Ejercicio Mongo Java

Ivan Nuñez Rodriguez

2º DAM

Acceso a Datos

Índice:

Código:

Clase AlumnoDAO

```java
package dao;

import com.mongodb.client.MongoCollection;
import database.ConexionDB;
import model.Alumno;
import org.bson.Document;

import java.util.Scanner;

public class AlumnoDAO {
    private final MongoCollection<Alumno> alumnosCollection;
    private final Scanner scanner = new Scanner(System.in);

    // constructor que resibe una conexion a la base de datos
    public AlumnoDAO(ConexionDB conexionDB) {
        this.alumnosCollection = conexionDB.getAlumnosCollection();
    }

    // metodo para insertar un nuevo alumno en la coleccion
    public void insertarAlumno() {
        System.out.println("insertando un nuevo alumno...");
        Alumno alumno = new Alumno();
        System.out.print("nombre: ");
        alumno.setName(scanner.nextLine());
        System.out.print("edad: ");
        alumno.setAge(Integer.parseInt(scanner.nextLine()));
        System.out.print("género: ");
        alumno.setGender(scanner.nextLine());
        System.out.print("email: ");
        alumno.setEmail(scanner.nextLine());
        System.out.print("teléfono: ");
        alumno.setPhone(scanner.nextLine());
        System.out.print("calificación: ");
        alumno.setCalification(Integer.parseInt(scanner.nextLine()));
        System.out.print("grado superior: ");
        alumno.setHigherGrade(scanner.nextLine());
        System.out.print("rating: ");
        alumno.setRating(Double.parseDouble(scanner.nextLine()));
        alumnosCollection.insertOne(alumno);
        System.out.println("alumno insertado correctamente.");
    }

    // metodo para mostrar todos los alumnos
    public void mostrarAlumnos() {
        System.out.println("mostrando todos los alumnos...");
        for (Alumno alumno : alumnosCollection.find()) {
            System.out.println(alumno);
        }
    }

    // metodo para buscar un alumno por su email
    public void buscarAlumno() {
        System.out.print("ingrese el email del alumno a buscar: ");
```

```
        String email = scanner.nextLine();
        Alumno alumno = alumnosCollection.find(new Document("email",
email)).first();
        if (alumno != null) {
            System.out.println(alumno);
        } else {
            System.out.println("alumno no encontrado.");
        }
    }

    // metodo para dar de baja a los alumnos xon calificacion 5 o
superior
    public void darBajaAlumnos() {
        alumnosCollection.deleteMany(new Document("calification", new
Document("$gte", 5)));
        System.out.println("alumnos con calificación 5 o superior
eliminados correctamente.");
    }
}
```

Clase ProfesorDAO

```
package dao;

import com.mongodb.client.MongoCollection;
import database.ConexionDB;
import model.Profesor;

import java.util.Arrays;
import java.util.Scanner;

public class ProfesorDAO {
    private final MongoCollection<Profesor> profesoresCollection;
    private final Scanner scanner = new Scanner(System.in);

    // constructor que resibe una conexion a la base de datos
    public ProfesorDAO(ConexionDB conexionDB) {
        this.profesoresCollection =
conexionDB.getProfesoresCollection();  // ahora se espera
MongoCollection<Profesor>
    }

    // metodo para insertar un nuevo profesor en la coleccion
    public void insertarProfesor() {
        System.out.println("insertando un nuevo profesor...");
        Profesor profesor = new Profesor();
        System.out.print("nombre: ");
        profesor.setNombre(scanner.nextLine());
        System.out.print("edad: ");
        profesor.setEdad(Integer.parseInt(scanner.nextLine()));
        System.out.print("género: ");
        profesor.setGender(scanner.nextLine());
        System.out.print("email: ");
        profesor.setEmail(scanner.nextLine());
        System.out.print("teléfono: ");
        profesor.setPhone(scanner.nextLine());
        System.out.print("título: ");
        profesor.setTitle(scanner.nextLine());
        System.out.print("materias (separadas por comas): ");
```

```
profesor.setSubjects(Arrays.asList(scanner.nextLine().split(",")));
        System.out.print("rating: ");
        profesor.setRating(Double.parseDouble(scanner.nextLine()));
        profesoresCollection.insertOne(profesor);
        System.out.println("profesor insertado correctamente.");
    }

    // metodo para mostrar todos los profesores
    public void mostrarProfesores() {
        System.out.println("mostrando todos los profesores...");
        for (Profesor profesor : profesoresCollection.find()) {
            System.out.println(profesor);
        }
    }

    // metodo para buscar profesores por rango de edad
    public void buscarProfesor() {
        System.out.print("ingrese el rango de edad (mínima y máxima
separadas por coma): ");
        String[] rangos = scanner.nextLine().split(",");
        int edadMin = Integer.parseInt(rangos[0].trim());
        int edadMax = Integer.parseInt(rangos[1].trim());
        for (Profesor profesor : profesoresCollection.find(new
org.bson.Document("age", new org.bson.Document("$gte",
edadMin).append("$lte", edadMax)))) {
            System.out.println(profesor);
        }
    }

    // metodo para actualizar la calificacion de un profesor
    public void actualizarProfesor() {
        System.out.print("ingrese el email del profesor a actualizar:
");
        String email = scanner.nextLine();
        System.out.print("ingrese la nueva calificación: ");
        double nuevaCalificacion =
Double.parseDouble(scanner.nextLine());
        org.bson.Document update = new org.bson.Document("$set", new
org.bson.Document("rating", nuevaCalificacion));
        profesoresCollection.updateOne(new org.bson.Document("email",
email), update);
        System.out.println("profesor actualizado correctamente.");
    }
}
```

