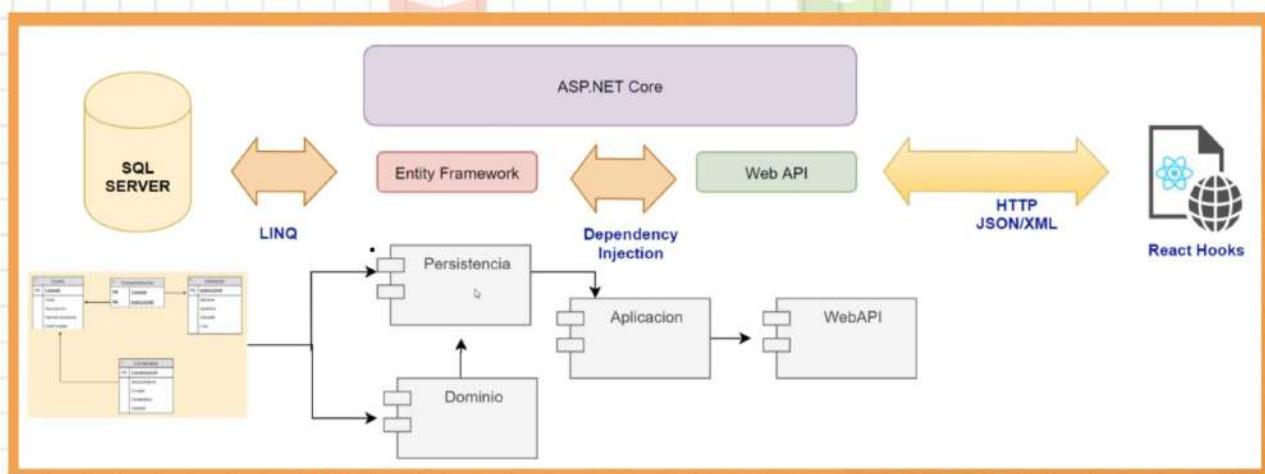
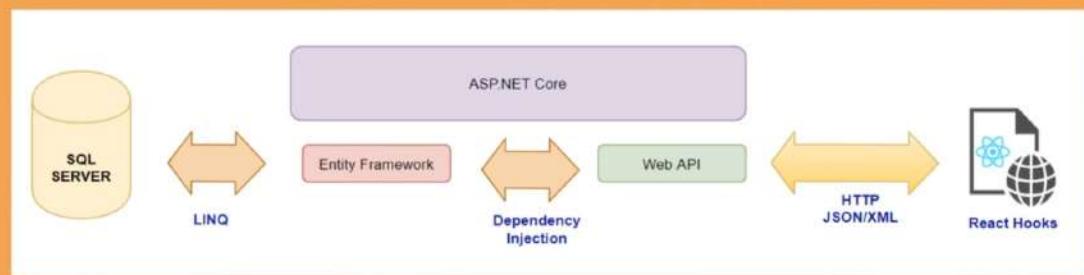


Nuestro Proyecto - Arquitectura



A) Agregar Proyectos A Solución.

B) Agregar Referencias.

Aplicación → **Dominio**
Aplicación → **Persistencia**

WebApi → **Aplicación**
Seguridad

Persistencia → **Dominio**.

Seguridad → **Aplicación**

C) Paquetes

WebApi →

```
<PackageReference Include="AspNetCoreRateLimit" Version="5.0.0" />
<PackageReference Include="AutoMapper.Extensions.Microsoft.DependencyInjection" Version="12.0.1" />
<PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="7.0.10" />
<PackageReference Include="Microsoft.AspNetCore.Mvc.Versioning" Version="5.1.0" />
<PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="7.0.10" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="7.0.10" />
```

Dominio →

```
<ItemGroup>
  <PackageReference Include="FluentValidation.AspNetCore" Version="11.3.0" />
  <PackageReference Include="itext7.pdfhtml" Version="5.0.1" />
  <PackageReference Include="Microsoft.EntityFrameworkCore" Version="7.0.10" />
</ItemGroup>
```

Persistencia →

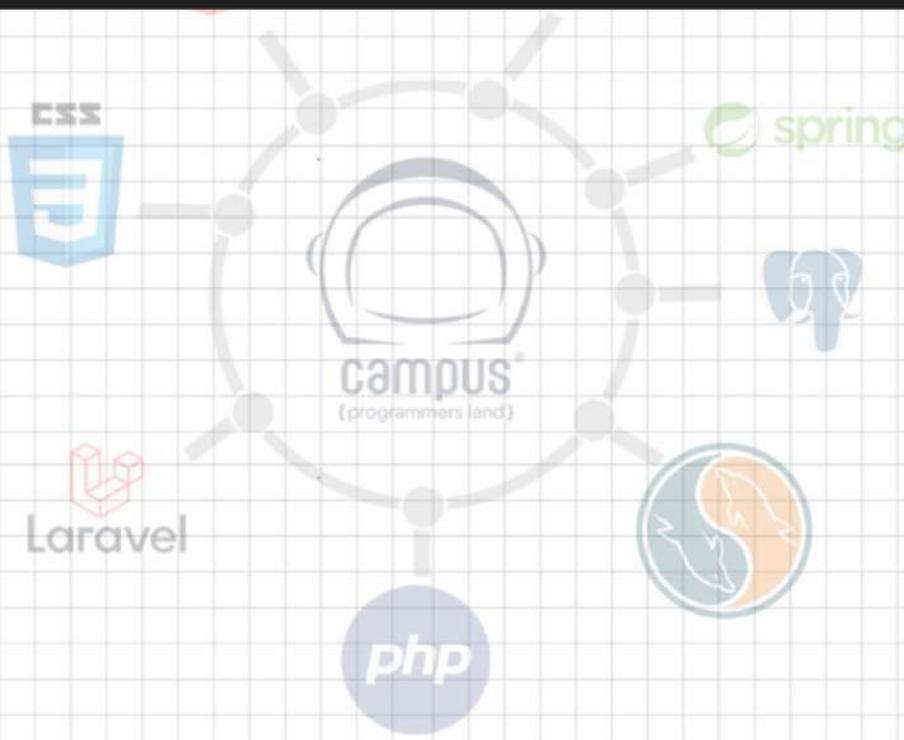
```
<ItemGroup>
  <PackageReference Include="Microsoft.EntityFrameworkCore" Version="7.0.10" />
  <PackageReference Include="Pomelo.EntityFrameworkCore.MySql" Version="7.0.0" />
</ItemGroup>
```

HTML

JS

Seguridad →

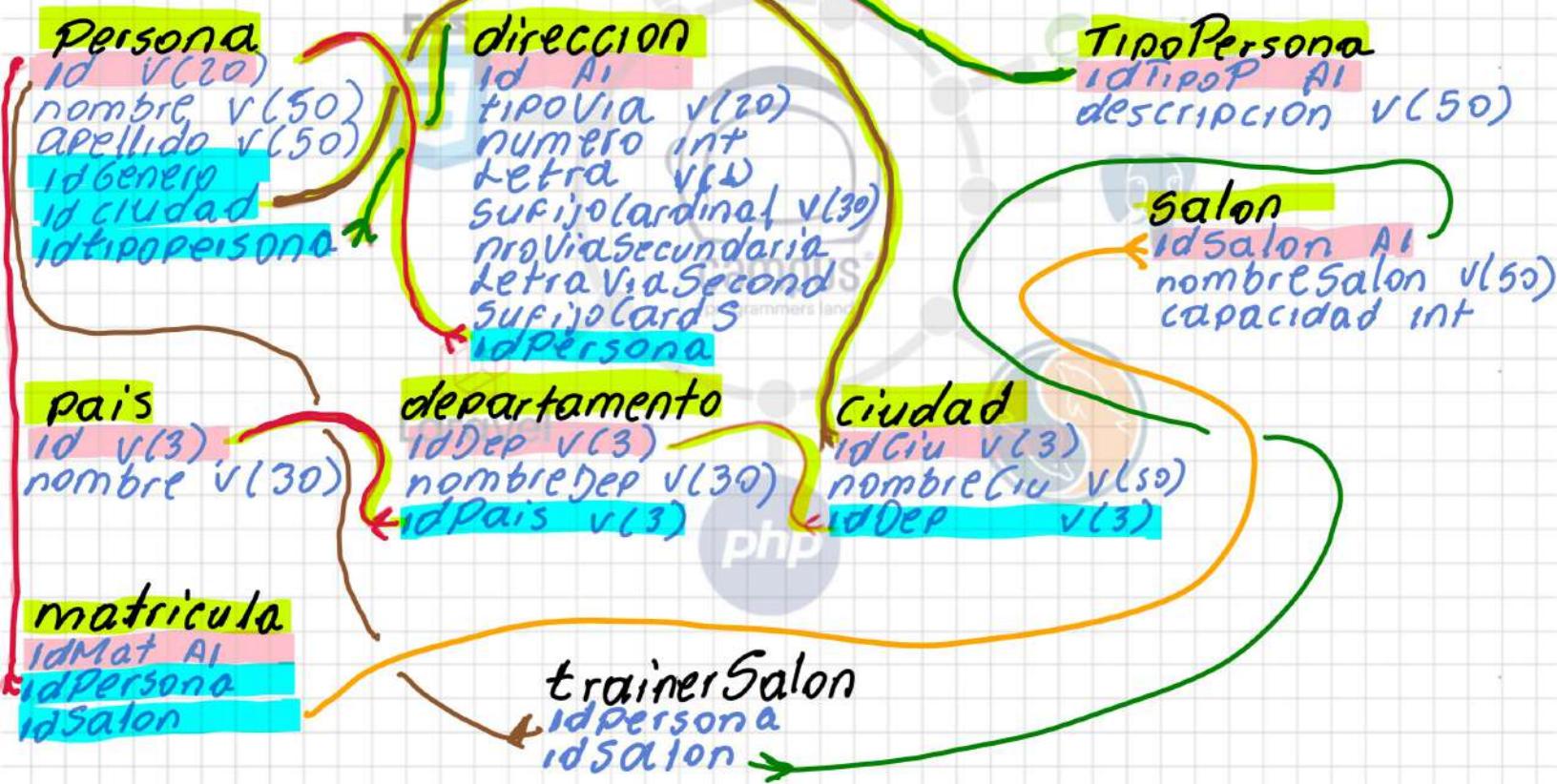
```
<ItemGroup>
  <PackageReference Include="System.IdentityModel.Tokens.Jwt" Version="6.32.2" />
</ItemGroup>
```



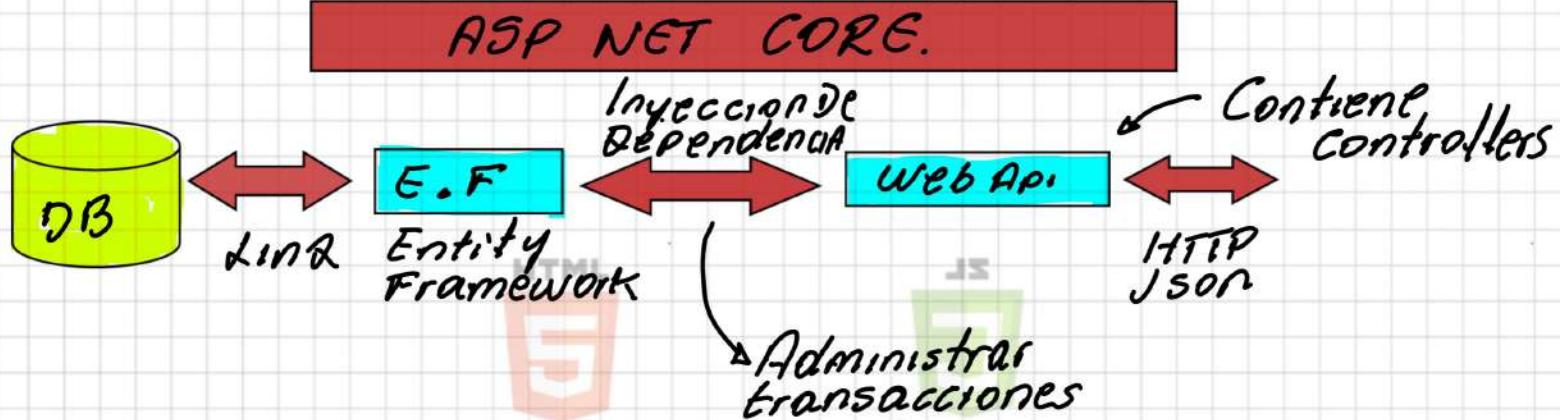
En Esta Guía Práctica Aprenderá A Crear Un Proyecto WebApi Usando .NetCore 7.0 y MySQL Como Gestor De Bases De Datos.

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

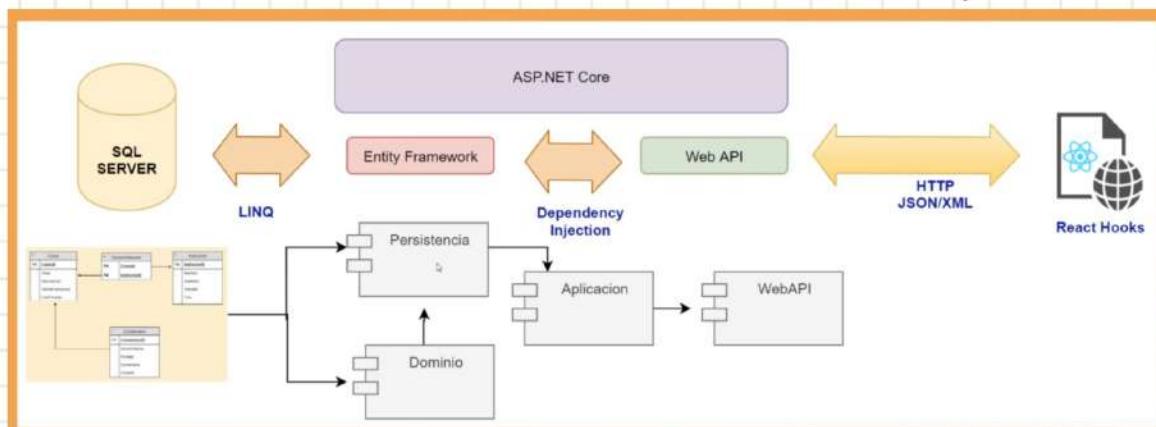
7) Agregar los Proyectos A la Solución Principales

Comando Para Visualizar los Proyectos Asociados a la solución principal.

```
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
```

Comando Usado Para agregar Un Proyecto a la Solución.

