

Prova pratica

Realizzare un database per la Gestione degli Ordini e dei Clienti.

- **Cliente:**
ID (int, PK), CodiceCliente (string), Nome (string), Cognome (string)
- **Ordine**
ID (int, PK), DataOrdine (date), CodiceOrdine (string), CodiceProdotto (string), Importo (decimal), Cliente(?, FK)

Utilizzare EF Code-first per la realizzazione dello strato di accesso al dato.

Parte 1

Realizzare un servizio WCF per la gestione della parte dei **Clienti**, che esponga i metodi per:

- Elencare i Clienti
- Creare un Cliente
- Modificare un Cliente
- Cancellare un Cliente

Parte 2

Realizzare un servizio REST per la gestione della parte di **Anagrafica Ordini**, che esponga i metodi per:

- Elencare gli Ordini
- Creare un Ordine
- Modificare un Ordine
- Cancellare un Ordine

Scheletro del progetto:

1. Progetto **Core** → comune

Class library → .NET Standard 2.0

- Models:

→ Customer: DataContract

→ Order

- Interfaces → IRepository, IOrderRepository, ICustomerRepository

- BusinessLayer → IMainBL, MainBL (costruttore a cui passo IOrderRepository e ICustomerRepository)

2. Progetto **EF** → comune

Class library → .NET Standard 2.0

Pacchetti:

- EFCore

- EFCore.Tools

- EFCore.SqlServer

versione: **3.1.13**

Project references:

- Core

- ...Context : DbContext

→ costruttore vuoto

→ costruttore con opzioni

```
public MyContext() : base() {}  
  
public MyContext(DbContextOptions<OrderContext> options)  
    : base(options) {}
```

3. Progetto WCF

4. Progetto API

Riepilogo per implementare Parte 1 :

Creare progetto **WCF** Service Library

Target .NET Framework 4.8

Project references:

- Core

- EF

Pacchetti:

- EFCore.Design

- EFCore.SqlServer

versione : **3.1.13**

IService1 → Service Contract. Contiene le operazioni indicate sopra (Elencare clienti, aggiungere cliente...)

Service1 → Classe di implementazione del servizio → si implementano le operazioni esposte nel service contract, in particolare saranno dei metodi che chiamano dei metodi del business layer.

N.B. Nel costruttore, bisogna creare l'istanza del business layer e del repository (perché non c'è dependency injection)

```
private readonly MainBL mainBusinessLayer;  
  
public CustomerService()  
{  
    mainBusinessLayer = new MainBL( new EFOOrderRepository(), new EFCustomerRepository()  
    );  
}
```

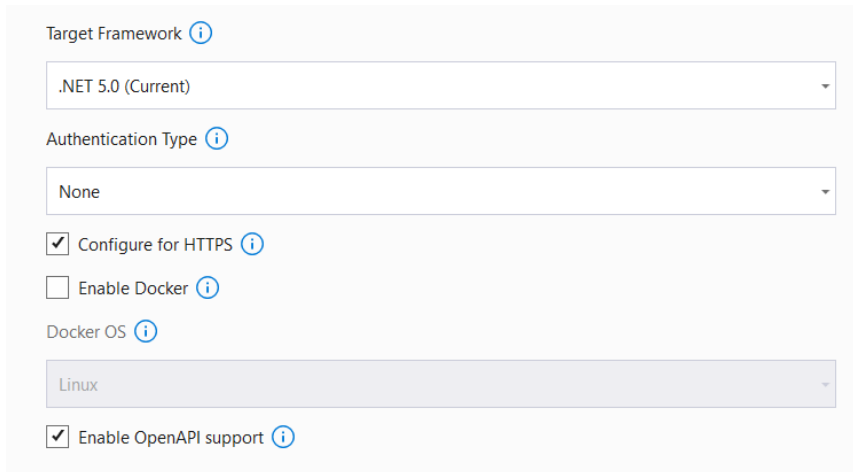
Se si rinominano IService1 e Service1, modificare in app.config nel baseAddress.

➔ Testare con WCF Test Client

Riepilogo per implementare parte 2

Creare progetto ASP .NET Core Web API

Target Framework .NET 5.0



The screenshot shows the 'Target Framework' settings in Visual Studio. The 'Target Framework' dropdown is set to '.NET 5.0 (Current)'. The 'Authentication Type' dropdown is set to 'None'. The 'Configure for HTTPS' checkbox is checked. The 'Enable Docker' checkbox is unchecked. The 'Docker OS' dropdown is set to 'Linux'. The 'Enable OpenAPI support' checkbox is checked.

References:

- Core
- EF

1. Creare Controller per Ordini (API Controller – Empty)
2. Inserire Costruttore con parametri:

```
private readonly IMainBL mainBusinessLayer;  
  
public OrderController(IOrderBL mainBusinessLayer)  
{  
    this.mainBusinessLayer = mainBusinessLayer;  
}
```

3. Implementare Action.
4. Configurare la dependency injection in Configure Services nella classe Startup

➔ Testare con Postman o Swagger