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

(Mapping objects through a data motor NoSQL case study: framework for web applications development)

Roger Calderón Moreno[1], Daniel Arenas Seleey[1]
[1] Facultad de Ingeniería, UNAB. Tecnologías de Información - GTI, rcalderon700@unab.edu.co, darenass@unab.edu.co.
Universidad Autónoma de Bucaramanga.

(Recibido septiembre 4 de 2015 y aceptado noviembre 17 de 2015)

Resumen
El presente artículo 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 - POO 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 del proyecto la posibilidad de utilizar un motor de almacenamiento de tipo no relacional o NoSQL.

Con el diseño y desarrollo del framework para generar aplicaciones web, el usuario podrá definir los objetos que considere incluir en la aplicación, los cuales se almacenarán en el motor 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.

Con el objetivo de socializar y evaluar el trabajo realizado, se diseñaron algunos instrumentos para recopilar información de usuarios con experiencia en el sector de las bases de datos y el desarrollo de software. Como resultado se resalta que los desarrolladores de software tienen 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 conscientes que estas implementaciones generan un bajo rendimiento en las aplicaciones. Además, se resalta la apertura de estos a optar por otras alternativas para organizar y almacenar la información, diferentes al enfoque relacional utilizado desde hace varios años.

Palabras clave: base de datos, framework, java, json, mapeo objeto relacional, mongodb, nosql, objetos.

Abstract
This article emerged as an academic initiative in which it is observed that the areas of knowledge in software development under the paradigm of Object-oriented programming (OOP) is confronted by a model data storage relational raising two scenarios different developers try to mitigate through conversions between types or using intermediate tools such as mapping relational objects that bring certain advantages and disadvantages, and therefore, was raised within the project the possibility of using a storage engine type non-relational or NoSQL.

With the design and development of the framework for generating Web applications, the user can define objects to consider including in the application, which will be stored in MongoDB engine, which arranges the data in the form of documents. The dynamic structure of these documents can be used in many projects, including many who traditionally would work on relational databases.

Aiming to socialize and evaluate the work done, some instruments were designed to collect information from users with experience in the field of databases and software development. As a result highlights that software developers have clear concepts of object persistence through object-relational mapping (ORM), that learning these techniques software development through implementing own code or using APIs have a high degree of complexity and mostly (60%) they are aware that these implementations generate low performance in applications. In addition, the opening of these highlights to choose alternative to organize and store information, different to the relational approach used for several years.

Key words: database, framework, object relational mapping, mongodb, nosql, java.

1. INTRODUCCIÓN

En la actualidad diversos frameworks de desarrollo de aplicaciones permiten el acceso a diferentes bases de datos relacionales a través del mapeo de objetos-relacional (ORM) con el fin de generar una relación entre los objetos que se definen en la aplicación y las entidades o tablas del modelo relacional, incluyendo en estas últimas sus correspondientes relaciones todo con el único fin de ocultar al desarrollador la complejidad implícita del *Structured Query Language* (SQL) y la dependencia de todo el desarrollo a un motor de almacenamiento.

Las bases de datos relacionales solo permiten almacenar datos de tipo primitivo o escalares para almacenar datos de tipo numérico, cadenas de texto, lógicos, fechas, binarios largos, entre otros. Por lo tanto, para almacenar un objeto, el cual se caracteriza por estar compuesto por diversos atributos, los cuales pueden ser de diferentes tipos de datos escalares u objetos, que a su vez pueden estar agrupados. Esta complejidad de los datos que representan al objeto no es posible llevarla directamente al modelo relacional, la estrategia utilizada por los desarrolladores consiste en descomponer el objeto en una o más tablas o entidades.

Para que estos dos componentes: aplicaciones y motores de almacenamiento puedan funcionar juntos, deben poder comunicarse entre ellos intercambiando información que cada uno debe adaptar a su modelo inicial.

