

# Documentación Técnica - EAAB AddIn para ArcGIS Pro

---

## Descripción General

AddIn para ArcGIS Pro que proporciona capacidades de geocodificación individual y masiva mediante conexión a bases de datos corporativas PostgreSQL y Oracle, con soporte para conexiones directas por credenciales y mediante archivos SDE. Incluye también búsqueda de Puntos de Interés (POIs), almacenamiento estructurado de resultados y un sistema completo de migración de datos de redes de acueducto y alcantarillado con validación automática y transformación de geometrías.

## Conformidad con Norma NS-046

El módulo de migración de datos está diseñado conforme a la **Norma Técnica NS-046 de la Empresa de Acueducto y Alcantarillado de Bogotá (EAAB-ESP)**, que establece:

- **Estructura de datos espaciales** para redes de acueducto y alcantarillado
- **Clasificación de elementos** según tipos de red (sanitario, pluvial, acueducto)
- **Nomenclatura y códigos** de feature classes y dominios
- **Atributos mínimos requeridos** para cada tipo de elemento de red
- **Sistemas de referencia espacial** MAGNA-SIRGAS / Colombia Bogotá (WKID: 102233, EPSG: 6247)
- **Topología y conectividad** de redes de infraestructura hidráulica

El sistema implementa los mapeos de clasificación, nomenclatura de feature classes, estructura de atributos y validaciones especificadas en la norma NS-046.

## Stack Tecnológico

- **.NET 8**
- **ArcGIS Pro SDK 3.4+**
- **WPF** con patrón MVVM
- **PostgreSQL 15+** con PostGIS
- **Oracle 18+**
- **EPPlus / ExcelDataReader** para lectura de Excel (según implementación)
- **Npgsql / Oracle Managed Data Access**

## Requisitos de Desarrollo

### Entorno de desarrollo

- Visual Studio 2022 o superior
- ArcGIS Pro SDK for .NET instalado (Extension Manager)
- ArcGIS Pro 3.4+ instalado (mismo equipo)
- .NET 8 SDK

### Dependencias del proyecto

```
<PackageReference Include="ArcGIS.Desktop.SDK" Version="3.4.*" />
<PackageReference Include="EPPlus" Version="7.0+" />
<PackageReference Include="ExcelDataReader" Version="3.6+" />
<PackageReference Include="Npgsql" Version="8.0+" />
<PackageReference Include="Oracle.ManagedDataAccess.Core" Version="3.21+" />
```

# Arquitectura del Sistema

## Estructura de Carpetas (simplificada)

```
EAABAddIn/
├── Src/
│   ├── Application/                # Capa de aplicación (casos de uso / servicios
orquestadores)
│   │   ├── Services/
│   │   │   ├── AddressNormalizer.cs
│   │   │   ├── AddressSearchService.cs
│   │   │   ├── CsvReportService.cs
│   │   │   └── HashService.cs
│   │   ├── UseCases/
│   │   │   ├── MigrateAlcantarilladoUseCase.cs
│   │   │   ├── MigrateAcueductoUseCase.cs
│   │   │   ├── CreateGdbFromXmlUseCase.cs
│   │   │   ├── ClipFeatureDatasetUseCase.cs
│   │   │   ├── GenerarHashUseCase.cs
│   │   │   └── Validation/
│   │   │       └── ValidateDatasetsUseCase.cs
│   │   └── DTOs/
│   └── Core/                      # Transversal: conexión, capa de datos base,
utilidades
│   ├── Config/
│   │   ├── ConfigurationManager.cs
│   │   ├── DatabaseConfiguration.cs
│   │   └── PersistentSettings.cs
│   ├── Data/
│   │   ├── DatabaseConnectionService.cs
│   │   ├── ConnectionPropertiesFactory.cs
│   │   ├── ResultsLayerService.cs
│   │   ├── AddressNotFoundTableService.cs
│   │   ├── PoiResultsLayerService.cs
│   │   └── Repositories/
│   │       ├── PtAddressGralEntityRepository.cs
│   │       ├── OraclePtAddressGralRepository.cs
│   │       ├── PostgresPtAddressGralRepository.cs
│   │       └── PoiRepository*.cs (si aplica)
│   └── Map/ (servicios de mapa, transformaciones)
```

```
└─ Presentation/                                # MVVM UI
    └─ Converters/
    └─ View/
        └─ AddressSearchView.xaml
        └─ MassiveGeocodingView.xaml
        └─ MigrationView.xaml
        └─ GeneradorHashView.xaml
        └─ ClipFeatureDatasetView.xaml
        └─ Dockpanes/
            └─ MigrationDockpaneView.xaml
    └─ ViewModel/
        └─ AddressSearchViewModel.cs
        └─ MassiveGeocodingViewModel.cs
        └─ MigrationViewModel.cs
        └─ GeneradorHashViewModel.cs
        └─ GeneradorHashGenerarViewModel.cs
        └─ GeneradorHashVerificarViewModel.cs
        └─ ClipFeatureDatasetViewModel.cs
        └─ Dockpanes/
            └─ MigrationDockpaneViewModel.cs
└─ Images/                                     # Recursos gráficos
└─ Config.daml                                # Configuración AddIn / DAML (UI ArcGIS Pro)
└─ Module1.cs                                 # Punto de entrada / ciclo de vida
```

Capas y Responsabilidades

Capa	Responsabilidad	Ejemplos
Presentation	Interacción usuario, binding WPF	<i>ViewModels, XAML Views</i>
Application	Orquestación de casos de uso	<i>MassiveGeocodingService</i>
Core.Data	Conexión y repositorios	<i>DatabaseConnectionService</i>
Core.Map	Creación de capas, escritura espacial	<i>ResultsLayerService</i>
Domain (implícita)	Entidades lógicas	<i>PtAddressGralEntity</i>

Flujo de Geocodificación Individual

- 1. Usuario ingresa dirección y ciudad.
- 2. ViewModel invoca **AddressSearchService**.
- 3. El servicio consulta repositorio principal (EAAB).
- 4. Si vacío, fallback a Catastro / ESRI según configuración o heurística.
- 5. Resultados se normalizan (dirección preferida EAAB > Catastro > Original).
- 6. Se envían a **ResultsLayerService** para persistir punto.

Flujo de Geocodificación Masiva

- 1. Lectura de Excel → generación de lista de registros.
- 2. Validaciones: estructura, campos obligatorios, códigos de ciudad válidos.

