MAPEO DE OBJETOS A TRAVÉS DE UN MOTOR DE DATOS NOSQL: CASO FRAMEWORK DE DESARROLLO PARA APLICACIONES WEB

Roger Calderón Moreno, Daniel Arenas Seleey
Facultad de Ingeniería, Universidad Autónoma de Bucaramanga
Bucaramanga, Colombia
rcalderonmoreno@gmail.com
darenass@unab.edu.co

Abstract—El presente proyecto surgió como una iniciativa académica en la cual se observa que las áreas de conocimiento en el desarrollo de software bajo el paradigma de programación Orientada a Objetos está confrontado por un modelo de almacenamiento de datos de tipo relacional lo que plantea dos escenarios diferentes que los desarrolladores tratan de mitigar a través de conversiones entre tipos o utilizando herramientas intermedias como el mapeo de objetos relacional que traen ciertas ventajas y desventajas, y por lo cual, se planteo dentro de este proyecto la posibilidad de utilizar un motor de almacenamiento de tipo no relacional o NoSQL.

Los objetos de la aplicación que se desarrollo se almacenan en el motor de almacenamiento MongoDB, el cual organiza los datos en forma de documentos. La estructura dinámica de estos documentos se puede utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales.

I. INTRODUCCION

Los objetos de nuestra aplicación serán almacenados directamente en un motor de almacenamiento no relacional sin necesidad de hacer conversiones para ingresar o para leerlos, para lo cual se construyo un Framework Prototipo para implementar Aplicaciones Web teniendo como base de almacenamiento MongoDB, de tal forma que el usuario final pueda construir sus aplicaciones con base en el modelo de objetos que abstrae de los requerimientos de la aplicación.

Con el fin de conocer la aceptación por parte de algunos profesionales desarrolladores de software se aplicaron encuestas a una población seleccionada indagándoles en temas como la persistencia de datos, el mapeo objeto relacional y las bases de datos NoSql.

II. ESTADO DEL ARTE

A partir del análisis que se realice del problema que intentemos solucionar a través de una aplicación de software, se deberá partir de un modelo de objetos que representen el mundo del problema y de cómo se relacionan, este proceso de abstracción de entidades del mundo real a entidades objetos nos arrojara un modelo de datos orientado objetos. Sobre este modelo que se puede expresar a través de un diagrama de clases se inicia la construcción de la aplicación.

La interacción de los objetos que se crean dinámicamente dentro de la aplicación se da sin ninguna dificultad, la problemática inicia cuando se debe interactuar con la plataforma de almacenamiento de datos relacional, la cual es el estándar que la industria ha adoptado desde hace casi 30 años, y funciona sobre el modelo de datos relacional, postulado sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California) (1)

En este punto, ya tenemos dos consideraciones importantes en el momento de desarrollar una aplicación de software con almacenamiento de datos sobre un motor relacional:

- Un modelo de objetos que representan el mundo de la aplicación.
- Un modelo para el almacenamiento de los datos condicionado al modelo entidad relación.

Como se evidencia son modelos totalmente diferentes que tienen que coexistir dentro del software, para lo cual los desarrolladores han utilizado estrategias, de las cuales podemos resaltar: desarrollar los objetos trabajar con ellos, y en el momento de almacenar los datos a través de operaciones tipo CRUD (Crear, Obtener, Actualizar y Borrar) en el motor de almacenamiento, se convierten los objetos que se requieran a la representación del modelo relacional establecido y para obtener los datos de motor del almacenamiento y poder operar sobre ellos se debe realizar el proceso inverso.

Este tipo de conversiones entre modelos se pueden implementar con programación manual de todas estas conversiones y/o través de APIS que permitan de alguna manera liberar al programador de estás conversiones a través del Mapeo de Objetos Relacional - ORM.

Entre las ventajas que ofrecen los ORM se encuentran: rapidez en el desarrollo, abstracción de la base de datos, reutilización, seguridad, mantenimiento del código, lenguaje propio para realizar las consultas. No obstante los ORM traen consigo algunas desventajas como el tiempo invertido en el aprendizaje.

