UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMA INFORMÁTICOS

Estudio del uso de MongoDB como alternativa a las bases de datos relacionales tradicionales en aplicaciones web que requieren rapidez de lectura/escritura de los datos almacenados

Estudiante: Diana Marisela Brito Zhunio

Tutor: Ing. Pablo Tamayo

Cuenca – Ecuador

Diciembre 2011

UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMAS INFORMÁTICOS

CERTIFICADO DE RESPONSABILIDAD DEL DIRECTOR DE TESIS

Ing. Pablo Tamayo
Director de Tesis

CERTIFICA:

Que el presente trabajo de investigación “Estudio del uso de MongoDB como alternativa a las bases de datos relacionales tradicionales en aplicaciones web que requieren rapidez de lectura/escritura de los datos almacenados” realizado por la Srta. **Diana Marisela Brito Zhunio**, egresada de Ingeniería en Sistemas Informáticos, se ajusta a los requerimientos técnico-metodológicos y legales establecidos por la Universidad Tecnológica Israel, por lo que se autoriza su presentación.

Cuenca, Noviembre 7 del 2011

Ing. Pablo Tamayo
DIRECTOR DE TESIS

UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMAS INFORMÁTICOS

ACTA DE SESIÓN DE DERECHOS

Yo, **Diana Marisela Brito Zhunio**, declaro conocer y aceptar la disposición de la Normativa de la Universidad Tecnológica Israel que en su parte pertinente textualmente dice: “Forma parte del Patrimonio de la Universidad la propiedad intelectual de las investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”.

Cuenca, Noviembre 7 del 2011

Diana Marisela Brito Zhunio

UNIVERSIDAD TECNOLÓGICA ISRAEL

FACULTAD DE SISTEMAS INFORMÁTICOS

CERTIFICADO DE AUTORÍA

Los contenidos, argumentos, exposiciones, conclusiones son de responsabilidad del autor.

Diana Marisela Brito Zhunio

DEDICATORIA

Este trabajo va dedicado a mis padres Fanny y Fidel por el apoyo brindado, a mis hermanos Erika, Rubí, Karen y Andrés por la comprensión que me tuvieron siempre, a mi precioso sobrino Erick, y en especial a esa persona tan importante en mi vida Edisson, quien ha sido mi fortaleza y mi inspiración para lograr gran parte de mis sueños.

AGRADECIMIENTO

Hoy celebro el fin de una etapa importante en mi vida, es por eso que quiero agradecer a todas las personas que participaron directa e indirectamente e hicieron posible la culminación de este trabajo.

También agradezco el apoyo incondicional de mi familia y amigos por estar siempre alentándome a seguir adelante.

RESUMEN

El término web ha tenido un increíble avance en los últimos años, en la actualidad son pocos los sitios web que ofrecen información estática, la mayoría de ellos ofrecen cierto nivel de dinamismo e interacción con el usuario por ejemplo en fórums, gestores de contenido, suscripciones rss, etc.

Este avance ha provocado que ya no se hable solo de sitios web, sino de aplicaciones web, surgiendo la Web 2.0 con la idea de una web más social dando origen a servicios como Facebook, MySpace, Hi5, Twitter, en otros; en los que la web se apoya en el uso de varias tecnologías combinadas.

El desarrollo de la web avanzó aún más y se acuñaron términos como “Software como Servicio” con nuevos retos para la web, entre ellos el de atender las peticiones de miles y millones de usuarios distribuyendo la carga de trabajo generada en varios equipos para que atiendan atienden estas solicitudes, a esto se le conoce como escalabilidad.

Toda la información generada por las aplicaciones web necesitan almacenarse en un motor de base de datos y es allí en donde se origina la necesidad de escalabilidad ha llevado a grandes empresas como Amazon, Google, Facebook, etc. A desarrollar alternativas a las bases de datos tradicionales y es así como se popularizan una variante de las bases de datos documentales llamadas NoSQL (“not only SQL”) que brindan sobre todo velocidad y escalabilidad. En la actualidad se aplican bases de datos NoSQL como complementos a las bases de datos relacionales tradicionales en empresas como Amazon que vende servicios “en la nube”, Google con su conocida aplicación “Google Maps”, Facebook, etc. Y la lista sigue creciendo día a día.

Es por eso que en el presente trabajo estudiaremos las bases de datos NoSQL y en particular a MongoDB, presentándola como alternativa y/o complemento a las bases de datos relacionales tradiciones especialmente en aplicaciones web.

SUMMARY

The Web had an incredible evolution in recent years, today there are few websites offering just static information, most of them offer some level of dynamism and interaction with the user for example forums, content management systems, rss subscriptions, etc.

This evolution has made that people no longer talk about just websites, but web applications, Web 2.0 emerged with the idea of a more social web, giving rise to services like Facebook, MySpace, Hi5, Twitter, and others; all of these services are supported by combined web technologies.

Terms such as "Software as a Service" had emerged with new challenges for the web, including meeting the requests of thousands and millions of users by distributing the workload generated in various teams to meet address these requests, this is called scalability.

All information generated by web applications need to be stored in a database engine and this is where the need of scalability has led to large companies like Amazon, Google, Facebook, etc. To develop alternatives to traditional databases and become popular as a variant of the document databases called NoSQL ("not only SQL") that provide speed and scalability. Currently NoSQL databases are applied as complements to traditional relational databases in companies like Amazon that sells services "in the cloud," Google with its well-known application "Google Maps", Facebook, etc. And the list keeps growing every day.

That's why in this paper will study the NoSQL databases, MongoDB in particular, presenting it as an alternative and / or complement to relational databases traditions especially in web applications.

TABLA DE CONTENIDOS

LISTA DE CUADROS Y GRÁFICOS

CAPITULO I: INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

¿Permitirá el estudio del uso de MongoDB presentarlo como alternativa a las bases de datos relacionales tradicionales en aplicaciones web que requieren rapidez de lectura/escritura de los datos almacenados?

1.2 SISTEMATIZACIÓN

1.2.1 Diagnóstico

- En nuestro medio las bases de datos NoSQL como MongoDB no son muy conocidas, es decir prima el uso de bases de datos relacionales en proyectos de todo tamaño.
- Cuando se programan aplicaciones web el uso de bases de datos complejas dificulta el desarrollo de las mismas.
- En una aplicación con bases de datos relacionales, para la más simple tarea tendremos 4 o 5 tablas y muchos joins para las consultas.
- El uso de consultas complejas juega en contra del tiempo de carga en las aplicaciones web.

1.2.2 Pronóstico

- El desconocimiento de herramientas como MongoDB y las bases de datos NoSQL limitará al desarrollador a las herramientas tradiciones (bases de datos relacionales), sin dejarlo aumentar su set de herramientas con las cuales proponer soluciones a los problemas planteados, pues MongoDB sería aplicable para muchos proyectos.
- Si el desarrollador necesita un conocimiento profundo del diseño de la base de datos relacional para la que programa su aplicación web, él deberá estudiar el esquema y contar con un sinnúmero de documentación para poder desarrollar, afectando la productividad del programador.
- El uso de múltiples joins en las consultas además de ser complejos de realizar, afectarán negativamente el rendimiento de la aplicación por el trabajo extra que el servidor realizará enlazando los datos, esto repercutirá

en numerosas técnicas que los desarrolladores deberán aplicar para que no afecte el rendimiento de la aplicación y además del lenguaje de programación deberá dominar SQL general y el SQL específico de cada motor de base de datos para el que desarrolle.

1.2.3 Control del Pronóstico

- Conociendo las bondades y limitaciones de MongoDB o las bases NoSQL los desarrolladores pueden evaluar las circunstancias en las que pueden aprovechar de las ventajas que estas tienen sobre las bases de datos relacionales tradicionales en determinados escenarios.
- Cuando el desarrollador usa MongoDB se olvida de complejos esquemas de base de datos pues está tecnología no usa esquemas o tablas simplemente graba la información como el desarrollador lo especifique. MongoDB, de ser aplicable al escenario del proyecto optimizará la productividad del desarrollador.
- Lo que en una base de datos relacional tomaría 10 tablas en MongoDB lo podemos hacer en un solo registro que puede tener atributos simples y complejos como subconjuntos de datos. Al no realizar joins el rendimiento de las consultas y de la base de datos en general aumenta considerablemente.

1.3 OBJETIVOS

1.3.1 Objetivo General

Realizar un estudio del uso de MongoDB como alternativa a las bases de datos relacionales tradicionales en aplicaciones web que requieren rapidez de lectura/escritura de los datos almacenados.

1.3.2 Objetivos Específicos

- Presentar un tutorial de uso general de MongoDB.
- Presentar un tutorial de uso de MongoDB en aplicaciones web con PHP como lenguaje de programación.

- Determinar los parámetros de evaluación para la matriz comparativa de bases de NoSQL.
- Presentar un estudio comparativo de los sistemas de base de datos NoSQL para determinar la mejor opción de ellas para escenarios específicos.

1.4 JUSTIFICACIÓN

1.4.1 Teórica

En la actualidad el uso de MongoDB y demás bases de datos NoSQL se ha popularizado a nivel de las grandes empresas como Amazon, Google, Facebook, Twitter que se han visto en la necesidad de almacenar datos complejos y requerir buenos tiempos de carga de los mismos en sus servicios. Cada día los desarrolladores prueban nuevos escenarios en los que las bases de datos NoSQL como MongoDB funcionan muy bien principalmente en el ámbito de desarrollo de aplicaciones web.

La aplicación de MongoDB no está limitada únicamente a las grandes empresas con millones de usuarios pues su propuesta sería muy buena opción para proyectos pequeños y medianos de aplicaciones en la web.

MongoDB brinda referencias técnicas en cuanto a su uso. Además pone a disposición los drivers para funcionar con distintos lenguajes de programación

1.4.2 Metodológica

El desarrollo de este proyecto se basará en la investigación aplicada, pues presentaremos el uso de la herramienta MongoDB como solución de Base de Datos en Aplicaciones Web.

1.4.3 Práctica

En la vida práctica el uso de MongoDB sería una muy buena alternativa para los desarrolladores web de todo tamaño ya que pueden evaluar y aplicar a sus proyectos en desarrollo, la curva de aprendizaje no sería tan elevada pues el uso de MongoDB se realiza desde el mismo lenguaje de programación.

También se ahorraría tiempo en el desarrollo de las consultas y comandos SQL que las bases de datos tradicionales usan.

1.5 ALCANCE Y LIMITACIONES

1.5.1 Alcance

Este proyecto se presenta como un estudio que busca profundizar los conocimientos acerca de la nueva tendencia tecnológica dada en las bases de datos NoSQL, más específicamente MongoDB y sus aplicaciones en el desarrollo web. Por ende los entregables del presente proyecto serán:

- Tutorial Básico de MongoDB
    - Instalación
    - Configuración
    - Uso General
- Tutorial del uso de MongoDB con PHP
    - Inserciones
    - Búsquedas
    - Actualizaciones
    - Borrado

1.5.2 Limitaciones

Este proyecto se limitará a ilustrar el uso de la base de datos NoSQL MongoDB.

1.6 ESTUDIOS DE FACTIBILIDAD

1.6.1 Factibilidad Técnica

El desarrollo del presente proyecto es factible tecnológicamente pues las herramientas necesarias para su desarrollo están disponles, entre ellas podemos citar:

- MongoDB: Un sistema de base de datos NoSQL Open Source.
- Driver de MongoDB para PHP: El driver necesario para comunicar PHP con MongoDB
- Documentación: Existen algunos libros de MongoDB de Apress y O'Reilly

1.6.2 Factibilidad Operativa

Este proyecto es factible operacionalmente pues tendrá un efecto positivo en la comunidad de desarrolladores que tengan acceso a los tutoriales.

1.2 MARCO DE REFERENCIA

1.2.1 Marco Teórico

Base de Datos

"Una base de datos es un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha base de datos. Por ejemplo, en un directorio de miembros de una asociación, un registro será la ficha completa de cada uno de los socios. En cada registro se recogerán determinados datos, como el nombre, la profesión, la dirección o el teléfono, cada uno de los cuáles constituye un campo." (MALDONADO, Ángeles, 2001)

Un grupo o conjunto de datos organizados que son almacenados en forma documental y no relacional, es decir no en tablas o esquemas predefinidos.

Un conjunto de datos organizados y almacenados pueden brindar soporte para la creación de servicios accesibles desde del Internet a través de un navegador web.

NoSQL (Documental)

"Un SGBDD (Sistema de Gestión de Bases de Datos Documentales) se ocupa de la gestión de documentos optimizando el almacenaje y facilitando su recuperación. A diferencia de cualquier otro SGBD, un SGBDD no realiza ningún tratamiento sobre la información. Simplemente la almacena y posibilita su recuperación." (UNIVERSIDAD DE JAEN, Bases de Datos Documentales)

Un conjunto de información almacenada en forma de documentos y no en un esquema relacional.

Un conjunto de datos organizados y almacenados en forma documental pueden brindar soporte para la creación de servicios accesibles desde del Internet a través de un navegador web, con muchos usuarios que requieren rapidez y escalabilidad.

Aplicaciones Web

"Se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web" (WIKIPEDIA "Enciclopedia Virtual", Línea: 2 http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web)

Los servicios e información a través del internet generalmente se apoyan en el almacenamiento de datos organizados y almacenados para su posterior recuperación o uso.

Los servicios accesibles desde el Internet con muchos usuarios pueden hallar respaldo en almacenamiento basado en documentos, pues supone un mejor rendimiento que los sistemas de base de datos tradicionales.

1.2.2 MARCO ESPACIAL

El presente proyecto será publicado en el Blog científico de tecnología KyoskoTech.

1.2.3 MARCO TEMPORAL

Para desarrollar el proyecto disponemos de 6 semanas, en este tiempo tenemos que cumplir con los objetivos propuestos.

1.3 METODOLOGÍA

1.3.1 Unidad de Análisis

El estudio, y el desarrollo del presente proyecto se lo realizarán sobre el producto de Software "MongoDB".

1.3.2 Tipo de Investigación

Se empleará la investigación exploratoria, pues se realizará sobre MongoDB. Un tema poco conocido y estudiado en nuestro medio. Sus resultados serán documentados como una fuente de conocimiento sobre la herramienta de software mencionada.

1.3.2 Método

Se empleará el método analítico pues estudiaremos MongoDB como un todo dividido en partes: características, instalación, etc. Con el objetivo de conocer más del objeto en estudio.

CAPITULO II: MARCO DE REFERENCIA

2.1 ANTECEDENTES

La web como tal ha tenido muchos avances desde la concepción de una web normal hace unos cuantos años cuando un sitio web se componía únicamente de páginas estáticas con texto y algunas imágenes que se usaban únicamente con fines informativos.

El término Web 2.0 surgió con la idea de una web más social dando origen a servicios como Facebook: Red social creada por Mark Zuckerberg. Inicialmente fue creada para uso de los estudiantes de la Universidad de Harvard, en la actualidad está disponible para todas las personas, y en la mayoría de idiomas, el único requisito es tener una cuenta de correo electrónico y conocimientos básicos de informática.

También uno de los servicios que tuvo mucho auge es MySpace

En nuestro medio Hi5 tuvo éxito temporal, llegando a tener una

En los servicios que mayor número de usuarios ha llegado a tener se encuentra Twitter:

Para lograr implementar dichos servicios, los desarrolladores se apoyaron en el uso de varias tecnologías combinadas como lenguajes de programación orientados a la web entre ellos se encuentran PHP que se ha convertido en el lenguaje de facto para la web; ASP.NET la apuesta de Microsoft para desarrollar para la web; podemos citar a Ruby on Rails que han revolucionado el desarrollo web con su paradigma de programación MVC[1], estos lenguajes se encargan de procesar las solicitudes enviadas desde el navegador web en un servidor; la mayor parte de los servicios que hemos citado se apoyan mucho en JavaScript para mejorar las interfaces web y por ende la experiencia de usuario, recordemos que la web se creó originalmente con el único propósito

[1] **MVC**: http://es.wikipedia.org/wiki/Modelo_Vista_Controlador: “Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos”

de compartir información y no de ofrecer los servicios que actualmente tenemos. Es debido a ese antecedente que la mayor parte de empresas que ofrecen servicios web y aún desarrolladores independientes han colaborado para crear nuevos estándares, tecnologías y herramientas que permitan hacer más dinámica a la web; una de las mejoras más importantes en lo que respecta a la experiencia de usuario es la tecnología llamada AJAX, esta hace posible que los navegadores actualicen solo parte de una aplicación web, de esta manera las aplicaciones funcionan mucho más rápido.

Todo esto apoyado en una base de datos. El desarrollo de la web avanzó aún más y se acuñaron términos como "Software como Servicio" [1] con nuevos retos para la web, entre ellos el de atender las peticiones de miles y millones de usuarios distribuyendo la carga de trabajo generada en varios equipos para que atiendan estas solicitudes, a esto se le conoce como escalabilidad.

La necesidad de escalabilidad ha llevado a grandes empresas como Amazon[2], Google, Twitter, etc. A desarrollar alternativas a las bases de datos tradicionales y es así como se popularizan una variante de las bases de datos documentales llamadas NoSQL ("not only SQL") que brindan sobre todo velocidad y escalabilidad.

En la actualidad se aplican bases de datos NoSQL como complementos a las bases de datos relacionales tradicionales en empresas como Amazon que vende servicios "en la nube", Google con su conocida aplicación "Google Maps"[3], Twitter, etc. Y la lista sigue creciendo día a día.

Entre las bases de datos NoSQL se encuentra MongoDB que estudiaremos en el presente proyecto, aunque en nuestro medio en mayor parte aún se desconoce el uso de las bases de datos NoSQL y por ende de MongoDB, esta herramienta sería una excelente alternativa y complemento a las bases de

[1] **Software como Servicio:** Es un modelo de distribución de software donde el software y los datos que maneja se alojan en servidores de la compañía de tecnologías de información y comunicación (TIC) y se accede con un navegador web a través de internet. La empresa TIC provee el servicio de mantenimiento, operación diaria, y soporte del software usado por el cliente. (WIKIPEDIA) Copiado de: http://es.wikipedia.org/wiki/Software_como_servicio

[2] **Amazon**: http://es.wikipedia.org/wiki/Amazon.com: Es una compañía estadounidense de comercio electrónico con sede en Seattle, Estado de Washington.

[3] **Google Maps**: http://es.wikipedia.org/wiki/Google_Maps: Es un servidor de aplicaciones de mapas en la Web

datos relacionales tradiciones, las posibilidades que su uso nos daría son muchos, especialmente en aplicaciones web.

2.2 BASES DE DATOS RELACIONALES TRADICIONALES Y NoSQL

Generalmente hemos aprendido que los sistemas gestores de bases de datos se clasifican bajo los siguientes criterios:

- **Bases de datos relacionales:** “Es una base de datos que cumple con el modelo relacional, el cual es el modelo más utilizado en la actualidad para implementar bases de datos ya planificadas. Permiten establecer interconexiones (relaciones) entre los datos (que están guardados en tablas), y a través de dichas conexiones relacionar los datos de ambas tablas”
  - **SQL Server:** El motor de base de datos de Microsoft, inicialmente fue adquirido de Sybase por 1989. Con el paso de los años SQL Server ha evolucionado hasta actualmente posicionarse entre las bases de datos más populares. Actualmente se comercializa la versión 2008 de la herramienta. SQL Server funciona únicamente bajo Windows. Se distribuyen versiones comerciales y una gratuita denominada “Express Edition”. Es usada en empresas generalmente de mediano tamaño.
  - **Oracle:** Tuvo su origen en 1979 en la empresa SDL, para con el tiempo convertirse en la base de datos más usada a nivel empresarial. Oracle ofrece el conjunto de herramientas más completo que va desde la base de datos, aplicaciones comerciales, herramientas de desarrollo, herramientas de soporte de decisiones o business intelligence. Con la adquisición de SUN también ofrece soluciones de hardware especializados para explotar al máximo su base de datos. Oracle es multiplataforma, es decir puede funcionar en distintos sistemas operativos y arquitectura de procesadores, como SQL Server Oracle tiene una versión gratuita de su base de datos denominada Oracle Data Base XE.
  - **MySQL:** Es la base de datos más utilizada para sistemas de contenido web, servicios web, etc. Esta base de datos puede gloriarse de ser una de las más instaladas, aunque a nivel empresarial no ha tenido tanta

aceptación. Inicialmente fue una subsidiaria de SUN, pero luego de la adquisición de Oracle ha pasado a ser propiedad del gigante de las bases de datos. MySQL puede ejecutarse en Linux, Windows, Solaris, Mac OS X, BSD. La versión “Comunity Server” e distribuye de forma gratuita, aunque hay productos comerciales como “Enterprise Server”, “MySQL Cluster” entre otros que se comercializan.

  - **PostgreSQL:** Es una base de datos multiplataforma (Linux, BSD, Solaris, Windows, Mac OS X). Generalmente se la ha conocido como la base de datos open source orientada al ámbito empresarial. Algunas empresas lo usan como alternativa a Oracle y se oferta productos basados en PostgreSQL como EnterpriseDB entre otras.

- **Bases de datos orientadas a objetos:** “Las bases de datos orientadas a objetos se diseñan para trabajar bien en conjunción con lenguajes de programación orientados a objetos como Java, C#, Visual Basic.NET y C++. Los ODBMS usan exactamente el mismo modelo que estos lenguajes de programación”

- **Bases de datos relacionales orientadas a objetos:** Proporcionan un modelo de cambio adecuado para los usuarios de las bases de datos relacionales que deseen utilizar características orientadas a objetos.

En la vida práctica la mayor parte de motores de base de datos usadas se basan en la arquitectura o modelo relacional y han estandarizado SQL como lenguaje de consulta y modificación de datos. En esta clasificación tradicional estamos obviando un par de categorías que por tiempo estuvieron sin recibir la atención que merecían:

- **Bases de datos basadas en clave/valor:** Se almacenan valores asociados a una clave. Son sencillas y las de mayor rendimiento.
  - **Cassandra:** Desarrollada inicialmente por facebook y luego entregado a la fundación Apache, Cassandra es un sistema de base de datos distribuida que permite almacenar cantidades muy grandes de información en un entorno distribuido sin punto de fallo, es decir en

sistema de replicación en el que todos los nodos son iguales (desempeñan la misma función) .

- **MemcacheDB:** MemcacheDB es una variante de la herramienta Memcached que es una herramienta construida para acelerar las aplicaciones web almacenando el contenido de una base de datos en memoria pero sin ofrecer persistencia en un medio de almacenamiento; y éste es precisamente el punto diferenciador ya que MemcacheDB si permite grabar los datos en un medio de almacenamiento. Generalmente MemcacheDB se usa en conjunto con la herramienta original Memcached.
- **Google BigTable:** Un sistema de base de datos desarrollado por Google, se basa principalmente en el uso del sistema de archivos GFS[1]. No utiliza un modelo relacional, más bien la podemos definir como un mapa ordenado y multidimensional de datos. Fue desarrollado con el principal objetivo de permitir alta escalabilidad y manejar datos en la dimensión de peta bytes, pudiendo distribuir la carga de trabajo entre cientos e incluso miles de servidores.

- **Bases de datos basadas en documentos:**

  Son una particularización de las clave/valor, en las que el valor puede ser un documento. Permiten consultas complejas.

  - **CouchDB:** Una base de datos documental desarrollada con el objetivo de escalar horizontalmente con facilidad. Una característica especial de CouchDB es que permite enlazarse a la base de datos incluso a través de peticiones http, para ello se basa completamente en el uso de JSON puro. Una diferencia fundamental con las demás bases de datos documentales el que implementa en parte características que garantizan ACID (transaccionalidad o integridad en los datos). También permite la fácil escalabilidad horizontal en entornos distribuidos de fácil manera. CouchDB fue diseñado específicamente para la web, y es debido e este particular que implementa un set completo de funciones como un API

[1] **GFS**: Google File System: Es un sistema de archivo cluster que Google desarrollo para su uso interno

web para realizar operaciones de inserción, actualización y eliminación de datos a través de http.

- **MongoDB:** Una base de datos documental, de alto desempeño, no utiliza esquemas de base de datos. Permite almacenar la información de forma más natural mediante documentos auto contenidos, es decir al no usar tablas con relaciones cada unidad de datos contiene en sí mismo las dependencias necesarias. Para almacenar datos utiliza la forma binaria de JSON denominada BSON. Se distribuye de forma gratuita para Windows, Linux, Mac OS X y Solaris.

Las bases de datos NoSQL se basan en estas dos últimas categorías revisadas, ahora revisemos la definición que **Ian Eure**[1] nos da en su artículo "*Looking to the future with Cassandra*":

NoSQL es un término usado en informática para agrupar una serie de almacenes de datos no relacionales que no proporcionan garantías ACID[2]. Normalmente no tienen esquemas fijos de tablas ni sentencias "join".[3]

En términos más generales el término NoSQL se refiere a sistemas de almacenamiento de información que no cumplen con el esquema entidad-relación, es decir no imponen una estructura de datos en forma de tablas y relaciones entre ellas (no existe un esquema pre-definido de tablas relacionadas), de tal manera que son más flexibles y suelen permitir almacenar información en otros formatos como clave-valor, documentos, etc.

