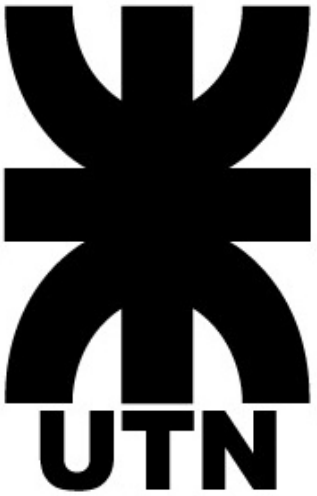



# Tecnologías de Desarrollo de Software IDE

# Trabajo Práctico Integrador

# Resolución de Anomalías

Comisión: 3EK01

Integrantes:

- Aguilera Tomas - Legajo: 53227
- Aronson Melina - Legajo: 52058
- Tomasino Alvaro - Legajo: 53082

Año: 2025

## Sistema de Resolución de Anomalías (Gestión de Denuncias)

El Sistema de Resolución de Anomalías es una plataforma diseñada para gestionar avistamientos de anomalías producidas por fantasmas en la provincia de Santa Fe. El sistema permite coordinar la respuesta a estos eventos mediante la asignación de cazadores especializados a pedidos de resolución generados por operadores, quienes reciben reportes de los Denunciantes.

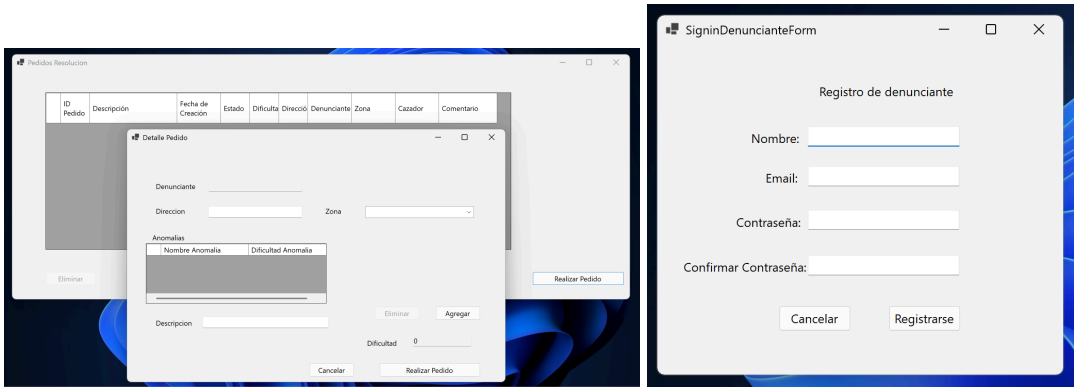
El sistema maneja tres tipos de usuarios con roles diferenciados:

- Operador: Encargado de gestionar el sistema, aceptar pedidos de agregación y administrar tipos de anomalías, zonas y localidades.
- Cazador: Profesional asignado a una zona específica que acepta y resuelve pedidos de resolución según su ubicación.
- Denunciante: Ciudadano que reporta avistamientos de anomalías al sistema.

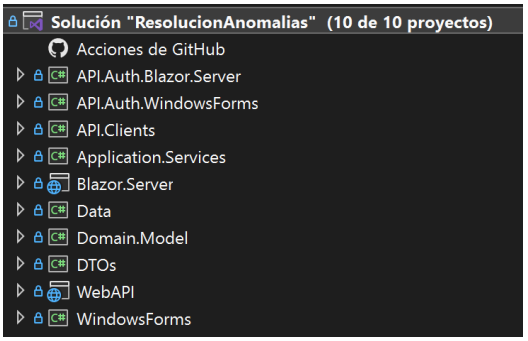
Los Denunciantes disponen de un catálogo de anomalías que pueden ser reportadas. Los cazadores pueden generar solicitudes para agregar nuevos tipos de anomalías al catálogo del sistema. Estas solicitudes son aprobadas por el Operador.

