

## ACCESO A DATOS

### PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB

1. Crea una base de datos en MongoDB con tu nombre y crea una colección de restaurantes, almacenando la siguiente información sobre cada uno: nombre, propietario, direccion que será un objeto que contendrá calle, cp, localidad y provincia, especialidad, teléfonos que será un array para contener varios teléfonos, y menú que será un array de objetos que contendrá comida, precio.

```
mongo - Acceso directo
---
---
    Enable MongoDB's free cloud-based monitoring service, which will then receive
and display
    metrics about your deployment (disk utilization, CPU, operation statistics, e
tc).

    The monitoring data will be available on a MongoDB website with a unique URL
accessible to you
    and anyone you share the URL with. MongoDB may use this information to make p
roduct
    improvements and to suggest MongoDB products and deployment options to you.

    To enable free monitoring, run the following command: db.enableFreeMonitoring
()
    To permanently disable this reminder, run the following command: db.disableFr
eeMonitoring()
---
> show dbs
admin                0.000GB
config               0.000GB
datospersonas        0.000GB
local                0.000GB
miBaseMongoAdrian    0.000GB
miBaseMongoAdrianBoado 0.000GB
> use AlbertoColmenarBD
switched to db AlbertoColmenarBD
> db.createCollection('restaurantes')
{ "ok" : 1 }
>
```

2. Inserta un restaurante rellenando todos los campos.

```
> db.restaurantes.insert({nombre: 'Ginos', propietario: 'Gianluca Virgolini', direccion: {calle: 'Calle Falsa 123', cp: 37492, localidad: 'Roma', provincia: 'Toscana'}, especialidad: 'pasta', telefonos:[5739573, 2384392], menu: ['Lasagna 10€', 'Tortellini 9€', 'Pizza napolitana 12€']})
WriteResult({ "nInserted" : 1 })
>
```

3. Inserta un restaurante sin indicar el menú.

```
> db.restaurantes.insert({nombre: 'Brasa Argentina', propietario: 'Mateo Rodriguez', direccion: {calle: 'Calle de la Plata', cp: 12392, localidad: 'Buenos Aires', provincia: 'Buenos Aires'}, especialidad: 'asados', telefonos:[54239573, 5742392]})
WriteResult({ "nInserted" : 1 })
>
```

4. Inserta un restaurante que indique en vez del atributo menú, menuDelDia que será un objeto que contendrá: primerPlato, segundoPlato, postre, precio.

## ACCESO A DATOS

### PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB

```
> db.restaurantes.insert({nombre: 'Elite Burger', propietario: 'Roberto Zacarias', direccion: {calle: 'Calle Escandinavia', cp: 43392, localidad: 'Fuenlabrada', provincia: 'Madrid'}, especialidad: 'hamburguesas', telefonos: [5435234, 1742392], menuDelDia: {primerPlato: 'aros de cebolla', segundoPlato: 'hamburguesa', postre: 'gofre', precio: '17€'}})
WriteResult({ "nInserted" : 1 })
>
```

5. Realiza una búsqueda de todos los restaurantes de la misma especialidad y ordénalos por nombre.

```
> db.restaurantes.find({especialidad: 'pasta'}).sort({nombre:1})
{ "_id" : ObjectId("65e5a16ffd8d91f48212c218"), "nombre" : "Ginos", "propietario" : "Gianluca Virgolini", "direccion" : { "calle" : "Calle Falsa 123", "cp" : 37492, "localidad" : "Roma", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 5739573, 2384392 ], "menu" : [ "Lasagna 10€", "Tortellini 9€", "Pizza napolitana 12€" ] }
{ "_id" : ObjectId("65e5b1c6fd8d91f48212c21c"), "nombre" : "Grecos", "propietario" : "Leonardo Greco", "direccion" : { "calle" : "Calle Mochila 134", "cp" : 67492, "localidad" : "Milán", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 95739573, 2584392 ], "menu" : [ "pescado", "pasta", "pollo" ] }
{ "_id" : ObjectId("65e5b11cfd8d91f48212c21b"), "nombre" : "Pizzeria", "propietario" : "María Gómez", "direccion" : { "calle" : "Calle Broma 34", "cp" : 67492, "localidad" : "Roma", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 75739573, 584392 ], "menu" : [ "pescado", "pasta", "pollo" ] }
>
```