Revisemos un concepto: clustering es una técnica usada para formar un sistema de varios servidores conectados entre sí, que actúan como si fuesen uno. Ésta técnica se realiza para dividir la carga de trabajo entre los servidores que componen el clúster. Se le realiza también para tener tolerancia a fallos haciendo que si alguna parte de nuestro clúster se dañara otro miembro

[1] **Ian Eure,** Ingeniero en DIGG: un sitio web principalmente sobre noticias de ciencia y tecnología. Combina marcadores sociales, blogging y sindicación con una organización sin jerarquías, con control editorial democrático. http://www.opiniontecnologica.com/aplicaciones-informaticas/130-bases-de-datos-nosql-y-escalabilidad-horizontal.html
[2] **ACID**: conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción
[3] EURE IAN, DIGG http://about.digg.com/blog/looking-future-cassandra

tomaría su lugar para continuar con las servicios prestados hasta que el equipo caído esté de vuelta.

Adicionalmente a la carencia de esquemas predefinidos, una de las principales características de las bases de datos NoSQL es que fueron pensadas para trabajar (manipular) con enormes cantidades de datos con tiempos de respuesta muy rápidos; para ello permiten escalar horizontalmente en varios servidores de fácil manera, vale la pena mencionar que el hardware requerido para hacer clustering es muy básico e inclusive se pueden usar PCs domésticas con un rendimiento muy aceptable, es importante mencionar también que se puede añadir nuevas máquinas "en caliente" es decir sin reiniciar todo el sistema.

2.3 ¿POR QUÉ APARECEN LOS SISTEMAS NOSQL?

Primero debemos aclarar que las bases de datos relacionales no tienen nada de malo, es más, vale la pena mencionar que con el avance de la tecnología patrocinada por las grandes empresas que se dedican a las bases de datos como Oracle, Microsoft, IBM, Informix, etc. Se han desarrollado técnicas para escalar sus productos en función a la demanda y uso al que son sometidas dichas implementaciones. Entre las más populares para la web están MySql, PostgreSQL, Oracle, etc.

Una vez mencionado lo anterior podemos analizar la evolución que la web ha sufrido. El factor determinante en la evolución de la web es el propósito que se le ha dado a la misma, por ejemplo:

- Web 2.0 [1] (Enfocado a lo social).
- Software como Servicio[2] (Relacionado con los servicios en la nube o "cloud computing").

[1] **WEB 2.0:** éste término está comúnmente asociado con aplicaciones web que facilitan el compartir información y la colaboración en la web. Ejemplos de la Web 2.0 son las comunidades web, los servicios de red social, las wikis, blogs, etc. (WIKIPEDIA) Copiado de: http://es.wikipedia.org/wiki/Web_2.0

[2] **SOFTWARE COMO SERVICIO:** http://es.wikipedia.org/wiki/Software_como_servicio: Es un modelo de distribución de software donde el software y los datos que maneja se alojan en servidores de la compañía de tecnologías de información y comunicación (TIC) y se accede con un navegador web a través de internet.

Estos dos términos son los que definen las aplicaciones web de hoy en día. Una combinación de las dos ha dado origen a un nuevo negocio los “startups”, su forma de funcionamiento podríamos resumirla de la siguiente manera: los desarrolladores crean ideas que facilitan la vida a los usuarios y los ofrecen como servicios web muchas veces gratuitos, para luego una vez que tengan una base de usuarios aceptable implementar un modelo de negocio basado en la publicidad, etc. Un ejemplo de esto puede ser Facebook que además de la publicidad hace también negocio con las aplicaciones, por ejemplo City Ville. Podemos citar también a Twitter, FourSquare como claros ejemplos. También se valen de otros modelos de negocio como los denominamos servicios web “freemium”, a los que nos suscribimos generalmente de forma gratuita y luego tenemos la posibilidad de contratar adicionales al servicio gratuito por los que si tenemos que pagar, por ejemplo Dropbox (sistema de almacenamiento online) ofrece su servicio de almacenamiento gratuito hasta 2GB y si deseamos más espacio debemos pagar una cuota anual. De esta manera podemos comprender que las aplicaciones web se han convertido en un negocio rentable, para el que necesitamos simplemente buenas ideas.

Una vez entendido lo explicado anteriormente es necesario observar la popularidad que dichos servicios han ido obteniendo, hay algunas ideas que tienen éxito y llegan a tener miles o incluso millones de suscripciones, y otros que no llegan a un nivel considerable de usuarios. El que nos interesa estudiar es el primer caso, donde dichos servicios o aplicaciones web comenzaron a popularizarse y a crecer en número de usuarios llegando hasta millones, las empresas enfrentaron nuevos desafíos para mantener la calidad del servicio. Uno de estos retos es la escalabilidad, si bien los modelos relacionales se pueden adaptar para hacerlos escalar incluso en los entornos más difíciles, sí que es cierto que, a menudo, se hacen cada vez menos intuitivos a medida que aumenta la complejidad. Triples y cuádruples JOINs[1] en consultas SQL, a veces poco eficientes, y sistemas de almacenamiento de resultados en cachés para acelerar la resolución de las peticiones y evitar ejecutar cada vez estas pesadas operaciones, son el pan de cada día en muchos de estos proyectos de software.

[1] **JOIN**: Permite combinar registros de dos o más tablas en una base de datos relacional

Por este motivo principalmente es que las empresas de gran nivel como Facebook, Google, Amazon han creado proyectos aplicando tecnologías de base de datos alternativas a las tradicionales relacionales. Han prestado principal atención a las bases de datos documentales y clave/valor; y, en base a estas han creado sus motores de almacenamiento como por ejemplo: Google creó BigTable, Facebook creó Cassandra, Amazon creó Dynamo[1], etc. Luego de desarrollar dichos proyectos la mayor parte de empresas han liberado el código de sus creaciones como software open Source, el mismo que ha ido mejorado por comunidades de desarrollo y empresas particulares que venden soporte y asesoramiento en el desarrollo de aplicaciones que usen dichas bases de datos.

De esta manera los sistemas NoSQL surgen como una nueva alternativa (basada en tecnología ya existente: bases de datos clave/valor y documentales) para solventar estos problemas proponiendo una estructura de almacenamiento más versátil, aunque sea a costa de perder ciertas funcionalidades como las transacciones que engloban operaciones en más de una colección de datos, o la incapacidad de ejecutar el producto cartesiano de dos tablas (también llamado JOIN) teniendo que recurrir a la desnormalización de datos.

Vale la pena recalcar que las bases de datos NoSQL no buscan reemplazar a las bases de datos tradicionales; es más, la mayor parte de empresas que usan NoSQL han desarrollado aplicaciones híbridas que usan en conjunto bases de datos tradicionales y NoSQL en un mismo ecosistema. Es por este motivo que los ingenieros de la herramienta en estudio MongoDB publicaron lo que han llamado “la filosofía de MongoDB” que se expresa en la frase “*Using the Right Tool for the Right Job*”, estas palabras quieren decir: usando la herramienta indicada para el trabajo indicado. Estas palabras dejan claro que las bases de datos NoSQL no han surgido como reemplazo a las bases de datos relacionales, sino como una herramienta más a considerar cuando desarrollemos un proyecto, es decir que en ciertos escenarios será mejor usar

[1] **DYNAMO**: http://es.zettapedia.com/dynamo-sistema-de-almacenamiento.htm: Es una alta disponibilidad, de propiedad de clave y valor del sistema de almacenamiento estructurado o un almacén de datos distribuidos.

una base de datos tradicional, en otros será mejor una combinación de bases de datos NoSQL con bases de datos tradicionales, y por supuesto habrán casos en los que podamos usar bases de datos NoSQL solas.

Aunque el propósito principal de las bases de datos NoSQL es mejorar los aspectos de escalabilidad, su aplicación no se limita únicamente a los grandes proyectos con millones de usuarios sino que es perfectamente aplicable en escenarios más pequeños principalmente en aplicaciones web.

Si bien, las bases de datos NoSQL no han reemplazado a las relacionales tradicionales, grandes empresas han realizado implementaciones mixtas obteniendo muy buenos resultados.

2.4 PARÁMETROS PARA EVALUAR UNA BASE DE DATOS

Como se mencionó con anterioridad, cuando estamos desarrollando un proyecto, luego de entender los objetivos a cumplir con el mismo, los escenarios en los que nuestro software se usará, las restricciones dadas por el cliente, por el hardware, por el presupuesto, los tiempos de entrega, etc. Debemos hacer un análisis de la mejor herramienta.

Antes de discutir la posibilidad de comparar las bases de datos relacionales y las NoSQL, debemos revisar ciertos conceptos que son básicos que en realidad serían los parámetros a considerar al momento de realizar dicha evaluación. Como hemos mencionado con anterioridad las bases de datos NoSQL constituyen una herramienta con un enfoque diferente a las tradicionales relacionales, de tal forma que revisaremos los parámetros de evaluación agrupándolos bajo tres categorías: los que se consideran en las bases de datos relacionales, las correspondientes a NoSQL, y las que podrían ser comunes a los dos enfoques.

Revisemos primero las que son aplicables a las bases de datos relacionales:

- **Características de Garantía de Integridad:**
  - **ACID**: Es el conjunto de características necesarias para que las transacciones en una base de datos tengan los siguientes atributos:
    - **Atomicidad**: Esta propiedad garantiza que una operación sobre los datos se ha realizado o no, sin dejar que una operación compleja quede a medias. Por ejemplo cuando realizamos una transferencia en el banco en realidad estamos haciendo varias operaciones sobre la base de datos: Bajar el saldo del cliente 1, subir el saldo de la cuenta del cliente 2; no podríamos solamente bajar el saldo del cliente 1 y no acreditarlo en la cuenta del cliente 2.
    - **Consistencia**: Esta propiedad garantiza que las operaciones que se ejecutan sobre los datos no rompan las reglas de integridad de los datos, por ejemplo no podrían existir 2 clientes con el mismo número de cédula.
    - **Aislamiento**: Esta propiedad garantiza que la ejecución de una transacción no afecte a otra, es decir, garantiza que no se pueda ejecutar dos transacciones sobre los mismos datos al mismo tiempo.
    - **Durabilidad**: Esta propiedad garantiza que una vez que se ha realizado una transacción, esta será persistente, es decir, los datos serán guardados en un medio de almacenamiento.

  - **Integridad Referencial**: Esta característica es considerada como un deseable en las bases de datos, gracias a esta característica se garantiza que cada registro de una tabla se relaciona con otra tabla que existe en la base de datos, que los datos sean correctos, que no se repitan datos innecesariamente.

  - **Transaccionalidad**: Esa característica permite que varias instrucciones dadas a la base de datos, formen una unidad no divisible. Es decir en una transacción compuesta por 15 instrucciones no pueden ejecutarse solo 7 y nos las 8 restantes, es decir se ejecutan todas o ninguna. De ésta manera se garantiza la integridad de los datos que se encuentran en varias tablas.

- **Unicode**: El uso de éste estándar permite que la base de datos almacenen contenido de texto de múltiples lenguajes.

- **Características complementarias:**
  - **Triggers**: Está es una característica que permite que un procedimiento (conjunto de instrucciones) sea ejecutado cuando se cumple con una determinada condición. Por ejemplo: en un sistema bancario, cada vez que realizo una transacción (almaceno un registro) mandamos a recalcular el saldo; de ésta manera dicho proceso queda automatizado en la base de datos, la misma que la mandará a ejecutar luego de cada operación de inserción en la base de datos.
  - **Procedimientos Almacenados**: Es un conjunto de instrucciones almacenados en la base de datos bajo un nombre común. Generalmente se utilizan para crear una capa entre los desarrolladores y la base de datos. Podríamos formar toda una interface del lado del servidor de base de datos, lo que supondría un mayor grado de seguridad y facilidad al crear logs de las operaciones en los datos debido a que los desarrolladores tendrían acceso a los procedimientos y no a las tablas directamente, pues de otra manera estaríamos en manos de los desarrolladores quienes deberían programar la creación de logs.

Entre los parámetros que podríamos considerar para las bases de datos NoSQL tenemos:

Finalmente entre los términos que podemos aplicar a los dos tipos de productos podemos mencionar:

- **Información básica:**
  - Nombre Comercial
  - Empresa que mantiene el producto
  - Última versión
  - Licenciamiento.

- **Plataformas:**
  - **Windows**: Es el nombre de la familia de sistemas operativos desarrollados y comercializados por Microsoft. Actualmente cuenta con versiones de escritorio tanto domésticas como profesionales y las que pertenecen al mercado de sistemas operativos para servidores. Windows es el sistema operativo mayormente usado a nivel mundial, debido a que soporta numerosas marcas de hardware. Actualmente Microsoft comercializa la versión denominada “Windows 7” para usuarios domésticos y profesionales, y “Windows Server 2008 R2” para servidores.
  - **Linux**: La palabra “Linux” se utiliza para describir una familia de sistemas operativos que utilizan el núcleo o Kernel que lleva el mismo nombre combinado con las herramientas distribuidas bajo licencia GNU (software de código abierto o libre distribuido son Copyright). Inicialmente se desarrollaron las denominadas distribuciones “padre o principales” entre las que encontramos a Red Hat, Gentoo, Debian; a partir de las cuales han surgido un sin número de distribuciones entre las cuales podemos mencionar como más utilizadas a:
    - **CentOS**: Es la versión empresarial y gratuita de Red Hat
    - **Fedora**: Basado también en Red Hat se lo utiliza generalmente para probar herramientas de software antes de incluirla en la versión oficial de Red Hat
    - **Suse**: Esta distribución está basada en Gentoo, y ofrece versiones para escritorio y servidores dándonos opción a contratar licenciamiento con soporte técnico oficial de la empresa Novell.
    - **Ubuntu**: Esta distribución es una de las más populares en usuarios domésticos, su desarrollo está en manos de la empresa Canonical que financia su avance y la distribuye de manera gratuita.
  - **Free BSD**: Es un sistema basado en Unix desarrollada por la universidad de California en Berkeley. Este sistema operativo se distribuye libre y ha sido la base para otros sistemas operativos como

por ejemplo Mac OS X, inclusive se le ha otorgado el título de “El gigante desconocido entre los sistemas operativos libres”

- **Unix**: Este sistema operativo fue desarrollado a partir de 1969, actualmente es propiedad de Novell. Se utiliza como una certificación a los sistemas operativos que cumplen con la especificación de Novell. Se podría decir que Unix es la base de muchos sistemas operativos, entre ellos las distribuciones de Linux, Free BSD, Mac OS X.
- **Mac OS X**: Este sistema es distribuido por Apple en sus ordenadores. Está basado en Free BSD y por ende en Unix. Este sistema operativo es uno de los más amigables y fáciles de utilizar con los que cuenta el mercado, además ofrece la estabilidad heredada de los sistemas Unix. El hecho de que este sistema operativo solo se distribuye con los equipos fabricados por la marca, ha provocado desinterés en los desarrolladores de virus por lo que no se enumeran muchos tipos de virus para este sistema.

2.5 HERRAMIENTAS NoSQL

Se ha seleccionado escogido las siguientes herramientas dada la popularidad que cada una de ellas goza:

- Cassandra
- CouchDB
- MongoDB

2.5.1 CASSANDRA

El proyecto Cassandra fue abierto al público en el 2008 por Facebook, fue muy influenciada por la base de datos Dynamo de Amazon; Cassandra implementa un modelo de replicación “sin puntos de falla” muy parecido al de Dynamo.

Algunas empresas como Digg, Twitter, Rackspace vieron el potencial de Cassandra y decidieron colaborar con el proyecto y participar con su desarrollo.

2.5.1.1 Características

A continuación vamos a analizar las características básicas de Cassandra:

- **Distribuida y Descentralizado:**

  Esto significa que la base de datos es capaz de ejecutarse en varios equipos de forma transparente para el usuario que se conecta con su “servidor”. Para obtener el beneficio real y ver el verdadero desempeño de Cassandra es necesario que tengamos varios nodos de trabajo. La herramienta desde su misma concepción y desarrollo fue optimizada para obtener un gran desempeño en servidores distribuidos e incluido en Centros de Datos dispersos geográficamente.

  A diferencia de otros motores de bases de datos un clúster de Cassandra es completamente descentralizado, es decir, que cada nodo que conforma el clúster es igual a los demás y no dependen de un nodo principal que organice las operaciones de los demás.

  En cuanto a la replicación Cassandra funciona de manera descentralizada en este aspecto también dándonos como resultado alta disponibilidad por su tolerancia a fallos ya que como se mencionó anteriormente no hay un nodo principal del que dependan los demás, de tal manera que, si uno de los nodos de Cassandra fallara, los demás seguirán desempeñando sus operaciones.

  También debemos mencionar que la configuración de un clúster con Cassandra es muy simple debido a que se configura como si lo estuviéramos configurando en un solo equipo.

- **Escalabilidad:**

  En términos de escalabilidad debemos hacer diferenciación entre: escalabilidad vertical (mejorar el hardware), y escalabilidad horizontal (distribuir la carga de trabajo entre varios servidores). Cassandra ofrece una característica de escalabilidad horizontal denominada “elastic escalability” es decir que puede escalar fácilmente hacia arriba y hacia abajo; en términos más sencillos, basta con agregar un servidor al clúster y Cassandra se encargará de ”ponerlo a trabajar” de tal manera que no es necesario reiniciar el servicio, no hay que cambiar las aplicaciones desarrolladas contra la base de datos, no es necesario volver a equilibrar

los datos (es decir dividir los datos en partes iguales en los nodos). Cuando necesitamos al contrario reducir nodos en nuestro clúster no es necesario reiniciar el servicio, Cassandra lo realiza por nosotros.

- **Alta disponibilidad y tolerancia a fallos:**
  La disponibilidad de un sistema se mide con su capacidad de atender solicitudes, pero debido a que las computadoras en general son susceptibles a sufrir fallas de hardware o podría temporalmente fallar la conectividad de red. Desde luego hay técnicas muy sofisticadas y generalmente costosas, que nos ayudan a solventar en parte estos inconvenientes; estas técnicas generalmente consisten en tener redundancia de hardware, notificaciones de fallas, equipos que permiten agregar, quitar o cambiar elementos “en caliente” es decir sin apagar el sistema, redundancia de los datos en cuestión. La redundancia generalmente se aplica en clústeres y réplicas de los datos, pero los sistemas que utilicen deberán ser capaces de reconocer cuando un nodo falla y enviar las peticiones a otro nodo activo.

- La sencillez con la que Cassandra nos deja configurar un clúster con alta tolerancia a fallos sumando a la opción de replicar los datos en varios centros incluso geográficamente dispersos, hacen de ésta característica uno de los principales atractivos de la herramienta.

- **Consistencia de datos configurable:**
  Antes de hablar sobre esta característica es necesario que comprendamos el teorema CAP, éste teorema se basa en la relación de tres conceptos principales:
  - **(C) Consistencia:** Todos los clientes leerán los mismos datos para la misma consulta, es decir no importa cuántos nodos de trabajo existan, las operaciones sobre los datos deben reflejarse para todos los nodos. Cuando una actualización se realiza en un nodo, los datos quedan bloqueados para lectura/escritura en todos los demás nodos hasta que reciban la última actualización.

- **(A) Disponibilidad:** Todos los clientes siempre podrán leer y escribir datos en los todos los nodos.
- **(P) Tolerancia en Particionamiento:** La base de datos puede ser dividida en varias máquinas, y podrán seguir funcionando si parte de la red falla.

El teorema de CAP dice que siempre un sistema de base de datos puede escoger cumplir solo dos de las características mencionadas con anterioridad. De tal forma que:

- Los sistemas relacionales se sitúan entre C y A: generalmente el sistema en entornos distribuidos (clústeres) cuando se de una actualización los datos estarán bloqueados (no disponibles) mientras no se den los cambios en todos los nodos de trabajo.
- Cassandra se ubica entre A y P: su principal objetivo es tener siempre los datos disponibles a pesar de fallas en los sistemas; y, aceptan que los datos no estén actualizados por milisegundos.

A pesar de que Cassandra se apegue más a AP según el teorema de CAP, nos brinda la posibilidad de configurar el nivel de consistencia a través de dos parámetros:

- **(RF) Factor de Replicación:** Este parámetro se refiere a el número de nodos en un clúster a los que deseamos propagar una operación en la base de datos (inserciones, actualizaciones, eliminaciones). En resumen nos indica cuando estamos dispuestos a pagar en desempeño para ganar consistencia.
- **(CL) Nivel de Consistencia:** Este parámetro permite que el usuario decida cuantas réplicas en un clúster deben actualizarse en una operación sobre los datos para que la operación sea considerada un éxito.

Mediante el uso de los parámetros anteriores podemos configurar el nivel de consistencia que deseamos, por ejemplo si queremos un alto nivel de consistencia igualaríamos el CL al número ingresado en el RF, pero nos

costaría mucho en el nivel de desempeño, pues cuando se actualice un nodo al igual que en las bases relacionales tendremos que esperar hasta que todos los nodos se actualicen para poder usar los datos en cuestión otra vez. Si configuramos con un alto nivel de consistencia perderíamos el enfoque de Cassandra que es son: rapidez y alta disponibilidad; debido a esto casi nunca encontraremos una configuración de Cassandra que implemente un alto nivel de consistencia.

- **Modelo de Datos**:

  Recordemos que Cassandra es una herramienta desarrollada bajo el enfoque NoSQL, es decir no tenemos esquemas predefinidos como en las tablas. Cassandra se basa en el modelo Llave/Valor. Para entender mejor el modelo de datos examinemos el siguiente gráfico:

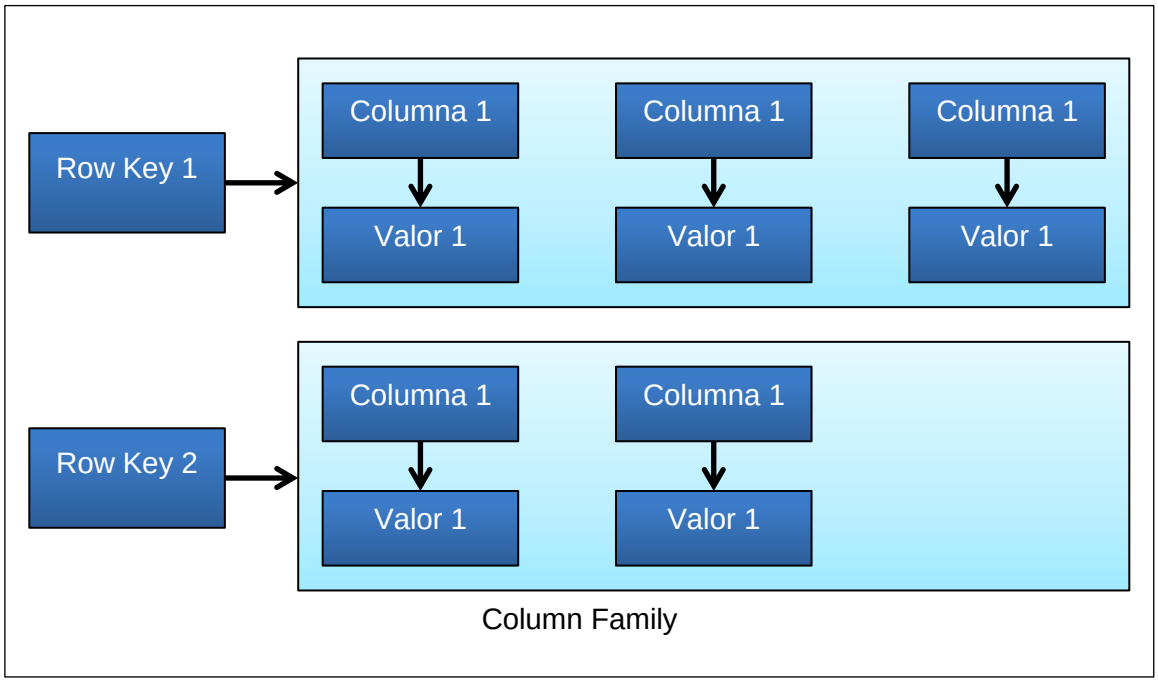

Ilustración 1 - Representación del Modelo de Datos Básico de Cassandra

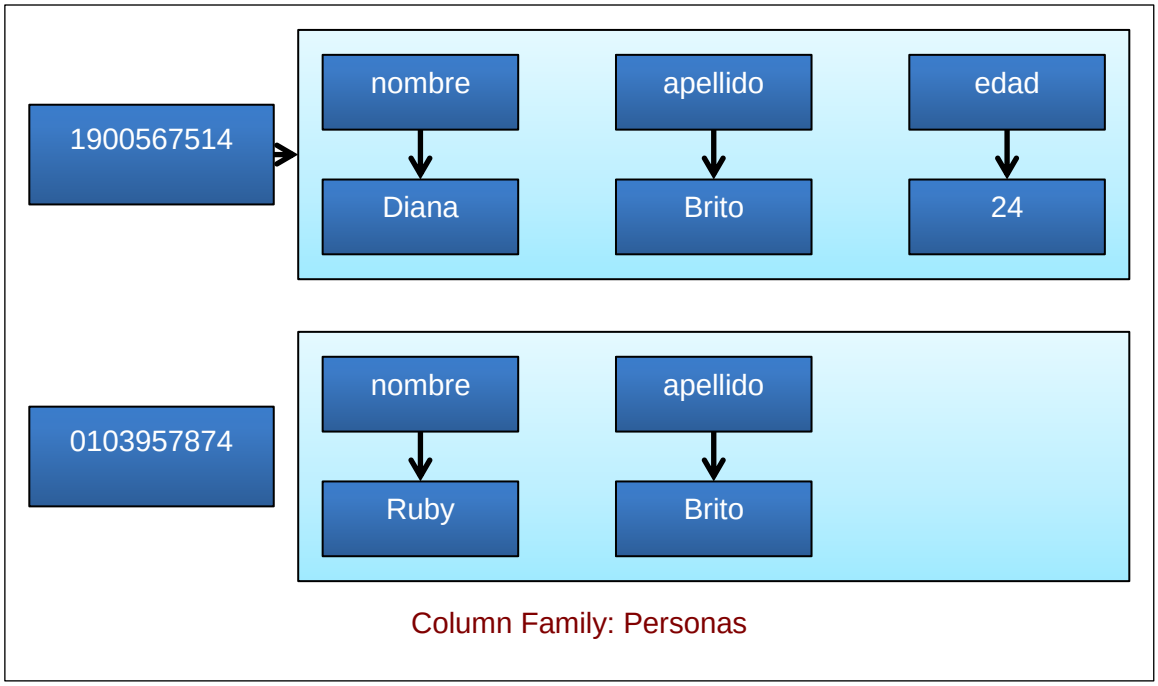

Ilustración 2 - Representación del Modelo de Datos Básico de Cassandra con datos

Analicemos los conceptos básicos para comprender el modelo de datos de Cassandra:

- **Columna:** el par llave/valor para almacenar los datos.
- **Fila:** el conjunto de columnas.
- **Row Key:** El identificador único de la fila.
- **Column Family:** El conjunto de filas que tienen datos semejantes, no necesariamente las mismas columnas.

