

Segunda Entrega
PDET Municipality boundaries dataset integration



**Pontificia Universidad
JAVERIANA**
Colombia

Juan David Castillo Laverde
Juan Pablo Dávila Martínez
Eugenia Victoria Dayoub Barito
Luis Fernando Lee Rodríguez

Pontificia Universidad Javeriana
Facultad de Ingeniería
Administración de Bases de Datos
Bogotá D.C.
Noviembre 2025

Introducción.....	3
Adquisición y verificación de datos.....	3
Integridad y formato de datos.....	5
Integración espacial NoSQL.....	10
Verificación de la Integración.....	11
Conclusión.....	12

Introducción

En esta entrega del proyecto se trabajó en la obtención, preparación y depuración del conjunto de datos necesario para identificar los municipios PDET dentro del territorio nacional. Para ello fue necesario buscar fuentes de geometría municipal, verificar su disponibilidad, seleccionar una alternativa utilizable, normalizar la información y filtrar únicamente los registros correspondientes a municipios PDET.

Con este proceso se obtuvo finalmente un subconjunto geográfico limpio y validado, exportado en formato GeoJSON, el cual posteriormente fue cargado e indexado en la base de datos NoSQL para habilitar su uso en las siguientes etapas del proyecto.

Adquisición y verificación de datos

La primera opción fue intentar descargar los datos del Marco Geoestadístico Nacional del DANE (MGN), ya que es la fuente oficial de geometrías de Colombia. Sin embargo, durante el proceso la página del DANE no permitió descargar los archivos: los enlaces no cargaban, la página fallaba y mostraba errores a la hora de descargar el archivo. Por esta razón, aunque esa era la fuente ideal, no fue posible trabajar con ella.

Como alternativa, se decidió usar GeoBoundaries (ADM2 – Colombia), que contiene los polígonos municipales, está en formato abierto y además ya viene en CRS EPSG:4326, lo cual facilita la exportación a GeoJSON.

Individual Country Files

The geoBoundaries open license ([CC-BY 4.0](#)) country dataset seeks to represent every nation "as they would represent themselves", with no special identification of disputed areas. If you would prefer data that explicitly includes disputed areas, please see the global composite dataset. [Attribution](#) is required for all uses of this dataset.

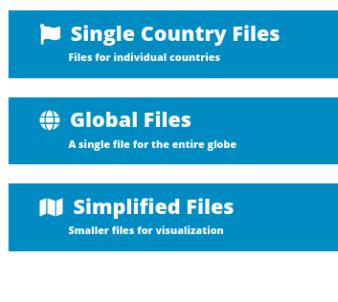
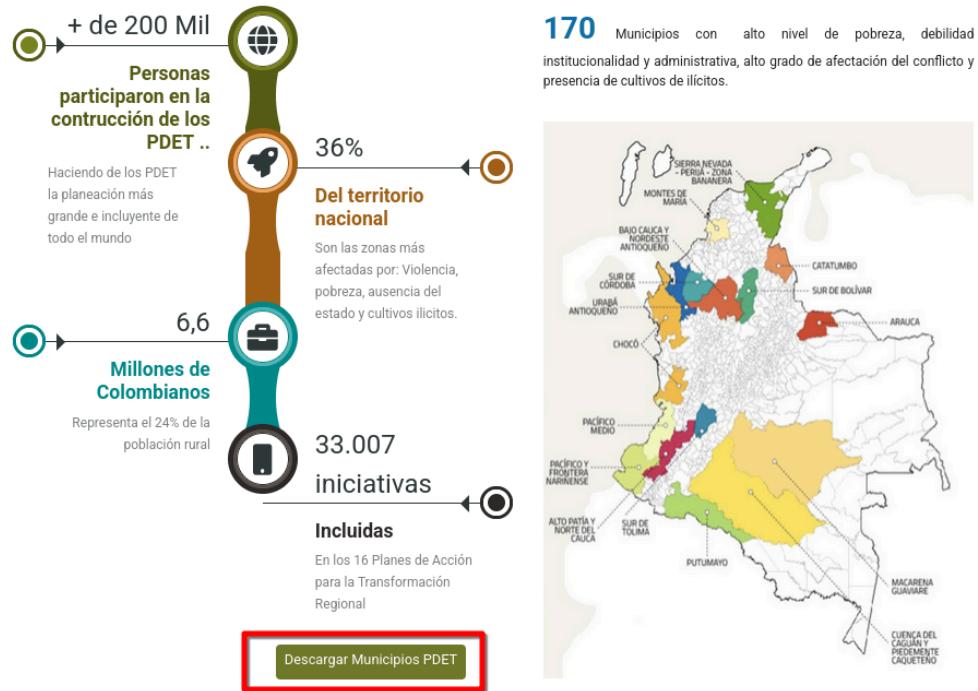


