HCL (BIG DATA)

CASO PRACTICO I UD5

ALUMNO CESUR

25/26

Alejandro Muñoz de la Sierra

PROFESOR

Óscar González Núñez

I N T R O D U C C I O N

Este proyecto nace de una realidad del sector tecnológico: el gran volumen de datos de internet. Diseñaremos la arquitectura de "FARO", un motor de búsqueda propio. No competiremos con Google de inmediato. Queremos demostrar nuestra capacidad para gestionar la ingestión y consulta de millones de páginas web con eficiencia.

El equipo estableció una premisa clara durante el análisis inicial. El modelo relacional clásico no funciona para este fin. Un buscador rastrea millones de páginas diarias. Cada una tiene una estructura distinta, como blogs, foros o tiendas. Forzar la web en un esquema rígido de tablas con Oracle o MySQL causaría fallos graves. Generaría cuellos de botella y dificultaría el mantenimiento.

Basamos la arquitectura de FARO en MongoDB. Es una base de datos NoSQL orientada a documentos. Su sistema sin esquema fijo usa documentos BSON. Esto permite guardar cada página web con su estructura natural. Los datos crecen de forma exponencial. MongoDB ofrece una escalabilidad horizontal nativa superior a las bases SQL por el mismo coste.

0 1
A N Á L I S I S D E N E C E S I D A D E S Y M O D E L A D O D E D A T O S

Aquí detallo las fases de diseño, modelado e implementación de este clúster.

El éxito de una base NoSQL no depende de crear colecciones sin orden e insertar archivos JSON. Aplicamos una regla estricta. Diseñamos el modelo según la lectura de los datos. En SQL normalizamos para evitar repeticiones. En NoSQL desnormalizamos para priorizar la velocidad de lectura.

1.1. Identificación Exhaustiva de Necesidades

Variedad Estructural (Polimorfismo): Guardamos artículos de prensa, productos de tiendas y vídeos. Estos tienen datos distintos como autor, fecha, precio o stock. MongoDB permite documentos con esquemas diferentes en una misma colección sin errores.

Velocidad de Respuesta Crítica: El usuario espera resultados en menos de 200 milisegundos. Evitaremos operaciones complejas de cruce de datos durante la consulta. Estas operaciones usan `$lookup` en MongoDB.

Datos Enriquecidos: Almacenaremos el texto de la web y sus métricas técnicas. Esto incluye velocidad de carga y optimización móvil. Usaremos estos datos para influir en la clasificación de resultados.

1.2. Diseño de Colecciones y Patrones de Modelado

Estructuramos la base de datos principal en dos grandes colecciones. Aplicamos patrones de diseño NoSQL distintos según la necesidad:

A) Colección Principal: web_pages (Patrón de Documento Embebido)

No usamos tablas separadas para URL, SEO y métricas como en SQL. Incrustamos todo en un único documento. El buscador lee un solo bloque del disco duro para obtener la información. El proceso es muy rápido.

