TXT

XML

JDBC

SPRING

HYBERNATE

Mapea solo los objetos con las tablas

Configuracion:

- El JPAUTIL que llame al persistence.xml en resources
- El persistence.xml con la unit creada con el driver de jdbc y llamando a los modelos
- Todas las notaciones en el modelo

No hay queries, es mejor, se llama HQL

Hay 3 maneras de pedir datos a base de datos, con sql, con hql a pelo, o con hql en una constante con la anotación nativa de hibernate en el model, que es como lo tenemos que hacer.

Si no metes @column en un atributo de un objeto del model no te lo devuelve el hibernate, si ese atributo es un objeto deberas poner @Transient

El get de un id como PK se hace con un entity.find(ObjetoQueDevuelve,id)

Siempre que modificas la base de datos tienes que hacer una transacción

Las excepciones se throwean de normal, luego haces un e.getcause instanceof “tuexcepciondehybernate” y la sacas, porque se hace asi ahora, también puedes hacer un catch de PersistenceException en vez de Exception

??que es cascade persist (solo para add) es que te guarde los datos en cascada

Cuando tienes un onetomany en la otra clase debes tener un manytoone, onetomany usa muchos recursos para que encuentre la fk a la que se refiere, por lo que no lo uses de mas o suspenso en el examen. Si en la lista de un objeto que hay relación 1 n ese objeto de la lista (n) tiene un id de referencia a su relación (1) y luego aparte esa relación tiene una lista declarada (lista de n), hybernate entra como en una especie de bucle al mostrar los datos, por que tienes que usar la notación @ToString.Exclude, también en la lista debes cambiar la manera de la que pide la lista de N con fetch= FetchType.EAGER o pedir un .size para que el vago por defecto detecte que se necesite la lista y lo cargue

MONGODB

No usa sql, sus datos son mas rapidos, mas consultas simultaneas, para almacenar datos masivamente.

Instituto: mongodb://informatica.iesquevedo.es:2323

Creando Container:

docker run --name mongodb -d -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=root - e MONGO_INITDB_ROOT_PASSWORD=root mongo

Acceso shell:

mongosh admin -u root -p root

Creando “tablas”, ahora se llaman Collections

Las proyecciones son para especificar lo que quieres que te devuelva, especificas que quieres recibir desde el “select” en vez se hacer un “select *”:

En Java:

Añadir dependencia del driver de mongo y el gson para el json

```
<dependency>
   <groupId>org.mongodb</groupId>
   <artifactId>mongo-java-driver</artifactId>
   <version>3.12.12</version>
</dependency>

<dependency>
   <groupId>com.google.code.gson</groupId>
   <artifactId>gson</artifactId>
   <version>2.10.1</version>
</dependency>
```

Declaras en un trywithresources

Assignment

Customers, orders, menu items, order items

Ejemplos Consultas

SQL (Structured Query Language)

1. **Selección Básica**
   - **Enunciado:** Selecciona todos los registros de la tabla `clientes`.
   - **Consulta:**
     ```
     SELECT * FROM clientes;
     ```
2. **Selección con Condiciones**
   - **Enunciado:** Selecciona todos los registros de la tabla `clientes` donde la edad sea mayor a 30.
   - **Consulta:**
     ```
     SELECT * FROM clientes WHERE edad > 30;
     ```
3. **Selección con LIKE**
   - **Enunciado:** Selecciona todos los registros de la tabla `clientes` donde el nombre empiece con 'J'.
   - **Consulta:**
     ```
     SELECT * FROM clientes WHERE nombre LIKE 'J%';
     ```
4. **Selección con IN**
   - **Enunciado:** Selecciona todos los registros de la tabla `clientes` donde el país sea 'España' o 'Francia'.
   - **Consulta:**
     ```
     SELECT * FROM clientes WHERE pais IN ('España', 'Francia')
     ;
     ```
5. **Ordenación**
   - **Enunciado:** Selecciona todos los registros de la tabla `clientes` ordenados por edad de manera descendente.
   - **Consulta:**
     ```
     SELECT * FROM clientes ORDER BY edad DESC;
     ```
6. **Agrupación**
   - **Enunciado:** Agrupa los registros de la tabla `clientes` por país y muestra el número de clientes en cada país.
   - **Consulta:**
     ```
     SELECT pais, COUNT(*) as total_clientes FROM clientes GROU
     P BY pais;
     ```
7. **Filtrado de Agrupación (HAVING)**
   - **Enunciado:** Agrupa los registros de la tabla `clientes` por país y muestra aquellos grupos que tengan más de 10 clientes.

