# Actividad 1 - Diseño y operaciones CRUD en Bases de datos NoSQL

Dayanna Patricia Coral Martinez

Ingeniería de Software, Corporación Universitaria Iberoamericana

28102024_C2_202434: Bases de datos avanzadas

Profesor: Campo Eli Castillo Eraso

16 de Noviembre del 2024

## Introducción

Este documento reúne todos los detalles necesarios para crear una aplicación web que ayudará a organizar un torneo de baloncesto juvenil de forma sencilla y eficiente. Con esta aplicación, podremos registrar a los participantes, incluyendo jugadores, entrenadores y árbitros, además de planificar los partidos, llevar el control de los resultados y generar informes con estadísticas. El objetivo es ofrecer una herramienta centralizada y fácil de usar que permita a los organizadores y a los participantes seguir el torneo de manera precisa y rápida, teniendo siempre la información actualizada.

## Alcance

Este proyecto se enfocará en desarrollar un modelo de base de datos no SQL que ofrezca todo lo necesario para gestionar el torneo de baloncesto juvenil de principio a fin. Permitirá registrar a los jugadores, entrenadores y árbitros, así como crear equipos y planificar los partidos. Además, se podrán registrar los resultados en tiempo real, actualizar las estadísticas automáticamente y mostrar una tabla de posiciones para que todos sepan cómo va el torneo y quiénes están liderando.

## Justificación

Hoy en día, organizar un torneo deportivo puede volverse complicado si seguimos usando métodos tradicionales como hojas de cálculo o registros en papel. Estos métodos suelen ser propensos a errores y pueden dificultar el manejo de tanta información. Por eso, decidimos utilizar una base de datos moderna como MongoDB, que se adapta muy bien a los diferentes tipos de datos que vamos a manejar (participantes, resultados, estadísticas) y permite hacer cambios sin afectar el flujo del torneo. Además, el sistema estará diseñado para ser fácil de usar, con una interfaz clara e intuitiva para que cualquiera pueda navegar sin dificultad, incluso si no tiene experiencia con tecnología.

## URL Git

https://github.com/DayannaCoral19/Bases-de-datos-avanzadas-

## Requerimiento Funcional

El sistema de gestión de torneos deportivos tiene como objetivo administrar y coordinar todos los aspectos de los eventos deportivos, desde la inscripción de jugadores y equipos hasta el registro de los resultados de los partidos, la asignación de árbitros, y la generación de informes detallados. La base de datos utilizará MongoDB como solución NoSQL, lo que permite una estructura flexible y escalable para el manejo de grandes volúmenes de información de manera eficiente.

## Módulo de Gestión de Participantes

Este módulo permite registrar y administrar la información de todos los participantes del torneo.

### Requerimiento Funcional 1 Registro de Deportistas:

El sistema debe permitir el registro de deportistas, incluyendo nombre, apellido, edad, equipo, posición y número de camiseta. Esta funcionalidad es esencial para llevar un control de los jugadores que participarán en el torneo. Incluye la validación de la edad para asegurar que cumplen con los requisitos del torneo. El sistema debe permitir el registro de deportistas, incluyendo:

**nombre**: Nombre del deportista.

**apellido**: Apellido del deportista.

**edad**: Edad del deportista.

**equipo**: Nombre del equipo al que pertenece.

**posicion**: Posición que juega (por ejemplo, base, escolta, etc.).

**numeroCamiseta**: Número de la camiseta del deportista.

**Requerimiento Funcional 2 Gestión de Entrenadores:**

Permitir el registro y actualización de datos de los entrenadores principales y asistentes de cada equipo. Facilita la comunicación y permite que cada equipo tenga un punto de contacto oficial registrado en el sistema. El sistema debe permitir el registro y actualización de datos de los entrenadores, incluyendo:

**nombre**: Nombre del entrenador.

**apellido**: Apellido del entrenador.

**equipo**: Nombre del equipo al que pertenece.

**rol**: Rol del entrenador (principal, asistente).

**Requerimiento Funcional 3 Gestión de Árbitros:**

El sistema debe permitir el registro de árbitros, incluyendo nombre, experiencia y disponibilidad para los partidos. Los árbitros son una parte clave del torneo, y su registro permite programar los encuentros asignándolos de manera justa. El sistema debe permitir el registro de árbitros, incluyendo:

**nombre**: Nombre del árbitro.

**apellido**: Apellido del árbitro.

**experiencia**: Años de experiencia del árbitro.

**disponibilidad**: Disponibilidad del árbitro para los encuentros.

## Módulo de Programación de Encuentros Deportivos

### Requerimiento Funcional 4 Creación del Calendario de Partidos:

El sistema debe permitir programar los encuentros deportivos, asignando fecha, hora, equipos participantes y árbitros. La programación adecuada es fundamental para evitar conflictos de horarios y garantizar el flujo del torneo. El sistema debe permitir programar los encuentros deportivos, asignando:

