Entrega Final
Final Technical Report and Recommendations

Pontificia Universidad
JAVERIANA
Colombia

Juan David Castillo Laverde
Juan Pablo Dávila Martinez
Eugenia Victoria Dayoub Barito
Luis Fernando Lee Rodriguez

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

Contenido

Introducción

El presente informe constituye el documento técnico final del proyecto desarrollado para la Unidad de Planeación Minero Energética (UPME), cuyo objetivo es diseñar y ejecutar un workflow reproducible, escalable y auditable para estimar el número total de edificaciones y el área de sus cubiertas dentro de los municipios PDET, como insumo para evaluar el potencial de energía solar en estas regiones.

El proyecto se desarrolló en cuatro fases previas:

1. Diseño de un modelo de datos en MongoDB, optimizado para manejar geometrías complejas mediante índices espaciales.
2. Integración y depuración del conjunto de municipios PDET.
3. Preparación, carga y auditoría de datos abiertos de edificaciones provenientes de Google Open Buildings y Microsoft Building Footprints.
4. Implementación del pipeline geoespacial reproducible, responsable del filtrado, conteo y cálculo de áreas sobre cada municipio.

Este documento consolida las cuatro entregas, presenta los resultados finales, las visualizaciones, las recomendaciones para UPME y evidencia el cumplimiento de todos los criterios solicitados: documentación del proceso, resultados reproducibles, visualizaciones, completitud, claridad y alineación con los objetivos institucionales.

Objetivos del proyecto

Objetivo General

Desarrollar un workflow reproducible basado en tecnologías NoSQL para estimar el número y área de edificaciones dentro de municipios PDET, como insumo para futuros análisis de potencial solar.

Objetivos Específicos

- Diseñar un modelo de base de datos NoSQL capaz de almacenar geometrías masivas y consultas espaciales.
- Integrar y depurar la capa geográfica oficial de municipios PDET.
- Procesar, auditar y cargar los datasets abiertos de Google y Microsoft.
- Construir un pipeline automatizado para filtrar footprints dentro de cada municipio.
- Calcular el área total útil de cubiertas mediante $geoArea.
- Generar un archivo final consolidado con resultados por municipio.
- Visualizar los resultados mediante mapas estáticos.
- Formular recomendaciones alineadas con los objetivos de UPME.

Metodología

La metodología se estructuró para permitir reproducibilidad completa y separación clara entre procesamiento y visualización. Todas las operaciones se realizaron en MongoDB, mientras que QGIS se utilizó únicamente para visualización.

Diseño del Modelo NoSQL

Para gestionar datos masivos, se definió un conjunto de colecciones especializadas:

- mgn_municipios_pdet: municipios filtrados PDET.
- buildings_google: edificaciones provenientes de Google.
- buildings_microsoft: edificaciones provenientes de Microsoft.
- resultados_municipio: resultados agregados por municipio.

Cada documento geométrico se modeló en formato GeoJSON, y se creó para cada colección un índice espacial 2dsphere, permitiendo ejecutar consultas $geoWithin y $geoIntersects de forma eficiente.

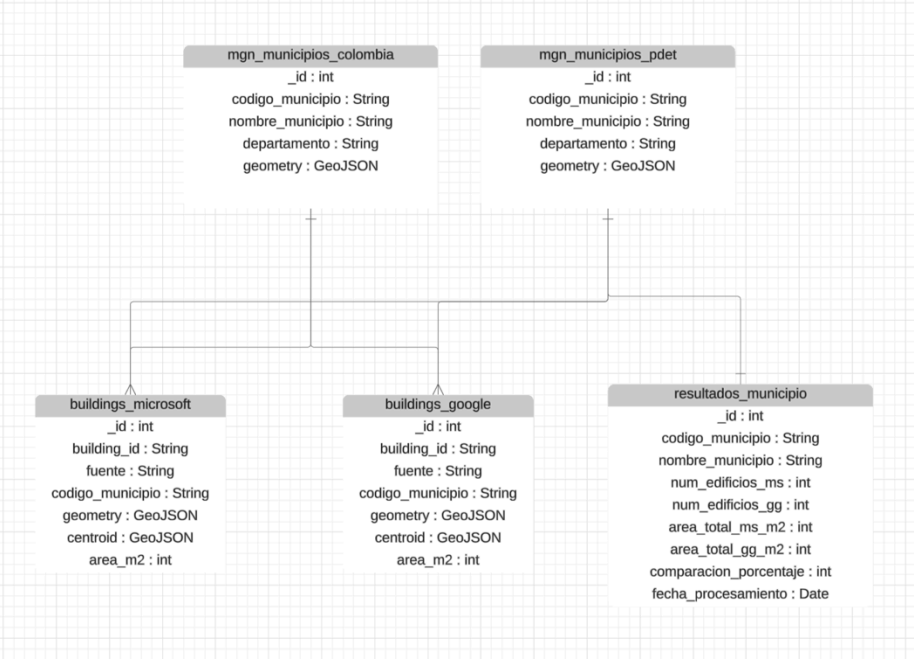


Figura 1: Diagrama modelo de datos

Integración de Municipios PDET

Para garantizar el uso de datos oficiales, se descargó desde el portal del DANE el archivo.

