

PDET Municipality Boundaries Dataset Integration

Segunda entrega

Eliana Katherine Cepeda

Juan Diego Gonzalez

Mateo Moreno

Noviembre 3, 2025

1. Resumen Ejecutivo

Fuente: MGN/DANE (municipios) en Shapefile/GeoJSON + listado oficial PDET.

Qué se hizo: normalización de campos DANE, carga en MongoDB y filtrado PDET.

Resultado: Colección geo.municipios con índice 2dsphere y consultas espaciales funcionando.

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).

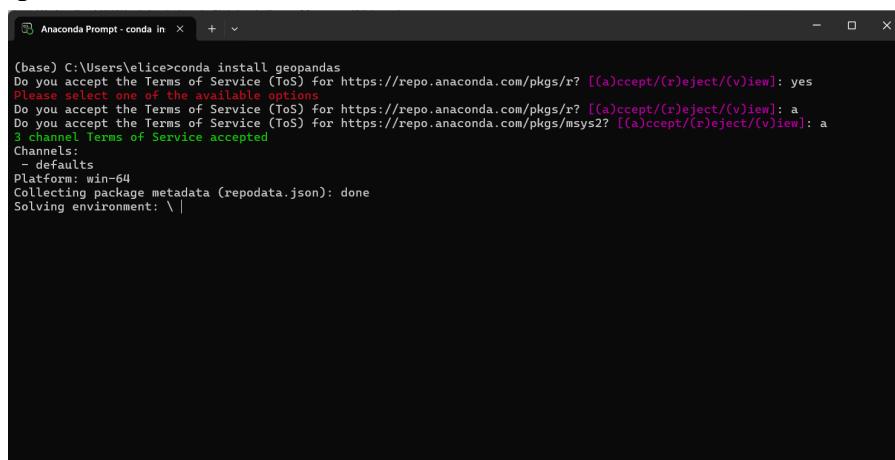
Fuera de alcance: Capas temáticas adicionales (vías, POI, edificios).

3. Fuentes, Versionado y Licencia

- **MGN/DANE-Municipios:** <https://geoportal.dane.gov.co/servicios/descarga-y-metadatos/datos-geoestadisticos/?cod=111> [cite: 36, 37]
- **Listado-PDET:** <https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fcentral-pdet.renovacionterritorio.gov.co%2Fwp-content%2Fuploads%2F2022%2F01%2FMunicipios-PDET.xlsx&wdOrigin=BROWSELINK>
- **Licencia/uso:** Los datos del MGN (Marco Geoestadístico Nacional) son de dominio público bajo la política de datos abiertos del DANE.

