Contenido

[MOCKING DE CLASES E NTERFACES 2](#_Toc152367064)

[MOCKING CON SETEO DE VARIABLES 2](#_Toc152367065)

[MOCKING PARA METODO CON DEPENDENCIAS 3](#_Toc152367066)

[CQRS PARA UNIT TEST 3](#_Toc152367067)

[Comando, consulta, responsabilidad y segregación 3](#_Toc152367068)

[Librerias en Persistencia 4](#_Toc152367069)

[Librerías Aplicación 4](#_Toc152367070)

[PATRÓN DE DISEÑO MEDIATR 4](#_Toc152367071)

[AUTOMAPPER 5](#_Toc152367072)

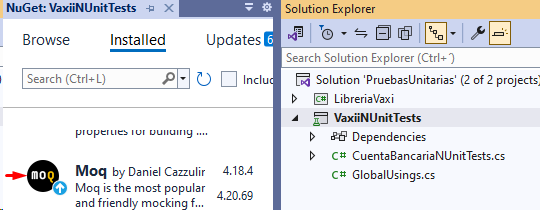
[CORS PARA REGISTRAR RECORD 6](#_Toc152367073)

[Consulta con parámetros CORS 6](#_Toc152367074)

[MIGRACION EF 6](#_Toc152367075)

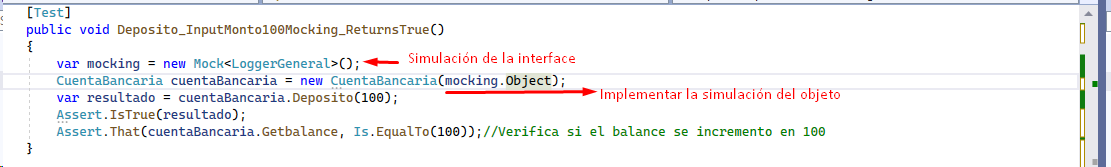
[CREACIÓN DE END-POINT Y CONTROLLER 7](#_Toc152367076)

[**UNIT TEST CQRS** 7](#_Toc152367077)



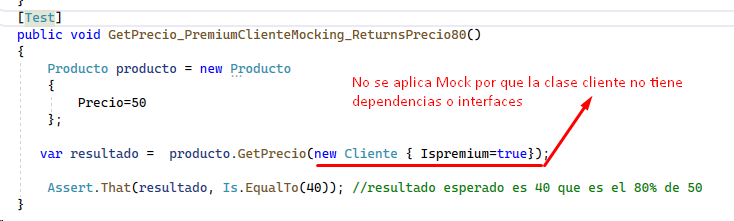
**MOQ** 🡺 Crea objetos simulados.

**NUnit** 🡺 es un Marco de prueba.

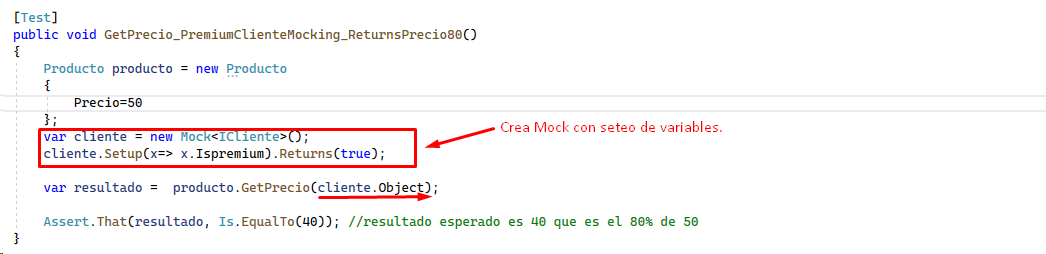


## MOCKING DE CLASES E NTERFACES

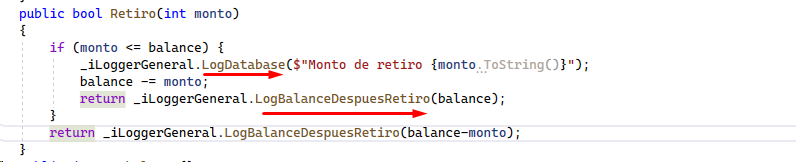
**Nota**: Cuando la clase no tiene dependencias de interfaces. Lo recomendable es usar directamente la clase para no abusar de la creación de **mocks**.