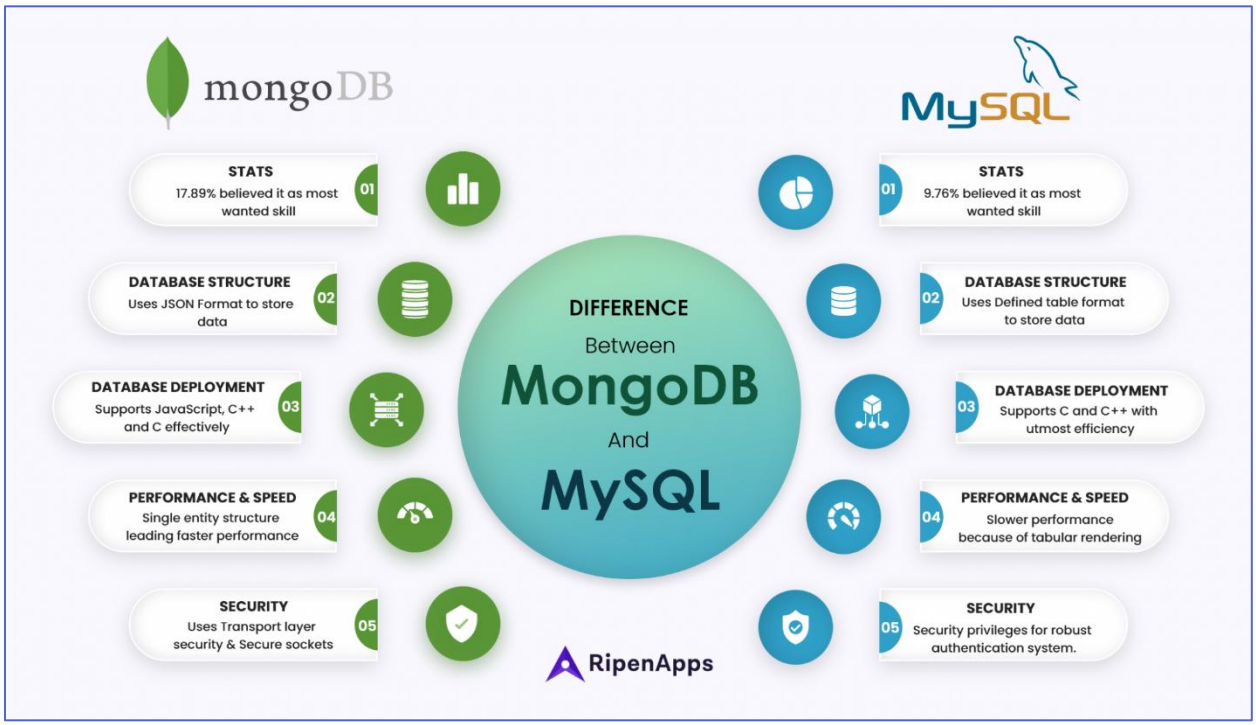
BIGDATAUD5_1.json 6, U

C: > Users > sibko > Documents > GitHub > Cesur-DAM > Asignaturas 2 > HCL (Big Data) > UD5 > Caso_Practico_1 > {} BIGDATAUD5_1.j

```
// Ejemplo del modelo de documento en la colección 'web_pages'
{
    "_id": ObjectId("652b4c9a3f..."),
    "url": "https://www.ejemplo.es/tecnologia/5g",
    "dominio": "ejemplo.es",
    "hash_url": "a1b2c3d4e5f6...", // Índice único para evitar indexar dos veces la misma web
    "contenido": {
        "titulo": "El avance del 5G en zonas rurales",
        "cuerpo_texto": "El despliegue de la red móvil avanza a buen ritmo...",
        "idioma": "es-ES",
        "tiene_multimedia": true
    },
    "metadatos_seo": {
        "keywords": [
            "5G",
            "tecnología",
            "redes",
            "rural"
        ],
        "fecha_ultimo_rastreo": ISODate("2026-02-21T09:00:00Z"),
    },
    "metricas_tecnicas": {
        "latencia_media_ms": 12,
        "optimizado_movil_amp": true
    },
    "autoridad": {
        "page_rank_score": 8.7, // Este dato lo pre-calculamos por la noche
        "trust_score": 9.1
    },
    "enlaces_salientes": [
        "https://www.otraweb.es/info",
        "https://www.gob.es/telecomunicaciones"
    ]
}
```

B) Colección Secundaria: search_analytics (Patrón de Referencia)

Guardamos los **registros estadísticos** de búsqueda en otra colección. Esto evita que los documentos de las webs crezcan demasiado. Aquí usamos referencias y guardamos el _id de la web.
Un usuario busca "**5G**" y hace clic en la web de ejemplo. Guardamos ese evento en search_analytics para alimentar nuestro algoritmo. Esta colección acepta escrituras masivas constantes. No bloquea las lecturas de las búsquedas.



0 2
I M P L E M E N T A C I Ó N T É C N I C A D E L B U S C A D O R

Un buscador es más que un almacén digital. Es un motor de recuperación de información. Explicaré la resolución de las operaciones críticas en MongoDB.

2.1. El Motor de Búsqueda Textual (Full-Text Search)

El mayor reto era buscar palabras concretas dentro de millones de párrafos. Un "**find**" normal con expresión regular recorre la base de datos documento por documento. Esto se llama **Collection Scan**. En Big Data, este proceso falla y tumba el servidor.

