Universidad de las Ciencias Informáticas

FACULTAD 6

Título: "Componente de conexión a MongoDB como sistema de base de datos no relacional para la generación de reportes en el Generador Dinámico de Reportes 2.0".

Trabajo de Diploma para optar el título de Ingeniero en Ciencias Informática

Autores:

Ibis Brito Amaya

Yader Lazaro Bravo Amador

Tutores:

Ing. Miguel Lezcano Ramos

Ing. Yudeily Ledesma Tamayo

La Habana, junio 2015

"Año 57 de la Revolución"

"La única lucha que se pierde
es la que se abandona."

Ernesto Guevara de la Serna

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2015.

Ibis Brito Amaya

_____

Firma del autor

Ing. Miguel Lezcano Ramos

_____

Firma del tutor

Yader Lazaro Bravo Amador

_____

Firma del autor

Ing. Yudeily Ledesma Tamayo

_____

Firma del tutor

Datos de contacto

Autores:

Ibis Brito Amaya

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: ibrito@estudiantes.uci.cu

Yader Lazaro Bravo Amador

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: ylbravo@estudiantes.uci.cu

Tutores:

Ing. Miguel Lezcano Ramos

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: mlezcano@uci.cu

Ing. Yudeily Ledesma Tamayo

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Correo electrónico: yledesma@uci.cu

Agradecimientos

Hoy celebro el fin una etapa importante en mi vida y el comienzo de otra: mi vida como Ingeniera en Ciencias Informáticas. Mi sueño desde la secundaria fue estudiar y graduarme en la UCI, y puedo decir que lo logré con esfuerzo y sacrificio, por eso me siento muy orgullosa de mí misma e indescriptiblemente feliz. Aprovecho la ocasión para agradecer a todas las personas que de una forma u otra han contribuido con mi formación a lo largo de estos años, personas que quedan en mi memoria formando un precioso recuerdo de mi tiempo como estudiante.

Primero, agradezco a mi Amayita linda y amada, a mi Papá Víctor, a mi Hermana Iris, a mis cuatro Abues, a Orlando, a mis Primis y a mis Tías(os) por ser personas especiales en mi vida, por darme todo su amor, por guiarme a ser quién soy, por aconsejarme y acompañarme física y espiritualmente, por ayudarme a mantener el optimismo que me caracteriza y por tantas otras bendiciones que me han otorgado. También agradezco a mis amistades Nayet y Adrianet, a mi compañero Yader, a mi grupo 6504, a mis tutores Ing. Miguel e Ing. Yudeily, a mis Profes de la carrera, a Yoander, a Beatriz, a mi Tribunal y al profe Rosnel; y a todos aquellos que como verdaderos colegas han sabido confiar en mí y fortalecer también de alguna forma mi crecimiento como profesional.

Y especialmente quiero agradecer a mi Cosi que ha sabido ser a la vez mi amigo, mi amante, mi confidente, mi punto de apoyo y empuje en cada situación difícil a la que he tenido que enfrentarme así como la mejor compañía en los buenos momentos durante todo el tiempo que he estado en la Universidad y principalmente en la temporada de tesis.

Por tenerlos a ustedes, estoy agradecida también con Dios.

A todos los quiero muchísimo más de lo que puedo expresar hoy y Gracias nuevamente por acompañarme y ser mi luz en este tiempo tan preciado.

Ibis Brito Amaya

Agradecimientos

No serviría de nada este resultado si no se da una muestra de gratitud a todas aquellas personas que de alguna manera aportaran su granito de arena. Primeramente quisiera agradecer a mis padres que más que padres son mis amigos y los que han hecho posible todos mis logros. Mami sabes que eres mi tesoro más grande, papi gracias por tu ejemplo. A mi segundo padre el Vala como le digo cariñosamente, gracias por ser como eres. A mis abuelitas por sus sabios consejos. A mis tíos por todo el apoyo que siempre me han dado, aquí presentes Isdenys y mi única tía Isneisy. A mis primos, en especial a Ediel por su ayuda incondicional y su disposición en todo momento. A mi compañera de tesis por soportarme y aguantar todas mis malcriadeces y peleas, por su dedicación insuperable y porque sin ti no hubiese sido posible este resultado. A Gustavo por guiarme y apoyarme, gracias mi hermano. A los tutores por estar pendientes en todo el trayecto de este trabajo. A todos los miembros d este tribunal y a mi oponente por sus buenos consejos y señalamientos. A mis amigos Yandrey, Alex y Sergio por todos los momentos q pasamos juntos y por estar ahí siempre q los necesité. A mis amistades del grupo 6504 con los que compartí mi vida como estudiante en la uci, Emilio, Bernardo, el favio. En general, agradezco a aquellos que formaron parte de mi vida durante estos 5 años en la universidad, a todos muchas gracias.

