Informe Seminario Bases de Datos NoSQL
Grupo 29

Integrantes:

- **Olea Osvaldo Juan**
- **Ricci Braian**
- **Saldain Gerardo**
- **Sánchez Fiadone Gonzalo Nicolás**
-

Fechas de entrega: 30/09/24.

Objetivos:

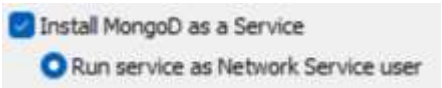
El objetivo del presente trabajo es estudiar el uso de bases de datos NoSQL, en particular MongoDB, en la implementación de un sistema de administración de ventas para una plataforma de e-commerce.
Particularmente nos centraremos en la flexibilidad en el modelado de datos de MongoDB, un aspecto básico, pero una ventaja esencial sobre las db tradicionales.
MongoDB también ofrece soluciones muy efectivas de gran escalabilidad horizontal y la capacidad de manejar grandes volúmenes de datos se beneficia tremendamente de la **no estructuración** de estos.
En nuestro ejemplo, nos acotaremos a ejemplificar las operaciones CRUD referentes a productos en una e-commerce de prendas de vestir.

Pasos de instalación:

1. Ingresamos al sitio web oficial de MongoDB para descargar el instalador, seleccionando nuestro sistema operativo y preferentemente la última versión: https://www.mongodb.com/try/download/community

2. Una vez descargado, ejecutamos el instalador en modo '**Complete**' :

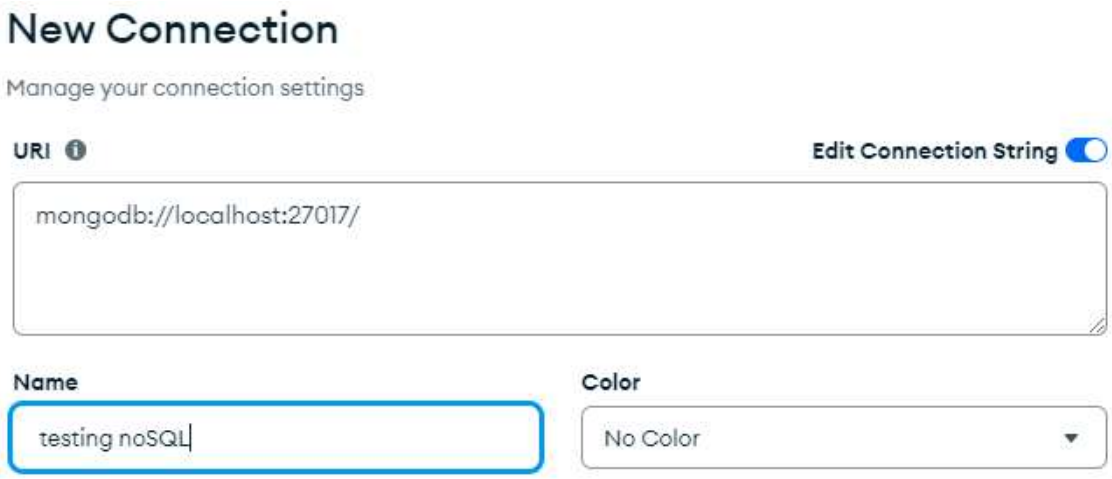
3. Instalamos MongoDB como un servicio web:



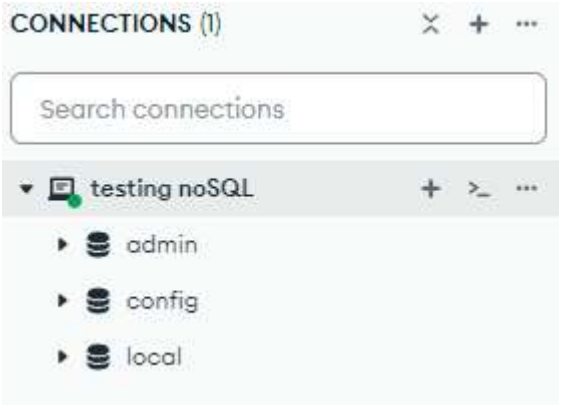
4. Para garantizarnos una GUI, nos aseguramos de tildar la opción de instalar **MongoDB Compass**, indicamos next y finalmente install. Es posible que durante la instalación se nos soliciten permisos de administración.

5. Completada la instalación, abrimos MongoDB Compass (si no se abrió de forma automática) y hacemos click en '**+ Add new connection':**

6. Aquí podremos ingresar los detalles de conexión a la instancia local especificando la dirección del servidor (por defecto '**mongodb://localhost:27017**') y darle un nombre, luego seleccionamos '**Connect**':



7. Ahora podemos ver en la barra lateral, la coneccion que acabamos de crear:



8. Utilizamos la consola **MongoDB Shell** integrada y ejecutamos el comando '**db.version()**' para verificar que MongoDB está corriendo**.** Podríamos utilizar la interfaz gráfica, pero a partir de ahora trabajaremos directamente en la consola para familiarizarnos con los comandos propios de Mongo:

Testeos realizados:

1. Empleamos el comando '**use ecommerce**' para crear una base de datos con ese nombre si no existe previamente.

2. Creamos la colección para almacenar productos de la siguiente forma: **'db.createCollection("productos")'**

```
>_MONGOSH

> use ecommerce
< switched to db ecommerce
> db.createCollection("productos")
< { ok: 1 }
ecommerce>
```

   Si volvemos a la interfaz y damos '**Refresh**' podremos ver en la barra lateral que se creó nuestra base de datos con la colección '**productos**'.

