**Ejercicio Mongo Java**

**Ivan Nuñez Rodriguez**

2º DAM

Acceso a Datos

**Índice:**

1. Código.
2. Creación Base de Datos
3. Ejecución y resultado.
4. Enlace a GitHub.

**Código:**

**Clase AlumnoDAO**

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**

**Texto

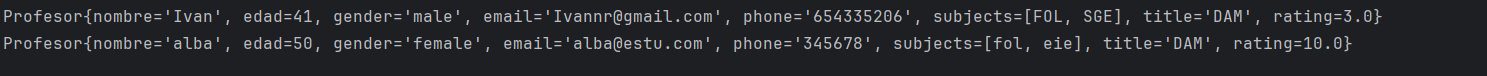
Descripción generada automáticamente**

1. **Insertar un profesor: tras seleccionar esta opción se pedirá al usuario la**

**introducción de todos los datos**

**Texto

Descripción generada automáticamente**

****

1. **Insertar un alumno: tras seleccionar esta opción se pedirá al usuario la**

**introducción de todos los datos**

**Texto

Descripción generada automáticamente**

****

1. **Mostrar todos los datos: se imprimirán todos los datos, tanto de usuarios**

**como de profesores**

**Imagen que contiene Texto

Descripción generada automáticamente**

1. **Mostrar profesores: se imprimirá solo los datos de los profesores**

**Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente**

1. **Mostrar alumnos: se imprimirá solo los datos de los alumnos**

**Imagen que contiene computadora, tabla, monitor, grande

Descripción generada automáticamente**

1. **Buscar alumno: se pedirá un email y se mostrará el alumno que cumpla la**

**condición de búsqueda**

**Texto

Descripción generada automáticamente**

1. **Buscar profesor: se pedirá un rando de edad y se mostrará solo aquellos**

**que cumpla la condición de búsqueda**

**Imagen que contiene Texto

Descripción generada automáticamente**

1. **Actualizar profesor: se pedirá el email del profesor que se quiere actualizar**

**y la nueva calificación que tendrña**

**Texto

Descripción generada automáticamente**

****

1. **Dar de baja alumnos: se borrarán todos los alumnos cuya calificación sea 5**

**o superior**

****

**Texto

Descripción generada automáticamente**

**Teclado de computadora

Descripción generada automáticamente con confianza media**

1. **Salir**

**Texto

Descripción generada automáticamente**

**Enlace a GitHub:**

https://github.com/Ivannunezrodriguez/Acceso\_a\_Datos\_24-25/tree/6f913baaacce7ae2abd4a606ba6ac49f186f53b8/Acitvidades/tema\_1/Ejercicio%20Mongo%20Java