



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL FORMATO GUÍA DE APRENDIZAJE – MATERIAL DE APOYO

VALIDACIÓN DE ESQUEMAS Y DOCUMENTOS

Reglas de validación del esquema de MongoDB

Para conseguir que todos los documentos cumplan con los campos anteriores, utilizaremos un esquema específico de **MongoDB**. Se puede pensar que un esquema MongoDB no es más que un conjunto de reglas para las propiedades (Keys) y valores de los documentos. Dichas reglas funcionan en función de cada colección. Las reglas deben seguirse (=validated) durante la inserción o actualización de cada documento en la colección específica. Este conjunto de reglas debe definirse mediante un archivo JSON de acuerdo con los estándares BSON.

La validación de esquemas y documentos permite crear criterios que deben cumplir los documentos para poder ser introducidos o actualizados en una colección. Para llevarlo a cabo se utilizan validadores con los que describimos pautas que deben cumplir los documentos en lenguaje MQL como si se tratase del criterio de una búsqueda o actualización en MongoDB.

Para la validación del esquema se utiliza el operador **\$jsonChema**, como se muestra en el ejemplo

```
{
  $jsonSchema: {
    required: ['customer'], // the customer field is required
    properties: {
      purchaseMethod: {
        enum: ['In Store','Online'],
        description: "can only be either 'In Store' or 'Online'"
      }
    }
  }
}
```



El operador **\$jsonSchema** admite varias palabras clave para especificar reglas de validación. Por ejemplo:

- El arreglo **required** define los campos obligatorios en su documento.
- El objeto **properties** define reglas para campos específicos del documento.

Considere el siguiente ejemplo de validación:

```
{
  $jsonSchema: {
    bsonType: "object",
    required: [ "name", "year", "major", "gpa", "address.city", "address.street" ],
    properties: {
      name: {
        bsonType: "string",
        description: "must be a string"
      },
      year: {
        bsonType: "int",
        minimum: 2017,
        maximum: 3017,
        exclusiveMaximum: false,
        description: "must be an integer in [ 2017, 3017 ]"
      },
      major: {
        bsonType: "string",
        enum: [ "Math", "English", "Computer Science", "History", null ],
        description: "can only be one of the enum values"
      },
      gpa: {
        bsonType: [ "double" ],
        minimum: 0,
        description: "must be a double"
      }
    }
  }
}
```



De acuerdo al ejemplo anterior encontramos las siguientes reglas:

- Lista de campos obligatorios
- El bsonType para todos los campos
- Los valores mínimo y máximo para el año
- Los valores aceptables para la especialidad (**major**), utilizando **enum**.
- El valor mínimo para el campo de promedio general (**gpa**)

Validación usando operadores de consulta

También se puede especificar la validación mediante operadores de consulta, con la excepción de los siguientes operadores de consulta: \$near, \$nearSphere, \$text y \$where.

```
{
  $or: [
    { telefono: { $type: "string" } },
    { correo: { $regex: /@mongodb\.com$/ } },
    { estado: { $in: [ "desconocido", "Incompleto" ] } }
  ]
}
```

Para utilizar esta validación, debe cumplirse **una de las siguientes** condiciones:

- El campo de teléfono debe ser una cadena de tipo BSON,
- El campo de correo electrónico debe coincidir con la expresión regular [/@mongodb\.com\\$/](#), o
- El campo de estado debe ser desconocido o incompleto



Acciones y niveles de validación

En la parte superior, especifique una acción de validación y un nivel de validación:

- La **acción de validación** determina si se debe advertir pero aceptar documentos no válidos, o generar un error y rechazar documentos no válidos.
 - **validateAction**: puede ser 'error' o 'warn'
- El **nivel de validación** determina con qué rigurosidad MongoDB aplica las reglas de validación a los documentos existentes.
 - La validación estricta aplica sus reglas a todas las inserciones y actualizaciones de documentos.
 - La validación moderada solo aplica sus reglas a los documentos nuevos y a los documentos válidos existentes. Los documentos no válidos existentes no se ven afectados.
 - **validateLevel**: puede ser 'moderate' o 'strict'

Para obtener más información sobre las acciones y los niveles de validación, consulte Especificar reglas de validación en el manual de MongoDB.

De acuerdo con lo anterior vamos a realizar el siguiente ejemplo en **MongoDB Compass**.

Desde la consola monosh realizar lo siguiente:

- Crear una base de datos llamada **TIENDAADSO**
- Crear la colección **PRODUCTOS** con los siguientes atributos y reglas.
 - **ATRIBUTOS**
 - **Código**: de tipo entero, valor mínimo 1000
 - **Nombre**: de tipo string
 - **Categoría**: de tipo string, con solo las siguientes posibilidades:
Electrodomesticos, Ropa, Calzado.
 - **Precio**: de tipo double, valor mínimo 0.
- Todos los **atributos** son **obligatorios**
- A todos los **atributos** colocarles una **descripción**.



