

Plan de Implementación NoSQL

Análisis Geoespacial de Potencial Solar en Territorios PDET

Entrega 1 - Diseño de Esquema y Base de Datos

Proyecto Final - Administración de Bases de Datos

Universidad de los Andes

Octubre 27, 2025

Índice

| | |
|--|-----------|
| 1. Resumen Ejecutivo | 4 |
| 1.1. Objetivos del Proyecto | 4 |
| 1.2. Alcance de la Entrega 1 | 4 |
| 2. Selección de Tecnología NoSQL | 5 |
| 2.1. Tecnología Seleccionada: MongoDB | 5 |
| 2.1.1. Justificación Técnica | 5 |
| 2.2. Comparación con Alternativas | 6 |
| 3. Arquitectura del Sistema | 7 |
| 3.1. Arquitectura General | 7 |
| 3.2. Componentes Tecnológicos | 7 |
| 3.3. Diagrama de Arquitectura | 7 |
| 4. Modelo de Datos NoSQL | 9 |
| 4.1. Paradigma de Modelado | 9 |
| 4.2. Entidades Principales | 9 |
| 4.2.1. 1. Municipios PDET | 9 |
| 4.2.2. 2. Edificios Microsoft | 9 |
| 4.2.3. 3. Edificios Google | 10 |
| 4.3. Relaciones entre Entidades | 10 |
| 5. Diseño del Esquema MongoDB | 11 |
| 5.1. Colección: municipalities | 11 |
| 5.2. Colección: buildings_microsoft | 11 |
| 5.3. Colección: buildings_google | 12 |
| 5.4. Estrategia de Índices | 13 |
| 5.4.1. Índices Espaciales (2dsphere) | 13 |
| 5.4.2. Índices Compuestos | 13 |
| 5.4.3. Consideraciones de Sharding | 13 |
| 6. Operaciones Espaciales Clave | 14 |
| 6.1. Asignación de Edificios a Municipios | 14 |
| 6.2. Agregación por Municipio | 14 |
| 6.3. Comparación entre Datasets | 14 |
| 7. Plan de Implementación Detallado | 16 |
| 7.1. Cronograma de Entregas | 16 |
| 7.2. Fases de Implementación | 17 |
| 7.2.1. Fase 1: Configuración del Entorno (27-28 Oct) | 17 |
| 7.2.2. Fase 2: Carga de Municipios PDET (29 Oct - 3 Nov) | 17 |
| 7.2.3. Fase 3: Integración de Edificios (4-10 Nov) | 17 |
| 7.2.4. Fase 4: Análisis Espacial (11-17 Nov) | 18 |
| 7.2.5. Fase 5: Reporte y Documentación (18-24 Nov) | 18 |

| | |
|--|-----------|
| 8. Consideraciones de Escalabilidad | 19 |
| 8.1. Gestión de Volumen de Datos | 19 |
| 8.1.1. Estrategias de Optimización | 19 |
| 8.2. Optimización de Consultas | 19 |
| 8.3. Monitoreo y Rendimiento | 19 |
| 9. Validación del Esquema | 20 |
| 9.1. JSON Schema Validation | 20 |
| 9.2. Validación de Geometrías | 20 |
| 10.Gestión de Riesgos | 21 |
| 11.Conclusiones | 21 |
| 12.Referencias | 22 |

1. Resumen Ejecutivo

Este documento presenta el plan de implementación para el desarrollo de una solución NoSQL orientada al análisis geoespacial de potencial solar en territorios PDET (Programas de Desarrollo con Enfoque Territorial) de Colombia. El proyecto tiene como objetivo principal diseñar e implementar un flujo de trabajo reproducible para estimar el potencial energético solar mediante el análisis de superficies de techos de edificaciones.