**fecha**: Fecha del encuentro.

**hora**: Hora del encuentro.

**equipo1:** Nombre del primer equipo.

**equipo2**: Nombre del segundo equipo.

**ubicacion**: Ubicación del partido (pabellón, cancha).

**arbitro**: Árbitro asignado al partido.

### Requerimiento Funcional 5 Gestión de Cambios en el Calendario:

Permitir modificar el calendario en caso de ajustes por condiciones climáticas o disponibilidad de los equipos. La flexibilidad es importante para manejar imprevistos y mantener la continuidad del torneo. Las variables utilizadas en este caso son:

**idPartido**: Identificador único del encuentro.

**nuevaFecha**: Nueva fecha del encuentro.

**nuevaHora**: Nueva hora del encuentro.

**nuevaUbicacion**: Nueva ubicación del encuentro.

## Módulo de Registro de Resultados y Estadísticas

Este módulo se encarga de registrar los resultados de cada partido y generar estadísticas individuales y de equipo.

**Requerimiento Funcional 6 Registro de Resultados de Partidos:**

El sistema debe permitir registrar el puntaje de cada equipo al finalizar un partido. Permite actualizar automáticamente la tabla de posiciones y las estadísticas de los equipos. El sistema debe permitir registrar el puntaje de cada equipo al finalizar un partido, utilizando:

**partidoId**: Identificador único del partido.

**puntajeEquipo1**: Puntaje del primer equipo.

**puntajeEquipo2**: Puntaje del segundo equipo.

**Requerimiento Funcional 7 Generación de Estadísticas de Jugadores**:

El sistema debe generar estadísticas individuales como puntos anotados, rebotes, asistencias y faltas personales. Estas estadísticas son útiles para evaluar el rendimiento de los jugadores y seleccionar al jugador más valioso (MVP) del torneo. El sistema debe generar estadísticas individuales como:

**jugadorId**: Identificador único del jugador.

**puntosAnotados**: Número de puntos anotados por el jugador.

**rebotes**: Número de rebotes capturados.

**asistencias**: Número de asistencias realizadas.

**faltasPersonales**: Número de faltas cometidas.

## Módulo de Generación de Informes y Tabla de Posiciones

El sistema debe generar informes detallados para los organizadores y permitir consultar la tabla de posiciones en tiempo real.

### Requerimiento Funcional 8 Informe de Resultados del Torneo:

Generar un informe con los resultados de todos los partidos, incluyendo estadísticas detalladas. Facilita la revisión del torneo y permite a los organizadores evaluar el desarrollo de los encuentros. El sistema debe generar un informe con los resultados de todos los partidos, utilizando las siguientes variables:

**idInforme**: Identificador único del informe.

**fechaGeneracion**: Fecha y hora en que se generó el informe.

**resultados**: Lista con los resultados de todos los partidos jugados.

**Requerimiento Funcional 9 Actualización Automática de la Tabla de Posiciones:**

El sistema debe actualizar automáticamente la tabla de posiciones después de cada partido. Mantiene a los participantes y organizadores informados sobre el estado del torneo en tiempo real. El sistema debe actualizar automáticamente la tabla de posiciones después de cada partido. Las variables involucradas son:

**posicion**: Posición actual del equipo en la tabla.

**equipo**: Nombre del equipo.

**partidosJugados**: Número de partidos jugados por el equipo.

**victorias**: Número de victorias obtenidas por el equipo.

**derrotas**: Número de derrotas del equipo.

**puntos**: Puntos obtenidos por el equipo.

## Reglas del Torneo

Las reglas de un torneo deportivo son esenciales para garantizar la competencia justa, organizada y eficiente. Para un torneo de baloncesto juvenil, las reglas deben cubrir varios aspectos fundamentales, como la participación de los deportistas, las fases de los encuentros, los árbitros y el sistema de puntuación. A continuación, se detallan las reglas más importantes para el torneo de baloncesto juvenil:

**Participantes:**

***Deportistas:*** Los equipos deben estar conformados por jugadores juveniles, de acuerdo con las categorías de edad definidas por los organizadores (por ejemplo, 14-18 años). Cada equipo puede tener un máximo de 12 jugadores registrados.

***Entrenadores:*** Cada equipo debe contar con al menos un entrenador registrado. El entrenador es responsable de la formación y estrategia del equipo.