Figura 1: Descarga datos GeoBoundaries

Además de la capa geográfica, se cargó también un archivo Excel con la lista oficial de municipios PDET. Este archivo fue el que permitió identificar cuáles municipios del dataset completo debían mantenerse. Los dos archivos fueron importados en QGIS, el GeoBoundaries como capa vectorial y el Excel como tabla. Al revisarlos se encontró que ambos contenían campos con nombres de municipios, pero no escritos exactamente igual, lo cual confirmó que era necesario normalizar el texto para poder unirlos correctamente.

PDET EN CIFRAS



geoBoundaries MunicipiosPDET.xlsx

MunicipiosPDET

File Home Insert Share Page Layout Formulas Data Review View Help Draw

14

Subregión PDET	Código DANE Departamento	Departamento	Código DANE Municipio	Municipio
2 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19050 ARGELIA		
3 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19075 BALBOA		
4 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19110 BUENOS AIRES		
5 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19130 CAJIBIÓN		
6 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19137 CALDONO		
7 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19142 CALOTO		
8 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19212 CORINTO		
9 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19256 EL TAMBO		
10 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19364 JAMBALÓ		
11 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19450 MERCADERES		
12 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19455 MIRANDA		
13 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19473 MORALES		
14 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19532 PATÍA		
15 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19546 PIENDAMÓ		
16 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19698 SANTANDER DE QUILOCHAO		
17 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19780 SUÁREZ		
18 ALTO PATÍA Y NORTE DEL CAUCA	19 CAUCA	19821 TORIBIO		
19 ALTO PATÍA Y NORTE DEL CAUCA	52 NARIÑO	52233 CUMBITARA		
20 ALTO PATÍA Y NORTE DEL CAUCA	52 NARIÑO	52256 EL ROSARIO		
21 ALTO PATÍA Y NORTE DEL CAUCA	52 NARIÑO	52405 LEIVA		
22 ALTO PATÍA Y NORTE DEL CAUCA	52 NARIÑO	52418 LOS ANDES		
23 ALTO PATÍA Y NORTE DEL CAUCA	52 NARIÑO	52540 POLICARPA		
24 ALTO PATÍA Y NORTE DEL CAUCA	76 VALLE DEL CAUCA	76275 FLORIDA		
25 ALTO PATÍA Y NORTE DEL CAUCA	76 VALLE DEL CAUCA	76565 PRADERA		
26 ARAUCA	81 ARAUCA	81065 ARAUQUITA		
27 ARAUCA	81 ARAUCA	81300 FORTUL		
28 ARAUCA	81 ARAUCA	81746 SARAVINA		

