

ACTIVIDAD N° 1 y 2

YIXCELA HERMINIA CONEJO CONEJO

KRISTIN XIOMARA MUÑOZ RIVERA

CORPORACIÓN UNIVERSITARIA IBEROAMERICANA

BASE DE DATOS AVANZADA

DOCENTE: WILLIAM RUIZ

POPAYÁN

9 DE JUNIO DE 2024

INTRODUCCIÓN

En el ámbito de la gestión de datos, la creación y mantenimiento de una base de datos robusta y eficiente es fundamental para el éxito de cualquier empresa o proyecto. En este contexto, se ha elaborado el presente documento que detalla los requerimientos funcionales y no funcionales de la base de datos Championship. Este sistema de gestión de datos ha sido diseñado específicamente para cubrir las necesidades de una plataforma deportiva en línea, que maneja información crítica sobre campeonatos, equipos, jugadores y partidos.

Los requerimientos funcionales delimitan las características y funcionalidades específicas que debe ofrecer la base de datos para satisfacer las necesidades del negocio. Por otro lado, los requerimientos no funcionales establecen los criterios de calidad y rendimiento que la base de datos debe cumplir para garantizar su eficacia y fiabilidad en todo momento.

Además de definir los requerimientos, este documento también detalla las pruebas no funcionales realizadas para evaluar aspectos clave como el criterio de calidad, la disponibilidad 24x7, la consistencia, la tolerancia a fallos y la estrategia de replicación. Estas pruebas son fundamentales para asegurar que la base de datos Championship cumpla con los más altos estándares de rendimiento y confiabilidad exigidos por su entorno operativo.

A lo largo de este documento, se analizarán en detalle tanto los requerimientos funcionales y no funcionales de la base de datos Championship, así como los resultados de las pruebas realizadas para garantizar su correcto funcionamiento en cualquier situación.

Requerimientos de reglas del partido: Reglas del Torneo Deportivo

El presente documento establece las reglas y el funcionamiento del torneo deportivo que será gestionado a través del sistema de gestión de bases de datos propuesto.

Formato del Torneo:

El torneo se llevará a cabo en formato de liga.

Cada equipo participante se enfrentará a todos los demás equipos en partidos de ida y vuelta.

Se otorgarán puntos según los resultados de los partidos (3 puntos por victoria, 1 punto por empate y 0 puntos por derrota).

Equipos Participantes:

El torneo contará con un número predeterminado de equipos participantes.

1.1. El torneo consistirá en una serie de partidos de fútbol entre equipos conformados por un total de 8 jugadores cada uno.

1.2. De estos 8 jugadores, 5 participarán como titulares en cada partido, mientras que los 3 restantes actuarán como suplentes.

1.3. Durante el transcurso de un partido, los equipos podrán realizar un máximo de 3 sustituciones, permitiendo así la rotación de jugadores y estrategias tácticas.

1.4. Cada partido tendrá una duración estándar de 90 minutos, divididos en dos tiempos de 45 minutos cada uno, con un descanso de 15 minutos entre ambos tiempos.

1.5. El equipo que acumule el mayor número de goles al finalizar el tiempo reglamentario será declarado como el ganador del partido.

1.6. En caso de empate al finalizar el tiempo reglamentario, se procederá a una prórroga de 30 minutos, dividida en dos tiempos de 15 minutos cada uno. Si persiste el empate al término de la prórroga, se definirá al ganador a través de una tanda de penales.

1.7. Se aplicarán las reglas estándar del fútbol en cuanto a faltas, tarjetas y fuera de juego, con el fin de garantizar la equidad y el fair play durante el desarrollo de los partidos.

1.8. En caso de que un jugador reciba dos tarjetas rojas durante un partido, será expulsado del mismo y no podrá continuar participando en ese encuentro.

1.9. Se establece una multa económica para aquellos jugadores que cometan faltas graves durante un partido, cuyo monto dependerá de la gravedad de la falta y será determinado por el comité organizador del torneo.

1.10 Se establece la posibilidad de ganar por W (Walkover) en los siguientes casos:

Si un equipo no se presenta al campo de juego dentro de los primeros 15 minutos del tiempo reglamentario.

Si un equipo abandona el partido de forma voluntaria antes de su conclusión.

Si un equipo es incapaz de continuar el partido debido a la expulsión de todos sus jugadores por acumulación de tarjetas rojas o por cualquier otra razón.

En caso de que un equipo gane por W, se le otorgarán los 3 puntos correspondientes a la victoria, mientras que al equipo perdedor se le asignará una derrota por 3-0.

Insertar, Actualizar y Eliminar Datos:

Los nombres y las identificaciones de los entrenadores deben ser únicos en la base de datos. Esto significa que no puede haber dos entrenadores con el mismo nombre ni con la misma identificación.

Calendario de Partidos:

Se establecerá un calendario de partidos que incluirá las fechas y horarios de cada encuentro.

Los partidos se programará de manera equitativa, garantizando la participación de todos los equipos en diferentes horarios.

Desarrollo de los Partidos:

Se designarán árbitros oficiales para cada encuentro, quienes serán responsables de hacer cumplir las reglas del juego.

Resultados y Tabla de Posiciones:

Se llevará un registro detallado de los resultados de cada partido, incluyendo los goles marcados por cada equipo.

Documento de Requerimientos No Funcionales

El objetivo de este documento es especificar los criterios de calidad necesarios para garantizar la redundancia y disponibilidad 24x7 de la base de datos de un campeonato deportivo, tal como se planteó en la primera actividad.

Criterios de Calidad

- Redundancia

Definición: La redundancia en la base de datos implica que los datos están replicados en múltiples nodos. Esto asegura que si un nodo falla, los datos aún pueden ser accedidos desde otro nodo.

Justificación: La redundancia es esencial para evitar la pérdida de datos en caso de fallos de hardware, problemas de red, o durante el mantenimiento planificado. Tener múltiples copias de los datos en diferentes nodos garantiza que siempre haya una copia disponible, mejorando la resiliencia del sistema y asegurando la continuidad del servicio.

- Disponibilidad 24x7

Definición: La disponibilidad 24x7 significa que la base de datos está accesible en todo momento, sin interrupciones.

Justificación: La disponibilidad continua es crucial para los usuarios y sistemas que dependen de la base de datos, especialmente durante eventos críticos como los partidos del campeonato. Un sistema de base de datos altamente disponible garantiza que las operaciones pueden llevarse a cabo sin interrupciones, lo que es fundamental para la satisfacción del usuario y la operación ininterrumpida de las aplicaciones dependientes.

- Consistencia

Definición: La consistencia asegura que los datos en todos los nodos del Replica Set estén sincronizados y sean precisos.

Justificación: Mantener la consistencia de los datos en todos los nodos es crucial para evitar discrepancias que puedan afectar la integridad y exactitud de la información. Una base de datos consistente asegura que los usuarios siempre vean una versión precisa y actualizada de los datos, lo que es fundamental para la toma de decisiones y la fiabilidad de las aplicaciones que utilizan esta información.

- Tolerancia a Fallos

Definición: La tolerancia a fallos se refiere a la capacidad del sistema de continuar operando adecuadamente en caso de fallo de uno o más de sus componentes.

Justificación: La capacidad de un sistema para manejar fallos sin interrumpir su funcionamiento es crucial para la fiabilidad del servicio. La tolerancia a fallos garantiza que la base de datos siga operativa incluso si un nodo falla, minimizando el tiempo de inactividad y asegurando que los servicios críticos no se vean afectados por fallos inesperados.

- Estrategia de Replicación

Para cumplir con los criterios de calidad mencionados, se implementará una replicación en MongoDB utilizando un Replica Set con tres nodos.

Replica Set: Configuración de un Replica Set en MongoDB con un nodo Primary y dos nodos Secondary.

Configuración de Electores: Asegurar que haya al menos un nodo elegible para convertirse en el Primary en caso de falla del nodo actual.

Sincronización: Configurar la sincronización de datos entre los nodos para mantener la consistencia.

Heartbeat: Asegurar que los nodos envíen señales de heartbeat para detectar fallas rápidamente.

- Disponibilidad y Rendimiento

Monitorización Continua: Implementación de herramientas de monitorización para supervisar la salud de los nodos y el rendimiento del Replica Set.

Alertas en Tiempo Real: Configuración de alertas para notificar a los administradores en caso de problemas de rendimiento o fallos en los nodos.

Mantenimiento Planificado: Realizar mantenimiento planificado durante horarios de menor actividad para minimizar el impacto en la disponibilidad.

- Plan de Implementación

Preparación del Entorno: Configurar los servidores necesarios para los tres nodos.

Instalación de MongoDB: Instalar MongoDB en cada uno de los servidores.

Configuración del Replica Set: Inicializar y configurar el Replica Set.

