OPERACIONES CRUD EN DATOS SEMIESTRUCTURADOS

Ing. Víctor Manuel Muñiz del Valle[1], Dr. Severino Feliciano Morales[2], Dr. Edgardo Solís Carmona[3], Dr. José Luis Hernández Hernandez[4], Dr. Mario Hernández Hernandez[5], Dr. Valentín Álvarez Hilario[6].

***Resumen*— La estructura de los datos puede evolucionar frecuentemente, sin necesidad de que haya cambios en la estructura de un esquema. La principal característica de este tipo de datos, es el hecho de que no depende de una estructura de un esquema explicito pero existe implícitamente entre los datos o el código. Se puede saber qué tipo de datos se necesitan para obtener un análisis completo de una problemática, para ofrecer soluciones, a pesar de ser schemaless. Los tipos de datos se pueden identificar como Estructurados (Bases de Datos Relacionales, Data Warehouses), Semiestructurados (JSON, XML) y No Estructurados (Texto, Audio y Video), pero lo más que han tomado mayor relevancia son los dos primeros, ya que hacen más fácil el tratamiento de los datos. En este artículo se pretenden abordar los datos semiestructurados y las operaciones fundamentales para los datos en una base de datos NoSQL, específicamente MongoDB.**

Palabras claves: **Datos Semiestructurados, Bases de Datos NoSQL, Schemaless, MongoDB.**

Introducción

En un mundo de constantes cambios a nivel de sistemas, es necesario volver a pensar acerca de los nuevos paradigmas que maneja la industria, se necesita ajustar nuestras herramientas a las necesidades reales que se tienen hoy en día con el fin de tener sistemas de vanguardia, como respuesta a los problemas a estos cambios, que maneja las industria, surgió NoSQL. Este tipo de sistemas no son un sustituto de las bases de datos relacionales, no obstante, es un gran movimiento que busca otras opciones en el almacenamiento de los datos ("No uses solo SQL"). El termino fue primero usado en los años 90 para nombrar una base de datos relacional open source, sin embargo, aunque el termino más correcto sería NoREL (Not Only Relational), como varios autores han señalado, el termino NoSQL ya tiene gran aceptación, es una forma de decir que no todos los problemas son clavos que pueden ser atacados con un RDMS.

Desde una perspectiva de clasificación de los datos pueden ser de tres tipos: datos estructurados, datos semiestructurados y datos no estructurados. Los datos estructurados tienen una larga historia y es el tipo de dato más utilizado comúnmente en las bases de datos tradicionales, recientemente los datos semiestructurados y no estructurados han salido a la luz a medida que la tecnología ha estado evolucionando y que permite aprovechar los datos y extraerlos para obtener información de una mejor manera [17].

El Marketing global de productos para la gestión de datos, define a los tipos de datos semiestructurados como un tipo de datos que contienen etiquetas semánticas o metadatos para facilitar la organización para su fluidez, pero no se ajustan a la estructura asociada con las bases de datos relacionales típicas, si bien las entidades semiestructuradas pertenecen a la misma clase, pueden tener atributos diferentes, los ejemplos incluyen: correo electrónico, XML y de otro tipos de lenguajes de marcado como HTML [16].

Datos Semiestructurados

La noción de los datos semiestructurados surgió para definir las propiedades de los datos involucrados en la web y las nuevas aplicaciones que emergieron en este contexto por ejemplo, base de datos genómicas o geográficas. Una de las principales característica que distingue a los datos semiestructurados es el hecho de no tener que de depender de una estructura que no esté definida mediante un esquema explicito pero que a su vez esto existe implícitamente entre los datos y el código, que se describen generalmente mediante una tupla de pares clave-valor.

Un concepto más claro para definir los pares de clave–valor, son aquellos campos que pueden denotar los atributos, las propiedades de datos y valores que pueden ser primitivos como los valores o que además pueden ser complejos (que están representado por tuplas y arrays).

