

TECNOLOGIA EN DESARROLLO DE SOFTWARE

PROGRAMACION DE SOFTWARE

580304006-6 (6-8 AM)

Entrega final 20%

ITM 2024

Grupo: Mateo Vallejo Álvarez, Samuel García

Docente: Andrés Felipe Albanes Gaviria

Proyecto: Localización, Tema No° [4]

I. Descripción del proyecto.

A. Ambiente de desarrollo

IDE: Visual Studio 2022 (Community) Microsoft.

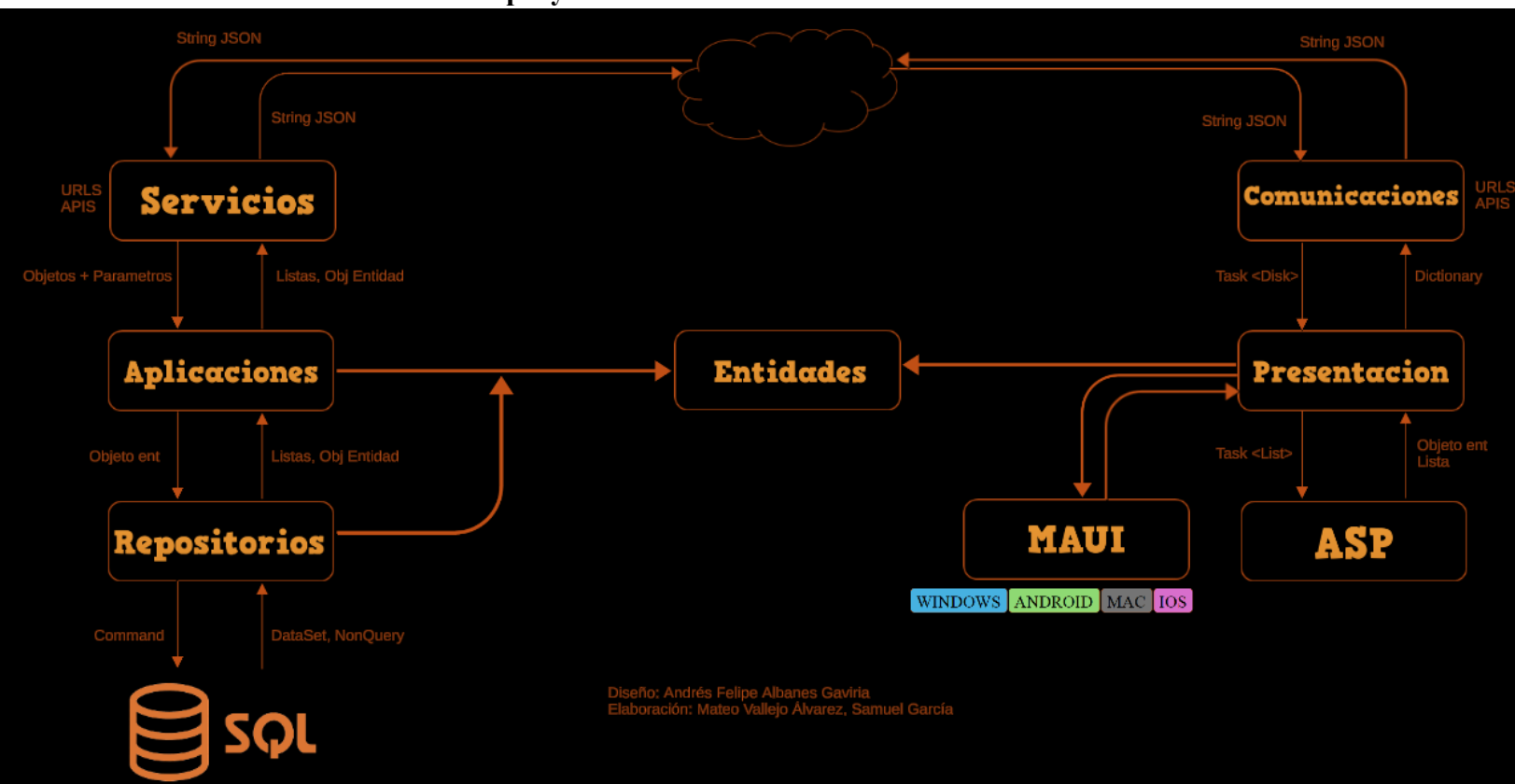
Lenguajes: C#, SQL, HTML, CSS.

Framework: .NET (Core, ASP, Entity, MAUI, Blazor).

DBMS: SQL Server Management Studio 20 (Microsoft).