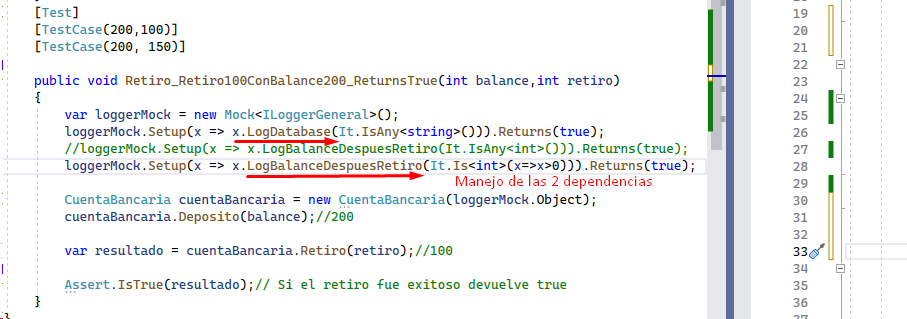


## MOCKING CON SETEO DE VARIABLES



## MOCKING PARA METODO CON DEPENDENCIAS





**Repositorio**

**FALSA DEPENDENCIA**  **Cuenta bancaria**

**LoggerGeneral**

# CQRS PARA UNIT TEST

## Comando, consulta, responsabilidad y segregación

(Command Query Responsibility Segregation). Es un patron de diseño arquitectónico que propone separar las operaciones de lectura (queries) de las operaciones de escritura (commands) en una aplicación. Esto ayuda a mejorar la escalabilidad, la mantenibilidad y la flexibilidad.

Fases lógicas de una prueba Automatizada.

**ARRANGE**

Configuración Inicial, Crear instancias de Objetos y datos de prueba.

**ACT**

Ejecutar código de la aplicación, métodos de llamada.

**ASSERT**

Verificar resultado, prueba pasa/ falla.