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.

nuget.org/packages?q=EntityFrameworkCore&frameworks=&tflms=&packageType=&prerelease=true&sortBy=relevance

nuget Packages Upload Statistics Documentation Downloads Blog Sign in

EntityFrameworkCore

Frameworks ⓘ

- .NET
- .NET Core
- .NET Standard
- .NET Framework

Package type

- All types
- Dependency
- .NET tool

4,137 packages returned for EntityFrameworkCore

Sort by Relevance

Microsoft.EntityFrameworkCore by: aspnet EntityFramework Microsoft

767,940,155 total downloads last updated 18 days ago Latest version: 8.0.0-preview.6.23329.4

Entity Framework Core entity-framework-core EF Data O/RM EntityFramework EntityFrameworkCore EFCore

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)

NuGet: Open NuGet Gallery

recently used

other commands

Dominio.csproj X NuGet Gallery X

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

Refresh FluentValidation.AspNetCore by Jeremy Skinner

Aplicacion.csproj Install
DinoApi.csproj Install
 Dominio.csproj Install
Persistencia.csproj Install

v11.3.0 v3.3.2 11.3.0 Install Uninstall

Dominio.csproj X NuGet Gallery X

iText

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-

Refresh iText7.pdfhtml by Apryse Software

Aplicacion.csproj Install
 Dominio.csproj Install
Persistencia.csproj Install

v8.0.0 v5.0.0 5.0.0 Install Uninstall

Dominio.csproj X NuGet Gallery X

.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

Refresh Microsoft.EntityFrameworkCore by Microsoft

Aplicacion.csproj Install
DinoApi.csproj Install
Dominio.csproj 7.0.9 Uninstall
 Persistencia.csproj Install

v7.0.9 v7.0.9 7.0.9 Install Uninstall

Dominio.csproj X NuGet Gallery X

mysql

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.

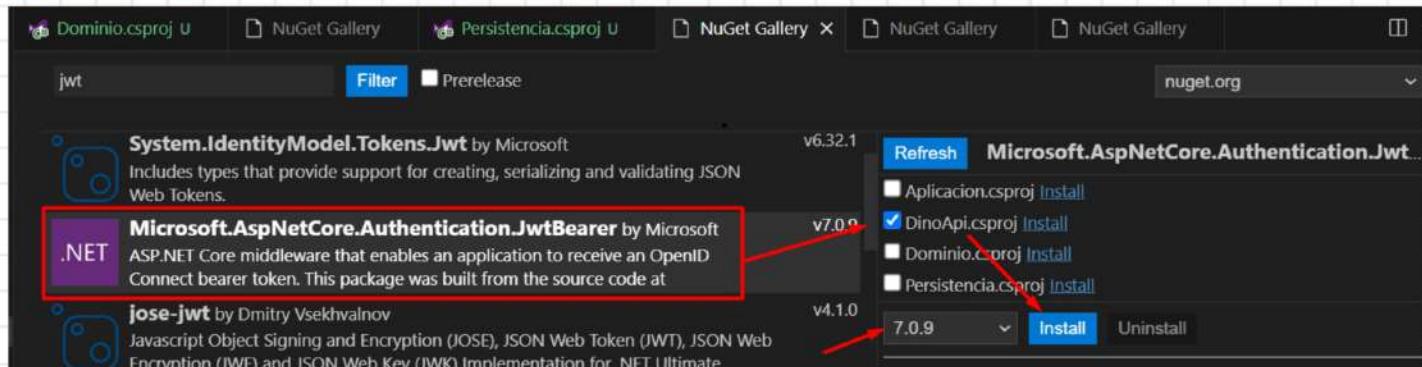
MySQL.Data.EntityFrameworkCore by Oracle

MySQL.Data.EntityFrameworkCore for Entity Framework.

Refresh Pomelo.EntityFrameworkCore.MySql by La...

Aplicacion.csproj Install
DinoApi.csproj Install
Dominio.csproj Install
 Persistencia.csproj Install

v7.0.0 v2.2.7 v8.0.22 7.0.0 Install Uninstall



Conexión Base De Datos

→ Configuración De La Cadena De Conexión.

{ } appsettings.Development.json

{ } appsettings.json

```

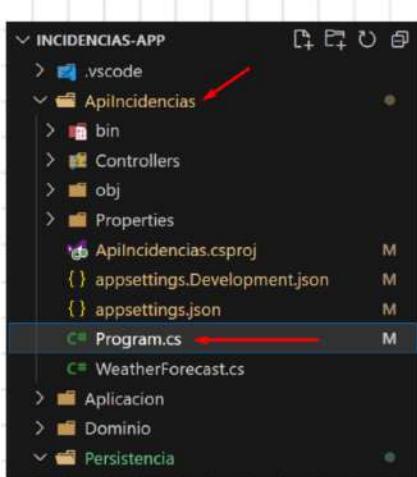
1  {
2    "ConnectionStrings": {
3      "ConexSqlServer": "Data Source=localhost\\sqlexpress;Initial Catalog=dB;Integrate Security=True",
4      "ConexMysql": "server=localhost;user=root;password=123456;database=incidenciasdb"
5    },
6    "Logging": {
7      "LogLevel": {
8        "Default": "Information",
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)



```
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     0 references
4     public class Persona
5     {
6         0 references
7         public string IdTipoPersona { get; set; }
8         0 references
9         public string NombrePersona { get; set; }
10        0 references
11        public int IdGeneroFk { get; set; }
12        0 references
13        public int IdCiudadFk { get; set; }
14        0 references
15        public int IdTipoPerFk { get; set; }
16    }
```

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

```
1 namespace Dominio;
2
3     0 references
4     public class Ciudad
5     {
6         0 references
7         public string IdCiudad { get; set; }
8         0 references
9         public string NombreCiudad { get; set; }
10        0 references
11         public string IdDepFk { get; set; }
12    }
```

```
1 namespace Dominio;
2
3     0 references
4     public class Genero
5     {
6         0 references
7         public int IdGeneroFk { get; set; }
8         0 references
9         public string NombreGenero { get; set; }
10    }
```

```
1 namespace Dominio;
2
3     0 references
4     public class Pais
5     {
6         0 references
7         public string IdPais { get; set; }
8         0 references
9         public string NombrePais { get; set; }
10    }
```

```

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             }
13

```

```

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             }
11

```

```

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             }
13

```

```

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             }
11

```

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                 {
11                     }
12                     0 references
13                         public DbSet<Ciudad> Ciudades { get; set; }
14                         0 references
15                             public DbSet<Persona> Personas { get; set; }
16                             0 references
17                                 public DbSet<Salon> Salones { get; set; }
18                                 0 references
19                                     public DbSet<Matricula> Matriculas { get; set; }
20                                     0 references
21                                         public DbSet<TipoPersona> TipoPersonas { get; set; }
22                                         0 references
23                                             public DbSet<TrainerSalon> TrainerSalones { get; set; }
24                                             0 references
25                                                 public DbSet<Departamento> Departamentos { get; set; }
26                                                 0 references
27                                                     public DbSet<Pais> Paises { get; set; }
28                                                     0 references
29                                                         public DbSet<Genero> Generos { get; set; }
30             }
31

```

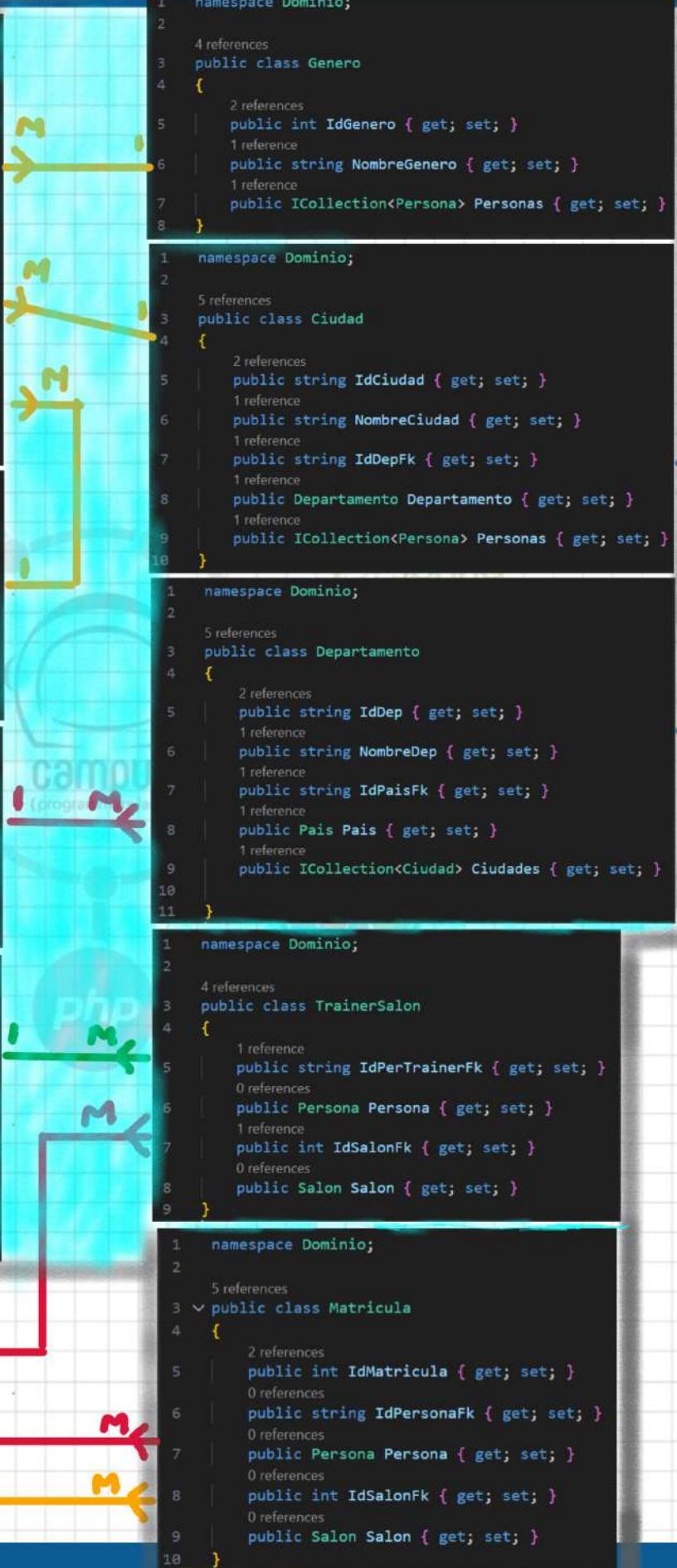
Definición Cardinalida Entre Entidades

```
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    }
```



Creación Archivos de Configuración

```
✓ Data\Configuration
  C# CiudadConfiguration.cs
  C# DepartamentoConfiguration.cs
  C# GeneroConfiguration.cs
  C# MatriculaConfiguration.cs
  C# PaisConfiguration.cs
  C# PersonaConfiguration.cs
  C# SalonConfiguration.cs
  C# TipoPersonaConfiguration.cs
  C# 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 DepartamentoConfiguration : IEntityTypeConfiguration<Departamento>
9     {
10         0 references
11         public void Configure(EntityTypeBuilder<Departamento> builder)
12         {
13             // Aquí puedes configurar las propiedades de la entidad Marca
14             // utilizando el objeto 'builder'.
15             builder.ToTable("departamento");
16
17             builder.HasKey(e => e.IdDep);
18             builder.Property(e => e.IdDep)
19                 .HasMaxLength(3);
20
21             builder.Property(p => p.NombreDep)
22                 .IsRequired()
23                 .HasMaxLength(50);
24
25             builder.HasOne(p => p.Pais)
26                 .WithMany(p => p.Departamentos)
27                 .HasForeignKey(p => p.IdPaisFk);
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 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 }
```

```
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
20             builder.Property(p => p.NombreGenero)
21                 .IsRequired()
22                 .HasMaxLength(50);
23         }
24     }
25 }
```

```
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 }
```

```

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             builderhasOne(p => p.Ciudad)
29                 .WithMany(p => p.Perso)
30                 .HasForeignKey(p => p.TipoPersona)
31                 .WithMany(p => p.Personas)
32                 .HasForeignKey(p => p.IdTipoPerFk);
33         }
34     }
35 }

