Primera Entrega
Base de Datos NoSQL Diseño de Esquema y Plan de Implementación

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.
Octubre 2025

Índice

Introducción

Como parte del proyecto final del curso Administración de Base de datos, este documento presenta la propuesta inicial desarrollada por la consultora, en colaboración con la Unidad de Planeación Minero Energética (UPME), para diseñar una metodología reproducible que permita estimar el potencial solar en las cubiertas de edificaciones dentro de los municipios PDET.

El proyecto busca identificar las áreas urbanas y rurales con mayor capacidad para la instalación de paneles solares, aprovechando fuentes de datos abiertas y de alta resolución, como Microsoft Building Footprints, Google Open Buildings y el Marco Geoestadístico Nacional (MGN) del DANE.

Con el fin de asegurar una infraestructura flexible, escalable y eficiente en el manejo de información geoespacial masiva, se adopta un enfoque NoSQL, particularmente utilizando MongoDB, para el almacenamiento, indexación y consulta espacial de los datos.

Este documento define el plan de implementación, el modelado de datos y el diseño del esquema NoSQL, sentando las bases para las siguientes fases del proyecto: integración de límites PDET, carga e integración de datasets y flujo reproducible de análisis geoespacial.

Plan de implementación

1. **Objetivo técnico**

    a. Almacenar eficientemente grandes volúmenes de geometrías de edificios provenientes de fuentes abiertas.
    b. Integrar la información geoespacial del DANE (MGN) para delimitar los municipios PDET.
    c. Realizar consultas y operaciones espaciales, como conteo y estimación de áreas dentro de cada municipio.
    d. Servir como base para la generación de tablas y mapas reproducibles en etapas posteriores.

2. **Tecnologías seleccionadas**

    a. MongoDB: base NoSQL con soporte nativo para índices 2dsphere y consultas espaciales.
    b. Python: lenguaje principal para carga, transformación y validación de datos.
    c. Docker: despliegue reproducible del entorno de base de datos y scripts ETL.
    d. GitHub: control de versiones y almacenamiento de commits.
    e. GeoJSON (EPSG:4326): formato estándar para representar datos espaciales.

3. **Flujo de trabajo propuesto**

1. Adquisición de datos: descargar los datasets de Microsoft, Google y MGN/DANE.
2. Transformación y limpieza: estandarizar a formato GeoJSON, calcular áreas y centroides, y validar geometrías.
3. Cargar en MongoDB: insertar los datos en colecciones separadas según la fuente.
4. Creación de índices espaciales: generación índices 2dsphere en los campos de geometría y centroides.
5. Verificación de integridad: ejecutar pruebas de consultas $geoWithin y $geoIntersects.
6. Documentación: registrar el proceso de implementación en el repositorio y documentos.

Modelo de datos

El modelo de datos propuesto sigue un enfoque NoSQL enfocado en manejar información geoespacial masiva de forma eficiente, flexible y escalable.

MongoDB se seleccionó por su capacidad nativa para manejar GeoJSON, consultas espaciales e índices 2dsphere, que permiten realizar operaciones como $geoWithin, $geoIntersects y $near con alto rendimiento.

El modelo propuesto considera tres niveles:

1. Datos administrativos: límites municipales PDET (DANE/MGN).

2. Datos de edificaciones: huellas de construcción (Microsoft y Google).

3. Datos analíticos: resultados agregados por municipio.

Colecciones que usaremos:

1.mgn_municipios_pdet

Campo	**Tipo**	**Descripción**
_id	ObjectId	Identificador único del documento
codigo_municipio	String	Código DANE del municipio
nombre_municipio	String	Nombre del municipio
departamento	String	Nombre del departamento
pdet	Boolean	Indica si el municipio pertenece al programa PDET
geometry	GeoJSON (Polygon)	Límite geográfico del municipio

2.buildings_microsoft

Campo	Tipo	Descripción
_id	ObjectId	Identificador interno
building_id	String	ID único del edificio
fuente	String	"Microsoft"
codigo_municipio	String	Código DANE del municipio asociado
geometry	GeoJSON (Polygon)	Forma del edificio
centroid	GeoJSON (Point)	Centroide calculado del edificio
area_m2	Number	Área del edificio en metros cuadrados