1.1. Objetivos del Proyecto

- Diseñar e implementar una base de datos NoSQL escalable para almacenar datos geoespaciales de edificaciones
- Integrar y comparar dos datasets masivos: Microsoft Building Footprints (999M edificios) y Google Open Buildings (1.8B edificios)
- Realizar análisis espaciales para calcular el número de edificios y área total de techos por municipio PDET
- Generar un reporte técnico con recomendaciones para la UPME sobre ubicaciones óptimas para proyectos piloto de energía solar

1.2. Alcance de la Entrega 1

Esta primera entrega establece los fundamentos tecnológicos del proyecto mediante:

- Selección y justificación de la solución NoSQL
- Diseño del modelo de datos
- Definición del esquema de base de datos
- Plan de implementación detallado

2. Selección de Tecnología NoSQL

2.1. Tecnología Seleccionada: MongoDB

Se ha seleccionado **MongoDB** como la solución NoSQL para este proyecto por las siguientes razones técnicas fundamentales:

2.1.1. Justificación Técnica

1. Soporte Geoespacial Nativo

- MongoDB proporciona soporte completo para datos geoespaciales mediante el estándar **GeoJSON**
- Implementa índices espaciales **2dsphere** que permiten consultas geométricas eficientes en superficies esféricas (crucial para coordenadas geográficas)
- Operadores espaciales nativos: `$geoWithin`, `$geoIntersects`, `$near`, fundamentales para determinar edificios dentro de límites municipales

2. Escalabilidad Horizontal

- Arquitectura de **sharding** automático que permite distribuir billones de documentos en múltiples nodos
- Capacidad de manejar los datasets masivos del proyecto (999M + 1.8B edificios)
- Replicación nativa para alta disponibilidad y tolerancia a fallos

3. Flexibilidad del Modelo de Documentos

- Esquema flexible que permite almacenar estructuras heterogéneas de Microsoft y Google
- Facilita la comparación de datasets con atributos diferentes
- Soporte para documentos embebidos y arrays, ideal para geometrías complejas

4. Rendimiento en Consultas Agregadas

- Framework de agregación (**aggregate pipeline**) optimizado para cálculos estadísticos complejos
- Capacidad de realizar agrupaciones por municipio y cálculos de área en tiempo eficiente
- Índices compuestos para optimizar consultas por ubicación y atributos

5. Ecosistema y Herramientas

- Excelente integración con Python (PyMongo, Motor) para análisis de datos
- MongoDB Atlas para gestión en la nube (si se requiere)
- MongoDB Compass para visualización y exploración de datos
- Soporte para MongoDB Charts para visualizaciones geoespaciales

2.2. Comparación con Alternativas

Cuadro 1: Comparación de Soluciones NoSQL

| Característica | MongoDB | Cassandra | Neo4j |
|---------------------------|-------------|------------------|-------------|
| Soporte Geoespacial | Excelente | Limitado | Básico |
| Escalabilidad | Excelente | Excelente | Buena |
| Consultas Espaciales | Nativas | Requiere plugins | Limitadas |
| Modelo de Datos | Documentos | Columnar | Grafos |
| Curva de Aprendizaje | Baja | Alta | Media |
| Integración Python | Excelente | Buena | Buena |
| Idoneidad Proyecto | 95 % | 60 % | 40 % |

3. Arquitectura del Sistema

3.1. Arquitectura General

El sistema se estructura en cuatro capas principales:

1. Capa de Almacenamiento (Storage Layer)

- MongoDB Database Server
- Colecciones: `municipalities`, `buildings_microsoft`, `buildings_google`
- Índices espaciales y de búsqueda

2. Capa de Ingesta de Datos (Data Ingestion Layer)

- Scripts Python para descarga de datasets
- Procesamiento ETL (Extract, Transform, Load)
- Validación de geometrías GeoJSON
- Carga masiva mediante `bulk.write`

3. Capa de Procesamiento (Processing Layer)

- Motor de consultas espaciales
- Agregaciones por municipio
- Cálculo de áreas y conteos
- Comparación entre datasets

4. Capa de Presentación (Presentation Layer)

- Scripts de análisis en Python (Pandas, GeoPandas)
- Generación de reportes
- Visualizaciones (Matplotlib, Folium)
- Exportación de resultados