Funcionalidades implementadas:

1. Alta, Baja, Modificaciones y Consulta de Usuarios (Cazadores y Denunciantes).
2. Alta, Baja, Modificaciones y Consulta de Tipos de Anomalías.
3. Alta, Baja, Modificaciones y Consulta de Localidades.
4. Alta, Baja, Modificaciones y Consulta de Zonas.
5. Alta, Baja, Modificaciones y Consulta de Pedidos de Resolución.
6. Alta, Baja, Modificaciones y Consulta de Pedidos de Agregación.
7. Registro de resolución de Pedidos de Resolución. Donde:
    - Los Cazadores documentan el resultado de la intervención.
    - Los Cazadores agregan comentarios sobre las acciones tomadas.
    - Los Cazadores actualizan el estado del pedido (Resuelto o No resuelto).
8. Registro de resolución de Pedidos de Agregación. Donde:
    - Operador actualiza el estado del pedido (Aceptado o Rechazado).
9. Reporte de cantidad de Pedidos de Resolucion realizados en el mes.
10. Reporte de cantidad de Pedidos de Agregación realizados.

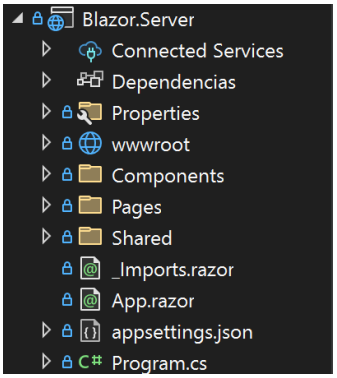


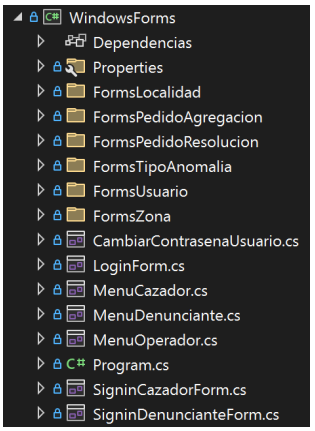
## Arquitectura del Sistema



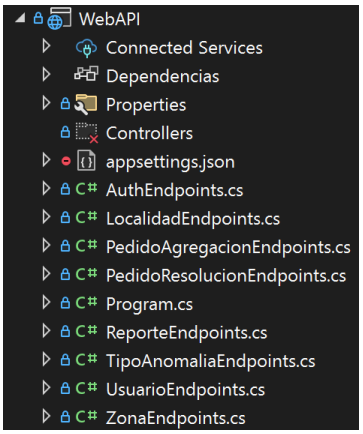

El sistema implementa una arquitectura en capas:

1. Capa de Presentación:
   - Blazor Server: Interfaz web interactiva para gestión administrativa
   - Windows Forms: Aplicación de escritorio para operadores



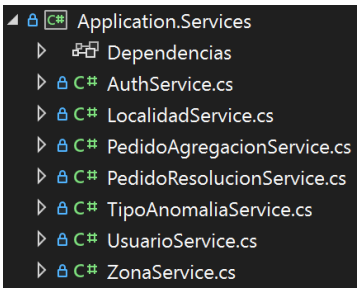




2. Rest API (WebAPI):
   - Endpoints para todas las entidades
   - Autenticación JWT para seguridad
   - Autorización basada en roles
   - Validación de datos