Este archivo contiene:

- Límites oficiales de los municipios
- Codificación DANE
- Geometría precisa y actualizada

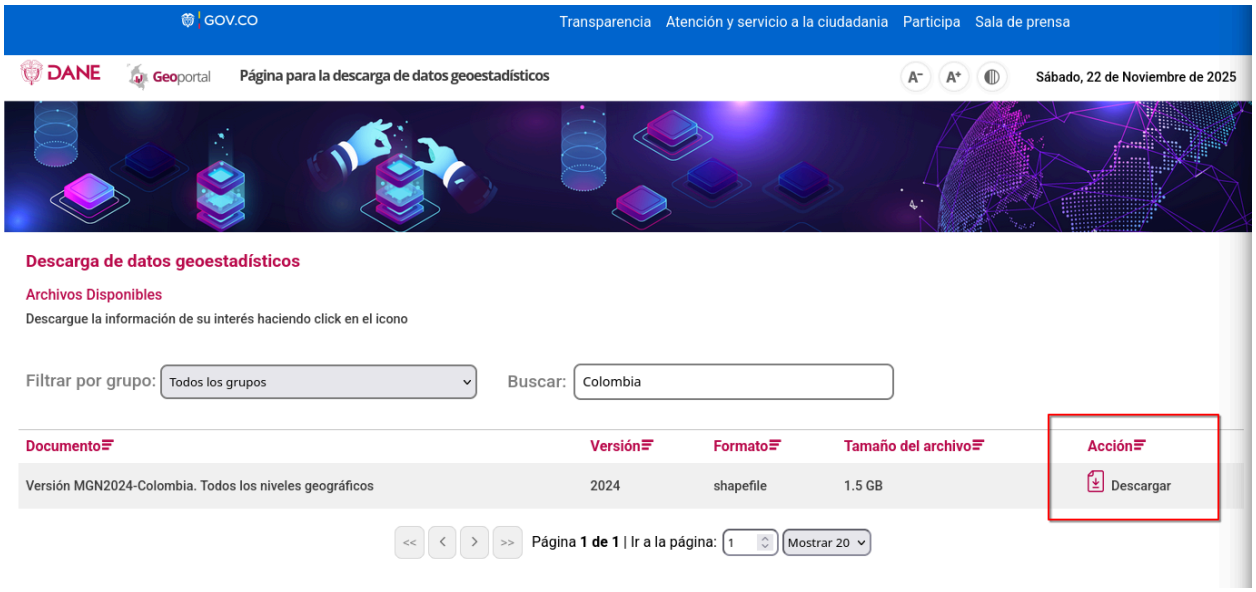


Figura 2: Descarga de datos desde el portal DANE

Proceso de integración:

1. Se verificó la estructura del shapefile.
2. Se normalizaron campos administrativos.
3. Se unió con la tabla oficial de municipios PDET.
4. Se exportó el resultado final: municipios_pdet_2024.geojson
5. Se cargó en MongoDB mediante cargar_municipios.py.
6. Se creó el índice espacial correspondiente.