```

```

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     }
25 }

```

```

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 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             builderhasOne(p => p.Persona)
24                 .WithMany(p => p.TrainerSalones)
25                 .HasForeignKey(p => p.IdPerTrainerFk);
26
27             builderhasOne(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

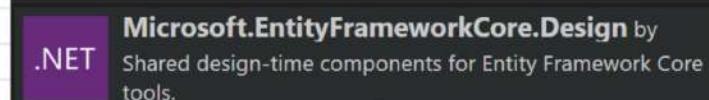
Carpeta Donde Se desea crear la migración.

proyecto Donde Se desea crear la migración.

Si al momento de crear la migración 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



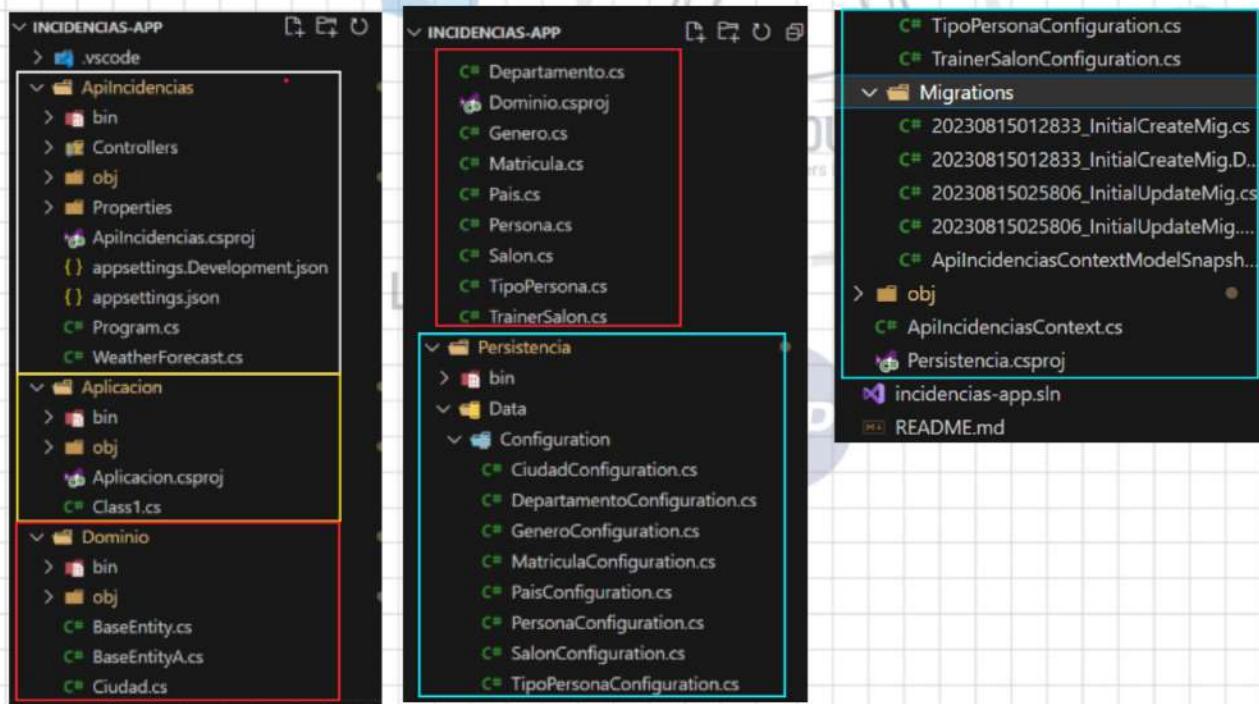
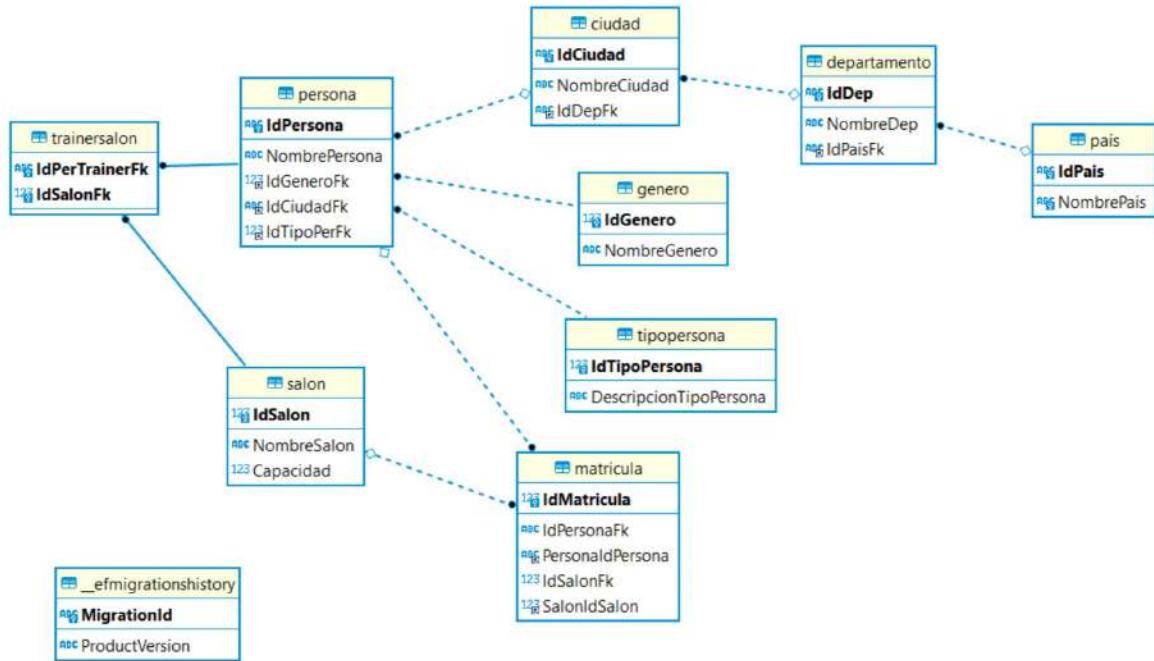
En el proyecto de tipo WebAPI

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 Migración



Creación De Clases Reutilizables

```
C# BaseEntity.cs X
Dominio > C# BaseEntity.cs > BaseEntity > Id
1  namespace Dominio;
2
3  public class BaseEntity
4  {
5
6      public string Id { get; set; }
7
8
C# BaseEntityA.cs X
Dominio > C# BaseEntityA.cs > BaseEntityA
1
2  namespace Dominio;
3
4  public class BaseEntityA
5  {
6
7      public int Id { get; set; }
8
```

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
Aplicaciones > Controllers > C# BaseApiController.cs > ...
1  using Microsoft.AspNetCore.Mvc; ←
2
3  namespace Aplicaciones.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 }
```

Laravel
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
30      1 reference
31      public virtual void AddRange(IEnumerable<T> entities)
32      {
33          _context.Set<T>().AddRange(entities);
34      }
35
36      1 reference
37      public virtual IEnumerable<T> Find(Expression<Func<T, bool>> expression)
38      {
39          return _context.Set<T>().Where(expression);
40      }
41
42      0 references
43      public virtual async Task<IEnumerable<T>> GetAllAsync()
44      {
45          return await _context.Set<T>().ToListAsync();
46      }
47
48      1 reference
49      public Task<T> GetByIdAsync(int id)
50      {
51          return await _context.Set<T>().FindAsync(id);
52      }
53
54      1 reference
55      public Task<T> GetByIdAsync(string id)
56      {
57          throw new NotImplementedException();
58      }
59
60      51
61
62      1 reference
63      public virtual void Remove(T entity)
64      {
65          _context.Set<T>().Remove(entity);
66      }
67
68      1 reference
69      public virtual void RemoveRange(IEnumerable<T> entities)
70      {
71          _context.Set<T>().RemoveRange(entities);
72      }
73
74      1 reference
75      public virtual void Update(T entity)
76      {
77          _context.Set<T>()
78              .Update(entity);
79      }
80  }
```

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(ApiIncidentesContext context) : base(context)
16  }
17 }
```



Creación De La Unidad De Trabajo

1) Crear la Interface De La Unidad De trabajo En El Folder **interfaces** Que Se Encuentra En **Dominio**

```
1  namespace Dominio.Interfaces;
2  1 reference
3  public interface IUnitOfWork
4  {
5      1 reference
6      IPais Paises { get; }
7  }
```

Dominio > Interfaces > C# IUnitOfWork.cs > ...

```
1  namespace Dominio.Interfaces;
2
3  0 references
4  public interface IUnitOfWork
5  {
6      0 references
7      IPaisRepository Paises { get; }
8  }
```

2) Implementar la Interface unidad De Trabajo. La Implementación Se Realiza En El Proyecto Aplicacion

Prop.
Privados de
Repo.

Inyección
close contexto

```
1  using Aplicacion.Repository;
2  using Dominio.Interfaces;
3  using Persistencia;
4  namespace Aplicacion.UnitOfWork
5  {
6      2 references
7      public class UnitOfWork : IUnitOfWork, IDisposable
8      {
9          private readonly ApiIncidenciasContext context;
10         private PaisRepository _paises;
11         0 references
12         public UnitOfWork(ApiIncidenciasContext _context)
13         {
14             context = _context;
15         }
16         1 reference
17         public IPaisRepository Paises
18         {
19             get
20             {
21                 if (_paises == null)
22                 {
23                     _paises = new PaisRepository(context);
24                 }
25                 return _paises;
26             }
27         }
28         1 reference
29         public int Save()
30         {
31             return context.SaveChanges();
32         }
33     }
34 }
```

Cada Una De Las Interfaces y Repositorios Se Deben Implementar En La Unidad De Trabajo.



Permite Guardar los Cambios En El Contexto.

Permite liberar Recursos en El Contexto.

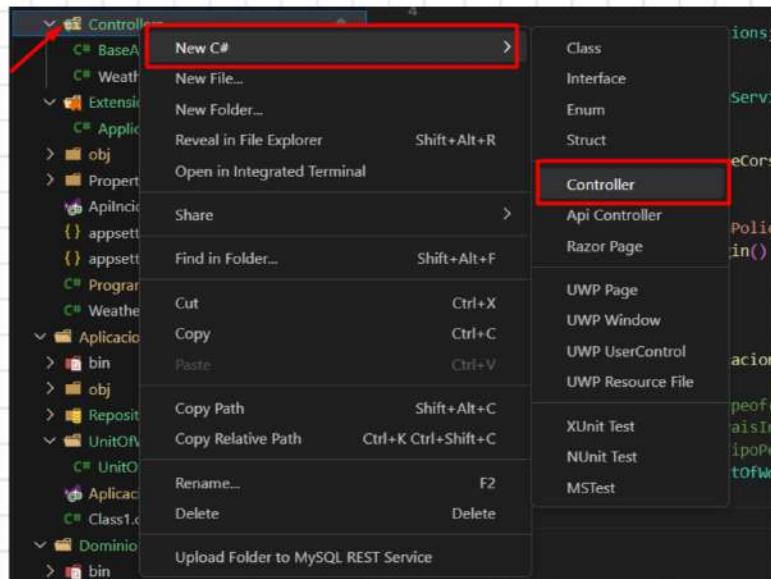
En El Método De Extension Agregaremos da Unidad De trabajo. Para Que Se Inyecte

```
Aplicacion > Extensions > C# ApplicationServiceExtension.cs > ApplicationServiceExtension
1  using Aplicacion.Repository;
2  using Aplicacion.UnitOfWork;
3  using Dominio.Interfaces;
4
5  namespace Aplicacion.Extensions;
6
7  0 references
8  public static class ApplicationServiceExtension
9  [
10     1 reference
11     public static void ConfigureCors(this IServiceCollection services) =>
12         services.AddCors(options =>
13         {
14             options.AddPolicy("CorsPolicy", builder =>
15                 builder.AllowAnyOrigin() //WithOrigins("https://domini.com")
16                 .AllowAnyMethod() //WithMethods("GET", "POST")
17                 .AllowAnyHeader()); //WithHeaders("accept", "content-type")
18         });
19
20     0 references
21     public static void AddAplicacionServices(this IServiceCollection services)
22     {
23         //services.AddScoped(typeof(IGenericRepository<>),typeof(GenericRepository<>));
24         //services.AddScoped<IPaisInterface,PaisRepository>();
25         //services.AddScoped<ITipoPersona,TipoPersonaRepository>(); //
26         services.AddScoped<IUnitOfWork, UnitOfWork>(); //
```

En Caso De Necesitar usar un Repo Generico q' no este En La Unidad De trabajo Se Puede Descomentar La linea De código indicada.

Trabajando Con Controladores

1) En El Proyecto WebAPI Crear El Controlador Requerido. Para El Ejemplo PaísController.



2) Modificar El Archivo Hasta Que Tenga El Siguiente Aspecto.

```
ApiIncidencias > Controllers > C# PaísController.cs > ...
1
2  namespace ApiIncidencias.Controllers;
3  public class PaísController : BaseApiController
4  {
5
6  }
```

3) Crear El Constructor De La Clase Controller Creada Segun Imagen Siguiente.

```
ApiIncidencias > Controllers > C# PaísController.cs > PaísController > .ctor
1
2  using Dominio.Interfaces;
3
4  namespace ApiIncidencias.Controllers;
5  public class PaísController : BaseApiController
6  {
7      private readonly IUnitOfWork unitOfWork;
8
9      public PaísController(IUnitOfWork unitOfWork)
10     {
11         this.unitOfWork = unitOfWork;
12     }
13 }
```

4) Creación De Los Métodos Get, Post, Delete Y Update.

Get

```
15 [HttpGet]
16 [ProducesResponseType(StatusCodes.Status200OK)]
17 [ProducesResponseType(StatusCodes.Status400BadRequest)]
18 0 references
19 public async Task<ActionResult<IEnumerable<Pais>>> Get()
20 {
21     var regiones = await _unitOfWork.Paises.GetAllAsync();
22     return Ok(regiones);
23 }
24 [HttpGet("{id}")]
25 [ProducesResponseType(StatusCodes.Status200OK)]
26 [ProducesResponseType(StatusCodes.Status400BadRequest)]
27 0 references
28 public async Task<IActionResult> Get(string id)
29 {
30     var region = await _unitOfWork.Paises.GetByIdAsync(id);
31     return Ok(region);
32 }
```

Retorna todos los registros contenidos en la tabla.

Permite Retornar un Registro Específico Apartir del Id Principal.

Post

```
31 [HttpPost]
32 [ProducesResponseType(StatusCodes.Status201Created)]
33 [ProducesResponseType(StatusCodes.Status400BadRequest)]
34 1 reference
35 public async Task<ActionResult<Pais>> Post(Pais pais){
36     this._unitOfWork.Paises.Add(pais);
37     await _unitOfWork.SaveChangesAsync();
38     if (pais == null)
39     {
40         return BadRequest();
41     }
42     return CreatedAtAction(nameof(Post), new { id = pais.Id }, pais);
43 }
```

Agrega la información al Contexto.

