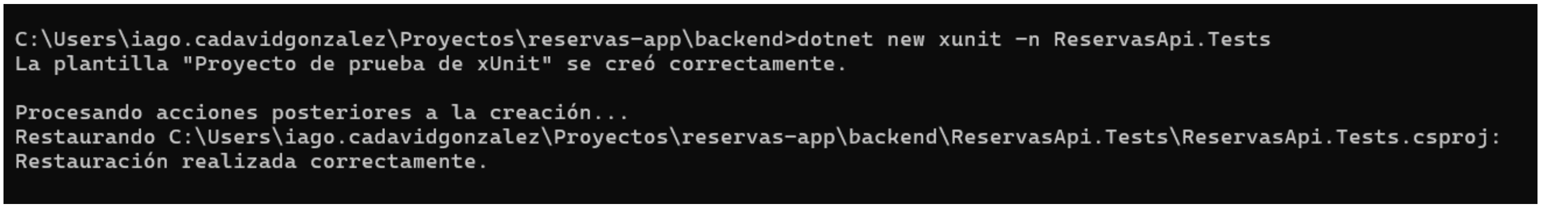
```markdown
backend/
    backend.sln
    ReservasApi/
        ReservasApi.csproj
```

If you open **Command Prompt (cmd)** from File Explorer, you should open it **from the `backend` folder**, the one that contains `backend.sln`.

---

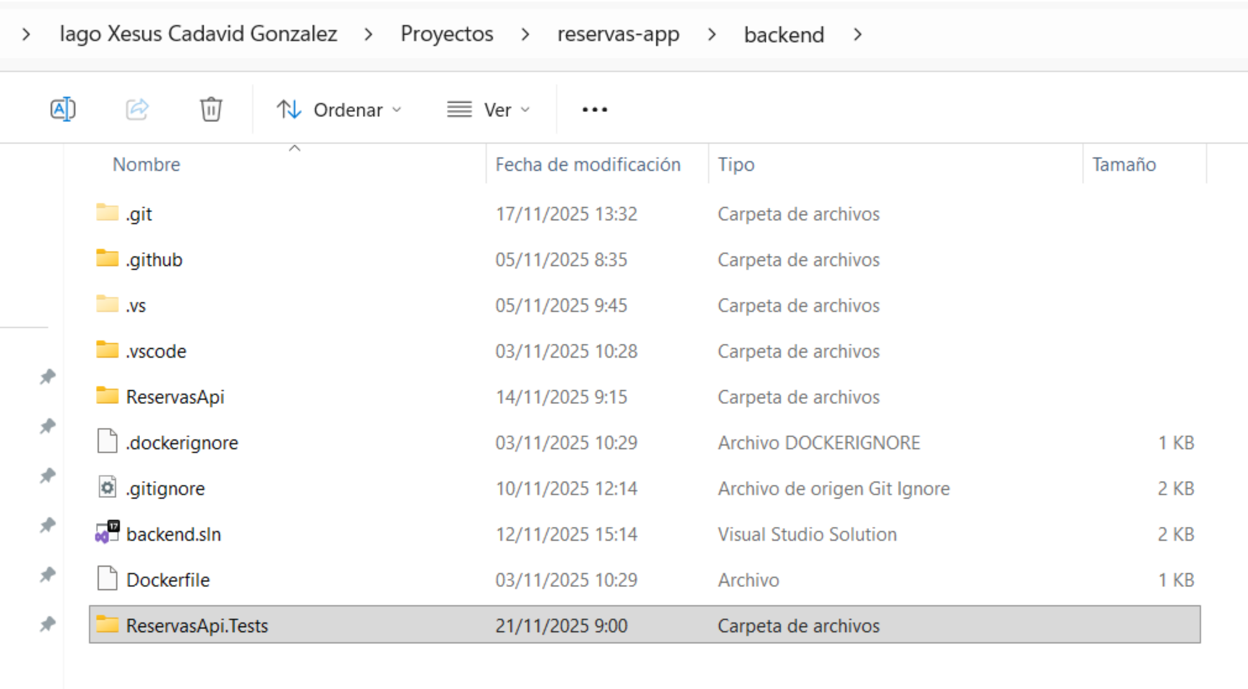
## ✅ Why the `backend` folder

- This is your **solution root**.
- Running the commands here keeps the test project **next to your API project**, not inside it.
- It allows you to **add the test project to the solution** and reference your API project correctly.



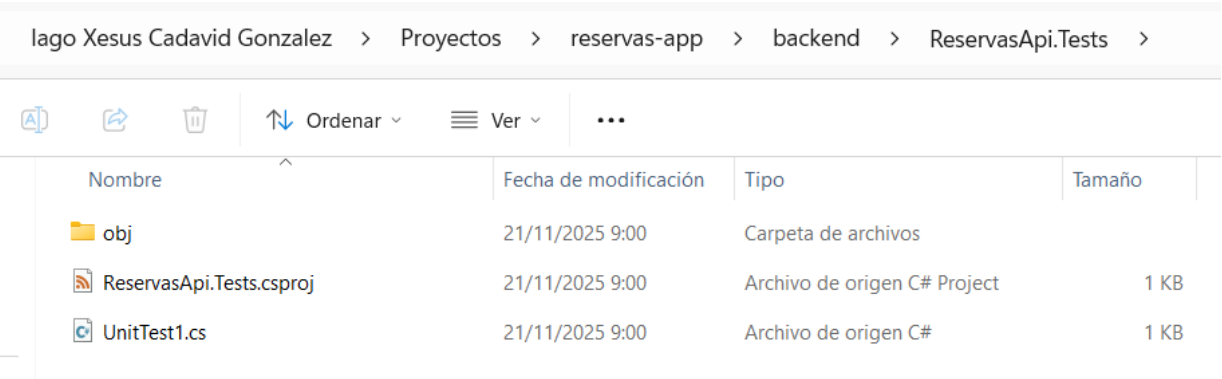
```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet new xunit -n ReservasApi.Tests
La plantilla "Proyecto de prueba de xUnit" se creó correctamente.

Procesando acciones posteriores a la creación...
Restaurando C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj:
Restauración realizada correctamente.
```

Ese comando crea el xUnit test project.



> Iago Xesus Cadavid Gonzalez > Proyectos > reservas-app > backend >

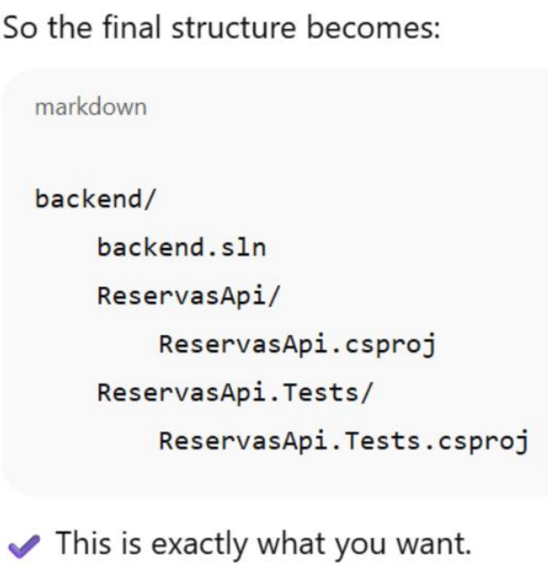
| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|---|---|---|
| .git | 17/11/2025 13:32 | Carpeta de archivos | |
| .github | 05/11/2025 8:35 | Carpeta de archivos | |
| .vs | 05/11/2025 9:45 | Carpeta de archivos | |
| .vscode | 03/11/2025 10:28 | Carpeta de archivos | |
| ReservasApi | 14/11/2025 9:15 | Carpeta de archivos | |
| .dockerignore | 03/11/2025 10:29 | Archivo DOCKERIGNORE | 1 KB |
| .gitignore | 10/11/2025 12:14 | Archivo de origen Git Ignore | 2 KB |
| backend.sln | 12/11/2025 15:14 | Visual Studio Solution | 2 KB |
| Dockerfile | 03/11/2025 10:29 | Archivo | 1 KB |
| ReservasApi.Tests | 21/11/2025 9:00 | Carpeta de archivos | |



Iago Xesus Cadavid Gonzalez > Proyectos > reservas-app > backend > ReservasApi.Tests >

| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|---|---|---|
| obj | 21/11/2025 9:00 | Carpeta de archivos | |
| ReservasApi.Tests.csproj | 21/11/2025 9:00 | Archivo de origen C# Project | 1 KB |
| UnitTest1.cs | 21/11/2025 9:00 | Archivo de origen C# | 1 KB |

PÁGINA SIGUIENTE



So the final structure becomes:

markdown

```
backend/
    backend.sln
    ReservasApi/
        ReservasApi.csproj
    ReservasApi.Tests/
        ReservasApi.Tests.csproj
```

✓ This is exactly what you want.

## El segundo paso es to **add the test project to** your **solution**

Still inside the backend folder i.e. habiendo abierto el cmd desde backend folder, ejecutamos:

**dotnet sln add ReservasApi.Tests**

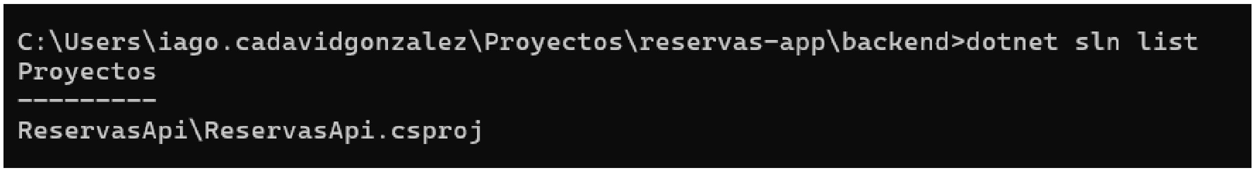
Pero antes de hacerlo y por motivos de comparación después, vamos ver que tengo ya ahora en el (backend) sln file, para ello lo abro en el bloc de notas:

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio Version 17
VisualStudioVersion = 17.5.2.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "ReservasApi", "ReservasApi\ReservasApi.csproj", "{0428B5C6-9D5D-6F3A-0F4C-43576921DA8E}"
EndProject
Global
        GlobalSection(SolutionConfigurationPlatforms) = preSolution
                Debug|Any CPU = Debug|Any CPU
                Release|Any CPU = Release|Any CPU
        EndGlobalSection
        GlobalSection(ProjectConfigurationPlatforms) = postSolution
                {0428B5C6-9D5D-6F3A-0F4C-43576921DA8E}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
                {0428B5C6-9D5D-6F3A-0F4C-43576921DA8E}.Debug|Any CPU.Build.0 = Debug|Any CPU
                {0428B5C6-9D5D-6F3A-0F4C-43576921DA8E}.Release|Any CPU.ActiveCfg = Release|Any CPU
                {0428B5C6-9D5D-6F3A-0F4C-43576921DA8E}.Release|Any CPU.Build.0 = Release|Any CPU
        EndGlobalSection
        GlobalSection(SolutionProperties) = preSolution
                HideSolutionNode = FALSE
        EndGlobalSection
        GlobalSection(ExtensibilityGlobals) = postSolution
                SolutionGuid = {E3B00B15-7A4A-435D-B5BE-3B45232BF96E}
        EndGlobalSection
EndGlobal
```

Me dice chatgpt de centrarme en la línea Project("{GUID}") = etc., "{PROJECT-GUID}"

A ese respecto, dice que "**each Project(...) block** represents **a project included in your solution**", since there is only one block for ReservasApi, that's the only project currently in the solution.

Podemos ver lo mismo using the dotnet CLI to list projects:

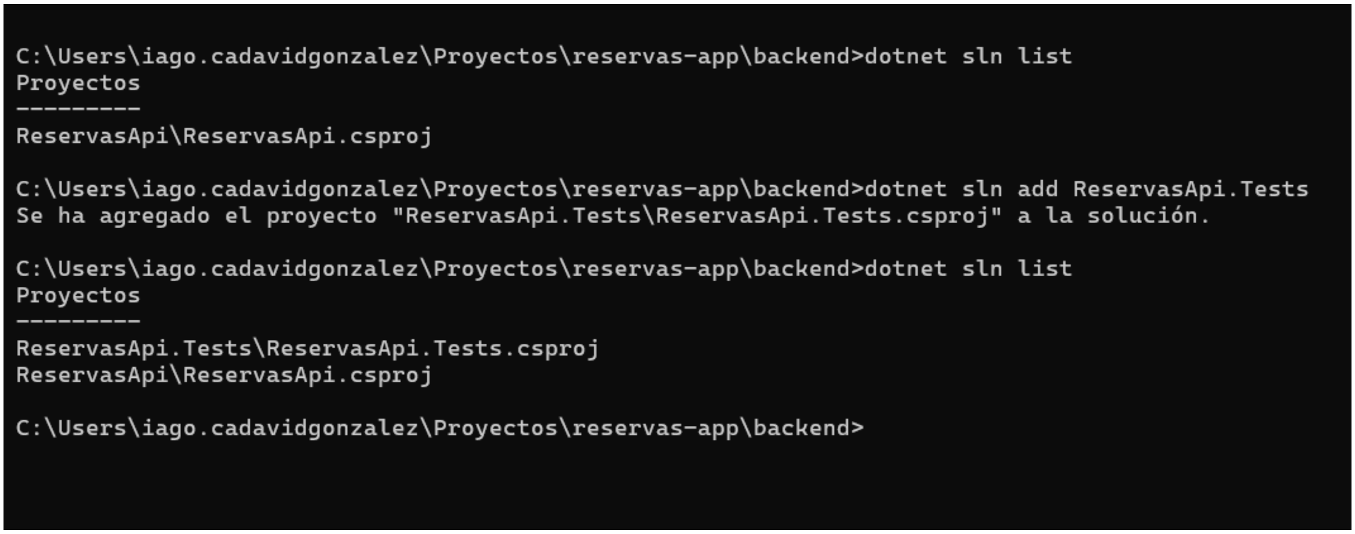


```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet sln list
Proyectos
---------
ReservasApi\ReservasApi.csproj
```

Pues bien, vamos ahora con:

habiendo abierto el cmd desde backend folder, ejecutamos:

**dotnet sln add ReservasApi.Tests**



```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet sln list
Proyectos
---------
ReservasApi\ReservasApi.csproj

C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet sln add ReservasApi.Tests
Se ha agregado el proyecto "ReservasApi.Tests\ReservasApi.Tests.csproj" a la solución.