6. Realiza una búsqueda de todos los restaurantes de la misma localidad y ordénalos por propietario.

```
> db.restaurantes.find({'direccion.localidad': 'Roma'}).sort({propietario:1})
{ "_id" : ObjectId("65e5a16ffd8d91f48212c218"), "nombre" : "Ginos", "propietario" : "Gianluca Virgolini", "direccion" : { "calle" : "Calle Falsa 123", "cp" : 37492, "localidad" : "Roma", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 5739573, 2384392 ], "menu" : [ "Lasagna 10€", "Tortellini 9€", "Pizza napolitana 12€" ] }
{ "_id" : ObjectId("65e5b11cfd8d91f48212c21b"), "nombre" : "Pizzeria", "propietario" : "María Gómez", "direccion" : { "calle" : "Calle Broma 34", "cp" : 67492, "localidad" : "Roma", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 75739573, 584392 ], "menu" : [ "pescado", "pasta", "pollo" ] }
>
```

7. Realiza una búsqueda de todos los restaurantes que sirvan pollo.

```
> db.restaurantes.find({menu: 'pollo'})
{ "_id" : ObjectId("65e5b11cfd8d91f48212c21b"), "nombre" : "Pizzeria", "propietario" : "María Gómez", "direccion" : { "calle" : "Calle Broma 34", "cp" : 67492, "localidad" : "Roma", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 75739573, 584392 ], "menu" : [ "pescado", "pasta", "pollo" ] }
{ "_id" : ObjectId("65e5b1c6fd8d91f48212c21c"), "nombre" : "Grecos", "propietario" : "Leonardo Greco", "direccion" : { "calle" : "Calle Mochila 134", "cp" : 67492, "localidad" : "Milán", "provincia" : "Toscana" }, "especialidad" : "pasta", "telefonos" : [ 95739573, 2584392 ], "menu" : [ "pescado", "pasta", "pollo" ] }
>
```

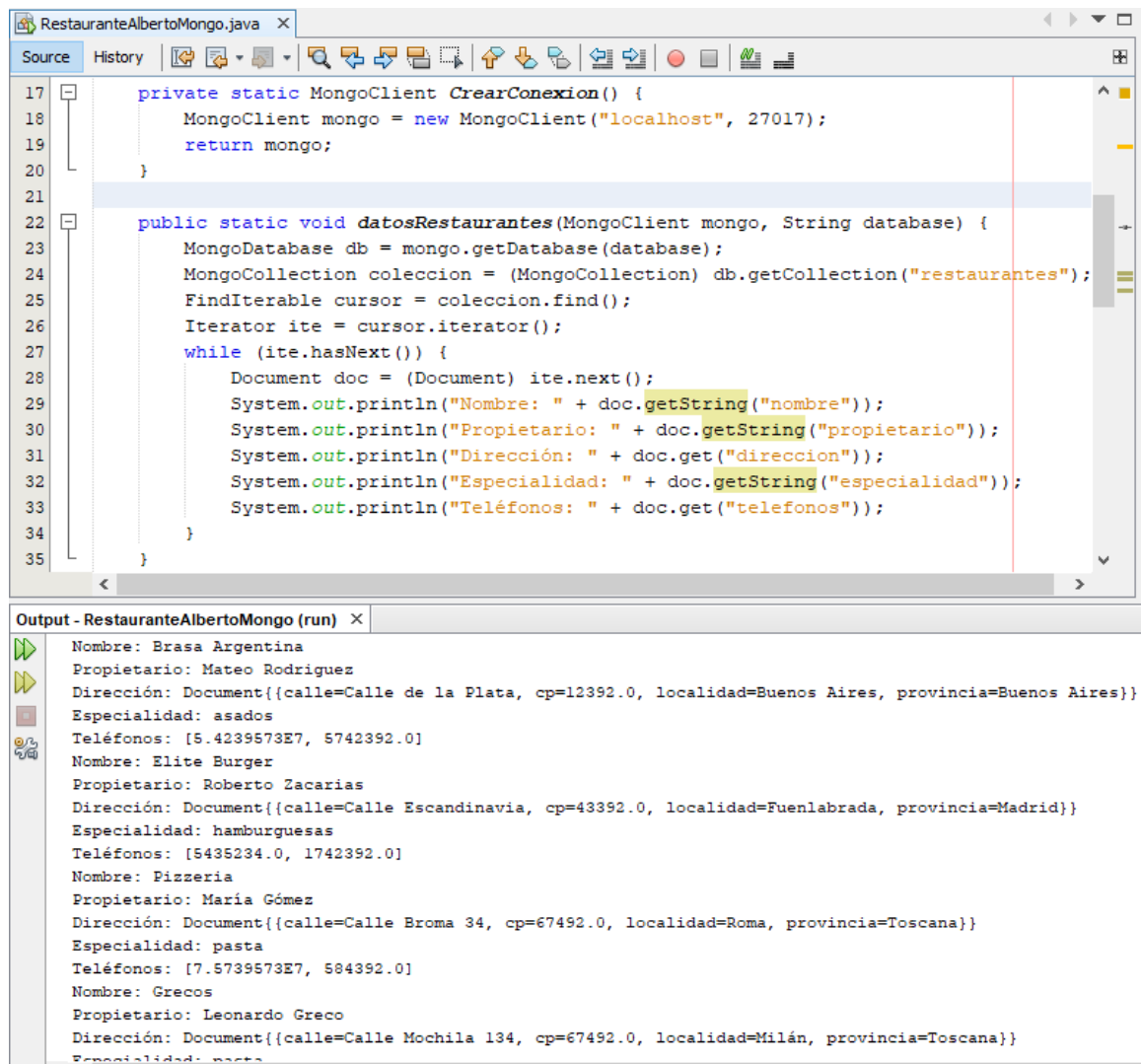
8. Modifica la especialidad de un restaurante determinado.

```
> db.restaurantes.update({nombre: 'Ginos'}, {$set: {especialidad: 'italiano'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
```

