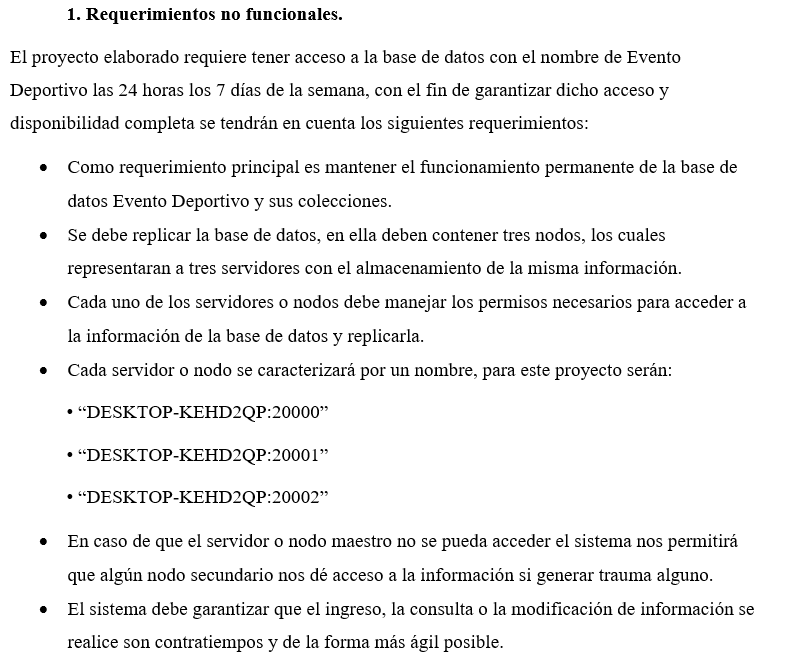
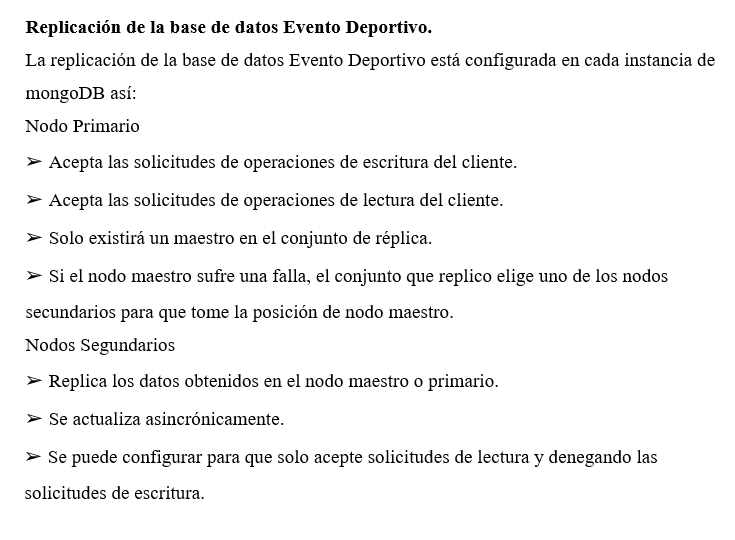
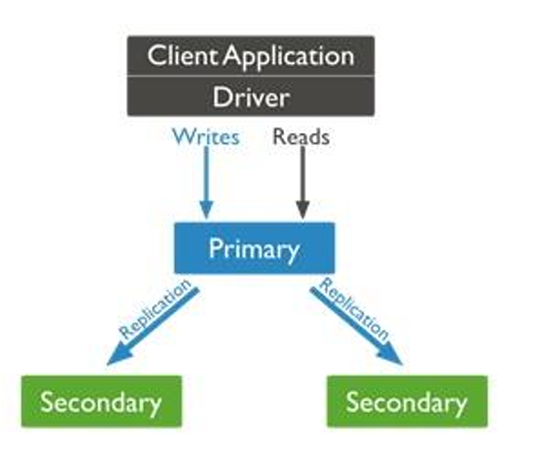
1. Desarrollo Actividad
   1. Especificar a través de un documento de requerimientos no funcionales los criterios de calidad en cuanto a la redundancia y disponibilidad 24x7 para el caso planteado en la primera actividad. 
   2. Desarrollar las consultas para replicar las bases de datos del evento deportivo en mínimo 3 nodos, planteado acorde al documento de requerimientos no funcionales, esto es:

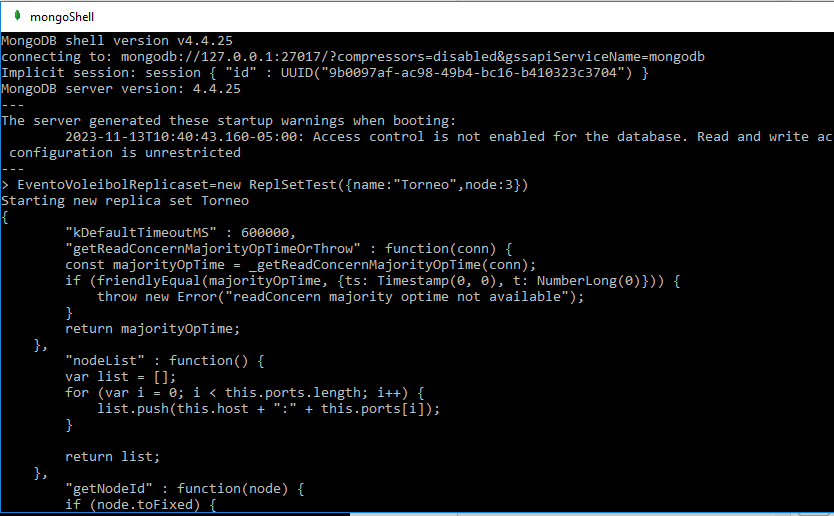


**Diagrama de replicación:**

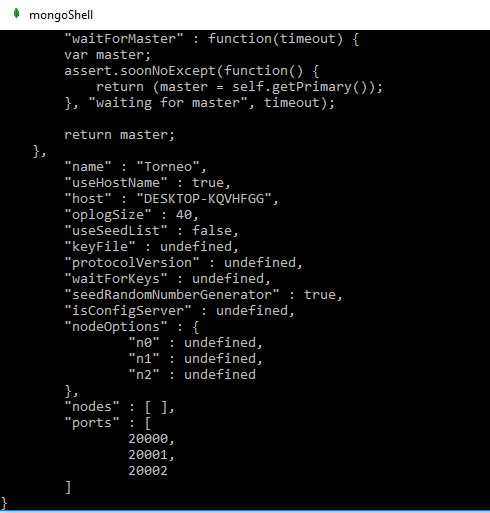
2.1. Scripts.

**EventoVoleibolReplicaset=new ReplSetTest({name:"Torneo",nodes:3})**

Este comando nos permitirá crear un conjunto de 3 réplicas.