Este tipo de herramientas suelen ser complejas por lo que su correcta utilización requiere un espacio de tiempo a emplear en conocer su funcionamiento adecuado para posteriormente aprovechar todo el partido que se le puede sacar. *Otra desventaja es que las aplicaciones suelen ser algo más lentas. Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá transformarlas al lenguaje propio de la herramienta, luego leer los registros y por último crear los objetos*. (2)

Como se ha mencionado hasta el momento, estos ORM generan una capa intermedia entre las aplicaciones y el repositorio de datos relacional, lo cual puede generar que las aplicaciones puedan ser un poco más lentas y no aprovechan el potencial del paradigma orientado a objetos. *Además, dentro de los ORM se pueden resaltar algunos aspectos poco positivos como:*

- *Curva de aprendizaje: Las herramientas (o librerías) ORM suelen ser muy amplias, por lo que llegar a explotar su máximo rendimiento costará tiempo* (3).
- *Menor rendimiento: Está claro que tener una capa tan enorme entre tu código y el sistema de base de datos, ralentizará un poco la aplicación* (4) (3)
- *Aumento de tiempo: mapeando objetos y relaciones innecesario y creciente* (5).
- *Realización de consultas complejas: para relaciones sencillas y cargas de innecesaria de datos* (6) (5).
- *Sistemas complejos: Normalmente la utilidad de ORM desciende con la mayor complejidad del sistema relacional. Es decir, si tienes una base de datos compleja, ORM también se te hará más complejo y perderás más tiempo adaptando tus clases que en un sistema de menor complejidad* (3).

Dentro de los ORM que más fuerza han tenido y se mantienen vigentes esta: Hibernate. El cual es software libre, distribuido bajo los términos de la licencia GNU LGPL.

Existen implementaciones de ORM a través de los denominados Frameworks de desarrollo como: .NET o Java, pero también el concepto se aplica a ámbitos más específicos, por ejemplo; dentro de Java en el ámbito específico de aplicaciones Web tenemos los framework: Struts, Java Server Faces o Spring. Estos frameworks de Java en la práctica son conjuntos de librerías (API's) para desarrollar aplicaciones Web, más librerías para su ejecución (o motor), y más un conjunto de herramientas para facilitar esta tarea (debuggers, ambientes de desarrollo como Eclipse, etc). (7)

Bases de datos no relacionales o NOSQL: *NoSQL significa "no sólo SQL". En sentido más amplio, que incluye todo sistema gestor de bases de datos no relacional (que pueden o no pueden hacer uso de un lenguaje de consulta). Dentro de NoSQL existen algunos motores no relacionales como: Redis, Cassandra, MongoDB, CouchDB,BigTable, etc.* (8)

Los sistemas NoSQL son distribuidos, diseñados para el almacenamiento de datos a gran escala y para el procesamiento de datos masivamente paralelo a través de un gran número de servidores básicos. También utilizan lenguajes y mecanismos de tipo No-SQL para interactuar con los datos, en algunos se encuentran API que permiten convertir el SQL tradicional a lenguaje de consulta propio del motor para facilitar la transición entre lo relacional y no relacional. Los sistemas de bases de datos NoSQL surgieron junto a las principales empresas de Internet, como Google, Amazon y Facebook; que tenía problemas para hacer frente a enormes cantidades de datos que con las soluciones RDBMS no podía hacer frente. (9)

Dentro de las bases de datos no relacionales más representativas encontramos MongoDB, el cual es un sistema orientado a documentos, desarrollado bajo el concepto de código abierto.

MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico. (10)

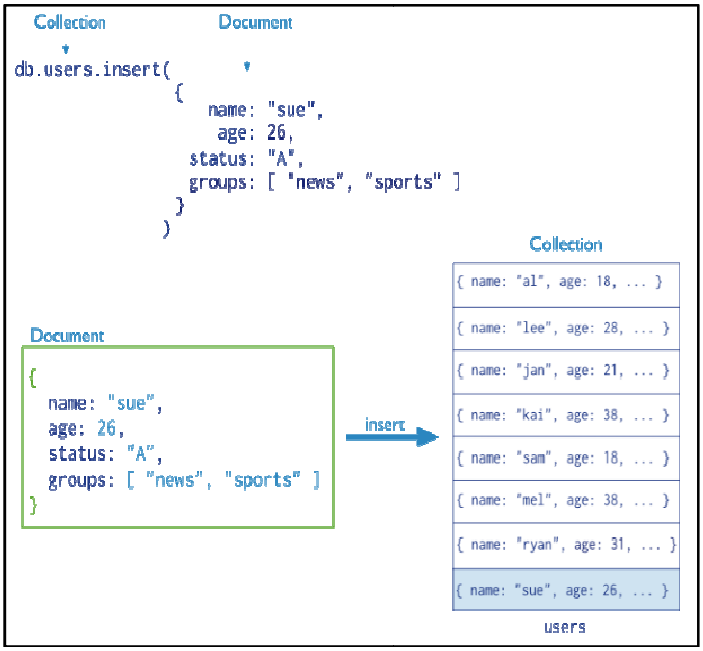

Ilustración 1: Ejemplo de inserción en una Collection. Imagen obtenida del Manual MongoDB.