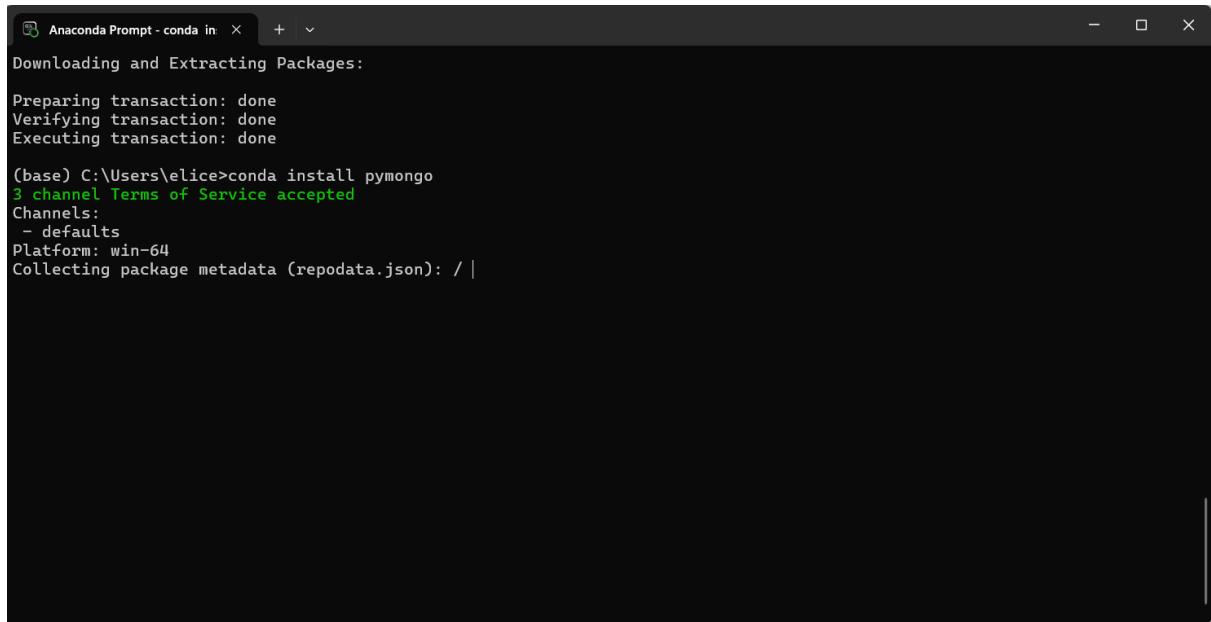
4. Pasos previos

Instalar geopandas



```
Anaconda Prompt - conda in ... + - x
(base) C:\Users\elice>conda install geopandas
Do you accept the Terms of Service (ToS) for https://repo.anaconda.com/pkgs/r? [(a)ccept/(r)eject/(v)iew]: yes
Please select one of the available options
Do you accept the Terms of Service (ToS) for https://repo.anaconda.com/pkgs/r?: [(a)ccept/(r)eject/(v)iew]: a
Do you accept the Terms of Service (ToS) for https://repo.anaconda.com/pkgs/msys2?: [(a)ccept/(r)eject/(v)iew]: a
3 channel Terms of Service accepted
Channels:
  defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: \ |
```

Instalar pymongo

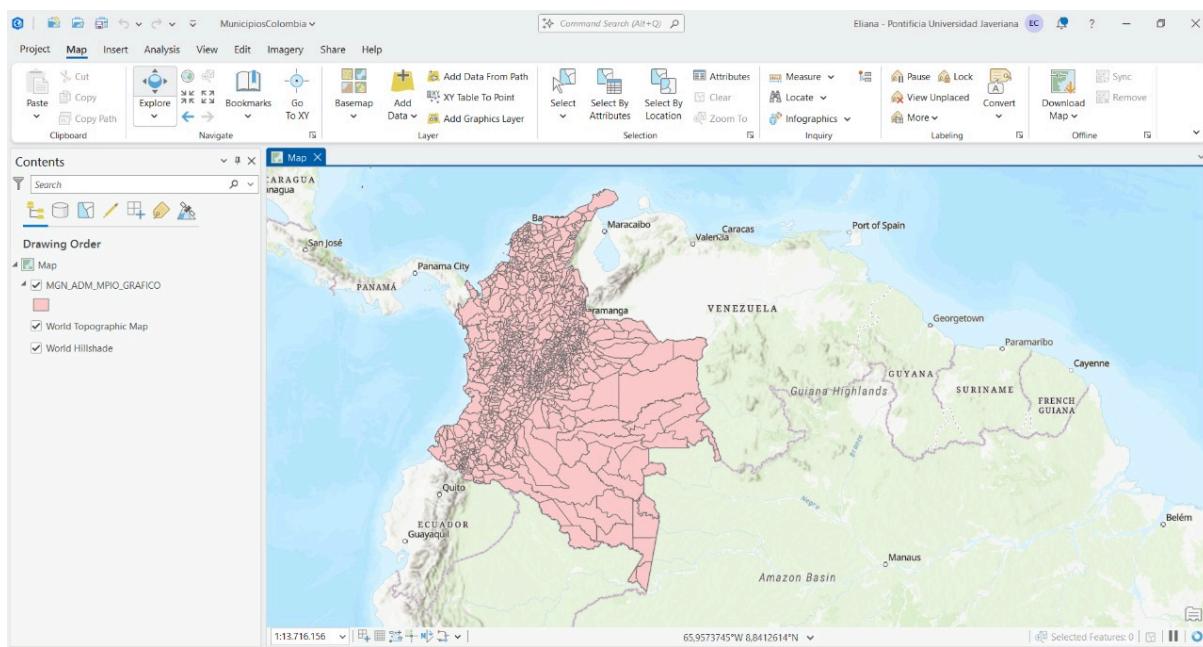


```
Anaconda Prompt - conda in + v
Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(base) C:\Users\elice>conda install pymongo
3 channel Terms of Service accepted
Channels:
- defaults
Platform: win-64
Collecting package metadata (repodata.json): / |
```

