PDET Municipality Boundaries Dataset Integration

Eliana Katherine Cepeda
Juan Diego Gonzalez
Mateo Moreno

Noviembre 1, 2025

1. Resumen Ejecutivo

Fuente: MGN/DANE (municipios) en Shapefile/GeoJSON + listado oficial PDET.
Qué se hizo: Conversión a EPSG:4326, normalización de campos DANE, carga en MongoDB y marcado pdet.
Resultado: Colección geo.municipios con índice 2dsphere y consultas espaciales funcionando.
Commits clave: <hash1>, <hash2>.

2. Objetivos y Alcance

Objetivos

- Importar los límites municipales oficiales a MongoDB.
- Filtrar/etiquetar municipios PDET.
- Habilitar consultas espaciales con índice 2dsphere.

Alcance: nivel municipal (no incluye veredas/centros poblados).
Fuera de alcance: capas temáticas adicionales (vías, POI, edificios).

3. Fuentes, Versionado y Licencia

MGN/DANE — Municipios: <URL oficial>, descargado el <fecha>;
versión <mgn_YYYYMMDD>.
Listado PDET: <URL/CSV/JSON>, descargado el <fecha>; versión <pdet_YYYYMMDD>.
Licencia/uso: <texto o enlace>.
Hashes de integridad (SHA256): ver **/docs/hashes.txt**.

4. Estructura de Datos (post-ETL)

4.1 Diccionario mínimo

Campo	Tipo	Descripción	Ejemplo
cod_dpto	string	Código DANE departamento	"11"
nom_dpto	string	Nombre departamento	"Bogotá"
cod_mpio	string	Código DANE municipio (5 dígitos)	"11001"
nom_mpio	string	Nombre municipio	"Bogotá"
pdet	boolean	true si municipio está en listado PDET	false
geom	GeoJSON	Polygon/MultiPolygon en EPSG:4326	{…}
version	string	Versión de la fuente	"mgn_20251101"
source	string	Trazabilidad de origen	"DANE/MGN 2025-11-01"

5. **Pipeline ETL (reproducible)**

Rutas de trabajo:

/data/raw/ADMINISTRATIVO/ (shp + dbf + prj + shx)
/data/processed/ (salidas GeoJSON/NDJSON)
/scripts/ (automatización)

5.1 Pasos

1. **Export & reproyección a WGS84 (EPSG:4326):**

```
ogr2ogr -t_srs EPSG:4326 \
  data/processed/mgn_mpio_4326.geojson \
  "data/raw/ADMINISTRATIVO/MGN_ADM_MPIO_GRAFICO.shp" \
  -lco RFC7946=YES
```

2. **Normalización de campos + armado de documentos:**

```
jq ' .features
   | map({
       cod_dpto: (.properties.COD_DPTO // .properties.cod_dpto),
       nom_dpto: (.properties.NOM_DPTO // .properties.nom_dpto),
       cod_mpio: (.properties.COD_MPIO // .properties.cod_mpio),
       nom_mpio: (.properties.NOM_MPIO // .properties.nom_mpio),
       geom: .geometry,
       pdet: false,
       version: "<mgn_YYYYMMDD>",
       source:  "DANE/MGN <YYYY-MM-DD>"
     })' \
  data/processed/mgn_mpio_4326.geojson \
  > data/processed/municipios.array.json
```

3. **Carga en MongoDB e índices:**

```
mongoimport --db geo --collection municipios --file data/processed/municipios.array.json
--jsonArray
mongosh --eval 'use geo; db.municipios.createIndex({ geom: "2dsphere" })'
mongosh --eval 'use geo; db.municipios.createIndex({ cod_mpio: 1 }, { unique: true })'
```

4. **Marcado PDET (join por cod_mpio):**
   - Convertir el CSV oficial a NDJSON ({"cod_mpio":"XXXXX"} por línea).
   - Importar lista y actualizar flag:

```
// mongosh
use geo;
const codes = db.pdet_list.distinct('cod_mpio');
db.municipios.updateMany({ cod_mpio: { $in: codes } }, { $set: { pdet: true } });
```

5. **Outputs:**
   - **Colección:** geo.municipios
   - **Índices:** geom:2dsphere, cod_mpio:unique
   - **Artefactos:** municipios.array.json, pdet_list.ndjson

6. Verificación de Integridad (QA/QC)

6.1 Checks automáticos

```
use geo;
// Conteos
db.municipios.countDocuments();
db.municipios.countDocuments({ pdet: true });

// Duplicados por cod_mpio (debe ser 0)
db.municipios.aggregate([
  { $group: { _id: "$cod_mpio", n: { $sum: 1 } } },
  { $match: { n: { $gt: 1 } } }
]);

// Campos obligatorios no nulos
db.municipios.countDocuments({ $or: [ {cod_mpio: {$in:[null,""]}}, {geom: {$exists:false}} ] });
```

6.2 Prueba de consulta espacial

```
db.municipios.findOne({
  pdet: true,
  geom: { $geoIntersects: { $geometry: { type:"Point", coordinates:[<lon>, <lat>] } } }
}, { cod_mpio:1, nom_mpio:1 });
```

7. Evidencia y Reproducibilidad

8. Decisiones de Diseño

9. Limitaciones y Supuestos

10. Criterios de Aceptación (Checklist)

11. Anexos