Figura 2: Descarga datos MunicipiosPDET

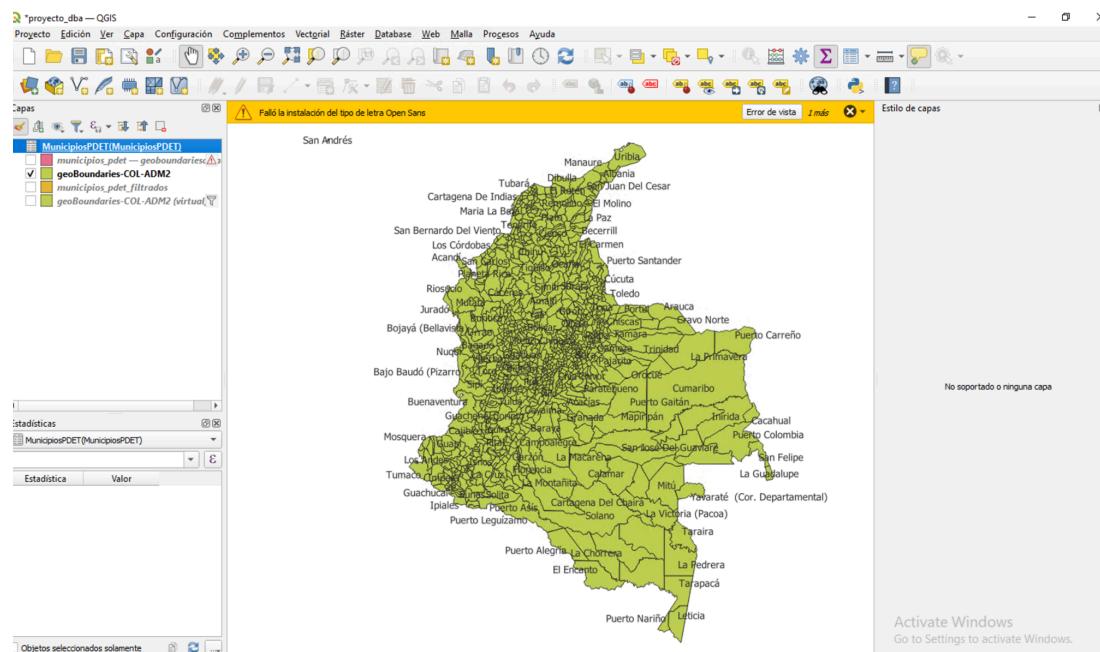
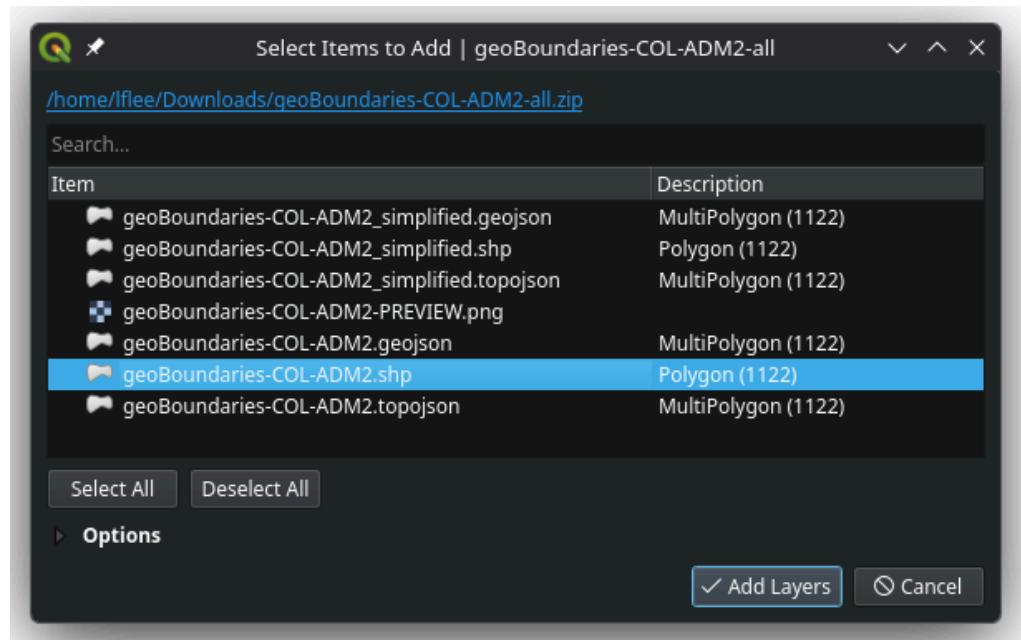


Figura 3: Carga de datos en QGIS

Integridad y formato de datos

Con ambos archivos cargados en QGIS se procedió a la normalización. Se creó un campo nuevo en la tabla municipal y otro en la tabla PDET aplicando expresiones para eliminar tildes, pasar todo a mayúscula y limpiar caracteres especiales, de manera que los nombres quedarán homologados y comparables entre sí.