Envía la Información Desde El Contexto a la BD.

Retorna El Id Generado En El Proceso de Post.

Put

```
43 [HttpPut("{id}")]
44 [ProducesResponseType(StatusCodes.Status200OK)]
45 [ProducesResponseType(StatusCodes.Status404NotFound)]
46 [ProducesResponseType(StatusCodes.Status400BadRequest)]
47 0 references
48 public async Task<ActionResult<Pais>> Put(string id, [FromBody]Pais pais){
49     if(pais == null)
50         return NotFound();
51     _unitOfWork.Paises.Update(pais);
52     await _unitOfWork.SaveChangesAsync();
53     return pais;
54 }
```

Contiene la info a Actualizar.

Actualiza En El Contexto

Guarda la Actualización En La BD.

Delete

```
55 [HttpDelete("{id}")]  
56 [ProducesResponseType(StatusCodes.Status204NoContent)]  
57 [ProducesResponseType(StatusCodes.Status404NotFound)]  
58 0 references  
59 public async Task<IActionResult> Delete(string id){  
60     var pais = await _unitOfWork.Paises.GetByIdAsync(id);  
61     if(pais == null){  
62         return NotFound();  
63     }  
64     _unitOfWork.Paises.Remove(pais);  
65     await _unitOfWork.SaveChangesAsync();  
66     return NoContent();  
67 }
```

Busca El Elemento a Eliminar.

Valida Si Se Encontro El Registro. Si No Encontro Nada Retorno NotFound y Sale Del Metodo.

Indica Que El Metodo No Retorna Ningun Contenido En la Rta.

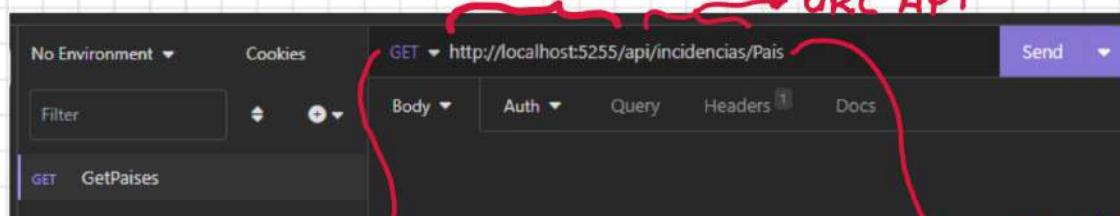
Probando los EndPoints.

1) Para Realizar las Pruebas De los EndPoints Se Debe Contar Con Un Programa Para el Desarrollo de web Api's como Postman o Insomnia. Puede Descargar Cualquier De Estos Programas Desde la Pagina Oficial.

Ej: Mostrar los Paises Que Se Encuentran En la Tabla De Paises usando el EndPoint. Correspondiente al Metodo Get.

Url y Puerto temporal.

URL API



Tipo De Petición

EndPoint.

```
1: [  
2: {  
3: "nombrePais": "Colombia",  
4: "departamentos": null,  
5: "id": "001"  
6: }  
7: ]
```

Respuesta Del Server a la Peticion

```
1 [  
2 {  
3   "nombrePais": "Colombia",  
4   "departamentos": null,  
5   "id": "001"  
6 }  
7 ]
```

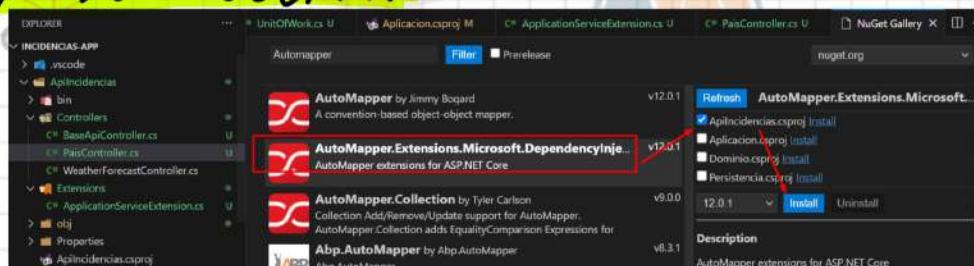
En Respueta Del Servidor Aparece el Atributo De Departamento Como Null Esto Se Puede Solucionar Implementando DTOs.

Trabajando Con Dto (Data Transfer Object) → Es Un Patrón de Diseño usado Para transferir Datos Entre Diferentes Componentes. o Capas De Una Aplicación.

- Ventajas →
- Envio de Información Necesaria
 - Permite Definir la Estructura Exacta de los Datos que Se transfieren.
 - Permiten Ocultar Información Antes De Ser transferida Por La Red.

Como Implementar Dto

1) Instalar Paquete AutoMapper. Se Debe Instalar En El Proyecto WebApi.

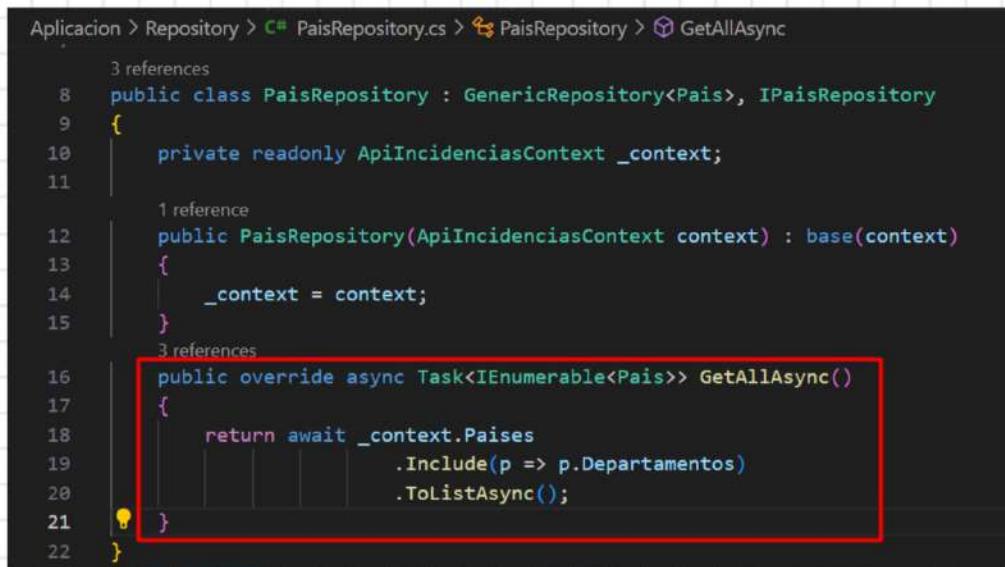


2) Agregar El Servicio de Automapper En Program.cs

```
1 using System.Reflection; ←  
2 using ApiIncidencias.Extensions;  
3 using Microsoft.EntityFrameworkCore;  
4 using Persistencia;  
5  
6 var builder = WebApplication.CreateBuilder(args);  
7  
8 // Add services to the container.  
9  
10 builder.Services.AddControllers();  
11 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle  
12 builder.Services.AddEndpointsApiExplorer();  
13 builder.Services.AddSwaggerGen();  
14 builder.Services.AddAutoMapper(Assembly.GetEntryAssembly()); ←  
15 builder.Services.ConfigureCors();  
16 builder.Services.AddAplicacionServices();
```

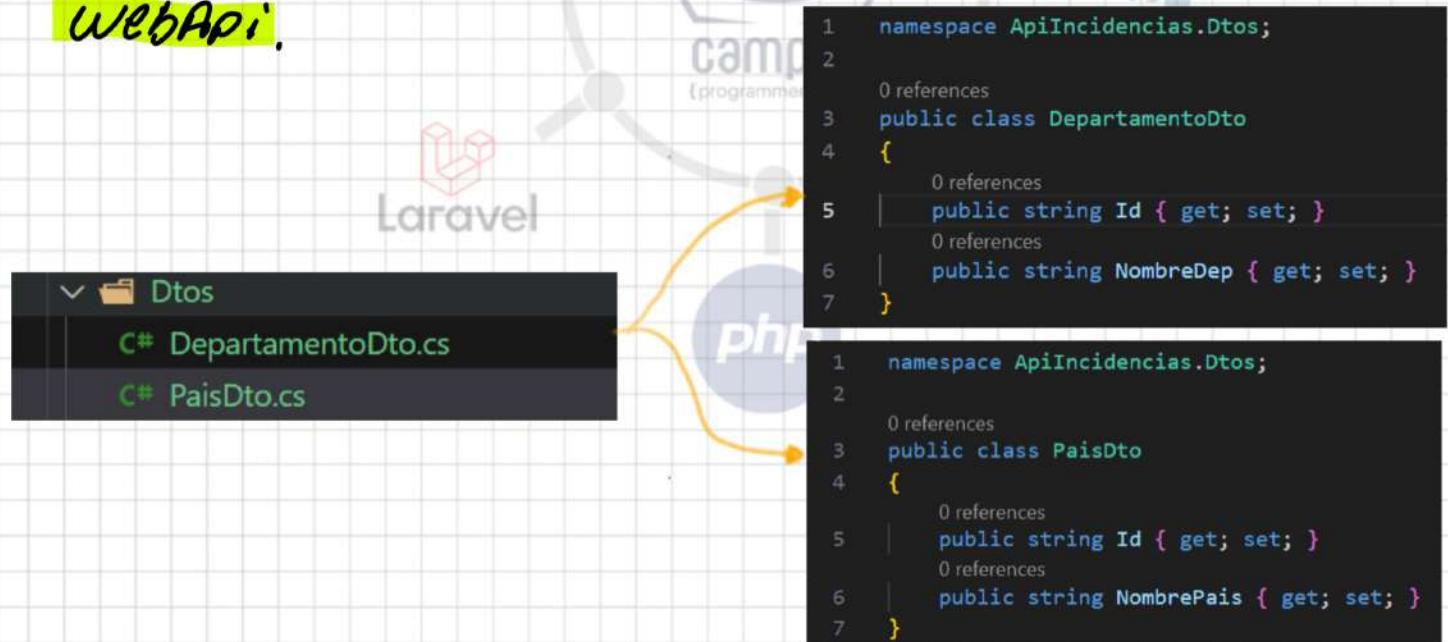
Resolver Null Con Automapper

⇒ Generar Un Override Al Método GetAllAsync



```
Aplicacion > Repository > C# PaisRepository.cs > PaisRepository > GetAllAsync
  8 public class PaisRepository : GenericRepository<Pais>, IPaisRepository
  9 {
10     private readonly ApiIncidenciasContext _context;
11
12     public PaisRepository(ApiIncidenciasContext context) : base(context)
13     {
14         _context = context;
15     }
16     public override async Task<IEnumerable<Pais>> GetAllAsync()
17     {
18         return await _context.Paises
19             .Include(p => p.Departamentos)
20             .ToListAsync();
21     }
22 }
```

2) Crear Las Clases Dto de las Entities. Se Recomienda Crear Una Carpeta Dtos En El Proyecto De Tipo WebApi.



3) Crear Carpeta Profiles En Proyecto WebAPI y En Su Interior Crear El Archivo MappingProfiles.cs

```

1  using ApiIncidencias.Dtos;
2  using Dominio;
3  using AutoMapper;
4  namespace ApiIncidencias.Profiles;
5
6  1 reference
7  public class MappingProfiles : Profile
8  {
9      0 references
10     public MappingProfiles()
11     {
12         CreateMap<Pais,PaisDto>().ReverseMap();
13         CreateMap<Departamento,DepartamentoDto>().ReverseMap();
14         /*CreateMap<Estado,EstadoDto>()
15         .ForMember(dest => dest.IdEstado, opt => opt.MapFrom(src => src.IdCod))
16         .ForMember(dest => dest.Name, opt => opt.MapFrom(src => src.nombreEstado))
17         .ReverseMap();*/
18     }
19 }

```

4) Modificar El Archivo De Controlador. De Países Para Usar Los Dto.

A) Inyectar Automapper En El Constructor. De La Clase

```

9  public class PaisController : BaseApiController
10 {
11     private readonly IUnitOfWork _unitOfWork;
12     private readonly IMapper _mapper; ↗
13
14     0 references
15     public PaisController(IUnitOfWork unitOfWork, IMapper mapper)
16     {
17         this._unitOfWork = unitOfWork;
18         _mapper = mapper; ↗
19     }

```

B) Modificar El Método Que Permite Obtener Los Registros De La Tabla. País.

```

19  /* [HttpGet]
20  [ProducesResponseType(StatusCodes.Status200OK)]
21  [ProducesResponseType(StatusCodes.Status400BadRequest)]
22  public async Task<ActionResult<IEnumerable<Pais>>> Get()
23  {
24      var regiones = await _unitOfWork.Paises.GetAllAsync();
25      return Ok(regiones);
26  }*/
27  [HttpGet]
28  [ProducesResponseType(StatusCodes.Status200OK)]
29  [ProducesResponseType(StatusCodes.Status400BadRequest)]
30  0 references
31  public async Task<ActionResult<IEnumerable<PaisDto>>> Get()
32  {
33      var paises = await _unitOfWork.Paises.GetAllAsync();
34      return _mapper.Map<List<PaisDto>>(paises);
35  }

```



Nota → Es Muy Importante Tener En Cuenta Que Se Deben Crear tantos Dto Como Sean Necesarios. Por Es Si Se Desea Listar Los Paises Con Los Departamentos Hijos.