- **Consulta:**

```
SELECT pais, COUNT(*) as total_clientes FROM clientes GROU
P BY pais HAVING COUNT(*) > 10;
```

8. **Joins**

   - **Enunciado:** Selecciona todos los pedidos junto con la información de los clientes que los realizaron.
   - **Consulta:**

```
SELECT pedidos.*, clientes.nombre, clientes.email
FROM pedidos
JOIN clientes ON pedidos.cliente_id = clientes.id;
```

9. **Subconsultas**

   - **Enunciado:** Selecciona todos los clientes que han realizado pedidos.
   - **Consulta:**

```
SELECT * FROM clientes WHERE id IN (SELECT cliente_id FROM
pedidos);
```

10. **Inserción**

    - **Enunciado:** Inserta un nuevo registro en la tabla `clientes`.
    - **Consulta:**

```
INSERT INTO clientes (nombre, edad, pais) VALUES ('Juan Pé
rez', 25, 'España');
```

11. **Actualización**

    - **Enunciado:** Actualiza el país de todos los clientes a 'España' donde el país sea 'Francia'.
    - **Consulta:**

```
UPDATE clientes SET pais = 'España' WHERE pais = 'Francia'
;
```

12. **Eliminación**

    - **Enunciado:** Elimina todos los clientes que tienen más de 80 años.
    - **Consulta:**

```
DELETE FROM clientes WHERE edad > 80;
```

HQL (Hibernate Query Language)

1. **Selección Básica**

   - **Enunciado:** Selecciona todos los registros de la entidad `Cliente`.
   - **Consulta:**

```
FROM Cliente;
```

2. **Selección con Condiciones**

- **Enunciado:** Selecciona todos los registros de la entidad `Cliente` donde la edad sea mayor a 30.
- **Consulta:**
  ```
  FROM Cliente WHERE edad > 30;
  ```

3. **Selección con LIKE**
   - **Enunciado:** Selecciona todos los registros de la entidad `Cliente` donde el nombre empiece con 'J'.
   - **Consulta:**
     ```
     FROM Cliente WHERE nombre LIKE 'J%';
     ```

4. **Selección con IN**
   - **Enunciado:** Selecciona todos los registros de la entidad `Cliente` donde el país sea 'España' o 'Francia'.
   - **Consulta:**
     ```
     FROM Cliente WHERE pais IN ('España', 'Francia');
     ```

5. **Ordenación**
   - **Enunciado:** Selecciona todos los registros de la entidad `Cliente` ordenados por edad de manera descendente.
   - **Consulta:**
     ```
     FROM Cliente ORDER BY edad DESC;
     ```

6. **Agrupación**
   - **Enunciado:** Agrupa los registros de la entidad `Cliente` por país y muestra el número de clientes en cada país.
   - **Consulta:**
     ```
     SELECT pais, COUNT(*) FROM Cliente GROUP BY pais;
     ```

7. **Filtrado de Agrupación (HAVING)**
   - **Enunciado:** Agrupa los registros de la entidad `Cliente` por país y muestra aquellos grupos que tengan más de 10 clientes.
   - **Consulta:**
     ```
     SELECT pais, COUNT(*) FROM Cliente GROUP BY pais HAVING CO
     UNT(*) > 10;
     ```

8. **Joins**
   - **Enunciado:** Selecciona todos los pedidos junto con la información de los clientes que los realizaron.
   - **Consulta:**
     ```
     FROM Pedido p JOIN p.cliente c;
     ```

9. **Subconsultas**
   - **Enunciado:** Selecciona todos los clientes que han realizado pedidos.

