Creación de Objetos de Base de Datos No Relacional del Proyecto

"Censo Rural"

Creación de colecciones y esquemas de seguridad en base de datos NoSQL

GA6-220501096-AA1-EV03

Isidro J Gallardo Navarro

Ficha:3070299

2025

Tecnología en Análisis y Desarrollo de Software.

ADSO

Introducción

El presente documento técnico desarrolla la creación de los objetos de base de datos para el proyecto "Censo Rural" utilizando un enfoque **no relacional (NoSQL)**, específicamente empleando MongoDB como motor de base de datos orientado a documentos. Esta arquitectura NoSQL ofrece ventajas significativas para el contexto rural del proyecto, incluyendo flexibilidad en la estructura de datos, escalabilidad horizontal y operación eficiente en entornos distribuidos con conectividad intermitente.

A diferencia del modelo relacional tradicional con tablas rígidas, el modelo NoSQL basado en documentos permite almacenar información en estructuras JSON flexibles (documentos BSON en MongoDB), facilitando la adaptación a diferentes tipos de censos sin modificar el esquema de base de datos. Esta característica es fundamental para un proyecto de censo rural donde los formularios pueden evolucionar y variar según las necesidades específicas de cada región.

El diseño implementa esquemas de seguridad robustos adaptados al paradigma NoSQL, incluyendo validación de esquemas, índices únicos, encriptación a nivel de documento y control de acceso basado en roles. La arquitectura propuesta cumple con los requisitos establecidos en la evidencia GA6-220501096-AA1-EV03, garantizando la integridad, seguridad y funcionalidad del sistema de recolección de datos personales en zonas rurales apartadas, todo ello bajo el marco legal de la Ley 1581 de 2012 de protección de datos personales en Colombia.

1. Fundamentos de Bases de Datos No Relacionales para Censo Rural

1.1 Ventajas del Modelo NoSQL para el Proyecto

El enfoque no relacional ofrece beneficios específicos para el contexto del censo rural que justifican su adopción frente al modelo relacional tradicional.

Flexibilidad de Esquema: Los documentos NoSQL no requieren una estructura rígida predefinida. En el censo rural, diferentes regiones pueden requerir campos adicionales específicos (por ejemplo, información sobre cultivos en zonas agrícolas o datos de pesca en zonas costeras) sin necesidad de modificar el esquema global de la base de datos. Esta flexibilidad permite que el sistema evolucione con los requerimientos cambiantes del censo.

Operación Offline Optimizada: MongoDB permite sincronización de réplicas y operación desconectada mediante Realm/Sync, ideal para encuestadores trabajando en zonas sin conectividad. Los datos capturados offline se sincronizan automáticamente cuando hay conexión disponible, resolviendo uno de los mayores desafíos del censo rural.

Escalabilidad Horizontal: A medida que el censo se expande a más regiones, el modelo NoSQL permite agregar servidores adicionales (sharding) distribuyendo la carga de forma transparente, sin las limitaciones de escalabilidad vertical típicas de bases de datos relacionales.

Rendimiento en Lecturas y Escrituras: Las operaciones de lectura/escritura en documentos completos son más rápidas que las operaciones con múltiples JOINs en bases relacionales, mejorando la experiencia del encuestador en campo al capturar y consultar datos.

1.2 Arquitectura de Datos Basada en Documentos

En el modelo no relacional, cada entidad del sistema se representa como una **colección** de **documentos JSON**. A diferencia de las filas en tablas relacionales, cada documento puede tener una estructura ligeramente diferente, permitiendo representar entidades complejas con subdocumentos anidados y arrays.

Ejemplo conceptual de documento:

```
{
  "_id": ObjectId("507f1f77bcf86cd799439011"),
  "numeroDocumento": "1234567890",
  "datosPersonales": {
    "nombres": ["Juan", "Carlos"],
    "apellidos": ["García", "López"]
  },
  "registrosCenso": [
    { "fecha": "2025-10-15", "encuestador": "María" }
  ]
}
```

Esta estructura permite:

- **Desnormalización estratégica:** Almacenar datos relacionados en un solo documento reduce la necesidad de operaciones JOIN
- **Atomicidad a nivel de documento:** Las actualizaciones de un documento completo son atómicas
- **Representación natural:** Los datos se modelan como objetos JSON, coincidiendo con la forma en que las aplicaciones modernas manejan datos

2. Definición de Esquemas de Seguridad e Integridad en NoSQL

Aunque las bases de datos NoSQL son "sin esquema" (schema-less), MongoDB y otros motores modernos permiten implementar **validación de esquemas** y restricciones de integridad para garantizar la calidad y seguridad de los datos.

