

**Actividad 2 – Conceptos y comando básicos de la replicación en bases de datos  
NoSQL**

Oscar Francisco Rodríguez Tobaría- Cod 100108446

Facultad de Ingeniería de Software Semestre VI, Corporación Universitaria Iberoamericana

Bases de Datos Avanzadas

Docente William Ruiz

Mayo 26 – 2024

***Documento de requerimientos NO funcionales para un sistema de gestión de torneos de Ping Pong***

Este documento detalla los requerimientos no funcionales para el sistema de gestión de torneos de ping pong. Los criterios de calidad incluidos son redundancia, disponibilidad 24x7, rendimiento, seguridad, usabilidad y mantenibilidad, entre otros. Estos criterios asegurarán que el sistema funcione de manera eficiente y continua, proporcionando una experiencia confiable y satisfactoria para todos los usuarios.

**Requerimientos no funcionales:**

***Redundancia***

**Descripción:** El sistema debe contar con mecanismos de redundancia para asegurar que no haya interrupciones en el servicio.

**Justificación:** La redundancia es crítica para evitar la pérdida de datos y minimizar el tiempo de inactividad.

**Criterios de calidad:**

**Replicación de bases de datos:** Implementar replicación de bases de datos en tiempo real en múltiples ubicaciones geográficas.

**Failover Automático:** Configurar sistemas de failover automático que tomen el control en caso de fallo del servidor principal.

**Backups:** Realizar copias de seguridad automáticas incrementales cada hora y copias completas diariamente.

### ***Disponibilidad 24x7***

**Descripción:** El sistema debe estar disponible y operativo las 24 horas del día, los 7 días de la semana.

**Justificación:** Los usuarios deben poder acceder al sistema en cualquier momento, especialmente durante torneos internacionales con participantes de diferentes zonas horarias.

### **Criterios de calidad:**

**Uptime:** Garantizar un tiempo de actividad del 99.9% mensualmente.

**Monitoreo:** Implementar monitoreo en tiempo real del sistema con alertas para el personal técnico ante cualquier fallo potencial.

**Mantenimiento Programado:** Planificar y comunicar el mantenimiento programado durante horas de menor actividad para minimizar el impacto en los usuarios.

### ***Rendimiento***

**Descripción:** El sistema debe responder de manera rápida y eficiente bajo diversas cargas de trabajo

**Justificación:** Un rendimiento óptimo es esencial para una experiencia de usuario positiva y para el manejo eficiente de torneos en tiempo real.

**Criterios de calidad:**

**Tiempo de respuesta:** El tiempo de respuesta de las páginas web y de las operaciones críticas no debe exceder los 2 segundos bajo carga normal.

**Escalabilidad:** El sistema debe ser escalable para manejar picos de tráfico, especialmente durante eventos importantes

**Optimización:** Optimizar el código y las consultas de base de datos para mejorar la velocidad y eficiencia.

***Seguridad***

**Descripción:** El sistema debe proteger la información de los usuarios y la integridad del torneo.

**Justificación:** La seguridad es vital para proteger datos sensibles y mantener la confianza de los usuarios.

**Criterios de Calidad:**

**Autenticación y Autorización:** Implementar autenticación multifactor (MFA) y controles de acceso basados en roles.

**Encriptación:** Utilizar encriptación SSL/TLS para la transmisión de datos y encriptación de datos sensibles en reposo.

**Auditoría y registro:** Mantener registros de auditoría detallados de todas las actividades del sistema y accesos a datos.

### ***Usabilidad***

**Descripción:** El sistema debe ser fácil de usar e intuitivo para todos los usuarios, independientemente de su habilidad técnica.

**Justificación:** Una buena usabilidad mejora la satisfacción del usuario y reduce la necesidad de soporte.

#### **Criterios de Calidad:**

**Interfaz de Usuario:** Diseñar una interfaz de usuario clara y amigable con navegación intuitiva.

**Documentación y ayuda:** Proveer documentación completa y accesible, junto con tutoriales y asistencia en línea.

**Accesibilidad:** Asegurar que el sistema cumpla con los estándares de accesibilidad (WCAG 2.1).

### ***Mantenibilidad***

**Descripción:** El sistema debe ser fácil de mantener y actualizar.

**Justificación:** Una buena mantenibilidad reduce el tiempo y el costo asociados con las actualizaciones y las correcciones de errores.