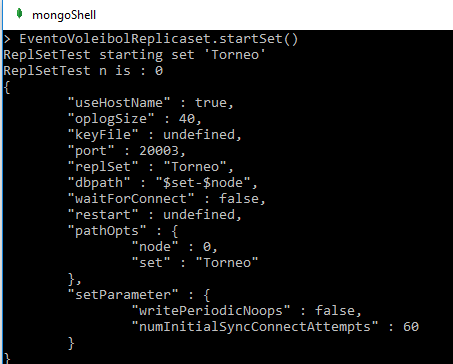


El sistema crea 3 puertos

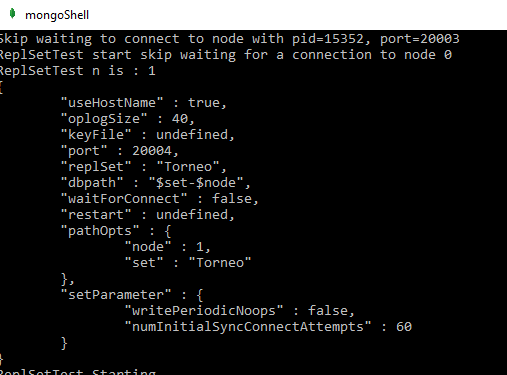


**EventoVoleibolReplicaSet.startSet()**

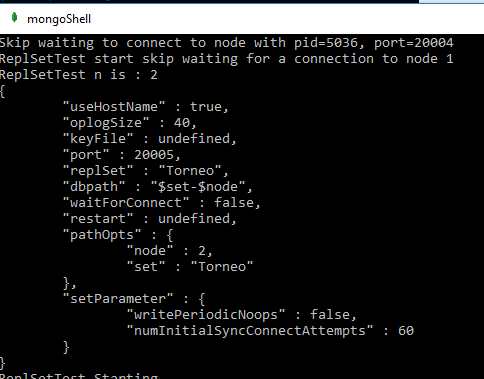
**Nodo #1**



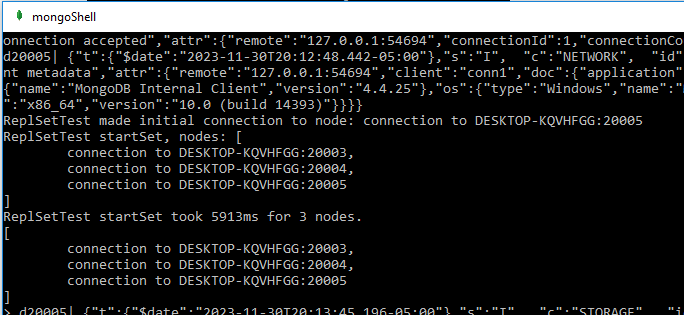
**Nodo #2**



**Nodo # 3**

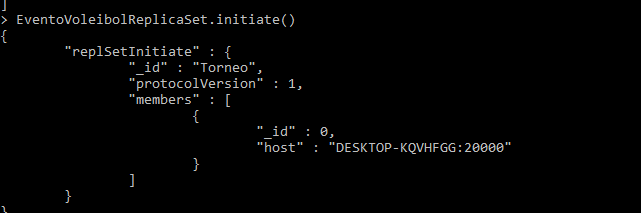


**Conexiones**



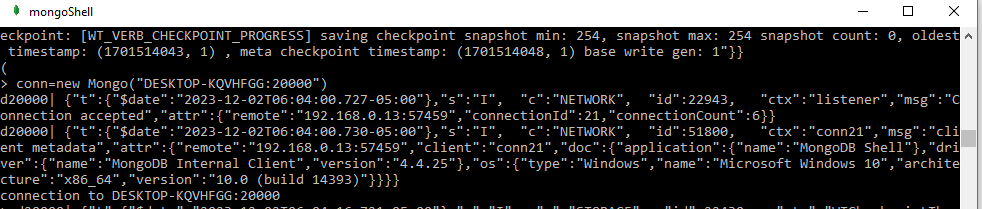
**EventoVoleibolReplicaset.initiate()**

Este comando nos permite iniciar con la replicación funcionalidad de réplica en el grupo.



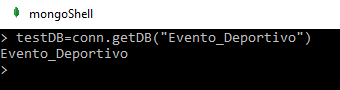
**conn=new Mongo("DESKTOP-KQVHFGG:20000")**

Este comando nos permite conectar al nodo #1 puerto 20000



**testDB=conn.getDB(“Evento\_Deportivo")**

Nos conectamos con la base de datos **Evento\_Deportivo**



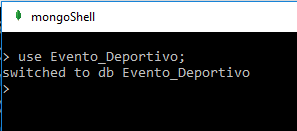
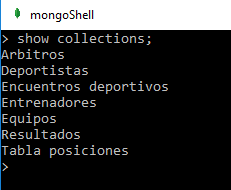
**testDB.isMaster()**

consultamos al nodo #1 o puerto 20000 si es el nodo principal o maestro, la respuesta es verdadero.



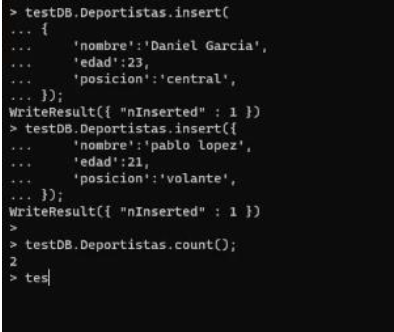
**use Evento\_Deportivo; - > show collections;**

Utilizamos la base de datos EVENTO DEPORTIVO y consultamos sus colecciones

**testDB.Deportistas.insert({})**

Con este comando ingresamos datos a la colección DEPORTISTAS en la base de datos Evento\_Deportivo, lo estamos aplicando en el nodo principal.



**testDB.Deportistas.count();**

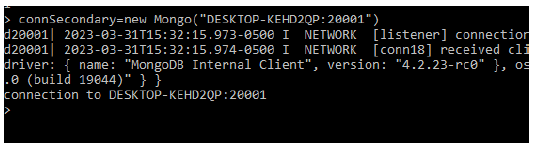
**> testDB.Deportistas.find().pretty();**

Con estos comandos nos permites verificar la cantidad de información y que información insertamos en la colección que se encuentra en el nodo principal.



