VALIDACIÓN DE DATOS DE LECTURA Y ESCRITURA EN BASES DE DATOS NOSQL USANDO MONGOOSE

Severino Feliciano Morales
Facultad de Ingeniería, Universidad Autónoma de Guerrero, Chilpancingo, Guerrero, México, 09552@uagro.mx

Johan Ramón Tellez Alvarez
Facultad de Ingeniería, Universidad Autónoma de Guerrero, Chilpancingo, Guerrero, México, 13452712@uagro.mx

Edgardo Solís Carmona
Facultad de Ingeniería, Universidad Autónoma de Guerrero, Chilpancingo, Guerrero, México, 09552@uagro.mx

Valentín Álvarez Hilario
Facultad de Ingeniería, Universidad Autónoma de Guerrero, Chilpancingo, Guerrero, México, 09552@uagro.mx

RESUMEN
La web 2.0 y las tecnologías modernas como las redes sociales, internet de las cosas y big data, han revolucionado la forma de organizar los datos en las bases de datos, ya que hoy en día los datos tienen una estructura más flexible, es decir, los datos son semiestructurados y no estructurados como se han venido manejando, es decir, que no tienen un esquema explícito (schemaless), sino que viene implícito en los datos o en el código. Esta característica permite que haya incluso campos opcionales. Esta forma de gestionar los datos, se hace a través de bases de datos NoSQL, donde los datos se pueden organizar mediante documentos, clave – valor, familia de columnas o grafos, las cuales pueden ser gestionadas de manera nativa o mediante una API. Sin embargo, esta flexibilidad hace más complicada la validación estática al leer y escribir los datos en la base de datos, por lo que se han desarrollado herramientas que permiten esta validación, definiendo un esquema, el cual asegura que los datos que son necesarios sean requeridos. Para este caso, se usó MongoDB y la herramienta Mongoose para definir el esquema.

PALABRAS CLAVE
MongoDB, NoSQL, Mongoose.

ABSTRACT
Web 2.0 and modern technologies, e. g., social media, internet of things, and big data have revolutionized how to organize data in databases, since nowadays the data has a more flexible structure, that is, the data is semi-structured and not structured as they have been handled, that is, they do not have an explicit schema (schemaless), but rather it is implicit in the data or in the code. This feature allows for even optional fields. This way of managing data is done through NoSQL databases, where data can be organized by document, wide column, key-value stores, and graph-based databases, which can be managed natively or through an API. However, this flexibility makes static validation more complicated when reading and writing the data in the database, hence tools have been developed that allow this validation trough schema defining, which ensures that necessary data is required. For this case, MongoDB and the Mongoose tool were used to define the schema.

KEYWORDS
MongoDB, NoSQL, Mongoose.

INTRODUCCIÓN
El manejo de la información cada vez se ha hecho más compleja con el desarrollo de nuevas tecnologías y grandes compañías que manejan altos volúmenes de información. Estas compañías se dieron cuenta que con la infraestructura que contaban no podrían manejar la cantidad de información que se proyecta para los próximos años, por eso, están invirtiendo y creando soluciones *NoSQL* [1]. Muchas empresas tanto medianas como grandes han optado por usar bases de datos NoSQL, ya que estas permiten manejar con más eficacia y rapidez una mayor cantidad de datos.

Las bases de datos NoSQL han ganado popularidad, ya que estas tienen usos específicos en la solución de problemas actuales, tales como almacenamiento de datos, análisis de datos, comercios electrónicos, aplicaciones web, redes sociales y desarrollo de aplicaciones móviles [3].
El termino NoSQL suele ser referido a un grupo diverso de paradigmas de modelado de datos que tienden a gestionar datos no estructurados [2]. Por ende, dentro de una base NoSQL, los datos son almacenados sin tener la necesidad de haber definido un esquema con anterioridad. Al no requerir un esquema, las bases de datos NoSQL ofrecen mucha más flexibilidad para el manejo de los datos. Son por naturaleza escalables, propiedad que poseen para poder reaccionar y afrontar el crecimiento sin fallar en los servicios que proporciona. Adicionalmente, estas bases de datos tienen la capacidad de ingresar datos con el mismo tipo de entidad, aunque sus registros tengan estructuras diferentes al contario de una base SQL la cual requiere establecer la estructura de los datos con anterioridad.
En la figura 1, se muestra la diferencia de la organización de los datos entre las bases de datos SQL y las bases de datos NoSQL.