Ej: Modificar El Metodo Post Del Controlador Para Que Soporte Dto.

```
43     /*[HttpPost]
44     [ProducesResponseType(StatusCodes.Status201Created)]
45     [ProducesResponseType(StatusCodes.Status400BadRequest)]
46     public async Task<ActionResult<Pais>> Post(Pais pais){
47         this._unitOfWork.Paises.Add(pais);
48         await _unitOfWork.SaveAsync();
49         if (pais == null)
50         {
51             return BadRequest();
52         }
53         return CreatedAtAction(nameof(Post), new {id= pais.Id}, pais);
54     }*/

```

Post. Sin Dto



```
55     [HttpPost]
56     [ProducesResponseType(StatusCodes.Status201Created)]
57     [ProducesResponseType(StatusCodes.Status400BadRequest)]
58     public async Task<ActionResult<Pais>> Post(PaisDto paisDto){
59         var pais = _mapper.Map<Pais>(paisDto);
60         this._unitOfWork.Paises.Add(pais);
61         await _unitOfWork.SaveAsync();
62         if (pais == null)
63         {
64             return BadRequest();
65         }
66         paisDto.Id = pais.Id;
67         return CreatedAtAction(nameof(Post), new {id= paisDto.Id}, paisDto);
68     }

```

Post Con Dto.



Test Método Post



POST http://localhost:5255/api/incidencias/Pais

Send

JSON Auth Query Headers Docs

```
1 {
2     "id": "002",
3     "nombrePais": "Portugal"
4 }
```

Petición

201 Created 546 ms 57 B

Preview Headers Cookies Timeline

```
1 {
2     "nombrePais": "Portugal",
3     "departamentos": null,
4     "id": "002"
5 }
```

Respuesta.

Rta Método Get.

```
1 [
2     {
3         "id": "001",
4         "nombrePais": "Colombia"
5     },
6     {
7         "id": "002",
8         "nombrePais": "Portugal"
9     }
10 ]
```

Nota → Si. Se Desea ignorar el Valor null De la Rta a la Petición Post. Realice El Siguiente Cambio En El Archivo **MappingProfiles**

ApiIncidencias > Profiles > C# MappingProfiles.cs > ...

```
1  using ApiIncidencias.Dtos;
2  using Dominio;
3  using AutoMapper;
4  namespace ApiIncidencias.Profiles;
5
6  1 reference
7  public class MappingProfiles : Profile
8  {
9      0 references
10     public MappingProfiles(){
11         CreateMap<Pais,PaisDto>()
12             .ReverseMap()
13             .ForMember(o => o.Departamentos,d => d.Ignore());
14         CreateMap<Departamento,DepartamentoDto>().ReverseMap();
15         /*CreateMap<Estado,EstadoDto>()
16             .ForMember(dest => dest.IdEstado, opt => opt.MapFrom(src => src.IdCod))
17             .ForMember(dest => dest.Name, opt => opt.MapFrom(src => src.nombreEstado))
18             .ReverseMap();*/
19     }
20 }
```

Permite Ignorar El mapeo del Atributo y Así Se Previene El Null.



Q: Modifique El Método Update Para permitir El Uso de Dto En El Proceso De Actualización.

```
69     /*[HttpPut("{id}")]
70     [ProducesResponseType(StatusCodes.Status200OK)]
71     [ProducesResponseType(StatusCodes.Status404NotFound)]
72     [ProducesResponseType(StatusCodes.Status400BadRequest)]*/
73     public async Task<ActionResult<Pais>> Put(string id, [FromBody]Pais pais){
74         if(pais == null)
75             return NotFound();
76         _unitOfWork.Paises.Update(pais);
77         await _unitOfWork.SaveAsync();
78         return pais;
79     }*/
80 
```

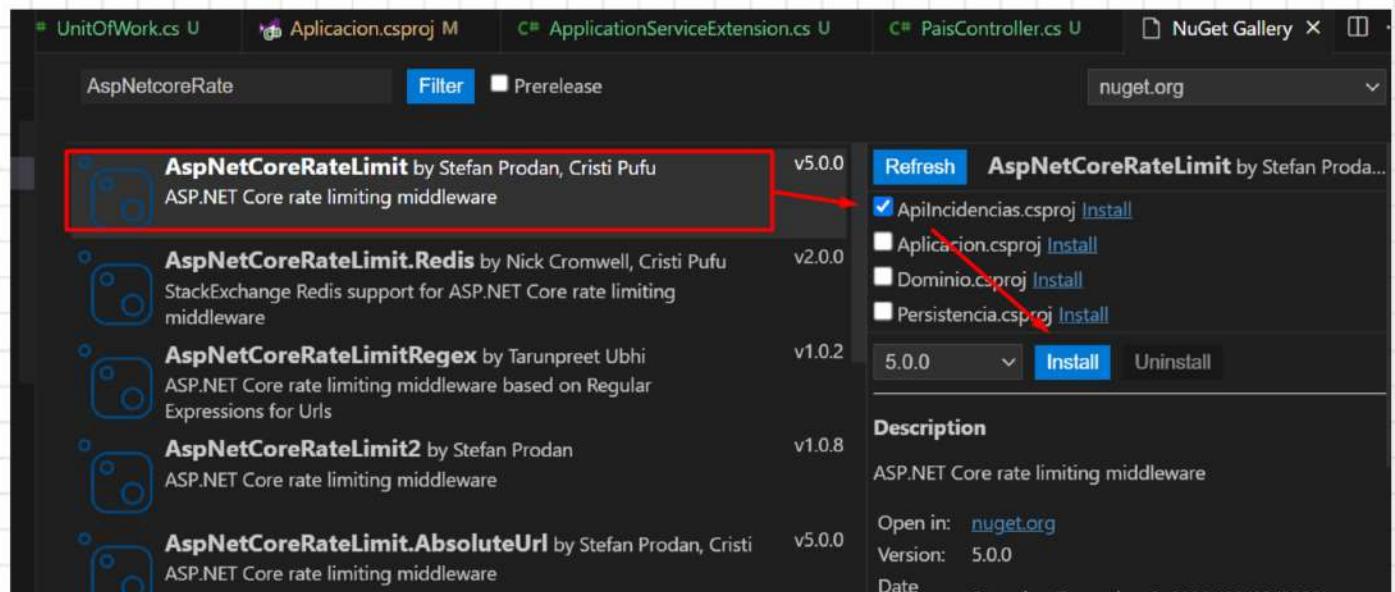


```
[HttpPut("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
0 references
public async Task<ActionResult<PaisDto>> Put(string id, [FromBody]PaisDto paisDto){
    if(paisDto == null)
        return NotFound();
    var paises = _mapper.Map<Pais>(paisDto);
    _unitOfWork.Paises.Update(paises);
    await _unitOfWork.SaveAsync();
    return paisDto;
}
```



Rate Limiting → Proceso De Controlar El Número de Peticiones Que Se Realizan hacia un Recurso En Un Periodo Determinado. De tiempo.

1) Instalar En El Proyecto Web Api



2) Implementar Un Nuevo Método De Extensión En La Clase

```
Apilncidencias > Extensions > C# ApplicationServiceExtension.cs > ...
22.     //services.AddScoped<IIpPersona, IpPersonaRepository>(); //
23.     services.AddScoped<IUnitOfWork, UnitOfWork>();
24. }
25. public static void ConfigureRateLimiting(this IServiceCollection services)
26. {
27.     services.AddMemoryCache();
28.     services.AddSingleton<IRateLimitConfiguration, RateLimitConfiguration>();
29.     services.AddInMemoryRateLimiting();
30.     services.Configure<IpRateLimitOptions>(options =>
31.     {
32.         options.EnableEndpointRateLimiting = true;
33.         options.StackBlockedRequests = false;
34.         options.HttpStatusCode = 429;
35.         options.RealIpHeader = "X-Real-IP";
36.         options.GeneralRules = new List<RateLimitRule>
37.         {
38.             new RateLimitRule
39.             {
40.                 Endpoint = "*",
41.                 Period = "10s",
42.                 Limit = 2
43.             };
44.         };
45.     });
46. }
```

3) Inyectar el Servicio en El Contenedor De Dependencias Program.cs

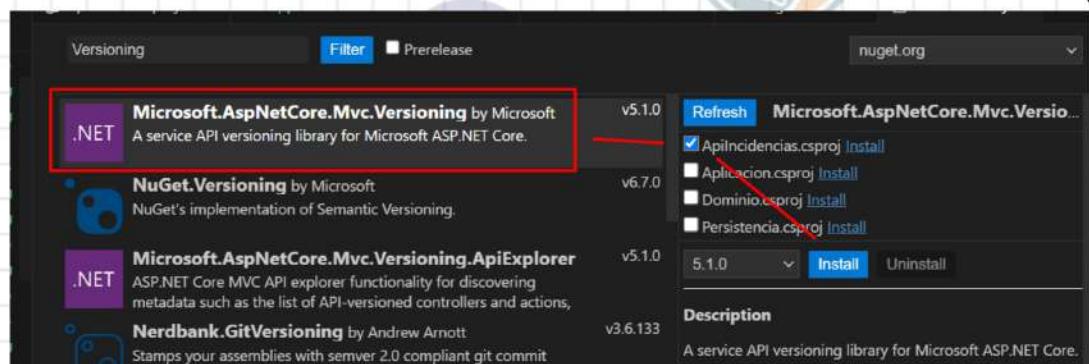
```
using System.Reflection;
using ApiIncidencias.Extensions;
using AspNetCoreRateLimit; -----
using Microsoft.EntityFrameworkCore;
using Persistencia;
```

```
builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
builder.Services.ConfigureRateLimiting(); -----
builder.Services.AddAutoMapper(Assembly.GetEntryAssembly());
builder.Services.ConfigureCors();
builder.Services.AddAplicacionServices();
```

```
app.UseHttpsRedirection();
app.UseIpRateLimiting(); -----
app.UseAuthorization();
```

Versionado API → Permite implementar cambios en el webAPI sin afectar el funcionamiento de las aplicaciones FrontEnd. Del lado del cliente. Para implementar el Versionado Siga los siguientes pasos:

1) Instale el Paquete De Versionado En El Proyecto WebAPI



2) Cree Un Método De Extensión Para El Versionado

```
public static void ConfigureApiVersioning(this IServiceCollection services)
{
    services.AddApiVersioning(options =>
    {
        });
}
```

3) Inyecte El Servicio En El Contenedor De Dependencias

Program.cs

```
12 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
13 builder.Services.AddEndpointsApiExplorer();
14 builder.Services.AddSwaggerGen();
15 builder.Services.ConfigureRateLimiting();
16 builder.Services.ConfigureApiVersioning(); ←
17 builder.Services.AddAutoMapper(Assembly.GetEntryAssembly());
18 builder.Services.ConfigureCors();
```

Inicie El Servidor y Realice Una Petición Get a la webApi:

```
1 {
2   "error": {
3     "code": "ApiVersionUnspecified",
4     "message": "An API version is required, but was not specified.",
5     "innerError": null
6   }
7 }
```

Como Se Puede Observar En La Imagen La Respuesta Por El Servidor Es Errada y Que No Se Ha Realizado La Especificación De Una Versión del API. Para Corregir Siga Los Siguientes Pasos.

Agregue las Siguientes Líneas de Código Al Método De Extensión Del Versionado.php

```
options.DefaultApiVersion = new ApiVersion(1, 0);
options.AssumeDefaultVersionWhenUnspecified = true;
```

Permite Especificar La Versión Del WebApi.

Permite tomar La Última Versión Del API En Caso De No Ser Especificada.

```
48 public static void ConfigureApiVersioning(this IServiceCollection services)
49 {
50     services.AddApiVersioning(options =>
51     {
52         options.DefaultApiVersion = new ApiVersion(1, 0);
53         options.AssumeDefaultVersionWhenUnspecified = true;
54     });
55 }
```

Reinicie El Servidor y Realice Nuevamente la Petición.

```
1 = [
2 = {
3 = "id": "001",
4 = "nombrePais": "Colombia"
5 = },
6 = {
7 = "id": "002",
8 = "nombrePais": "Portugal"
9 = },
10 = {
11 = "id": "003",
12 = "nombrePais": "Ecuador"
13 = }
14 = ]
```

Versionado Con Query String

Ej: Se Requiere Implementar Una Nueva Versión Del Listado De Paises Con Los Departamentos Hijos.

1) Genera Dto.

```
ApIIncidencias > Dtos > C# PaisxDepDto.cs > ...
1   namespace ApIIncidencias.Dtos;
2
3   0 references
4   public class PaisxDepDto
5   {
6       0 references
7       public string Id { get; set; }
8       0 references
9       public string NombrePais { get; set; }
10      0 references
11      public List<DepartamentoDto> departamentos { get; set; }
12  }
```

2) Implemente El Mappeo Del Dto.

```
ApIIncidencias > Profiles > C# MappingProfiles.cs > MappingProfiles > .ctor
7  {
8      0 references
9      public MappingProfiles(){}
10     CreateMap<Pais,PaisDto>()
11         .ReverseMap()
12         .ForMember(o => o.Departamentos,d => d.Ignore());
13
14     CreateMap<Departamento,DepartamentoDto>().ReverseMap();
15
16     CreateMap<Pais,PaisxDepDto>().ReverseMap();
```

3) Modifique El Controller De Pais Implementando La nueva Version Del Método Get Que Retorna todos los Paises.

```

ApilIncidencias > Controllers > C# PaisController.cs > PaisController > Get11
26     }*/  

27     [HttpGet]  

28     [MapToApiVersion("1.0")]
29     [ProducesResponseType(StatusCodes.Status200OK)]
30     [ProducesResponseType(StatusCodes.Status400BadRequest)]
31     0 references
32     public async Task<ActionResult<IEnumerable<PaisDto>>> Get()
33     {
34         var paises = await _unitOfWork.Paises.GetAllAsync();
35         return _mapper.Map<List<PaisDto>>(paises);

```

Permite Especificar A Que Versión Pertenece el EndPoint

```

36     [HttpGet]
37     [MapToApiVersion("1.1")]
38     [ProducesResponseType(StatusCodes.Status200OK)]
39     [ProducesResponseType(StatusCodes.Status400BadRequest)]
40     0 references
41     public async Task<ActionResult<IEnumerable<PaisxDepDto>>> Get11()
42     {
43         var paises = await _unitOfWork.Paises.GetAllAsync();
44         return _mapper.Map<List<PaisxDepDto>>(paises);

```



```

8     namespace ApiIncidencias.Controllers;
9     [ApiVersion("1.0")]
10    [ApiVersion("1.1")]
11    1 reference
12    public class PaisController : BaseApiController
13    {
14        private readonly IUnitOfWork _unitOfWork;
15        private readonly IMapper _mapper;

```

Se Deben Aplicar Las Anotaciones De Versión Antes Del Inicio De La Declaración De Clase



2) En El Método De Extension Del Versionado Realice Los Siguientes Cambios Para Configurar La Diferentes Versiones Soportadas.

```

public static void ConfigureApiVersioning(this IServiceCollection services)
{
    services.AddApiVersioning(options =>
    {
        options.DefaultApiVersion = new ApiVersion(1, 0);
        options.AssumeDefaultVersionWhenUnspecified = true;
        options.ApiVersionReader = new QueryStringApiVersionReader("ver");
    });
}

```

Variable Que Permitira Especificar La Versión A Utilizar En La Petición.