En cambio, los programas han usado desde los años 80, un modelo llamado "orientado a objetos", que difiere en mucho del modelo relacional y que se ha extendido cada vez más. Es por ello que aparece un conflicto a la hora de reunir estos dos componentes en una aplicación, ya que cada uno responde a diferente modelo y forma de operar. Cada componente maneja los datos con un formato diferente. Metafóricamente, podríamos afirmar que el programa y la base de datos hablan idiomas diferentes y, por lo tanto, la comunicación entre ellos resulta difícil. En el caso de que toda la aplicación siga el modelo relacional, perdemos las ventajas de la orientación a objetos. En el caso de que toda la aplicación siga el modelo orientado a objetos, tenemos que las bases de datos orientadas a objetos que podríamos usar están inmaduras y tienen un bajo nivel de estandarización. (Neward, 2006) (ZonaDiegum, 2007)

Como una solución a esta dificultad surge el concepto de: mapeo objeto-relacional (ORM*), el cual es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo)* (Zuñiga, 2003).

La adaptación de modelos relacionales a modelos de objetos y viceversa, por parte de los programadores a través de diferentes estrategias como la APIS o frameworks, genera una serie de ventajas como la disminución en la escritura de código y por lo tanto en los tiempos de desarrollo, pero también desventajas como la complejidad de en el aprendizaje de la herramienta y los tiempos de respuestas de las aplicaciones, lo

anterior motivó la realización del proyecto, en el cual para evitar las conversiones entre objetos y tablas, se propuso la utilización de un motor de almacenamiento de datos no relacional o NoSQL, para nuestro caso fue seleccionado MongoDB que posee una estructura de información dinámica basada en documentos, sobre la cual se implementó la definición de los objetos y su correspondiente almacenamiento. Otro factor por el cual se seleccionó MongoDB, es el soporte que tiene para diversos lenguajes de programación a través de las Interfaces de Programación de Aplicaciones - APIS, existen para Java, Python, C entro otros. Además, la constante evolución del producto gracias a la empresa que lo soporta: MongoDB, Inc., a su apertura al Open Source, a la cantidad de documentación relacionada a través del sitio web oficial y de las diversas comunidades de usuarios que lo han implementado para desarrollar diversos proyectos.

Para demostrar esta propuesta de organización de la información, se diseñó y desarrolló un framework para desarrollo de aplicaciones web que permitiera almacenar los objetos dentro del motor no relacional respetando en todo momento su estructura inicial, con lo cual se logró almacenar la información de los objetos creados en la aplicación, de tal forma que su estructura se representó y almacenó correctamente, sin tener que desagregar la información de los objetos.

El framework desarrollado tiene como una característica especial que el usuario solo debe definir las estructura de sus objetos y a partir de allí se genera todo el código fuente para la aplicación web que mapeara los objetos entre la aplicación y motor de almacenamiento no relacional.

El presente artículo se organizó en las siguientes secciones: antecedentes que soportan el trabajo realizado, una metodología, la cual muestra cómo se realizó y organizo el trabajo propuesto, interpretación de los resultados obtenidos y las respectivas conclusiones.

2. ANTECEDENTES

2.1 Mapeo de Objetos Relacional

Cuando se está desarrollando software bajo el paradigma de Programación Orientada a Objetos, se deberá partir de un modelo de objetos que representen el mundo del problema lo que comúnmente se denomina un proceso de abstracción de entidades del mundo real a objetos, lo cual da como resultado una representación a través de un diagrama de clases y sus correspondientes relaciones, y de esta forma iniciar la construcción de la aplicación.