3. Podemos insertar varios documentos en la colección con un comando con el siguiente formato:

```
ecommerce> db.productos.insertMany([
            {id: 01, nombre: "Remera Deportiva", precio: 10200, stock: 100, talle: "XL", color: "azul"},
            {id: 542, nombre: "Pantalon Jean", precio: 25000, stock: 55, talle: 4, marca: "denim"}
          ])
```

   O también podemos agregar una única entrada con el comando '**db.productos.insertOne()**'.

4. Consultamos todos los documentos registrados en la colección empleando el comando '**db.productos.find()**', que se verán reflejados de forma legible con una estructura indentada (en formato JSON) gracias al empleo de la GUI:

```
> db.productos.find()
< {
    _id: ObjectId('66f8fdacadfd53e47a17fc02'),
    id: 1,
    nombre: 'Remera Deportiva',
    precio: 10200,
    stock: 100,
    talle: 'XL',
    color: 'azul'
  }
  {
    _id: ObjectId('66f8fdacadfd53e47a17fc03'),
    id: 542,
    nombre: 'Pantalon Jean',
    precio: 25000,
    stock: 55,
    talle: 4,
    marca: 'denim'
  }
```

5. También podemos buscar un registro en específico utilizando '**db.productos.findOne({ color: "azul" })**'.
   O utilizar algo como '**db.productos.find({ precio: { $gt: 10000 } })**', dónde *$gt* significa *greater than,* y buscará todos las entradas que cumplan con el parámetro.

   Mongo incluye numeroso operadores:
   $gt: *greater than*, más que.
   $lt: *less than*, menos que.
   $eq: *equal to*, igual a.
   $ne: *not equal to*, no igual a.
   $and, **$or**: operadores lógicos que combinan otras sentencias.
   etc.

   También podemos limitar los campos retornados en un query, agregando un **1** para mostrarlos, o un **0** para excluirlos.
   Agregando **{ nombre: 1 }:**

```
> db.productos.find({ precio: { $gt: 10000 }}, {nombre: 1})
< {
    _id: ObjectId('66f8fdacadfd53e47a17fc02'),
    nombre: 'Remera Deportiva'
  }
  {
    _id: ObjectId('66f8fdacadfd53e47a17fc03'),
    nombre: 'Pantalon Jean'
  }
```

   Nótese que el id interno de la db ('**_id**', conocido como **ObjectId**) se muestra por defecto, pero usando la siguiente sentencia indicando **{ nombre:1, _id:0 }**, podemos omitirlo:

```
> db.productos.find({ precio: { $gt: 10000 } }, { nombre:1, _id:0 })
< {
    nombre: 'Remera Deportiva'
  }
  {
    nombre: 'Pantalon Jean'
  }
```

6. Finalmente, con una sentencia como la siguiente, podemos incluir referencias a otros documentos de otras colecciones, en este ejemplo, referencias a comentarios que tiene un determinado producto, que a su vez también referencian el usuario que hizo el comentario, documentos que podrían ser parte de las colecciones '**comentarios**' y '**usuarios**' respectivamente.

```
db.productos.insertOne({
  nombre: "Campera Rompevientos",
  precio: 15500,
  stock: 20,
  reviews: [
    {
      _id: ObjectId("6514a3d7d458f00016f9a6c1"),
      userId: ObjectId("6514a3d7d458f00016f9a6a1"),
      comment: "Excelente campera!.",
      rating: 5
    },
    {
      _id: ObjectId("6514a3d7d458f00016f9a6c2"),
      userId: ObjectId("6514a3d7d458f00016f9a6a2"),
      comment: "Buen rendimiento, buenos materiales.",
      rating: 4
    }
  ]
})
```

Conclusiones:

Solo con esta simple inducción podemos apreciar el nivel extremo de flexibilidad que los documentos de MongoDB proporcionan, permitiendo almacenar de manera formateada y recuperable, items con diversas propiedades y tipos, sin la necesidad de modificar la base de datos para conformar a nuevas propiedades de los objetos.
Consideramos también muy prácticas estas características a la hora de implementar una base de datos en un sistema backend orientado a objetos (como los que estamos viendo en la asignatura Arquitecturas Web) flexibilizando el modelado de las entidades/objetos que usamos en codigo y como se relacionan.
Igual de importante, quedamos impresionados con la facilidad para referenciar otras colecciones y objetos dentro de la db, sin la necesidad de definir claves foráneas u otros requerimientos de integridad.
Finalmente, sentimos que la combinación de funciones y formato json para las operaciones CRUD en Mongo, tiene una gran ventaja en cuanto a aprendizaje y adopción, siendo que en bases tradicionales existe la barrera de aprender la sintaxis específica del lenguaje SQL.