Pruebas de Failover: Realizar pruebas de failover para asegurar que los nodos secundarios pueden convertirse en primarios cuando sea necesario.

Monitorización y Alertas: Configurar herramientas de monitorización y alertas.

- Plan de Mantenimiento

Actualización de Software: Mantener MongoDB y el sistema operativo actualizados.

Revisión Periódica de la Configuración: Revisar y ajustar la configuración del Replica Set según sea necesario.

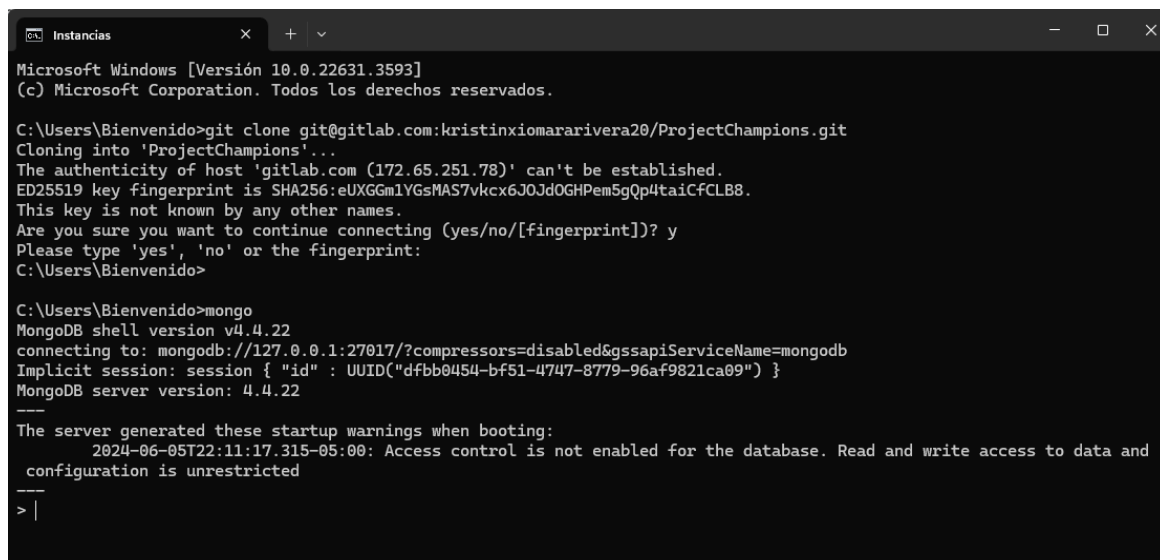
Pruebas de Recuperación de Desastres: Realizar pruebas regulares de recuperación de desastres para asegurar que los datos pueden recuperarse en caso de fallo catastrófico.

2. Preparar el entorno

Replicación 1:

Comenzamos creando el conjunto de réplicas.

Primero que todo, abriremos un mongo shell en el cual empezaremos a realizar los comando y le cambiaremos el nombre del documento a Instancias



```
Microsoft Windows [Versión 10.0.22631.3593]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Bienvenido>git clone git@gitlab.com:kristinxiomararivera20/ProjectChampions.git
Cloning into 'ProjectChampions'...
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ED25519 key fingerprint is SHA256:eUXGGmlYGsMAS7vkcx6JOJd0GHPem5gQp4taiCfCLB8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint:
C:\Users\Bienvenido>

C:\Users\Bienvenido>mongo
MongoDB shell version v4.4.22
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("dfbb0454-bf51-4747-8779-96af9821ca09") }
MongoDB server version: 4.4.22

---
The server generated these startup warnings when booting:
 2024-06-05T22:11:17.315-05:00: Access control is not enabled for the database. Read and write access to data and
configuration is unrestricted
---
> |
```

Ahora crearemos una variable que se llama MieJReplicaSet y le indico con new que se cree un nuevo conjunto de réplicas, a este le debe asignar un nombre el cual será MiReplicaSet y la cantidad de nod


```
> Mireplicaset = new ReplSetTest ({name: "Mireplicaset", nodes: 3})
Starting new replica set Mireplicaset
{
  "kDefaultTimeoutMS" : 600000,
  "getReadConcernMajorityOpTimeOrThrow" : function(conn) {
    const majorityOpTime = _getReadConcernMajorityOpTime(conn);
    if (friendlyEqual(majorityOpTime, {ts: Timestamp(0, 0), t: NumberLong(0)})) {
      throw new Error("readConcern majority optime not available");
    }
    return majorityOpTime;
  },
  "nodeList" : function() {
    var list = [];
    for (var i = 0; i < this.ports.length; i++) {
      list.push(this.host + ":" + this.ports[i]);
    }

    return list;
  },
  "getNodeId" : function(node) {
    if (node.toFixed) {
      return parseInt(node);
    }

    for (var i = 0; i < this.nodes.length; i++) {
      if (this.nodes[i] == node) {
        return i;
      }
    }

    if (node instanceof ObjectId) {
      for (i = 0; i < this.nodes.length; i++) {
```

La información no brinda información de la creación de cada nodo, con su respectivo puerto

```
    }, "waiting for master", timeout);

    return master;
  },
  "name" : "Mireplicaset",
  "useHostName" : true,
  "host" : "LAPTOP-DHVLJMAP",
  "oplogSize" : 40,
  "useSeedList" : false,
  "keyFile" : undefined,
  "protocolVersion" : undefined,
  "waitForKeys" : undefined,
  "seedRandomNumberGenerator" : true,
  "isConfigServer" : undefined,
  "nodeOptions" : {
    "n0" : undefined,
    "n1" : undefined,
    "n2" : undefined
  },
  "nodes" : [ ],
  "ports" : [
    20000,
    20001,
    20002
  ]
}
```

Ahora vamos arrancar los procesos mongod de la réplica

```
> MieiReplicaSet.startSet()
ReplSetTest starting set 'MireplicaSet'
ReplSetTest n is : 0
{
  "useHostName" : true,
  "oplogSize" : 40,
  "keyFile" : undefined,
  "port" : 20000,
  "replSet" : "MireplicaSet",
  "dbpath" : "$set-$node",
  "waitForConnect" : false,
  "restart" : undefined,
  "pathOpts" : {
    "node" : 0,
    "set" : "MireplicaSet"
  },
  "setParameter" : {
    "writePeriodicNoops" : false,
    "numInitialSyncConnectAttempts" : 60
  }
}
ReplSetTest Starting....
Resetting db path '/data/db/MireplicaSet-0'
{"t":{"$date":"2024-06-09T14:58:27.921Z"},"s":"I", "c":"-","id":22810, "ctx":"js" "msg":"shell: Started program" "attr":{"pid":"8876","ar
gv":["mongod.exe","--oplogSize","40","--port","20000","--replSet","MireplicaSet","--dbpath","/data/db/MireplicaSet-0","--setParameter","writePeriodi
cNoops=false","--setParameter","numInitialSyncConnectAttempts=60","--bind_ip","0.0.0.0","--setParameter","disableLogicalSessionCacheRefresh=true","-
setParameter","minNumChunksForSessionsCollection=1","--setParameter","orphanCleanupDelaySecs=1"]}}
Skip waiting to connect to node with pid=8876, port=20000
ReplSetTest start skip waiting for a connection to node 0
ReplSetTest n is : 1
{
  "useHostName" : true,
  "oplogSize" : 40,
  "keyFile" : undefined,
  "port" : 20001,