**connSecondary=new Mongo("DESKTOP-KEHD2QP:20001")**

Nos conectamos al nodo #2 con el fin para verificar que la información que insertamos en el nodo primario esta replicando en los demás nodos.



**secondaryTestDB.isMaster()**

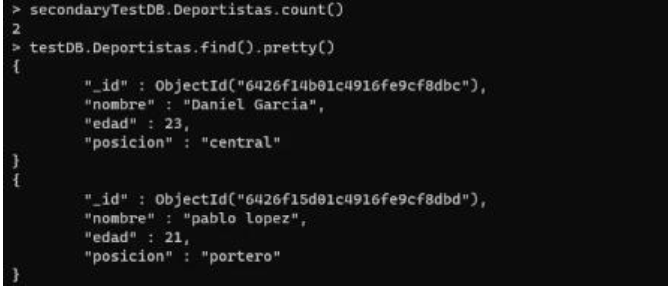
Preguntamos al nodo si es el principal, la respuesta es FALSO



**secondaryTestDB.Deportistas.count()**

**> secondaryTestDB.Deportistas.find().pretty()**

Con estos comandos verificamos que la información que ingresamos en el nodo principal se encuentra replicada en el nodo #2.



**connSecondary=new Mongo("DESKTOP-KEHD2QP:20002")**

Nos conectamos al nodo #3 para verifica si se está replicando la información ingresada a la colección DEPORTITAS.



**secondaryTestDB.Deportistas.count()**

**> secondaryTestDB.Deportistas.find().pretty()**

Con estos comandos verificamos que la información que ingresamos en el nodo principal se encuentra replicada en el nodo #3



**> secondaryTestDB.isMaster()**

Preguntamos al nodo si es el principal, la respuesta es **FALSO**

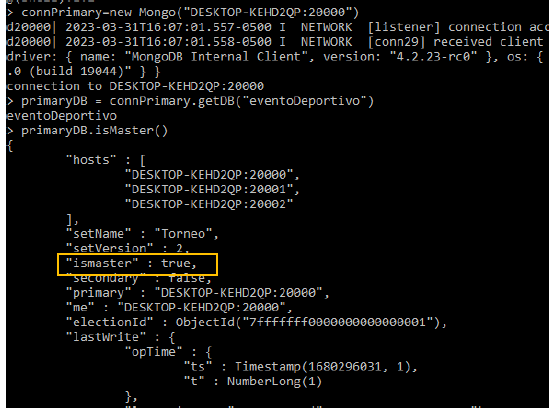


**connPrimary=new Mongo("DESKTOP-KEHD2QP:20000")**

Nos conectamos al nodo #1 para proceder a detenerlo, esto con el fin de realizar una simulación de fallo, con el fin de comprobar que los demás nodos quedan funcionales.

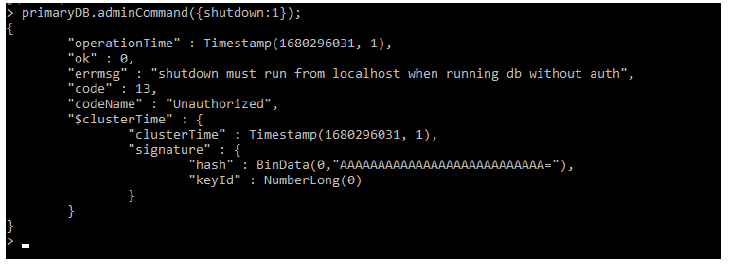
**> testDB.isMaster()**

Le preguntamos al nodo #1 o puerto 20000 si es el nodo principal o maestro, la respuesta es verdadero.



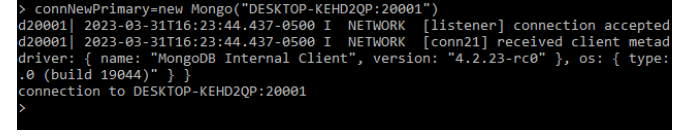
**> primaryDB.admin.Command({shutdown:1});**

Con el comando detenemos el funcionamiento del nodo #1 puerto 20000.



