IMPLEMENTACIÓN DE UN FRAMEWORK PROTOTIPO DE DESARROLLO DE APLICACIONES WEB PARA UN MOTOR DE ALMACENAMIENTO NO RELACIONAL, QUE PERMITA EL MAPEO DE OBJETOS

ROGER CALDERON MORENO

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERIA
2016

IMPLEMENTACIÓN DE UN FRAMEWORK PROTOTIPO DE DESARROLLO DE APLICACIONES WEB PARA UN MOTOR DE ALMACENAMIENTO NO RELACIONAL, QUE PERMITA EL MAPEO DE OBJETOS

ROGER CALDERON MORENO

Trabajo de grado presentado como requisito parcial para optar por el titulo de:
Magister En Software Libre

ASESOR: M.Sc. DANIEL ARENAS SELEEY

UNIVERSIDAD AUTONOMA DE BUCARAMANGA
FACULTAD DE INGENIERIA
2016

Nota de aceptación

_____

_____

_____

Jurado

_____

Jurado

_____

Bucaramanga, 25 de abril de 2016

A mi esposa e hijos, quienes fueron testigos del trabajo realizado a lo largo de mi formación y a los cuales no les pude dedicar el tiempo que se merecen.

AGRADECIMIENTOS

Doy gracias a Dios, que en su infinita misericordia me ha permitió iniciar este proyecto de formación y terminarlo satisfactoriamente.

Al ingeniero M.Sc. DANIEL ARENAS SELEEY director de la Maestría en Software Libre, quien me oriento y corrigió a lo largo del desarrollo de este proyecto.

Al Consejo de Facultad de Ciencias Básicas e Ingeniería de la Universidad de los Llanos, quienes me colaboran semestralmente con un apoyo económico para el pago de la matricula.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

TABLA DE GRÁFICAS

RESUMEN

Palabras claves: base de datos, nosql, mongodb, mapeo objeto relacional, persistencia, java, jpa, hibérnate y framework.

El presente proyecto surgió como una iniciativa académica en la cual se observa que las áreas de conocimiento en el desarrollo de software bajo el paradigma de programación Orientada a Objetos está confrontado por un modelo de almacenamiento de datos de tipo relacional lo que plantea dos escenarios diferentes que los desarrolladores tratan de mitigar a través de conversiones entre tipos o utilizando herramientas intermedias como el mapeo de objetos relacional que traen ciertas ventajas y desventajas, y a lo cual se planteo dentro de este proyecto una solución a través de la utilización del almacenamiento de tipo no relacional o NoSQL, con la cual se dio una solución adicional a la formas de desarrollo antes mencionadas las cuales pueden ser de utilidad para la enseñanza de los estudiantes y profesionales que plantean soluciones de software.

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

Se definió como objetivo general: evaluar el grado de aceptación de un Framework Prototipo para Desarrollar Aplicaciones Web para un motor de almacenamiento no relacional, que permita el mapeo de objetos. Para consecución de los objetivos propuestos, se realizaron actividades como: diseñar y desarrollar un Framework Prototipo De Desarrollo De Aplicaciones Web, se aplicaron encuestas a una

población o grupo de profesionales desarrolladores de software de la región del Departamento del Meta, a los cuales se tomo una muestra no probabilística a 15 usuarios que pertenecen a diversas empresas del sector y posteriormente se procedió a la revisión y análisis de los resultados. Teniendo en cuenta los objetivos planteados en el proyecto, se definieron variables como: Interacción con herramientas de tipo mapeo objeto-relacional (ORM), Dificultad en el proceso de aprendizaje de las herramientas tipo ORM, ORM utilizadas, Conocimiento de los problemas de rendimiento de los ORM, Conocimiento de las BD NOSQL, BD NOSQL como medio de almacenamiento de objetos, MongoDB como repositorio de objetos, Concepto del Framework web para MongoDB.

Como conclusiones finales del desarrollo del proyecto, se plantearon las siguientes afirmaciones:

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

La población encuestada evidencia un conocimiento alto (93.3%) sobre las nuevas formas de almacenamiento de información denominadas Bases de Datos NoSql, y se resalta que han utilizado diversas tecnologías como: MongoDB, Redis y Cassandra, para el desarrollo de proyectos. También consideran que es posible almacenar en esta clase de motores no relaciones los datos de los objetos de nuestras aplicaciones respetando en todo momento su definición inicial.

La propuesta de almacenar objetos dentro del motor de almacenamiento MongoDB a través de su estructura basada en documentos fue aceptada por el 73.3% de los encuestados.

INTRODUCCIÓN

El presente proyecto surge como una iniciativa académica en la cual se observa que las áreas de conocimiento en el desarrollo de software bajo el paradigma de programación Orientada a Objetos está confrontado por un modelo de almacenamiento de datos de tipo relacional lo que plantea dos escenarios diferentes que los desarrolladores tratan de mitigar a través de conversiones entre tipos o utilizando herramientas intermedias como el mapeo de objetos relacional que traen ciertas ventajas y desventajas, y a lo cual se plantea dentro de este proyecto una solución a través de la utilización del almacenamiento de tipo no relacional o NoSQL, con la cual se pretende dar una solución adicional a la formas de desarrollo antes mencionadas las cuales pueden ser de utilidad para la enseñanza de los estudiantes y profesionales que plantean soluciones de software.

Los objetos de la aplicación que se desarrollaron se insertan en el motor de almacenamiento no relacional MongoDB, el cual almacena los datos en forma de documentos, que son pares de campos y valores como JSON. La estructura dinámica de documentos de MongoDB, se puede utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. De aquí, que la Consultora Gartner, Inc., ha publicado en el Octubre de 2014 el “Magic Quadrant for Operational Database Management Systems”, que representa la tendencia entre las empresas de tecnologías más influyentes en el mercado sobre sus expectativas sobre los motores de almacenamiento de datos relacionales y no relacionales, y allí han incluido a MongoDB.

Teniendo claro que los objetos de nuestra aplicación serán almacenados directamente en un motor de almacenamiento sin necesidad de hacer conversiones para ingresar o para leerlos, el siguiente aspecto que se propuso es realizar un Framework Prototipo para Construir Aplicaciones Web teniendo como base de almacenamiento MongoDB, de tal forma que el usuario final pueda construir sus aplicaciones con base en el modelo de objetos que abstrae de los requerimientos de la aplicación.

Se definió como objetivo general: Evaluar el grado de aceptación de un Framework Prototipo para Desarrollar Aplicaciones Web para un motor de almacenamiento no relacional, que permita el mapeo de objetos. Como objetivos específicos:

- Determinar el grado de comprensión y la facilidad de uso de las tecnologías como JPA - Hibernate para realizar una implementación completa de la programación orientada a objetos.

- Diseñar e Implementar un Framework Prototipo Para Desarrollar Aplicaciones Web: Formularios para captura de información y Reportes, que permita una representación de los objetos sobre un motor de almacenamiento no relacional.

- Determinar el grado de aceptación, sobre las ventajas y desventajas de este nuevo Framework prototipo de desarrollo en contraste con aquellos que utilizan el concepto de persistencia a través del mapeo objeto-relacional.

Para consecución de los objetivos propuestos, se realizaron actividades como: diseñar y desarrollar un Framework Prototipo De Desarrollo De Aplicaciones Web que permitan el almacenamiento de objetos en un motor de almacenamiento no relacional, se aplicaron encuestas a una población o grupo de profesionales

desarrolladores de software de la región del Departamento del Meta, a los cuales se propone tomar una muestra no probabilística a 15 usuarios que pertenecen a diversas empresas del sector y posteriormente se procedió a la revisión y análisis de los resultados.

1. REVISIÓN BIBLIOGRÁFICA O MARCO TEÓRICO

Dentro de esta área del proyecto se pretende explicar al lector una serie de conceptos que le permitan conocer con mayor claridad la problemática que se plantea alrededor del mismo, que tiene como fin demostrar que se puede utilizar un motor de almacenamiento no relacional como MONGODB para almacenar los objetos de nuestra aplicación, algo que el modelo relacional no es posible hacer debido a su naturaleza de tablas y relaciones entre ellas, dejando de un lado a los objetos que se utilizan en el lenguaje de programación, y forzándolos a realizar una conversión en ambas direcciones: aplicación → base de datos relacionales y base de datos relacionales → aplicación, lo cual se puede hacer a través de programación directa o a través de una API o frameworks para generar mapeo objetos (Ver Ilustración 1).

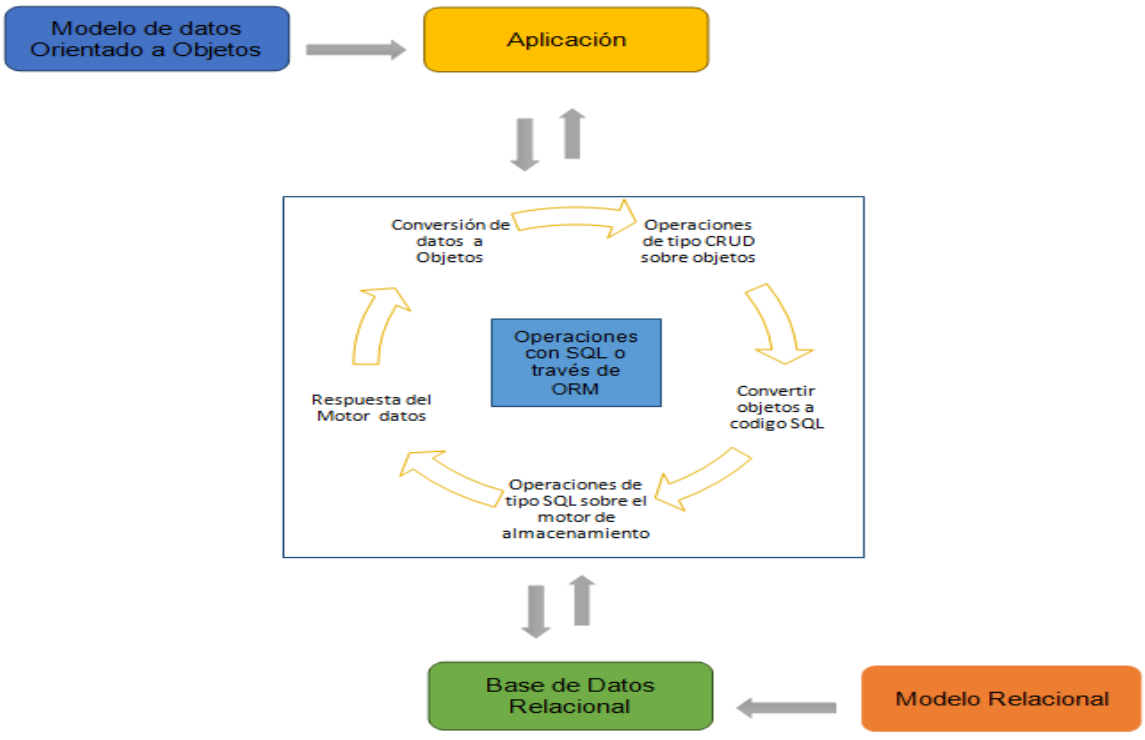


Ilustración 1: Proceso de desarrollo

Programación Orientada a Objetos. La orientación a objetos se puede definir como “una disciplina de ingeniería de desarrollo y modelado de software que permite construir más fácilmente sistemas complejos a partir de componentes individuales”. La Orientación a Objetos permite una representación más directa del modelo del mundo real, reduciendo fuertemente la transformación radical normal desde los requerimientos del sistema, definidos en términos del usuario, a las especificaciones del sistema, definidas en términos del computador.

La orientación a objetos proporciona mejores herramientas para:

- Modelar el mundo real de un modo más cercano a la perspectiva del usuario.
- Construir componentes reutilizables de software.
- Modificar y ampliar con facilidad la implementación de estos componentes sin afectar al resto de su estructura. (Fernando Alonso Amo, 2008)

Recordemos que dentro de la Programación Orientada a Objetos, se definen como entidades que tienen un determinado estado, comportamiento (método) e identidad:

- El estado está compuesto de datos o informaciones; serán uno o varios atributos a los que se habrán asignado unos valores concretos (datos).
- El comportamiento está definido por los métodos o mensajes a los que sabe responder dicho objeto, es decir, qué operaciones se pueden realizar con él.
- La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una

misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento. (Wikipedia - Mapeo objeto-relacional, 2015)

De aquí, que a partir del análisis que se realice del problema que intentemos solucionar a través de una aplicación, deberá partir de un modelo de objetos que representen el mundo del problema y de cómo se relacionan, este proceso de abstracción de entidades del mundo real a entidades objetos nos arrojara un modelo de datos orientado objetos. Sobre este modelo (Ver ilustración 2) que se puede expresar a través de un diagrama de clases se inicia la construcción de la aplicación.

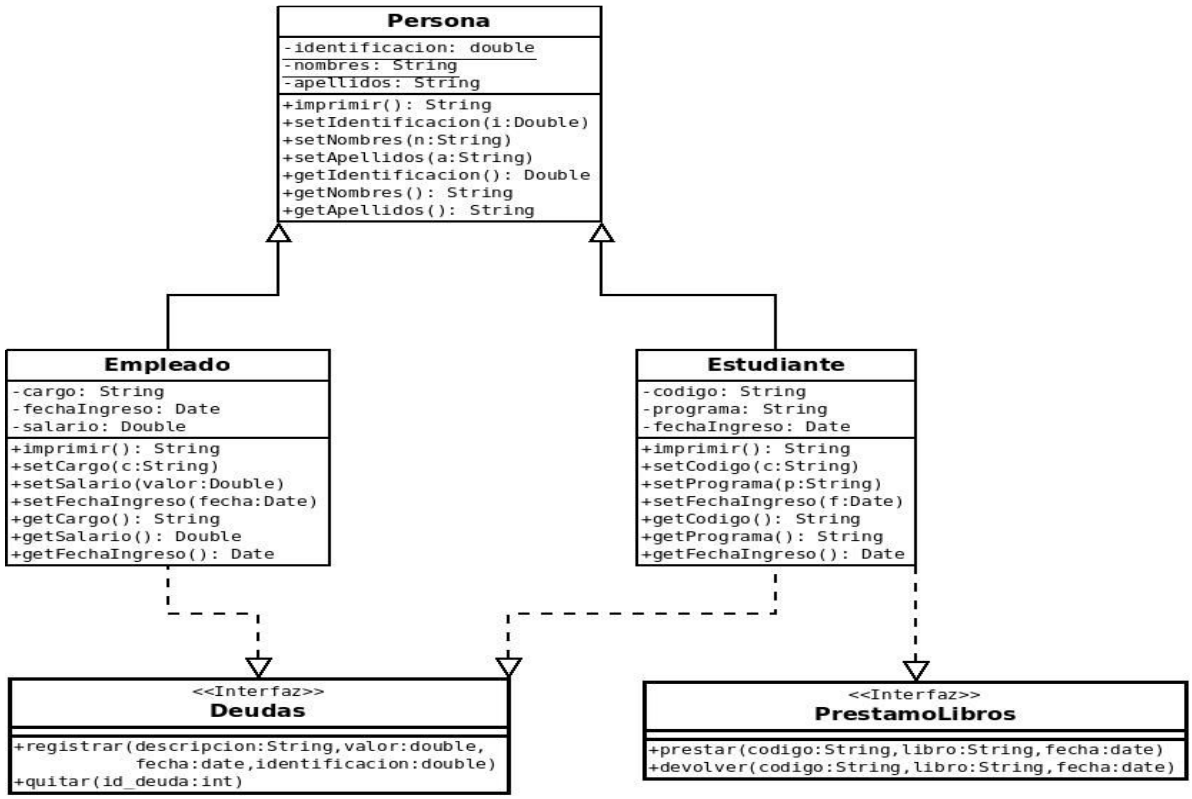


Ilustración 2. Diagrama de Clases sobre UML.

