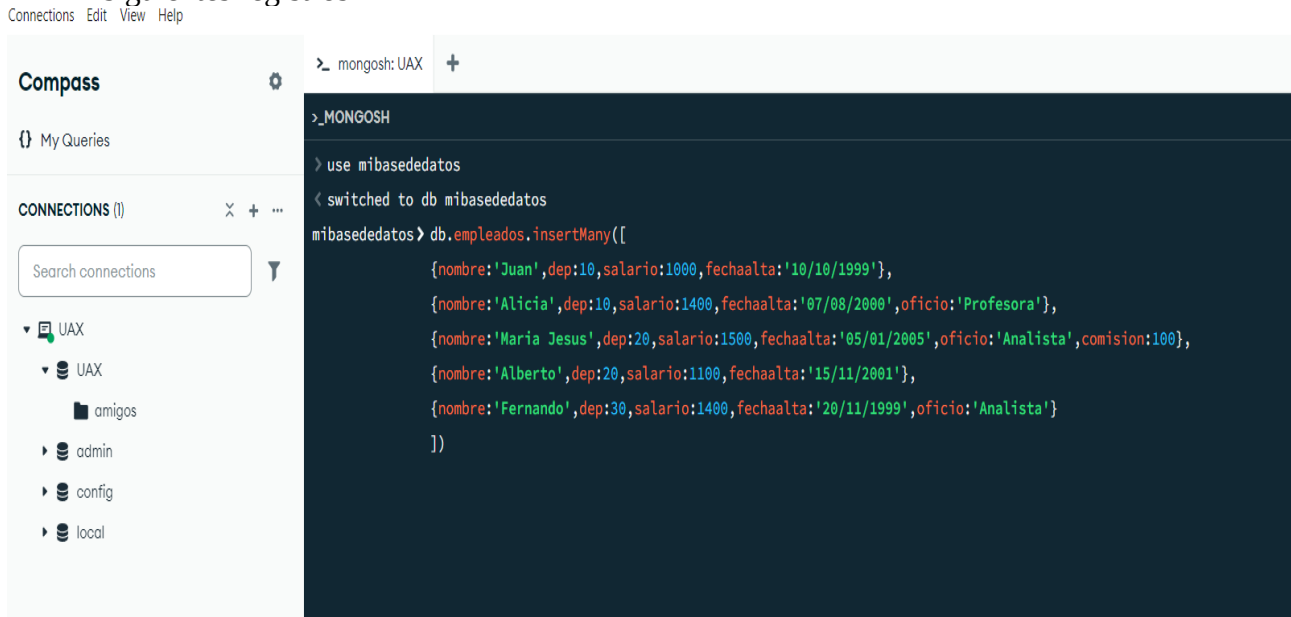


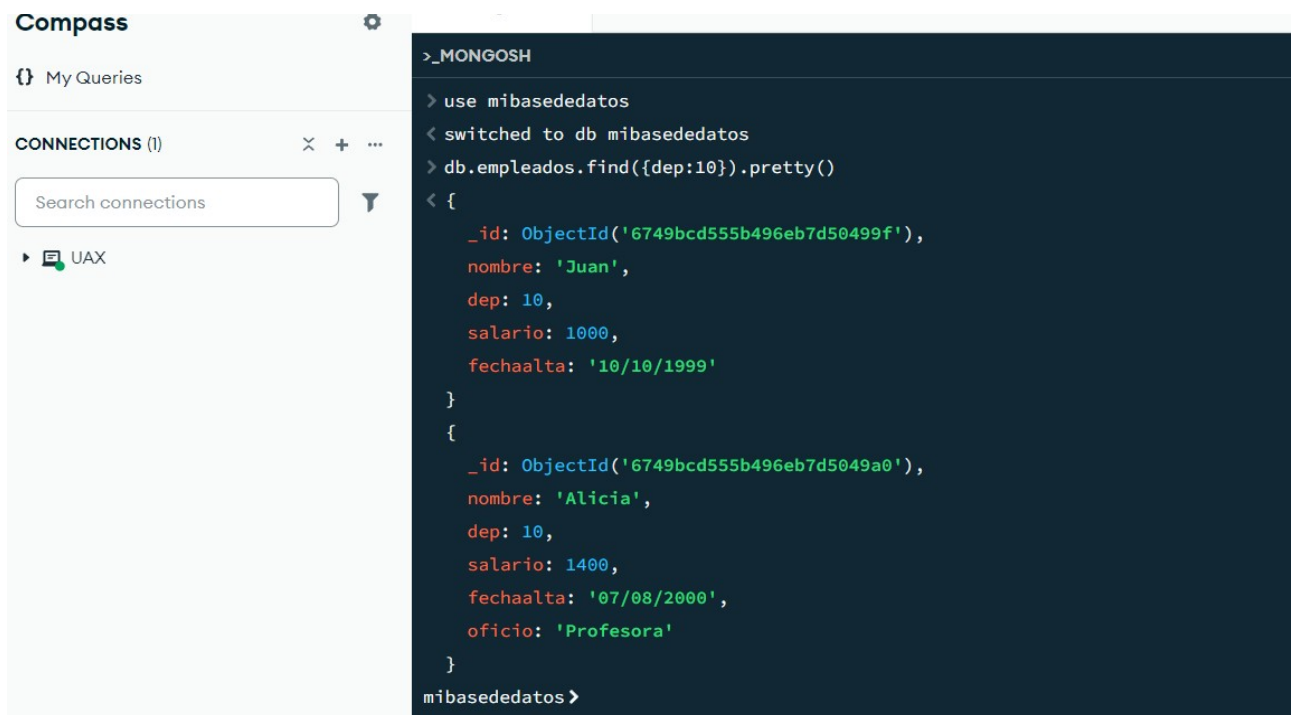
RETO5 – Por Gerard Luque Oliver

- Crea una colección empleados dentro de la base de datos mibasededatos y añade los siguientes registros