C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet sln list
Proyectos
---------
ReservasApi.Tests\ReservasApi.Tests.csproj
ReservasApi\ReservasApi.csproj

C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>
```

## En tercer lugar, **add a reference from the test project -> API project**

Since the API project is in ReservasApi/ReservasApi.csproj, run:

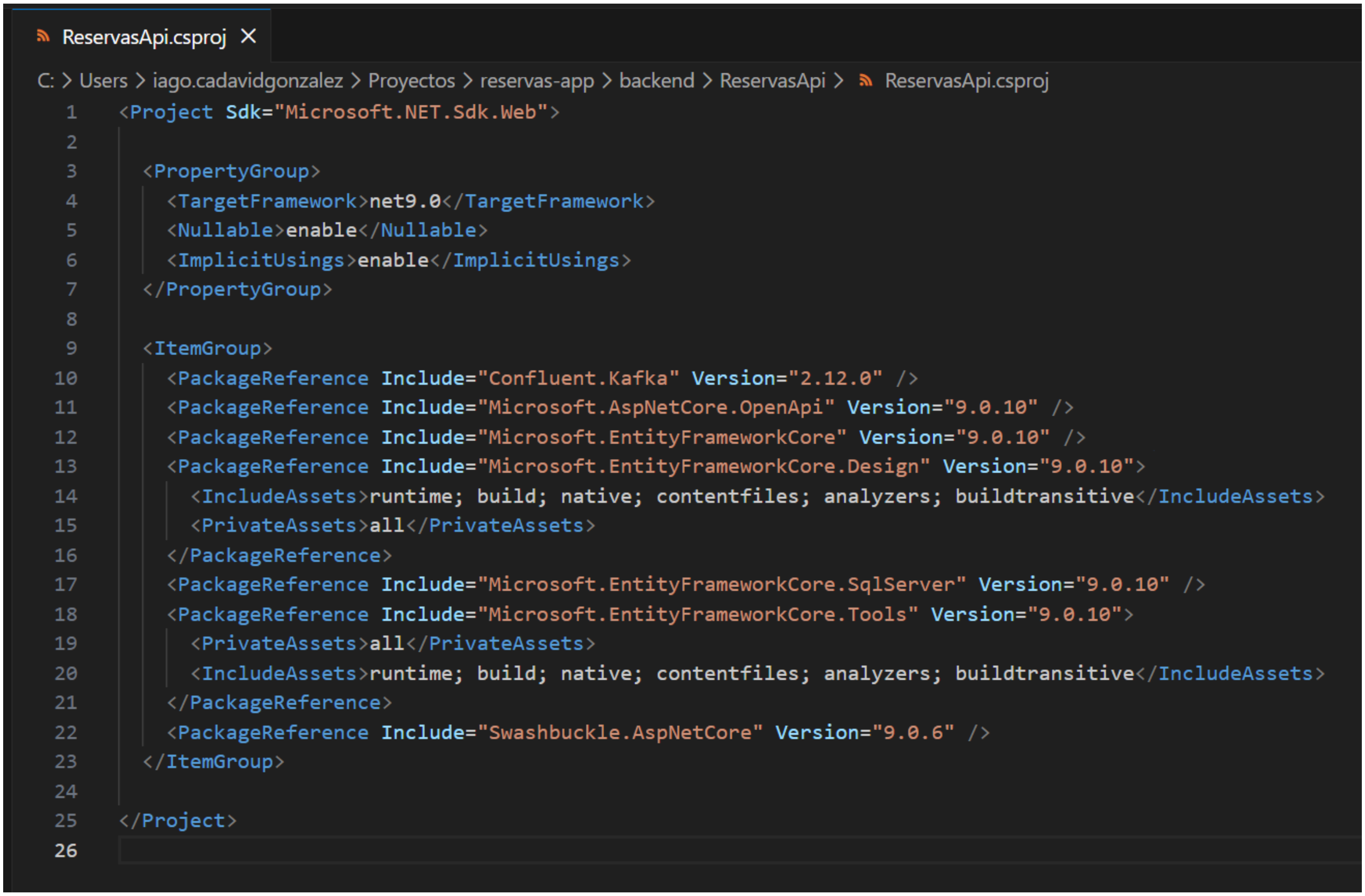
**dotnet add ReservasApi.Tests reference ReservasApi** (once again we run it from the solution root i.e. the ‘backend’ folder that contains backend.sln).

This **links the test project to the API code**.

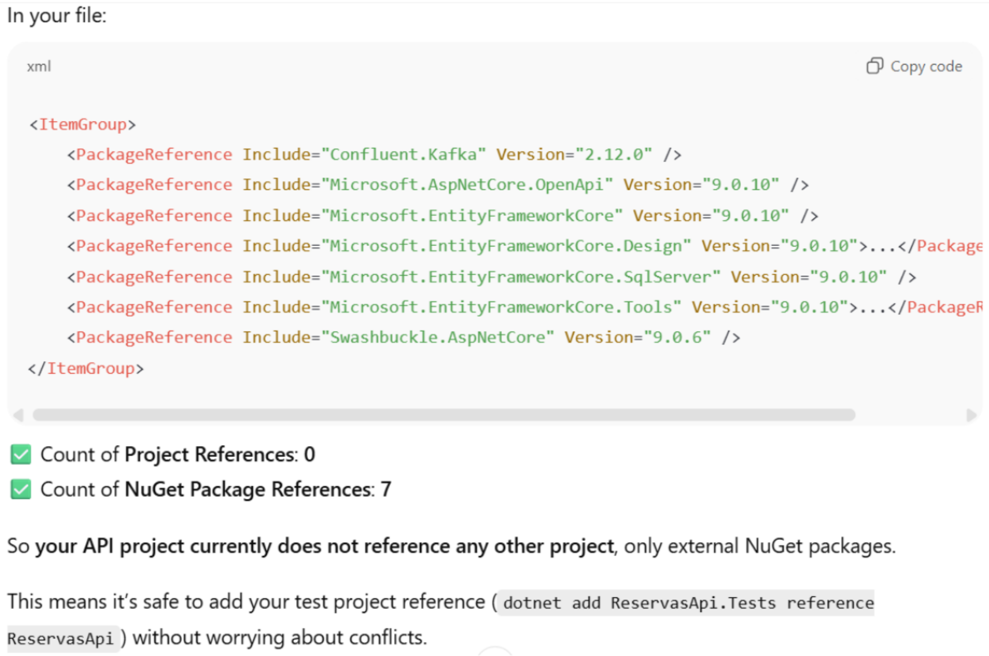
Pero nuevamente vamos ver primero what project references are already added to my API project (ReservasApi.csproj) before adding the test project reference (to the API project).

Para ello, en un editor de texto abrimos el .csproj file (hijo de ReservasApi) and look for **<ItemGroup>** sections that contain **<ProjectReference>** tags.

Mi ReservasApi.csproj pre-añadir la referencia : NEXT PAGE


ReservasApi.csproj
C: > Users > iago.cadavidgonzalez > Proyectos > reservas-app > backend > ReservasApi > ReservasApi.csproj

```xml
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Confluent.Kafka" Version="2.12.0" />
    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="9.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="9.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="9.0.10">
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
      <PrivateAssets>all</PrivateAssets>
    </PackageReference>
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="9.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="9.0.10">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="9.0.6" />
  </ItemGroup>

</Project>

```



In your file:

```xml
<ItemGroup>
    <PackageReference Include="Confluent.Kafka" Version="2.12.0" />
    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="9.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="9.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="9.0.10">...</Package
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="9.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="9.0.10">...</PackageR
    <PackageReference Include="Swashbuckle.AspNetCore" Version="9.0.6" />
</ItemGroup>
```

✅ Count of **Project References**: 0

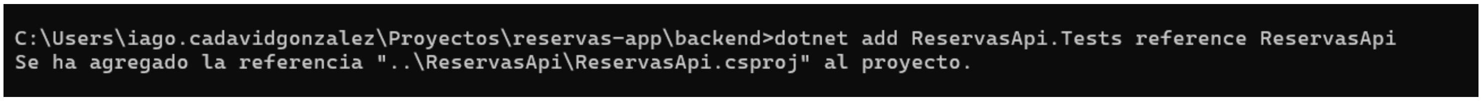
✅ Count of **NuGet Package References**: 7

So **your API project currently does not reference any other project**, only external NuGet packages.

This means it's safe to add your test project reference (`dotnet add ReservasApi.Tests reference ReservasApi`) without worrying about conflicts.

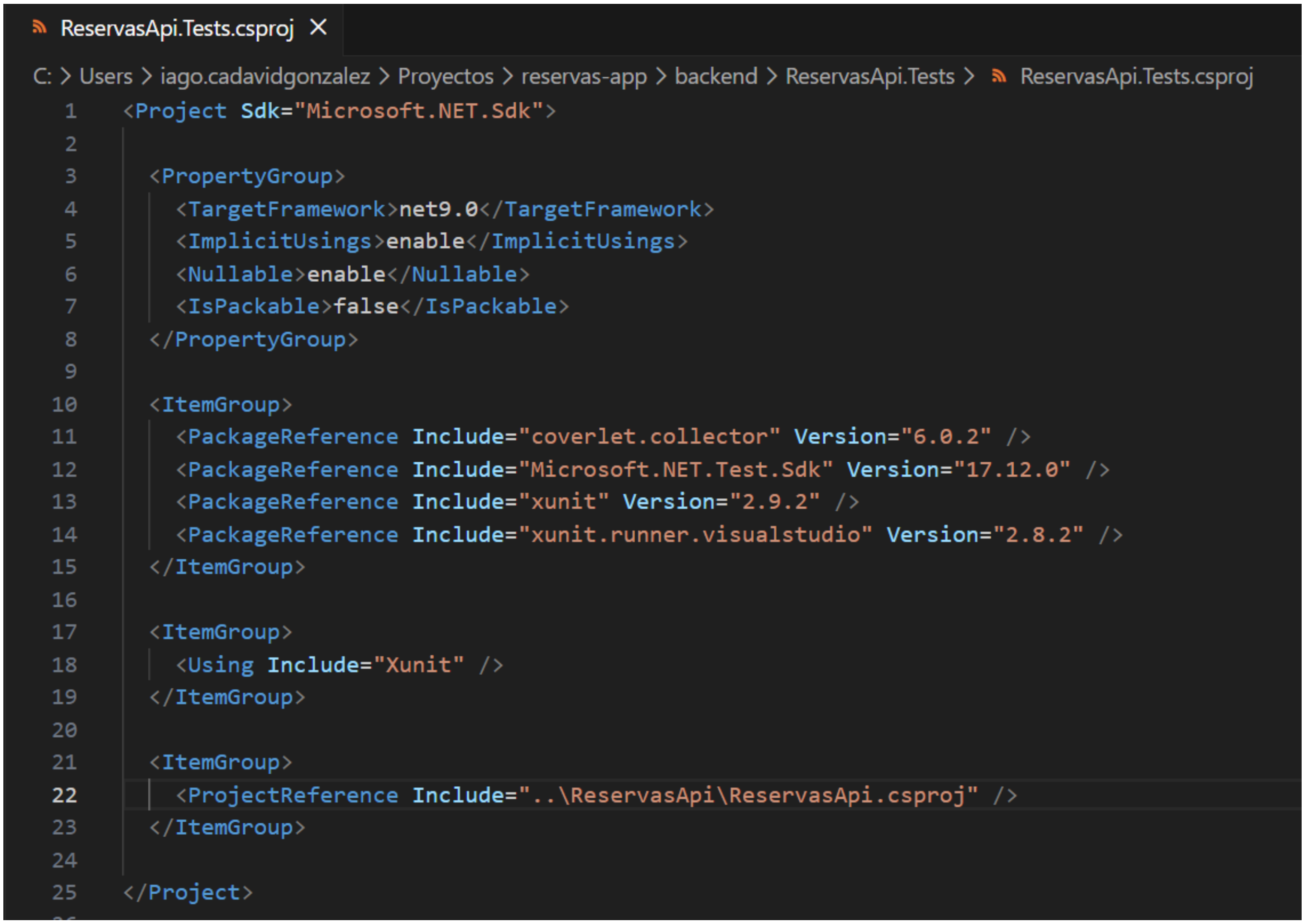
Nótese que ahí en negrita pone al API project como sujeto activo de referenciar, mientras que antes dijimos que es lo referenciado.

Bueno, vamos a añadir ahora la referencia y ver que aparece en ReservasApi.csproj :



```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet add ReservasApi.Tests reference ReservasApi
Se ha agregado la referencia "..\ReservasApi\ReservasApi.csproj" al proyecto.
```

Resulta que si ahora voy a ReservasApi.csproj, sigo viendo lo mismo que antes osease NO reference, pero esto se debe a que donde realmente aparece esta refe que acabamos de añadir es en ReservasApi.Tests.csproj:


ReservasApi.Tests.csproj ×
C: > Users > iago.cadavidgonzalez > Proyectos > reservas-app > backend > ReservasApi.Tests > ReservasApi.Tests.csproj

```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsPackable>false</IsPackable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="coverlet.collector" Version="6.0.2" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.12.0" />
    <PackageReference Include="xunit" Version="2.9.2" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.8.2" />
  </ItemGroup>

  <ItemGroup>
    <Using Include="Xunit" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\ReservasApi\ReservasApi.csproj" />
  </ItemGroup>