Herramientas: Control de versiones (Git, GitHub), APIs (Postman), Docker

B. Estructura del proyecto

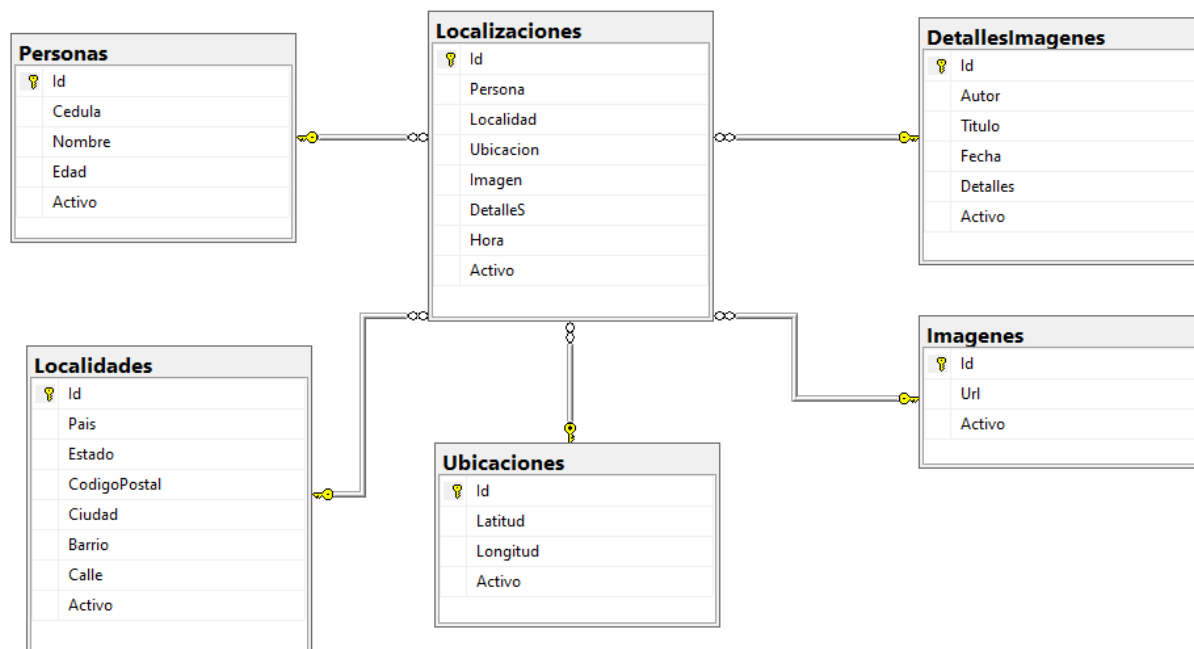


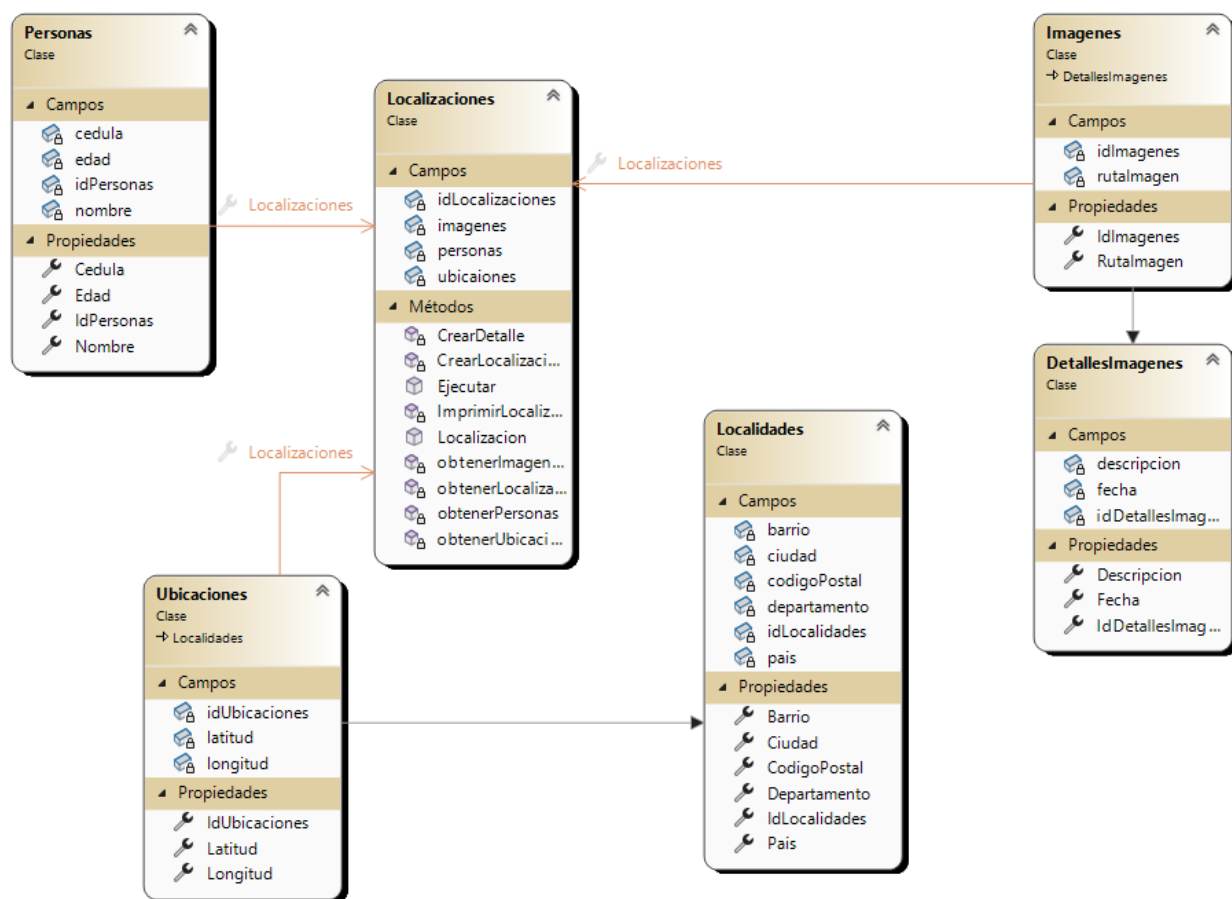
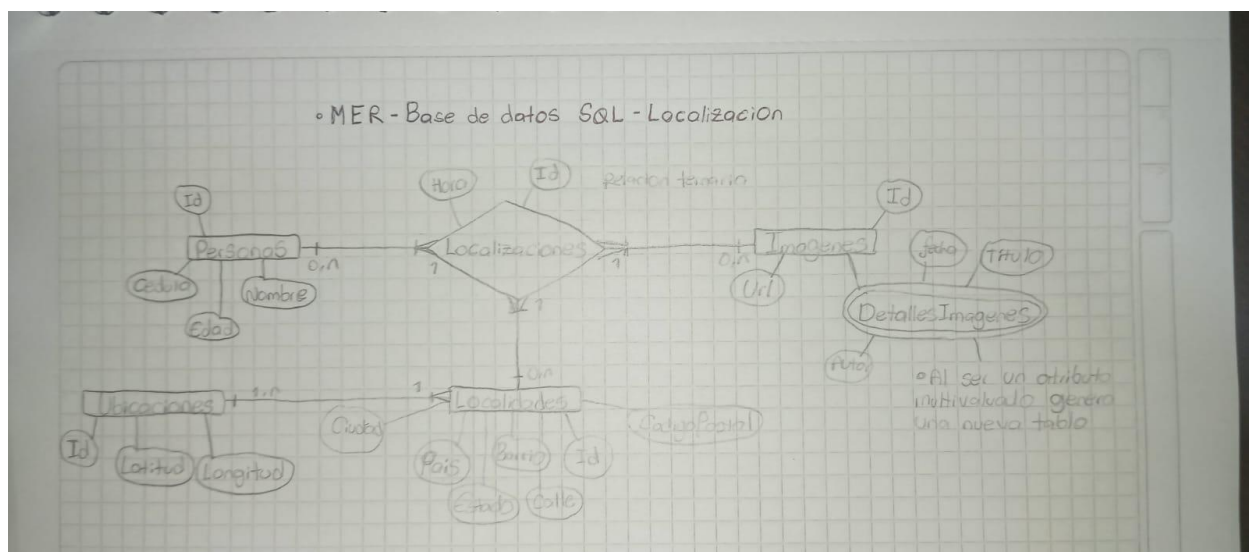
C. Contexto del proyecto

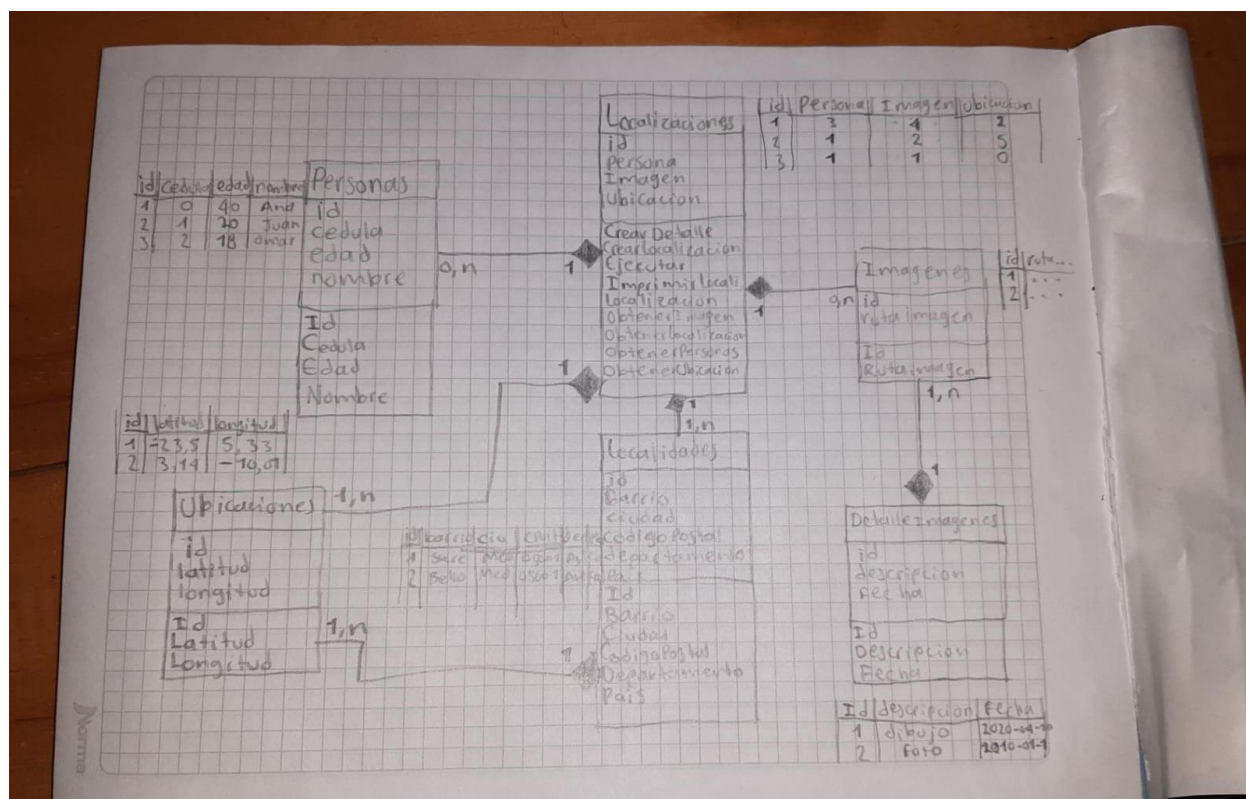
El proyecto se desarrollo en base al tema de localizacion asignado, todos los conocimientos vistos en clase y el codigo proporcionado que sirvio como base, en el desarrollo de la aplicación web completa se integraron las siguientes competencias