- **Consulta:**
  ```
  FROM Cliente c WHERE c.id IN (SELECT p.cliente.id FROM Ped
  ido p);
  ```

MongoDB

1. **Selección Básica**
   - **Enunciado:** Selecciona todos los documentos de la colección `clientes`.
   - **Consulta:**
     ```
     db.clientes.find();
     ```
2. **Selección con Condiciones**
   - **Enunciado:** Selecciona todos los documentos de la colección `clientes` donde la edad sea mayor a 30.
   - **Consulta:**
     ```
     db.clientes.find({ edad: { $gt: 30 } });
     ```
3. **Selección con LIKE (Regex)**
   - **Enunciado:** Selecciona todos los documentos de la colección `clientes` donde el nombre empiece con 'J'.
   - **Consulta:**
     ```
     db.clientes.find({ nombre: /^J/ });
     ```
4. **Selección con IN**
   - **Enunciado:** Selecciona todos los documentos de la colección `clientes` donde el país sea 'España' o 'Francia'.
   - **Consulta:**
     ```
     db.clientes.find({ pais: { $in: ['España', 'Francia'] } })
     ;
     ```
5. **Ordenación**
   - **Enunciado:** Selecciona todos los documentos de la colección `clientes` ordenados por edad de manera descendente.
   - **Consulta:**
     ```
     db.clientes.find().sort({ edad: -1 });
     ```
6. **Agrupación**
   - **Enunciado:** Agrupa los documentos de la colección `clientes` por país y muestra el número de clientes en cada país.
   - **Consulta:**
     ```
     db.clientes.aggregate([
       { $group: { _id: "$pais", total_clientes: { $sum: 1 } }
     }
     ]);
     ```
7. **Filtrado de Agrupación (HAVING)**

- **Enunciado:** Agrupa los documentos de la colección `clientes` por país y muestra aquellos grupos que tengan más de 10 clientes.
- **Consulta:**

```
db.clientes.aggregate([
  { $group: { _id: "$pais", total_clientes: { $sum: 1 } }
},
  { $match: { total_clientes: { $gt: 10 } } }
]);
```

8. **Joins (Lookup)**

- **Enunciado:** Selecciona todos los pedidos junto con la información de los clientes que los realizaron.
- **Consulta:**

```
db.pedidos.aggregate([
  {
    $lookup: {
      from: "clientes",
      localField: "cliente_id",
      foreignField: "_id",
      as: "cliente_info"
    }
  }
]);
```

9. **Inserción**

- **Enunciado:** Inserta un nuevo documento en la colección `clientes`.
- **Consulta:**

```
db.clientes.insertOne({ nombre: "Juan Pérez", edad: 25, pa
is: "España" });
```

10. **Actualización**

- **Enunciado:** Actualiza el país de todos los documentos en la colección `clientes` donde el país sea 'Francia' a 'España'.
- **Consulta:**

```
db.clientes.updateMany
```

({ pais: "Francia" }, { $set: { pais: "España" } }); ```

11. **Eliminación**
    - **Enunciado:** Elimina todos los documentos en la colección `clientes` que tengan más de 80 años.
    - **Consulta:**

```
db.clientes.deleteMany({ edad: { $gt: 80 } });
```

Estos ejemplos cubren una amplia gama de consultas posibles en SQL, HQL y MongoDB. Tener estas consultas a mano te ayudará a resolver rápidamente cualquier tipo de problema que pueda surgir durante el examen.

MONGO AGGREGATIONS

Aquí tienes varios ejemplos de agregaciones en MongoDB con consultas que cubren diversas necesidades. Además, cada ejemplo incluye una explicación sobre cómo extraer atributos específicos de los documentos resultantes.

