Actividad 2

Conceptos y Comandos Básicos de la Replicación en Bases de Datos NoSQL

Fausto Alejandro Gómez Valbuena

Código Banner: 100095458

Facultad de Ingeniería

Ingeniería de Software Virtual

Bases de Datos Avanzadas

William Ruiz Martínez

Mayo 25 de 2024

Corporación Universitaria Iberoamericana

Virtual

**Introducción**

Este documento describe los requerimientos no funcionales enfocados en garantizar la redundancia y la disponibilidad 24x7 de la base de datos **torneo\_futbol\_de\_salon** desarrollada en MongoDB. La finalidad es asegurar la continuidad del servicio y la integridad de los datos en todo momento.

***Punto 1:*** *Especificar a través de un documento de requerimientos no funcionales los criterios de calidad en cuanto a la redundancia y disponibilidad 24x7 para el caso planteado en el primer ejercicio.*

1. **Redundancia**
   1. **Replicación**

* ***Requerimiento:*** Implementar una configuración de replicación en MongoDB.
* ***Descripción:*** La base de datos debe estar configurada en un clúster de replicación (Replica Set) para asegurar que exista al menos una copia secundaria de los datos en tiempo real.
* ***Justificación:*** La replicación asegura que, en caso de fallo de un nodo, otro nodo pueda tomar su lugar sin pérdida de datos.
* ***Medida de Implementación:*** Configurar un Replica Set con un nodo primario y al menos dos nodos secundarios distribuidos localmente.
  1. **Sincronización de Datos**
* ***Requerimiento:*** Asegurar la sincronización continua de datos entre los nodos del Replica Set.
* ***Descripción:*** Los datos deben ser sincronizados en tiempo real entre todos los nodos del Replica Set para evitar inconsistencias.
* ***Justificación:*** La sincronización continua asegura que todas las réplicas tienen la misma información, manteniendo la integridad de los datos.
* ***Medida de Implementación:*** Utilizar la configuración de Write Concern adecuada para asegurar que las escrituras se replican en múltiples nodos antes de confirmar la operación al cliente.

1. **Disponibilidad 24 x 7**
   1. **Tolerancia a Fallos**

* ***Requerimiento:*** Configurar la base de datos para tolerancia a fallos.
* ***Descripción:*** La arquitectura debe permitir el cambio automático a un nodo secundario en caso de que el nodo primario falle (failover).
* ***Justificación:*** Esto asegura que la base de datos esté siempre disponible y operativa, incluso en caso de fallos de hardware o red.
* ***Medida de Implementación:*** Implementar y probar regularmente el proceso de failover automático en el Replica Set de MongoDB.
  1. **Monitoreo y Alertas**
* ***Requerimiento:*** Implementar un sistema de monitoreo y alertas.
* ***Descripción:*** Utilizar herramientas de monitoreo para supervisar el estado de los nodos, el rendimiento de la base de datos y el espacio de almacenamiento.
* ***Justificación:*** El monitoreo continuo permite detectar y solucionar problemas proactivamente antes de que afecten la disponibilidad del servicio.
* ***Medida de Implementación:*** Integrar MongoDB con herramientas de monitoreo como MongoDB Cloud Manager, Prometheus, o ELK Stack, y configurar alertas en caso de anomalías.
  1. **Mantenimiento y Actualizaciones**
* ***Requerimiento:*** Planificar ventanas de mantenimiento y actualizaciones.
* ***Descripción:*** Programar actualizaciones y mantenimientos de manera que se minimice el impacto en la disponibilidad del servicio.
* ***Justificación:*** Asegurar que las actualizaciones de software y hardware no interrumpan el acceso a la base de datos.
* ***Medida de Implementación:*** Realizar mantenimiento en horarios de baja demanda y utilizar estrategias como la actualización en rollo (rolling updates) para mantener el servicio activo durante el proceso.

1. **Escalabilidad**

* ***Requerimiento:*** Asegurar la escalabilidad horizontal.
* ***Descripción:*** La arquitectura debe permitir añadir nuevos nodos para manejar incrementos en la carga de trabajo.
* ***Justificación:*** La escalabilidad horizontal garantiza que el sistema puede manejar un mayor volumen de datos y transacciones sin comprometer el rendimiento.
* ***Medida de Implementación:*** Configurar la base de datos en un diseño sharded (fragmentado) si se prevé un crecimiento significativo en el volumen de datos y transacciones.

1. **Seguridad**
   1. **Autenticación y Autorización**

* ***Requerimiento:*** Implementar autenticación y autorización robustas.
* ***Descripción:*** Usar mecanismos de autenticación para asegurar que solo usuarios autorizados puedan acceder y modificar los datos.
* ***Justificación:*** Proteger los datos de accesos no autorizados y posibles amenazas internas y externas.
* ***Medida de Implementación:*** Configurar autenticación SCRAM-SHA-256 y roles de usuario con permisos específicos en MongoDB.
  1. **Cifrado de Datos**
* ***Requerimiento:*** Cifrar los datos en reposo y en tránsito.
* ***Descripción:*** Asegurar que los datos están cifrados tanto en el disco como durante la transmisión entre clientes y servidores.
* ***Justificación:*** El cifrado protege la confidencialidad de los datos contra intercepciones y accesos no autorizados.
* ***Medida de Implementación:*** Utilizar TLS/SSL para la transmisión de datos y cifrado a nivel de almacenamiento (data-at-rest encryption).

***Punto 2:*** *Desarrollar las consultas para replicar las bases de datos del evento deportivo planteado en el ejercicio N° 1, en mínimo 3 nodos, acorde al documento de requerimientos no funcionales, se debe tener en cuenta: a) Definir la estrategia de replicación que cumplan los requerimientos planteados. a) Ejecutar los comandos para crear el entorno de replicación definido.*

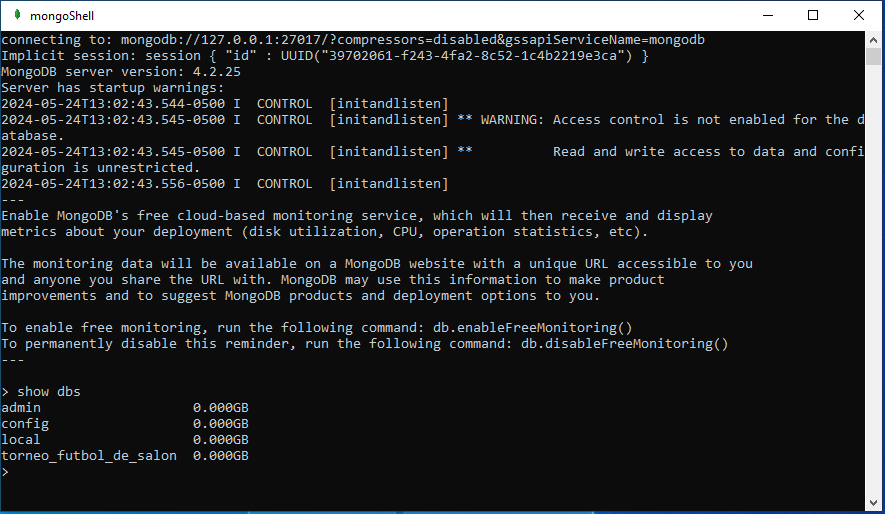
1. **Definir la Estrategia de Replicación**

Vamos a utilizar un Replica Set de MongoDB, que es una configuración de alta disponibilidad y redundancia en la que un grupo de servidores de MongoDB mantiene las mismas copias de los datos. En un Replica Set, uno de los nodos actúa como el primario y los otros como secundarios.

1. **Ejecutar los Comandos para Crear el Entorno de Replicación**

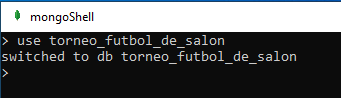
* **Show dbs**

Muestra todas las bases de datos NoSql que tenemos.



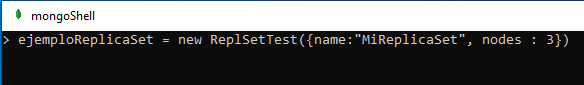
* **use torneo\_futbol\_de\_salon**

Hacemos uso de la base de datos selecionada.

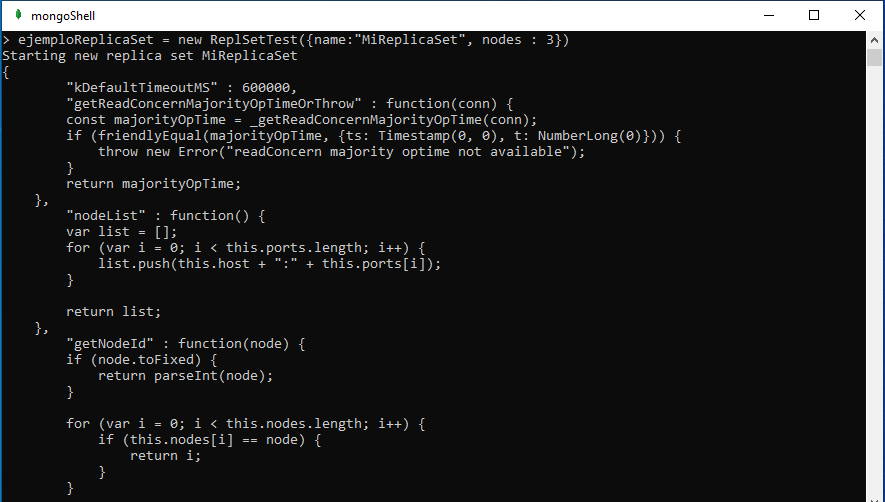


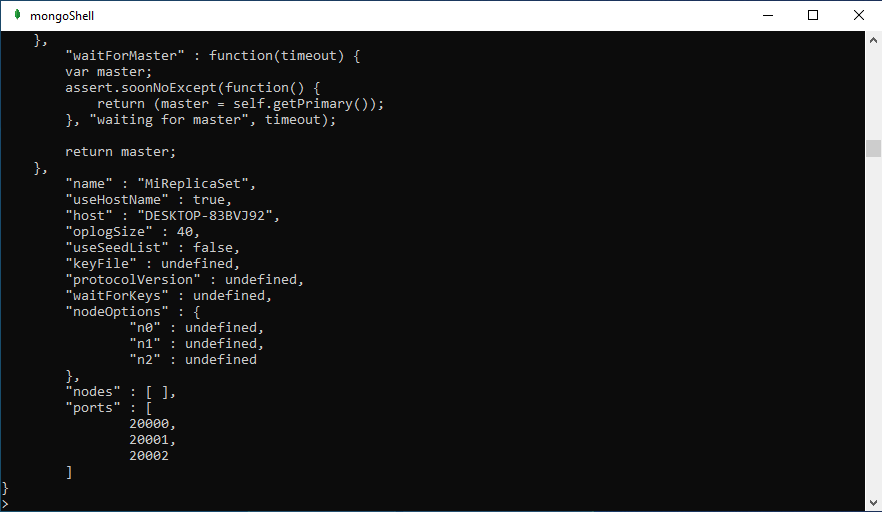
**Paso 1:** Configurar cada nodo de MongoDB.