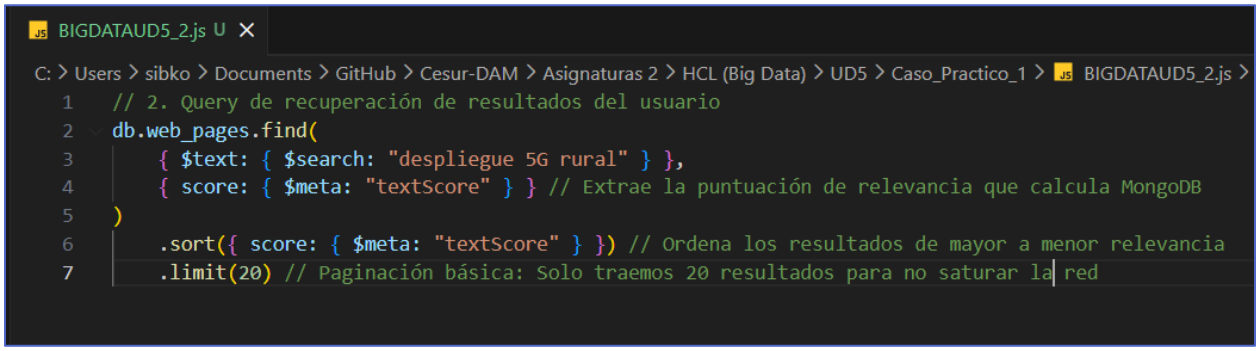
Implementaremos **Índices de Texto ($text)** como solución. MongoDB crea una estructura de **árbol (B-Tree)** interna. Esta mapea cada palabra (**token**) a los documentos que la contienen.
Aplicamos pesos (**weights**). La palabra "**5G**" tiene más valor en el título principal de la web. Vale menos en la letra pequeña del pie de página.

BIGDATAUD5_1.js U

C: > Users > sibko > Documents > GitHub > Cesur-DAM > Asignaturas 2 > HCL (Big Data) > UD5 > Caso_Practico_1 > BIGDATAUD5_1.js > ...

```
// 1. Script de creación del índice (Setup inicial)
db.web_pages.createIndex(
  {
    "contenido.titulo": "text",
    "contenido.cuerpo_texto": "text",
    "metadatos_seo.keywords": "text",
    "dominio": "text"
  },
  {
    weights: {
      "contenido.titulo": 15,       // Prioridad máxima (si está en el título, puntúa más)
      "metadatos_seo.keywords": 10, // Prioridad alta
      "dominio": 5,                 // Prioridad media
      "contenido.cuerpo_texto": 1   // Prioridad normal (texto genérico)
    },
    default_language: "spanish" // Vital: aplica stemming (raíces) y elimina stopwords (el, la, los)
  }
)
```

Un usuario busca "**despliegue 5G rural**". El backend ejecuta esta consulta eficiente:



2.2. Algoritmo de Ranking y el Aggregation Framework

La búsqueda simple de palabras favorece a las webs con spam. Estas repiten el mismo término muchas veces. Debemos calcular la "autoridad" de cada web según los enlaces recibidos. Esto sigue el concepto clásico del PageRank.

MongoDB no es una base de datos orientada a grafos. Pero tiene una herramienta potente para **cálculos masivos:** el Aggregation Framework. Es la evolución de MapReduce.
Diseñamos un proceso por lotes (**Batch Processing**). MongoDB ejecuta un Pipeline de agregación cada madrugada con poco tráfico. El proceso realiza lo siguiente:

$**unwind**: Desglosa los arrays de enlaces_salientes.

$**group**: Agrupa las webs por dominio y cuenta los enlaces entrantes de sitios de alta autoridad.

$**merge**: Actualiza la colección original y sobrescribe el campo autoridad.page_rank_score.

Realizamos los cálculos complejos de noche. Las lecturas de los usuarios son instantáneas durante el día.