2.1 Validación de Esquemas con JSON Schema

MongoDB permite definir reglas de validación usando JSON Schema que verifican la estructura y

tipos de datos de cada documento insertado o actualizado. Esta validación actúa como una capa de seguridad que previene datos inconsistentes.

Características de la validación:

- **Obligatoriedad de campos:** Especificar campos requeridos (required)
- **Tipos de datos:** Validar que cada campo tenga el tipo correcto (string, number, date, etc.)
- **Patrones y rangos:** Validar formatos (email, teléfono) y rangos numéricos
- **Enumeraciones:** Restringir valores a opciones predefinidas

2.2 Integridad Referencial en NoSQL

A diferencia del modelo relacional con claves foráneas automáticas, en NoSQL la integridad referencial se implementa mediante:

Referencias por ID: Almacenar ObjectId de documentos relacionados, similar a foreign keys pero sin validación automática por el motor de base de datos. La aplicación debe verificar la existencia de referencias.

Documentos Embebidos: Para relaciones uno-a-muchos con datos que siempre se consultan juntos, se pueden embeber subdocumentos, eliminando la necesidad de referencias externas.

Transacciones Multi-Documento: MongoDB 4.0+ soporta transacciones ACID que abarcan múltiples documentos y colecciones, permitiendo operaciones complejas con garantías de consistencia.

2.3 Unicidad de Datos

Los índices únicos en MongoDB garantizan que no existan valores duplicados en campos específicos, implementando la misma funcionalidad que UNIQUE constraints en bases relacionales.

Implementación:

Esta restricción es crítica para prevenir la duplicación de habitantes censados, asegurando la integridad de las estadísticas del censo.

2.4 Seguridad de Datos Personales

La protección de información sensible en NoSQL se implementa mediante múltiples capas:

Encriptación a Nivel de Campo: MongoDB Enterprise ofrece encriptación automática de campos sensibles (Field Level Encryption), permitiendo que datos como números de documento se almacenen encriptados sin modificar el código de la aplicación.

Encriptación en Reposo: Toda la base de datos puede encriptarse en disco usando encriptación transparente (Encryption at Rest), protegiendo contra acceso físico no autorizado a los archivos de datos.

Control de Acceso Basado en Roles: MongoDB implementa RBAC granular donde se definen roles con permisos específicos sobre colecciones y operaciones (lectura, escritura, actualización, eliminación).

Auditoría de Accesos: El sistema de auditoría registra todas las operaciones realizadas por usuarios, permitiendo trazabilidad completa para cumplimiento normativo.

3. Definición de Colecciones y Estructuras de Documentos

Las siguientes secciones definen las colecciones principales del sistema Censo Rural, incluyendo la estructura de documentos, validación de esquemas e índices de seguridad.

3.1 Colección: categorias

La colección categorias gestiona clasificaciones jerárquicas de datos por región, comunidad, vereda o grupo etario, permitiendo organización territorial y análisis desagregados.

Estructura de Documento

```
{
  "_id": ObjectId(" ... "),
  "nombreCategoria": "Antioquia - Municipio Envigado",
  "tipo": "municipio",
  "nivelJerarquico": 2,
  "codigoDANE": "05266",
  "jerarquia": {
    "departamento": "Antioquia",
    "municipio": "Envigado",
    "vereda": null
  },
  "ubicacionGeografica": {
    "type": "Point",
    "coordinates": [-75.5812, 6.1701]
  },
  "poblacionEstimada": 235000,
  "metadata": {
    "fechaCreacion": ISODate("2025-10-15T10:30:00Z"),
    "creadoPor": ObjectId(" ... "),
    "ultimaActualizacion": ISODate("2025-10-20T14:22:00Z")
  }
}
```