Ejemplos de Agregaciones en MongoDB

11. **Conteo de Documentos**
    - **Enunciado:** Cuenta el número total de clientes en la colección `clientes`.
    - **Consulta:**
```
db.clientes.aggregate([
  { $count: "total_clientes" }
]);
```

12. **Agrupación y Conteo**
    - **Enunciado:** Agrupa los clientes por país y cuenta cuántos clientes hay en cada país.
    - **Consulta:**
```
db.clientes.aggregate([
  { $group: { _id: "$pais", total_clientes: { $sum: 1 } }
}
]);
```

13. **Filtrado y Agrupación**
    - **Enunciado:** Agrupa los clientes por país y cuenta solo aquellos con edad mayor a 30.
    - **Consulta:**
```
db.clientes.aggregate([
  { $match: { edad: { $gt: 30 } } },
  { $group: { _id: "$pais", total_clientes: { $sum: 1 } }
}
]);
```

14. **Promedio**
    - **Enunciado:** Calcula la edad promedio de los clientes en cada país.
    - **Consulta:**
```
db.clientes.aggregate([
  { $group: { _id: "$pais", edad_promedio: { $avg: "$edad"
} } }
]);
```

15. **Máximo**
    - **Enunciado:** Encuentra la mayor edad de los clientes en cada país.
    - **Consulta:**

```
db.clientes.aggregate([
  { $group: { _id: "$pais", edad_maxima: { $max: "$edad" }
} }
]);
```

16. **Mínimo**

    – **Enunciado:** Encuentra la menor edad de los clientes en cada país.

    – **Consulta:**

```
db.clientes.aggregate([
  { $group: { _id: "$pais", edad_minima: { $min: "$edad" }
} }
]);
```

17. **Sumatoria**

    – **Enunciado:** Calcula la sumatoria de edades de los clientes en cada país.

    – **Consulta:**

```
db.clientes.aggregate([
  { $group: { _id: "$pais", sum_edad: { $sum: "$edad" } }
}
]);
```

18. **Agrupación y Filtrado Posterior (HAVING)**

    – **Enunciado:** Agrupa los clientes por país y muestra aquellos grupos que tengan más de 10 clientes.

    – **Consulta:**

```
db.clientes.aggregate([
  { $group: { _id: "$pais", total_clientes: { $sum: 1 } }
},
  { $match: { total_clientes: { $gt: 10 } } }
]);
```

19. **Joins (Lookup)**

    – **Enunciado:** Muestra todos los pedidos junto con la información de los clientes que los realizaron.

    – **Consulta:**

```
db.pedidos.aggregate([
  {
    $lookup: {
      from: "clientes",
      localField: "cliente_id",
      foreignField: "_id",
      as: "cliente_info"
    }
  }
]);
```

20. **Unwind**

- **Enunciado:** Desenrolla un array de productos en pedidos para analizar cada producto por separado.
- **Consulta:**

```
db.pedidos.aggregate([
  { $unwind: "$productos" }
]);
```

21. **Pipeline Completo**

- **Enunciado:** Encuentra el cliente con el mayor número de pedidos y muestra su nombre.
- **Consulta:**

```
db.pedidos.aggregate([
  { $group: { _id: "$cliente_id", total_pedidos: { $sum: 1
} } },
  { $sort: { total_pedidos: -1 } },
  { $limit: 1 },
  {
    $lookup: {
      from: "clientes",
      localField: "_id",
      foreignField: "_id",
      as: "cliente_info"
    }
  },
  { $unwind: "$cliente_info" },
  { $project: { nombre: "$cliente_info.nombre", total_pedi
dos: 1 } }
]);
```

Extracción de Atributos Específicos

Para extraer un atributo específico de los resultados de una agregación en MongoDB y usarlo en un lenguaje de programación como Java, podrías hacer algo similar a lo siguiente:

Supongamos que tienes un resultado de agregación y quieres obtener el nombre del empleado con más compras.

