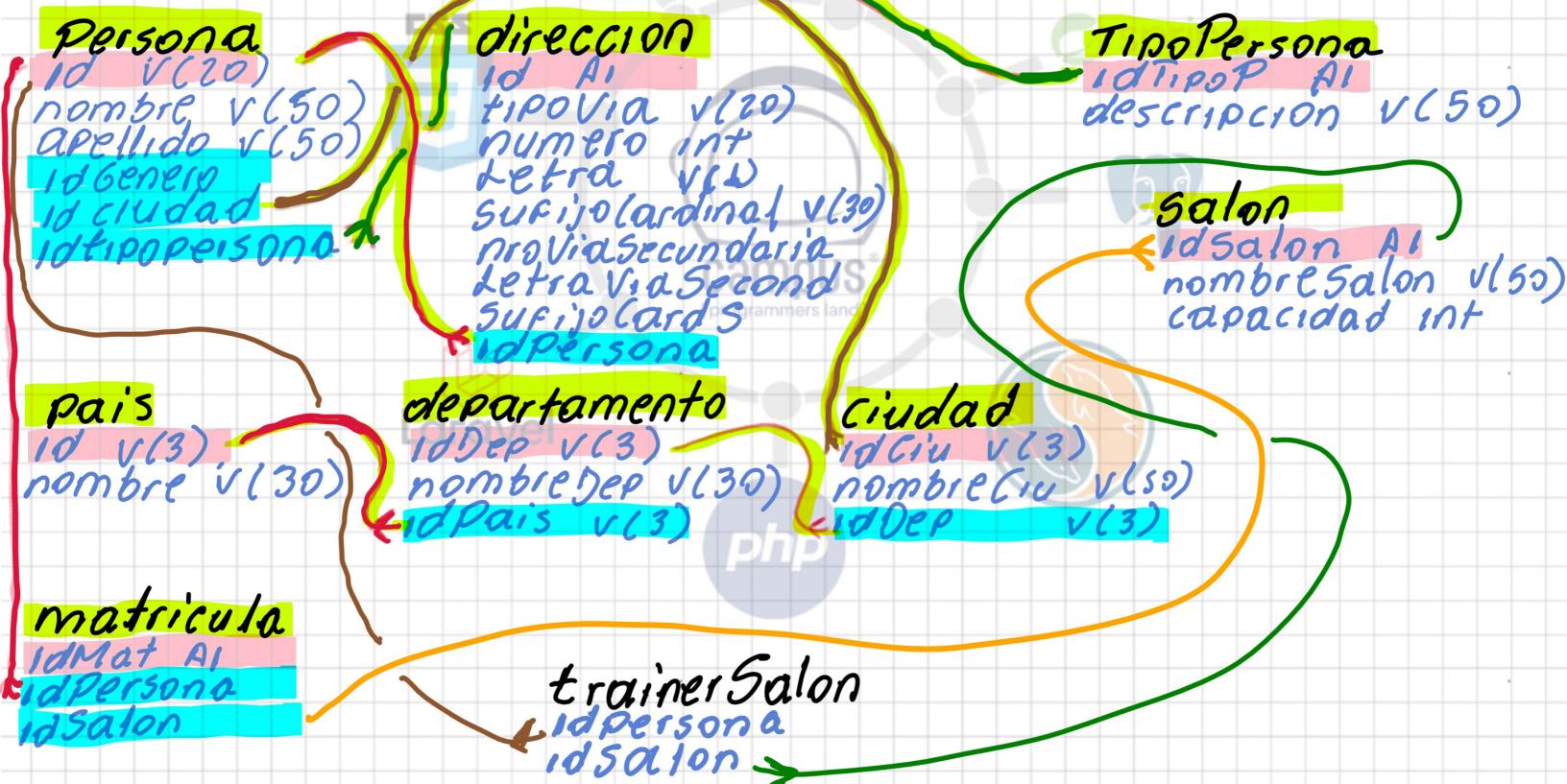


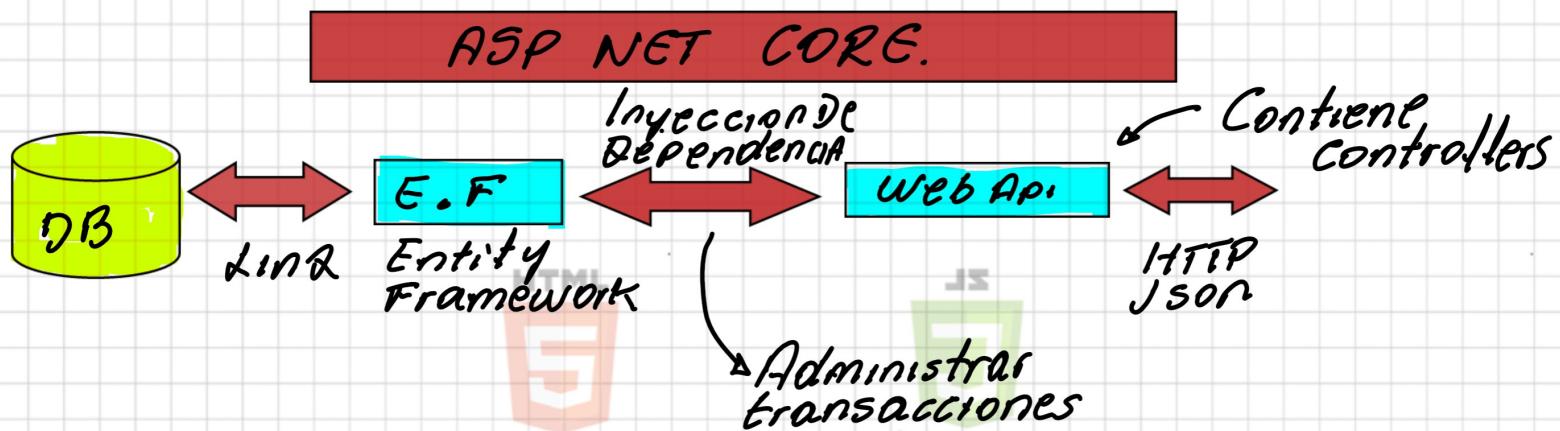
En Esta Guía Práctica Aprenderá A Crear Un Proyecto WebApi Usando .NetCore 7.0 y MySQL Como Gestor De Bases De Datos.