La interacción de los objetos que se crean dinámicamente dentro de la aplicación se da sin ninguna dificultad, la problemática inicia cuando se debe interactuar con la plataforma de almacenamiento de datos relacional, la cual es el estándar que la industria ha adoptado desde hace casi 30años, y funciona sobre el modelo de datos relacional, postulado sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), donde su idea fundamental es el uso de «relaciones» entre unas entidades (tablas), las cuales están compuestas por registros (cada fila de la tabla sería un registro), y columnas (también llamadas campos). (Management, 1990)

Como ejemplo, se plantea el siguiente diagrama relacional (Ver ilustración 3) resultante del proceso de un modelo entidad relación y la respectiva normalización de las tablas resultantes:

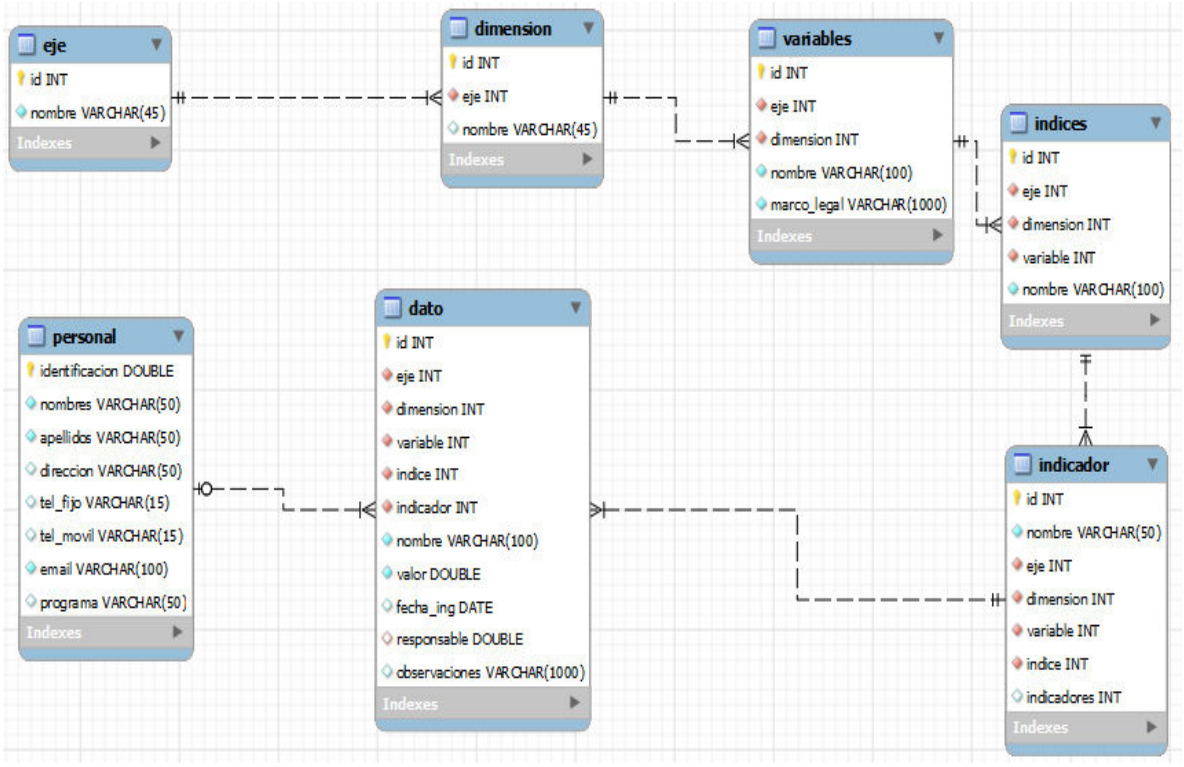


Ilustración 3. Ejemplo de un Diagrama Relacional.

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

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

Como se evidencia son modelos totalmente diferentes que tienen que coexistir dentro del software, para lo cual los desarrolladores han utilizado estrategias, de las cuales podemos resaltar: desarrollar los objetos trabajar con ellos, y en el momento de almacenar los datos a través de operaciones tipo CRUD (Crear, Obtener, Actualizar y Borrar) en el motor de almacenamiento, se convierten los objetos que se requieran a la representación del modelo relacional establecido y para obtener los datos de motor del almacenamiento y poder operar sobre ellos se debe realizar el proceso inverso. Este tipo de conversiones entre modelos se pueden implementar con programación manual de todas estas conversiones y/o través de APIS que permitan de alguna manera liberar al programador de estás conversiones a través del Mapeo de Objetos Relacional - ORM.

Entre las ventajas que ofrecen los ORM se encuentran: rapidez en el desarrollo, abstracción de la base de datos, reutilización, seguridad, mantenimiento del código, lenguaje propio para realizar las consultas. No obstante los ORM traen consigo algunas desventajas como el tiempo invertido en el aprendizaje. Este tipo de herramientas suelen ser complejas por lo que su correcta utilización requiere un espacio de tiempo a emplear en conocer su funcionamiento adecuado para posteriormente aprovechar todo el partido que se le puede sacar. Otra desventaja es que las aplicaciones suelen ser algo más lentas. Esto es debido a que todas las consultas que se hagan sobre la base de datos, el sistema primero deberá

transformarlas al lenguaje propio de la herramienta, luego leer los registros y por último crear los objetos. (Busto, 2011)

Dentro de los ORM que más fuerza han tenido y se mantienen vigentes esta: Hibernate. El cual es software libre, distribuido bajo los términos de la licencia GNU LGPL. Como todas las herramientas de su tipo, Hibernate busca solucionar el problema de la diferencia entre los dos modelos de datos coexistentes en una aplicación: el usado en la memoria de la computadora (orientación a objetos) y el usado en las bases de datos (modelo relacional). Para lograr esto permite al desarrollador detallar cómo es su modelo de datos, qué relaciones existen y qué forma tienen. Con esta información Hibernate le permite a la aplicación manipular los datos en la base de datos operando sobre objetos, con todas las características de la POO. Hibernate convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución.” (Wikipedia-Hibernate, 2015)

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

- Ámbito: Webservices =>FrameWork: Axis.
- Ámbito: Interfaz de Usuario Web Dinámica =>FrameWork: Ajax – DWR ,Laravel, OpenXava, Zend, Django, Ruby onRails entre otros.
- Ámbito: Procesos de Negocio => BPMS (Que son los FrameWorks) (agenda, 2012)

Los objetos de la aplicación que desarrollemos se almacenaran en el motor de almacenamiento no relacional MongoDB, el cual almacena los datos en forma de documentos, que son pares de campos y valores como JSON (Ver ilustración 4).

MongoDB. MongoDB (de la palabra en inglés “humongous” que significa enorme) es un sistema de base de datos NoSQL orientado a documentos, desarrollado bajo el concepto de código abierto. MongoDB forma parte de la nueva familia de sistemas de base de datos NoSQL. En vez de guardar los datos en tablas como se hace en las base de datos relacionales, MongoDB guarda estructuras de datos en documentos tipo JSON con un esquema dinámico (MongoDB llama ese formato BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. Los documentos son parecidos a las estructuras de los lenguajes de programación de las llaves asociados con valores, donde las llaves pueden contener otros pares de llaves y valores (por ejemplo, diccionarios, hashes, mapas y arrays asociativos).

Formalmente, los documentos de MongoDB son documentos BSON, que es una representación binaria de JSON con información de tipo adicional. (MongoDB, 2015). Dentro de las ventajas de utilizar documentos se puede apreciar:

- Los documentos corresponden con los tipos de datos nativos en muchos lenguajes de programación.
- Los documentos y matrices incorporadas reducen necesidad de combinaciones costosas.

MongoDB contienen colecciones, una colección está hecha de documentos y cada documento está compuesto de campos de tipo BSON, los cuales permiten tener dentro de cada campo otros subdocumentos, lo cual, facilita tener una estructura de la información dinámica (ver ilustración 5).

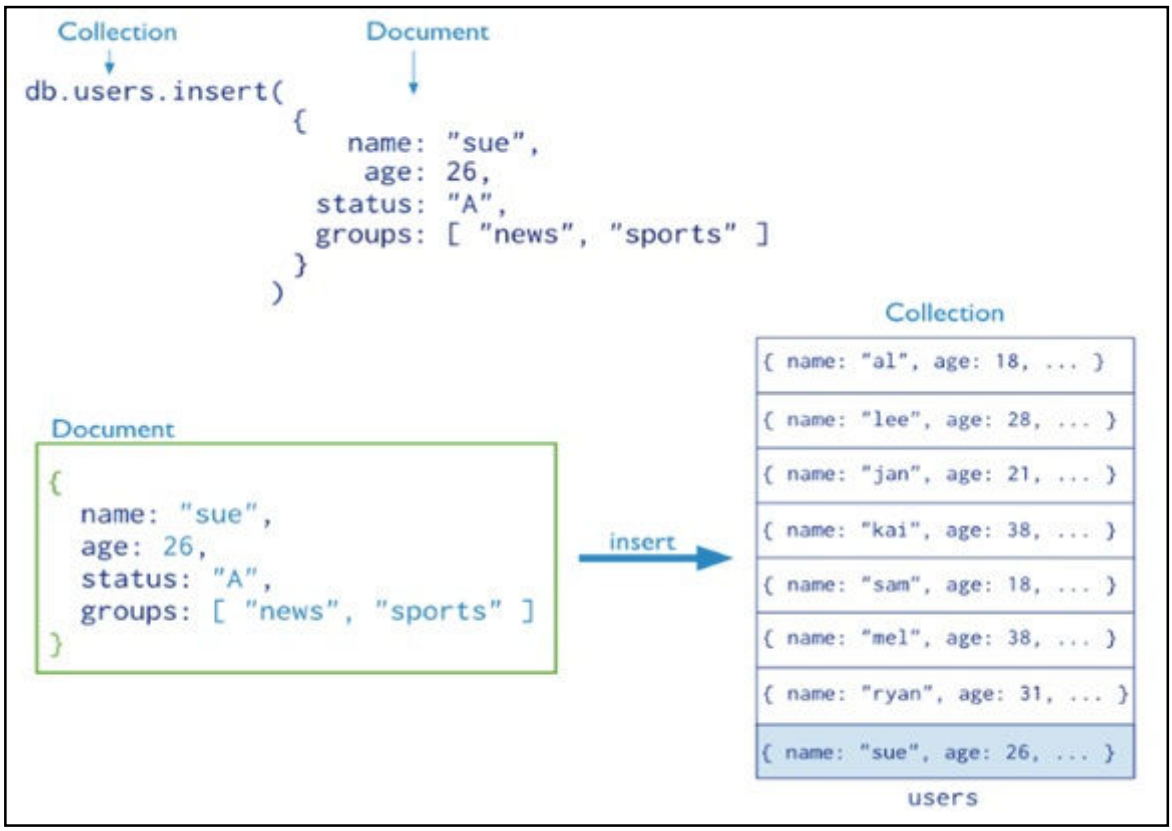


Ilustración 4: Ejemplo de inserción en una Collection. Imagen obtenida del Manual MongoDB.

La estructura dinámica de documentos de MongoDB, se puede utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales. De aquí, que la Consultora Gartner, Inc., ha publicado en el Octubre de 2014 el "Magic Quadrant for Operational Database Management Systems", que representa la tendencia entre las empresas de tecnologías más influyentes en el mercado sobre sus expectativas sobre los motores de almacenamiento de datos relacionales y no relacionales, y allí han incluido a MongoDB (Ver ilustración 6).

Ilustración 5.Magic Quadrant for Operational Database Management Systems - https://www.mongodb.com/lp/misc/gartner-mq-op-db-report

Se resalta de MongoDB: “Más de 1000 clientes, desde StartUps hasta las organizaciones empresariales, se basan en MongoDB para construir aplicaciones nunca antes posible. MongoDB es la comunidad de más rápido crecimiento en Big Data, con una gran base de usuarios y la posición trayectoria meteórica.” (MongoDB, 2014)

De MongoDB, podemos resaltar algunas características que los hacen atractivo a los desarrolladores:

- Sin esquemas fijos para organizar la información.
- Soportan la escritura de un gran volumen de datos simultáneamente.
- Auto-sharding y Escalado Horizontal sin afectar a la funcionalidad del producto.
- Soporta varios lenguajes de programación.
- De fácil Administración.
- Open Source.
- Buena documentación, comunidad creciente que aportan al proyecto.
- Uso de Memoria en vez del disco como principal ubicación de escritura.

En la actualidad existe un motor de almacenamiento muy similar a MongoDB denominado Apache CouchDB, comúnmente llamada CouchDB, es un gestor de bases de datos de código abierto, cuyo foco está puesto en la facilidad de su uso y en ser "una base de datos que asume la web de manera completa". Se trata de una base de datos NoSQL que emplea JSON para almacenar los datos, JavaScript como lenguaje de consulta por medio de MapReduce y HTTP como API. (couchdb)

CouchDB está clasificado como una base de datos " NoSQL ", un término que se hizo cada vez más popular a finales de 2009 y principios de 2010. Si bien este término es más bien una caracterización genérica de una base de datos o almacén de datos, sí define claramente un cambio respecto a las bases de datos tradicionales o basadas en SQL. Una base de datos CouchDB carece de un esquema o estructuras rígidas de datos predefinidos, tales como tablas. Los datos almacenados en CouchDB se organizan documento(s) de JSON . La estructura de los datos, o documento (s), puede cambiar de forma dinámica para adaptarse a las necesidades cambiantes. (Meet CouchDB)

Teniendo claro dónde vamos almacenar nuestros objetos, debemos recordar el concepto de persistencia: “es almacenar los objetos en algún medio de manera de

poder recuperarlos en el mismo estado en que los guardé. De esa manera cada objeto persistido trasciende más allá del ambiente donde fue creado". (Objetos, 2013)

En definitiva, como afirma Juan Mármol Castillo, "El programador debería disponer de algún medio para poder convertir el estado de un objeto a una representación adecuada sobre un soporte de información, que permitirá con posterioridad revivir o reconstruir el objeto, logrando que no debamos preocuparnos de cómo esta operación es llevada a cabo." (Castillo, 2013)

La persistencia de los objetos sobre MongoDB se realizara sobre la tecnología que ofrece la Plataforma Java Enterprise Edition (Java EE), las cuales nos permitirá mapear los objetos de nuestra aplicación y almacenarlos en un motor de almacenamiento no relacional. Dentro de la revisión de la literatura se encontró una API Mongo Engineque, que permite Mapear Objetos – Documentos para trabajar con MongoDB, algo similar a un ORM relacional (MongoEngine, 2014), del cual solo existe soporte para Python.