1. Implementaion de POO en el desarrollo de funcionalidades
2. Desarrollo, Conexión y manipulacion de bases de datos con EntityFramework y ORM
3. Diseño e implementacion de WebAPIs RESTFull
4. Desarrollo Back-End y Front-End en la aplicación
5. Uso de control de versiones con git y manejo de branches
6. Despliegue de la aplicación en un entorno de nube y monitoreo de su desempeño

II. Diagramas

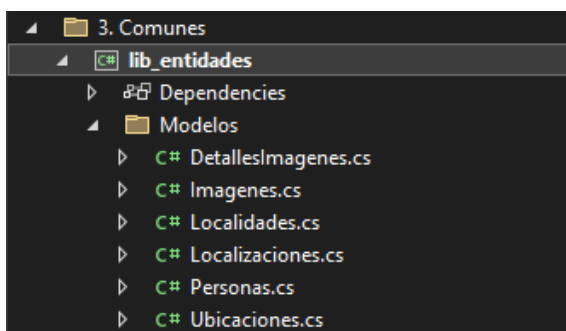






III. Proyecto

A. Librería de entidades



Representa las entidades y objetos de nuestro programa y será utilizada en las demás capas de la aplicación para asegurar la consistencia en la representación de los datos, con esto conseguimos.

1. Organización y reutilización del código
2. Mantenimiento y escalabilidad
3. Encapsulación y modularidad
4. Integración con las demás capas

```
DetallesImagenes.cs  X
lib_entidades  lib_entidades.Modelos.DetallesImagenes  Titulo
1 // Librerias
2 using System.ComponentModel.DataAnnotations;
3
4 // Proyecto
5 namespace lib_entidades.Modelos
6 {
7     // Clase que proporcionara los detalles de la clase imagen y respetara los derechos de autor
8     public class DetallesImagenes
9     {
10         // Atributos y propiedades
11         [Key] public int Id { get; set; }
12         public string? Autor { get; set; }
13         public string? Titulo { get; set; }
14         public DateTime Fecha { get; set; }
15         public string? Detalles { get; set; }
16         public bool Activo { get; set; }
17
18         // Metodos
19     }
20 }
21

Imagenes.cs  X
lib_entidades  lib_entidades.Modelos.Imagenes  Url
1 // Librerias
2 using System.ComponentModel.DataAnnotations;
3
4 // Proyecto
5 namespace lib_entidades.Modelos
6 {
7     // Esta clase estara asociada a una localizacion y proyectara una imagen a traves de su url en la web
8     public class Imagenes
9     {
10         // Atributos y propiedades
11         [Key] public int Id { get; set; }
12         public string? Url { get; set; }
13
14         // Metodos
15     }
16 }
17
```

```

Localidades.cs
lib_entidades lib_entidades.Modelos.Localidades Pais
1 // Librerias
2 using System.ComponentModel.DataAnnotations;
3
4 // Proyecto
5 namespace lib_entidades.Modelos
6 {
7     // La clase proporcionara la informacion de una ubicacion geografica especifica
8     public class Localidades
9     {
10         // Atributos y propiedades
11         [Key] public int Id { get; set; }
12         public string? Pais { get; set; }
13         public string? Estado { get; set; }
14         public string? CodigoPostal { get; set; }
15         public string? Ciudad { get; set; }
16         public string? Barrio { get; set; }
17         public string? Calle { get; set; }
18
19         // Metodos
20     }
21 }
22
Localizaciones.cs
lib_entidades lib_entidades.Modelos.Localizaciones ubicaciones
1 // Librerias
2 using System.ComponentModel.DataAnnotations;
3
4 // Proyecto
5 namespace lib_entidades.Modelos
6 {
7     // Esta clase sera la mas esencial ya que se relaciona con las demas para poder generar una localizacion
8     internal class Localizaciones
9     {
10         // Atributos y propiedades
11         [Key] public int Id { get; set; }
12         public int personas { get; set; }
13         public int localidades { get; set; }
14         public int ubicaciones { get; set; }
15         public int imagenes { get; set; }
16         public int detalles { get; set; }
17         public TimeSpan Hora { get; set; }
18         public bool Activo { get; set; }
19
20         // Metodos
21     }
22 }
23
Personas.cs
lib_entidades lib_entidades.Modelos.Personas Nombre
1 // Librerias
2 using System.ComponentModel.DataAnnotations;
3
4 // Proyecto
5 namespace lib_entidades.Modelos
6 {
7     // Clase que representa la informacion de la personas registradas en el sistema
8     public class Personas
9     {
10         // Atributos y propiedades
11         [Key] public int Id { get; set; }
12         public string? Cedula { get; set; }
13         public string? Nombre { get; set; }
14         public int Edad { get; set; }
15         public bool Activo { get; set; }
16
17         // Metodos
18     }
19 }
20

```



```

Ubicaciones.cs
lib_entidades
lib_entidades.Modelos.Ubicaciones
Longitud

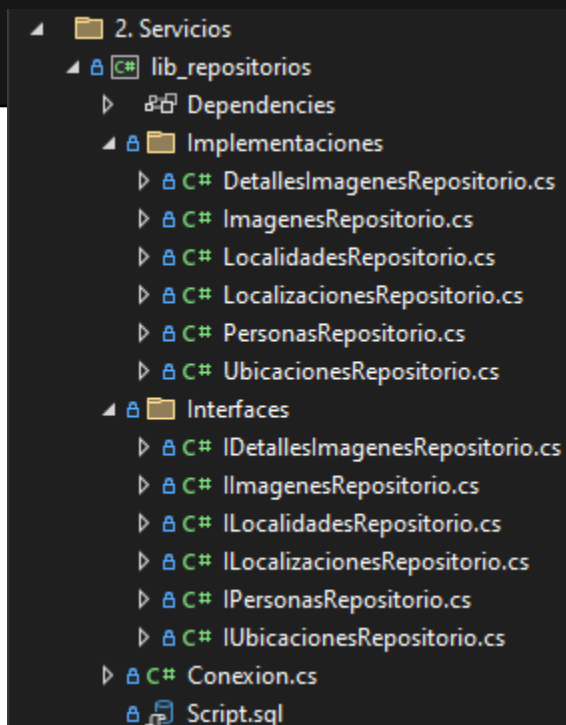
1 // Librerias
2 using System.ComponentModel.DataAnnotations;
3
4 // Proyecto
5 namespace lib_entidades.Modelos
6 {
7     // La clase representa las coordenadas geograficas exactas
8     public class Ubicaciones
9     {
10         // Atributos y propiedades
11         [Key] public int Id { get; set; }
12         public decimal Latitud { get; set; }
13         public decimal Longitud { get; set; }
14         public bool Activo { get; set; }
15
16         // Metodos
17     }
18
19

```

B. Librería de repositorios

Esta capa será la encargada de interactuar con la base de datos a través de la conexión con comandos mediante EntityFramework y su Nugget Sql, es decir es la que accede y obtiene DataSets y luego hace la conversión para dar listas retornando entidades mediante la lógica de las implementaciones y interfaces de cada entidad, con esto conseguimos.

