**Corporación Universitaria Iberoamericana**

**Asignatura a:**

Bases de datos avanzadas

Actividad 3 – Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

**Presenta:**

Juan Sebastián Quiroz Soto

**ID:** 100143989

**Docente:**

William Ruiz

Apartadó, Diciembre, 17 del 2023

**Documento de Requerimientos No Funcionales para Torneo de Fútbol - Particionamiento**

**1. Introducción**

El propósito de este documento es especificar los requerimientos no funcionales para el sistema de gestión de torneos de fútbol, centrándose en aspectos relacionados con el particionamiento del sistema.

**2. Requerimientos No Funcionales**

**2.1 Escalabilidad**

El sistema deberá ser escalable para soportar un aumento gradual en la cantidad de torneos, equipos y usuarios. El particionamiento debe permitir la distribución eficiente de la carga y recursos, garantizando un rendimiento óptimo incluso en momentos de alta demanda.

**2.2 Seguridad**

La arquitectura de particionamiento debe garantizar la seguridad de los datos. Se requiere un aislamiento efectivo entre las instancias de torneos para prevenir accesos no autorizados y proteger la integridad de la información.

**2.3 Mantenibilidad**

El particionamiento debe facilitar las actualizaciones, mejoras y correcciones sin afectar a otros torneos en ejecución. Deberá ser posible realizar cambios en una instancia específica sin interferir en el funcionamiento general del sistema.

**2.4 Disponibilidad**

El sistema deberá ser altamente disponible, garantizando que la caída de una instancia o componente no afecte negativamente a las demás. Se debe implementar un mecanismo de redundancia y failover para minimizar los tiempos de inactividad.

**2.5 Rendimiento**

El particionamiento debe optimizar el rendimiento del sistema, distribuyendo eficientemente los recursos computacionales, de red y de almacenamiento. Se deben establecer métricas de rendimiento y realizar pruebas periódicas para asegurar que se cumplen los objetivos establecidos.

**2.6 Cumplimiento normativo**

El sistema debe cumplir con las regulaciones y normativas aplicables a la gestión de datos personales y financieros. El particionamiento debe facilitar la implementación de controles de cumplimiento y auditorías independientes para cada instancia de torneo.

**2.7 Interoperabilidad**

El particionamiento debe permitir la interoperabilidad con otros sistemas relevantes, como sistemas de pago, sistemas de transmisión en vivo y aplicaciones móviles. Se deben establecer interfaces y protocolos de comunicación estandarizados.

**2.8 Adaptabilidad**

La arquitectura de particionamiento debe ser adaptable a diferentes tipos de torneos y formatos de competición. Debe ser fácilmente configurable para satisfacer las necesidades específicas de cada instancia de torneo.

**3. Restricciones y Limitaciones**

**3.1 Hardware y Recursos**

El particionamiento debe ser diseñado considerando las limitaciones y capacidades del hardware y los recursos disponibles. Se deben especificar los requisitos mínimos y recomendados para cada instancia de torneo.

**3.2 Costos**

Se deben establecer restricciones presupuestarias para el particionamiento del sistema. El diseño debe ser eficiente en términos de costos, minimizando la infraestructura necesaria sin comprometer la calidad y el rendimiento.

**4. Aprobaciones**

Este documento de requerimientos no funcionales para el particionamiento del sistema de gestión de torneos de fútbol debe ser revisado y aprobado por todas las partes interesadas antes de proceder con la implementación.

Comandos para el particionamiento de bases de datos con *sharding:*

Estos comandos, lo que abrían de hacer seria copiar una colección de una base de datos, y copiar cierta información en otras bases de datos.

// Conectar a la base de datos de origen

use torneo**;**

// Definir las colecciones de origen

var colecciones\_origen = **[**"arbitros"**,** "deportistas"**];**

// Definir los nombres de las bases de datos de destino

var nombre\_basedatos\_destino\_1 = "torneo1"**;**

var nombre\_basedatos\_destino\_2 = "torneo2"**;**

// Iterar sobre las colecciones de origen

colecciones\_origen.forEach**(**function**(**coleccion\_origen**)** **{**

// Iterar sobre los documentos de la colección de origen

var documentos = db**[**coleccion\_origen**]**.find**()**.toArray**();**

for **(**var i = 0**;** i < documentos.length**;** i++**)** **{**

// Determinar a cuál base de datos de destino pertenece el documento

var baseDatosDestino = **(**i % 2 === 0**)** ? nombre\_basedatos\_destino\_1 **:** nombre\_basedatos\_destino\_2**;**

