Universidad Nacional Autónoma de México
Instituto de Investigaciones en Matemáticas
Aplicadas y en Sistemas (IIMAS)
Licenciatura en Ciencia de Datos

BASES DE DATOS NO ESTRUCTURADAS
EJERCICIO PRÁCTICO 08

1. INTERACTUANDO CON MONGODB Y JAVA

1.1. Creación del programa básico

En este ejercicio se realizará la creación de una pequeña aplicación Java que ilustra la forma en la que se puede interactuar con la *replica set* de MongoDB creada en el ejercicio práctico anterior.

1.1.1. Creación de un proyecto Java en IntelliJ

A. Similar al procedimiento realizado para la interacción de Cassandra con Java, crear un proyecto Gradle llamado `<iniciales>-mongodb`.
B. En el archivo `build.gradle` agregar las siguientes dependencias que permiten agregar el driver de mongodb encargado de realizar la conexión con la replica set

```
dependencies {
    testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'
    testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'
    implementation 'org.mongodb:mongodb-driver-sync:5.0.1'

    runtimeOnly 'ch.qos.logback:logback-classic:1.5.0'
    implementation 'org.slf4j:slf4j-api:2.0.12'
}
```

C. No olvidar agregar el archivo `logback.xml` similar al agregado en el ejercicio práctico con Cassandra. Este archivo se emplea para configurar la salida de mensajes a bitácora. En este caso a la salida estándar.
D. Crear la clase `iimas.bdne.jrc.MongoClientUtils` encargada de establecer la conexión hacia la replica set. Modificar los valores de conexión como se indica en el siguiente código.

```
public class MongoClientUtils {
  //TODO: actualizar el hostname de cada nodo según corresponda
  private static final String
   URI="mongodb://jrc-mongo1:27017,jrc-mongo2:27017,jrc-mongo3:27017,";

  private static final Logger LOG =
    LoggerFactory.getLogger(MongoClientUtils.class);

  public static MongoClient createMongoClient(){
    final ConnectionString cs =  new ConnectionString(URI);
    MongoClient client = MongoClients.create(cs);
    return client;
  }
```

```
  public static void main(String[] args) {
    LOG.debug("Verificando instancia de  MongoClient");
    try(MongoClient mongoClient = createMongoClient()){
      LOG.debug("Cluster description: {}",mongoClient.getClusterDescription());
      MongoIterable<String> dbList = mongoClient.listDatabaseNames();
      LOG.debug("Mostrando BDs ");
      for (String db:dbList) {
        LOG.debug("Db: {}",db);
      }
      LOG.debug("{}",mongoClient.listDatabaseNames());
      LOG.debug(
        "Al final del bloque try, se cierra la instancia de MongoClient");
    }
  }
}
```

- El método `creaMongoClient` se encarga de realizar la conexión con la replica set y obtener una instancia del tipo `MongoClient`. A partir de este objeto se podrá acceder a una base de datos y a su vez a una colección en particular
- Revisar el método main. Se realiza una pequeña prueba para verificar la correcta conexión. Se imprime la descripción del replica set, y la lista de bases de datos que existen.
- Notar que la generación de la instancia MongoClient se realiza dentro de un bloque try-catch de tipo `Autocloseable`. Esto significa que de forma implícita se ejecuta `mongoClient.close()` justo al terminar el bloque try-catch. Es ***importante*** cerrar la conexión una vez que ya no se requiera interactuar con MongoDB. ***C1. Incluir en la entrega*** la salida de ejecución de esta clase.