</Project>
```

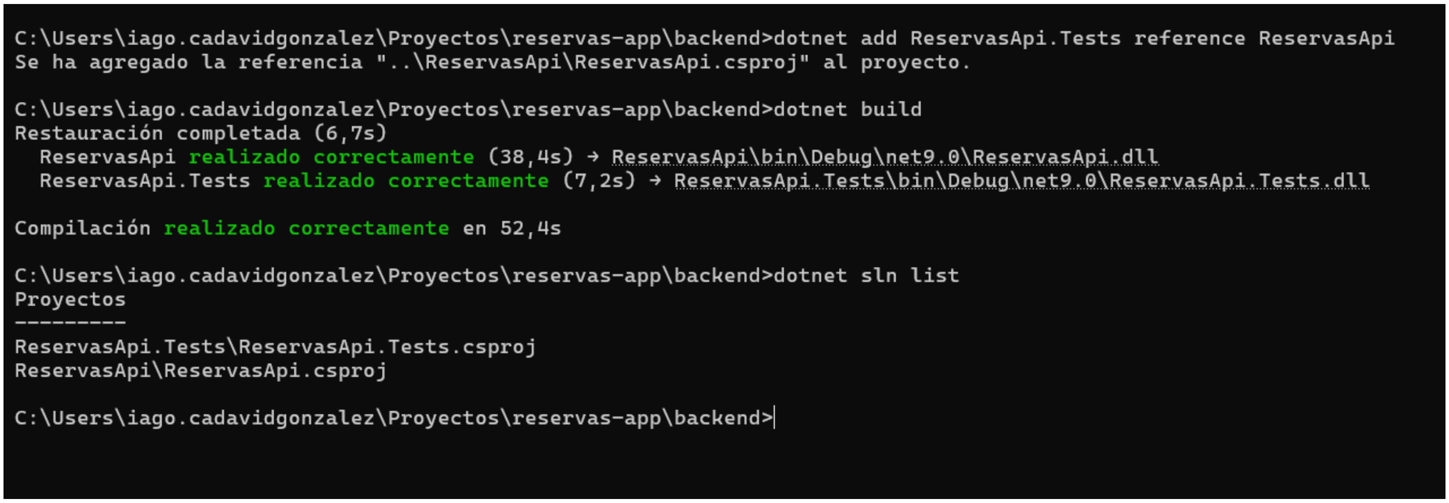
Porque el comando **dotnet add ReservasApi.Tests reference ReservasApi** means:

“**Add** a **reference** to the ReservasApi project **inside** the **ReservasApi.Tests project”** (inside se refiere al add)

Lo siguiente parece ser que es “**generate** the first **test file** for your ReservasApi.project” y “**test** your first **controller** or service”.

Since I’ve linked my test project (ReservasApi.Tests) to my API (ReservasApi), now my **test project can see all** my **controllers**, services **and models** (los de ReservasApi, supongo).

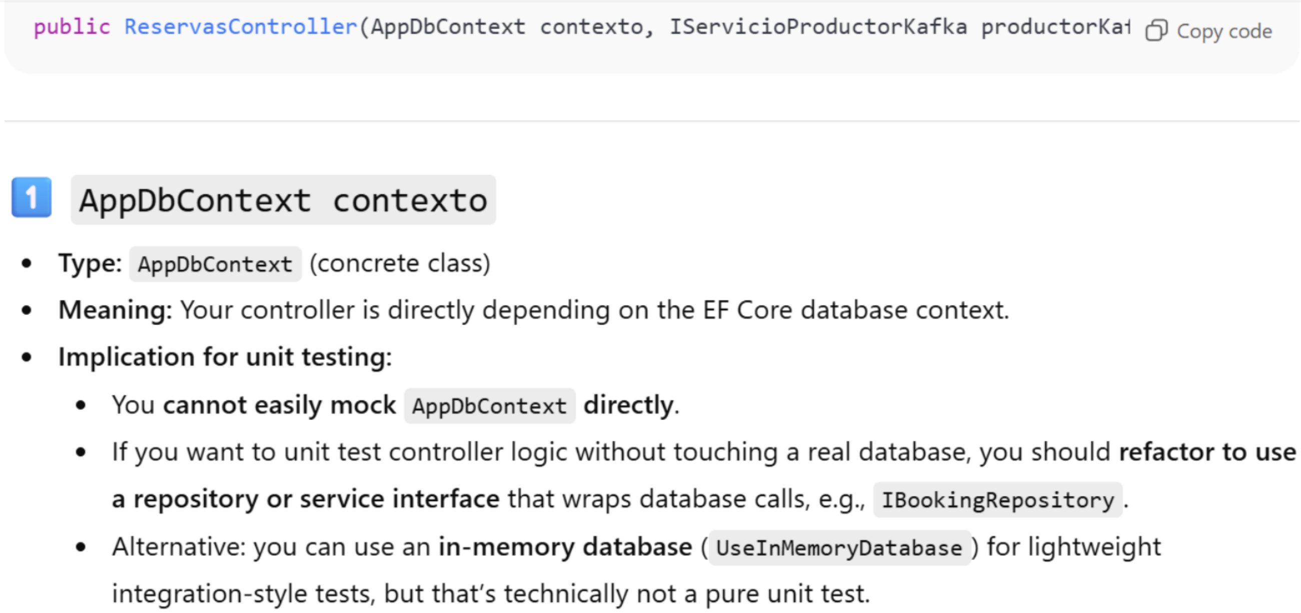
To verify that everything is set up correctly, corremos dotnet build from the solution root (‘backend’ folder)



```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet add ReservasApi.Tests reference ReservasApi
Se ha agregado la referencia "..\ReservasApi\ReservasApi.csproj" al proyecto.

C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet build
Restauración completada (6,7s)
  ReservasApi realizado correctamente (38,4s) → ReservasApi\bin\Debug\net9.0\ReservasApi.dll
  ReservasApi.Tests realizado correctamente (7,2s) → ReservasApi.Tests\bin\Debug\net9.0\ReservasApi.Tests.dll

Compilación realizado correctamente en 52,4s

C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet sln list
Proyectos
---------
ReservasApi.Tests\ReservasApi.Tests.csproj
ReservasApi\ReservasApi.csproj

C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>
```

If there are no errors, your test project (ReservasApi.Tests) correctly references your API (ReservasApi).

Now let’s carefully go through how to **determine whether your controllers depend on interfaces or repositories**, because **this affects how you write unit tests** and **wheter you need mocks**.



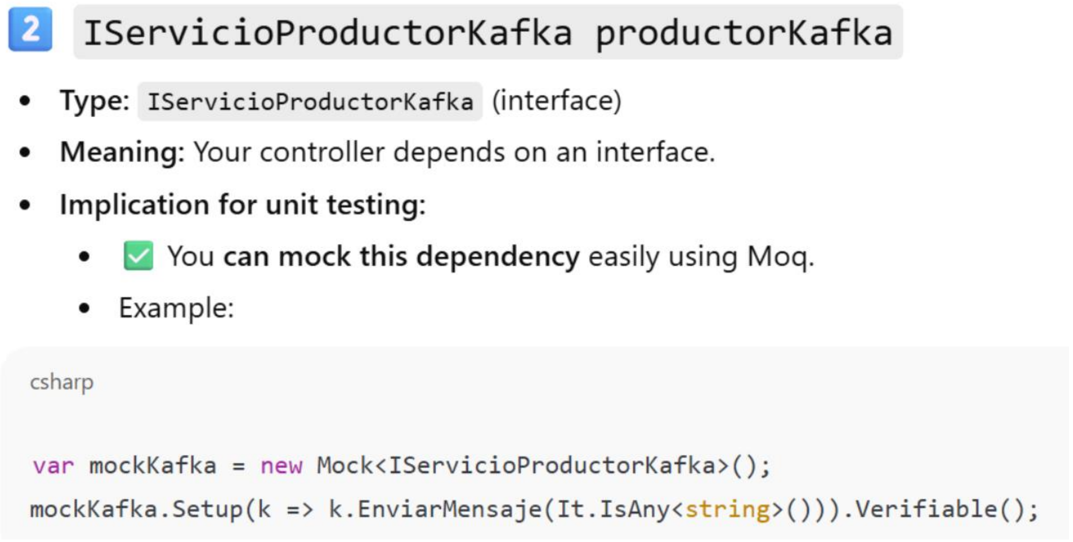
```
public ReservasController(AppDbContext contexto, IServicioProductorKafka productorKa
```

---

### 1️⃣ `AppDbContext contexto`

- **Type:** `AppDbContext` (concrete class)
- **Meaning:** Your controller is directly depending on the EF Core database context.
- **Implication for unit testing:**
  - You **cannot easily mock** `AppDbContext` **directly**.
  - If you want to unit test controller logic without touching a real database, you should **refactor to use a repository or service interface** that wraps database calls, e.g., `IBookingRepository`.
  - Alternative: you can use an **in-memory database** (`UseInMemoryDatabase`) for lightweight integration-style tests, but that’s technically not a pure unit test.

NEXT PAGE

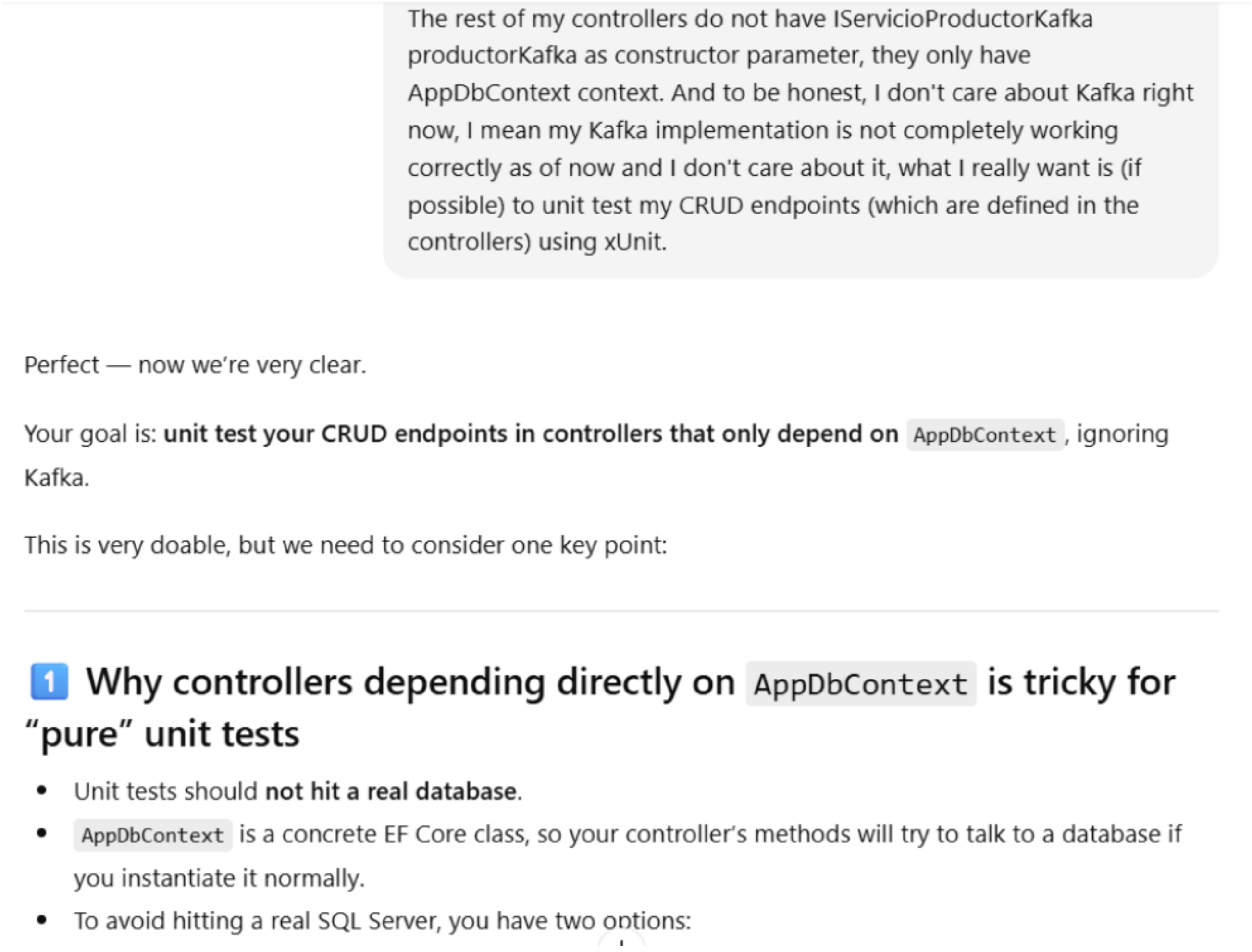


## 2️⃣ `IServicioProductorKafka productorKafka`

- **Type:** `IServicioProductorKafka` (interface)
- **Meaning:** Your controller depends on an interface.
- **Implication for unit testing:**
  - ✅ You **can mock this dependency** easily using Moq.
  - Example:

```csharp
var mockKafka = new Mock<IServicioProductorKafka>();
mockKafka.Setup(k => k.EnviarMensaje(It.IsAny<string>())).Verifiable();
```

F



The rest of my controllers do not have IServicioProductorKafka productorKafka as constructor parameter, they only have AppDbContext context. And to be honest, I don't care about Kafka right now, I mean my Kafka implementation is not completely working correctly as of now and I don't care about it, what I really want is (if possible) to unit test my CRUD endpoints (which are defined in the controllers) using xUnit.

Perfect — now we're very clear.

Your goal is: **unit test your CRUD endpoints in controllers that only depend on** `AppDbContext`, ignoring Kafka.

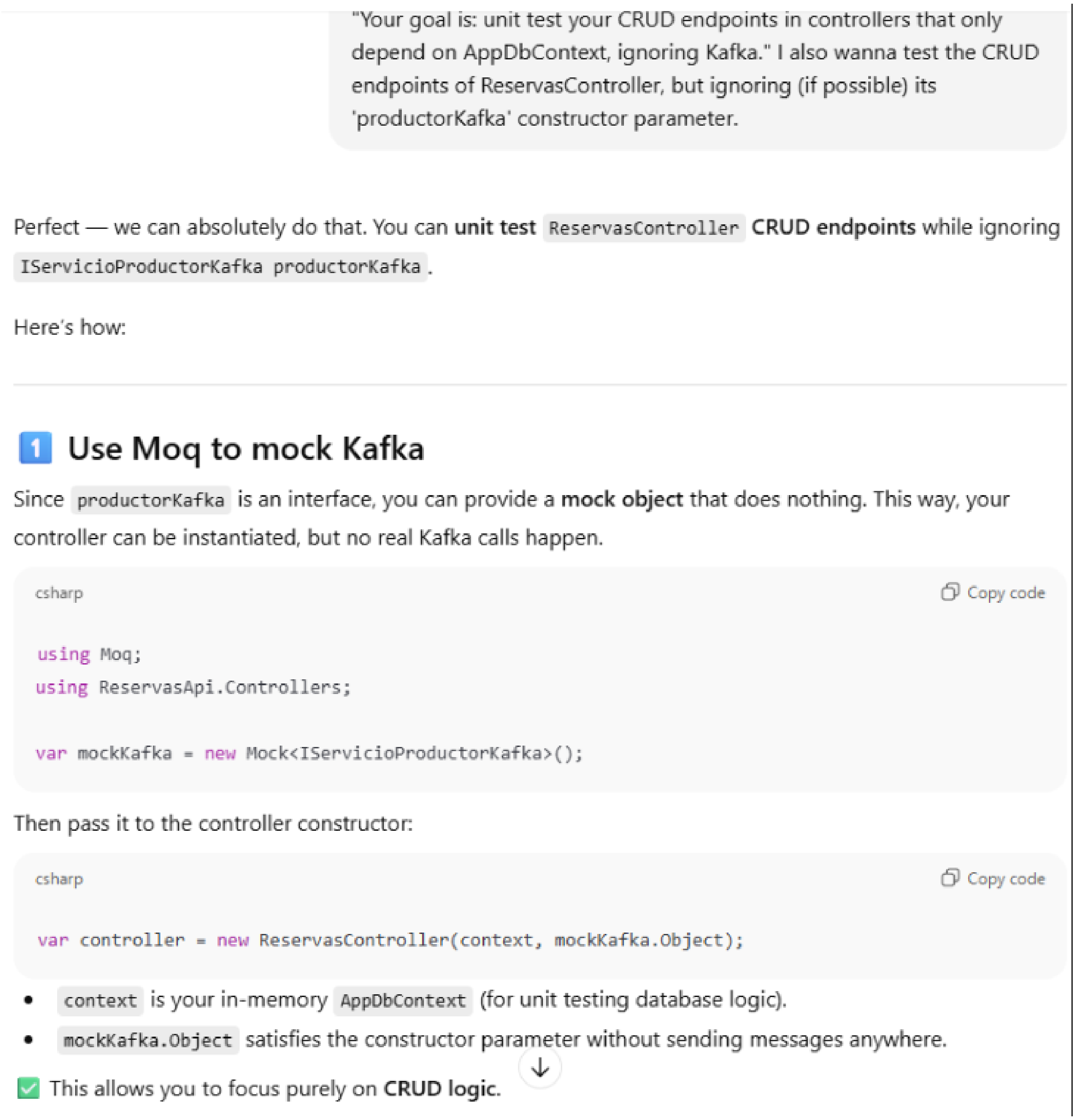
This is very doable, but we need to consider one key point:

---

## 1️⃣ Why controllers depending directly on `AppDbContext` is tricky for "pure" unit tests

- Unit tests should **not hit a real database**.
- `AppDbContext` is a concrete EF Core class, so your controller's methods will try to talk to a database if you instantiate it normally.
- To avoid hitting a real SQL Server, you have two options:

Antes de ir con las 2 opciones, muestro la aclaración que le hice sobre "your goal is:…"



"Your goal is: unit test your CRUD endpoints in controllers that only depend on AppDbContext, ignoring Kafka." I also wanna test the CRUD endpoints of ReservasController, but ignoring (if possible) its 'productorKafka' constructor parameter.

Perfect — we can absolutely do that. You can **unit test** `ReservasController` **CRUD endpoints** while ignoring `IServicioProductorKafka productorKafka`.

Here's how:

---

## 1️⃣ Use Moq to mock Kafka

Since `productorKafka` is an interface, you can provide a **mock object** that does nothing. This way, your controller can be instantiated, but no real Kafka calls happen.

```csharp
using Moq;
using ReservasApi.Controllers;

var mockKafka = new Mock<IServicioProductorKafka>();
```

Then pass it to the controller constructor:

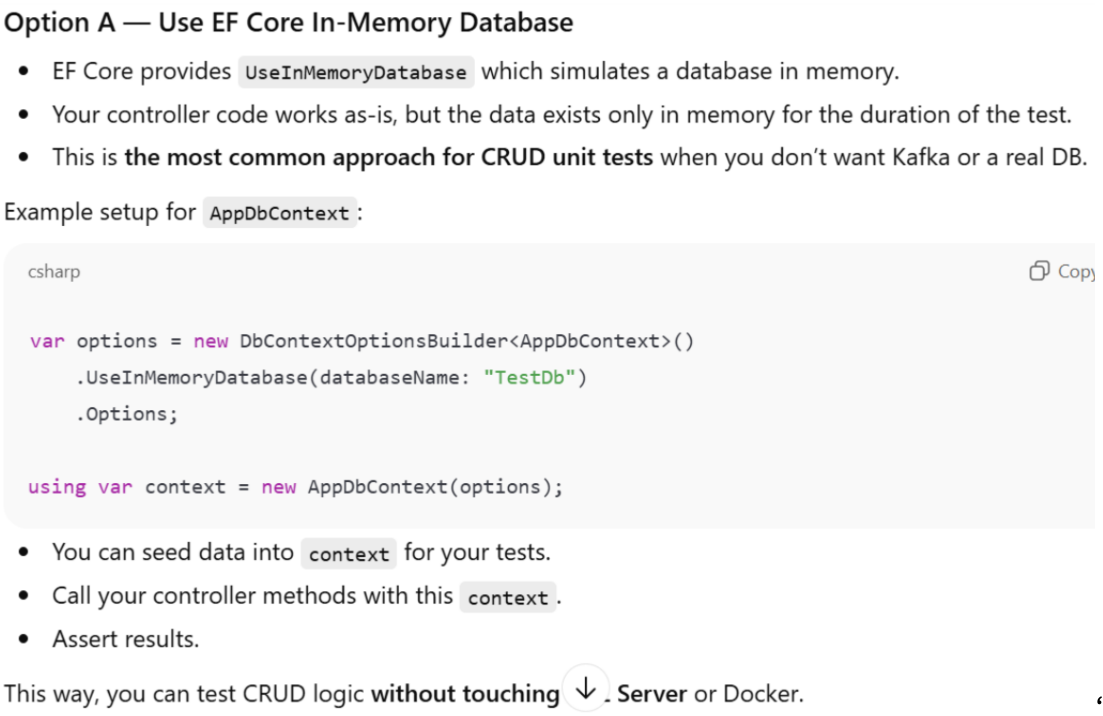
```csharp
var controller = new ReservasController(context, mockKafka.Object);
```

- `context` is your in-memory `AppDbContext` (for unit testing database logic).
- `mockKafka.Object` satisfies the constructor parameter without sending messages anywhere.

✅ This allows you to focus purely on **CRUD logic**.

Ahora sí, vamos con las 2 opcns que chatgpt me da “to avoid hitting a real sql server”:

NEXT PAGE OPTION 1



### Option A — Use EF Core In-Memory Database

- EF Core provides `UseInMemoryDatabase` which simulates a database in memory.
- Your controller code works as-is, but the data exists only in memory for the duration of the test.
- This is **the most common approach for CRUD unit tests** when you don't want Kafka or a real DB.

Example setup for `AppDbContext`:

```csharp
var options = new DbContextOptionsBuilder<AppDbContext>()
    .UseInMemoryDatabase(databaseName: "TestDb")
    .Options;

using var context = new AppDbContext(options);
```

- You can seed data into `context` for your tests.
- Call your controller methods with this `context`.
- Assert results.

This way, you can test CRUD logic **without touching** ↓ **. Server** or Docker.

“without touching SQL Server or Docker.”

Pero recordar que antes dijo que esa opción NO es test unitarios PUROS.

Option B: **Refactor to a repository or service interface** (“you should refactor to use a repo or service int. that wraps db calls”, dijimos antes):

- Create an interface like IReservasRepository that wraps CRUD operations.
- Controller depends on the interface isntead of AppDbContext.
- In unit tests, you mock the interface using Moq.

Advantage of option B is: true unit test with no database involved.

Disadvantage: requires refactoring existing controllers.

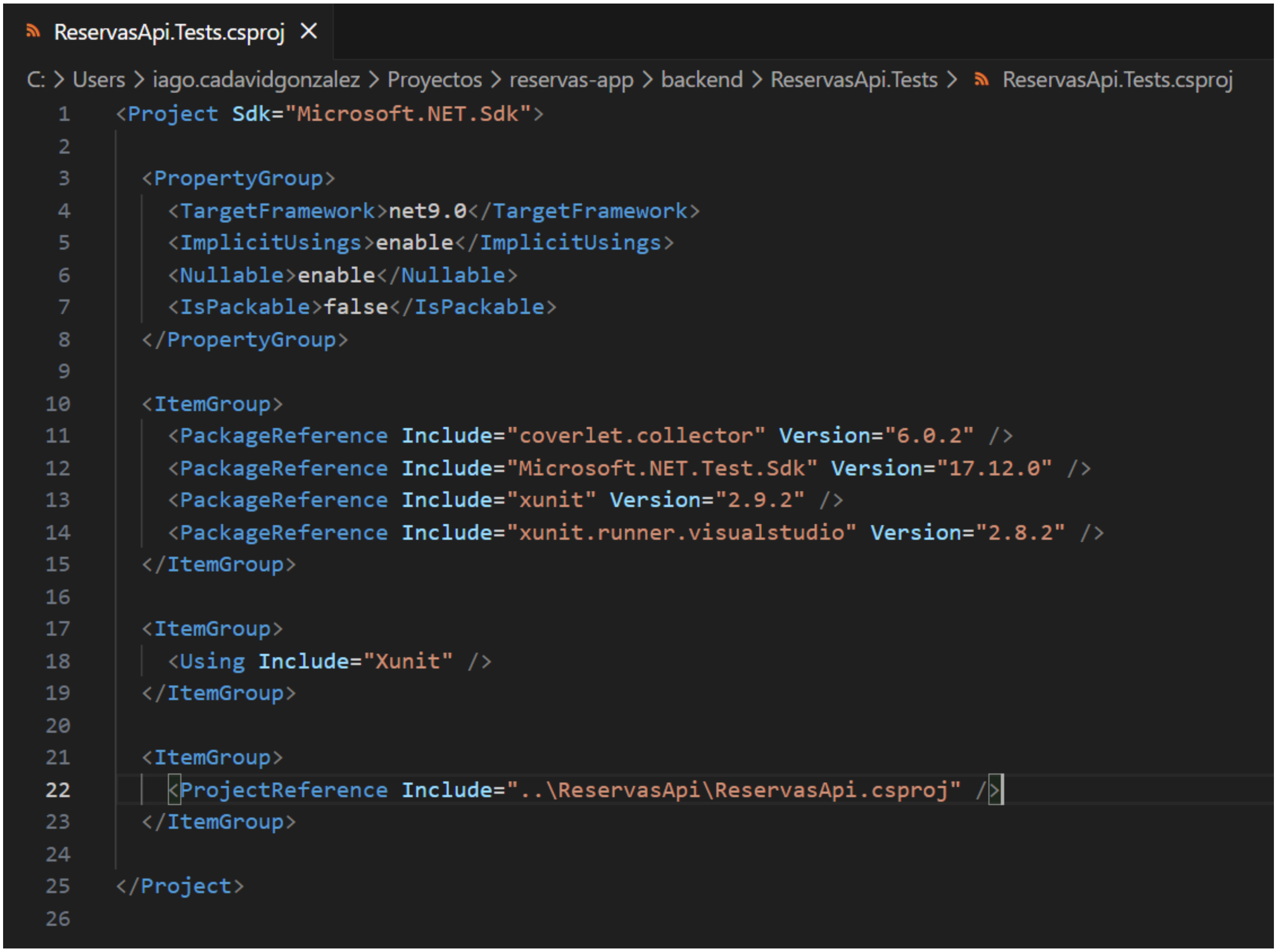
Vamos a hace Option A que le vale a Victor y es más sencilla que la option B.

Pero antes de ponerme con Option A lo que hago es comentar todas las líneas de código Kafka osease la de Program.cs, la de ReservasController, y todas las de ProductorKafka.cs y la interfaz que implementa. Pa que no me interfiera el Kafka con los test unitarios CRUD que voy a hacer con xUnit y UseInMemoryDatabase.

Since Kafka is completely removed, your ReservasController now only depends on AppDbContext (ya que es el único param de su constructor). That means you are ready to write xUnit tests using EFC In-Memory Database.

First of all, make sure your test project is ready i.e. you should already have ReservasApi.Tests created and added to the solution (indeed lo tengo), also ensure it references your API project (indeed ya ejecuté “dotnet add ReservasApi.Tests reference ReservasApi”),

Now make sure you have xUnit installed i.e. package **xunit** and package **xunit.runner.visualstudio** :

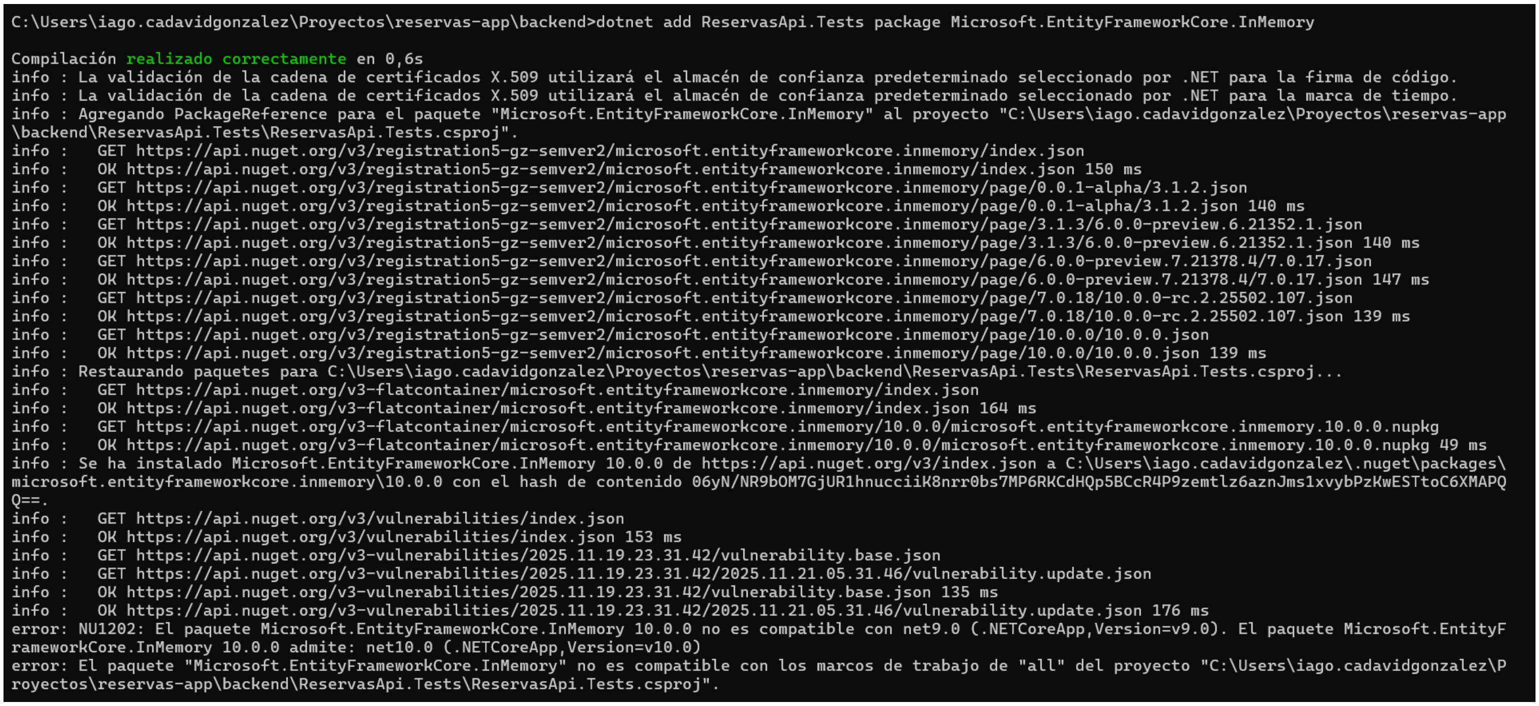

ReservasApi.Tests.csproj
C: > Users > iago.cadavidgonzalez > Proyectos > reservas-app > backend > ReservasApi.Tests > ReservasApi.Tests.csproj