Objetivo: Crear Un WebApi Que Permita Administrar las Peticiones De Las Aplicaciones Front y Móvil de Un Gestor De Incidencias tecnológicas y Administrativas De Una Organización

Base De Datos



Arquitectura WebApi



Estructura Proyecto:

- Dominio → Se Crean Tablas Que Representan La BD.
- Persistencia → Instancia Conex DB
- Aplicación → Se Crea La Inyección De Dependencia Para La Comunicación Con El WebAPI.
- WebAPI → Se Creen Clases Encargadas De Recibir Peticiones De Los Clientes

Creación Proyecto

- 1) Genere Un Repositorio Nuevo Con El Proyecto.
- 2) Clone El Repositorio.

```
PS D:\NetCore> git clone https://github.com/trainingLeader/incidencias-app.git
```

- 3) Genere La SOLUCION PRINCIPAL.

```
PS D:\NetCore\incidencias-app> dotnet new sln ←  
La plantilla "Archivo de la solución" se creó correctamente.
```

- 4) Crear El Proyecto Dominio

```
PS D:\NetCore\incidencias-app> dotnet new classlib -o Dominio  
La plantilla "Biblioteca de clases" se creó correctamente.
```

- 5) Crear Proyecto Persistencia

```
PS D:\NetCore\incidencias-app> dotnet new classlib -o Persistencia ←  
La plantilla "Biblioteca de clases" se creó correctamente.
```

6) Crear Proyecto De Aplicación y WebAPI:

```
PS D:\NetCore\incidencias-app> dotnet new classlib -o Aplicacion  
La plantilla "Biblioteca de clases" se creó correctamente.
```

```
PS D:\NetCore\incidencias-app> dotnet new webapi -o ApiIncidencias  
La plantilla "ASP.NET Core Web API" se creó correctamente.
```

```
PS D:\NetCore\incidencias-app> dir
```

Directorio: D:\NetCore\incidencias-app

Mode	LastWriteTime	Length	Name
d----	12/08/2023 5:58 p. m.		
d----	12/08/2023 5:58 p. m.		ApiIncidencias
d----	12/08/2023 5:57 p. m.		Aplicacion
d----	12/08/2023 5:57 p. m.		Dominio
-a---	12/08/2023 5:53 p. m.	441	Persistencia
-a---	12/08/2023 5:52 p. m.	17	incidencias-app.sln
			README.md

7) Agregar los Proyectos A la Solución Principales

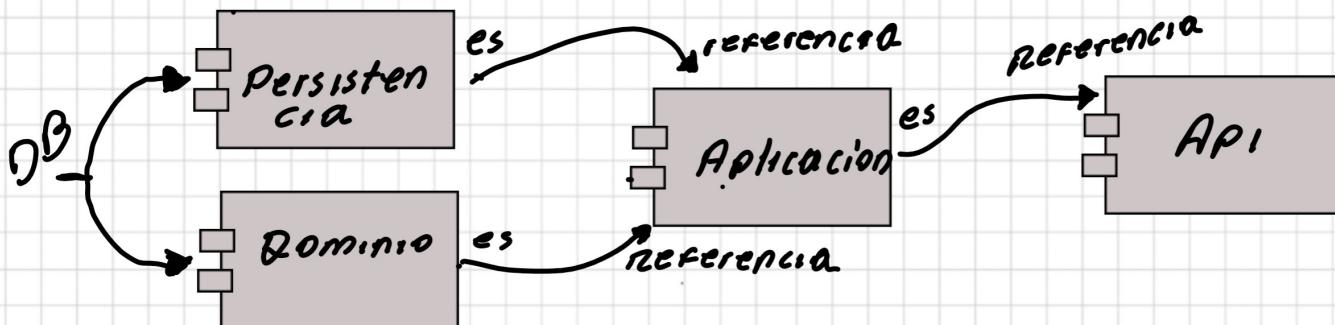
Comando Para Visualizar los Proyectos ASOCIADOS a la SOLUCION PRINCIPAL.

Comando Usado Para

```
PS D:\NetCore\incidencias-app> dotnet sln list  
No se han encontrado proyectos en la solución.  
PS D:\NetCore\incidencias-app> dotnet sln add .\Dominio\  
Se ha agregado el proyecto "Dominio\Dominio.csproj" a la solución.  
PS D:\NetCore\incidencias-app> dotnet sln add .\Persistencia\  
Se ha agregado el proyecto "Persistencia\Persistencia.csproj" a la solución.  
PS D:\NetCore\incidencias-app> dotnet sln add .\ApiIncidencias\  
Se ha agregado el proyecto "ApiIncidencias\ApiIncidencias.csproj" a la solución.  
PS D:\NetCore\incidencias-app> dir
```

agregar UN PROYECTO A LA SOLUCION.

8) Establecer Referencia Entre Proyectos



```

PS D:\NetCore\incidencias-app> cd ..\Aplicacion\
PS D:\NetCore\incidencias-app\Aplicacion> dotnet add reference ..\Dominio\
Se ha agregado la referencia "..\Dominio\Dominio.csproj" al proyecto.
PS D:\NetCore\incidencias-app\Aplicacion> dotnet add reference ..\Persistencia\
Se ha agregado la referencia "..\Persistencia\Persistencia.csproj" al proyecto.
PS D:\NetCore\incidencias-app\Aplicacion> cd ..
PS D:\NetCore\incidencias-app> cd ..\ApiIncidencias\
PS D:\NetCore\incidencias-app\ApiIncidencias> dotnet add reference ..\Aplicacion\
Se ha agregado la referencia "..\Aplicacion\Aplicacion.csproj" al proyecto.
PS D:\NetCore\incidencias-app\ApiIncidencias> cd ..
PS D:\NetCore\incidencias-app> cd persistencia
PS D:\NetCore\incidencias-app\persistencia> dotnet add reference ..\Dominio\
Se ha agregado la referencia "..\Dominio\Dominio.csproj" al proyecto.
PS D:\NetCore\incidencias-app\persistencia> cd..
PS D:\NetCore\incidencias-app> code .
PS D:\NetCore\incidencias-app>

```

Instalación De Paquetes → La instalación de paquetes se puede realizar de 2 formas; la primera desde la consola y la segunda desde el Gestor de Paquetes Nugets.