1. Abstracción de la lógica
2. Mantenimiento y escalabilidad
3. Reutilización del código
4. Consistencia



Conexión

```

Conexion.cs
lib_repositorios  lib_repositorios.Conexion  ObtenerSet<T>()

1  // Librerias
2  using lib_entidades.Modelos;
3  using Microsoft.EntityFrameworkCore;
4  using System.Linq.Expressions;
5
6  // Proyecto
7  namespace lib_repositorios
8  {
9      // Clase que actua como puente entre la aplicacion y la base de datos relacionando las entidades y sus representaciones en la DB
10     public class Conexion : DbContext
11     {
12         // Propiedad para almacenar nuestra conexion a la base de datos
13         public string? StringConnection { get; set; }
14         // Configuración del contexto para usar SQL Server
15         protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
16         {
17             optionsBuilder.UseSqlServer(this.StringConnection!, p => { });
18             optionsBuilder.UseQueryTrackingBehavior(QueryTrackingBehavior.NoTracking);
19         }
20         // Conjuntos de datos DbSet para gestionarlas como tablas en la base de datos
21         protected DbSet<Personas>? Personas { get; set; }
22         protected DbSet<Ubicaciones>? Ubicaciones { get; set; }
23         protected DbSet<Localidades>? Localidades { get; set; }
24         protected DbSet<Imagenes>? Imagenes { get; set; }
25         protected DbSet<DetallesImagenes>? DetallesImagenes { get; set; }
26         protected DbSet<Localizaciones>? Localizaciones { get; set; }
27         // Metodos
28         public virtual DbSet<T> ObtenerSet<T>() where T : class, new()
29         {
30             return this.Set<T>();
31         }
32
33         public virtual List<T> Listar<T>() where T : class, new()
34         {
35             return this.Set<T>().ToList();
36         }
37
38         public virtual List<T> Buscar<T>(Expression<Func<T, bool>> condiciones) where T : class, new()
39         {
40             return this.Set<T>().Where(condiciones).ToList();
41         }
42
43         public virtual bool Existe<T>(Expression<Func<T, bool>> condiciones) where T : class, new()
44         {
45             return this.Set<T>().Any(condiciones);
46         }
47
48         public virtual void Guardar<T>(T entidad) where T : class, new()
49         {
50             this.Set<T>().Add(entidad);
51         }
52
53         public virtual void Modificar<T>(T entidad) where T : class
54         {
55             var entry = this.Entry(entidad);
56             entry.State = EntityState.Modified;
57         }
58
59         public virtual void Borrar<T>(T entidad) where T : class, new()
60         {
61             this.Set<T>().Remove(entidad);
62         }
63
64         public virtual void Separar<T>(T entidad) where T : class, new()
65         {
66             this.Entry(entidad).State = EntityState.Detached;
67         }
68
69         public virtual void GuardarCambios()
70         {
71             this.SaveChanges();
72         }
73     }

```

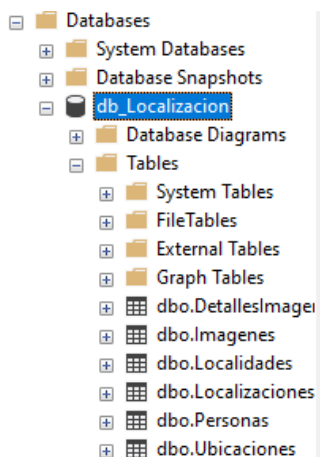
Base de datos

```
Script.sql
1  CREATE DATABASE db_Localizacion;
2  GO
3
4  USE db_Localizacion;
5  GO
6
7  CREATE TABLE Personas
8  (
9      [Id] INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
10     [Cedula] NVARCHAR(20) NOT NULL,
11     [Nombre] NVARCHAR(50) NOT NULL,
12     [Edad] INT NOT NULL,
13     [Activo] BIT NOT NULL
14 );
15 GO
16
17 INSERT INTO [Personas] ([Cedula],[Nombre],[Edad],[Activo])
18 VALUES ('1020333366','Samuel Garcia', 20, 1);
19
20 INSERT INTO [Personas] ([Cedula],[Nombre],[Edad],[Activo])
21 VALUES ('1044733299','Mateo Vallejo', 20, 1);
22 GO
23
24 CREATE TABLE Localidades
25 (
26     [Id] INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
27     [Pais] NVARCHAR(50) NOT NULL,
28     [Estado] NVARCHAR(50) NOT NULL,
29     [CodigoPostal] NVARCHAR(50) NOT NULL,
30     [Ciudad] NVARCHAR(50) NOT NULL,
31     [Barrio] NVARCHAR(50) NOT NULL,
32     [Calle] NVARCHAR(50) NOT NULL,
33     [Activo] BIT NOT NULL
34 );
35 GO
36
37 INSERT INTO [Localidades] ([Pais],[Estado],[CodigoPostal],[Ciudad],[Barrio],[Calle],[Activo])
38 VALUES ('Colombia','Antioquia','05001','Medellin','Villa Hermosa','Cl. 54a #30-01',1);
39
40 INSERT INTO [Localidades] ([Pais],[Estado],[CodigoPostal],[Ciudad],[Barrio],[Calle],[Activo])
41 VALUES ('Colombia','Antioquia','05002','Medellin','Robledo','Calle 73 No. 76A - 354',1);
42 GO
43
44 CREATE TABLE Ubicaciones
45 (
46     [Id] INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
47     [Latitud] DECIMAL(9, 6) NOT NULL,
48     [Longitud] DECIMAL(9, 6) NOT NULL,
49     [Activo] BIT NOT NULL
50 );
51 GO
52
53 INSERT INTO [Ubicaciones] ([Latitud],[Longitud],[Activo])
54 VALUES (6.244922, -75.550010, 1);
55
56 INSERT INTO [Ubicaciones] ([Latitud],[Longitud],[Activo])
57 VALUES (6.273538, -75.588527, 1);
58 GO
```

```

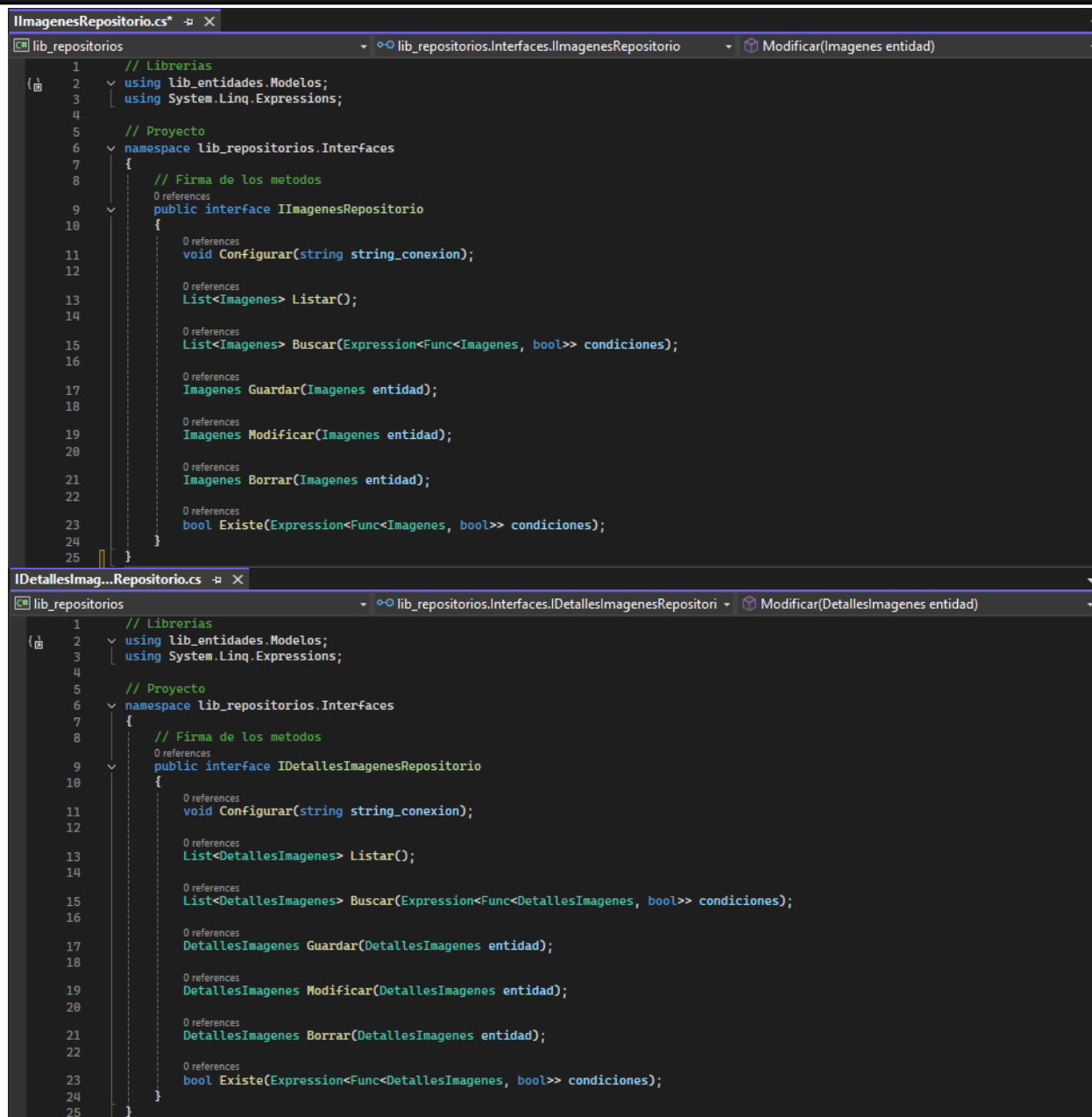
59
60 CREATE TABLE Imagenes
61 (
62     [Id] INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
63     [Url] NVARCHAR(2040) NOT NULL,
64     [Activo] BIT NOT NULL
65 );
66 GO
67
68 INSERT INTO [Imagenes] ([Url],[Activo])
69 VALUES ('https://www.itm.edu.co/wp-content/uploads/noticias/fraternidad-2.jpg', 1);
70
71 INSERT INTO [Imagenes] ([Url],[Activo])
72 VALUES ('https://www.itm.edu.co/wp-content/uploads/campus/Robledo/fotos_campusRobledo00.jpg', 1);
73 GO
74
75 CREATE TABLE DetallesImagenes
76 (
77     [Id] INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
78     [Autor] NVARCHAR(50) NOT NULL,
79     [Titulo] NVARCHAR(50) NOT NULL,
80     [Fecha] DATE NOT NULL,
81     [Detalles] NVARCHAR(2040) NULL,
82     [Activo] BIT NOT NULL
83 );
84 GO
85
86 INSERT INTO [DetallesImagenes] ([Autor],[Titulo],[Fecha],[Detalles],[Activo])
87 VALUES ('ITM', 'SEDE FRATERNIDAD', GETDATE(), 'Campus Universitario', 1);
88
89 INSERT INTO [DetallesImagenes] ([Autor],[Titulo],[Fecha],[Detalles],[Activo])
90 VALUES ('ITM', 'SEDE ROBLEDO', GETDATE(), 'Campus Universitario', 1);
91 GO
92
93 CREATE TABLE Localizaciones
94 (
95     [Id] INT NOT NULL IDENTITY(1, 1) PRIMARY KEY,
96     [personas] INT NOT NULL REFERENCES [Personas]([Id]),
97     [localidades] INT NOT NULL REFERENCES [Localidades]([Id]),
98     [ubicaciones] INT NOT NULL REFERENCES [Ubicaciones]([Id]),
99     [imagenes] INT NOT NULL REFERENCES [Imagenes]([Id]),
100     [detalles] INT NOT NULL REFERENCES [DetallesImagenes]([Id]),
101     [Hora] TIME NOT NULL,
102     [Activo] BIT NOT NULL
103 );
104 GO
105
106 INSERT INTO [Localizaciones] ([personas],[localidades],[ubicaciones],[imagenes],[detalles],[Hora],[Activo])
107 VALUES (1,1,1,1,1,CAST(GETDATE() AS TIME),1);
108
109 INSERT INTO [Localizaciones] ([personas],[localidades],[ubicaciones],[imagenes],[detalles],[Hora],[Activo])
110 VALUES (2,2,2,2,2,CAST(GETDATE() AS TIME),1);
111 GO
112
113 SELECT *
114 FROM [Personas] P
115     INNER JOIN [Localizaciones] L ON P.Id = L.personas;
116 GO