Las principales características que contiene los datos semiestructurados son:

- Los datos no se ajustan a un modelo de datos, pero tienen alguna estructura.
- Los datos no pueden almacenarse en forma de filas y columnas como las bases de datos.
- Los datos semiestructurados contienen etiquetas y elementos (metadatos) que se utilizan para agrupar datos y

describir como se almacenan los datos.

- Las entidades en el mismo grupo pueden o no tener los mismos atributos o propiedades.

Además los datos semiestructurados contienen ventajas como:

- Los datos no están restringidos por un esquema fijo.
- Es flexible, es decir, el esquema se puede cambiar fácilmente.
- Los datos son portables.
- Es posible ver datos estructurados como datos semiestructurados.
- Es compatible con usuarios que no puedan expresar su necesidad en SQL.

XML es considerado un lenguaje de marcado, es decir, utiliza *etiquetas* (etiquetas de texto con formato especial) para identificar la función en los elementos de los datos variados dentro de un documento. XML había sido el formato que se utilizaba para almacenar datos semiestructurados a partir de la llegada de la web con su adopción por el Consorcio Wold Wide Web (W3C) como uno de los estándares para el intercambio de datos en la web, sim embargo, desde la aparición de JSON (JavaScript Object Notation), XML ha perdido predominio como formato de intercambios de datos. De hecho es el formato utilizado para almacenar datos en varias bases de datos NoSQL, además es un formato de texto legible con un estándar ampliamente utilizado para representar los datos semiestructurados, esta notación está tomando el lugar de XML como el formato de intercambio de datos principales por ser simple y legible, JSON es un pequeño subconjunto de JavaScript, los datos JSON son literales de JavaScript. En la figura 1, se muestra una representación equivalente en JSON y XML.

```json
{
    "sessionStart": "16-03-18-12-33-09",
    "sessionEnd": "16-03-18-12-33-12",
    "mapName": "TestMap",
    "logSections": [{
        "sector": {
            "x": 2.0,
            "y": -1.0,
            "z": 0.0
        },
        "logLines": [{
            "time": 37.84491729736328,
            "state": 0,
            "action": 1,
            "playerPosition": {
                "x": 24.560218811035158,
                "y": -8.940696716308594e-8,
                "z": 3.3498525619506838
            },
            "cameraRotation": {
                "x": 0.24549755454063416,
                "y": 0.017123013734817506,
                "z": 0.031348951160907748,
                "w": -0.9687389135360718
            },
    ...
```