The screenshot shows the NuGet.org website interface. The URL in the address bar is `nuget.org/packages?q=EntityFrameworkCore&frameworks=&tfms=&packageType=&prerelease=true&sortby=relevance`. The main search query is "EntityFrameworkCore". On the left, there are filters for "Frameworks" (checkboxes for .NET, .NET Core, .NET Standard, .NET Framework) and "Package type" (radio buttons for All types, Dependency, .NET tool). The results panel displays "4,137 packages returned for EntityFrameworkCore". The top result is "Microsoft.EntityFrameworkCore" by aspnet EntityFramework Microsoft, with 767,940,155 total downloads, last updated 18 days ago, and the latest version being 8.0.0-preview.6.23329.4. A brief description states: "Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database, SQLite, Azure... More information".

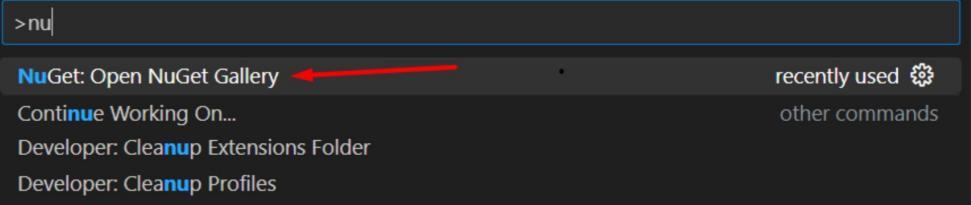
Seleccionar el Enlace y Dar Clic. En El Listado De Versiones
 Seleccionar Ver. 7.0.9. Copiar El Comando Sugerido.
 y Pegarlo El la terminal **se debe ejecutar El Comando**
desde la carpeta Dominio

```

desarrollo@DESKTOP-L4V647N MINGW64 /d/projectsNetCore/dinoShop-app/Dominio (main)
$ dotnet add package Microsoft.EntityFrameworkCore --version 7.0.9

```

Forma 2: Abrir Gestor De Paquetes Nuget. En Visual Studio code. (Ctrl + P)



Dominio.csproj X NuGet Gallery X

Fluent Filter Prerelease nuget.org

FluentValidation.DependencyInjectionExtensions by Jeremy Skinner
Dependency injection extensions for FluentValidation

FluentValidation.AspNetCore by Jeremy Skinner
AspNetCore integration for FluentValidation

FluentMigrator by Sean Chambers, Josh Coffman, Tom Marien, Mark Junker
FluentMigrator is a database migration framework for .NET written in C#. The basic idea is that you can create migrations which are simply classes that derive from the

FluentValidation.AspNetCore by Jeremy Skinner
v11.3.0 Refresh Install
Aplicacion.csproj Install
DinoApi.csproj Install
Domino.csproj **Install** Persistencia.csproj Install
11.3.0 Install Uninstall

Dominio.csproj X NuGet Gallery X

iText Filter Prerelease nuget.org

iText7.commons by Apryse Software
Commons module

iText7.pdfhtml by Apryse Software
pdfHTML is an iText 7 add-on that lets you to parse (X)HTML snippets and the associated CSS and converts them to PDF.

iText7.licensekey by iText Software
The iText 7 license key library enables the use of iText 7 (and iTextSharp version 5.5.13 and newer) in non-AGPL mode, and also provides access to the iText 7 add-

iText7.pdfhtml by Apryse Software
v5.0.0 Refresh Install
Aplicacion.csproj Install
Domino.csproj **Install** Persistencia.csproj Install
v3.1.5 5.0.0 Install Uninstall

Dominio.csproj X NuGet Gallery X NuGet Gallery X NuGet Gallery X

entity Filter Prerelease nuget.org

.NET Microsoft.EntityFrameworkCore by Microsoft
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with

.NET Microsoft.EntityFrameworkCore.Relational by Microsoft
Shared Entity Framework Core components for relational database providers.

.NET Microsoft.EntityFrameworkCore.Abstractions by Microsoft
Provides abstractions and attributes that are used to configure Entity Framework Core

Microsoft.EntityFrameworkCore by Microsoft
v7.0.9 Refresh Install
Aplicacion.csproj Install
DinoApi.csproj Install
Domino.csproj **7.0.9 Uninstall** Persistencia.csproj Install
7.0.9 Install Uninstall

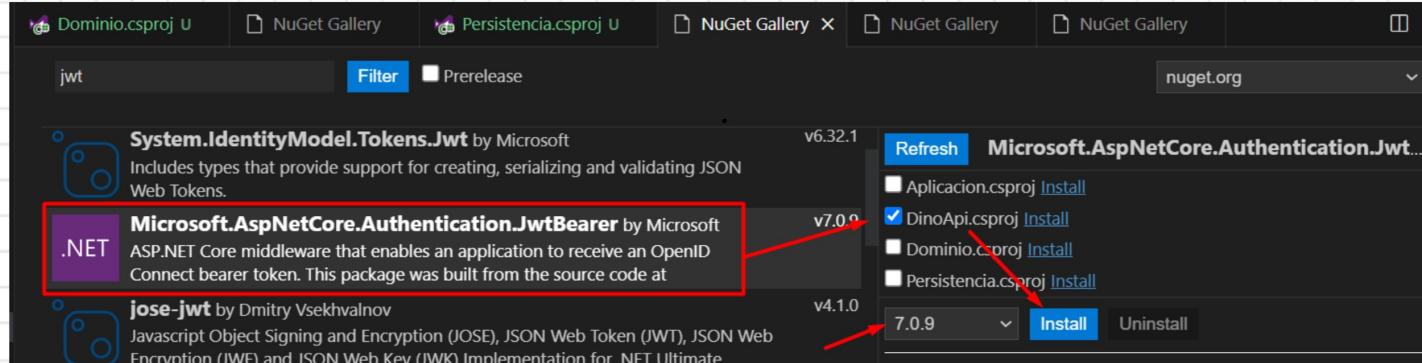
mysql Filter Prerelease nuget.org

Pomelo.EntityFrameworkCore.MySql by Laurents Meyer, Caleb Lloyd
Pomelo's MySQL database provider for Entity Framework Core.