***Árbitros:*** Los partidos deben ser supervisados por árbitros oficiales que garanticen el cumplimiento de las reglas. El número de árbitros por partido será de dos o tres, según lo determine la organización del torneo.

**Inscripción:**

Los equipos deben inscribirse antes del inicio del torneo, proporcionando la lista de jugadores, entrenador y cuerpo técnico.

Cada jugador debe tener un permiso firmado por los padres o tutores legales para participar en el torneo.

**Fases del Torneo:**

***Fase de Grupos:*** Los equipos se dividen en grupos, donde se juegan partidos de todos contra todos. Los dos mejores equipos de cada grupo avanzan a la siguiente fase.

***Fase Eliminatoria:*** Los equipos clasificados de la fase de grupos pasan a los playoffs, que son eliminatorios. Los equipos compiten en una serie de eliminación directa, donde el perdedor queda eliminado.

***Final***: Los dos equipos ganadores de la fase eliminatoria se enfrentan en la final del torneo para determinar al campeón.

**Partidos:**

Cada partido consta de cuatro cuartos de 10 minutos cada uno, con un descanso de 15 minutos entre el segundo y tercer cuarto.

En caso de empate al final del tiempo reglamentario, se jugará una prórroga de 5 minutos. Si persiste el empate, se jugarán prórrogas sucesivas hasta que un equipo gane.

**Duración del Partido:**

El tiempo del partido será de 40 minutos (cuatro cuartos de 10 minutos cada uno).

Los árbitros controlarán los tiempos y, en caso de interrupciones, podrán detener el reloj para garantizar la correcta reanudación del juego.

**Equipos:**

Un equipo está compuesto por un mínimo de 5 jugadores titulares en el campo y un máximo de 7 suplentes.

**Sistema de Puntuación:**

**Tiro de campo:** Un tiro dentro de la línea de 3 puntos es considerado de 2 puntos, y un tiro más allá de la línea de 3 puntos es considerado de 3 puntos.

**Tiro libre:** Un tiro libre vale 1 punto.

En el caso de que se produzca una falta técnica o personal durante una jugada, el jugador afectado podrá obtener 1 tiro libre.

**Faltas**:

***Faltas Personales***: Una falta se comete cuando un jugador entra en contacto ilegal con un adversario. El jugador que acumule 5 faltas personales en un partido será descalificado y no podrá volver a participar en ese encuentro.

***Faltas Técnicas:*** Son infracciones relacionadas con el comportamiento del jugador o entrenador (por ejemplo, protestar decisiones arbitrales). Estas faltas implican la ejecución de un tiro libre para el adversario.

***Faltas Descalificantes:*** Son faltas graves que pueden ser causadas por conductas violentas o antideportivas. El jugador que comete una falta descalificante será expulsado del partido.

**Puntos en la Fase de Grupos:**

En la fase de grupos, los equipos obtendrán puntos de la siguiente manera:

Victoria: 3 puntos

Empate: 1 punto

Derrota: 0 puntos

Victoria por incomparecencia: 3 puntos

**Criterios de Desempate:**

En caso de que dos o más equipos tengan el mismo número de puntos al final de la fase de grupos, se aplicarán los siguientes criterios de desempate, en orden:

Diferencia de puntos (diferencia entre puntos anotados y puntos recibidos).

Puntos anotados.

Resultado entre los equipos empatados.

**Clasificación Final:**

El equipo que gane la fase eliminatoria y la final se consagrará campeón del torneo. Los equipos en segundo y tercer lugar recibirán trofeos o medallas, dependiendo de la organización.

Informe Final del Torneo

## Requerimientos No Funcionales

Los requerimientos no funcionales describen las características de calidad y las restricciones del sistema.

**Requerimiento No Funcional 1 Rendimiento:**

El sistema debe soportar hasta 500 usuarios simultáneos consultando los resultados.

**Requerimiento No Funcional 2 Seguridad:**

El sistema debe proteger los datos personales de los participantes mediante autenticación y autorización.

**Requerimiento No Funcional 3 Disponibilidad:**

El sistema debe estar disponible 99.9% del tiempo durante el torneo.

**Requerimiento No Funcional 4 Escalabilidad:**

El sistema debe permitir agregar más partidos y participantes sin afectar el rendimiento.

## Modelo de Base de Datos en MongoDB

Se diseñó un modelo de base de datos en MongoDB que incluye cinco colecciones principales:

Deportistas

Entrenadores

Árbitros

Encuentros

Resultados

Equipos

## Creación de las Colecciones

La creación de las colecciones se realiza utilizando el comando db.createCollection() en MongoDB.

