Taller de CRUD: READ en MongoDB

Requisitos previos

Antes de comenzar, asegúrate de tener **uno de los siguientes entornos de ejecución**:

- MongoDB instalado localmente
- MongoDB Compass (interfaz gráfica)
- Docker instalado para contenedor MongoDB
- Mongo Shell (mongosh) o conexión mediante terminal

Crear la base de datos LabTestDB y la colección clientes

A) Si usas la terminal con mongosh:

1. Abre el terminal y ejecuta: `mongosh`
2. Crea o cambia a la base de datos: `use LabTestDB`
3. Verifica que estás en la base correcta: `db`

B) Importar los datos del archivo .json (opcional si se desea precargar datos, no es necesario para el taller CREATE)

Puedes utilizar el archivo LabTestDB.clientes.json para precargar datos de ejemplo antes de hacer inserts nuevos. En el entorno local, haz lo siguiente:

1. Guarda el archivo como clientes.json
2. Ejecuta desde la terminal:

```
mongoimport --db LabTestDB --collection clientes --file clientes.json --
jsonArray
```

Asegúrate de estar en el mismo directorio donde está el archivo clientes.json.

C) Con Docker (entorno desechable para pruebas):

1. Levanta un contenedor de MongoDB:

```
docker run -d -p 27017:27017 --name mongo-taller mongo
```

2. Accede al contenedor: `docker exec -it mongo-taller mongosh`
3. Luego crea la base: `use LabTestDB`

Verifica que la colección clientes está vacía (para los ejercicios de CREATE)

```
db.clientes.find()
```

- Si devuelve una lista vacía [], estás listo para comenzar.
- Si hay datos, puedes limpiar la colección antes del taller con:
  `db.clientes.deleteMany({})`

READ en MongoDB

Base de trabajo:

```
use("LabTestDB");
const coleccion = db.getCollection("clientes_read");
```

Comparación

1. Buscar clientes con puntos de fidelidad exactamente 100 ($eq)

```
coleccion.find({ puntos_fidelidad: { $eq: 100 } })
```

2. Clientes con más de 500 puntos ($gt)

```
coleccion.find({ puntos_fidelidad: { $gt: 500 } })
```

3. Clientes con puntos mayores o iguales a 1000 ($gte)

```
coleccion.find({ puntos_fidelidad: { $gte: 1000 } })
```

4. Clientes que tienen el estatus en una lista ($in)

```
coleccion.find({ estatus: { $in: ["activo", "verificado"] } })
```

5. Clientes con puntos menores a 100 ($lt)

```
coleccion.find({ puntos_fidelidad: { $lt: 100 } })
```

6. Clientes con puntos menores o iguales a 50 ($lte)

```
coleccion.find({ puntos_fidelidad: { $lte: 50 } })
```

7. Clientes cuyo estatus no es "activo" ($ne)

```
coleccion.find({ estatus: { $ne: "activo" } })
```

8. Clientes que NO tienen estatus en una lista ($nin)

```
coleccion.find({ estatus: { $nin: ["activo", "inactivo"] } })
```

Lógicos y Elementos

9. Clientes activos y con más de 500 puntos ($and)

```
coleccion.find({ $and: [
 { estatus: "activo" },
 { puntos_fidelidad: { $gt: 500 } }
] })
```

10. Clientes con estatus inactivo o sin email ($or)

```
coleccion.find({ $or: [
 { estatus: "inactivo" },
 { email: { $exists: false } }
] })
```

11. Clientes que NO tengan puntos mayores a 1000 ($not)

```
coleccion.find({ puntos_fidelidad: { $not: { $gt: 1000 } } })
```

12. Clientes que NO son activos ni tienen puntos altos ($nor)

```
coleccion.find({ $nor: [
 { estatus: "activo" },
 { puntos_fidelidad: { $gt: 1000 } }
] })
```

13. Clientes que tienen el campo telefono ($exists)

```
coleccion.find({ telefono: { $exists: true } })
```

14. Clientes donde puntos_fidelidad es de tipo int ($type)

```
coleccion.find({ puntos_fidelidad: { $type: "int" } })
```

Evaluación

15. Clientes cuyo campo de puntos es divisible por 100 ($mod)

```
coleccion.find({ puntos_fidelidad: { $mod: [100, 0] } })
```

16. Clientes cuyo nombre empiece con "Carlos" ($regex)

```
coleccion.find({ nombre: { $regex: /^Carlos/ } })
```

17. Clientes cuyo email termina en .com (regex insensible a mayúsculas)

```
coleccion.find({ email: { $regex: /\.com$/, $options: "i" } })
```

