Venta de entradas. Parte 2.

Creamos la colección usuarios

Comenzamos creando colección usuarios con 2 usuarios dentro, la cual nos servirán para iniciar sesión en nuestro programa:

```
use entradas_db;
db.createCollection("usuarios");
db.usuarios.insertMany([
{
	name: "oscar",
	password: crypto.createHash('sha256').update("1234").digest('hex')
},
{
	name: "adolfo",
	password: crypto.createHash('sha256').update("4321").digest('hex')
}]);
```

Los usuarios los creamos encriptando sus contraseñas en sha256.

Ahora pasamos al código del programa java:

```
package entradas;

import java.security.MessageDigest;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.MongoCursor;
import org.bson.Document;

public class ProgramaVentas {

	private static final String MONGO_URI =
"mongodb://entradas:1234@79.72.63.217:27017/entradas_db?authSource=entradas_db";
	private static final String DATABASE_NAME = "entradas_db";

	public static void main(String[] args) {
		// Deshabilitar los logs de MongoDB
		Logger mongoLogger = Logger.getLogger("org.mongodb.driver");
		mongoLogger.setLevel(Level.WARNING);

		try (MongoClient mongoClient = MongoClients.create(MONGO_URI)) {
			MongoDatabase database = mongoClient.getDatabase(DATABASE_NAME);
			Scanner scanner = new Scanner(System.in);

			System.out.println("=== Inicio de Sesión ===");
			System.out.print("Ingrese su nombre de usuario: ");
			String nombreUsuario = scanner.nextLine();
```

```
                        System.out.print("Ingrese su contraseña: ");
                        String contraseña = scanner.nextLine();

                        // Validamos el usuario
                        if (!validarUsuario(database, nombreUsuario, contraseña)) {
                                System.out.println("Usuario o contraseña incorrectos. Cerrando
programa.");
                                return;
                        }
                        System.out.println("Inicio de sesión exitoso.");

                        // Bucle de operaciones
                        while (true) {
                                System.out.println("\n=== Programa de Ventas ===");
                                System.out.println("1. Ver entradas disponibles y comprar");
                                System.out.println("2. Salir");
                                System.out.print("Seleccione una opción: ");
                                int opcion = scanner.nextInt();
                                scanner.nextLine();
                                if (opcion == 1) {
                                        mostrarColecciones(database);
                                        System.out.print("Seleccione el evento (nombre de la
colección): ");
                                        String nombreEvento = scanner.nextLine();

                                        System.out.print("¿Cuántas entradas desea comprar? ");
                                        int cantidad = scanner.nextInt();
                                        scanner.nextLine();

                                        procesarCompra(database, nombreEvento, cantidad,
nombreUsuario);
                                } else if (opcion == 2) {
                                        System.out.println("Programa finalizado.");
                                        break;
                                } else {
                                        System.out.println("Opción no válida.");
                                }
                        }
                } catch (Exception e) {
                        System.err.println("Error en el programa: " + e.getMessage());
                        e.printStackTrace();
                }
        }

        private static boolean validarUsuario(MongoDatabase database, String nombreUsuario, String
contraseña) {
                MongoCollection<Document> usuariosCollection = database.getCollection("usuarios");
                Document usuario = usuariosCollection.find(new Document("name",
nombreUsuario)).first();

                if (usuario != null) {
                        String contraseñaCifrada = cifrarContraseña(contraseña);
                        return contraseñaCifrada.equals(usuario.getString("password"));
                }
                return false;
        }

        private static void mostrarColecciones(MongoDatabase database) {
                System.out.println("\n=== Entradas Disponibles ===");
                for (String coleccion : database.listCollectionNames()) {
                        if (!coleccion.equals("usuarios")) {
                                System.out.println("- " + coleccion);
                        }
                }
        }
```

```
	private static void procesarCompra(MongoDatabase database, String nombreEvento, int cantidad,
			String nombreUsuario) {
		MongoCollection<Document> collection = database.getCollection(nombreEvento);

		long entradasLibres = collection.countDocuments(new Document("nombreCliente", null));
		if (entradasLibres < cantidad) {
			System.out
					.println("No hay suficientes entradas disponibles. Quedan " +
entradasLibres + " entradas libres.");
			return;
		}

		MongoCursor<Document> cursor = collection.find(new Document("nombreCliente",
null)).limit(cantidad).iterator();
		int entradasAsignadas = 0;

		while (cursor.hasNext()) {
			Document entrada = cursor.next();
			collection.updateOne(new Document("_id", entrada.get("_id")),
					new Document("$set", new Document("nombreCliente",
nombreUsuario)));
			entradasAsignadas++;
		}

		System.out.println("Se han asignado " + entradasAsignadas + " entradas.");
		long entradasRestantes = collection.countDocuments(new Document("nombreCliente",
null));
		System.out.println("Entradas libres restantes: " + entradasRestantes);
	}

	private static String cifrarContraseña(String contraseña) {
		try {
			MessageDigest digest = MessageDigest.getInstance("SHA-256");
			byte[] encodedhash = digest.digest(contraseña.getBytes("UTF-8"));
			StringBuilder hexString = new StringBuilder();
			for (byte b : encodedhash) {
				String hex = Integer.toHexString(0xff & b);
				if (hex.length() == 1)
					hexString.append('0');
				hexString.append(hex);
			}
			return hexString.toString();
		} catch (Exception e) {
			throw new RuntimeException("Error al cifrar la contraseña", e);
		}
	}
}
```