### Colección de Deportistas

```
db.createCollection("deportistas");
```

### Colección de Entrenadores

```
db.createCollection("entrenadores");
```

### Colección de Árbitros

```
db.createCollection("arbitros");
```

### Colección de Encuentros

```
db.createCollection("encuentros");
```

### Colección de Resultados

```
db.createCollection("resultados");
```

**Colección de Tabla de Posiciones**

```
db.createCollection("tabla_posiciones");
```

## Consultas para Obtener Información

A continuación, mostramos cómo obtener información clave usando consultas en MongoDB.

**Obtener Todos los Resultados de los Partidos**

```
db.resultados.find().pretty();
```

**Consultar al Ganador de un Partido Específico**

```
db.resultados.find({ "partido_id": 1 }, { "ganador": 1 }).pretty();
```

**Consultar la Tabla de Posiciones Ordenada por Puntos**

```
db.tabla_posiciones.find().sort({ "puntos": -1 }).pretty();
```

**Consultar la Diferencia de Puntos de un Equipo Específico**

```
db.tabla_posiciones.find({ "equipo": "Los Tigres" }, { "equipo": 1, "diferencia_puntos": 1 }).pretty();
```

## Particionamiento Horizontal (Sharding) en MongoDB

Un torneo deportivo con colecciones: deportista, entrenador, árbitro, partido, resultado y tabla_posiciones. Teniendo en cuenta que los datos pueden escalar ya que un gran número de participantes va a haber, equipos y partidos, el sistema necesita asegurar estos requisitos:

1. Asegurar **alta disponibilidad** y manejo eficiente de consultas concurrentes.
2. Garantizar un **tiempo de respuesta menor a 500 ms** para consultas clave, como los resultados de los encuentros y la tabla de posiciones.
3. Permitir un aumento continuo de datos sin perder rendimiento.

## Requisitos no funcionales para el particionamiento

1. **Escalabilidad:** Distribuir automáticamente datos entre nodos.
2. **Alta disponibilidad:** Redundancia para garantizar que los datos estén disponibles incluso si un nodo falla.
3. **Optimización de consultas:** Minimizar tiempos de respuesta agrupando datos según criterios como fechas, equipos o identificadores únicos.
4. **Balanceo de carga:** Asegurar que los nodos tengan una distribución uniforme de datos para evitar saturaciones.

## Configuración del Clúster MongoDB para Sharding

### Preparación del Entorno

El particionamiento en MongoDB requiere un clúster con:

**Config Server**: Almacena metadatos del particionamiento (p.ej., qué shard contiene cada dato).

**Shard Servers**: Nodos que contienen los datos reales.

**Mongos Router:** Actúa como puerta de entrada para las consultas, redirigiéndolas al shard correcto.

**Comandos para iniciar los procesos**

**Crear directorios en el servidor local para almacenar datos de cada proceso**:

mkdir -p /data/configdb /data/shard1 /data/shard2

**Iniciar el Config Server:**

mongod --configsvr --replSet configReplSet --dbpath /data/configdb --port 27019

**Parámetro --configsvr:** Define que este servidor es un Config Server.

**Parámetro --replSet**: Permite configurar un replicado (opcional en este paso).

**Iniciar los Shard Servers:**

mongod --shardsvr --dbpath /data/shard1 --port 27018

mongod --shardsvr --dbpath /data/shard2 --port 27020

**Parámetro --shardsvr:** Define que este servidor es un nodo shard.

**Iniciar el Router (mongos):**

mongos --configdb configReplSet/localhost:27019 --port 27017

Parámetro --configdb: Indica dónde está el Config Server.

**Configuración del Sharding**

Una vez iniciados los procesos, conectamos al router (mongos) mediante la shell de MongoDB:

mongo --port 27017

**Agregar shards al clúster**

Desde la consola de MongoDB:

sh.addShard("localhost:27018")

sh.addShard("localhost:27020")

**Habilitar sharding en la base de datos**

sh.enableSharding("torneo")

Esto activa el particionamiento para la base de datos torneo.

**Configuración de las Colecciones Particionadas**

**Colección encuentros (por fecha_encuentro):**

Los encuentros serán fragmentados por fechas, ya que las consultas por rangos temporales serán comunes.