MongoDB contienen colecciones, una colección está hecha de documentos y cada documento está compuesto de campos de tipo BSON, los cuales permiten tener dentro de cada campo otros subdocumentos, lo cual, facilita tener una estructura de la información dinámica, según lo representa la ilustración 1.

La estructura dinámica de documentos de MongoDB, se puede utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. De aquí, que la Consultora Gartner, Inc., ha publicado en el Octubre de 2014 el "Magic Quadrant for Operational Database Management Systems", que representa la tendencia entre las empresas de tecnologías más influyentes en el mercado sobre sus expectativas sobre los motores de almacenamiento de datos relacionales y no relacionales, y allí han incluido a MongoDB.

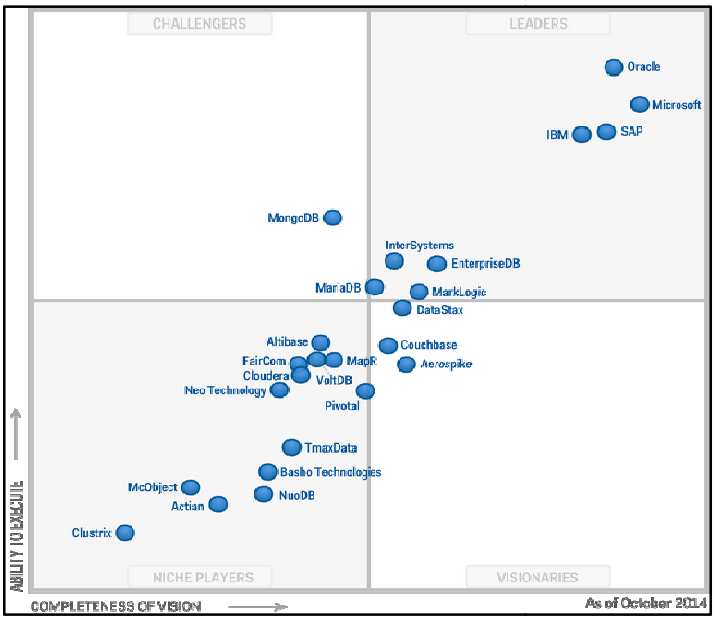

Ilustración 2: Magic Quadrant for Operational Database Management Systems - https://www.mongodb.com/lp/misc/gartner-mq-op-db-report

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) *es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999.*

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros.

JSON está constituido por dos estructuras:

- *Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.*

- *Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.* (11)

A continuación se describe un ejemplo en formato JSON:

```
{
  "tipo": "Class",
  "nombre": "Persona",
  "operaciones":  ["Consultar","Insertar","Eliminar","Actualizar"],
  "vistas": [ "PDF", "cargo;Contenga una cadena;" ]
}
```

III. DESARROLLO DEL PROYECTO

Motivado por los avances que se ha estado generando en ámbito de las bases de datos NoSQL y conociendo las ventajas y desventajas que ofrece el desarrollo de software basado en ORM, se propone generar un Framework Prototipo de Desarrollo Web, que permita acceder a un repositorio de datos ubicado en MONGODB.

Con esta implementación, se pretende ofrecer una alternativa de acceso a datos, diferente a la propuesta por la persistencia de datos relacional, facilitando el proceso de desarrollo orientado a objetos, de tal forma que entre la aplicación resultante y en el motor de almacenamiento de datos, siempre se pueda mapear y trabajar todo el tiempo con objetos, como lo indica la ilustración 3.

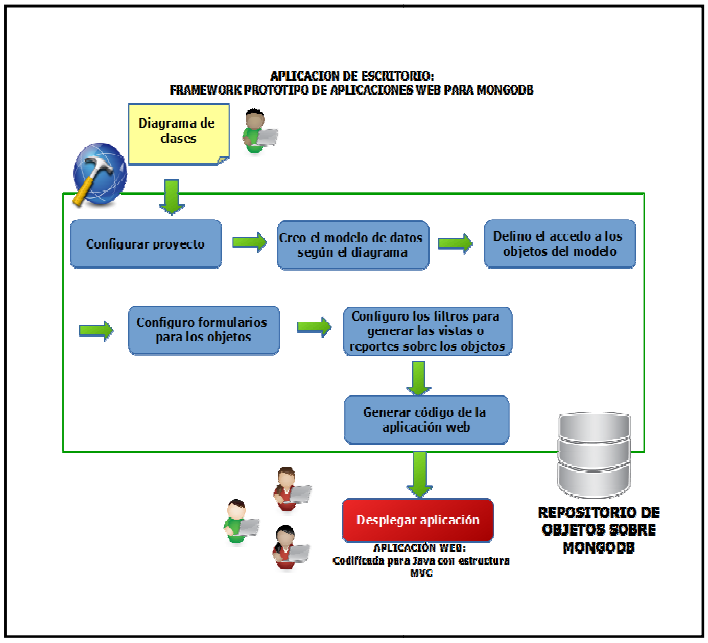

Ilustración 3: Ciclo de acceso a datos propuesto

Un objeto definido en modelo se podrá almacenar con todos sus atributos gracias a la estructura dinámica de MongoDB, como se evidencia en la ilustración 4.