```

y podemos vsiaulizar cómo se conectan los nodos

```
d20002| {"t":{"$date":"2024-06-09T14:58:31.003-05:00"},"s":"I", "c":"NETWORK",
r":{"port":20002,"ssl":"off"}}
d20002| {"t":{"$date":"2024-06-09T14:58:31.948-05:00"},"s":"I", "c":"NETWORK",
"remote":"127.0.0.1:57392","connectionId":1,"connectionCount":1}}
d20002| {"t":{"$date":"2024-06-09T14:58:31.958-05:00"},"s":"I", "c":"NETWORK",
":"127.0.0.1:57392","client":"conn1","doc":{"application":{"name":"MongoDB Shell
s":{"type":"Windows","name":"Microsoft Windows 10","architecture":"x86_64","vers
ReplSetTest made initial connection to node: connection to LAPTOP-DHVLJMAP:20002
ReplSetTest startSet, nodes: [
  connection to LAPTOP-DHVLJMAP:20000,
  connection to LAPTOP-DHVLJMAP:20001,
  connection to LAPTOP-DHVLJMAP:20002
]
ReplSetTest startSet took 4726ms for 3 nodes.
[
  connection to LAPTOP-DHVLJMAP:20000,
  connection to LAPTOP-DHVLJMAP:20001,
  connection to LAPTOP-DHVLJMAP:20002
]

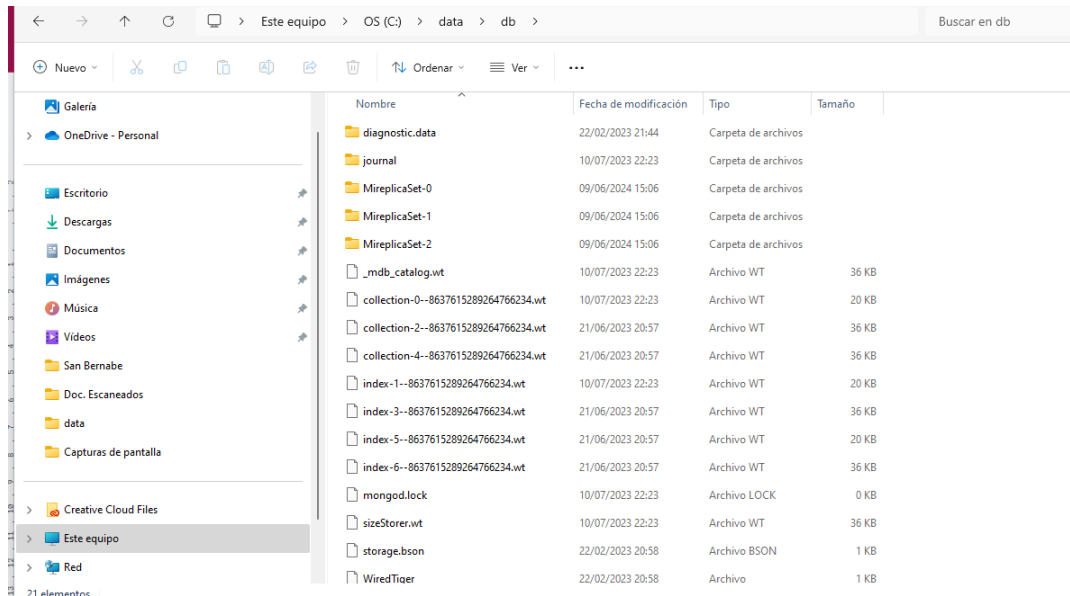
```

Procedemos a arrancar el proceso de replica

```
> MieiReplicaSet.initiate()
{
  "replSetInitiate" : {
    "_id" : "MireplicaSet",
    "protocolVersion" : 1,
    "members" : [
      {
        "_id" : 0,
        "host" : "LAPTOP-DHVLJMAP:20000"
      }
    ]
  }
}
d20000| {"t":{"$date":"2024-06-09T15:02:43.308-05:00"},"s":"I", "c":"REPL", "id":21356, "ctx":"conn1","msg":"replSetInitiate admin command re
ceived from client"}
d20000| {"t":{"$date":"2024-06-09T15:02:43.308-05:00"},"s":"I", "c":"REPL", "id":6015317, "ctx":"conn1","msg":"Setting new configuration state"
,"attr":{"newState":"ConfigInitiating","oldState":"ConfigUninitialized"}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.315-05:00"},"s":"I", "c":"REPL", "id":21357, "ctx":"conn1","msg":"replSetInitiate config object pa
rses ok","attr":{"numMembers":1}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.315-05:00"},"s":"I", "c":"REPL", "id":21251, "ctx":"conn1","msg":"Creating replication oplog","att
r":{"oplogSizeMB":40}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.315-05:00"},"s":"I", "c":"STORAGE", "id":20320, "ctx":"conn1","msg":"createCollection","attr":{"names
pace":"local.oplog.rs","uuidDisposition":"generated","uuid":{"uuid":{"$uuid":"52047e80-caf9-444c-ab6a-eb247e6e2068"},"options":{"capped":true,"size
":41943040,"autoIndexId":false}}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.340-05:00"},"s":"I", "c":"STORAGE", "id":22383, "ctx":"conn1","msg":"The size storer reports that the
oplog contains","attr":{"numRecords":0,"dataSize":0}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.340-05:00"},"s":"I", "c":"STORAGE", "id":22382, "ctx":"conn1","msg":"WiredTiger record store oplog pr
ocessing finished","attr":{"durationMillis":0}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.344-05:00"},"s":"I", "c":"STORAGE", "id":22430, "ctx":"conn1","msg":"WiredTiger message","attr":{"mes
sage":["1717963363:344513][4112:140712610047824], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 52, snapshot
max: 52 snapshot count: 0, oldest timestamp: (0, 0), meta checkpoint timestamp: (0, 0) base write gen: 1"}}
d20000| {"t":{"$date":"2024-06-09T15:02:43.396-05:00"},"s":"I", "c":"STORAGE", "id":20320, "ctx":"conn1","msg":"createCollection","attr":{"names
pace":"local.system.replset","uuidDisposition":"generated","uuid":{"uuid":{"$uuid":"22a9167e-903f-4ad5-911b-a6c210832e3e"},"options":{}}}}

```

Ya podemos verificar las carpetas de mis réplicas que se crearon según la dirección que le indique



Nombre	Fecha de modificación	Tipo	Tamaño
diagnostic.data	22/02/2023 21:44	Carpeta de archivos	
journal	10/07/2023 22:23	Carpeta de archivos	
MireplicaSet-0	09/06/2024 15:06	Carpeta de archivos	
MireplicaSet-1	09/06/2024 15:06	Carpeta de archivos	
MireplicaSet-2	09/06/2024 15:06	Carpeta de archivos	
_mdb_catalog.wt	10/07/2023 22:23	Archivo WT	36 KB
collection-0--8637615289264766234.wt	10/07/2023 22:23	Archivo WT	20 KB
collection-2--8637615289264766234.wt	21/06/2023 20:57	Archivo WT	36 KB
collection-4--8637615289264766234.wt	21/06/2023 20:57	Archivo WT	36 KB
index-1--8637615289264766234.wt	10/07/2023 22:23	Archivo WT	20 KB
index-3--8637615289264766234.wt	21/06/2023 20:57	Archivo WT	36 KB
index-5--8637615289264766234.wt	21/06/2023 20:57	Archivo WT	20 KB
index-6--8637615289264766234.wt	21/06/2023 20:57	Archivo WT	36 KB
mongod.lock	10/07/2023 22:23	Archivo LOCK	0 KB
sizeStorer.wt	10/07/2023 22:23	Archivo WT	36 KB
storage.bson	22/02/2023 20:58	Archivo BSON	1 KB
WiredTiger	22/02/2023 20:58	Archivo	1 KB

Ahora realizaremos el grupo de la réplica, abrimos un nuevo mongo shell y ponemos el nombre de Conjunto réplicas. Empezaré conectado al nodo primario

```
> conn=new Mongo("LAPTOP-DHVLJMAP:20000")
connection to LAPTOP-DHVLJMAP:20000
>
```

Ahora, vamos a almacenar la conexión a una base de datos que se llama “**Championship**”, y con esto nos conectamos a la base de datos

```
> testDB=conn.getDB("Championship")
Championship
>
```

Con esto, ya nos conectamos a el nodo 0 en el puerto 20000 y nos conectamos a un db que se llama **Championship**. Ahora podremos hacer unas cuantas inserciones de ejemplo para poder verificar la interacción en esta forma de replica

```
> testDB.Coaches.insert({"name":"Dilan Santiago","nationality":"Germany","last_name":"Agredo","document":"1026662512"});
WriteResult({ "nInserted" : 1 })
>
```

y verificaremos cuantos datos hay en la colección

```
inserted : 1
> testDB.Coaches.count();
1
>
```

Y podríamos verificar más a fondo esta información con el siguiente comando

```
> testDB.Coaches.find().pretty();
{
  "_id" : ObjectId("66660ee6b0ad0a40cb7246b9"),
  "name" : "Dilan Santiago",
  "nationality" : "Germany",
  "last_name" : "Agredo",
  "document" : "1026662512"
}
```

Esta colección es sumamente importante para poder realizar y cumplir los requerimientos funcionales de la plataforma.

