

Operaciones de Agregación

Trabajando la Shell

Comenzamos creando la base de datos, para ello entramos en la shell y lo creamos con los siguientes comandos:

```
mongosh -u admin1 -p admin1
use facturas_db
```

E insertamos 3 facturas iniciales de la siguiente forma:

```
db.facturas.insertMany([
  {
    numeroFactura: 1,
    fecha: new Date("2023-01-10"),
    cliente: "Cliente1",
    lineas: [
      { articulo: "ProductoA", precioUnitario: 10, cantidad: 2 },
      { articulo: "ProductoB", precioUnitario: 5, cantidad: 4 }
    ]
  },
  {
    numeroFactura: 2,
    fecha: new Date("2023-01-15"),
    cliente: "Cliente2",
    lineas: [
      { articulo: "ProductoC", precioUnitario: 20, cantidad: 3 },
      { articulo: "ProductoB", precioUnitario: 5, cantidad: 1 }
    ]
  },
  {
    numeroFactura: 3,
    fecha: new Date("2023-02-01"),
    cliente: "Cliente1",
    lineas: [
      { articulo: "ProductoA", precioUnitario: 10, cantidad: 5 },
      { articulo: "ProductoD", precioUnitario: 15, cantidad: 2 }
    ]
  }
]);
```

Añadimos 3 más:

```

db.facturas.insertMany([
  {
    numeroFactura: 4,
    fecha: new Date("2023-02-05"),
    cliente: "Cliente3",
    lineas: [
      { articulo: "ProductoA", precioUnitario: 10, cantidad: 1 },
      { articulo: "ProductoB", precioUnitario: 5, cantidad: 2 }
    ]
  },
  {
    numeroFactura: 5,
    fecha: new Date("2023-02-10"),
    cliente: "Cliente2",
    lineas: [
      { articulo: "ProductoC", precioUnitario: 20, cantidad: 1 },
      { articulo: "ProductoD", precioUnitario: 15, cantidad: 1 }
    ]
  },
  {
    numeroFactura: 6,
    fecha: new Date("2023-02-15"),
    cliente: "Cliente1",
    lineas: [
      { articulo: "ProductoB", precioUnitario: 5, cantidad: 10 },
      { articulo: "ProductoD", precioUnitario: 15, cantidad: 3 }
    ]
  }
]);

```

Y ahora creamos un usuario para acceder:

```

db.createUser({
  user: "facturas",
  pwd: "1234",
  roles:
  [
    {role: "readWrite", db: "facturas_db" },
    {role: "dbAdmin", db: "facturas_db" }
  ]
});

```

Operaciones de agregación

Cuántas facturas tiene un cliente, en este caso Cliente1:

```
db.facturas.aggregate([
  { $match: { cliente: "Cliente1" } },
  {
    $group: {
      _id: "$cliente",
      totalFacturas: { $sum: 1 }
    }
  }
]);
```

Importe total de una factura, en este caso factura 1:

```
db.facturas.aggregate([
  { $match: { numeroFactura: 1 } },
  { $unwind: "$lineas" },
  {
    $group: {
      _id: "$numeroFactura",
      totalFactura: {
        $sum: {
          $multiply: [ "$lineas.precioUnitario", "$lineas.cantidad" ]
        }
      }
    }
  }
]);
```

Total de ventas de un cliente, en este caso el Cliente2:

```
db.facturas.aggregate([
  { $match: { cliente: "Cliente2" } },
  { $unwind: "$lineas" },
  {
    $group: {
      _id: "$cliente",
      totalVentas: {
        $sum: {
          $multiply: [ "$lineas.precioUnitario", "$lineas.cantidad" ]
        }
      }
    }
  }
]);
```

Productos más vendidos:

```
db.facturas.aggregate([
  { $unwind: "$lineas" },
```

```

    {
        $group: {
            _id: "$lineas.articulo",
            unidadesVendidas: { $sum: "$lineas.cantidad" }
        }
    },
    { $sort: { unidadesVendidas: -1 } }
]);

```

Trabajando la API de Java

Creamos el programa en java:

```

package agregacion;

import com.mongodb.client.*;
import org.bson.Document;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;

import static com.mongodb.client.model.Aggregates.*;
import static com.mongodb.client.model.Filters.*;
import static com.mongodb.client.model.Sorts.*;
import static com.mongodb.client.model.Accumulators.*;
import java.util.Arrays;

public class Facturas {
    // Configuración de la conexión
    private static final String MONGO_URI =
"mongodb://facturas:1234@79.72.63.217:27017/facturas_db?authSource=facturas_db";
    private static final String DATABASE_NAME = "facturas_db";

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

        // Conexión a la base de datos
        MongoClient mongoClient = MongoClient.create(MONGO_URI);
        MongoDBDatabase database = mongoClient.getDatabase(DATABASE_NAME);
        MongoCollection<Document> facturas = database.getCollection("facturas");

        Scanner scanner = new Scanner(System.in);
        int opcion;

        do {
            System.out.println("Menú de operaciones:");
            System.out.println("1. Ver cuántas facturas tiene un cliente");
            System.out.println("2. Calcular el importe total de una factura");
            System.out.println("3. Calcular el importe total de ventas de un cliente");
            System.out.println("4. Ver los productos más vendidos");
            System.out.println("0. Salir");
            System.out.print("Seleccione una opción: ");
            opcion = scanner.nextInt();

            switch (opcion) {
                case 1:
                    System.out.print("Ingrese el nombre del cliente: ");
                    String cliente1 = scanner.next();
                    obtenerFacturasPorCliente(facturas, cliente1);
                    break;