Teniendo claro que los objetos de nuestra aplicación serán almacenados directamente en un motor de almacenamiento sin necesidad de hacer conversiones para ingresar o para leerlos, el siguiente aspecto que se propone es realizar un Framework Prototipo para Construir Aplicaciones Web teniendo como base de almacenamiento MongoDB, de tal forma que el usuario final pueda construir sus aplicaciones con base en el modelo de objetos que abstrae de los requerimientos de la aplicación.

El framework se diseñara "para promover el acoplamiento débil y la estricta separación entre las piezas de la aplicación, de tal forma que resulte fácil hacer cambios en un lugar particular de la aplicación sin afectar otras piezas. Para ello se implementara el patrón de arquitectura de software Modelo-Vista-Controlador (MVC). En este patrón, el "Modelo" hace referencia al acceso a la capa de datos, la "Vista" se refiere a la parte del sistema que selecciona qué mostrar y cómo

mostrarlo, y el “Controlador” implica la parte del sistema que decide qué vista usar, dependiendo de la entrada del usuario, accediendo al modelo si es necesario*”.* (Kaplan-Moss.)*.*

2. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

En la actualidad diversos framework de desarrollo de aplicaciones, permiten el acceso a diferentes bases de datos relacionales a través del mapeo de objetos-relacional, con el fin de generar una relación entre los objetos que se definen en la aplicación y las entidades o tablas del modelo relacional, incluyendo en estas últimas sus correspondientes relaciones, todo con el único fin, de ocultar al desarrollador la complejidad implícita del SQL y la dependencia de todo el desarrollo a un motor de almacenamiento relacional, ya que en teoría, se podría cambiar en cualquier momento, sin afectar el desarrollo. Es de resaltar que estas implementaciones inicialmente son complejas en su aprendizaje, y además, terminan sacrificando al final el rendimiento de la aplicación por la conversión del modelo orientado a objetos al modelo relacional y viceversa, entre otros. (Wikipedia - Mapeo objeto-relacional, 2015).

Las bases de datos relacionales solo permiten almacenar datos de tipo primitivo o escalares como: enteros, varchar, boleanos, entre otros, por lo tanto para almacenar un dato compuesto como lo es un objeto el cual puede tener datos como: escalares, agrupaciones de datos escalares, otros objetos y datos binarios, es imposible realizarlo directamente, hay que des-componerlo, para adaptarlo al modelo relacional. Para que estos dos componentes: aplicaciones y motores de almacenamiento puedan funcionar junto, deben poder comunicarse intercambiando datos. En otras palabras, deben ser compatibles. Sin embargo, durante los últimos treinta años la evolución de estos dos componentes ha sido divergente, de forma que cada vez se ha hecho más difícil que colaboren en una misma aplicación. Así, desde los años 70 a la actualidad, las bases de datos utilizan un modelo teórico llamado “relacional”, que se ha convertido en un estándar y que es utilizado en la práctica para el desarrollo de aplicaciones de software.

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

Como una solución a esta dificultad surge el concepto de: mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relationalmapping, o sus siglas O/RM, ORM, y O/R mapping), el cual es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo) (Motores de Persistencia, 2005). Este mapeo de objeto-relacional, se puede implementar de forma manual a través de diversas funcionalidades que ofrecen los lenguajes de programación, como por ejemplo las implementación en Java de la API Hibernate o con JPA - Java Persistence API, o a través del uso de frameworks especializados como OpenXava, el cual permite el desarrollo rápido de aplicaciones web empresariales, donde se oculta la complejidad de trabajar con el SQL de la base de datos. La mayoría de los frameworks ORM generan el código requerido para las operaciones CRUD (Create-Read-Update-Delete) en las bases de datos automáticamente, lo cual reduce los tiempos de desarrollo, pero se

debe aclarar que los tiempos de aprendizajes inicialmente de la herramienta son altos debido a su complejidad de funcionamiento.

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

- Curva de aprendizaje: Las herramientas (o librerías) ORM suelen ser muy amplias, por lo que llegar a explotar su máximo rendimiento costará tiempo (Guardado, 2010).
- Menor rendimiento: Está claro que tener una capa tan enorme entre tu código y el sistema de base de datos, ralentizará un poco la aplicación (Mauro CALLEJAS CUERVO, 2011) (Guardado, 2010)
- Aumento de tiempo mapeando objetos y relaciones innecesario y creciente (Vondra, 2010).
- Realización de consultas complejas para relaciones sencillas y cargas de innecesaria de datos (Fink, 2010) (Vondra, 2010).
- Sistemas complejos: Normalmente la utilidad de ORM desciende con la mayor complejidad del sistema relacional. Es decir, si tienes una base de datos compleja, ORM también se te hará más complejo y perderás más tiempo adaptando tus clases que en un sistema de menor complejidad (Guardado, 2010).

Motivado por los avances que se ha estado generando en ámbito de las bases de datos NoSQL y conociendo las ventajas y desventajas que ofrece el desarrollo de software basado en ORM, se propone generar un Framework Prototipo de Desarrollo Web, que permita acceder a una base de datos de tipo No Relacional, que para nuestro caso será: MONGODB, teniendo en cuenta todas las bondades

que ella ofrece (ver Estado del Arte), y así tomar esta estructura interna basada en documentos con el fin de almacenar en ella la información de los objetos de la aplicación, con sus respectivos atributos y métodos, ya que este tipo de motores NoSQL no siguen una estructura rígida para el almacenamiento de la información, lo cual, es ideal para implementar un modelo de objetos, sin la necesidad de colocar una capa de software intermedia, y de esta forma obtener una estructura de almacenamiento de datos acorde a nuestras necesidades presentes y que se flexible a los requerimientos futuros (Ghosh, 2010). Con esta implementación, se pretende ofrecer una alternativa de acceso a datos, diferente a la propuesta por la persistencia de datos relacional, de tal forma que se generen ventajas como:

- Facilitar el proceso de desarrollo orientado a objetos, de tal forma que entre la aplicación resultante y en el motor de almacenamiento de datos, siempre se pueda mapear y trabajar todo el tiempo con objetos (ver ilustración 7).

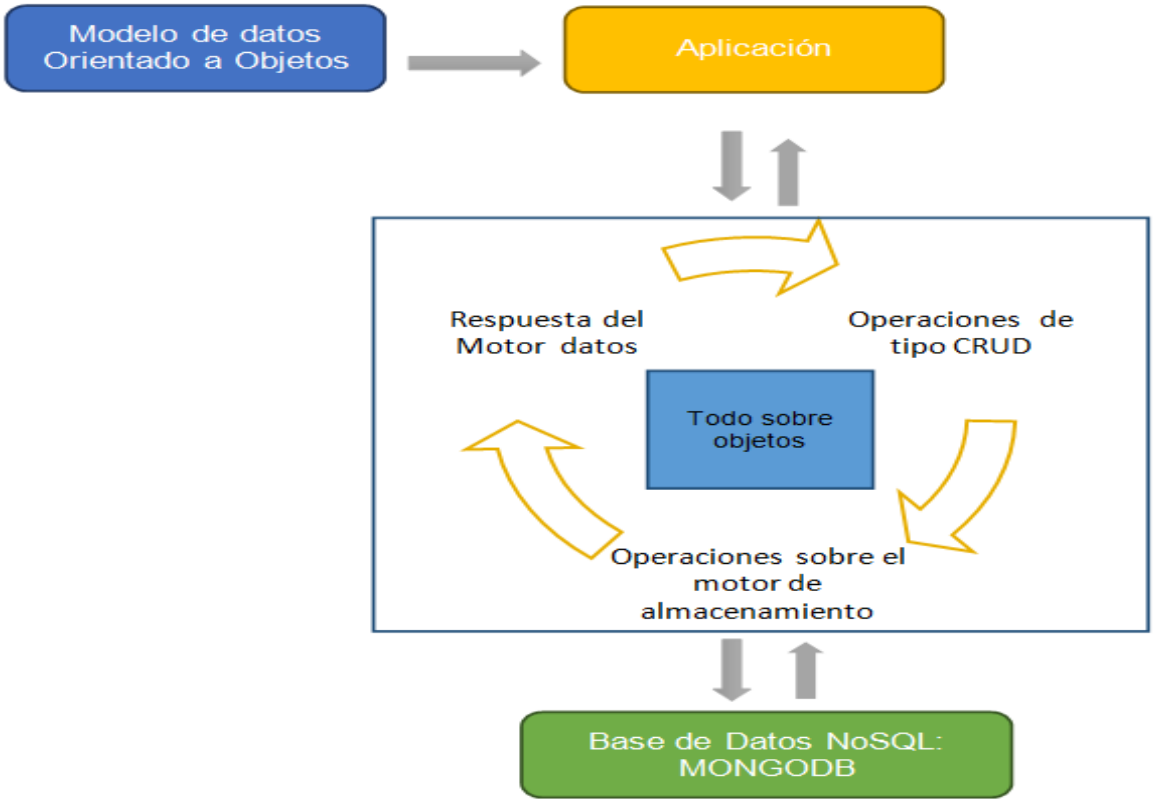


Ilustración 6: Ciclo de acceso a datos propuesto

- Se ofrece otra alternativa para el almacenamiento de datos en el desarrollo de aplicaciones que no depende de una base de datos relacional ni de una orientada a objetos, sino de una que permite manejar una estructura dinámica para el almacenamiento de los datos (Zhang, Song, & Liu, 2014).

Con este framework basado en documentos sobre MONGODB, podremos realizar las comparaciones con algunos frameworks tradicionales relacionales y así, lograr determinar el grado de aceptación por parte de los estudiantes, donde podremos determinar mejoras, respecto a: instalación y configuración, facilidad de aprendizaje, calidad de las aplicaciones desarrolladas (en el código fuente y la aplicación final), disminución de tiempos de desarrollo e implementación del paradigma orientado a objetos y seguridad de la capa de acceso a datos contra ataques de tipo SQL Injection, ya que no se trabaja con la SQL.

3. OBJETIVOS PLANTEADOS

Objetivo general

Evaluar el grado de aceptación de un Framework Prototipo para Desarrollar Aplicaciones Web para un motor de almacenamiento no relacional, que permita el mapeo de objetos.

Objetivos específicos

- Determinar el grado de comprensión y la facilidad de uso de las tecnologías como JPA - Hibernate para realizar una implementación completa de la programación orientada a objetos.

- Diseñar e Implementar un Framework Prototipo Para Desarrollar Aplicaciones Web: Formularios para captura de información y Reportes, que permita una representación de los objetos sobre un motor de almacenamiento no relacional.

- Determinar el grado de aceptación, sobre las ventajas y desventajas de este nuevo Framework prototipo de desarrollo en contraste con aquellos que utilizan el concepto de persistencia a través del mapeo objeto-relacional.

4. MÉTODO DE INVESTIGACIÓN

Para el desarrollo de este proyecto se planteo una metodología con un enfoque cuantitativo, el cual se oriento para la obtención de los datos en un estudio con encuestas. En la metodología se establecieron las siguientes etapas:

- **Selección de la población de la muestra.** Se determinó como población objetivo, a un grupo de profesionales desarrolladores de software de la región del Departamento del Meta, a los cuales se propone tomar una muestra no probabilística a 15 usuarios que pertenecen de diversas empresas del sector.

- **Definición de variables.** Teniendo en cuenta los objetivos planteados en el proyecto, se definieron variables como:

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

- **Diseño de los instrumentos (Encuestas).** Se determino realizar dos instrumentos, de los cuales, la Encuesta 1 se aplico antes de mostrar el Framework propuesto y la Encuesta 2 se aplico después de realizar las demostraciones del nuevo producto.

- **Recolección de la información.** Para el proceso de recolección de la información se realizaron las encuestas a través del envió de la misma a través de correo electrónico, las cuales eran diligenciadas en línea.

- **Análisis de la información recolectada.** La información recolectada se organice y se tabulo, con el fin de generar las graficas correspondientes y el análisis de los datos encontrados.

5. RESULTADOS DE LA INVESTIGACIÓN

5.1. Actividades

Para poder llevar a cabo proyecto planteado, se definieron las siguientes actividades:

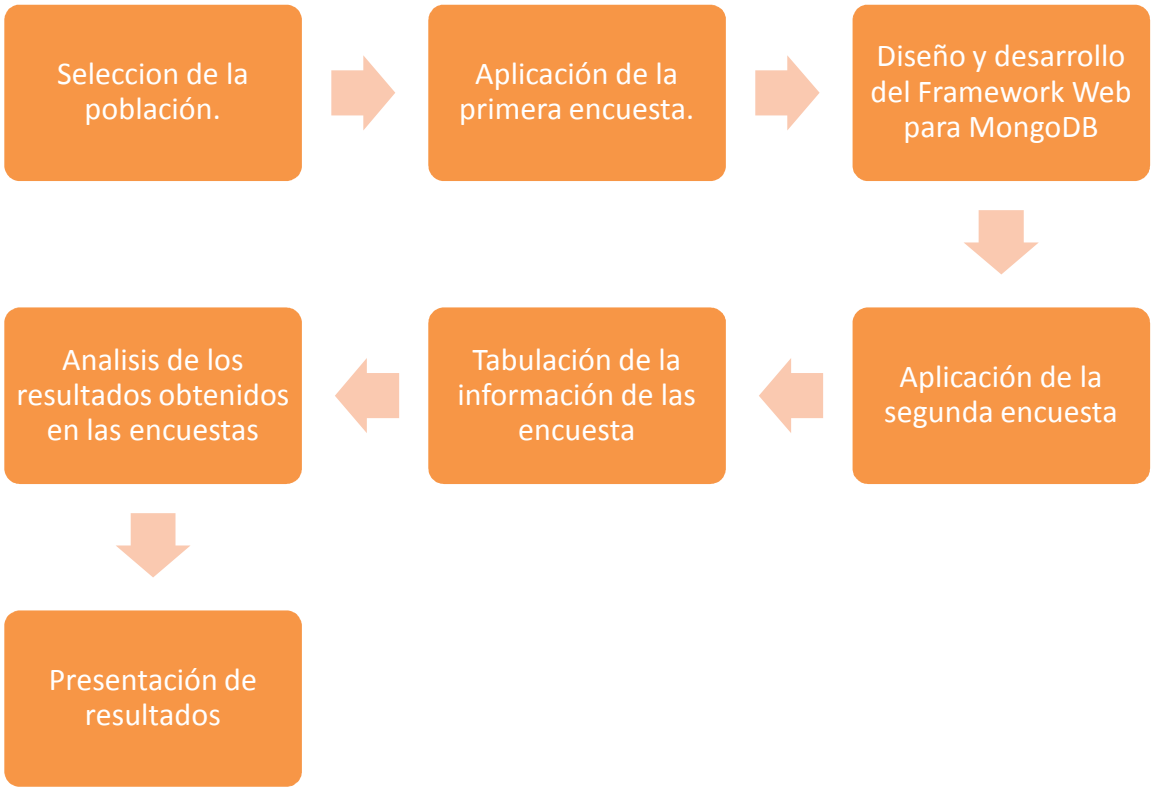


Ilustración 7: Actividades planteadas

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

encuestas. A continuación se describe la información suministrada por los participantes de la muestra, donde indican el cargo que desempeña dentro su empresa y las funciones que tradicionalmente tiene encomendadas.