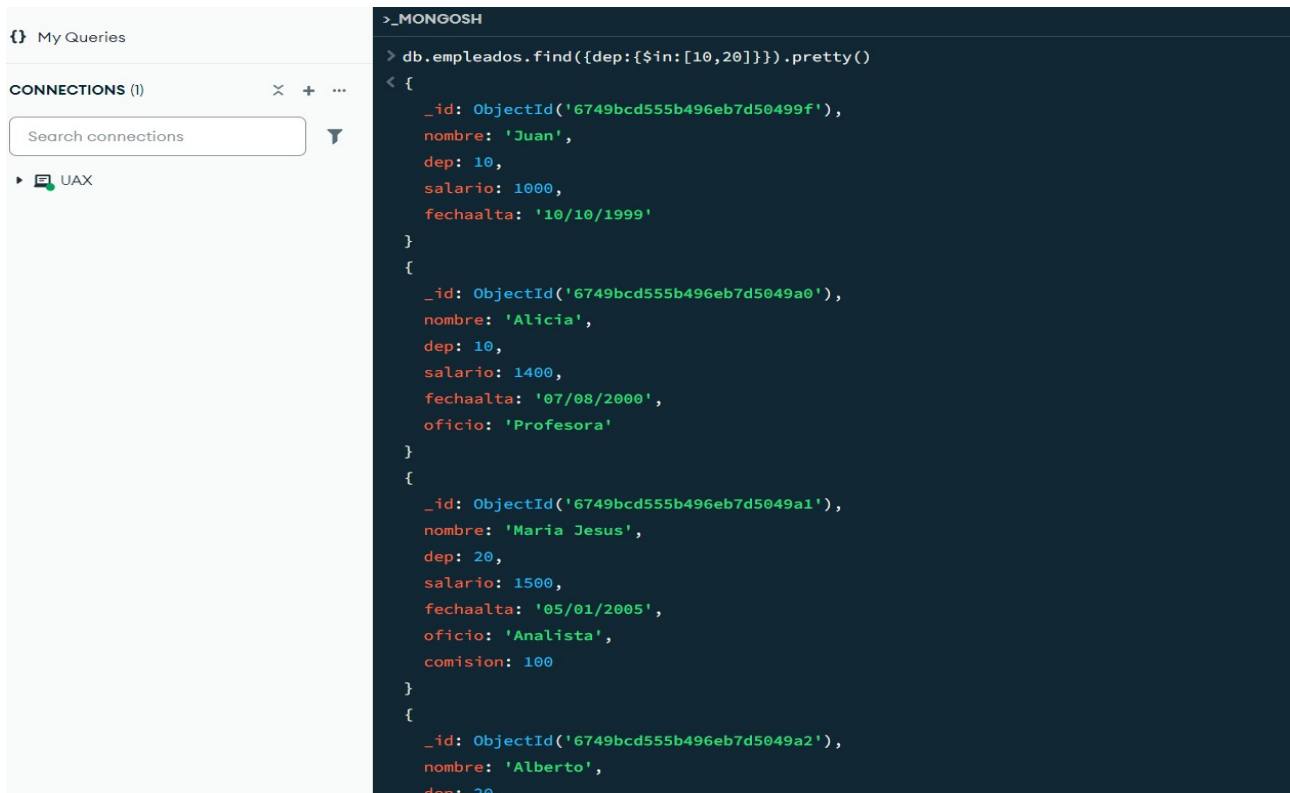


Realiza las siguientes consultas:

- Visualiza los empleados del departamento 10



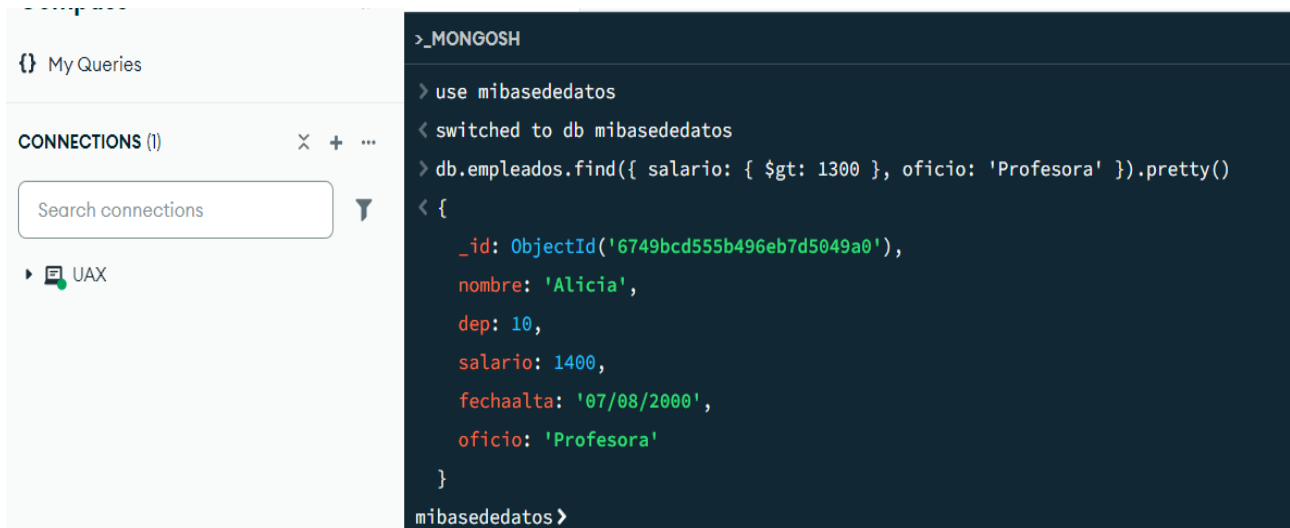
- Visualiza los empleados del departamento 10 y 20