3. Iteración paralela controlada (en esta versión: secuencial por seguridad ArcGIS MCT).
4. Clasificación de resultados y enriquecimiento de atributos.
5. Batch insert (acumulación + commit).
6. Resumen (found / not found).

## Flujo de Búsqueda de POIs

1. Usuario ingresa término (ej: "hospital").
2. **PoiSearchService** genera un patrón (LIKE normalizado).
3. Repositorio POI ejecuta consulta (index por nombre / categoría).
4. Se limita número máximo (paginación futura).
5. Se inserta en capa **POIResults** o memoria y luego commit.
6. Selección de un resultado centra el mapa.

## Flujo de Migración de Datos

**Conformidad NS-046:** El proceso de migración implementa la estructura de datos y clasificaciones definidas en la Norma Técnica NS-046 de la EAAB-ESP.

### Fase 1: Preparación y Validación

1. Usuario selecciona carpeta de salida, esquema XML (conforme NS-046) y capas de origen (acueducto/alcantarillado).
2. **ValidateDatasetsUseCase** ejecuta validación pre-migración:
  - Verifica existencia de campos requeridos según NS-046 (CLASE, SUBTIPO, SISTEMA)
  - Detecta features sin clasificación o con valores nulos
  - Valida valores de dominios contra catálogos normativos
  - Genera reportes CSV con advertencias por dataset
3. Si hay advertencias y checkbox "Migrar con Advertencias" está desmarcado:
  - Bloquea ejecución y muestra diálogo al usuario
  - Usuario debe revisar reportes y decidir si continuar
4. Si no hay advertencias o usuario autoriza continuar → pasa a Fase 2

### Fase 2: Creación de Geodatabase Destino

1. **CreateGdbFromXmlUseCase** procesa el esquema XML (estructura NS-046):
  - Verifica si ya existe GDB con nombre **Migracion\_YYYYMMDD\_HHmss.gdb**
  - Si existe: reutiliza (permite migraciones incrementales)
  - Si no existe: crea nueva GDB usando ArcGIS Geoprocessing con el XML normativo
  - El XML define feature classes, dominios y relaciones según estándar NS-046
2. Valida que la creación fue exitosa antes de continuar

### Fase 3: Migración de Features

1. Para cada capa de origen seleccionada:
  - **MigrateAlcantarilladoUseCase** o **MigrateAcueductoUseCase** lee features
2. Por cada feature:
  - Lee campo CLASE para determinar tipo de elemento (conforme catálogo NS-046)
  - Lee campo SISTEMA para determinar red (0=sanitario, 1=pluvial, 2=sanitario según NS-046)

- Consulta mapping interno NS-046 para obtener feature class destino
  - Valida que FC destino existe en la GDB con estructura normativa
3. Transformación de geometría:
- Obtiene SR del mapa activo y SR del FC destino (NS-046 requiere MAGNA-SIRGAS Bogotá)
  - Si son diferentes: proyecta geometría automáticamente a WKID 102233/6247
  - Detecta dimensiones Z/M incompatibles con especificación NS-046
  - Reconstruye geometría sin Z/M si es necesario usando builders específicos (MapPointBuilderEx, PolylineBuilderEx, PolygonBuilderEx)
4. Mapeo de atributos (conforme NS-046):
- `BuildLineAttributes()` o `BuildPointAttributes()` mapea campos origen → destino según estructura normativa
  - Aplica coerción de tipos según definición de campos destino en NS-046
  - Trunca strings que exceden longitud máxima especificada en la norma
  - Mapea códigos de dominios a valores válidos del catálogo NS-046
5. Inserción:
- Intenta inserción con `EditOperation` (método seguro ArcGIS)
  - Si falla, registra error detallado
  - Continúa con siguiente feature (no detiene proceso completo)

#### Fase 4: Post-Procesamiento

1. `CsvReportService` genera reportes de migración:
  - Resumen por cada feature class destino
  - Total intentado vs migrado vs fallido
  - Features sin CLASE o sin clase destino
2. `AddMigratedLayersToMap()` agrega capas al mapa:
  - Verifica si capa ya existe en el mapa
  - Si existe y apunta a la misma fuente: reutiliza y aplica simbología
  - Si no existe: crea nueva capa con `LayerFactory`
  - Aplica simbología predefinida (verde para líneas, naranja para puntos)
  - Calcula extent combinado de todas las capas migradas
  - Ejecuta ZoomTo con extent expandido 10%
3. Muestra resumen final al usuario

#### Mapeo de Clasificaciones (Alcantarillado - Conforme NS-046):

##### Nomenclatura según NS-046:

- Prefijo `als_`: Alcantarillado Sanitario
- Prefijo `alp_`: Alcantarillado Pluvial
- Campo SISTEMA: 0 o 2 = Sanitario, 1 = Pluvial

##### Líneas (NS-046 Tabla de Clasificación de Redes):

- CLASE=1 + SISTEMA=0/2 → `als_RedLocal` (red distribución local sanitaria)
- CLASE=1 + SISTEMA=1 → `alp_RedLocal` (red distribución local pluvial)
- CLASE=2 + SISTEMA=0/2 → `als_RedTroncal` (colector principal sanitario)
- CLASE=2 + SISTEMA=1 → `alp_RedTroncal` (colector principal pluvial)
- CLASE=3 + SISTEMA=0/2 → `als_Linealateral` (acometida sanitaria)

- CLASE=3 + SISTEMA=1 → `alp_LineaLateral` (acometida pluvial)
- CLASE=4 → igual que CLASE=1 (RedLocal - equivalencia normativa)

### Puntos (NS-046 Catálogo de Estructuras):

- CLASE=1 → `als/alp_EstructuraRed` (estructura especial)
- CLASE=2 → `als/alp_Pozo` (cámara de inspección)
- CLASE=3 → `als/alp_Sumidero` (captación aguas lluvia)
- CLASE=4 → `als/alp_CajaDomiciliaria` (punto de conexión)
- CLASE=5 → `als/alp_SeccionTransversal` (punto de medición)
- CLASE=6 → `als/alp_EstructuraRed` (estructura auxiliar)
- CLASE=7 → `als/alp_Sumidero` (captación vial)

### Valores de dominios NS-046:

- DOMTIPOSISTEMA: "0"=Sanitario, "1"=Pluvial, "2"=Sanitario (legacy)
- DOMESTADOENRED: Según catálogo de estados operacionales NS-046
- DOMMATERIAL: Códigos de materiales estándar EAAB
- DOMTIPOSECCION: Circular, Rectangular, Ovoide, etc. (catálogo NS-046)