Yader Lazaro Bravo Amador

Dedicatoria

Yo, **Ibis Brito Amaya**, dedico este triunfo con el que tanto he soñado

.. A Irma, esa maravillosa mujer que tengo por Madre,

.. A mi Papá Víctor, a Iris mi Tata amada,

.. A mis Abues, a Orlando y a Gustavo por estar siempre para mí;

.. Ustedes son el verdadero motivo por el cual me convierto hoy en Ingeniera.

Es mi deseo como sencillo gesto de agradecimiento, dedicarle mi humilde obra de Trabajo de Grado plasmada en el presente Informe:

En primera instancia a la memoria de mis abuelos quienes estarían orgullosos de este resultado, a mis padres y mi novia, quienes permanentemente me apoyaron con espíritu alentador, contribuyendo incondicionalmente a lograr las metas y objetivos propuestos.

Yader Lazaro Bravo Amador

Resumen

En el Centro de Tecnologías de Gestión de Datos (DATEC) se encuentra el proyecto Generador Dinámico de Reportes 2.0, que permite generar reportes de forma dinámica partiendo de datos persistentes en los orígenes de datos soportados por el sistema. Cuenta con un conjunto de módulos que brindan las funcionalidades para dar soporte al ciclo de vida de los reportes. El módulo Diseñador de Modelos es el encargado de realizar las conexiones con los sistemas gestores de base de datos relacionales PostgreSQL, MySQL y SQLServer. La presente investigación está enmarcada en el desarrollo de una nueva conexión del GDR con MongoDB como sistema gestor de bases de datos no relacional, sustentado en los beneficios que reportaría contar con este gestor de base de datos a la aplicación y en la gran aceptación que han adquirido a nivel mundial. Para el desarrollo de la presente investigación se utilizaron un conjunto de herramientas y tecnologías que permitieron diseñar, implementar y validar la solución siguiendo las pautas que propone la metodología OpenUP para construir un software. Como producto final se obtuvo un componente de conexión a MongoDB para el GDR.

Palabras clave: orígenes de datos, Sistemas Gestores de Bases de Datos, MongoDB, Generador Dinámico de Reportes

Abstract

In the center of data management technologies (DATEC) is the GDR 2.0 project, which allows to generate reports dynamically based on persistent data on the origins of data supported by the system. It has a set of modules that provide the capabilities to support the life cycle of reports. Designer models module is what allows to make connections with the relational database managers, PostgreSQL, MySQL and SQLServer. This research is framed in the development of a new connection of the GDR with MongoDB as non-relational database system, based on the benefits of these database managers and the great acceptance that have acquired around the world. A set of tools and technologies which allowed design and implement the component classes and followed the guidelines proposed by the OpenUP methodology for building a software were used for the development of this research. As a final product was a component of MongoDB to the DDR connection.

Keywords: data sources, Management Systems databases, MongoDB, Generator Dynamic Reporting

Tabla de contenido

Tabla de contenido

Índice de tablas

Índice de figuras

Introducción

El desarrollo actual ha aumentado considerablemente a partir de la constante evolución y utilización de las Tecnologías de la Información y las Comunicaciones (TICs) a nivel mundial. La informática como parte esencial de estas tecnologías representa la vía más utilizada para garantizar la innovación y el avance en todas las esferas de la sociedad, permitiendo a las instituciones optimizar la forma en que operan. Al utilizarla se logran importantes mejoras en las organizaciones, lo que trae consigo beneficios como la automatización de los procesos operativos, el almacenamiento de grandes volúmenes de datos; al igual que la búsqueda, acceso y manipulación de la información de forma ágil y segura, logrando ventajas competitivas que suministran una plataforma de información necesaria para la toma de decisiones.

Desde este punto de vista la creación de la Universidad de las Ciencias Informáticas (UCI) en el año 2002 fue un paso de avance para la nación cubana, teniendo como misión la formación de profesionales comprometidos con su patria y altamente calificados en la rama de la informática para cumplir con la tarea de producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación (1). Para ello la universidad cuenta con varios centros productivos, entre los que se encuentra el Centro de Tecnologías de Gestión de Datos (DATEC), encargado de crear productos informáticos, desarrollar tecnologías y proveer servicios relacionados con la gestión de datos y el análisis de información (2), contribuyendo así al cumplimiento de la misión de la universidad. Como componente fundamental de este centro, se tiene el proyecto productivo Generador Dinámico de Reportes (GDR), el cual es una aplicación web compuesta por seis módulos que garantizan el ciclo de desarrollo de los reportes que genera en distintos formatos con gran variedad de opciones en su diseño y de forma rápida. Uno de ellos es el diseñador de modelos que gestiona orígenes de datos a través de los sistemas gestores de bases de datos relacionales PostgreSQL, SQL Server y MySQL a partir de consultas que se les realizan para diseñar los modelos que serán utilizados posteriormente en la confección de los reportes.