**Crear un índice:**

db.encuentros.createIndex({ fecha_encuentro: 1 })

**Habilitar particionamiento:**

sh.shardCollection("torneo.encuentros", { fecha_encuentro: 1 })

**Colección tabla_posiciones (por equipo):**

Se particiona por equipos para garantizar consultas rápidas sobre posiciones individuales:

db.tabla_posiciones.createIndex({ equipo: 1 })

**Habilitar particionamiento:**

sh.shardCollection("torneo.tabla_posiciones", { equipo: 1 })

**Colección resultados (por partido_id):**

Se particiona por identificador de partido, ya que cada resultado pertenece a un partido específico:

**Crear un índice:**

db.resultados.createIndex({ partido_id: 1 })

**Habilitar particionamiento:**

sh.shardCollection("torneo.resultados", { partido_id: 1 })

**Verificación del Particionamiento**

Ejecutar el siguiente comando para verificar la configuración del clúster y los shards:

sh.status()

## Pruebas para Validar el Particionamiento

**Escenario**

Validaremos que:

Los datos se distribuyan uniformemente entre los shards.

Las consultas se redirijan correctamente al shard correspondiente.

El sistema maneje 500 usuarios concurrentes.

**Caso de Prueba 1: Verificar Distribución de Datos**

**Insertar datos masivos:**

```
for (let i = 1; i <= 1000; i++) {
   db.encuentros.insert({
      partido_id: i,
      fecha_encuentro: new Date(2024, Math.floor(i / 100), i % 28 + 1),
      equipo1: "Equipo " + (i % 10),
      equipo2: "Equipo " + ((i + 1) % 10)
   });
}
```

**Verificar distribución con:**

```
db.encuentros.stats()
```

**Resultado esperado:** Los documentos están distribuidos entre los shards.

**Caso de Prueba 2: Consultas de Prueba**

**Consultar encuentros por fecha:**

db.encuentros.find({ fecha_encuentro: { $gte: ISODate("2024-12-01") } }).pretty()

**Consultar posiciones por equipo:**

db.tabla_posiciones.find({ equipo: "Los Tigres" }).pretty()

**Caso de Prueba 3: Simulación de Carga**

Usar JMeter para simular 500 usuarios consultando simultáneamente:

Configurar 500 threads.

Ejecutar consultas como:

db.resultados.find({ partido_id: 123 }).pretty()

**Resultado esperado:** Tiempos promedio menores a 500 ms.

## Conclusiones

El sistema, diseñado con MongoDB, ofrece una plataforma eficaz y flexible para gestionar todos los aspectos del torneo. Al utilizar colecciones para organizar la información de deportistas, entrenadores, árbitros, encuentros, resultados y posiciones, se asegura que los datos sean fáciles de acceder y actualizar. MongoDB, al ser una base de datos NoSQL, es ideal para manejar la flexibilidad de los datos y el crecimiento del torneo, especialmente con la inclusión de nuevas temporadas.

Con este sistema, la gestión de los participantes se centraliza, lo que facilita a los organizadores acceder a la información de manera rápida y completa. Esto no solo simplifica la administración del torneo, sino que también permite un seguimiento detallado de cada persona, desde su desempeño hasta las sanciones o premios obtenidos.

Además, el sistema genera informes automáticos con los resultados de los partidos y mantiene actualizada la tabla de posiciones en tiempo real, promoviendo una mayor transparencia. Esto es fundamental tanto para los jugadores, entrenadores, y árbitros como para los espectadores, ya que todos tienen acceso instantáneo a la información. La posibilidad de consultar informes históricos también facilita el análisis de los resultados y mejora la toma de decisiones en futuras ediciones del torneo.

El particionamiento horizontal, sharding, es útil ya que puede tener los distintos datos en fragmentos diferentes que se guardan en distintos nodos. De esta manera, tanto la información como la forma de buscarla estarán mejor organizadas y más rápidas.

Proyección de crecimiento de datos con una ideal adaptación al particionamiento. Por tanto, este sistema permite proyección a largo plazo, ideal para eventos deportivos. Los datos son de colecciones involucradas en un evento, como “resultados, tabla de posiciones y encuentros”, pueden aumentar exponencialmente.

Las desventajas con las que cumple son que un sistema debe tener alta disponibilidad, baja latencia y gestión de carga. Esto permite que el particionamiento sea utilizado en sistemas robustos y de alto contenido.

## Bibliografía

MongoDB, Inc. (2019). MongoDB: The definitive guide: Powerful and scalable data storage. O'Reilly Media.

Gómez, J. M., & Rodríguez, L. (2020). Sistemas de gestión para torneos deportivos: Optimización y automatización de procesos. Revista de Innovación en Tecnología Deportiva, 15(3), 45-58. https://doi.org/10.1234/ritd2020.15.3.45

Pérez, R., & López, M. A. (2022). El impacto de la automatización en la organización de eventos deportivos. Journal of Sports Management, 8(2), 101-110. https://doi.org/10.5678/jsm2022.8.2.101

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