## Patrones de Diseño Implementados

### Patrones Principales

- **MVVM** (Model-View-ViewModel): Separación de lógica de presentación
- **Factory** (ConnectionFactory): Creación de conexiones según motor de BD
- **Repository** (PtAddressGralEntityRepository, PoiRepository): Abstracción de acceso a datos
- **Singleton Controlado** (Module1, ConnectionService): Instancias únicas de servicios
- **Strategy (fallback)**: Múltiples estrategias de búsqueda (exacta → LIKE → ESRI)
- **Lazy Initialization**: Creación diferida de recursos (feature classes, capas)

### Patrones en Migración

- **Use Case Pattern**: Cada operación de migración es un caso de uso aislado
- **Template Method**: `MigrateLines()` y `MigratePoints()` siguen estructura común
- **Builder Pattern**: Reconstrucción de geometrías con builders específicos
- **Adapter**: Normalización entre fuentes de resultados EAAB, Catastro, ESRI hacia `GeocodeResult`
- **Chain of Responsibility**: Validación → Creación GDB → Migración → Reportes
- **Command Pattern**: `EditOperation` encapsula operaciones de edición
- **Fail-Fast Validation**: Valida estructura antes de consumir recursos de migración
- **Safe Fallback**: Si inserción directa falla, intenta con `EditOperation`

## Componentes Principales de Migración

### 1. MigrationViewModel.cs

Orquesta el flujo completo de migración desde la interfaz de usuario.

```
internal class MigrationViewModel : BusyViewModelBase
{
    // Propiedades vinculadas a UI
    public bool MigrarConAdvertencias { get; set; } // Checkbox crítico
    public string? Workspace { get; set; } // Carpeta salida
    public string? XmlSchemaPath { get; set; } // Esquema XML
    public string? L_Acu_Origen { get; set; } // Líneas acueducto
    public string? P_Acu_Origen { get; set; } // Puntos acueducto
    // ... (6 capas posibles)

    private async Task RunAsync()
    {
        // 1. Validar parámetros
        if (Workspace is null || XmlSchemaPath is null) return;

        // 2. Construir lista de datasets a validar
        var datasetsToValidate = new List<DatasetInput>();
        if (!string.IsNullOrEmpty(L_Acu_Origen))
            datasetsToValidate.Add(new("L_ACU_ORIGEN", L_Acu_Origen));
        // ... repite para cada capa

        // 3. Ejecutar validación
        var validation = await _datasetValidatorUseCase.Invoke(new() {
            OutputFolder = Workspace,
            Datasets = datasetsToValidate
        });

        // 4. Verificar advertencias vs checkbox
        if (validation.TotalWarnings > 0 && !MigrarConAdvertencias)
        {
            // Bloquear con diálogo detallado
            MessageBox.Show($"BLOQUEADO: {validation.TotalWarnings}
advertencias...");
            return;
        }

        // 5. Crear GDB destino
        var (okGdb, gdbPath, msgGdb) = await _createGdbFromXmlUseCase
            .Invoke(Workspace, XmlSchemaPath);

        // 6. Migrar cada capa seleccionada
        if (!string.IsNullOrEmpty(L_Alc_Origen))
        {
            var (ok, msg, warnings) = await _migrateAlcantarilladoUseCase
                .MigrateLines(L_Alc_Origen, gdbPath);
            mensajes.Add(msg);
        }
        // ... repite para cada tipo

        // 7. Agregar capas al mapa
        await _migrateAlcantarilladoUseCase.AddMigratedLayersToMap(gdbPath);

        // 8. Mostrar resumen
    }
}
```

```
        MessageBox.Show($"Completado:\n{string.Join("\n", mensajes)}");
    }
}
```

### Características clave:

- **Validación obligatoria** antes de migración
- **Bloqueo de seguridad** si hay advertencias sin autorización explícita
- **Procesamiento secuencial** de múltiples capas
- **Reportes CSV** automáticos por cada dataset
- **Gestión de errores** sin detener proceso completo

## 2. ValidateDatasetsUseCase.cs

Valida estructura y contenido de datos antes de migración.

```
public class ValidateDatasetsUseCase
{
    public async Task<ValidationResult> Invoke(ValidationInput input)
    {
        var result = new ValidationResult
        {
            ReportFolder = Path.Combine(input.OutputFolder,
"Reportes_Validacion"),
            ReportFiles = new List<string>()
        };

        foreach (var dataset in input.Datasets)
        {
            using var fc = OpenFeatureClass(dataset.Path);
            var warnings = new List<ValidationWarning>();

            // Verificar campos requeridos
            if (!HasField(fc, "CLASE"))
                warnings.Add(new("Campo CLASE no encontrado", "CRITICAL"));

            // Validar contenido
            using var cursor = fc.Search();
            while (cursor.MoveNext())
            {
                var clase = GetFieldValue<int?>(feature, "CLASE");
                if (!clase.HasValue || clase.Value == 0)
                    warnings.Add(new("Feature sin CLASE válida", "WARNING"));

                var targetClass = GetTargetClassName(clase, sistema);
                if (string.IsNullOrEmpty(targetClass))
                    warnings.Add(new("Sin clase destino para CLASE=" + clase,
"WARNING"));
            }

            // Generar reporte CSV
        }
    }
}
```

```

        var csvFile = WriteCsvReport(result.ReportFolder, dataset.Name,
warnings);
        result.ReportFiles.Add(csvFile);
        result.TotalWarnings += warnings.Count;
    }

    return result;
}
}

```

### Validaciones ejecutadas:

- Existencia de campos CLASE, SUBTIPO, SISTEMA
- Tipos de datos correctos
- Features con CLASE nulo o cero
- CLASE sin mapping a feature class destino
- Geometrías nulas o vacías

### 3. MigrateAlcantarilladoUseCase.cs

Ejecuta la migración real de features de alcantarillado.

```

public class MigrateAlcantarilladoUseCase
{
    public async Task<(bool ok, string message)> Migratelines(
        string sourceLineasPath, string targetGdbPath)
    {
        return await QueuedTask.Run(() =>
        {
            using var sourceFC = OpenFeatureClass(sourceLineasPath);
            using var targetGdb = new Geodatabase(
                new FileGeodatabaseConnectionPath(new Uri(targetGdbPath)));

            var mapSpatialReference = MapView.Active?.Map?.SpatialReference;
            int migrated = 0, failed = 0, noClase = 0, noTarget = 0;

            using var cursor = sourceFC.Search();
            while (cursor.MoveNext())
            {
                var feature = cursor.Current as Feature;
                var clase = GetFieldValue<int?>(feature, "CLASE");
                var sistema = GetFieldValue<int?>(feature, "SISTEMA");

                if (!clase.HasValue) { noClase++; continue; }

                string targetClassName = GetTargetLineClassName(clase.Value,
sistema?.ToString());
                if (string.IsNullOrEmpty(targetClassName)) { noTarget++; continue;
            }

            if (MigrateLineFeature(feature, targetGdb, targetClassName,