Validación de Esquema

```
db.createCollection("categorias", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["nombreCategoria", "tipo", "nivelJerarquico"],
      properties: {
        nombreCategoria: {
          bsonType: "string",
          minLength: 3,
          maxLength: 150,
          description: "Nombre descriptivo de la categoría (requerido)"
        },
        tipo: {
          enum: ["departamento", "municipio", "vereda", "grupoEtario", "zona"],
          description: "Tipo de categoría (debe ser uno de los valores permitidos)"
        },
        nivelJerarquico: {
          bsonType: "int",
          minimum: 1,
          maximum: 5,
          description: "Nivel en la jerarquía administrativa (1-5)"
        },
        codigoDANE: {
          bsonType: "string",
          pattern: "^[0-9]{5,8}$",
          description: "Código DANE oficial"
        },
        jerarquia: {
          bsonType: "object",
          properties: {
            departamento: { bsonType: "string" },
            municipio: { bsonType: ["string", "null"] },
            vereda: { bsonType: ["string", "null"] }
          }
        },
        ubicacionGeografica: {
          bsonType: "object",
          required: ["type", "coordinates"],
          properties: {
            type: { enum: ["Point"] },
            coordinates: {
              bsonType: "array",
              minItems: 2,
              maxItems: 2,
              items: { bsonType: "double" }
            }
          }
        },
        poblacionEstimada: {
          bsonType: "int",
          minimum: 0
        }
      }
    }
  },
  validationLevel: "strict",
  validationAction: "error"
})
```

Índices

```
// Índice único para código DANE
db.categorias.createIndex(
  { "codigoDANE": 1 },
  { unique: true, sparse: true, name: "idx_codigo_dane_unique" }
)

// Índice para búsquedas jerárquicas
db.categorias.createIndex(
  { "nivelJerarquico": 1, "jerarquia.departamento": 1, "jerarquia.municipio": 1 },
  { name: "idx_jerarquia" }
)

// Índice geoespacial para consultas de proximidad
db.categorias.createIndex(
  { "ubicacionGeografica": "2dsphere" },
  { name: "idx_ubicacion_geo" }
)
```

Justificación funcional: La estructura jerárquica con subdocumento jerarquia permite consultas

eficientes por nivel administrativo. El índice geoespacial facilita búsquedas de "categorías cercanas" para asignación inteligente de zonas a encuestadores.

3.2 Colección: usuarios

La colección usuarios centraliza la gestión de autenticación, autorización y perfil de todos los usuarios del sistema.

Estructura de Documento

Justificación de seguridad: El campo contrasenaHash almacena solo el hash bcrypt de la contraseña, nunca el texto plano. El subdocumento seguridad implementa protección contra ataques de fuerza bruta mediante contador de intentos fallidos y bloqueo temporal. La separación del perfilEncuestador permite extender roles sin contaminar la estructura base.

3.3 Colección: habitantesCensados

La colección habitantesCensados almacena los datos personales verificados de cada habitante de zonas rurales, siendo la colección más sensible del sistema.

Justificación de seguridad crítica: El índice único en numeroDocumento es la garantía fundamental contra duplicación de habitantes. El subdocumento seguridad indica qué campos están encriptados y el consentimiento informado. El array historicoRegistros mantiene trazabilidad sin necesidad de JOINs.

3.4 Colección: registrosCenso

La colección registrosCenso almacena cada instancia transaccional de captura de datos en campo, vinculando encuestador, habitante, formulario y metadatos.

Estructura de Documento

```
{
  "_id": ObjectId("..."),
  "idEncuestador": ObjectId("..."),
  "idHabitante": ObjectId("..."),
  "idFormulario": ObjectId("..."),
  "captura": {
    "fechaHoraCaptura": ISODate("2025-10-15T14:30:00Z"),
    "ubicacionGPS": {
      "type": "Point",
      "coordinates": [-75.6789, 6.2345],
      "altitud": 1450,
      "precision": 5
    },
```