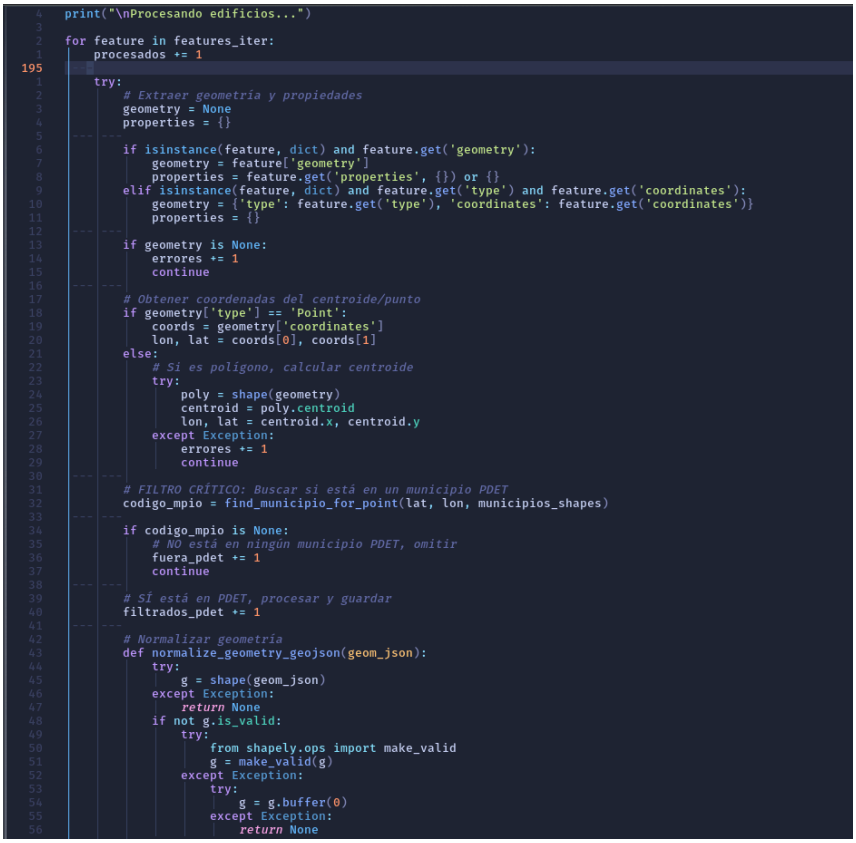
Carga y Auditoría de Footprints

En esta fase se realizó la carga completa de los datasets oficiales de edificaciones, tanto de Google Open Buildings como de Microsoft Building Footprints.

Google Open Buildings

- Se identificaron los tiles del dataset que contienen territorio colombiano.
- Cada tile fue descargado en formato CSV.

- Los archivos completos fueron procesados mediante un script Python que:
  1. Lee cada archivo completo
  2. Estandariza la geometría a formato GeoJSON
  3. Inserta cada documento en la colección buildings_google
  4. Crea el índice 2dsphere necesario para operaciones espaciales
  5. Registra logs de progreso (importación de millones de registros)



```
print("\nProcesando edificios...")

for feature in features_iter:
    procesados += 1

    try:
        # Extraer geometría y propiedades
        geometry = None
        properties = {}

        if isinstance(feature, dict) and feature.get('geometry'):
            geometry = feature['geometry']
            properties = feature.get('properties', {}) or {}
        elif isinstance(feature, dict) and feature.get('type') and feature.get('coordinates'):
            geometry = {'type': feature.get('type'), 'coordinates': feature.get('coordinates')}
            properties = {}

        if geometry is None:
            errores += 1
            continue

        # Obtener coordenadas del centroide/punto
        if geometry['type'] == 'Point':
            coords = geometry['coordinates']
            lon, lat = coords[0], coords[1]
        else:
            # Si es polígono, calcular centroide
            try:
                poly = shape(geometry)
                centroid = poly.centroid
                lon, lat = centroid.x, centroid.y
            except Exception:
                errores += 1
                continue

        # FILTRO CRÍTICO: Buscar si está en un municipio PDET
        codigo_mpio = find_municipio_for_point(lat, lon, municipios_shapes)

        if codigo_mpio is None:
            # NO está en ningún municipio PDET, omitir
            fuera_pdet += 1
            continue

        # SÍ está en PDET, procesar y guardar
        filtrados_pdet += 1

        # Normalizar geometría
        def normalize_geometry_geojson(geom_json):
            try:
                g = shape(geom_json)
            except Exception:
                return None
            if not g.is_valid:
                try:
                    from shapely.ops import make_valid
                    g = make_valid(g)
                except Exception:
                    try:
                        g = g.buffer(0)
                    except Exception:
                        return None
```