MySQLConnector by Bradley Grainger
A truly async MySQL ADO.NET provider, supporting MySQL Server, MariaDB, Percona Server, Amazon Aurora, Azure Database for MySQL and more.

MySQL.Data.EntityFrameworkCore by Oracle
MySQL.Data.EntityFrameworkCore for Entity Framework.

Pomelo.EntityFrameworkCore.MySql by Laurents Meyer, Caleb Lloyd
v7.0.0 Refresh Install
Aplicacion.csproj Install
DinoApi.csproj Install
Domino.csproj **Install** Persistencia.csproj Install
7.0.0 Install Uninstall



Conexión Base De Datos

→ Configuración de la Cadena De Conexión.

{ } appsettings.Development.json	M
{ } appsettings.json	M

```

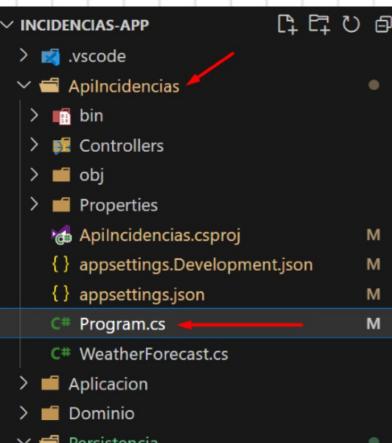
1 {
2   "ConnectionStrings": {
3     "ConexSqlServer": "Data Source=localhost\\sqlexpress;Initial Catalog=dB;Integrate Security=True",
4     "ConexMysql": "server=localhost;user=root;password=123456;database=incidenciasdb"
5   },
6   "Logging": {
7     "LogLevel": {
8       "Default": "Information",
9       "Microsoft.AspNetCore": "Warning"
10    }
11  },
12  "AllowedHosts": "*"
13 }
```

2) Creación Del **DBContext** → El DBcontext Es Una Capa De Abstracción Que Permitirá La Interacción Con La Base De Datos; Lo Que Facilitará La Creación De Un **(CRUD)**

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Microsoft.EntityFrameworkCore; ←
6
7  namespace Persistencia
8  {
9    2 references
10   public class ApiIncidenciasContext : DbContext
11   {
12     0 references
13     public ApiIncidenciasContext(DbContextOptions<ApiIncidenciasContext> options) : base(options)
14     {
15     }
16   }
}
```

3) Inyectar el DBContext (Proyecto WebApi > Program.cs)



```
INCIDENCIAS-APP
  - .vscode
  - Aplicacion
  - Dominio
  - Persistencia
    - ApiIncidencias (highlighted)
      - Program.cs (highlighted)
      - WeatherForecast.cs
```

```
1 | using Microsoft.EntityFrameworkCore;
2 | using Persistencia;
3 |
4 | var builder = WebApplication.CreateBuilder(args);
5 |
6 | // Add services to the container.
7 |
8 | builder.Services.AddControllers();
9 | // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
10| builder.Services.AddEndpointsApiExplorer();
11| builder.Services.AddSwaggerGen();
12|
13| builder.Services.AddDbContext<ApiIncidenciasContext>(options =>
14| {
15|   string connectionString = builder.Configuration.GetConnectionString("ConexMysql");
16|   options.UseMySql(connectionString, ServerVersion.AutoDetect(connectionString));
17| });
18|
19| var app = builder.Build();
```

Creación Entidades (Entities) → Las Entidades se deben crear en el proyecto Dominio.

```
1 namespace Dominio;
2
3 public class Persona
4 {
5   public string IdTipoPersona { get; set; }
6   public string NombrePersona { get; set; }
7   public int IdGeneroFk { get; set; }
8   public int IdCiudadFk { get; set; }
9   public int IdTipoPerFk { get; set; }
10}
```

```
1 namespace Dominio;
2
3 public class Matricula
4 {
5   public int IdMatricula { get; set; }
6   public string IdPersonaFk { get; set; }
7   public int IdSalonFk { get; set; }
8 }
```

```
1 namespace Dominio;
2
3 public class Ciudad
4 {
5   public string IdCiudad { get; set; }
6   public string NombreCiudad { get; set; }
7   public string IdDepFk { get; set; }
8 }
```

```
1 namespace Dominio;
2
3 public class Genero
4 {
5   public int IdGeneroFk { get; set; }
6   public string NombreGenero { get; set; }
7 }
```

```
1 namespace Dominio;
2
3 public class Pais
4 {
5   public string IdPais { get; set; }
6   public string NombrePais { get; set; }
7 }
```

```

1 namespace Dominio;
2
3 0 references
4 public class Departamento
5 {
6     0 references
7     public string IdDep { get; set; }
8     0 references
9     public string NombreDep { get; set; }
10    0 references
11    public string IdPaisFk { get; set; }
12 }

```

```

1 namespace Dominio;
2
3 0 references
4 public class TipoPersona
5 {
6     0 references
7     public int IdTipoPersona { get; set; }
8     0 references
9     public string DescripcionTipoPersona { get; set; }
10 }

```

```

1 namespace Dominio;
2
3 0 references
4 public class Salon
5 {
6     0 references
7     public int IdSalon { get; set; }
8     0 references
9     public string NombreSalon { get; set; }
10    0 references
11    public int Capacidad { get; set; }
12 }

```

```

1 namespace Dominio;
2
3 0 references
4 public class TrainerSalon
5 {
6     0 references
7     public string IdPerTrainerFk { get; set; }
8     0 references
9     public int IdSalonFk { get; set; }
10 }

```

Configuración DbSet → En el EF core los DbSet representan una colección de Entidades en una Base de Datos. Los DbSet se configuran en el Archivo De Contexto.