9. Crea una aplicación Java que permita conectar con la base de datos que has creado de forma que se pueda listar la información de los restaurantes introducidos.

## ACCESO A DATOS

### PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB



```
17 private static MongoClient CrearConexion() {
18     MongoClient mongo = new MongoClient("localhost", 27017);
19     return mongo;
20 }
21
22 public static void datosRestaurantes(MongoClient mongo, String database) {
23     MongoDBDatabase db = mongo.getDatabase(database);
24     MongoCollection coleccion = (MongoCollection) db.getCollection("restaurantes");
25     FindIterable cursor = coleccion.find();
26     Iterator ite = cursor.iterator();
27     while (ite.hasNext()) {
28         Document doc = (Document) ite.next();
29         System.out.println("Nombre: " + doc.getString("nombre"));
30         System.out.println("Propietario: " + doc.getString("propietario"));
31         System.out.println("Dirección: " + doc.get("direccion"));
32         System.out.println("Especialidad: " + doc.getString("especialidad"));
33         System.out.println("Teléfonos: " + doc.get("telefonos"));
34     }
35 }
```

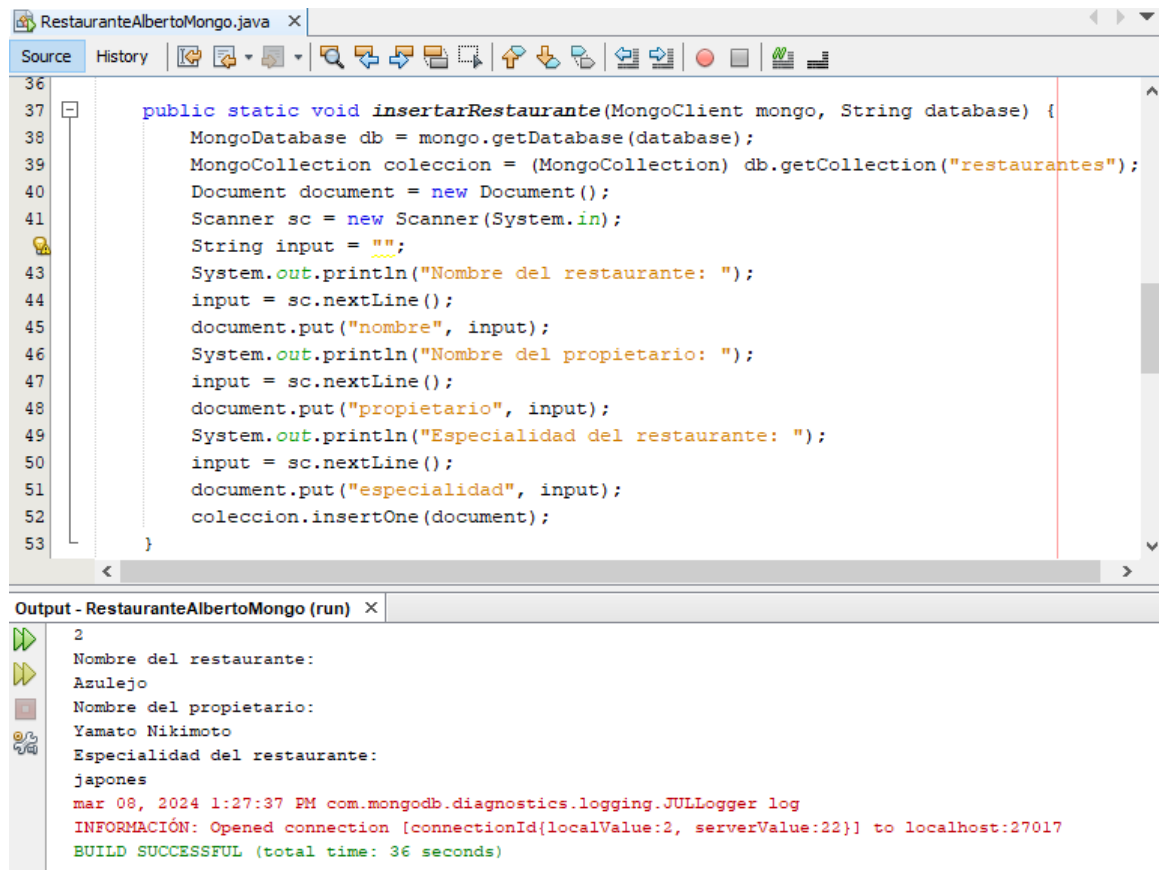
Output - RestauranteAlbertoMongo (run) X

```
Nombre: Brasa Argentina
Propietario: Mateo Rodriguez
Dirección: Document{{calle=Calle de la Plata, cp=12392.0, localidad=Buenos Aires, provincia=Buenos Aires}}
Especialidad: asados
Teléfonos: [5.4239573E7, 5742392.0]
Nombre: Elite Burger
Propietario: Roberto Zacarias
Dirección: Document{{calle=Calle Escandinavia, cp=43392.0, localidad=Fuenlabrada, provincia=Madrid}}
Especialidad: hamburguesas
Teléfonos: [5435234.0, 1742392.0]
Nombre: Pizzeria
Propietario: María Gómez
Dirección: Document{{calle=Calle Broma 34, cp=67492.0, localidad=Roma, provincia=Toscana}}
Especialidad: pasta
Teléfonos: [7.5739573E7, 584392.0]
Nombre: Grecos
Propietario: Leonardo Greco
Dirección: Document{{calle=Calle Mochila 134, cp=67492.0, localidad=Milán, provincia=Toscana}}
Especialidad: pasta
```