## Librerias en Persistencia

Microsoft.EntityFrameworkCore

Microsoft.EntityFrameworkCore.sqlserver

Microsoft.EntityFrameworkCore.Tools

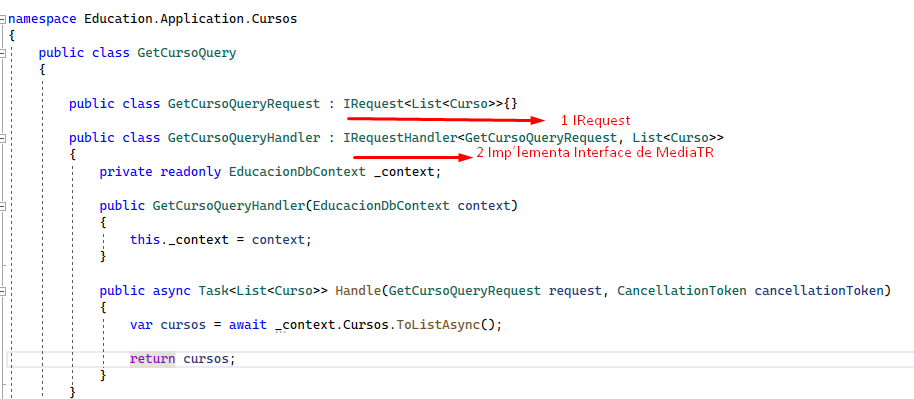
## Librerías Aplicación

MediatR.Extensions.Microsoft.DependencyInjection

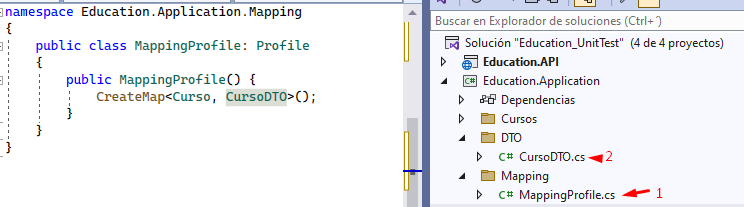
FluentValidation.AspNetCore

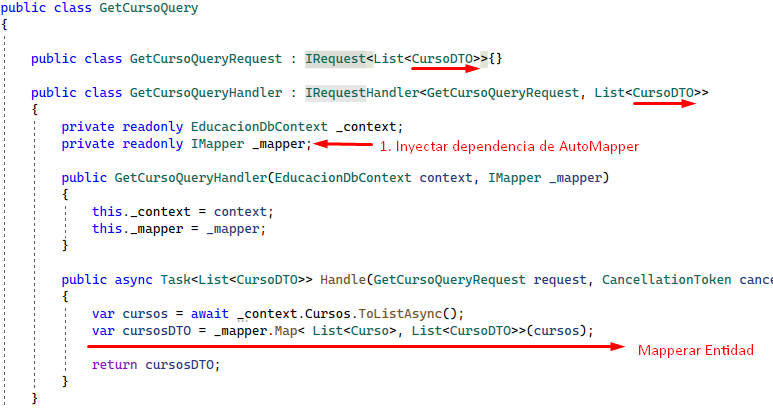
AutoMapper.Extensions.Microsoft.DependencyInjection