* **ejemploReplicaSet = new ReplSetTest({name:"MiReplicaSet", nodes : 3})**



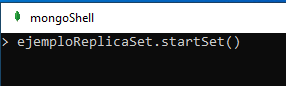
Al ejecutar el comando, por consola nos imprimirá el contenido del objeto **ReplSetTest** que acabamos de crear.





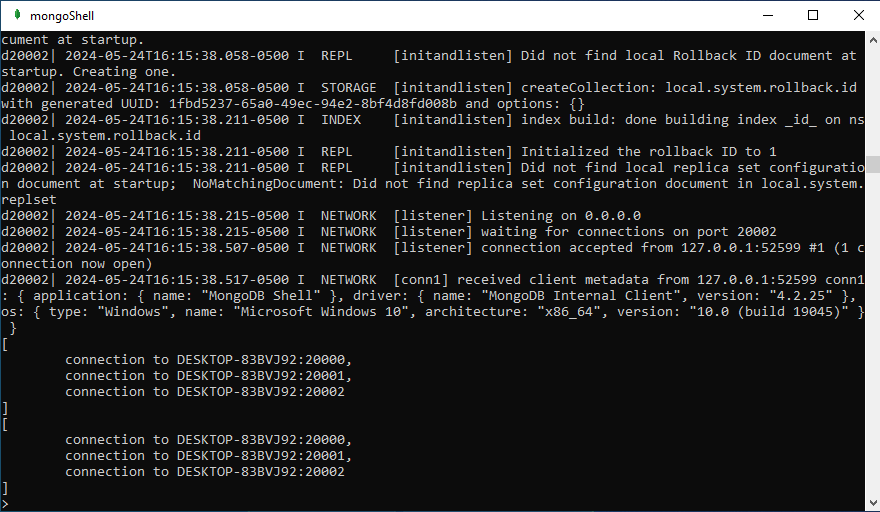
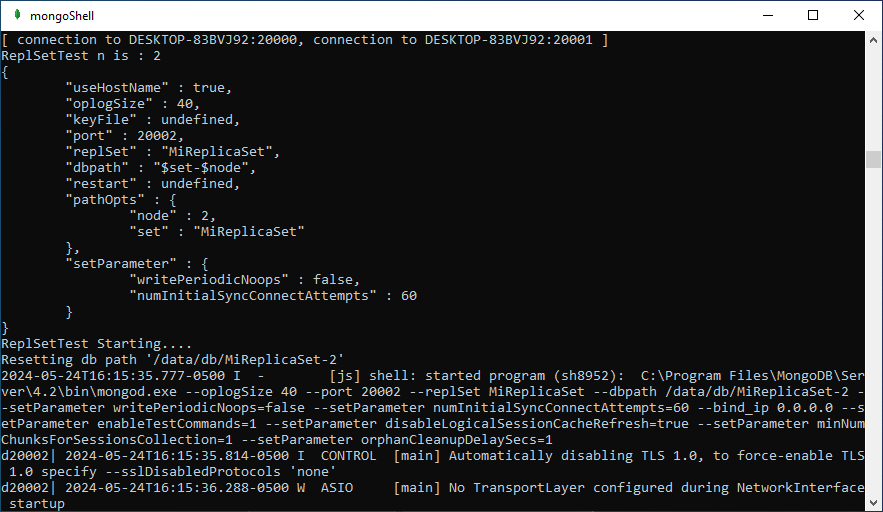
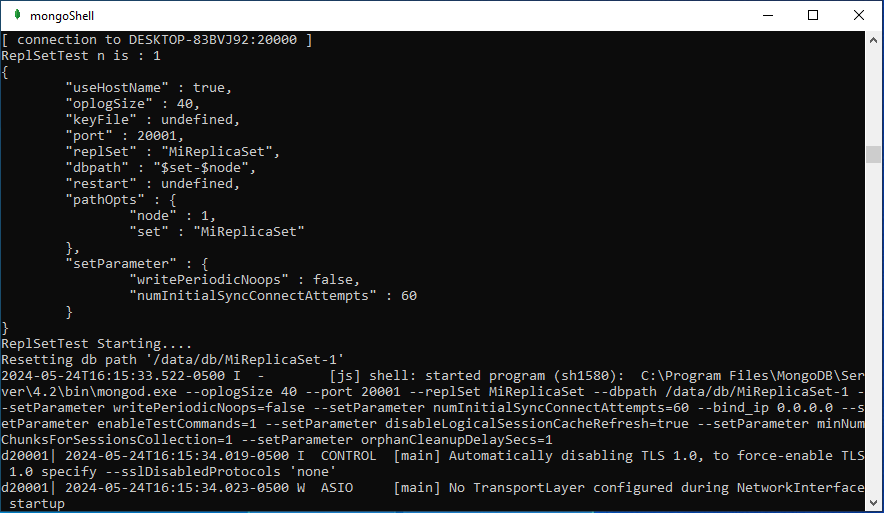
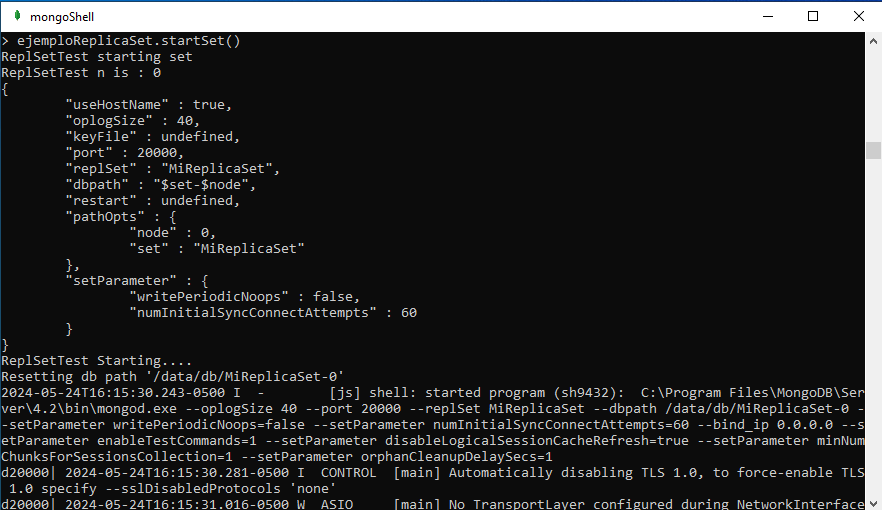
**Paso 2:** Inicializar el Replica Set.

* **ejemploReplicaSet.startSet()**



Para arrancar los nodos del grupo de réplica ejecutaremos la función **startSet()** sobre el objeto que representa el grupo de réplica.

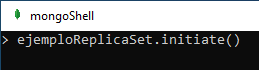
Como salida del comando veremos la configuración de cada nodo del grupo de réplica y los mensajes de logs que nos indican que están arrancando cada una de las instancias.



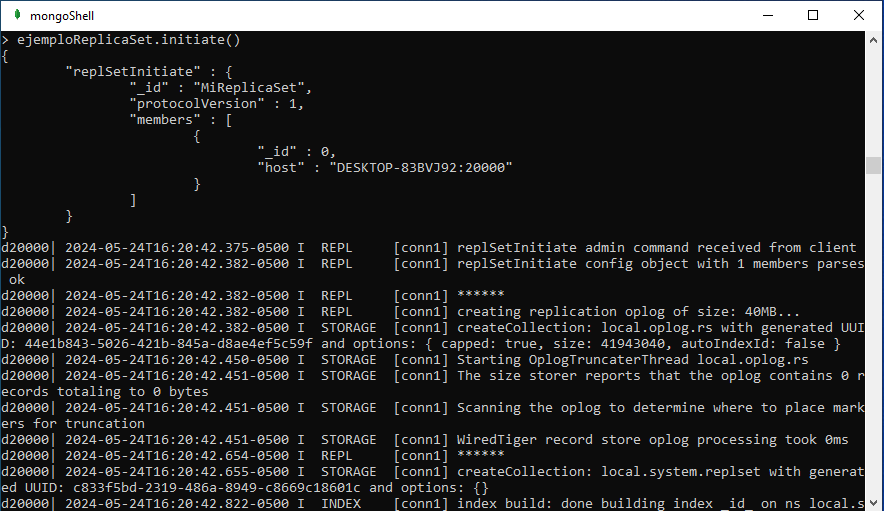
Durante la inicialización vemos los puertos en los que se va levantando cada uno de los nodos. En este caso, y por defecto, se levantan en los puertos **20000**, **20001** y **20002**.