3.2. Componentes Tecnológicos

3.3. Diagrama de Arquitectura

La arquitectura sigue un patrón de procesamiento por lotes (batch processing) donde:

Fuentes de Datos (Microsoft, Google, DANE)

|

v

ETL Pipeline (Python)

|

v

MongoDB Database

- `municipalities`

- `buildings_microsoft`

Cuadro 2: Stack Tecnológico

| Componente | Tecnología |
|---------------------------|-------------------------------|
| Base de Datos | MongoDB 7.0+ |
| Driver de Conexión | PyMongo 4.5+ |
| Procesamiento Geoespacial | GeoPandas 0.14+, Shapely 2.0+ |
| Análisis de Datos | Pandas 2.0+, NumPy 1.24+ |
| Visualización | Matplotlib, Folium, Plotly |
| Lenguaje Principal | Python 3.11+ |
| Gestión de Entorno | Conda / Poetry |
| Control de Versiones | Git / GitHub |
| Documentación | Jupyter Notebooks, LaTeX |

```
- buildings_google
  |
  v
Spatial Queries & Aggregations
  |
  v
Analysis & Reporting
```


4. Modelo de Datos NoSQL

4.1. Paradigma de Modelado

El diseño del modelo de datos sigue los principios de modelado NoSQL orientado a documentos:

- **Desnormalización estratégica:** Los datos geométricos se embeben en los documentos para reducir joins
- **Índices especializados:** Uso de índices 2dsphere para consultas espaciales
- **Colecciones separadas por fuente:** Permite comparaciones directas entre datasets
- **Metadatos en cada documento:** Trazabilidad y auditoría de origen de datos

4.2. Entidades Principales

4.2.1. 1. Municipios PDET

Almacena los límites administrativos de los municipios designados como territorios PDET según el Marco Geoestadístico Nacional (MGN) del DANE.

Atributos clave:

- Código DANE (identificador único)
- Nombre del municipio y departamento
- Geometría del límite municipal (Polygon/MultiPolygon)
- Indicador PDET
- Metadatos administrativos

4.2.2. 2. Edificios Microsoft

Detecciones de edificios del dataset Microsoft Building Footprints.

Atributos clave:

- Geometría del footprint (Polygon)
- Código del municipio al que pertenece
- Área calculada en m²
- Fuente y fecha de captura
- Nivel de confianza (si disponible)

4.2.3. 3. Edificios Google

Detecciones de edificios del dataset Google Open Buildings.

Atributos clave:

- Geometría del footprint (Polygon)
- Código del municipio al que pertenece
- Área calculada en m²
- Confianza del modelo (confidence score)
- Versión del dataset

4.3. Relaciones entre Entidades

En un modelo NoSQL orientado a documentos, las relaciones se manejan mediante:

- **Referencia:** Los edificios referencian municipios mediante `municipality_code`
- **Índices espaciales:** Permiten consultas de contención (edificios dentro de municipios)
- **No hay joins tradicionales:** Se usan agregaciones con `$lookup` cuando es necesario

Relación conceptual:

Municipality (1) ----< (N) Building Microsoft

Municipality (1) ----< (N) Building Google

5. Diseño del Esquema MongoDB

5.1. Colección: municipalities

```
{
  "_id": ObjectId("..."),
  "codigo_dane": "05001",
  "nombre_municipio": "Medellin",
  "nombre_departamento": "Antioquia",
  "is_pdet": true,
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [-75.6068, 6.2442],
        [-75.6000, 6.2500],
        ...
      ]
    ]
  },
  "area_km2": 105.2,
  "metadata": {
    "source": "DANE-MGN",
    "version": "2.0",
    "load_date": ISODate("2025-10-27T00:00:00Z")
  }
}
```

Índices:

- codigo_dane: unique index
- geometry: 2dsphere spatial index
- is_pdet: regular index (filtrado)