Caso de prueba de Sincronización

Ahora vamos a conectarnos a los nodos esclavos para poder realizar las pruebas de conexión.

```
> connSecondary = new Mongo("LAPTOP-DHVLJMAP:20001")
connection to LAPTOP-DHVLJMAP:20001
>
```

Realizaremos la prueba para que con una variable pueda acceder a la base de datos de "Championship" y podemos visualizar que si se conecta de manera exitosa.

```
> secondaryTestDB = connSecondary.getDB("Championship")
Championship
>
```

Ahora probemos si este es el nodo principal ejecutando el comando:

```
> secondaryTestDB.isMaster();
{
  "topologyVersion" : {
    "processId" : ObjectId("66660a55f3f51e367212506e"),
    "counter" : NumberLong(4)
  },
  "hosts" : [
    "LAPTOP-DHVLJMAP:20000",
    "LAPTOP-DHVLJMAP:20001",
    "LAPTOP-DHVLJMAP:20002"
  ],
  "setName" : "MireplicaSet",
  "setVersion" : 3,
  "ismaster" : false,
  "secondary" : true,
  "primary" : "LAPTOP-DHVLJMAP:20000",
  "me" : "LAPTOP-DHVLJMAP:20001",
  "lastWrite" : {
    "opTime" : {
      "ts" : Timestamp(1717964518, 2),
      "t" : NumberLong(1)
    },
    "lastWriteDate" : ISODate("2024-06-09T20:21:58Z"),
    "majorityOpTime" : {
      "ts" : Timestamp(1717964518, 2),
      "t" : NumberLong(1)
    },
    "majorityWriteDate" : ISODate("2024-06-09T20:21:58Z")
  },
  "maxBsonObjectSize" : 16777216,
  "maxMessageSizeBytes" : 48000000,
}
```

Como podemos ver, nos indica que no, ya que estamos en el nodo esclavo y eso podemos visualizarlo en la variable isMaster que está en false y el campo secondary aparece en true.

- Ahora el segundo caso de prueba, es asegurarnos de que los registros realizados desde el nodo primario, se puedan visualizar desde el nodo secundario

```
> secondaryTestDB.Coaches.count();
uncaught exception: Error: count failed: {
  "topologyVersion" : {
    "processId" : ObjectId("66660a55f3f51e367212506e"),
    "counter" : NumberLong(4)
  },
  "operationTime" : Timestamp(1717964518, 2),
  "ok" : 0,
  "errmsg" : "not master and slaveOk=false",
  "code" : 13435,
  "codeName" : "NotPrimaryNoSecondaryOk",
  "$clusterTime" : {
    "clusterTime" : Timestamp(1717964518, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DBCollection.prototype.count@src/mongo/shell/collection.js:1401:15
@(shell):1:1
> secondaryTestDB = connSecondary.getDB("Championship")
Championship
> secondaryTestDB.Coaches.count();
```

Como no se tienen permisos actualmente de lectura y escritura, lo que vamos a hacer es activarlos con el siguiente comando:

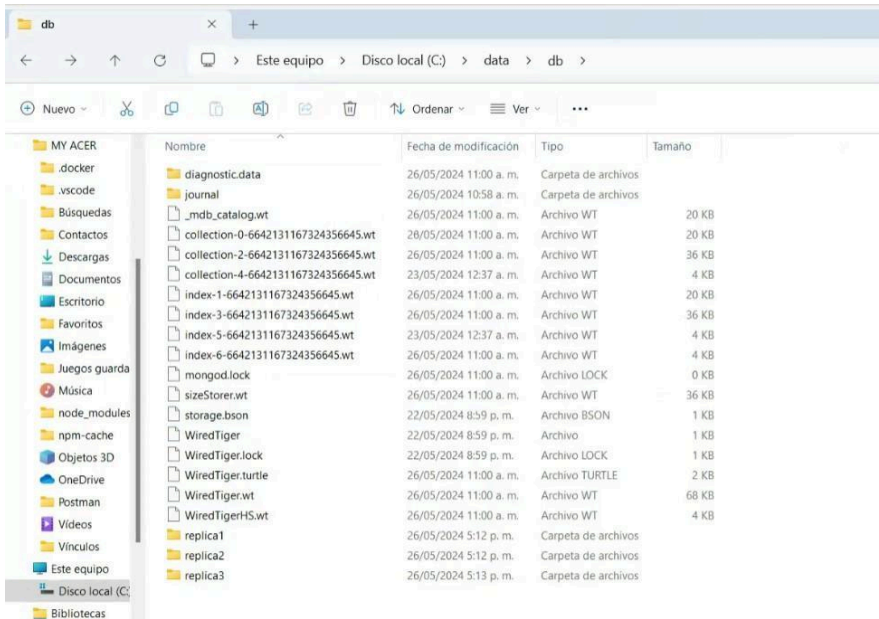
```
> connSecondary.setSecondaryOk()
> secondaryTestDB = connSecondary.getDB("Championship")
Championship
> secondaryTestDB.Coaches.count();
1
>
```

Si verificamos nuevamente el comando nos aparece el registro que se realizó porque ya tiene los permisos

```
> secondaryTestDB.Coaches.find().pretty();
{
  "_id" : ObjectId("66660ee6b0ad0a40cb7246b9"),
  "name" : "Dilan Santiago",
  "nationality" : "Germany",
  "last_name" : "Agredo",
  "document" : "1026662512"
}
>
```

Replicación 2 :

Comenzamos creando las carpetas que van a servirnos para almacenar las réplicas



Ahora abrimos una terminal o cmd para poder dar de alta la primera máquina, corremos la base de datos con mongo, la cual referencia el lugar donde se va a almacenar la base de datos y el puerto.

```

Microsoft Windows [Versión 10.0.22631.3593]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\MY ACER>mongo --replSet RS --dbpath="C:\data\db\replica1" --port 27018
{"t":{"$date":"2024-05-26T17:17:28.038-05:00"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"thread1", "msg":"Automatica
lly disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2024-05-26T17:17:28.039-05:00"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1", "msg":"Initialize
d wire specification", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingIntern
alClient":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"maxWireVersion":21},"isInternalClient"
:true}}}
{"t":{"$date":"2024-05-26T17:17:29.903-05:00"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"thread1", "msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2024-05-26T17:17:29.905-05:00"},"s":"I",  "c":"REPL",       "id":5123008, "ctx":"thread1", "msg":"Successful
ly registered PrimaryOnlyService", "attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDon
ors"}}
{"t":{"$date":"2024-05-26T17:17:29.905-05:00"},"s":"I",  "c":"REPL",       "id":5123008, "ctx":"thread1", "msg":"Successful
ly registered PrimaryOnlyService", "attr":{"service":"TenantMigrationRecipientService","namespace":"config.tenant Migratio
nRecipients"}}
{"t":{"$date":"2024-05-26T17:17:29.905-05:00"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"thread1", "msg":"Multi thre
ading initialized"}
{"t":{"$date":"2024-05-26T17:17:29.906-05:00"},"s":"I",  "c":"TENANT_M",   "id":7091600, "ctx":"thread1", "msg":"Starting T
enantMigrationAccessBlockerRegistry"}
{"t":{"$date":"2024-05-26T17:17:29.908-05:00"},"s":"I",  "c":"CONTROL",  "id":4615611, "ctx":"initandlisten", "msg":"Mong
oDB starting", "attr":{"pid":18660, "port":27018, "dbPath":"C:/data/db/replica1", "architecture":"64-bit", "host":"DESKTOP-R7
E07L0"}}
{"t":{"$date":"2024-05-26T17:17:29.908-05:00"},"s":"I",  "c":"CONTROL",  "id":23398,   "ctx":"initandlisten", "msg":"Targ
et operating system minimum version", "attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2024-05-26T17:17:29.908-05:00"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten", "msg":"Buil
d Info", "attr":{"buildInfo":{"version":"7.0.9", "gitVersion":"3ff3a3925c36ed277cf5eafca5495f2e3728dd67", "modules":[], "all
ocator":"tcmalloc", "environment":{"distmod":"windows", "distarch":"x86_64", "target_arch":"x86_64"}}}}
  
```

Esto mismo realizamos con las otras dos réplicas


```

Microsoft Windows [Versión 10.0.22631.3593]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\MY ACER>mongod --replSet RS --dbpath="C:\data\db\replica2" --port 27019
{"t":{"$date":"2024-05-26T17:18:26.244-05:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"thread1","msg":"Automatica
lly disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2024-05-26T17:18:26.247-05:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"thread1","msg":"Initialize
d wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingIntern
alClient":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"maxWireVersion":21},"isInternalClient
":true}}}}
{"t":{"$date":"2024-05-26T17:18:26.248-05:00"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"thread1","msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2024-05-26T17:18:26.250-05:00"},"s":"I", "c":"REPL", "id":5123808, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDon
ors"}}
{"t":{"$date":"2024-05-26T17:18:26.250-05:00"},"s":"I", "c":"REPL", "id":5123808, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationRecipientService","namespace":"config.tenantMigratio
nRecipients"}}
{"t":{"$date":"2024-05-26T17:18:26.250-05:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"thread1","msg":"Multi thr
eading initialized"}
{"t":{"$date":"2024-05-26T17:18:26.250-05:00"},"s":"I", "c":"TENANT_M", "id":7091600, "ctx":"thread1","msg":"Starting T
enantMigrationAccessBlockerRegistry"}
{"t":{"$date":"2024-05-26T17:18:26.252-05:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten","msg":"Mong
oDB starting","attr":{"pid":6204,"port":27019,"dbPath":"C:\data\db\replica2","architecture":"64-bit","host":"DESKTOP-R7E
07L0"}}
{"t":{"$date":"2024-05-26T17:18:26.252-05:00"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg":"Targ
et operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2024-05-26T17:18:26.252-05:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg":"Build
Info","attr":{"buildInfo":{"version":"7.0.9","gitVersion":"3ff3a3925c36ed277cf5eafca5495f2e3728dd67","modules":[],"all
ocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","distarch":"x86_64"}}}}

