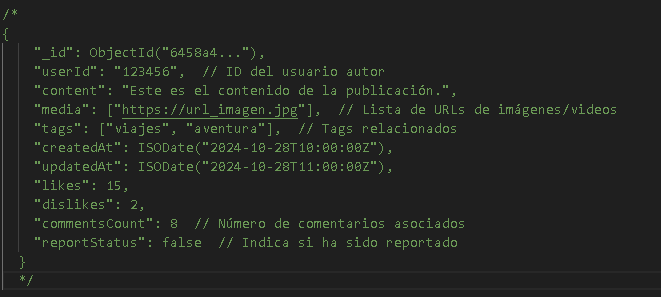
**1. Esquema General**

Tendremos dos colecciones principales:

* **Posts** (publicaciones)
* **Comments** (comentarios)

**Relación entre Publicaciones y Comentarios**

* **1 Publicación → N Comentarios.**  
  Cada comentario estará relacionado con una publicación a través de un **ID de referencia**. Podríamos optar por **anidar comentarios dentro de la publicación** si el volumen es bajo, pero para evitar problemas de rendimiento con grandes cantidades, usaremos **colecciones separadas**.
* **2. Esquemas de las Colecciones**
* **2.1. Colección posts (Publicaciones)**
* Cada documento en esta colección representa una **publicación**. Incluye información relevante del autor y una lista de reacciones o interacciones básicas.



**Explicación:**

* **userId**: Relaciona la publicación con el microservicio de **gestión de usuarios**.
* **media**: Almacena URLs para archivos multimedia.
* **likes y dislikes**: Reacciones simples para cada publicación.
* **commentsCount**: Campo para evitar consultas adicionales al contar comentarios.
* **reportStatus**: Marca si la publicación ha sido reportada (enlazado al microservicio de **reportes**).
* **2.2. Colección comments (Comentarios)**
* Esta colección almacena los comentarios realizados en las publicaciones. Cada comentario se relaciona con una publicación mediante el **postId**.

Texto

Descripción generada automáticamente

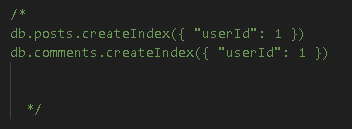
**Explicación:**

* **postId**: Relaciona el comentario con la publicación correspondiente.
* **replies**: Estructura de **respuestas anidadas** para comentarios.
* **likes y dislikes**: Reacciones individuales por comentario o respuesta.

**3. Índices Recomendados**

Para optimizar las consultas frecuentes, es importante definir índices en los campos que se usan como filtros:

* **Índice en el campo userId** en **posts** y **comments**:  
  Facilita las consultas por publicaciones/comentarios de un usuario específico.



**Índice compuesto en postId y createdAt** en **comments**:  
Optimiza la búsqueda de comentarios ordenados cronológicamente por publicación.

Texto

Descripción generada automáticamente

**Índice en el campo reportStatus** en **posts**:  
Facilita las consultas de publicaciones reportadas.

Texto

Descripción generada automáticamente

**6. Gestión de Reportes e Integración con Otros Microservicios**

* Cuando se reporta una publicación, se enviará un mensaje al microservicio de **reportes** usando **RabbitMQ**.
* Cuando se crea un comentario o publicación, se notificará al microservicio de **notificaciones y mensajería** para enviar alertas a los usuarios.

**RabbitMQ** es un **sistema de mensajería** basado en el protocolo **AMQP (Advanced Message Queuing Protocol)**. En pocas palabras, se encarga de **enviar, recibir y gestionar mensajes** entre distintos sistemas o microservicios de forma eficiente y segura. Es ideal en arquitecturas de **microservicios** porque permite la **comunicación asíncrona**, asegurando que los sistemas puedan interactuar sin depender de que el otro esté disponible en tiempo real.

**¿Por qué usar RabbitMQ?**

* **Desacoplamiento**: Los microservicios pueden funcionar de forma independiente; solo envían o reciben mensajes.
* **Escalabilidad**: Permite gestionar grandes volúmenes de mensajes.
* **Fiabilidad**: Garantiza que los mensajes no se pierdan, incluso si uno de los servicios está temporalmente fuera de línea.
* **Colas de mensajes**: Los mensajes se almacenan en **colas** hasta que un consumidor esté disponible para procesarlos.

**Conceptos Clave de RabbitMQ**

1. **Producer (Productor)**:  
   El servicio que **envía mensajes** a RabbitMQ. Por ejemplo, tu microservicio de **publicaciones** podría notificar a RabbitMQ cuando se cree una publicación.
2. **Queue (Cola)**:  
   Es un **almacén temporal** donde los mensajes esperan a ser procesados por un consumidor. Cada cola tiene un nombre y puede configurarse para que persista en disco si es necesario.
3. **Consumer (Consumidor)**:  
   El servicio que **recibe y procesa mensajes** desde una cola. Por ejemplo, el microservicio de **notificaciones** puede consumir mensajes para enviar alertas a los usuarios sobre nuevas publicaciones o comentarios.
4. **Exchange (Intercambio)**:  
   El **exchange** recibe mensajes de los productores y decide en qué colas colocarlos. Existen diferentes tipos:
   * **Direct**: Envía mensajes a una cola específica.
   * **Fanout**: Envía el mensaje a **todas las colas** vinculadas.
   * **Topic**: Rutea mensajes a colas basadas en patrones específicos.
5. **Binding (Vinculación)**:  
   Define la relación entre un **exchange** y una **cola**, indicando qué mensajes deben enviarse a cuál cola.

**Ejemplo de Uso en tu Arquitectura**

**Escenario: Crear una Publicación**

Cuando un usuario crea una publicación, tu microservicio de publicaciones podría enviar un mensaje a RabbitMQ. Los microservicios de **notificaciones**, **mensajería** y **reportes** pueden recibir este mensaje y realizar acciones específicas.

1. **Productor (Microservicio de Publicaciones)**:  
   Envía un mensaje indicando que se creó una nueva publicación.
2. **Exchange**:  
   Rutea el mensaje a las colas correspondientes de **notificaciones** y **reportes**.
3. **Consumidores (Notificaciones y Reportes)**:
   * El microservicio de **notificaciones** envía una alerta a los seguidores del autor.
   * El microservicio de **reportes** monitorea publicaciones para verificar si necesitan ser revisadas.