Solución:

1. Crear la base de datos

```
>_MONGOSH  
  
> use TIENDAADSO  
< switched to db TIENDAADSO  
TIENDAADSO> |
```

2. Crear el Json de la creación de la colección

```
db.createCollection("PRODUCTOS",{  
  validator: {  
    $jsonSchema: {  
      bsonType: "object",  
      required: [ "codigo", "nombre", "categoria", "precio" ],  
      properties: {  
        codigo: {  
          bsonType: "int",  
          minimum: 1000,  
          description: "Código que identifica el producto"  
        },  
        nombre: {  
          bsonType: "string",  
          description: "Nombre del producto ]"  
        },  
        categoria: {  
          bsonType: "string",  
          enum: [ "Electrodomestico", "Ropa", "Calzado" ],  
          description: "solo puede ser una de esas categorías"  
        },  
        precio: {  
          bsonType: "int",  
          minimum: 0,  
          description: "Precio del producto tipo entero"  
        }  
      }  
    }  
  }  
})
```




3. En la consola ejecutar la operación de crear la colección:

```
>_MONGOSH
P
< switched to db TIENDAADSO
TIENDAADSO> db.createCollection("PRODUCTOS",{
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "codigo", "nombre", "categoria", "precio" ],
      properties: {
        codigo: {
          bsonType: "int",
          minimum: 1000,
          description: "Código que identifica el producto"
        },
        nombre: {
          bsonType: "string",
          description: "Nombre del producto "]"
        },
        categoria: {
          bsonType: "string",
          enum: [ "Electrodomestico", "Ropa", "Calzado" ],
          description: "solo puede ser una de esas categorías"
        },
        precio: {
          bsonType: "int",
          minimum: 0,
          description: "Precio del producto tipo entero"
        }
      }
    }
  }
})
```



4. Respuesta del servidor:

```
>_MONGOSH
> db.createCollection("PRODUCTOS",{
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: [ "codigo", "nombre", "categoria", "precio" ],
      properties: {
        codigo: {
          bsonType: "int",
          minimum: 1000,
          description: "Código que identifica el producto"
        },
        nombre: {
          bsonType: "string",
          description: "Nombre del producto"
        },
        categoria: {
          bsonType: "string",
          enum: [ "Electrodomestico", "Ropa", "Calzado" ],
          description: "solo puede ser una de esas categorías"
        },
        precio: {
          bsonType: "int",
          minimum: 0,
          description: "Precio del producto tipo entero"
        }
      }
    }
  }
})
< { ok: 1 }
```

A large green arrow points from the right towards the response line "< { ok: 1 }" at the bottom of the terminal output.



5. Realizar pruebas haciendo inserción de documentos

Insertar un documento vacío:

```
> db.PRODUCTOS.insertOne({})  
✖ ▶ MongoServerError: Document failed validation  
TIENDAADS0>
```

Genera error. Todos los atributos son requeridos.

Insertar otro documento con campo que no cumple la regla de validación.

```
{  
  'codigo': 500,  
  'nombre': 'Televisor',  
  'categoría': 'Electrodomestico',  
  'precio': 2500000  
}
```

El producto anterior tiene código de 500 y debe ser como mínimo 1000.

```
TIENDAADS0> db.PRODUCTOS.insertOne({  
  'codigo': 500,  
  'nombre': 'Televisor',  
  'categoría': 'Electrodomestico',  
  'precio': 2500000  
})
```




El servidor responde informando error validación

```
> db.PRODUCTOS.insertOne({
  'codigo': 500,
  'nombre': 'Televisor',
  'categoria': 'Electrodomestico',
  'precio': 2500000
})
```

❌ ▶ **MongoServerError:** Document failed validation

Ahora cambiemos el código por un valor valido

```
> db.PRODUCTOS.insertOne({
  'codigo': 1500,
  'nombre': 'Televisor',
  'categoria': 'Electrodomestico',
  'precio': 2500000
}
)
< {
  acknowledged: true,
  insertedId: ObjectId('66993747a68393a9568af1ea')
}
TIENDAADSO >
```



Ahora vamos a intentar insertar un producto con una categoría no posible así:

```
> db.PRODUCTOS.insertOne({
  'codigo': 1501,
  'nombre': 'Nevera',
  'categoria': 'Comida',
  'precio': 3150000
})
```

✖ ▶ **MongoServerError:** Document failed validation

Cambiamos categoría a Electrodomestico y debe permitir hacer la inserción.

```
> db.PRODUCTOS.insertOne({
  'codigo': 1501,
  'nombre': 'Nevera',
  'categoria': 'Electrodomestico',
  'precio': 3150000
})
< {
  acknowledged: true,
  insertedId: ObjectId('6699385aa68393a9568af1ec')
}
TIENDAADSO> |
```