Cargo que desempeña dentro de la empresa: *Ingeniera de desarrollo, Ingeniero desarrollador, Profesor, Desarrolladora de software, Ingeniero desarrollador senior, Desarrollador Web, Ingeniero de desarrollo, Profesional de apoyo, Ingeniero desarrollador, DBA, Android developer, Director operativo, Desarrollo, Docente, Support engineer*.

Describa las funciones que tiene encomendadas dentro de la empresa: *Analista, maquetador y desarrollador de aplicaciones web y móviles, Desarrollo, Levantamiento de requerimientos, Arquitectura y desarrollo de software, Análisis de requerimientos y procesos, desarrollo en diferentes tecnologías para llevar a cabo dichos requerimientos, y coordinación de grupos de desarrollo, Monitoreo de la Bases de datos de la Corporación. Creación de objetos en las bases de datos de la corporación. Backups. Instalación de BD Administración de las mismas, desarrollo en ASP.NET MVC, administración de la BD Oracle del equipo de trabajo, Desarrollo front-end y backend, Orientar cursos de grado y posgrado en el área de Ingeniería de Sistemas. - Dirigir trabajos de grado, Desarrollo de aplicaciones móviles a la medida, desarrollo web y móvil, Desarrollo de funcionalidades de un modulo del sistema y reportes en Jaspersoft, Coordinar actividades Análisis de requerimientos Desarrollo de software y Líder en desarrollo de software.*

Nota: *Los nombres de las personas encuestadas y la empresa a la que pertenecen no fueron solicitas por confidencialidad de la información.*

Posteriormente, se comenzó el diseño y desarrollo del Framework Web para MongoDB. En este proceso se identificaron los siguientes requerimientos de software:

5.2. Funcionales.

Se definieron los siguientes requerimientos funcionales:

- Almacenar la estructura de los objetos en MongoDB.
- Almacenar la información de los objetos en MongoDB.
- Permitir conectar a bases de datos MongoDB locales o remotas.
- Permitir conectar a bases de datos MongoDB locales o remotas, con restricciones de usuario y clave.
- Para cada objeto creado se podrá configurar: su estructura, permisos (Consultar, Insertar, Actualizar y Eliminar), formulario de ingreso de información y las vistas de los datos.
- El formulario de cada objeto deberá controlar: el tipo de dato, restringir los posibles valores, determinar si un atributo puede ser nulo, el tamaño máximo, el orden de presentación, el mensaje de error y si el atributo identifica al objeto.
- Para cada objeto se podrán crear varias vistas, las cuales estarán condicionadas por el filtro que se agregue.
- Se debe generar una aplicación web con la configuración realizada por el usuario sin que tenga escribir código (Ver ilustración 9).

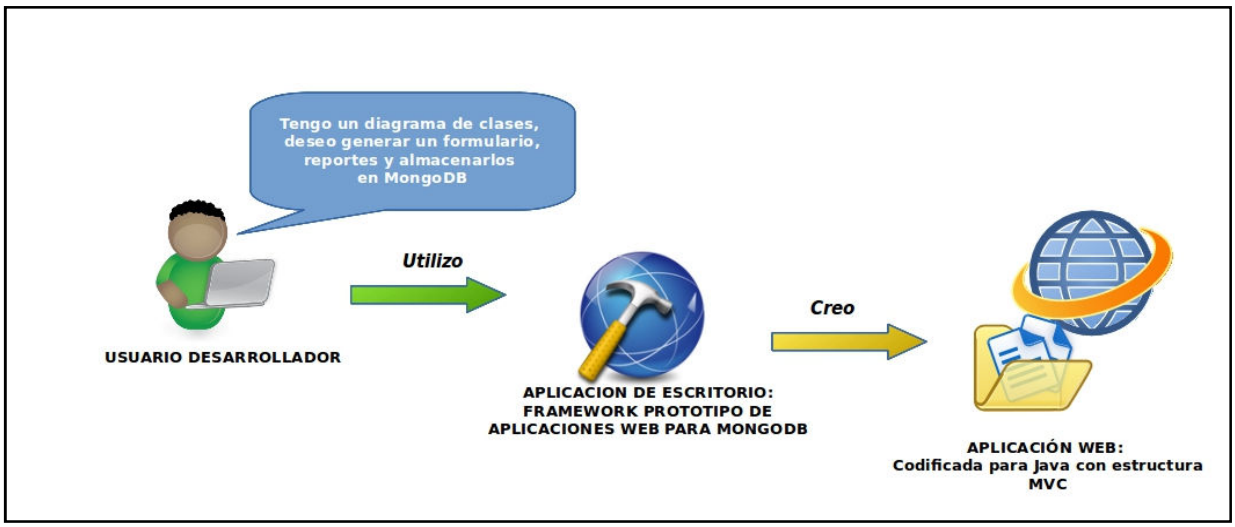


Ilustración 8: Generación de Aplicación web sobre el Framework Prototipo

La aplicación se debe organizar de la siguiente forma:

- Un archivo index.html que mostrara todos los objetos del usuario, su configuración y los enlaces a su formulario de captura de información y las respectivas vistas para acceder a la información.
- Los siguientes directorios:

Configuración. Este directorio almacena el archivo de configuración del proyecto y se utiliza para abrir el proyecto.

Formularios. Este directorio contiene los formularios con extensión .jsp, los cuales permiten el ingreso de la información a nuestros objetos.

Reportes. Este directorio contiene las vistas con extensión .jsp, las cuales nos permite acceder a la información según los filtros asignados.

Paquetes. Este directorio contiene los paquete requeridos para el funcionamiento de la aplicación web. Debe contener los siguientes paquetes: ConectarDB, Controlador, Modelo y ObjetosVarios.

Librerías. Este directorio contiene las librerías requeridas para el funcionamiento de la aplicación web.

Jquery. Este directorio contiene la librería Jquery en su versión 2.1.4 y algunas utilidades que requiere para su funcionamiento.

Las Imágenes. Este directorio contiene imágenes que utiliza la aplicación web.

- La aplicación web resultante debe implementar el patrón de diseño Singleton, con el fin de restringir la creación de objetos que se conecten a la base de datos.
- La aplicación web resultante debe controlar el acceso a los datos a través de la utilización de la Librería Jquery

- La aplicación web resultante debe organizar el código siguiente los lineamientos del Modelo-Vista-controlador (Ver ilustración 10).

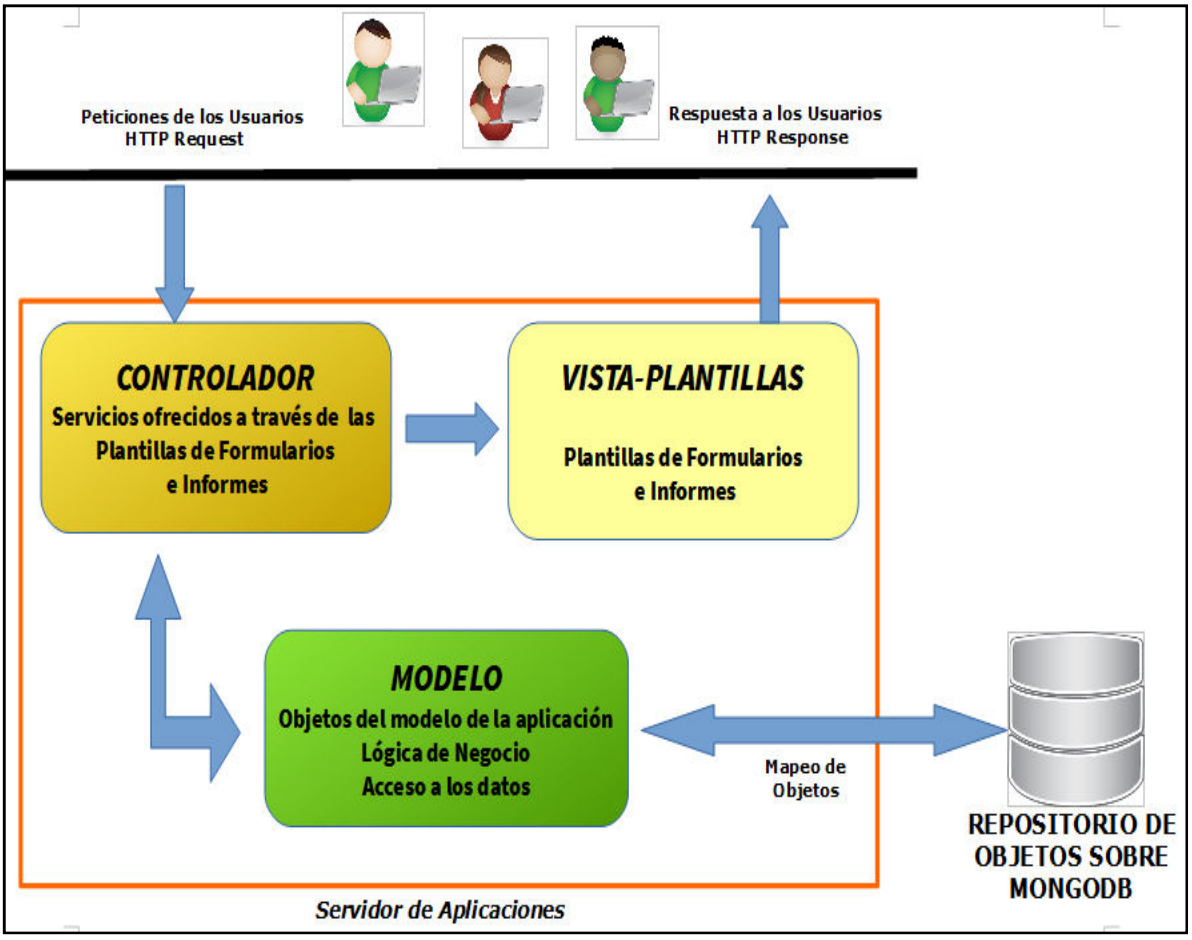


Ilustración 9: Modelo-Vista-controlador para las aplicaciones generadas

5.3. No Funcionales

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

5.4. Funcionamiento del Framework.

Se definió el siguiente modelo como guia para utilizar el Framework propuesto:

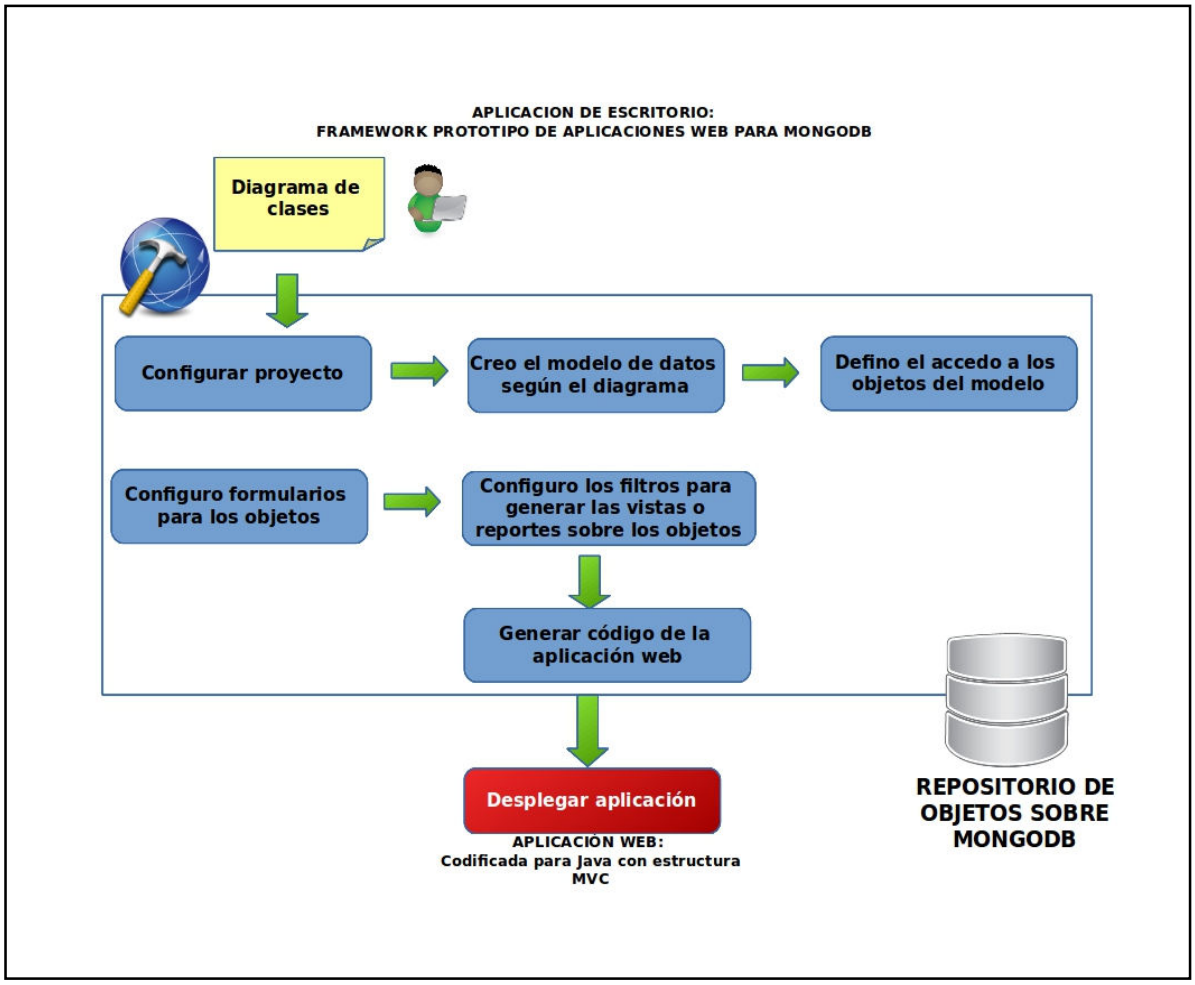


Ilustración 10: Funcionamiento del Framework Prototipo

5.5. Diseño e implementación del Framework

Para el desarrollo del Framework se utilizaron las siguientes herramientas:

- Java Platform, Enterprise Edition (Java EE) SDK en la versión 7.0
- NetBeans 8.1
- Contenedor de aplicaciones web para Java: Apache Tomcat 7
- MongoDB 3.0

- RideMongo
- Librerías:
  - Driver MongoDb para Java: bson-3.1.1, mongodb-driver-3.1.1 y mongodb-driver-core-3.1.1
  - Procesamiento de archivos y/o directorios: commons-io-2.4
  - Jquery: jquery-2.1.4.js, jquery-ui.js y jquery.validate.js