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <sessionStart>16-03-18-12-33-09</sessionStart>
    <sessionEnd>16-03-18-12-33-12</sessionEnd>
    <mapName>TestMap</mapName>
    <logSections>
        <sector>
            <x>2</x>
            <y>-1</y>
            <z>0</z>
        </sector>
        <logLines>
            <time>37.84491729736328</time>
            <state>0</state>
            <action>1</action>
            <playerPosition>
                <x>24.560218811035156</x>
                <y>-8.940696716308594e-8</y>
                <z>3.3498525619506836</z>
            </playerPosition>
            <cameraRotation>
                <x>0.24549755454063416</x>
                <y>0.017123013734817505</y>
                <z>0.031348951160907745</z>
                <w>-0.9687389135360718</w>
            </cameraRotation>
    ...
```

Figura 1. JSON vs MXL (Tomado de [14]).

Base de datos NoSQL

El término de NoSQL se refiere a un conjunto de paradigmas de modelado de datos, destinados para gestionar los datos semiestructurados y no estructurados [1].

Las bases de datos NoSQL surgieron para la solución de constantes requerimientos en los procesamientos y en los análisis de las grandes cantidades de datos, por lo que los sistemas tradicionales no son lo suficientemente aptos. NoSQL ha evolucionado como un *no lenguaje SQL*, para referirse a sistemas que no son DBMS (database management system) tradicionales. Las base de datos NoSQL (Not only SQL) es el nuevo conjunto de nuevas tecnologías que están contribuyendo al manejo de la información que no cumple con un esquema Entidad-Relación (E-R) que son las que están basadas en los funcionamientos de tablas, Joins y transacciones, en cambio, en las bases de datos NoSQL, no se impone a una estructura de datos en forma de tablas y relaciones entre ellas, sino que ayuda a proveer un esquema más flexible y con una escalabilidad horizontal [4].

Las bases de datos permiten ser dinámicos, lo que permite realizarse en los desarrollos ágiles. Las bases de datos NoSQL están clasificadas en cuatro categorías:

- Orientados a clave-valor, rendimiento excelente en volúmenes de datos muy grandes, sin embargo, no permiten verificación de la integridad de datos, llaves foráneas ni disparadores. La validación de datos se debe realizar en la aplicación del cliente (DyaboDB, Azure Table Storage, Couchbase Server, Riak y Redis).
- Orientados a columnas, Ofrecen alto rendimiento en las consultas de agregación, alta eficiencia en la comprensión y distribución de datos, adicionalmente la carga de grandes volúmenes de datos es rápida (Hadoop / HBase, Cassandra, Hypertable, Accumulo, Amazon SimpleDB).
- Almacenamiento de Documentos, gestiona información orientada a datos semiestructurados o documentos. Se

consideran como un escalón superior de los gestores sencillos llave-valor debido a que permiten encapsular pares de llave-valor, por ejemplo CouchDB, MongoDB, HBase, SimpleDB, Base X.

- Sistemas de base de datos por medio de grafos, es un sistema de almacenamiento que permite la adyacencia libre de índice. Se usa la teoría de grafos para recorrer la base. Existen diversos modelos de grafos, uno de ellos es el Grafo de Propiedad, que es el resultado de un esfuerzo por unificar diferentes implementaciones de grafos, como ejemplo de este tipo, tenemos Nej, OrientDB.

Mongo DB

Es un sistema de base de datos orientado a documentos, es *High Performance*, es decir, su rendimiento es bastante alto y rápido y *High Availability* (disponibilidad de datos), lo que hace que sea robusto y con una buena funcionalidad. En la figura 2, se muestra la organización de las bases de datos en MongoDB.

Es una nueva generación de base de datos que almacenan datos de una forma nativa en un formato JSON que aprovechan una arquitectura de los datos semiestructurados; MongoDB es una base de datos distribuida de código abierto, está basado en documentos y está diseñada para los grandes desarrolladores de aplicaciones modernas y de las nuevas tecnologías; además permite el almacenamiento de datos en la nube. Desde esta perspectiva, se cree que es la mejor manera para poder concebir los datos, frente al modelo que tradicionalmente se conoce con tablas que contienen filas y columnas, lo que la hace, más expresiva y potente, además de ser un gestor de base de datos muy popular a nivel internacional. En la actualidad existen una variedad de empresas internacionales trabajando con este enfoque y que realmente son muy innovadoras para impulsar sus productos, como Facebook, Invision, Amazon, Google, Adobe, Verizon, entre otras [5].

De acuerdo a la Figura 2, las bases de datos documentales se organizan como un grupo de colecciones y las colecciones se agrupan en la base de datos. En una instancia de MongoDB, se puede gestionar varias bases de datos, cada una agrupando colecciones

Una colección, es el conjunto de documentos que desempeña una gran función de una manera análoga a las tablas de las base de datos relacionales, las colecciones contienen elementos dinámicos, es decir, puede haber documentos con diferentes estructuras que están dentro de una colección.

Los documentos se encargan de almacenar los registros de datos como documentos BSON, que es una representación binaria del documento JSON, aunque contiene más tipos de datos que el mismo JSON. Los documentos tienen la estructura de pares *campo-valor*, como se muestra en la figura 3, la cual, visualiza como se define la estructura de un documento. Dentro de los documentos, se encuentran los documentos embebidos que son subdocumentos que se integran como parte de otros documentos, los documentos embebidos tienen las mismas características que sus documentos raíz [7].

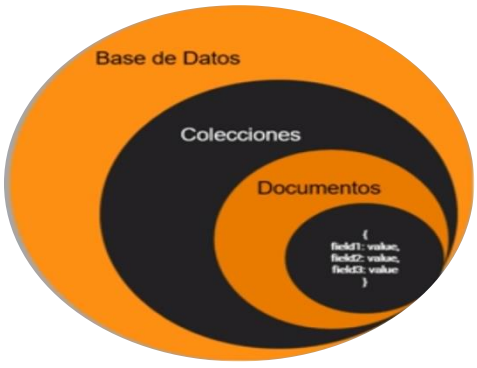

Figura 2. Organización de MongoDB.