Column Family sería el análogo a una tabla en una base de datos relacional, las filas de un mismo Column Family no necesariamente tienen que tener las mismas filas, por ejemplo en un Column Family para almacenar personas, una fila podría tener los datos: cédula, apellidos, nombres y teléfono fijo; pero otra fila podría tener cédula, apellidos, nombres. Recordemos que en las bases de datos NoSQL no dejamos campos en NULL como en las bases relacionales, simplemente grabamos los datos que disponemos.

Si deseamos almacenar subconjuntos de datos en Cassandra, la herramienta pone a disposición nuestra un tipo especial de Column Family denominado Súper Column Family, que nos permite tener como valor un conjunto de columnas.

Para crear el esquema básico de una base de datos se puede definir mediante el formato YAML para luego cargarlo en el motor de base de datos, o simplemente se generará desde el cliente cuando realicemos la primera inserción.

- **Modelo de Datos Libre de Esquemas**:
  Cassandra necesita únicamente que definamos los Column Family, que en realidad serían los contenedores de datos. Además debemos definir los denominados keyspace, que en realidad son espacios de nombres para englobar conjuntos de Column Family.

- **Alto Nivel de Desempeño:**
  Cassandra ha sido concebido desde sus inicios mismos como una base de datos de alto desempeño para aprovechar al máximo la potencia de los equipos que tienen multiprocesadores, o procesadores multinúcleo. Además se consideró desde sus diseños mismos para escalar con facilidad y manejar altos volúmenes de datos (cientos de terabytes).

2.5.2 COUCHDB

CouchDB es un sistema de base de datos documental orientado a documentos, que se distribuye bajo licencia Open Source. CouchDB no utiliza esquemas predefinidos como las bases de datos relacionales tradicionales, es decir no tendremos tablas, columnas, llaves primarias, llaves foráneas, joins, relaciones, etc. En lugar de ello CouchDB almacena los datos como documentos.

CouchDB se diseñó con el objetivo de ser altamente escalable y a bajo costo, pudiendo usarse servidores de bajo coste. A diferencia de las bases de datos relacionales tradicionales en las que crear un clúster o replicación no es una tarea sencilla, CouchDB busca ofrecer al igual que la mayor parte de bases de datos NoSQL Replicación y Clústeres sencillos de configurar, pero garantizando alta fiabilidad y disponibilidad.

CouchDB comenzó a ser desarrollado en el año 2005 por Damien Krats, quién previamente había trabajado en el proyecto de Lotus Notes[1]. Inicialmente fue desarrollado en C++, pero luego se migró toda la herramienta a Erlang, un lenguaje de programación creado por la empresa de telecomunicaciones Ericsson.

El equipo de trabajo de CouchDB pensó en la web como principal objetivo para su herramienta, por lo que en el año 2006 el equipo de desarrollo anunció que CouchDB contaría con un API basada en http, es decir que para acceder a los datos y realizar operaciones sobre ellos, podríamos usar cualquier herramienta que soporte http. Esta acción incrementó notablemente el rendimiento de la base de datos. De esta manera los desarrolladores cuentan con dos alternativas para conectarse a la base de datos: mediante URLs o mediante un lenguaje de programación que soporte trabajar con peticiones http.

2.5.2.1 Características

- **Base de Datos Basada en Documentos:**
  CouchDB no utiliza esquemas predefinidos como las bases de datos relacionales en las que antes de ingresar los datos debemos especificar un esquema denominado tabla, en la que creamos las columnas o campos necesarios; en lugar de ello CouchDB almacena los datos en forma de documentos, que contienen valores basados en clave/valor.

  Cuando creamos una colección de documentos (el equivalente a una tabla en un sistema tradicional) la herramienta no exige que cada documento cumpla estrictamente un esquema predefinido, sino que cada documento será independiente el uno del otro, aunque sean semejantes, por ejemplo en la colección Personas puedo tener un documento con los atributos: cedula, nombres, apellidos; y, puedo tener un segundo documento que tenga: cedula, nombres, apellidos y fechas de nacimiento. De esta manera los documentos de CouchDB son dinámicos. En las bases de datos documentales debemos olvidar los campos en NUL.

[1] **Lotus Notes**: Es un sistema cliente/servidor de colaboración y correo electrónico

Para identificar cada documento CouchDB utiliza un identificador único denominado ID, dicho identificador puede ser ingresado automáticamente mediante el aplicativo en desarrollo, o dejar que CouchDB lo haga por nosotros.

A diferencia de una base de datos relacional, las dependencias de un documento en CouchDB están contenidas en el mismo documento, por ejemplo consideremos el documento:

Persona: [nombre:[nombre], apellidos:[apellidos], teléfonos[móvil:[móvil], celular:[celular]]]

En el anterior ejemplo podemos observar que el documento posee un subdocumento para los teléfonos. De esta manera CouchDB se encarga de almacenar las dependencias de cada documento en sí mismo, sin dividir los datos en 2 o más tablas como es común.

- **Documentos en CouchDB:**
  Los datos en CouchDB consisten en una serie de documentos, cada uno de estos documentos están formados por campos (llave/valor); los valores almacenados pueden ser cadenas, números, fechas, etc. E incluso pueden ser un subdocumento que a su vez puede tener subdocumentos. Adicionalmente CouchDB mantiene metadatos de cada documento, estos metadatos son gestionados por la misma herramienta y generalmente consisten en un versionado de los cambios realizados en dicho documento. Cuando implementamos un clúster de CouchDB, la herramienta en cuestión no bloquea los datos como las bases relacionales; recordemos que en un clúster de una base de datos relacional cuando se realiza una operación sobre los datos (inserciones, actualizaciones, eliminaciones), los datos quedan bloqueados a lectura/escritura hasta que los nodos del clúster sean actualizados. CouchDB enfrenta este problema de otra manera, si dos usuarios editan los mismos datos al mismo tiempo, el primero en intentar su actualización lo conseguirá, y mientras que el usuario en segundo lugar recibirá una advertencia, cuando se de esta advertencia el usuario que

quedó en segundo lugar recibirá la versión más actualizada de los datos y tendrá oportunidad de realizar su actualización otra vez. CouchDB cuenta con un poderoso sistema de versionado que almacenará en un historial los cambios realizados a un documento.

CouchDB implementa las características necesarias para garantizar ACID, y de esta manera asegurar la integridad de los datos.

- **Motor de vistas JavaScript:**
  Debido a que CouchDB no utiliza esquemas predefinidos los datos son altamente no estructurados; aunque esto sea favorable en el aspecto de poder cambiar (agregar, cambiar) campos, no es tan favorable a la hora de realizar reportes o vistas de los datos.

  Afortunadamente CouchDB ofrece un motor de vistas basado en JavaScript que nos permite crear vistas con agregados, y de esta manera crear los reportes necesarios de la base. Aunque las vistas no son almacenadas en la base de datos, los resultados son generados cada vez que realicemos una consulta a través de la vista.

- **API RESTful http:**
  Para acceder y trabajar con bases de datos tradicionales relacionales generalmente debemos implementar las interfaces de acceso en un lenguaje de programación utilizando el respectivo driver provisto por la base de datos en combinación con instrucciones SQL, o podemos usar un ORM[1] que nos permita “mapear” la base de datos y trabajar con ella en lenguaje nativo.

  CouchDB hace las cosas un poco diferente, REST se refiere al hecho de que podemos acceder a los contenidos de la base de datos y operar sobre

[1] ORM: http://es.wikipedia.org/wiki/Mapeo_objeto-relacional: Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia

ella mediante una serie de servicios web que se implementan mediante el protocola HTTP.

CouchDB configura las APIS de acceso para operar sobre los datos, los mismos que son transferidos en formato JSON. Una vez que tenemos los datos JSON en nuestra aplicaciones debemos procesarlos para visualizarlos en nuestro aplicativo, o caso contrario convertir los datos ingresados en nuestro aplicativo para enviarlos en formato JSON al servidor de base de datos.

- **Clustering y Alta Disponibilidad:**

  CouchDB utiliza su misma API RESTful para gestionar la replicación en clústeres. CouchDB es capaz de sincronizar los cambios realizados en los datos en los nodos involucrados, así como de balancear la carga de trabajo, pero CouchDB implementa adicionalmente una característica para que cuando por algún problema perdamos la conectividad (red) entre los nodos de trabajo y el proceso de sincronización se vea interrumpido, cuando los nodos en cuestión estén “en línea” la operación de sincronización continuará desde el punto en el que fue interrumpida. Cada nodo del clúster es independiente el uno del otro, y los datos se sincronizan periódicamente.

- **Administrador Web “FUTON”:**

  CouchDB incluye de serie una herramienta en forma de una aplicación web desde la cual podemos conectarnos a nuestro servidor y realizar tareas básicas de administración sobre nuestra base de datos.

- **Plataformas Soportadas:**

  CouchDB puede ser instalado en las siguientes plataformas:

  - Linux
  - FreeBSD
  - Unix
  - Solaris
  - Mac OS X
  - Windows (Beta)

2.5.3 MONGODB

Para el estudio y desarrollo del presente proyecto se ha escogido a MongoDB como herramienta de base de datos NoSQL.

Los desarrolladores de MongoDB decidieron no tratar de crear un producto que abarque todas las necesidades de todos los usuarios, en lugar de ello se propusieron crear una base de datos basada en documentos en lugar de filas, que fuese rápida, que escalara fácilmente y que fuese fácil de usar. Para lograr los objetivos antes mencionados los desarrolladores de MongoDB se vieron en la necesidad de prescindir de algunas características, eso hace que MongoDB no sea la mejor opción para todos los escenarios, por ejemplo no podríamos usar la herramienta para una aplicación contable debido a su carencia de transacciones, en este caso podríamos usar una base de datos relacional tradicional para almacenar los datos contables y usar MongoDB para almacenar documentos por ejemplo o datos más complejos; este tipo de soluciones "híbridas" que usan varios tipos de base de datos pueden ser comunes, por ejemplo Facebook, Sourceforge[1], etc.

Aunque MongoDB no solucione todos los problemas, puede ser excelente en los casos que ameriten sus funcionalidades, por ejemplo para realizar analítica de datos (analytics)[2].

Mongo DB es multiplataforma con versiones para Windows, Linux, Mac y Solaris. Otro concepto clave de MongoDB es la disponibilidad, desde su arquitectura MongoDB pensó en clustering[3] para la recuperación de desastres; y la escalabilidad horizontal para garantizar el rendimiento en aplicaciones que así lo requieran. Debido a estas razones recalcamos que aunque MongoDB no sea la mejor opción para todos los escenarios, definitivamente puede ser una gran herramienta para ciertas soluciones, especialmente aplicaciones Web.

---

[1] **SOURCEFORGE**: Es un servicio web que permite publicar proyectos de desarrollo de software

[2] **ANALYTICS**: Combinar los datos generados en un sistema transaccional, para posteriormente analizarlos a detalle y sacar conclusiones beneficiosas para el negocio

[3] **CLUSTERING**: Se aplica a los conjuntos o conglomerados de computadoras construidos mediante la utilización de hardwares comunes y que se comportan como si fuesen una única computadora.

Podemos recalcar que el hecho de la falta de soporte para transacciones no debería asustarnos, MySQL con su motor MyIsam[1] que es el estándar de facto para las aplicaciones web y en hostings compartidos que la mayor parte de desarrolladores usan para sus aplicaciones, carece de transaccionalidad también, y sin embargo tiene la popularidad que goza hoy en día. El sacrificio de la transaccionalidad ha sido debido al esfuerzo por mantener a MongoDB simple y rápido. Normalmente cuando hablamos de escalabilidad tenemos dos definiciones básicas: Escalabilidad Vertical y Escalabilidad Horizontal; normalmente los sistemas de base de datos relacionales mejoran su rendimiento haciendo más potente el hardware en el que dichos motores corren (escalabilidad vertical), en lugar de ello MongoDB busca que la carga de trabajo sea repartida en equipos menos potentes pero todos trabajando en paralelo.

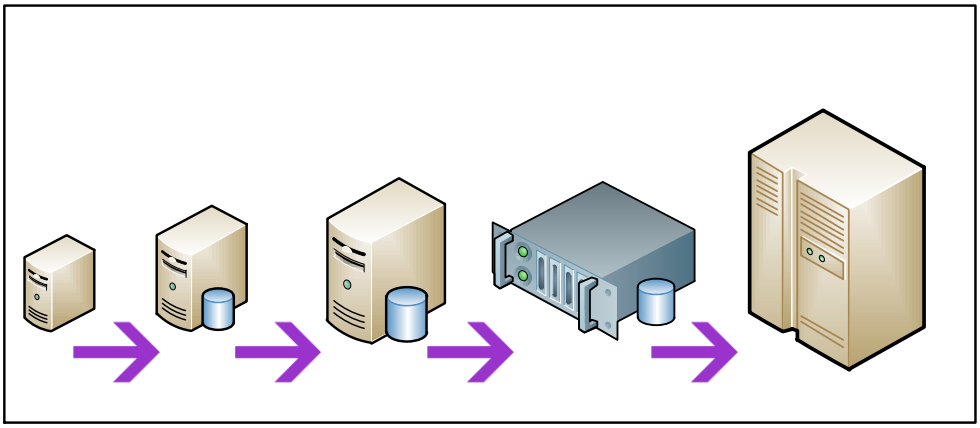

Ilustración 3 - Ejemplo de Escalabilidad Vertical

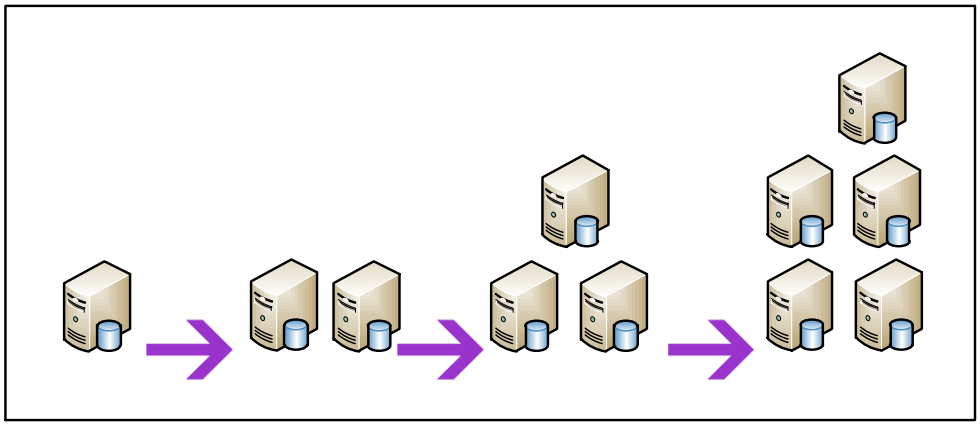

Ilustración 4 - Ejemplo de Escalabilidad Horizontal

[1] **MYISAM**: Es la tecnología de almacenamiento de datos usada por defecto por el sistema administrador de bases de datos relacionales MySQL

2.5.3.1 Características

- **Enfoque no relacional**

  Aunque MongoDB haya sido concebido como una solución enfocada a la escalabilidad, no debemos olvidar que la sencillez y facilidad de uso son dos puntos clave de su diseño. Debemos recalcar que MongoDB no solo tiene aplicaciones de la escala de Facebook o Twitter sino que también podría ser una excelente herramienta para soluciones de menor tamaño, o un complemento perfecto a las bases de datos relacionales tradicionales; esto se ha dicho debido a que el enfoque no relacional tiene mucho que ver con la escalabilidad de la herramienta, pero no debemos tomar esta característica como la más importante al momento de evaluar sus aplicaciones en proyectos de menor envergadura. Una vez aclarado aquello, podemos analizar por qué los creadores de MongoDB adoptaron un enfoque no relacional.

  El hecho de incrementar el rendimiento de las bases de datos relacionales es usualmente sencillo, compramos un servidor más grande y potente. Esta solución funcionará bien hasta que lleguemos al punto en el que no existe un servidor más potente que el que tenemos actualmente. En esta situación la única alternativa es tener dos o más servidores. Esto puede sonar fácil pero debemos saber que ni MySQL ni PostgreSQL pueden implementar clustering del tipo active/active, es decir correr una base de datos en dos servidores, en la que cualquiera de ellos pueda leer o escribir datos. MySQL se basa más en balancear la carga de trabajo. El clustering es más complejo debido a que cuando consultamos a la base de datos, ésta tiene que encontrar toda la información relevante a la consulta que hicimos (generalmente dividido en tablas) y enlazar todo junto para darnos una respuesta. Los sistemas de base de datos tradicionales se basan en características que para mejorar el desempeño de su motor necesitan tener una imagen total de los datos en cuestión; este enfoque no funciona cuando tenemos la mitad de los datos en un servidor y segunda mitad en otro adicional.

Vale la pena recalcar que Oracle puede realizar clustering de tipo activo/activo con su muy impresionante solución llamada “Real Aplication Clustering” más conocida como RAC[1]; desafortunadamente una solución de este tipo involucra costos extremadamente elevados.

El hecho de dividir la carga de trabajo en MySQL y PostgreSQL conlleva algunos inconvenientes adicionales, por ejemplo cuando dividimos la carga de trabajo debemos asegurarnos que los datos escritos en el servidor uno estén disponibles para el segundo servidor, y si las actualizaciones se realizan en dos o más servidores master simultáneamente debemos determinar cuál de las actualizaciones es la correcta. Toda esta serie de inconvenientes nos llevan a considerar el porqué del elevado precio de RAC de Oracle que resuelve todos estos problemas que son tan difíciles de tratar.

MongoDB es inmune a dichos inconvenientes, pues recordemos que los datos son almacenados en BSON, de tal manera que los datos son “auto contenidos”, es decir cada documento tiene sus dependencias dentro del mismo, por lo que todas las operaciones se simplifican al no tener los datos divididos en diferentes tablas. Los documentos similares se almacenan conjuntamente. Aunque MongoDB no ofrezca de fábrica replicación Master/Master en la que los dos servidores reciban actualizaciones, ofrece sharding que permite dividir los datos en dos o más servidores, cada una de estas máquinas es responsable de actualizar las diferentes partes del conjunto de datos. El beneficio de esta característica es que mientras algunas soluciones ofrecen tener dos masters en sus servidores, MongoDB puede potencialmente escalar en cientos de servidores tan fácilmente como en dos de ellos.

- **JSON Y BSON**

  Para continuar con el análisis de MongoDB es necesario mencionar que dicha herramienta para almacenar datos utiliza un formato denominado

[1] RAC (Real Aplication Clustering): http://www.oracle.com/es/products/database/options/rac/index.html: Permite ejecutar una sola base de datos en un grupo de servidores y proporciona una tolerancia a fallos, un rendimiento y una capacidad de ampliación inigualables, sin necesidad de cambios de aplicaciones

BSON, antes de entender cómo funciona BSON es necesario hablar de un formato más conocido para los desarrolladores web: JSON[1]; JSON es una gran alternativa para el intercambio de datos pues provee una rica y expresiva forma de almacenarlos. Dado que JSON describe cada parte de contenido que compone un documento, no es necesario especificar una previa estructura. Si bien es cierto MongoDB no utiliza JSON para almacenar los datos, pero un su lugar usa una variante del mismo desarrollado por los ingenieros de MongoDB al que denominaron BSON, o JSON Binario.

```
{
        "apellidos":    "Brito Zhunio";
        "nombres":   "Diana Marisela";
        "números":
        [
                { "fijo":          "072803506"},
                { "móvil":        "090040126"},
        ]
}
```

En el ejemplo de JSON podemos ver que cada documento puede tener n niveles de "subdocumentos", es decir por ejemplo el documento perteneciente a Diana Marisela Brito Zhunio tiene un subdocumento denominado números con sus respectivos atributos.

- **LLAVE / VALOR**

  En MongoDB un "documento" es la unidad básica de almacenamiento, en una base de datos relacional su análogo sería una "fila". Desde luego un documento es mucho más que una fila común, debido a que puede almacenar información compleja como listas, diccionarios y aún una lista de diccionarios; en contraste con las bases de datos relacionales tradicionales en la que cada "fila" es estática, cada documento puede tener un número ilimitado de llaves y valores. Una llave es una etiqueta que describe el dato, su equivalente sería el nombre de una columna en una base de datos relacional.

[1] **JSON**: Es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML

En las bases de datos relacionales usamos un campo único especial al que denominamos Llave Primaria[1] (PK) que nos permite identificar a cada registro de manera única. MongoDB también usa la misma definición bajo el nombre _id. A menos que especifiquemos un valor para dicho dato MongoDB generará automáticamente el id del documento. La decisión de usar o no el id automático dependerá de cada desarrollador o del escenario en el que se aplique.

Para comprender mejor el funcionamiento del modelo clave/valor usaremos el siguiente ejemplo:

Supongamos el documento de una persona:

```
{
    "apellidos":    "Brito Zhunio";
    "nombres":    "Rubí Adriana";
    "números":    [
                    "072803525",
                    "099510976"
        ]
}
```

Como podemos observar en el ejemplo las llaves son las etiquetas que definen un dato, por ejemplo: "apellidos".

Las llaves y valores siempre se usan en pares; a diferencia de los sistemas relacionales tradicionales que si una columna no tiene valor es necesario dejar NULL, los documentos de MongoDB no necesitan tener un esquema común. Por ejemplo si no sabemos el número telefónico de una persona simplemente no lo incluimos. Esta política se explica mejor en el siguiente ejemplo: en una tarjeta de presentación profesional no podríamos tener FAX: ninguno, simplemente no lo pondríamos.

- **ALMACENAMIENTO ORIENTADO A DOCUMENTOS (BSON)**

  Ya hemos hablado un poco de BSON, mediante los ejemplos usados hemos comprobado que JSON hace mucho más fácil almacenar y recuperar datos en su "forma real" eliminando la necesidad de mapear los

[1] **Llave Primaria**: Es un conjunto de uno o más atributos de una tabla

resultados a los objetos del lenguaje de programación. BSON es un estándar abierto. BSON es la forma binaria de JSON, esto hace que a nuestra mente venga la idea de que BSON ocupa menos espacio que JSON (texto plano), pero en realidad BSON ocupa un poco de espacio más que JSON.

Generalmente las personas se preguntan por qué los Ingenieros de MongoDB decidieron usar BSON en lugar de JSON puro como CouchDB por ejemplo; son dos los principales motivos por los que se realizó dicha selección:

- Primero, debemos recordar que el enfoque de MongoDB es la velocidad, dicho esto debemos saber que BSON hace mucho más fácil el procesamiento de los datos y se indexa[1] mucho más rápido de JSON. Aunque JSON sea más eficiente en el uso del espacio de almacenamiento, la diferencia no es tan considerable, además la ganancia en rendimiento hace que valga la pena gastar unos cuantos bytes adicionales.
- Segundo, es mucho más fácil y rápido convertir BSON a su representación nativa en los lenguajes de programación soportados. Si los datos estuvieran almacenados en JSON puro necesitaríamos una conversión más compleja a los lenguajes de programación. MongoDB provee de drivers para algunos lenguajes de programación, entre ellos: Python, Ruby, PHP, C, C++, Java, C#, etc. Estos lenguajes funcionan diferente cada uno del otro, de tal manera que usando un simple formato binario (BSON) se puede construir representaciones nativas de los datos en cada uno de los lenguajes soportados. Los resultados son sencillez y rapidez, ambas metas de MongoDB.