5. Pasos Realizados:

1. Descarga de Datos Geográficos del DANE:

- Se descargó un archivo pesado del DANE (Departamento Administrativo Nacional de Estadística) que contiene todos los niveles geográficos a nivel nacional de Colombia, incluyendo departamentos, municipios y otros.
- Se obtuvo la versión completa en formato shapefile (.shp), que representa mapas vectoriales.
- Para acceder al archivo específico: se navegó al nivel administrativo en el geoportal del DANE, seleccionando la capa "MGN_ADMINISTRATIVO_GRAFICO" (Marco Geoestadístico Nacional - Administración Municipal Gráfica).
- Se cargó el shapefile y se verificó que correspondiera al archivo correcto, confirmando su estructura y contenido.



Prueba Validación SHP.

2. Preparación de Datos Adicionales:

- Se utilizó un archivo Excel proporcionado por el profesor, proveniente del DANE, que lista exclusivamente los municipios PDET.
- Este archivo incluye columnas como: código del departamento, nombre del departamento, nombre del municipio y código del municipio.
- Nota: El archivo solo contiene municipios PDET (170 en total), lo que facilita el filtrado posterior.

3. Procesamiento en Entorno Jupyter con Python:

- Se inició un notebook en Jupyter para el análisis y procesamiento.
- Se importaron las librerías necesarias: GeoPandas (para manejar archivos shapefile), Pandas (para manipular datos tabulares) y otras librerías estándar como JSON para conversiones.
- Se definieron constantes y rutas de archivos, incluyendo:
 - La ruta al shapefile descargado (.shp).
 - La ruta al archivo Excel con los municipios PDET.
 - Detalles de conexión a la base de datos MongoDB (por ejemplo, URI de conexión).
- Se creó una colección en MongoDB llamada "municipios" para almacenar los datos procesados.

4. Exploración y Verificación de Datos:

- Se inspeccionaron las columnas del shapefile: se cargó el archivo con GeoPandas y se imprimieron los nombres de las columnas para confirmar su estructura (resultado: Index(['dpto_ccdgo', 'mpio_ccdgo', 'mpio_cdpmp', 'dpto_cnmbr', 'mpio_cnmbr', 'mpio_crslc', 'mpio_tipo', 'mpio_narea', 'mpio_nano', 'shape_Leng', 'shape_Area', 'geometry'], dtype='object')).
- Con Pandas, se cargó el archivo Excel y se extrajo la lista de códigos de municipios PDET.

- Se imprimieron y verificaron estos códigos para asegurar que tuvieran el mismo formato (por ejemplo, códigos válidos de 5 dígitos, consistentes con la codificación del DANE).

5. Carga y Filtrado de Datos Geográficos:

- Se cargó el shapefile completo utilizando GeoPandas, lo que permitió manejar las geometrías espaciales sin necesidad de convertir directamente a CSV (opción descartada por pérdida potencial de información geográfica).
- Se filtraron las geometrías solo para los municipios PDET, cruzando los códigos del shapefile con la lista del Excel.
- Los datos filtrados se transformaron a formato JSON, compatible con MongoDB, preservando las geometrías (usando métodos como `to_json()` de GeoPandas).

6. Carga en la Base de Datos MongoDB:

- Se estableció la conexión a la base de datos MongoDB.
- Antes de insertar, se eliminó todo el contenido existente en la colección "municipios" para evitar duplicados en cargas posteriores (operación de limpieza).
- Se insertaron los datos JSON filtrados en la colección.
- Se verificó la carga exitosa: se realizaron consultas en MongoDB (por ejemplo, conteo de documentos) para confirmar que se insertaron exactamente 170 municipios y que los datos (incluyendo geometrías) se almacenaron correctamente.

```

_id: ObjectId('690782a6d2bc313c49f82e9a')
codigo_dane_municipio: "05031"
nombre: "AMALFI"
departamento: "ANTIOQUIA"
geometria: Object

_id: ObjectId('690782a6d2bc313c49f82e9b')
codigo_dane_municipio: "05040"
nombre: "ANORT"
departamento: "ANTIOQUIA"
geometria: Object

_id: ObjectId('690782a6d2bc313c49f82e9c')
codigo_dane_municipio: "05045"
nombre: "APARTADÓ"
departamento: "ANTIOQUIA"
geometria: Object

_id: ObjectId('690782a6d2bc313c49f82e9d')
codigo_dane_municipio: "05107"
nombre: "BRICEÑO"
departamento: "ANTIOQUIA"
geometria: Object

```

Prueba Validación de carga de datos en Mongo.