ConexionDB

```
package database;

import com.mongodb.MongoClientSettings;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import model.Alumno;
import model.Profesor;
import org.bson.codecs.configuration.CodecProvider;
import org.bson.codecs.configuration.CodecRegistries;
import org.bson.codecs.configuration.CodecRegistry;
import org.bson.codecs.pojo.PojoCodecProvider;
```

```
public class ConexionDB {
    private MongoClient mongoClient;
    private MongoDatabase database;
    private MongoCollection<Alumno> alumnosCollection;
    private MongoCollection<Profesor> profesoresCollection;

    // constructor que inicializa la conexion a la base de datos
    public ConexionDB() {

        // configurar el proveedor de codecs para soportar pojos
        CodecProvider pojoCodecProvider =
PojoCodecProvider.builder().automatic(true).build();
        CodecRegistry pojoCodecRegistry =
CodecRegistries.fromRegistries(
                MongoClientSettings.getDefaultCodecRegistry(),
                CodecRegistries.fromProviders(pojoCodecProvider));
        MongoClientSettings settings = MongoClientSettings.builder()
                .applyConnectionString(new
com.mongodb.ConnectionString(DBScheme.LINK_CONEXION))
                .codecRegistry(pojoCodecRegistry)
                .build();

        // crear el cliente mongo usando la configuracion
        mongoClient = MongoClients.create(settings);

        // obtener la base de datos "centro_estudios"
        database = mongoClient.getDatabase(DBScheme.DATABASE);

        // obtener las colecciones especificando el tipo de pojo
        alumnosCollection =
database.getCollection(DBScheme.COLECION_ALUMNOS, Alumno.class);
        profesoresCollection =
database.getCollection(DBScheme.COLECCION_PROFESOR, Profesor.class);
    }

    // metodo para obtener la coleccion de alumnos
    public MongoCollection<Alumno> getAlumnosCollection() {
        return alumnosCollection;
    }

    // metodo para obtener la coleccion de profesores
    public MongoCollection<Profesor> getProfesoresCollection() {
        return profesoresCollection;
    }

    // metodo para cerrar la conexion a la base de datos
    public void cerrarConexion() {
        mongoClient.close();
    }
}
```

DBScheme

```
package database;

public interface DBScheme {

    String USER= "unir";
    String PASS= "unir";
```

```
    String DATABASE = "centro_estudios";
    String COLECION_ALUMNOS = "alumnos";
    String COLECCION_PROFESOR = "profesores";
    String
LINK_CONEXION="mongodb+srv://"+USER+":"+PASS+"@cluster0.qrerh.mongodb.
net/?retryWrites=true&w=majority&appName=Cluster0";
}
```

Alumno

```
package model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.bson.codecs.pojo.annotations.BsonProperty;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Alumno {

    @BsonProperty("name")
    private String name;  // El nombre del alumno

    @BsonProperty("age")
    private int age;  // Edad del alumno

    @BsonProperty("gender")
    private String gender;  // Género del alumno

    @BsonProperty("email")
    private String email;  // Correo electrónico del alumno

    @BsonProperty("phone")
    private String phone;  // Número de teléfono

    @BsonProperty("calification")
    private int calification;  // Calificación del alumno

    @BsonProperty("higher_grade")
    private String higherGrade;  // Nivel superior del alumno (e.g.,
DAM, DAW)

    @BsonProperty("rating")
    private double rating;  // Valoración del alumno

    @Override
    public String toString() {
        return "Alumno{" +
                "name='" + name + '\'' +
                ", age=" + age +
                ", gender='" + gender + '\'' +
                ", email='" + email + '\'' +
                ", phone='" + phone + '\'' +
                ", calification=" + calification +
                ", higherGrade='" + higherGrade + '\'' +
                ", rating=" + rating +
```

```
                '}';
    }
}
```