5.2. Colección: buildings_microsoft

```
{
  "_id": ObjectId("..."),
  "municipality_code": "05001",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [-75.6050, 6.2450],
        [-75.6048, 6.2450],
        [-75.6048, 6.2448],
        [-75.6050, 6.2448],
        [-75.6050, 6.2450]
      ]
    ]
  },
}
```

```

    "properties": {
      "area_m2": 156.8,
      "confidence": 0.95,
      "height_estimated": null
    },
    "metadata": {
      "source": "Microsoft",
      "capture_date_range": "2014-2021",
      "load_date": ISODate("2025-11-03T00:00:00Z"),
      "version": "v1.0"
    }
  }
}

```

Índices:

- geometry: 2dsphere spatial index
- municipality_code: regular index
- compound: {municipality_code: 1, "properties.area_m2": 1}

5.3. Colección: buildings_google

```

{
  "_id": ObjectId("..."),
  "municipality_code": "05001",
  "geometry": {
    "type": "Polygon",
    "coordinates": [
      [
        [-75.6052, 6.2451],
        [-75.6050, 6.2451],
        [-75.6050, 6.2449],
        [-75.6052, 6.2449],
        [-75.6052, 6.2451]
      ]
    ]
  },
  "properties": {
    "area_m2": 148.3,
    "confidence": 0.87,
    "full_plus_code": "67Q6+6RW Medellín"
  },
  "metadata": {
    "source": "Google",
    "dataset_version": "v3",
    "region": "Latin America",
    "load_date": ISODate("2025-11-03T00:00:00Z")
  }
}

```

Índices:

- geometry: 2dsphere spatial index

- `municipality_code`: regular index
- `compound`: `{municipality_code: 1, "properties.area_m2": 1}`

5.4. Estrategia de Índices

5.4.1. Índices Espaciales (2dsphere)

Los índices 2dsphere son fundamentales para:

- Consultas `$geoWithin`: Encontrar edificios dentro de límites municipales
- Consultas `$geoIntersects`: Detectar intersecciones entre geometrías
- Consultas `$near`: Búsquedas por proximidad (si se requieren)

5.4.2. Índices Compuestos

Permiten optimizar consultas que filtran por municipio y ordenan/filtran por área:

```
db.buildings_microsoft.createIndex(  
  { municipality_code: 1, "properties.area_m2": -1 }  
)
```

5.4.3. Consideraciones de Sharding

Para datasets masivos, el sharding se configurará con:

- **Shard Key**: `municipality_code` (distribución equitativa por municipio)
- **Alternativa**: Hash de `_id` para distribución uniforme

6. Operaciones Espaciales Clave

6.1. Asignación de Edificios a Municipios

Consulta espacial para determinar a qué municipio pertenece un edificio:

```
# Pseudocódigo
for building in buildings_collection:
    municipality = municipalities.find_one({
        "geometry": {
            "$geoIntersects": {
                "$geometry": building.geometry
            }
        }
    })
    building.municipality_code = municipality.codigo_dane
    building.save()
```

6.2. Agregación por Municipio

Pipeline de agregación para calcular totales por municipio:

```
pipeline = [
    {
        "$match": {
            "municipality_code": {"$exists": True}
        }
    },
    {
        "$group": {
            "_id": "$municipality_code",
            "total_buildings": {"$sum": 1},
            "total_area_m2": {"$sum": "$properties.area_m2"},
            "avg_area_m2": {"$avg": "$properties.area_m2"}
        }
    },
    {
        "$sort": {"total_area_m2": -1}
    }
]

results = db.buildings_microsoft.aggregate(pipeline)
```

6.3. Comparación entre Datasets

Unión de resultados de Microsoft y Google para comparación:

```
# Agregacion para Microsoft
ms_results = db.buildings_microsoft.aggregate([...])

# Agregacion para Google
google_results = db.buildings_google.aggregate([...])
```

```
# Comparacion en Python/Pandas
comparison_df = pd.merge(
    ms_results_df,
    google_results_df,
    on='municipality_code',
    suffixes=('_ms', '_google')
)
```