// Insertar el documento en la colección de la base de datos de destino correspondiente

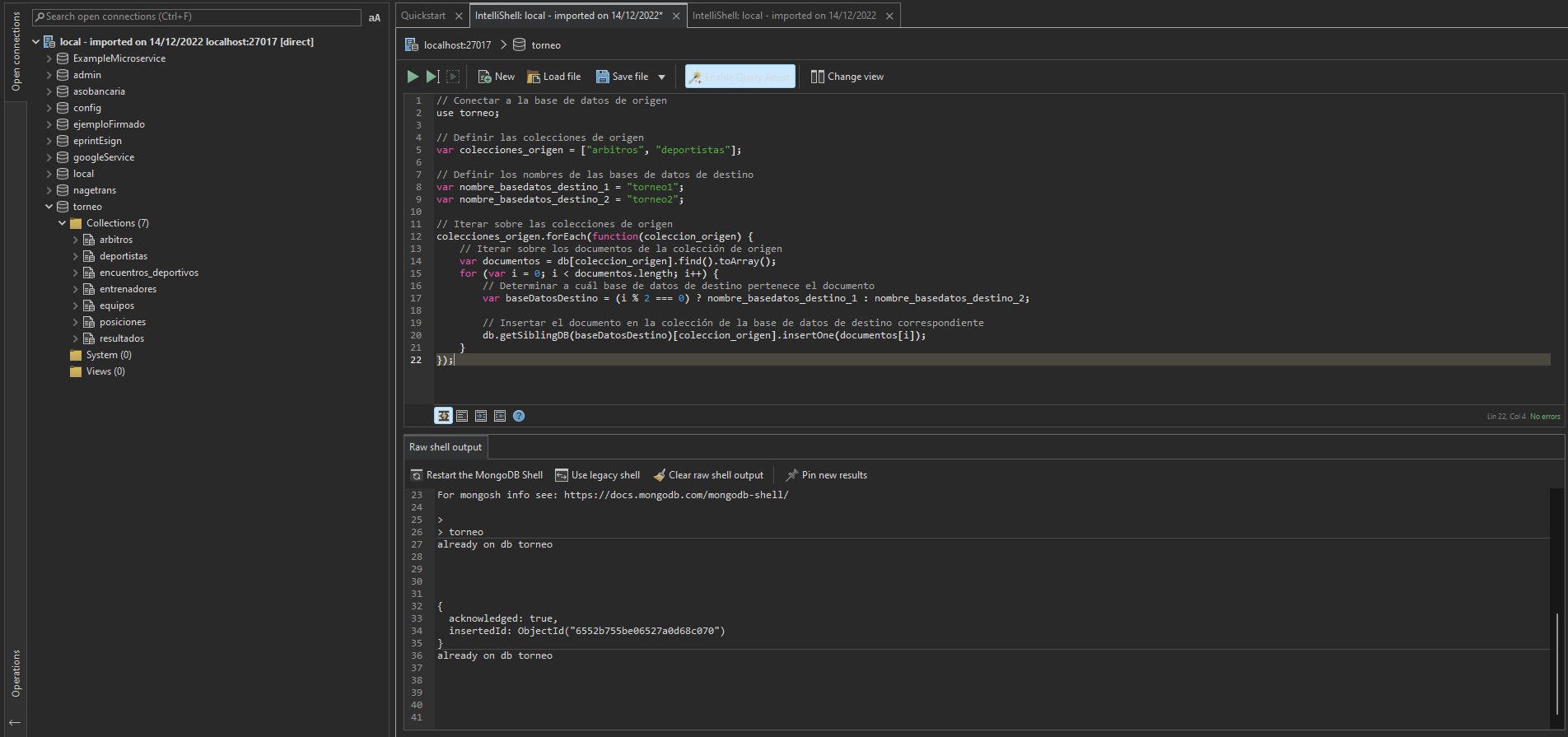
db.getSiblingDB**(**baseDatosDestino**)[**coleccion\_origen**]**.insertOne**(**documentos**[**i**]);**

**}**

**});**

Resultados:

Se va a copiar una de las colecciones de la base de datos llamada torneo,



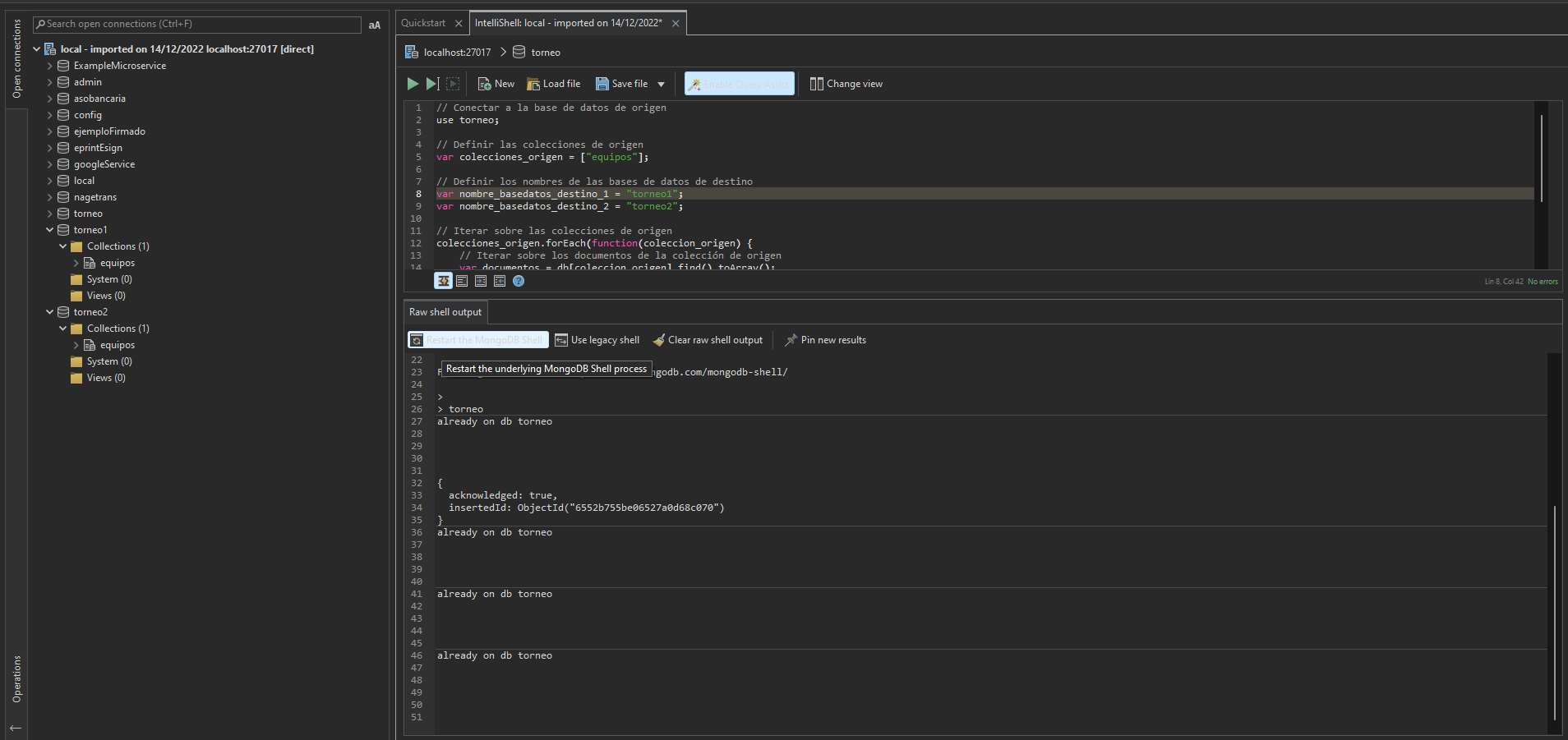
se podrían copiar todas las colecciones, pero para este caso, solo voy a copiar una de las colecciones, la cual contiene la siguiente información,

Captura de pantalla de un celular

Descripción generada automáticamente

En este caso se ejecutaron los comandos anteriores, el resultado fue un éxito, donde en dos bases de datos diferentes, llamados torneo1 y torneo2, se distribuyó la información, también se podría hacer con grandes cantidades de información, el resultado obtenido fue el siguiente.

Mensaje de retorno donde me dice que fue un éxito.



Torneo 1:

Captura de pantalla de computadora

Descripción generada automáticamente

Torneo 2:

Captura de pantalla de computadora

Descripción generada automáticamente