```
{
    field1: value1,
    field2: value2,
    field3: value3,
    ...
    fieldN: valueN
}
```

Figura 3. Estructura de un documento.

Las referencias se pueden implementar de dos formas:

1. Referencias donde se guarda el *campo_id* de un documento en otro documento como referencia, luego su aplicación se puede ejecutar con una segunda consulta para que pueda devolver los datos relacionados.
2. DBRefs son referencias de un documento a otro utilizando el valor del campo *_id* del primer documento, nombre de la colección y opcionalmente, su nombre de base de datos, al incluir estos nombres DBRefs permite que los documentos ubicados en múltiples colecciones se vinculen más fácilmente de una sola colección [8].

MongoDB cuenta por defecto con una consola construida en JavaScript desde la cual nosotros podemos ejecutar una variedad de comandos, que permite también definir variables, funciones y se pueden utilizar algunos bucles (for, while,

do while). Si se requiere utilizar MongoDB en un lenguaje de programación distinto, existen drivers oficiales para C#, Java, Node.js, PHP, Python, Ruby, C, C++, Perl o Scala. En la figura 4, se visualiza cómo trabaja MongoDB en una aplicación web con Node.js [9].

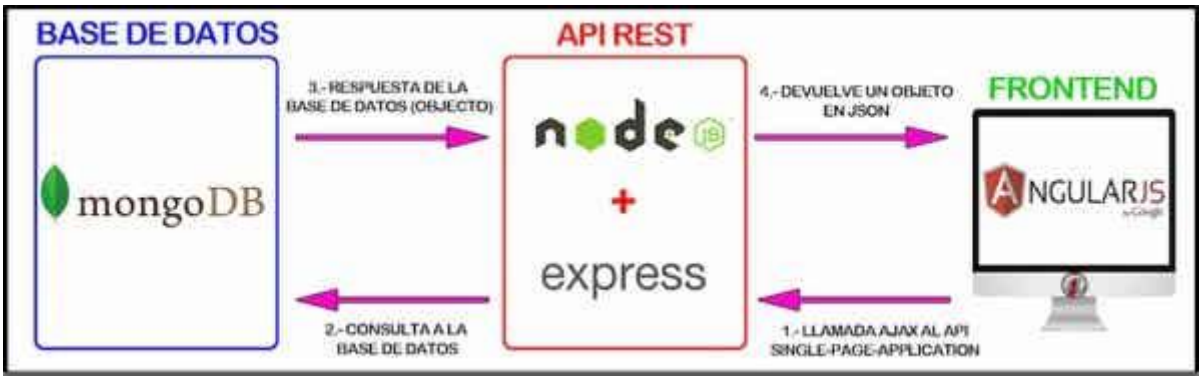

Figura 4. Modelado de Datos en MongoDB (Tomado de [15]).

Operaciones CRUD

En cualquier aplicación que pueda almacenar datos semiestructurados puede usar MongoDB y se pueden implementar todas las típicas operaciones CRUD. Para usar MongoDB no se necesita definir algún tipo de esquema (Entidad-Relación, Relacional), ya que es *schemaless*. Este sistema de base de datos está apto para los entornos que requirieran una gran escalabilidad, por lo que se puede conseguir un sistema escalable sin ninguna dificultad. Para mostrar la implementación de las operaciones CRUD, primeramente se crea una base de datos llamada *Horarios* y en la figura 5 se pueden observar las colecciones que para la base de datos.