Consulta:

```
db.pedidos.aggregate([
  { $group: { _id: "$empleado_id", total_compras: { $sum: 1 } } },
  { $sort: { total_compras: -1 } },
  { $limit: 1 },
  {
    $lookup: {
      from: "empleados",
      localField: "_id",
      foreignField: "_id",
      as: "empleado_info"
    }
  },
```

```
  { $unwind: "$empleado_info" },
  { $project: { nombre: "$empleado_info.nombre", total_compras: 1 } }
]);
```

Código Java para obtener el atributo específico:

```
import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import org.bson.Document;
import static com.mongodb.client.model.Aggregates.*;
import static com.mongodb.client.model.Accumulators.*;
import static com.mongodb.client.model.Filters.*;
import static com.mongodb.client.model.Sorts.*;
import static com.mongodb.client.model.Projections.*;

import java.util.Arrays;
import java.util.List;

public class MongoDBExample {
    public static void main(String[] args) {
        MongoClient mongoClient = new MongoClient("localhost", 27017);
        MongoDatabase database = mongoClient.getDatabase("tu_base_de_d
atos");
        MongoCollection<Document> collection = database.getCollection(
"pedidos");

        List<Document> result = collection.aggregate(Arrays.asList(
            group("$empleado_id", sum("total_compras", 1)),
            sort(descending("total_compras")),
            limit(1),
            lookup("empleados", "_id", "_id", "empleado_info"),
            unwind("$empleado_info"),
            project(fields(include("empleado_info.nombre", "total_comp
ras")))
        )).into(new ArrayList<>());

        if (!result.isEmpty()) {
            Document empleado = result.get(0);
            String nombre = empleado.get("empleado_info", Document.cla
ss).getString("nombre");
            System.out.println("Nombre del empleado con más compras: "
+ nombre);
        }

        mongoClient.close();
    }
}
```

Claro, aquí tienes todos los ejemplos de agregaciones de MongoDB, incluyendo el código Java necesario para realizar cada una de las consultas y extraer atributos específicos.

Configuración Previa

Primero, asegúrate de tener las dependencias necesarias en tu proyecto. Si estás utilizando Maven, añade la siguiente dependencia en tu `pom.xml`:

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.6.0</version>
</dependency>
```

Ejemplos de Agregaciones en MongoDB con Java

22. **Conteo de Documentos**

    - **Enunciado:** Cuenta el número total de clientes en la colección `clientes`.
    - **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_de_datos");
MongoCollection<Document> collection = database.getCollection("clientes");

List<Document> result = collection.aggregate(Arrays.asList(
    Aggregates.count("total_clientes")
)).into(new ArrayList<>());

if (!result.isEmpty()) {
    int totalClientes = result.get(0).getInteger("total_clientes");
    System.out.println("Total de clientes: " + totalClientes);
}

mongoClient.close();
```

23. **Agrupación y Conteo**

    - **Enunciado:** Agrupa los clientes por país y cuenta cuántos clientes hay en cada país.
    - **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_de_datos");
MongoCollection<Document> collection = database.getCollection("clientes");

List<Document> result = collection.aggregate(Arrays.asList(
```

```
    Aggregates.group("$pais", Accumulators.sum("total_clie
ntes", 1))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    int totalClientes = doc.getInteger("total_clientes");
    System.out.println("País: " + pais + ", Total de clien
tes: " + totalClientes);
}

mongoClient.close();
```

24. **Filtrado y Agrupación**

- **Enunciado:** Agrupa los clientes por país y cuenta solo aquellos con edad mayor a 30.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("clientes");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.match(Filters.gt("edad", 30)),
    Aggregates.group("$pais", Accumulators.sum("total_clie
ntes", 1))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    int totalClientes = doc.getInteger("total_clientes");
    System.out.println("País: " + pais + ", Total de clien
tes con edad > 30: " + totalClientes);
}

mongoClient.close();
```

25. **Promedio**

- **Enunciado:** Calcula la edad promedio de los clientes en cada país.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("clientes");

List<Document> result = collection.aggregate(Arrays.asList
```

```
(
    Aggregates.group("$pais", Accumulators.avg("edad_prome
dio", "$edad"))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    double edadPromedio = doc.getDouble("edad_promedio");
    System.out.println("País: " + pais + ", Edad promedio:
" + edadPromedio);
}

mongoClient.close();
```

26. **Máximo**

- **Enunciado:** Encuentra la mayor edad de los clientes en cada país.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("clientes");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.group("$pais", Accumulators.max("edad_maxim
a", "$edad"))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    int edadMaxima = doc.getInteger("edad_maxima");
    System.out.println("País: " + pais + ", Edad máxima: "
+ edadMaxima);
}

mongoClient.close();
```