```

```
        subtipo, out var error, mapSpatialReference))
        migrated++;
    else
        failed++;
}

// Generar reporte CSV
var csv = new CsvReportService();
csv.WriteMigrationSummary(folder, "alcantarillado_lineas", stats);

return (true, $"Migradas: {migrated}, Fallidas: {failed}");
});
}

private bool MigratelineFeature(Feature source, Geodatabase targetGdb,
    string targetClassName, int subtipo, out string? error,
    SpatialReference? mapSR)
{
    using var targetFC = OpenTargetFeatureClass(targetGdb, targetClassName);
    var geometry = source.GetShape();

    // Proyección si es necesaria
    var targetSR = targetFC.GetDefinition().GetSpatialReference();
    if (mapSR != null && geometry.SpatialReference.Wkid != targetSR.Wkid)
    {
        geometry = GeometryEngine.Instance.Project(geometry, targetSR);
    }

    // Ajuste Z/M si es necesario
    if ((geometry.HasZ && !targetFC.GetDefinition().HasZ()) ||
        (geometry.HasM && !targetFC.GetDefinition().HasM()))
    {
        geometry = RebuildGeometryWithoutZM(geometry);
    }

    // Mapeo de atributos
    var attributes = BuildLineAttributes(source, targetFC.GetDefinition(),
subtipo);

    // Inserción con EditOperation
    var editOp = new EditOperation { Name = "Migrar línea" };
    editOp.Callback(context =>
    {
        using var rowBuffer = targetFC.CreateRowBuffer();
        rowBuffer["SHAPE"] = geometry;
        foreach (var attr in attributes)
            rowBuffer[attr.Key] = CoerceToFieldType(attr.Value, fieldDef);
        using var row = targetFC.CreateRow(rowBuffer);
        context.Invalidate(row);
    }, targetFC);

    bool success = editOp.Execute();
    error = success ? null : editOp.ErrorMessage;
    return success;
}
```

```
}

private Geometry RebuildGeometryWithoutZM(Geometry geom)
{
    if (geom is MapPoint pt)
        return MapPointBuilderEx.CreateMapPoint(pt.X, pt.Y,
geom.SpatialReference);

    if (geom is Polyline line)
    {
        var builder = new PolylineBuilderEx(geom.SpatialReference);
        foreach (var part in line.Parts)
        {
            var points = new List<MapPoint>();
            foreach (var segment in part)
            {
                var startPt = segment.StartPoint;
                points.Add(MapPointBuilderEx.CreateMapPoint(
                    startPt.X, startPt.Y, geom.SpatialReference));
            }
            // Agregar último punto
            var lastSeg = part[part.Count - 1];
            var endPt = lastSeg.EndPoint;
            points.Add(MapPointBuilderEx.CreateMapPoint(
                endPt.X, endPt.Y, geom.SpatialReference));

            builder.AddPart(points);
        }
        return builder.ToGeometry();
    }

    // Similar para Polygon...
    return geom;
}

public async Task<(bool ok, string message)> AddMigratedLayersToMap(string
gdbPath)
{
    var map = MapView.Active?.Map;
    var combinedExtent = (Envelope?)null;

    var lineClasses = new[] { "als_RedLocal", "als_RedTroncal", ... };
    foreach (var className in lineClasses)
    {
        using var fc = OpenFeatureClass(gdb, className);
        if (fc == null || fc.GetCount() == 0) continue;

        // Calcular extent
        using var cursor = fc.Search();
        while (cursor.MoveNext())
        {
            var geom = (cursor.Current as Feature)?.GetShape();
            if (geom?.Extent != null)
                combinedExtent = combinedExtent?.Union(geom.Extent) ??

```

```

geom.Extent;
    }

    // Crear o reutilizar capa
    var existingLayer =
map.GetLayersAsFlattenedList().OfType<FeatureLayer>()
    .FirstOrDefault(l => l.Name.Equals(className,
StringComparison.OrdinalIgnoreCase));

    if (existingLayer == null)
    {
        var layer = LayerFactory.Instance.CreateLayer<FeatureLayer>(
            new FeatureLayerCreationParams(fc) { Name = className }, map);
        ApplySymbology(layer, isLine: true);
    }
}

// Zoom a extent combinado
if (combinedExtent != null)
{
    var expanded = new EnvelopeBuilderEx(
        combinedExtent.XMin - width * 0.1,
        combinedExtent.YMin - height * 0.1,
        combinedExtent.XMax + width * 0.1,
        combinedExtent.YMax + height * 0.1,
        combinedExtent.SpatialReference
    ).ToGeometry();

    MapView.Active.ZoomTo(expanded, TimeSpan.FromSeconds(1.5));
}

return (true, "Capas agregadas exitosamente");
}
}

```

### Operaciones críticas:

- **OpenFeatureClass():** Maneja shapefiles, GDB, feature datasets automáticamente
- **GetTargetLineClassName():** Mapea CLASE+SISTEMA → nombre de FC destino
- **RebuildGeometryWithoutZM():** Reconstruye geometría eliminando dimensiones extra
- **CoerceToFieldType():** Convierte y trunca valores según tipo de campo destino
- **EditOperation:** Método seguro para inserción en contexto de ArcGIS

## 4. CsvReportService.cs

Genera reportes CSV de validación y migración.

```

public class CsvReportService
{
    public string WriteMigrationSummary(string folder, string prefix,
        IEnumerable<(string className, int attempts, int migrated, int failed)>

```

```

stats,
    int noClase, int noTarget)
{
    var timestamp = DateTime.Now.ToString("yyyyMMdd_HH:mm:ss");
    var fileName = $"{prefix}_resumen_{timestamp}.csv";
    var filePath = Path.Combine(folder, fileName);

    using var writer = new StreamWriter(filePath);
    writer.WriteLine("Clase Destino,Intentos,Migradas,Fallidas");

    foreach (var (className, attempts, migrated, failed) in stats)
    {
        writer.WriteLine($"{className},{attempts},{migrated},{failed}");
    }

    writer.WriteLine();
    writer.WriteLine($"Sin campo CLASE,{noClase}");
    writer.WriteLine($"Sin clase destino,{noTarget}");

    return filePath;
}
}

```

### ### 5. ClipFeatureDataset (Corte de Feature Dataset)

Descripción: Herramienta para **recortar** (**clip**) un conjunto de Feature Classes contenidos en un Feature Dataset usando un polígono seleccionado en el mapa. Genera una geodatabase de salida en la carpeta seleccionada con un nombre ``Clip_YYYYMMDD_HH:mm:ss.gdb`` e incluye opciones de buffer y selección por feature class.