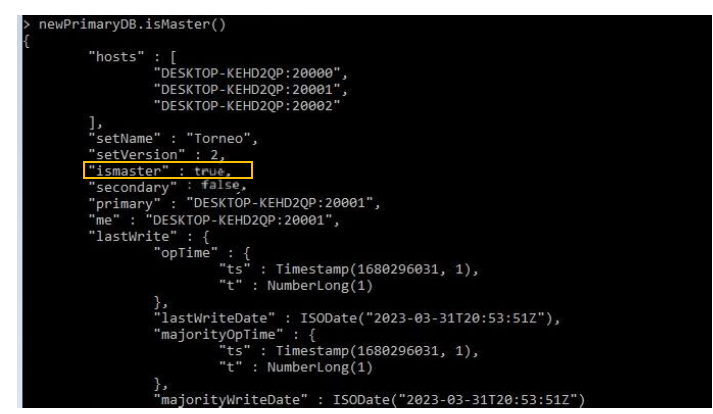
**connNewPrimary=new Mongo("DESKTOP-KEHD2QP:20001")**

Nos conectamos a un nuevo nodo primario, en este caso es el nodo #2 o puerto 20001.



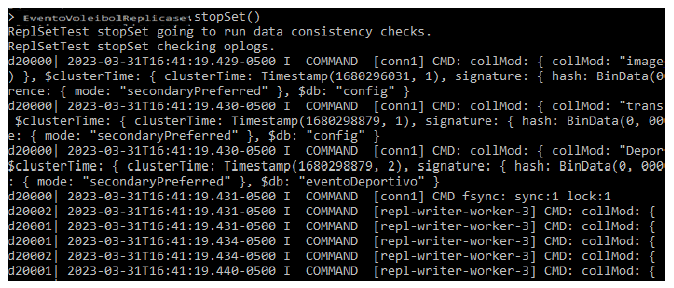
> newPrimaryDB.isMaster()

Consultamos si el nuevo nodo primario, el nodo #2 es el principal o maestro, la respuesta es VERDADERO.

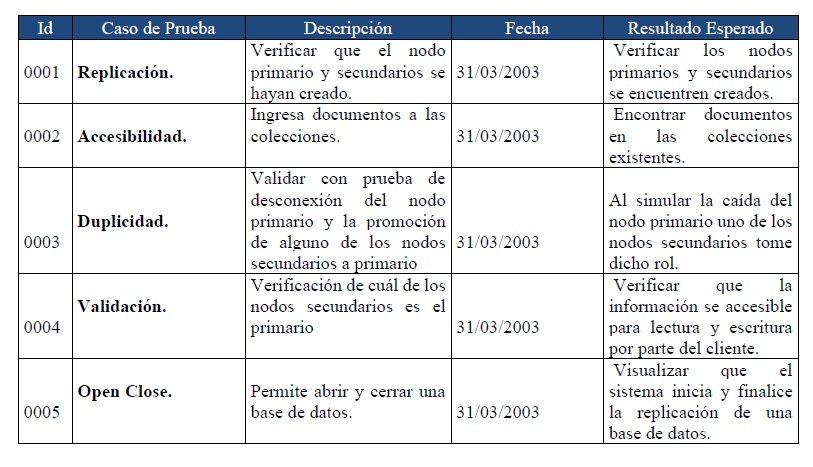


EventoVoleibolReplicaSet.stopSet()

Con el comando detenemos la replicación de la base de datos EVENTO DEPORTIVO



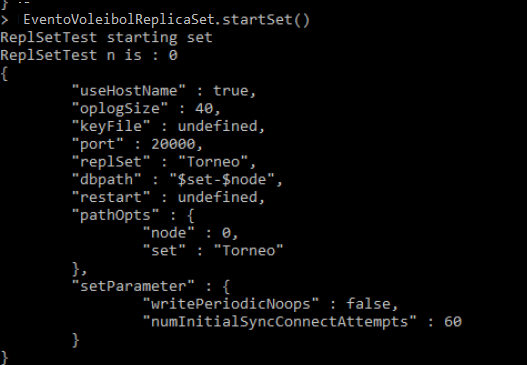
* 1. Especificar a través de un documento de casos de pruebas la verificación del mecanismo de replicación planteado funciona en cuanto a la redundancia y disponibilidad 24x7 para el caso planteado en la primera actividad.



**Ejecución caso de prueba # 0001.**

Verificar los nodos primarios y secundarios se encuentren creados.

Node #1:





**Node # 3:** 