10. Añade a la aplicación la posibilidad de registrar nuevos restaurantes, considerando el nombre como campo obligatorio.

## ACCESO A DATOS

### PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB



The screenshot shows an IDE window titled 'RestauranteAlbertoMongo.java'. The code defines a static method 'insertarRestaurante' that interacts with MongoDB. It prompts the user for the restaurant name, owner name, and specialty, then inserts a document into the 'restaurantes' collection. Below the code, the 'Output' window shows the execution results, including the user input and system logs.

```
36
37 public static void insertarRestaurante(MongoClient mongo, String database) {
38     MongoDBDatabase db = mongo.getDatabase(database);
39     MongoCollection coleccion = (MongoCollection) db.getCollection("restaurantes");
40     Document document = new Document();
41     Scanner sc = new Scanner(System.in);
42     String input = "";
43     System.out.println("Nombre del restaurante: ");
44     input = sc.nextLine();
45     document.put("nombre", input);
46     System.out.println("Nombre del propietario: ");
47     input = sc.nextLine();
48     document.put("propietario", input);
49     System.out.println("Especialidad del restaurante: ");
50     input = sc.nextLine();
51     document.put("especialidad", input);
52     coleccion.insertOne(document);
53 }
```

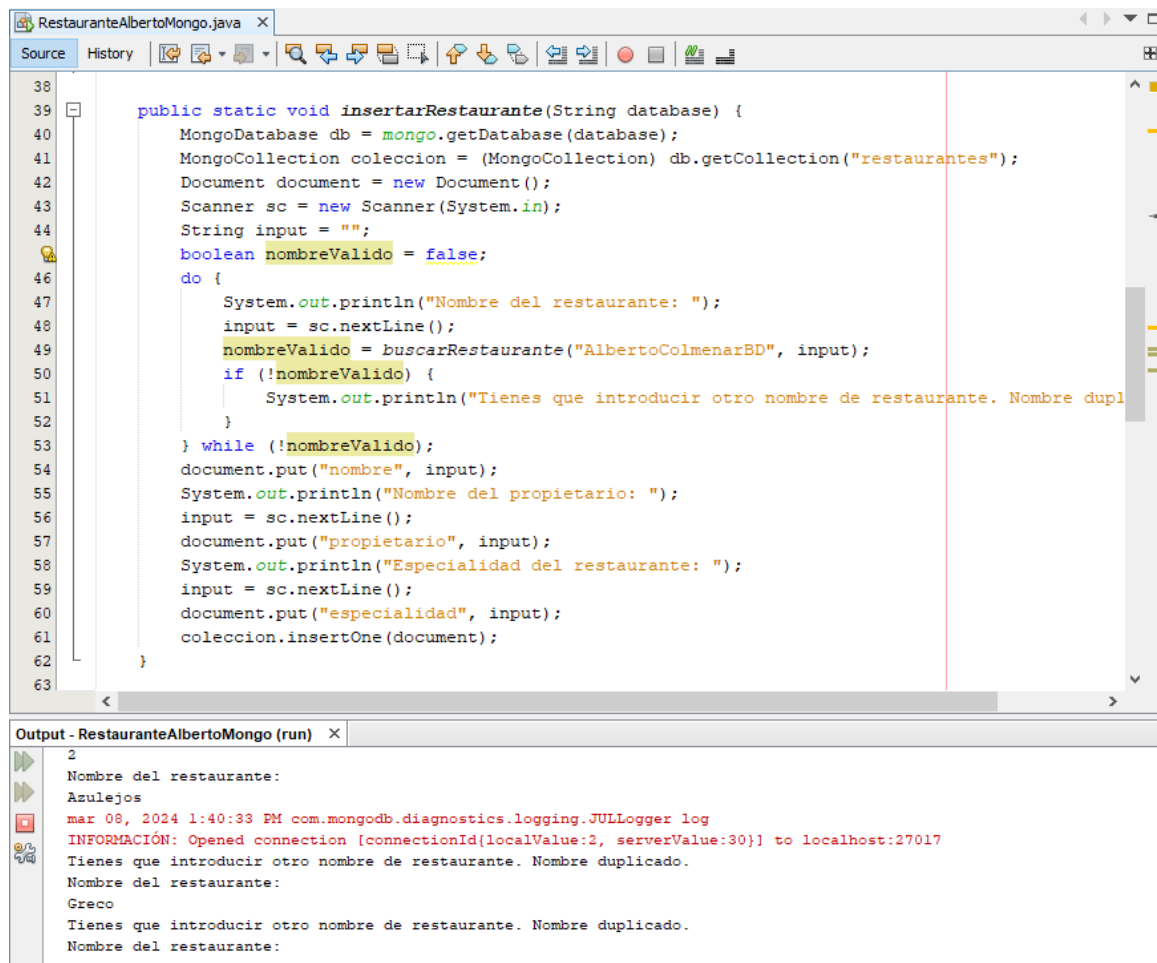
Output - RestauranteAlbertoMongo (run) ×

```
2
Nombre del restaurante:
Azulejo
Nombre del propietario:
Yamato Nikimoto
Especialidad del restaurante:
japones
mar 08, 2024 1:27:37 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:2, serverValue:22}] to localhost:27017
BUILD SUCCESSFUL (total time: 36 seconds)
```

11. Añade a la funcionalidad de registrar nuevos restaurantes un control para que no sea posible registrar dos restaurantes con el mismo nombre. En ese caso se mostrará un mensaje de error al usuario y tendrá que proporcionar otro nombre diferente.