Ilustración 11: Herramientas tecnológicas utilizadas para el desarrollo

5.5.1. Casos de Uso

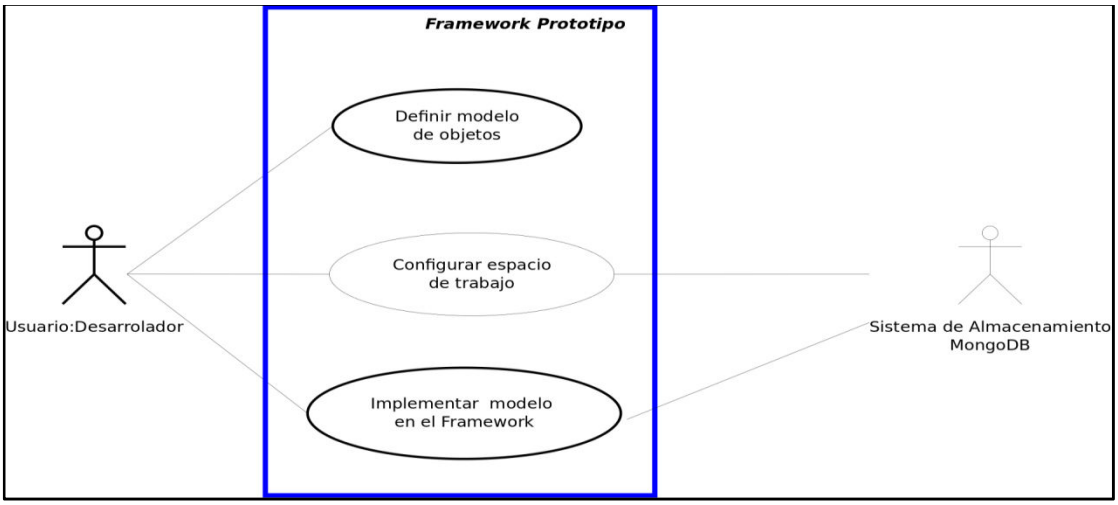


Ilustración 12: Caso de uso para utiliza Framework Prototipo

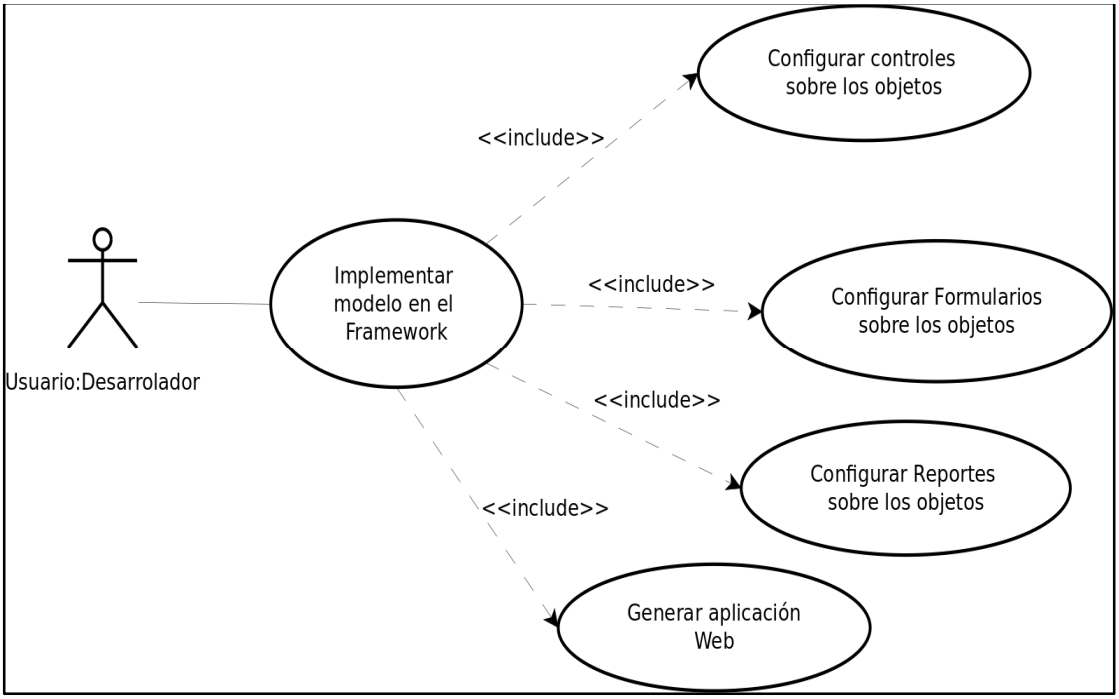


Ilustración 13: Caso de uso para implementar modelo

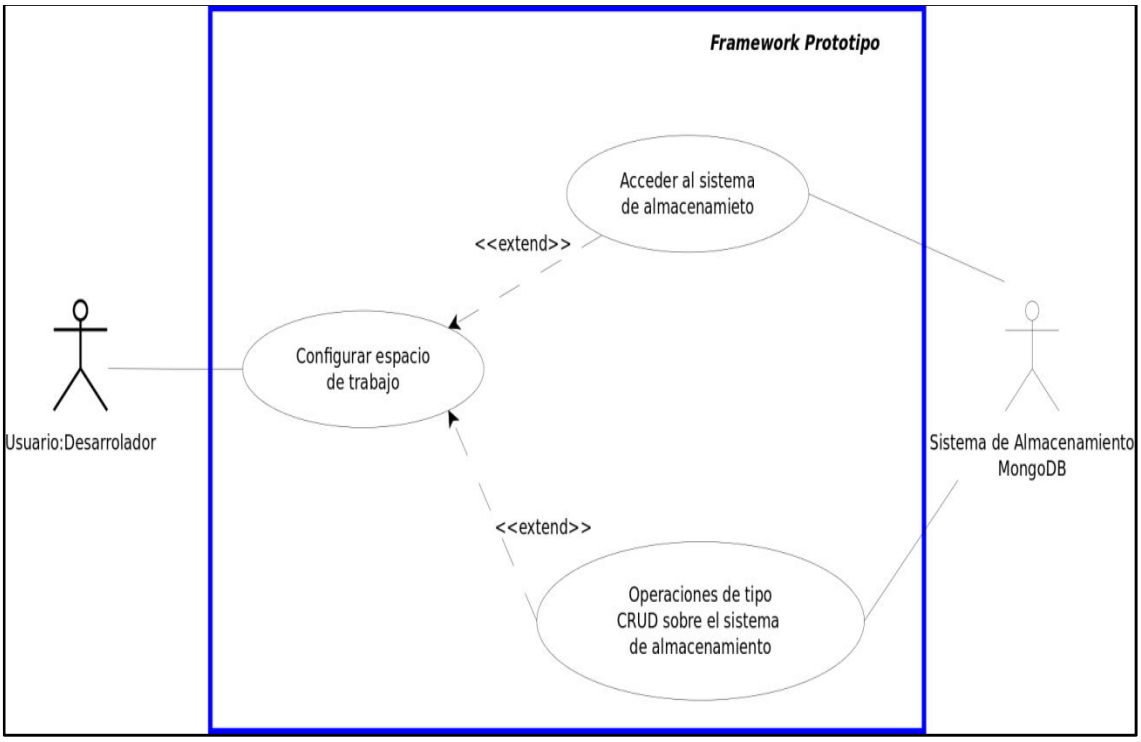


Ilustración 14: Caso de uso para acceder al motor de almacenamiento

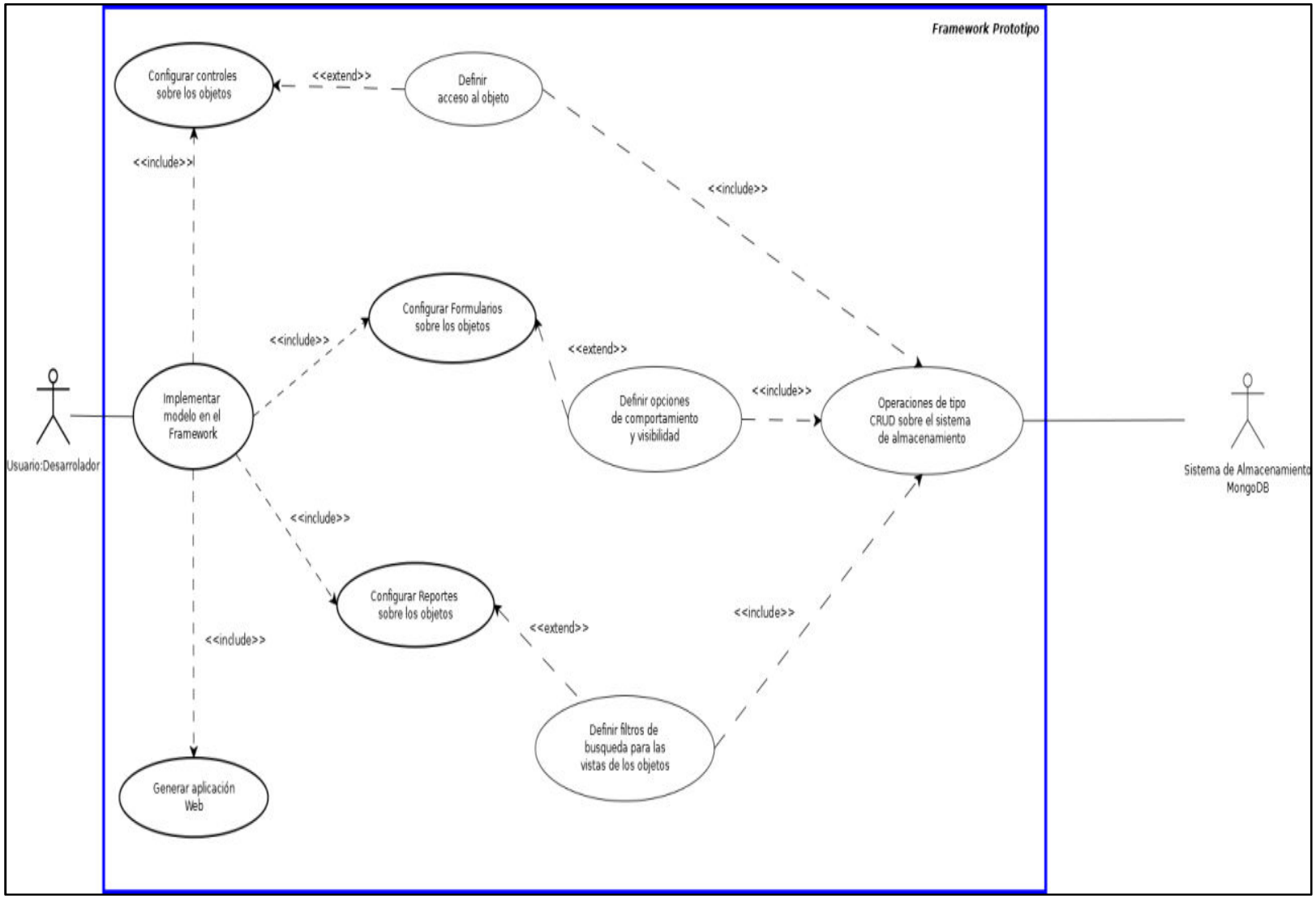


Ilustración 15: Caso de uso detallado de operaciones y acceso a datos

5.5.2. Diagrama de Clases.

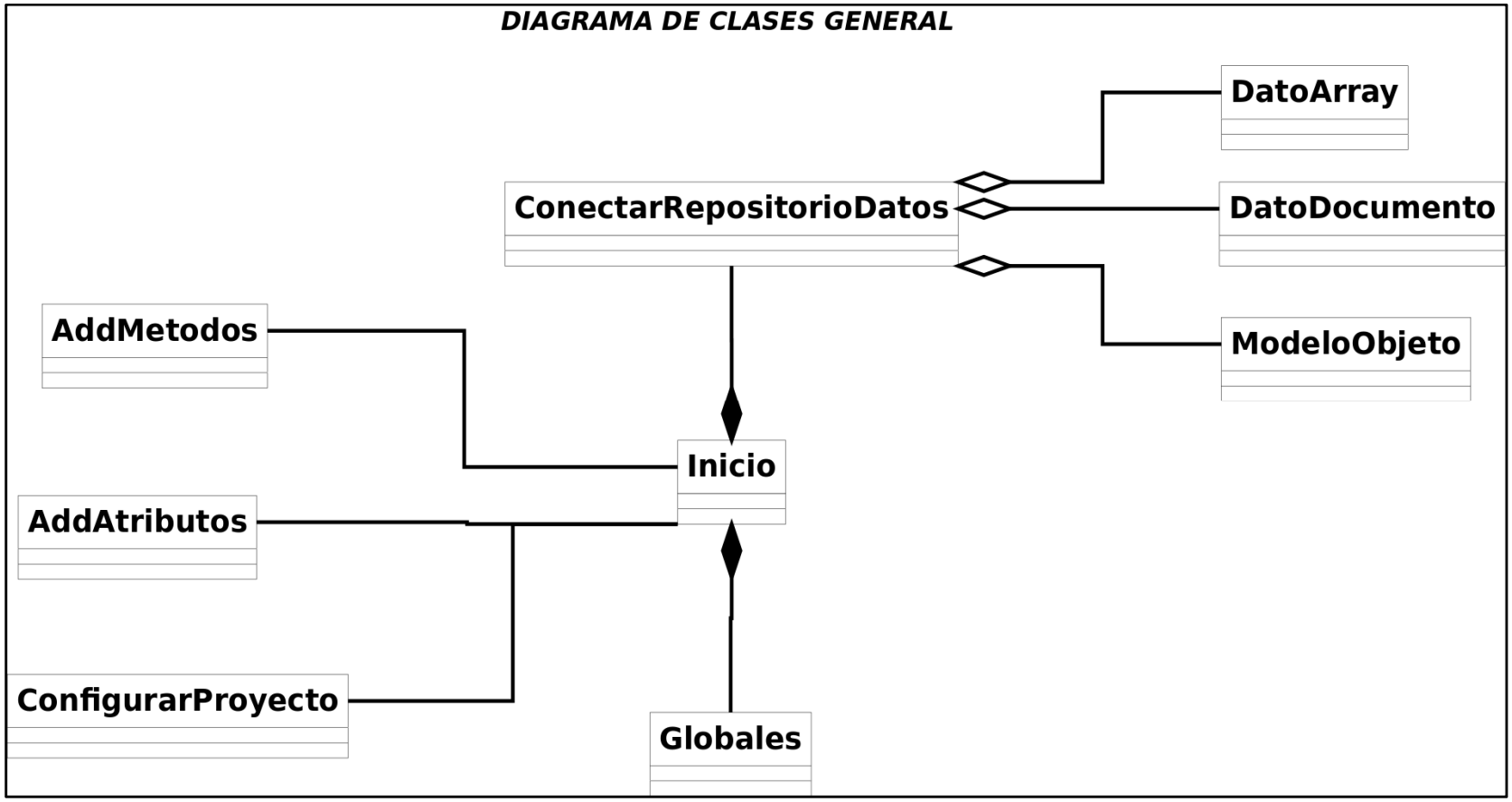


Ilustración 16: Diagrama de clases general para el desarrollo del Framework, los detalles de las clases aparecen en la documentación anexa