Profesor

```
package model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.bson.codecs.pojo.annotations.BsonProperty;

import java.io.Serializable;
import java.util.List;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Profesor implements Serializable {
    @BsonProperty("name")
    private String nombre;

    @BsonProperty("age")
    private int edad;

    @BsonProperty("gender")
    private String gender;

    @BsonProperty("email")
    private String email;

    @BsonProperty("phone")
    private String phone;

    @BsonProperty("subjects")
    private List<String> subjects;

    @BsonProperty("title")
    private String title;

    @BsonProperty("rating")
    private double rating;

    @Override
    public String toString() {
        return "Profesor{" +
                "nombre='" + nombre + '\'' +
                ", edad=" + edad +
                ", gender='" + gender + '\'' +
                ", email='" + email + '\'' +
                ", phone='" + phone + '\'' +
                ", subjects=" + subjects +
                ", title='" + title + '\'' +
                ", rating=" + rating +
                '}';
    }
}
```

CentroEstudios

```
import dao.AlumnoDAO;
import dao.ProfesorDAO;
import database.ConexionDB;

import java.util.Scanner;

public class CentroEstudios {
    private final Scanner scanner = new Scanner(System.in);
    private final ProfesorDAO profesorDAO;
    private final AlumnoDAO alumnoDAO;

    // constructor qe inicializa la conexion y los daos
    public CentroEstudios() {
        ConexionDB conexionDB = new ConexionDB();
        this.profesorDAO = new ProfesorDAO(conexionDB);
        this.alumnoDAO = new AlumnoDAO(conexionDB);
        ejecutarMenu();
        conexionDB.cerrarConexion();
    }

    // metodo main para iniciar el programa
    public static void main(String[] args) {
        new CentroEstudios();
    }

    // metodo quee ejecuta el menu principal
    public void ejecutarMenu() {
        int option;
        do {
            showMenu();
            option = Integer.parseInt(scanner.nextLine());
            switch (option) {
                case 1 -> profesorDAO.insertarProfesor();
                case 2 -> alumnoDAO.insertarAlumno();
                case 3 -> mostrarTodos();
                case 4 -> profesorDAO.mostrarProfesores();
                case 5 -> alumnoDAO.mostrarAlumnos();
                case 6 -> alumnoDAO.buscarAlumno();
                case 7 -> profesorDAO.buscarProfesor();
                case 8 -> profesorDAO.actualizarProfesor();
                case 9 -> alumnoDAO.darBajaAlumnos();
                case 10 -> System.out.println("saliendo...");
                default -> System.out.println("opción inválida, por
favor intente nuevamente.");
            }
        } while (option != 10);
    }

    // metodo para mostrar el menu de opciones
    private void showMenu() {
        System.out.println("\nmenú de opciones:");
        System.out.println("1- insertar un profesor");
        System.out.println("2- insertar un alumno");
        System.out.println("3- mostrar todos los datos");
        System.out.println("4- mostrar profesores");
        System.out.println("5- mostrar alumnos");
        System.out.println("6- buscar alumno por mail");
```

```
        System.out.println("7- buscar profesor por edad");
        System.out.println("8- actualizar calificación profesor");
        System.out.println("9- dar de baja alumnos con suspenso");
        System.out.println("10- salir");
        System.out.print("seleccione una opción: ");
    }

    // metodo para mostrar todos los profesores y alumnos
    private void mostrarTodos() {
        System.out.println("mostrando todos los profesores y
alumnos...");
        profesorDAO.mostrarProfesores();
        alumnoDAO.mostrarAlumnos();
    }
}
```

Ejecución y resultado.