#### **Criterios de Calidad:**

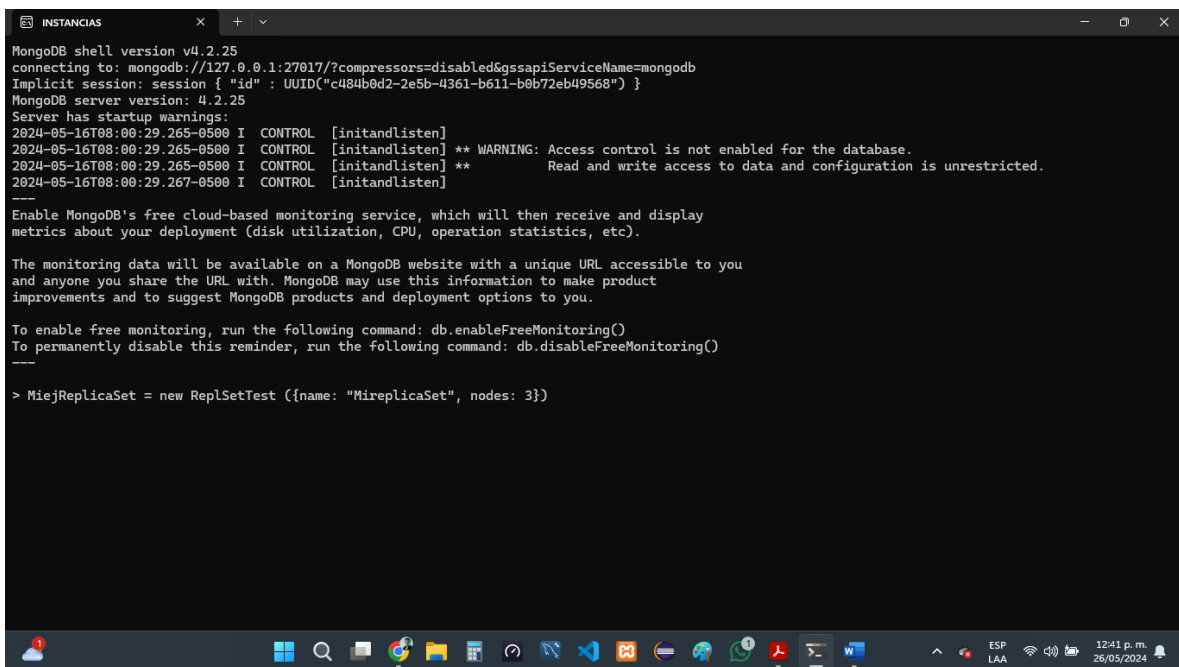
**Modularidad:** Utilizar una arquitectura modular que permita actualizaciones y mantenimiento sin afectar al sistema completo.

**Documentación del código:** Mantener una documentación de código clara y detallada para facilitar el trabajo de los desarrolladores.

**Testing Automatizado:** Implementar pruebas automatizadas para asegurar que las actualizaciones no introduzcan nuevos errores.

Estos requerimientos no funcionales son esenciales para asegurar que el sistema de gestión de torneos de ping pong sea robusto, confiable y eficiente. La implementación adecuada de estos criterios de calidad garantizará una experiencia positiva y continua para todos los usuarios del sistema.

### *Soporte con capturas de pantallas del proceso de creación de nodos*



```
INSTANCIAS
MongoDB shell version v4.2.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c484b0d2-2e5b-4361-b611-b0b72eb49568") }
MongoDB server version: 4.2.25
Server has startup warnings:
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten]
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2024-05-16T08:00:29.267-0500 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> MijReplicaSet = new ReplSetTest ({name: "MireplicaSet", nodes: 3})
```

```
INSTANCIAS
"usesBridge" : function() {
  return _useBridge;
},
"waitForState" : function(node, state, timeout, reconnectNode) {
  _waitForIndicator(node, state, "state", timeout, reconnectNode);
},
"waitForMaster" : function(timeout) {
  var master;
  assert.soonNoExcept(function() {
    return (master = self.getPrimary());
  }, "waiting for master", timeout);

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