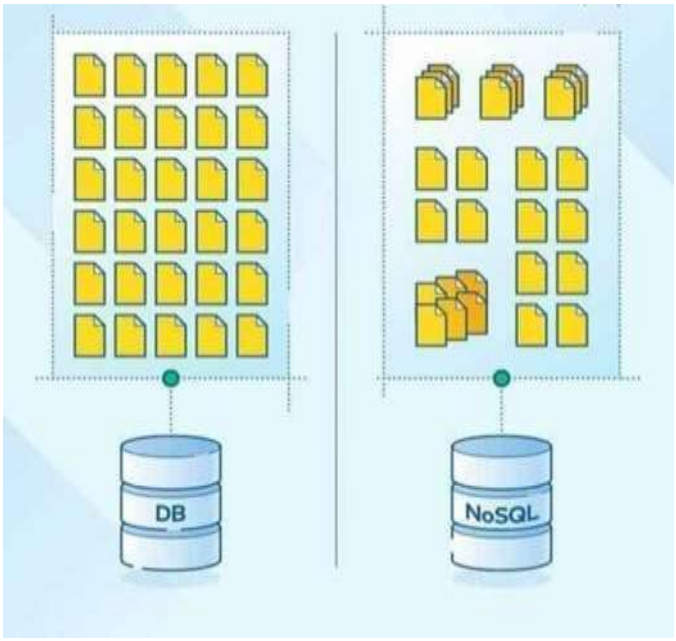

Figura 1- Estructuras de SQL y NoSQL [15].

El hecho de que no tenga un esquema de datos explícito convierte a NoSQL en una propuesta bastante atractiva para los desarrolladores de bases de datos. Sin embargo, el hecho de que se no haya necesidad de declarar esquemas no tiene que confundirse con la idea de que los esquemas aquí no existen en su totalidad, ya que éstos están implícitos en los datos almacenados y en el código de las aplicaciones. Por ende, los desarrolladores siempre deben tomar en cuenta un esquema implícito cuando se desarrolla una base de datos con el propósito de acceder a los datos almacenados. Para ello, cuando se escriben operaciones *CRUD* (Create, Read, Update and Delete), se suelen respetar los nombres y los tipos de los campos correspondientes. Al tener la idea de la existencia de un esquema, es posible tener un mejor control de errores. Por otro lado, varias herramientas para bases de datos requieren conocer el esquema para ofrecer varias funciones como realizar consultas, migrar datos de manera automática o bien procesar datos de manera eficiente.
Además, las bases de datos NoSQL permiten almacenar información en momentos en las que las bases de datos relacionales generan ciertos problemas, como problemas de escalabilidad y rendimiento, un ejemplo son las APIs de servicios web donde miles de usuarios realizan millones de consultas a toda hora.

Con el transcurso del tiempo, las bases de datos NoSQL han estado en una constante evolución, lo que ha ocasionado que se desarrollen 4 diferentes tipos de bases de datos NoSQL (figura 2), las cuales tienen su manera distintiva de organizar la información y de este modo satisfacer las necesidades que se requieran.

- ***Bases de datos de tipo Clave-Valor***: Los valores se almacenan dentro de una estructura muy similar a una tabla en donde la primera columna se asigna la clave *id* de los registros, y en la segunda columna los datos ingresados.

- ***Bases de datos de tipo Documentales:*** En este tipo de estructura, las claves y valores de los datos se modelan a través de Documentos y se escriben en JSON. Dentro de este tipo de bases de datos no es necesario que los datos guardados en Documentos sigan el mismo tipo de estructura entre ellos, por lo que se dice que otorga flexibilidad.
- ***Bases de datos de tipo Orientadas a Columnas:*** En este tipo de estructura los datos son insertados en tablas. A diferencia de las del tipo Clave-Valor, este tipo de bases de datos tienen una organización que se basa en crear grupos de familias de columnas. Se considera que la primera columna contiene la clave con la que se identifica la familia de columnas que le siguen.
- ***Bases de datos de tipo Grafo:*** Este tipo de estructura almacena los datos en nodos, y estos mismos se conectan por aristas las cuales contienen los atributos que permiten relacionarse entre sí.