Ejecución de CentroEstudios

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción:
```

1- **Insertar un profesor: tras seleccionar esta opción se pedirá al usuario la introducción de todos los datos**

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 1
insertando un nuevo profesor...
nombre: alba
edad: 50
género: female
email: alba@estu.com
teléfono: 345678
título: DAM
materias (separadas por comas): fol,eie
rating: 10
profesor insertado correctamente.
```

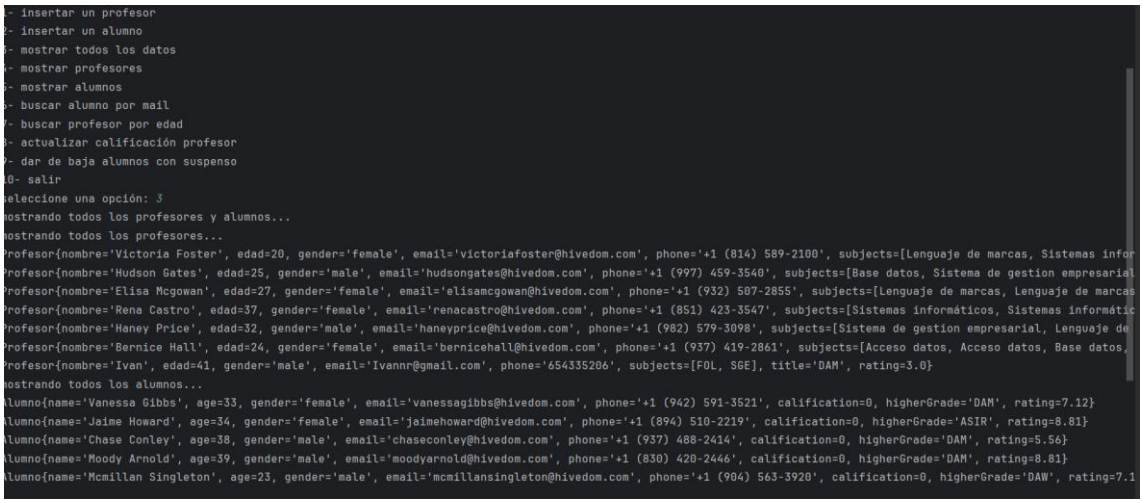
```
Profesor{nombre='Ivan', edad=41, gender='male', email='Ivannr@gmail.com', phone='654335206', subjects=[FOL, SGE], title='DAM', rating=3.0}
Profesor{nombre='alba', edad=50, gender='female', email='alba@estu.com', phone='345678', subjects=[fol, eie], title='DAM', rating=10.0}
```

2- Insertar un alumno: tras seleccionar esta opción se pedirá al usuario la introducción de todos los datos

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 2
insertando un nuevo alumno...
nombre: miguel
edad: 18
género: male
email: miguel@noestu.com
teléfono: 123456
calificación: 8
grado superior: DAW
rating: 9
alumno insertado correctamente.
```

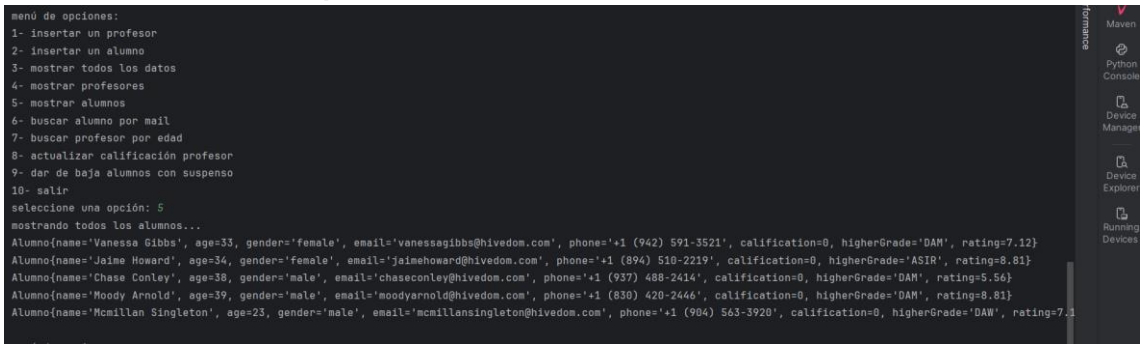
```
Alumno{name='Mcmillan Singleton', age=23, gender='male', email='mcmillansingleton@hivedom.com', phone='+1 (904) 563-3920', calification=0, higherGrad
Alumno{name='miguel', age=18, gender='male', email='miguel@noestu.com', phone='123456', calification=8, higherGrade='DAW', rating=9.0}
```

3- Mostrar todos los datos: se imprimirán todos los datos, tanto de usuarios como de profesores



4- Mostrar profesores: se imprimirá solo los datos de los profesores

5- Mostrar alumnos: se imprimirá solo los datos de los alumnos



6- Buscar alumno: se pedirá un email y se mostrará el alumno que cumpla la condición de búsqueda

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 6
ingrese el email del alumno a buscar: miguel@noestu.com
Alumno{name='miguel', age=18, gender='male', email='miguel@noestu.com', phone='123456', calification=8, higherGrade='DAW', rating=9.0}
```