Figura 3: Script de procesamiento Google

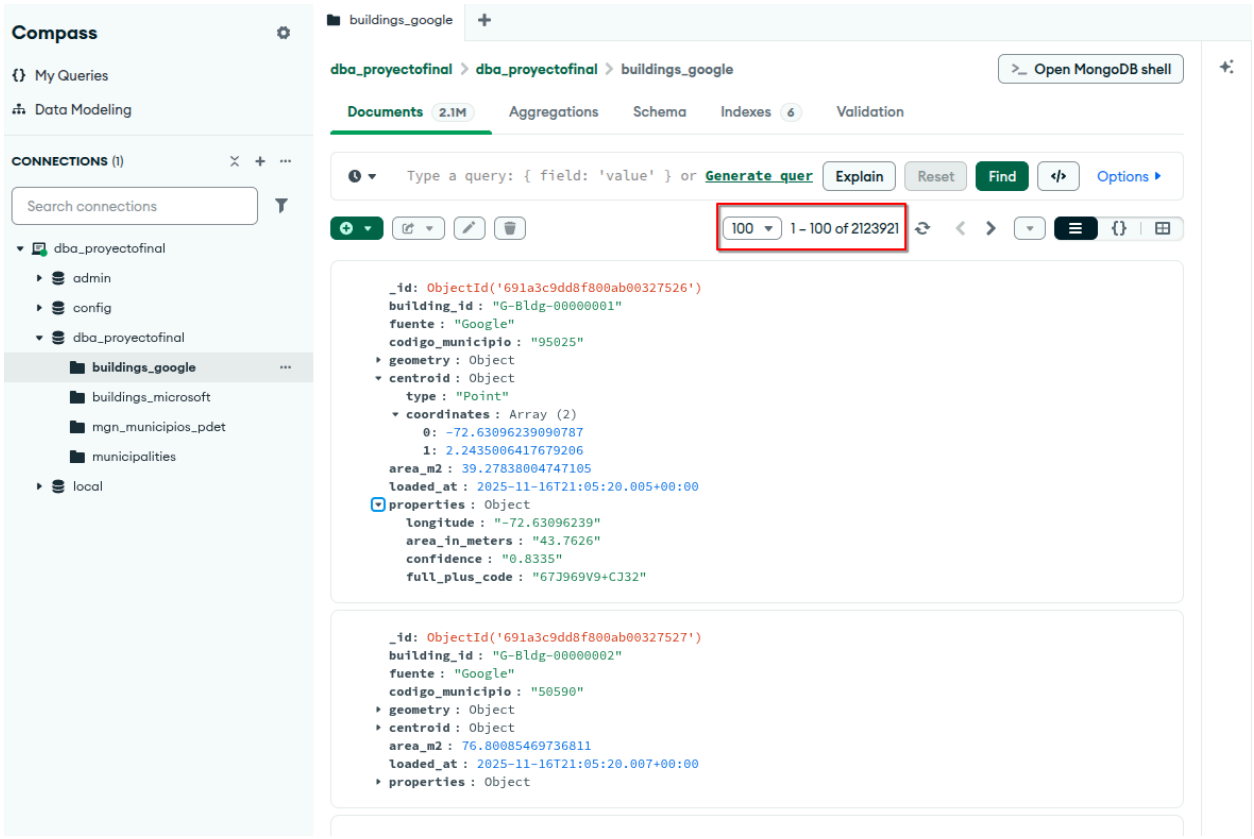
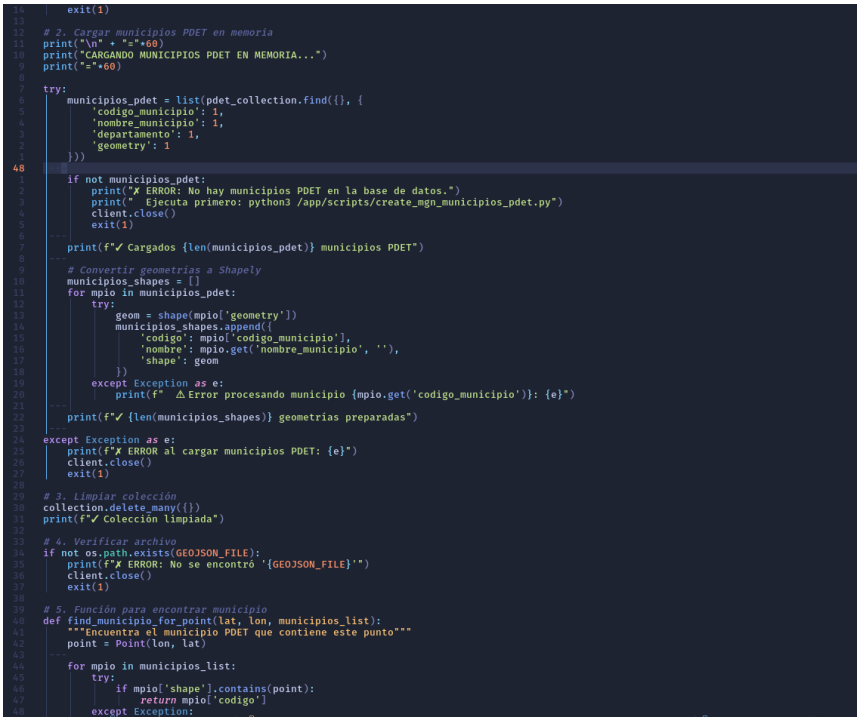


Figura 4: Datos de google cargados en la base de datos

Microsoft Building Footprints

- Se descargó el archivo nacional .geojsonl de Colombia (millones de edificaciones).
- Debido al tamaño del archivo, se utilizó un script de lectura por streaming que:
  1. Procesa el archivo línea por línea
  2. Convierte cada geometría a GeoJSON
  3. Inserta cada documento en la colección buildings_microsoft
  4. Crea el índice 2dsphere
  5. Usa batch inserts optimizados para alta escala



```
    exit(1)

# 2. Cargar municipios PDET en memoria
print("\n" + "="*60)
print("CARGANDO MUNICIPIOS PDET EN MEMORIA...")
print("="*60)

try:
    municipios_pdet = list(pdet_collection.find({}, {
        'codigo_municipio': 1,
        'nombre_municipio': 1,
        'departamento': 1,
        'geometry': 1
    }))

    if not municipios_pdet:
        print("✗ ERROR: No hay municipios PDET en la base de datos.")
        print("   Ejecuta primero: python3 /app/scripts/create_mgn_municipios_pdet.py")
        client.close()
        exit(1)

    print(f"✓ Cargados {len(municipios_pdet)} municipios PDET")

    # Convertir geometrias a Shapely
    municipios_shapes = []
    for mpio in municipios_pdet:
        try:
            geom = shape(mpio['geometry'])
            municipios_shapes.append({
                'codigo': mpio['codigo_municipio'],
                'nombre': mpio.get('nombre_municipio', ''),
                'shape': geom
            })
        except Exception as e:
            print(f"  ⚠ Error procesando municipio {mpio.get('codigo_municipio')}: {e}")

    print(f"✓ {len(municipios_shapes)} geometrias preparadas")

except Exception as e:
    print(f"✗ ERROR al cargar municipios PDET: {e}")
    client.close()
    exit(1)

# 3. Limpiar colección
collection.delete_many({})
print(f"✓ Colección limpiada")

# 4. Verificar archivo
if not os.path.exists(GEOJSON_FILE):
    print(f"✗ ERROR: No se encontró '{GEOJSON_FILE}'")
    client.close()
    exit(1)

# 5. Función para encontrar municipio
def find_municipio_for_point(lat, lon, municipios_list):
    """Encuentra el municipio PDET que contiene este punto"""
    point = Point(lon, lat)

    for mpio in municipios_list:
        try:
            if mpio['shape'].contains(point):
                return mpio['codigo']
        except Exception:
```