Cuando la aplicación de software inicia su ejecución, se crean dinámicamente una serie de objetos necesarios para su funcionamiento o para representar la información, estos objetos son ubicados en la memoria principal asignada por el Sistema Operativo. Los objetos se crean y se eliminan 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, cuyas bases fueron postuladas en 1970 por Edgar Frank Codd de los laboratorios IBM en San José (California) (Management, 1990)

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:

1. Un modelo de objetos que representan el mundo de la aplicación.

2. Un modelo para el almacenamiento de los datos condicionado al modelo entidad-relación.

Se parte de dos escenarios totalmente diferentes que tienen que coexistir dentro del software, para lo cual en el desarrollo de software se han utilizado diversas 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 del acrónimo Crear, Obtener, Actualizar y Borrar (Wikipedia, 2014) en el motor de almacenamiento relacional, se convierten los objetos que representan la información que debe persistir a la representación del modelo relacional establecido, y para obtener los datos de motor del almacenamiento relacional y poder operar sobre ellos se debe realizar el proceso inverso.

Este tipo de conversiones entre modelos se pueden implementar con programación manual de acuerdo con las anteriores conversiones y/o través de diversas APIS que permitan de alguna manera liberar al programador de estás conversiones a través del Mapeo de Objetos

Relacional (*ORM)* de la definición en inglés: *Object-Relational mapping* .

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*. (Busto, 2011)

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 de tipo ORM tienen un grado de complejidad inherente para su aprendizaje lo cual genera que los tiempos de aprendizaje y aprovechamiento completo de la herramienta sean altos. (Guardado, 2010).

- *Menor rendimiento:* Con esta clase de herramientas de tipo ORM tienen inmerso una serie de capas adicionales entre la aplicación desarrollada y los datos, los cuales afectan los tiempos de respuestas debido a las conversiones internas que tiene que realizar para ocultar la complejidad al usuario. (Mauro CALLEJAS CUERVO, 2011) (Guardado, 2010)

- *Aumento de tiempo y realización de consultas complejas:* Con el fin de facilitar las operaciones CRUD, los ORM de forma constante realizan un mapeo de objetos y relaciones, lo cual afecta los tiempos de respuesta, además, para la realización de consultas que resultarían simples en código SQL, el ORM genera consultas muy complejas generando carga de datos innecesarios. (Vondra, 2010). (Fink, 2010) (Vondra, 2010).

- *Sistemas complejos:* En la medida que el modelo relacional que contiene los datos comience a crecer según los requerimientos propios del negocio, la gestión del ORM será más compleja y no son muy amigables a la gestión del cambio, puesto que habrá que destinar tiempo a crear, modificar y adaptar clases. (Guardado, 2010).

Dentro de los ORM que más fuerza han tenido y se mantienen vigentes esta: Hibernate (projects, 2015). El cual es software libre, distribuido bajo los términos de la Licencia Pública General Reducida de GNU (GNU/LGPL) por su nombre en inglés GNU Lesser General Public License (Foundation, 2014).

Existen implementaciones de ORM a través de los denominados frameworks para lenguajes de programación 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 frameworks: Struts, Java Server Faces o Spring. (agenda, 2012)

2.2 Bases de datos no relacionales

Las bases datos NoSQL o no solo es SQL, son un conjunto de bases datos no relacionales, donde las bases de datos no se construyen sobre la teoría relacional y por lo general no utilizan el SQL como lenguaje de consulta. Estos entornos de bases de datos ofrecen esquemas datos flexibles y tiempos de respuestas cortos ante volúmenes de datos muy altos. (B M Moniruzzaman, 2013). Dentro de las bases de datos NoSQL más representativas se encuentran: Redis, Cassandra, MongoDB, CouchDB, BigTable, etc.

Las bases de datos 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 NoSQL para interactuar con los datos, en algunos se encuentran APIS 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 de los sistemas de administración de bases de datos relacionales por sus siglas en inglés RDBMS no podía hacer frente. (Hossain, 2013) (Mohammed-Ali Khan, 2012)

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, MongoDB Licensing)

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 en formato JavaScript Object Notation (JSON) con un esquema dinámico. (MongoDB, MongoDB, 2015)

MongoDB organiza la información dentro de colecciones, la cual está integrada de documentos y cada documento está compuesto de campos, los cuales permiten tener dentro de cada campo otros subdocumentos, lo cual, facilita tener una estructura de la información dinámica (ver figura 1).