7. Creación del índice geoespacial para el campo geometria en MongoDB:

- Se creó el índice geoespacial para el campo `geometria` con el comando `"collection.create_index([("geometria", pymongo.GEOSPHERE)])"` en el JupyterNotebook.

The screenshot shows the MongoDB Compass interface. The top bar indicates the connection is to 'orion.javeriana.edu.co:27017/is394508_db.municipios'. The left sidebar shows 'Connections' with one entry: 'orion.javeriana.edu.co:27017'. The main area is focused on the 'municipios' collection under the 'is394508_db' database. The 'Indexes' tab is selected, showing two entries:

Name & Definition	Type	Size	Usage	Properties	Status
id	REGULAR	20.5 kB	Usage data unavailable	UNIQUE	READY
geometria_2dsphere	GEOSPATIAL	65.5 kB	Usage data unavailable		READY

Prueba Validación de creación de indice en Mongo.

6. Estructura de Datos (post-ETL)

6.1 Diccionario mínimo

Campo	Tipo	Descripción	Ejemplo
codigo_dane_municipio	string	Código DANE municipio (5 dígitos)	"11001"
nombre	string	Nombre municipio	"Bogotá"
departamento	string	Nombre departamento	"Bogotá"
geometria	Object	Polygon/MultiPolygon en EPSG:4326	{...}

6.2 Pasos con código y verificación de éxito:

Conexión a Base de datos MONGO

```
In [12]: try:
    # Intenta conectarse al cliente de MongoDB
    client = MongoClient(CADENA_CONEXION_MONGO)

    # Selecciona la base de datos
    db = client[NOMBRE_BASE_DATOS]

    # Selecciona la colección
    collection = db[NOMBRE_COLECCION]

    # Ping al servidor para confirmar la conexión
    client.admin.command('ping')

    print(f"Conexión exitosa a MongoDB! Listo para trabajar con la colección '{NOMBRE_COLECCION}'.")

except Exception as e:
    print(f"Error conectando a MongoDB: {e}")
    # Si hay un error, el script se detendrá
    raise

¡Conexión exitosa a MongoDB! Listo para trabajar con la colección 'municipios'.
```

Carga y preparación Lista de PDET

```
In [13]: print(f"Cargando lista de PDET desde: {RUTA_EXCEL_PDET}")

# Carga el archivo Excel en un DataFrame de pandas
try:
    df_pdet_excel = pd.read_excel(RUTA_EXCEL_PDET, dtype={COLUMNA_CODIGOS_PDET: str})

    # Convierte la columna de códigos a una lista de Python
    # .dropna() elimina valores nulos y .unique() asegura que no haya duplicados
    # Convertimos la columna a string y luego rellenamos con ceros a la izquierda hasta tener 5 dígitos
    lista_codigos_pdet = df_pdet_excel[COLUMNA_CODIGOS_PDET].astype(str).str.zfill(5).dropna().unique().tolist()

    print(f"Lista de PDET cargada. Se encontraron {len(lista_codigos_pdet)} códigos de municipios PDET.")
    print("Primeros 5 códigos:", lista_codigos_pdet[:5])

except FileNotFoundError:
    print(f"¡ERROR! No se encontró el archivo Excel en: {RUTA_EXCEL_PDET}")
except KeyError:
    print(f"¡ERROR! No se encontró la columna '{COLUMNA_CODIGOS_PDET}' en el Excel. Revisa la celda del Paso")
except Exception as e:
    print(f"Error inesperado leyendo el Excel: {e}")

Cargando lista de PDET desde: D:\Javeriana\Bases de datos\proyecto\MunicipiosPDET\MunicipiosPDET.xlsx
Lista de PDET cargada. Se encontraron 170 códigos de municipios PDET.
Primeros 5 códigos: ['19050', '19075', '19110', '19130', '19137']
```

```

Cargando Shapefile de municipios desde: D:\Javeriana\Bases de datos\ proyecto\ Municipios\ADMINISTRATIVO\MGN_AI
M_MPIO_GRAFICO.shp

--- NOMBRES DE LAS COLUMNAS ---
Index(['dpto_ccdgo', 'mpio_ccdgo', 'mpio_cdmp', 'dpto_cnmbr', 'mpio_cnmbr',
       'mpio_crslc', 'mpio_tipo', 'mpio_narea', 'mpio_nano', 'shape_Leng',
       'shape_Area', 'geometry'],
      dtype='object')
Shapefile cargado. Total de municipios en Colombia: 1121
Filtro aplicado. Total de municipios PDET encontrados: 170

Verificación de municipios PDET encontrados (primeros 5):
    mpio_cdmp  mpio_cnmbr  dpto_cnmbr
5      05031     AMALFI  ANTIOQUIA
9      05040     ANORÍ   ANTIOQUIA
12     05045  APARTADÓ  ANTIOQUIA
22     05107   BRICEÑO  ANTIOQUIA
24     05120   CÁCERES  ANTIOQUIA