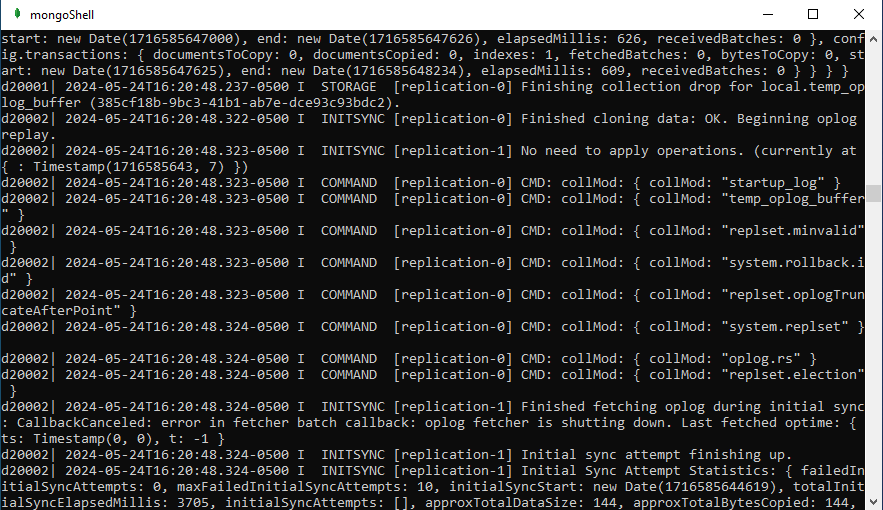
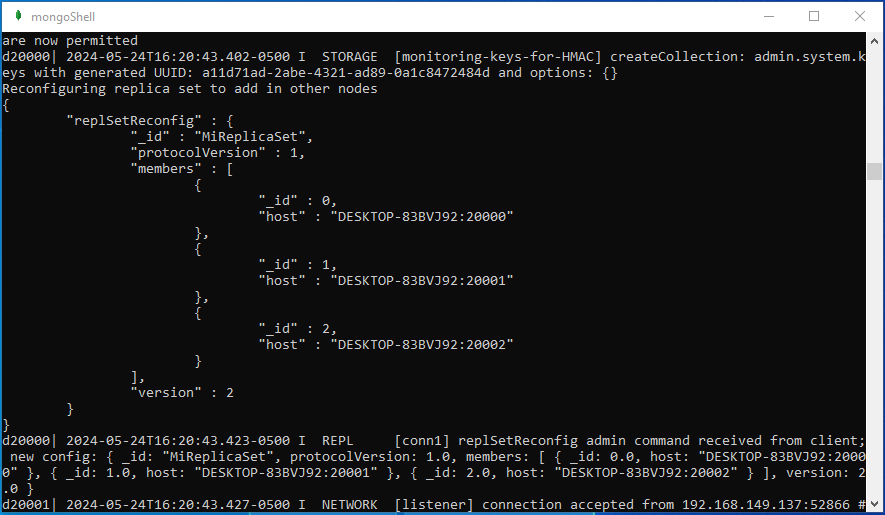
**Paso 3:** Arranque del proceso de réplica.

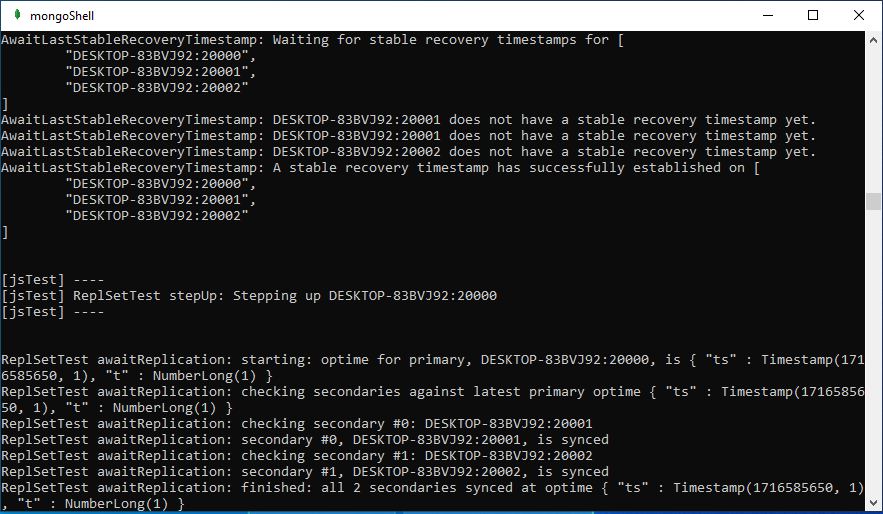
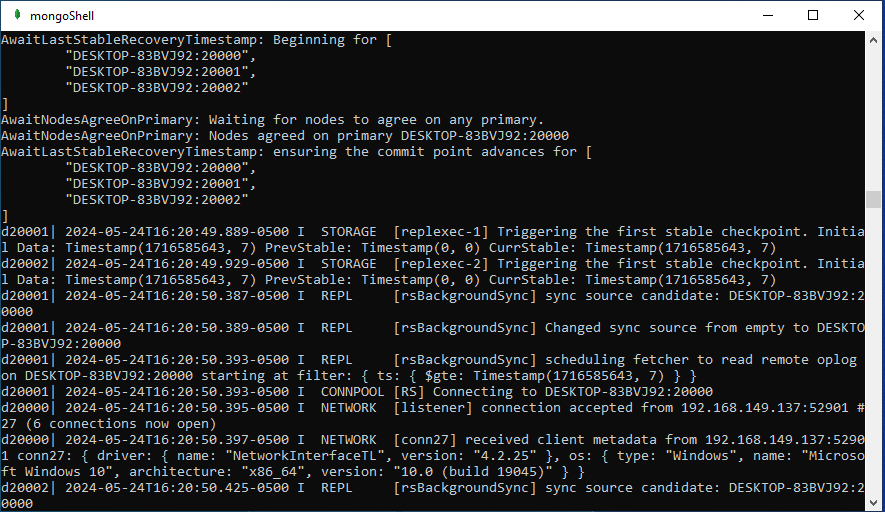
* **ejemploReplicaSet.initiate()**



Al ejecutar la función, veremos como la salida de consola nos muestra la configuración de miembros de réplica set y a continuación se activa la funcionalidad de réplica en el grupo.



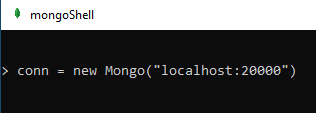




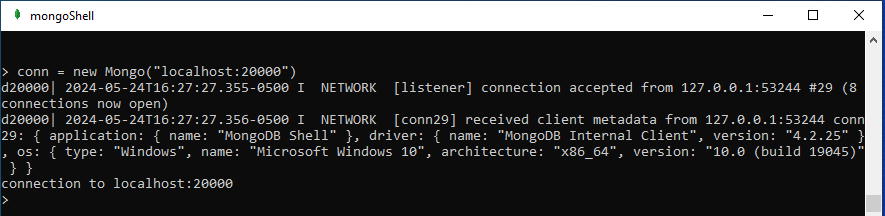
**Paso 4:** Prueba del grupo de réplica.

Conexión al nodo primario del grupo de réplica

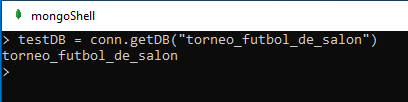
* **conn = new Mongo("localhost:20000")**



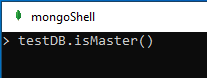
Una vez obtenida la conexión, obtenemos la BD sobre la que realizaremos la prueba.



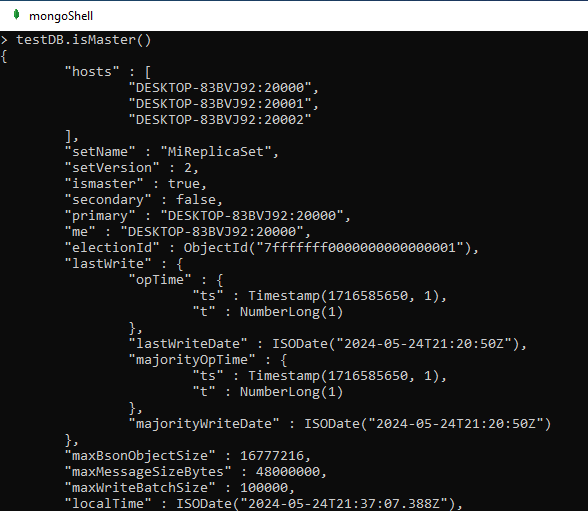
* **testDB = conn.getDB("torneo\_futbol\_de\_salon")**



* **testDB.isMaster()**

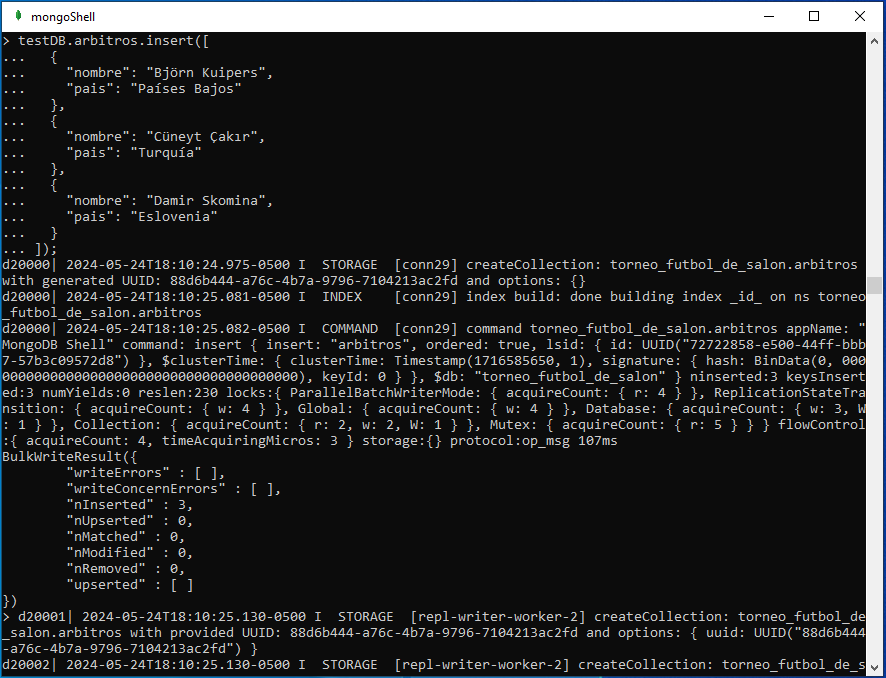


Por último, sobre la BD, preguntaremos si es el nodo primario, utilizando la función isMaster():