```
INSTANCIAS
t-2/diagnostic.data'
d20002| 2024-05-26T12:50:08.167-0500 I STORAGE [initandlisten] createCollection: local.replset.oplogTruncateAfterPoint with generated UUID: 7d4d02
0a-398b-4f39-b61e-cb1d8acf873f and options: {}
d20002| 2024-05-26T12:50:08.177-0500 I INDEX [initandlisten] index build: done building index _id_ on ns local.replset.oplogTruncateAfterPoint
d20002| 2024-05-26T12:50:08.177-0500 I STORAGE [initandlisten] createCollection: local.replset.minvalid with generated UUID: c5bcab61-5a5d-4c3b-ab
32-807147084e67 and options: {}
d20002| 2024-05-26T12:50:08.189-0500 I INDEX [initandlisten] index build: done building index _id_ on ns local.replset.minvalid
d20002| 2024-05-26T12:50:08.189-0500 I STORAGE [initandlisten] createCollection: local.replset.election with generated UUID: 6473d53c-6023-4eed-be
e7-e00668bdf8c and options: {}
d20002| 2024-05-26T12:50:08.202-0500 I INDEX [initandlisten] index build: done building index _id_ on ns local.replset.election
d20002| 2024-05-26T12:50:08.202-0500 I REPL [initandlisten] Did not find local initialized voted for document at startup.
d20002| 2024-05-26T12:50:08.202-0500 I REPL [initandlisten] Did not find local Rollback ID document at startup. Creating one.
d20002| 2024-05-26T12:50:08.202-0500 I STORAGE [initandlisten] createCollection: local.system.rollback.id with generated UUID: 2d720de1-3ea7-445f-
93cd-fe2957f9f0d8 and options: {}
d20002| 2024-05-26T12:50:08.212-0500 I INDEX [initandlisten] index build: done building index _id_ on ns local.system.rollback.id
d20002| 2024-05-26T12:50:08.213-0500 I REPL [initandlisten] Initialized the rollback ID to 1
d20002| 2024-05-26T12:50:08.213-0500 I REPL [initandlisten] Did not find local replica set configuration document at startup; NoMatchingDocume
nt: Did not find replica set configuration document in local.system.replset
d20002| 2024-05-26T12:50:08.214-0500 I NETWORK [listener] Listening on 0.0.0.0
d20002| 2024-05-26T12:50:08.214-0500 I NETWORK [listener] waiting for connections on port 20002
d20002| 2024-05-26T12:50:08.398-0500 I NETWORK [listener] connection accepted from 127.0.0.1:60877 #1 (1 connection now open)
d20002| 2024-05-26T12:50:08.398-0500 I NETWORK [conn1] received client metadata from 127.0.0.1:60877 conn1: { application: { name: "MongoDB Shell"
}, driver: { name: "MongoDB Internal Client", version: "4.2.25" }, os: { type: "Windows", name: "Microsoft Windows 10", architecture: "x86_64", ver
sion: "10.0 (build 22631)" } }
[
  connection to LAPTOP-EUSHEITM:20000,
  connection to LAPTOP-EUSHEITM:20001,
  connection to LAPTOP-EUSHEITM:20002
]
[
  connection to LAPTOP-EUSHEITM:20000,
  connection to LAPTOP-EUSHEITM:20001,
  connection to LAPTOP-EUSHEITM:20002
]
> |
```

## Imagen Nodo “Cero”

```
INSTANCIAS
ReplSetTest starting set
ReplSetTest n is : 0
{
  "useHostName" : true,
  "oplogSize" : 40,
  "keyFile" : undefined,
  "port" : 20000,
  "replSet" : "MireplicaSet",
  "dbpath" : "$set-$node",
  "restart" : undefined,
  "pathOpts" : {
    "node" : 0,
    "set" : "MireplicaSet"
  },
  "setParameter" : {
    "writePeriodicNoops" : false,
    "numInitialSyncConnectAttempts" : 60
  }
}
ReplSetTest Starting....
Resetting db path '/data/db/MireplicaSet-0'
2024-05-26T12:50:03.253-0500 I - [js] shell: started program (sh20788): C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe --oplogSize 40 -
-port 20000 --replSet MireplicaSet --dbpath /data/db/MireplicaSet-0 --setParameter writePeriodicNoops=false --setParameter numInitialSyncConnectAtte
mpts=60 --bind_ip 0.0.0.0 --setParameter enableTestCommands=1 --setParameter disableLogicalSessionCacheRefresh=true --setParameter minNumChunksForSe
ssionsCollection=1 --setParameter orphanCleanupDelaySecs=1
d20000| 2024-05-26T12:50:03.354-0500 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'non
e'
d20000| 2024-05-26T12:50:03.870-0500 W ASIO [main] No TransportLayer configured during NetworkInterface startup
d20000| 2024-05-26T12:50:03.875-0500 I CONTROL [initandlisten] MongoDB starting : pid=20788 port=20000 dbpath=/data/db/MireplicaSet-0 64-bit host=
LAPTOP-EUSHEITM
d20000| 2024-05-26T12:50:03.875-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
d20000| 2024-05-26T12:50:03.875-0500 I CONTROL [initandlisten] db version v4.2.25
d20000| 2024-05-26T12:50:03.875-0500 I CONTROL [initandlisten] git version: 41b59c2bfb5121e66f18cc3ef40055a1b5fb6c2e
d20000| 2024-05-26T12:50:03.875-0500 I CONTROL [initandlisten] allocator: tcmalloc
d20000| 2024-05-26T12:50:03.875-0500 I CONTROL [initandlisten] modules: none
```

## Imagen Nodo 1