```



Interfaces

Definen el contrato para la implementación de cada entidad específica las operaciones que deben seguir como (configurar, listar, buscar, guardar, modificar, borrar, comprobar existencia)



```

IImagenesRepositorio.cs
lib_repositorios
1 // Librerías
2 using lib_entidades.Modelos;
3 using System.Linq.Expressions;
4
5 // Proyecto
6 namespace lib_repositorios.Interfaces
7 {
8     // Firma de los metodos
9     public interface IImagenesRepositorio
10     {
11         void Configurar(string string_conexion);
12
13         List<Imagenes> Listar();
14
15         List<Imagenes> Buscar(Expression<Func<Imagenes, bool>> condiciones);
16
17         Imagenes Guardar(Imagenes entidad);
18
19         Imagenes Modificar(Imagenes entidad);
20
21         Imagenes Borrar(Imagenes entidad);
22
23         bool Existe(Expression<Func<Imagenes, bool>> condiciones);
24     }
25
IDetallesImagenesRepositorio.cs
lib_repositorios
1 // Librerías
2 using lib_entidades.Modelos;
3 using System.Linq.Expressions;
4
5 // Proyecto
6 namespace lib_repositorios.Interfaces
7 {
8     // Firma de los metodos
9     public interface IDetallesImagenesRepositorio
10     {
11         void Configurar(string string_conexion);
12
13         List<DetallesImagenes> Listar();
14
15         List<DetallesImagenes> Buscar(Expression<Func<DetallesImagenes, bool>> condiciones);
16
17         DetallesImagenes Guardar(DetallesImagenes entidad);
18
19         DetallesImagenes Modificar(DetallesImagenes entidad);
20
21         DetallesImagenes Borrar(DetallesImagenes entidad);
22
23         bool Existe(Expression<Func<DetallesImagenes, bool>> condiciones);
24     }
25
  
```

```

ILocalidadesRepositorio.cs* -p X
lib_repositorios lib_repositorios.Interfaces.ILocalidadesRepositorio Modificar(Localidades entidad)
1 // Librerias
2 using lib_entidades.Modelos;
3 using System.Linq.Expressions;
4
5 // Proyecto
6 namespace lib_repositorios.Interfaces
7 {
8     // Firma de los metodos
9     public interface ILocalidadesRepositorio
10    {
11        void Configurar(string string_conexion);
12
13        List<Localidades> Listar();
14
15        List<Localidades> Buscar(Expression<Func<Localidades, bool>> condiciones);
16
17        Localidades Guardar(Localidades entidad);
18
19        Localidades Modificar(Localidades entidad);
20
21        Localidades Borrar(Localidades entidad);
22
23        bool Existe(Expression<Func<Localidades, bool>> condiciones);
24    }
25
ILocalizacionesRepositorio.cs -p X
lib_repositorios lib_repositorios.Interfaces.ILocalizacionesRepositorio Modificar(Localizaciones entidad)
1 // Librerias
2 using lib_entidades.Modelos;
3 using System.Linq.Expressions;
4
5 // Proyecto
6 namespace lib_repositorios.Interfaces
7 {
8     // Firma de los metodos
9     public interface ILocalizacionesRepositorio
10    {
11        void Configurar(string string_conexion);
12
13        List<Localizaciones> Listar();
14
15        List<Localizaciones> Buscar(Expression<Func<Localizaciones, bool>> condiciones);
16
17        Localizaciones Guardar(Localizaciones entidad);
18
19        Localizaciones Modificar(Localizaciones entidad);
20
21        Localizaciones Borrar(Localizaciones entidad);
22
23        bool Existe(Expression<Func<Localizaciones, bool>> condiciones);
24    }
25

```

```

IUbicacionesRepositorio.cs*
lib_repositorios
lib_repositorios.Interfaces.IUbicacionesRepositorio
Modificar(Ubicaciones entidad)

1 // Librerias
2 using lib_entidades.Modelos;
3 using System.Linq.Expressions;
4
5 // Proyecto
6 namespace lib_repositorios.Interfaces
7 {
8     // Firma de los metodos
9     public interface IUbicacionesRepositorio
10    {
11        void Configurar(string string_conexion);
12
13        List<Ubicaciones> Listar();
14
15        List<Ubicaciones> Buscar(Expression<Func<Ubicaciones, bool>> condiciones);
16
17        Ubicaciones Guardar(Ubicaciones entidad);
18
19        Ubicaciones Modificar(Ubicaciones entidad);
20
21        Ubicaciones Borrar(Ubicaciones entidad);
22
23        bool Existe(Expression<Func<Ubicaciones, bool>> condiciones);
24    }
25 }

IPersonasRepositorio.cs
lib_repositorios
lib_repositorios.Interfaces.IPersonasRepositorio
Modificar(Personas entidad)

1 // Librerias
2 using lib_entidades.Modelos;
3 using System.Linq.Expressions;
4
5 // Proyecto
6 namespace lib_repositorios.Interfaces
7 {
8     // Firma de los metodos
9     public interface IPersonasRepositorio
10    {
11        void Configurar(string string_conexion);
12
13        List<Personas> Listar();
14
15        List<Personas> Buscar(Expression<Func<Personas, bool>> condiciones);
16
17        Personas Guardar(Personas entidad);
18
19        Personas Modificar(Personas entidad);
20
21        Personas Borrar(Personas entidad);
22
23        bool Existe(Expression<Func<Personas, bool>> condiciones);
24    }
25 }