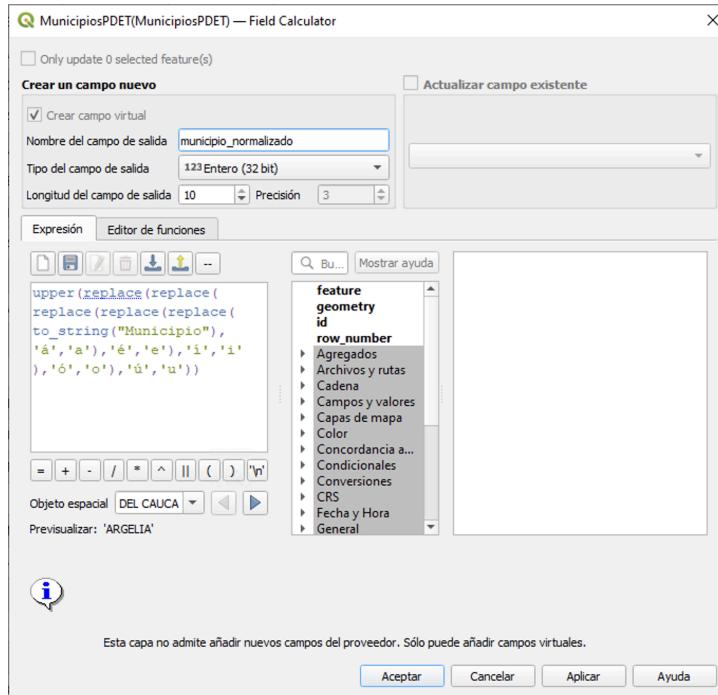


Figura 4: Normalización de datos en tabla MunicipiosPDET

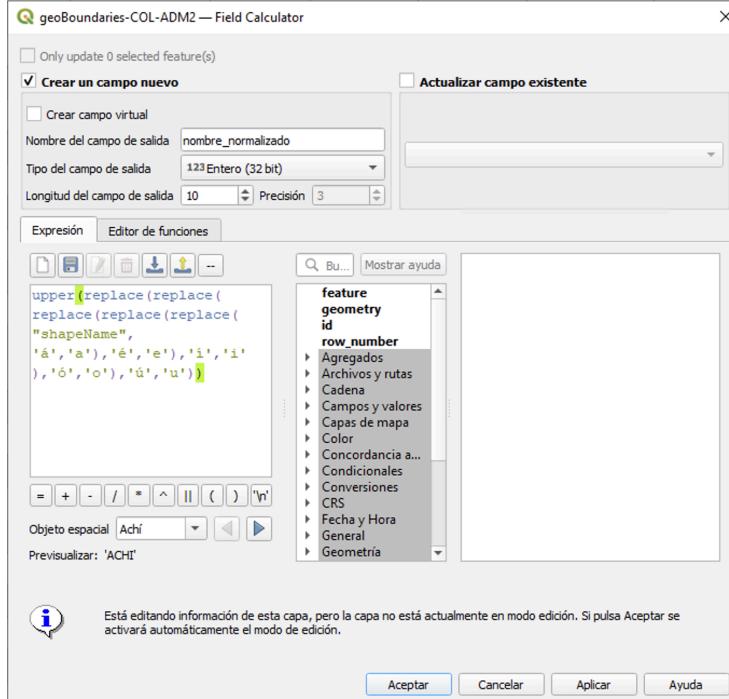


Figura 5: Normalización de datos en tabla geoBoundaries

Luego, en las propiedades de la capa municipal, se configuró una unión (JOIN) utilizando estos campos normalizados como clave. Esto permitió identificar directamente qué municipios dentro del dataset nacional coincidían con la lista oficial PDET.