Figura 5: Script de procesamiento Microsoft

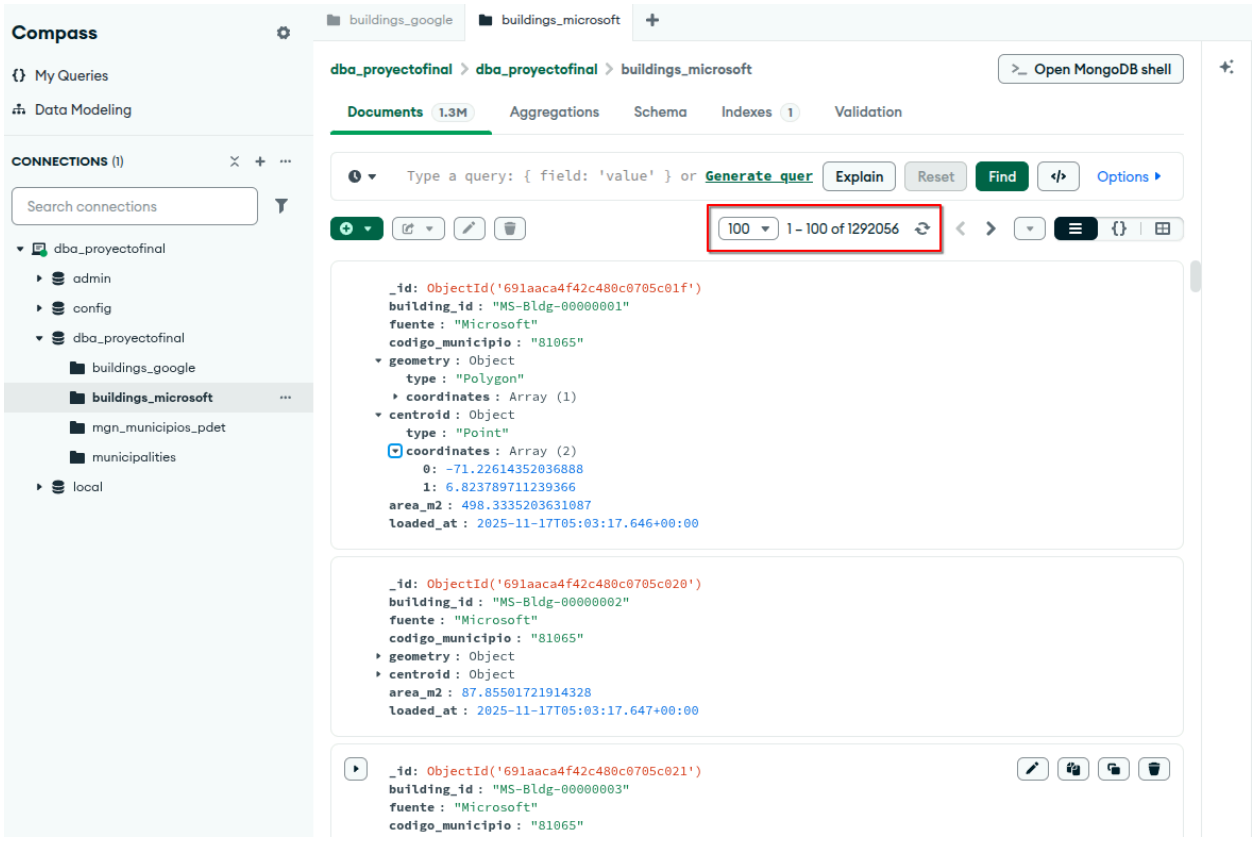
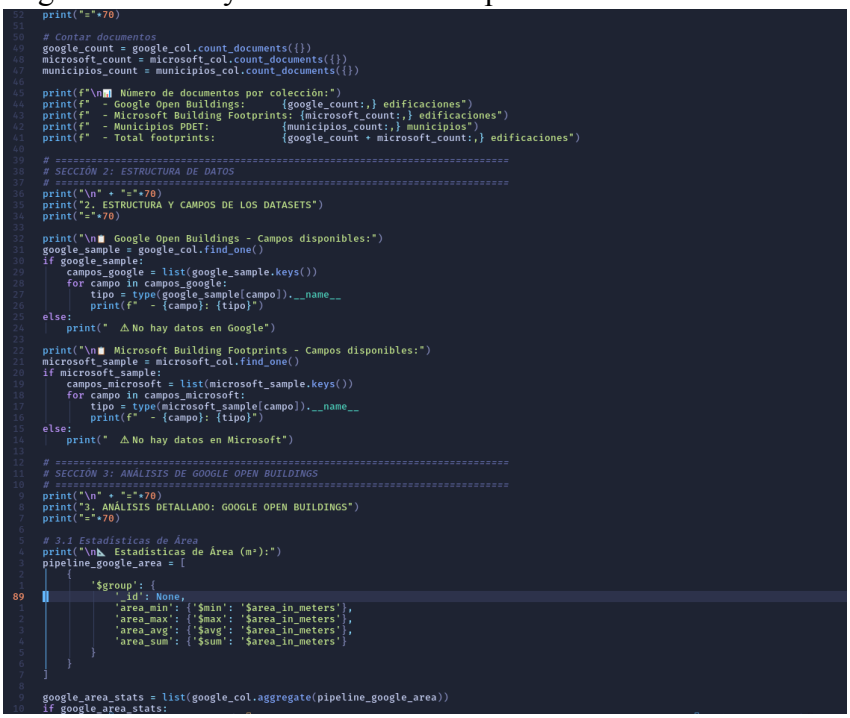


