Taller de CRUD: CREATE 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({})`

CREATE en MongoDB

1. Insertar un cliente básico con solo `nombre` y `email`:

```
db.clientes.insertOne({
  nombre: "Carlos Martínez",
  email: "carlos.martinez@example.com"
})
```

2. Insertar con `_id` personalizado:

```
db.clientes.insertOne({
  _id: "CL100",
  nombre: "Laura Peña",
  email: "laura.pena@example.com"
})
```

3. Insertar un documento completo como en el JSON base:

```
db.clientes.insertOne({
  _id: "CL101",
  nombre: "Andrés López",
  direccion: {
    calle: "Calle 55",
    ciudad: "Guadalajara",
    codigoPostal: "45710"
  },
  email: "andres.lopez@example.com",
  telefono: "+52-1-555-1234",
  fecha_registro: new Date("2024-04-19T12:00:00Z"),
  estatus: "activo",
  puntos_fidelidad: 100,
  preferencias: {
    categorias_favoritas: ["ropa", "electrónicos"],
    notificaciones: true
  },
  historial_acceso: [],
  metodos_pago: [],
  contacto_emergencia: {
    nombre: "María",
    telefono: "+52-1-555-7890",
    relacion: "pareja"
  }
})
```

4. Insertar múltiples documentos a la vez:

```
db.clientes.insertMany([
  { nombre: "Pedro Sánchez", email: "pedro.sanchez@example.com" },
  { nombre: "Ana Ruiz", email: "ana.ruiz@example.com" }
])
```

5. Insertar con un array vacío para `historial_acceso` y `metodos_pago`:

```
db.clientes.insertOne({
  nombre: "Lucía Gómez",
  email: "lucia.gomez@example.com",
  historial_acceso: [],
```

```
  metodos_pago: []
})
```

6. Insertar con fechas usando `ISODate()`:

```
db.clientes.insertOne({
  nombre: "Manuel Ortega",
  fecha_registro: ISODate("2024-01-01T00:00:00Z")
})
```

7. Insertar usando un objeto `preferencias` complejo:

```
db.clientes.insertOne({
  nombre: "Natalia Vélez",
  preferencias: {
    categorias_favoritas: ["libros", "hogar"],
    notificaciones: false
  }
})
```

8. Insertar un documento sin campo `email`:

```
db.clientes.insertOne({
  nombre: "Samuel Hernández",
  telefono: "+52-1-999-9999"
})
```

9. Insertar un documento con un array `metodos_pago`:

```
db.clientes.insertOne({
  nombre: "Claudia Romero",
  metodos_pago: [
    { tipo: "tarjeta_credito", numero: "**** **** **** 5678",
caducidad: "11/25" },
    { tipo: "PayPal", correo: "claudia.romero@paypal.com" }
  ]
})
```

10. Insertar con `contacto_emergencia` como subdocumento:

```
db.clientes.insertOne({
  nombre: "Héctor Ríos",
  contacto_emergencia: {
    nombre: "Carolina Ríos",
    telefono: "+52-1-777-7777",
    relacion: "hermana"
  }
})
```

11. Insertar con múltiples accesos en `historial_acceso`:

```
db.clientes.insertOne({
  nombre: "Erika Méndez",
  historial_acceso: [
    { fecha: ISODate("2024-04-01T12:00:00Z"), dispositivo: "Laptop" },
    { fecha: ISODate("2024-04-02T08:00:00Z"), dispositivo: "Tablet" }
  ]
})
```

12. Insertar con `estatus` igual a `"inactivo"`:

```
db.clientes.insertOne({
  nombre: "Tomás García",
  estatus: "inactivo"
})
```

13. Insertar documento con campo adicional no esperado:

```
db.clientes.insertOne({
  nombre: "Sofía Ramírez",
  email: "sofia.ramirez@example.com",
  notas: "Cliente VIP, requiere seguimiento mensual"
})
```

14. Insertar documento con valores numéricos personalizados:

```
db.clientes.insertOne({
  nombre: "Pablo Navarro",
  puntos_fidelidad: 999
})
```

15. Insertar con estructura anidada personalizada:

```
db.clientes.insertOne({
  nombre: "Rosa Martínez",
  direccion: {
    calle: "Av. Reforma",
    ciudad: "CDMX",
    detalles: {
      edificio: "Torre Norte",
      piso: 5
    }
  }
})
```