7. Plan de Implementación Detallado

7.1. Cronograma de Entregas

| Fecha | Entrega | Actividades Principales |
|----------------|---------------------------------|--|
| 27 Oct | Entrega 1: Diseño de Esquema | <ul style="list-style-type: none"> - Documento LaTeX del plan - Diagramas PlantUML - Definición de esquemas JSON - README actualizado |
| 28 Oct - 2 Nov | Implementación MongoDB | <ul style="list-style-type: none"> - Instalación y configuración de MongoDB - Creación de base de datos y colecciones - Implementación de índices - Scripts de validación |
| 3 Nov | Entrega 2: Integración PDET | <ul style="list-style-type: none"> - Descarga de datos DANE/MGN - Procesamiento de geometrías - Carga a MongoDB - Validación espacial - Documentación del proceso |
| 4-9 Nov | Preparación Datasets Edificios | <ul style="list-style-type: none"> - Análisis de formatos Microsoft/-Google - Scripts de ETL - Pruebas de carga masiva - Optimización de rendimiento |
| 10 Nov | Entrega 3: Carga de Edificios | <ul style="list-style-type: none"> - Carga completa datasets - Índices espaciales - EDA inicial - Reporte de auditoría |
| 11-16 Nov | Análisis Espacial | <ul style="list-style-type: none"> - Asignación municipio-edificio - Agregaciones por territorio - Cálculos de área - Comparación datasets |
| 17 Nov | Entrega 4: Workflow Geoespacial | <ul style="list-style-type: none"> - Scripts reproducibles - Resultados tabulares - Mapas y visualizaciones - Documentación metodológica |
| 18-23 Nov | Reporte Final | <ul style="list-style-type: none"> - Redacción del informe técnico - Análisis de resultados - Recomendaciones UPME - Visualizaciones finales |
| 24 Nov | Entrega 5: Reporte Final | <ul style="list-style-type: none"> - Documento completo - Presentación de defensa |

| Fecha | Entrega | Actividades Principales |
|-------|---------|-------------------------------|
| | | - Repositorio GitHub completo |

7.2. Fases de Implementación

7.2.1. Fase 1: Configuración del Entorno (27-28 Oct)

1. Instalación de MongoDB Community Edition
2. Configuración de Python environment (conda/venv)
3. Instalación de dependencias: PyMongo, GeoPandas, Shapely, etc.
4. Configuración de conexión a MongoDB
5. Pruebas de conectividad

7.2.2. Fase 2: Carga de Municipios PDET (29 Oct - 3 Nov)

1. Descarga del MGN desde portal DANE
2. Filtrado de municipios PDET
3. Conversión de geometrías a GeoJSON
4. Validación de geometrías (topología)
5. Carga a colección `municipalities`
6. Creación de índices espaciales
7. Verificación de datos

7.2.3. Fase 3: Integración de Edificios (4-10 Nov)

1. Descarga de datasets (Microsoft y Google)
2. Procesamiento ETL por lotes
3. Asignación espacial a municipios
4. Carga masiva con `bulk_write`
5. Creación de índices
6. Análisis exploratorio de datos
7. Documentación de anomalías

7.2.4. Fase 4: Análisis Espacial (11-17 Nov)

1. Desarrollo de pipelines de agregación
2. Cálculo de métricas por municipio
3. Comparación Microsoft vs Google
4. Generación de tablas de resultados
5. Creación de mapas interactivos
6. Validación de resultados

7.2.5. Fase 5: Reporte y Documentación (18-24 Nov)

1. Redacción de metodología
2. Análisis de resultados
3. Generación de visualizaciones finales
4. Recomendaciones para UPME
5. Revisión y correcciones
6. Preparación de presentación

8. Consideraciones de Escalabilidad

8.1. Gestión de Volumen de Datos

8.1.1. Estrategias de Optimización

- **Compresión:** MongoDB utiliza compresión Snappy/Zstd por defecto
- **Proyecciones:** Cargar solo campos necesarios en consultas
- **Índices selectivos:** Índices parciales solo para municipios PDET
- **Carga incremental:** Procesar datasets en batches de 10,000-100,000 documentos
- **Sharding:** Distribuir colecciones grandes en múltiples shards