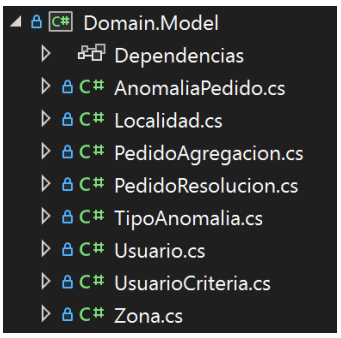



3. Capa de Aplicación (Application.Services):
   - Lógica de negocio
   - Servicios para cada entidad

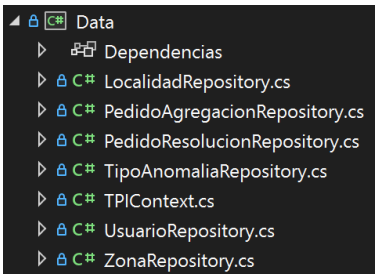
- Validaciones de reglas de negocio



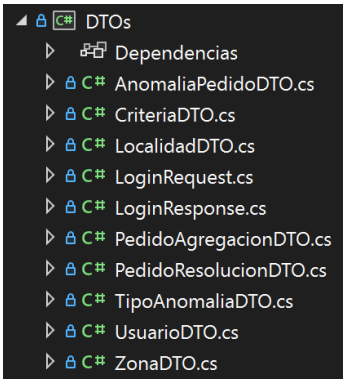

4. Capa de Dominio (Domain.Model):
   - Entidades de negocio con encapsulamiento
   - Validaciones en setters
   - Relaciones entre entidades




5. Capa de Datos (Data):
   - Repositorios para acceso a datos
   - DbContext de Entity Framework Core




6. Capa de Transferencia (DTOs):
   - Objetos de transferencia de datos
   - Serialización JSON para APIs

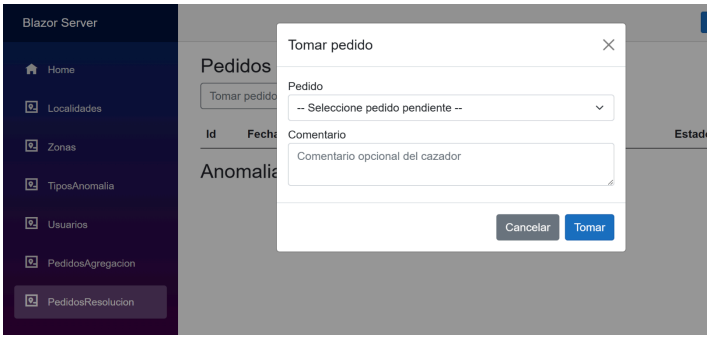


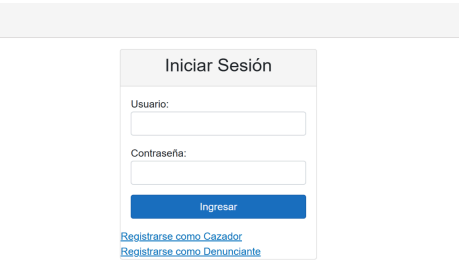
## Tecnologías Utilizadas

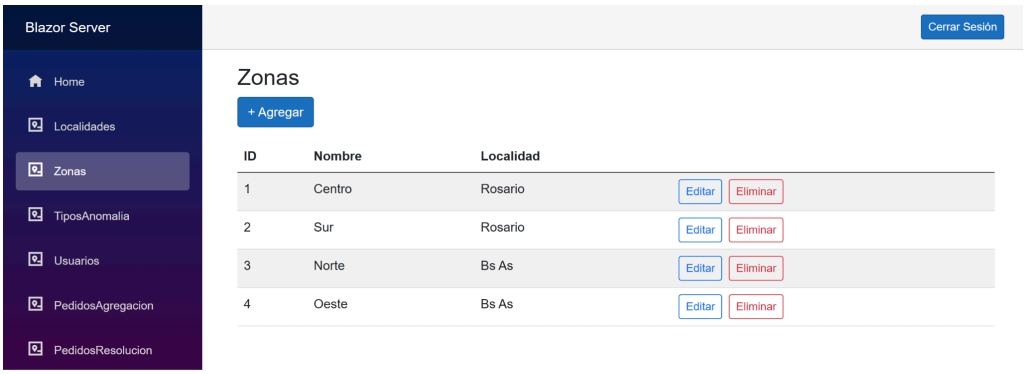
- NET 8.
- C#.
- Entity Framework Core:

```
using Domain.Model;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;

namespace Data
{
    13 referencias
    public class TPIContext : DbContext
    {
        7 referencias
        public DbSet<TipoAnomalia> TipoAnomalias { get; set; }
        7 referencias
        public DbSet<Localidad> Localidades { get; set; }
        8 referencias
        public DbSet<Zona> Zonas { get; set; }
        9 referencias
        public DbSet<Usuario> Usuarios { get; set; }
        6 referencias
        public DbSet<PedidoAgregacion> PedidosAgregacion { get; set; }
        7 referencias
        public DbSet<PedidoResolucion> PedidosResolucion { get; set; }

        6 referencias
        internal TPIContext()
        {
            this.Database.EnsureCreated();
        }

        0 referencias
        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            if (!optionsBuilder.IsConfigured)
            {
                var configuration = new ConfigurationBuilder()
                    .SetBasePath(Directory.GetCurrentDirectory())
                    .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                    .Build();

                string connectionString = configuration.GetConnectionString("DefaultConnection");
                optionsBuilder.UseSqlServer(connectionString);
            }
        }
```