```
INSTANCIAS
}, driver: { name: "MongoDB Internal Client", version: "4.2.25" }, os: { type: "Windows", name: "Microsoft Windows 10", architecture: "x86_64", ver
sion: "10.0 (build 22631)" } }
[ connection to LAPTOP-EUSHEITM:20000 ]
ReplSetTest n is : 1
{
  "useHostName" : true,
  "oplogSize" : 40,
  "keyFile" : undefined,
  "port" : 20001,
  "replSet" : "MireplicaSet",
  "dbpath" : "$set-$node",
  "restart" : undefined,
  "pathOpts" : {
    "node" : 1,
    "set" : "MireplicaSet"
  },
  "setParameter" : {
    "writePeriodicNoops" : false,
    "numInitialSyncConnectAttempts" : 60
  }
}
ReplSetTest Starting....
Resetting db path '/data/db/MireplicaSet-1'
2024-05-26T12:50:05.302-0500 I - [js] shell: started program (sh5008): C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe --oplogSize 40 --
port 20001 --replSet MireplicaSet --dbpath /data/db/MireplicaSet-1 --setParameter writePeriodicNoops=false --setParameter numInitialSyncConnectAtte
mpts=60 --bind_ip 0.0.0.0 --setParameter enableTestCommands=1 --setParameter disableLogicalSessionCacheRefresh=true --setParameter minNumChunksForSe
ssionsCollection=1 --setParameter orphanCleanupDelaySecs=1
d20001| 2024-05-26T12:50:05.339-0500 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'non
e'
d20001| 2024-05-26T12:50:05.857-0500 W ASIO [main] No TransportLayer configured during NetworkInterface startup
d20001| 2024-05-26T12:50:05.859-0500 I CONTROL [initandlisten] MongoDB starting : pid=5008 port=20001 dbpath=/data/db/MireplicaSet-1 64-bit host=L
APTOP-EUSHEITM
d20001| 2024-05-26T12:50:05.859-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
d20001| 2024-05-26T12:50:05.859-0500 I CONTROL [initandlisten] db version v4.2.25
d20001| 2024-05-26T12:50:05.859-0500 I CONTROL [initandlisten] git version: 41b59c2bfb5121e66f18cc3ef40055a1b5fb6c2e
```



## Imagen nodo 2

```
INSTANCIAS
sion: "10.0 (build 22631)" } }
[ connection to LAPTOP-EUSHEITM:20000, connection to LAPTOP-EUSHEITM:20001 ]
ReplSetTest n is : 2
{
  "useHostName" : true,
  "oplogSize" : 40,
  "keyFile" : undefined,
  "port" : 20002,
  "replSet" : "MireplicaSet",
  "dbpath" : "$set-$node",
  "restart" : undefined,
  "pathOpts" : {
    "node" : 2,
    "set" : "MireplicaSet"
  },
  "setParameter" : {
    "writePeriodicNoops" : false,
    "numInitialSyncConnectAttempts" : 60
  }
}
ReplSetTest Starting....
Resetting db path '/data/db/MireplicaSet-2'
2024-05-26T12:50:06.853-0500 I - [js] shell: started program (sh20680): C:\Program Files\MongoDB\Server\4.2\bin\mongod.exe --oplogSize 40 -
-port 20002 --replSet MireplicaSet --dbpath /data/db/MireplicaSet-2 --setParameter writePeriodicNoops=false --setParameter numInitialSyncConnectAtte
mpts=60 --bind_ip 0.0.0.0 --setParameter enableTestCommands=1 --setParameter disableLogicalSessionCacheRefresh=true --setParameter minNumChunksForSe
ssionsCollection=1 --setParameter orphanCleanupDelaySecs=1
d20002| 2024-05-26T12:50:07.378-0500 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'non
e'
d20002| 2024-05-26T12:50:07.383-0500 W ASIO [main] No TransportLayer configured during NetworkInterface startup
d20002| 2024-05-26T12:50:07.386-0500 I CONTROL [initandlisten] MongoDB starting : pid=20680 port=20002 dbpath=/data/db/MireplicaSet-2 64-bit host=
LAPTOP-EUSHEITM
d20002| 2024-05-26T12:50:07.386-0500 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
d20002| 2024-05-26T12:50:07.386-0500 I CONTROL [initandlisten] db version v4.2.25
d20002| 2024-05-26T12:50:07.386-0500 I CONTROL [initandlisten] git version: 41b59c2bfb5121e66f18cc3ef40055alb5fb6c2e
d20002| 2024-05-26T12:50:07.386-0500 I CONTROL [initandlisten] allocator: tcmalloc
```

```
INSTANCIAS
"LAPTOP-EUSHEITM:20000",
"LAPTOP-EUSHEITM:20001",
"LAPTOP-EUSHEITM:20002"
]

[jsTest] ----
[jsTest] ReplSetTest stepUp: Stepping up LAPTOP-EUSHEITM:20000
[jsTest] ----

ReplSetTest awaitReplication: starting: optime for primary, LAPTOP-EUSHEITM:20000, is { "ts" : Timestamp(1716746467, 1), "t" : NumberLong(1) }
ReplSetTest awaitReplication: checking secondaries against latest primary optime { "ts" : Timestamp(1716746467, 1), "t" : NumberLong(1) }
ReplSetTest awaitReplication: checking secondary #0: LAPTOP-EUSHEITM:20001
ReplSetTest awaitReplication: secondary #0, LAPTOP-EUSHEITM:20001, is synced
ReplSetTest awaitReplication: checking secondary #1: LAPTOP-EUSHEITM:20002
ReplSetTest awaitReplication: secondary #1, LAPTOP-EUSHEITM:20002, is synced
ReplSetTest awaitReplication: finished: all 2 secondaries synced at optime { "ts" : Timestamp(1716746467, 1), "t" : NumberLong(1) }
d20000| 2024-05-26T13:01:07.300-0500 I COMMAND [conn1] Received replSetStepUp request
d20000| 2024-05-26T13:01:07.300-0500 I ELECTION [conn1] Not starting an election for a replSetStepUp request, since we are not electable due to: No
t standing for election again; already primary
AwaitNodesAgreeOnPrimary: Waiting for nodes to agree on any primary.
AwaitNodesAgreeOnPrimary: Nodes agreed on primary LAPTOP-EUSHEITM:20000