## ACCESO A DATOS

### PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB

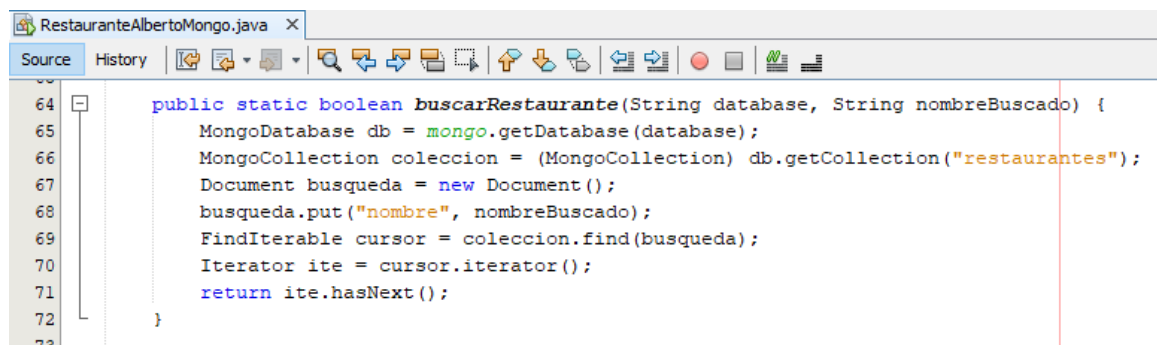


The screenshot shows an IDE window titled 'RestauranteAlbertoMongo.java'. The code defines a method `insertarRestaurante` that interacts with a MongoDB database. It prompts the user for a restaurant name, checks for duplicates, and then prompts for the owner's name and the restaurant's specialty. The output window shows the execution of this method, with the user entering 'Azulejos' and 'Greco', and the program outputting confirmation messages and a MongoDB connection log.

```
38
39 public static void insertarRestaurante(String database) {
40     MongoDB db = mongo.getDatabase(database);
41     MongoCollection coleccion = (MongoCollection) db.getCollection("restaurantes");
42     Document document = new Document();
43     Scanner sc = new Scanner(System.in);
44     String input = "";
45     boolean nombreValido = false;
46     do {
47         System.out.println("Nombre del restaurante: ");
48         input = sc.nextLine();
49         nombreValido = buscarRestaurante("AlbertoColmenarBD", input);
50         if (!nombreValido) {
51             System.out.println("Tienes que introducir otro nombre de restaurante. Nombre duplicado.");
52         }
53     } while (!nombreValido);
54     document.put("nombre", input);
55     System.out.println("Nombre del propietario: ");