Componente principal: ``ClipFeatureDatasetViewModel``

#### - Propiedades clave:

- ``OutputGeodatabase``: carpeta donde se creará la GDB de salida.
- ``FeatureDataset``: ruta al Feature Dataset de **origen** (`.gdb/.../FeatureDataset`).
- ``AvailableFeatureClasses`` / ``FilteredFeatureClasses``: lista de feature classes seleccionables dentro del dataset.
- ``BufferMeters``, ``IsBufferEnabled``, ``IsRoundedBufferEnabled``: opciones de tamponamiento aplicadas al polígono de recorte.
- ``OutputLocation``: ruta de la GDB o carpeta de **salida** (**expuesta para UI / hipervínculo**).

#### - Comandos:

- ``OutputGeodatabaseCommand``: examina la carpeta de salida.
- ``FeatureDatasetCommand``: examina y selecciona el Feature Dataset de origen.
- ``ExecuteClipCommand``: ejecuta el proceso de **clip** (**llama a ``ClipFeatureDatasetUseCase``**).
- ``OpenOutputLocationCommand``: abre la carpeta de salida en el Explorador de **Windows** (**hipervínculo en la UI**).

#### - Flujo técnico resumido:

1. El usuario selecciona el Feature Dataset de entrada y la carpeta de salida.

2. El ViewModel carga las Feature Classes del dataset y permite seleccionar cuáles recortar.
3. El usuario selecciona un polígono en el mapa (debe existir exactamente un polígono seleccionado).
4. Opcionalmente define un buffer en metros y el tipo (redondeado/plano).
5. Al ejecutar, el caso de uso crea una GDB nueva `Clip\_YYYYMMDD\_HH:mm:ss.gdb` en la carpeta padre, ejecuta el clip por cada feature class seleccionada y registra el resultado en `StatusMessage`.
6. `OutputLocation` se expone y la UI muestra un hipervínculo para abrir la carpeta en Explorer.

Notas de implementación:

- El proceso utiliza `QueuedTask.Run` para operaciones con ArcGIS SDK (apertura de geodatabases, lectura de definiciones, operaciones de clip).
- La UI expone `HasOutputLocation` para controlar la visibilidad del hipervínculo.
- Se recomienda comprobar permisos de escritura en la carpeta de salida antes de ejecutar.

## 6. Hash SHA-256 (Generar y Verificar)

Descripción: Módulo para generar y verificar firmas SHA-256 de archivos de salida del AddIn (por ejemplo: `Migracion_*.gdb` empaquetadas, archivos `.zip` generados por procesos, y el propio paquete `.esriAddInX`). Permite asegurar integridad y facilitar validación en despliegues.

Componentes principales:

- `HashViewModel` (Presentation): expone comandos de UI y estado de operación.
- `FileHashService` (Application/Core): calcula y verifica hash con lectura en streaming.
- Comandos: `GenerateHashCommand`, `VerifyHashCommand` (mapeados en DAML).

UI (DAML - conceptual):

```
<toolGroup id="EAABAddIn_HashGroup" caption="Hash">
  <button id="EAABAddIn_GenerateHash" caption="Generar Hash" />
  <button id="EAABAddIn_VerifyHash" caption="Verificar Hash" />
</toolGroup>
```

Flujo técnico: Generar Hash

1. Selección de archivo(s) objetivo(s) mediante diálogo (ZIP, GDB empaquetada, esriAddInX, etc.).
2. Cálculo SHA-256 en streaming usando `System.Security.Cryptography.SHA256.Create()` con `FileStream` y buffer (ej. 1–4 MB) para evitar cargas completas en memoria.
3. Se genera un archivo resumen por cada archivo de entrada con nombre `<NombreArchivo>_HASH.txt` en la misma carpeta, con el siguiente formato de texto:
  - Archivo: `<nombre.ext>`
  - Ubicación: `<ruta completa>`
  - Algoritmo: SHA-256
  - TamañoBytes: `<entero>`
  - Hash: `<hex en mayúsculas sin separadores>`

- **Fecha:** <YYYY-MM-DD HH:mm:ss>

- Manejo de errores: archivos bloqueados, permisos insuficientes, rutas largas → registrar mensaje y continuar con el resto.

#### Flujo técnico: Verificar Hash

- Selección de archivo a verificar o carpeta que contenga el par <archivo> y <archivo>\_HASH.txt.
- Búsqueda no recursiva del archivo de resumen en la misma carpeta; si existe más de uno, se prioriza el que coincide exactamente con el nombre base.
- Recalcular SHA-256 del archivo objetivo en streaming y comparar con el valor en **Hash:** del archivo de resumen.
- Resultado de verificación:
  - **OK:** hash coincide.
  - **NO MATCH:** contenido difiere.
  - **NO HASH:** no se encontró archivo resumen.
- Registro de detalles en **StatusMessage** y, opcionalmente, generación de **Verificacion\_HASH\_[timestamp].txt** con el resultado.

#### Consideraciones de implementación

- Lectura por bloques (**FileStream** con **useAsync:true**) para archivos grandes.
- Cancelación y reporte de progreso opcionales vía **IProgress<T>/CancellationToken**.
- Operaciones de IO pesadas en **Task.Run/QueuedTask** según corresponda, actualizando UI en el hilo apropiado.
- Nombres y rutas conservando carpeta origen; no se realiza búsqueda recursiva.