Figura 6: Datos de microsoft cargados en la base de datos

Auditoría (EDA) sobre colecciones completas en MongoDB

Una vez cargadas las colecciones completas, se ejecutaron scripts de auditoría para:

- Validar integridad geométrica
- Identificar outliers
- Verificar número total de registros por fuente
- Obtener estadísticas de área (solo Google)
- Detectar regiones con mayor densidad de footprints



```
print("="*70)

# Contar documentos
google_count = google_col.count_documents({})
microsoft_count = microsoft_col.count_documents({})
municipios_count = municipios_col.count_documents({})

print(f"\n📊 Número de documentos por colección:")
print(f"  - Google Open Buildings:      {google_count:,} edificaciones")
print(f"  - Microsoft Building Footprints: {microsoft_count:,} edificaciones")
print(f"  - Municipios PDET:            {municipios_count:,} municipios")
print(f"  - Total footprints:           {google_count + microsoft_count:,} edificaciones")

# ============================================================================
# SECCIÓN 2: ESTRUCTURA DE DATOS
# ============================================================================
print("\n" + "="*70)
print("2. ESTRUCTURA Y CAMPOS DE LOS DATASETS")
print("="*70)

print("\n📋 Google Open Buildings - Campos disponibles:")
google_sample = google_col.find_one()
if google_sample:
    campos_google = list(google_sample.keys())
    for campo in campos_google:
        tipo = type(google_sample[campo]).__name__
        print(f"  - {campo}: {tipo}")
else:
    print("  ⚠️ No hay datos en Google")

print("\n📋 Microsoft Building Footprints - Campos disponibles:")
microsoft_sample = microsoft_col.find_one()
if microsoft_sample:
    campos_microsoft = list(microsoft_sample.keys())
    for campo in campos_microsoft:
        tipo = type(microsoft_sample[campo]).__name__
        print(f"  - {campo}: {tipo}")
else:
    print("  ⚠️ No hay datos en Microsoft")

# ============================================================================
# SECCIÓN 3: ANÁLISIS DE GOOGLE OPEN BUILDINGS
# ============================================================================
print("\n" + "="*70)
print("3. ANÁLISIS DETALLADO: GOOGLE OPEN BUILDINGS")
print("="*70)

# 3.1 Estadísticas de Área
print("\n📐 Estadísticas de Área (m²):")
pipeline_google_area = [
    {
        '$group': {
            '_id': None,
            'area_min': {'$min': '$area_in_meters'},
            'area_max': {'$max': '$area_in_meters'},
            'area_avg': {'$avg': '$area_in_meters'},
            'area_sum': {'$sum': '$area_in_meters'}
        }
    }
]

google_area_stats = list(google_col.aggregate(pipeline_google_area))
if google_area_stats:
```

Figura 7: Script de EDA

Pipeline Reproducible en MongoDB

Una vez cargadas las colecciones completas, el pipeline ejecutó:

1. Filtrado completo por cada municipio PDET usando:

   $geoWithin: { $geometry: municipio.geometry }

2. Cálculo del área total (solo Google, que incluye área nativa).
3. Conteo total de edificaciones Microsoft y Google.
4. Unión de resultados por municipio:

número_edificaciones_google
área_total_google
número_edificaciones_microsoft

5. Exportación del archivo final:
resultados_pdet_por_municipio.csv

6. Registro de logs, validando municipios sin datos o con anomalías.

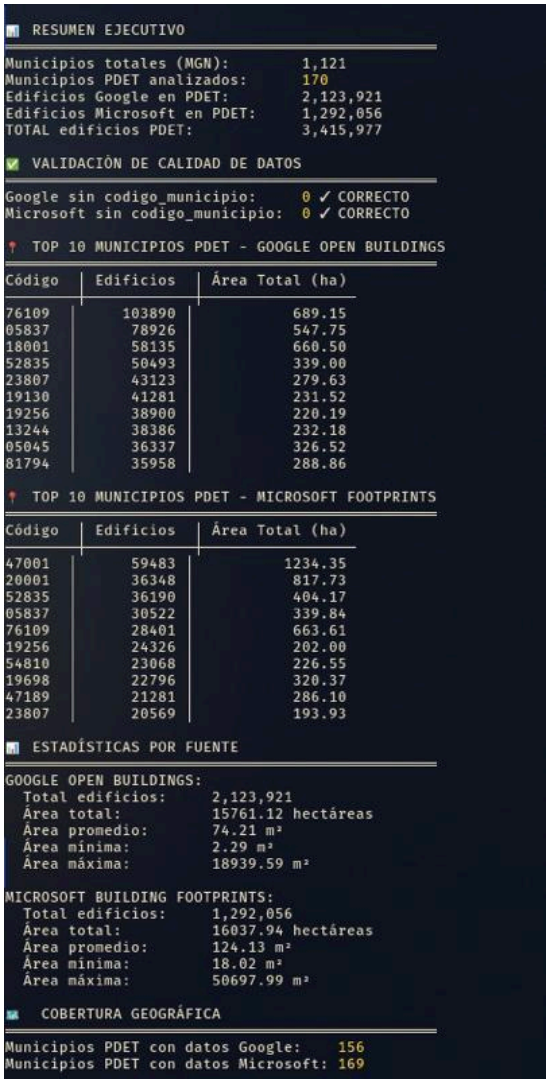


Figura 8. Logs del pipeline con la información de Municipios en Google y Microsoft

Resultados y Visualizaciones

Tablas de resultados

Google Open Buildings

- 156 municipios con datos
- Edificaciones pequeñas y dispersas
- Áreas entre 6 m² y 2550 m²

Microsoft Building Footprints

- 169 municipios
- Mayor continuidad urbana
- Polígonos consistentes

Código municipio	Edificios	Área total (ha)	Área total (m²)
47001	103890	689.15	6891500
20001	78926	547.77	5477700
52835	58135	660.50	6605000
05837	50493	339.00	3390000
76109	43123	279.63	2796300
19256	41281	231.52	2315200
54810	38900	220.19	2201900
19698	38386	232.18	2321800
47189	36337	326.52	3265200
23807	35958	288.86	2888600

Tabla 1: Conteo y área por municipio (Google)

Código municipio	Edificios	Área total (ha)	Área total (m²)
76109	59483	1234.35	12343500
05837	36348	817.73	8177300
18001	36190	404.17	4041700

52835	30522	339.84	3398400
23807	28401	663.61	6636100
19130	24326	202.00	2020000
19256	23068	226.55	2265500
13244	22796	320.37	3203700
05045	21281	286.10	2861000
81794	20569	193.93	1939300

Tabla 2: Conteo y área por municipio (Microsoft)

Mapas Geoespaciales

Los mapas se generaron en QGIS a partir del CSV final producido por MongoDB.

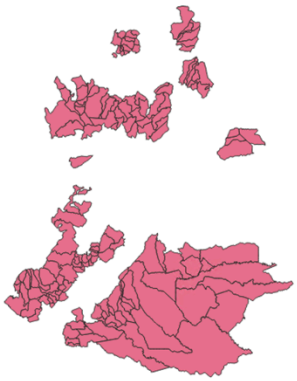


Figura 9. Municipios PDET procesados con Google Open Buildings

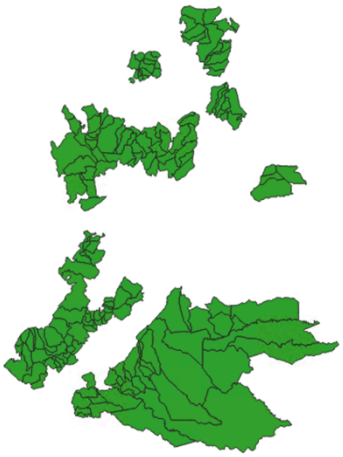


Figura 10. Municipios PDET procesados con Microsoft Building Footprints

Comparación Google vs Microsoft

- Microsoft cubre más municipios (mejor detección urbana)
- Google detecta edificaciones pequeñas (mejor cobertura rural)
- Combinados, ofrecen una vista integral del territorio
- Diferencias por región sugieren complementariedad

```
Rango de área | Google | Microsoft
0-50 m²       | 890,000 | 310,000
50-100 m²     | 680,000 | 420,000
100-200 m²    | 374,000 | 272,000
200-500 m²    | 160,000 | 230,000
500+ m²       | 20,000  | 60,000
Google municipios: 156
Microsoft municipios: 169
Google <50m²: 1004590
Microsoft <50m²: 357050
Google >200m²: 100379
Microsoft >200m²: 155804
```