```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsPackable>false</IsPackable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="coverlet.collector" Version="6.0.2" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.12.0" />
    <PackageReference Include="xunit" Version="2.9.2" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.8.2" />
  </ItemGroup>

  <ItemGroup>
    <Using Include="Xunit" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\ReservasApi\ReservasApi.csproj" />
  </ItemGroup>

</Project>

```

When you run **dotnet new xunit –n ReservasApi.Tests** (que ya ejecuté anteriormente), the **xUnit project template automatically adds** the **xunit** and **xunit.runner.visualstudio packages** to he new project (i.e. the test project ReservasApi.Tests).

You do need to manually **install Microsoft.EntityFrameworkCore.InMemory** because that is NOT included in the xUnit template –you will use it for your **in-memory EFCore tests**. Para ello ejecutamos from the root folder of my solution i.e. the directory where my backend.sln file is located, dicho directory en mi caso es la folder ‘backend’:

**dotnet add ReservasApi.Tests package Microsoft.EntityFrameworkCore.InMemory**



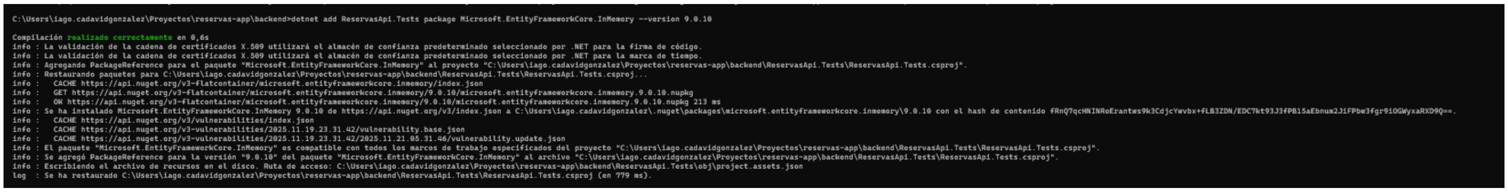
```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet add ReservasApi.Tests package Microsoft.EntityFrameworkCore.InMemory

Compilación realizado correctamente en 0,6s
info : La validación de la cadena de certificados X.509 utilizará el almacén de confianza predeterminado seleccionado por .NET para la firma de código.
info : La validación de la cadena de certificados X.509 utilizará el almacén de confianza predeterminado seleccionado por .NET para la marca de tiempo.
info : Agregando PackageReference para el paquete "Microsoft.EntityFrameworkCore.InMemory" al proyecto "C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app
\backend\ReservasApi.Tests\ReservasApi.Tests.csproj".
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/index.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/index.json 150 ms
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/0.0.1-alpha/3.1.2.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/0.0.1-alpha/3.1.2.json 140 ms
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/3.1.3/6.0.0-preview.6.21352.1.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/3.1.3/6.0.0-preview.6.21352.1.json 140 ms
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/6.0.0-preview.7.21378.4/7.0.17.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/6.0.0-preview.7.21378.4/7.0.17.json 147 ms
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/7.0.18/10.0.0-rc.2.25502.107.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/7.0.18/10.0.0-rc.2.25502.107.json 139 ms
info :   GET https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/10.0.0/10.0.0.json
info :   OK https://api.nuget.org/v3/registration5-gz-semver2/microsoft.entityframeworkcore.inmemory/page/10.0.0/10.0.0.json 139 ms
info : Restaurando paquetes para C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj...
info :   GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/index.json
info :   OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/index.json 164 ms
info :   GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/10.0.0/microsoft.entityframeworkcore.inmemory.10.0.0.nupkg
info :   OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/10.0.0/microsoft.entityframeworkcore.inmemory.10.0.0.nupkg 49 ms
info : Se ha instalado Microsoft.EntityFrameworkCore.InMemory 10.0.0 de https://api.nuget.org/v3/index.json a C:\Users\iago.cadavidgonzalez\.nuget\packages\
microsoft.entityframeworkcore.inmemory\10.0.0 con el hash de contenido 06yN/NR9bOM7GjUR1hnucciiK8nrr0bs7MP6RKCdHQp5BCcR4P9zemtlz6aznJms1xvybPzKwESTtoC6XMAPQ
Q==.
info :   GET https://api.nuget.org/v3/vulnerabilities/index.json
info :   OK https://api.nuget.org/v3/vulnerabilities/index.json 153 ms
info :   GET https://api.nuget.org/v3-vulnerabilities/2025.11.19.23.31.42/vulnerability.base.json
info :   GET https://api.nuget.org/v3-vulnerabilities/2025.11.19.23.31.42/2025.11.21.05.31.46/vulnerability.update.json
info :   OK https://api.nuget.org/v3-vulnerabilities/2025.11.19.23.31.42/vulnerability.base.json 135 ms
info :   OK https://api.nuget.org/v3-vulnerabilities/2025.11.19.23.31.42/2025.11.21.05.31.46/vulnerability.update.json 176 ms
error: NU1202: El paquete Microsoft.EntityFrameworkCore.InMemory 10.0.0 no es compatible con net9.0 (.NETCoreApp,Version=v9.0). El paquete Microsoft.EntityF
rameworkCore.InMemory 10.0.0 admite: net10.0 (.NETCoreApp,Version=v10.0)
error: El paquete "Microsoft.EntityFrameworkCore.InMemory" no es compatible con los marcos de trabajo de "all" del proyecto "C:\Users\iago.cadavidgonzalez\P
royectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj".
```

Según chatgpt, ese error es porque “you are using .NET 9 (net9.0) for your projects, but you tried to install EF Core In-Memory 10.0.0, which only supports .NET 10, that is why NuGet reports “error: NU1202:…Microsoft.EFC.InMemory 10.0.0 NO es compatible con net 9.0”.

To fix it, you need to isntall a version of M.EFC.IM that supports .NET 9, so run:

dotnet add ReservasApi.Tests package Microsoft.EntityFrameworkCore.InMemory **--version 9.0.10**

**--version 9.0.10** is EF Core 9 stable version compatible with .NET 9 (net9.0). This will work with your current projects without upgrading to .NET 10.

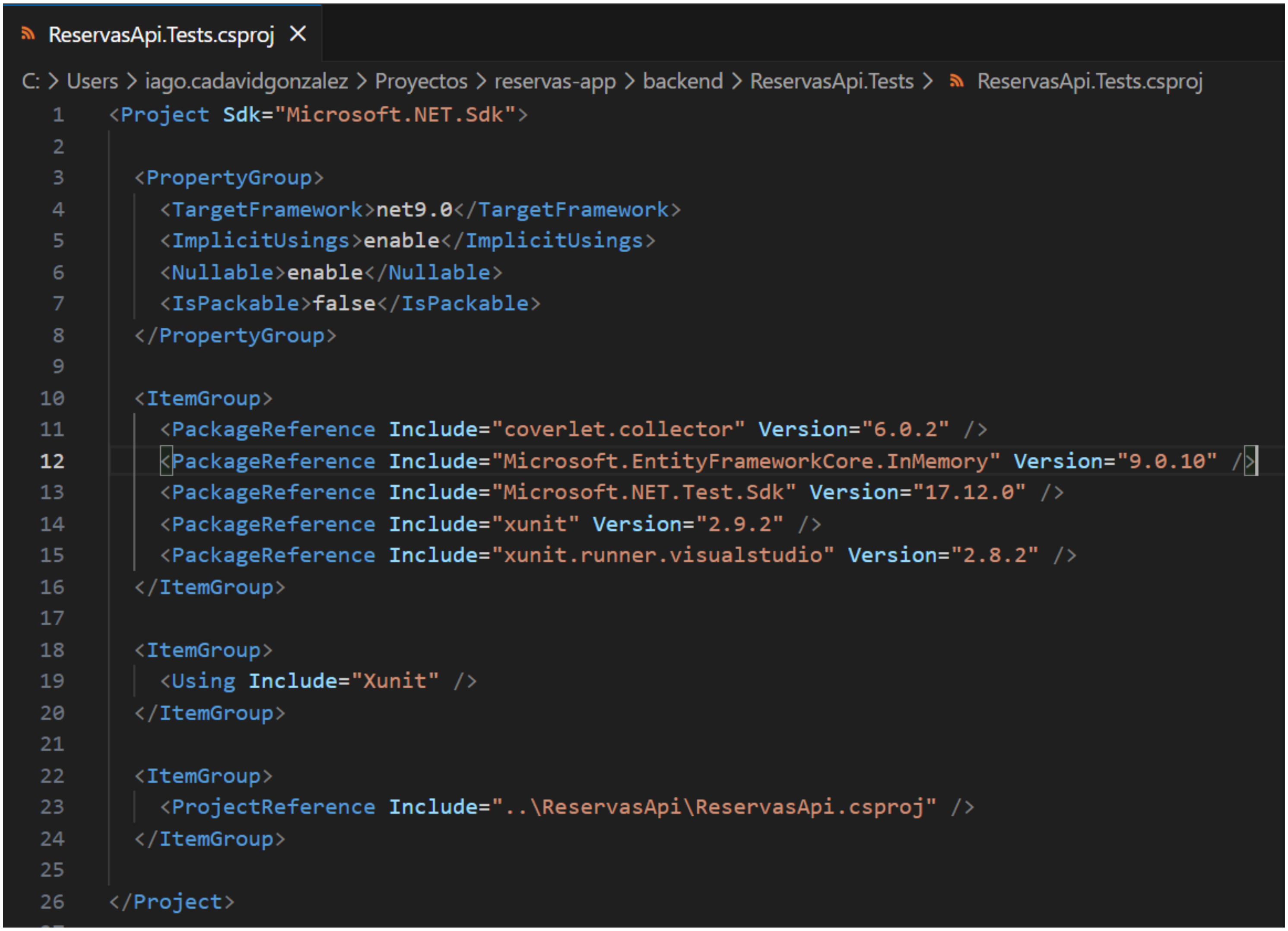


```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet add ReservasApi.Tests package Microsoft.EntityFrameworkCore.InMemory --version 9.0.10

Compilación realizado correctamente en 0,6s
info : La validación de la cadena de certificados X.509 utilizará el almacén de confianza predeterminado seleccionado por .NET para la firma de código.
info : La validación de la cadena de certificados X.509 utilizará el almacén de confianza predeterminado seleccionado por .NET para la marca de tiempo.
info : Agregando PackageReference para el paquete "Microsoft.EntityFrameworkCore.InMemory" al proyecto "C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj".
info : Restaurando paquetes para C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj...
info :   CACHE https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/index.json
info :   GET https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/9.0.10/microsoft.entityframeworkcore.inmemory.9.0.10.nupkg
info :   OK https://api.nuget.org/v3-flatcontainer/microsoft.entityframeworkcore.inmemory/9.0.10/microsoft.entityframeworkcore.inmemory.9.0.10.nupkg 213 ms
info : Se ha instalado Microsoft.EntityFrameworkCore.InMemory 9.0.10 de https://api.nuget.org/v3/index.json a C:\Users\iago.cadavidgonzalez\.nuget\packages\microsoft.entityframeworkcore.inmemory\9.0.10 con el hash de contenido fRnQ7qcHNINRoErantws9k3CdjcYwvbx+fLB3ZDN/EDC7kt93J3fPB15aEbnum2JiFPbw3fgr9iOGWyxaRXD9Q==.
info :   CACHE https://api.nuget.org/v3/vulnerabilities/index.json
info :   CACHE https://api.nuget.org/v3-vulnerabilities/2025.11.19.23.31.42/vulnerability.base.json
info :   CACHE https://api.nuget.org/v3-vulnerabilities/2025.11.19.23.31.42/2025.11.21.05.31.46/vulnerability.update.json
info : El paquete "Microsoft.EntityFrameworkCore.InMemory" es compatible con todos los marcos de trabajo especificados del proyecto "C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj".
info : Se agregó PackageReference para la versión "9.0.10" del paquete "Microsoft.EntityFrameworkCore.InMemory" al archivo "C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj".
info : Escribiendo el archivo de recursos en el disco. Ruta de acceso: C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\obj\project.assets.json
log  : Se ha restaurado C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasApi.Tests.csproj (en 779 ms).
```

“El paquete es compatible con todos los marcos de trabajo del proyecto.

Se agregó PackageRefrence para la versión 9.0.10 del paquete al archivo ReservasApi.Tests.csproj”.

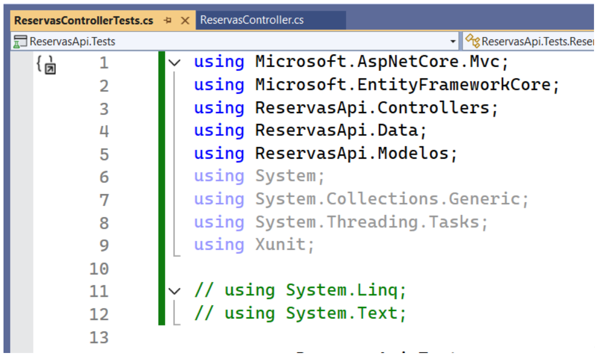
NEXT PAGE


ReservasApi.Tests.csproj
C: > Users > iago.cadavidgonzalez > Proyectos > reservas-app > backend > ReservasApi.Tests > ReservasApi.Tests.csproj