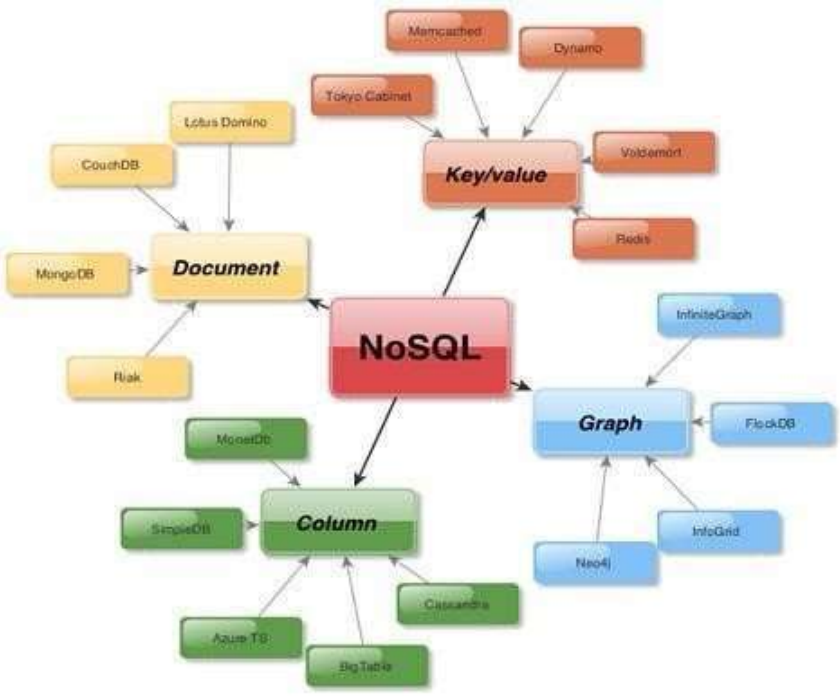

Figura 2- Tipos de bases NoSQL [12].

1 MONGODB.

MongoDB es un gestor NoSQL el cual permite manejar bases de datos no relacionales, está desarrollada de tal manera que nos facilita el trabajo de desarrollo y escalado de las mismas.

Este gestor de bases de datos es usado en aplicaciones famosas como Foursquare, Sourceforge, Google, Facebook o New York Times, entre otros [6].

El nombre *Mongo* proviene de la palabra *humongous* que significa gigante, haciendo semejanza a las grandes cantidades de datos con las que puede trabajar. Fue desarrollado bajo el concepto de código abierto por la empresa de software 10gen, comenzando su desarrollo a partir del 2007 y teniendo un lanzamiento al mercado en el año 2009.

MongoDB es una base de datos ágil que permite a los esquemas cambiar rápidamente al paso del que evolucionan las aplicaciones, sin dejar de ofrecer las funcionalidades que los desarrolladores esperan de las bases de datos tradicionales, como realizar operaciones CRUD.

1.1 ESTRUCTURA DE ALMACENAMIENTO DE MONGODB

Dentro de una base de datos NoSQL los valores que son ingresados se les da el nombre de ***Documentos***. Un Documento tiene una estructura que contiene pares de campos y valores. Los documentos permiten realizar diversas funciones sobre los datos como lo son agregar, modificar, eliminar, renombrar, etcétera, y esto se puede realizar en cualquier momento, ya que no hay un esquema predefinido.

Los valores de los campos dentro de los documentos pueden incluir otros documentos y vectores de documentos. Adicionalmente facilita la búsqueda de campos, consultas en ciertos rangos y expresiones regulares.
La estructura de un documento contiene uno o varios pares de campos en forma de pares con nombre y valor. Esta estructura al ser un objeto de JSON, formato de texto que deriva de la sintaxis de JavaScript, inician y terminan con llaves ({}) y su gramática se muestra en la figura 3.

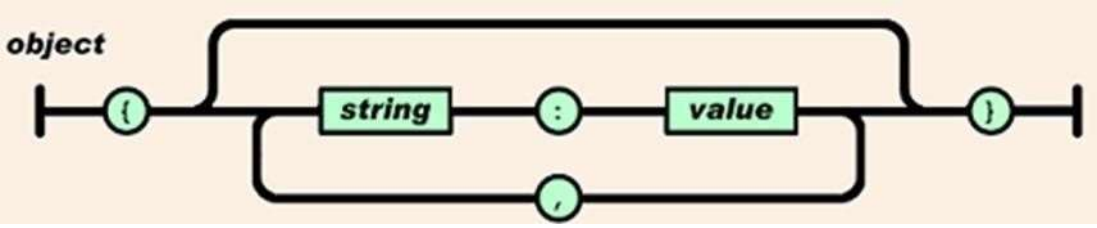

Figura 3- Gramática JSON [13].