5.5.3. Diagrama de Paquetes

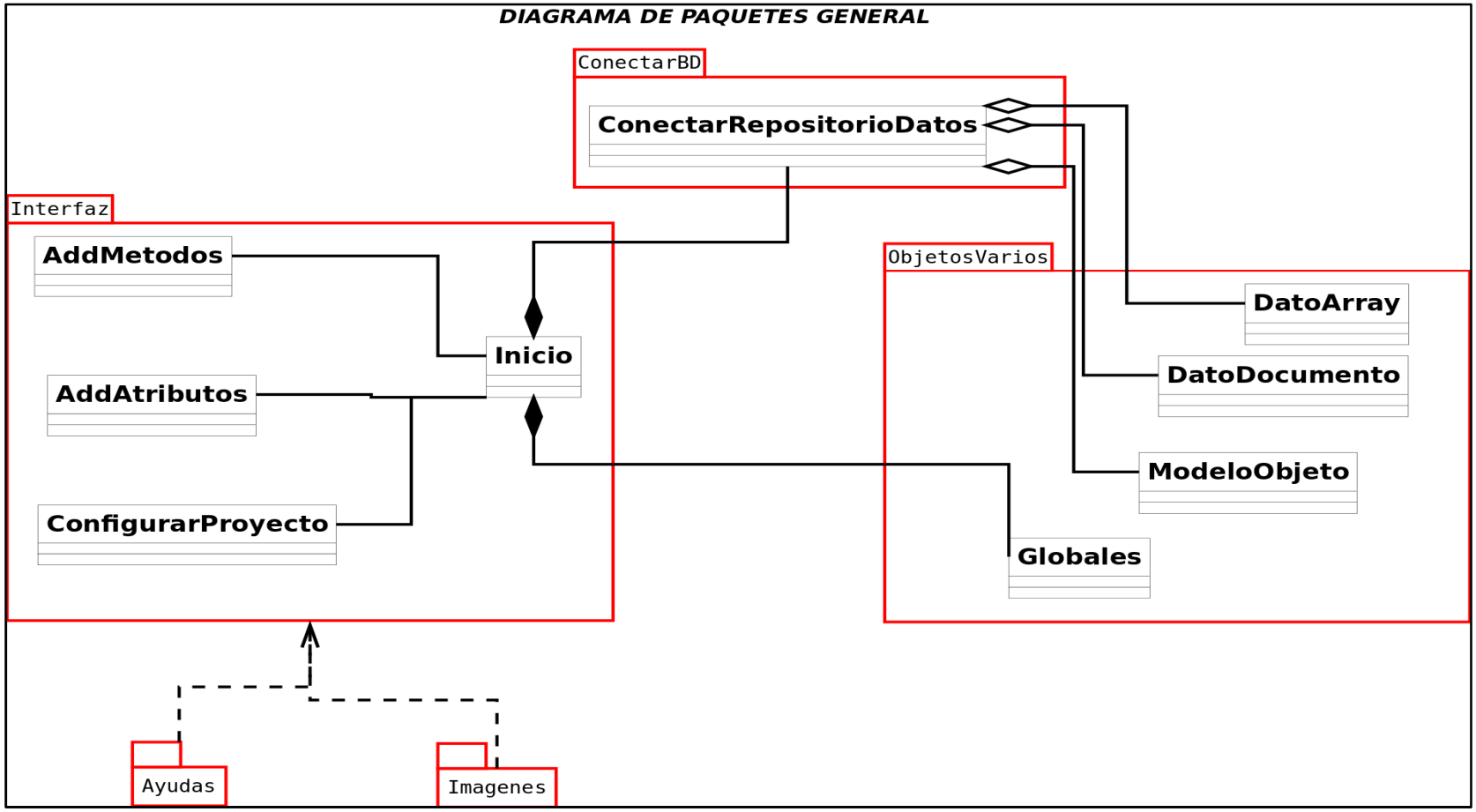


Ilustración 17: Diagrama de paquetes

5.5.4. Diagrama de Componentes.

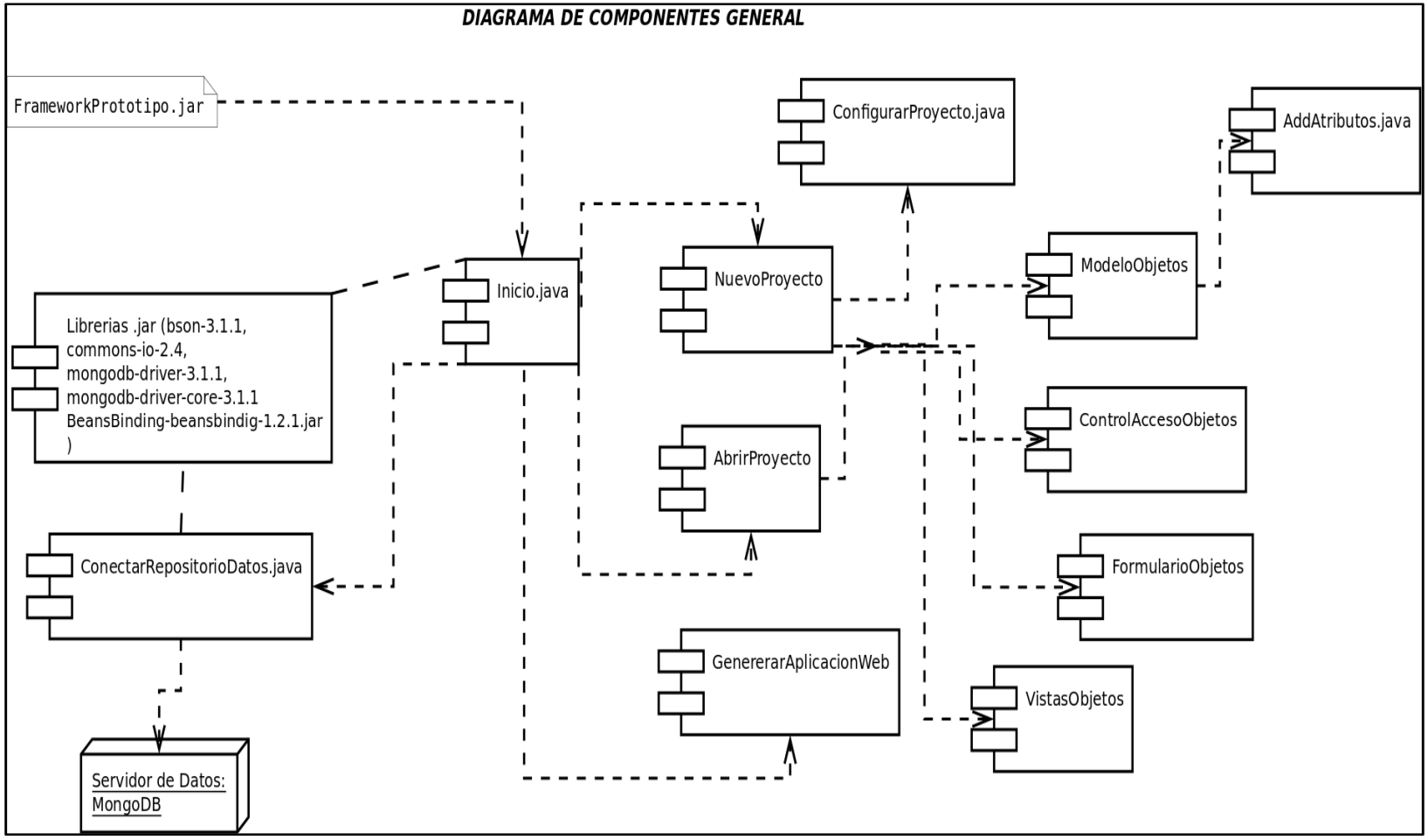


Ilustración 18: Diagrama de componentes

5.5.5. Interfaz grafica de usuario

Se definieron las siguientes interfaces graficas las cuales cubren los requerimientos de captura y visualización de la información del Framework.

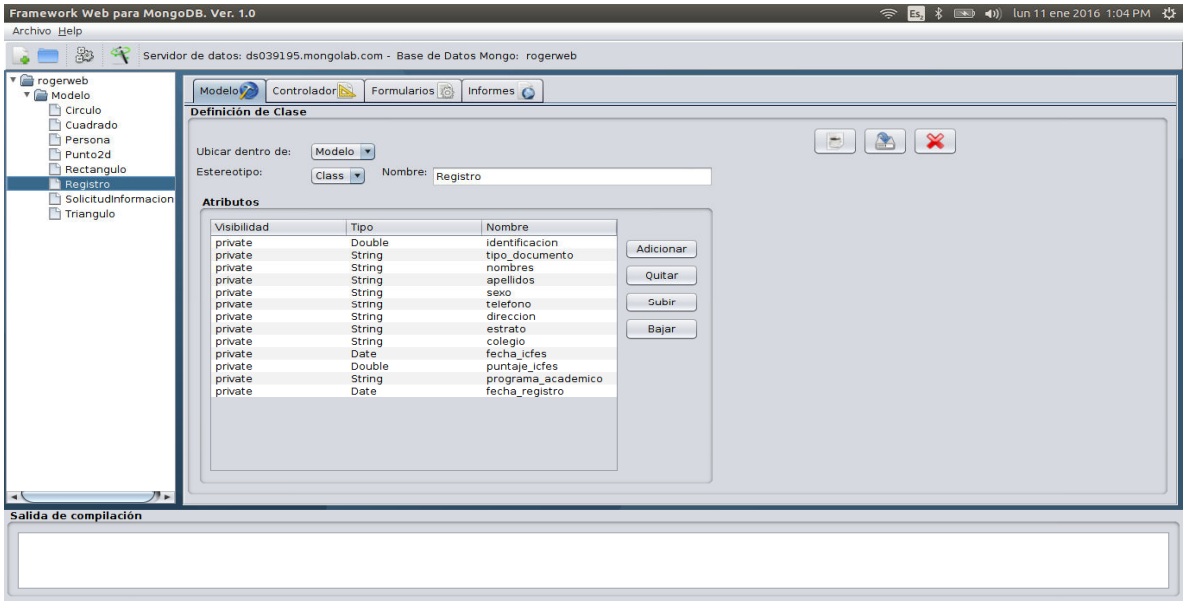


Ilustración 19: Pestaña Modelo para crear los objetos

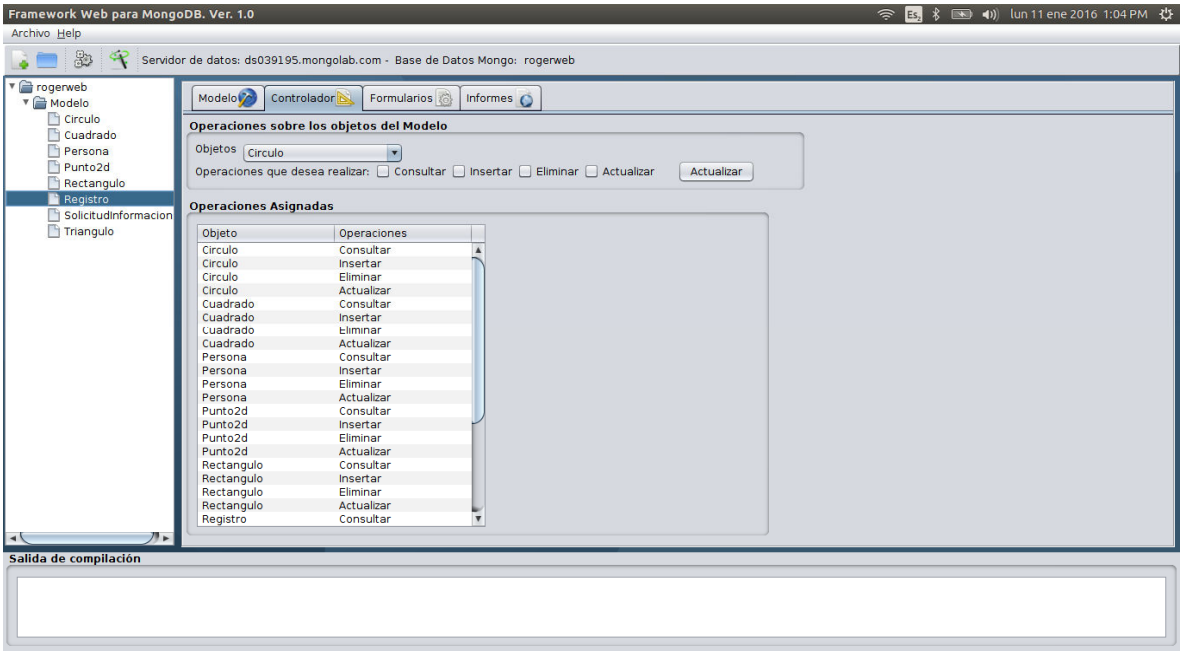


Ilustración 20: Pestaña controlador - Asignación de permisos sobre los objetos

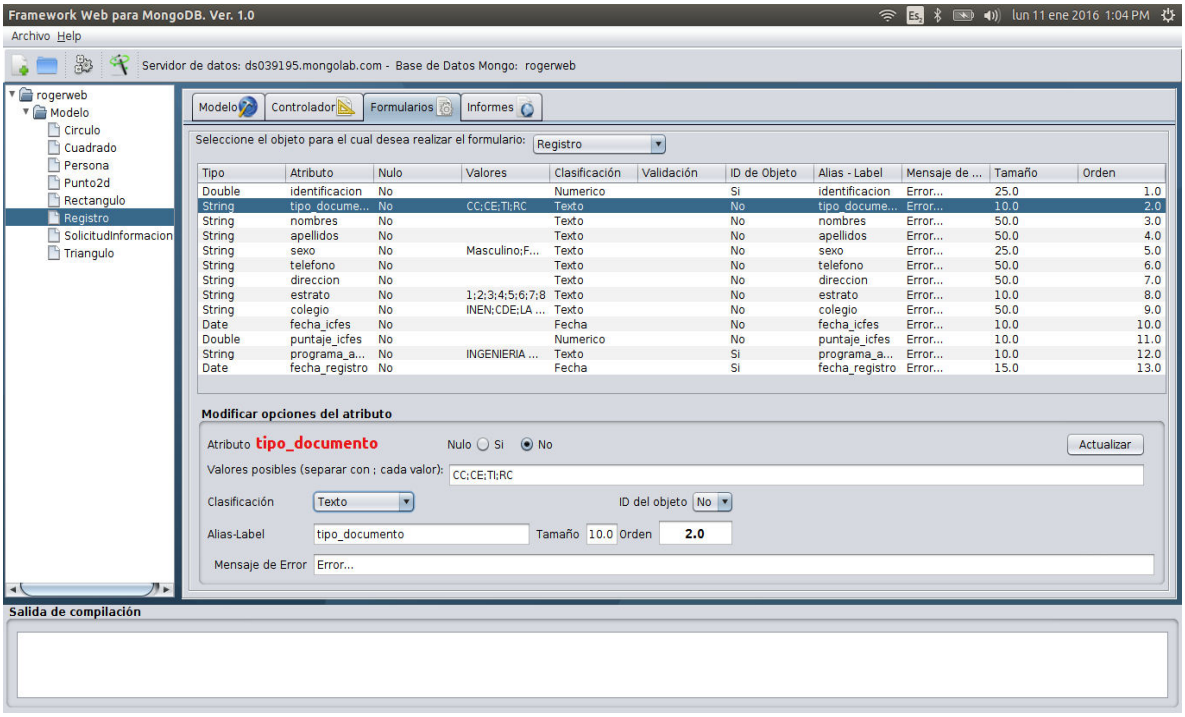


Ilustración 21: Pestaña Formularios - Para modificar la apariencia y comportamiento