[jsTest] ----
[jsTest] ReplSetTest stepUp: Finished stepping up LAPTOP-EUSHEITM:20000
[jsTest] ----

> d20000| 2024-05-26T13:11:07.185-0500 I QUERY [clientcursormon] Cursor id 3339532583471557219 timed out, idle since 2024-05-26T13:01:06.286-050
0
d20000| 2024-05-26T13:11:07.186-0500 I QUERY [clientcursormon] Cursor id 7488196934926482485 timed out, idle since 2024-05-26T13:01:06.286-0500
> |
```

## *Creo otro Mondo Shell nombrándolo “ConjuntodeReplicas”*

```
ConjuntodeReplicas
MongoDB shell version v4.2.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("01ffa844-f40d-4c48-8203-9d650e87d4d1") }
MongoDB server version: 4.2.25
Server has startup warnings:
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten]
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2024-05-16T08:00:29.267-0500 I CONTROL [initandlisten]
-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
> |
```

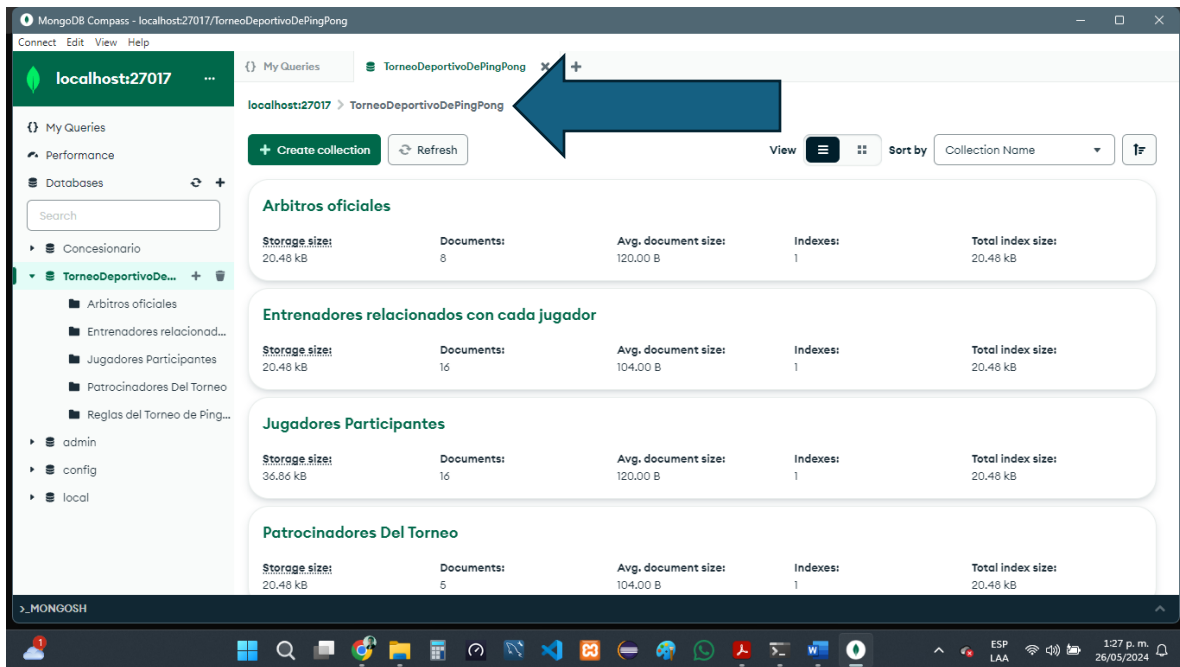
## *Me arrija error al escribir el comando que usted utilizo en clase y que esta en el documento PDF que usted compartió*

```
ConjuntodeReplicas
MongoDB shell version v4.2.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("01ffa844-f40d-4c48-8203-9d650e87d4d1") }
MongoDB server version: 4.2.25
Server has startup warnings:
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten]
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] **      Read and write access to data and configuration is unrestricted.
2024-05-16T08:00:29.267-0500 I CONTROL [initandlisten]
-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