Los documentos se encuentran dentro de las estructuras llamadas *Colecciones* (figura 4). Las Colecciones se comportan de manera similar a las tablas que se encuentran en las bases de datos relacionales, ya que estas son las que almacenan a los documentos y lo hacen sin importar si tienen diferente estructura.

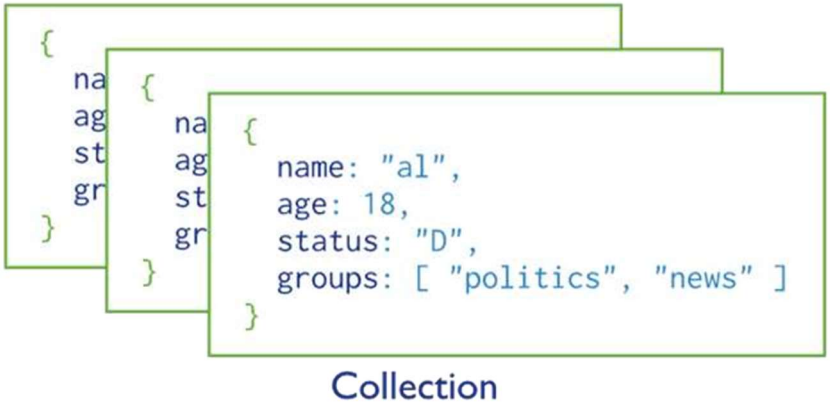

Figura 4- Grupo de Colecciones [8].

Tener varios conjuntos de colecciones que almacenan distintos tipos de documentos hace que se logre conformar una base de datos NoSQL. En la figura 5 se muestra la composición de las bases de datos NoSQL.

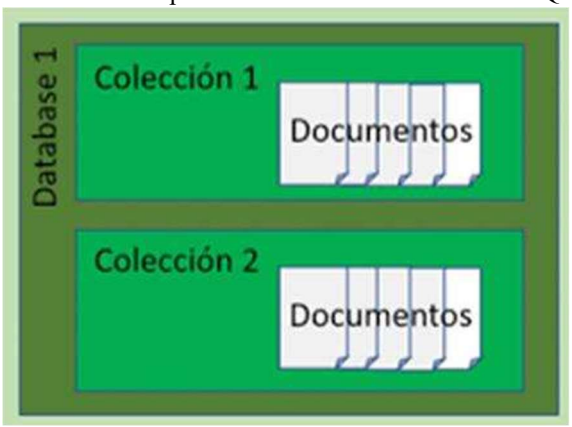

Figura 5- Composición de una base NoSQL [14].

Como prueba de concepto, se desarrolló la base de datos mostrada en la figura 6. Se crearon 3 Colecciones con los nombres de *Publisher, Book y Journal*, y en los documentos ingresados se puede apreciar que poseen información sobre libros, revistas y editoriales, así que la base puede tener como finalidad el almacenamiento de la información acerca de libros y revistas.

Se implementó con el uso de MongoDB, creando la base de datos con la instrucción *use OurDataBase* y seguido se crearon las colecciones junto a los documentos que contienen.

```
[

{
 publisher:
 [
 {
   name: "Academic Press",
   city: [ "Londres", "Oxford", "Boston", "New York", "
San Diego"],
   _id: 98543
 },
 {
  name: "Omega",
  city: "Barcelona",
   _id : 12345
 },
 {
  name: "Limusa",
  city: "México",
   _id : 12377
 },
 {
   _id : 34567,
   name: "Addison -Wesley "
 },
 {
   _id : 67892,
   name: "Mc Graw Hill"
 },
 {
   name: "Institute of Electrical and Electronics
          Engineers publications",
  year  : 1975,
  journals: [1,2],
  field  : "electrical and electronics engineering and
          computer science",
   _id: 90786
 } ] }
```

```
{
 book:
 [
  {
   _id      :1,
   title     : "El Lenguaje Unificado de Modelado",
   publisher_id: " Addison -Wesley ",
   year      : 2013,
   authors:[
        {
         _id      :1,
         name     :"Grady Booch",
         company:{name:"IBM", country:"USA"}
        },
        {
         _id      :2,
         name     :"James Rumbaugh",
         company:{name:"IBM", country:"USA"}
        },
        {
         _id       :3,
         name     :"Ivar Jacobson",
         country  :"USA"
        }
        ],
   content:{pages: 527, chapters:33}
  },
  {
   "_id" : 2,
   "title" : "MongoDB: The Definitive Guide",
   "authors" : [ "Kristina Chodorow", "Mike Dirolf" ],
   "published_date" :
         { "$date" : "2010-09-T02:00:00.000+0200" },
   "pages" : 216
  }
 ]
}
```

```
{
 "journal":
 [
 {
  name: "IEEE Transactions on Software Engineering ",
  issn: ["0098-5589","1939-3520"],
  discipline:"software engineering",
  _id : 1
 },
 {
  name:"IEEE Transactions on Wireless
        Communications",
  _id : 2
 },
 {
  _id : 3,
  name: "Automated Software Engineering",
  issn: ["0928-8910","1573-7535"],
  discipline:"software engineering",
  number: 10515
 }
 ]
}

]
```