DEFINICIÓN DEL OBJETO	OBJETO ALMACENADO EN MONGODB – FORMATO JSON

public class Persona {
 	
private Double identificacion;
private String tipo_documento;
private String nombres;
private String apellidos;
private String direccion;
private Date fecha_nacimiento;
private String telefonos;
private String email;
private String usuario;
private String clave;
private Integer hijos;
private Historial historial;
 	
}
	{
	"_id": {"$oid": "569138fe9e59e308e0583e53" },
	"identificacion": 86072892,
	"tipo_documento": "CC",
	"nombres": "Roger",
	"apellidos": "Calderon Moreno",
	"direccion": "Calle 5a 31a-20",
	"fecha_nacimiento": "12/03/1982",
	"telefonos": "3118371736",
	"email": "rcalderonmoreno@gmail.com",
	"usuario": "rcalderonmoreno",
	"clave": "123456789",
	"hijos": 2,
	"historial":{
	"fecha_ingreso": "10/01/2015",
	"cargo": " DESARROLLADOR",
	"salario": 2500000 }
	}

Ilustración 4: Relación objeto con estructura MongoDB

- *Estructura de la información en el motor de almacenamiento MongoDB*

Para cada proyecto creado desde el Framework prototipo se creara una base de daos dentro de MongoDB con el nombre definido por el usuario, la cual estará compuesta por siguientes colecciones: Modelo, EstructuraObjeto y DatosObjetos, como se representa en la imagen 5.

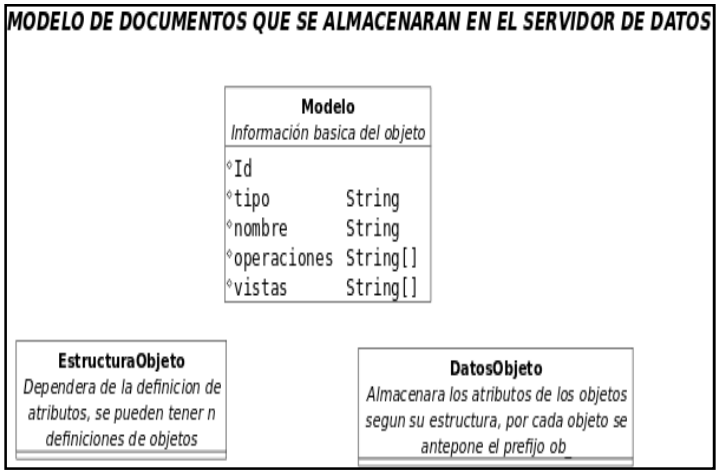

Ilustración 5: Estructura de datos del proyecto para MongoDB

Modelo. Define la información básica para cada objeto definido por el usuario.

EstructuraObjeto. Define la información relacionada con los atributos y métodos (get/set) del objeto definido por el usuario. El nombre de esta colección será el definido por el usuario.

DatosObjetos. Contiene los objetos creados desde la aplicación web, cada objeto tendrá el prefijo ob_. La información de los atributos será almacenada en formato de documentos tipo Json dentro de MongoDB, como lo muestra la ilustración 4.

El proceso de convertir los objetos definidos por los usuarios desarrolladores se realiza a través de:

1. Identificar el tipo de dato compuesto definido por el usuario dentro de la colección denominada Modelo.
2. Determinar los atributos que tiene asignado el objeto con su respectivo nombre y tipo de dato.
3. Crear el documento en formato JSON con la información de los atributos y valores asignados al objeto, para cada atributo es importante almacenarlo según el tipo de dato que tenga asignado. El documento tipo JSON se crea con el objeto *Document* del paquete org.bson.Document.
4. Enviar el documento en formato JSON al motor de datos NoSQL, los datos serán almacenados en una colección que tendrá por nombre el mismo nombre definido para el objeto. Para insertar el documento se utilizo el método *insertOne* del paquete com.mongodb.DBCollection.