```

Implementaciones

Proporciona la implementación de los métodos necesarios para llevar a cabo las operaciones CRUD (operaciones básicas de la base de datos utilizando la instancia de la conexión) sobre cada entidad.

```

DetallesImage...epositorio.cs
lib_repositorios
lib_repositorios.Implementaciones.DetallesImagenesR
conexion

1 // Librerías
2 using lib_entidades.Modelos;
3 using lib_repositorios.Interfaces;
4 using System.Linq.Expressions;
5
6 // Proyecto
7 namespace lib_repositorios.Implementaciones
8 {
9     1 reference
10     public class DetallesImagenesRepositorio : IDetallesImagenesRepositorio
11     {
12         // Variable de tipo conexion para interactuar con la DB
13         private Conexion? conexion;
14         // Constructor
15         0 references
16         public DetallesImagenesRepositorio(Conexion conexion)
17         {
18             this.conexion = conexion;
19         }
20         // Metodos definidos por la interfaz y llamado a metodos de la clase Conexion
21         1 reference
22         public void Configurar(string string_conexion)
23         {
24             this.conexion!.StringConnection = string_conexion;
25         }
26         // Obtener todas las entidades
27         1 reference
28         public List<DetallesImagenes> Listar()
29         {
30             return conexion!.Listar<DetallesImagenes>();
31         }
32         // Buscar entidades bajo determinadas condiciones
33         1 reference
34         public List<DetallesImagenes> Buscar(Expression<Func<DetallesImagenes, bool>> condiciones)
35         {
36             return conexion!.Buscar(condiciones);
37         }
38         // Guardar una nueva entidad
39         1 reference
40         public DetallesImagenes Guardar(DetallesImagenes entidad)
41         {
42             conexion!.Guardar(entidad);
43             conexion!.GuardarCambios();
44             conexion!.Separar(entidad);
45             return entidad;
46         }
47         // Modificar una entidad ya existente
48         1 reference
49         public DetallesImagenes Modificar(DetallesImagenes entidad)
50         {
51             conexion!.Modificar(entidad);
52             conexion!.GuardarCambios();
53             conexion!.Separar(entidad);
54             return entidad;
55         }
56         // Borrar una entidad ya existente
57         1 reference
58         public DetallesImagenes Borrar(DetallesImagenes entidad)
59         {
60             conexion!.Borrar(entidad);
61             conexion!.GuardarCambios();
62             conexion!.Separar(entidad);
63             return entidad;
64         }
65         // Verificar la existencia de una entidad que cumple ciertas condiciones y devuelve un valor booleano
66         1 reference
67         public bool Existe(Expression<Func<DetallesImagenes, bool>> condiciones)
68         {
69             return conexion!.Existe(condiciones);
70         }
71     }
72 }

```



```

ImagenesRepositorio.cs
lib_repositorios
lib_repositorios.Implementaciones.ImagenesRepositorio
conexion

1 // Librerias
2 using lib_entidades.Modelos;
3 using lib_repositorios.Interfaces;
4 using System.Linq.Expressions;
5
6 // Proyecto
7 namespace lib_repositorios.Implementaciones
8 {
9     public class ImagenesRepositorio : IImagenesRepositorio
10     {
11         // Variable de tipo conexion para interactuar con la DB
12         private Conexion? conexion;
13         // Constructor
14         public ImagenesRepositorio(Conexion conexion)
15         {
16             this.conexion = conexion;
17         }
18         // Metodos definidos por la interfaz y llamado a metodos de la clase Conexion
19         public void Configurar(string string_conexion)
20         {
21             this.conexion!.StringConnection = string_conexion;
22         }
23         // Obtener todas las entidades
24         public List<Imagenes> Listar()
25         {
26             return conexion!.Listar<Imagenes>();
27         }
28         // Buscar entidades bajo determinadas condiciones
29         public List<Imagenes> Buscar(Expression<Func<Imagenes, bool>> condiciones)
30         {
31             return conexion!.Buscar(condiciones);
32         }
33         // Guardar una nueva entidad
34         public Imagenes Guardar(Image entity)
35         {
36             conexion!.Guardar(entity);
37             conexion!.GuardarCambios();
38             conexion!.Separar(entity);
39             return entity;
40         }
41         // Modificar una entidad ya existente
42         public Imagenes Modificar(Image entity)
43         {
44             conexion!.Modificar(entity);
45             conexion!.GuardarCambios();
46             conexion!.Separar(entity);
47             return entity;
48         }
49         // Borrar una entidad ya existente
50         public Imagenes Borrar(Image entity)
51         {
52             conexion!.Borrar(entity);
53             conexion!.GuardarCambios();
54             conexion!.Separar(entity);
55             return entity;
56         }
57         // Verificar la existencia de una entidad que cumple ciertas condiciones y devuelve un valor booleano
58         public bool Existe(Expression<Func<Imagenes, bool>> condiciones)
59         {
60             return conexion!.Existe(condiciones);
61         }
62     }
63 }

```

```

LocalidadesRepositorio.cs
lib_repositorios
lib_repositorios.Implementaciones.LocalidadesReposit
conexion

1 // Librerias
2 using lib_entidades.Modelos;
3 using lib_repositorios.Interfaces;
4 using System.Linq.Expressions;
5
6 // Proyecto
7 namespace lib_repositorios.Implementaciones
8 {
9     1 reference
10     public class LocalidadesRepositorio : ILocalidadesRepositorio
11     {
12         // Variable de tipo conexion para interactuar con la DB
13         private Conexion? conexion;
14         // Constructor
15         public LocalidadesRepositorio(Conexion conexion)
16         {
17             this.conexion = conexion;
18         }
19         // Metodos definidos por la interfaz y llamado a metodos de la clase Conexion
20         1 reference
21         public void Configurar(string string_conexion)
22         {
23             this.conexion!.StringConnection = string_conexion;
24         }
25         // Obtener todas las entidades
26         1 reference
27         public List<Localidades> Listar()
28         {
29             return conexion!.Listar<Localidades>();
30         }
31         // Buscar entidades bajo determinadas condiciones
32         1 reference
33         public List<Localidades> Buscar(Expression<Func<Localidades, bool>> condiciones)
34         {
35             return conexion!.Buscar(condiciones);
36         }
37         // Guardar una nueva entidad
38         1 reference
39         public Localidades Guardar(Localidades entidad)
40         {
41             conexion!.Guardar(entidad);
42             conexion!.GuardarCambios();
43             conexion!.Separar(entidad);
44             return entidad;
45         }
46         // Modificar una entidad ya existente
47         1 reference
48         public Localidades Modificar(Localidades entidad)
49         {
50             conexion!.Modificar(entidad);
51             conexion!.GuardarCambios();
52             conexion!.Separar(entidad);
53             return entidad;
54         }
55         // Borrar una entidad ya existente
56         1 reference
57         public Localidades Borrar(Localidades entidad)
58         {
59             conexion!.Borrar(entidad);
60             conexion!.GuardarCambios();
61             conexion!.Separar(entidad);
62             return entidad;
63         }
64         // Verificar la existencia de una entidad que cumple ciertas condiciones y devuelve un valor booleano
65         1 reference
66         public bool Existe(Expression<Func<Localidades, bool>> condiciones)
67         {
68             return conexion!.Existe(condiciones);
69         }
70     }
71 }