The screenshot shows the MongoDB Compass interface. On the left, the 'My Queries' panel is visible with a 'CONNECTIONS (1)' section showing a connection to 'UAX'. The main panel displays a MongoDB query in the '_MONGOSH' terminal:

```
>_MONGOSH
> db.empleados.find({dep:{$in:[10,20]}}).pretty()
< {
  _id: ObjectId('6749bcd555b496eb7d50499f'),
  nombre: 'Juan',
  dep: 10,
  salario: 1000,
  fechaalta: '10/10/1999'
}
{
  _id: ObjectId('6749bcd555b496eb7d5049a0'),
  nombre: 'Alicia',
  dep: 10,
  salario: 1400,
  fechaalta: '07/08/2000',
  oficio: 'Profesora'
}
{
  _id: ObjectId('6749bcd555b496eb7d5049a1'),
  nombre: 'Maria Jesus',
  dep: 20,
  salario: 1500,
  fechaalta: '05/01/2005',
  oficio: 'Analista',
  comision: 100
}
{
  _id: ObjectId('6749bcd555b496eb7d5049a2'),
  nombre: 'Alberto',
  dep: 20,
```

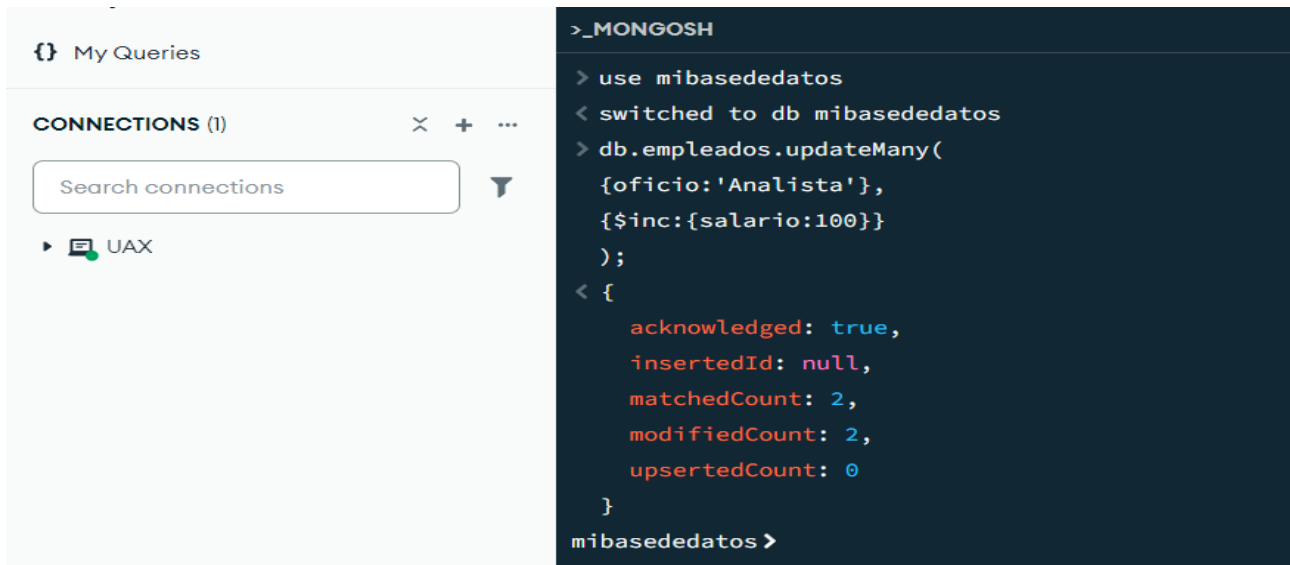
- Obtén los empleados con salario >1300 y oficio Profesora