El GDR constituye un sistema de suma importancia para la gestión de la información de cualquier institución, ya que contribuye al proceso de toma de decisiones. Su capacidad de integración con otros sistemas, lo convierte en componente indispensable de cualquier

sistema de información o software de gestión. Actualmente se utiliza en más de 20 proyectos de la Universidad de las Ciencias Informáticas (UCI), además de estar activo en varias entidades nacionales e internacionales, se tienen contratos con varios clientes de la República Bolivariana de Venezuela, manteniéndose como clientes fijos de la aplicación por los beneficios que la misma le aporta.

En la actualidad existen herramientas para la generación de reportes que brindan soporte a sistemas gestores de bases de datos no relacionales con el fin común de optimizar sus funciones y aumentar su competitividad en el mercado, entre ellos se encuentra iReport. Por ello, el GDR necesita gestionar orígenes de datos mediante algún sistema de base de datos no relacional para incrementar la oportunidad de información de la aplicación y evitar la pérdida de oportunidades de negocio con algunas de las instituciones que lo utilizan debido al gran auge y aceptación que han tenido los sistemas de bases de datos no relacionales a nivel mundial.

A partir de la problemática anteriormente mencionada queda definido el siguiente **problema de investigación***:* ¿Cómo consultar los datos almacenados en bases de datos no relacionales para la generación de reportes en el GDR 2.0?

Definiéndose como **objeto de estudio:** Sistemas gestores de bases de datos no relacionales, enmarcado en el **campo de acción:** Sistemas gestores de bases de datos no relacionales para el Generador Dinámico de Reportes 2.0.

Para solucionar el problema planteado se define como **objetivo general:** Desarrollar un componente de conexión a MongoDB como sistema gestor de base de datos no relacional para la generación de reportes en el GDR 2.0.

Para cumplir este objetivo se desglosan los siguientes **objetivos específicos**:

- Analizar conceptos, herramientas, tecnologías y metodología necesaria para el desarrollo del componente de conexión a MongoDB como sistema gestor de base de datos no relacional para la generación de reportes en el GDR 2.0.
- Realizar el análisis y diseño del componente de conexión a MongoDB como sistema de base de datos no relacional para la generación de reportes en el GDR 2.0.

- Implementar el componente de conexión al sistema gestor de base de datos no relacional MongoDB para la generación de reportes en el GDR 2.0.
- Realizar las pruebas al componente de conexión al sistema de base de datos no relacional MongoDB para la generación de reportes en el GDR 2.0.

Para cumplir con los objetivos específicos propuestos se plantearon las siguientes **tareas de investigación***:*

- Análisis de elementos conceptuales asociados a los sistemas gestores de bases de datos no relacionales para definir el marco teórico de la investigación.
- Análisis de los sistemas de bases de datos no relacionales para seleccionar cuál de estos sistemas va a utilizarse.
- Selección de la metodología y herramientas a utilizar para guiar el desarrollo del componente de conexión a MongoDB como sistema de bases de datos no relacional para el GDR 2.0.
- Estudio basado en los servicios web para su posterior implementación en la solución.
- Identificación de las necesidades funcionales del componente de conexión a MongoDB como sistema de bases de datos no relacional seleccionado para lograr el correcto funcionamiento de la solución.
- Modelado de la propuesta de solución para guiar el proceso de implementación del componente de conexión al sistema de bases de datos no relacional MongoDB para la generación de reportes en el GDR 2.0.
- Implementación de los principios de diseño y funcionamiento para satisfacer las necesidades funcionales y tecnológicas del componente de conexión a MongoDB como sistema de base de datos no relacional seleccionado para incorporarle al GDR 2.0.
- Implementación de un servicio web en java para ejecutar la consulta en formato SQL a partir de consultar un origen de datos MongoDB creado previamente en el sistema.
- Diseño de los casos de pruebas para determinar el correcto funcionamiento de los requisitos definidos para el sistema.
- Realización de las pruebas al componente de conexión al sistema gestor de base de datos no relacional MongoDB para la generación de reportes en el GDR 2.0 para validar su calidad.

Métodos científicos de la investigación

Para dar cumplimiento a las tareas planteadas, para la búsqueda y procesamiento de la información se utilizan los siguientes métodos de investigación. Los **métodos teóricos** utilizados en la investigación fueron:

- **Histórico-lógico:** para investigar las características y funcionalidades de los sistemas gestores de bases de datos no relacionales, profundizando en los sistemas orientados a documentos.
- **Analítico-sintético:** para analizar las principales potencialidades que brindan los sistemas de base de datos no relacionales a los sistemas generadores de reportes que los utilizan.
- **Modelación:** para elaborar los diagramas necesarios con el fin de guiar el proceso de implementación del componente de conexión a realizar.