```

```

LocalizacionesRepositorio.cs
lib_repositorios
lib_repositorios.Implementaciones.LocalizacionesRepc
conexion

1 // Librerías
2 using lib_entidades.Modelos;
3 using lib_repositorios.Interfaces;
4 using System.Linq.Expressions;
5
6 // Proyecto
7 namespace lib_repositorios.Implementaciones
8 {
9     1 reference
10     public class LocalizacionesRepositorio : ILocalizacionesRepositorio
11     {
12         // Variable de tipo conexion para interactuar con la DB
13         private Conexion? conexion;
14         // Constructor
15         0 references
16         public LocalizacionesRepositorio(Conexion conexion)
17         {
18             this.conexion = conexion;
19         }
20         // Metodos definidos por la interfaz y llamado a metodos de la clase Conexion
21         1 reference
22         public void Configurar(string string_conexion)
23         {
24             this.conexion!.StringConnection = string_conexion;
25         }
26         // Obtener todas las entidades
27         1 reference
28         public List<Localizaciones> Listar()
29         {
30             return conexion!.Listar<Localizaciones>();
31         }
32         // Buscar entidades bajo determinadas condiciones
33         1 reference
34         public List<Localizaciones> Buscar(Expression<Func<Localizaciones, bool>> condiciones)
35         {
36             return conexion!.Buscar(condiciones);
37         }
38         // Guardar una nueva entidad
39         1 reference
40         public Localizaciones Guardar(Localizaciones entidad)
41         {
42             conexion!.Guardar(entidad);
43             conexion!.GuardarCambios();
44             conexion!.Separar(entidad);
45             return entidad;
46         }
47         // Modificar una entidad ya existente
48         1 reference
49         public Localizaciones Modificar(Localizaciones entidad)
50         {
51             conexion!.Modificar(entidad);
52             conexion!.GuardarCambios();
53             conexion!.Separar(entidad);
54             return entidad;
55         }
56         // Borrar una entidad ya existente
57         public Localizaciones Borrar(Localizaciones entidad)
58         {
59             conexion!.Borrar(entidad);
60             conexion!.GuardarCambios();
61             conexion!.Separar(entidad);
62             return entidad;
63         }
64         // Verificar la existencia de una entidad que cumple ciertas condiciones y devuelve un valor booleano
65         public bool Existe(Expression<Func<Localizaciones, bool>> condiciones)
66         {
67             return conexion!.Existe(condiciones);
68         }
69     }
70 }

```

```

PersonasRepositorio.cs
lib_repositorios
lib_repositorios.Implementaciones.PersonasRepositorio
conexion

1 // Librerias
2 using lib_entidades.Modelos;
3 using lib_repositorios.Interfaces;
4 using System.Linq.Expressions;
5
6 // Proyecto
7 namespace lib_repositorios.Implementaciones
8 {
9     public class PersonasRepositorio : IPersonasRepositorio
10    {
11        // Variable de tipo conexion para interactuar con la DB
12        private Conexion? conexion;
13        // Constructor
14        public PersonasRepositorio(Conexion conexion)
15        {
16            this.conexion = conexion;
17        }
18        // Metodos definidos por la interfaz y llamado a metodos de la clase Conexion
19        public void Configurar(string string_conexion)
20        {
21            this.conexion!.StringConnection = string_conexion;
22        }
23        // Obtener todas las entidades
24        public List<Personas> Listar()
25        {
26            return conexion!.Listar<Personas>();
27        }
28        // Buscar entidades bajo determinadas condiciones
29        public List<Personas> Buscar(Expression<Func<Personas, bool>> condiciones)
30        {
31            return conexion!.Buscar(condiciones);
32        }
33        // Guardar una nueva entidad
34        public Personas Guardar(Personas entidad)
35        {
36            conexion!.Guardar(entidad);
37            conexion!.GuardarCambios();
38            conexion!.Separar(entidad);
39            return entidad;
40        }
41        // Modificar una entidad ya existente
42        public Personas Modificar(Personas entidad)
43        {
44            conexion!.Modificar(entidad);
45            conexion!.GuardarCambios();
46            conexion!.Separar(entidad);
47            return entidad;
48        }
49        // Borrar una entidad ya existente
50        public Personas Borrar(Personas entidad)
51        {
52            conexion!.Borrar(entidad);
53            conexion!.GuardarCambios();
54            conexion!.Separar(entidad);
55            return entidad;
56        }
57        // Verificar la existencia de una entidad que cumple ciertas condiciones y devuelve un valor booleano
58        public bool Existe(Expression<Func<Personas, bool>> condiciones)
59        {
60            return conexion!.Existe(condiciones);
61        }
62    }
63 }

```

```

UbicacionesRepositorio.cs
lib_repositorios
lib_repositorios.Implementaciones.UbicacionesReposil
conexion

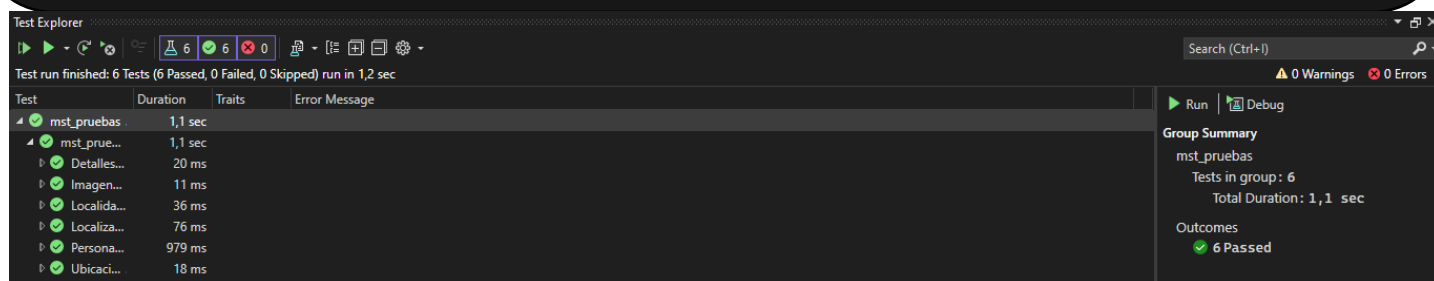
1 // Librerias
2 using lib_entidades.Modelos;
3 using lib_repositorios.Interfaces;
4 using System.Linq.Expressions;
5
6 // Proyecto
7 namespace lib_repositorios.Implementaciones
8 {
9     public class UbicacionesRepositorio : IUbicacionesRepositorio
10     {
11         // Variable de tipo conexion para interactuar con la DB
12         private Conexion? conexion;
13         // Constructor
14         public UbicacionesRepositorio(Conexion conexion)
15         {
16             this.conexion = conexion;
17         }
18         // Metodos definidos por la interfaz y llamado a metodos de la clase Conexion
19         public void Configurar(string string_conexion)
20         {
21             this.conexion!.StringConnection = string_conexion;
22         }
23         // Obtener todas las entidades
24         public List<Ubicaciones> Listar()
25         {
26             return conexion!.Listar<Ubicaciones>();
27         }
28         // Buscar entidades bajo determinadas condiciones
29         public List<Ubicaciones> Buscar(Expression<Func<Ubicaciones, bool>> condiciones)
30         {
31             return conexion!.Buscar(condiciones);
32         }
33         // Guardar una nueva entidad
34         public Ubicaciones Guardar(Ubicaciones entidad)
35         {
36             conexion!.Guardar(entidad);
37             conexion!.GuardarCambios();
38             conexion!.Separar(entidad);
39             return entidad;
40         }
41         // Modificar una entidad ya existente
42         public Ubicaciones Modificar(Ubicaciones entidad)
43         {
44             conexion!.Modificar(entidad);
45             conexion!.GuardarCambios();
46             conexion!.Separar(entidad);
47             return entidad;
48         }
49         // Borrar una entidad ya existente
50         public Ubicaciones Borrar(Ubicaciones entidad)
51         {
52             conexion!.Borrar(entidad);
53             conexion!.GuardarCambios();
54             conexion!.Separar(entidad);
55             return entidad;
56         }
57         // Verificar la existencia de una entidad que cumple ciertas condiciones y devuelve un valor booleano
58         public bool Existe(Expression<Func<Ubicaciones, bool>> condiciones)
59         {
60             return conexion!.Existe(condiciones);
61         }
62     }
63 }

```

Pruebas unitarias (Repositorio)

El propósito de las pruebas unitarias es comprobar que los componentes individuales de la aplicación tales como (Funciones, Métodos, Clases) funcionan correctamente de manera aislada con esto conseguimos:

1. Verificación de funcionalidad
2. Detección temprana de errores
3. Refactoring, confianza, seguridad, mantenimiento (Código en el proyecto).