```

1 using Dominio;
2 using Microsoft.EntityFrameworkCore;
3 namespace Persistencia
4 [
5     3 references
6     public class ApiIncidenciasContext : DbContext
7     {
8         0 references
9         public ApiIncidenciasContext(DbContextOptions<ApiIncidenciasContext> options) : base(options)
10        0 references
11        public DbSet<Ciudad> Ciudades { get; set; }
12        0 references
13        public DbSet<Persona> Personas { get; set; }
14        0 references
15        public DbSet<Salon> Salones { get; set; }
16        0 references
17        public DbSet<Matricula> Matriculas { get; set; }
18        0 references
19        public DbSet<TipoPersona> TipoPersonas { get; set; }
20        0 references
21        public DbSet<TrainerSalon> TrainerSalones { get; set; }
22        0 references
23        public DbSet<Departamento> Departamentos { get; set; }
24        0 references
25        public DbSet<Pais> Paises { get; set; }
26        0 references
27        public DbSet<Genero> Generos { get; set; }
28    }

```

Definición Cardinalida Entre Entidades

```
1 namespace Dominio;
2
3     8 references
4     public class Persona
5     {
6         2 references
7         public string IdPersona { get; set; }
8         1 reference
9         public string NombrePersona { get; set; }
10        1 reference
11        public int IdGeneroFk { get; set; }
12        1 reference
13        public Genero Genero { get; set; }
14        1 reference
15        public int IdCiudadFk { get; set; }
16        1 reference
17        public Ciudad Ciudad { get; set; }
18        1 reference
19        public int IdTipoPerFk { get; set; }
20        1 reference
21        public TipoPersona TipoPersona { get; set; }
22        0 references
23        public ICollection<Matricula> Matriculas { get; set; }
24        0 references
25        public ICollection<TrainerSalon> TrainerSalones { get; set; }
26    }
```

```
1 namespace Dominio;
2
3     4 references
4     public class TipoPersona
5     {
6         2 references
7         public int IdTipoPersona { get; set; }
8         1 reference
9         public string DescripcionTipoPersona { get; set; }
10        1 reference
11        public ICollection<Persona> Personas { get; set; }
12    }
```

```
1 namespace Dominio;
2
3     5 references
4     public class Pais
5     {
6         2 references
7         public string IdPais { get; set; }
8         2 references
9         public string NombrePais { get; set; }
10        1 reference
11        public ICollection<Departamento> Departamentos { get; set; }
12    }
```

```
1
2 namespace Dominio;
3
4     5 references
5     public class Salon
6     {
7         2 references
8         public int IdSalon { get; set; }
9         1 reference
10        public string NombreSalon { get; set; }
11        1 reference
12        public int Capacidad { get; set; }
13        0 references
14        public ICollection<Matricula> Matriculas { get; set; }
15        0 references
16        public ICollection<TrainerSalon> TrainerSalones { get; set; }
17    }
```



```
1 namespace Dominio;
2
3     4 references
4     public class Genero
5     {
6         2 references
7         public int IdGenero { get; set; }
8         1 reference
9         public string NombreGenero { get; set; }
10        1 reference
11        public ICollection<Persona> Personas { get; set; }
12    }
```

```
1 namespace Dominio;
2
3     5 references
4     public class Ciudad
5     {
6         2 references
7         public string IdCiudad { get; set; }
8         1 reference
9         public string NombreCiudad { get; set; }
10        1 reference
11        public string IdDepFk { get; set; }
12        1 reference
13        public Departamento Departamento { get; set; }
14        1 reference
15        public ICollection<Persona> Personas { get; set; }
16    }
```

```
1 namespace Dominio;
2
3     5 references
4     public class Departamento
5     {
6         2 references
7         public string IdDep { get; set; }
8         1 reference
9         public string NombreDep { get; set; }
10        1 reference
11        public string IdPaisFk { get; set; }
12        1 reference
13        public Pais Pais { get; set; }
14        1 reference
15        public ICollection<Ciudad> Ciudades { get; set; }
16    }
```

```
1 namespace Dominio;
2
3     4 references
4     public class TrainerSalon
5     {
6         1 reference
7         public string IdPerTrainerFk { get; set; }
8         0 references
9         public Persona Persona { get; set; }
10        1 reference
11        public int IdSalonFk { get; set; }
12        0 references
13        public Salon Salon { get; set; }
14    }
```

```
1 namespace Dominio;
2
3     5 references
4     public class Matricula
5     {
6         2 references
7         public int IdMatricula { get; set; }
8         0 references
9         public string IdPersonaFk { get; set; }
10        0 references
11        public Persona Persona { get; set; }
12        0 references
13        public int IdSalonFk { get; set; }
14        0 references
15        public Salon Salon { get; set; }
16    }
```

Creación Archivos de Configuración

```

    Data\Configuration
    CiudadConfiguration.cs
    DepartamentoConfiguration.cs
    GeneroConfiguration.cs
    MatriculaConfiguration.cs
    PaisConfiguration.cs
    PersonaConfiguration.cs
    SalonConfiguration.cs
    TipoPersonaConfiguration.cs
    TrainerSalonConfiguration.cs

```