Los **métodos empíricos** utilizados en la investigación fueron:

- **Análisis de documentos:** para recopilar información referente al funcionamiento del Generador Dinámico de Reportes buscando definir los requerimientos del componente de conexión a desarrollar.
- **Observación**: para determinar la forma en que el Generador Dinámico de Reportes se conecta a los sistemas gestores de bases de datos a los que utiliza y para comprobar el correcto funcionamiento del sistema.

Para lograr el cumplimiento de los objetivos propuestos, el documento de tesis se estructura en:

Capítulo I: Fundamentos teóricos del componente de conexión.

Se desglosa el marco teórico de la investigación analizando los principales conceptos relacionados con los sistemas gestores de bases de datos no relacionales. Se selecciona el sistema de base de datos no relacional a utilizar en el desarrollo del componente de conexión a partir del estudio realizado. Se determina la metodología a seguir, además de fundamentar las tecnologías y herramientas en las que se apoya la solución al problema.

Capítulo II: Análisis de la solución propuesta.

Se desarrollan los artefactos correspondientes a la metodología de desarrollo de software OpenUP. Se realiza el modelo conceptual a partir de la definición previa de los conceptos que serán manejados, al igual que las relaciones existentes entre ellos, obteniéndose los requisitos funcionales y no funcionales. Se identifican, especifican y describen los actores y casos de uso necesarios para el desarrollo del componente de conexión a MongoDB para el Generador Dinámico de Reportes 2.0; además de definirse los patrones de diseño utilizados.

Capítulo III: Implementación y prueba.

Se modela el sistema en términos de componentes, se especifican los tipos, niveles, métodos y casos de pruebas que se realizarán a la solución. Se muestran ejemplos de las implementaciones más relevantes. Finalmente se especifican los casos de pruebas realizados para validar el correcto funcionamiento de la solución, persiguiendo la obtención de un sistema que satisfaga las principales necesidades del cliente

CAPÍTULO 1

Fundamentos teóricos del componente de conexión.

Capítulo I: Fundamentos teóricos del componente de conexión.

Introducción

En este capítulo se realiza un estudio acerca de los sistemas gestores de bases de datos no relacionales, sus clasificaciones, beneficios y principales limitaciones. Se guía la investigación a la selección de uno de estos para utilizarlo en la implementación del componente de conexión para la generación de reportes desde el Generador Dinámico de Reportes 2.0. Además se fundamenta el uso de las herramientas, tecnologías y metodología de desarrollo de software a seguir.

1.1 Sistemas gestores de bases de datos

Un sistema gestor de bases de datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. Actúa como interfaz entre los programas de aplicación y el sistema operativo para propiciar un entorno eficiente a la hora de almacenar y recuperar la información de la base de datos. (3) Se clasifican en relacionales o no relacionales, conocidos estos últimos también como NoSQL.

1.1.1 Sistemas gestores de bases de datos no relacionales

Los sistemas gestores de bases de datos no relacionales son una amplia clase de sistemas de gestión de bases de datos que a diferencia del modelo relacional no usan SQL como principal lenguaje de consultas, los datos almacenados no requieren estructuras fijas como tablas y normalmente no soportan operaciones JOIN. Las bases de datos no relacionales están altamente optimizadas para las operaciones recuperar y agregar, y mayormente son utilizadas en el almacenamiento de registros de información. (4)

Algunos de estos sistemas permiten realizar consultas del tipo *Map-Reduce*, las cuales pueden ejecutarse en todos los nodos a la vez (cada uno operando sobre una porción de los datos) y reunir luego los resultados antes de devolverlos al cliente. (5) La gran mayoría permiten también indicar otros detalles como el número de réplicas en que se hará una operación de escritura para garantizar la disponibilidad por el buen rendimiento que ofrecen. Hoy en día se utilizan principalmente en sistemas para los que el aspecto primordial es la velocidad. (6)

Ventajas de los sistemas gestores de bases de datos no relacionales

- No generan cuellos de botella.
- Ofrecen escalamiento sencillo.
- Pueden manejar enormes cantidades de datos.
- Pueden ejecutarse en *clusters* de máquinas con precios relativamente bajos.
- Responden a las necesidades presentes en un creciente número de empresas en cuanto a la escalabilidad horizontal.

Existen varias aproximaciones para clasificar las bases de datos no relacionales tales como: orientadas a columnas, de clave-valor y orientadas a documentos. (7)

Clasificaciones de los sistemas gestores de bases de datos no relacionales

1. **De clave-valor**

Estos son los sistemas gestores de bases de datos no relacionales más simples en cuanto a su uso (la implementación puede ser muy complicada), ya que simplemente almacenan valores identificados por una clave. Generalmente el valor se almacena como un arreglo de *bytes* (BLOB[1], por sus siglas en inglés). De esta forma el tipo de contenido no es importante para la base de datos, solo la clave y su valor asociado. Su Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) es simple ya que es una variación de tres operaciones: insertar, obtener, eliminar. (8) Este tipo de modelo de almacenamiento de información es utilizado para guardar los datos en sesiones de usuarios, opciones de configuración, video juegos y comercio electrónico entre otros ejemplos.