```

y el último

```

C:\Users\WY ACER> mongod --replSet --dbpath="C:\data\db\replica3" --port 27028
{"t":{"$date":"2024-05-26T17:19:10.202-05:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"thread1", "msg":"Initialize wire specification", "attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":21},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":21},"outgoing":{"minWireVersion":6,"maxWireVersion":21},"isInternalClient":true}}}}
{"t":{"$date":"2024-05-26T17:19:10.204-05:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"thread1", "msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'", "attr":{"$date":"2024-05-26T17:19:10.205-05:00"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"thread1", "msg":"Implicit TCP FastOpen in use."}}
{"t":{"$date":"2024-05-26T17:19:10.208-05:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"thread1", "msg":"Successful ly registered PrimaryOnlyService", "attr":{"service":"TenantMigrationDonorService", "namespace":"config.tenantMigrationDonors"}}
{"t":{"$date":"2024-05-26T17:19:10.208-05:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"thread1", "msg":"Successful ly registered PrimaryOnlyService", "attr":{"service":"TenantMigrationRecipientService", "namespace":"config.tenantMigrationRecipients"}}
{"t":{"$date":"2024-05-26T17:19:10.208-05:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"thread1", "msg":"Multi threading initialized"}
{"t":{"$date":"2024-05-26T17:19:10.208-05:00"},"s":"I", "c":"TENANT_M", "id":7091600, "ctx":"thread1", "msg":"Starting TenantMigrationAccessBlockerRegistry"}
{"t":{"$date":"2024-05-26T17:19:10.209-05:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten", "msg":"MongoD starting", "attr":{"pid":924, "port":27028, "dbPath":"C:/data/db/replica3", "architecture":"64-bit", "host":"DESKTOP-R7E07L0"}}
{"t":{"$date":"2024-05-26T17:19:10.210-05:00"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten", "msg":"Target operating system minimum version", "attr":{"targetMinOS":"Windows 7/Windows Server 2008 R2"}}
{"t":{"$date":"2024-05-26T17:19:10.210-05:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten", "msg":"Build Info", "attr":{"buildInfo":{"version":"7.0.9", "gitVersion":"3ff3a3925c36ed277cf5eafca5495f2e3728dd67", "modules":[], "allocator":"tcmalloc", "environment":{"distmod":"windows", "distarch":"x86_64", "target_arch":"x86_64"}}}}}

```

Ahora abrimos una nueva terminal y accedemos a nuestro servidor con localhost e iniciamos nuestro proceso de replica

```

C:\Users\MY ACER>mongosh --host localhost --port 27018
Current Mongosh Log ID: 6653b5efb566f19f51cdcdf5
Connecting to: mongodb://localhost:27018/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.6
Using MongoDB: 7.0.9
Using Mongosh: 2.2.6

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

-----
The server generated these startup warnings when booting
2024-05-26T17:17:30.006-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-05-26T17:17:30.007-05:00: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
-----

test> |

```


Procedemos a inicializar las réplicas, con el `rs.initiate` iniciamos la réplicas, le damos un id será con el cual identificamos la réplica, los member serán quienes se van a identificar con el id 0, 1,2 y cada miembro tendrá un puerto

```
test> rs.initiate ({
...  _id : "RS",
...  members: [
...    { _id : 0, host: "localhost:27018" },
...    { _id : 1, host: "localhost:27019" },
...    { _id : 2, host: "localhost:27020" }
...  ]
... })
{ ok: 1 }
RS [direct: other] test>
```

con el `rs.status()` vamos a verificar si se crearon correctamente las réplicas

```
mongosh mongodb://localhost:27018 > use test
> rs.status()
{
  members: [
    {
      _id: 0,
      name: 'localhost:27018',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 554,
      optime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2024-05-26T22:26:35.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T22:26:35.141Z'),
      lastDurableWallTime: ISODate('2024-05-26T22:26:35.141Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: 'Could not find member to sync from',
      electionTime: Timestamp({ t: 1716762334, i: 1 }),
      electionDate: ISODate('2024-05-26T22:25:34.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'localhost:27019',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 78,
      optime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
      optimeDurable: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2024-05-26T22:26:35.000Z'),
      optimeDurableDate: ISODate('2024-05-26T22:26:35.000Z'),
      lastAppliedWallTime: ISODate('2024-05-26T22:26:35.141Z'),
      lastDurableWallTime: ISODate('2024-05-26T22:26:35.141Z'),
      lastHeartbeat: ISODate('2024-05-26T22:26:41.184Z'),
      lastHeartbeatRecv: ISODate('2024-05-26T22:26:40.202Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:27018',

```

```
RS [direct: other] test> rs.status()
{
  set: 'RS',
  date: ISODate('2024-05-26T22:26:42.205Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2024-05-26T22:26:35.141Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2024-05-26T22:26:35.141Z'),
    lastDurableWallTime: ISODate('2024-05-26T22:26:35.141Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1716762375, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-05-26T22:25:34.893Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1716762323, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1716762323, i: 1 }), t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-05-26T22:25:35.002Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-05-26T22:25:35.562Z')
  },
  members: [
    {
      _id: 0,
      name: 'localhost:27018',
      health: 1,

```

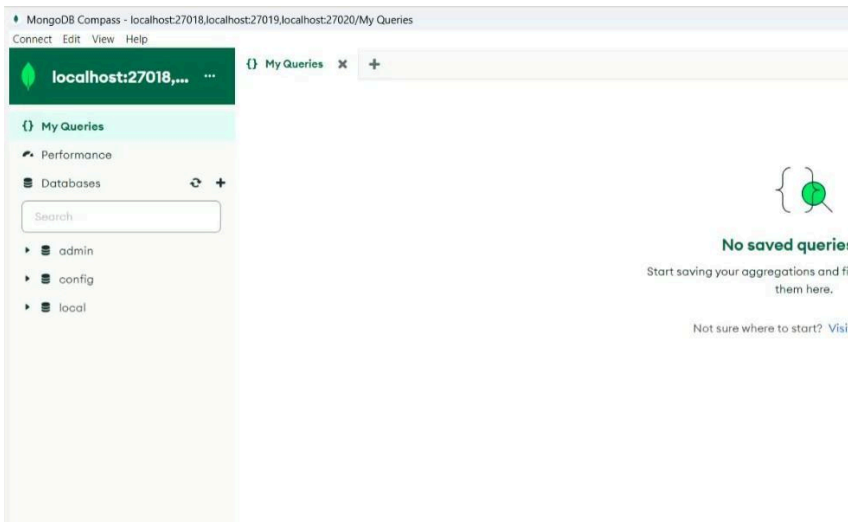
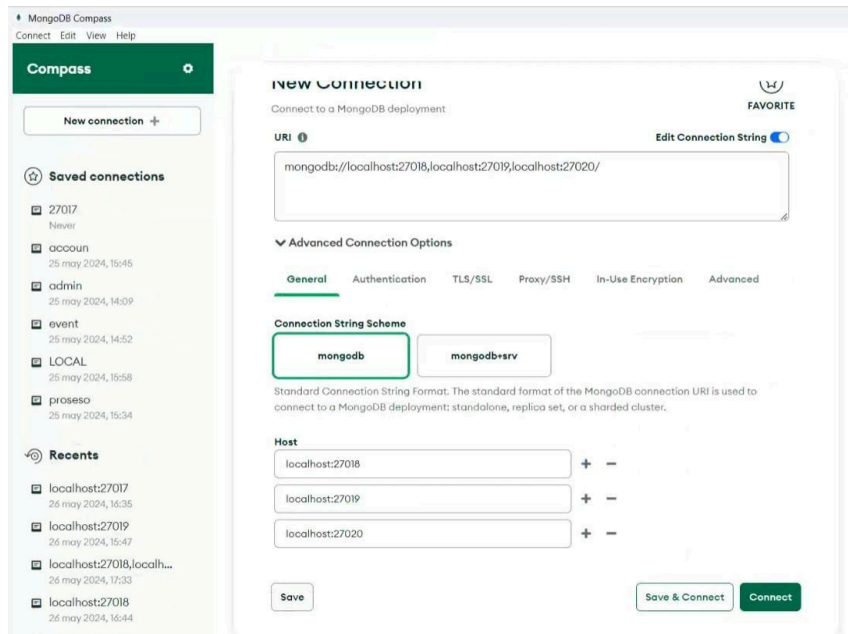
```
},
{
  _id: 2,
  name: 'localhost:27020',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 78,
  optime: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
  optimeDurable: { ts: Timestamp({ t: 1716762395, i: 1 }), t: Long('1') },
  optimeDate: ISODate('2024-05-26T22:26:35.000Z'),
  optimeDurableDate: ISODate('2024-05-26T22:26:35.000Z'),
  lastAppliedWallTime: ISODate('2024-05-26T22:26:35.141Z'),
  lastDurableWallTime: ISODate('2024-05-26T22:26:35.141Z'),
  lastHeartbeat: ISODate('2024-05-26T22:26:41.184Z'),
  lastHeartbeatRecv: ISODate('2024-05-26T22:26:40.249Z'),
  pingMs: Long('0'),
  lastHeartbeatMessage: '',
  syncSourceHost: 'localhost:27018',
  syncSourceId: 0,
  infoMessage: '',
  configVersion: 1,
  configTerm: 1
}
],
ok: 1,
'$clusterTime': {
  clusterTime: Timestamp({ t: 1716762395, i: 1 }),
  signature: {
    hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
    keyId: Long('0')
  }
},
operationTime: Timestamp({ t: 1716762395, i: 1 })
}
RS [direct: primary] test>
```