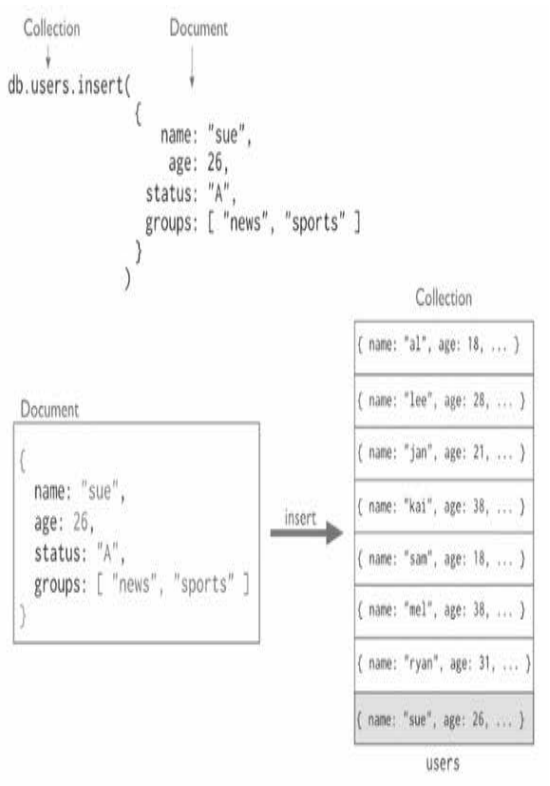

Figura 1. Ejemplo de inserción en una Collection. Imagen obtenida del Manual MongoDB - https://docs.mongodb.com/manual/introduction/ .

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 2015 el "*Magic Quadrant for Operational Database Management Systems"* (ver figura 2)*,* 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.

Lo anterior indica que este motor está incluido en el cuadrante de los líderes en gestores de bases de datos, lo cual representa la adopción por parte de numerosas empresas de este tipo de tecnologías para almacenar la información, y en el proyecto sirvió como respaldo a la selección que se hizo para almacenar la información de los objetos de nuestras aplicaciones.

Figura 2. MongoDB Named a 'Leader' in the 2015 Gartner Magic Quadrant for Operational Database Management Systems - https://www.mongodb.com/lp/misc/gartner-mq-op-db-report

2.3 JavaScript Object Notation - JSON

JavaScript Object Notation - Notación de Objetos de JavaScript (JSON) es un formato ligero de intercambio de datos. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - diciembre 1999 (Ecma International ,Rue du Rhone 114, 2015).

JSON está constituido por dos estructuras: u*na colección de pares de clave/valor, en varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. Y por otro lado, una lista ordenada de valores, en la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.* (JSON, org, 2015)

A continuación se representa un ejemplo en formato JSON (ver figura 3).

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

Figura 3. Ejemplo de la estructura de documento en formato JSON, Fuente: autor.

3. METODOLOGÍA

Motivado por los avances que se han 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 propuso 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 (ver figura 4).