```xml
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFramework>net9.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <IsPackable>false</IsPackable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="coverlet.collector" Version="6.0.2" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.InMemory" Version="9.0.10" />
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="17.12.0" />
    <PackageReference Include="xunit" Version="2.9.2" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.8.2" />
  </ItemGroup>

  <ItemGroup>
    <Using Include="Xunit" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\ReservasApi\ReservasApi.csproj" />
  </ItemGroup>

</Project>
```

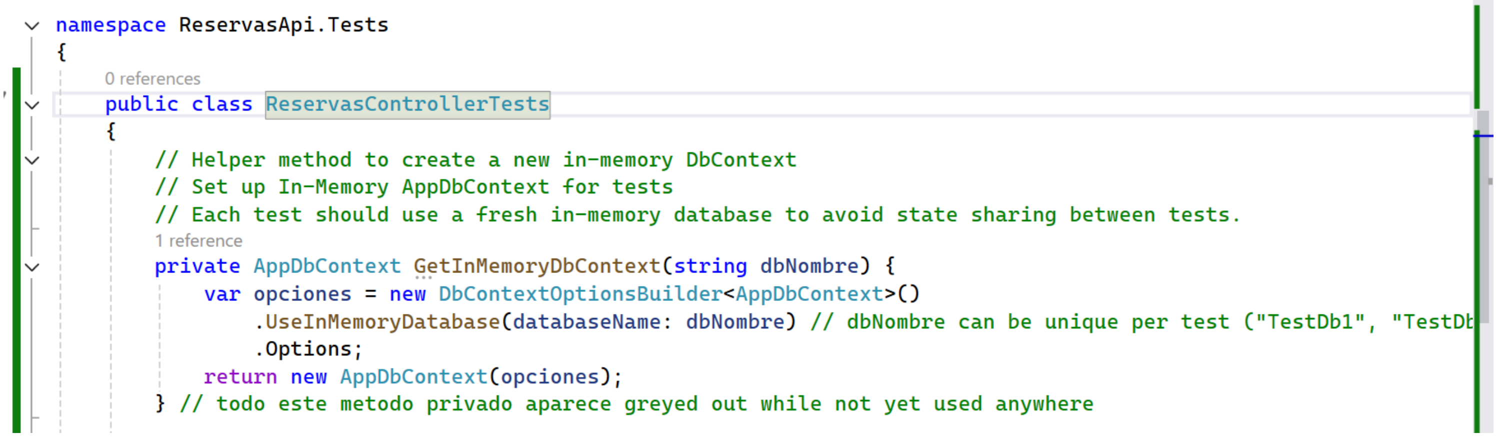
At this point, your test project has everything it needs to start unit testing the CRUD endpoints of ReservasController.

Now it's time to **create a new C# file inside ReservasApi.Tests** i.e. create a test class, call it **ReservasControllerTests.cs,** this class **will contain all** your **unit tests for CRUD endpoints (of ReservasController)**.

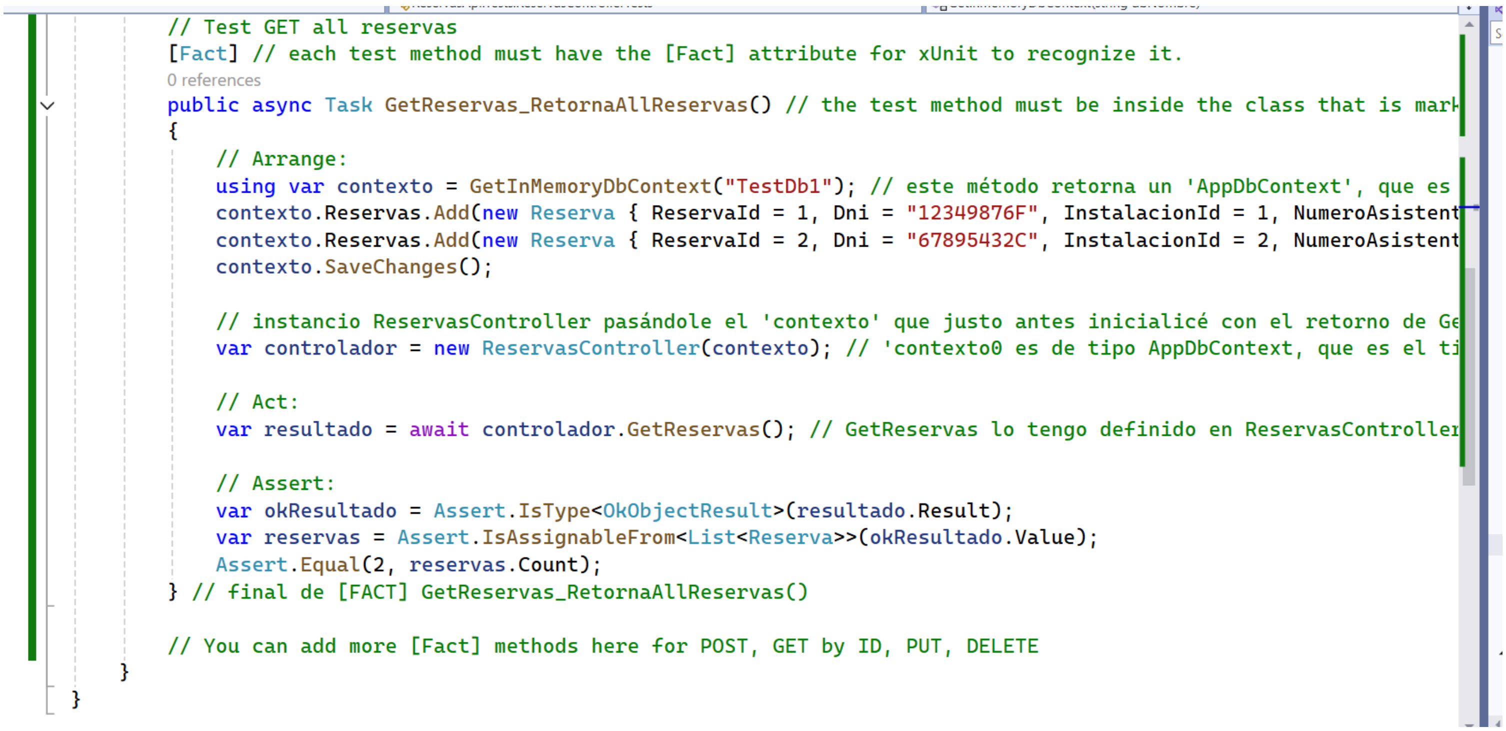


```csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using ReservasApi.Controllers;
using ReservasApi.Data;
using ReservasApi.Modelos;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Xunit;

// using System.Linq;
// using System.Text;
```

Sigue en pág siguiente



```
namespace ReservasApi.Tests
{
    0 references
    public class ReservasControllerTests
    {
        // Helper method to create a new in-memory DbContext
        // Set up In-Memory AppDbContext for tests
        // Each test should use a fresh in-memory database to avoid state sharing between tests.
        1 reference
        private AppDbContext GetInMemoryDbContext(string dbNombre) {
            var opciones = new DbContextOptionsBuilder<AppDbContext>()
                .UseInMemoryDatabase(databaseName: dbNombre) // dbNombre can be unique per test ("TestDb1", "TestDb
                .Options;
            return new AppDbContext(opciones);
        } // todo este metodo privado aparece greyed out while not yet used anywhere
```



```
        // Test GET all reservas
        [Fact] // each test method must have the [Fact] attribute for xUnit to recognize it.
        0 references
        public async Task GetReservas_RetornaAllReservas() // the test method must be inside the class that is mark
        {
            // Arrange:
            using var contexto = GetInMemoryDbContext("TestDb1"); // este método retorna un 'AppDbContext', que es
            contexto.Reservas.Add(new Reserva { ReservaId = 1, Dni = "12349876F", InstalacionId = 1, NumeroAsistent
            contexto.Reservas.Add(new Reserva { ReservaId = 2, Dni = "67895432C", InstalacionId = 2, NumeroAsistent
            contexto.SaveChanges();

            // instancio ReservasController pasándole el 'contexto' que justo antes inicialicé con el retorno de Ge
            var controlador = new ReservasController(contexto); // 'contexto0 es de tipo AppDbContext, que es el ti

            // Act:
            var resultado = await controlador.GetReservas(); // GetReservas lo tengo definido en ReservasControlle

            // Assert:
            var okResultado = Assert.IsType<OkObjectResult>(resultado.Result);
            var reservas = Assert.IsAssignableFrom<List<Reserva>>(okResultado.Value);
            Assert.Equal(2, reservas.Count);
        } // final de [FACT] GetReservas_RetornaAllReservas()

        // You can add more [Fact] methods here for POST, GET by ID, PUT, DELETE
    }
}
```

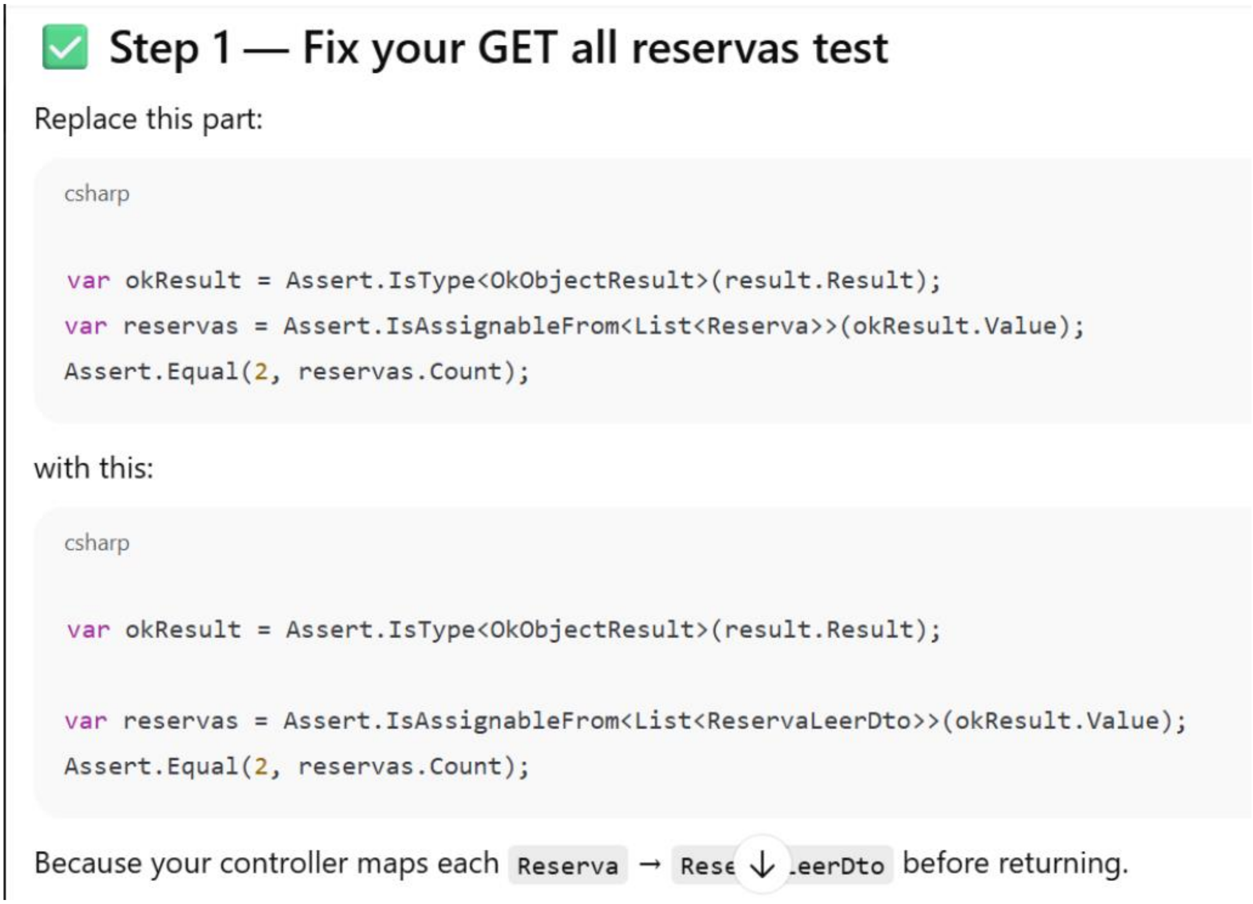
Nº asistentes puse primero 2 y luego 3, nótese que no puse FechaHora, dijo chatgpt que no hacía falta, a pesar de estar en mi modelo de reservas.

You now have a test class, a working in-memory DbContext and your first test method *scaffolded*.

Pero dice que cuando me dio ese código "you had not yeet shown me your actual GetReservas() (Reservas)controller code, so I gave you the *generic* pattern used when a controller returns entities directly".

Pero como ahora sí conoce mi código concreto de ese GetReservas(), dice que debo reemplazar la parte del Assert osease cambiarle un poco el código, quedando la cosa así:

NEXT PAGE



## ✅ Step 1 — Fix your GET all reservas test

Replace this part:

csharp

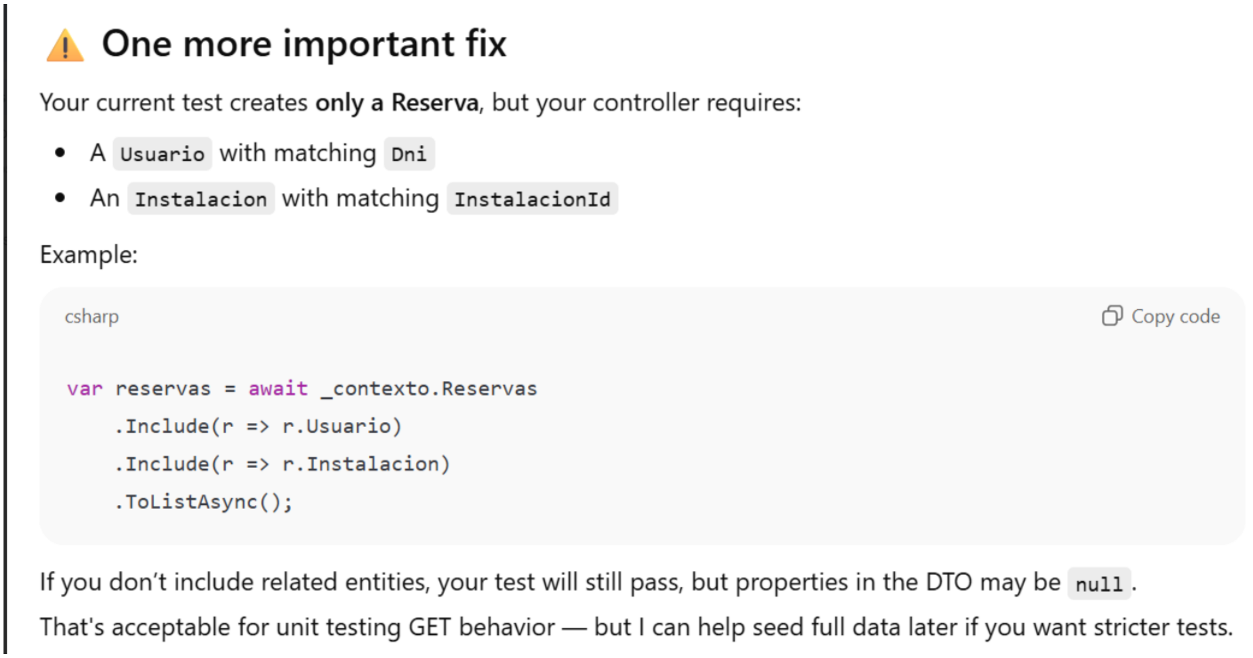
```csharp
var okResult = Assert.IsType<OkObjectResult>(result.Result);
var reservas = Assert.IsAssignableFrom<List<Reserva>>(okResult.Value);
Assert.Equal(2, reservas.Count);
```

with this:

csharp