con esto verificamos que se creó el nodo primario y los nodos secundarios, El miembro con `_id: 0` es el primario (PRIMARY). Está en un estado saludable y ha estado en funcionamiento durante 554 segundos. Los miembros con `_id: 1` y `_id: 2` son secundarios (SECONDARY). También están en un estado saludable y han estado en funcionamiento durante 78 segundos. El miembro con `_id: 1` parece estar sincronizando desde el miembro primario con `_id: 0`, mientras que el miembro con `_id: 2` también está sincronizando desde el mismo miembro primario. El conjunto de réplicas parece estar funcionando bien, y todos los miembros están sincronizados entre sí. El último registro de estado muestra que todas las operaciones se realizaron correctamente.

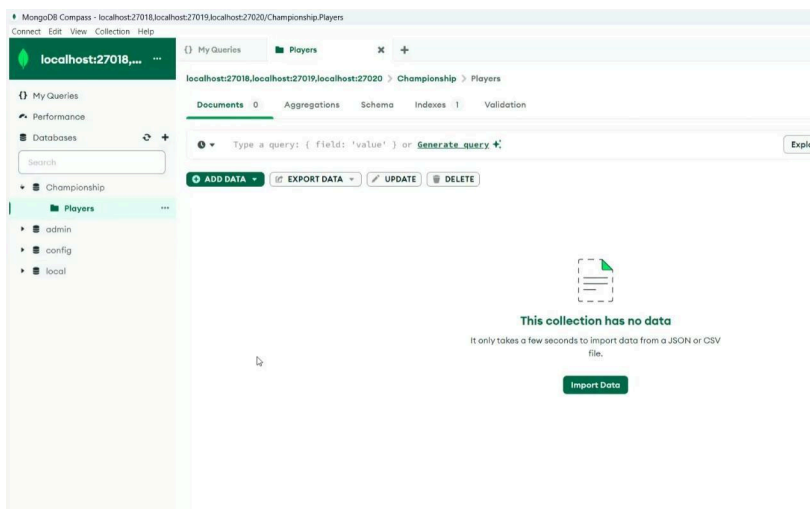
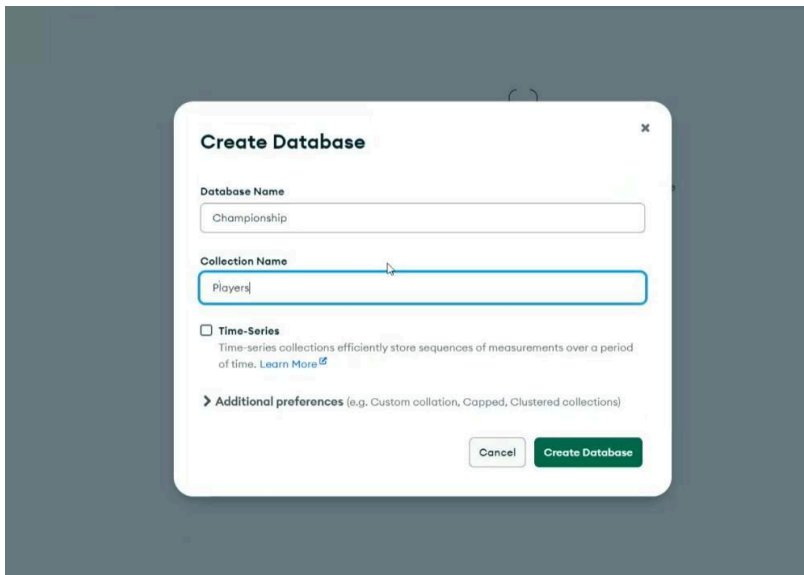
Disponibilidad:

Procedemos a ingresar a mongo compás e iniciamos una nueva conexión con los puertos

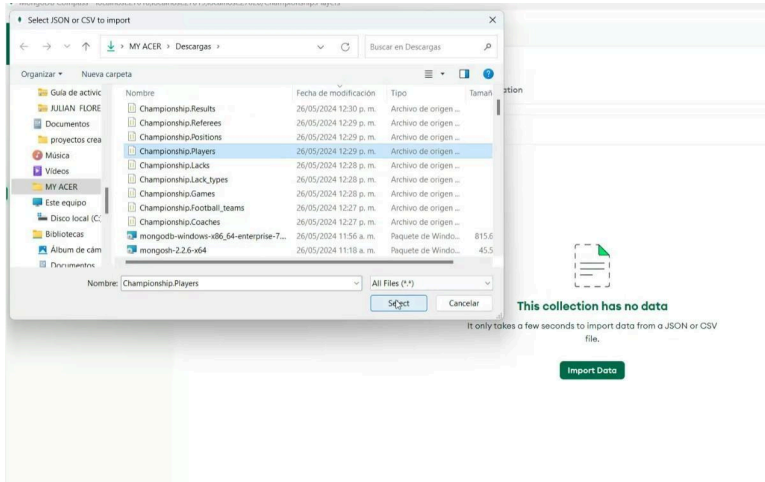
27028, 27029, 27020

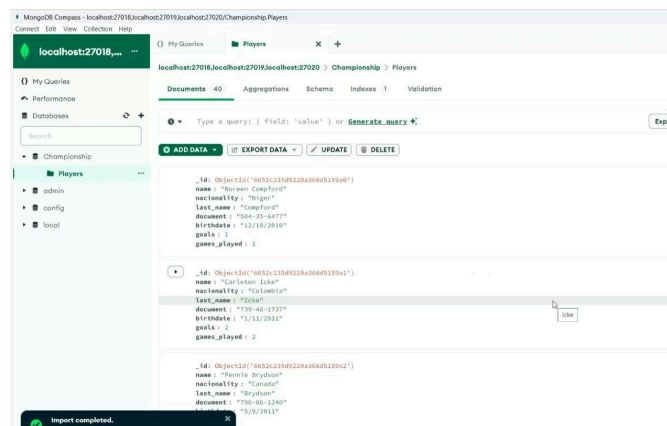
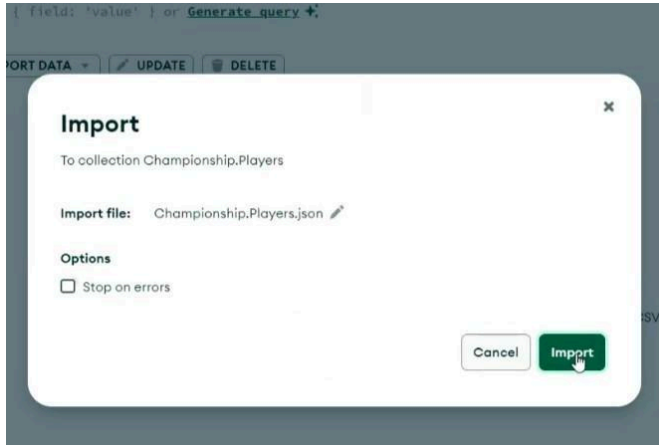