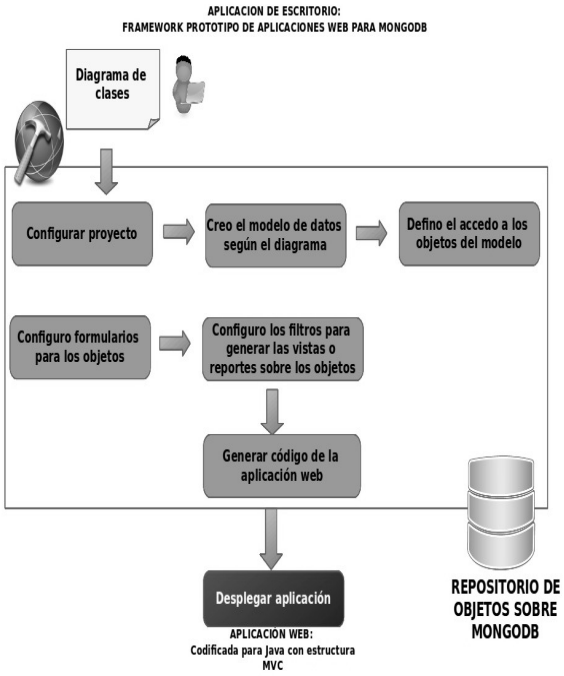

Figura 4. Ciclo de acceso a datos propuesto, Fuente: autor.

Un objeto definido en el diagrama de clases se podrá almacenar con todos sus atributos gracias a la estructura dinámica de MongoDB, lo que permitirá tener bajo un solo documento la información definida para el objeto sin realizar desagregaciones, como lo sugiere el modelo relacional (ver figura 5).

DEFINICIÓN DEL OBJETO

```
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 ArrayList <Historial>  historial;
}
```

OBJETO ALMACENADO EN MONGODB – FORMATO JSON

```
{
   "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,
            "actual":"SI"},
        {
           "fecha_ingreso": "05/02/2013",
           "cargo": "AUXILIAR-TECNICO",
           "salario": 1200000
         "actual":"NO"
         "fecha_retiro": "30/12/2014",
         }]
}
```

Figura 5. Relación objeto con estructura MongoDB, Fuente: autor.

3.1 Estructura de la información en el motor de almacenamiento MongoDB

Para cada proyecto creado desde el framework prototipo se creará una base de datos dentro de MongoDB con el nombre definido por el usuario, la cual estará compuesta por siguientes colecciones: Modelo, EstructuraObjeto y DatosObjetos (ver figura 6).

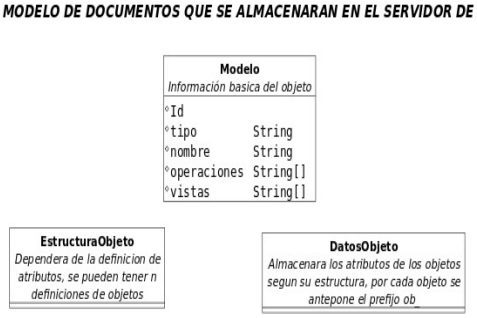

Figura 6. Estructura de datos del proyecto para MongoDB, Fuente: autor.

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

EstructuraObjeto. Define la información relacionada con los atributos y métodos: get y 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 (ver figura 5).

3.2 Proceso de conversión de objetos

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 (API Documentation for MongoDB Drivers, 2015) .

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 utilizó el método *insertOne* (Documentation, 2015) del paquete com.mongodb.DBCollection (Class DBCollection, 2015).

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

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.

3.3 Requerimientos definidos para el framework de desarrollo de aplicaciones web

Dentro del proceso de análisis de requerimientos para desarrollar nuestro framework de caso de estudio, podemos resaltar los siguientes requerimientos:

Almacenar la estructura y la respectiva información de los objetos en MongoDB, 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. Cada objeto se podrá crear varias vistas las cuales estarán condicionadas por el filtro que se agregue.

Al finalizar se debe generar una aplicación Web con la configuración realizada por el usuario sin que tenga que escribir código. La aplicación web resultante debe implementar el patrón de diseño singleton (DEBRAUWER, 2012), con el fin de restringir la creación de objetos que se conecten a la base de datos y debe organizar el código generado según los lineamientos del Modelo-Vista-controlador (ver figura 7).