```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class CiudadConfiguration : IEntityTypeConfiguration<Ciudad>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<Ciudad> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("ciudad");
16
17             builder.HasKey(e => e.IdCiudad);
18             builder.Property(e => e.IdCiudad)
19                 .HasMaxLength(3);
20
21             builder.Property(p => p.NombreCiudad)
22                 .IsRequired()
23                 .HasMaxLength(50);
24
25             builder.HasOne(p => p.Departamento)
26                 .WithMany(p => p.Ciudades)
27                 .HasForeignKey(p => p.IdDepFk);
28         }
29     }
30 }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class GeneroConfiguration : IEntityTypeConfiguration<Genero>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<Genero> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("genero");
16
17             builder.HasKey(e => e.IdGenero);
18             builder.Property(e => e.IdGenero)
19                 .IsRequired()
20                 .HasMaxLength(50);
21
22         }
23     }
24 }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class PaisConfiguration : IEntityTypeConfiguration<Pais>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<Pais> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("pais");
16
17             builder.HasKey(e => e.IdPais);
18             builder.Property(e => e.IdPais)
19                 .HasMaxLength(3);
20
21             builder.Property(p => p.NombrePais)
22                 .IsRequired()
23                 .HasMaxLength(50);
24
25         }
26     }
27 }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class MatriculaConfiguration : IEntityTypeConfiguration<Matricula>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<Matricula> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("matricula");
16
17             builder.HasKey(e => e.IdMatricula);
18             builder.Property(e => e.IdMatricula);
19
20         }
21     }
22 }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4  namespace Persistencia.Data.Configuration
5  {
6      0 references
7      public class PersonaConfiguration : IEntityTypeConfiguration<Persona>
8      {
9          0 references
10         public void Configure(EntityTypeBuilder<Persona> builder)
11         {
12             // Aquí puedes configurar las propiedades de la entidad Marca
13             // utilizando el objeto 'builder'.
14             builder.ToTable("persona");
15
16             builder.HasKey(e => e.IdPersona);
17             builder.Property(e => e.IdPersona)
18                 .HasMaxLength(20);
19
20             builder.Property(p => p.NombrePersona)
21                 .IsRequired()
22                 .HasMaxLength(50);
23
24             builder.HasOne(p => p.Genero)
25                 .WithMany(p => p.Personas)
26                 .HasForeignKey(p => p.IdGeneroFk);
27
28             builder.HasOne(p => p.ciudad)
29                 .WithMany(p => p.Perso)
30                 .HasForeignKey(p => p.TipoPersona)    'TipoPersona' is not null here.
31
32             builder.HasOne(p => p.TipoPersona)
33                 .WithMany(p => p.Personas)
34                 .HasForeignKey(p => p.IdTipoPerFk);
35         }
36     }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class SalonConfiguration : IEntityTypeConfiguration<Salon>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<Salon> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("salon");
16
17             builder.HasKey(e => e.IdSalon);
18             builder.Property(e => e.IdSalon);
19
20             builder.Property(p => p.NombreSalon)
21                 .IsRequired()
22                 .HasMaxLength(50);
23
24             builder.Property(p => p.Capacidad)
25                 .HasColumnType("int");
26
27         }
28     }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class TipoPersonaConfiguration : IEntityTypeConfiguration<TipoPersona>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<TipoPersona> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("tipopersona");
16
17             builder.HasKey(e => e.IdTipoPersona);
18             builder.Property(e => e.IdTipoPersona);
19
20             builder.Property(p => p.DescripcionTipoPersona)
21                 .IsRequired()
22                 .HasMaxLength(50);
23         }
24     }

```

```

1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration
6  {
7      0 references
8      public class TrainerSalonConfiguration : IEntityTypeConfiguration<TrainerSalon>
9      {
10         0 references
11         public void Configure(EntityTypeBuilder<TrainerSalon> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("trainersalon");
16
17             builder.Property(p => p.IdPerTrainerFk)
18                 .HasMaxLength(20);
19
20             builder.Property(p => p.IdSalonFk)
21                 .HasColumnType("int");
22
23             builder.HasOne(p => p.Persona)
24                 .WithMany(p => p.TrainerSalones)
25                 .HasForeignKey(p => p.IdPerTrainerFk);
26
27             builder.HasOne(p => p.Salon)
28                 .WithMany(p => p.TrainerSalones)
29                 .HasForeignKey(p => p.IdSalonFk);
30
31         }
32     }

```

Migraciones

→ Verificar si se encuentra instalada la herramienta de migraciones de EF.

```

PS C:\Users\developer> dotnet tool list -g
Id. de paquete      Versión      Comandos
-----
PS C:\Users\developer> -

```

✓ Comando Usado Para
Verificar Si Están
Instaladas Herramientas

2) Instalar la Herramienta dotnet-ef

```
PS C:\Users\developer> dotnet tool install --global dotnet-ef
Puede invocar la herramienta con el comando siguiente: dotnet-ef
La herramienta "dotnet-ef" (versión '7.0.10') se instaló correctamente.
PS C:\Users\developer> dotnet tool list -g
Id. de paquete      Versión      Comandos
-----
dotnet-ef            7.0.10       dotnet-ef
PS C:\Users\developer>
```

3) Crear La migración.

```
PS D:\NetCore\incidencias-app> dotnet ef migrations add InitialCreateMig --project .\Persistencia\ --startup-project
.\ApiIncidencias\ --output-dir ./Data/Migrations
Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'
PS D:\NetCore\incidencias-app>
```

Nombre Migración

Carpeta Donde Se
desea crear la mi-
gracion.

proyecto Donde Se
desea crear la migra-
cion.

Si al momento de crear la migracion obtiene el
siguiente mensaje