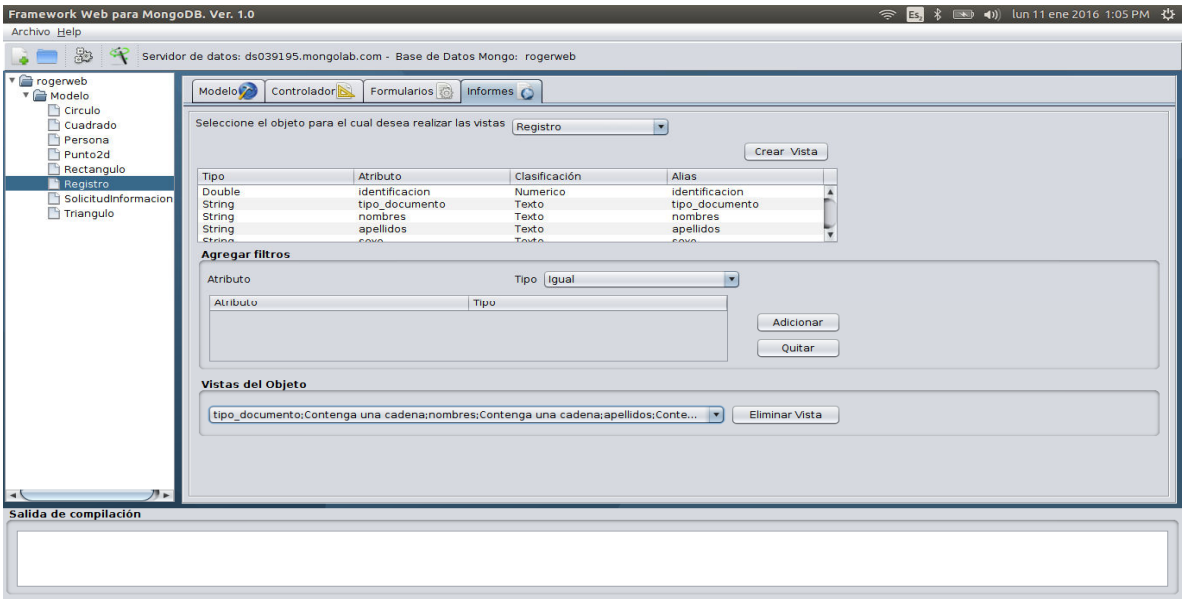


Ilustración 22: Pestaña Informes para definir las vistas que tendra el objeto

Ilustración 23: Configurar el proyecto

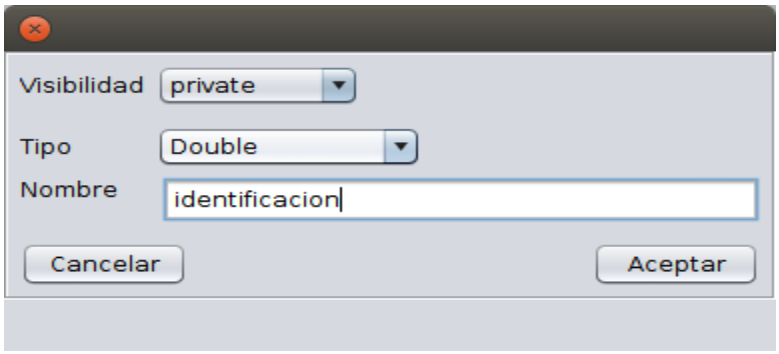


Ilustración 24: Crear atributos

Para la aplicación web resultante la página inicial tendrá la siguiente presentación

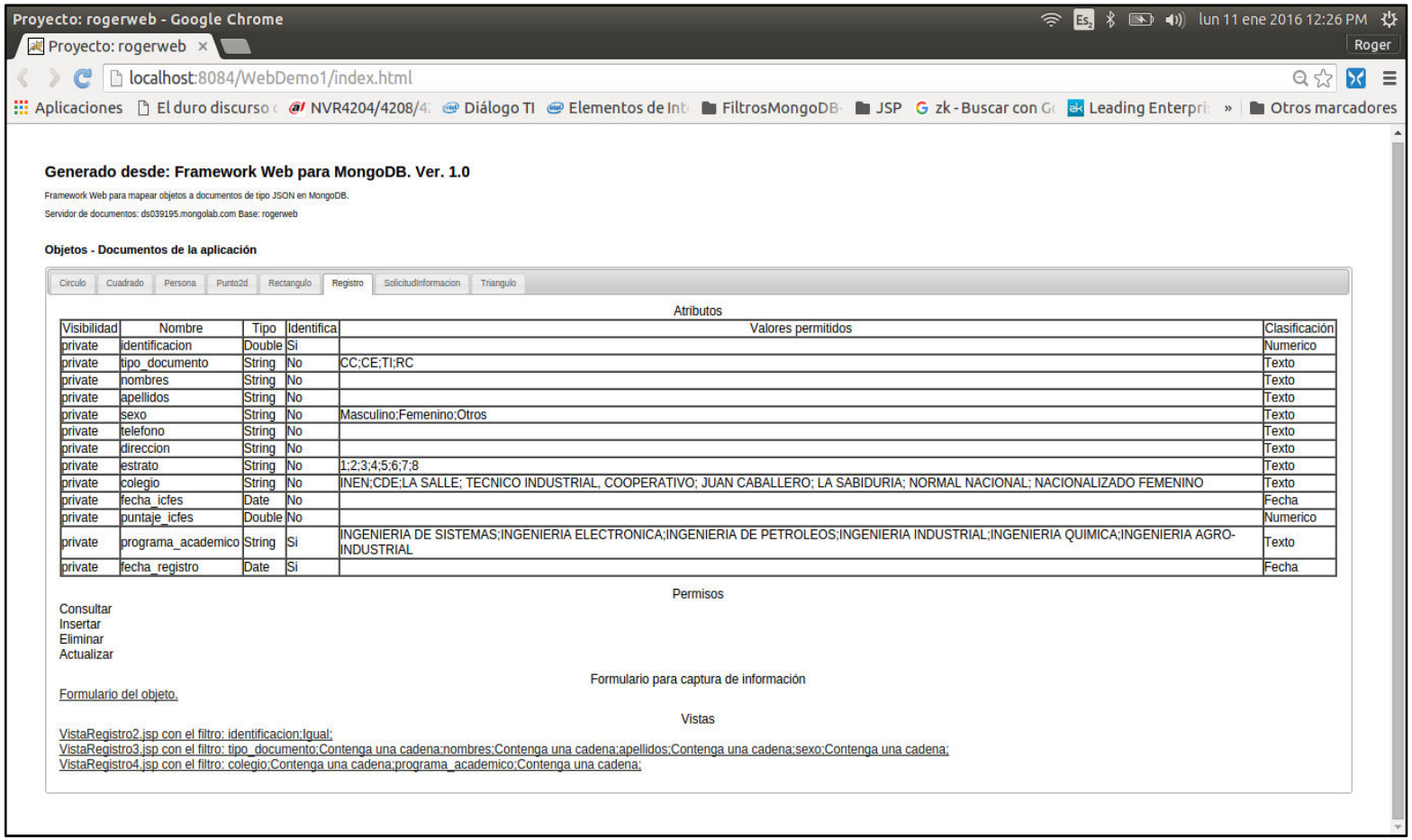


Ilustración 25: Vista de la página principal de la aplicación generada

Para conocer más detalles del funcionamiento del Framework y del proceso de desarrollo puede dirigirse a los manuales de usuario y técnico respectivamente.

5.6. Estructura de la información en el motor de almacenamiento MongoDB

Para cada proyecto creado desde el Framework prototipo se creara una base de daos dentro de MongoDB con el nombre definido por el usuario, la cual estará compuesta por siguientes colecciones: Modelo, EstructuraObjeto y DatosObjetos (ver ilustración 27).

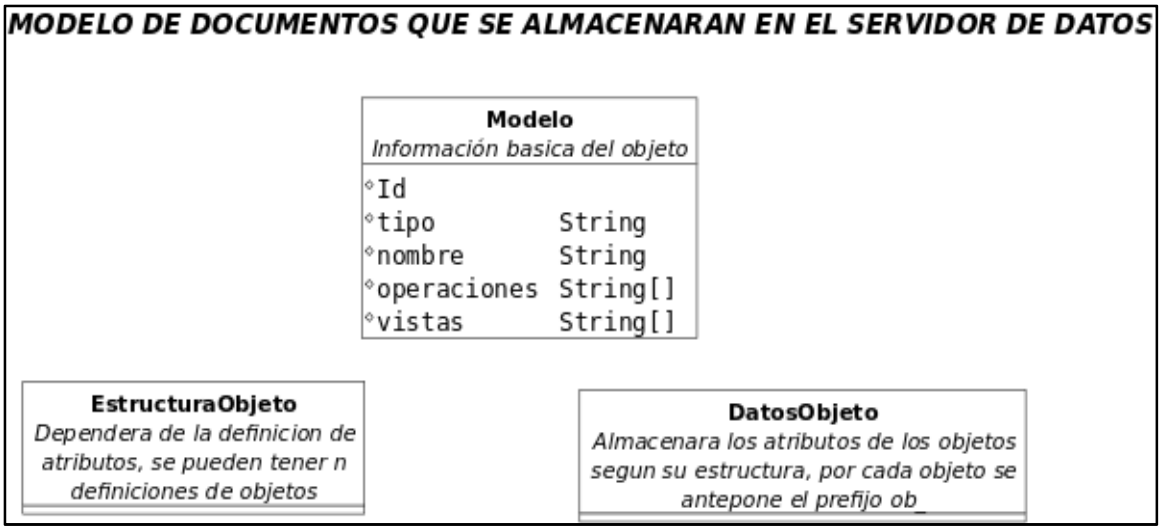


Ilustración 26: Estructura de datos del proyecto para MongoDB

Modelo. Define la información básica para cada objeto definido por el usuario.
EstructuraObjeto. Define la información relacionada con los atributos y métodos (get/set) del objeto definido por el usuario. El nombre de esta colección será el definido por el usuario.
DatosObjetos. Contiene los objetos creados desde la aplicación web, cada objeto tendrá el prefijo ob_. La información de los atributos será almacenada en formato de documentos tipo Json dentro de MongoDB (ver ilustración 28).

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

Ilustración 27: Estructura del objeto vs Estructura en formato Json

A continuación, en la ilustración 29 se muestra través de la visor de documentos Robomongo la estructura de información que se genera desde el Framework.

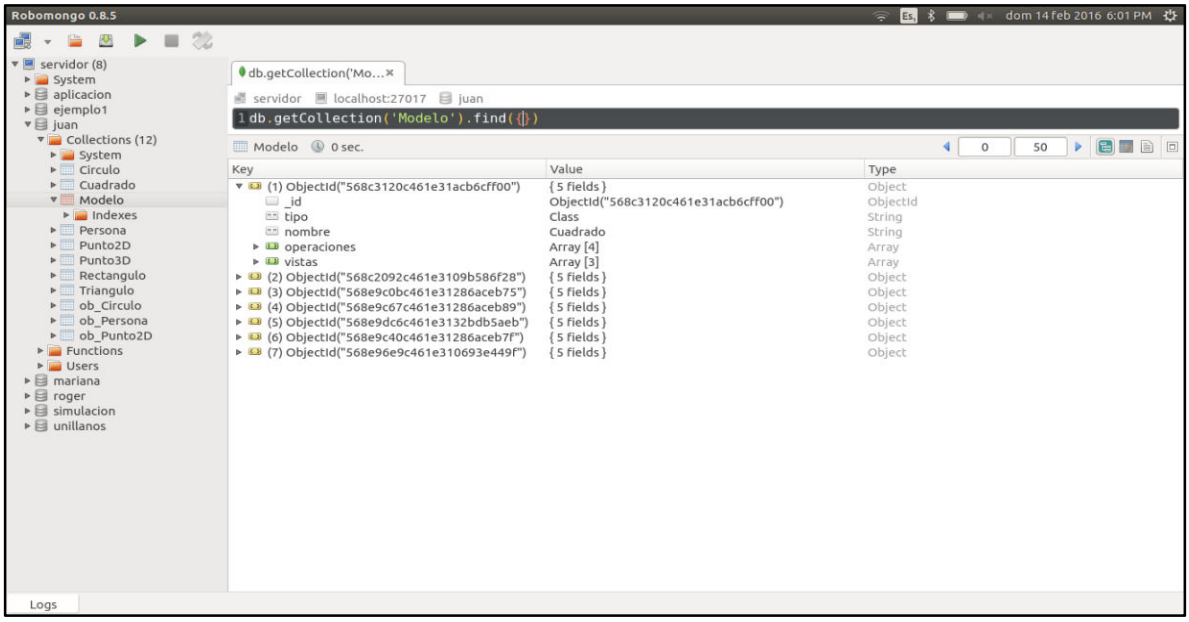


Ilustración 28: Estructura de datos basada en colecciones desde Robomongo

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

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

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

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

6. ANÁLISIS DE LA INFORMACIÓN

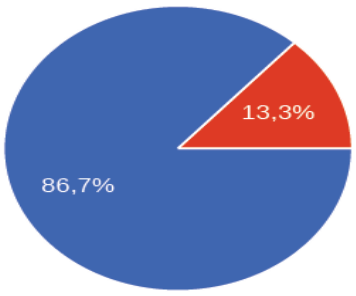
Cuando se termino el Framework Web para MongoDB Ver. 1.0, se procedió a socializarlo entre los miembros de la población de la muestra, a través de exposiciones presenciales y virtuales, en las cuales se mostraban las bondades de esta aplicación. Posteriormente se realizo la segunda encuesta y se procedió a la tabulación de la información y análisis. Con el fin de comprender el trabajo realizado, los resultados se agruparon desde la clasificación dada a las variables:

6.1. Herramientas de tipo mapeo objeto-relacional (ORM)

Dentro de esta clasificación se encuentran las siguientes variables: Interacción con herramientas de tipo mapeo objeto-relacional (ORM), Dificultad en el proceso de aprendizaje de las herramientas tipo ORM, ORM utilizadas y Conocimiento de los problemas de rendimiento de los ORM. Para cada una de ella presentamos los resultados obtenidos de las encuestas:

Interacción con herramientas de tipo mapeo objeto-relacional (ORM)

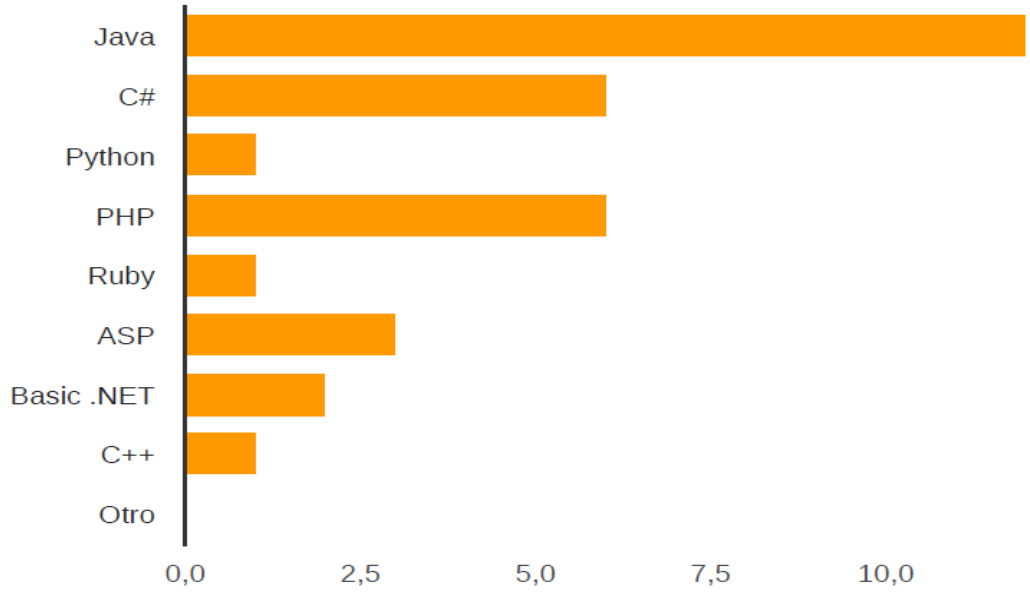
Ha tenido la posibilidad de generar un desarrollo de software utilizando el mapeo objeto relacional.