```

Transformación a Json

```

In [15]: print("Iniciando transformación a formato JSON...")

documentos_para_insertar = []

# Iteramos sobre cada fila del GeoDataFrame filtrado
for index, municipio in gdf_municipios_pdet.iterrows():

    # Crea un diccionario (JSON) con la estructura definida en La Entrega 1
    documento_municipio = {
        "codigo_dane_municipio": municipio[COLUMNA_CODIGO_SHP],
        "nombre": municipio[COLUMNA_NOMBRE_SHP],
        "departamento": municipio[COLUMNA_DEPTO_SHP],

        # Esta función convierte la geometría de geopandas al formato GeoJSON estándar
        # que MongoDB entiende perfectamente.
        "geometria": municipio["geometry"].__geo_interface__
    }

    # Agrega el documento a nuestra lista
    documentos_para_insertar.append(documento_municipio)

print(f"Transformación completa. Se prepararon {len(documentos_para_insertar)} documentos.")
print("\nEjemplo del primer documento JSON preparado:")
print(json.dumps(documentos_para_insertar[0], indent=2, ensure_ascii=False))

```

Iniciando transformación a formato JSON...
 Transformación completa. Se prepararon 170 documentos.

Ejemplo del primer documento JSON preparado:
{
 "codigo_dane_municipio": "05031",
 "nombre": "AMALFI",
 "departamento": "ANTIOQUIA",

Verificación de Integridad (QA/QC y Resultados)

```

# Coordenadas de prueba dentro de un municipio PDET
test_point = {
    "type": "Point",
    # [Longitud, Latitud] en GeoJSON
    "coordinates": [-74.764, 2.138] # Ejemplo: San Vicente del Caguán
}

result = municipios_col.find_one({
    "pdet": True,
    "geom": { "$geoIntersects": { "$geometry": test_point } }
}, {"cod_mpio": 1, "nom_mpio": 1})

print("Resultado de consulta espacial (geocercado):")

```

```
print(result)

# Resultado esperado: {'_id': ObjectId('...'), 'cod_mpio': '18753', 'nom_mpio': 'SAN VICENTE DEL CAGUÁN'}
```

Resultados de ejecución en Jupyter:

```
Iniciando carga a MongoDB...
Limpiendo colección 'municipios' (método: drop)...
Colección antigua eliminada (si existía).
Insertando nuevos documentos...

--- ¡CARGA COMPLETA! ---
Se insertaron 170 municipios PDET en la colección.
```

```
Verificación: La colección 'municipios' ahora tiene 170 documentos.

Documento de ejemplo recuperado de MongoDB:
{
    "codigo_dane_municipio": "05031",
    "nombre": "AMALFI",
    "departamento": "ANTIOQUIA"
}

Conexión a MongoDB cerrada.
```

7. Conclusión de la Integración de Datos PDET

En esta fase del proyecto y la segunda entrega titulada "Integración del Dataset de Límites Municipales PDET", logramos terminar con éxito la obtención, el procesamiento y la incorporación de los límites administrativos de los municipios en nuestra base de datos NoSQL. Filtramos y cargamos los 170 municipios PDET designados, tomando como fuente oficial el Marco Geoestadístico Nacional (MGN) del DANE.

Usamos MongoDB como motor principal y creamos un índice espacial 2dsphere, lo que nos permite hacer consultas geográficas rápidas, escalables y sin complicaciones.

Este avance nos da la base geoespacial sólida que necesitábamos para las próximas etapas: ahora podremos filtrar miles de millones de huellas de edificios de manera eficiente, solo en los territorios priorizados por la UPME.