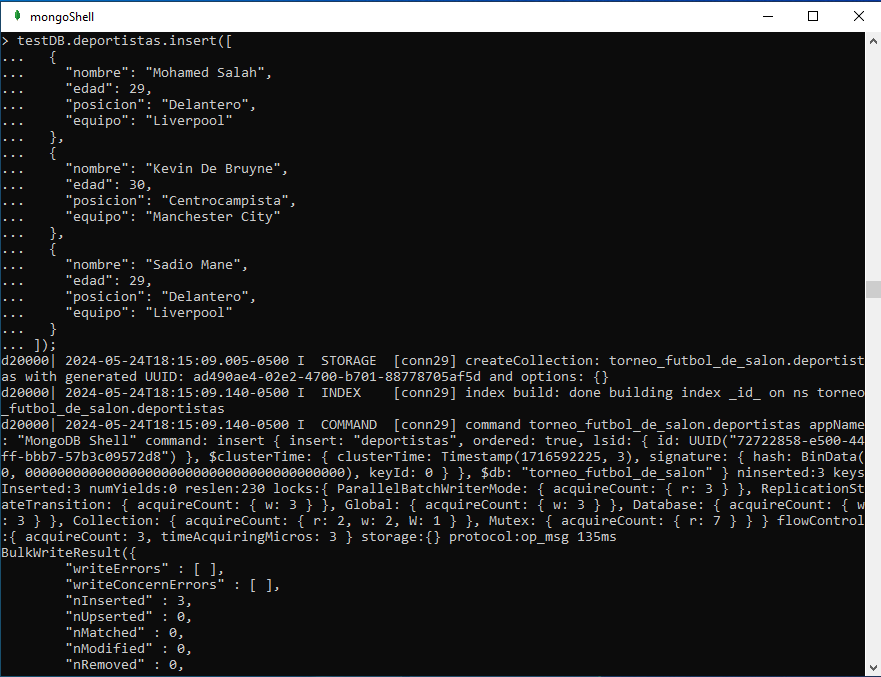


**Paso 5:** Insertar un conjunto de datos en el nodo primario.

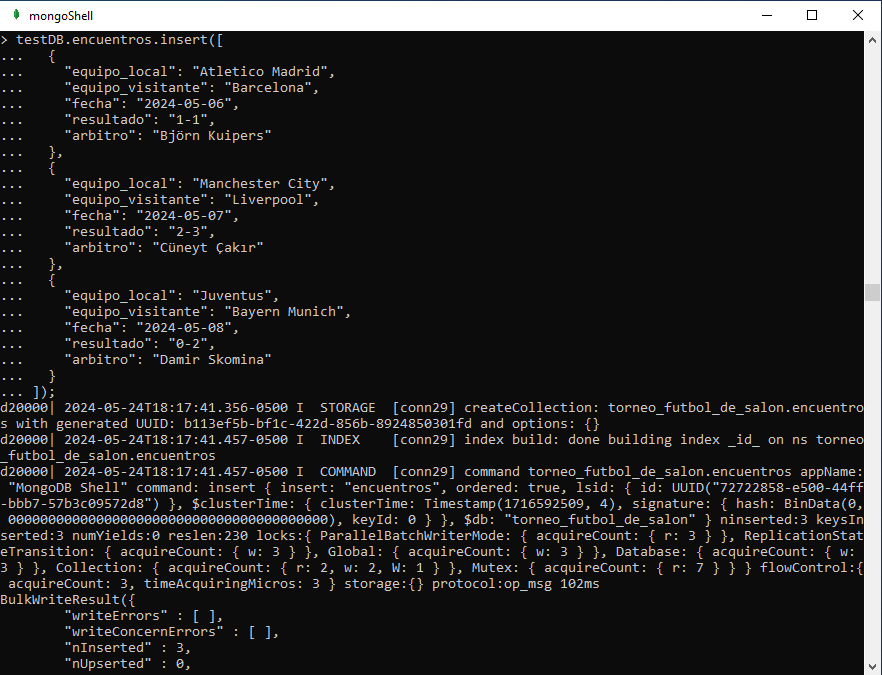
* **testDB.arbitros.insert()**



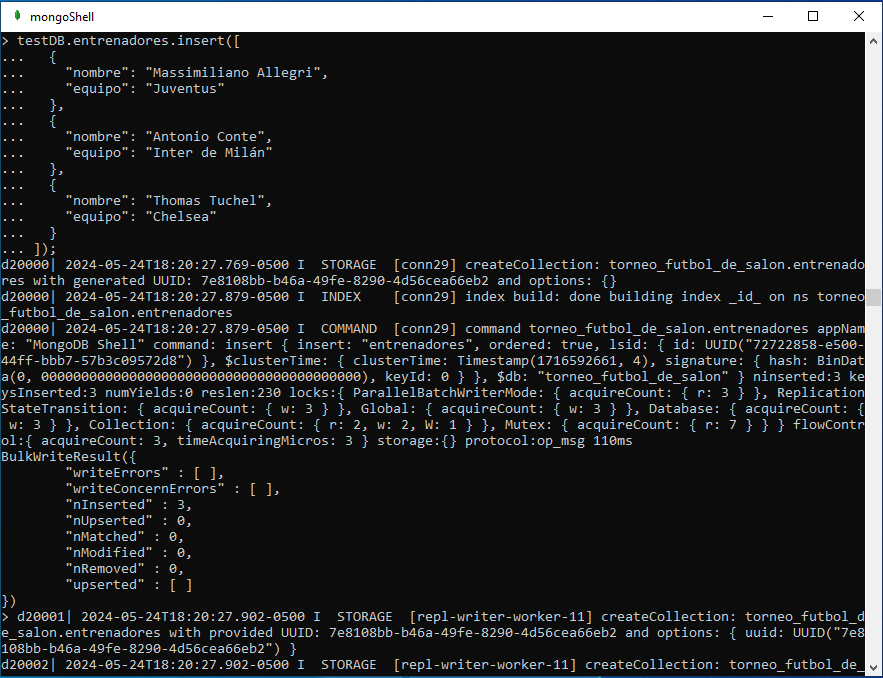
* **testDB.deportistas.insert()**