56     input = sc.nextLine();
57     document.put("propietario", input);
58     System.out.println("Especialidad del restaurante: ");
59     input = sc.nextLine();
60     document.put("especialidad", input);
61     coleccion.insertOne(document);
62 }
63
```

Output - RestauranteAlbertoMongo (run) X

```
2
Nombre del restaurante:
Azulejos
mar 08, 2024 1:40:33 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:2, serverValue:30}] to localhost:27017
Tienes que introducir otro nombre de restaurante. Nombre duplicado.
Nombre del restaurante:
Greco
Tienes que introducir otro nombre de restaurante. Nombre duplicado.
Nombre del restaurante:
```



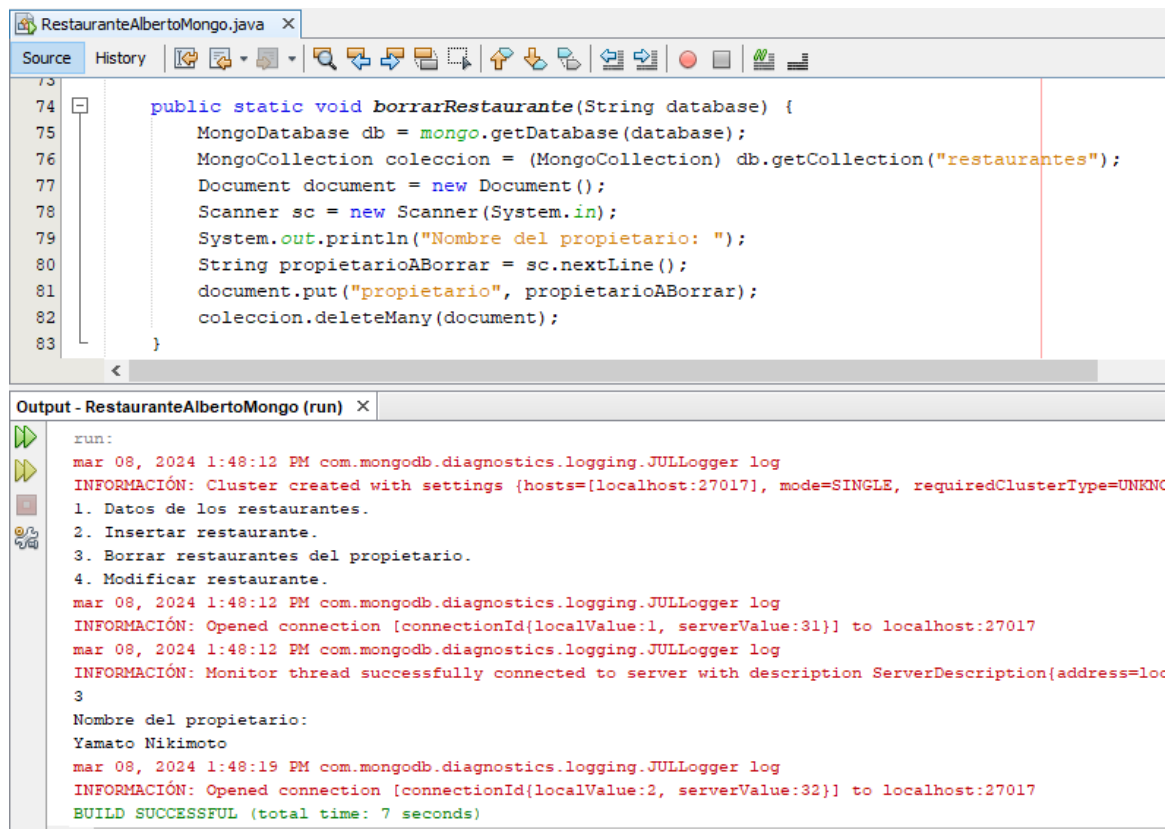
The screenshot shows the same IDE window, but now displaying the `buscarRestaurante` method. This method takes a database name and a restaurant name as input and returns a boolean indicating whether the restaurant exists in the database.

```
64 public static boolean buscarRestaurante(String database, String nombreBuscado) {
65     MongoDB db = mongo.getDatabase(database);
66     MongoCollection coleccion = (MongoCollection) db.getCollection("restaurantes");
67     Document busqueda = new Document();
68     busqueda.put("nombre", nombreBuscado);
69     FindIterable cursor = coleccion.find(busqueda);
70     Iterator ite = cursor.iterator();
71     return ite.hasNext();
72 }
73
```