Para recuperar la información de un objeto almacenado en MongoDB, se debe realizar:

1. Indicar el nombre del objeto, el cual servirá para filtrar la colección en la cual se debe buscar, si lo desea puede pasar valores tipo Document para filtrar los datos de la colección.
2. Los datos son devueltos a través del método *obtenerDatosColeccion* que devuelve ArrayList <Document>.
3. Determinar los atributos que tiene asignado el objeto con su respectivo nombre y tipo de dato.
4. Se itera el ArrayList <Document> que contiene los datos solicitados y según su tipo de dato se realiza su respectiva conversión.
5. Por último, se instancia el objeto con la información obtenida.

- *Requerimientos definidos para el Framework*
  - *Funcionales.*
    - Almacenar la estructura de los objetos en MongoDB.

- Almacenar la información de los objetos en MongoDB.
- Permitir conectar a bases de datos MongoDB locales o remotas.
- Permitir conectar a bases de datos MongoDB locales o remotas, con restricciones de usuario y clave.
- Para cada objeto creado se podrá configurar: su estructura, permisos (Consultar, Insertar, Actualizar y Eliminar), formulario de ingreso de información y las vistas de los datos.
- El formulario de cada objeto deberá controlar: el tipo de dato, restringir los posibles valores, determinar si un atributo puede ser nulo, el tamaño máximo, el orden de presentación, el mensaje de error y si el atributo identifica al objeto.
- Para cada objeto se podrán crear varias vistas, las cuales estarán condicionadas por el filtro que se agregue.
- Se debe generar una aplicación web con la configuración realizada por el usuario sin que tenga escribir código.
- La aplicación web resultante debe implementar el patrón de diseño singleton, con el fin de restringir la creación de objetos que se conecten a la base de datos.
- La aplicación web resultante debe controlar el acceso a los datos a través de la utilización de la Librería Jquery.
- La aplicación web resultante debe organizar el código siguiente los lineamientos del Modelo-Vista-controlador, como lo indica la ilustración 6.

- *No Funcionales*

  - La aplicación web resultante debe funcionar en los sistemas operativos Windows y Linux.

Con este Framework basado en documentos se propuso darlo a conocer a un grupo de desarrolladores con el fin de conocer el nivel de conocimiento en áreas como: la utilización de ORM en proyectos con sus correspondientes implicaciones, el uso de tecnologías emergentes de almacenamiento de datos no estructurado y su implementación en proyectos. Para lo cual se plantearon dos encuestas las cuales se organizaron y clasificaron, como se indica en la tabla 1, con el fin de conocer la opinión de estos usuarios.

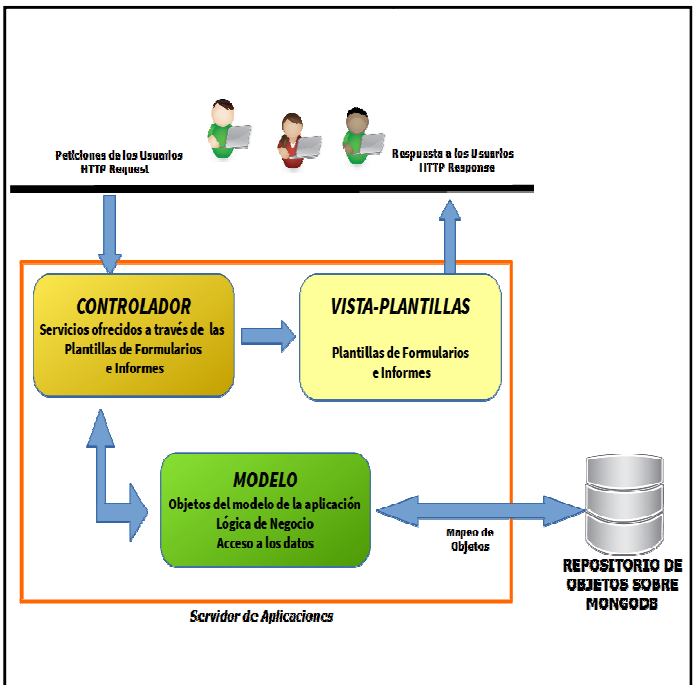

Ilustración 6: Modelo-Vista-controlador para las aplicaciones generadas

Tabla 1. Clasificación de variables

Clasificación	Variable
Herramientas de tipo mapeo objeto-relacional (ORM)	Interacción con herramientas de tipo mapeo objeto-relacional (ORM)
	Dificultad en el proceso de aprendizaje de las herramientas tipo ORM
	ORM utilizadas
	Conocimiento de los problemas de rendimiento de los ORM
Conceptos de Bases de Datos NoSQL	Conocimiento de las BD NOSQL
	BD NOSQL como medio de almacenamiento de objetos
	Mongodb como repositorio de objetos
Framework desarrollado	Concepto del Framework web para Mongodb

La población de la muestra fue seleccionada del personal que trabaja dentro de las distintas empresas que pertenecen a al sector del desarrollo de software del departamento del Meta y demás profesionales de la región. Esta población fue contactada vía correo electrónico, a través del cual se les enviaron las respectivas encuestas.