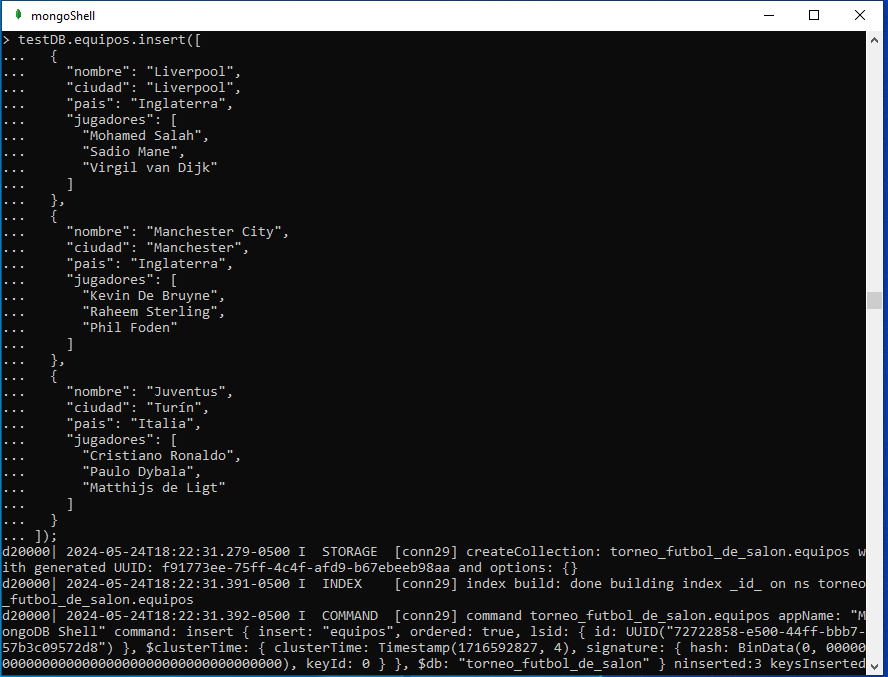
**testDB.encuentros.insert()**

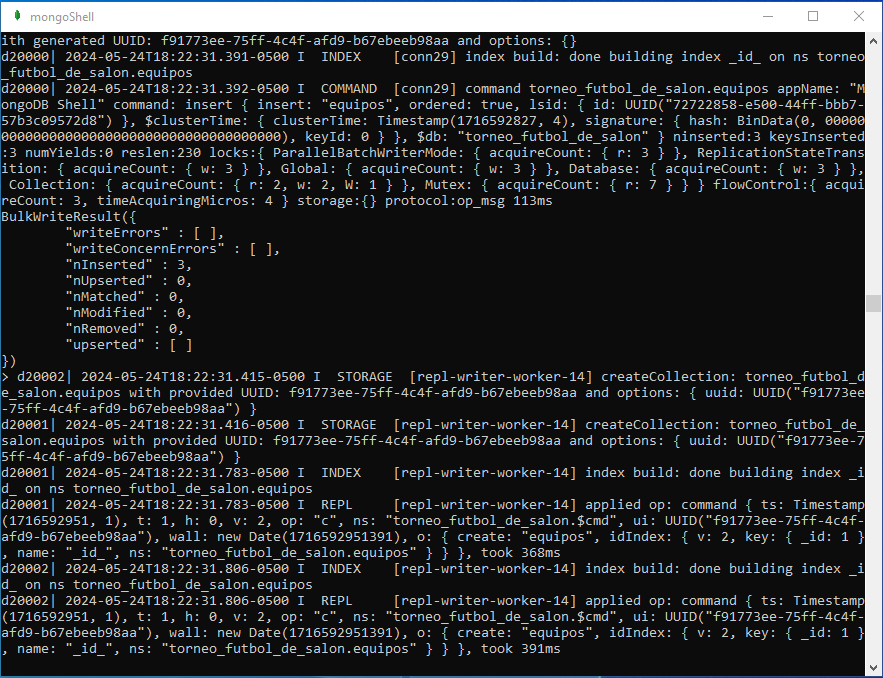


* **testDB.entrenadores.insert()**

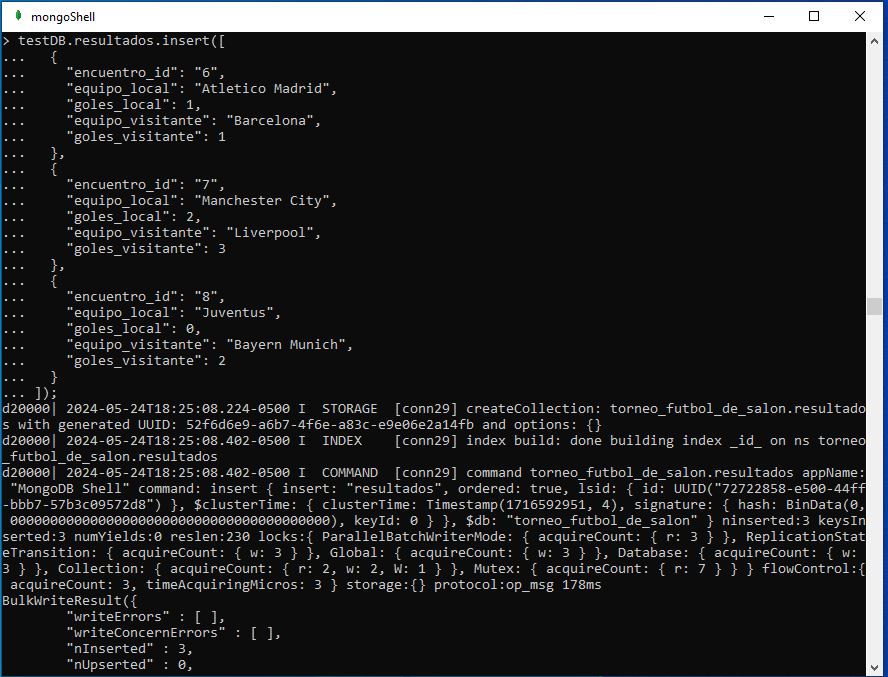


* **testDB.equipos.insert()**

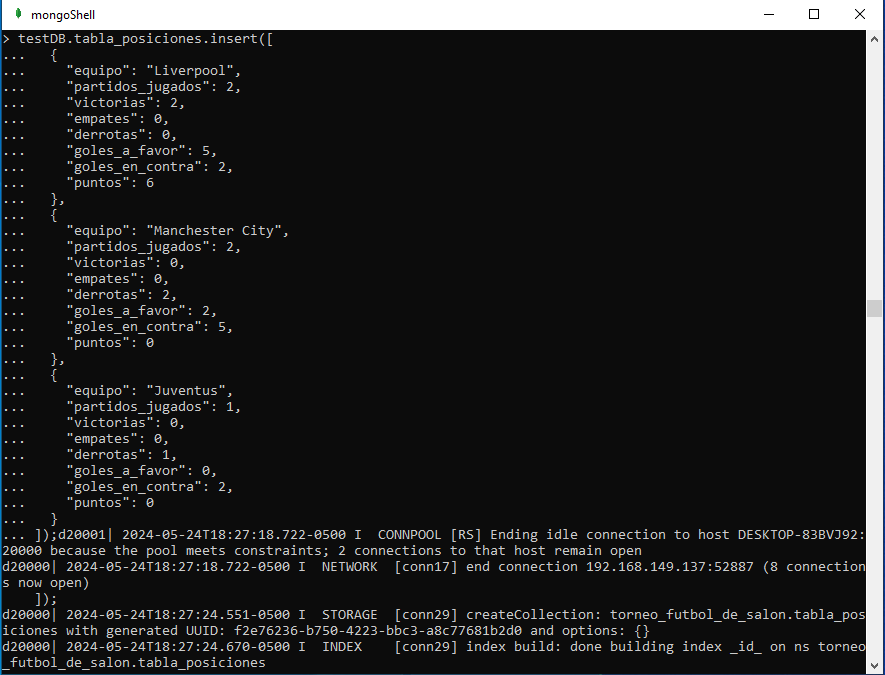


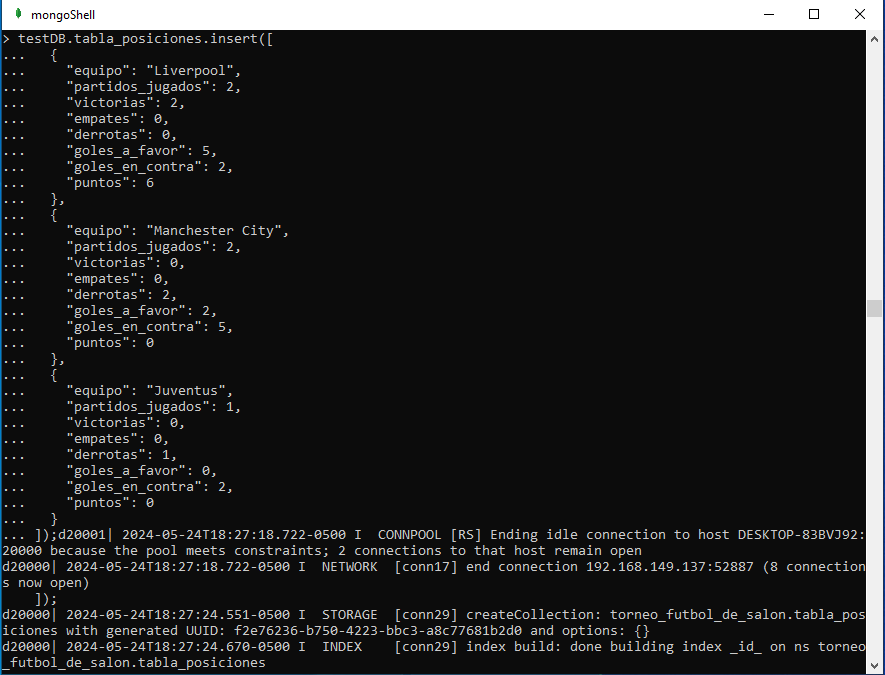