Figura 6- Base de datos NoSQL con las Colecciones Publisher, Book y Journal [2].

Como se puede apreciar en el código, los documentos que se insertan no tienen estructura alguna y estos están contenidos en las Colecciones correspondientes. Para corroborar que las colecciones que se crearon no presentan algún inconveniente se puede utilizar la instrucción *show collections* y seguido MongoDB mostrará las colecciones creadas hasta el momento (figura 7).

```
OurDataBase> show collections
Book
Journal
Publisher
OurDataBase>
```

Figura 7- Visualización de las Colecciones existentes.

Para visualizar una colección en específico y sus documentos contenidos se puede usar la instrucción *find()*, adicionalmente se puede incluir a la instrucción anterior la extensión .p*retty()* para que MongoDB muestre a la Colección de una manera más organizada (figura 8).

Figura 8- Visualización del contenido de una Colección.

2 IMPLEMENTACIÓN DE MONGOOSE

2.1 ODM

Con la ausencia de los esquemas explícitos, existen herramientas que requieren conocerlo para soportar algunas funcionalidades específicas o incluso existen casos donde desarrolladores desean o necesitan usar un esquema para diversos propósitos informáticos, los ***ODM*** (Object Document Mapping) proporcionan la posibilidad de implementar los esquemas. Los ODM son modelos de programación que se encargan de *mapear* bases de datos NoSQL sobre una estructura lógica basada en entidades con la intención de crear un esquema explícito que permita acelerar y simplificar la manipulación de la información. Los ODM mantienen una persistencia entre los datos de una base de datos NoSQL junto a una flexibilidad que simplifican las consultas CRUD, entre otras tareas.

2.2 MONGOOSE

Mongoose es una biblioteca ODM la cual se encarga de crea una conexión entre MongoDB y el entorno de ejecución *Node.js* (figura 9), lo que permite escribir consultas para una base de datos de MongooDB y realizar operaciones como validaciones y construcción de *querie*s, middlewares, conversión de tipos, etc. Todo esto enriquece la funcionalidad de la base de datos en MongoDB [11]. También guarda la información de las validaciones y de los valores por default, y si una propiedad en particular es requerida.

Debido a que Mongoose fue desarrollado para MongoDB, lo convierte en uno de los ODM más utilizados gracias a la popularidad de este sistema. Además, se tiene un sistema de validación y de tipado que puede ser personalizado acorde a las necesidades.

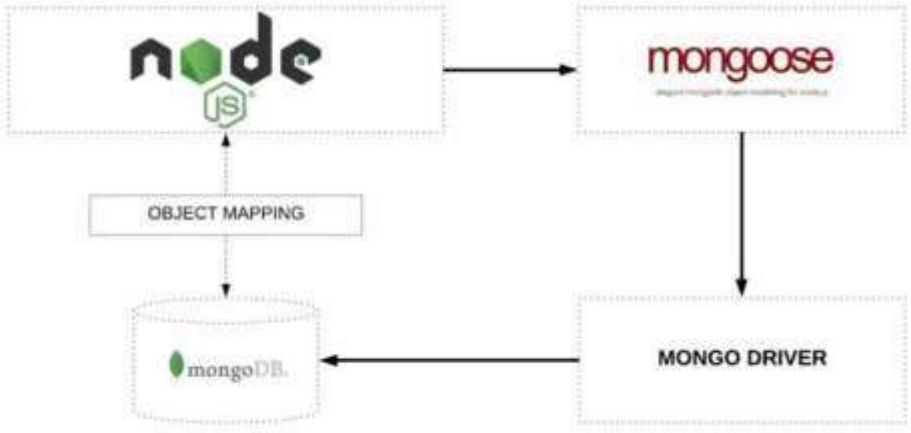

Figura 9- Diagrama de mapeo de objeto con Mongoose [8].