12. Añade una nueva funcionalidad que permita eliminar todos los restaurantes de un mismo propietario.

## ACCESO A DATOS

### PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB



The screenshot shows an IDE window titled 'RestauranteAlbertoMongo.java'. The code defines a method `borrarRestaurante` that takes a database name as input. It connects to a MongoDB instance, retrieves the 'restaurantes' collection, prompts the user for a restaurant owner's name, and then deletes all documents belonging to that owner. The output window shows the execution of the program, including MongoDB connection logs and a menu of options. The user has selected option 3, 'Borrar restaurantes del propietario', and the program has successfully deleted the documents.

```
73
74 public static void borrarRestaurante(String database) {
75     MongoDBDatabase db = mongo.getDatabase(database);
76     MongoCollection coleccion = (MongoCollection) db.getCollection("restaurantes");
77     Document document = new Document();
78     Scanner sc = new Scanner(System.in);
79     System.out.println("Nombre del propietario: ");
80     String propietarioABorrar = sc.nextLine();
81     document.put("propietario", propietarioABorrar);
82     coleccion.deleteMany(document);
83 }
```

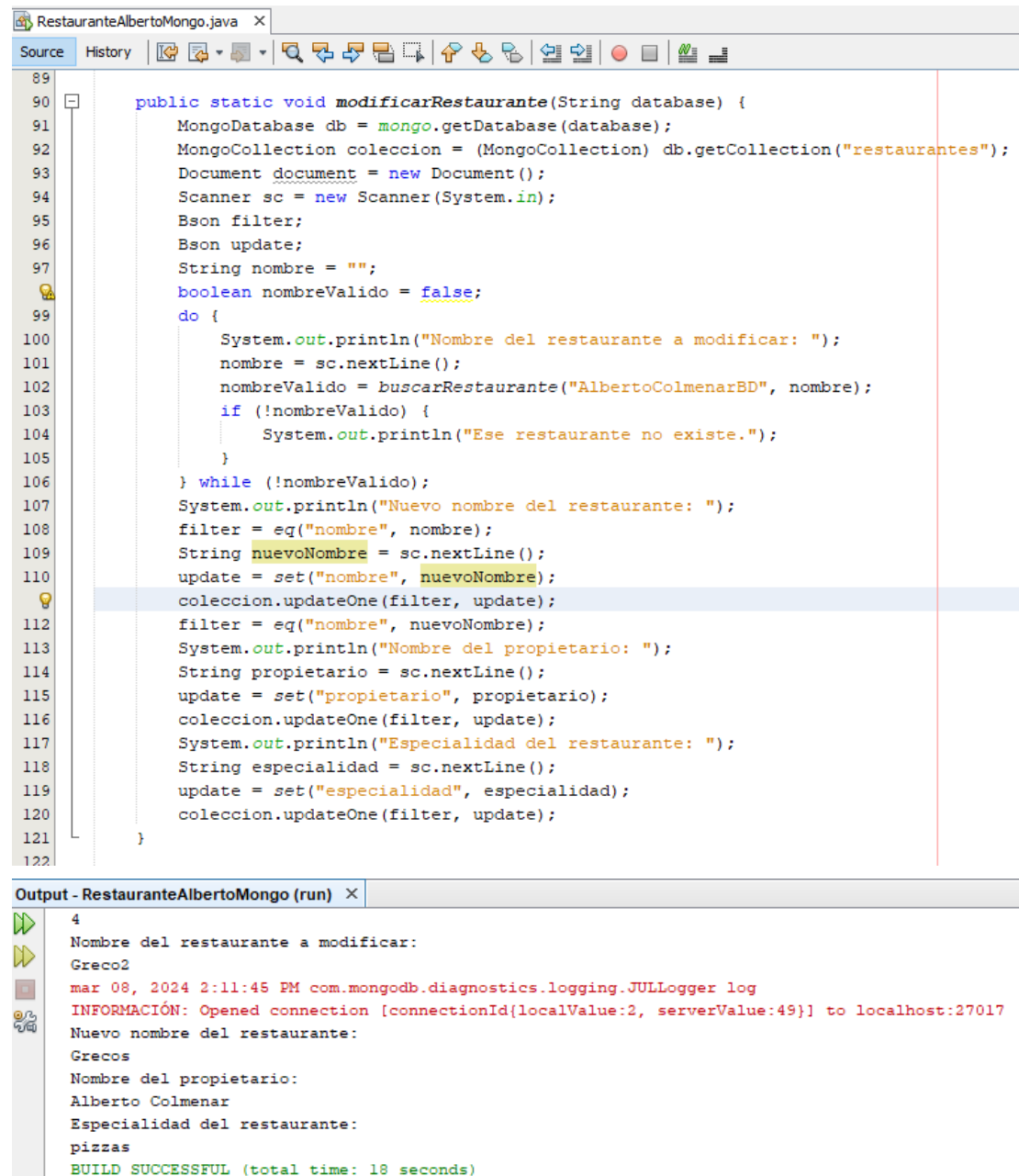
Output - RestauranteAlbertoMongo (run) X

```
run:
mar 08, 2024 1:48:12 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNK
1. Datos de los restaurantes.
2. Insertar restaurante.
3. Borrar restaurantes del propietario.
4. Modificar restaurante.
mar 08, 2024 1:48:12 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:1, serverValue:31}] to localhost:27017
mar 08, 2024 1:48:12 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Monitor thread successfully connected to server with description ServerDescription{address=loc
3
Nombre del propietario:
Yamato Nikimoto
mar 08, 2024 1:48:19 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:2, serverValue:32}] to localhost:27017
BUILD SUCCESSFUL (total time: 7 seconds)
```