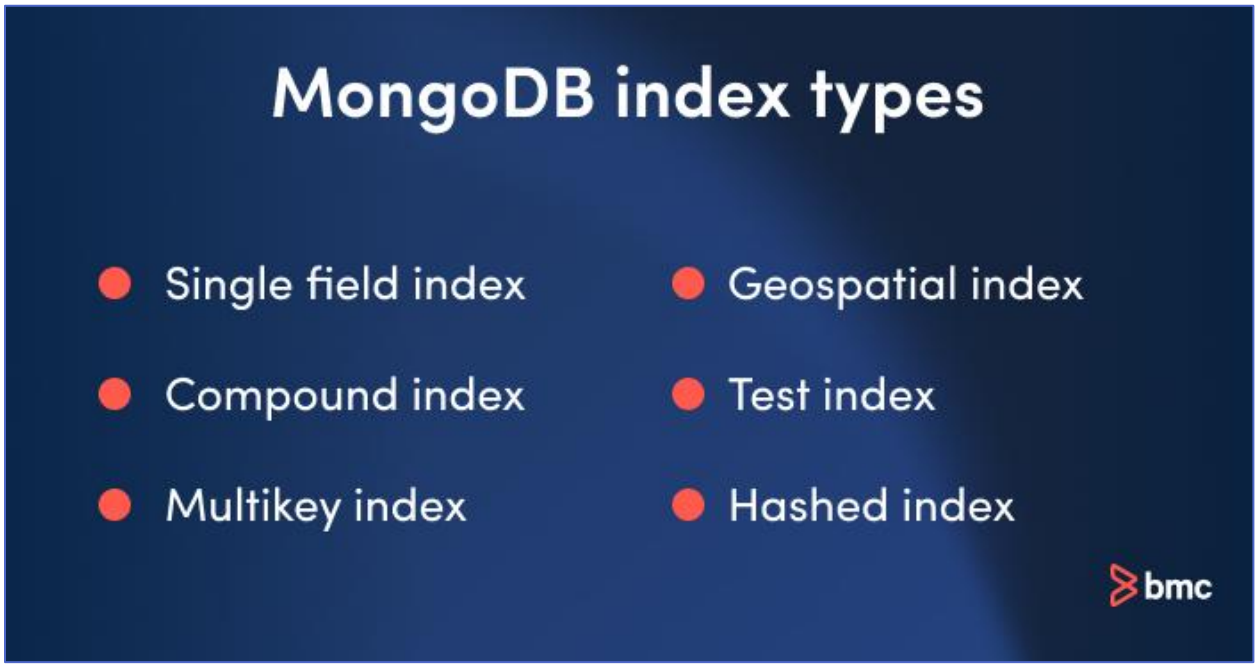
2.3. Hiper-Localización (Búsqueda Geoespacial)

Queremos aportar más valor. El usuario busca "restaurante". No tiene sentido mostrarle uno que está a 500 km. **MongoDB** permite **añadir índices geoespaciales** (**2dsphere**) a los documentos. Conocemos las coordenadas aproximadas del usuario. El backend puede añadir un filtro espacial a la búsqueda:

BIGDATAUD5_3.js U

C: > Users > sibko > Documents > GitHub > Cesur-DAM > Asignaturas 2 > HCL (Big Data) > UD5 > Caso_Practico_1 > BIGDATAUD5_3.js > ...

```
db.web_pages.find({
    "contenido.cuerpo_texto": /restaurante/i,
    "localizacion_negocio": {
        $near: {
            $geometry: { type: "Point", coordinates: [-3.703790, 40.416775] }, // Coordenadas del usuario
            $maxDistance: 2000 // Busca solo en un radio de 2 kilómetros
        }
    }
})
```



0 3
A R Q U I T E C T U R A T É C N I C A Y E S C A L A B I L I D A D

FARO funcionará a gran escala. La base de datos manejará **Terabytes o Petabytes** de información. Un solo servidor se saturará aunque tenga mucha RAM (escalabilidad vertical). Debemos usar la **Escalabilidad Horizontal** y mantener la Alta Disponibilidad.

3.1. Sharding (Fragmentación de la Base de Datos)

Usaremos un **clúster de Sharding.** Tomaremos el total de las páginas web guardadas. Las repartiremos físicamente entre varios servidores distintos.
La arquitectura de Sharding en MongoDB planteada tiene tres piezas:

Mongos (Routers): Son los servidores de "puerta de entrada". Reciben la búsqueda del usuario. No guardan datos y solo dirigen el tráfico hacia el servidor correcto.

Config Servers: Servidores pequeños que guardan el "mapa" de ubicación de cada fragmento de datos.

Shards (Nodos de Datos): Los servidores donde residen los documentos (ej. Servidor 1, Servidor 2...).

La decisión crítica: La Clave de Fragmentación **(Shard Key)**
Una mala elección de la Shard Key arruina el rendimiento del clúster. Si fragmentamos por "fecha", todas las webs rastreadas hoy irían al mismo servidor físico. Esto lo saturaría y crearía Jumbo Chunks.
Usaremos **Hashed Sharding** sobre el campo dominio. Aplicaremos una función hash criptográfica al nombre del dominio. Los documentos se repartirán de forma uniforme y aleatoria entre todos los servidores. Así equilibramos la carga de trabajo.