8.2. Optimización de Consultas

- Uso de `explain()` para analizar planes de ejecución
- Índices cubrientes (covering indexes) cuando sea posible
- Agregaciones con `$match` temprano en el pipeline
- Límite de resultados con `$limit` cuando sea apropiado
- Uso de `allowDiskUse: true` para agregaciones grandes

8.3. Monitoreo y Rendimiento

- MongoDB Compass para monitoreo visual
- Métricas de `serverStatus` y `dbStats`
- Logs de queries lentas (slow query log)
- Profiler de MongoDB para análisis detallado

9. Validación del Esquema

9.1. JSON Schema Validation

MongoDB permite definir reglas de validación para garantizar la integridad de los datos:

```
db.createCollection("municipalities", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["codigo_dane", "nombre_municipio",
        "geometry", "is_pdet"],
      properties: {
        codigo_dane: {
          bsonType: "string",
          pattern: "[0-9]{5}$",
          description: "Codigo DANE de 5 digitos"
        },
        geometry: {
          bsonType: "object",
          required: ["type", "coordinates"],
          properties: {
            type: {
              enum: ["Polygon", "MultiPolygon"]
            }
          }
        },
        is_pdet: {
          bsonType: "bool"
        }
      }
    }
  }
})
```

9.2. Validación de Geometrías

- Verificación de formato GeoJSON válido
- Validación de topología (no auto-intersecciones)
- Verificación de coordenadas dentro de rangos válidos para Colombia
- Uso de Shapely para validación: `is_valid`, `is_simple`

10. Gestión de Riesgos

| Riesgo | Impacto | Mitigación |
|------------------------------|---|---|
| Tamaño excesivo de datasets | Problemas de almacenamiento y rendimiento | <ul style="list-style-type: none"> - Filtrar solo municipios PDET - Sharding de colecciones - Usar servidor con capacidad adecuada |
| Geometrías inválidas | Errores en consultas espaciales | <ul style="list-style-type: none"> - Validación con Shapely - Limpieza con buffer(0) - Logging de problemas |
| Baja calidad de datos | Resultados inexactos | <ul style="list-style-type: none"> - EDA exhaustivo - Comparación entre datasets - Documentar limitaciones |
| Tiempos de carga excesivos | Retraso en entregas | <ul style="list-style-type: none"> - Carga en paralelo - Bulk operations - Procesamiento incremental |
| Incompatibilidad de formatos | Problemas en integración | <ul style="list-style-type: none"> - Estandarización a GeoJSON - Scripts de transformación robustos - Pruebas tempranas |

11. Conclusiones

Este plan de implementación establece una base sólida para el desarrollo del proyecto de análisis geoespacial de potencial solar en territorios PDET. La selección de MongoDB como solución NoSQL se justifica por su:

- Soporte geoespacial nativo y robusto
- Capacidad de escalar a billones de documentos
- Flexibilidad para comparar datasets heterogéneos
- Ecosistema maduro de herramientas

El diseño del esquema propuesto optimiza el almacenamiento y consulta de datos geoespaciales masivos, mientras que el plan de implementación detallado proporciona una ruta clara hacia la entrega exitosa del proyecto.

Las siguientes entregas construirán sobre esta fundación, implementando progresivamente la carga de datos, análisis espacial y generación de reportes que culminarán en recomendaciones accionables para la UPME.

12. Referencias

1. MongoDB Documentation. *Geospatial Queries*. <https://docs.mongodb.com/manual/geospatial-queries/>
2. Microsoft Building Footprints. *Planetary Computer*. <https://planetarycomputer.microsoft.com/dataset/ms-buildings>
3. Google Open Buildings. *Research Dataset*. <https://sites.research.google/gr/open-buildings/>
4. DANE. *Marco Geoestadístico Nacional*. <https://geoportal.dane.gov.co/>
5. MongoDB, Inc. *Data Modeling Introduction*. <https://docs.mongodb.com/manual/core/data-modeling-introduction/>