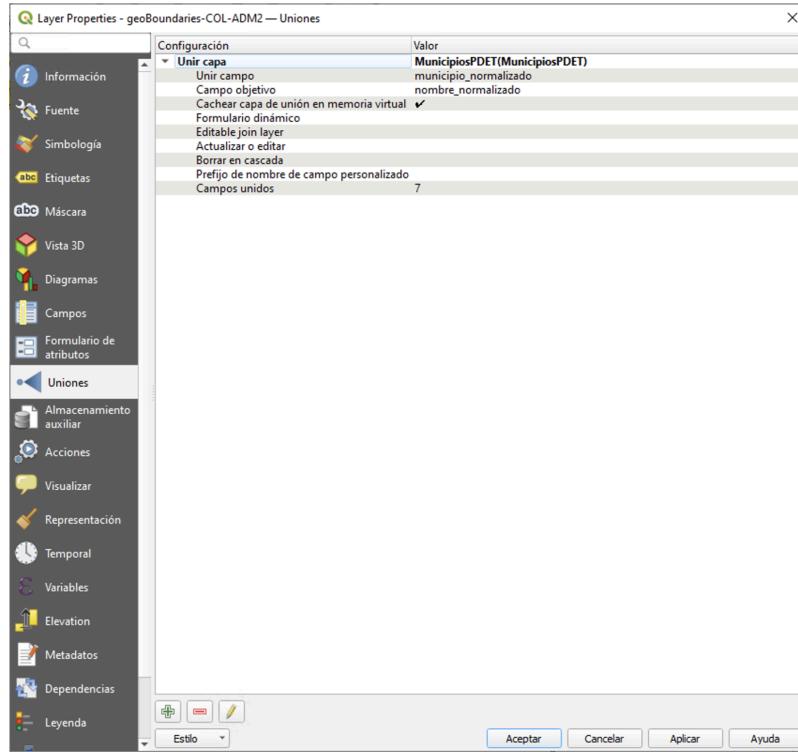


Figura 6: Unión de tablas con datos normalizados

Una vez aplicada la unión, se filtraron los registros en QGIS seleccionando únicamente aquellos donde el campo proveniente de la tabla PDET no fuera nulo. De esta forma se obtuvo únicamente el subconjunto de municipios PDET, que fue aproximadamente de 101 municipios frente a los 1122 datos cargados.