Creamos la base de datos de champiosship, con una colección de jugadores

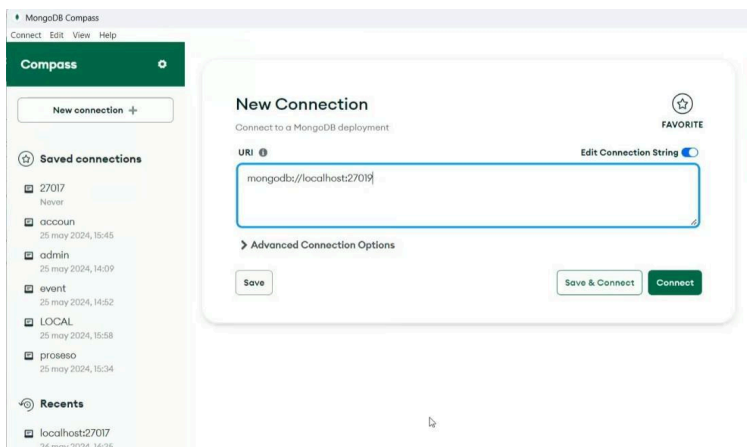


Ahora procedemos a insertar data que ya teníamos exportada en formato json

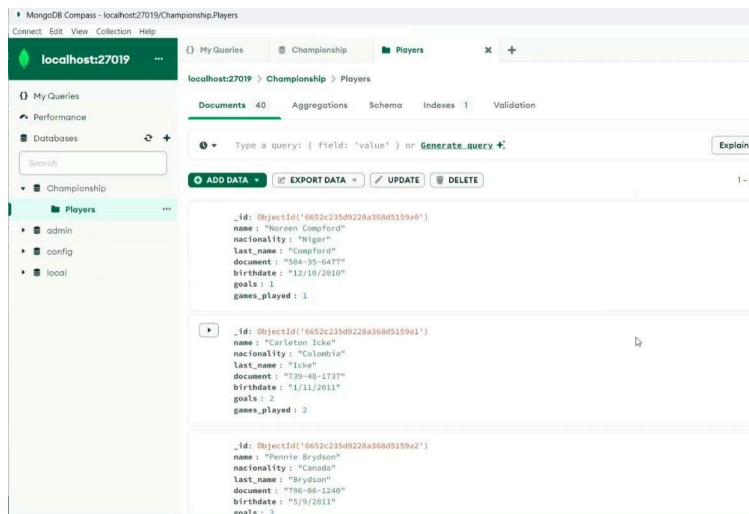




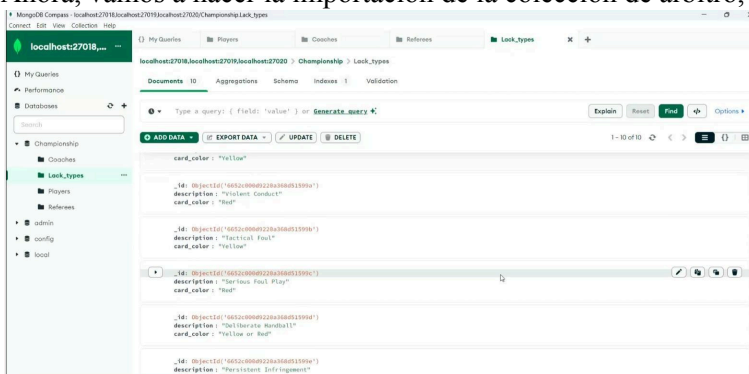
Ahora verificamos si este registro de datos, aparece en otro puerto, abrimos nueva ventana de mongo compass y nos conectamos en el puerto 27029



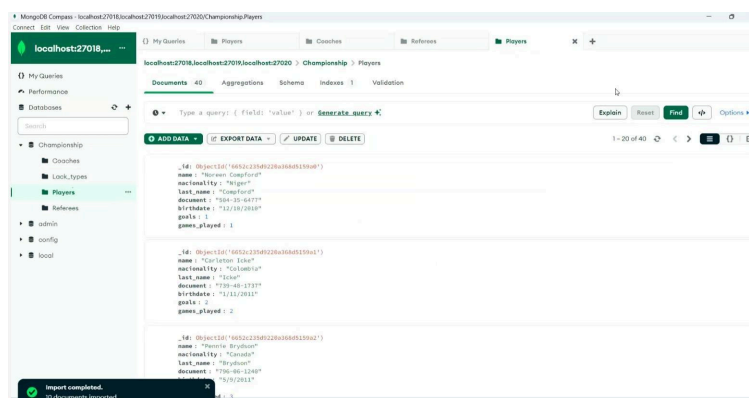
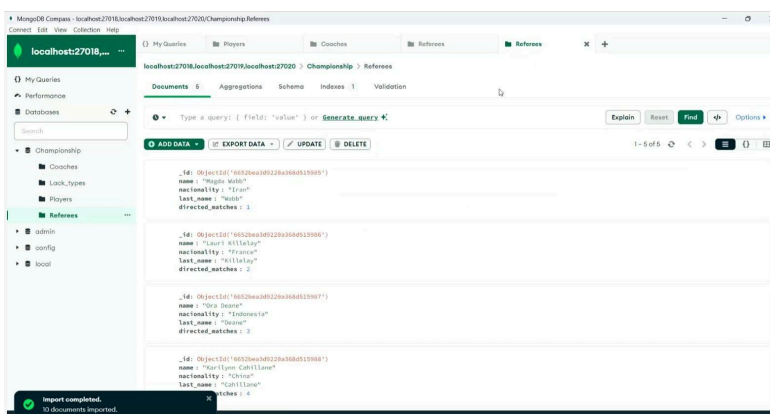
Efectivamente aparece los datos registrados



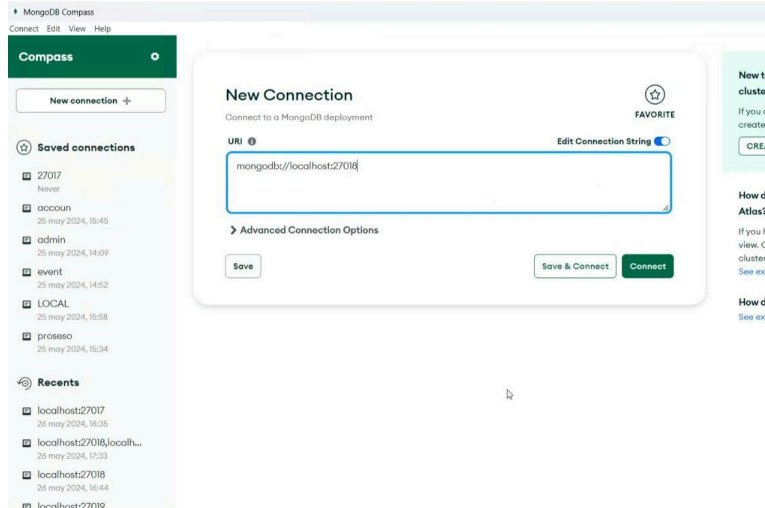
Ahora, vamos a hacer la importación de la colección de árbitro, entrenadores, y tipo de falta, para



verificar si en el nodo maestro se verifican estas nuevas colecciones y registros

Ahora vamos a verificar estos registros en el nodo maestro



Efectivamente están las colecciones y registros

Prueba de rendimiento y Falla

Lo primero que hacemos es acceder a el nodo principal, realizar la conexión con la db y preguntar si es el nodo Master como se presenta a continuación

```
> testDB.isMaster()
{
  "topologyVersion" : {
    "processId" : ObjectId("66660a55afcdecea63346b25"),
    "counter" : NumberLong(8)
  },
  "hosts" : [
    "LAPTOP-DHVLJMAP:20000",
    "LAPTOP-DHVLJMAP:20001",
    "LAPTOP-DHVLJMAP:20002"
  ],
  "setName" : "MireplicaSet",
  "setVersion" : 3,
  "ismaster" : true,
  "secondary" : false,
  "primary" : "LAPTOP-DHVLJMAP:20000",
  "me" : "LAPTOP-DHVLJMAP:20000",
  "electionId" : ObjectId("7fffffff0000000000000001"),
  "lastWrite" : {
    "opTime" : {
      "ts" : Timestamp(1717964518, 2),
      "t" : NumberLong(1)
    },
    "lastWriteDate" : ISODate("2024-06-09T20:21:58Z"),
    "majorityOpTime" : {
      "ts" : Timestamp(1717964518, 2),
      "t" : NumberLong(1)
    },
    "majorityWriteDate" : ISODate("2024-06-09T20:21:58Z")
  },
  "maxBsonObjectSize" : 16777216,
  "maxMessageSizeBytes" : 48000000,
}
```

Lo primero Verificamos que si lo es debido a que la variable isMaster esta en true.

