Mini Projecto Mongo

Juan Diego Collazos

May 2025

Lista de colecciones y justificación de atributos

Se propone el siguiente diseño de base de datos en MongoDB para la plataforma de intercambio de libros. El diseño busca garantizar eficiencia en consultas, flexibilidad para futuras ampliaciones y escalabilidad horizontal mediante sharding.

Colección: `users`

- **_id**: Identificador único del usuario (generado automáticamente por MongoDB).
- **username**: Nombre de usuario único utilizado para autenticación e identificación.
- **name**: Nombre real del usuario (opcional pero útil para una mejor experiencia de usuario).
- **password_hash**: Hash seguro de la contraseña (nunca se almacena la contraseña en texto plano).
- **email**: Correo electrónico de contacto.
- **registration_date**: Fecha de registro del usuario en la plataforma.

Colección: `books` (opcional, si se desea normalizar los libros)

- **_id**: Identificador único del libro.
- **title**: Título del libro.
- **author**: Autor del libro.
- **genre**: Género literario.
- **tags**: Arreglo de etiquetas descriptivas que ayudan en las búsquedas.

Nota: En este miniproyecto, los datos del libro se embeben directamente en los documentos de la colección `trades`, por lo que esta colección puede ser opcional.

Colección: `trades`

Esta colección almacena las propuestas de intercambio de libros hechas por los usuarios. A continuación se presenta el esquema de un documento típico de la colección, junto con comentarios sobre la función de cada campo y la eficiencia del diseño:

```
{
  _id: ObjectId,
  proposed_by: string,
  created_at: datetime,
  status: string,  // 'open' o 'negotiated'
  book: {
    title: string,
    author: string,
    genre: string,
    quality: string,
    tags: [string]  // etiquetas que describen el libro
  },
  negotiation: {    // Solo presente si status = 'negotiated'
    with_user: string,
    exchanged_for: {
      title: string,
      author: string,
      genre: string
    },
    date: datetime
  }
}
```

Justificación del diseño:

- `proposed-by`: indica el usuario que propone el intercambio. Es clave para la eficiencia, ya que se usa como **clave de sharding**, permitiendo distribuir los documentos según el usuario y realizar consultas eficientes sobre sus intercambios.
- `created-at`: permite ordenar las propuestas cronológicamente. Es útil para listar propuestas en orden de antigüedad (por ejemplo, las más viejas primero).
- `status`: indica si el intercambio está abierto o ya fue negociado. Este campo permite filtrar eficientemente las propuestas activas (`open`).
- `book`: es un **documento anidado** que contiene los datos del libro. Esto agrupa toda la información relevante del libro en un solo campo, lo que facilita el acceso y evita hacer *joins* o referencias externas.

- `tags`: es un **arreglo** de etiquetas, útil para búsquedas por categorías o filtros múltiples en futuras versiones de la aplicación.
- `negotiation`: es un **documento opcional** que aparece solo cuando el estado es `negotiated`. Permite almacenar los detalles del intercambio sin necesidad de crear una colección separada.

Claves de eficiencia:

- Se elige `proposed-by` como **clave de sharding**, ya que muchas consultas están centradas en las propuestas de un usuario específico. Esto permite distribuir los documentos de forma equilibrada en el clúster y escalar horizontalmente.
- El diseño es **auto-contenido**, ya que toda la información de una propuesta (incluido el libro y la negociación) está dentro del mismo documento. Esto permite consultas rápidas y sin operaciones de agregación complejas.
- El uso de **documentos anidados** y **arreglos** permite representar estructuras ricas y flexibles que crecen con la aplicación, alineándose con las fortalezas de MongoDB.

Colección `complete_trades`

La colección `complete_trades` registra los intercambios que han sido exitosamente negociados entre usuarios. Cada documento representa una transacción permanente y contiene la información clave de la propuesta original, el usuario con quien se realizó el intercambio y los libros intercambiados. A continuación, se muestra un esquema típico de documento:

```
{
  "_id": ObjectId,
  "proposed_by": "juan123",
  "with_user": "laura456",
  "date": ISODate("2025-05-13T16:45:00Z"),
  "proposed_book": {
    "title": "Cien años de soledad",
    "author": "Gabriel García Márquez",
    "genre": "Realismo mágico"
  },
  "exchanged_for": {
    "title": "1984",
    "author": "George Orwell",
    "genre": "Distopía"
  }
}
```

Justificación y eficiencia: Separar los intercambios finalizados en una colección distinta ofrece múltiples beneficios desde el punto de vista del modelado de datos y del rendimiento:

- **Eficiencia en consultas comunes:** Al mantener los intercambios abiertos en la colección `trades` y mover los finalizados a `complete_trades`, las consultas que muestran al usuario las propuestas activas (que son las más frecuentes en la plataforma) se ejecutan más rápido, ya que el conjunto de datos es más pequeño y relevante.
- **Historial desacoplado:** Permite consultar y analizar el historial de intercambios sin interferir con los datos actuales de la aplicación. Esto facilita realizar agregaciones, estadísticas o auditorías sin afectar el rendimiento del sistema en tiempo real.
- **Escalabilidad lógica:** En un entorno distribuido, tener colecciones separadas para operaciones activas y finalizadas facilita el diseño de particiones (sharding) basadas en el estado de la transacción y permite balancear mejor la carga de lectura y escritura.
- **Diseño orientado a operaciones CRUD específicas:** Las operaciones de lectura sobre `complete_trades` son mayoritariamente de tipo `read-only`, lo que permite optimizaciones de almacenamiento e índices específicos para análisis históricos.