```
docente
{
    "id": "xxx",
  "nombre":"xxx",
   "área:":"xxx",
 "especialidad": "xxx"
```

```
materia
{
    "id": "xxx",
  "edificio": "xxx",
 "semestre:":"xxx",
   "turno": "xxx",
   "área": "xxx"
```

```
grupo
{
    "id": "xxx",
  "nombre": "xxx",
 "créditos:": "xxx",
 "semestre": "xxx",
   "hora": "xxx",
"capacidad": "xxx",
 "edificio": "xxx",
```

```
aula
{
    "id": "xxx",
 "capacidad": "xxx",
 "ubicación:": "xxx",
}
```

Figura 5. Colecciones para la base de datos

Las operaciones CRUD (Create, Read, Update, Delete, por sus siglas en Ingles) hace referencia a las operaciones básicas de **Crear, Leer, Actualizar y Eliminar**, en este caso para la base de datos MongoDB.

- **Operación Create**, esta operación de creación o de inserción pueden agregar nuevos documentos a una colección, si una colección no existe, la operación de inserción podrá crear una colección, en la figura 6 se crea la colección Docente y en la figura 7 se agrega un dato a la misma colección.

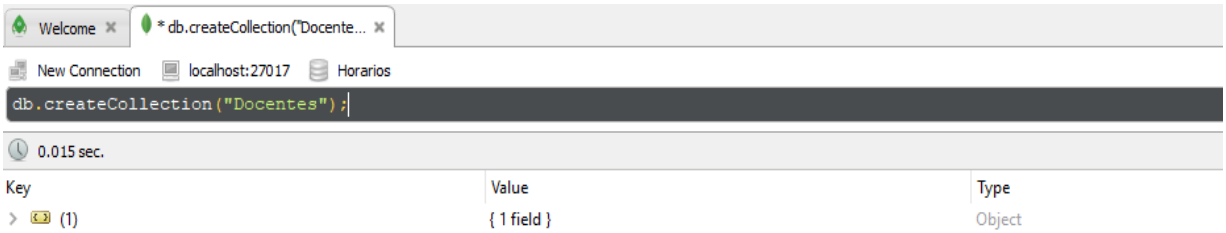

Figura 6. Implementación de la estructura del método createCollection().

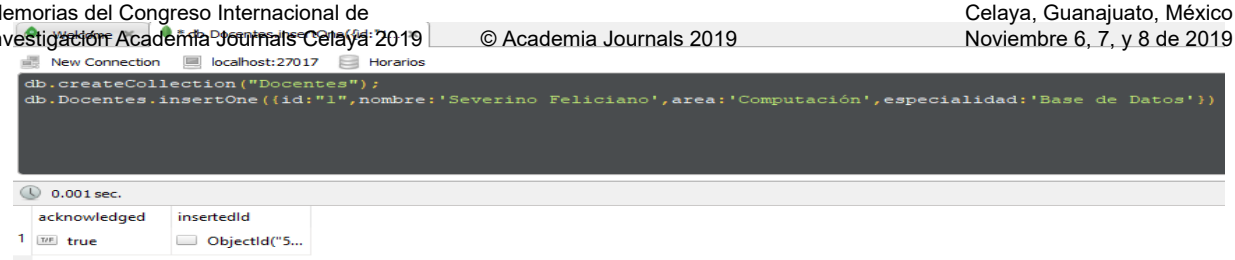

Figura 7. Implementación de la estructura del método insertOne().

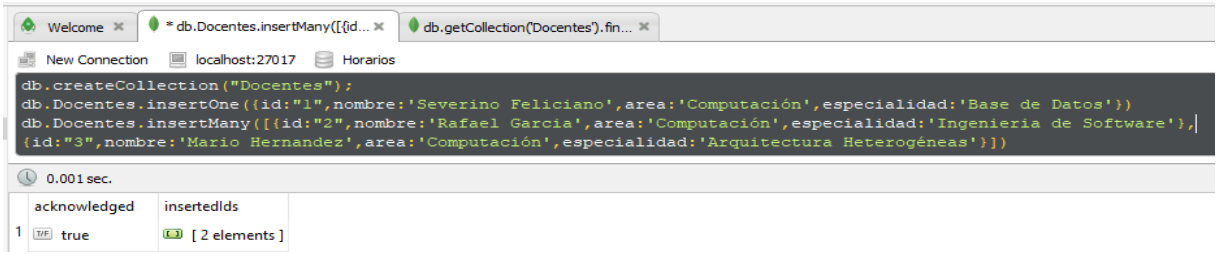

Figura 8. Implementación de la estructura del método insertMany() .

Como se puede observar en la figura 8, se agregan varios documentos en la misma colección de la figura 6 y aunque se utiliza las misma estructura que la figura 7, acá se utiliza un arreglo y los documentos, son separados mediante comas (,).

- **Operación Read:** aquí se recuperan los documentos de una colección, es decir, se lleva a cabo una consulta de una o varias colecciones de documentos.

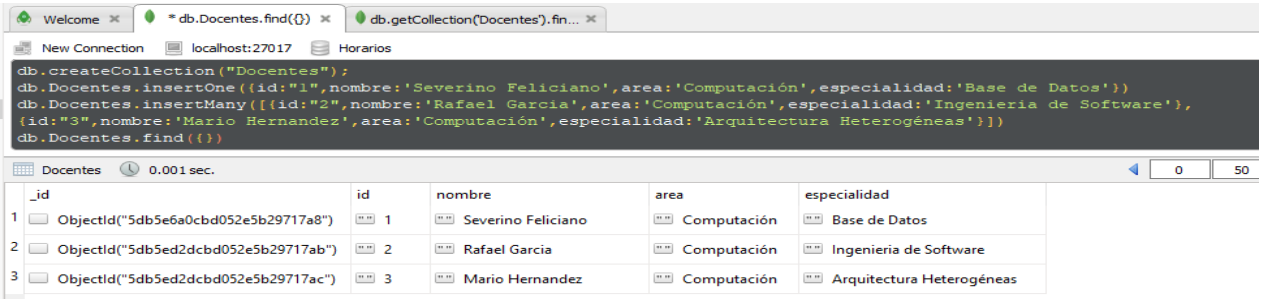

Figura 9. Implementación de la estructura del método find().