```
Your startup project 'ApiIncidencias' doesn't reference Microsoft.EntityFrameworkCore.Design. This package is required for the Entity Framework Core Tools to work. Ensure your startup project is correct, install the package, and try again.
```

Debe instalar el paquete



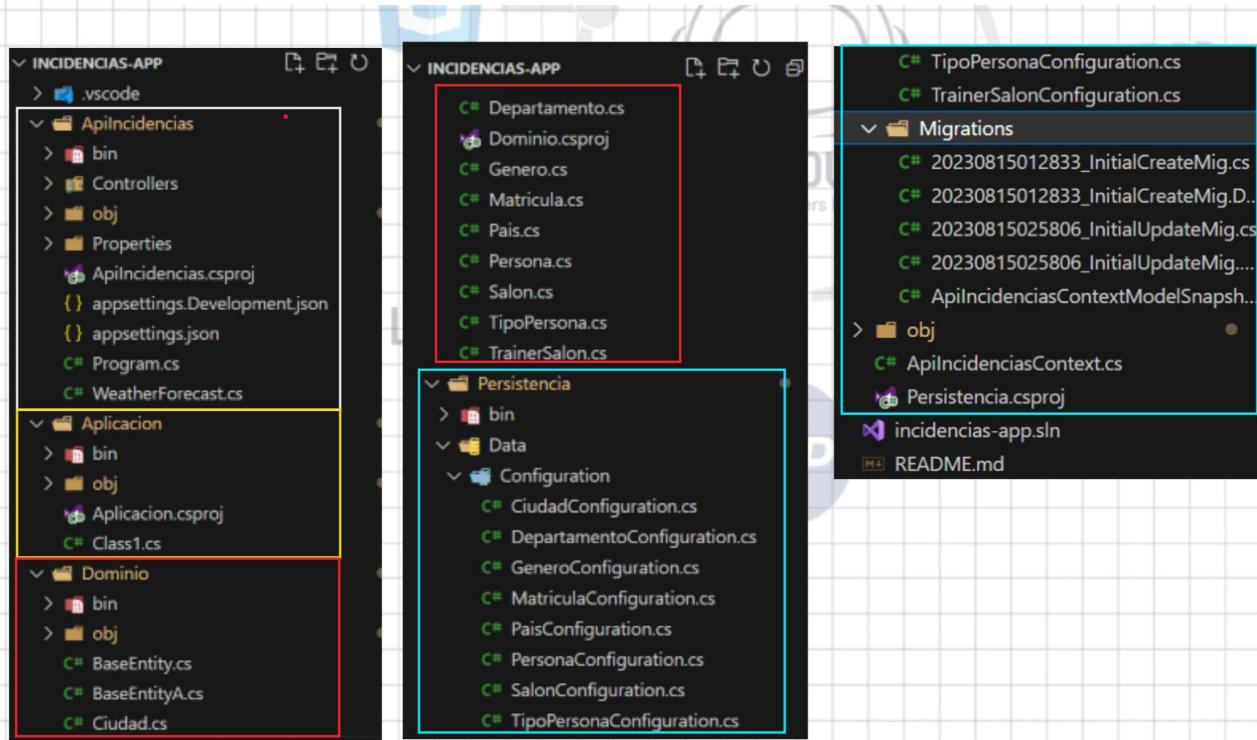
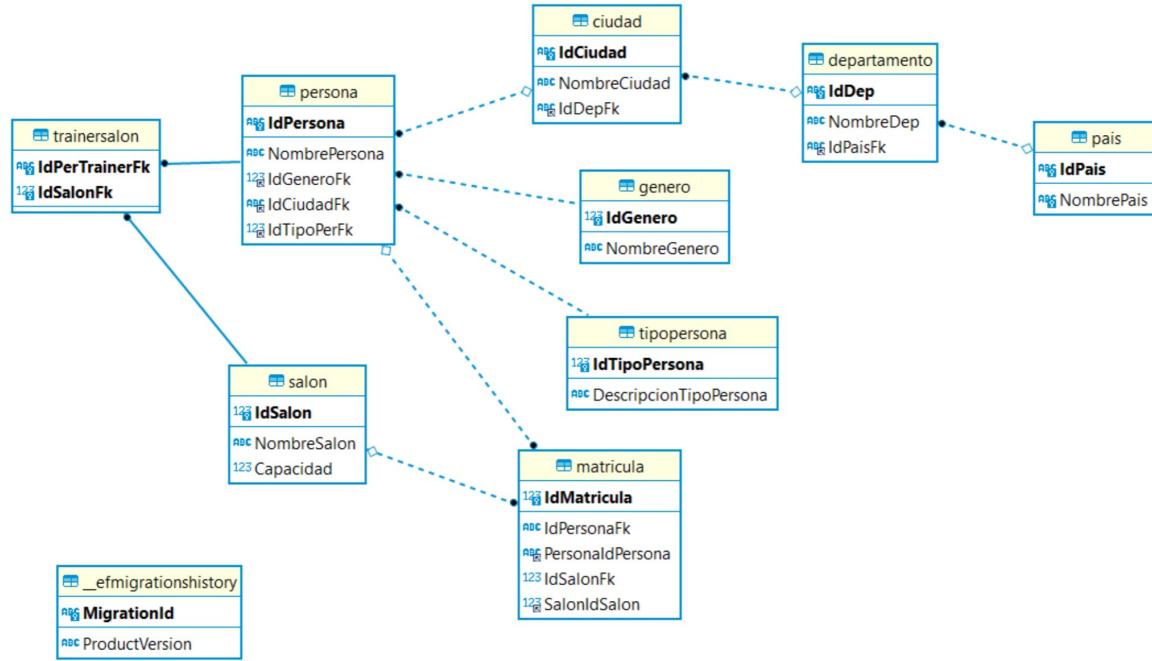
En el proyecto de tipo WebApi

4) Aplicar La Migración

```
PS D:\NetCore\incidencias-app> dotnet ef database update --project .\Persistencia\ --startup-project .\ApiIncidencias\
```

dotnet ef database update --project .\Persistencia\ --startup-project .\ApiIncidencias\

Proyecto donde se genero Migracion



Creación De Clases Reutilizables

```
C# BaseEntity.cs X  
Dominio > C# BaseEntity.cs > BaseEntity > Id  
1  namespace Dominio;  
2  
3  public class BaseEntity  
4  {  
5      public string Id { get; set; }  
6  }
```

```
C# BaseEntityA.cs X  
Dominio > C# BaseEntityA.cs > BaseEntityA  
1  
2  namespace Dominio;  
3  
4  public class BaseEntityA  
5  {  
6      public int Id { get; set; }  
7  }
```

Se Crean En El Proyecto
Dominio

Repetimos El Mismo Proceso Para Los Controllers.
Creando Una Clase Llamada **BaseApiController**