# PATRÓN DE DISEÑO MEDIATR

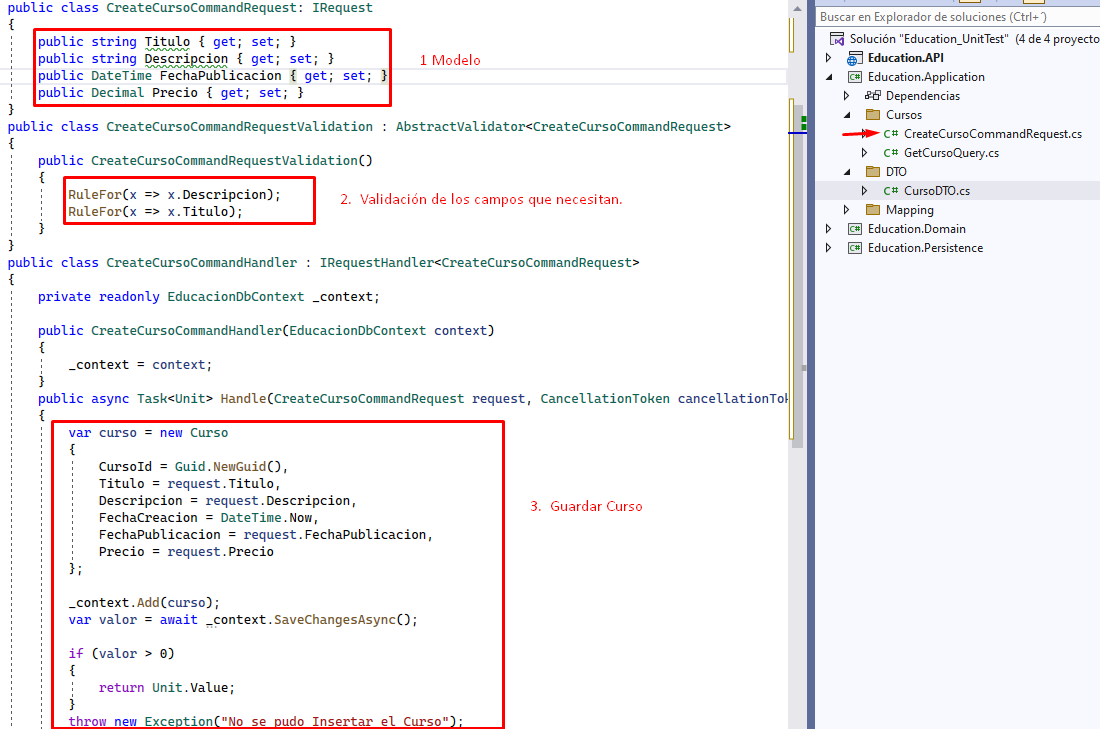


## AUTOMAPPER





## CORS PARA REGISTRAR RECORD



## Consulta con parámetros CORS



## MIGRACION EF

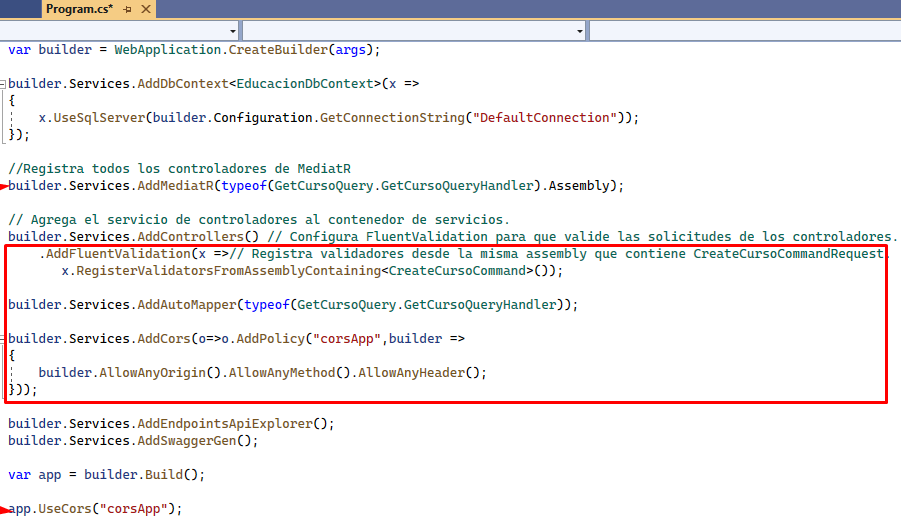


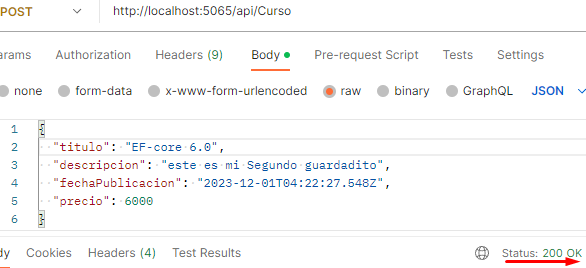
dotnet tool update --global dotnet-ef --version 6.0.0

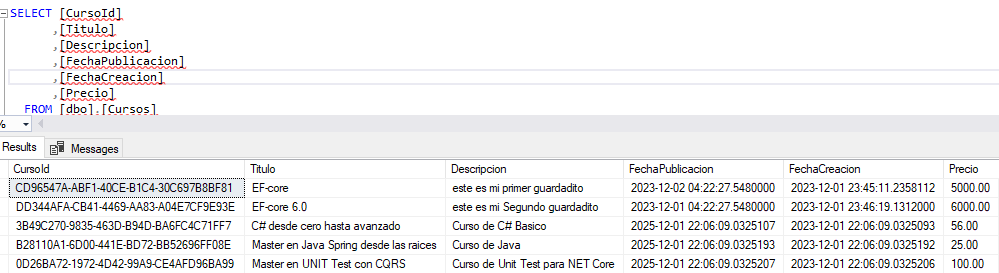
dotnet ef migrations add Inicial --project Education.Persistence