Como se observa en la figura 9, se pueden visualizar todos los documentos que contiene la colección, además se puede realizar búsquedas de acuerdo a un filtro que estaría compuesto con su *campo: valor*, por ejemplo ejemplo, db.Docentes.find({nombre:"Severino"}), además se puede observar que MongoDB asigna un id automático, cuando el programador no define alguno.

- **Operación Update:** se implementa la modificación de los documentos que existen en una colección.

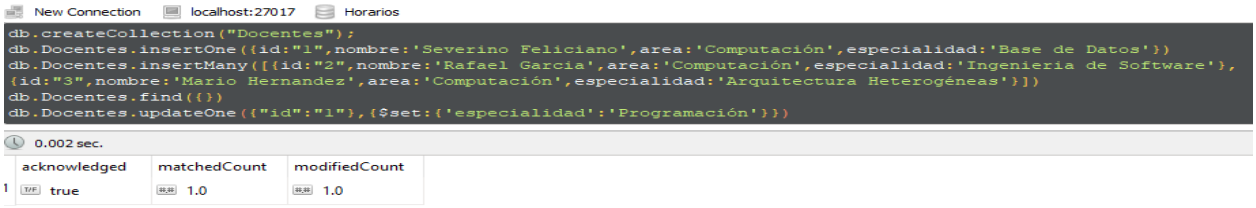

Figura 10. Implementación de la estructura del método updateOne().

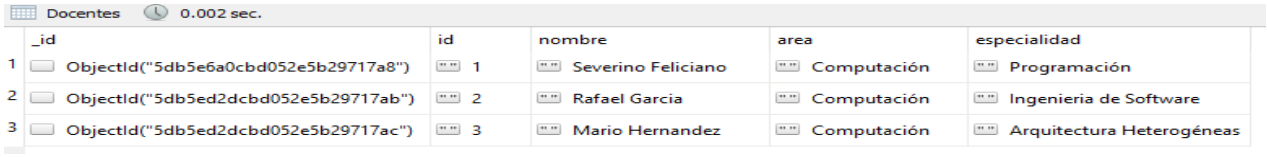

Figura 11. Resultado de updateOne.

De acuerdo a la figura 10, se ha hecho el cambio de la especialidad del id 1, además este método también se puede implementar en Many(), no solo es el caso de actualizar los datos de un documento, sino también de las colecciones, como se puede observar en la figura 11, donde se muestra el resultado del cambio de un campo .

- **Operación Delete:** elimina los documentos de una colección.

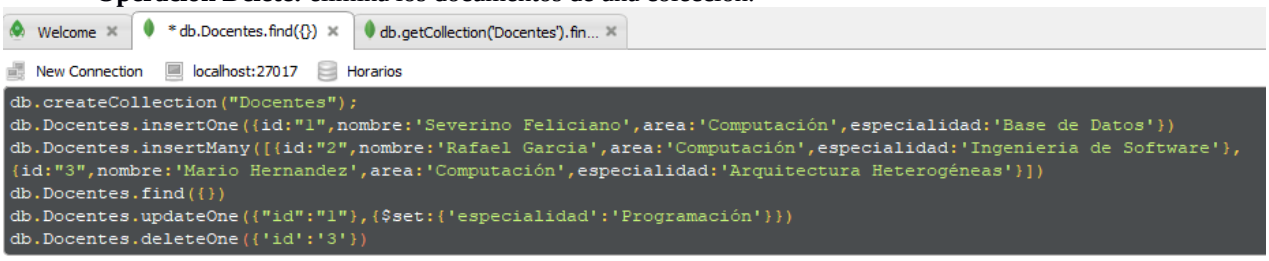

Figura 12. Implementación de la estructura del método deleteOne().

Figura 13. Resultado de la aplicación de deleteOne().

En la figura 12 se está implementando la eliminación y en la figura 13 se puede apreciar que se ha eliminado un documento, sin embargo, también se puede implementar para varios documentos con deleteMany(), pero para esto, deben estar de acuerdo un *campo:valor* para que se puedan eliminar. También se pueden eliminar una o varias colecciones.

Conclusiones

Los datos semiestructurados contienen una estructura irregular y muy parcial, algunas fuentes tienen una estructura de datos implícitas, que puede dificultar una interpretación de la relación entre los datos, en cambio, en un esquema los datos están estrechamente acoplados, es decir, no solo están unidos entre sí, sino que también dependen uno del otro pero en la falta de un esquema fijo