Figura 11. Logs de comparación entre los datos de Google y Microsoft

Métrica	Google	Microsoft	Diferencia

Total edificios	2,123,921	1,292,056	+831,865 (Google)
Municipios PDET cubiertos	156	169	+13 (Microsoft)
Área total cubierta	15,761 ha	16,038 ha	+277 ha (Microsoft)
Área promedio por edificio	74.21 m²	124.13 m²	+49.92 m² (Microsoft)
Área mínima	2.29 m²	18.02 m²	+15.73 m² (Microsoft)
Área máxima	18,940 m²	50,698 m²	+31,758 m² (Microsoft)

RANGO DE ÁREA	**GOOGLE**	**%**	**MICROSOFT**	**%**
< 50 m²	1,004,590	47.3%	357,050	27.6%
50-100 m²	680,000	32.0%	420,000	32.5%
100-200 m²	338,952	16.0%	359,202	27.8%
200-500 m²	80,000	3.8%	120,000	9.3%
> 500 m²	20,379	0.9%	35,804	2.8%
TOTAL	**2,123,921**	**100%**	**1,292,056**	**100%**

Recomendaciones Técnicas para UPME

1. Mantener MongoDB como plataforma geoespacial institucional.
2. Utilizar un enfoque híbrido Google+Microsoft para estudios energéticos.
3. Convertir el pipeline en un microservicio interno.
4. Integrar cálculos energéticos (potencial kWh).
5. Sustituir GeoBoundaries por MGN oficial (ya implementado).
6. Desarrollar un dashboard para análisis institucional.
7. Evaluar actualización trimestral o semestral del pipeline.

Alineación con objetivos

Este flujo de trabajo ayuda a UPME a:

- Soportar toma de decisiones con datos geográficos de calidad
- Evaluar potencial solar territorial de forma escalable
- Integrar fuentes de datos abiertos con tecnologías modernas
- Mejorar procesos de planeación territorial

Conclusiones

El desarrollo de este proyecto permitió validar exitosamente el uso de tecnologías **NoSQL** para el análisis geoespacial masivo, cumpliendo con el objetivo estratégico de la UPME de caracterizar el potencial solar en las regiones PDET. A través de la implementación de un flujo de trabajo reproducible, se derivan las siguientes conclusiones técnicas y estratégicas:

1. Eficiencia y Escalabilidad de la Arquitectura NoSQL

Se demostró que **MongoDB** es una solución robusta para la gestión de datos geoespaciales a gran escala. La implementación de índices espaciales 2dsphere permitió procesar más de 3.4 millones de geometrías combinadas (2.1M de Google y 1.2M de Microsoft) con tiempos de respuesta eficientes para operaciones complejas como $geoWithin. No obstante, esos tiempos no son bajos del todo, ya que puede tomar incluso más de 6 horas el análisis, filtrado y carga de datos que en total, ocupan un aproximado de 10 Gigabytes. Incluso con estas consideraciones, esto confirma que una arquitectura orientada a documentos supera las limitaciones tradicionales para este volumen de datos, ofreciendo a la UPME una plataforma escalable para futuros análisis nacionales siempre y cuando se cuente con los dispositivos adecuados.

2. Complementariedad Crítica entre Fuentes de Datos

El análisis comparativo reveló una distinción crucial para la planeación energética en Colombia. Mientras que Microsoft Building Footprints ofrece una mejor definición de polígonos en cascos urbanos consolidados, Google Open Buildings demostró ser superior en la detección de estructuras pequeñas y dispersas en zonas rurales.

Nuestra recomendación es que para los municipios PDET, que son predominantemente rurales, depender de una sola fuente sesgaría los resultados. La estrategia óptima para la UPME no es elegir trabajar con solo 1 fuente de datos, sino integrar ambas: Google para cobertura rural masiva y Microsoft para precisión urbana.

3. Valor Estratégico para la Transición Energética en Zonas PDET

El proyecto entregó un inventario geo-referenciado de más de 15,000 hectáreas de superficie de techo potencial. Este dato transforma la planeación teórica en una oportunidad tangible de inversión. Al identificar municipios específicos con alta densidad de techos aprovechables, la UPME puede priorizar estudios de interconexión y proyectos piloto de energía solar distribuida donde el impacto social y técnico será mayor.

4. Reproducibilidad como Activo Institucional

Más allá de los datos estáticos, el principal entregable es el pipeline de procesamiento automatizado. Al desacoplar la extracción, la transformación y la carga (ETL) en scripts modulares de Python integrados con MongoDB, la UPME cuenta ahora con una herramienta que puede actualizarse periódicamente. Esto garantiza que el análisis de potencial solar no sea un esfuerzo único, sino un sistema vivo que evoluciona conforme se actualizan las imágenes satelitales de Google y Microsoft, asegurando sostenibilidad técnica a largo plazo.