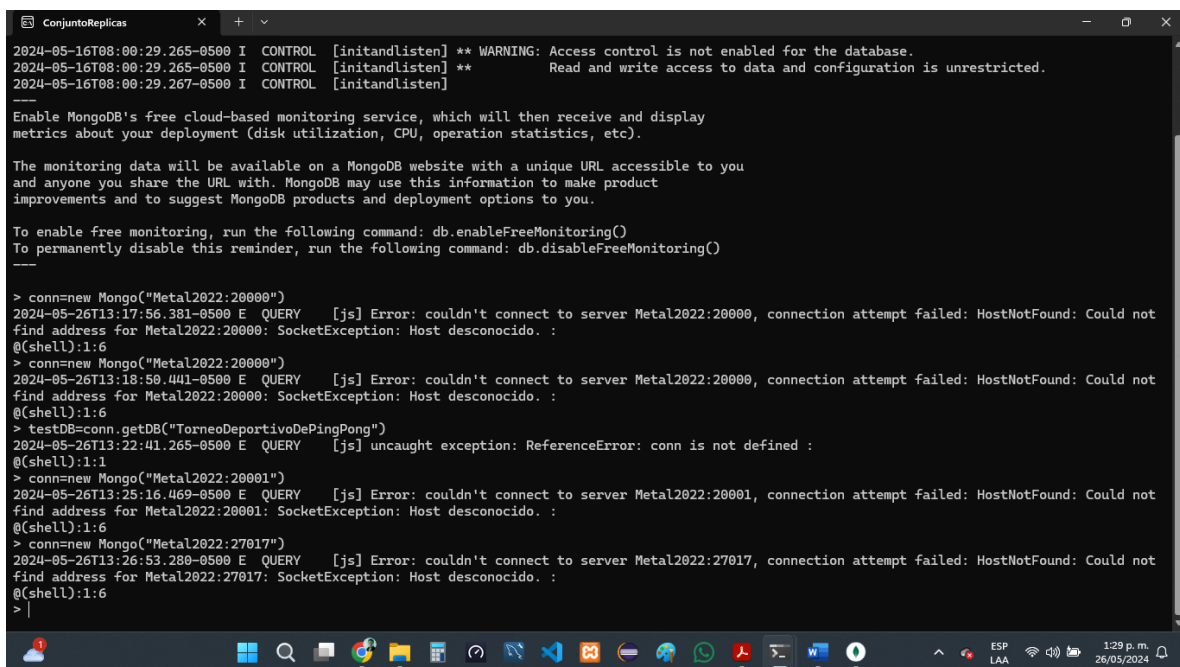
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
> conn=new Mongo("Metal2022:20000")
2024-05-26T13:17:56.381-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20000, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20000: SocketException: Host desconocido. :
@(<shell>):1:6
> conn=new Mongo("Metal2022:20000")
2024-05-26T13:18:50.441-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20000, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20000: SocketException: Host desconocido. :
@(<shell>):1:6
> testDB=conn.getDB("TorneoDeportivoDePingPong")
2024-05-26T13:22:41.265-0500 E QUERY [js] uncaught exception: ReferenceError: conn is not defined :
@(<shell>):1:1
>
```

***Evidencio que Sí tengo creada mi base en Mongo DB del torneo de ping pong***



***Sin embargo, no me conecta como en el paso a paso que usted hizo en clase, me sigue apareciendo error***



***Cambie el Puerto a 27017 para verificar si me conectaba, pero me sigue saliendo error***

```
ConjuntoReplicas x + v
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2024-05-16T08:00:29.265-0500 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
-----
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----

> conn=new Mongo("Metal2022:20000")
2024-05-26T13:17:56.381-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20000, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20000: SocketException: Host desconocido. :
@(shell):1:6
> conn=new Mongo("Metal2022:20000")
2024-05-26T13:18:50.441-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20000, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20000: SocketException: Host desconocido. :
@(shell):1:6
> testDB=conn.getDB("TorneoDeportivoDePingPong")
2024-05-26T13:22:41.265-0500 E QUERY [js] uncaught exception: ReferenceError: conn is not defined :
@(shell):1:1
> conn=new Mongo("Metal2022:20001")
2024-05-26T13:25:16.469-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20001, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20001: SocketException: Host desconocido. :
@(shell):1:6
> conn=new Mongo("Metal2022:27017")
2024-05-26T13:26:53.280-0500 E QUERY [js] Error: couldn't connect to server Metal2022:27017, connection attempt failed: HostNotFound: Could not
find address for Metal2022:27017: SocketException: Host desconocido. :
@(shell):1:6
> testDB=conn.getDB("TorneoDeportivoDePingPong")
```

*El sistema no me permite escribir otro comando como el de preguntar si es la rama  
máster*