y rígido, se puede dificultar la validación en el almacenamiento de los datos, sin embargo, se puede ganar bastante en la flexibilidad, ya que al ser *echemaless*, los campos pueden no ser uniformes, es decir, un campo dado, puede ser de cualquier tipo, incluso pueden existir campos opcionales.

En MongoDB, las operaciones CRUD se pueden implementar de manera muy sencilla, pero ofrecen una gran funcionalidad y robustez, lo que ha hecho que MongoDB, sea preferido por los desarrolldores.

Referencias

1. Feliciano Morales, Severino et. al. (2017). Inferring NoSQL data schemas with model-driven engineering techniques, Tesis de Doctorado. Universidad de Murcia. Consultada por Internet el 10 de septiembre del 2018. Dirección de internet: http://hdl.handle.net/10201/53472.
2. Hernández Chillón, Alberto, Severino, Feliciano, Sevilla, Diego y Molina, Jesús. (2017). Visualización de Esquemas en Bases de Datos NoSQL basadas en documentos.
3. Martin Adriana, Chávez Susana, Rodríguez Nelson, Valenzuela Adriana y Murazzo María.(2013), Bases de Datos NoSql en Cloud Computing, XV Workshop de Investigadores en Ciencias de la Computación ,San Juan. http://sedici.unlp.edu.ar/bitstream/handle/10915/27121/Documento_completo.pdf?sequence=1&isAllowed=y
4. Castro Romero A., González Sanabria, Sebastián Juan y Callejas Cuervo M. , (2012), Utilidad y funcionamiento de la bases de datos NoSQL, Facultad de Ingeniería, Vol. 21 (Núm. 33) , pp. 21-32, Universidad PEdagogixa y Tecnológica de Colombia, Tunja Colombia., https://www.redalyc.org/pdf/4139/413940772003.pdf
5. Página de MongoDB, (Visitado, Octubre 7 de 2019), https://www.mongodb.com/es
6. Tutorial MongoDB - ¿Qué son los Documentos y Colecciones en una Base de Datos Mongo DB? (Visitado, Octubre 15 de 2019),https://www.youtube.com/watch?v=IGhVWruOhuA
7. Emmita 2017, Base de datos NoSQL (Mongo DB), (Visitado, Octubre 20 de 2019), https://medium.com/@Emmitta/base-de-datos-nosql-mongodb-3eaa8a1b1866