16. Insertar usando `ObjectId` generado manualmente:

```
db.clientes.insertOne({
  _id: ObjectId("661e1fe3e7d07b4f4d5f0001"),
  nombre: "Mario Díaz"
})
```

17. Insertar con `null` en campos opcionales:

```
db.clientes.insertOne({
  nombre: "Andrea Silva",
  email: null,
  telefono: null
})
```

18. Insertar con campo `telefono` en formato internacional:

```
db.clientes.insertOne({
  nombre: "Juan Valdez",
  telefono: "+1-305-555-1212"
})
```

19. Insertar utilizando campo booleano explícito:

```
db.clientes.insertOne({
  nombre: "Carmen Soto",
  preferencias: {
    notificaciones: false
  }
})
```

20. Insertar documento mínimo válido:

```
db.clientes.insertOne({
  nombre: "Usuario Anónimo"
})
```

Evaluación: Ejercicios sobre CREATE en MongoDB

Instrucciones para los estudiantes

1. Crea un archivo JavaScript llamado: `evaluacion_create.js`
2. Dentro del archivo, debes usar el siguiente bloque inicial para conectarte y seleccionar la base de datos y colección:

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

3. Debes completar los siguientes **10 ejercicios incompletos** de inserción. Todos utilizan coleccion.insertOne() o coleccion.insertMany().
4. Una vez completado el archivo, puedes ejecutarlo en MongoDB Shell (o mongosh) con: `mongosh < evaluacion_create.js`

Ejercicios Incompletos (debes completar los campos faltantes)

1. Insertar cliente con nombre, email y fecha_registro como fecha ISO

```
coleccion.insertOne({
  nombre: "Miguel Ángel",
  email: "miguel.angel@example.com",
  fecha_registro: ISODate(/* completa aquí */)
});
```

2. Insertar cliente con estructura anidada para dirección

```
coleccion.insertOne({
  nombre: "Isabel Ruiz",
  direccion: {
    calle: "Calle 50",
    ciudad: /* completa aquí */,
    codigoPostal: /* completa aquí */
  }
});
```

3. Insertar un documento con múltiples métodos de pago (estructura de array de objetos)

```
coleccion.insertOne({
  nombre: "Tomás Fernández",
  metodos_pago: [
    {
      tipo: "tarjeta_credito",
      numero: "**** **** **** 1111",
      caducidad: /* completa aquí */
    },
    {
      tipo: "PayPal",
      correo: /* completa aquí */
    }
  ]
});
```

4. Insertar un cliente con preferencias anidadas y dos categorías favoritas

```
coleccion.insertOne({
 nombre: "Luciana Gómez",
 preferencias: {
  categorias_favoritas: [/* completa aquí */],
  notificaciones: /* true o false */
 }
});
```

5. Insertar con historial de acceso: dos entradas con fecha y dispositivo

```
coleccion.insertOne({
 nombre: "Andrés Mejía",
 historial_acceso: [
  { fecha: ISODate("2024-04-01T10:00:00Z"), dispositivo: "Laptop" },
  { /* completa aquí */ }
 ]
});
```

6. Insertar varios clientes a la vez (insertMany) con solo nombre y email

```
coleccion.insertMany([
 { nombre: "María Ortega", email: /* completa aquí */ },
 { nombre: /* completa aquí */, email: "carlos.soto@example.com" }
]);
```

7. Insertar con contacto de emergencia completo

```
coleccion.insertOne({
 nombre: "Natalia Prado",
 contacto_emergencia: {
  nombre: "Raúl Prado",
  telefono: "+52-1-800-1234",
  relacion: /* completa aquí */
 }
});
```

8. Insertar un documento con campos numéricos y booleanos

```
coleccion.insertOne({
 nombre: "Elena Torres",
 puntos_fidelidad: /* número */,
 preferencias: {
  notificaciones: /* true o false */
 }
});
```

9. Insertar cliente con campo extra personalizado (por ejemplo, comentarios)

```
coleccion.insertOne({
 nombre: "Gabriel López",
 email: "gabriel.lopez@example.com",
 comentarios: /* escribe un texto personalizado */
});
```

10. Insertar un cliente con _id definido por el usuario

```
coleccion.insertOne({
 _id: "CLTEST001",
 nombre: "Verónica Salinas",
 email: /* completa aquí */
});
```

Revisión

Una vez completado el fichero, el profesor o tutor puede ejecutarlo con:

```
mongosh < evaluacion_create.js
```

Y luego revisar los documentos insertados con:

```
use LabTestDB;
db.clientes_test.find().pretty();
```