Uso de $jsonSchema en MongoDB

En MongoDB, el uso de esquemas se logra a través de **validadores** que permiten establecer reglas sobre los documentos que se pueden almacenar en una colección. Esto se implementa utilizando la opción $jsonSchema.

A continuación, te presento un ejemplo completo de cómo definir y usar un esquema en MongoDB para la colección clientes.

Definir un esquema para la colección clientes

El siguiente esquema valida que:

1. El campo _id debe ser una cadena (string).
2. El campo nombre debe ser obligatorio y de tipo cadena.
3. El campo direccion es un objeto que contiene:
   - calle: Cadena obligatoria.
   - ciudad: Cadena obligatoria.
   - codigoPostal: Cadena opcional.
4. El campo email debe ser único y tener formato de correo electrónico.
5. El campo puntos_fidelidad debe ser un número entero mayor o igual a 0.
6. El campo metodos_pago debe ser un arreglo donde cada elemento sea un objeto que contenga tipo y numero.

Crear la colección con un esquema:

```
db.createCollection("clientes2", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "nombre", "direccion", "email", "puntos_fidelidad"],
      properties: {
        _id: {
          bsonType: "string",
          description: "El ID debe ser una cadena y es obligatorio"
        },
        nombre: {
          bsonType: "string",
          description: "El nombre es obligatorio y debe ser una cadena"
        },
        direccion: {
          bsonType: "object",
          required: ["calle", "ciudad"],
          properties: {
            calle: {
              bsonType: "string",
              description: "La calle es obligatoria y debe ser una cadena"
            },
            ciudad: {
              bsonType: "string",
              description: "La ciudad es obligatoria y debe ser una cadena"
            },
            codigoPostal: {
              bsonType: "string",
```

```
            description: "El código postal es opcional y debe ser una
cadena"
          }
        }
      },
      email: {
        bsonType: "string",
        pattern: "^.+@.+$",
        description: "El correo es obligatorio y debe tener formato válido"
      },
      puntos_fidelidad: {
        bsonType: "int",
        minimum: 0,
        description: "Los puntos de fidelidad deben ser un entero mayor o
igual a 0"
      },
      metodos_pago: {
        bsonType: "array",
        items: {
          bsonType: "object",
          required: ["tipo", "numero"],
          properties: {
            tipo: {
              bsonType: "string",
              description: "El tipo de pago debe ser una cadena"
            },
            numero: {
              bsonType: "string",
              description: "El número de pago debe ser una cadena"
            }
          }
        },
        description: "Debe ser un arreglo de objetos de métodos de pago"
      }
    }
  }
  }
})
```

Insertar documentos en la colección validada

Ejemplo válido:

```
db.clientes2.insertOne({
  _id: "CL1001",
  nombre: "Carlos Pérez",
  direccion: {
    calle: "Calle 50",
    ciudad: "Monterrey",
    codigoPostal: "64000"
  },
  email: "carlos.perez@example.com",
  puntos_fidelidad: 500,
  metodos_pago: [
    { tipo: "tarjeta_credito", numero: "**** **** **** 1234" }
  ]
})
```

Ejemplo inválido: Si intentas insertar un documento sin un campo obligatorio o con un tipo incorrecto, el validador lo rechazará:

```
db.clientes2.insertOne({
  _id: "CL1002",
  nombre: "María López",
  direccion: {
    calle: "Calle 80",
    ciudad: "Guadalajara"
  },
  email: "correo_invalido", // Error: no cumple con el formato de correo
  puntos_fidelidad: -50 // Error: debe ser >= 0
})
```

Resultado: MongoServerError: Document failed validation

Actualizar el esquema de validación

Si necesitas modificar el esquema más tarde, puedes usar el comando collMod:

```
db.runCommand({
  collMod: "clientes",
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["_id", "nombre", "direccion", "email", "puntos_fidelidad"],
      properties: {
        _id: { bsonType: "string" },
        nombre: { bsonType: "string" },
        direccion: {
          bsonType: "object",
          required: ["calle", "ciudad"],
          properties: {
            calle: { bsonType: "string" },
            ciudad: { bsonType: "string" },
            codigoPostal: { bsonType: "string" }
          }
        },
        email: { bsonType: "string", pattern: "^.+@.+$" },
        puntos_fidelidad: { bsonType: "int", minimum: 0 }
      }
    }
  },
  validationLevel: "moderate" // Cambiar a "moderate" o "strict"
})
```

Beneficios del uso de esquemas:

1. **Validación automática:** Previene datos inconsistentes.
2. **Definición clara:** Mejora la comprensión de la estructura de los datos.
3. **Control flexible:** Puedes aplicar reglas estrictas o moderadas según tus necesidades.