IV. RESULTADOS OBTENIDOS

Se desarrollo el "Framework Web para MongoDB Ver 1.0" cumpliendo con los requerimientos establecidos para el proyecto, como se evidencia en la ilustración 7.

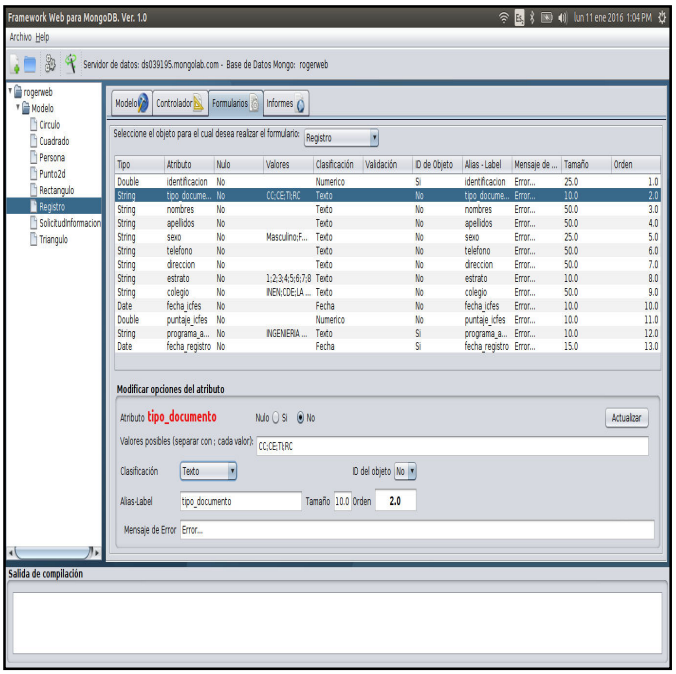

Ilustración 7: Pestaña Formularios - Para modificar la apariencia y comportamiento

Respecto a los resultados de las encuestas, se obtuvieron los siguientes resultados:

En la clasificación Herramientas de tipo mapeo objeto-relacional (ORM), se evidencia que la que un 83.7% población de la muestra conoce o a tenido la posibilidad de trabajar con el ORM. Para ellos, el lenguaje de programación más utilizado para realizar estos trabajos es Java en un 80%, seguido de C# con un 40%.

De los que han desarrollado ORM manifiestan que prefieren hacerlo a través de API´s como JPA o Hibernate, pero evidencia que existe un grado de dificultad en el proceso de aprendizaje, que los podemos catalogar como medio o regular con un porcentaje de 66.7%.

Dentro de la clasificación de Conceptos de Bases de Datos NoSql, se observa que este tema es conocido por la mayoría de los encuestados en un 93.3%, que de los motores de almacenamiento NoSql que mas conocen son: MongoDB con un 76.9%, seguido de Redis con 46.2% y Cassandra con un 23.1%.

Respecto a la posibilidad de almacenar la información de un objeto respetando su definición en un motor de almacenamiento NoSQL los encuestados en un 86.7% manifiestan que sí es posible realizar esta operación. Con relación a la posibilidad de utilizar la estructura basada en documentos ofrecida por MongoDB un 73.3% considera que es posible, mientras un 26.7% considera que no es posible.

Dentro de la clasificación del Framework desarrollado, se observa que los encuestados manifiestan que la aplicación web generada sin programar una sola línea de código para un 73.3% es buena y para 26.6 % no es tan buena. Como caso de estudio el Framework fue recomendado por los encuestados en un 93.3%, pero no fue recomendado como herramienta para generar aplicaciones web a nivel profesional.

V. CONCLUSIONES

• A través del modelo de datos basado en documentos que ofrece MongoDB se logro almacenar la información de los objetos creados en nuestra aplicación, de tal forma que su estructura se represento y almaceno correctamente dentro del motor de almacenamiento no relacional.

• El formato de documentos (Json) utilizado por el motor de datos MongoDB permitió almacenar los objetos definidos por los usuarios del Framework de tal forma que en una sola entidad se tiene organizada toda información, y no se segmenta como en el modelo de datos relacional, se respeta la definición inicial del objeto modelado, a partir de esta premisa, consideramos que se debe generar en mejoras de rendimiento de acceso a los datos, ya que la información estará ubicada en una misma colección y no desagregada en un grupo de tablas.