3. buildings_google

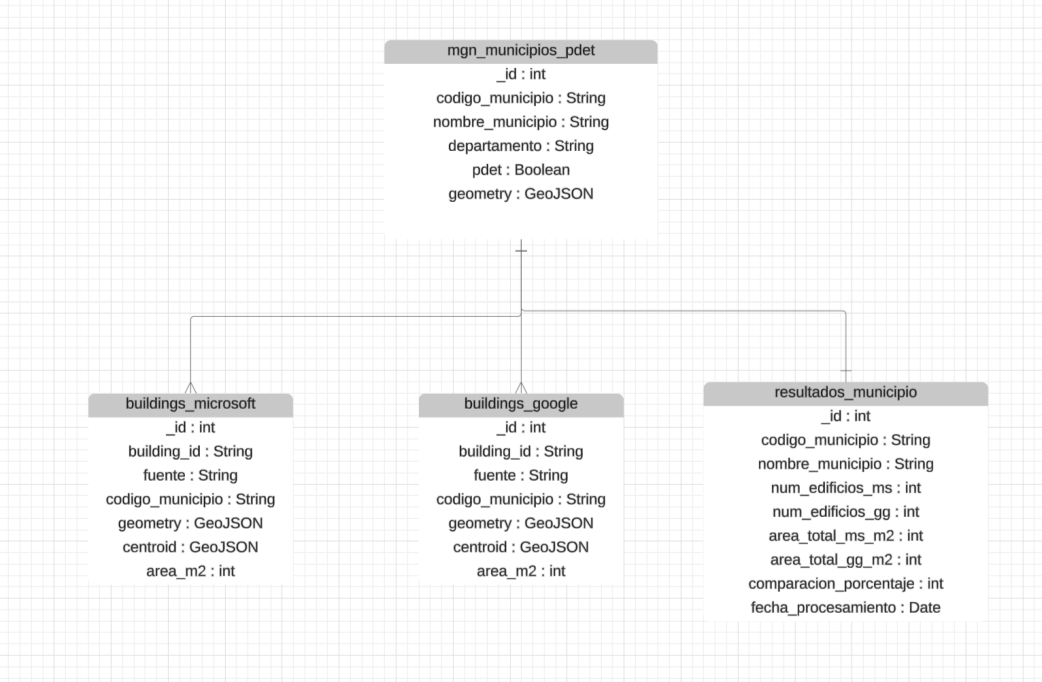
Campo	Tipo	Descripción
_id	ObjectId	Identificador interno
building_id	String	ID único del edificio
fuente	String	"Google"
codigo_municipio	String	Código DANE del municipio asociado
geometry	GeoJSON (Polygon)	Forma del edificio
centroid	GeoJSON (Point)	Centroide calculado
area_m2	Number	Área estimada del edificio

4. resultados_municipio

Campo	Tipo	Descripción
_id	ObjectId	Identificador del documento
codigo_municipio	String	Código DANE
nombre_municipio	String	Nombre del municipio
num_edificios_ms	Number	Número de edificios (Microsoft)
num_edificios_gg	Number	Número de edificios (Google)

area_total_ms_m2	Number	Suma de áreas (Microsoft)
area_total_gg_m2	Number	Suma de áreas (Google)
comparacion_porcentaje	Number	Diferencia porcentual entre fuentes
fecha_procesamiento	Date	Fecha de la última actualización de resultados

Diagrama del modelo de datos



Apropiaciones

La estructura se eligió principalmente por seis razones:

1. **Flexibilidad:** Se eligió un enfoque NoSQL con MongoDB porque permite agregar fácilmente nuevos campos a futuro (como potencial_kWh) sin tener que modificar toda la configuración del modelo.
2. **Eficiencia Espacial:** La estructura está diseñada para usar los índices espaciales 2dsphere de MongoDB. Esto es crucial porque el proyecto depende de consultas geográficas.

3. **Facilidad de Comparación:** La estructura separa deliberadamente los datos de Microsoft y Google en colecciones independientes (buildings_microsoft y buildings_google). Esto se hizo para cumplir con el requisito de la UPME de facilitar la comparación entre las dos fuentes de datos, permitiendo analizar diferencias de cobertura y precisión.
4. **Escalabilidad**: permite manejar millones de geometrías distribuidas por fuentes y municipios sin afectar el rendimiento de la base de datos. Esta característica es esencial para procesar los volúmenes de datos provenientes de Microsoft y Google.
5. **Reproducibilidad:** el modelo es completamente replicable y se integra fácilmente con flujos ETL desarrollados en Python y ejecutados en entornos Docker, garantizando consistencia y trazabilidad en el proceso de análisis.
6. **Separación por fuente:** el almacenamiento independiente de los datos de Microsoft y Google facilita la comparación entre conjuntos, permitiendo analizar diferencias en cobertura y precisión, y cumpliendo con los requerimientos establecidos por la UPME.