Probando En Insomnia & Postman.

```
GET http://localhost:5255/api/incidencias/Pais
200 OK
[{"id": "001", "nombrePais": "Colombia"}, {"id": "002", "nombrePais": "Portugal"}, {"id": "003", "nombrePais": "Ecuador"}]
```

Petición A Versión x Defecto.

```
GET http://localhost:5255/api/incidencias/Pais?ver=1.1
200 OK
[{"id": "001", "nombrePais": "Colombia", "departamentos": []}, {"id": "002", "nombrePais": "Portugal", "departamentos": []}, {"id": "003", "nombrePais": "Ecuador", "departamentos": []}]
```

Se Especifica La Version

```
GET http://localhost:5255/api/incidencias/Pais?ver=1.1
200 OK
[{"id": "001", "nombrePais": "Colombia", "departamentos": []}, {"id": "002", "nombrePais": "Portugal", "departamentos": []}, {"id": "003", "nombrePais": "Ecuador", "departamentos": []}]
```

Petición Especificando la VERSIÓN

http://localhost:5255/api/incidencias/Pais?ver=1.1

Especificación Del Versionado x Medio De Query String.

```
GET http://localhost:5255/api/incidencias/Pais?ver=1.1
200 OK
[{"id": "001", "nombrePais": "Colombia", "departamentos": [{"id": "001", "nombreDep": "Santander"}]}, {"id": "002", "nombrePais": "Portugal", "departamentos": []}, {"id": "003", "nombrePais": "Ecuador", "departamentos": []}]
```

Especificando Versión Header

1) Agregue El Siguiente Código En El Método De Extensión

```
55 options.ApiVersionReader = ApiVersionReader.Combine(
56     new QueryStringApiVersionReader("ver"),
57     new HeaderApiVersionReader("X-Version")
58 );
```

Este Código Permite Usar Query String o Headers para Especificar la versión de API

2) Asegúrese de Definir El Encabezado En Insomnia o Postman

```
GET ▼ http://localhost:5255/api/incidencias/Pais
Send ▼ 200 OK 39.8 ms 111 B
Body ▼ Auth ▼ Query Headers ▼ Docs
Add Delete All Toggle Description
User-Agent Insomnia/2023.5.5
X-Version 1.0
Preview ▼ Headers ▼ Cookies Timeline
1 - [
2 - {
3 -   "id": "ee1",
4 -   "nombrePais": "Colombia"
5 - },
6 - {
7 -   "id": "ee2",
8 -   "nombrePais": "Portugal"
9 - },
10 - {
11 -   "id": "ee3",
12 -   "nombrePais": "Ecuador"
13 - }
14 ]
```

Cambiar Formato De Respuesta

1) Sobreescibir En El Header De Insomnia o Postman
El Atributo Accept

```
GET ▼ http://localhost:5255/api/incidencias/Pais
Send ▼
Body ▼ Auth ▼ Query Headers ▼ Docs
Add Delete All Toggle Description
User-Agent Insomnia/2023.5.5
X-Version 1.0
Accept application/xml
Preview ▼ Headers ▼ Cookies Timeline
1 - [
2 - {
3 -   "id": "ee1",
4 -   "nombrePais": "Colombia"
5 - },
6 - {
7 -   "id": "ee2",
8 -   "nombrePais": "Portugal"
9 - },
10 - {
11 -   "id": "ee3",
12 -   "nombrePais": "Ecuador"
13 - }
14 ]
```

2) Modificar El Contenedor De Dependencias Program.cs
Segun Imagen De Referencia.

```
Aplicaciones > C# Program.cs
8
9 // Add services to the container.
10
11 builder.Services.AddControllers(options =>
12 {
13     options.RespectBrowserAcceptHeader = true;
14 }).AddXmlSerializerFormatters();
15 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
16 builder.Services.AddEndpointsApiExplorer();
17 builder.Services.AddSwaggerGen();
```

GET ▾ http://localhost:5255/api/incidencias/Pais

Send ▾ 200 OK 1.21 s 323 B

Body ▾	Auth ▾	Query	Headers ▾ 3	Docs
Add Delete All Toggle Description				
User-Agent	Insomnia/2023.5.5			
X-Version	1.0			
Accept	application/xml			

Preview ▾ Headers ▾ Cookies Timeline

```

1 <ArrayOfPaisDto
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
4     <PaisDto>
5       <Id>001</Id>
6       <NombrePais>Colombia</NombrePais>
7     </PaisDto>
8     <PaisDto>
9       <Id>002</Id>
10      <NombrePais>Portugal</NombrePais>
11     </PaisDto>
12     <PaisDto>
13       <Id>003</Id>
14       <NombrePais>Ecuador</NombrePais>
15     </PaisDto>
16   </ArrayOfPaisDto>

```

Nota → Si Se Desea Enviar Un Mensaje De Error Indicando Que El Formato Del Accept no Es Soportado Agregue La linea de Código Como Se Indica En La Imagen Inferior.

```

11 builder.Services.AddControllers(options =>
12 {
13     options.RespectBrowserAcceptHeader = true;
14     options.ReturnHttpNotAcceptable=true; ←
15 }).AddXmlSerializerFormatters();

```

A Continuación se listan Algunos Valores Usados en Accept.

- `text/html`: Acepta contenido HTML.
- `application/json`: Acepta contenido en formato JSON.
- `application/xml`: Acepta contenido en formato XML.
- `text/plain`: Acepta contenido de texto sin formato.
- `image/jpeg`: Acepta contenido de imagen en formato JPEG.
- `image/png`: Acepta contenido de imagen en formato PNG.
- `application/pdf`: Acepta contenido en formato PDF.
- `application/octet-stream`: Acepta cualquier tipo de contenido binario.
- `application/xhtml+xml`: Acepta contenido en formato XHTML.
- `audio/mpeg`: Acepta contenido de audio en formato MP3.
- `video/mp4`: Acepta contenido de video en formato MP4.
- `application/javascript`: Acepta contenido JavaScript.
- `text/css`: Acepta contenido de hoja de estilos en cascada (CSS).
- `application/vnd.openxmlformats-officedocument.wordprocessingml.document`: Acepta contenido de documento de Microsoft Word en formato Office Open XML.
- `application/vnd.openxmlformats-officedocument.spreadsheetml.sheet`: Acepta contenido de hoja de cálculo de Microsoft Excel en formato Office Open XML.
- `application/vnd.openxmlformats-officedocument.presentationml.presentation`: Acepta contenido de presentación de Microsoft PowerPoint en formato Office Open XML.

`text/html`: Indica que el cliente acepta contenido HTML, que es el formato estándar para mostrar páginas web en un navegador.
 `application/json`: Indica que el cliente acepta contenido en formato JSON, que es ampliamente utilizado para intercambiar datos estructurados entre el cliente y el servidor.
 `application/xml`: Indica que el cliente acepta contenido en formato XML, que también se utiliza para intercambiar datos estructurados.
 `text/plain`: Indica que el cliente acepta contenido de texto sin formato.

<https://learn.microsoft.com/en-us/aspnet/core/web-api/advanced/custom-formatters?view=aspnetcore-6.0>

Paginación

- 1) Cree Una Carpeta Llamada **Helpers** en El Proyecto **Laravel**
- 2) Cree Una Clase En **Helpers** y Llámela **Pager**. Modifique la clase Teniendo En Cuenta La Siguiente Imagen.

```
ApiIncidencias > Helpers > C# Pager.cs > ...
1  namespace ApiIncidencias.Helpers;
2
3      1 reference
4  public class Pager<T> where T : class
5  {
6      1 reference
7      public string Search { get; private set; }
8      3 references
9      public int PageIndex { get; private set; }
10     2 references
11     public int PageSize { get; private set; }
12     2 references
13     public int Total { get; private set; }
```

```

9     public IEnumerable<T> Registers { get; private set; }
10
11     0 references
12     public Pager(IEnumerable<T> registers, int total, int pageIndex,
13                 int pageSize, string search)
14     {
15         Registers = registers;
16         Total = total;
17        PageIndex = pageIndex;
18         PageSize = pageSize;
19         Search = search;
20     }
21
22     1 reference
23     public int TotalPages
24     {
25         get
26         {
27             return (int)Math.Ceiling(Total / (double)PageSize);
28         }
29     }
30
31     0 references
32     public bool HasPreviousPage
33     {
34         get
35         {
36             return (PageIndex > 1);
37         }
38     }
39
40     0 references
41     public bool HasNextPage
42     {
43         get
44         {
45             return (PageIndex < TotalPages);
46         }
47     }

```

3) Cree Una Nueva Clase En Helpers y llamela Params.cs y modifiquela teniendo en cuenta la Siguiente Imagen.

```

1  namespace ApiIncidencias.Helpers;
2
3      0 references
4  public class Params
5  {
6      private int _pageSize = 5;
7      private const int MaxPageSize = 50;
8      private int _pageIndex = 1;
9      private string _search;
10     0 references
11     public int PageSize
12     {
13         get => _pageSize;
14         set => _pageSize = (value > MaxPageSize) ? MaxPageSize : value;
15     }
16
17     0 references
18     public int pageIndex
19     {
20         get => _pageIndex;
21         set => _pageIndex = (value <= 0) ? 1 : value;
22     }
23
24     0 references
25     public string Search
26     {
27         get => _search;
28         set => _search = (!String.IsNullOrEmpty(value)) ? value.ToLower() : "";
29     }
30 }
```

4) En la Interfaz Generica Agregue El Siguiente Código

C# IGenericRepository.cs

```

Dominio > Interfaces > C# IGenericRepository.cs > IGenericRepository
1  using System.Linq.Expressions;
2  namespace Dominio.Interfaces;
3  public interface IGenericRepository<T> where T : BaseEntity
4  {
5      Task<T> GetByIdAsync(string id);
6      Task<IEnumerable<T>> GetAllAsync();
7      IEnumerable<T> Find(Expression<Func<T, bool>> expression);
8      Task<(int totalRegistros, IEnumerable<T> registros)> GetAllAsync(int pageIndex, int pageSize, string search);
9      void Add(T entity);
10     void AddRange(IEnumerable<T> entities);
11     void Remove(T entity);
12     void RemoveRange(IEnumerable<T> entities);
13     void Update(T entity);
14 }
```

5) Implemente El Nuevo Metodo En El Repository Generico

```
1 reference
67     public virtual async Task<(int totalRegistros, IEnumerable<T> registros)> GetAllAsync(int pageIndex, int pageSize, string _search)
68     {
69         var totalRegistros = await _context.Set<T>().CountAsync();
70         var registros = await _context.Set<T>()
71             .Skip((pageIndex - 1) * pageSize)
72             .Take(pageSize)
73             .ToListAsync();
74         return (totalRegistros, registros);
75     }
```

6) Modifique El Controlador De Pais En La Peticion Get De La Version 1.1 Como Se Muestra En La Siguiente Imagen

Aplicaciones > Controllers > C# PaisController.cs > PaisController > Get11

```
39     [HttpGet]
40     [MapToApiVersion("1.1")]
41     [ProducesResponseType(StatusCodes.Status200OK)]
42     [ProducesResponseType(StatusCodes.Status400BadRequest)]
43     0 references
44     public async Task<ActionResult<Pager<PaisxDepDto>>> Get11([FromQuery] Params paisParams)
45     {
46         var pais = await _unitOfWork.Paises.GetAllAsync(paisParams.PageIndex,paisParams.PageSize,paisParams.Search);
47         var lstPaisesDto = _mapper.Map<List<PaisxDepDto>>(pais.registros);
48         return new Pager<PaisxDepDto>(lstPaisesDto,pais.totalRegistros,paisParams.PageIndex,paisParams.PageSize,paisParams.Search);
```

7) En PaisRepository Sobrecargar El Metodo

```
22     public override async Task<(int totalRegistros, IEnumerable<Pais> registros)> GetAllAsync(int pageIndex, int pageSize, string search)
23     {
24         var query = _context.Paises as IQueryable<Pais>;
25         if (!string.IsNullOrEmpty(search))
26         {
27             query = query.Where(p => p.NombrePais.ToLower().Contains(search));
28         }
29         var totalRegistros = await query.CountAsync();
30         var registros = await query
31             .Include(u => u.Departamentos)
32             .Skip((pageIndex - 1) * pageSize)
33             .Take(pageSize)
34             .ToListAsync();
35
36         return (totalRegistros, registros);
37     }
```