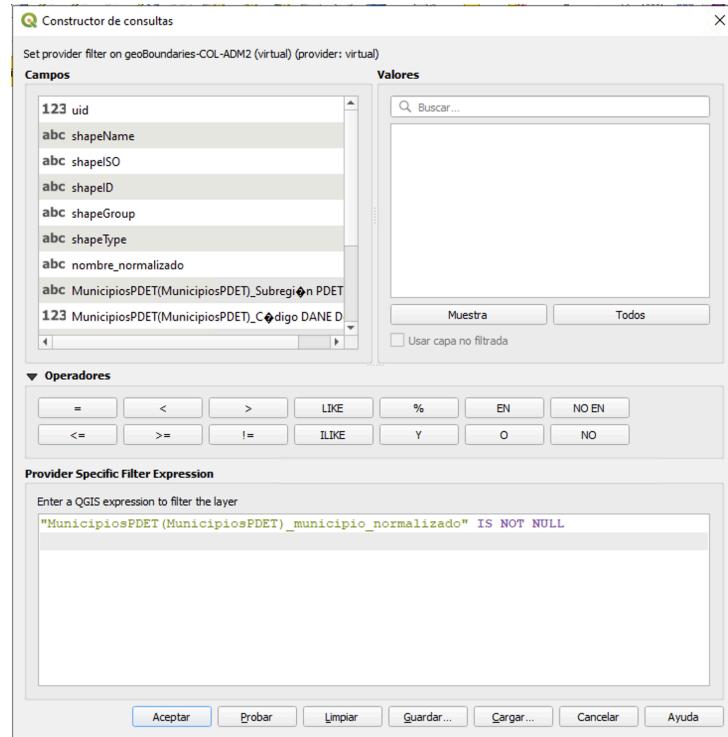


Figura 7: Filtración de datos

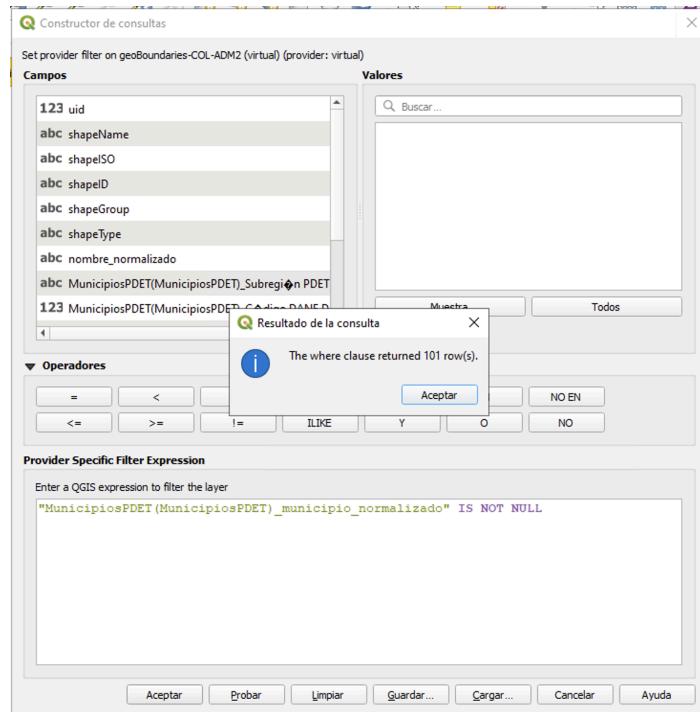


Figura 8: Resultado después de filtración de datos

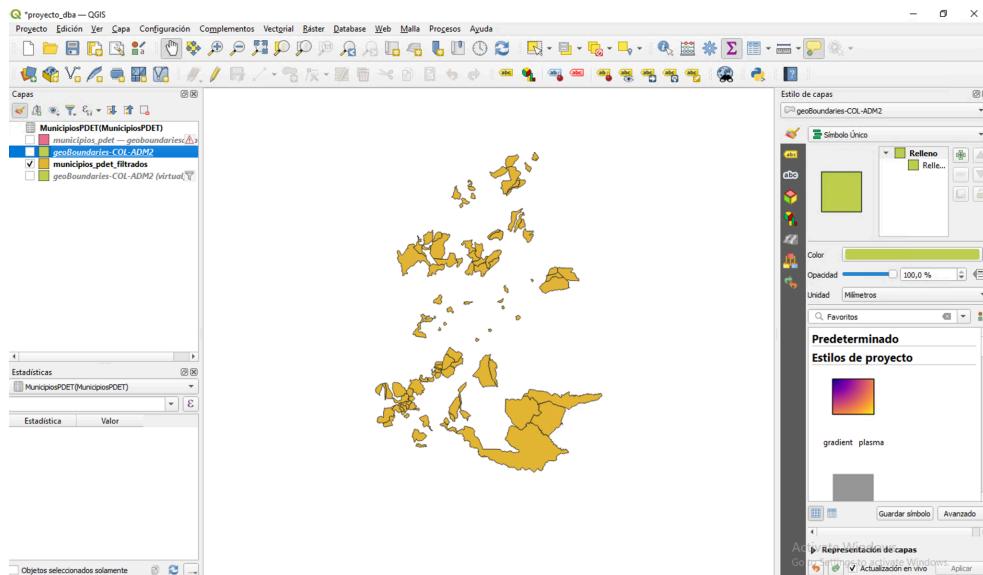


Figura 9: Resultado 2 después de filtración de datos

Con el subconjunto ya verificado, se exportó exclusivamente esa selección a formato GeoJSON, asegurando que el CRS continuara siendo EPSG:4326 e incluyendo todos los campos para tener todos los datos necesarios.