Consulta de los productos registrados

```
> db.PRODUCTOS.find()
< [
  {
    _id: ObjectId('66993747a68393a9568af1ea'),
    codigo: 1500,
    nombre: 'Televisor',
    categoria: 'Electrodomestico',
    precio: 2500000
  },
  {
    _id: ObjectId('6699385aa68393a9568af1ec'),
    codigo: 1501,
    nombre: 'Nevera',
    categoria: 'Electrodomestico',
    precio: 3150000
  }
]
TIENDAADS0> |
```

Desde la consola podemos revisar información de las reglas de validación de la colección **PRODUCTOS**.

```
> db.getCollectionInfos()
< [
  {
    name: 'PRODUCTOS',
    type: 'collection',
    options: { validator: [Object] },
    info: {
      readOnly: false,
      uuid: UUID('5e488d4e-5c75-49fb-8bed-4fc5de760a31')
    },
    idIndex: { v: 2, key: [Object], name: '_id_' }
  }
]
TIENDAADS0>
```



```
> db.getCollectionInfos({name:"PRODUCTOS"})[0].options.validator
< {
  '$jsonSchema': {
    bsonType: 'object',
    required: [ 'codigo', 'nombre', 'categoria', 'precio' ],
    properties: {
      codigo: [Object],
      nombre: [Object],
      categoria: [Object],
      precio: [Object]
    }
  }
}
```

También se puede consultar las reglas de validación desde la interfaz gráfica. Debe seleccionar la colección y en la parte principal donde se muestran los datos dar clic en la opción validation como lo muestra la siguiente imagen:

The screenshot shows the MongoDB Compass interface. At the top, there are tabs for Documents (2), Aggregations, Schema, Indexes (1), and Validation. A green arrow points to the Validation tab. Below the tabs, there is a search bar with the text 'Type a query: { field: 'value' } or [Generate query](#)'. To the right of the search bar are buttons for Explain, Reset, Find, and Options. Below the search bar, there are buttons for ADD DATA, EXPORT DATA, UPDATE, and DELETE. On the right side, there is a status bar showing '1 - 2 of 2' and navigation icons. The main area displays two documents with their validation rules:

```
_id: ObjectId('66993747a68393a9568af1ea')
codigo: 1500
nombre: "Televisor"
categoria: "Electrodomestico"
precio: 2500000
```

```
_id: ObjectId('6699385aa68393a9568af1ec')
codigo: 1501
nombre: "Nevera"
categoria: "Electrodomestico"
precio: 3150000
```



Después aparece la información de las reglas de validación de la colección así:

localhost:27017 > TIENDAADSO > PRODUCTOS

Documents 2

Aggregations

Schema

Indexes 1

Validation

Validation Action ⓘ

Error

Validation Level ⓘ

Strict

```
1 ▼ {
2 ▼   $jsonSchema: {
3     bsonType: 'object',
4     required: [
5       'codigo',
6       'nombre',
7       'categoria',
8       'precio'
9     ],
10    properties: {
11      codigo: {
12        bsonType: 'int',
13        minimum: 1000,
14        description: 'Código que identifica el producto'
15      },
16      nombre: {
17        bsonType: 'string',
18        description: 'Nombre del producto ]'
19      },
20      categoria: {
21        bsonType: 'string',
22        'enum': [
23          'Electrodomestico',
24          'Ropa',
25          'Calzado'
26        ],
27        description: 'solo puede ser una de esas categorías'
28      },
29      precio: {
30        bsonType: 'int',
31        minimum: 0,
32        description: 'Precio del producto tipo entero'
33      }
34    }
35  }
36 }
```

Desde la interfaz anterior se pueden hacer cambios a las reglas de validación y la interfaz cuando usted realiza un cambio se habilita un botón para actualizar.



Referencias

- **Reglas de validación del esquema de MongoDB:** <https://blog.itechnova.com/reglas-de-validacion-del-esquema-de-mongodb/>
- **Validación Nativa en MongoDB:** <https://www.paradigmadigital.com/dev/validacion-nativa-documentos-mongodb-3-2/>
- **Validación de esquemas:** <https://www.youtube.com/watch?v=t-Z3xIcrMgg>
- <https://docs.mongodb.com/compass/current/validation/>
- <https://docs.mongodb.com/manual/core/schema-validation/#schema-validation>
- <https://docs.mongodb.com/manual/core/schema-validation/#specify-validation-rules>
- <https://docs.mongodb.com/manual/reference/command/collMod/#add-document-validation-to-an-existing-collection>
- <https://docs.mongodb.com/manual/reference/bson-types/>
- <https://www.mongodb.com/docs/manual/reference/operator/query/jsonSchema/>

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	César Marino Cuéllar Chacón	Instructor	CTPI-CAUCA	18-07-2024