E. Crear la clase `iimas.bdne.jrc.OnlineLibraryExamples`. Este código incluye algunos ejemplos de operaciones básicas con MongoDB. Notar que se hace uso de la clase creada en el punto anterior para obtener conexiones hacia MongoDB

```
public class OnlineLibraryExamples {

  private static final String DB_NAME = "OnlineLibrary";
  private static final String COLLECTION_NAME = "books";
  private static final Logger LOG =
    LoggerFactory.getLogger(MongoClientUtils.class);
```

```
/**
 * Ejemplo de creación de un libro a través de este documento JSON
 * {
 *   "_id": 1,
 *   "title": "Unlocking Android",
 *   "isbn": "1933988673",
 *   "pageCount": 416,
 *   "publishedDate": {
 *     "$date": "2009-04-01T00:00:00.000-0700"
 *   },
 *   "thumbnailUrl": "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-images/ableson.jpg",
 *   "shortDescription": "Unlocking Android: A Developer's Guide",
 *   "longDescription": "Android is an open source mobile phone platform based on the Linux operating system.",
 *   "status": "PUBLISH",
 *   "authors": [
 *     "W. Frank Ableson",
 *     "Charlie Collins",
 *     "Robi Sen"
 *   ],
 *   "categories": [
 *     "Open Source",
 *     "Mobile"
 *   ]
 * }
 */
private static void createBook(MongoCollection<Document> collection) {

  Document doc = new Document("_id", new ObjectId());
  doc.append("title", "Unlocking Android");
  doc.append("isbn", "1933988673");
  doc.append("pageCount", 416);
  doc.append("publishedDate", LocalDate.of(2009, 04, 01));
  doc.append("thumbnailUrl",
    "https://s3.amazonaws.com/AKIAJC5RLADLUMVRPFDQ.book-thumb-images/ableson.jpg");
  doc.append("shortDescription", "Unlocking Android: A Developer's Guide");
  doc.append("longDescription",
    "Android is an open source mobile phone platform based on the Linux operating system");
  doc.append("status","PUBLISH");
  doc.append("authors", List.of(
    "W. Frank Ableson","Charlie Collins","Robi Sen"));
  doc.append("categories",List.of("Open source","Mobile"));
  LOG.debug("Creando un nuevo libro ");
  collection.insertOne(doc);
  LOG.debug("Libro creado con Id : {}",doc.get("_id"));
}
```

```
  public static void showBook(MongoCollection<Document> collection){
    FindIterable<Document> docs =
      collection.find(new Document("title","Unlocking Android"));
    docs.forEach(doc -> {
      LOG.debug("Documento encontrado");
      LOG.debug("{}",doc.toString());
    });
  }

  public static void deleteBook(MongoCollection<Document> collection){
    collection.deleteOne(new Document("title","Unlocking Android"));
  }

  public static void main(String[] args) {
    LOG.debug("Obteniendo referencia a la colección");
    try (MongoClient client = MongoClientUtils.createMongoClient()) {
      MongoDatabase db = client.getDatabase(DB_NAME);
      MongoCollection collection = db.getCollection(COLLECTION_NAME);

      LOG.debug("Ejemplo - create");
     createBook(collection);

      LOG.debug("Ejemplo - leer documento");
      showBook(collection);

      LOG.debug("Ejemplo - Eliminar documento");
      deleteBook(collection);

    }
  }
}
```

- La clase define 3 métodos `createBook`, `showBook`, `deleteBook` empleados para realizar operaciones básicas con en una base de datos llamada `OnlineLibrary` para una colección `books`.
- Notar que se emplean objetos tipo `org.bson.Document` para crear documentos JSON.
- En el método `main` se invocan a estos 3 métodos para verificar su funcionamiento.

Realizar las siguientes actividades:

A. Buscar un dataset en formato JSON.
B. Actualizar los 3 métodos anteriores para crear, mostrar o eliminar un documento. Actualizar el nombre de método con el nombre del objeto seleccionado, es decir,

en lugar de `createBook`, cambiar a `create<Objeto>` ,sustituyendo `<Objeto>` con el nombre real. ***C2. Incluir en la entrega*** la salida de estos 3 métodos.

C. Agregar los siguientes métodos:

- `createMany<Objetos>` En este método se deberán crear varios documentos JSON del dataset seleccionado. Sustituir `<Objetos>` con el nombre del objeto seleccionado.
- `showMany<Objetos>` En este método se deberá realizar una consulta que genere varios documentos. Emplear un cursor, es decir, emplear `collection.find().cursor()`.
- `update<Objeto>` En este método realizar la actualización de un documento.
- ***C3. Incluir en la entrega*** el código de estos métodos así como la salida de ejecución

1.2. Contenido de la entrega

A. Elementos comunes de los ejercicios prácticos.
B. C1. Salida de ejecución del comando `docker compose` que permite la creación del cluster con mongodb
C. C2. Salida de ejecución de los métodos `create<Objeto>`, `show<Objeto`, `delete<Objeto>`
D. C3. Código y salida de los métodos `createMany<Objetos>, showMany<Objetos>, update<Objeto>` .