3.2. Alta Disponibilidad (Replica Sets) y Teorema CAP

Un buscador no puede "caerse". Por cada Shard físico, montaremos un **Replica Set (un grupo de replicación).** Este tendrá 3 nodos distribuidos en distintos Centros de Datos:

Nodo Primario: Es el jefe y recibe todas las operaciones de escritura (cuando guardamos webs nuevas).

Nodos Secundarios (x2): Mantienen una copia exacta de los datos del Primario en tiempo real.

El servidor Primario puede quemarse o la red puede caer. Los nodos Secundarios lo notan enseguida mediante **pulsos internos (heartbeats).** Celebran una "elección" automática en un par de segundos. Uno de ellos se convierte en el nuevo Primario. El sistema se **repara solo** y sin nuestra intervención.

Consideración del **Teorema CAP:**
El Teorema CAP aplica a bases de datos distribuidas. Solo puedes elegir dos opciones entre Consistencia (C), Disponibilidad (A) y Tolerancia a Particiones (P). MongoDB prioriza 'CP' por defecto (Consistencia estricta). Pero para un buscador configuraremos la prioridad en AP (Disponibilidad). Configuraremos las llamadas (Read Preferences) para leer de los nodos secundarios. Preferimos mostrar resultados con unos minutos de desfase (consistencia eventual). Esto es mejor que mostrar una pantalla de error por servicio no disponible.

0 4

S E G U R I D A D Y G O B E R N A N Z A D E L D A T O

Seguimos las normativas de protección de datos (RGPD). La seguridad es obligatoria. La implementación en MongoDB incluirá:

Cifrado en Reposo (Encryption at Rest): Usaremos el motor de almacenamiento WiredTiger de MongoDB. Lo configuraremos con cifrado AES-256 a nivel de disco. Alguien puede robar un disco duro físico del servidor. Los datos serán ilegibles sin la llave. Cifrado en Tránsito: Usaremos certificados TLS/SSL para proteger todas las comunicaciones internas entre los routers mongos, los shards y las aplicaciones cliente.

Control de Acceso (RBAC): Crearemos usuarios de base de datos con roles estrictos. Los bots de rastreo solo tendrán permisos de escritura. La API del buscador mantendrá permisos exclusivos de lectura. Bloquearemos comandos destructivos como dropDatabase. Así, ni siquiera un administrador podrá borrar el clúster por error.

0 5
C O N C L U S I O N E S

Diseñé la arquitectura del buscador FARO sobre MongoDB. Este trabajo confirmó que las tecnologías NoSQL transformaron la ingeniería de datos masiva.

Este informe explica cómo la flexibilidad del formato BSON evita los fallos típicos de las bases de datos relacionales ante la variedad de la web. Trazamos una guía técnica clara. Usamos índices $text junto con búsquedas geoespaciales. También diseñamos pipelines de agregación para calcular la relevancia de los sitios por la noche.

El diseño distribuido usa Sharding y Replica Sets. El sistema soportará la información actual. Podrá escalar horizontalmente en el futuro con más servidores económicos. Esta estructura mantiene la alta disponibilidad. La arquitectura está lista para el despliegue.

0 6
R E F E R E N C I A S

https://www.mongodb.com/es/docs/manual/installation/

https://www.mongodb.com/es/docs/manual/text-search/

https://studio3t.com/knowledge-base/articles/mongodb-aggregation-framework/

https://www.mongodb.com/es/docs/manual/geospatial-queries/

https://www.mongodb.com/es/docs/manual/sharding/

https://www.mongodb.com/es/docs/manual/core/replica-set-elections/

MongoDB Curso, Introducción Practica a NoSQL
https://www.youtube.com/watch?v=lWMemPN9t6Q

MongoDB in 5 Minutes with Eliot Horowitz
https://www.youtube.com/watch?v=EE8ZTQxa0AM

How Google searches one document among Billions of documents quickly?
https://www.youtube.com/watch?v=CeGtqouT8eA

Aggregation | MongoDB | Tutorial 10
https://www.youtube.com/watch?v=Kk6Er0c7srU