8) Probar En Postman o Insomnia Los Cambios Realizados

GET <http://localhost:5255/api/incidencias/Pais>

Body ▾ Auth ▾ Query Headers Docs

Add Delete All Toggle Description

User-Agent: Insomnia/2023.5.5

X-Version: 1.1

Accept: application/xml

```
1: {
2:   "search": null,
3:   "pageIndex": 1,
4:   "pageSize": 5,
5:   "total": 3,
6:   "registros": [
7:     {
8:       "id": "001",
9:       "nombrePais": "Colombia",
10:      "departamentos": [
11:        {
12:          "id": "001",
13:          "nombreDep": "Santander"
14:        }
15:      ],
16:      "totalPages": 1,
17:      "hasPreviousPage": false,
18:      "hasNextPage": false
19:    },
20:    {
21:      "id": "002",
22:      "nombrePais": "Portugal",
23:      "departamentos": []
24:    },
25:    {
26:      "id": "003",
27:      "nombrePais": "Ecuador",
28:      "departamentos": []
29:    }
30  ]
```

Estableciendo Query Params

1) En Insomnia Ingresar A La Pestaña Query y Crear Los Parámetros **pageIndex** y **pageSize**

GET ▾ http://localhost:5255/api/incidencias/Pais

Body ▾ Auth ▾ Query ▾ Headers ▾ Docs

URL PREVIEW

http://localhost:5255/api/incidencias/Pais?pageIndex=1&pageSize=1

Add Delete All Toggle Description

pageIndex 1

pageSize 1

```
1 * {
2   "search": null,
3   "pageIndex": 1,
4   "pageSize": 1,
5   "total": 3,
6   "registers": [
7     {
8       "id": "001",
9       "nombrePais": "Colombia",
10      "departamentos": [
11        {
12          "id": "001",
13          "nombreDep": "Santander"
14        }
15      ]
16    },
17    "totalPages": 3,
18    "hasPreviousPage": false,
19    "hasNextPage": true
20  }
```

GET ▾ http://localhost:5255/api/incidencias/Pais

Body ▾ Auth ▾ Query ▾ Headers ▾ Docs

URL PREVIEW

http://localhost:5255/api/incidencias/Pais?pageIndex=2&pageSize=1

Add Delete All Toggle Description

pageIndex 2

pageSize 1

```
1 * {
2   "search": null,
3   "pageIndex": 2,
4   "pageSize": 1,
5   "total": 3,
6   "registers": [
7     {
8       "id": "003",
9       "nombrePais": "Ecuador",
10      "departamentos": []
11    }
12  ],
13  "totalPages": 3,
14  "hasPreviousPage": true,
15  "hasNextPage": true
16 }
```

2) Aplicando Busqueda.

a) En Postman o Insomnia Agregar El Parámetro De Search

Se Debe Deshabilitar los
Parametros de pageIndex y PageSize.

```
1: {
2:   "search": "colombia",
3:   "pageIndex": 1,
4:   "pageSize": 5,
5:   "total": 1,
6:   "registers": [
7:     {
8:       "id": "001",
9:       "nombrePais": "Colombia",
10:      "departamentos": [
11:        {
12:          "id": "001",
13:          "nombreDep": "Santander"
14:        }
15:      ],
16:    },
17:    {
18:      "totalPages": 1,
19:      "hasPreviousPage": false,
20:      "hasNextPage": false
21:    }
}
```

Autenticación Jwt

JWT (JSON Web Token) es un estándar abierto (RFC 7519) que define un formato compacto y autónomo para transmitir información entre partes de manera segura como un objeto JSON. Está diseñado especialmente para ser utilizado en aplicaciones web y servicios API, y es ampliamente utilizado para implementar autenticación y autorización en sistemas distribuidos.

Un token JWT está compuesto por tres partes separadas por puntos:

1. **Header:** Contiene el tipo de token (JWT) y el algoritmo de firma utilizado.
2. **Payload:** Contiene las declaraciones (claims) del token. Estas declaraciones representan información sobre el usuario, roles, permisos, caducidad, etc.
3. **Signature:** Es la parte que verifica que el token no haya sido modificado en el camino. Se crea utilizando el header, payload, una clave secreta y el algoritmo especificado en el header.

Para El Caso Práctico Se Crearan Una Entidad Llamada rol y se agregar nuevos campos a Persona.

Dominio > C# Persona.cs > Persona > PersonaRoles

```
1 namespace Dominio;
2
3 public class Persona : BaseEntity
4 {
5
6     1 reference
7     public string NombrePersona { get; set; }
8     0 references
9     public string ApellidoPaterno { get; set; }
10    0 references
11    public string ApellidoMaterno { get; set; }
12    0 references
13    public string Username { get; set; }
14    0 references
15    public string Email { get; set; }
16    0 references
17    public string Password { get; set; }
18    1 reference
19    public int IdGeneroFk { get; set; }
20    1 reference
21    public Genero Genero { get; set; }
22
23     public Ciudad Ciudad { get; set; }
24     1 reference
25     public int IdTipoPerFk { get; set; }
26     1 reference
27     public TipoPersona TipoPersona { get; set; }
28     0 references
29     public ICollection<Matricula> Matriculas { get; set; }
30     1 reference
31     public ICollection<TrainerSalon> TrainerSalones { get; set; }
32     1 reference
33     public ICollection<Rol> Roles { get; set; } = new HashSet<Rol>();
34     1 reference
35     public ICollection<PersonaRoles> PersonaRoles { get; set; }
36 }
```

Campos Nuevos.

clase Para obtener una Colección de Elementos únicos Sin Duplicados.

```
1 namespace Dominio;
2
3 public class Rol : BaseEntityA
4 {
5     public string Nombre { get; set; }
6     public ICollection<Persona> Personas { get; set; } = new HashSet<Persona>();
7     public ICollection<PersonaRoles> PersonaRoles { get; set; }
8 }
```

Dominio > C# PersonaRoles.cs ...

```
1 namespace Dominio;
2
3 public class PersonaRoles
4 {
5     2 references
6     public string UsuarioId { get; set; }
7     1 reference
8     public Persona Persona { get; set; }
9     2 references
10    public int RolId { get; set; }
11    1 reference
12    public Rol Rol { get; set; }
13 }
```

Entidad Intermedia Entre Persona y Rol.

Genere los Archivos De Configuración Para Usuarios y Roles.

Persistencia > Data > Configuration > C# RolConfiguration.cs > ...

```
1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4
5  namespace Persistencia.Data.Configuration;
6
7  public class RolConfiguration : IEntityTypeConfiguration<Rol>
8  {
9      public void Configure(EntityTypeBuilder<Rol> builder)
10     {
11         builder.ToTable("Rol");
12         builder.Property(p => p.Id)
13             .IsRequired();
14         builder.Property(p => p.Nombre)
15             .IsRequired()
16             .HasMaxLength(200);
17     }
18 }
```

Persistencia > Data > Configuration > C# PersonaConfiguration.cs > C# PersonaConfiguration > Configure

```
1  using Dominio;
2  using Microsoft.EntityFrameworkCore;
3  using Microsoft.EntityFrameworkCore.Metadata.Builders;
4  namespace Persistencia.Data.Configuration
5  {
6      public class PersonaConfiguration : IEntityTypeConfiguration<Personas>
7  {
8      public void Configure(EntityTypeBuilder<Persona> builder)
9  {
10         // Aquí puedes configurar las propiedades de la entidad Marca
11         // utilizando el objeto 'builder'.
12         builder.ToTable("persona");
13
14         builder.HasKey(e => e.Id);
15         builder.Property(e => e.Id)
16             .HasMaxLength(20);
17     }
18     builder.Property(p => p.NombrePersona)
19         .IsRequired()
20         .HasMaxLength(50);
21
22     builder.HasOne(p => p.Genero)
23         .WithMany(p => p.Personas)
24         .HasForeignKey(p => p.IdGeneroFk);
25
26     builder.HasOne(p => p.Ciudad)
27         .WithMany(p => p.Personas)
28         .HasForeignKey(p => p.IdCiudadFk);
29
30     builder.HasOne(p => p.TipoPersona)
31         .WithMany(p => p.Personas)
32         .HasForeignKey(p => p.IdTipoPerFk);
33
34     builder
35         .HasMany(p => p.Roles)
36         .WithMany(p => p.Personas)
37         .UsingEntity<PersonasRoles>(
38             j => j
39                 .HasOne(pt => pt.Rol)
40                     .WithMany(t => t.PersonasRoles)
41                     .HasForeignKey(pt => pt.RolId),
42             j => j
43                 .HasOne(pt => pt.Persona)
44                     .WithMany(p => p.PersonasRoles)
45                     .HasForeignKey(pt => pt.UsuarioId),
46             j =>
47             {
48                 j.HasKey(t => new { t.UsuarioId, t.RolId });
49             }
50         );
51     }
52 }
```

Define La PK Compu -
Esta en la Union

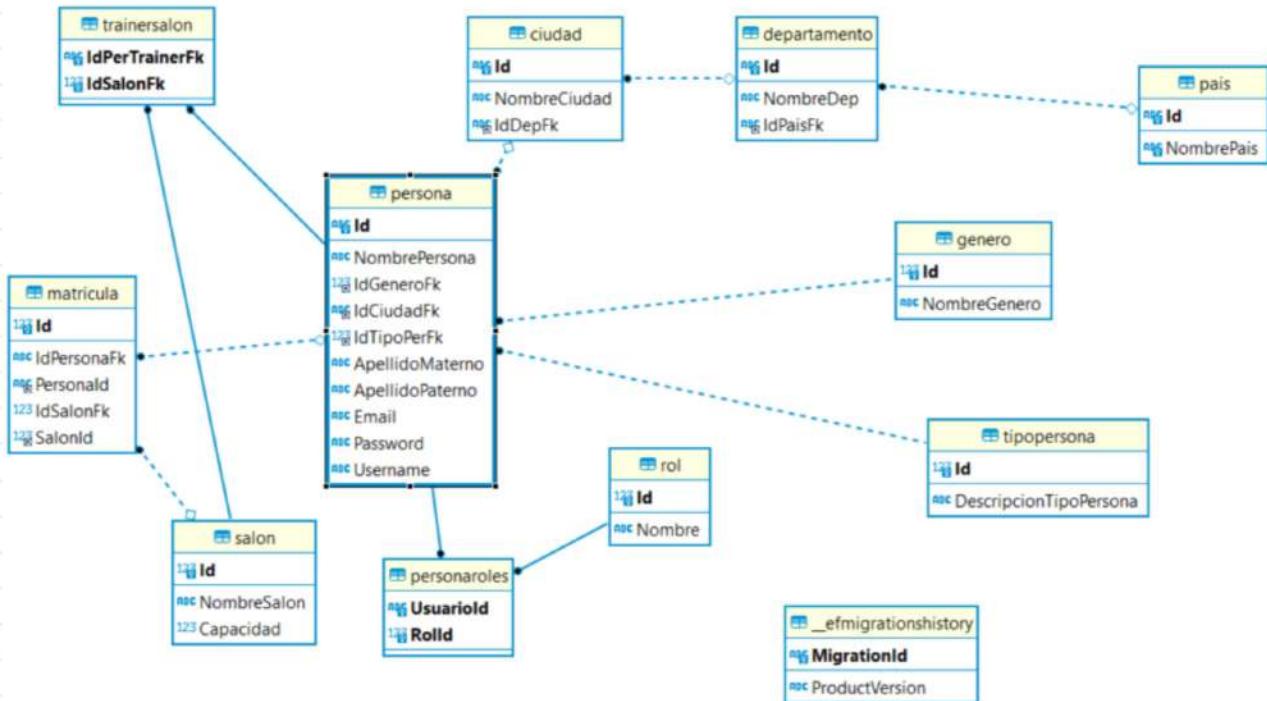
PersonasRoles

Permite Establecer la Rela-
ción Muchos A Muchos Entre
Las Entidades Usuarios Y
Roles. Dentro de)

.UsingEntity<UsuariosRoles>

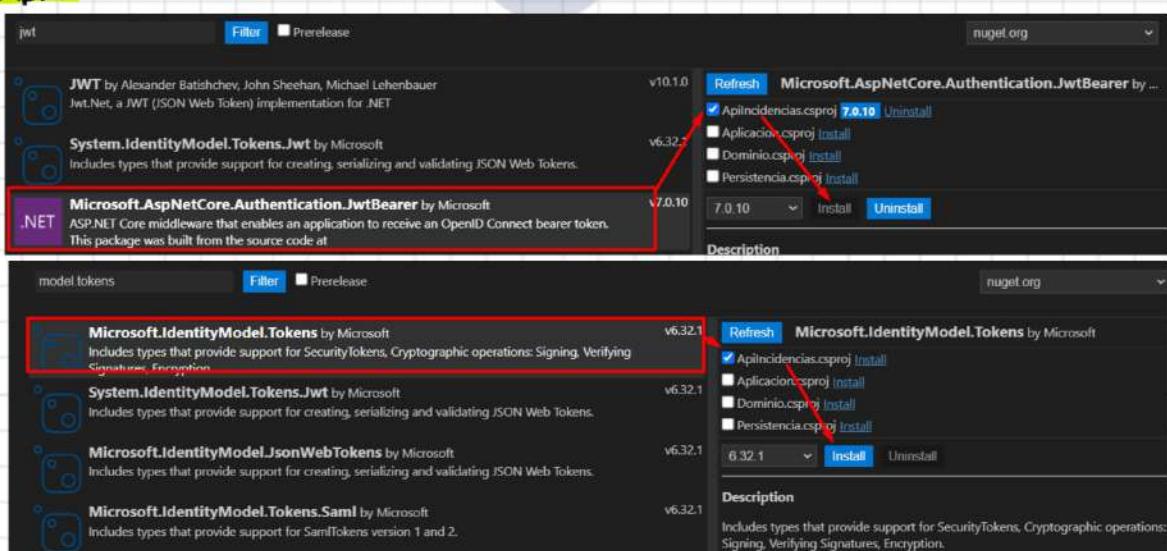
se Realiza la Configuración
de las Claves Foraneas.

Genere La migración y Actualice la Estructura De la Base De Datos.



• Genere La Interfaz De Repositorio Generico de Rol e implemente Los Metodos Necesarios; Haga lo mismo para Persona.