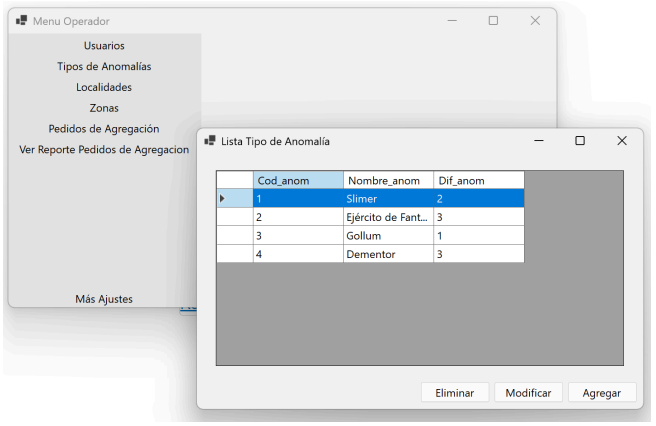
- SQL Server, con interfaz SQL Server Management Studio 21.
- Blazor Server junto con Bootstrap y CSS:

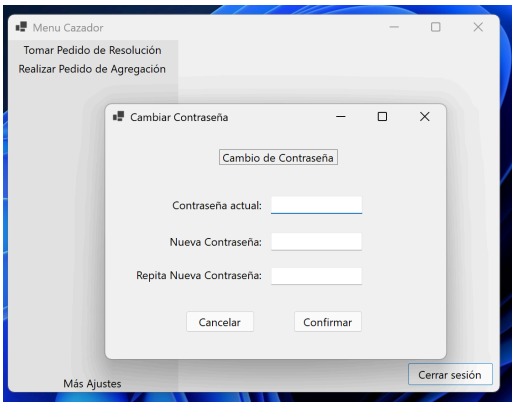


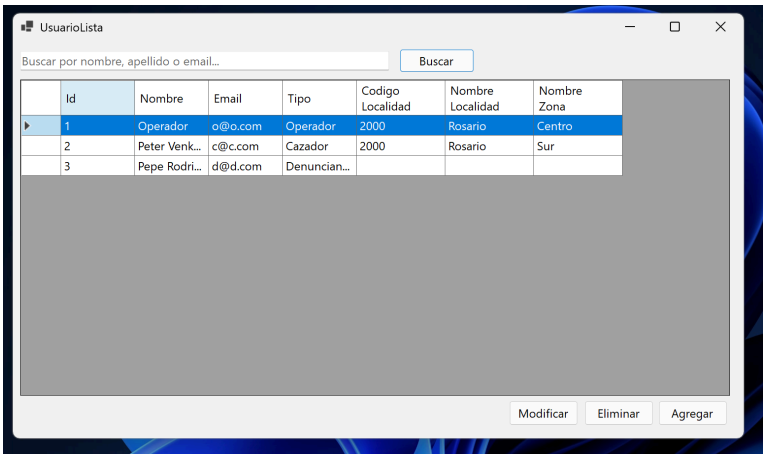




- Windows Forms:



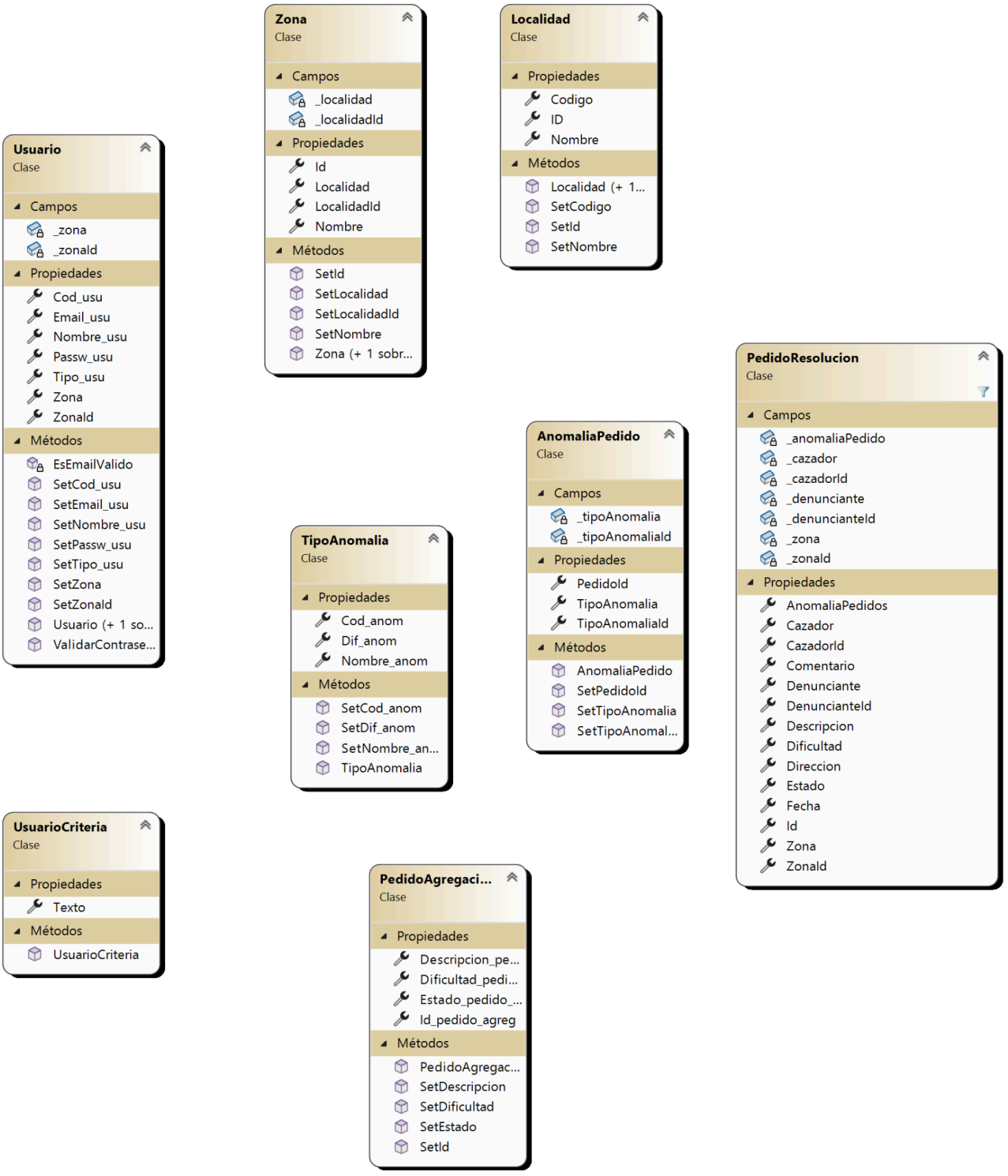




- JWT para seguridad y autenticación:

```
1 referencia
public async Task<LoginResponse?> LoginAsync(LoginRequest request)
{
    if (string.IsNullOrWhiteSpace(request.Email) || string.IsNullOrWhiteSpace(request.Password))
        return null;

    var usuario = await usuarioRepository.GetByEmailAsync(request.Email);

    if (usuario == null || !usuario.ValidatePassword(request.Password))
        return null;

    var token = GenerateJwtToken(usuario);
    var expiresAt = DateTime.UtcNow.AddMinutes(GetExpirationMinutes());

    return new LoginResponse
    {
        Token = token,
        ExpiresAt = expiresAt,
        Email = usuario.Email_usu
    };
}
```

- Swagger/OpenAPI (para desarrollo).
- ADO.NET.
- LINQ.

## Modelo de Objetos



## Modelo de Datos