dotnet ef database update --project Education.Persistence

# CREACIÓN DE END-POINT Y CONTROLLER







# **UNIT TEST CQRS**

Código

Unit Test Código

Data Falsa

FUNCIONALIDAD

Data Real

EF

EF

Moq

Database

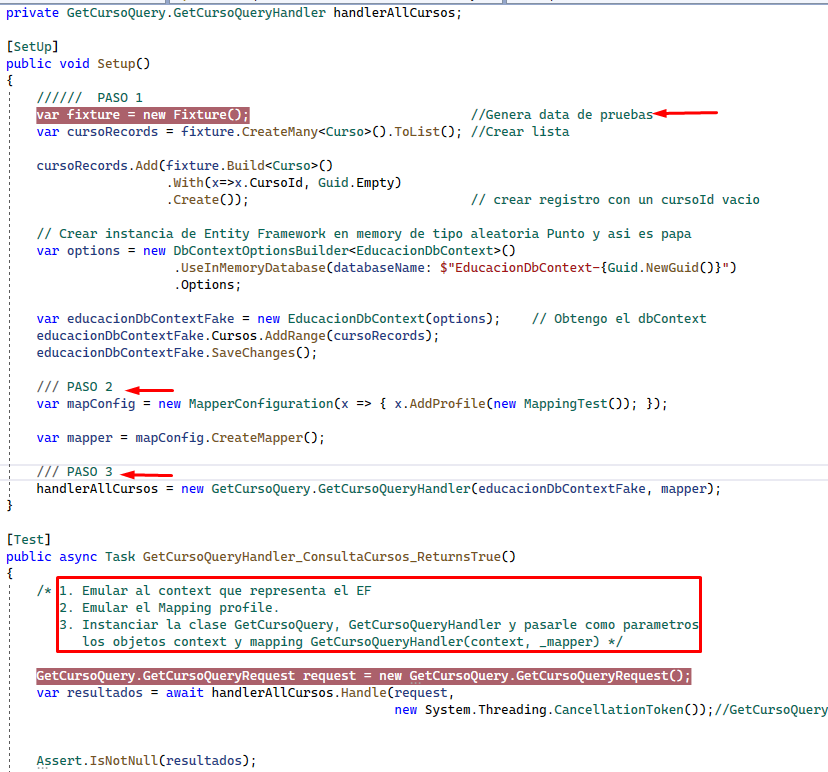
## Instalar 3 Paquetes

Microsoft.EntityFrameworkCore.InMemory

Moq

AutoFixture ==🡺 Genera Data de Prueba.

## Test Consulta

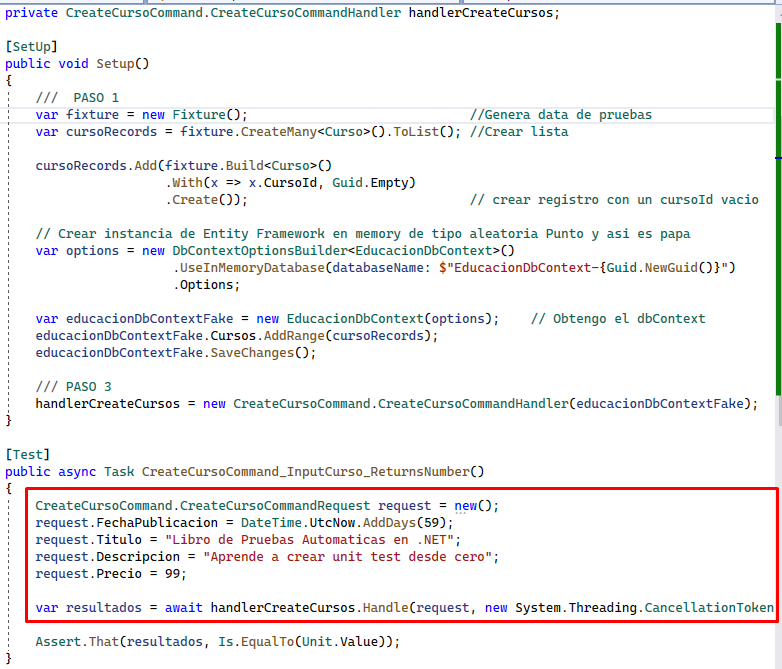


# **UNIT TEST CQRS CON PARAMETROS**

## Guardar registro

**PASO 1** Emular al contexto que representa el EF.

**PASO 2** Instanciar la clase CreateCursoCommand , CreateCursoCommandHandler y pasarle como parámetros el objeto context.



## Parámetros para consulta