18. Clientes donde la suma de campos cumpla condición ($expr)

```
coleccion.find({ $expr: { $gt: ["$puntos_fidelidad", 1000] } })
```

19. Clientes con texto en el campo indexado ($text)

```
coleccion.find({ $text: { $search: "urgente" } })
```

20. Clientes usando condición en JS (no recomendado, pero posible) ($where)

```
coleccion.find({ $where: function() {
 return this.puntos_fidelidad > 500 && this.estatus === "activo";
} })
```

Arrays y Proyección

21. Clientes con categorías favoritas que incluyen "ropa" ($all)

```
coleccion.find({ "preferencias.categorias_favoritas": { $all: ["ropa"] } })
```

22. Clientes con alguna categoría que sea "hogar" y notificaciones activadas ($elemMatch)

```
coleccion.find({
 "preferencias": {
  $elemMatch: {
   categorias_favoritas: "hogar",
   notificaciones: true
  } } })
```

23. Clientes con exactamente 3 accesos ($size)

```
coleccion.find({ historial_acceso: { $size: 3 } })
```

24. Proyectar solo nombre y puntos (proyección con campos)

```
coleccion.find({}, { nombre: 1, puntos_fidelidad: 1, _id: 0 })
```

25. Mostrar solo el primer acceso del historial ($slice)

```
coleccion.find({}, { historial_acceso: { $slice: 1 } })
```

26. Mostrar los 2 últimos accesos

```
coleccion.find({}, { historial_acceso: { $slice: -2 } })
```

Consultas Avanzadas

27. Clientes que viven en Guadalajara (campo anidado)

```
coleccion.find({ "direccion.ciudad": "Guadalajara" })
```

28. Clientes cuyo nombre tiene más de 10 caracteres

```
coleccion.find({
 $expr: { $gt: [ { $strLenCP: "$nombre" }, 10 ] }
})
```

29. Clientes registrados entre dos fechas

```
coleccion.find({
 fecha_registro: {
  $gte: ISODate("2023-01-01"),
  $lt: ISODate("2024-01-01")
 }
})
```

30. Clientes con "tarjeta_credito" en sus métodos de pago ($elemMatch)

```
coleccion.find({
 metodos_pago: {
  $elemMatch: { tipo: "tarjeta_credito" }
 }
})
```

31. Clientes sin preferencias registradas ($exists)

```
coleccion.find({ preferencias: { $exists: false } })
```

32. Clientes con puntuación múltiplo de 50

```
coleccion.find({ puntos_fidelidad: { $mod: [50, 0] } })
```

33. Proyectar el campo comentarios y ocultar _id

```
coleccion.find({}, { comentarios: 1, _id: 0 })
```

34. Clientes con nombres que contengan "ana" en cualquier parte (insensible)

```
coleccion.find({ nombre: { $regex: /ana/i } })
```

35. Buscar clientes con texto VIP en un campo indexado

```
coleccion.find({ $text: { $search: "VIP" } })
```

Paginación, orden y límites

36. Obtener los primeros 5 clientes activos ordenados por nombre

```
coleccion.find({ estatus: "activo" }).sort({ nombre: 1 }).limit(5)
```

37. Saltar 5 y obtener 5 más (paginación)

```
coleccion.find({ estatus: "activo" }).skip(5).limit(5)
```

38. Ordenar por fecha de registro descendente

```
coleccion.find({}).sort({ fecha_registro: -1 })
```

39. Mostrar solo clientes con puntos en el top 10

```
coleccion.find({}).sort({ puntos_fidelidad: -1 }).limit(10)
```

40. Contar documentos con categoría "hogar"

```
coleccion.countDocuments({ "preferencias.categorias_favoritas": "hogar" })
```

Combinaciones Complejas

41. Clientes activos con más de 2 accesos y al menos un método de pago

```
coleccion.find({
 $and: [
  { estatus: "activo" },
  { historial_acceso: { $size: { $gte: 3 } } },
  { "metodos_pago.0": { $exists: true } }
 ]
})
```

42. Clientes que tengan algún método de pago caducado (regex)

```
coleccion.find({
 "metodos_pago.caducidad": { $regex: /^0[1-9]\/2[0-3]$/ }
})
```

43. Clientes donde al menos un acceso fue hecho desde "Mobile"

```
coleccion.find({
 historial_acceso: { $elemMatch: { dispositivo: "Mobile" } }
})
```