* **testDB.resultados.insert()**

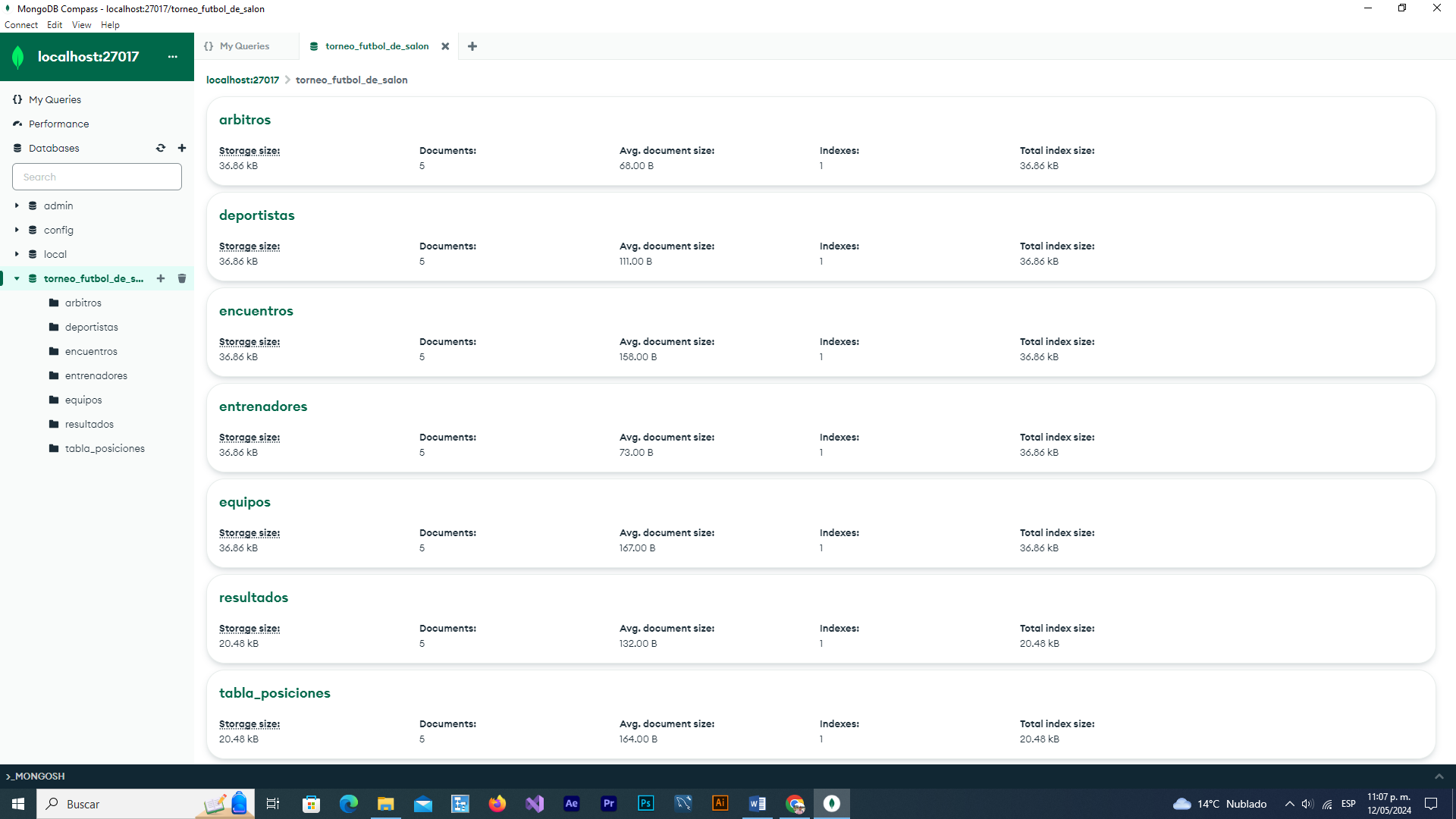


* **testDB.tabla\_posiciones.insert()**

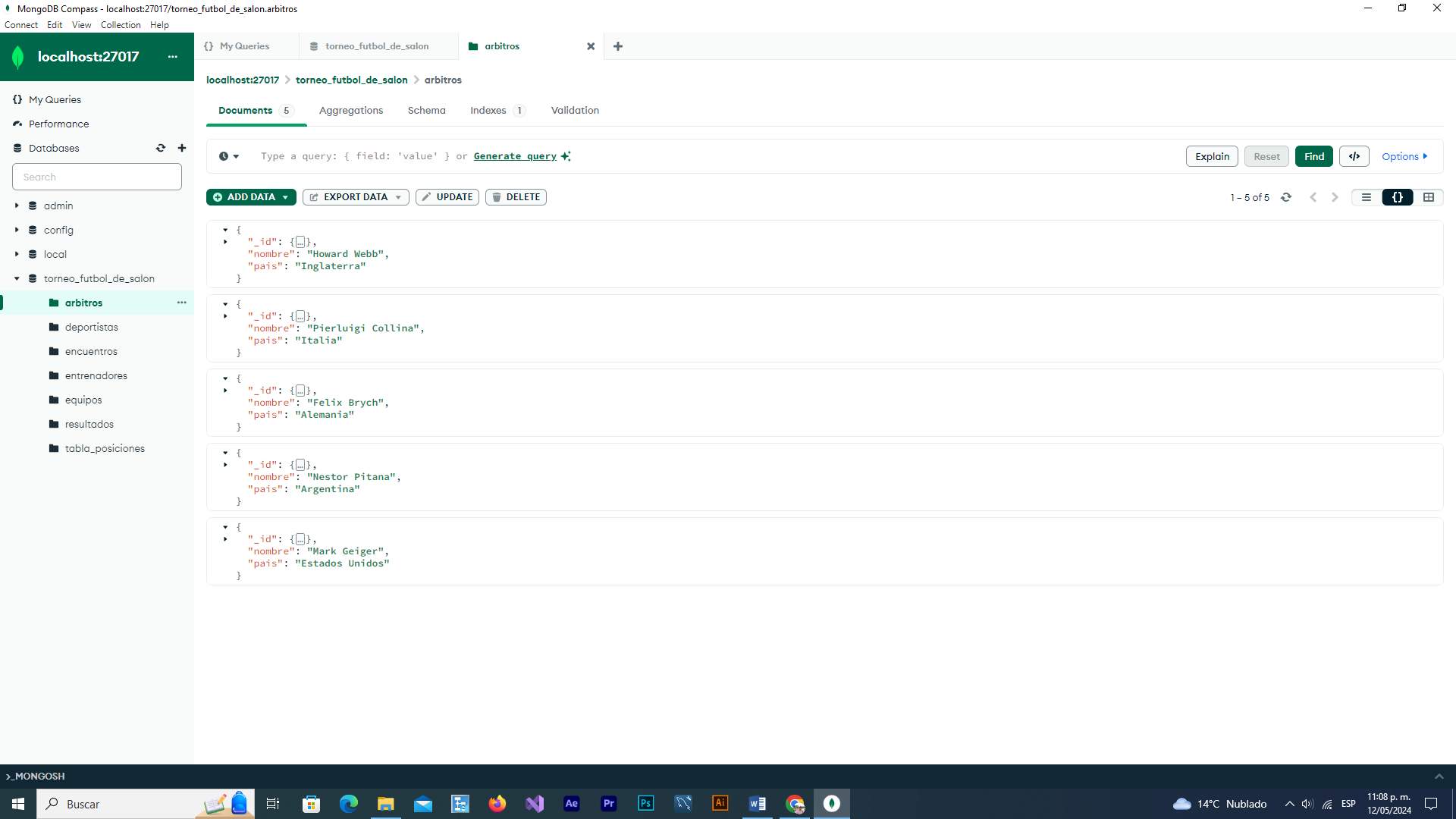




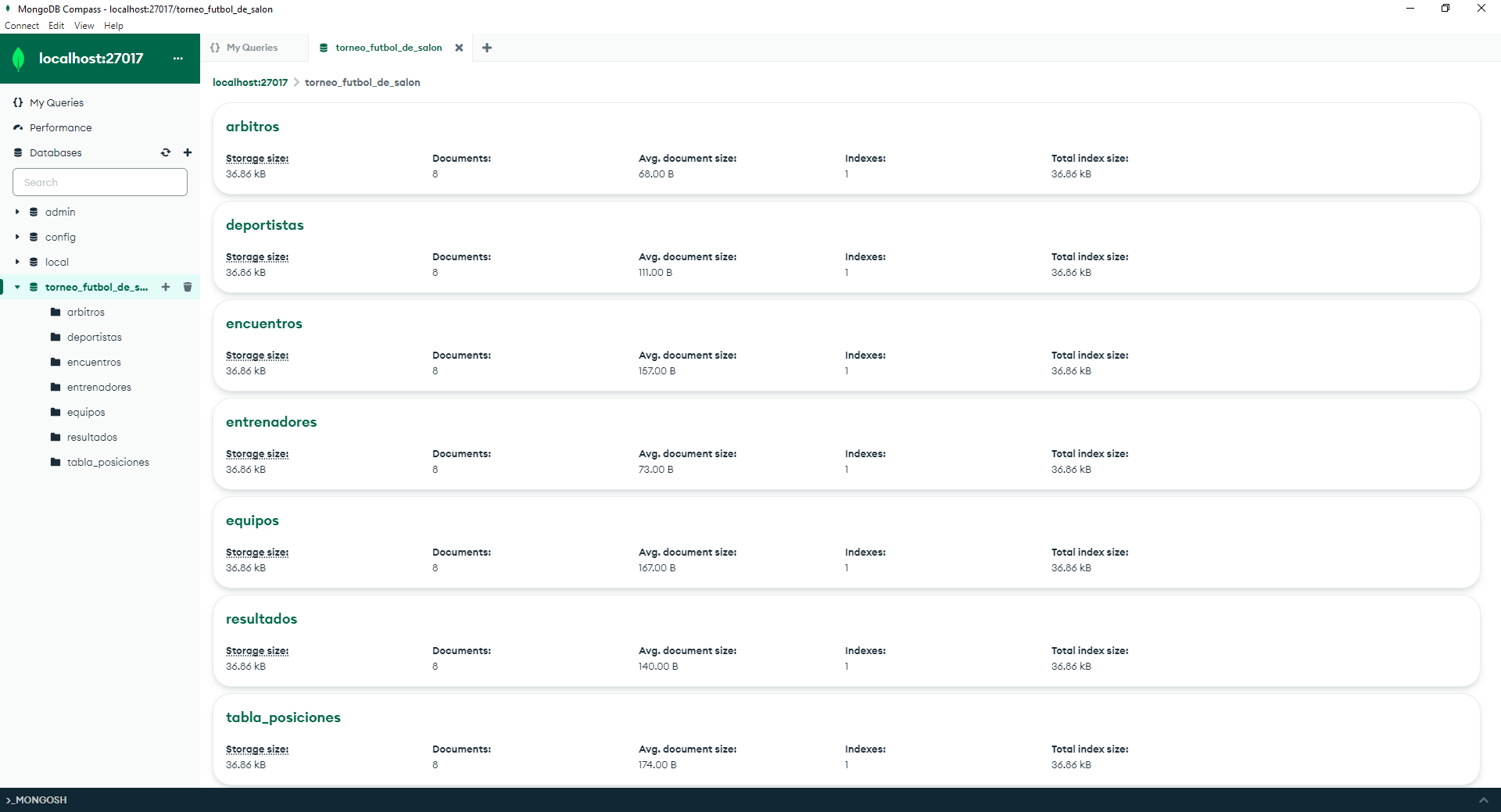
**Imagen de mongo DB antes de la inserción de datos**



