# Entity Framework – Prática

Documentação: https://learn.microsoft.com/en-us/ef/core/

## 1. Configuração do Ambiente

### 1. Instalação do SDK do .NET e CLI do Entity Framework

#### Certifique-se de ter o .NET SDK instalado e qual versão.

No terminal, verifique com:

```
dotnet --version
```

#### Instale a ferramenta de linha de comando (CLI) do Entity Framework:

```
dotnet tool install --global dotnet-ef
dotnet ef --version
```

# 2. Criando o Projeto

## 1. Criação de um Projeto Console no .NET

Inicie um novo projeto console para o curso:

```
dotnet new console -o MeuProjetoEF
cd MeuProjetoEF
```

## 2. Configuração do Projeto para Entity Framework

- Adicionar o pacote Microsoft.EntityFrameworkCore:

```
dotnet add package Microsoft.EntityFrameworkCore
```

Adiciona o pacote principal do Entity Framework Core ao seu projeto.

- Adicionar o pacote Microsoft.EntityFrameworkCore.Design:

```
dotnet add package Microsoft.EntityFrameworkCore.Design
```

Adiciona o pacote EntityFrameworkCore.Design ao projeto, o qual fornece ferramentas adicionais para desenvolvimento. Esse pacote contém funcionalidades específicas para projeto e desenvolvimento, incluindo suporte para o CLI (dotnet ef) ao gerar e gerenciar migrações do banco de dados.

- Adicionar pacotes necessários ao Entity Framework Core relativos ao seu provider.

(https://learn.microsoft.com/en-us/ef/core/providers)

```
dotnet add package  Microsoft.EntityFrameworkCore.SqlServer
```

### 3. Verificar Dependências no Arquivo de Projeto (.csproj)

Confirme que as versões do EntityFrameworkCore, SQLServer e Design estão corretas e configuradas no arquivo .csproj. Certifique-se de que o Visual Studio Code reconheça essas dependências ao compilar o projeto.

# 3. Configuração do DbContext e Modelo de Dados

## 1. Criar a Classe de Modelo Produto

No arquivo Produto.cs, defina a classe de entidade:

```
namespace MeuProjetoConsole.Models;

public class Produto
{
    public int Id { get; set; }
    public string Nome { get; set; }
}
```

## 2. Criar a Classe ProdutoContext com DbContext

No arquivo ProdutoContext.cs, defina o DbContext para configurar o provider do banco de dados:

```
using Microsoft.EntityFrameworkCore;

public class ProdutoContext : DbContext
{
   public DbSet<Produto> Produtos { get; set; }

   protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
   {
      optionsBuilder.UseInMemoryDatabase("TarefasDB");
   }
}
```

4. Implementando Operações CRUD

```
using Microsoft.EntityFrameworkCore;

namespace MeuProjetoConsole.Models;

public class AppDbContext : DbContext
{
    public DbSet<Produto> Produtos { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlServer(@"Server=192.168.0.120;Database=MeuProjetoConsole;User
Id=SA;Password=DellITAcademyef@;Encrypt=False;");
        optionsBuilder.EnableSensitiveDataLogging().LogTo(Console.WriteLine);
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        base.OnModelCreating(modelBuilder);
```

```
        // Mapeamento de Produto
        modelBuilder.Entity<Produto>()
            .ToTable("Produtos")
            .HasKey(p => p.Id);

        modelBuilder.Entity<Produto>()
            .Property(p => p.Nome)
            .HasMaxLength(100);

    }
}
```

Program.cs

```
using MeuProjetoConsole.Models;
using System.Data.SqlClient;
using Microsoft.Data.SqlClient;
using System;

class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Iniciando conexão com BD...");
      using(var contexto = new AppDbContext())
      {
         Console.WriteLine("Inserindo dados");
         contexto.Produtos.Add(new Produto{ Id = 100, Nome = "Prego" });
         contexto.Produtos.Add(new Produto{ Id = 200, Nome = "Parafuso"});
         contexto.SaveChanges();
         }
   }
}
```