```
C# BaseApiController.cs U X C# WeatherForecastController.cs  
Apilncidencias > Controllers > C# BaseApiController.cs > ...  
1  using Microsoft.AspNetCore.Mvc; ←  
2  
3  namespace Apilncidencias.Controllers;  
4  
5  [ApiController]  
6  [Route("api/incidencias/[controller]")]  
7  public class BaseApiController : ControllerBase  
8  {  
9  
10 }  
11 |
```

Metodos De Extensiones → Los Metodos De Extensiones permiten agregar nuevos metodos a tipos existentes Sin modificar el Original. Es Decir Permite Extender la Funcionalidad de las Clases y tipos Sin Necesidad de implementar herencia.

Para Agregar Método De Extensión Siga los Siguientes Pasos:

- 1) Cree Una Carpeta llamada **Extensions** En El Proyecto webAPI.
- 2) Cree Una Nueva Clase En La Carpeta Extensions y Llámela **C# ApplicationServiceExtension.cs**
La clase Debe Ser Estática.

```
1 namespace ApiIncidentes.Extensions;
2
3     0 references
4     public static class ApplicationServiceExtension
5     {
6         0 references
7         public static void ConfigureCors(this IServiceCollection services) =>
8             services.AddCors(options =>
9                 options.AddPolicy("CorsPolicy", builder =>
10                     builder.AllowAnyOrigin()           //WithOrigins("https://domini.com")
11                     .AllowAnyMethod()                //WithMethods(*GET*, "POST")
12                     .AllowAnyHeader());            //WithHeaders(*accept*, "content-type")
13             );
14     }
15 }
```

Configurar El Servicio En Program.cs

ApiIncidentes > C# Program.cs

```
1 using ApiIncidentes.Extensions;
2 using Microsoft.EntityFrameworkCore;
3 using Persistencia;
4
5 var builder = WebApplication.CreateBuilder(args);
6
7 // Add services to the container.
8
9 builder.Services.AddControllers();
10 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
11 builder.Services.AddEndpointsApiExplorer();
12 builder.Services.AddSwaggerGen();
13 builder.Services.ConfigureCors(); ←
14 builder.Services.AddDbContext<ApiIncidentesContext>(options =>
15     options.UseSqlServer("MyConnection"));
16
17 app.UseHttpsRedirection();
18 app.UseAuthorization();
19 app.MapControllers();
20 app.Run(); ←
```

Unidad Trabajo → Es Un Patrón De Diseño Que Permite agrupar una o mas operaciones (creación, lectura, Actualización y Eliminación) en Una Única transacción.

Repositorio Generico → En El Proyecto **Dominio** Crear Una Carpeta llamada **Interfaces** y Crear Una Interface llamada **IGenericRepository**

```
1  using System.Linq.Expressions;
2
3  namespace Dominio.Interfaces;
4
5  public interface IGenericRepository<T> where T : BaseEntity
6  {
7      Task<T> GetByIdAsync(string id);
8      Task<IEnumerable<T>> GetAllAsync();
9      IEnumerable<T> Find(Expression
```

Implementación de la Interface Generica:

- 1) Cree Una Carpeta llamada **Repository** En El Proyecto **Aplicacion**
- 2) Cree Una Clase Que Permita Implementar **IGenericRepository**.

```
Aplicacion > Repository > C# GenericRepository.cs > GenericRepository
  1  using System;
  2  using System.Collections.Generic;
  3  using System.Linq;
  4  using System.Linq.Expressions;
  5  using System.Threading.Tasks;
  6  using Dominio;
  7  using Dominio.Interfaces;
  8  using Microsoft.EntityFrameworkCore;
  9  using Persistencia;
10
11 namespace Aplicacion.Repository;
12
13     1 reference
14     public class GenericRepository<T> : IGenericRepository<T> where T : BaseEntity
15     {
16         private readonly ApiIncidenciasContext _context;
17
18         0 references
19         public GenericRepository(ApiIncidenciasContext context)
20         {
21             _context = context;
22         }
23
24         1 reference
25         public virtual void Add(T entity)
26         {
27             _context.Set<T>().Add(entity);
28
29         1 reference
30         public virtual void AddRange(IEnumerable<T> entities)
31         {
32             _context.Set<T>().AddRange(entities);
33
34         1 reference
35         public virtual IEnumerable<T> Find(Expression<Func<T, bool>> expression)
36         {
37             return _context.Set<T>().Where(expression);
38
39         1 reference
40         public virtual async Task<IEnumerable<T>> GetAllAsync()
41         {
42             return await _context.Set<T>().ToListAsync();
43
44         0 references
45         public virtual async Task<T> GetByIdAsync(int id)
46         {
47             return await _context.Set<T>().FindAsync(id);
48
49         1 reference
50         public Task<T> GetByIdAsync(string id)
51         {
52             throw new NotImplementedException();
53
54         51
55             1 reference
56             public virtual void Remove(T entity)
57             {
58                 _context.Set<T>().Remove(entity);
59
60             1 reference
61             public virtual void RemoveRange(IEnumerable<T> entities)
62             {
63                 _context.Set<T>().RemoveRange(entities);
64
65             1 reference
66             public virtual void Update(T entity)
67             {
68                 _context.Set<T>()
69                     .Update(entity);
70             }
71         }
72     }
73 }
```

Creación De Repositorios Para El Ejercicio Práctico Crearemos El Repositorio de Países. (IPaisRepository)

Dominio > Interfaces > C# IPaisRepository.cs > IPaisRepository

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5
6  namespace Dominio.Interfaces
7  {
8      0 references
9      public interface IPaisRepository : IGenericRepository<Pais>
10     {
11     }
12 }
```

A continuación se debe implementar la Interface.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Threading.Tasks;
5  using Dominio;
6  using Dominio.Interfaces;
7  using Persistencia;
8
9  namespace Aplicacion.Repository;
10
11  1 reference
12  public class PaisRepository : GenericRepository<Pais>, IPaisRepository
13  {
14      0 references
15      public PaisRepository(ApiIncidenciasContext contex) : base(contex)
16      {
17      }
18 }
```