Debemos mencionar también que aprender el formato de BSON no supondrá ninguna dificultad para los desarrolladores, pues MongoDB se encargará de gestionarlo y abstraerá este trabajo en el driver específico para cada lenguaje, de tal manera que los desarrolladores trabajaran en su lenguaje nativo.

[1] **INDEXAR**: http://es.wikipedia.org/wiki/%C3%8Dndice_(base_de_datos): Permite un rápido acceso a los registros de la tabla

Cuando tenemos un conjunto de documentos formamos una Colección, las colecciones en MongoDB son análogas a lo que las tablas son en una base de datos relacional tradicional, pero mucho más flexibles. Una colección es como una caja etiquetada, por ejemplo en casa tal vez tengamos una caja con una etiqueta que dice “revistas”, pero nada impide que dentro de ella también coloquemos revistas sino diarios o libros si lo deseamos. En una base de datos tradicional, las tablas están estrictamente definidas y debido a esto únicamente podremos ingresar los datos que ese diseño nos permita.

MongoDB tiene la capacidad de crear las colecciones “bajo demanda”, es decir cuando intentamos crear un documento que hace referencia a esa colección. En las bases de datos tradicionales también podemos crear la base de datos desde nuestra aplicación, pero en MongoDB lo realizamos de una forma muy fácil. Vale la pena recalcar también que MongoDB nos permite definir las colecciones desde el administrador de la base de datos.

Finalmente una base de datos en MongoDB es decir que es una colección de colecciones. Así como en las colecciones, la herramienta nos da la facilidad de crear la base de datos “bajo demanda” o de la manera tradicional; como siempre MongoDB deja a nuestro criterio cuál de los métodos usar.

- **CONSULTAS DINÁMICAS**

  Soportar consultas dinámicas quiere decir que la base de datos pude ejecutar consultas que no han sido previamente definidas o planificadas. Esto es similar al hecho de ejecutar consultas SQL en un motor de base de datos relacional. Esta característica está presente en todas las bases de datos relacionales, recordemos que dichas herramientas tienen el esquema de datos definido en sus tablas, lo cual hace realmente simple ejecutar consultas dinámicas; no así en las bases de datos documentales, en las que los documentos de una misma colección pueden variar en estructura, y es precisamente este aspecto el que dificulta el soporte de consultas dinámicas.

A pesar de la dificultad de soportar consultas dinámicas debido a la falta de estructuración de los datos, MongoDB a diferencia de otras bases de datos NosQL soporta consultas dinámicas sin mayor dificultad, permitiendo que dichas consultas no necesiten ser predefinidas antes de usarlas.

- **SEGUIMIENTO DE CONSULTAS.**
  MongoDB incluye de serie una herramienta que permite monitorizar cómo se comporta una determinada consulta, de tal manera que si tenemos una consulta que se ejecuta demasiado lento, podremos hacer seguimiento para optimizarla.
- **ACTUALIZACION EN EL"LUGAR ESPECÍFICO"**
  La mayor parte de base de datos al realizar actualizaciones utilizan el enfoque de "Control de Concurrencia de Multiversiones" más conocido como MVCC. Este enfoque versiona los cambios realizados en los datos. MongoDB no utiliza este enfoque sino que actualiza la información final, es decir no la versiona. MongoDB utiliza el concepto de "lazy writes" que escribe en memoria primero los cambios y luego los escribe en la memoria persistente (disco duro), por ejemplo un escenario poco sería un contador de visitas que se actualiza varias veces por segundo, los cambios se realizan en memoria del servidor de base de datos y cada segundo este escribe los cambios en el medio de almacenamiento.

  CouchDB por ejemplo tiene un muy buen sistema de versionado de datos, y escribe las actualizaciones cada vez que estas se dan, esto reduce el desempeño pero asegura que los cambios han sido realizados ye están grabados en el medio de almacenamiento. En MongoDB cuando utilizamos esta característica podemos en casos extremos sufrir la perdida de los datos en memoria por un corte de energía por ejemplo.
  Esta característica se la puede considerar una ventaja desde el punto de vista del rendimiento, pero una desventaja para información extremadamente delicada como contabilidad, facturación entre otros.

- **ALMACENAMIENTO DE DATOS BINARIOS**
  MongoDB utiliza GridFS para almacenar datos binarios de gran tamaño. BSON soporta datos de hasta 4MB en un documento, lo cual para

necesidades comunes como una imagen, un clip de audio corto es suficiente; pero si necesitamos almacenar datos como clips de video, audio de alta calidad o archivos de muchos megas, MongoDB utiliza GridFS para satisfacer dichas necesidades.

GridFS trabaja grabando los metadatos[1] del dato en cuestión en la colección llamada “files collection”, el dato como tal es dividido en pequeñas partes denominadas “chunks” y son almacenados en la colección llamada “chunks collection”. De esta manera grabar datos grandes (binarios) es muy sencillo, rápido y soporta alta escalabilidad, incluso podemos recuperar solo partes del archivo.

Como en las demás características de almacenamiento GridFS y sus funciones son gestionadas y abstraídas en el driver del lenguaje de programación que usemos, de tal manera que no tendremos que “ensuciarnos las manos” trabajando a bajo nivel para almacenar este tipo de datos, MongoDB nos tiene cubiertos, además GridFS ha sido diseñado para garantizar velocidad, sencillez, y escalabilidad.

- **REPLICACIÓN**

  La seguridad de los datos es importante, MongoDB nos permite realizar réplicas de tipo maestro/esclavo para asegurar la facilidad de recuperarnos y tener disponible la base de datos, si nuestro servidor fallara. La replicación de tipo maestro/esclavo consiste en tener un servidor habilitado para actualizaciones, luego de que las actualizaciones son realizadas en el maestro, los nodos o esclavos son actualizados, de tal forma que si el maestro falla un esclavo podría tomar su lugar hasta que éste entre nuevamente en funcionamiento.

- **AUTO SHARDING**

  Para quienes desean escalar en alto nivel, la función de auto sharding es uno de los principales atractivos de MongoDB. Aunque la mayor parte de

[1] **Metadatos:** http://antares.inegi.org.mx/metadatos/metadat1.htm**:** Son datos altamente estructurados que describen información, describen el contenido, la calidad, la condición y otras características de los datos

usuarios se sentirán satisfechos con las características de réplica, la función de sharding permitirá escalar en muy alto nivel.

La función e auto sharding permite que la base de datos sea dividida en dos o más servidores de forma transparente para el desarrollador quien programará como si la base de datos estuviera en un solo servidor; MongoDB se encargará de actualizar la parte adecuada de los datos en el debido servidor, y de ensamblar los datos cuando realicemos una consulta.

MongoDB permite también que hagamos sharding de forma manual, pero si optamos por esta alternativa perderíamos una de las mejores características de MongoDB: su simplicidad.

Aunque con un solo servidor, con la replicación maestro/esclavo o con el auto sharding en dos servidores tendríamos suficiente para nuestras aplicaciones sencillas, es bueno usar una herramienta que nos permita escalar a muy altos niveles son morir en el intento.

- **BASE DE DATOS MULTIPLATAFORMA**

  MongoDB fue desarrollado en C++, esto hace posible correr la aplicación prácticamente en cualquier sistema operativo. MongoDB actualmente cuenta con versiones para Linux, Mac, Windows, Solaris. El código fuente de la herramienta está disponible para compilarlo[1] según nuestras necesidades. Vale la pena recalcar que para las plataformas oficiales es mejor descargar los instaladores disponibles en la web de la empresa que mantiene la herramienta.

[1] **Compilar:** http://es.wikipedia.org/wiki/Compilador: Es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar

CAPITULO III: METODOLOGIA

3.1 ANTECEDENTES

Las bases de datos NoSQL han sido creadas con un enfoque diferente a las bases de datos relacionales tradicionales, mientras que las relacionales buscan cada vez mejorar temas como transaccionalidad, características ACID, soporte para Triggers, Procedimientos Almacenados, etc. Mientras que las NoSQL se han concebido como soluciones que ofrezcan altas velocidades, simplicidad, escalabilidad a costa de algunas características comunes a las bases de datos relacionales.

Aunque para darnos cuenta de la verdadera diferencia en velocidad entre los productos de ambos paradigmas (relacionales y NoSQL) deberíamos tener un clúster con muchos nodos de trabajo y una aplicación con millones de usuarios. Este tipo de pruebas es prácticamente imposible de llevarla a cabo en proyectos de media escala, enfocados a pequeñas empresas, es más aún las grandes industrias de nuestro medio tal vez lleguen a involucrar a miles de usuarios, pero estoy segura que ninguna llegará ni siquiera al millón de usuarios concurrentes como en Facebook o Twitter. Es por esto que debemos valorar algunas de las características adicionales que nos ofrecen las bases de datos NoSQL a parte de la escalabilidad.

El desarrollo web es uno de los principales campos de aplicación para la tecnología NoSQL, en ella incluimos directorios, wikis, blogs, entornos colaborativos, etc. El desarrollo web se beneficia especialmente de la simplicidad de las herramientas, entre ellas MongoDB. Aunque generalmente los desarrolladores web tengan conocimientos acerca de bases de datos y SQL[1], el simple hecho de que un analista haya diseñado una base para determinado proyecto web involucra que el desarrollador tenga que aprender de dicho diseño para poder plasmarlo en el aplicativo; vale la pena recalcar que no hablamos de sistemas ERP, o te transacciones en línea de un banco, en lo absoluto, hablamos de aplicaciones web orientadas a servicios del tipo “startup”

[1] SQL: http://es.wikipedia.org/wiki/SQL: Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en éstas

en los que alguien tuvo una idea de un servicio novedoso que beneficiaría mucho a sus usuarios, por ejemplo Digg, FourSquare, etc.

No debemos dejar de lado también la posibilidad de implementar soluciones hibridas que usen bases de datos relacionales y NoSQL en conjunto, por ejemplo Facebook usa MySQL para ciertos datos y Cassandra para cubrir otros requerimientos.

El simple hecho de conocer más herramientas, sus principales características (ventajas, desventajas, carencias, etc.) y los posibles escenarios de aplicación permite que siempre dispongamos de la herramienta correcta para cubrir determinado trabajo.

Aunque los tradicionales defensores de las bases de datos relacionales generalmente no le den la importancia que las nuevas herramientas merecen, deberíamos prestarles la debida atención, pues si empresas como Facebook, Twitter, Digg, FourSquare, Google, Amazon, etc. Que son gigantes de la informática y tienen millones de usuarios suscritos a sus servicios y cuyo principal activo lo constituyen los datos de sus usuarios; han encontrado útiles las herramientas NoSQL debe ser porque cumplen con sus objetivos de excelente manera.

3.2 BASES DE DATOS RELACIONALES VS. NOSQL

Este término debe ser bastante familiar para aquellos que gustan de leer acerca de las nuevas tendencias que la tecnología toma. Si digitamos en Google “bases de datos relacionales vs. NoSQL” seguramente obtendremos no solo un resultado sino muchos. Como es común en el mundo de la tecnología, al aparecer una tendencia nueva la mayor parte de usuarios querrá compararla con la que ya existía; esto es factible en algunos temas pero definitivamente en otros no.

Bien, entonces, ¿es factible comparar las bases de datos relacionales con las NoSQL?

La respuesta es sencilla: no deberíamos compararlas pues son dos herramientas con enfoques diferentes. Pongamos un ejemplo: cuando se publicó el framework para desarrollo web Rails[1] para el lenguaje de programación Ruby (más conocido como Ruby on Rails o RoR); debido al éxito que la herramienta tuvo en la comunidad de desarrolladores, esta tecnología ganó muchos adeptos, quienes comenzaron a compararlo a la "tecnología de facto" para el desarrollo web: PHP. Esta comparación se realizó seguramente en miles de temas de foros de tecnología, pero en realidad no era una comparación justa debido a que Ruby on Rails es un framework para desarrollo, mientras que PHP es un lenguaje de programación; y aunque el objetivo de los dos sea el mismo "desarrollo web" son herramientas con enfoques distintos: RoR es en realidad con conjunto de herramientas que implementan el paradigma de programación MVC orientado a la web, mientras que PHP es sencillamente un lenguaje de programación como tal que no implementa ningún paradigma. A pesar de ello los adeptos a RoR decían que éste era mucho mejor que PHP.

El ejemplo anterior ilustra perfectamente el caso de comparar las bases de datos relacionales con las NoSQL, pues aunque las dos tecnologías tienen como objetivo almacenar datos, el enfoque que cada una tiene es diferente, por lo tanto no podríamos calificar a ninguna como mejor que la otra. Se darán casos en los que efectivamente las bases de datos relacionales funcionen completamente mejor que las NoSQL, pero también habrá escenarios en los que las NoSQL sean una mejor alternativa.

En conclusión las bases de datos NoSQL deberían considerarse como una herramienta más para el desarrollo de proyectos de software, pues como se mencionó en textos anteriores no buscan reemplazar a las tradicionales relacionales; la razón de ser de las bases de datos NoSQL se resume en ofrecer características como sencillez, escalabilidad, y demás que la convierten en una alternativa muy funcional para ciertos proyectos.

[1] Rails: Framework de Ruby

3.3 BASES DE DATOS NOSQL EN EL DESARROLLO WEB

El desarrollo web es uno de los principales beneficiados de las bases de datos NoSQL, pero ¿qué pasa si la escalabilidad no es una de nuestras preocupaciones? Pues definitivamente existen otros motivos por los que el uso de NoSQL es perfectamente viable:

- La base de datos es de crucial importancia en el desarrollo en general, y el desarrollo web no es la excepción. Los desarrolladores web se ven forzados a pensar en entidades, joins, agregados, relaciones, llaves primarias, llaves foráneas, restricciones, etc. Si somos observadores nos daremos cuenta los datos nunca se almacenarán en la base tal como son presentados al usuario final, hasta en la más mínima operación intervendrán dos o más tablas; como desarrolladores tendremos que interpretar los errores dados por ejemplo al tratar de eliminar un registro del que dependen otras tablas, y mostrar un mensaje entendible para el usuario.

  Además, cuando consultamos datos tenemos que realizar joins relacionando correctamente las tablas, etc. Esto consume tiempo y esfuerzo en el desarrollador.
  Estos problemas se solventan en parte usando ORMs, que son herramientas que mapean la base de datos relacional para darnos objetos en formato nativo del lenguaje de programación que estamos usando; desde luego esto impacta el rendimiento de nuestra aplicación, y de todas maneras no resuelve el principal problema: la información no se graba en la base de datos tal como la visualizamos al usuario final.

- Las bases de datos NoSQL, principalmente las basadas en documentos (CouchDB y MongoDB) brindan un alto grado de comodidad al desarrollador pues la forma de grabar los datos es mucho más parecida a la realidad y mucho más fácil de entender que en una base de datos relacional.

- Las bases de datos NoSQL al no tener un esquema estático (pues basta con definir el nombre de la colección) brinda al desarrollador alta flexibilidad

al momento de agregar campos por ejemplo de determinado objeto; es decir el desarrollador diseña su esquema al tiempo que programa.

- MongoDB y CouchDB utilizan JSON como formato para almacenar los datos, la sencillez de JSON lo hace muy fácil de comprender, interpretar y manipular desde nuestro lenguaje de programación.

Todas las características mencionadas con anterioridad favorecen a la productividad del desarrollador, al ser más simple y fácil de usar, el tiempo de desarrollo de un proyecto se verá acortado.

3.4 EVALUACIÓN DE LAS HERRAMIENTAS NO SQL

En el Capítulo I de la presente investigación se ha seleccionado las siguientes herramientas para su estudio y selección:

- Cassandra
- CouchDB
- MongoDB

A continuación realizaremos un cuadro comparativo de las tres herramientas con el fin de evaluar los escenarios de aplicación de cada una de ellas, ventajas y desventajas, etc.

INFORMACIÓN BÁSICA			
Nombre:	Cassandra	CouchDB	MongoDB
Empresa:	Apache	Apache	10Gen
Licencia:	Licencia Apache	Licencia Apache	GNU
Suscripciones con Soporte:	Con terceras empresas	Producto comercial disponible bajo el nombre de CouchBase	Planes de suscripciones que ponen a nuestra disposición profesionales de la herramienta para ayudarnos con nuestras implementaciones.
Documentación:	Pobre documentación	Buena documentación	Excelente documentación
Comunidad de Usuarios:	No tiene tan amplia comunidad de usuarios	Tiene una comunidad de usuarios media	Gran comunidad de usuarios

Tabla 1 - Cuadro comparativo

CARACTERÍSTICAS TÉCNICAS	
A. ALMACENAMIENTO Y MODELO DE DATOS	
Cassandra	• Utiliza el modelo llave/valor. • Casandra almacena los datos en forma de tablas hash, utiliza familias de columnas (Column Family) para almacenar los datos. • Para grabar colecciones dentro de un campos utiliza las denominadas “súper Column Family”
CouchDB	• Graba los datos en forma documental, apoyada en el modelo llave/valor. • Utiliza JSON puro. • Podemos almacenar subdocumentos como valores de los campos.
MongoDB	• Graba los datos en forma documental, apoyada en el modelo llave/valor. • Utiliza la versión binaria de JSON denominada BSON.

Tabla 2 - Características técnicas – Almacenamiento y modelo de datos

CARACTERÍSTICAS TÉCNICAS	
B. INRERFACES	
Cassandra	• Para las conexiones a la base utiliza su propio protocolo de comunicación denominado Thrift.
CouchDB	• Para conexiones a la base utiliza una API RESTful HTTP que funciona mediante servicios web. CouchBase (la versión comercial de CouchDB) ofrece drivers para usarlos con varios lenguajes de programación.
MongoDB	• Dispone de un gran número de drivers nativos oficiales para usarse con varios lenguajes de programación.

Tabla 3 - Características técnicas – Interfaces

CARACTERÍSTICAS TÉCNICAS	
C. ESCALABILIDAD HORIZONTAL	
Cassandra	• Basada en replicación. • Alta tolerancia a fallos. • Permite agregar y quitar nodos “en caliente” sin reiniciar los servicios.
CouchDB	• Para la replicación utiliza su API RESTful HTTP. • Permite resumir la sincronización de los datos entre los nodos si existiera algún error de hardware o conectividad.
MongoDB	• Permite escalar usando su función de Auto sharding o de auto segmentar los datos en sus diferentes nodos de trabajo.

Tabla 4 - Características técnicas – Escalabilidad horizontal

CARACTERÍSTICAS TÉCNICAS	
D. REPLICACIÓN	
Cassandra	• Basada en replicación. • Alta tolerancia a fallos. • Permite agregar y quitar nodos “en caliente” sin reiniciar los servicios.
CouchDB	• Replicación Maestro/ Maestro, pero los desarrolladores deben proveer las sentencias para resolver los conflictos que puedan darse en este tipo de replicación.
MongoDB	• Replicación Maestro/Esclavo. MongoDB usa su sistema de replicación solo para alta disponibilidad.

Tabla 5 - Características técnicas – Replicación

CARACTERÍSTICAS TÉCNICAS	
E. SOPORTE PARA ALMACENAR ARCHIVOS DE GRAN TAMAÑO	
Cassandra	• No brinda soporte para archivos de gran tamaño, por lo que tenemos que dividir los archivos grandes en partes más pequeñas para poder almacenarlos.
CouchDB	• Brinda soporte para archivos de gran tamaño mediante “adjuntos”.
MongoDB	• Brinda soporte para archivos de gran tamaño mediante GridFS.

Tabla 6 - Características técnicas – Soporte para almacenar archivos de gran tamaño

CARACTERÍSTICAS TÉCNICAS	
F. CONSULTAS DINAMICAS	
Cassandra	• Soporta consultas dinámicas.
CouchDB	• No soporta consultas dinámicas, las consultas deben ser programadas para luego ser consumidas.
MongoDB	• Soporta consultas dinámicas.

Tabla 7 - Características técnicas – Consultas dinámicas

CARACTERÍSTICAS TÉCNICAS	
G. CONTROL DE CAMBIOS Y CONSISTENCIA	
Cassandra	• Soporta MVCC, esta característica permite asegurar que las operaciones sobre los datos se reflejen en todos los nodos de trabajo de un entorno distribuid (clúster). • Cassandra nos permite configurar el nivel de consistencia.
CouchDB	• Soporta MVCC, versionando los cambios y sincronizándolos a los demás nodos de trabajo.
MongoDB	• No versiona, las actualizaciones se realizan sobre los datos como tal para distribuirlos a los demás nodos de trabajo.

Tabla 8 - Características técnicas – Control de cambios y consistencia

CARACTERÍSTICAS TÉCNICAS	
H. PLATAFORMAS SOPORTADAS	
Cassandra	• Windows • Linux • Mac OS X
CouchDB	• Mac OS X • Linux • Solaris • BSD • Android (plataforma móvil)
MongoDB	• Windows • Linux • Mac OS X • Solaris

Tabla 9 - Características técnicas – Plataformas soportadas

CARACTERÍSTICAS TÉCNICAS	
I. DRIVERS NATIVOS OFICIALES PARA LENGUAJES DE PROGRAMACIÓN	
Cassandra	• Java • Python
CouchDB	• Soporta todos los lenguajes de programación que puedan trabajar con servicios web vía su API RESTful usando JSON.
MongoDB	• C • C++ • Erlang • Haskell • Java • JavaScript • .NET (C# F#, PowerShell, etc) • Perl • PHP • Python • Ruby • Scala

Tabla 10 - Características técnicas – Drivers nativos oficiales para lenguajes de programación

Una vez que hemos representado en matrices las tres bases de datos, a continuación vamos hacer una revisión de cada uno de los puntos:

a) **Almacenamiento Y Modelo De Datos**

Es importante conocer el modelo de datos, pues al trabajar con ellos es de mucha utilidad entender cómo están almacenados.

Cassandra tiene un modelo de datos un poco complejo de entender con sus "Family Columns" y "súper Family columns" basadas en llave/valor; mientras que CouchDB y MongoDB ofrecen un modelo documental mucho más fácil de comprender y más parecido a la realidad.

CouchDB y MongoDB utilizan una notación basada en JSON, que en realidad permite estructurar muy bien los datos sin perder entendimiento de los mismos.

Para comprender mejor las definiciones podemos retroceder un poco en la historia hasta los archivos CSV[1], luego XML[2] y JSON. Bien para ejemplificar el uso de estas tres tecnologías vamos a suponer que deseamos almacenar registros de personas con los siguientes datos de cada una: apellidos, nombres, teléfono fijo, teléfono móvil.

En CSV tendríamos un archivo con un contenido parecido a:

```
Apellidos, Nombres, Fijo, Movil
Brito Zhunio, Diana Marisela, 072803525, 099510976
Brito Zhunio, Rubí Adriana, 072803506, 090040126
```

En XML tendríamos algo parecido a:

```
<persona>
        <apellidos>
                Brito Zhunio
        </apellidos>
        <nombres>
                Diana Marisela
        </ nombres >
        <fijo>
                072803525
        </ fijo >
        <móvil>
                099510976
        </ móvil >
</persona>
<persona>
        <apellidos>
                Brito Zhunio
        </apellidos>
        <nombres>
                Rubí Adriana
```

[1] **CSV**: Son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.
[2] **XML**: Formato para representar información

```
        </ nombres >
        <fijo>
                072803506
        </ fijo >
        <móvil>
                090040126
        </ móvil >
</persona>
```

Y finalmente en JSON tendríamos:

```
{
        "apellidos":    "Brito Zhunio";
        "nombres":      "Rubí Adriana";
        "números":
        [
                { "fijo":           "072803525"},
                { "móvil":          "099510976"},
        ]
}
{
        "apellidos":    "Brito Zhunio";
        "nombres":      "Diana Marisela";
        "números":
        [
                { "fijo":           "072803506"},
                { "móvil":          "090040126"},
        ]
}
```

Luego de comparar estos tres formatos podemos llegar a las siguientes conclusiones:

- CSV: se complicaría con datos complejos, por ejemplo si ya no solo deseamos dos números sino varios números fijos, faxes, móviles con su respectiva operadora, etc. Entonces CSV comienza a mostrar sus limitaciones con datos estructurados.
- XML: Podría solucionar este problema, pero es más complejo de usar, aunque ofrezca la flexibilidad necesaria, sería más útil para intercambios de información muy estructurada, desde luego hasta el intercambio de datos más simple necesita un trabajo considerable.
- Por otra parte JSON se sitúa entre los dos formatos anteriores, pues a diferencia de CSV, JSON soporta datos estructurados, pero a diferencia de XML JSON es mucho más fácil de entender y de usar.

b) **Interfaces**

Una vez más en este apartado Cassandra era un poco más complicado de usar debido a que familiarizarse con su API Thrift que usar un driver nativo para nuestro lenguaje de programación. El equipo de desarrollo de Cassandra se ha esforzado en tratar de ofrecer drivers nativos pero solo encontramos disponibles para Java y Python.