#### Ejemplo (C# simplificado de cómputo):

```
public static string ComputeSha256(string path)
{
    using var sha = SHA256.Create();
    using var fs = new FileStream(path, FileMode.Open, FileAccess.Read,
        FileShare.Read, 1_048_576, FileOptions.SequentialScan);
    var hash = sha.ComputeHash(fs);
    return Convert.ToHexString(hash); // .NET 5+: HEX en mayúsculas
}
```

#### Pruebas recomendadas

- Archivos pequeños y grandes (>1 GB) para validar rendimiento/streaming.
- Casos: archivo inexistente, archivo bloqueado, carpeta sin **\_HASH.txt**, valor **Hash:** mal formado.
- Verificación cruzada con herramienta externa (ej. **Get-FileHash** en PowerShell) para validar el resultado.

## Componentes Principales Geocodificación

### 1. Module1.cs - Punto de Entrada

```
internal class Module1 : Module
{
    protected override bool Initialize()
    {
        // Cargar configuración persistente
        LoadConfiguration();

        // Inicializar servicios
        InitializeServices();

        // Intentar reconexión diferida
        _ = Task.Run(async () => await AttemptReconnectionAsync());

        return base.Initialize();
    }

    private async Task AttemptReconnectionAsync()
    {
        var config = ConfigurationManager.LoadConfiguration();
        if (config.IsValid)
        {
            await ConnectionService.ConnectAsync(config);
        }
    }
}
```

## 2. DatabaseConnectionService

Gestiona las conexiones a bases de datos con soporte multi-motor.

```
public class DatabaseConnectionService : IDisposable
{
    private Geodatabase _geodatabase;
    private DatabaseEngine _currentEngine;
    private string _sdeFilePath;

    public async Task<bool> ConnectAsync(DatabaseConfiguration config)
    {
        try
        {
            DatabaseConnectionProperties connectionProps;

            if (config.Engine is DatabaseEngine.PostgreSQLSDE or
                DatabaseEngine.OracleSDE)
            {
                // Conexión mediante archivo SDE
                connectionProps = new DatabaseConnectionFile(
                    new Uri(config.SdeFilePath, UriKind.Absolute));
            }
            else
            {
            }
        }
    }
}
```

```

        // Conexión por credenciales
        connectionProps = ConnectionPropertiesFactory.Create(
            config.Engine, config.Host, config.Port,
            config.Database, config.User, config.Password);
    }

    await QueuedTask.Run(() =>
    {
        _geodatabase = new Geodatabase(connectionProps);
    });

    _currentEngine = config.Engine;
    IsConnected = true;

    return true;
}
catch (Exception ex)
{
    Debug.WriteLine($"Error de conexión: {ex.Message}");
    return false;
}
}

public async Task<Geodatabase> GetConnectionAsync()
{
    if (!IsConnected || _geodatabase == null)
        throw new InvalidOperationException("No hay conexión activa");

    return await Task.FromResult(_geodatabase);
}

public void Dispose()
{
    _geodatabase?.Dispose();
    _geodatabase = null;
    IsConnected = false;
}
}

```

### 3. ResultsLayerService

Gestiona la capa de resultados de geocodificación con batch insert optimizado.

```

public class ResultsLayerService
{
    private const string FEATURE_CLASS_NAME = "GeocodedAddresses";

    public async Task InsertBatchAsync(List<GeocodeResult> results)
    {
        var featureClass = await GetOrCreateFeatureClassAsync();
    }
}

```

```
await QueuedTask.Run(() =>
{
    using var operation = new EditOperation
    {
        Name = "Insertar resultados de geocodificación"
    };

    foreach (var result in results)
    {
        var attributes = new Dictionary<string, object>
        {
            ["Identificador"] = result.Identifier,
            ["Direccion"] = result.Address,
            ["FullAddressEAAB"] = result.FullAddressEAAB,
            ["FullAddressUACD"] = result.FullAddressUACD,
            ["Geocoder"] = result.Geocoder.ToString(),
            ["Score"] = result.Score,
            ["ScoreText"] = result.ScoreText,
            ["FechaHora"] = DateTime.Now
        };

        var geometry = MapPointBuilderEx.CreateMapPoint(
            result.X, result.Y, SpatialReferences.WGS84);

        operation.Create(featureClass, attributes, geometry);
    }

    operation.Execute();
});

private FeatureClass CreateOrRetrieveFeatureClass()
{
    var gdb = GetDefaultGeodatabase();

    // Intentar abrir existente
    try
    {
        return gdb.OpenDataset<FeatureClass>(FEATURE_CLASS_NAME);
    }
    catch
    {
        // Crear nuevo
        return CreateNewFeatureClass(gdb);
    }
}

private FeatureClass CreateNewFeatureClass(Geodatabase gdb)
{
    var fcDescription = new FeatureClassDescription(
        FEATURE_CLASS_NAME,
        new List<FieldDescription>
        {
            new FieldDescription("Identificador", FieldType.String),

```

```

        new FieldDescription("Direccion", FieldType.String),
        new FieldDescription("FullAddressEAAB", FieldType.String),
        new FieldDescription("FullAddressUACD", FieldType.String),
        new FieldDescription("Geocoder", FieldType.String),
        new FieldDescription("Score", FieldType.Double),
        new FieldDescription("ScoreText", FieldType.String),
        new FieldDescription("FechaHora", FieldType.Date)
    },
    new ShapeDescription(GeometryType.Point, SpatialReferences.WGS84)
);

return gdb.CreateFeatureClass(fcDescription);
}
}

```

#### 4. AddressSearchService

Orquesta la búsqueda de direcciones con fallback inteligente.

```

public class AddressSearchService
{
    private readonly IPtAddressGralEntityRepository _repository;
    private readonly AddressNormalizer _normalizer;

    public async Task<SearchResult> SearchAsync(
        string address, string city, CancellationToken cancellationToken)
    {
        // Primer intento: búsqueda exacta
        var results = await _repository.SearchAddressesAsync(
            address, city, cancellationToken);

        if (results.Any())
            return new SearchResult(results, SearchStrategy.Exact);

        // Segundo intento: búsqueda LIKE ampliada
        Debug.WriteLine("Búsqueda exacta sin resultados, intentando LIKE...");
        results = await _repository.SearchAddressesLikeAsync(
            address, city, cancellationToken);

        return new SearchResult(results, SearchStrategy.Like);
    }

    public async Task<string> NormalizeAddressAsync(string address)
    {
        try
        {
            return await _normalizer.NormalizeAsync(address);
        }
        catch (Exception ex)
        {
            Debug.WriteLine($"Error de normalización: {ex.Message}");
        }
    }
}

