Ejercicio 9 – Operaciones CRUD en MongoDB con Java

En esta práctica vamos a realizar una implementación muy similar a la que hicimos en el ejercicio 6 cuando implementamos operaciones CRUD en una base de datos SQLite.

En este caso, trabajaremos con la base de datos MongoDB instalada en una instancia AWS, pero el funcionamiento es muy similar al realizado en esa práctica de SQLite y en la siguiente de PostgreSQL.

Parte 1: Crear base de datos y usuario para manipularla

Dentro de la instancia que contiene MongoDB, acceder a su shell mediante el comando ***mongosh***. Si realizaste el apartado de seguridad de la práctica anterior, requerirá autenticación con un comando similar a ***mongosh -u usuario -p --authenticationDatabase admin***.

Vamos a crear la base de datos con el comando ***use f1-2006***. La base de datos estará vacía y será borrada si al finalizar la sesión no tiene datos.

A continuación, vamos a crear un usuario no administrador que podrá hacer lecturas y escrituras sobre esa base de datos:

db.createUser({user: "usuario", pwd: "mipassword", roles: [{role: "readWrite", db: "f1-2006"}] })

```
ubuntu@ip-172-31-20-161:~$ mongosh -u adminAlejandro -p --authenticationDatabase admin
Enter password: ****
Current Mongosh Log ID: 6565b8a10bbe34b1e76c3930
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&authSo
urce=admin&appName=mongosh+2.1.0
Using MongoDB:          7.0.4
Using Mongosh:          2.1.0

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

test> use f1-2006
switched to db f1-2006
f1-2006> db.createUser({user:"alejandro", pwd: passwordPrompt(), roles: [{ role: "readWrite", db: "f1-2006"}]})
Enter password
1234
****{ ok: 1 }
```

Parte 2: Importar los datos de un archivo JSON

La herramienta de MongoDB, **mongoimport**, es una herramienta bastante potente que nos permite realizar importaciones en MongoDB.

Como en nuestro caso requerimos de autenticación, la sintaxis del comando a utilizar es la siguiente:

mongoimport --username=usuario --db=base_datos --collection=coleccion --file=fichero.json

Puedes hacerlo también desde un sitio remoto, añadiendo el parámetro ***--host***.

```
ubuntu@ip-172-31-20-161:~$ mongoimport --username=alejandro --db=f1-2006 --collection=drivers --file=drivers.json
Enter password for mongo user:

2023-11-28T10:13:40.641+0000    connected to: mongodb://localhost/
2023-11-28T10:13:40.661+0000    27 document(s) imported successfully. 0 document(s) failed to import.
```

A partir de este momento, ya podrás consumir la base de datos desde tu aplicación Java usando las clases adecuadas.

Ejercicio práctico

De la misma forma que hiciste en el ejercicio 6, crea una aplicación con:

- Una clase **Piloto** que facilite la lectura y escritura de pilotos en la colección ***drivers***. Crea también la clase **Escuderia**, que pueda parsear la información contenida de una Escudería. Ten en cuenta, respecto a los ejercicios que realizamos con bases de datos relacionales, que las escuderías no tienen un identificador numérico.

- Crea una clase **OperacionesCRUDPilotos**, la cual disponga de los siguientes métodos:

    - ***CrearPiloto()***, que reciba un objeto Piloto y lo añada a la base de datos.

    - ***LeerPiloto()***, que reciba un entero y devuelva un objeto Piloto con la información del piloto con el driverid coincidente.

    - ***LeerPilotos()***, que devuelva un listado completo de objetos Piloto.

    - ***ActualizarPiloto()***, que reciba un objeto Piloto y actualice los datos del registro coincidente en la base de datos con el mismo driverid.

    - ***BorrarPiloto()***, que reciba un objeto Piloto y lo elimine de la base de datos.

- Agrega también dos métodos adicionales:

    - ***MostrarPilotosOrdenadoresPorEdadDescendente()***, que muestre el listado de pilotos ordenador con edad de mayor a menor y su edad en el inicio de la temporada (año 2006).

    - ***MostrarPilotosConEdadMayorQue()***, que reciba como parámetro un entero y muestre el listado de pilotos con edad mayor o igual que ese entero en el inicio de la temporada (año 2006) ordenados de mayor a menor edad.

Si has acabado las tareas anteriores, puede ser muy interesante investigar sobre el uso del campo ***driverid*** cuando MongoDB utiliza su propio ***_id*** de tipo ***ObjectId***. Uno de los problemas de asignar un nuevo driverid a un nuevo Piloto sería tener en cuenta que sería autoincrementable y único. ¿Cómo puede solucionarse esto en MongoDB? ¿Tiene sentido ese driverid?