7- Buscar profesor: se pedirá un rando de edad y se mostrará solo aquellos que cumpla la condición de búsqueda

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 7
ingrese el rango de edad (mínima y máxima separadas por coma): 40,60
Profesor{nombre='Ivan', edad=41, gender='male', email='Ivannr@gmail.com', phone='654335206', subjects=[FOL, SGE], title='DAM', rating=3.0}
Profesor{nombre='alba', edad=50, gender='female', email='alba@estu.com', phone='345678', subjects=[fol, eie], title='DAM', rating=10.0}
```

8- Actualizar profesor: se pedirá el email del profesor que se quiere actualizar y la nueva calificación que tendrña

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 8
ingrese el email del profesor a actualizar: alba@estu.com
ingrese la nueva calificación: 5
profesor actualizado correctamente.
```

```
Profesor{nombre='Ivan', edad=41, gender='male', email='Ivannr@gmail.com', phone='654335206', subjects=[FOL, SGE], title='DAM', rating=3.0}
Profesor{nombre='alba', edad=50, gender='female', email='alba@estu.com', phone='345678', subjects=[fol, eie], title='DAM', rating=5.0}
```

9- Dar de baja alumnos: se borrarán todos los alumnos cuya calificación sea 5 o superior

```
lumno{name='Vanessa Gibbs', age=33, gender='female', email='vanessagibbs@hivedom.com', phone='+1 (942) 591-3521', calification=0, higherGrade='DAM', rating=7.12}
lumno{name='Jaime Howard', age=34, gender='female', email='jaimehoward@hivedom.com', phone='+1 (894) 510-2219', calification=0, higherGrade='ASIR', rating=8.81}
lumno{name='Chase Conley', age=38, gender='male', email='chaseconley@hivedom.com', phone='+1 (937) 488-2414', calification=0, higherGrade='DAM', rating=5.56}
lumno{name='Moody Arnold', age=39, gender='male', email='moodyarnold@hivedom.com', phone='+1 (830) 420-2446', calification=0, higherGrade='DAM', rating=8.81}
lumno{name='Mcmillan Singleton', age=23, gender='male', email='mcmillansingleton@hivedom.com', phone='+1 (904) 563-3920', calification=0, higherGrade='DAW', rating
lumno{name='miguel', age=18, gender='male', email='miguel@noestu.com', phone='123456', calification=8, higherGrade='DAW', rating=9.0}
lumno{name='tomas', age=30, gender='male', email='tom@tom.com', phone='038775', calification=3, higherGrade='DAM', rating=3.0}
```

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 9
alumnos con calificación 5 o superior eliminados correctamente.
```

```
mostrando todos los alumnos...
Alumno{name='Vanessa Gibbs', age=33, gender='female', email='vanessagibbs@hivedom.com', phone='+1 (942) 591-3521', calification=0, higherGrade='DAM', rating=7.12}
Alumno{name='Jaime Howard', age=34, gender='female', email='jaimehoward@hivedom.com', phone='+1 (894) 510-2219', calification=0, higherGrade='ASIR', rating=8.81}
Alumno{name='Chase Conley', age=38, gender='male', email='chaseconley@hivedom.com', phone='+1 (937) 488-2414', calification=0, higherGrade='DAM', rating=5.56}
Alumno{name='Moody Arnold', age=39, gender='male', email='moodyarnold@hivedom.com', phone='+1 (830) 420-2446', calification=0, higherGrade='DAM', rating=8.81}
Alumno{name='Mcmillan Singleton', age=23, gender='male', email='mcmillansingleton@hivedom.com', phone='+1 (904) 563-3920', calification=0, higherGrade='DAW', rating=7.
Alumno{name='tomas', age=30, gender='male', email='tom@tom.com', phone='038775', calification=3, higherGrade='DAM', rating=3.0}
```

10- Salir

```
menú de opciones:
1- insertar un profesor
2- insertar un alumno
3- mostrar todos los datos
4- mostrar profesores
5- mostrar alumnos
6- buscar alumno por mail
7- buscar profesor por edad
8- actualizar calificación profesor
9- dar de baja alumnos con suspenso
10- salir
seleccione una opción: 10
saliendo...
```

Enlace a GitHub:

https://github.com/Ivannunezrodriguez/Acceso_a_Datos_24-25/tree/6f913baaacce7ae2abd4a606ba6ac49f186f53b8/Acitvidades/tema_1/Ejercicio%20Mongo%20Java