• Se observa que la población objetivo de la muestra tiene claro los conceptos de persistencia de objetos a través del mapeo objeto relacional (ORM), que el aprendizaje de estas técnicas de desarrollo de software a través de la implementación de código propio o de la utilización de API´s tiene un grado alto de complejidad y en su mayoría (un 60%) son consientes que estas implementaciones generan un bajo rendimiento en las aplicaciones.

• La aceptación del Framework Web los para MongoDB fue satisfactorio y la consideran como una aplicación buena en un 73.3%, pues sus características de fácil manejo y no programar una sola línea de código para generar una aplicación web, hacen de este Framework una opción interesante para los usuarios desarrolladores que están iniciando como para aquellos que ya tienen cierta experiencia.

• La población encuestada evidencia un conocimiento alto (93.3%) sobre las nuevas formas de almacenamiento de información denominadas Bases de Datos NoSql, y se resalta que han utilizado diversas tecnologías como: MongoDB, Redis y Cassandra, para el desarrollo de proyectos. También consideran que es posible almacenar en esta clase de motores

no relaciones los datos de los objetos de nuestras aplicaciones respetando en todo momento su definición inicial.

BIBLIOGRAFÍA

1. **Management, The Relational Model for Database.** [book auth.] E. F. Codd. Boston, MA, USA : Addison-Wesley Longman Publishing, 1990.
2. *Mapeo Objeto / Relacional (ORM).* **Busto, Osmel Yanes Enriquez y Hansel Gracia del.** 2011, Revista Telem@tica, pp. 1-7.
3. **Guardado, Iván.** [Online] 5 2010. http://web.ontuts.com/tutoriales/introduccion-a-object-relational-mapping-orm/.
4. *Evaluación y análisis de rendimiento de los frameworks de persistencia Hibernate y Eclipselink*1.* **Mauro CALLEJAS CUERVO, Diego Iván PEÑALOSA PARRA,Andrea Catherine ALARCÓN ALDANA.** Manizales, Colombia : s.n., 2011, Ventana Informatica.
5. **Vondra, Tomas.** are benefits of orm tools real? [Online] 5 2010. http://www.fuzzy.cz/en/about-me/.
6. **Fink, Gil.** Select N+1 Problem – How to Decrease Your ORM Performance. [Online] Agosto http://www.codeproject.com/Articles/102647/Select-N-1-Problem-How-to-Decrease-Your-ORM-Perfor, 2010.
7. **agenda, The SOA.** http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/. *The SOA agenda.* [Online] sep 25, 2012.
8. *SQL vs NoSQL.* **Mohammed-Ali Khan, ,October 21, 2012.** YORK UNIVERSITY : s.n., 2012.
9. *NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison.* **Hossain, A B M Moniruzzaman and Syed Akhter.** 2013, International Journal of Database Theory and Application Vol. 6.
10. **MongoDB.** MongoDB. [Online] 06 2015. http://docs.mongodb.org/manual/core/introduction/.
11. json.org. [Online] [Cited: 02 29, 2016.] http://www.json.org/json-es.html.
12. **Villugas, Junny.** webespacio. [Online] Abril 10, 2012. [Cited: Agosto 23, 2014.] http://myspace.wihe.net/twitter-trabajos-codigo-mysql/.
13. **Martinez, Daniel.** Working in solutions. [Online] Julio 20, 2010. [Cited: Agosto 23, 2014.] http://blog.danielmartinez.info/?p=12..
14. **Aniszczyk, Chris.** GitHub. [Online] Marzo 26, 2014. [Cited: Agosto 23, 2014.] https://github.com/twitter/mysql/wiki.
15. **Crane, Kevin.** GitHub. [Online] Agosto 21, 2014. [Cited: Agosto 23, 2014.] https://github.com/kevincrane.
16. **Fernández, Rubén.** Genbetadev. [Online] Enero 27, 2014. [Cited: Agosto 23, 2014.] http://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades.
17. **Sobel, Jason.** Facebook. [Online] Diciembre 21, 2007. [Cited: Agosto 23, 2014.] https://www.facebook.com/notes/facebook/keeping-up/7899307130.
18. **Milutinovich, Jovana.** Object Management Group. [Online] Febrero 27, 2014. [Cited: Agosto 23, 2014.] http://www.omg.org/gettingstarted/what_is_uml.htm.