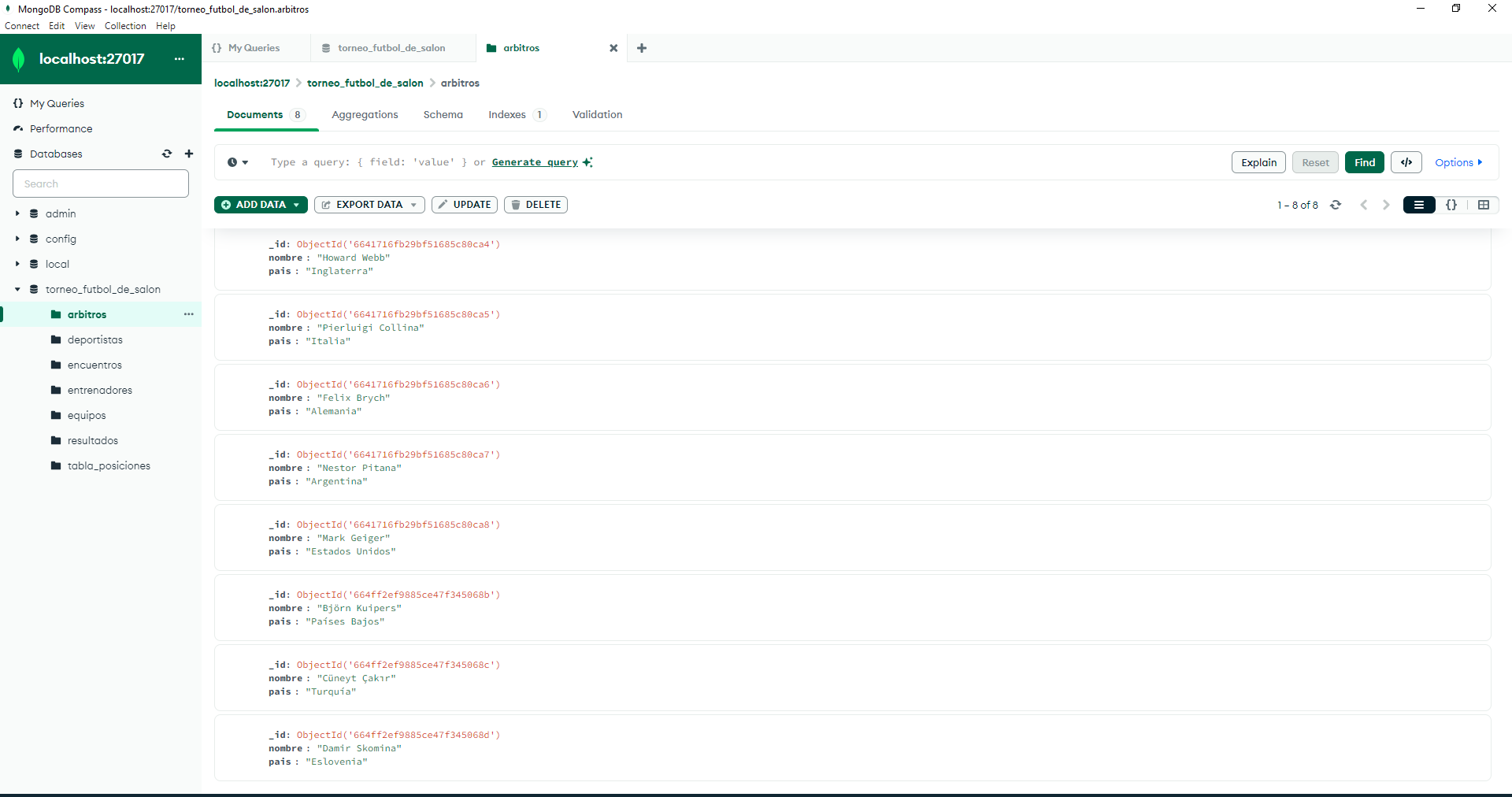
**Árbitros**



**Imagen de mongo DB después de la inserción de datos**

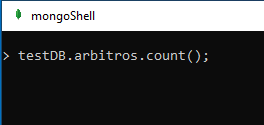


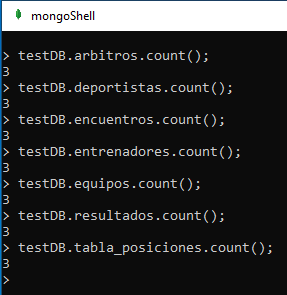
**Árbitros**



Comprobamos que se han almacenado los registros en la colección.

* **testDB.arbitros.count();**

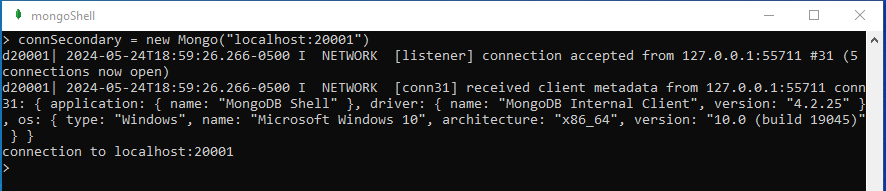




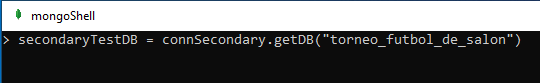
**Paso 6:** Comprobación de la réplica sobre los nodos secundarios.

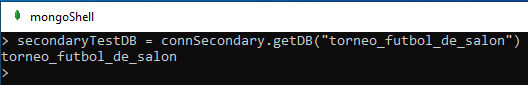
* **connSecondary = new Mongo("localhost:20001")**



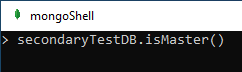


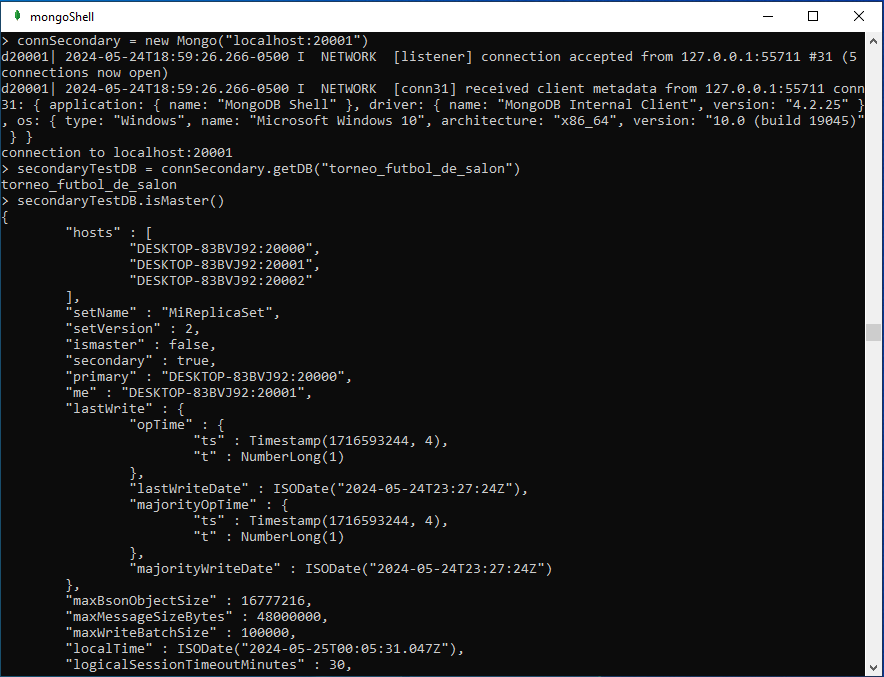
* **secondaryTestDB = connSecondary.getDB("torneo\_futbol\_de\_salon")**

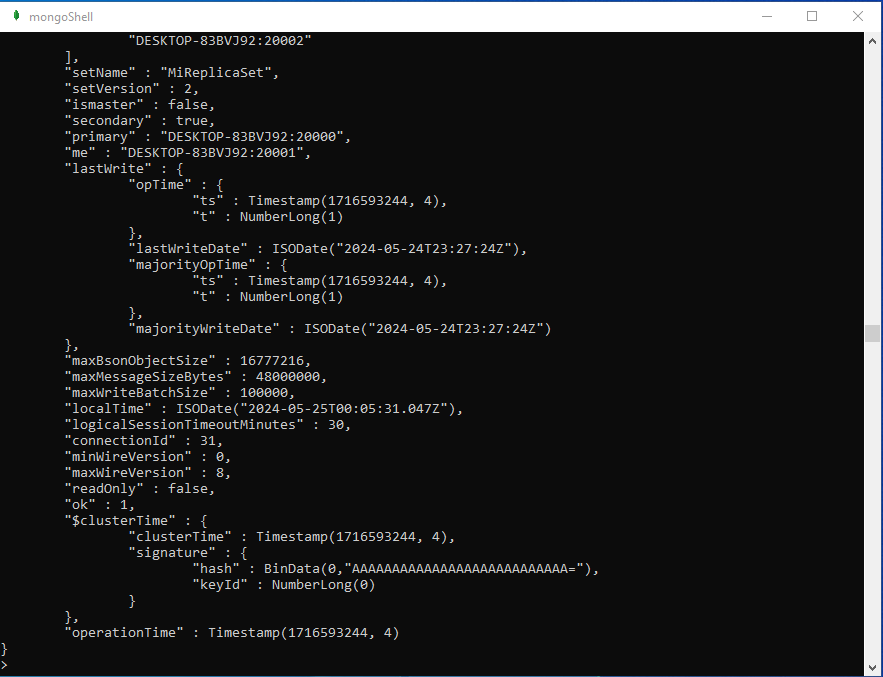




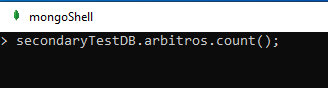
* **secondaryTestDB.isMaster()**

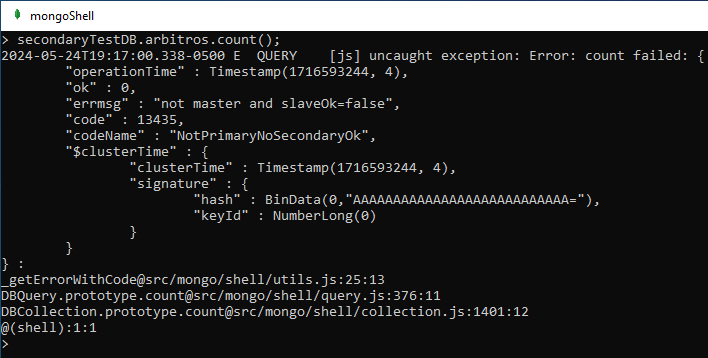






* **secondaryTestDB.arbitros.count();**

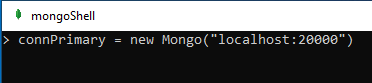


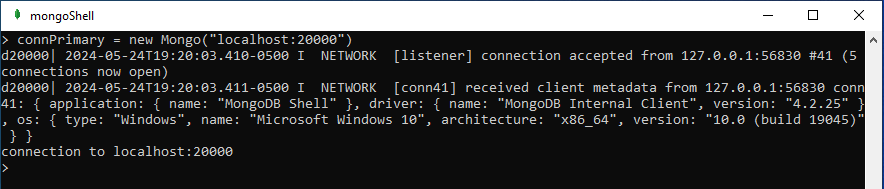


En este caso se obtiene un error porque por defecto, en los nodos secundarios en un grupo de réplica, no se admiten operaciones ni de escritura ni de lectura. Todas las operaciones deben realizarse siempre sobre el nodo principal.

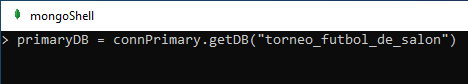
**Paso 7:** Detener el nodo primario.