```
ConjuntoReplicas x + v

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----

> conn=new Mongo("Metal2022:20000")
2024-05-26T13:17:56.381-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20000, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20000: SocketException: Host desconocido. :
@(shell):1:6
> conn=new Mongo("Metal2022:20000")
2024-05-26T13:18:50.441-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20000, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20000: SocketException: Host desconocido. :
@(shell):1:6
> testDB=conn.getDB("TorneoDeportivoDePingPong")
2024-05-26T13:22:41.265-0500 E QUERY [js] uncaught exception: ReferenceError: conn is not defined :
@(shell):1:1
> conn=new Mongo("Metal2022:20001")
2024-05-26T13:25:16.469-0500 E QUERY [js] Error: couldn't connect to server Metal2022:20001, connection attempt failed: HostNotFound: Could not
find address for Metal2022:20001: SocketException: Host desconocido. :
@(shell):1:6
> conn=new Mongo("Metal2022:27017")
2024-05-26T13:26:53.280-0500 E QUERY [js] Error: couldn't connect to server Metal2022:27017, connection attempt failed: HostNotFound: Could not
find address for Metal2022:27017: SocketException: Host desconocido. :
@(shell):1:6
> testDB=conn.getDB("TorneoDeportivoDePingPong")
2024-05-26T13:33:22.518-0500 E QUERY [js] uncaught exception: ReferenceError: conn is not defined :
@(shell):1:1
> testDB.isMaster()
2024-05-26T13:35:44.880-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> |
```

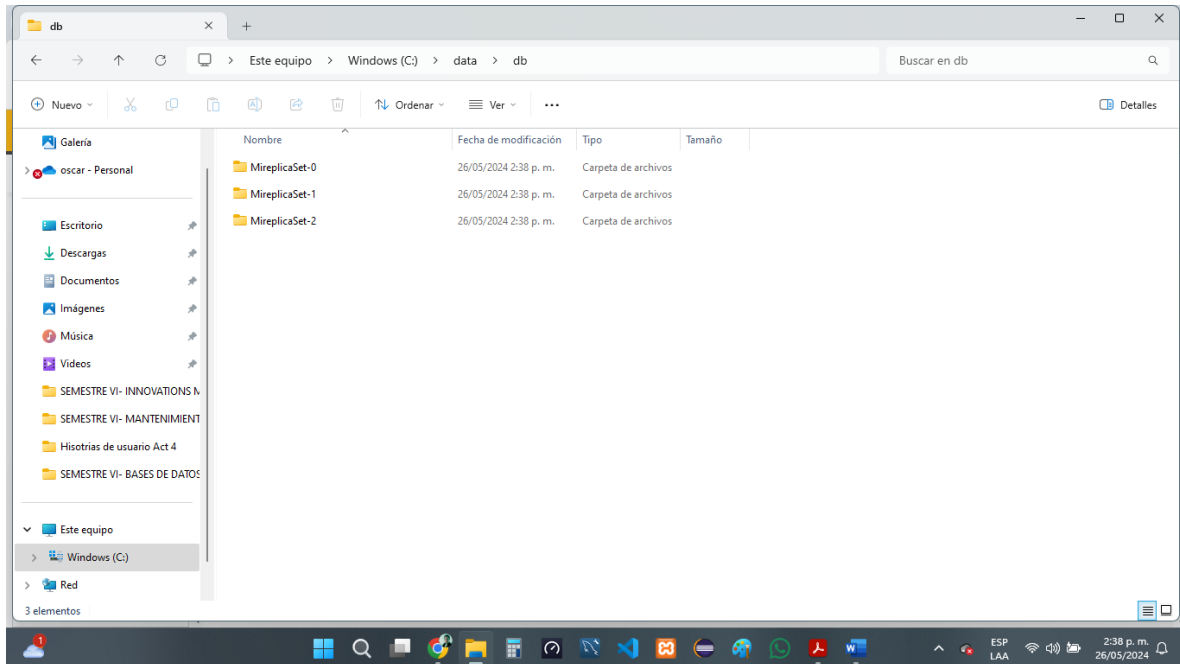
*He realizado cada paso como los que hizo en clase, pero sinceramente no sé cómo corregir el error de inicio*

```
ConjuntoReplicas
> testDB.Editoriales.insert(
... {
... id_editorial: "1 ",
... Nombre : " McGraw Hill ",
... Pais: " USA ",
... Grupo: " McGraw Hill Ed. "
... });
2024-05-26T13:39:08.778-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> testDB.Editoriales.insert(
... {
... id_editorial: "2 ",
... Nombre : " Pearson ",
... Pais: " USA ",
... Grupo: " Pearson Editorial"
... });
2024-05-26T13:39:47.099-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> testDB.Editoriales.insert(
... {
... id_editorial: "3 ",
... Nombre : " Oveja Negra ",
... Pais: " Colombia ",
... Grupo: " Oveja Negra Editores. "
... });
2024-05-26T13:39:54.484-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> testDB.Editoriales.insert (
... {id_editorial: "4",
... Nombre: "Editorial Planeta ",
... Pais: " Colombia ",
... Grupo: " Grupo Planeta "});
2024-05-26T13:40:13.602-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
>|
```

*Al tratar de conectarme al Nodo 2 con el puerto 20001 también me arroja error*