Diseño de esquema

El diseño de esquema define la forma exacta en la que se almacenarán los documentos dentro de MongoDB, estableciendo los campos, tipos de datos y estructuras compatibles con el formato GeoJSON. Este diseño busca mantener la coherencia con el modelo lógico, garantizar la eficiencia en las operaciones espaciales y permitir la expansión futura del sistema.

A continuación, se detalla el esquema propuesto para cada colección.

Esquema de mgn_municipios_pdet

Descripción: Almacena los límites geográficos oficiales de los municipios clasificados como PDET según el DANE. Esta colección sirve como capa de referencia para las consultas espaciales.

```
{
  "_id": "ObjectId('67f4c6e7a1f8f3cde3b5e7d1')",
  "codigo_municipio": "13433",
  "nombre_municipio": "María La Baja",
  "departamento": "Bolívar",
  "pdet": true,
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [ -75.3116, 9.9745 ],
        [ -75.3106, 9.9744 ],
        // ... miles de coordenadas ...
        [ -75.3116, 9.9745 ]
      ]
    ]
  }
}
```


Figura 1. Esquema mgn_municipios_pdet

Esquema de buildings_google y buildings_microsoft

Descripción: Almacenan los polígonos individuales (huellas) de los edificios. Se crean dos colecciones separadas, pero con idéntica estructura, para cada fuente de datos (Google y Microsoft). Esto es un requisito clave para poder comparar los resultados de cada proveedor.


```
{
  "_id": "ObjectId('67f4c6e7a1f8f3cde3b5e7d2')",
  "building_id": "MS-Bldg-12345ABC",
  "fuente": "Microsoft",
  "codigo_municipio": "13433",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [ -75.1451, 4.8431 ],
        [ -75.1452, 4.8432 ],
        // ... coordenadas del edificio ...
        [ -75.1451, 4.8431 ]
      ]
    ]
  },
  "centroid": {
    "type": "Point",
    "coordinates": [ -75.14515, 4.84315 ]
  },
  "area_m2": 120.5
}
```

Figura 2. Esquema buildings_google y microsoft

Esquema de resultados_municipio

Descripción: Esta es la colección analítica final, diseñada para almacenar los resultados agregados y comparativos del proyecto. Contendrá un único documento por cada municipio PDET, resumiendo los hallazgos de ambas fuentes.

```
{
  "_id": "ObjectId('67f4c6e7a1f8f3cde3b5e7d3')",
  "codigo_municipio": "13433",
  "nombre_municipio": "María La Baja",
  "num_edificios_ms": 42100,
  "num_edificios_gg": 45200,
  "area_total_ms_m2": 1198030.20,
  "area_total_gg_m2": 1250340.75,
  "comparacion_porcentaje": 4.35, // Ej. (Area_gg - Area_ms) / Area_ms
  "fecha_procesamiento": "ISODate('2025-11-16T20:30:00Z')"
}
```

Figura 3. Esquema resultados_municipio

Conclusiones

MongoDB se selecciona como la base de datos NoSQL central del proyecto, una decisión justificada por su capacidad nativa para manejar eficientemente volúmenes masivos de información geoespacial. El modelo aprovecha los índices *2dsphere* para optimizar las consultas geográficas, que son cruciales para el análisis. Esta arquitectura NoSQL no solo proporciona la flexibilidad necesaria para añadir futuros campos analíticos (como potencial_kWh) sin modificar el esquema, sino que también garantiza un flujo de trabajo reproducible al integrarse con Python y Docker (herramientas que se usarán para la implementación del proyecto).

Por otro lado, el diseño del esquema responde directamente a un requisito clave de la UPME al separar deliberadamente los datos de Microsoft y Google en colecciones independientes (buildings_microsoft y buildings_google). Esta separación es fundamental para facilitar la comparación de cobertura y precisión entre ambas fuentes. El modelo define una clara estructura de tres niveles (administrativo, datos crudos y resultados analíticos), sentando una base sólida para las siguientes fases del proyecto, que incluyen la carga masiva de datos y la ejecución del análisis geoespacial.