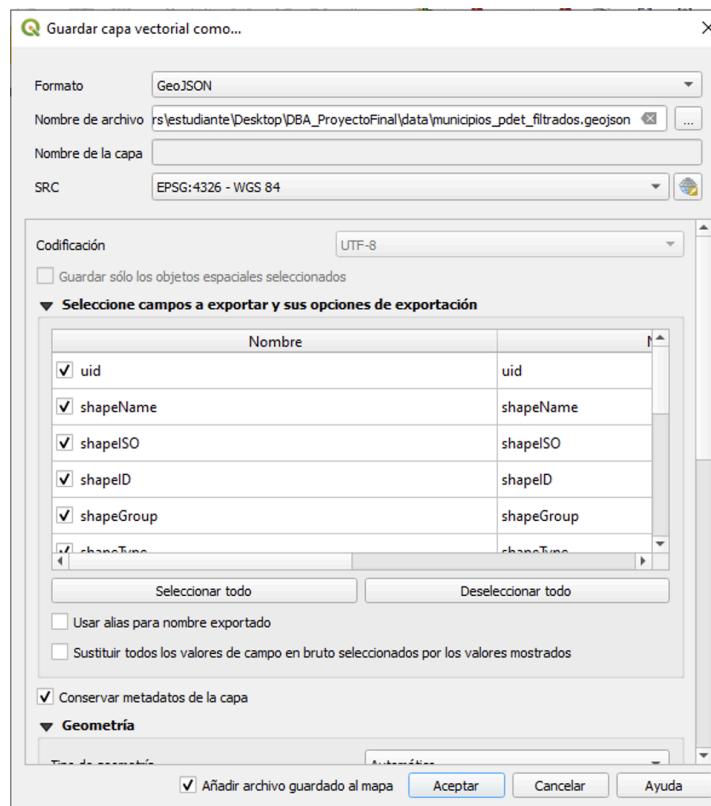


Figura 10: Exportación de datos finales en GeoJSON

El archivo final se guardó con el nombre municipios_pdet.geojson.

Integración espacial NoSQL

Una vez obtenido el archivo municipios_pdet.geojson, el siguiente paso fue cargarlo en la base de datos MongoDB. Este proceso se automatizó mediante un script de Python utilizando la librería pymongo.

El entorno de la base de datos y de Python se gestionó con Docker, asegurando la reproducibilidad del sistema, tal como se estableció en nuestro plan de implementación .

1. Script de Carga y Transformación (ETL)

El script de Python (cargar_municipios.py) se diseñó para realizar el proceso de carga y transformación. El script realiza las siguientes acciones clave:

Conexión y Limpieza: Se conecta a la base de datos proyecto_upme en Docker y ejecuta delete_many({}) sobre la colección mgn_municipios_pdet para garantizar que cada ejecución sea una carga limpia (idempotencia).

Lectura de Datos: Lee el archivo municipios_pdet_filtrados.geojson.

Transformación y Mapeo (ETL): Itera sobre cada "feature" del GeoJSON y "mapea" los campos de las propiedades (con nombres largos y caracteres especiales) a nuestro esquema limpio definido en la Entrega 1 .

Carga de Datos: Utiliza insert_many() para cargar la lista de 101 documentos de municipios de forma eficiente (bulk).

Creación de Índice: Ejecuta create_index([("geometry", GEOSPHERE)]) para crear el índice 2dsphere sobre el campo de geometría. Este es el paso técnico más crítico, ya que habilita las consultas geoespaciales de alto rendimiento que usaremos en las siguientes entregas.

```

FOLDERS DBA PROYECTO          docker-compose.yml  cargar_municipios.py  municipios_pdet_filtros.py
├── DBA_ProyectoFinal
│   ├── data
│   │   ├── cargar_municipios.py
│   │   └── municipios_pdet_filtros.geojson
│   ├── docker-compose.yml
│   └── Primera Entrega DBA.pdf