Sí	13	86,70%
No	2	13,30%

Gráfica 1. Porcentaje de uso de software OMR

En el desarrollo que ha realizado para implementar el mapeo de objetos relacional, que lenguajes de programación ha utilizado:

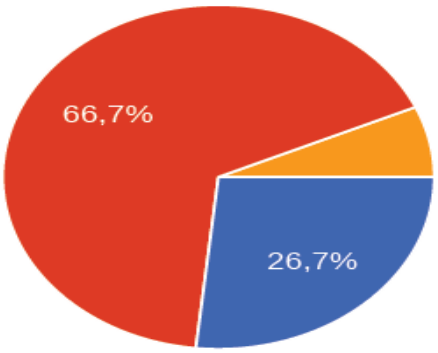


Gráfica 2. Lenguajes utilizados para implementar OMR

Java	12	80%
C#	6	40%
Python	1	6.7%
PHP	6	40%
Ruby	1	6.7%
ASP	3	20%
Basic .NET	2	13.3%
C++	1	6.7%
Otro	0	0%

Dificultad en el proceso de aprendizaje de las herramientas tipo ORM

El proceso de aprendizaje de herramientas para desarrollos con persistencia de datos, le pareció:



Fácil	4	26.7%
Regular	10	66.7%
Difícil	1	6.7%
No llena sus expectativas	0	0%

Gráfica 3. Dificultad en el aprendizaje de las OMR

ORM utilizadas.

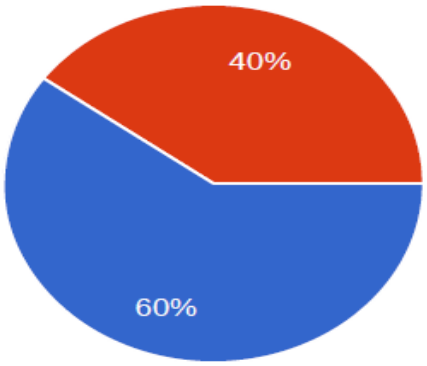
En el proceso que ha realizado para implementar el mapeo de objetos relacional, lo ha construido con:

Gráfica 4. Implementación de ORM

Código propio	5	33.3%
A través de una API (Por ejemplo JPA)	12	80%
Con un framework (Por ejemplo Hibernate)	8	53.3%

Conocimiento de los problemas de rendimiento de los ORM

Sabe usted, que el ocultar al desarrollador lo relacionado con las instrucciones de tipo SQL dentro de un proyecto con mapeo de objetos relacional, genera que las consultas sean más complejas para relaciones sencillas y cargas de innecesaria de datos.



Sí	9	60%
No	6	40%

Gráfica 5. Cocimiento sobre los problemas de rendimiento

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

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

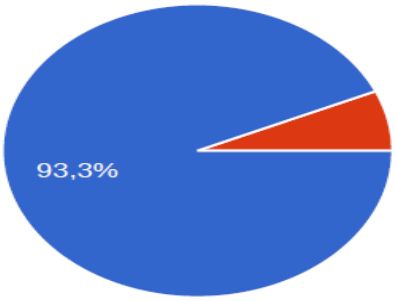
Respecto a los problemas de rendimiento, un 60% conoce de estos problemas frente a un 40% que los desconoce.

6.2. Conceptos de Bases de Datos NoSQL

Dentro de esta clasificación se encuentran las siguientes variables: Conocimiento de las BD NOSQL, Base de Datos NoSQL como medio de almacenamiento de objetos y MongoDB como repositorio de objetos. Para cada una de ella presentamos los resultados obtenidos de las encuestas:

Conocimiento de las BD NOSQL

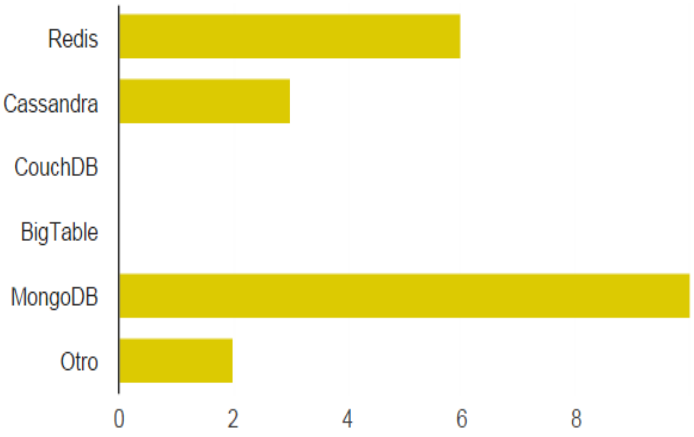
Conoce el concepto de NoSQL o Bases de Datos No Estructuradas.



Sí	14	93.3%
No	1	6.7%

Gráfica 6. Conocimiento sobre NoSQL

Con cuales motores de almacenamiento NSQL ha tenido la posibilidad de tener algún tipo de trabajo académicos o profesional:

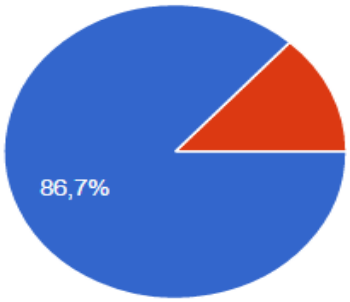


Redis	6	46.2%
Cassandra	3	23.1%
CouchDB	0	0%
BigTable	0	0%
MongoDB	10	76.9%
Otro	2	15.4%

Gráfica 7. Motores de almacenamiento NoSQL

Base de Datos NoSQL como medio de almacenamiento de objetos

Considera que con un motor de almacenamiento NoSQL se pueden almacenar los objetos de la aplicación en la base de datos sin utilizar persistencia de datos, conservando la información del objeto completa en la base de datos:

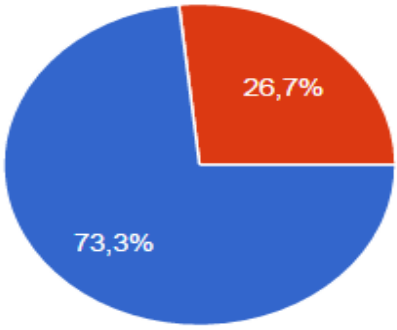


Sí	13	86.7%
No	2	13.3%

Gráfica 8. NoSQL y el almacenamiento de objetos

MongoDB como repositorio de objetos

Respecto al motor de almacenamiento MongoDB, considera que el modelo de almacenamiento basado en documentos, se puede adaptar al modelo de objetos de la aplicación.



Sí	**11**	73.3%
No	**4**	26.7%

Gráfica 9. MongoDB y el almacenamiento de objetos

Dentro de la clasificación de Conceptos de Bases de Datos NoSql, se observa que este tema es conocido por la mayoría de los encuestados en un 93.3%, que de

los motores de almacenamiento NoSql que mas conocen son: MongoDB con un 76.9%, seguido de Redis con 46.2% y Cassandra con un 23.1%.

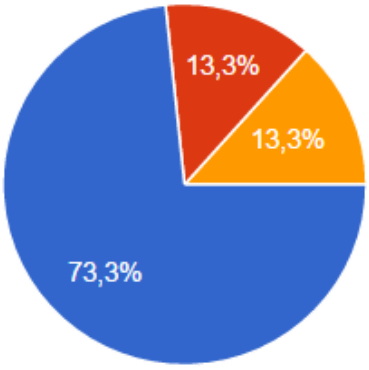
Respecto a la posibilidad de almacenar la información de un objeto respetando su definición en un motor de almacenamiento NoSQL los encuestados en un 86.7% manifiestan que sí es posible realizar esta operación. Con relación a la posibilidad de utilizar la estructura basada en documentos ofrecida por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

6.3. Framework desarrollado

Dentro de esta clasificación se encuentran la variable: Concepto del Framework web para MongoDB. Para cada una de ella presentamos los resultados obtenidos de las encuestas:

Concepto del Framework web para Mongodb

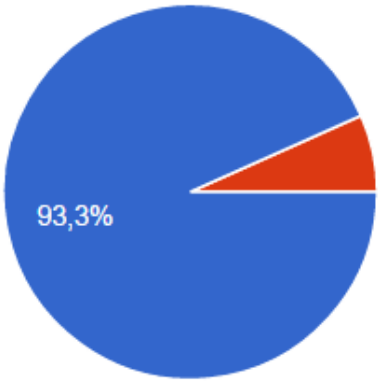
La aplicación que se desarrollo con la nueva herramienta para generar aplicaciones web con almacenamiento de objetos sobre MongoDB, es:



Buena	11	73.3%
Regular	2	13.3%
No llena sus expectativas	2	13.3%

Gráfica 10. Consideraciones de facilidad de uso del Framework

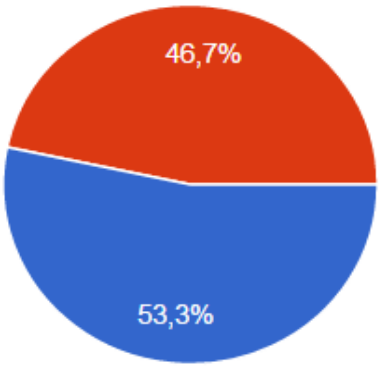
Recomendaría esta herramienta como caso de estudio.



Sí	14	93.3%
No	1	6.7%

Gráfica 11. Recomendaría el Framework desarrollado

Recomendaría esta herramienta para generar aplicaciones web a nivel profesional.



Sí	8	53.3%
No	7	46.7%

Gráfica 12. Recomendación a nivel profesional

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

7. CONCLUSIONES

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

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

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

- La aceptación del Framework Web los para MongoDB fue satisfactorio y la consideran como una aplicación buena en un 73.3%, pues sus características de fácil manejo y no programar una sola línea de código para generar una aplicación web, hacen de este Framework una opción interesante para los

usuarios desarrolladores que están iniciando como para aquellos que ya tienen cierta experiencia.

- La población encuestada evidencia un conocimiento alto (93.3%) sobre las nuevas formas de almacenamiento de información denominadas Bases de Datos NoSql, y se resalta que han utilizado diversas tecnologías como: MongoDB, Redis y Cassandra, para el desarrollo de proyectos. También consideran que es posible almacenar en esta clase de motores no relaciones los datos de los objetos de nuestras aplicaciones respetando en todo momento su definición inicial.

- La aceptación del Framework Web los para MongoDB que genera aplicaciones web fue satisfactorio y la consideran como una aplicación buena en un 73.3%, la cual puede ser utilizada como caso de estudio, pero se resalta que no tiene aprobación para ser utilizada en proyectos a nivel profesional, algo que es comprensible, ya que es la primera versión que se desarrolló, que faltan muchos componentes comparado con las frameworks tradicionales de tipo relacional, a lo que los encuestados están acostumbrados a utilizar.

8. RECOMENDACIONES Y TRABAJOS FUTUROS

Durante el desarrollo del proyecto se resaltan elementos como la indagación sobre el conocimiento que tienen los profesionales en desarrollo de software sobre las nuevas tecnologías para el almacenamiento y organización de la información, tema que puede servir como una guía para futuras investigaciones con muestras poblacionales más grandes.

Otro elemento importante, es el Framework Web para MongoDB, al cual se le pueden realizar mejoras y adicionar mas funcionalidades con el fin de tener un producto más robusto, sostenible y que sea de libre distribución, para lo cual se le debe adicionar una licencia de software de tipo GPL o MIT y así, pueda ser liberado a la comunidad del Software Libre.

Se recomienda que con el material recopilado para el desarrollo del Framework y su respectivo código fuente, se generara un documento o texto el cual pueda ser utilizado por la comunidad de desarrolladores que utilizan las bases de datos basadas en Documentos como lo es MongoDB y su correspondiente interacción con el driver del lenguaje de programación Java.

Respecto al Framework Prototipo, se recomienda:

- Realización de pruebas de carga y desempeño a la aplicación web generada utilizando para ello herramientas como Jmeter.
- Mejorar aspectos de seguridad a la aplicación web generada para mejorar las validaciones a nivel de clientes y de servidor de datos.
- Mejorar el aspecto de interfaz de usuario con el fin de ser más fácil de usar por parte de los usuarios finales.
- Realizar comparaciones de rendimiento con ORM.

9. REFERENCIAS BIBLIOGRAFICAS

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

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

Castillo, J. M. (2013). *Persistencia de objetos. JDO, Solución Java.* Facultad de Informática, Universidad de Murcia.

couchdb.apache.org. (n.d.). *CouchDB.* Retrieved 02 25, 2016, from http://couchdb.apache.org/

cwiki.apache.org. (n.d.). Retrieved 02 25, 2016, from https://cwiki.apache.org/confluence/display/COUCHDB/Introduction

Fernando Alonso Amo, L. A. (2008). *Introducción a la ingeniería del software - Modelos de desarrollo de programas.* Delta Publicaciones.

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

Ghosh, D. (2010). Multiparadigm Data Storage for Enterprise Applications. *Software, IEEE , vol.27, no.5* , 57,60.

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

Kaplan-Moss., A. H. (n.d.). *django-book*. Retrieved from http://django-book.mkaufmann.com.ar/chapter05.html

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

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

MongoDB. (2014, 10). Retrieved from https://www.mongodb.com/press/mongodb-recognized-only-%E2%80%9Cchallenger%E2%80%9D-gartner-2014-magic-quadrant-operational-database

MongoDB. (2015, 06). *MongoDB.* Retrieved from http://docs.mongodb.org/manual/core/introduction/

MongoEngine. (2014). Retrieved from http://mongoengine.org/#home

Objetos, P. e. (2013, 09). *Documentos de Google*. Retrieved from https://docs.google.com/document/d/1nCy-Xk00lBUrBFQvTWk9P5xsw8ee6JOVklSUlRN3mUI/edit

Programación .net. (2005). Retrieved from http://programacion.net/articulo/motores_de_persistencia_231

SOA-agenda. (n.d.). Retrieved from http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/

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

Wikipedia - Mapeo objeto-relacional. (2015, 05). Retrieved from http://es.wikipedia.org/wiki/Mapeo_objeto-relacional

Wikipedia-Hibernate. (2015, 05). *Wikipedia.* Retrieved from https://es.wikipedia.org/wiki/Hibernate

Zhang, X., Song, W., & Liu, L. (2014, Junio). An implementation approach to store GIS spatial data on NoSQL database. *Geoinformatics (GeoInformatics), 2014 22nd International Conference on* .

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