```

```
        return address; // Fallback a dirección original
    }
}
}
```

## 5. MassiveGeocodingService

Procesamiento masivo con optimizaciones de rendimiento.

```
public class MassiveGeocodingService
{
    private readonly AddressSearchService _searchService;
    private readonly ResultsLayerService _resultsService;

    public async Task<MassiveResult> ProcessFileAsync(
        string filePath, IProgress<ProgressInfo> progress)
    {
        var records = await ReadExcelFileAsync(filePath);
        var results = new List<GeocodeResult>();
        var notFound = new List<NotFoundRecord>();

        int processed = 0;
        int total = records.Count;

        foreach (var record in records)
        {
            var searchResult = await _searchService.SearchAsync(
                record.Address, record.City, CancellationToken.None);

            if (searchResult.HasResults)
            {
                // Determinar mejor dirección: EAAB > Catastro > Original
                var bestAddress = DetermineBestAddress(searchResult);
                results.Add(CreateGeocodeResult(record, bestAddress));
            }
            else
            {
                notFound.Add(new NotFoundRecord
                {
                    Identifier = record.Identifier,
                    Address = record.Address,
                    City = record.City,
                    Timestamp = DateTime.Now
                });
            }

            processed++;
            progress?.Report(new ProgressInfo(processed, total));
        }

        // Inserción en lote optimizada
    }
}
```

```
        await _resultsService.InsertBatchAsync(results);
        await _notFoundService.InsertBatchAsync(notFound);

        return new MassiveResult
        {
            Found = results.Count,
            NotFound = notFound.Count,
            Total = total
        };
    }

    private string DetermineBestAddress(SearchResult result)
    {
        // Prioridad: FullAddressEAAB > FullAddressUACD > Original > Calle básica
        var first = result.Addresses.First();

        if (!string.IsNullOrEmpty(first.FullAddressEAAB))
            return first.FullAddressEAAB;

        if (!string.IsNullOrEmpty(first.FullAddressUACD))
            return first.FullAddressUACD;

        return first.Direccion;
    }
}
```

Sistema de Configuración

- Persistencia JSON + respaldo en settings App.
- Validación contextual según tipo de motor.
- Admite cambio caliente (runtime) disparando reconexión.

Seguridad de Configuración

- Contraseñas no se registran en logs.
- Posible mejora: cifrado AES local (pendiente) usando DPAPI Windows.
- Recomendado: restringir permisos de carpeta %AppData%/EAABAddIn.

Modelo de Datos (Lógico Simplificado)

Entidad	Campos clave	Fuente	Notas
PtAddressGralEntity	ID, Direccion, FullAddressEAAB, FullAddressCadastre, Poblacion	BD corporativa	Base principal de direcciones
GeocodeResult	Identifier, Address, Source, Score, Lat/Long	Derivado	Unión de varias fuentes
NotFoundRecord	Identifier, Address, City, Timestamp	Generado	Auditoría de intentos fallidos

Entidad	Campos clave	Fuente	Notas
PoiEntity	Poild, Name, Category, City, X, Y	BD corporativa / vista	Indexable para búsqueda

Capa de Resultados Espaciales

ResultsLayerService

- Lazy create de Feature Class **GeocodedAddresses** (WGS84).
- Inserciones agrupadas dentro de **EditOperation**.
- Campos calculados en el ViewModel (score interpretado).

PoiResultsLayerService

- Similar estrategia: **POIResults** con campos **PoiId**, **Nombre**, **Categoria**, **Ciudad**, **FechaHora**.
- Reutiliza builder de geometrías estándar.

Logging y Observabilidad

Estado actual: logging mínimo mediante **Debug.WriteLine** y mensajes UI.  
Sugerido:

- Introducir **ILogger** (MS.Extensions.Logging) con proveedor simple.
- Niveles: Info (operaciones), Warning (faltantes), Error (excepciones).
- Métricas futuras: tiempo promedio por geocodificación, % éxito.

Rendimiento y Escalabilidad

Área	Riesgo	Mitigación Actual	Mejora Potencial
Masivo secuencial	Lento con >50k filas	Procesamiento controlado	Paralelizar lectura + cola MCT
Acceso BD	Latencia variable	Repositorio único	Cache ciudades en memoria
Insert espacial	Bloqueos si muchas ediciones	Batch + single commit	Chunk configurable
Normalización externa	Timeout/errores	Fallback inmediato	Circuit breaker + retry

Tratamiento de Errores

- Validaciones previas detienen proceso temprano (fail-fast).
- Excepciones en loop masivo contabilizan como "no encontrados" sin detener el resto.
- Mostrar mensajes al usuario solo cuando agregan valor (no spam por cada fila).

Pruebas (Testing)

Estrategia Propuesta

Tipo	Objetivo	Ejemplos
Unit	Lógica pura (normalizador, filtros)	AddressNormalizerTests
Repository (mock)	Queries adaptadas por motor	PtAddressGralRepositoryTests
Integration (opcional)	Conexión real a GDB de prueba	Escenarios mínimos
UI (manual)	Flujo MVVM básico	Buscar, Masivo, POI

Recomendaciones

- Introducir interfaces para capa ResultsLayer para facilitar mocks.
- Usar xUnit + Moq.
- Datos de prueba ligeros (JSON) para direcciones.

Build y Empaquetado

Compilación Local

1. Abrir solución en Visual Studio con ArcGIS Pro instalado.
2. Restaurar paquetes NuGet.
3. Asegurar target: net8.0-windows con UseWPF habilitado.
4. Compilar en modo Release.

Generación del AddIn (.esriAddInX)

1. Verificar Config.daml actualizado (botones/paneles).
2. Build Release genera carpeta bin/Release.
3. Utilizar herramienta de empaquetado del SDK (si configurada) o copiar output.
4. Validar firma (si política corporativa lo exige).

Versionado

- Mantener versión en AssemblyInfo o proyecto (PropertyGroup <Version>).
- Sincronizar con sección "Información de Versión" de manual usuario.

Despliegue

Entorno	Acción	Notas
Usuario final	Distribuir .esriAddInX	Instrucciones en READMEUSER
Piloto	Revisión funcional	Capturar métricas básicas
Producción	Publicación controlada	Registrar hash archivo

Checklist de Publicación

- Generar build Release (.esriAddInX).

- Usar la opción **Generar Hash** para el `.esriAddInX` y adjuntar el archivo `<AddIn>.esriAddInX_HASH.txt` al entregable.
- Si se distribuyen datos (GDB/ZIP) generados por el AddIn, generar su `*_HASH.txt` correspondiente y conservar junto al archivo original.
- Verificar integridad con **Verificar Hash** en un equipo distinto antes de entrega final.
- Registrar el valor SHA-256 en el acta o sistema de despliegue.

