Texto

Descripción generada automáticamente con confianza media**ESCUELA DE INGENIERÍAS**

**Facultad de ingeniería y Tecnologías**

**de la Información y la Comunicación**

|  |  |
| --- | --- |
| Curso | **Tópicos avanzados en bases de datos** |
| Docente | **Juan Darío Rodas Marín** |
| Nombre de la actividad | **Examen 4 – Bases de datos embebidas NoSQL** |
| Autor(es) | **Juan Esteban Galeano Herrera** |

**Abastecimiento de infraestructura – 20%**

Realice instalación de MongoDB en contenedor utilizando la herramienta Docker. Realice la documentación del proceso de creación del contenedor y la forma como se conecta a él. Demuestre la conexión usando **MongoDB Compass.**

**Descargar imagen de mongo**

Utilice en la terminal el comando

*docker pull mongo*

Por defecto se comenzará a descargar la imagen con la etiqueta *latest*, la última versión de mongoDB

Texto

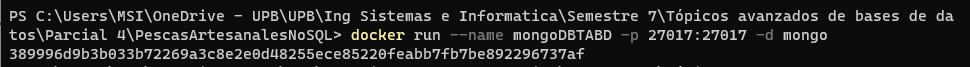
Descripción generada automáticamente

**Correr la imagen de MongoDB**

Inicie el contenedor ejecutando el comando

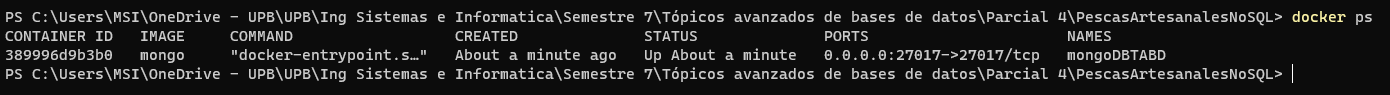
*docker run –name unNombre -p 27017:27017 -d mongo*

* Tenga en cuenta que debe modificar *unNombre* por el nombre que desea usar para su contenedor.



Puede validar que la imagen de mongo de está ejecutando, ingresando el comando

*docker ps*



**Conexión a mongoDB**

La verificación de la conexión se realizará a través de MongoDB Compass.

Se abre la aplicación MongoDB Compass, se hace click en el botón *New Connection*, se habilita la opción *Edit Connection String* y se ingresa el string de conexión para luego hacer click en el botón *Connect*:

*Mongodb://localhost:27017*

* Tenga en cuenta que en el connection string debe ingresar la dirección IP (o nombre de host) en el que se está corriendo MongoDB y el número de puerto sobre el cual está corriendo el servicio. Para el caso del ejercicio, Mongo está corriendo en *localhost* y en el puerto *27017*

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

En MongoDB Compass podrá observar entonces en host al que está conectado y las bases de datos disponibles en MongoDB.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**Implementación del modelo de datos NoSQL – 20%**

Realice la implementación de la base de datos y las respectivas colecciones que se pueda necesitar. El diseño es libre, pero debe permitir la ejecución de las operaciones CRUD requeridas. Puede utilizar los mismos datos utilizados en la entrega del examen anterior.

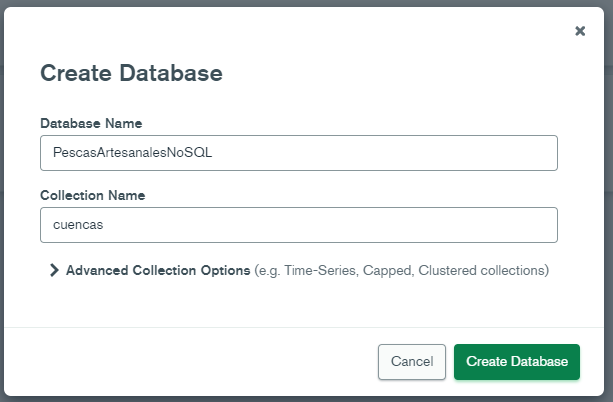
**Creación del modelo**

En MongoDB Compass, vaya a *Databases* y haga click en *Create database*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ingrese el nombre de la base de datos y de la colección que desea agregar a la base de datos (analogía con tabla)



Ingrese a la base de datos que acaba de construir para crear las colecciones faltantes. Haga click en *Create collection* e ingrese el nombre de la colección. Repita tantas veces como colecciones deba crear.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para el caso del ejercicio, solo se requieren 3 colecciones: *métodos, cuencas y pescas*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