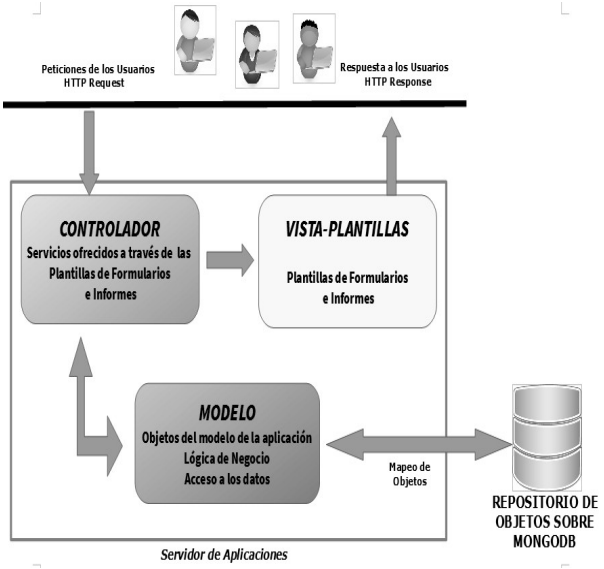

Figura 7. Modelo-Vista-controlador para las aplicaciones generadas, Fuente: autor.

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.

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 al sector del desarrollo de software del departamento del Meta y demás profesionales de la región relacionados con las Tecnologías de Información (TI), los cuales debían tener experiencias en el desarrollo de aplicaciones de tipo escritorio, web o móviles. Esta población fue contactada vía correo electrónico a través del cual se les enviaron las respectivas encuestas, en total se realizaron 32 encuestas.

4. INTERPRETACIÓN DE LOS RESULTADOS

Se desarrolló el "Framework Web para MongoDB Ver 1.0" cumpliendo con los requerimientos establecidos para el proyecto, como se evidencia en la figura 8.

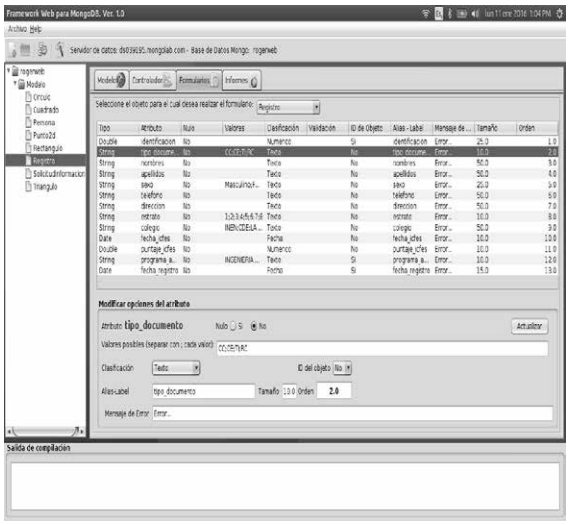

Figura 8. Pestaña Formularios para modificar la apariencia y comportamiento del ingreso de datos, Fuente: autor.

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 un 83.7% de la población conoce o ha tenido la posibilidad de trabajar con diversos ORM. Para ellos, el lenguaje de programación más utilizado para realizar estos trabajos son: Java en un 80%, seguido de C# con un 40%.

De los que han desarrollado software con ORM manifiestan que prefieren hacerlo a través de API´s como JPA o Hibernate, pero manifiestan que existe un grado de dificultad en el proceso de aprendizaje, catalogado 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%, los motores de almacenamiento NoSQL que más 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 que genera el framework 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.

5. CONCLUSIONES

A través del modelo de datos basado en documentos que ofrece MongoDB se logró almacenar la información de los objetos creados en nuestra aplicación, de tal forma que su estructura se representó y almacenó 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 la información y no se segmenta como en el modelo de datos relacional. Se respeta la definición inicial del objeto modelado en el diagrama de clases, a partir de esta premisa consideramos que se debe generar 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 ORM, que el aprendizaje de estas técnicas de desarrollo de software a través de la implementación de código propio o de la utilización de APIS 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 de desarrollo Web fue satisfactorio y la consideran como una aplicación buena en un 73.3%, pues gracias a sus características que facilitan una rápida comprensión de la herramienta y a la generación automática de código de la 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 de software. También consideran que es posible almacenar en esta clase de motores no relaciones los datos de los objetos definidos inicialmente respetando en todo momento su estructura.