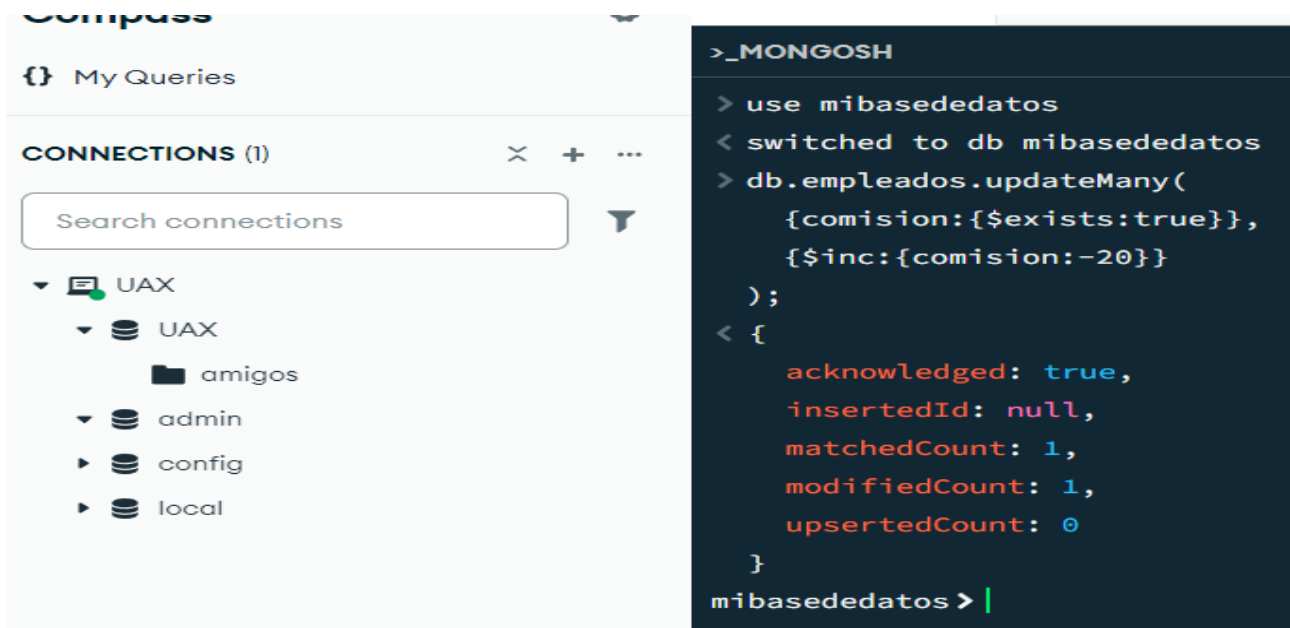
The screenshot shows the MongoDB Compass interface. On the left, the 'My Queries' panel is visible with a 'CONNECTIONS (1)' section showing a connection to 'UAX'. The main panel displays a MongoDB query in the '_MONGOSH' terminal:

```
>_MONGOSH
> use mibasededatos
< switched to db mibasededatos
> db.empleados.find({ salario: { $gt: 1300 }, oficio: 'Profesora' }).pretty()
< {
  _id: ObjectId('6749bcd555b496eb7d5049a0'),
  nombre: 'Alicia',
  dep: 10,
  salario: 1400,
  fechaalta: '07/08/2000',
  oficio: 'Profesora'
}
mibasededatos>
```

- Sube el salario de los analistas en 100€, a todos los analistas



- Decrementa la comisión en 20€, sólo a los que tengan comisión



MongoDB en Java, Conexión a la base de datos

Para realizar la conexión a la base de datos, tras incluir las dependencias correspondientes en mi fichero pom.xml:

```
<dependencies>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.10.0</version>
  </dependency>
</dependencies>
```

He creado dos clases, una que he llamado MongoDBConnection y otra Principal:

- **MongoDBConnection:**
 - Establece la conexión a MongoDB: Utiliza el cliente de MongoDC (MongoClients.create(uri)) para conectarse a una instancia de MongoDB mediante un URI proporcionado con parámetros.
Selecciona la base de datos deseada con mongoClient.getDatabase(dbNombre).
 - Manejo de errores: Si hay algún problema al intentar establecer la conexión, captura la excepción con MongoException y muestra un mensaje de error por consola.
 - Dar acceso a la base de datos: Con el método público getDatabase() permite obtener el objeto MongoDB, que puede ser usado para interactuar con las colecciones y los documentos de la base de datos.
 - Cerrar la conexión: Con el método close(), cerrando la conexión de manera segura.

```
package MongoDBConnect;
import com.mongodb.MongoException;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoDatabase;
/**
 * Clase para gestionar la conexión a una base de datos MongoDB.
 */
public class MongoDBConnection {
    // Cliente para interactuar con MongoDB
    private MongoClient mongoClient;
    // Objeto que representa la base de datos seleccionada
    private MongoDatabase database;
    /**
     * Constructor que establece una conexión a MongoDB.
     * @param uri URI de conexión a MongoDB.
     * @param dbName Nombre de la base de datos a conectar.
     */
    public MongoDBConnection(String uri, String dbName) {
        try {
            // Crea un cliente MongoDB usando la URI proporcionada
            mongoClient = MongoClients.create(uri);
            // Obtiene la base de datos especificada por su nombre
            database = mongoClient.getDatabase(dbName);
        } catch (MongoException e) {
            // Manejo de errores: muestra un mensaje si ocurre una excepción al conectar
            System.err.println("Error al conectar con MongoDB: " + e.getMessage());
            e.printStackTrace();
        }
    }
    /**
     * Método para obtener la base de datos conectada.
     * @return Un objeto MongoDatabase que representa la base de datos.
     */
    public MongoDatabase getDatabase() {
        return database;
    }
    /**
     * Método para cerrar la conexión con MongoDB.
     * Libera los recursos del cliente si está abierto.
     */
    public void close() {
        if (mongoClient != null) { // Verifica si el cliente está inicializado
            mongoClient.close(); // Cierra la conexión
        }
    }
}
```

- Principal: En esta clase es donde trabajo con la base de datos. Lo primero que hago es
 - Crear dos variables para tener los datos de conexión a la base de datos.

```
String uri = "mongodb://Gery:63ry@localhost:27017/mibasededatos";
String dbName = "mibasededatos";
```

- Crear la instancia de conexión con MongoDB

```
MongoDBConnection dbConect = new MongoDBConnection(uri, dbName);
```

- Obtener la base de datos.

```
MongoDatabase db = dbConect.getDatabase();
System.out.println("Conectado a la BBDD: "+db.getName());
```

- Los datos de una colección se pueden cargar en una lista utilizando el método find().into():

```
MongoCollection<Document> coleccion = db.getCollection("empleados");
List<Document> consulta = coleccion.find().into(new ArrayList<Document>());
System.out.println("Empleados:");
for (int i = 0; i < consulta.size(); i++) {
    Document empleado = consulta.get(i);
    System.out.println("\tNOMBRE: "+empleado.getString("nombre")+ ", DEPARTAMENTO: "+
        empleado.getInteger("dep")+ ", SALARIO: "+empleado.getInteger("salario")+
        ", FECHA DE ALTA: "+empleado.getString("fechaalta")+
        ", OFICIO: "+empleado.getString("oficio")+ ", COMISION: "+empleado.getInteger("comision"));
}
```

- Insertar un empleado con el método insertOne().
Primero creo un nuevo Documento empleado y en él mediante el método put asigno los parámetros clave-valor necesarios. Finalmente introduzco el documento en la colección empleados:

```
//Insertar un empleado con insertOne()
Document empleado = new Document();
empleado.put("nombre", "Joselito");
empleado.put("dep", 20);
empleado.put("salario", 3600);
empleado.put("fechaalta", "04/03/2002");
empleado.put("oficio", "Analista");
empleado.put("comision", 100);
coleccion.insertOne(empleado);
```

- Insertar documentos utilizando el método append de Document.

```
Document empleado2 = new Document("nombre", "Martina")
    .append("dep", 20)
    .append("salario", 40000)
    .append("fechaalta", "04/03/2002")
    .append("oficio", "Programador")
    .append("comision", 300);
coleccion.insertOne(empleado2);
```

- Utilizar el método iterator() para recorrer el cursor.

```
MongoCursor<Document> cursor = coleccion.find().iterator();
while (cursor.hasNext()) {
    Document empleadoCursor = cursor.next();
    System.out.println(empleadoCursor.toJson());
}
cursor.close();
```