Un uso primordial de Mongoose está en que se pueden definir esquemas que permiten la manipulación de los documentos provenientes de una base de datos NoSQL de MongoDB. El uso de un esquema brindado por Mongoose permite normalizar la información sin perder flexibilidad.

Mongoose hace un mapeo de la información de la base de datos en MongoDB por medio de NodeJS y lo convierte en esquemas. Mongoose permite que las bases de datos de MongoDB sean muy fáciles de modificar y actualizar. Muchos desarrolladores están acostumbrados a tener esquemas, así que Mongoose facilita la creación de uno.

2.3 ESQUEMAS Y MODELOS

Un *Esquema* en Mongoose es una estructura que tiene información acerca de las propiedades de un Documento proveniente de una Colección de una base de datos de MongoDB. Puede tener la información sobre la validación , su contener lógica y valores por default de los Documentos.

En Mongoose se define un esquema y se declaran los Documentos que se encuentran dentro de una Colección específica. Los esquemas sirven para definir la estructura del documento y se pueden entender que son las guías para la estructura de los documentos dentro de Mongoose. Los Esquemas cumplen con representar las instancias de los Documentos que estén dentro de una Colección en una base de datos NoSQL. Adicionalmente, permiten la construcción de documentos a través de un modelo permitiéndo crear métodos estáticos para realizar funciones de una base de datos.
Los *Modelos* son constructores que toman un esquema y crean una instancia de un documento equivalente a los registros en una base de datos relacional. Cuando usas el constructor del modelo, creas un nuevo documento con los nuevos objetos y tipos que se hayan creado en base a la colección en la que se toman los documentos. Los modelos realizados con Mongoose admiten funciones para almacenar, eliminar, consultar y también para la validaciones y procesamientos previos para garantizar la integridad de la base de datos y manejar los errores.

2.4 IMPLEMENTACIÓN DEL ESQUEMA DEFINIDO EN MONGOOSE

Para realizar un proyecto con mongoose es necesario usar el lenguaje de programación *Node.js*. Este último es un entorno controlado por eventos para la capa del servidor y está basado en JavaScript. Node.Js nos permite realizar aplicaciones web escalables permitiendo la gestión de varias conexiones.
Se crea la conexión entre Mongoose y la base de datos creada en el servidor local de MongoDB como se muestra en la figura 10.

```
const url = 'mongodb://127.0.0.1:27017/OurDataBase'
```

Figura 10- Conexión entre Mongoose y MongoDB.

Una vez creada la conexión, se crea el Esquema donde se definen los tipos de datos de las colecciones que se tienen en nuestra base de datos NoSQL.

Se debe declarar con el nombre y el tipo de dato que le corresponde, se crea la estructura de manera similar que una base de datos relacionales. Se debe prestar atención a los valores que no declaremos ya que Mongoose los ignorará. En el Esquema de la figura 11 se utilizan los Documentos de la colección *Book*.

```
const booksSchema = mongoose.Schema({

    title:{
        type: String,
        required: true
    },
    year:{
        type: Number,
        required: true
    },
    publisher_id:{
        type: String,
        required: undefined,
    },
    authors:[{
        name:{
            type: String,
            required: undefined
        },
        company:[{
            name:{
                type: String,
                required: undefined
            },
            country:{
                type: String,
                required: undefined
            }
        }],
        published_date:{
            type: String,
            required: undefined
        }
    }],
    content:[{
        pages:{
            type: Number,
            required: undefined
        },
        chapter:{
            type: Number,
            required: undefined
        }
    }]


});
```

Figura 11- Esquema de la colección "Book".

Posteriormente, se crea el Modelo para la colección Book (figura 12). El Modelo recibe los valores generados del Esquema *Books* y también el nombre de la Colección *Book* de la base de datos de MongoDB con la que se está trabajando respectivamente.

```
const BooksModel = mongoose.model('Book', booksSchema)
```

Figura 12- Creación del Modelo de la Colección Book.