6. REFERENCIAS

Agenda, T. S. (25 de sep de 2012). *http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/.* Obtenido de The SOA agenda.

B M Moniruzzaman, S. A. (2013). NoSQL Database: New Era of Databases for Big data Analytics -Classification, Characteristics and Comparison. *International Journal of Database Theory and Application* .

Busto, O. Y. (2011). Mapeo Objeto / Relacional (ORM). *Revista Telem@tica* , 1-7.

Class DBCollection. (2015). Obtenido de http://api.mongodb.com/java/current/com/mongodb/DBCollection.html

Debrauwer, L. (2012). *Patrones de diseño en Java: Los 23 modelos de diseño: descripción y solución ilustradas en UML 2 y Java.* ENI EDICIONES.

Documentation. (2015). Obtenido de https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/

Ecma International, Rue du Rhone 114. (2015). *ECMAScript® 2015 Language Specification.*

Fink, G. (http://www.codeproject.com/Articles/102647/Select-N-1-Problem-How-to-Decrease-Your-ORM-Perfor de Agosto de 2010). *Select N+1 Problem – How to Decrease Your ORM Performance.*

Foundation, F. S. (2014). Obtenido de http://www.gnu.org/licenses/lgpl-3.0.html

Guardado, I. (5 de 2010). Obtenido de http://web.ontuts.com/tutoriales/introduccion-a-object-relational-mapping-orm/

Hossain, A. B. (2013). NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison. *International Journal of Database Theory and Application Vol. 6* .

JSON, org. (2015). *json.org.* Recuperado el 29 de 02 de 2016, de http://www.json.org/json-es.html

Management, T. R. (1990). En E. F. Codd. Boston, MA, USA: Addison-Wesley Longman Publishing.

Mauro CALLEJAS CUERVO, D. I. (2011). Evaluación y análisis de rendimiento de los frameworks de persistencia Hibernate y Eclipselink*1. *Ventana Informatica* .

Mohammed-Ali Khan. (2012). SQL vs NoSQL. YORK UNIVERSITY.

MongoDB. (s.f.). Obtenido de https://www.mongodb.com/community/licensing

MongoDB API Documentation for Java. (2015). Obtenido de http://api.mongodb.com/java/current/org/bson/Document.html

MongoDB. (06 de 2015). *MongoDB.* Obtenido de http://docs.mongodb.org/manual/core/introduction/

Neward, T. (26 de 6 de 2006). *Ted Neward's Blog.* Obtenido de ttp://blogs.tedneward.com/post/the-vietnam-of-computer-science/

Projects, R. H. (2015). Obtenido de http://hibernate.org/orm/what-is-an-orm/

Vondra, T. (5 de 2010). *are benefits of orm tools real?* Obtenido de http://www.fuzzy.cz/en/about-me/

Wikipedia, t. f. (2014). Obtenido de https://en.wikipedia.org/wiki/Create,_read,_update_and_delete#cite_note-james-martin-1

ZonaDiegum. (2007). *diegumzone.wordpress.* Recuperado el 6 de 2015, de https://diegumzone.wordpress.com/2007/04/01/mapeo-de-objetos-y-tablas-relacionales-or-m-lo-que-a-mi-me-sirvio/

ZonaDiegum. (2007). *ZonaDiegum.* Obtenido de https://diegumzone.wordpress.com/2007/04/01/mapeo-de-objetos-y-tablas-relacionales-or-m-lo-que-a-mi-me-sirvio/

Zuñiga, R. (2003). Recuperado el 6 de 2015, de http://programacion.net/articulo/motores_de_persistencia_231