[1] **BLOB:** (*Binary Large Objects*, objetos binarios grandes) son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica.

Su limitante está en que su estructura no sigue un modelo de datos, todo lo que almacenan es un valor binario. Son extremadamente rápidos pero no permiten consultas complejas más allá de buscar por su clave. No posee un lenguaje de consulta estándar o herramientas de consulta (9), por lo que no garantiza la consistencia en los resultados de las consultas.

A continuación se describen algunos de los principales sistemas de base de datos de clave-valor:

DynamoDB

Es un sistema gestor de base de datos propietario desarrollado por la compañía estadounidense *Amazon.com, Inc.* de comercio electrónico y servicios de Internet. Implementa un sistema de persistencia tipo clave-valor que es un servicio gestionado, por lo que no es necesario disponer de expertos en el trabajo con sistemas gestores de bases de datos no relacionales, sino que los desarrolladores pueden encargarse de las tareas de administración. Es altamente escalable ya que no existen límites en la cantidad de datos que permite almacenar entre varios servidores. (10)

Es compatible con otros modelos de datos de clave-valor y orientados a documentos. Su rendimiento la convierte en una herramienta utilizada para móviles, web, juegos, tecnología publicitaria y muchas otras aplicaciones. (11)

Redis

Fue creado en el Instituto Nacional Estadounidense de Estándares (ANSI, por sus siglas en inglés) e implementado en el lenguaje C bajo los principios de código abierto, por lo tanto es compatible y funciona sin problemas en sistemas Unix, Linux y sus derivados, Solaris, OS/X. No ofrece soporte oficial para plataformas Windows. (12) Implementa el paradigma clave-valor- Es similar a un arreglo en memoria para almacenar datos y esos datos pueden ser cadenas, conjuntos (ordenados y desordenados) y listas. (13) No permite realizar consultas complejas, solo insertar y obtener datos además de las operaciones comunes sobre conjuntos (diferencia, unión, intersección). (6)

2. Orientadas a columnas

Los sistemas gestores de bases de datos orientados a columnas almacenan los datos por columnas en lugar de filas para realizar consultas y agregaciones sobre grandes cantidades

de datos; es decir, funcionan de forma similar a los sistemas gestores de bases de datos relacionales. Su principal diferencia es que realizan este proceso almacenando columnas de datos en lugar de registros, lo que lleva a ganar velocidad en lecturas. (14) Por ello se consideran soluciones aplicables generalmente en aplicaciones con un índice bajo de escrituras pero alto de lecturas.

Su principal desventaja reside en que requieren que cada tabla que utilizan se almacene en archivo o memoria, y para ello necesita datos estructurados y esquemas relacionales, heredando muchos de los inconvenientes que estos poseen, incluyendo el esfuerzo y el tiempo de procesamiento del sistema necesario para mantener las tablas canónicas.

A continuación se describen algunos de los principales sistemas gestores de bases de datos orientados a columnas:

Cassandra

Es un sistema gestor de bases de datos incluido en esta clasificación, aunque en realidad sigue un modelo híbrido entre orientado a columnas y clave-valor. Ofrece tolerancia a fallos, puesto que los datos se replican de forma automática en distintos nodos, o incluso en distintos centros de datos y posee buena disponibilidad; sin embargo es consistente sólo eventualmente. (15) Fue diseñado debido a la verticalidad de soluciones de datos relacionales y a la necesidad de ajustar el coste de la implementación para que las configuraciones de explotación fuesen altamente escalables y relativamente económicas. Estas características hacen de Cassandra un sistema gestor de bases de datos no relacional importante, pues combina lo mejor de Dynamo (consistencia eventual) con lo mejor de Google BigTable (familias de columnas) además de ser gratuita y de libre uso y distribución. (16)

Cassandra comparte con Dynamo el mecanismo de membresía de los nodos mediante la alta disponibilidad que alcanza con la replicación entre nodos, mientras que por otra parte implementa un mecanismo de estimación/detección de fallos mediante acumulación. (17)

HBase

Es un sistema gestor de bases de datos orientado a columnas escrito en Java. Básicamente es un conjunto de datos de código abierto que se utiliza para procesar grandes cantidades de datos, es distribuido puesto que los datos se particionan y fragmentan sobre múltiples

servidores; siendo escalable y permitiendo la configuración de mecanismos de tolerancia a fallos, motivos por los cuales es utilizado por reconocidas redes sociales como Facebook (18), Twitter o Yahoo (19).