Para la implementacion del JWT es necesario tener instalado los siguientes paquetes en el proyecto WebApi :



El siguiente paso a seguir realizar la siguiente configuración en el archivo appsettings.json

The screenshot shows the file structure of a .NET Core project named 'INCIDENCIAS-APP'. The 'AppSettings.json' file is open, showing its contents. A red box highlights the 'JWT' section. Four arrows point from the right side of the slide to specific parts of this section:

- Cadena secreta: Points to the 'Key' field.
- Emisor del token: Points to the 'Issuer' field.
- Destinatarios: Points to the 'Audience' field.
- Tiempo de duracion: Points to the 'DurationInMinutes' field.

```
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    "JWT": {
14      "Key": "rgfZs3pNboV0hbg6Fat",
15      "Issuer": "ApiIncidencias",
16      "Audience": "ApiIncidencias",
17      "DurationInMinutes": 20
18    }
19 }
```

Crear una nueva clase para Mapear los atributos del JWT dentro de la carpeta helpers

```
1  namespace ApiIncidencias.Helpers;
2
3  public class JWT
4  {
5    public string HasKey { get; set; }
6    public string Issuer { get; set; }
7    public string Audience { get; set; }
8    public double DurationInMinutes { get; set; }
9 }
```

Crear una nueva carpeta en el proyecto WebApi y dentro de la carpeta crear una interface llamada IUsuarioService y crear la implementacion de la interface en la misma carpeta.

```
1  using ApiIncidencias.Dtos;
2
3  namespace ApiIncidencias.Services;
4
5  public interface IUserService
6  {
7    Task<string> RegisterAsync(RegisterDto model);
8    Task<DatosUsuarioDto> GetTokenAsync(LoginDto model);
9
10   Task<string> AddRoleAsync(AddRoleDto model);
11 }
```

ApiIncidentes > Services > C# UserService.cs > RegisterAsync

```

1  using System.IdentityModel.Tokens.Jwt;
2  using System.Security.Claims;
3  using System.Text;
4  using ApiIncidentes.Dtos;
5  using ApiIncidentes.Helpers;
6  using Dominio;
7  using Dominio.Interfaces;
8  using Microsoft.AspNetCore.Identity;
9  using Microsoft.Extensions.Options;
10 using Microsoft.IdentityModel.Tokens;
11
12 namespace ApiIncidentes.Services;
13
14 public class UserService : IUserService
15 {
16     private readonly JWT _jwt;
17     private readonly IUnitOfWork _unitOfWork;
18     private readonly IPasswordHasher<Persona> _passwordHasher;
19
20     public UserService(IUnitOfWork unitOfWork, IOptions<JWT> jwt,
21         IPasswordHasher<Persona> passwordHasher)
22     {
23         _jwt = jwt.Value;
24         _unitOfWork = unitOfWork;
25         _passwordHasher = passwordHasher;
26     }
27
28     public async Task<string> RegisterAsync(RegisterDto registerDto)
29     {
30         var persona = new Persona
31         {
32             Id = registerDto.Id,
33             NombrePersona = registerDto.Nombres,
34             ApellidoMaterno = registerDto.ApellidoMaterno,
35             ApellidoPaterno = registerDto.ApellidoPaterno,
36             Email = registerDto.Email,
37             Username = registerDto.Username,
38             IdGeneroFk = registerDto.IdGeneroFk,
39             IdCiudadFk = registerDto.IdCiudadFk,
40             IdTipoPerFk = registerDto.IdTipoPerFk,
41         };
42
43         persona.Password = _passwordHasher.HashPassword(persona, registerDto.Password);
44
45         var usuarioExiste = _unitOfWork.Personas
46             .Find(u => u.Username.ToLower() == registerDto.Username.ToLower())
47             .FirstOrDefault();
48
49         if (usuarioExiste == null)
50         {
51             var rolPredeterminado = _unitOfWork.Roles
52                 .Find(u => u.Nombre == Autorizacion.rol_predeterminado.ToString())
53                 .First();
54
55             try
56             {
57                 persona.Roles.Add(rolPredeterminado);
58                 _unitOfWork.Personas.Add(persona);
59                 await _unitOfWork.SaveAsync();
59
60                 return $"El usuario {registerDto.Username} ha sido registrado exitosamente";
61             }
62             catch (Exception ex)
63             {

```

```

63             var message = ex.Message;
64             return $"Error: {message}";
65         }
66     }
67     else
68     {
69         return $"El usuario con {registerDto.Username} ya se encuentra registrado.";
70     }
71 }
72
73
74     2 references
75     public async Task<DatosUsuarioDto> GetTokenAsync(LoginDto model)
76     {
77         DatosUsuarioDto datosUsuarioDto = new DatosUsuarioDto();
78         var usuario = await _unitOfWork.Personas
79                         .GetByUsernameAsync(model.Username);
80
81         if (usuario == null)
82         {
83             datosUsuarioDto.EstaAutenticado = false;
84             datosUsuarioDto.Mensaje = $"No existe ningún usuario con el username {model.Username}.";
85             return datosUsuarioDto;
86         }
87
88         var resultado = _passwordHasher.VerifyHashedPassword(usuario, usuario.Password, model.Password);
89
90         if (resultado == PasswordVerificationResult.Success)
91         {
92             datosUsuarioDto.EstaAutenticado = true;
93             JwtSecurityToken jwtSecurityToken = CreateJwtToken(usuario);
94             datosUsuarioDto.Token = new JwtSecurityTokenHandler().WriteToken(jwtSecurityToken);
95             datosUsuarioDto.Email = usuario.Email;
96             datosUsuarioDto.UserName = usuario.Username;
97             datosUsuarioDto.Roles = usuario.Roles
98                             .Select(u => u.Nombre)
99                             .ToList();
100            return datosUsuarioDto;
101        }
102        datosUsuarioDto.EstaAutenticado = false;
103        datosUsuarioDto.Mensaje = $"Credenciales incorrectas para el usuario {usuario.Username}.";
104        return datosUsuarioDto;
105    }
106
107
108
109     2 references
110     public async Task<string> AddRoleAsync(AddRoleDto model)
111     {
112
113         var usuario = await _unitOfWork.Personas
114                         .GetByUsernameAsync(model.Username);
115
116         if (usuario == null)
117         {
118             return $"No existe algún usuario registrado con la cuenta {model.Username}.";
119         }
120
121         var resultado = _passwordHasher.VerifyHashedPassword(usuario, usuario.Password, model.Password);
122
123         if (resultado == PasswordVerificationResult.Success)
124         {

```

```

123
124     var rolExiste = _unitOfWork.Roles
125         .Find(u => u.Nombre.ToLower() == model.Role.ToLower())
126         .FirstOrDefault();
127
128     if (rolExiste != null)
129     {
130         var usuarioTieneRol = usuario.Roles
131             .Any(u => u.Id == rolExiste.Id);
132
133         if (usuarioTieneRol == false)
134         {
135             usuario.Roles.Add(rolExiste);
136             _unitOfWork.Personas.Update(usuario);
137             await _unitOfWork.SaveAsync();
138         }
139
140         return $"Rol {model.Role} agregado a la cuenta {model.Username} de forma exitosa.";
141     }
142
143     return $"Rol {model.Role} no encontrado.";
144 }
145
146 }
147
1 reference
148 private JwtSecurityToken CreateJwtToken(Persona usuario)
149 {
150     var roles = usuario.Roles;
151     var roleClaims = new List<Claim>();
152     foreach (var role in roles)
153     {
154         roleClaims.Add(new Claim("roles", role.Nombre));
155     }
156     var claims = new[]
157     {
158         new Claim(JwtRegisteredClaimNames.Sub, usuario.Username),
159         new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString()),
160         new Claim(JwtRegisteredClaimNames.Email, usuario.Email),
161         new Claim("uid", usuario.Id.ToString())
162     }
163     .Union(roleClaims);
164     var symmetricSecurityKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_jwt.Key));
165     var signingCredentials = new SigningCredentials(symmetricSecurityKey, SecurityAlgorithms.HmacSha256);
166     var jwtSecurityToken = new JwtSecurityToken(
167         issuer: _jwt.Issuer,
168         audience: _jwt.Audience,
169         claims: claims,
170         expires: DateTime.UtcNow.AddMinutes(_jwt.DurationInMinutes),
171         signingCredentials: signingCredentials);
172     return jwtSecurityToken;
173 }
174 }
```

UserService es responsable de las funciones relacionadas con la gestión de usuarios en una aplicación.

Explicación Código Fuente

Variables Miembro

_jwt: Almacena configuraciones relacionadas con JWT.

_unitOfWork: Proporciona métodos para interactuar con la base de datos.

_passwordHasher: Usado para hashear contraseñas y verificar contraseñas hasheadas.

Constructor → Se encarga de inicializar las variables Miembro a través de la Inyección de Dependencias.

Método RegisterAsync

 Toma como argumento un **DTO (Data Transfer Object)** que contiene la información del usuario a registrar.

 Crea una nueva instancia de **Persona** con la información proporcionada.

 Hashea la contraseña proporcionada y la asigna a la instancia **Persona**.

 Verifica si el usuario ya existe en la base de datos a través del nombre de usuario.

 Si el usuario no existe, le asigna un rol predeterminado y guarda el usuario en la base de datos.

 Devuelve un mensaje según si la operación fue exitosa o no.

Método GetTokenAsync

 Toma como entrada un DTO con información de inicio de sesión.

 Verifica si el usuario existe y compara la contraseña proporcionada con la contraseña hasheada almacenada.

 Si las credenciales son correctas, crea un token JWT para el usuario y establece información relevante en un DTO DATOSUSUARIODTO.



Devuelve el DTO.

Método AddRoleAsync

 Acepta un DTO que contiene información del usuario y el rol que se desea agregar.

 Verifica si el usuario existe y compara la contraseña proporcionada.

 Si las credenciales son correctas y el rol existe, añade el rol al usuario y actualiza la base de datos.

 Devuelve un mensaje según si la operación fue exitosa o no.

Método CreateJWTToken

 Se utiliza para crear tokens JWT.

 Toma un objeto Persona como entrada y, basándose en su información y roles, construye un token JWT.

 Devuelve el token JWT.

Clase Autorización → Permite Especificar Por Medio De Una Estructura Enum Los Roles Soportado. En El WebAPI:

```
Aplicaciones > Helpers > C# Autorizacion.cs > ...
1  namespace ApiAplicaciones.Helpers;
2
3      1 reference
4  public class Autorizacion
5  {
6      2 references
7          public enum Roles
8          {
9              Administrador,
10             Gerente,
11             Empleado
12         }
13     public const Roles rol_predeterminado = Roles.Empleado;
```

Controlador Registro Usuarios

```
ApiIncidencias > Controllers > C# UsuariosController.cs > UsuariosController > _userService
1  using ApiIncidencias.Dtos;
2  using ApiIncidencias.Services;
3  using Microsoft.AspNetCore.Mvc;
4
5  namespace ApiIncidencias.Controllers;
6
7  1 reference
8  public class UsuariosController : BaseApiController
9  {
10    4 references
11    private readonly IUserService _userService;
12
13    0 references
14    public UsuariosController(IUserService userService)
15    {
16      _userService = userService;
17    }
18    [HttpPost("register")]
19    0 references
20    public async Task<ActionResult> RegisterAsync(RegisterDto model)
21    {
22      var result = await _userService.RegisterAsync(model);
23      return Ok(result);
24    }
25
26    [HttpPost("token")]
27    0 references
28    public async Task<IActionResult> GetTokenAsync(LoginDto model)
29    {
30      var result = await _userService.GetTokenAsync(model);
31      return Ok(result);
32    }
33
34    [HttpPost("addrole")]
35    0 references
36    public async Task<IActionResult> AddRoleAsync(AddRoleDto model)
37    {
38      var result = await _userService.AddRoleAsync(model);
39      return Ok(result);
40    }
41  }
```

Definición del controlador:

public class UsuariosController : BaseApiController: Aquí se declara la clase UsuariosController que hereda de BaseApiController. Los controladores son clases en ASP.NET Core que manejan las solicitudes HTTP y generan respuestas.

Constructor y servicios:

private readonly IUserService _userService;: Declaración de un campo privado llamado _userService que hace referencia a una instancia de IUserService, que generalmente es una interfaz que define métodos relacionados con la gestión de usuarios.

public UsuariosController(IUserService userService): Constructor del controlador que toma una instancia de IUserService como parámetro. Esto permite la inyección de dependencias, donde se proporciona una implementación concreta de IUserService cuando se crea una instancia de este controlador.

Implementación de los métodos:

async Task<ActionResult> RegisterAsync(RegisterDto model):
Este método asincrónico toma un objeto RegisterDto como parámetro. Se espera que este método registre un nuevo usuario y devuelva un ActionResult. El resultado exitoso (OK) generalmente contiene algún tipo de resultado o mensaje.

async Task<IActionResult> GetTokenAsync(LoginDto model):
Similar al anterior, este método toma un objeto LoginDto y genera un token para autenticación.

async Task<IActionResult> AddRoleAsync(AddRoleDto model):
Similar al anterior, este método toma un objeto AddRoleDto y agrega un rol al usuario.

Registrando Usuarios

1) En Postman, Insomnia o Thunder Client Generar una nueva Petición de tipo POST.

POST ▾ http://localhost:5255/api/incidencias/Usuarios/register

Send ▾

JSON ▾ Auth ▾ Query Headers 2 Docs

```

1  {
2      "Id": "123456",
3      "Nombres": "Johlver José",
4      "ApellidoPaterno": "Pardo",
5      "ApellidoMaterno": "Garcia",
6      "Email": "jjpardo2002@gmail.com",
7      "Username": "jjpardo2002@gmail.com",
8      "Password": "jjpsrdo2002",
9      "IdGeneroFk": "1",
10     "IdCiudadFk": "001",
11     "IdTipoPerFk": "2"
12 }

```

Obtener token