```csharp
var okResult = Assert.IsType<OkObjectResult>(result.Result);

var reservas = Assert.IsAssignableFrom<List<ReservaLeerDto>>(okResult.Value);
Assert.Equal(2, reservas.Count);
```

Because your controller maps each `Reserva` → `Rese ↓ .eerDto` before returning.



## ⚠️ One more important fix

Your current test creates **only a Reserva**, but your controller requires:

- A `Usuario` with matching `Dni`
- An `Instalacion` with matching `InstalacionId`

Example:

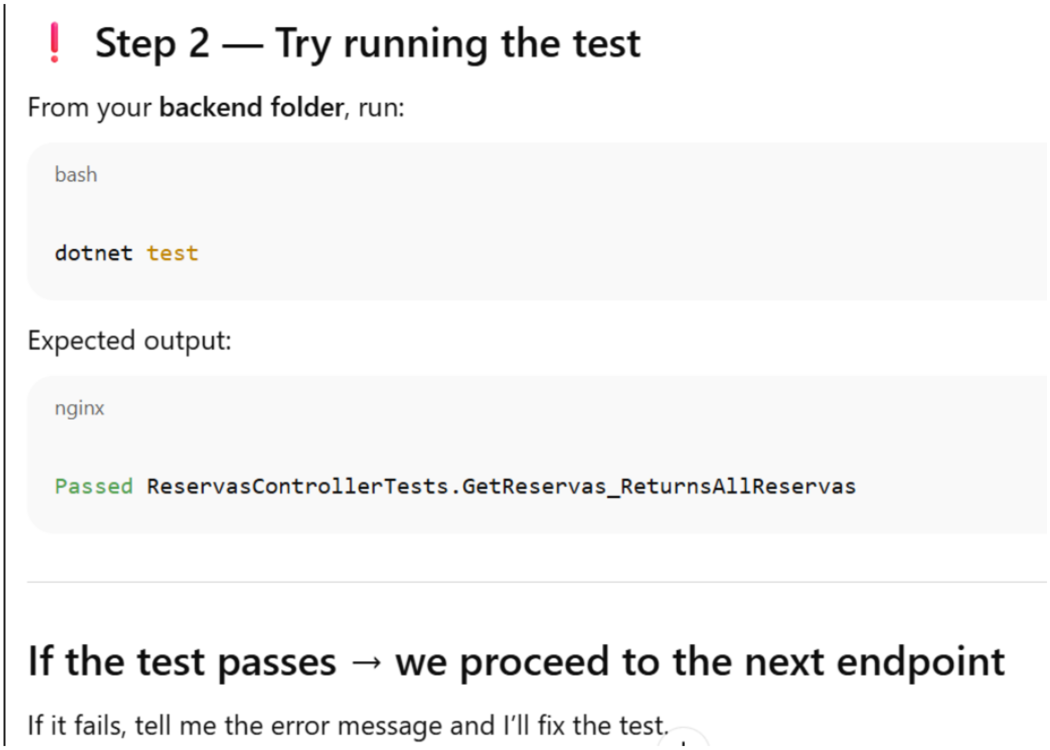
csharp

```csharp
var reservas = await _contexto.Reservas
    .Include(r => r.Usuario)
    .Include(r => r.Instalacion)
    .ToListAsync();
```

If you don't include related entities, your test will still pass, but properties in the DTO may be `null`.
That's acceptable for unit testing GET behavior — but I can help seed full data later if you want stricter tests.

Ese código del “Example” está en mi get api reservas de ReservasController.cs.

De momento no hago ese fix ya que dice “your test will still pass”.



## ❗ Step 2 — Try running the test

From your **backend folder**, run:

```bash
dotnet test
```

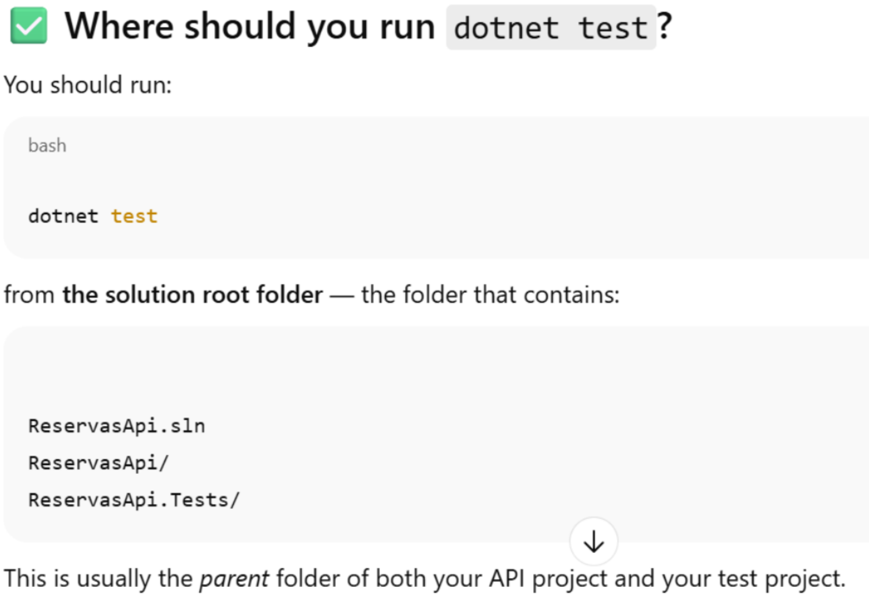
Expected output:

```nginx
Passed ReservasControllerTests.GetReservas_ReturnsAllReservas
```

---

## If the test passes → we proceed to the next endpoint

If it fails, tell me the error message and I'll fix the test.



## ✅ Where should you run `dotnet test`?

You should run:

```bash
dotnet test
```

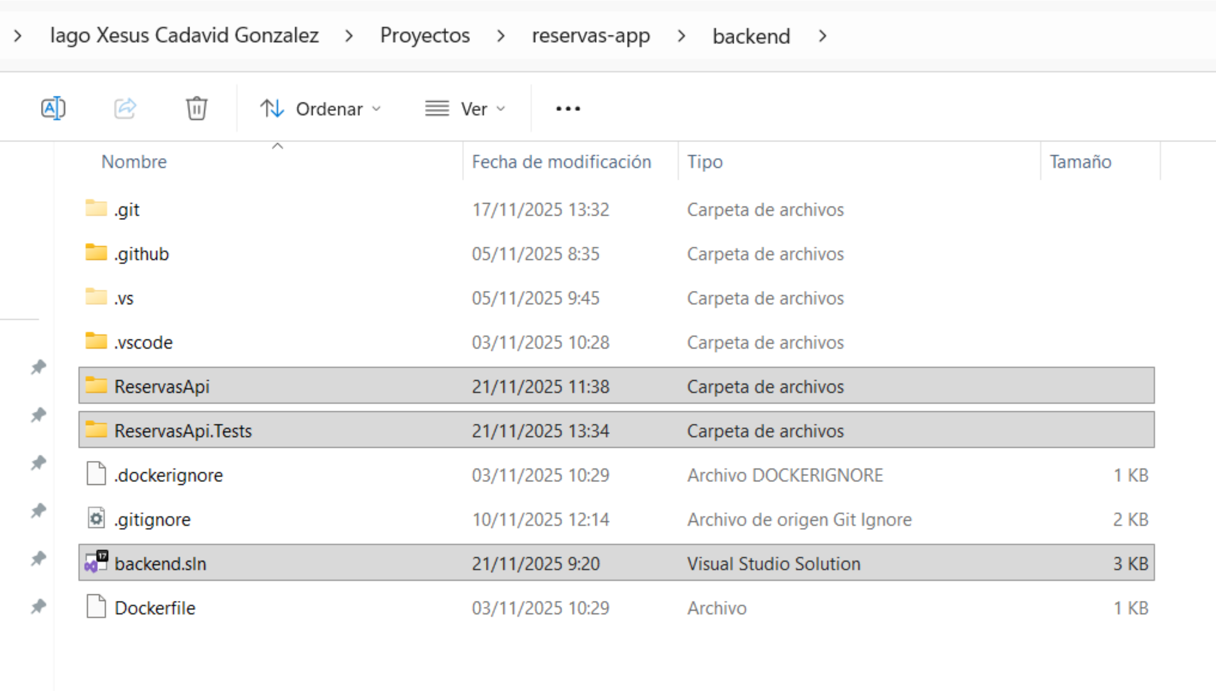
from **the solution root folder** — the folder that contains:

```
ReservasApi.sln
ReservasApi/
ReservasApi.Tests/
```

This is usually the *parent* folder of both your API project and your test project.

O sea que en mi caso corro dotnet test desde ‘backend’ folder:

NEXT PAGE


> Iago Xesus Cadavid Gonzalez > Proyectos > reservas-app > backend >
Ordenar
Ver
•••

| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|---|---|---|
| .git | 17/11/2025 13:32 | Carpeta de archivos | |
| .github | 05/11/2025 8:35 | Carpeta de archivos | |
| .vs | 05/11/2025 9:45 | Carpeta de archivos | |
| .vscode | 03/11/2025 10:28 | Carpeta de archivos | |
| ReservasApi | 21/11/2025 11:38 | Carpeta de archivos | |
| ReservasApi.Tests | 21/11/2025 13:34 | Carpeta de archivos | |
| .dockerignore | 03/11/2025 10:29 | Archivo DOCKERIGNORE | 1 KB |
| .gitignore | 10/11/2025 12:14 | Archivo de origen Git Ignore | 2 KB |
| backend.sln | 21/11/2025 9:20 | Visual Studio Solution | 3 KB |
| Dockerfile | 03/11/2025 10:29 | Archivo | 1 KB |

Before running dotnet test, shouldn't I first do some sort of build for the changes (the testing code we've written) to take effect ?

Great question — and **the answer is: NO, you do not need to build first.**

Here's why:

---

## ✅ `dotnet test` automatically builds your solution

When you run:

```bash
dotnet test
```

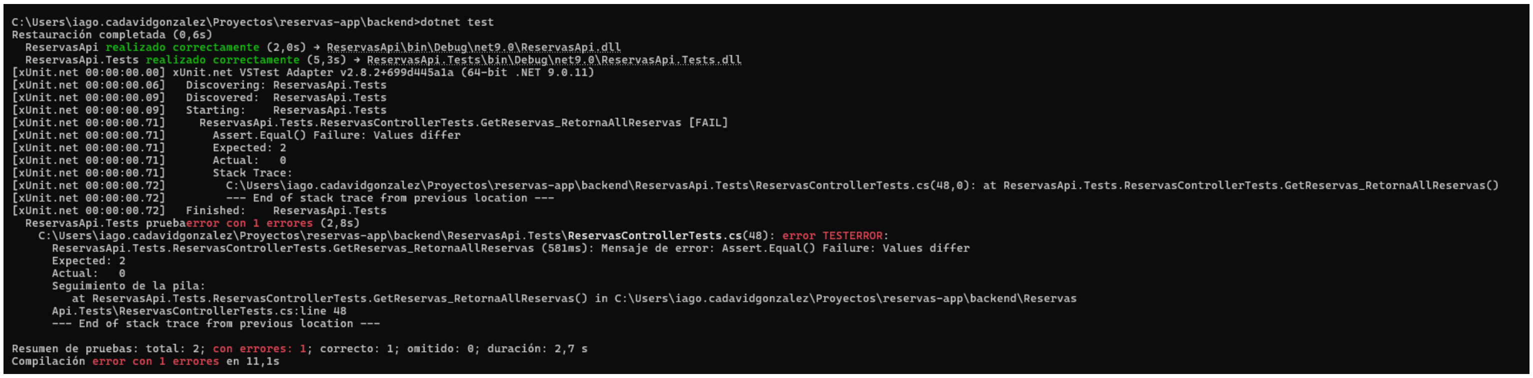
.NET automatically performs these steps **in order**:

1. **Restores packages** (if needed)
2. **Builds the solution/projects**
3. **Discovers the tests**
4. **Runs the tests**

So you do **not** need to run:

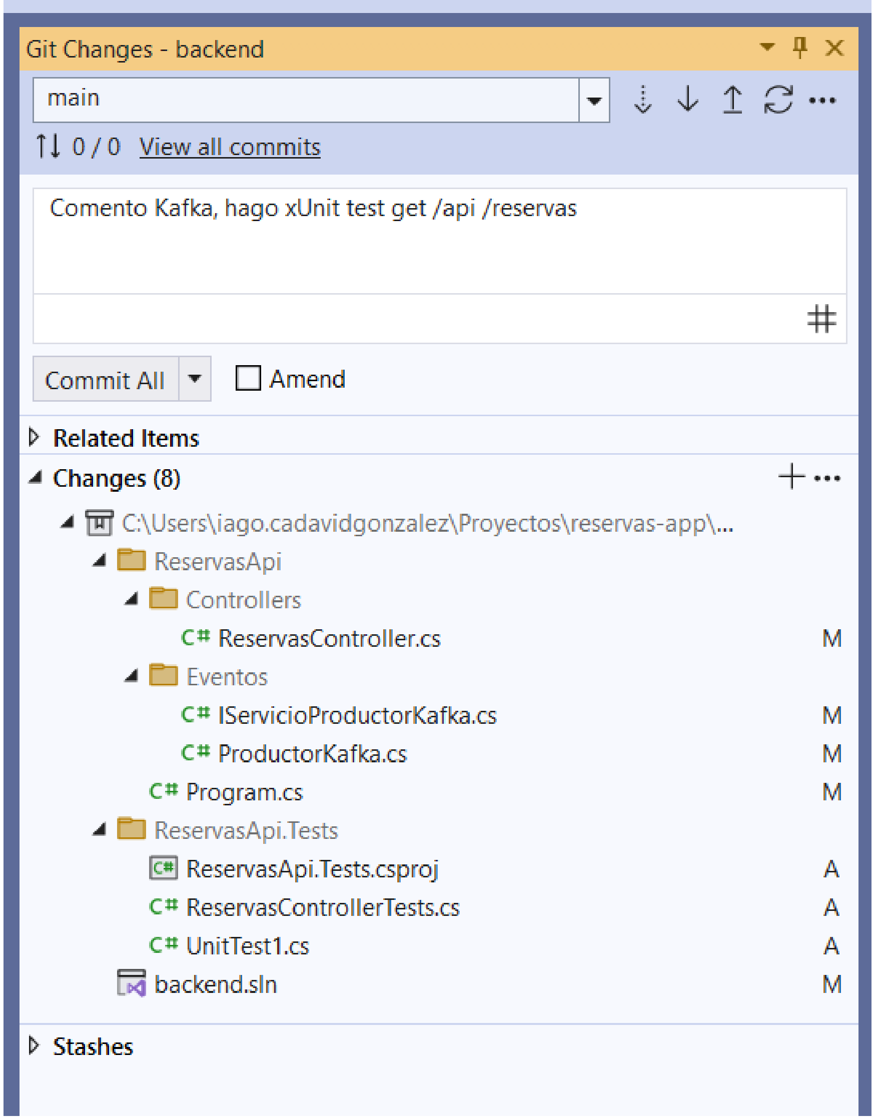
```nginx
dotnet build
```

before `dotnet test`.



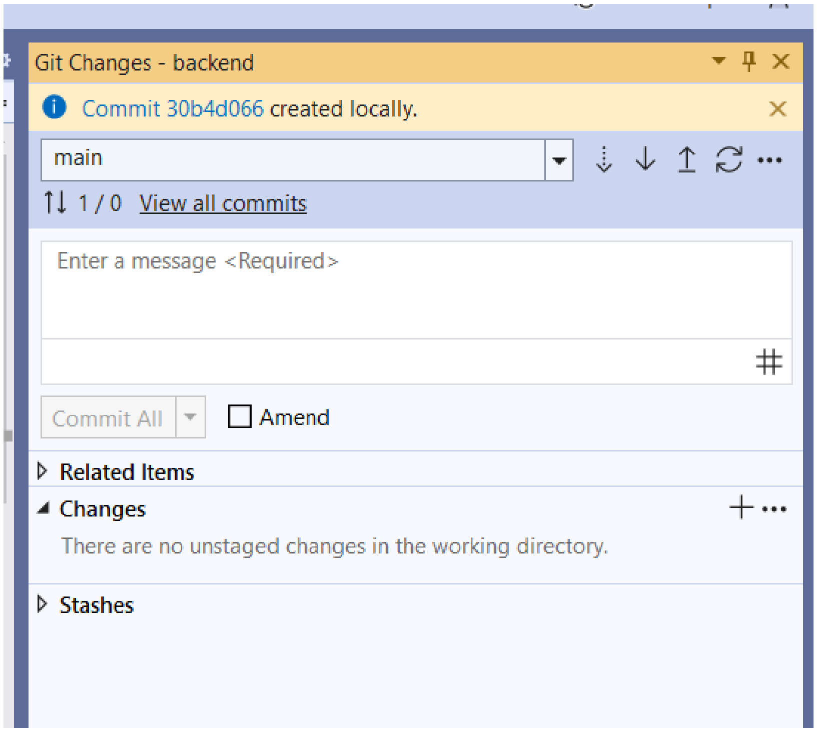
```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend>dotnet test
Restauración completada (0,6s)
  ReservasApi realizado correctamente (2,0s) → ReservasApi\bin\Debug\net9.0\ReservasApi.dll
  ReservasApi.Tests realizado correctamente (5,3s) → ReservasApi.Tests\bin\Debug\net9.0\ReservasApi.Tests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v2.8.2+699d445a1a (64-bit .NET 9.0.11)
[xUnit.net 00:00:00.06]   Discovering: ReservasApi.Tests
[xUnit.net 00:00:00.09]   Discovered:  ReservasApi.Tests
[xUnit.net 00:00:00.09]   Starting:    ReservasApi.Tests
[xUnit.net 00:00:00.71]     ReservasApi.Tests.ReservasControllerTests.GetReservas_RetornaAllReservas [FAIL]
[xUnit.net 00:00:00.71]       Assert.Equal() Failure: Values differ
[xUnit.net 00:00:00.71]       Expected: 2
[xUnit.net 00:00:00.71]       Actual:   0
[xUnit.net 00:00:00.71]       Stack Trace:
[xUnit.net 00:00:00.72]         C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasControllerTests.cs(48,0): at ReservasApi.Tests.ReservasControllerTests.GetReservas_RetornaAllReservas()
[xUnit.net 00:00:00.72]         --- End of stack trace from previous location ---
[xUnit.net 00:00:00.72]   Finished:    ReservasApi.Tests
  ReservasApi.Tests pruebaerror con 1 errores (2,8s)
    C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\ReservasApi.Tests\ReservasControllerTests.cs(48): error TESTERROR:
      ReservasApi.Tests.ReservasControllerTests.GetReservas_RetornaAllReservas (581ms): Mensaje de error: Assert.Equal() Failure: Values differ
      Expected: 2
      Actual:   0
      Seguimiento de la pila:
         at ReservasApi.Tests.ReservasControllerTests.GetReservas_RetornaAllReservas() in C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\backend\Reservas
      Api.Tests\ReservasControllerTests.cs:line 48
      --- End of stack trace from previous location ---

Resumen de pruebas: total: 2; con errores: 1; correcto: 1; omitido: 0; duración: 2,7 s
Compilación error con 1 errores en 11,1s
```

Chatgpt no dio con la tecla de por qué en “actual” sale 0 en vez de 2.

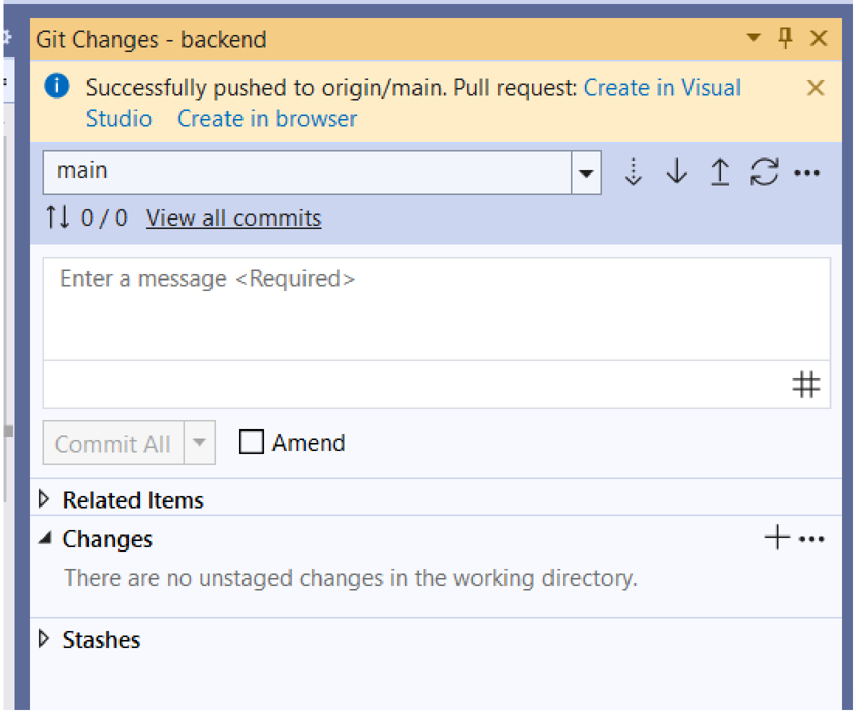


Comento Kafka, hago xUnit test get /api /reservas

```
C:\Users\iago.cadavidgonzalez\Proyectos\reservas-app\...
  ReservasApi
    Controllers
      ReservasController.cs            M
    Eventos
      IServicioProductorKafka.cs       M
      ProductorKafka.cs                M
    Program.cs                         M
  ReservasApi.Tests
    ReservasApi.Tests.csproj           A
    ReservasControllerTests.cs         A
    UnitTest1.cs                       A
  backend.sln                          M
```

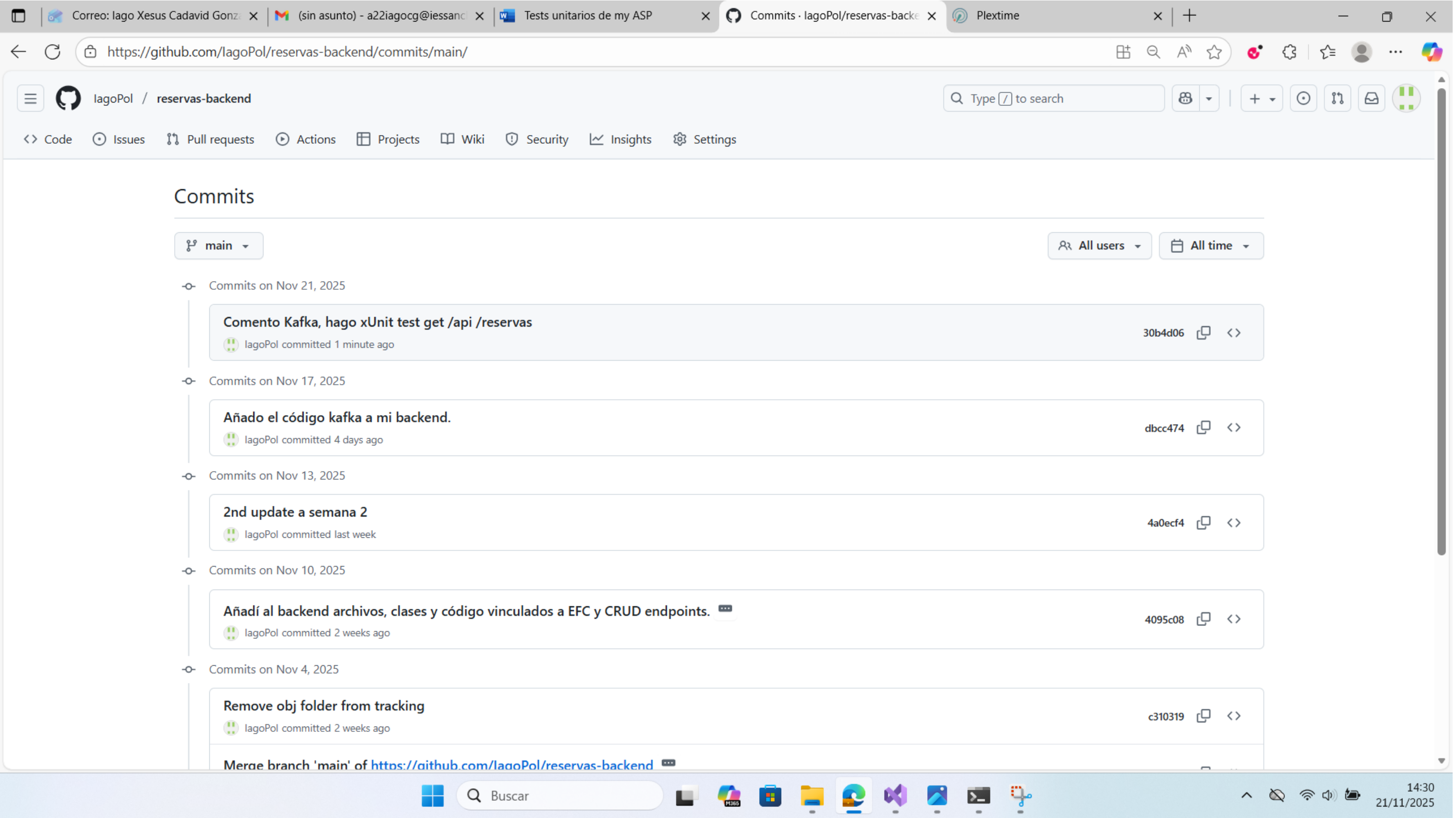


Commit 30b4d066 created locally.



Successfully pushed to origin/main. Pull request: Create in Visual Studio Create in browser

NEXT PAGE


Correo: Iago Xesus Cadavid Gonz
(sin asunto) - a22iagocg@iessanc
Tests unitarios de my ASP
Commits · IagoPol/reservas-backe
Plextime
https://github.com/IagoPol/reservas-backend/commits/main/
IagoPol / reservas-backend
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
All users
All time
Buscar
14:30
21/11/2025

## Commits

Commits on Nov 21, 2025

**Comento Kafka, hago xUnit test get /api /reservas**
IagoPol committed 1 minute ago — 30b4d06

Commits on Nov 17, 2025

**Añado el código kafka a mi backend.**
IagoPol committed 4 days ago — dbcc474

Commits on Nov 13, 2025

**2nd update a semana 2**
IagoPol committed last week — 4a0ecf4

Commits on Nov 10, 2025

**Añadí al backend archivos, clases y código vinculados a EFC y CRUD endpoints.**
IagoPol committed 2 weeks ago — 4095c08

Commits on Nov 4, 2025

**Remove obj folder from tracking**
IagoPol committed 2 weeks ago — c310319

**Merge branch 'main' of https://github.com/IagoPol/reservas-backend**