Comenzaremos la simulación caída del sistema con el siguiente comando

```
> primaryDB.adminCommand({shutdown : 1})
uncaught exception: Error: error doing query: failed: network error while attempting to run command 'shutdown' on host 'localhost:20000' :
DB.prototype.runCommand@src/mongo/shell/db.js:169:19
DB.prototype.adminCommand@src/mongo/shell/db.js:187:12
@(shell):1:1
>
```

Ahora procedemos a ver lo que sucede en la consola de Instancias. Podemos visualizar errores de conexión

```
d20002 [{"t":{"$date":"2024-05-21T20:01:11.631-05:00"},"s":"I", "c":"REPL", "id":6015317, "ctx":"ReplCoord-4","msg":"Setting new configuration
attr":{"newState":"ConfigReconfiguring","oldState":"ConfigSteady"}}
d20001 [{"t":{"$date":"2024-05-21T20:01:11.631-05:00"},"s":"I", "c":"REPL", "id":20657, "ctx":"OpLogApplier-0","msg":"IndexBuildsCoordinate
p - this node is stepping up to primary"}
d20001 [{"t":{"$date":"2024-05-21T20:01:11.631-05:00"},"s":"I", "c":"REPL", "id":21331, "ctx":"OpLogApplier-0","msg":"Transition to primary
database writes are now permitted"}
d20002 [{"t":{"$date":"2024-05-21T20:01:11.636-05:00"},"s":"I", "c":"REPL", "id":6015317, "ctx":"ReplCoord-7","msg":"Setting new configuration
attr":{"newState":"ConfigSteady","oldState":"ConfigReconfiguring"}}
d20002 [{"t":{"$date":"2024-05-21T20:01:11.636-05:00"},"s":"I", "c":"REPL", "id":21392, "ctx":"ReplCoord-7","msg":"New replica set config d
t":{"config":{"id":"MireplicaSet","version":3,"term":2,"protocolVersion":1,"writeConcernMajorityJournalDefault":true,"members":[{"_id":0,"host":
:20000,"arbiterOnly":false,"buildIndexes":true,"hidden":false,"priority":1.0,"tags":{"slaveDelay":0,"votes":1},{"_id":1,"host":"Metal2022:20000
Only":false,"buildIndexes":true,"hidden":false,"priority":1.0,"tags":{"slaveDelay":0,"votes":1},{"_id":2,"host":"Metal2022:20002","arbiterOnly":
ldIndexes":true,"hidden":false,"priority":1.0,"tags":{"slaveDelay":0,"votes":1},"settings":{"chainingAllowed":true,"heartbeatIntervalMillis":26
eatTimeoutSecs":10,"electionTimeoutMillis":30000,"catchUpTimeoutMillis":1,"catchUpTakeoverDelayMillis":30000,"getLastErrorModes":{},"getLastError
":{"m":1,"timeout":0},"replicaSetId":{"$oid":"664d3fe916a83aad60948f39"}}}}}]
d20002 [{"t":{"$date":"2024-05-21T20:01:11.636-05:00"},"s":"I", "c":"REPL", "id":21393, "ctx":"ReplCoord-7","msg":"Found self in config","a
tAndPort":"Metal2022:20002"}}
d20002 [{"t":{"$date":"2024-05-21T20:01:11.636-05:00"},"s":"I", "c":"CONNPPOOL", "id":22576, "ctx":"ReplNetwork","msg":"Connecting","attr":{"hos
Metal2022:20000"}}
d20002 [{"t":{"$date":"2024-05-21T20:01:12.592-05:00"},"s":"I", "c":"CONNPPOOL", "id":22572, "ctx":"ReplNetwork","msg":"Dropping all pooled conn
attr":{"hostAndPort":"Metal2022:20000","error":"HostUnreachable: Error connecting to Metal2022:20000 (192.168.1.9:20000) :: caused by :: No se pue
cer una conexiufffdn ya que el equipo de destino denegufffd expresaamente dicha conexiufffdn."}]
d20002 [{"t":{"$date":"2024-05-21T20:01:13.681-05:00"},"s":"I", "c":"CONNPPOOL", "id":22572, "ctx":"ReplNetwork","msg":"Dropping all pooled conn
attr":{"hostAndPort":"Metal2022:20000","error":"HostUnreachable: Error connecting to Metal2022:20000 (192.168.1.9:20000) :: caused by :: No se pue
cer una conexiufffdn ya que el equipo de destino denegufffd expresaamente dicha conexiufffdn."}]
d20001 [{"t":{"$date":"2024-05-21T20:01:13.681-05:00"},"s":"I", "c":"CONNPPOOL", "id":22576, "ctx":"ReplNetwork","msg":"Connecting","attr":{"hos
Metal2022:20000"}}
d20002 [{"t":{"$date":"2024-05-21T20:01:13.681-05:00"},"s":"I", "c":"REPL_HB", "id":23974, "ctx":"ReplCoord-5","msg":"Heartbeat failed after #
","attr":{"target":"Metal2022:20000","maxHeartbeatRetries":2,"error":{"code":6,"codeName":"HostUnreachable"},"errmsg":"Error connecting to Metal202
92.168.1.9:20000) :: caused by :: No se puede establecer una conexiufffdn ya que el equipo de destino denegufffd expresaamente dicha conexiufffd
d20002 [{"t":{"$date":"2024-05-21T20:01:13.681-05:00"},"s":"I", "c":"REPL", "id":21216, "ctx":"ReplCoord-5","msg":"Member is now in state R
ttr":{"hostAndPort":"Metal2022:20000","heartbeatMessage":"Error connecting to Metal2022:20000 (192.168.1.9:20000) :: caused by :: No se puede esta
conexiufffdn ya que el equipo de destino denegufffd expresaamente dicha conexiufffdn."}]
d20002 [{"t":{"$date":"2024-05-21T20:01:14.354-05:00"},"s":"I", "c":"CONNPPOOL", "id":22576, "ctx":"ReplNetwork","msg":"Connecting","attr":{"hos
Metal2022:20000"}}
d20002 [{"t":{"$date":"2024-05-21T20:01:15.366-05:00"},"s":"I", "c":"CONNPPOOL", "id":22576, "ctx":"ReplNetwork","msg":"Connecting","attr":{"hos
Metal2022:20000"}}
d20001 [{"t":{"$date":"2024-05-21T20:01:15.722-05:00"},"s":"I", "c":"CONNPPOOL", "id":22576, "ctx":"ReplNetwork","msg":"Connecting","attr":{"hos
Metal2022:20000"}}]
```

Ahora promovemos para que nodo con secundario se convierta en primario. Promovemos a que se conecte y que acceda a la base de datos de Championship

```
> connSecondary = new Mongo("LAPTOP-DHVLJMAP:20001")
connection to LAPTOP-DHVLJMAP:20001
>
```

```
> secondaryTestDB = connSecondary.getDB("Championship")
Championship
>
```

y verificamos con el comando isMaster para verificar si es el maestro

```
"hosts" : [
  "Metal2022:20000",
  "Metal2022:20001",
  "Metal2022:20002"
],
"setName" : "MireplicaSet",
"setVersion" : 3,
"ismaster" : true,
"secondary" : false,
```

Y como tal podemos verificar que ahora es el nuevo maestro en la variable de isMaster que ahora está en true

Conclusión sobre el Trabajo con Réplicas en MongoDB

La implementación de réplicas en MongoDB ha demostrado ser una estrategia fundamental para garantizar la disponibilidad, la tolerancia a fallos y el rendimiento óptimo de nuestra base de datos.

Durante el desarrollo de este proyecto, pudimos explorar y comprender en profundidad el funcionamiento y los beneficios de las réplicas en entornos distribuidos.

Al utilizar dos métodos diferentes para configurar y gestionar las réplicas, pudimos comparar y contrastar sus ventajas y desventajas. El método propuesto por el profesor nos proporcionó una configuración robusta y estable, mientras que el enfoque alternativo que exploramos con mi compañera nos permitió personalizar y adaptar la configuración según nuestras necesidades específicas.

Durante las pruebas realizadas, todas las réplicas funcionaron de manera excepcional, demostrando su capacidad para mantener la integridad de los datos y proporcionar una alta disponibilidad incluso en situaciones de fallo de uno o más nodos. Las pruebas de conmutación por error y recuperación automática también se llevaron a cabo con éxito, lo que confirma la fiabilidad y la eficacia de nuestro sistema de réplicas.

En resumen, la implementación de réplicas en MongoDB no solo ha fortalecido la infraestructura de nuestra base de datos, sino que también ha mejorado significativamente la escalabilidad, la confiabilidad y la capacidad de recuperación de nuestro sistema. Este proyecto ha sido una experiencia enriquecedora que nos ha permitido adquirir un conocimiento profundo sobre las réplicas en MongoDB y su impacto en entornos empresariales reales.

¡Nos sentimos muy satisfechos con los resultados obtenidos y confiados en la robustez de nuestra solución de réplicas en MongoDB!

Bibliografía

Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC.

<https://elibro.net/es/lc/biblioibero/titulos/58524>.

Introducción a las bases de datos NoSQL usando MongoDB Sarasa, A. (2016) Editorial UOC.

disponible en: <https://elibro.net/es/lc/biblioibero/titulos/58524>.

Capítulo 1 (Introducción a las bases de datos NoSQL) y del Capítulo 2 (Conceptos Básicos) del libro de Sarasa, A. (2016) Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC, disponible en: <https://elibro.net/es/lc/biblioibero/titulos/58524>