En este ejercicio, el modelo de las colecciones es:

**Colección metodos:**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Colección cuencas**

Interfaz de usuario gráfica

Descripción generada automáticamente

**Colección pescas**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Adicionalmente**

El modelo que se creó cuenta con una validación de esquemas denominado *scheme\_validation* en cada uno de los documentos declarado de la siguiente manera:

**Colección métodos**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Colección cuencas**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Colección pescas**

La colección pescas tiene un *scheme\_validation­* especial, ya que es dinámico. Este *scheme\_validation* se actualiza cada vez que se crea, modifica o elimina una cuenca o un método de pesca, ya que contiene atributos *enum* para las cuencas y los métodos de pesca.

Imagen que contiene Tabla

Descripción generada automáticamente

**Implementación de operaciones CRUD – 30%**

Realice una aplicación en Windows Forms C#.NET que demuestre las operaciones de CRUD requeridas. Si el examen anterior lo realizó en otro lenguaje (Java, Python) puede realizar este también en la misma tecnología.

La solución de esta implementación está adjunta y en el repositorio compartido. No obstante, hay evidencia de la aplicación funcionando en formato de video, también adjunto.

**Comparativo de la capa de datos – 30%**

Realice una comparación de cada una de las operaciones CRUD, presentando su opinión sobre cual paradigma de base de datos es el más pertinente para este dominio de problema en particular.

**READ SQL / SELECT SQL**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**READ NOSQL / SELECT NOSQL**

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

En ambos casos, la conexión a la base de datos se realiza de manera muy similar y, pese a que las sentencias se realizan de forma distinta, se puede decir que implica un coste algorítmico similar, debido a que en ambos casos se está realizando una iteración que depende del número de registros o documentos que hay en las tablas o colecciones.

**CREATE SQL**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**CREATE NOSQL / INSERT**

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

Antes del análisis, vale la pena mencionar que, para la creación de registros en SQL, perfectamente se pudo haber utilizado solo un método, en lugar de dos, como lo realicé. Teniendo en cuenta esto, se evidencia que es más sencillo realizar la creación de registros SQL por varias razones:

1. El modelo SQL admitía la fecha en formato texto, por lo que no existió una validación para ese tipo de dato en ese modelo.
2. En el modelo NOSQL se realizaron más validaciones para la creación de documentos.

Ambos casos tienen un coste algorítmico similar, siendo el caso del método NOSQL el más costoso debido a que está llamando más métodos para validaciones y conversiones.

**UPDATE SQL**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**UPDATE NOSQL**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La actualización de registros o documentos, según el paradigma que se utiliza, es muy similar al proceso de creación de registros o documentos.

Al igual que en la creación de registros o documentos, el método en SQL se dividió en 2 métodos, pero pudo haberse logrado en un solo método. Teniendo esto en cuenta, al igual que con la creación, la actualización tiene un coste algorítmico mayor para el paradigma NOSQL debido a algunas validaciones que se tuvieron en cuenta y a la cantidad de métodos que llama ese mismo método de actualización.

**DELETE SQL**

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

**DELETE NOSQL**

****

En el caso de la eliminación de registros o documentos, se puede decir que lo único que varía es la forma en la que se dice directamente a la base de datos que ejecute la acción. En ambos casos se realizan validaciones y se llaman métodos externos para toda la lógica creada, pero sigue siendo un poco más costo hablando de tiempo de procesamiento, el método para el paradigma NOSQL.

**Conclusión**

En ambos paradigmas se prestan perfectamente para cumplir con el dominio del problema y lo que se solicitó. Es verdad también que la forma de dar respuesta a lo solicitado bien pudo haber sido de otra manera, ya que hay muchas formas de realizar exactamente lo mismo.

Para este dominio del problema, y lo que se solicitó particularmente, al tratarse de ser tan pocas tablas, con tan pocas variables, la opción más adecuada es el paradigma SQL, debido a que facilita el desarrollo de la solución. Adicionalmente, el modelo de datos está muy bien establecido y no es dinámico o variable, por lo que no hay riesgos significativos respecto a la integridad de los datos y las tablas, ya que siempre serán las mismas tablas, con los mismos atributos.

Si hubieran existido muchas más tablas, o atributos variables dentro del modelo de datos, entonces en ese caso el paradigma NOSQL era perfecto, pero como se mencionó, no fue el caso.

El paradigma más pertinente para este dominio del problema, tal y como se presentó, es el paradigma SQL.