19. **Balsamiq.** Balsamiq Mockups. [Online] 2014. [Cited: Agosto 23, 2014.] http://balsamiq.com/products/mockups/.
20. **Adell, Josh.** github. [Online] 2012. [Cited: Agosto 24, 2014.] https://github.com/jadell/neo4jphp.
21. **Neo Technology, Inc.** Neo4j. [Online] 2014. [Cited: Agosto 25, 2014.] http://neo4j.com/use-cases/geo/.
22. —. Neo4j. [Online] 2014. [Cited: Agosto 25, 2014.] http://neo4j.com/customers/.
23. **Ávila, Patricia Chávez.** Patricia Chávez Ávila: Perfil público . [Online] [Cited: 01 05, 2014.] http://virtual2.unillanos.edu.co/moodle/user/profile.php?id=2337.
24. **Madariaga Orozco, Camilo, Abello Llanos, Raimundo and Sierra García, Omar.** *Redes sociales: infancia, familia y comunidad.* Colombia : Editorial Universidad del Norte, 2003.
25. **Fundación Wikimedia, Inc.** Wikipedia. [Online] Julio 21, 2014. [Cited: Agosto 28, 2014.] http://es.wikipedia.org/wiki/Historia_de_Facebook.
26. —. Wikipedia. [Online] Agosto 25, 2014. [Cited: Agosto 28, 2014.] http://es.wikipedia.org/wiki/Twitter.
27. —. Wikipedia. [Online] Agosto 26, 2014 . [Cited: Agosto 28, 2014.] http://es.wikipedia.org/wiki/Instagram.
28. **Tecnimedios, Blog.** Blog Tecnimedios. [Online] Marzo 26, 2013. [Cited: Agosto 28, 2014.] http://tecnimedios.com/blog/redes_sociales/redes-sociales-verticales/.
29. **Santos, Félix Requena.** *Análisis de redes sociales.* Madrid : CIS Centro de Investigaciones Sociológicas, 2003.
30. **Urzola, Alan Jara.** [Online] [Cited: 11 16, 2012.]
31. **Perez, Carlos Sanchez.** MIS IDEAS Internet, Desarrollo y Televisión. [Online] Febrero 14, 2011. http://carlossanchezperez.wordpress.com/2011/02/14/bases-de-datos-rdbms-vs-no-sql-una-r-evolucion/.
32. Wikipedia - Mapeo objeto-relacional. [Online] 05 2015. http://es.wikipedia.org/wiki/Mapeo_objeto-relacional.
33. **Wikipedia-Hibernate.** Wikipedia. [Online] 05 2015. https://es.wikipedia.org/wiki/Hibernate.
34. SOA-agenda. [Online] http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/.
35. **MongoDB.** [Online] 10 2014. https://www.mongodb.com/press/mongodb-recognized-only-%E2%80%9Cchallenger%E2%80%9D-gartner-2014-magic-quadrant-operational-database.
36. **Objetos, Persistencia en.** Documentos de Google. [Online] 09 2013. https://docs.google.com/document/d/1nCy-Xk00lBUrBFQvTWk9P5xsw8ee6JOVklSUlRN3mUI/edit.
37. **Castillo, Juan Mármol.** *Persistencia de objetos. JDO, Solución Java.* s.l. : Facultad de Informática, Universidad de Murcia, 2013.
38. MongoEngine. [Online] 2014. http://mongoengine.org/#home.
39. **Kaplan-Moss., Adrian Holovaty & Jacob.** django-book. [Online] http://django-book.mkaufmann.com.ar/chapter05.html.
40. **Fernando Alonso Amo, Loïc A. Martínez Normand, Francisco Javier Segovia Pérez.** *Introducción a la ingeniería del software - Modelos de desarrollo de programas.* s.l. : Delta Publicaciones, 2008.
41. **ZonaDiegum.** ZonaDiegum. [Online] 2007. https://diegumzone.wordpress.com/2007/04/01/mapeo-de-objetos-y-tablas-relacionales-or-m-lo-que-a-mi-me-sirvio/.
42. Programación .net. [Online] 2005. http://programacion.net/articulo/motores_de_persistencia_231.
43. *Multiparadigm Data Storage for Enterprise Applications.* **Ghosh, D.** 2010, Software, IEEE , vol.27, no.5, pp. 57,60.
44. *An implementation approach to store GIS spatial data on NoSQL database.* **Zhang, Xiaomin, Song, Wei and Liu, Liming.** Junio 2014, Geoinformatics (GeoInformatics), 2014 22nd International Conference on.
45. **Piattini, A. de Miguel y M.** Fundamentos y Modelos de Bases de Datos. s.l. : RA-MA, 1999.
46. **acenswhitepaper.** [Online] 02 2014. http://www.acens.com/wp-content/images/2014/02/bbdd-nosql-wp-acens.pdf.
47. **couchdb.apache.org.** CouchDB. [Online] [Cited: 02 25, 2016.] http://couchdb.apache.org/.
48. cwiki.apache.org. [Online] [Cited: 02 25, 2016.] https://cwiki.apache.org/confluence/display/COUCHDB/Introduction.