Es considerado una arquitectura híbrida que habilita la búsqueda más rápida y la recuperación de los datos. Es importante evaluar en él la actuación de lecturas aleatorias ya que recupera y almacena los datos respectivamente en el sistema de archivo *Hadoop Distributed File System* (HDFS, por sus siglas en inglés). (20)

3. Orientados a documentos

Son aquellos sistemas gestores de bases de datos que gestionan datos semi-estructurados. Este tipo de gestores de bases de datos son en esencia un almacén clave-valor con la excepción de que el valor no se almacena exclusivamente como un campo binario, sino con un formato definido de forma tal que el servidor pueda entenderlo. Esto no significa que siempre sigan el mismo esquema, sino que sólo se tienen dos campos donde uno de ellos es binario y puede ser entendido por la base de datos. Dicho formato puede ser Notación de Objetos de JavaScript (JSON, por sus siglas en inglés), *Binary JSON* (BSON, por sus siglas en inglés), Lenguaje de Etiquetado eXtensible (XML, por sus siglas en inglés) o cualquier otro formato estándar.

Si el servidor entiende los datos, puede realizar operaciones sobre ellos. De hecho varias de sus implementaciones permiten ejecutar consultas muy avanzadas sobre los datos, e incluso establecer relaciones entre ellos sin permitir JOINs. Por lo que ofrecen buen rendimiento y escalabilidad sin perder del todo los beneficios del modelo relacional (14).

Otras características de los sistemas gestores de bases de datos orientados a documentos:

- No siguen un esquema estándar por lo que cada documento por lo general asocia cualquier número de campos de cualquier longitud.
- Utilizan formatos estándares como PDF y Microsoft Office (Word, Excel) para encapsular y codificar la información.
- Son considerados como sistemas altamente disponibles y poseen gran velocidad de acceso a datos.

A continuación se describen algunos de los principales sistemas de base de datos orientados a documentos:

CouchDB

Es un sistema gestor de base de datos de código abierto, accesible mediante una Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) que hace uso extensivo de la Notación de Objetos de JavaScript (JSON, por sus siglas en inglés). Está escrito en un lenguaje llamado *Erlang*.

Algunas de sus principales características son las siguientes:

- Es altamente concurrente, diseñada para ser replicada de forma horizontal, a través de varios dispositivos y tolerante a fallos.
- Utiliza *Map-Reduce* para indexar y consultar la base de datos.
- Guarda datos de forma local en la propia máquina cliente para reducir la latencia, gestionando una replicación a la nube por el usuario.
- Está compuesta por colecciones de documentos a los que se les pueden incluir nuevos campos sin que afecte a otros documentos ya existentes en la base de datos.
- CouchDB no tiene una funcionalidad de autoincremento o secuencia.
- Ofrece una mejor solución a servicios como wikis, blogs y sistemas de gestión documental, que las bases de datos relacionales.

Se basa en JSON para almacenar datos y JavaScript. Una de sus características principales es la facilidad con la que permite hacer replicaciones. Está basado principalmente en solicitudes realizadas mediante el protocolo de transferencia de hipertextos (HTTP, por sus siglas en inglés). Es un motor de base de datos orientado a documentos para entornos web, altamente escalable y con un sistema de replicación bastante fácil y potente, el cual permite la reducción de costes en entornos de producción ya que no se necesitarán máquinas tan potentes para su uso. (21)

Google BigTable

Google BigTable es un mapa distribuido multidimensional que está formado por cadenas ordenadas. Cada cadena en él consta de una fila, las columnas y un valor de marca de tiempo que se utiliza para la indexación.

Las claves de las filas de Google BigTable son cadenas de tamaño arbitrario y cualquier operación sobre ellas es atómica. Las filas, son particionadas para ser separadas entre varios nodos servidores; mientras que las claves de las columnas forman las "familias de columnas", a las que se les puede asignar un valor determinado para cada fila. Finalmente, el tiempo permite mantener varias versiones de un mismo dato que haya ido variando. (22)

MongoDB

MongoDB está desarrollado bajo el concepto de código abierto. Este sistema gestor de bases de datos posibilita el almacenamiento orientado a documentos con esquemas dinámicos, empleando el formato JSON que ofrecen simplicidad y poder a las aplicaciones. Brinda soporte completo de indexado sobre cualquier atributo al igual que la replicación proporcionando una alta disponibilidad. (23) Emplea el modelo de escalamiento horizontal de modo que no compromete la funcionalidad del sistema. Permite realizar una amplia variedad de consultas independientemente de si se trata de búsquedas por campos, rangos o expresiones regulares. Además de ejecutar consultas Map/Reduce y permitir agregaciones flexibles que fortalecen el procesamiento de los datos; así como el almacenamiento GridFS[2], para archivos de cualquier tamaño. (24) El desarrollo de MongoDB comenzó en octubre de 2007 por la compañía de software 10gen. (25)

MongoDB se puede obtener de una forma gratuita bajo la Licencia Pública General de Affero de GNU (AGPL, por sus siglas en inglés) y los drivers para los lenguajes de programación están bajo la licencia de Apache.