Índices

```
db.createCollection("registrosCenso", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["idEncuestador", "idHabitante", "idFormulario", "captura", "sincronizacion"],
      properties: {
        idEncuestador: {
          bsonType: "objectId",
          description: "Referencia al encuestador que capturó los datos"
        },
        idHabitante: {
          bsonType: "objectId",
          description: "Referencia al habitante censado"
        },
        idFormulario: {
          bsonType: "objectId",
          description: "Referencia al formulario utilizado"
        },
        captura: {
          bsonType: "object",
          required: ["fechaHoraCaptura", "ubicacionGPS"],
          properties: {
            fechaHoraCaptura: {
              bsonType: "date",
              description: "Timestamp de captura de datos"
            },
            ubicacionGPS: {
              bsonType: "object",
              required: ["type", "coordinates"],
              properties: {
                type: { enum: ["Point"] },
                coordinates: {
                  bsonType: "array",
                  minItems: 2,
                  maxItems: 2
                },
                altitud: { bsonType: ["double", "int"] },
                precision: {
                  bsonType: ["double", "int"],
                  minimum: 0
                }
              }
            },
            duracionMinutos: {
              bsonType: "int",
              minimum: 1,
              maximum: 300
            }
          }
        },
        datosFormulario: {
          bsonType: "object",
          properties: {
            respuestas: {
              bsonType: "array",
              items: {
                bsonType: "object",
                required: ["idPregunta", "pregunta", "respuesta"],
                properties: {
                  idPregunta: { bsonType: "string" },
                  pregunta: { bsonType: "string" },
                  respuesta: {},
                  tipoDato: {
                    enum: ["string", "number", "boolean", "date", "array"]
                  }
                }
              }
            },
            puntajeCompletitud: {
              bsonType: ["int", "double"],
              minimum: 0,
              maximum: 100
            }
          }
        },
        sincronizacion: {
          bsonType: "object",
          required: ["estadoSincronizacion"],
          properties: {
            estadoSincronizacion: {
              enum: ["Offline", "Sincronizando", "Sincronizado", "Error"],
              description: "Estado de sincronización del registro"
            },
            fechaHoraCreacionOffline: { bsonType: "date" },
            fechaHoraSincronizacion: { bsonType: ["date", "null"] },
            intentosSincronizacion: {
              bsonType: "int",
              minimum: 0
            }
          }
        },
        archivosAdjuntos: {
          bsonType: "array",
          items: {
            bsonType: "object",
            required: ["idArchivo", "tipoArchivo", "nombreArchivo"],
            properties: {
              idArchivo: { bsonType: "objectId" },
              tipoArchivo: { bsonType: "string" },
              nombreArchivo: { bsonType: "string" },
              tamanoBytes: {
                bsonType: "int",
                minimum: 0
              }
            }
          }
        },
        validacion: {
          bsonType: "object",
          properties: {
            estadoValidacion: {
              enum: ["Pendiente", "En revisión", "Validado", "Rechazado"]
            },
            calificacionCalidad: {
              bsonType: ["double", "int"],
              minimum: 0,
              maximum: 10
            }
          }
        }
      }
    }
  },
  validationLevel: "strict",
  validationAction: "error"
})
```

Índice para consultas por encuestador

```
db.registrosCenso.createIndex(
  { "idEncuestador": 1, "captura.fechaHoraCaptura": -1 },
  { name: "idx_encuestador_fecha" }
)
```

Índice para historial de habitante

```
db.registrosCenso.createIndex(
  { "idHabitante": 1, "captura.fechaHoraCaptura": -1 },
  { name: "idx_habitante_historial" }
)
```

Índice para registros pendientes de sincronización

```
db.registrosCenso.createIndex(
  { "sincronizacion.estadoSincronizacion": 1, "captura.fechaHoraCaptura": 1 },
  { name: "idx_sincronizacion_pendiente" }
)
```

Indice geoespacial para ubicacion de captura

```
// Índice geoespacial para ubicación de captura
db.registrosCenso.createIndex(
  { "captura.ubicacionGPS": "2dsphere" },
  { name: "idx_ubicacion_captura" }
)
```

Indice para consulta por formulario

```
// Índice para consultas por formulario
db.registrosCenso.createIndex(
  { "idFormulario": 1 },
  { name: "idx_formulario" }
)
```

Indice para registros por validar

```
// Índice para registros por validar
db.registrosCenso.createIndex(
  { "validacion.estadoValidacion": 1, "captura.fechaHoraCaptura": -1 },
  { name: "idx_validacion_estado" }
)
```

Justificación funcional: El subdocumento sincronizacion es crítico para la operación offline, permitiendo identificar registros pendientes de envío al servidor. El array archivosAdjuntos mantiene referencias a archivos sin necesidad de una colección separada. La estructura

datosFormulario.respuestas permite flexibilidad total en las preguntas del censo.

3.5 Colección: archivos

La colección archivos gestiona el almacenamiento de fotografías, documentos escaneados y otros archivos binarios adjuntos a los registros de censo.

Justificación de seguridad crítica: El campo esEncriptado marca archivos que contienen información sensible. El hashIntegridad permite verificar que el archivo no ha sido modificado o corrompido. El subdocumento procesamientoIA habilita capacidades avanzadas como OCR y validación automática de documentos.