```

```

# Script para cargar los 101 municipios de Colombia en MongoDB
# Utiliza la colección 'municipios_pdet_filtros' en la base de datos 'DBA_NAME'
# Requiere conexión a MongoDB con la URL 'MONGO_URI'

from pymongo import MongoClient
from bson.json_util import loads
import json
import os

# Configuración de MongoDB
MONGO_URI = "mongodb://localhost:27017"
DB_NAME = "DBA_NAME"
COLLECTION_NAME = "municipios_pdet_filtros"

# Verificar conexión a MongoDB
try:
    client = MongoClient(MONGO_URI)
    db = client[DB_NAME]
    collection = db[COLLECTION_NAME]
    print(f"Conectado a MongoDB (Base de datos: {DB_NAME}, Colección: {COLLECTION_NAME})")
except Exception as e:
    print(f"ERROR: No se pudo conectar a MongoDB...")
    print(f"Detalle: {e}")
    exit()

# Limpiar colección existente
collection.delete_many({})
print("Colección limpia (delete_many).")

if not os.path.exists(GEOJSON_FILE):
    print(f"ERROR: No se encontró el archivo '{GEOJSON_FILE}'")
    client.close()
    exit()

with open(GEOJSON_FILE, 'r', encoding='utf-8') as f:
    geojson_data = f.read()
features = geojson.loads(geojson_data.get('features', []))
if not features:
    print("ERROR: El archivo GeoJSON no tiene 'features' o está vacío.")
    client.close()
    exit()

print(f"Leídos {len(features)} municipios del archivo GeoJSON...")

documentos_para_insertar = []
for feature in features:
    if feature.get('geometry') and feature.get('properties'):
        props = feature['properties']
        documento = {
            'codigo_municipio': props.get('MunicipiosPOET(MunicipioPOET)', 'Tipo DANE Municipio'),
            'nombre_municipio': props.get('MunicipiosPOET(MunicipioPOET)', 'Municipio'),
            'departamento': props.get('MunicipiosPOET(MunicipiosPOET)_Departamento'),
            'geometry': feature['geometry']
        }
        documentos_para_insertar.append(documento)

if not documentos_para_insertar:
    print("No se pudieron cargar documentos. Revisa el GeolJSON.")
    client.close()
    exit()

try:
    collection.insert_many(documentos_para_insertar)
    print(f"Exitoso la inserción de {len(documentos_para_insertar)} municipios en la colección.")
except Exception as e:
    print(f"ERROR: Falló la inserción de datos. Detalle: {e}")
    client.close()
    exit()

try:
    collection.create_index([('geometry', '2dsphere')])
    print("Índice 'geometry_2dsphere' creado exitosamente!")
except Exception as e:
    print(f"ERROR: Falló la creación del índice. Detalle: {e}")
    client.close()
    exit()

count = collection.count_documents({})
print(f"Verificación: La colección '{COLLECTION_NAME}' ahora tiene {count} documentos.")

print("Último documento de la BD:")
print(collection.find_one())
client.close()

```

Figura 11: Script de carga y creación de índice (cargar_municipios.py)

Verificación de la Integración

Tras ejecutar el script, la integración se verificó directamente en MongoDB Compass. La verificación confirmó dos puntos:

- Datos Cargados:** Los 101 municipios se cargaron exitosamente. La estructura de los documentos coincide con nuestro esquema limpio, conteniendo solo los campos `codigo_municipio`, `nombre_municipio`, `departamento` y `geometry`.
- Índice Creado:** El índice `geometry_2dsphere` fue creado exitosamente sobre la colección.

The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar titled 'Compass' with sections for 'My Queries', 'CONNECTIONS (2)', and a search bar. The main area is titled 'localhost:27017 > proyecto_upme > mgn_municipios_pdet'. It shows a table with 101 documents. The first document previewed is:

```

_id: ObjectId('6906743c81e1aaa90994145')
codigo_municipio : 5736
nombre_municipio : "SEGOVIA"
departamento : "ANTIOQUIA"
geometry : Object
  type : "MultiPolygon"
  coordinates : Array (1)

```

Other documents shown have IDs 6906743c81e1aaa90994146 and 6906743c81e1aaa90994147, with similar field structures.

Figura 12: Documentos cargados en la colección mgn_municipios_pdet

The screenshot shows the 'Indexes' tab for the 'mgn_municipios_pdet' collection. It lists two indexes:

Name & Definition	Type	Size	Usage	Properties	Status
id	REGULAR	32.8 KB	5 (since Sat Nov 01 2025)	UNIQUE	READY
geometry_2dsphere	GEOSPATIAL	57.3 KB	0 (since Sat Nov 01 2025)		READY

Figura 13: Índice geometry_2dsphere creado exitosamente

Conclusión

La segunda entrega se completó con éxito. Se logró superar el desafío de la indisponibilidad de la fuente oficial (DANE) mediante el uso de fuentes alternativas (GeoBoundaries y Excel PDET).

Se aplicó un riguroso proceso de limpieza y transformación en QGIS, seguido de un script de carga automatizado en Python. El resultado es una colección en MongoDB (mgn_municipios_pdet) que contiene los polígonos limpios de los 101 municipios PDET,

indexada espacialmente y lista para servir como la capa de referencia geoespacial para la siguiente fase del proyecto: la carga e integración de las huellas de edificios.