44. Clientes cuyo contacto_emergencia.telefono contiene "800"

```
coleccion.find({ "contacto_emergencia.telefono": /800/ })
```

45. Clientes que no tengan ninguna categoría favorita ($size: 0)

```
coleccion.find({ "preferencias.categorias_favoritas": { $size: 0 } })
```

Ejercicios de Refuerzo con $type

46. Clientes donde telefono es tipo string

```
coleccion.find({ telefono: { $type: "string" } })
```

47. Clientes donde puntos_fidelidad es double

```
coleccion.find({ puntos_fidelidad: { $type: "double" } })
```

48. Clientes donde historial_acceso es de tipo array

```
coleccion.find({ historial_acceso: { $type: "array" } })
```

Uso de $jsonSchema

49. Clientes que cumplan un esquema básico

```
coleccion.find({
 $jsonSchema: {
  required: ["nombre", "email"],
  properties: {
   nombre: { bsonType: "string" },
   email: { bsonType: "string" },
   puntos_fidelidad: { bsonType: "int" }
  }
 }
})
```

Filtrar por expresión avanzada usando $expr con condicional

50. Clientes cuyo nombre empieza igual que el contacto de emergencia

```
coleccion.find({
 $expr: {
  $eq: [
   { $substrCP: ["$nombre", 0, 3] },
   { $substrCP: ["$contacto_emergencia.nombre", 0, 3] }
  ]
 }
})
```

Evaluación: Ejercicios sobre READ en MongoDB

Colección a utilizar:

```
use("LabTestDB");
const coleccion = db.getCollection("clientes_eval_read");
```

1. Encuentra clientes que tengan el campo comentarios presente y cuyo valor contenga la palabra “urgente” sin importar mayúsculas/minúsculas.

Completa el uso de $regex y $options.

```
coleccion.find({
 comentarios: {
  $regex: /* completa aquí */,
  $options: /* completa aquí */
 }
});
```

2. Muestra solo los campos nombre, puntos_fidelidad y los dos últimos elementos de historial_acceso. **Tip:** usa $slice.

```
coleccion.find(
 {},
 {
  nombre: 1,
  puntos_fidelidad: 1,
  historial_acceso: { $slice: /* completa aquí */ },
  _id: 0
 }
);
```

3. Devuelve los clientes cuya categoría favorita sea exactamente "hogar" y tengan activadas las notificaciones.

Usa $elemMatch si es necesario. Si no existe estructura adecuada, se deben modificar los datos.

```
coleccion.find({
 preferencias: {
  /* completa aquí */
 }
});
```

4. Encuentra todos los clientes que no tienen métodos de pago registrados (array vacío).

Si los documentos no tienen arrays vacíos, **modifícalos previamente** para que al menos un caso se evalúe.

```
coleccion.find({
 metodos_pago: { $size: /* completa aquí */ }
});
```

5. Devuelve solo los clientes que tengan exactamente 4 letras en el nombre, sin importar mayúsculas. Usa $regex con cuantificadores.

```
coleccion.find({
 nombre: { $regex: /* expresión regular */, $options: "i" }
});
```

6. Muestra clientes donde puntos_fidelidad sea múltiplo de 200, usando $mod.

```
coleccion.find({
 puntos_fidelidad: { $mod: [/* divisor */, /* residuo */] }
});
```

7. Busca todos los clientes cuyo email termine en @outlook.com y proyecta solo su nombre y correo. Usa expresiones regulares.

```
coleccion.find(
 { email: { $regex: /* completa aquí */ } },
 { nombre: 1, email: 1, _id: 0 }
);
```

8. Muestra los clientes cuya ciudad en la dirección tenga exactamente 7 caracteres.

Se requiere $expr y $strLenCP.

```
coleccion.find({
 $expr: {
  $eq: [ { $strLenCP: "$direccion.ciudad" }, /* completa aquí */ ]
 }
});
```

9. Encuentra todos los clientes donde el nombre del cliente y el nombre del contacto de emergencia empiecen con la misma letra. Se debe usar $expr con $substrCP.

```
coleccion.find({
 $expr: {
  $eq: [
   { $substrCP: [ "$nombre", 0, 1 ] },
   { $substrCP: [ "$contacto_emergencia.nombre", 0, 1 ] }
  ]
 }
});
```

10. Devuelve los primeros 5 clientes activos ordenados por puntos de fidelidad descendente. Aplica sort, limit y un filtro adecuado.

```
coleccion.find({ estatus: /* completa aquí */ })
      .sort({ puntos_fidelidad: /* completa aquí */ })
      .limit(5);
```