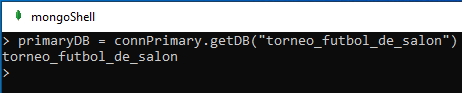
* **connPrimary = new Mongo("localhost:20000")**



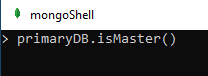


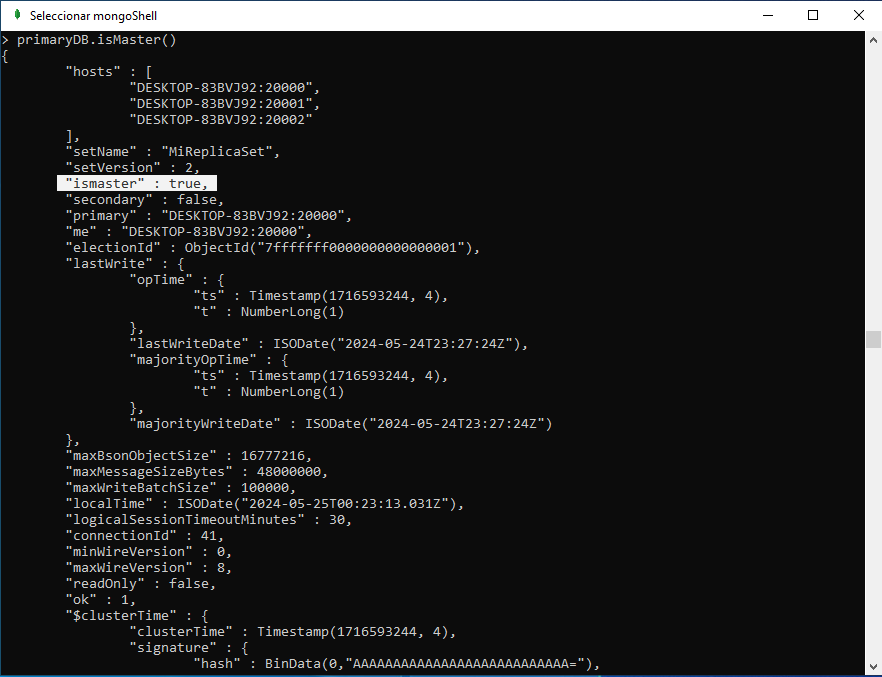
* **primaryDB = connPrimary.getDB("torneo\_futbol\_de\_salon")**



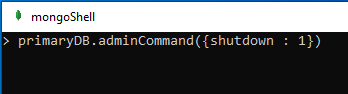


* **primaryDB.isMaster()**





* **primaryDB.adminCommand({shutdow : 1})**



***Punto 3:*** *Especificar a través de un documento de casos de pruebas la verificación del mecanismo de replicación planteado funciona en cuanto a la redundancia y disponibilidad 24x7 para el caso planteado en la primera actividad.*

**Objetivo**

Verificar que el mecanismo de replicación en MongoDB garantiza la redundancia y disponibilidad 24x7 para la base de datos del torneo de fútbol de salón, conforme a los requerimientos no funcionales.

**Alcance**

Las pruebas se realizarán sobre un Replica Set configurado en tres nodos (servidores) con los nombres “20000”, “20001” y “20002”.

**Casos de Prueba**

**Prueba 1:** Verificación de la Configuración Inicial del Replica Set

* + **Objetivo:** Asegurar que el Replica Set esté correctamente configurado.
  + **Precondición:** Los nodos están configurados y el Replica Set se ha iniciado.
  + **Pasos:**

1. Conectar al nodo primario (“localhost:20000”).
2. Ejecutar el comando “conn = new Mongo("localhost:20000")
3. Verificar que todos los nodos están presentes y en estado “SECONDARY” o “PRIMARY”.
   * **Resultado Esperado:** La salida del comando “conn = new Mongo("localhost:20000")” muestra que todos los nodos están correctamente configurados y sincronizados.

**Prueba 2:** Inserción de Datos y Propagación

* + **Objetivo:** Verificar que los datos insertados en el nodo primario se replican en los nodos secundarios.
  + **Precondición:** El Replica Set está en funcionamiento.
  + **Pasos:**

1. Conectar al nodo primario ("localhost:20000").
2. Insertar un nuevo documento en una colección.
3. Conectar a los nodos secundarios ("localhost:20001" y "localhost:20002).
4. Verificar que el nuevo documento esté presente en los nodos secundarios.

* **Resultado Esperado:** El documento insertado en el nodo primario está presente en todos los nodos secundarios.

**Prueba 3:** Failover Automático

* + **Objetivo:** Verificar que el sistema realiza un failover automático en caso de que el nodo primario falle.
  + **Precondición:** El Replica Set está en funcionamiento y sincronizado.
  + **Pasos:**

1. Identificar el nodo primario actual.
2. Apagar el nodo primario.
3. Esperar a que uno de los nodos secundarios se convierta en primario.
4. Verificar el estado del Replica Set con “rs.status()”.
5. Verificar que las operaciones de escritura y lectura pueden continuar en el nuevo nodo primario.
   * **Resultado Esperado:** Un nodo secundario se convierte en primario y el sistema continúa funcionando sin interrupciones.

**Prueba 4:** Reintegración del Nodo Primario Original

* + **Objetivo:** Verificar que el nodo primario original puede reintegrarse al Replica Set y volver a su estado secundario.
  + **Precondición:** El nodo primario original está apagado y otro nodo ha tomado su lugar.
  + **Pasos:**

1. Reiniciar el nodo primario original.
2. Conectar al Replica Set y verificar el estado con “rs.status()”.
3. Verificar que el nodo reintegrado esté en estado “secondary”.
   * **Resultado Esperado:** El nodo primario original se reintegra al Replica Set y se sincroniza correctamente como nodo secundario.

**Prueba 5:** Consistencia de Datos

* + **Objetivo:** Verificar la consistencia de datos entre todos los nodos del Replica Set.
  + **Precondición:** El Replica Set está en funcionamiento y sincronizado.
  + **Pasos:**

1. Realizar una serie de operaciones de escritura en el nodo primario.
2. Conectar a cada nodo secundario y realizar consultas para verificar la presencia y exactitud de los datos.
   * **Resultado Esperado:** Los datos en los nodos secundarios son consistentes con los datos en el nodo primario.

***Punto 4:*** *Ejecutar los casos de prueba planteados y generar un reporte de los resultados y análisis.*

**Reporte de Resultados y Análisis de los Casos de Prueba**

Este documento presenta los resultados de la ejecución de los casos de prueba para verificar el mecanismo de replicación de MongoDB, garantizando la redundancia y disponibilidad 24x7 para la base de datos del torneo de fútbol de salón.

1. **Configuración del Entorno**

Nodos del Replica Set:

* + Nodo 1: “20000” (Primario inicial)
  + Nodo 2: “20001” (Secundario)
  + Nodo 3: “20002” (Secundario)

1. **Caso de Prueba 1:** Verificación de la Configuración Inicial del Replica Set

* **Objetivo:** Asegurar que el Replica Set esté correctamente configurado.
* **Comando Ejecutado:** rs.status()
* **Análisis:** Todos los nodos están correctamente configurados. El nodo “20000” es el primario, y los nodos “20001” y “20002” son secundarios.

1. **Caso de Prueba 2:** Inserción de Datos y Propagación

* **Objetivo:** Verificar que los datos insertados en el nodo primario se replican en los nodos secundarios.
* **Inserción de Documento:**
* **Verificación en Nodo Secundario (“20001”):**
* **Análisis:** El documento insertado en el nodo primario se replicó correctamente a los nodos secundarios.

1. **Caso de Prueba 3:** Failover Automático

* **Objetivo:** Verificar que el sistema realiza un failover automático en caso de que el nodo primario falle.
* **Acción:** Apagar el nodo primario (“20000”):
* **Análisis:** El nodo (“20001”) se convirtió en primario, lo que confirma que el failover automático funcionó correctamente.

1. **Caso de Prueba 4:** Reintegración del Nodo Primario Original

* **Objetivo:** Verificar que el nodo primario original puede reintegrarse al Replica Set y volver a su estado secundario.
* **Acción:** Reiniciar el nodo primario original (“20000”).
* **Análisis:** El nodo (“20001”) se reintegró correctamente y está en estado `SECONDARY`, indicando que la reintegración fue exitosa.

1. **Caso de Prueba 5:** Consistencia de Datos

* **Objetivo:** Verificar la consistencia de datos entre todos los nodos del Replica Set.
* **Análisis:** Los datos son consistentes en todos los nodos, confirmando que las operaciones de escritura en el nodo primario se replican correctamente.

**Conclusión**

Todas las pruebas han sido exitosas, verificando que el mecanismo de replicación de MongoDB cumple con los requerimientos de redundancia y disponibilidad 24x7. El sistema es capaz de manejar fallos del nodo primario con failover automático, mantener la consistencia de datos y permitir la reintegración de nodos sin interrupciones.

**Bibliografía**

* Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC. <https://elibro.net/es/lc/biblioibero/titulos/58524>
* Aramburu Cabo, M. J. y Sanz Blasco, I. (2012). Bases de datos avanzadas. D - Universitat Jaume I. Servei de Comunicació i Publicacions, <https://elibro.net/es/lc/biblioibero/titulos/51741>
* Chen, Y., & Gibbons, P. B. (2018). \*\*Scalable and Elastic NoSQL Data Stores: Technologies and Principles\*\*. Morgan & Claypool Publishers.
* Chodorow, K. (2019). \*\*MongoDB: The Definitive Guide: Powerful and Scalable Data Storage\*\* (3rd ed.). O'Reilly Media.
* Hows, D., Plugge, E., Membrey, P., & Hawkins, T. (2015). \*\*MongoDB Basics\*\*. Apress.
* Meier, J. D., Farre, C., Bhatia, P., & Mackman, A. (2010). \*\*Improving .NET Application Performance and Scalability\*\*. Microsoft Press.
* MongoDB, Inc. (n.d.). \*\*MongoDB Manual\*\*. Retrieved from https://docs.mongodb.com/manual/
* Patel, J. (2020). \*\*NoSQL Database: A Comparison of MongoDB and Cassandra\*\*. International Journal of Scientific & Technology Research, 9(4), 102-108.
* Sadalage, P. J., & Fowler, M. (2012). \*\*NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence\*\*. Addison-Wesley.