8. Database References, (Visitado, Octubre 20 de 2019), https://docs.mongodb.com/manual/reference/database-references/
9. Zambrana Aldunate Nelio F y Claute Fuentes María A, (2018), NoSQL La Nueva Generacion de Base de Datos, (Trabajo Final Presentado Para Obtener el Certificado de Diplomado Experto en Desarrollo de Aplicaciones Empresariales Versión 1) Cochabamba –Bolivia, https://es.scribd.com/document/414366255/NoSQL-La-Nueva-Generacion-de-Base-de-Datos
10. What is Semi-structured data? (Visitado Octubre 25 de 2019), https://www.geeksforgeeks.org/what-is-semi-structured-data/
11. Obtenga lo básico sobre las bases de datos NoSQL: bases de datos XML (Visitado Octubre 25 de 2019) https://www.forbes.com/sites/metabrown/2018/03/31/get-the-basics-on-nosql-databases-xml-databases/#560959fb710e
12. CRUD Operations in MongoDB (Visitado Octubre 25 de 2019) https://www.studytonight.com/mongodb/crud-operations-mongodb
13. MongoDB CRUD Operations (Visitado Octubre 25 de 2019), https://docs.mongodb.com/manual/crud/
14. Representation of JSON vs XML formatted log files (Visitado Octubre 28 de 2019) https://www.researchgate.net/figure/Representation-of-JSON-vs-XML-formatted-log-files_fig8_326434680
15. MEAN (Mongo-Express-Angular-Node) Ejemplo de Aplicación Web (Parte II),(Visitado Octubre 28 de 2019), https://jarroba.com/mean-mongo-express-angular-node-ejemplo-de-aplicacion-web-parte-ii/
16. SemiStructured Data (Visitado Octubre 2 de 2019) https://www.datamation.com/big-data/semi-structured-data.html
17. NoSQL la evolución de las bases de datos (Visitado Octubre 15 dr 2019), https://sg.com.mx/revista/28/nosql-evolucion-bases-datos
18. Gloria Concepción Tenorio Sepúlveda Fundamentos de NoSQL. http://tesch.edu.mx/cw2/tesch.edu.mx/oa/clasificacin_de_las_bases_de_datos_nosql.html?nav=false (Visitado Octubre 22 de 2019).