3.6 Colección: formulariosCenso

La colección formulariosCenso define las plantillas de formularios dinámicos utilizados para capturar datos en campo.

Índices

```
db.formulariosCenso.createIndex(
  { "codigoFormulario": 1 },
  { unique: true, name: "idx_codigo_formulario_unique" }
)
```

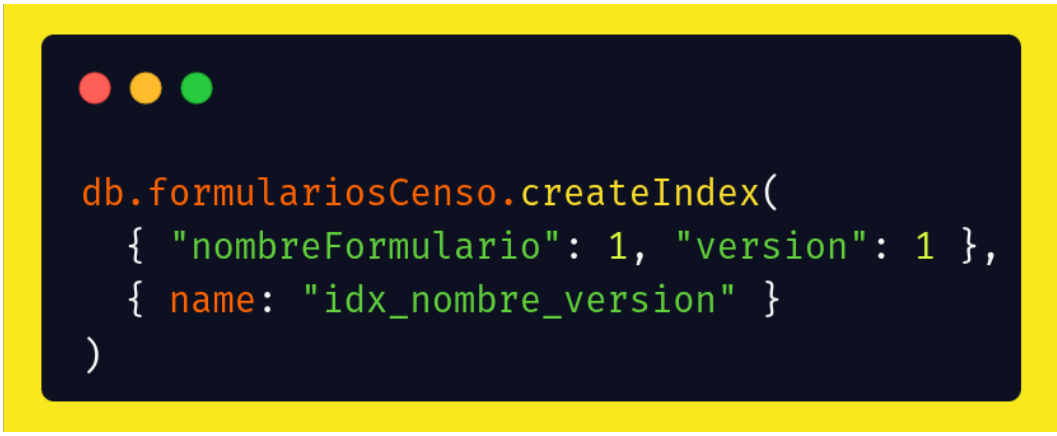
Índice único para código de formulario

Índice para formularios activos

```
db.formulariosCenso.createIndex(
  { "configuracion.activo": 1, "version": -1 },
  { name: "idx_activo_version" }
)
```

Índice compuesto para búsquedas por nombre y versión

4.



```
db.formulariosCenso.createIndex(
  { "nombreFormulario": 1, "version": 1 },
  { name: "idx_nombre_version" }
)
```

Implementación de Seguridad Avanzada en MongoDB

4.1 Encriptación a Nivel de Campo (Field Level Encryption)

MongoDB Enterprise ofrece encriptación automática de campos sensibles sin modificar código de aplicación:

5. Operaciones CRUD y Consultas Comunes

5.1 Inserción de Documentos

// Insertar habitante censado

```
db.habitantesCensados.insertOne({
  identificacion: {
    tipoDocumento: "CC",
    numeroDocumento: "1234567890"
  },
  datosPersonales: {
    nombres: { primero: "Juan", segundo: "Carlos" },
    apellidos: { primero: "García", segundo: "López" },
    nombreCompleto: "Juan Carlos García López",
    fechaNacimiento: new Date("1985-05-15"),
    edad: 40,
    sexo: "M"
  },
  ubicacion: {
    domicilio: {
      direccion: "Vereda El Carmen",
      zona: "Rural"
    },
    jerarquiaAdministrativa: {
      departamento: "Antioquia",
      municipio: "Envigado",
      vereda: "El Carmen"
    }
  }
})
```

5.2 Consultas Avanzadas

Buscar habitantes por rango de edad en una zona específica

```
db.habitantesCensados.find({
  "datosPersonales.edad": { $gte: 18, $lte: 65 },
  "ubicacion.jerarquiaAdministrativa.departamento": "Antioquia",
  "ubicacion.domicilio.zona": "Rural"
}).sort({ "datosPersonales.edad": 1 })
```

Agregar estadísticas por municipio

```
db.habitantesCensados.aggregate([
  {
    $group: {
      _id: "$ubicacion.jerarquiaAdministrativa.municipio",
      totalHabitantes: { $sum: 1 },
      edadPromedio: { $avg: "$datosPersonales.edad" },
      distribucionSexo: {
        $push: "$datosPersonales.sexo"
      }
    }
  },
  { $sort: { totalHabitantes: -1 } }
])

// Consultar registros pendientes de sincronización
db.registrosCenso.find({
  "sincronizacion.estadoSincronizacion": "Offline",
  "captura.fechaHoraCaptura": {
    $gte: new Date("2025-10-01"),
    $lt: new Date("2025-11-01")
  }
}).sort({ "captura.fechaHoraCaptura": -1 })
```