La interface de CouchDB se basa enteramente en servicios web, por ese motivo es compatible con todo lenguaje de programación que soporte trabajar con dichos servicios, para interactuar con los datos los desarrolladores deben usar JSON.

MongoDB da la solución más sencilla ofreciendo drivers para los lenguajes de programación más usados, de esta manera los desarrolladores pueden trabajar en su lenguaje nativo sin preocuparse por JSON o BSON, simplemente trabajarían con representaciones en forma de arreglos para la recuperación de datos y para enviarlos a la base para su respectivo almacenamiento.

c) **Escalabilidad Horizontal**

Tanto Cassandra como CouchDB se valen de la replicación como una vía para escalar horizontalmente además de usarlo como una herramienta que provea alta disponibilidad; es aquí en donde radica una gran diferencia: MongoDB usa la replicación solamente como un medio de alta disponibilidad, pero se vale del Auto sharding (Particionamiento de los datos) para proveer escalabilidad horizontal. Esto aumenta considerablemente en la velocidad de MongoDB.

d) **Replicación**

Cassandra y CouchDB soportan replicación maestro/ maestro; no así MongoDB nos permite implementar replicación maestro /esclavo.

e) **Soporte Para Almacenar Archivos De Gran Tamaño**

MongoDB con GridFS, y CouchDB con sus “adjuntos”, brindan soporte para almacenar datos de gran tamaño como archivos de video, imágenes en alta definición, clips de sonido, etc. En este apartado Cassandra es la que más problema tiene pues el desarrollador debe dividir su archivo en cuestión para poder almacenarlo, mientras que en CouchDB y en MongoDB la herramienta se encargar de particionar los archivos de ser necesario; siendo favorecido el desarrollador para quien la gestión de dichos datos es transparente pues ellos siguen tratándolo como un valor común.

f) **Consultas Dinámicas**

MongoDB y Cassandra soportan consultas dinámicas, que no es más que ejecutar una consulta que el sistema de base de datos no espera que se haga; en el mundo de las bases de datos relacionales el soporte de consultas dinámicas es estándar a todas las herramientas debido a que las tablas se presentan como esquemas estáticos por lo la base de datos siempre sabe que campos tiene a disposición; no así en las bases de datos NoSQL en donde cada registro o documento que conforma una colección puede tener una estructura diferente a los demás. Debido a este inconveniente para las bases NoSQL resulta un poco más complicado implementar esta funcionalidad.

Como se mencionó Cassandra y MongoDB brindan soporte para este tipo de consultas, no así CouchDB en el que tenemos que especificar las consultas que se pueden consumir desde el aplicativo que estemos desarrollando.

g) **Control De Cambios Y Consistencia**

El control de cambios y consistencia permite que la base de datos en entornos distribuidos asegure que todos los clientes que consultan unos mismos datos reciban la última actualización realizada.

Ese concepto trae algunos inconvenientes que CouchDB y Cassandra manejan un poco mejor que MongoDB.

Cassandra nos permite configurar el nivel de consistencia que queremos tener a coste de rendimiento.

CouchDB ofrece un muy buen sistema de sincronización de datos a través de sus APIs web, incluso puede resumir la sincronización si ésta se viera interrumpida por algún motivo.

h) **Plataformas Soportadas**

Las tres herramientas soportan los sistemas operativos más populares, solamente CouchDB tiene problemas en Windows, pues no se ha distribuido un instalador oficial.

i) **Drivers Nativos Oficiales Para Lenguajes De Programación**

MongoDB es superior en este aspecto a las otras dos herramientas, pues cuenta con drivers oficiales para la mayoría de lenguajes de programación más populares de nuestro medio. Esto repercute en que MongoDB sea muy fácil de usar, pues las estructuras de datos se manejan en representaciones en el lenguaje nativo que estemos usando, es decir no usaremos ni JSON o BSON, simplemente el lenguaje de nuestra preferencia.

No debemos desmerecer las APIs Web con las que cuenta CouchDB y aunque la transformación que debe hacerse de JSON al lenguaje nativo no es tan sencilla o rápida, es una implementación muy interesante y podría ser muy útil en algunos casos.

Luego de comparar las características de las herramientas seleccionadas es hora de que evaluemos los mejores escenarios en los que cada una de las bases puede ser aplicada:

Comencemos por Cassandra, debemos mencionar que independientemente al hecho que esté desarrollada en Java, se integra muy bien en aplicaciones desarrolladas con dicha tecnología. Sería ideal para aplicarla en sistema de logins, y para realizar minería de datos e inteligencia de negocios basada en el

análisis de datos. Es un poco difícil de usarla hasta acostumbrarse a la herramienta.

CouchDB veríamos usarla en aplicaciones cuyos datos no sean cambiantes debido al limitante de no soportar consultas dinámicas, que es uno de los principales inconvenientes que ésta herramienta presta. Aun así esta base de datos es muy recomendada para quienes necesiten implementar réplicas maestro/maestro que estén distribuidas geográficamente y necesiten las capacidades de sincronización que CouchDB utiliza. También es recomendad como una base de datos embebida para móviles por ejemplo, ya que aprovecharía sus capacidades de sincronización fuera de línea.

Finalmente MongoDB es una herramienta recomendada para quienes necesitan alto desempeño, para quienes desean altas velocidades de escritura de datos; pero principalmente para quienes necesiten una base de datos flexible, fácil de usar, rápida, con buena integración en los lenguajes de programación, así como buena comunidad y documentación. En general para todo proyecto en el que usaríamos MySQL podríamos también usar MongoDB sabiendo sus virtudes y sus defectos en función al tipo de proyecto que estemos implementando.

CAPITULO IV: DESARROLLO

4.1 El MODELO DE DATOS DE MONGODB

La unidad básica de almacenamiento se define como un "*documento*" que sería el equivalente a un registro en una base de datos relacional. Los "*documentos*" con contenidos en una estructura de mayor jerarquía denominada "*colección*" que almacena los datos de similar tipo; las colecciones serían el equivalente las tablas en un sistema relacional. Finalmente las colecciones se agrupan en una base de datos que las contendrá.

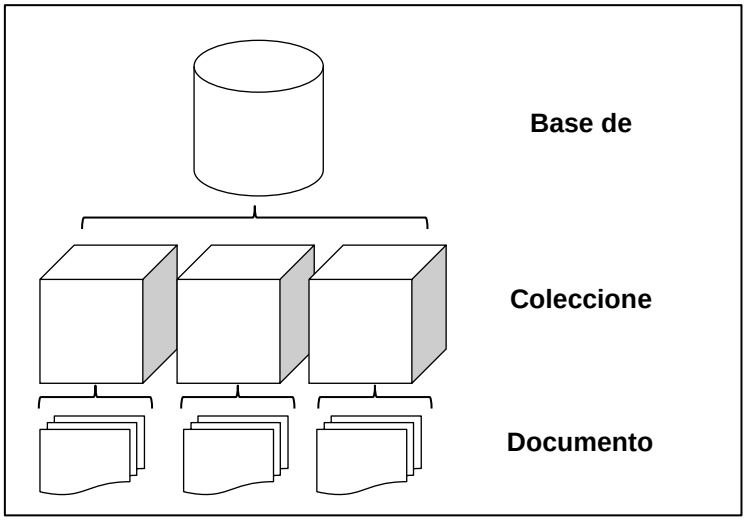

Ilustración 5 - Modelo de Datos MongoDB

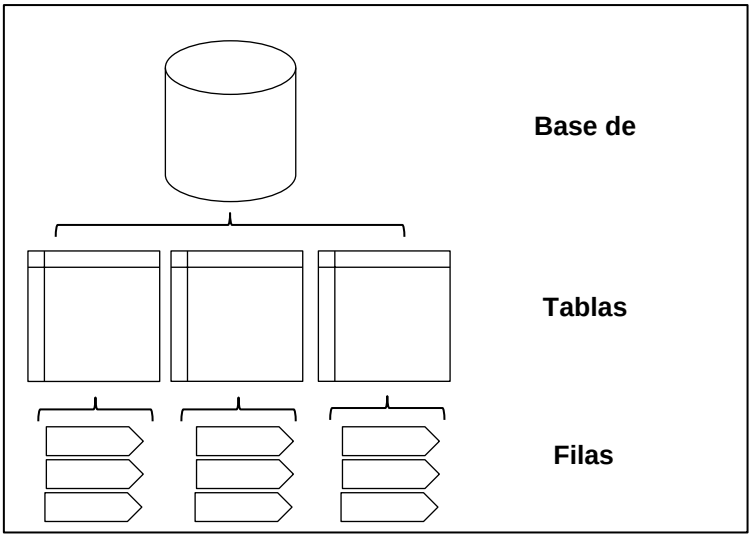

Ilustración 6 - Modelo De Datos Relacional

Debemos recordar que MongoDB es una base de datos documental que no usa esquemas de datos con columnas predefinidas. El principal beneficio se refleja en la flexibilidad que ganamos al almacenar los datos, pues de requerir un campo extra no tendremos realizar el proceso de cambiar la estructura de nuestra base de datos como lo haríamos en una relacional.

Las colecciones de datos de MongoDB son extremadamente flexibles, por ejemplo supongamos la colección "*media*" que en JSON podría tener los siguientes documentos.

```
{
        "Tipo": "CD",
        "Artista": "Chayanne",
        "Titulo": "Me Enamore de Ti",
        "Genero": "Baladas",
        "ListaCanciones":
        [
                {
                        "NumeroPista" : "1",
                        "Titulo": " Me Enamore de Ti ",
                },
                {
                        "NumeroPista ": "2",
                        "Titulo ": "Tu Boca",
                }
        ]
}
{
        "Tipo": "Libro",
        "Titulo": "Tesina de Grado",
        "Autores":
        [
                "BRITO, Diana",
                "BRITO, Rubí",
        ]
}
```

Como podemos ver en el ejemplo anterior los dos documentos de la colección "*media*"; si bien es cierto tienen dos campos en común: "*Tipo*" y *"Titulo*", los demás campos son diferentes entre ellos a pesar de pertenecer a una misma colección.

Este ejemplo sería difícil de implementar en una base de datos relacional, la única opción sería crear una tabla con todas las posibilidades que podría darse; con pocos campos podría hacerse, pero si hablamos de un número más alto de campos entonces no sería realizable.

No debemos abusar de la flexibilidad de MongoDB, pues si usamos documentos extremadamente diferentes los unos de los otros no explotaremos todo el potencial de la herramienta.

Para nombrar las columnas, colecciones y bases de datos debemos usar letras y números sin símbolos. El máximo de caracteres permitido para los nombres es de 128, pero es recomendable mantener los nombres cortos.

4.2 LOS DOCUMENTOS EN MONGODB

Los documentos son un conjunto de pares llave/valor. Por ejemplo el par **"*Tipo*": "*CD*"** consiste de una llave denominada "***Tipo***" y de su respectivo valor "***CD***". Las llaves generalmente son un string, pero los valores pueden ser de varios tipos , pueden ser arreglos o hasta datos binarios.

Los tipos de datos que regularmente podemos usar en MongoDB son:

TIPO	DESCRIPCIÓN
String	Puede contener cualquier colección de caracteres, por ejemplo: "**Nombre**": "Diana"
Integer **(32 y 64)**	Permite almacenar valores numéricos sin decimales, por ejemplo: "**Cantidad**": 1987
Boolean	Puede contener valores de falso o verdadero (true y false).
Double	Puede contener valores numéricos con decimales, por ejemplo: "**Valor**": 10,65
Arrays	Puede contener arreglos, por ejemplo: "**Autores**": ["BRITO, Diana", "BRITO, Rubí"]
Timestamp	Puede contener fechas y horas.
Object	Puede contener subdocumentos.
Null	Puede contener valores nulos (NULL).
Symbol	Puede contener caracteres al igual que String, pero se aplica en lenguajes que usan símbolos.
Object ID	Almacena el ID del documento,
Binary	Puede contener datos binarios, por ejemplo una imagen.
Regular Expression	Puede contener expresiones regulares[1].
JavaScript	Puede contener como su nombre lo indica código JavaScript.

Tabla 11 - Tipo de datos que maneja MongoDB

[1] **Expresión regular**: Describe un conjunto de cadenas sin enumerar sus elementos

4.3 ¿DEBEMOS USAR DOCUMENTOS INCRUSTADOS O DOCUMENTOS REFERENCIADOS?

Antes de hablar de este tema es importante comprender por qué necesitamos usarlos. Los documentos incrustados y los documentos referenciados son las alternativas que MongoDB nos brinda para resolver las dependencias de los datos. En el JSON del ejemplo anterior encontramos el caso en el que cada CD debe tener su lista de canciones, de tal manera que en una base de datos relacional tendríamos una tabla denominada ***CDs*** relacionada de uno a varios con una tabla llamada ***Canciones***.

Bien, revisemos las dos alternativas que MongoDB nos brinda:

- **Documentos Incrustados**: permite que en un par llave/valor, dicho valor almacene un subdocumento, por ejemplo:

```
{
        "Tipo": "CD",
        "Artista": "Chayanne",
        "Titulo": "Me Enamore de Ti",
        "Genero": "Baladas",
        "ListaCanciones":
        [
                {
                        "NumeroPista”: "1",
                        "Titulo”: " Me Enamore de Ti ",
                },
                {
                        “NumeroPista “: "2",
                        “Titulo “: "Tu Boca",
                }
        ]
}
```

  El documento de ejemplo usa documentos incrustados para almacenar la lista de canciones en el par “***ListaCanciones***”.

- **Documentos Referenciados:** Nos permite aplicar el concepto de las relaciones del enfoque relacional pero en MongoDB, es decir una colección denominada “***ListaCanciones***” que referencia a la colección “***CDs***”.

4.4 OBJECT ID EN MONGODB

Al igual que una tabla en una base de datos relacional identifica cada una de sus filas con un valor único denominado Llave Primaria; MongoDB también identifica de manera única sus documentos, y lo hace en un par llamado “***_id***”.

En los ejemplos anteriores nunca referenciamos al par "***_id***", sin embargo a pesar de que no lo referenciemos MongoDB se encarga de crearlo de manera automática.

Desde luego que también podemos ingresar el valor del par "_id" de forma manual, simplemente referenciándolo cuando creemos un documento.

4.5 ¿COMO CONSEGUIR MONGODB?

Como se mencionó en capítulos anteriores, MongoDB se encuentra disponible para varios Sistemas Operativos; los archivos de instalación pueden descargarse de forma gratuita del sitio web:

http://www.mongodb.org/downloads

Además desde esa misma web podremos conseguir los drivers para los distintos lenguajes de programación soportados por la base de datos.

4.6 INSTALACION DE MONGDB EN WINDOWS

1. Para instalar MongoDB en windows, necesitamos los archivos de instalación, para ello descargar la última versión estable de la herramienta en la web:

http://www.mongodb.org/downloads

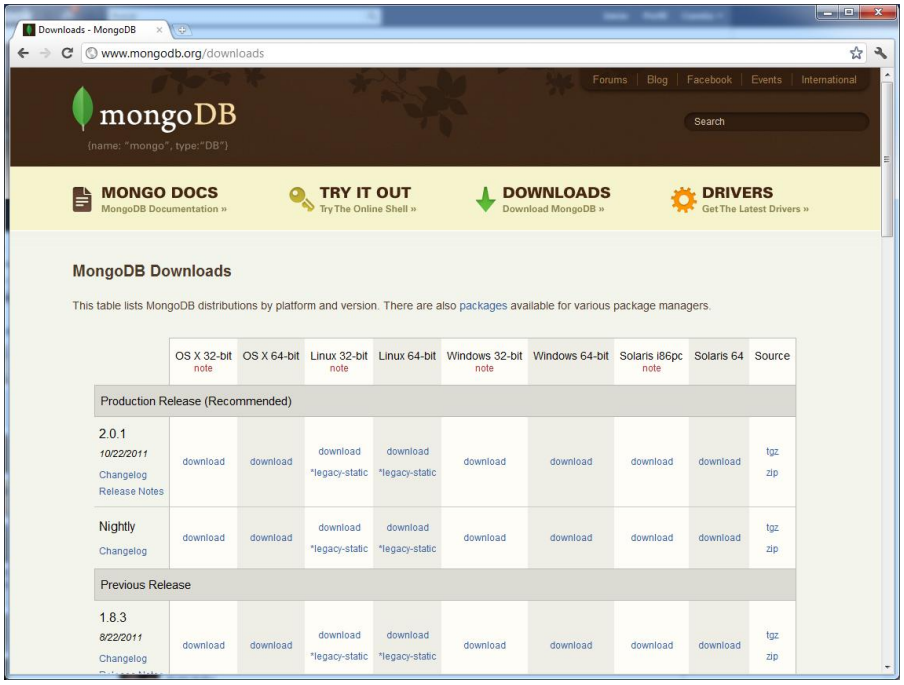

Ilustración 7 - Captura de Pantalla

Nota: Debemos recordar, como ya fue explicado en capítulos anteriores que es mejor usar la versión de 64 bits de MongoDB para que no tengamos límites en el tamaño de la base de datos.

2. Extraer el contenido del archivo comprimido descargado en el paso anterior, en la ruta **C:\mongodb**:

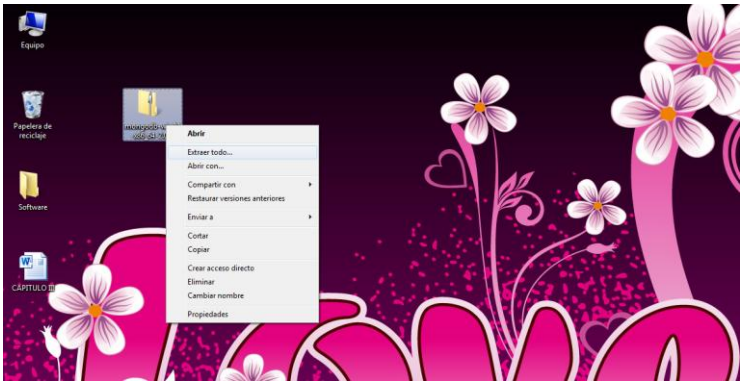

Ilustración 8 - Captura de Pantalla

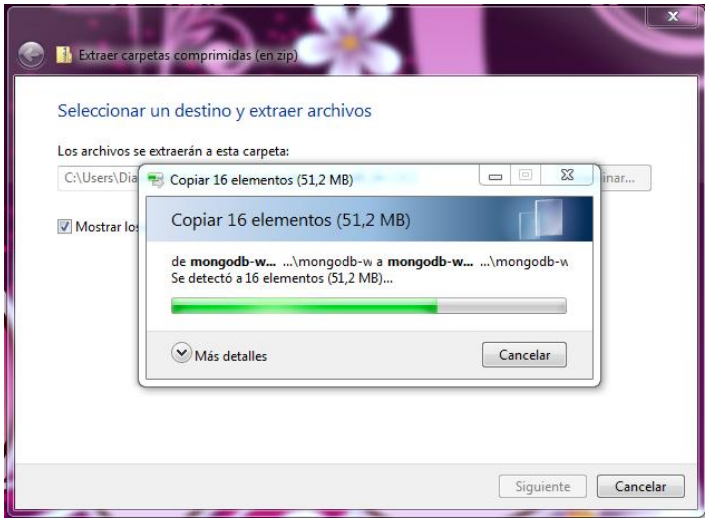

Ilustración 9 - Captura de Pantalla

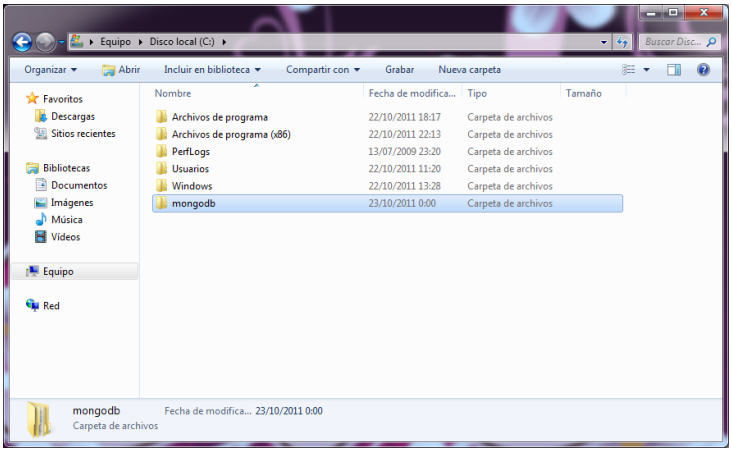

Ilustración 10 - Captura de Pantalla

3. Para una instalación normal bastaría con ejecutar el archivo de arranque del servicio dentro de la carpeta **C:\mongodb\bin\mongod,** pero esto implicaría que cada vez que reiniciemos la máquina necesitemos iniciar el servicio nuevamente; para evitar esto vamos a configurar MongoDB como un servicio del sistema, de tal forma que arranque cuando iniciemos nuestro ordenador de manera automática. Para ello vamos a iniciar la consola de comandos de Windows, pero la vamos a iniciar con privilegios de administrador:

3.1. Clic en inicio, buscar cmd.

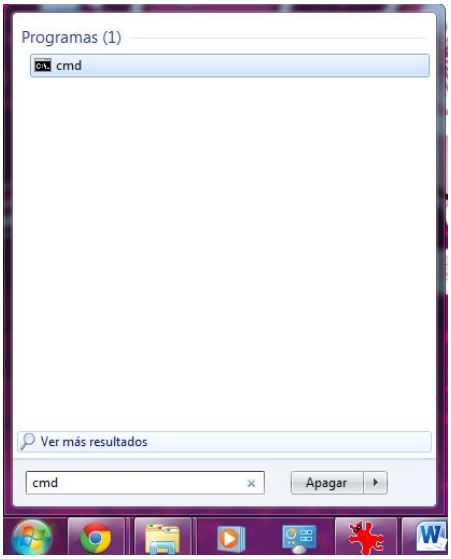

Ilustración 11 – Captura de Pantalla

3.2. Clic con el botón derecho en cmd y seleccionar ejecutar como administrador.

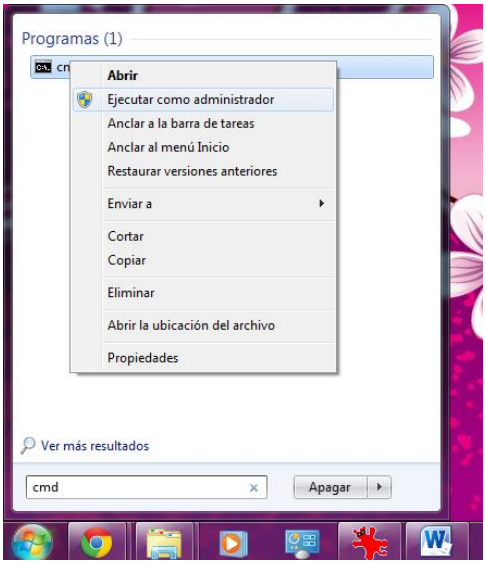

Ilustración 12 - Captura de Pantalla

4. Crear el directorio: **data** en la ruta **C:\mongodb**

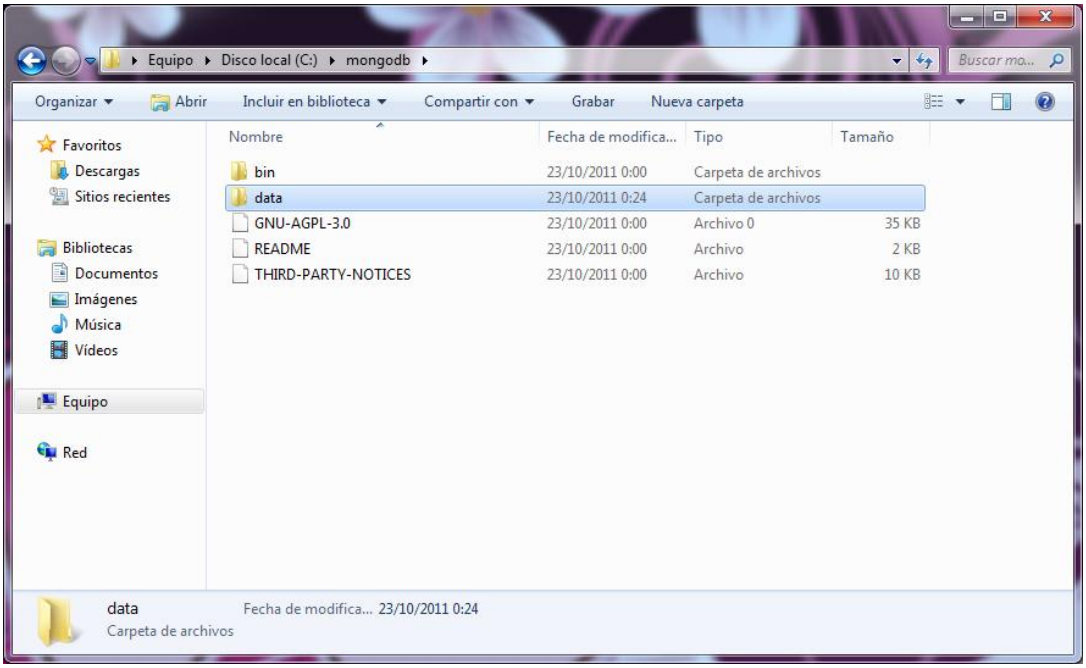

Ilustración 13 - Captura de Pantalla

5. Crear un archivo de texto llamado **logs.txt** en la ruta **C:\mongodb,** este archivo nos permitirá almacenar los logs generados por MongoDB en su periodo de funcionamiento:

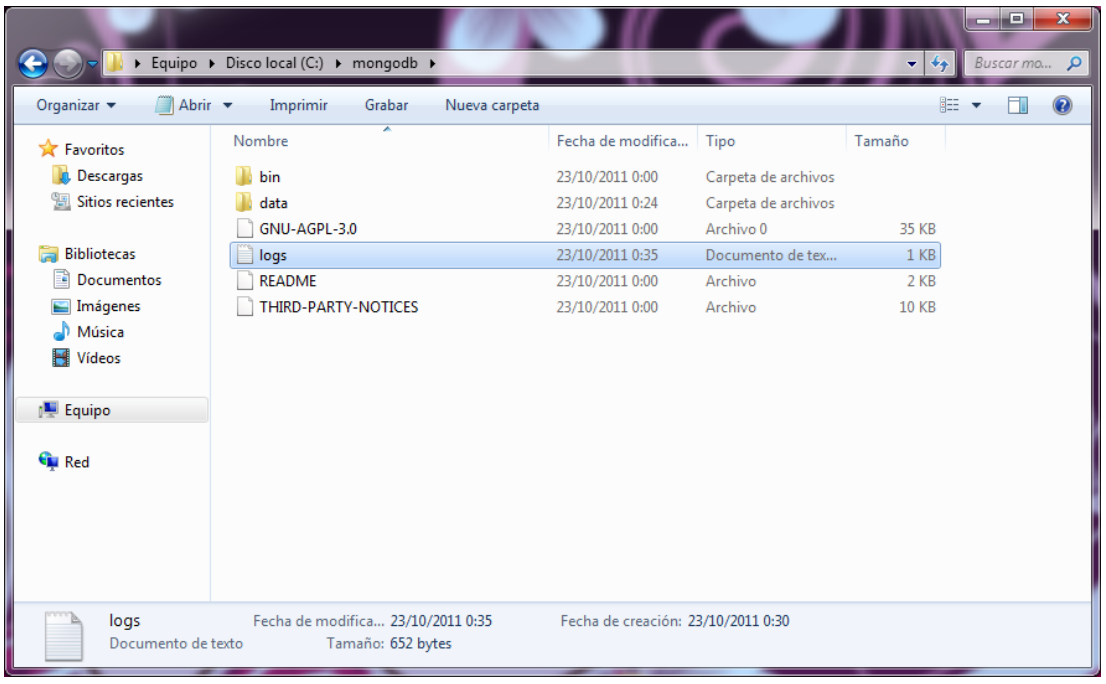

Ilustración 14 - Captura de Pantalla

6. En la consola de comandos de Windows con privilegios de administrador ejecutar el siguiente comando:

C:\mongodb\bin\mongod –install –logpath c: mongodb\logs.txt

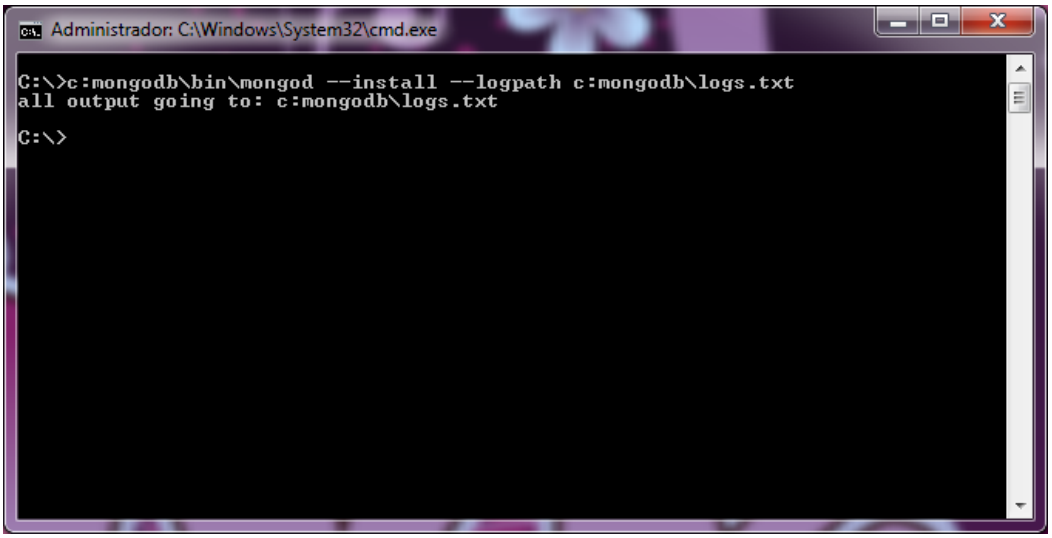

Ilustración 15 - Captura de Pantalla

7. Abrir el archivo **C:\mongodb\logs.txt** y verificar la confirmación del servicio como indica la siguiente imagen.

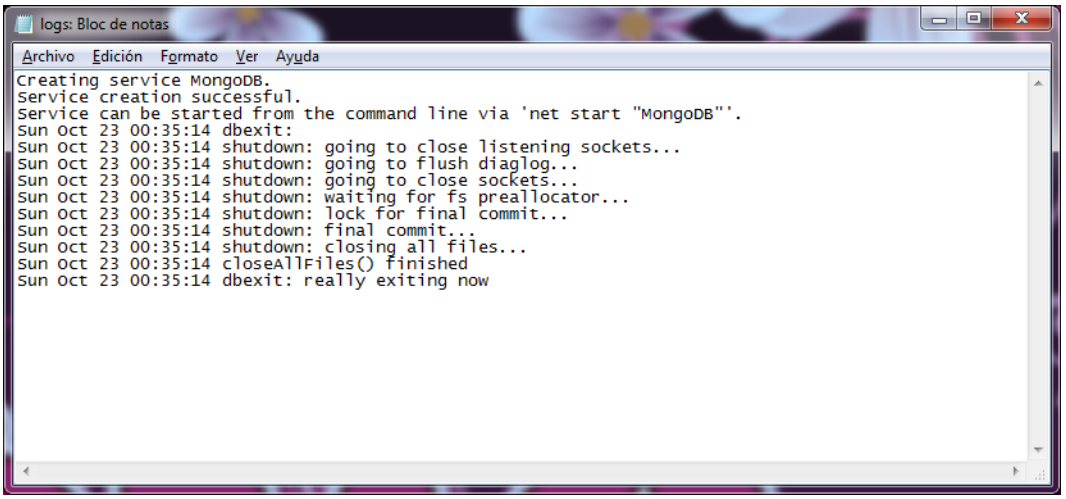

Ilustración 16 - Captura de Pantalla

8. Abrir el editor de registro de Windows, para ello desde el comando Ejecutar abrir: REGEDIT

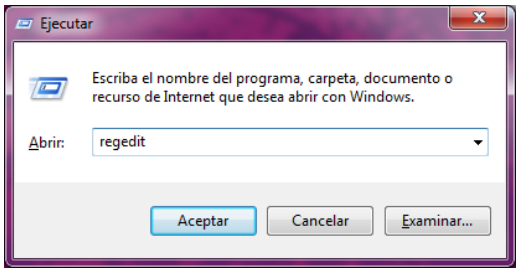

Ilustración 17 - Captura de Pantalla

9. En el Editor de Registro de Windows, seleccionar la ruta **HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services**

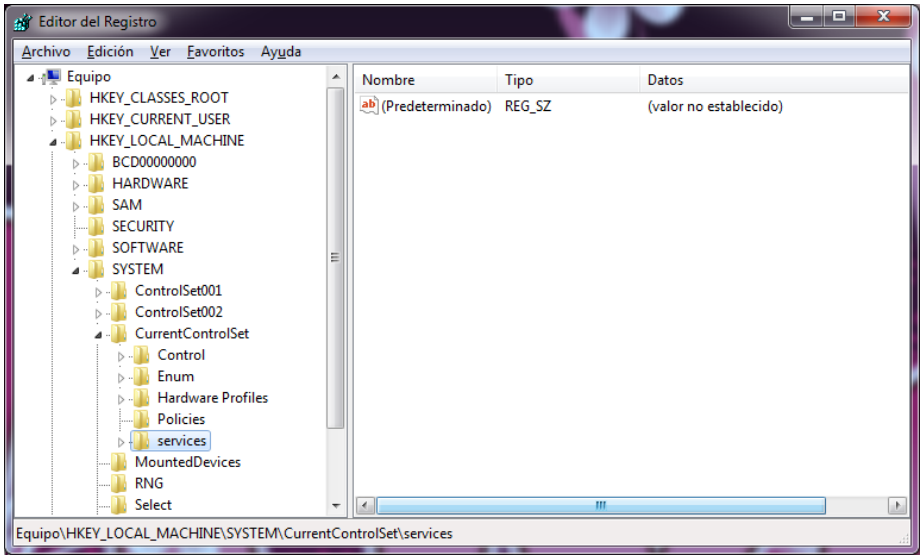

Ilustración 18 - Captura de Pantalla

10. Dentro de **Services** vamos a seleccionar la carpeta **MongoDB**

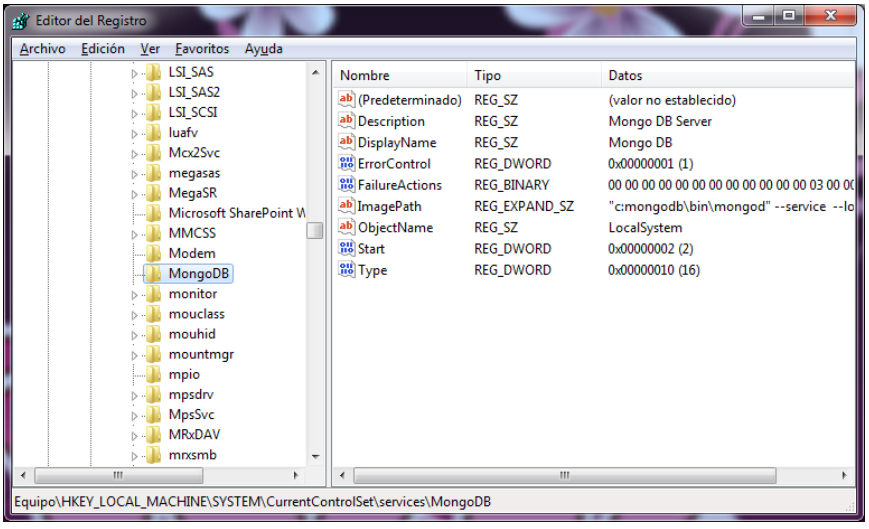

Ilustración 19 - Captura de Pantalla

11. Editar el valor de la variable **ImagePath** poner el valor:

"C: mongodb\bin\mongod" --service --logpath "c:mongodb\logs.txt" --dbpath "c:mongodb\data"

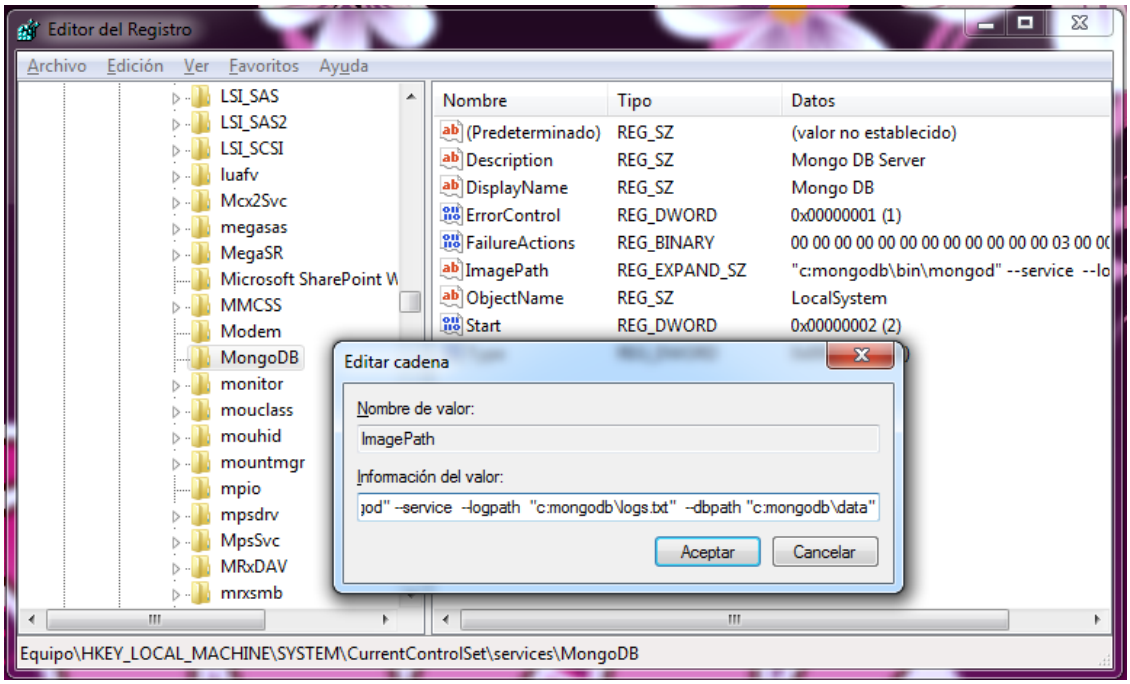

Ilustración 20 - Captura de Pantalla

12. Abrir en el Panel de Control el apartado de servicios

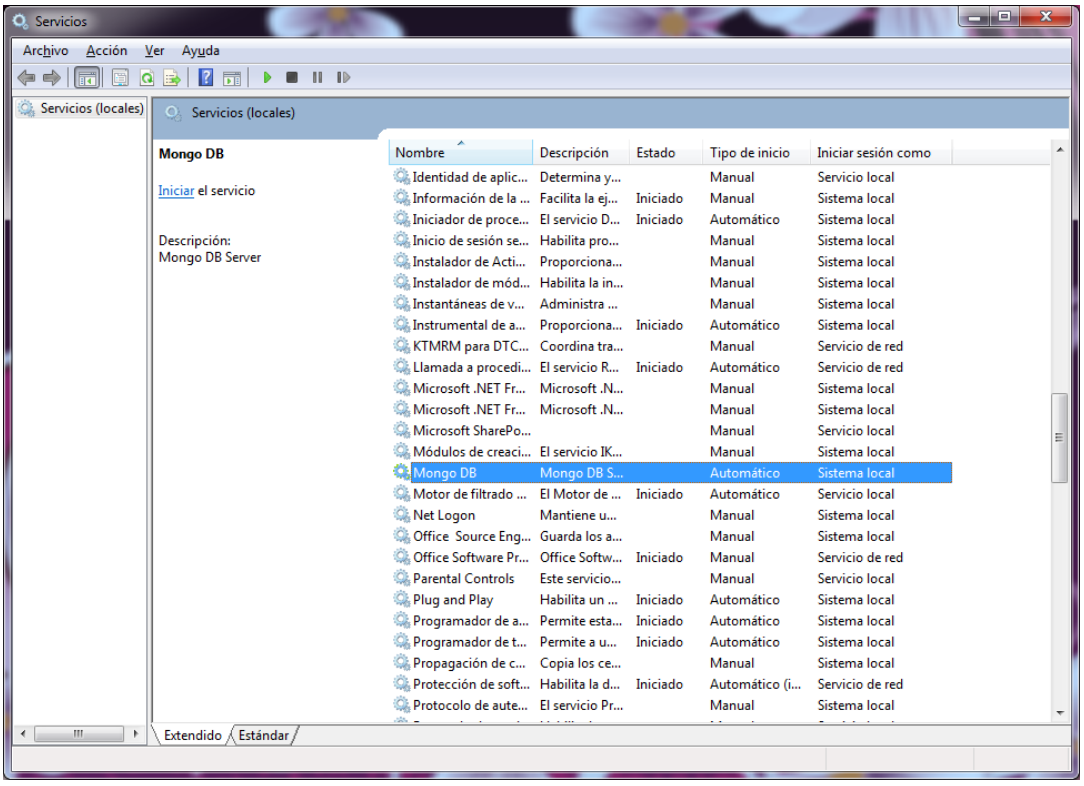

Ilustración 21 - Captura de Pantalla

13. Doble clic en el servicio **MongoDB** y seleccionar Automático como tipo de inicio.

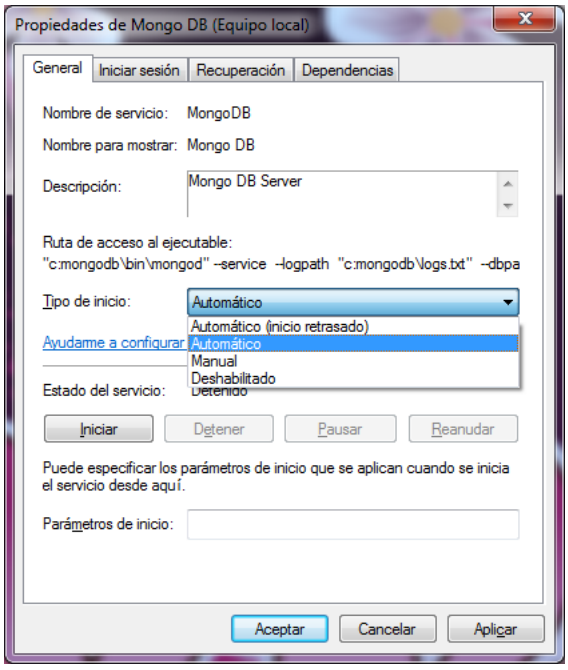

Ilustración 22 - Captura de Pantalla

14. Para comprobar que MongoDB ha sido instalado correctamente ingresamos desde un Browser a la dirección:

http://localhost:28017/

Y deberíamos visualizar la siguiente página

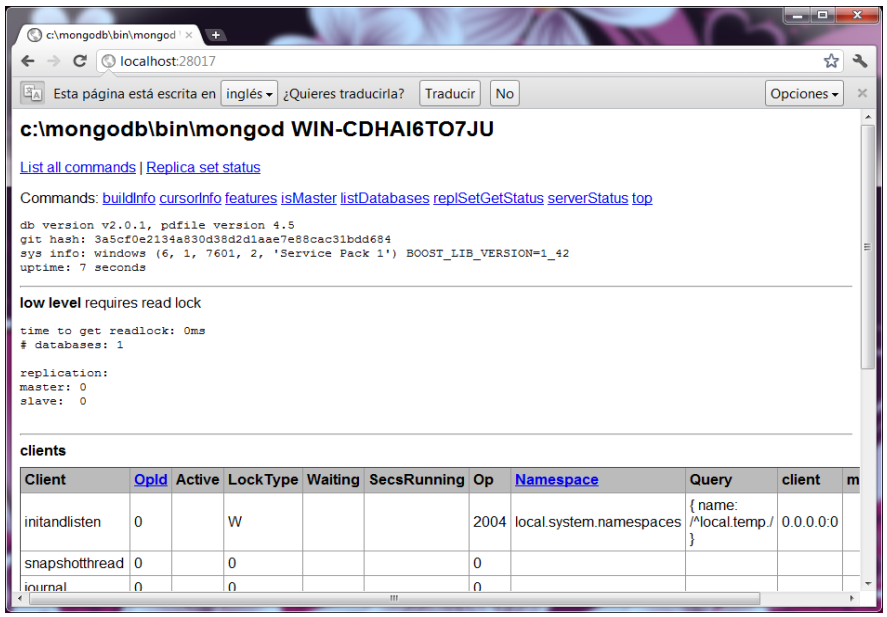

Ilustración 23 - Captura de Pantalla

4.7 USANDO MONGODB A TRAVES DEL GESTOR: MONGOVUE

Generalmente podemos administrar nuestra instancia de MongoDB desde la consola de comandos, desde luego no es una alternativa muy atractiva que digamos. Por este antecedente y debido a la gran popularidad que goza MongoDB existen numerosas herramientas de administración con interface de usuario de escritorio, orientado a la web, etc.

MongoVUE es una de las herramientas más completas, sencillas, y útiles que podemos encontrar, cuenta con versiones de pago y una gratuita.

4.8 INSTALACION DEL IDE DE ADMINISTRACION DE MONGODB

Para administrar MongoDB de una manera más amigable vamos a descargar la interface de administración:

- MongoVUE

4.8.1 INSTALACION DE MONGOVUE

MongoVUE es un excelente producto que nos permite administrar MongoDB y sus funciones de una manera muy amigable. MongoVUE es un producto comercial, pero cuenta con una versión gratuita.
Para instalarlo seguiremos los siguientes pasos:

1. Descargar el paquete de instalación de la dirección:

http://www.mongovue.com/downloads/

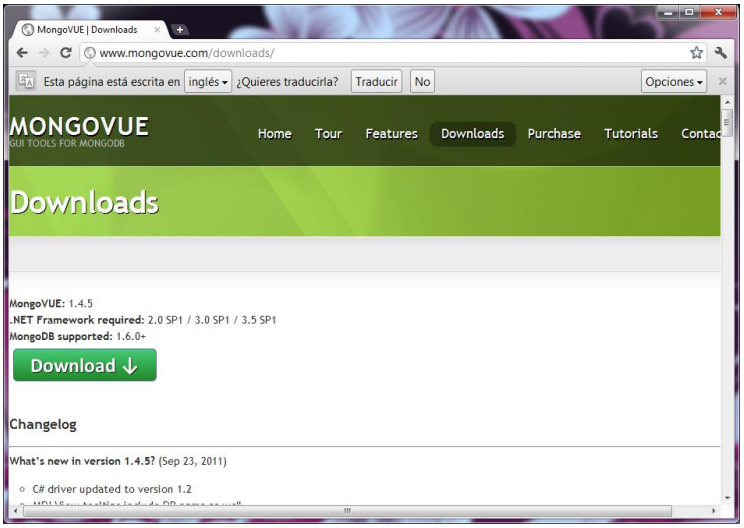

Ilustración 24 - Captura de Pantalla

2. Ejecutar el asistente de instalación y seguir los pasos indicados hasta recibir la confirmación de correcta instalación:

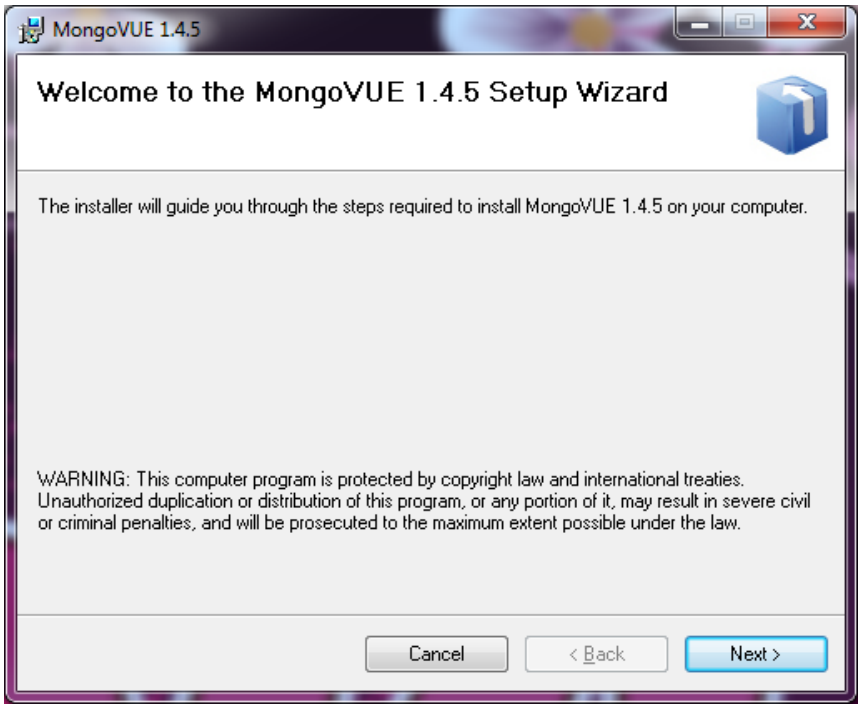

Ilustración 25 - Captura de Pantalla

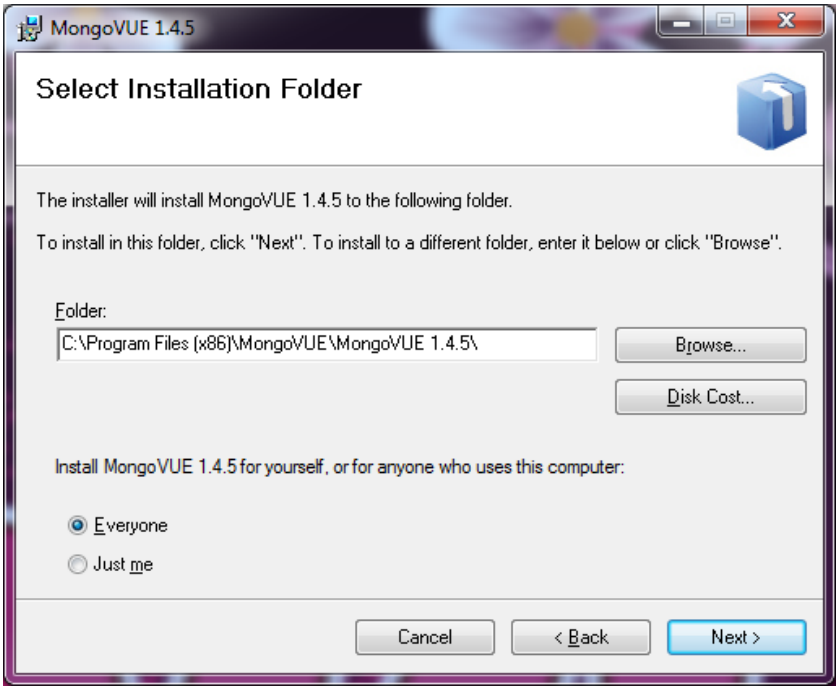

Ilustración 26 - Captura de Pantalla

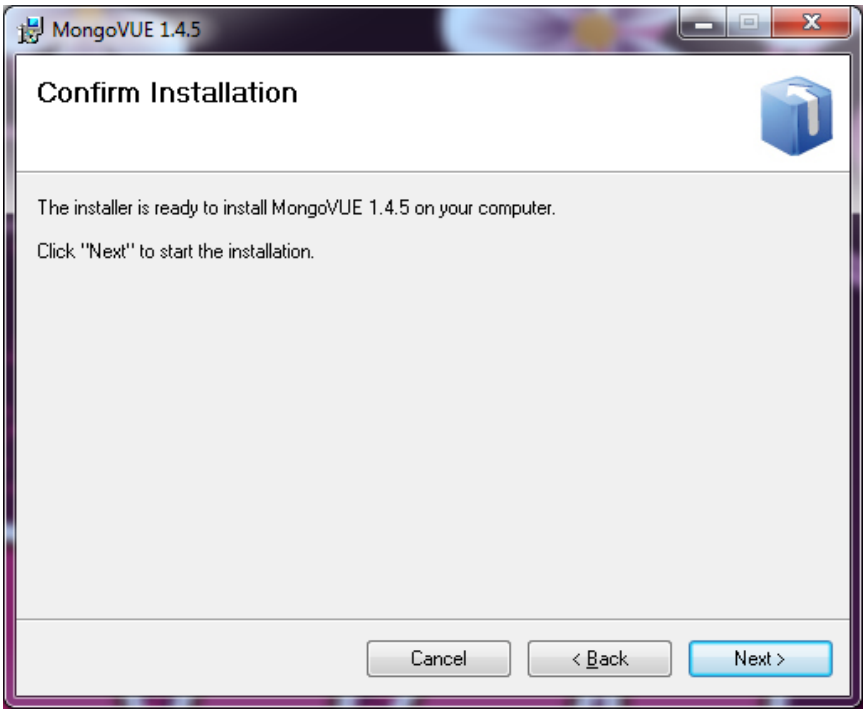

Ilustración 27 - Captura de Pantalla

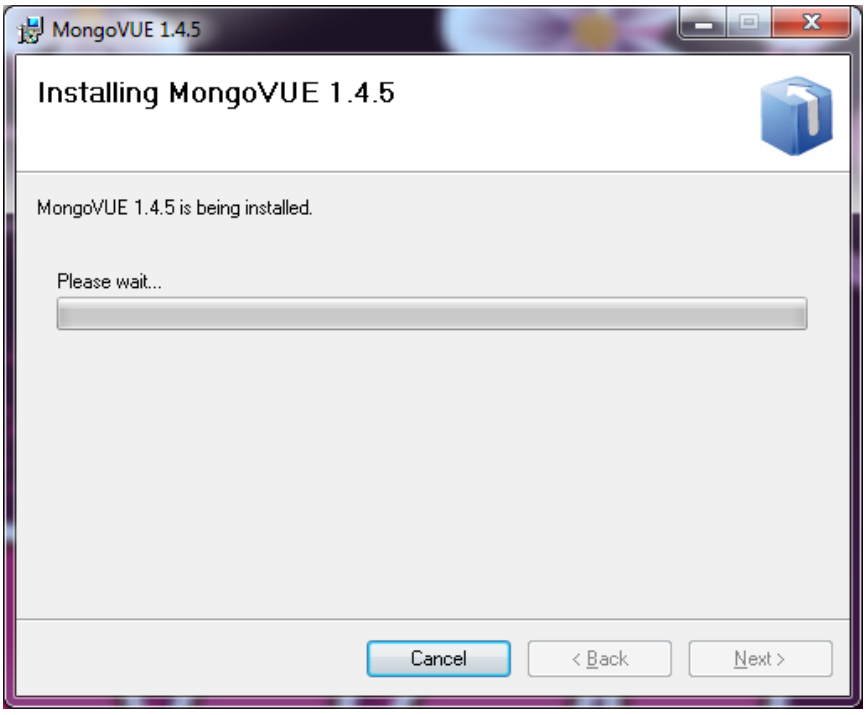

Ilustración 28 - Captura de Pantalla

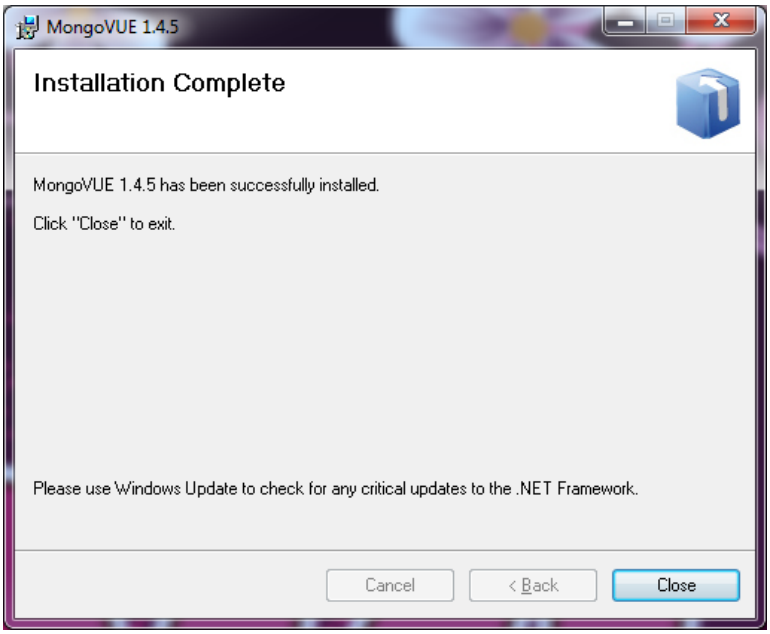

Ilustración 29 - Captura de Pantalla

Una vez instalada la herramienta lo único que queda por hacer es configurar la conexión, para ello el programa cuenta con un asistente que nos permite configurar la conexión.

4.8.2 REALIZAR UNA CONEXIÓN DESDE MONGOVUE

Para configurar una conexión desde la herramienta seguir los siguientes pasos:

1. Dar clic en el botón connect de la parte izquierda del programa:

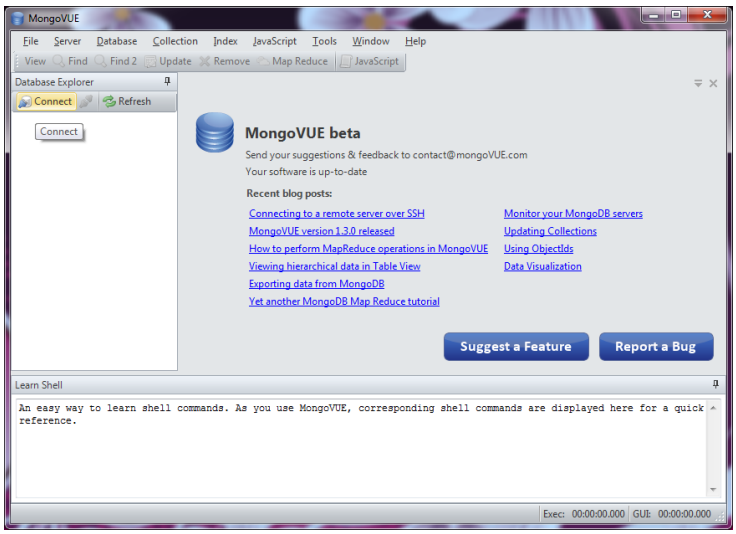

Ilustración 30 - Captura de Pantalla

2. En el Administrador de Conexiones dar clic en el botón “Agregar Nueva conexión”

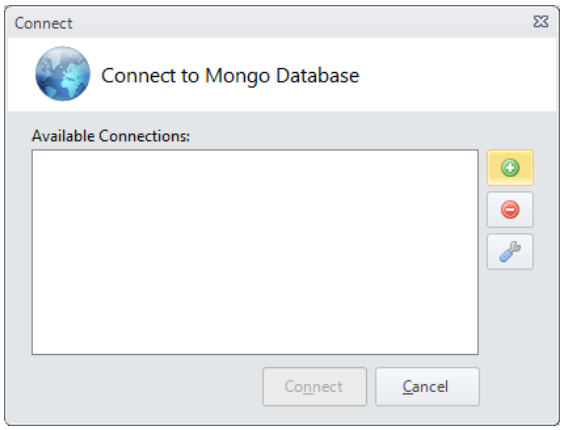

Ilustración 31 - Captura de Pantalla

3. Llenar los datos solicitados por el Asistente:

- Nombre de la conexión
- Servidor al que deseamos conectarnos
- Puerto a través del cual nos conectamos al servicio, generalmente es 27017
- Los campos de usuario y clave los usaremos únicamente si hemos establecido un usuario y una clave para MongoDB. Podemos probar la conexión mediante el botón “Test”

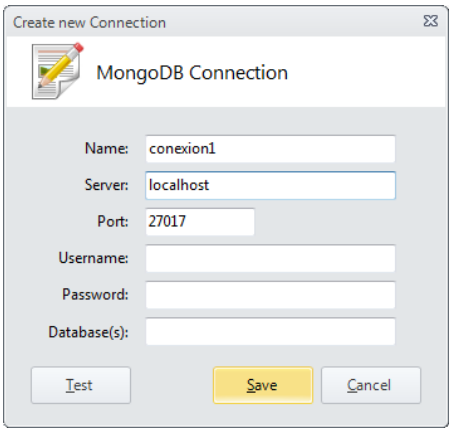

Ilustración 32 - Captura de Pantalla

Si la conexión está correcta tendremos el siguiente mensaje;

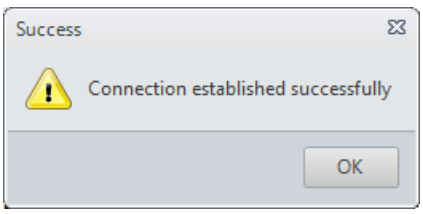

Ilustración 33 - Captura de Pantalla

4. Una vez que hemos completado el asistente de conexión podremos administrar la base de datos:

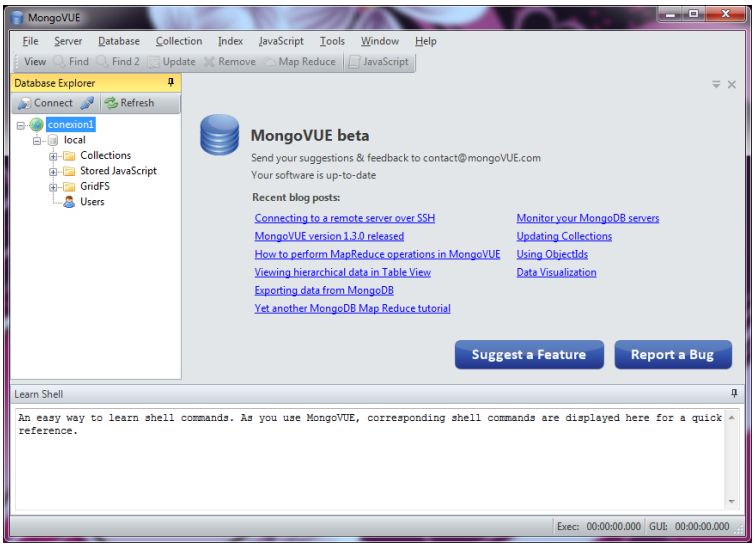

Ilustración 34 - Captura de Pantalla

4.8.3 CREAR UNA BASE DE DATOS DESDE MONGOVUE

Para crear una base de datos desde nuestra herramienta realizar los siguientes pasos:

1. Clic con el botón derecho sobre el nombre de nuestra conexión y seleccionar la opción **Add DataBase**

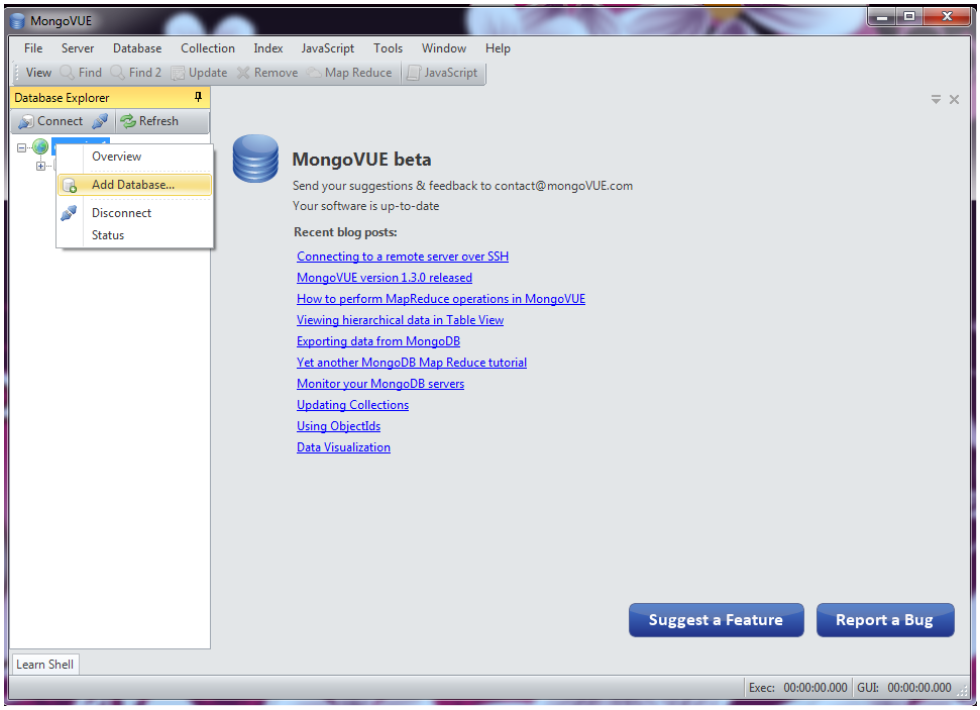

Ilustración 35 - Captura de Pantalla

2. Ingresar el nombre de la base de datos que queremos crear, y damos clic en el botón **OK**

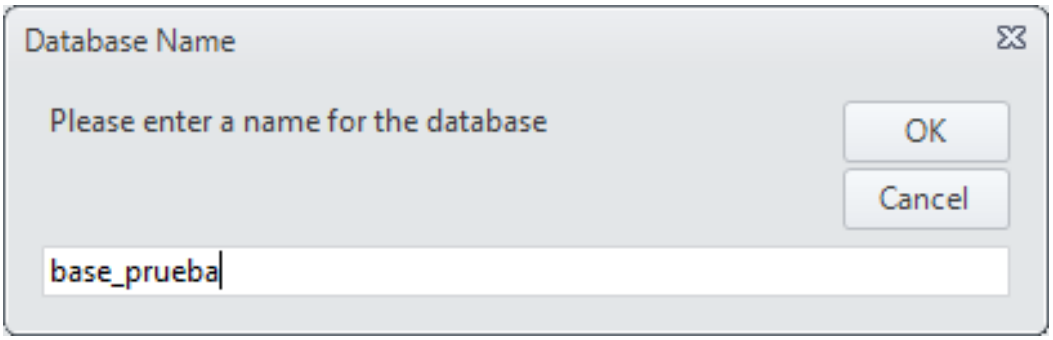

Ilustración 36 - Captura de Pantalla

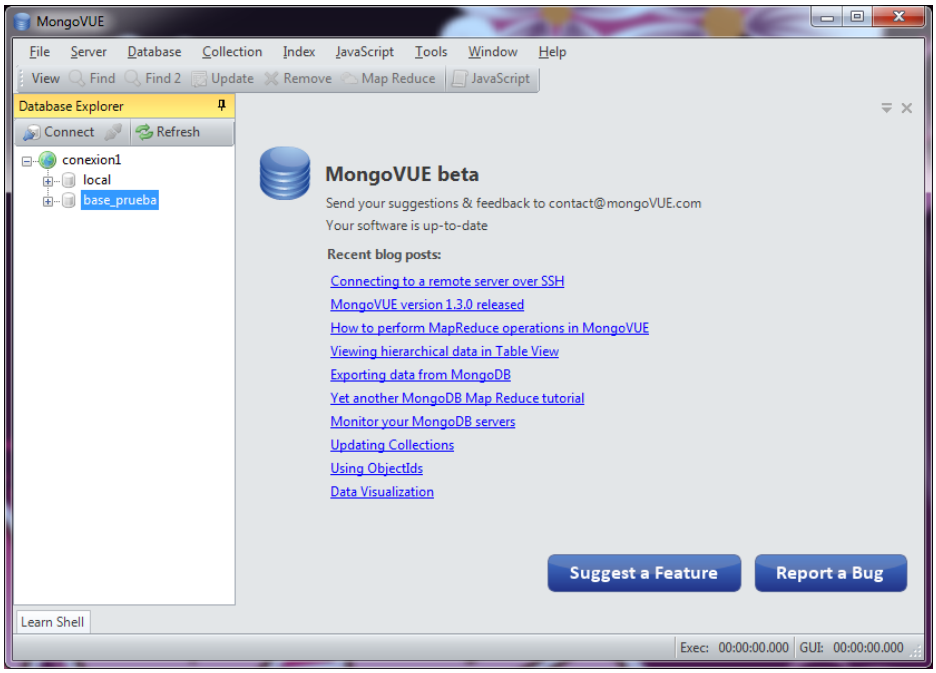

Ilustración 37 - Captura de Pantalla

4.8.4 CREAR COLECCIONES DESDE MONGOVUE

Para crear una colección desde la herramienta debemos seguir los siguientes pasos:

1. Expandimos la base de datos sobre la que queremos trabajar

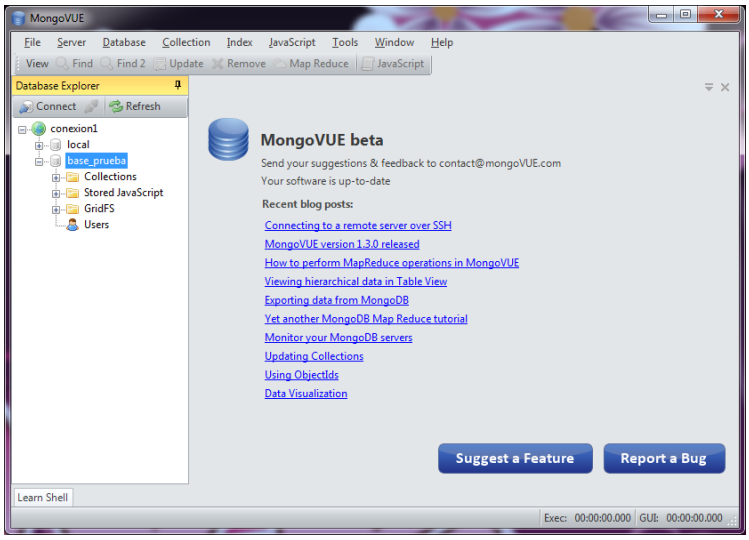

Ilustración 38 - Captura de Pantalla

2. Clic con el botón derecho sobre la base de datos en la que deseamos trabajar, y seleccionamos la opción **Add Collection**

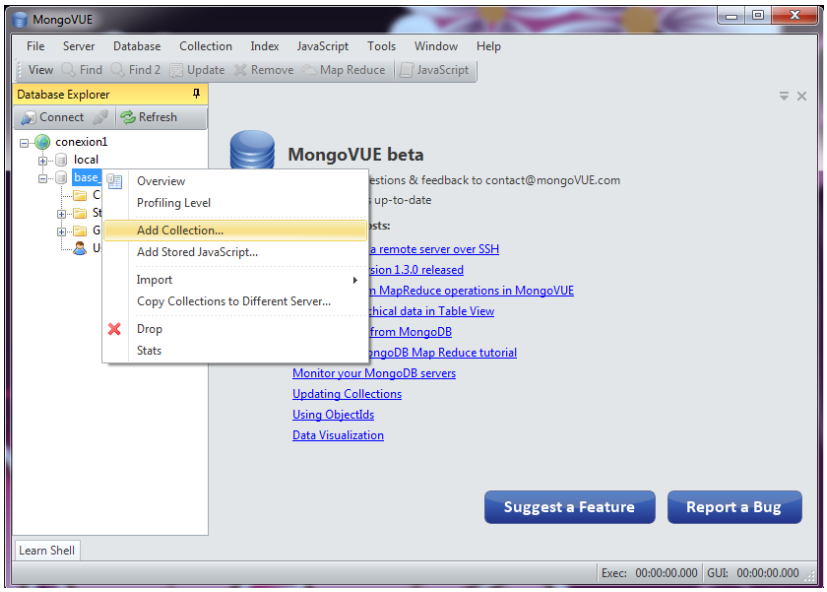

Ilustración 39 - Captura de Pantalla

3. Ingresar el nombre para nuestra nueva colección y dar clic en OK

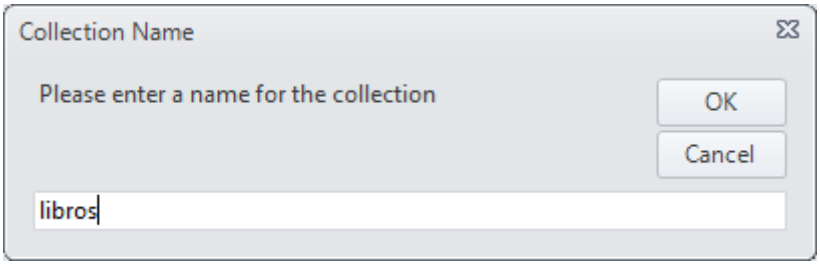

Ilustración 40 - Captura de Pantalla

4.8.5 CREAR DOCUMENTOS DESDE MONGOVUE

Para crear documentos desde la herramienta debemos seguir los siguientes pasos:

1. Clic con el botón derecho sobre la colección en la que deseamos crear un documento, y seleccionamos **Insert Document**