```

```

        case 2:
            System.out.print("Ingrese el número de la factura: ");
            int numeroFactura = scanner.nextInt();
            calcularTotalFactura(facturas, numeroFactura);
            break;
        case 3:
            System.out.print("Ingrese el nombre del cliente: ");
            String cliente2 = scanner.next();
            calcularTotalVentasCliente(facturas, cliente2);
            break;
        case 4:
            obtenerProductosMasVendidos(facturas);
            break;
        case 0:
            System.out.println("Saliendo ... ");
            break;
        default:
            System.out.println("Opción no válida.");
    }
} while (opcion != 0);

scanner.close();
mongoClient.close();
}

private static void obtenerFacturasPorCliente(MongoCollection<Document> facturas, String
cliente) {
    AggregateIterable<Document> resultado = facturas
        .aggregate(Arrays.asList(match(eq("cliente", cliente)),
group("$cliente", sum("totalFacturas", 1))));

    for (Document doc : resultado) {
        System.out.println("Cliente: " + doc.getString("_id"));
        System.out.println("Total de facturas: " + doc.getInteger("totalFacturas"));
    }
    System.out.println();
}

private static void calcularTotalFactura(MongoCollection<Document> facturas, int
numeroFactura) {
    AggregateIterable<Document> resultado = facturas.aggregate(Arrays.asList(
        match(eq("numeroFactura", numeroFactura)), unwind("$lineas"),
        group("$numeroFactura", sum("totalFactura",
            new Document("$multiply",
Arrays.asList("$lineas.precioUnitario", "$lineas.cantidad"))))));

    for (Document doc : resultado) {
        System.out.println("Número de factura: " + doc.getInteger("_id"));
        // Usamos get() y convertimos a double explícitamente
        Number totalFactura = doc.get("totalFactura", Number.class);
        System.out.println("Total de la factura: " + totalFactura.doubleValue());
    }
    System.out.println();
}

private static void calcularTotalVentasCliente(MongoCollection<Document> facturas, String
cliente) {
    AggregateIterable<Document> resultado = facturas.aggregate(
        Arrays.asList(match(eq("cliente", cliente)), unwind("$lineas"),
group("$cliente", sum("totalVentas",
            new Document("$multiply",
Arrays.asList("$lineas.precioUnitario", "$lineas.cantidad"))))));

    for (Document doc : resultado) {
        System.out.println("Cliente: " + doc.getString("_id"));
        // Usamos get() y convertimos a double explícitamente

```

```

        Number totalVentas = doc.get("totalVentas", Number.class);
        System.out.println("Total de ventas: " + totalVentas.doubleValue());
    }
    System.out.println();
}

private static void obtenerProductosMasVendidos(MongoCollection<Document> facturas) {
    AggregateIterable<Document> resultado = facturas.aggregate(
        Arrays.asList(unwind("$lineas"), group("$lineas.articulo",
sum("unidadesVendidas", "$lineas.cantidad")),
                    sort(descending("unidadesVendidas"))));

    System.out.println("Productos más vendidos:");
    for (Document doc : resultado) {
        System.out.println("Producto: " + doc.getString("_id") + " | Unidades
vendidas: "
                                + doc.getInteger("unidadesVendidas"));
    }
    System.out.println();
}
}

```

Y nos aseguramos que nuestro pom.xml tiene 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>
        <dependency>
            <groupId>org.mongodb</groupId>
            <artifactId>mongodb-driver-sync</artifactId>
            <version>4.10.1</version>
        </dependency>
    </dependencies>
</project>

```

Y como podemos ver, nos conectamos sin problema y hacemos las operaciones de agregación correspondientes.

```
Problems Servers Terminal Data Source Explorer Properties Console X
Facturas [Java Application] C:\Program Files\ eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x
Menú de operaciones:
1. Ver cuántas facturas tiene un cliente
2. Calcular el importe total de una factura
3. Calcular el importe total de ventas de un cliente
4. Ver los productos más vendidos
0. Salir
Seleccione una opción: 1
Ingrese el nombre del cliente: Cliente1
Cliente: Cliente1
Total de facturas: 2

Menú de operaciones:
1. Ver cuántas facturas tiene un cliente
2. Calcular el importe total de una factura
3. Calcular el importe total de ventas de un cliente
4. Ver los productos más vendidos
0. Salir
Seleccione una opción: 2
Ingrese el número de la factura: 1
Número de factura: 1
Total de la factura: 40.0

Menú de operaciones:
1. Ver cuántas facturas tiene un cliente
2. Calcular el importe total de una factura
3. Calcular el importe total de ventas de un cliente
4. Ver los productos más vendidos
0. Salir
Seleccione una opción:
```

```
Problems Servers Terminal Data Source Explorer Properties Console X
Facturas [Java Application] C:\Program Files\ eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
Menú de operaciones:
1. Ver cuántas facturas tiene un cliente
2. Calcular el importe total de una factura
3. Calcular el importe total de ventas de un cliente
4. Ver los productos más vendidos
0. Salir
Seleccione una opción: 3
Ingrese el nombre del cliente: Cliente1
Cliente: Cliente1
Total de ventas: 120.0

Menú de operaciones:
1. Ver cuántas facturas tiene un cliente
2. Calcular el importe total de una factura
3. Calcular el importe total de ventas de un cliente
4. Ver los productos más vendidos
0. Salir
Seleccione una opción: 4
Productos más vendidos:
Producto: ProductoA | Unidades vendidas: 7
Producto: ProductoB | Unidades vendidas: 5
Producto: ProductoC | Unidades vendidas: 3
Producto: ProductoD | Unidades vendidas: 2

Menú de operaciones:
1. Ver cuántas facturas tiene un cliente
2. Calcular el importe total de una factura
3. Calcular el importe total de ventas de un cliente
4. Ver los productos más vendidos
0. Salir
Seleccione una opción:
```