Y en pom.xml me aseguro de configurar las dependencias:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>1</groupId>
	<artifactId>AD404_VentaDeEntradas11</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<dependencies>
		<dependency>
			<groupId>org.mongodb</groupId>
			<artifactId>mongo-java-driver</artifactId>
			<version>3.12.14</version>
		</dependency>
	</dependencies>

</project>
```

Y al ejecutarlo, veremos que podemos iniciar sesión correctamente y coger las entradas correspondientes

```
Problems  Servers  Terminal  Data Source Explorer  Properties  Console
ProgramaVentas [Java Application] C:\Program Files\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.4
=== Inicio de Sesión ===
Ingrese su nombre de usuario: oscar
Ingrese su contraseña: 1234
Inicio de sesión exitoso.

=== Programa de Ventas ===
1. Ver entradas disponibles y comprar
2. Salir
Seleccione una opción: 1

=== Entradas Disponibles ===
- monologo
Seleccione el evento (nombre de la colección): monologo
¿Cuántas entradas desea comprar? 2
Se han asignado 2 entradas.
Entradas libres restantes: 3

=== Programa de Ventas ===
1. Ver entradas disponibles y comprar
2. Salir
Seleccione una opción: 1

=== Entradas Disponibles ===
- monologo
Seleccione el evento (nombre de la colección): monologo
¿Cuántas entradas desea comprar? 4
No hay suficientes entradas disponibles. Quedan 3 entradas libres.

=== Programa de Ventas ===
1. Ver entradas disponibles y comprar
2. Salir
Seleccione una opción:
```

```
entradas_db> db.monologo.find()
[
  {
    _id: ObjectId('67992be3d506da63cb03a7e5'),
    numeroEntrada: 1,
    observaciones: '',
    nombreCliente: null
  },
  {
    _id: ObjectId('67992be3d506da63cb03a7e6'),
    numeroEntrada: 2,
    observaciones: '',
    nombreCliente: null
  },
  {
    _id: ObjectId('67992be3d506da63cb03a7e7'),
    numeroEntrada: 3,
    observaciones: '',
    nombreCliente: null
  },
  {
    _id: ObjectId('67992be3d506da63cb03a7e8'),
    numeroEntrada: 4,
    observaciones: '',
    nombreCliente: null
  },
  {
    _id: ObjectId('67992be3d506da63cb03a7e9'),
    numeroEntrada: 5,
    observaciones: '',
    nombreCliente: null
  }
]
entradas_db>
```

```
entradas_db> db.monologo.find()
[
  {
    _id: ObjectId('67992cf8e192b35c5ab51705'),
    numeroEntrada: 1,
    observaciones: '',
    nombreCliente: 'oscar'
  },
  {
    _id: ObjectId('67992cf8e192b35c5ab51706'),
    numeroEntrada: 2,
    observaciones: '',
    nombreCliente: 'oscar'
  },
  {
    _id: ObjectId('67992cf8e192b35c5ab51707'),
    numeroEntrada: 3,
    observaciones: '',
    nombreCliente: null
  },
  {
    _id: ObjectId('67992cf8e192b35c5ab51708'),
    numeroEntrada: 4,
    observaciones: '',
    nombreCliente: null
  },
  {
    _id: ObjectId('67992cf8e192b35c5ab51709'),
    numeroEntrada: 5,
    observaciones: '',
    nombreCliente: null
  }
]
entradas_db>
```