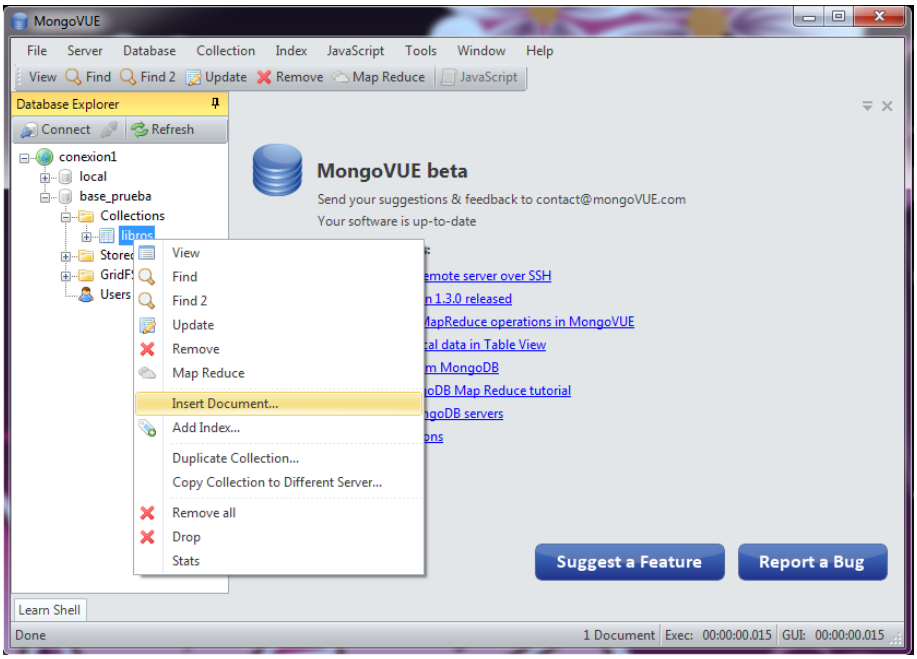

Ilustración 41 - Captura de Pantalla

2. Ingresamos el documento que deseamos insertar en formato JSON y damos clic en **Insert**

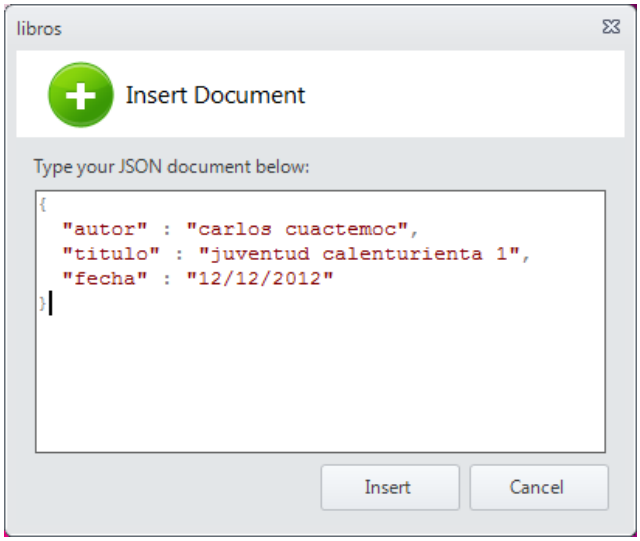

Ilustración 42 - Captura de Pantalla

3. A continuación podemos revisar los datos insertados, dando doble clic sobre el nombre de la colección.

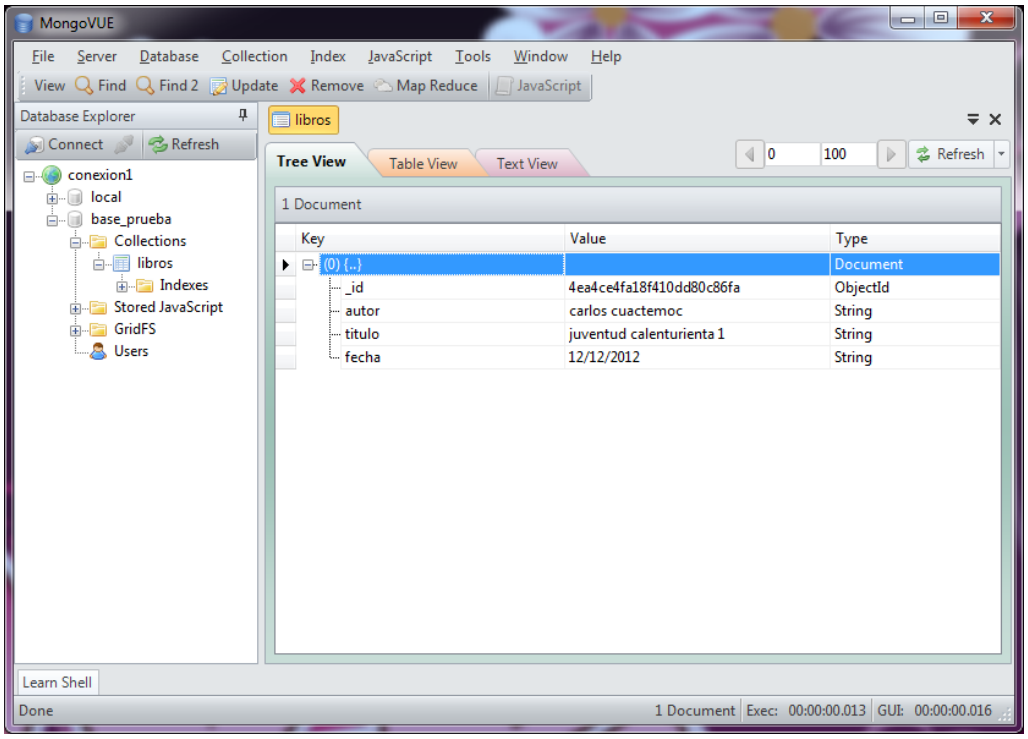

Ilustración 43 - Captura de Pantalla

4.8.6 ACTUALIZAR UN DOCUMENTO DESDE MONGOVUE

Para actualizar un documento desde MongoVUE es necesario seguir los siguientes pasos:

1. Clic derecho sobre el nombre de la colección que deseamos editar, seleccionaremos la opción **Send to: Update View**

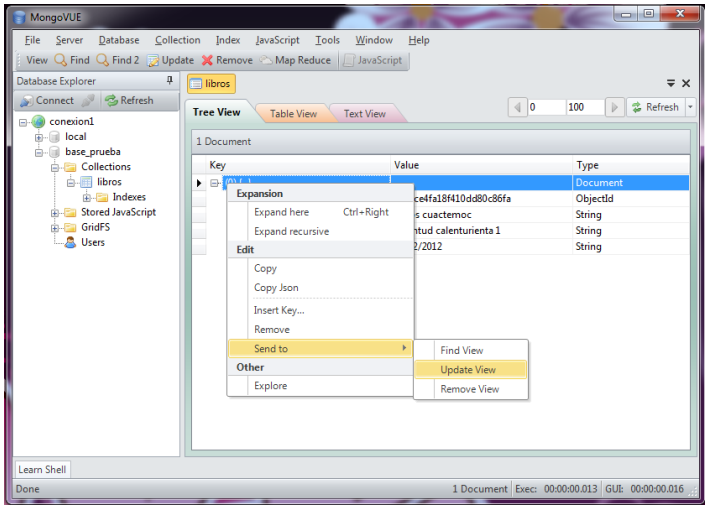

Ilustración 44 - Captura de Pantalla

2. Ingresamos la actualización de nuestro documento en formato JSON y luego damos clic en Update

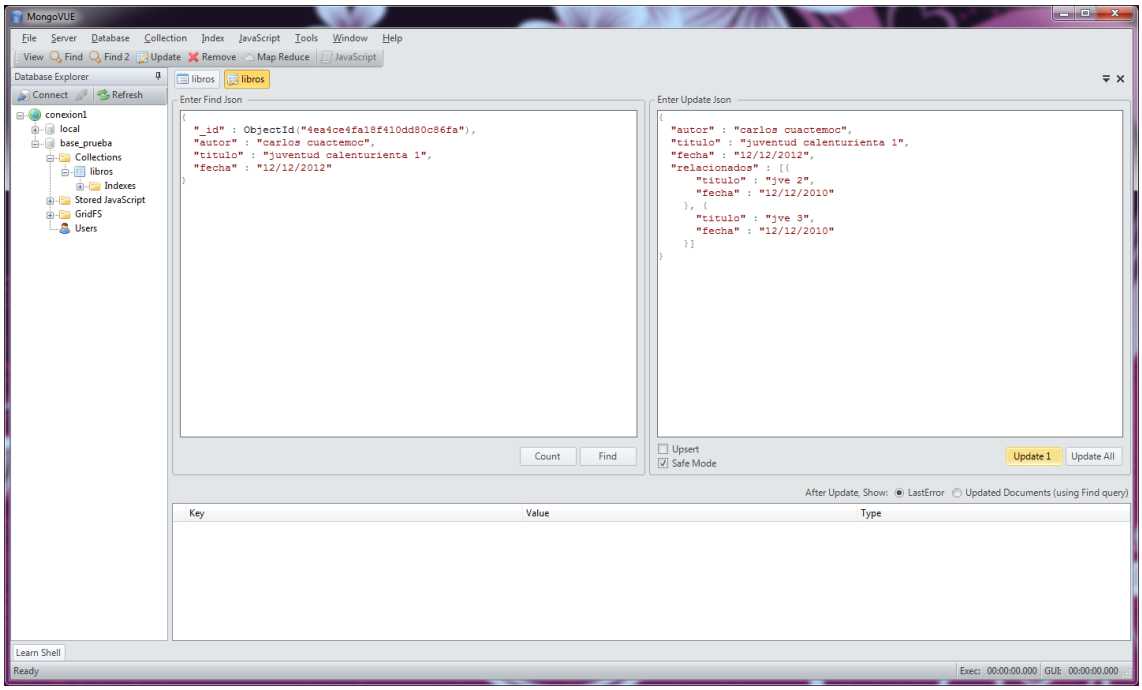

Ilustración 45 - Captura de Pantalla

4.8.7 ELIMINAR DOCUMENTOS DESDE MONGOVUE

Para eliminar un documento desde la herramienta debemos seguir los siguientes pasos:

1. Clic con el botón derecho sobre el documento que deseamos eliminar y clic en **Remove**

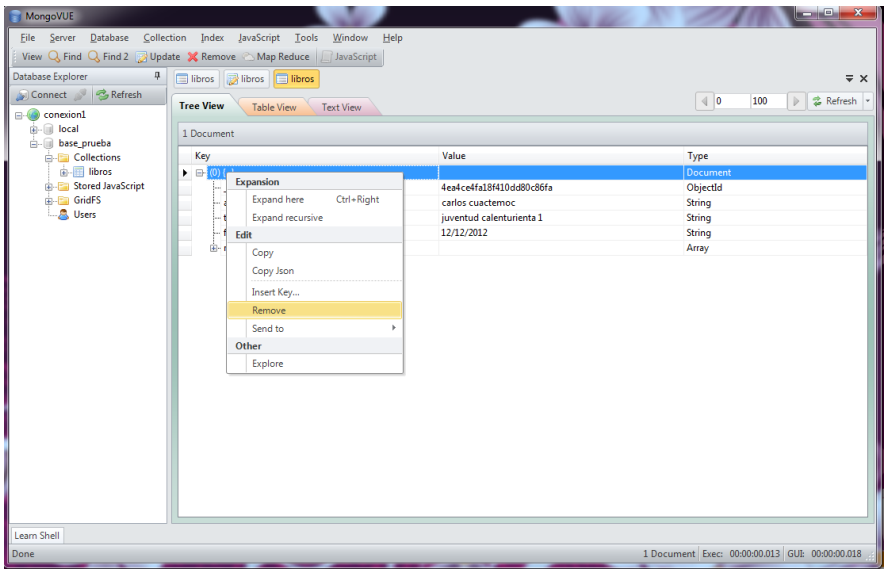

Ilustración 46 - Captura de Pantalla

2. Confirmamos la eliminación del documento dando clic en **Yes**

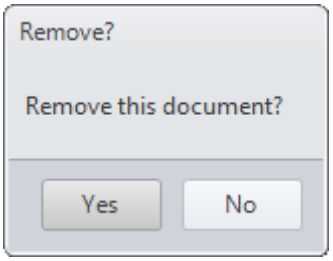

Ilustración 47 - Captura de Pantalla

4.9 PHP Y MONGODB

Como se mencionó anteriormente MongoDB cuenta con drivers nativos para un elevado número de lenguajes de programación, entre ellos PHP.

PHP se ha convertido en el lenguaje de Facto (por así decirlo) para el desarrollo web, su popularidad ha crecido tanto que la mayor parte de sitios web están desarrollados en dicho lenguaje, y algunas de las aplicaciones más conocidas como Facebook, entre otras la utilizan a gran escala.

El principal tema de esta investigación es precisamente demostrar el potencial que MongoDB tiene para ser implementado como base de datos para aplicaciones web, por lo que a continuación desarrollaremos algunas comparativas entre estructuras de datos nativas de MongoDB (en JSON) y su representación en forma el lenguaje natural (PHP).

4.9.1 ESTRUCTURAS (JSON Y PHP)

Para apreciar la potencia del driver de MongoDB para los lenguajes de programación, en este caso para PHP vamos a realizar una comparativa entre una estructura e JSON y su equivalente en lenguaje nativo PHP.

Consideremos la estructura del primer ejemplo:

JSON	PHP
{ "Tipo": "CD", "Artista": "Chayanne", "Titulo": "Me Enamore de Ti", "Genero": "Baladas",	$media = array ("Tipo" => "CD", "Artista" => "Chayanne", "Titulo" => "Me Enamore de Ti",

"ListaCanciones":
[
{
"NumeroPista" : "1",
"Titulo" : " Me Enamore de Ti "
},
{
" NumeroPista " : "2",
" Titulo " : "Tu Boca"
}
]
}
	"Genero" => "Baladas",
	"ListaCanciones" => array
	(
	array
	(
	"NumeroPista" => "1",
	"Titulo" => " Me Enamore de Ti "
	),
	array
	(
	" NumeroPista " => "2",
	" Titulo " => "Tu Boca"
	)
	)
	)

Tabla 12 - Comparativa entre PHP y JSON

Como podemos apreciar las dos estructuras se parecen mucho, se notan dos diferencias principalmente:

- El reemplazo de "**:**" por **"=>"**
- El reemplazo de "**{ }**" y "**[ ]**" por "**( )**"

4.9.2 CLASES DE MONGODB EN EL DRIVER DE PHP

El Driver de MongoDB para PHP se compone principalmente por cuatro clases principales, obviamente el driver tiene muchas más clases, pero las más importantes son cuatro:

- **Mongo:** Sirve para inicializar conexiones con la base de datos y nos provee de los siguientes métodos:
  - connect()
  - close()
  - listDBs()
  - selectDBs()
  - selectCollection()
- **MongoDB:** Nos permite interactuar con la base de datos y nos provee de los siguientes métodos:
  - createCollection()
  - selectCollection()
  - createDBRef()
  - getDBRef()
  - drop()
  - getGridFS()

- **MongoCollection:** Nos permite interactuar con las colecciones, nos provee de los métodos:
  - count()
  - find()
  - findOne()
  - insert()
  - remove()
  - save()
  - update()
- **MondoCursor:** Nos permite interactuar con los resultados de una consulta dados por el comando find(), nos provee de los siguientes métodos:
  - getNext()
  - count()
  - hint()
  - limit()
  - skip()
  - sort()

4.9.3 CONEXIONES A LA BASE DE DATOS

La mejor manera de entender cómo se realiza una conexión a la base de datos desde PHP es mediante un ejemplo. Para realizar una conexión debemos usar una instancia de la clase **Mongo.**

Para realizar una conexión a MongoDB y seleccionar una base de datos podemos hacerlo de la siguiente manera:

```
// Conexion a la base de datos
$c = new Mongo();

// Seleccionando la base de datos base_pruebas para trabajar
$c->selectDB("base_pruebas ");
```

También podemos usar una notación abreviada (->); el mismo fragmento de código anterior lo podemos hacer de la siguiente manera rápida:

```
// Conexion a la base de datos
$c = new Mongo();

// Seleccionando la base de datos base_pruebas para trabajar
$c->base_pruebas;
```

Una vez que hemos realizado una conexión a MongoDB y hemos seleccionado una base de datos, es necesario que también seleccionemos una colección con la que deseamos trabajar. Esto lo hacemos de la siguiente manera:

```
t// Conexion a la base de datos
$c = new Mongo();

// Seleccionando la base de datos base_pruebas y dentro de esta la colección media para trabajar
$c-> selectDB("base_pruebas")->selectCollection("media");
```

Si deseamos usar la notación rápida quedaría de la siguiente manera:

```
// Conexion a la base de datos
$c = new Mongo();

// Seleccionando la base de datos base_pruebas y dentro de esta la colección media para trabajar
$c -> base_pruebas -> media;
```

Pude darse el caso en el que necesitemos listar todas las bases de datos que estén disponibles en una instancia MongoDB. Esto lo podemos realizar a través del siguiente comando:

```
// Conexion a la base de datos
$c = new Mongo();

// Listar las bases de datos disponibles
print_r($c->listDBs());
```

Podemos aplicar una función parecida que nos permita listar las colecciones existentes en una base de datos.

```
// Conexion a la base de datos
$c = new Mongo();

// Listar todas las colecciones disponibles dentro de la base de datos base_pruebas
print_r($c-> base_pruebas ->listCollections());
```

Nota: El comando **print_r** es una función de PHP que permite sacar a pantalla el contenido de un arreglo.

Luego de interactúa con nuestra base de datos, es necesario que cerremos la conexión establecida. Usando el driver de MongoDB, generalmente no es necesario que cerremos la conexión manualmente, pues el driver se encarga de cerrar una conexión abierta inactiva de forma automática. De todas formas podemos cerrar la conexión manualmente si lo deseamos mediante el siguiente código:

```
// Conectando al servidor
$c = new Mongo();

// Cerrando la conexión
$c->close();
```

4.9.4 INSERCIONES DE DATOS

Ya hemos aprendido como establecer una conexión con nuestro servidor, seleccionar bases de datos y seleccionar colecciones; ahora vamos a realizar la creación o inserción de un documento (recordemos que un documento es el equivalente a un registro o fila en una base de datos relacional).

Para crear un documento en MongoDB no necesitamos obligatoriamente conocer a profundidad el formato de JSON, en lugar de ello el driver para PHP abstrae dicha dificultad y nos permite que para crear un documento simplemente lo hagamos en forma de un arreglo con llaves y valores:
Miremos el siguiente ejemplo:

```
$media = array
(
   "Tipo" => "CD",
   "Artista" => "Chayanne",
   "Titulo" => "Me Enamore de Ti",
   "Genero" => "Baladas",
   "ListaCanciones" => array
    (
        array
        (
          "NumeroPista" => "1",
          "Titulo" => " Me Enamore de Ti "
        ),
        array
        (
          " NumeroPista " => "2",
          " Titulo " => "Tu Boca"
        )
   )
)

// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Insertar el documento “$media” en la coleccion “$coleccion”
$coleccion ->insert($media);
```

Adicionalmente a la operación de creación realizada en el ejemplo anterior, la función **insert()** también recibe dos parámetros extras:

- **safe:** Este parámetro acepta valores de:
    - **true**: entonces la instrucción espera una confirmación de éxito de la operación de inserción realizada antes de continuar con la ejecución del resto de código PHP. Cuando utilizamos true, la función **insert()** nos devolverá como resultado un arreglo indicando el resultado de la operación.
    - **false:** (valor por defecto si no lo definimos) entonces la instrucción permitirá que se ejecute el resto del código sin esperar una confirmación de la base de datos.

- **fsync:** Este parámetro acepta valores de:
    - **true**: la instrucción de insert() escribirá los datos en el disco duro antes de confirmar el éxito de la operación (recordemos que MongoDB escribe los datos en la RAM para luego hacerlos persistentes). Cuando usamos true automáticamente
    - **false:** (valor por defecto si no lo definimos) si usamos este valor los datos serán escritos primero a la RAM para luego hacerlos persistentes y el sistema nos devolverá una confirmación de éxito aun cuando los datos no hayan sido escritos en el disco duro.

El ejemplo anterior usando la opción safe:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Creamos un arreglo con los parámetros
$opciones = array("safe" => True);

// Insertar el documento “$media” en la coleccion “$coleccion”
$coleccion ->insert($media, $opciones);
```

4.9.5 BUSQUEDA DE DOCUMENTOS

Para realizar una búsqueda de varios documentos debemos usar la función **find().** Esta función no solo nos permite realizar búsquedas que nos entreguen varios documentos como resultado, en lugar de ello también nos permite que filtremos los resultados en base a criterios personalizados.

4.9.5.1 BUSQUEDAS QUE DEVUELVEN UN SOLO DOCUMENTO

Realizar una búsqueda que devuelva un único documento es relativamente fácil gracias a la función **findOne(),** esta función devuelve como resultado un arreglo con los datos del documento en cuestión.

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Definimos los criterios de búsqueda
$artista = array(“Artista” => “Chayanne”);

// Busqueda del docuemento en función al criterio
print_r($coleccion->findOne($artista));
```

4.9.5.2 BUSQUEDAS QUE DEVUELVEN VARIIOS DOCUMENTOS

Mientras que la función **findOne**() nos permite encontrar un documento específico, la función **find**() nos permite buscar conjuntos de documentos o un documento específico en función de los parámetros que pongamos. Los datos devueltos por la función **find**() generalmente vienen en un arreglo dentro de un cursor, para listar los datos o realizar cualquier otra forma de visualización debemos recorrer el cursor de arreglos uno por uno, para ello usamos la función **getNext**().

A continuación una ilustración de la función:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Ejecutamos la consulta y almacenamos los resultados temporalmente en
la variable $cursor
$cursor = $coleccion ->find();

// Recorremos el cursos y visualizamos su contenido uno por uno
while ($document = $cursor->getNext())
{
   print_r($document);
}
```

4.9.5.3 FILTROS Y OPERADORES DE CONSULTA

Como mencionamos anteriormente la función **find**() también nos permite que ingresemos filtros, criterios de orden, etc. En los resultados que la búsqueda bride.

A continuación un ejemplo de una búsqueda parametrizada:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Creacion de los paramteros de busqueda
$artistas = array("Media.Artistas " => "Chayanne");

// Ejecutamos la consulta y almacenamos los resultados temporalmente en
la variable $cursor
$cursor = $coleccion ->find($artistas);

// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

4.9.5.4 CRITERIOS DE ORDEN, LÍMITES, Y SKIPPING EN UNA CONSULTA

MongoDB provee como un complemento al comando **find**() una serie opciones entre las que podemos mencionar: sort(), limit() y skip().

Un ejemplo para aplicar un criterio de orden a una consulta:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find();

// Creamos el criterio de orden
$cursor ->sort(array(“Artista”=>1));


// Recorremos el cursos y visualizamos su contenido uno por uno
while ($document = $cursor->getNext())
{
    print_r($document);
}
```

Si deseamos limitar el número de documentos que una consulta puede traernos, podemos usar la función **limit**():

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find();

// Indicamos a la base de datos que necesitamos
$cursor ->limit(2)


// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

Si deseamos simplemente obviar cierto número de documentos podemos usar la función skip:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find();

// Indicamos cuantos registros oviamos
$cursor -> skip(2)


// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

4.9.5.5 OPERADORES EN LAS CONSULTAS DE MONGODB

MongoDB al igual que toda base de datos nos permite usar criterios de búsqueda ciertos operadores como:

- **$lt:** Operador equivalente a <.
- **$gt:** Operador equivalente a >.
- **$lte:** Operador equivalente a <=.
- **$gte:** Operador equivalente a >=.

Podemos realizar un ejemplo para indicar como se aplicarían dichos operadores en una consulta normal.

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Parametros de busqueda
$condicionales = array(“Publicado En” => array(‘$gt’ => 2009));

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find($condicionales);


// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

También disponemos de un par de operadores más:

- **$ne**: Este operador nos permite obtener todos los documentos que no tengan el valor indicado en dicha variable. Su aplicación es muy sencilla:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Parametros de busqueda
$condicionales = array(“Publicado En” => array(‘$ne’ => 2009));

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find($condicionales);


// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

- **$in**: Nos permite encontrar todos los documentos que cumplan con los datos contenido en un array, por ejemplo:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Parametros de busqueda
$condicionales = array(“Publicado En” => array(‘$in’ =>array (2009, 2010, 2011)));

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find($condicionales);

// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

- **$all**: Este operador devuelve todos los documentos que cumplan con todos los parámetros ingresados. A continuación un ejemplo:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Parametros de busqueda
$condicionales = array(“Publicado En” => array(‘$all’ =>array (2009, 2010, 2011)));

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find($condicionales);

// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
    print_r($documento);
}
```

- **$or**: Este operador nos devuelve todos los documentos que cumplan con cualquiera de los parámetros ingresados en el **$or**. A continuación un ejemplo:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Parametros de busqueda
$condicionales =
array(
   ‘$or’ =>array(
                  array(“Publicado En” => 2009),
                  array( “Autor” =>“Chayanne”)
                  )
      );

// Ejecutamos la consulta y almacenamos los resultados temporalmente en la variable $cursor
$cursor -> $coleccion->find($condicionales);

// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
   print_r($documento);
}
```

4.9.5.6 VALIDAR SI UN DOCUMENTO EXISTEN EN BASE A UN CRITERIO

Generalmente es importante que podamos verificar la existencia de un determinado atributo en los documentos pertenecientes a una colección. Aunque a priori no parezca útil, en realidad lo es, por ejemplo podríamos buscar todos los clientes que no tengan Email.

MongoDB nos provee de un comando que nos permite realizar dicha validación de forma muy sencilla:

```
// Conectandose a la base de datos
$c = new Mongo();

// Seleccionamos la coleccion “Media”
$coleccion = $c->base_pruebas->media;

// Buscamos los documentos que no tengan el atributo disquera
$condicionales = array(“Media.Disquera”=> array(“$exist” => false))

// Ejecutamos la consulta y almacenamos los resultados temporalmente en
la variable $cursor
$cursor -> $coleccion->find($condicionales);

// Recorremos el cursos y visualizamos su contenido uno por uno
while ($documento = $cursor->getNext())
{
   print_r($documento);
}
```

BIBLIOGRAFIA

Wikipedia, (2011), http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

Wikipedia, (2011), http://es.wikipedia.org/wiki/Software_como_servicio

Wikipedia, (2011), http://es.wikipedia.org/wiki/Amazon.com

Wikipedia, (2011), http://es.wikipedia.org/wiki/Google_Maps

Juan Carlos Garcia Candela (2010) Universidad de Alicante, http://www.opiniontecnologica.com/aplicaciones-informaticas/130-bases-de-datos-nosql-y-escalabilidad-horizontal.html

Ian Eure, http://about.digg.com/blog/looking-future-cassandra

Wikipedia, (2011), http://es.wikipedia.org/wiki/Web_2.0

Wikipedia, (2011), http://es.wikipedia.org/wiki/Software_como_servicio

Zettapedia, http://es.zettapedia.com/dynamo-sistema-de-almacenamiento.htm

Wikipedia, (2011), http://es.wikipedia.org/wiki/SQL

Wikipedia, (2011), http://es.wikipedia.org/wiki/Mapeo_objeto-relacional

Oracle, http://www.oracle.com/es/products/database/options/rac/index.html

http://antares.inegi.org.mx/metadatos/metadat1.htm

10GEN.Inc, http://www.mongodb.org/downloads