13. Añade una funcionalidad que permita modificar los datos de un restaurante.

# ACCESO A DATOS

## PRÁCTICA 5.1 – BASES DE DATOS NO-SQL. MONGODB



The screenshot displays an IDE window titled "RestauranteAlbertoMongo.java". The code is a Java program that interacts with a MongoDB database to modify restaurant information. The code is as follows:

```
89
90 public static void modificarRestaurante(String database) {
91     MongoDB db = mongo.getDatabase(database);
92     MongoCollection coleccion = (MongoCollection) db.getCollection("restaurantes");
93     Document document = new Document();
94     Scanner sc = new Scanner(System.in);
95     Bson filter;
96     Bson update;
97     String nombre = "";
98     boolean nombreValido = false;
99     do {
100         System.out.println("Nombre del restaurante a modificar: ");
101         nombre = sc.nextLine();
102         nombreValido = buscarRestaurante("AlbertoColmenarBD", nombre);
103         if (!nombreValido) {
104             System.out.println("Ese restaurante no existe.");
105         }
106     } while (!nombreValido);
107     System.out.println("Nuevo nombre del restaurante: ");
108     filter = eq("nombre", nombre);
109     String nuevoNombre = sc.nextLine();
110     update = set("nombre", nuevoNombre);
111     coleccion.updateOne(filter, update);
112     filter = eq("nombre", nuevoNombre);
113     System.out.println("Nombre del propietario: ");
114     String propietario = sc.nextLine();
115     update = set("propietario", propietario);
116     coleccion.updateOne(filter, update);
117     System.out.println("Especialidad del restaurante: ");
118     String especialidad = sc.nextLine();
119     update = set("especialidad", especialidad);
120     coleccion.updateOne(filter, update);
121 }
122
```

The output window, titled "Output - RestauranteAlbertoMongo (run)", shows the execution of the program. It displays the prompts and user input for modifying a restaurant's name, owner, and specialty. The output is as follows:

```
4
Nombre del restaurante a modificar:
Greco2
mar 08, 2024 2:11:45 PM com.mongodb.diagnostics.logging.JULLogger log
INFORMACIÓN: Opened connection [connectionId{localValue:2, serverValue:49}] to localhost:27017
Nuevo nombre del restaurante:
Grecos
Nombre del propietario:
Alberto Colmenar
Especialidad del restaurante:
pizzas
BUILD SUCCESSFUL (total time: 18 seconds)
```