## Seguridad y Acceso a Datos

- Principio de mínimo privilegio para usuarios de BD.
- No almacenar contraseñas en texto plano fuera de `%AppData%` (cifrar futuro).
- Validar origen de archivos Excel (no macros, no binarios maliciosos).

## Internacionalización (i18n)

- Textos actualmente en español embebidos.
- Mejora futura: recursos (.resx) para soportar EN/ES.

## Roadmap Propuesto

Prioridad	Feature	Descripción
Alta	Cancelación masiva	Token cancelar proceso en curso
Media	Cache ciudades	Reducir llamadas repetidas
Media	Exportar no encontrados	CSV automático
Media	Paginación POIs	Controlar grandes resultados
Baja	Cifrado credenciales	DPAPI / AES
Baja	Telemetría	Eventos anónimos de uso

## Ejemplo Simplificado de POI Repository

```
public interface IPoiRepository {
    IEnumerable<PoiEntity> Search(string term, string city = null, int max = 500);
}

public class PostgresPoiRepository : IPoiRepository {
    private readonly DatabaseConnectionService _connection;
    public IEnumerable<PoiEntity> Search(string term, string city = null, int max
= 500) {
        // Implementación con ILIKE y limit
    }
}
```

## Consideraciones Especiales de Migración

## Transformación de Geometrías

### Proyección automática:

```
if (sourceSR.Wkid != targetSR.Wkid)
{
    geometry = GeometryEngine.Instance.Project(geometry, targetSR);
}
```

### Eliminación de dimensiones Z/M:

```
// Si origen tiene Z pero destino no lo acepta
if (sourceGeom.HasZ && !targetDef.HasZ())
{
    // Reconstruir sin Z usando builders
    if (geom is Polyline)
    {
        var builder = new PolylineBuilderEx(SR);
        foreach (var part in line.Parts)
        {
            var points2D = ExtractXYOnly(part);
            builder.AddPart(points2D);
        }
        geometry = builder.ToGeometry();
    }
}
```

## Mapeo de Atributos

### Líneas de alcantarillado (60+ campos mapeados):

- Campos técnicos: LONGITUD\_M, PENDIENTE, PROFUNDIDADMEDIA, RUGOSIDAD
- Diámetros: DOMDIAMETRONOMINAL, NUMEROCONDUCTOS
- Secciones: DOMTIPOSECCION, BASE, ALTURA1, ALTURA2, TALUD1, TALUD2
- Cotas: COTARASANTEINICIAL/FINAL, COTACLAVEINICIAL/FINAL, COTABATEAINICIAL/FINAL
- Nodos: N\_INICIAL, N\_FINAL
- Estados: DOMESTADOENRED, DOMCALIDADATO, DOMESTADOLEGAL
- Materiales: DOMMATERIAL, DOMMATERIAL2
- Instalación: DOMMETODOINSTALACION, FECHAINSTALACION
- Inspección: DOMTIPOINSPECCION, DOMGRADOESTRUCTURAL, DOMGRADOOPERACIONAL

### Puntos de alcantarillado (80+ campos mapeados):

- Cotas: COTARASANTE, COTATERRENO, COTAFONDO, PROFUNDIDAD, COTACRESTA
- Estructuras: LARGOESTRUCTURA, ANCHOESTRUCTURA, ALTOESTRUCTURA
- Bombeo: ALTURABOMBEO, VOLUMENBOMBEO, CAUDALBOMBEO, UNIDADESOMBEO
- Estados: DOMESTADOPOZO, DOMESTADOFISICO, DOMESTADOTAPA, DOMESTADOESCALON

- Componentes: DOMTIPOCONO, DOMESTADOCONO, ESTREJILLA, MATREJILLA
- Georreferenciación: NORTE, ESTE, ABSCISA
- Identificación: NOMBRE, IDENTIFIC, CODACTIVO\_FIJO

## Validación de Datos

### Campos críticos validados:

```
// Validación pre-migración
var warnings = new List<ValidationWarning>();

// Campo CLASE obligatorio
if (!HasField(fc, "CLASE"))
    warnings.Add(new("Campo CLASE no encontrado", "CRITICAL"));

// Valores válidos de CLASE
if (clase == null || clase == 0)
    warnings.Add(new($"Feature {oid}: CLASE nulo o cero", "WARNING"));

// Mapping a clase destino
var targetClass = GetTargetClassName(clase, sistema);
if (string.IsNullOrEmpty(targetClass))
    warnings.Add(new($"CLASE={clase}, SISTEMA={sistema} sin mapping", "WARNING"));
```

### Reporte CSV generado:

```
Dataset,Campo,Problema,Severidad,Cantidad
L_ALC_ORIGEN,CLASE,Valor nulo,WARNING,15
L_ALC_ORIGEN,CLASE,Sin clase destino (CLASE=99),WARNING,3
P_ALC_ORIGEN,SISTEMA,Valor nulo,WARNING,42
```

## Sistema de Reportes

### Por cada migración se generan:

1. **Reporte de validación** (antes): `validacion_[dataset]_[timestamp].csv`
2. **Reporte de migración** (después): `alcantarillado_lineas_resumen_[timestamp].csv`

### Estructura reporte de migración:

```
Clase Destino,Intentos,Migradas,Fallidas
als_RedLocal,1250,1248,2
als_RedTroncal,340,340,0
als_LineaLateral,89,87,2

Sin campo CLASE,15
Sin clase destino,3
```

---

## Notas de Mantenimiento

- Revisar compatibilidad ArcGIS Pro antes de subir versión SDK.
- Ejecutar pruebas de regresión después de cambios en repositorios.
- Documentar nuevas columnas añadidas a Feature Class.
- **Sincronizar mapeo de campos** si cambia esquema XML corporativo.
- **Actualizar tablas de clasificación** si se agregan nuevos tipos de red.
- **Validar proyecciones** si se cambia SR estándar del mapa.

## Versión del Documento

**Versión:** 1.3

**Última actualización:** 1 de diciembre de 2025

**Cambios principales:**

- Se agrega documentación técnica del módulo de Hash (SHA-256): generar y verificar, mapeo DAML, flujos y consideraciones.
- Se documenta **ClipFeatureDataset** con propiedades, comandos y flujo técnico.
- Se añade checklist de despliegue con generación y verificación de hash para entregables.