Se trata de un servicio similar a CouchDB, que pretende combinar lo mejor de las bases clave-valor con los sistemas de gestión documental. Hace uso de JSON y tiene su propio

[2] **GridFS**: Sistema de almacenamiento de MongoDB que permite dividir archivos. Está incluido en los drivers de MongoDB y disponible para los lenguajes de programación que soporta.

lenguaje de consultas. Está implementado en C++ y es usada en aplicaciones como *SourceForge*, *Foursquare* o *GitHub*.

Algunas de sus principales características son las siguientes:

- Cualquier campo en un documento de MongoDB puede ser indexado, al igual que es posible hacer índices secundarios. El concepto de índices en MongoDB es similar a los encontrados en base de datos relacionales.
- Se puede escalar de forma horizontal usando el concepto de *shard*.
- MongoDB tiene la capacidad de ejecutarse en múltiple servidores, balanceando la carga y/o duplicando los datos para mantener el sistema funcionando en caso de que exista un fallo de *hardware*.
- MongoDB tiene drivers oficiales para varios lenguajes de programación, tales como: C, C++, Lisp, C# / .NET, Haskell, node.JS, Erlang, Java, Python, JavaScript, PHP, Perl, Ruby.
- Puede ser utilizado con un sistema de archivos, tomando la ventaja de la capacidad que tiene MongoDB para el balanceo de carga y la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos.

Comparación de los sistemas de bases de datos no relacionales

Con la intención de tomar una decisión sobre cuál sería el sistema gestor de base de datos más indicado para incluirle al GDR 2.0 en correspondencia a su filosofía y funcionamiento, se realiza un estudio sobre los diferentes tipos de gestores existentes, obteniéndose como resultado:

Clave Valor (limitaciones):

- No permiten realmente un modelo de datos, todo lo que guardan es un valor binario.
- No permiten consultas complejas más allá de buscar por su clave.
- No existe un lenguaje de consulta estándar o herramientas de consulta (9) por lo que no garantiza la consistencia en los resultados de las consultas.

Orientadas a columnas (limitaciones):

- Al ser tan similares a los sistemas relacionales, heredan muchos de sus inconvenientes, incluyendo el esfuerzo y el tiempo de procesamiento del sistema necesario para mantener las tablas canónicas; necesitando que la tabla que utiliza se almacene en archivo o memoria, y para ello requiere de datos estructurados y esquemas relacionales.

Orientadas a documentos (potencialidades):

- Son una opción en caso de tener que realizar consultas complejas optando por no perder la velocidad de los almacenes clave-valor.
- Son horizontalmente escalables, lo que significa que a medida de que su base de datos crece, pueden agregárseles más hardware.
- Es considerado como una aplicación altamente disponible y posee gran velocidad de acceso a datos.
- Las bases de datos orientadas a documentos pueden asociar cualquier número de campos de cualquier longitud en un documento.

A partir de este estudio acerca de las diferentes clasificaciones de los SGBD no relacionales se detectaron las limitaciones descritas anteriormente en el caso de los almacenes clave-valor y los sistemas orientados a columnas, que pudiesen atentar contra el rendimiento y eficiencia del proyecto en desarrollo; y por tal motivo se desecharon como posibles candidatas a tener en cuenta a la hora de seleccionar el sistema gestor de base de datos no relacional a incorporar al GDR. En cambio, los SGBD orientados a documentos por sus propias características pudieran brindar grandes beneficios en cuanto a rendimiento y escalabilidad; por tal motivo la investigación dirige su atención a los gestores de bases de datos documentales.

Selección del sistema gestor de base de datos orientado a documentos a utilizar

A continuación se muestra un gráfico que representa las tendencias de búsqueda en Google para las entradas MongoDB y CouchDB a partir del año 2005 hasta la actualidad.

Fig. 1. Tendencias de búsqueda en Google para MongoDB y CouchDB

.

Los números del gráfico reflejan la cantidad de búsquedas que se han realizado acerca de ambos términos en relación con el número de búsquedas totales realizadas en Google a lo largo del tiempo. Una tendencia descendente no significa que haya descendido el número de búsquedas absoluto o total de ese término sino que ha disminuido su popularidad, como es el caso de CouchDB en la Figura 1. En cambio MongoDB presenta una tendencia ascendente.

DB-Engines clasifica mensualmente en un ranking los sistemas de gestión de base de datos de acuerdo a su popularidad. Al realizar el estudio se ubicaba en la 5ta posición de 277 y en la 1ra de las de su tipo seguida por Cassandra en la 8va posición. (26) Dado el auge que ha tomado este SGBD han surgido alternativas de desarrollo y cada vez son más los sistemas que lo requieren; como es el caso de MEAN.io [3]y MeteorJS[4].