```
ConjuntoReplicas
... Pais: " USA ",
... Grupo: " McGraw Hill Ed. "
... });
2024-05-26T13:39:08.778-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> testDB.Editoriales.insert(
... {
... id_editorial: "2 ",
... Nombre : " Pearson ",
... Pais: " USA ",
... Grupo: " Pearson Editorial"
... });
2024-05-26T13:39:47.099-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> testDB.Editoriales.insert(
... {
... id_editorial: "3 ",
... Nombre : " Oveja Negra ",
... Pais: " Colombia ",
... Grupo: " Oveja Negra Editores. "
... });
2024-05-26T13:39:54.484-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> testDB.Editoriales.insert (
... {id_editorial: "4",
... Nombre: "Editorial Planeta ",
... Pais: " Colombia ",
... Grupo: " Grupo Planeta "});
2024-05-26T13:40:13.602-0500 E QUERY [js] uncaught exception: ReferenceError: testDB is not defined :
@(shell):1:1
> connSecondary = new Mongo("Meta2022:20001")
2024-05-26T13:45:34.904-0500 E QUERY [js] Error: couldn't connect to
find address for Meta2022:20001: SocketException: Host desconocido. :
@(shell):1:17
>|
```

***Soporte de prueba que los archivos se crearon en la carpeta db correctamente como usted o hizo en clase***



***Casos de pruebas para disponibilidad y redundancia en el proceso de replicación de un sistema de gestión del Torneo de Ping Pong***

Este documento detalla los casos de pruebas diseñados para verificar la disponibilidad y la redundancia en el proceso de replicación de un sistema de gestión de torneos de ping pong. Las pruebas se enfocan en asegurar que el sistema pueda manejar fallos, realizar replications correctas y mantener la integridad y disponibilidad de los datos en todo momento.

## **Casos de Pruebas**

### ***Prueba de Failover Automático***

#### **ID de Prueba: DR-01**

**Descripción:** Verificar que el sistema realice un failover automático correctamente cuando el nodo primario falla.

#### **Precondiciones:**

- Nodo primario y nodos secundarios activos
- Replicación configurada y en funcionamiento

#### **Pasos:**

1. Simular un fallo en el nodo primario (apagado del nodo).
2. Observar si uno de los nodos secundarios asume el rol de nodo primario
3. Verificar que las transacciones se continúan procesando sin interrupciones.

#### **Criterios de éxito:**

- El nodo secundario asume el rol de nodo primario automáticamente.
- No se pierden transacciones durante el proceso de failover.
- El sistema sigue disponible para los usuarios.

## ***Prueba de Consistencia de Datos en Replicación***

### **Id de la Prueba: DR-02**

**Descripción:** Asegurar que los datos replicados en los nodos secundarios sean consistentes con los del nodo primario.

#### **Pasos**

1. Insertar, actualizar y eliminar varios registros en el nodo primario.
2. Verificar que las mismas operaciones se reflejan correctamente en los nodos secundarios.
3. Comparar los datos entre el nodo primario y los nodos secundarios.

#### **Criterios de éxito:**

- Los datos en los nodos secundarios son idénticos a los del nodo primario.
- No hay discrepancias en los registros de la base de datos.

## ***Prueba de recuperación de desastres***

### **ID de la Prueba: DR-03**

**Descripción:** Verificar que el sistema pueda recuperarse de un fallo catastrófico sin pérdida de datos

#### **Precondiciones:**

Sistema con copias de seguridad regulares y replicación activa entre nodos



**Pasos:**

1. Simular una falla catastrófica que afecte a todos los nodos.
2. Restaurar el sistema desde las copias de seguridad
3. Verificar que todos los datos se recuperen correctamente y que el sistema vuelva a estar operativo.

**Criterios de Éxito:**

- Todos los datos se restauran sin pérdida.
- El sistema vuelve a estar operativo y accesible para los usuarios.

***Prueba de Balanceo de Carga*****ID de la Prueba: DR-04**

**Descripción:** Asegurar que el balanceador de carga distribuya las solicitudes de manera equitativa entre los nodos disponibles.

**Precondiciones:**

Balanceador de carga configurado y operativo y Nodos primarios y secundarios activos.

**Pasos:**

1. Enviar un gran número de solicitudes de lectura y escritura al sistema
2. Monitorear la distribución de las solicitudes entre los nodos.
3. Verificar que las cargas de trabajo se distribuyan uniformemente.

**Criterios de Éxito:**

- Las solicitudes se distribuyen equitativamente entre los nodos.
- No hay sobrecarga en ningún nodo específico.
- El rendimiento del sistema se mantiene dentro de los parámetros aceptables.

***Pruebas de replicación en tiempo real:*****ID de la prueba DR-05**

**Descripción:** Verificar que la replicación de datos ocurra en tiempo real sin demoras significativas.

**Precondiciones:** Sistema en estado estable con replicación activa

**Pasos:**

1. Realizar varias operaciones de escritura en el nodo primario.
2. Monitorear el tiempo que tarda cada operación en reflejarse en los nodos secundarios.

**Criterios de Éxito:**

- La replicación ocurre en tiempo real o con una demora mínima aceptable.
- No se observan retrasos significativos que puedan afectar la integridad de los datos.

***En conclusión***, estos casos de pruebas están diseñados para asegurar que el sistema de gestión de torneos de ping pong mantenga una alta disponibilidad y redundancia mediante procesos efectivos de replicación. Implementar y superar estas pruebas garantizará que el sistema sea robusto, confiable y capaz de manejar fallos y mantener la integridad y disponibilidad de los datos en todo momento.

***Enlace del video explicativo del Trabajo presentado***

<https://youtu.be/bhTl6S5op30>

### ***Referencias Bibliográficas***

[Sarasa, A. \(2016\). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC. \(Capítulo 7- Replicacion\)](#)