C. Librería de aplicaciones

El propósito de esta capa de la aplicación es manejar la lógica de negocio y orquestar el flujo de datos entre la capa de servicios y la capa de repositorios, su función es separa la lógica de servicios y presentación y la lógica del acceso a los datos beneficiándonos en:

1. Mantenibilidad y escalabilidad
2. Reutilización del código
3. Separación de preocupaciones
4. Aplicación de pruebas de negocio

Interfaces

```

IDetallesImag...sAplicacion.cs  X
lib_aplicaciones  lib_aplicaciones.Interfaces.IDetallesImagenesAplicacio  Configurar(string string_conexion)
1  // Librerias
2  using lib_entidades.Modelos;
3
4  // Proyecto
5  namespace lib_aplicaciones.Interfaces
6  {
7      // Firma de los metodos
8      public interface IDetallesImagenesAplicacion
9      {
10         void Configurar(string string_conexion);
11
12         List<DetallesImagenes> Listar();
13
14         List<DetallesImagenes> Buscar(DetallesImagenes entidad, string tipo);
15
16         DetallesImagenes Guardar(DetallesImagenes entidad);
17
18         DetallesImagenes Modificar(DetallesImagenes entidad);
19
20         DetallesImagenes Borrar(DetallesImagenes entidad);
21     }
22
IImagenesAplicacion.cs  X
lib_aplicaciones  lib_aplicaciones.Interfaces.IImagenesAplicacion  Configurar(string string_conexion)
1  // Librerias
2  using lib_entidades.Modelos;
3
4  // Proyecto
5  namespace lib_aplicaciones.Interfaces
6  {
7      // Firma de los metodos
8      public interface IImagenesAplicacion
9      {
10         void Configurar(string string_conexion);
11
12         List<Imagenes> Listar();
13
14         List<Imagenes> Buscar(Image es entidad, string tipo);
15
16         Imagenes Guardar(Image es entidad);
17
18         Imagenes Modificar(Image es entidad);
19
20         Imagenes Borrar(Image es entidad);
21     }
22

```

```

ILocalidadesAplicacion.cs
lib_aplicaciones
lib_aplicaciones.Interfaces.ILocalidadesAplicacion
Configurar(string string_conexion)

1 // Librerias
2 using lib_entidades.Modelos;
3
4 // Proyecto
5 namespace lib_aplicaciones.Interfaces
6 {
7     // Firma de los metodos
8     public interface ILocalidadesAplicacion
9     {
10         void Configurar(string string_conexion);
11
12         List<Localidades> Listar();
13
14         List<Localidades> Buscar(Localidades entidad, string tipo);
15
16         Localidades Guardar(Localidades entidad);
17
18         Localidades Modificar(Localidades entidad);
19
20         Localidades Borrar(Localidades entidad);
21     }
22 }

ILocalizacionesAplicacion.cs
lib_aplicaciones
lib_aplicaciones.Interfaces.ILocalizacionesAplicacion
Configurar(string string_conexion)

1 // Librerias
2 using lib_entidades.Modelos;
3
4 // Proyecto
5 namespace lib_aplicaciones.Interfaces
6 {
7     // Firma de los metodos
8     public interface ILocalizacionesAplicacion
9     {
10         void Configurar(string string_conexion);
11
12         List<Localizaciones> Listar();
13
14         List<Localizaciones> Buscar(Localizaciones entidad, string tipo);
15
16         Localizaciones Guardar(Localizaciones entidad);
17
18         Localizaciones Modificar(Localizaciones entidad);
19
20         Localizaciones Borrar(Localizaciones entidad);
21     }
22 }

```


IPersonasAplicacion.cs

lib_aplicaciones

lib_aplicaciones.Interfaces.IPersonasAplicacion

Configurar(string string_conexion)

```

1 // Librerias
2 using lib_entidades.Modelos;
3
4 // Proyecto
5 namespace lib_aplicaciones.Interfaces
6 {
7     // Firma de los metodos
8
9     public interface IPersonasAplicacion
10     {
11
12         void Configurar(string string_conexion);
13
14         List<Personas> Listar();
15
16         List<Personas> Buscar(Personas entidad, string tipo);
17
18         Personas Guardar(Personas entidad);
19
20         Personas Modificar(Personas entidad);
21
22         0 references
23         Personas Borrar(Personas entidad);
24     }
25 }

```

IUbicacionesAplicaciones.cs

lib_aplicaciones

lib_aplicaciones.Interfaces.IUbicacionesAplicacion

Configurar(string string_conexion)

```

1 // Librerias
2 using lib_entidades.Modelos;
3
4 // Proyecto
5 namespace lib_aplicaciones.Interfaces
6 {
7     // Firma de los metodos
8
9     public interface IUbicacionesAplicacion
10     {
11
12         0 references
13         void Configurar(string string_conexion);
14
15         0 references
16         List<Ubicaciones> Listar();
17
18         0 references
19         List<Ubicaciones> Buscar(Ubicaciones entidad, string tipo);
20
21         0 references
22         Ubicaciones Guardar(Ubicaciones entidad);
23
24         0 references
25         Ubicaciones Modificar(Ubicaciones entidad);
26
27         0 references
28         Ubicaciones Borrar(Ubicaciones entidad);
29     }
30 }

```

Implementaciones

```

1 // Librerías
2 using lib_entidades.Modelos;
3 using lib_aplicaciones.Interfaces;
4 using lib_repositorios.Interfaces;
5 using System.Linq.Expressions;
6
7 // Proyecto
8 namespace lib_aplicaciones.Implementaciones
9 {
10     1 reference
11     public class DetallesImagenesAplicacion : IDetallesImagenesAplicacion
12     {
13         // Variable para interactuar con el repositorio
14         private IDetallesImagenesRepositorio? iRepositorio = null;
15         // Constructor
16         public DetallesImagenesAplicacion(IDetallesImagenesRepositorio iRepositorio)
17         {
18             this.iRepositorio = iRepositorio;
19         }
20         // Metodos
21         1 reference
22         public void Configurar(string string_conexion)
23         {
24             this.iRepositorio!.Configurar(string_conexion);
25         }
26
27         1 reference
28         public DetallesImagenes Borrar(DetallesImagenes entidad)
29         {
30             if (entidad == null || !entidad.Validar())
31                 throw new Exception("lbFaltaInformacion");
32
33             if (entidad.Id == 0)
34                 throw new Exception("lbNoSeGuardo");
35
36             entidad = iRepositorio!.Borrar(entidad);
37             return entidad;
38         }
39
40         1 reference
41         public DetallesImagenes Guardar(DetallesImagenes entidad)
42         {
43             if (entidad == null || !entidad.Validar())
44                 throw new Exception("lbFaltaInformacion");
45
46             if (entidad.Id != 0)
47                 throw new Exception("lbYaSeGuardo");
48
49             entidad = iRepositorio!.Guardar(entidad);
50             return entidad;
51         }
52
53         1 reference
54         public List<DetallesImagenes> Listar()
55         {
56             return iRepositorio!.Listar();
57         }
58
59         1 reference
60         public List<DetallesImagenes> Buscar(DetallesImagenes entidad, string tipo)
61         {
62             Expression<Func<DetallesImagenes, bool>>? condiciones = null;
63             switch (tipo.ToUpper())
64             {
65                 case "NOMBRE": condiciones = x => x.Autor!.Contains(entidad.Autor!); break;
66                 default: condiciones = x => x.Id == entidad.Id; break;
67             }
68             return this.iRepositorio!.Buscar(condiciones);
69         }
70
71         1 reference
72         public DetallesImagenes Modificar(DetallesImagenes entidad)
73         {
74             if (entidad == null || !entidad.Validar())
75                 throw new Exception("lbFaltaInformacion");
76
77             if (entidad.Id == 0)
78                 throw new Exception("lbNoSeGuardo");
79
80             entidad = iRepositorio!.Modificar(entidad);
81             return entidad;
82         }
83
84         /* Metodo vacio
85         private DetallesImagenes Calcular(DetallesImagenes entidad)
86         {
87         }
88         */
89     }
90 }

```

Falta implementar la lógica del método validar en las entidades y las aplicaciones

ASP_Servicios