27. **Mínimo**

- **Enunciado:** Encuentra la menor edad de los clientes en cada país.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("clientes");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.group("$pais", Accumulators.min("edad_minim
```

```
a", "$edad"))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    int edadMinima = doc.getInteger("edad_minima");
    System.out.println("País: " + pais + ", Edad mínima: "
+ edadMinima);
}

mongoClient.close();
```

28. **Sumatoria**

- **Enunciado:** Calcula la sumatoria de edades de los clientes en cada país.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("clientes");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.group("$pais", Accumulators.sum("sum_edad",
"$edad"))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    int sumEdad = doc.getInteger("sum_edad");
    System.out.println("País: " + pais + ", Sumatoria de e
dades: " + sumEdad);
}

mongoClient.close();
```

29. **Agrupación y Filtrado Posterior (HAVING)**

- **Enunciado:** Agrupa los clientes por país y muestra aquellos grupos que tengan más de 10 clientes.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("clientes");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.group("$pais", Accumulators.sum("total_clie
```

```
ntes", 1)),
    Aggregates.match(Filters.gt("total_clientes", 10))
)).into(new ArrayList<>());

for (Document doc : result) {
    String pais = doc.getString("_id");
    int totalClientes = doc.getInteger("total_clientes");
    System.out.println("País: " + pais + ", Total de clien
tes > 10: " + totalClientes);
}

mongoClient.close();
```

30. **Joins (Lookup)**

- **Enunciado:** Muestra todos los pedidos junto con la información de los clientes que los realizaron.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("pedidos");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.lookup("clientes", "cliente_id", "_id", "cl
iente_info")
)).into(new ArrayList<>());

for (Document doc : result) {
    System.out.println("Pedido: " + doc.toJson());
}

mongoClient.close();
```

31. **Unwind**

- **Enunciado:** Desenrolla un array de productos en pedidos para analizar cada producto por separado.
- **Código Java:**

```
MongoClient mongoClient = MongoClients.create("mongodb://l
ocalhost:27017");
MongoDatabase database = mongoClient.getDatabase("tu_base_
de_datos");
MongoCollection<Document> collection = database.getCollect
ion("pedidos");

List<Document> result = collection.aggregate(Arrays.asList
(
    Aggregates.unwind("$productos")
)).into(new ArrayList<>());
```

```
for (Document doc : result) {
    System.out.println("Pedido con producto desenrollado:
" + doc.toJson());
}

mongoClient.close();
```

32. **Pipeline Completo**

- **Enunciado:** Encuentra el cliente con el mayor número de pedidos y muestra su nombre.
- **Código Java:**

```
try (MongoClient mongoClient =
MongoClients.create("mongodb://localhost:27017")) {
        MongoDatabase database =
mongoClient.getDatabase("tu_base_de_datos");
        MongoCollection<Document> pedidosCollection =
database.getCollection("pedidos");
        MongoCollection<Document> clientesCollection =
database.getCollection("clientes");

        List<Document> pipeline = Arrays.asList(
          Aggregates.group("$cliente_id",
Accumulators.sum("total_pedidos", 1)),
          Aggregates.sort(Sorts.descending("total_pedidos")),
          Aggregates.limit(1),
          Aggregates.lookup("clientes", "_id", "_id", "cliente_info"),
          Aggregates.unwind("$cliente_info"),
          Aggregates.project(Projections.fields(
            Projections.include("cliente_info.nombre"),
            Projections.excludeId()
          ))
        );

        AggregateIterable<Document> results =
pedidosCollection.aggregate(pipeline);

        for (Document doc : results) {
          String nombreCliente =
doc.getEmbedded(Arrays.asList("cliente_info", "nombre"), String.class);
          System.out.println("Nombre del cliente con más pedidos: " +
nombreCliente);
        }
      } catch (Exception e) {
        e.printStackTrace();
```

```
}
```