Estando el Esquema y el Modelo ya definidos, se realiza la operación de agregación de un nuevo registro. La figura 13 muestra el ejemplo de una adición de un nuevo documento en la Colección Book usando el constructor del Modelo *Books*.

```
const crear= async()=>{
    const book = new BooksModel({
      title:'Libro de Ciencias',
      year: 1998,
      authors:[{
        name:'Alicia A.',
        company:
        [{
            name: 'SEP',
            country: 'Mexico'
        }]
        }]
    })
    const resultado = await book.save()
    console.log(resultado)
}
```

Figura 13- Insertando datos por medio del Modelo.

Una vez agregados varios registros distintos, se puede realizar la consulta para visualizar los Documentos ingresados. La figura 14 muestra la consulta del registro que se agregó recientemente.

```
[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
Conectado a la base
{
  title: 'Libro de Ciencias',
  year: 1998,
  authors: [
    {
      name: 'Alicia A.',
      company: [Array],
      _id: new ObjectId("63ba4758dfc5377ab3b5398d")
    }
  ],
  _id: new ObjectId("63ba4758dfc5377ab3b5398c"),
  content: [],
  __v: 0
}
```

Figura 14- Visualización de los datos insertados.

Por supuesto, también, por medio de MongoDB se puede corroborar que los Documentos fueron insertados exitosamente, como se muestra en las figuras 15 y 16.

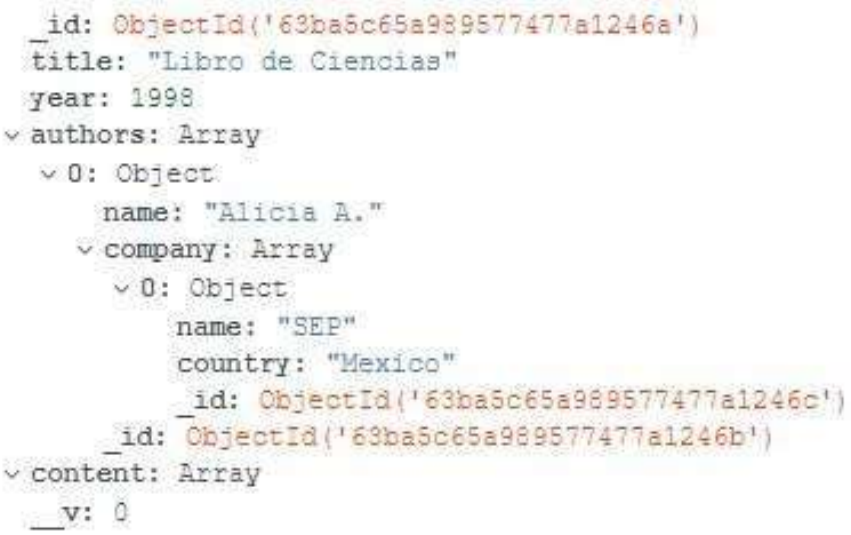

Figura 15- Visualización de los datos en MongoDB.

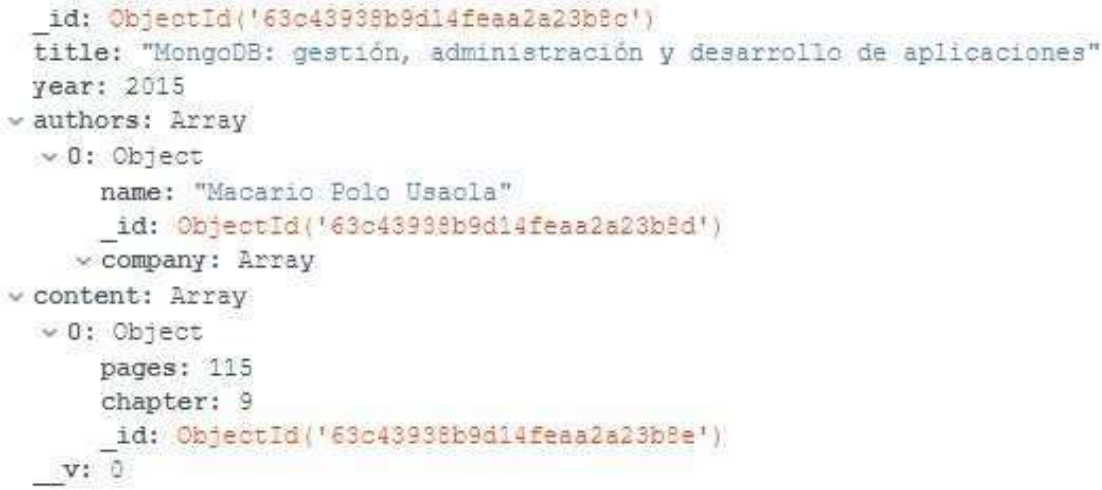

Figura 16- Ejemplo de más datos.