6. Cumplimiento de Criterios de Evaluación

Tabla 1. Matriz de Cumplimiento (GA6-220501096-AA1-EV03)

Criterio de Evaluación	Cumplimiento en Modelo NoSQL	Evidencia
Crea la base de datos integrando objetos según funcionalidad del software	Se definieron 6 colecciones (categorias, usuarios, habitantesCensados, registrosCenso, archivos, formulariosCenso) con estructuras de documentos JSON flexibles que cubren gestión de usuarios, datos censados, captura en campo y almacenamiento seguro	Secciones 3.1 a 3.6
Crea sentencias para creación de colecciones	Se proporcionaron comandos completos `db.createCollection()` con validación JSON Schema para todas las colecciones, definiendo estructura, tipos de datos y restricciones	Secciones 3.1 a 3.6
Define esquemas de seguridad para integridad de información	Implementación de: (1) Validación JSON Schema obligatoria, (2) Índices únicos en campos críticos, (3) Encriptación a nivel de campo, (4) RBAC con roles diferenciados, (5) Auditoría de operaciones, (6) Hash de integridad en archivos	Secciones 2, 4.1, 4.2, 4.3
Implementa mecanismos de sincronización offline	El campo `sincronizacion` en registrosCenso con estados (Offline, Sincronizando, Sincronizado, Error) permite operación desconectada esencial para zonas sin conectividad	Sección 3.4
Protege datos personales sensibles	Múltiples capas: encriptación automática de campos (FLE), hash bcrypt de contraseñas, indicador `esEncriptado` en archivos, consentimiento informado, control de acceso granular	Secciones 2.4, 3.2, 3.3, 3.5

Conclusiones

El presente documento ha desarrollado exhaustivamente un modelo de base de datos no relacional para el proyecto "Censo Rural" utilizando MongoDB como motor NoSQL orientado a documentos. El diseño implementado cumple integralmente con los requisitos establecidos en la evidencia GA6-220501096-AA1-EV03, adaptando los conceptos de integridad, unicidad y seguridad al paradigma no relacional.

Las principales ventajas del modelo NoSQL implementado incluyen flexibilidad de esquema para adaptarse a diferentes tipos de censos regionales, operación offline optimizada mediante sincronización de réplicas, escalabilidad horizontal para crecimiento del proyecto, y rendimiento superior en operaciones de lectura y escritura de documentos completos. La validación JSON Schema proporciona garantías de integridad de datos equiparables a las restricciones de bases relacionales, mientras mantiene la flexibilidad característica de NoSQL.

Los esquemas de seguridad implementados satisfacen los requerimientos de protección de datos personales sensibles mediante encriptación a nivel de campo, control de acceso basado en roles, auditoría completa de operaciones y hash de integridad para archivos adjuntos. La arquitectura de documentos embebidos y referencias por ObjectId proporciona un balance óptimo entre desnormalización estratégica y mantenimiento de relaciones lógicas entre entidades.

El modelo presentado constituye una base sólida y escalable para el desarrollo del sistema "Censo Rural", ofreciendo las capacidades técnicas necesarias para gestionar eficientemente la recolección, almacenamiento y análisis de datos de población en zonas rurales apartadas de Colombia, cumpliendo con los estándares de calidad, seguridad y normativa de protección de datos vigente.

Referencias

American Psychological Association. (2020). Publication manual of the American Psychological Association (7th ed.). https://doi.org/10.1037/0000165-000

Banker, K., Garrett, D., Bakkum, P., & Verch, S. (2016). MongoDB in action (2nd ed.). Manning Publications.

Bradshaw, S., Brazil, E., & Chodorow, K. (2019). MongoDB: The definitive guide (3rd ed.). O'Reilly Media.

Moniruzzaman, A. B. M., & Hossain, S. A. (2013). NoSQL database: New era of databases for big data analytics. International Journal of Database Theory and Application, 6(4), 1-14.

MongoDB, Inc. (2024). MongoDB manual: Security. https://docs.mongodb.com/manual/security/

República de Colombia. (2012). Ley 1581 de 2012: Régimen general de protección de datos personales. Diario Oficial No. 48.587.

Sadalage, P. J., & Fowler, M. (2012). NoSQL distilled: A brief guide to the emerging world of polyglot persistence. Addison-Wesley Professional.