CONCLUSIONES

Con los avances tecnológicos, las bases de datos NoSQL son las mejores opciones para manejar grandes cantidades de datos porque no tienen un esquema explicito lo que brinda una flexibilidad de elección sobre los datos que se proporciones, siendo esto la principal razón de que muchas compañías lo eligen para su uso de almacenamiento de datos, más en estos tiempos de redes sociales donde este tipo de bases de datos le da al usuario la libertad de elegir qué información ingresar a su perfil y eso también significa que el usuario podría omitir ingresar información que podría servir para ser reconocido, como nombre, edad, etcétera.

Para tener un mayor control sobre los datos se crean esquemas explícitos con ayuda de herramientas ODM que se integran a los manejadores de base de datos NoSQL. Al crear el esquema se pueden estructurar los datos asignando campos obligatorios y así tener un mejor control en la información con la que se trabaja.

Elegir a MongoDB como gestor de bases de datos NoSQL resulta ser una de las mejores opciones porque es muy fácil e intuitivo trabajar con él. Implementar Mongoose, una de las herramientas ODM más populares para MongoDB, incrementa aún más su utilidad al permitir la creación de esquemas para un mayor control de información por los que se vuelven en un par inseparable para muchos de los desarrolladores.

REFERENCIAS

[1] Alexander, C. R., Juan Sebastián, G. S., Mauro, C. C. (2012) Utilidad y funcionamiento de las bases de datos NoSQL. Revista Facultad de Ingeniería, Vol. 21.

[2] Feliciano, M. S. (2017). Inferring NoSQL Data Schemas with Model-Driven Engineering Techniques (tesis doctoral). Universidad de Murcia, Murcia, España.

[3] NoSQL-Market(2018). NoSQLMarket. https://www.alliedmarketresearch.com/NoSQL-market. Consultado en Diciembre, 2022.

[4] Robles, D., Sánchez, M., Serrano, R., Adárraga, B., & Vizcaíno, D. H. (2017). ¿Qué características tienen los esquemas NOSQL? Revista Investigación y Desarrollo en TIC.

[5] Shakuntala, G. E., Navin, S. (2015). Practical MongoDB: Architecting, Developing, and Administering MongoDB.

[6] HispaBigData; Base de Datos NoSQL. http:// www.hispabigdata.es/2013/08/bases-dedatos-nosql.html. Consultada en Noviembre, 2022.

[7] Compton, R. J. (2017). Shift, Global Research and Programmer Collaboration (Doctoral dissertation, Appalachian State University).

[8] MongoDB, Inc. (2022). MongoDB Documentation. https://www.mongodb.com/docs/. Consultado en Diciembre 2022.

[9] Jesse, H. (2022). Getting Started with MongoDB & Mongoose. https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/. Consultado en Diciembre 2022.

[10] Vidhi, S. (2022). Mongoose CRUD Operations Simplified. https://hevodata.com/learn/mongoosecrudoperations/#:~:text=Mongoose%20CRUD%20Operations%20MongoDB%20and%20Node.JS%20are%20comm only,abstraction%20failure%20when%20they%20used%20Node.JS%20and%20MongoDB. Consultado en Noviembre, 2022.

[11] Qué es Mongoose. https://codigofacilito.com/articulos/que-es-mongoose. Consultado en Diciembre, 2022

[12] Baas, B. (2012). NoSQL spatial: Neo4j versus PostGIS. Delft University of Technology [13] Introducción a JSON. https://www.json.org/json-es.html. Consultado en Diciembre, 2022.

[14] Silvia, C. (2019) Instalación y utilización de MongoDB. http://www.formadoresit.es/instalacion-y-utilizaciondemongodb/.Consultada en Noviembre, 2022.

[15] Sara, M. (2021). Bases de datos NoSQL: Guía definitiva. https://pandorafms.com/blog/es/bases-de-datos-nosql/. Consultada en Noviembre, 2022.