Luego de analizar los resultados arrojados por la investigación se concluye que el sistema de gestión documental MongoDB es el sistema de bases de datos no relacional indicado

[3] **Stack MEAN:** es una solución JavaScript utilizado para construir aplicaciones web rápidas, robustas y mantenibles usando MongoDB, Express, AngularJS y Node.js.

[4] **MeteorJS:** es una nueva plataforma con la que cualquier desarrollador va a poder escribir sus propias aplicaciones web.

para incorporarle al Generador Dinámico de Reportes 2.0 con el fin de consultar orígenes de datos MongoDB a través de él y finalmente diseñar reportes con la información seleccionada de la fuente de datos de este tipo; debido a que admite la replicación de datos utilizando múltiples servidores para el almacenamiento de archivos, es escalable, permite realizar consultas complejas a las bases de datos, es de código abierto, posee una comunidad de desarrollo muy activa dada su popularidad, además de que la herramienta iReport 5.2 soporta MongoDB como único sistema de base de datos no relacional, elemento que lo hace compatible con las tecnologías utilizadas en el Generador Dinámico de Reportes actualmente.

1.2 Sistemas Generadores de Reportes

Los generadores de reportes son herramientas complementarias de los sistemas de información incluidos en la mayoría de los productos de software empresariales. Proveen una forma transparente al usuario para realizar consultas a las bases de datos y obtener información de ella en forma de reporte. Tienen la capacidad de interactuar con los resultados obtenidos basándose en los datos para tomar sus propias decisiones. (27) Cuentan con una serie de características importantes para construir reportes permitiendo exportarlos. En la actualidad la forma de generarlos se restringe a las facilidades que presten las herramientas propias de las bases de datos o herramientas creadas con este fin como es el caso de iReport. (28)

1.2.1 iReport

El iReport es un potente generador de reportes visuales que utiliza Jasper Reports. (29) Está escrito en Java y posee una interfaz gráfica intuitiva y fácil de usar. Esta herramienta permite a los usuarios realizar diseños sofisticados que contienen gráficos, imágenes, subinformes, tablas de contingencia, entre otros elementos que lo fortalecen como generador de reportes. (30) Posibilita acceder a los datos a través de JDBC, *TableModels*, JavaBeans, XML, Hibernate, CSV, y fuentes personalizadas para publicar los reportes en PDF, RTF, XML, XLS, CSV, HTML, XHTML, texto, DOCX, u OpenOffice. (29)

1.2.2 Generador Dinámico de Reportes 2.0

El GDR 2.0 es una aplicación web que provee una forma transparente al usuario para realizar consultas a bases de datos y obtener información de ellas en forma de reporte. Se

encuentra desarrollado sobre el framework Symfony 2 para facilitar la gestión de la información (31). Presenta un área de trabajo adecuada para el diseño de los reportes. Soporta varios orígenes de datos relacionales, proporcionando la generación de reportes en los formatos PDF, HTML, XLS, CSV, XLSX, PPTX, ODS, ODT, RTF, TXT, XML, XHTML y DOCX de forma rápida y segura. Para la protección y auditoría de los recursos de la aplicación le han sido asegurados los principios de seguridad y el trabajo con las listas de control de acceso. La aplicación incluye potenciales avances en sus interfaces ya que emplea el novedoso framework Lycan en su diseño basado en ExtJS (30), brindando una serie de comportamientos y apariencia que favorecen la interacción con el sistema.

Arquitectura del GDR 2.0

La arquitectura del software ha pasado a ser un elemento esencial dentro del proceso de desarrollo de cualquier sistema informático, debido a sus múltiples usos. Construir una arquitectura para el software que sea apropiada para dar cabida a las exigencias de las distintas partes interesadas y buena en términos absolutos, no es una tarea sencilla. El GDR está diseñado con una arquitectura modular y distribuida que ayuda a obtener tanto escalabilidad como flexibilidad. Los procesos se distribuyen entre varios componentes que se pueden ampliar e integrar con soluciones personalizadas. Una típica aplicación para la generación de informes atraviesa por las tres etapas del ciclo de vida de los reportes: creación, administración y entrega; el GDR ofrece las herramientas necesarias para llevar a cabo estos procesos.

Para soportar el ciclo de creación de los reportes se implementaron seis módulos. El ciclo se inicia con el diseñador de modelos, que es donde se realiza la conexión a uno de los gestores de base de datos soportados por el sistema, para porteriormente diseñar los modelos semánticos que contienen la información en forma de metadatos y los que serán utilizados en la confección de los reportes.

En el diseñador de consultas se encuentra el área donde se diseñan las consultas; para ello la aplicación cuenta con dos vistas básicas, la vista de diseño y la vista SQL; en la vista de diseño se elabora la consulta de forma gráfica mostrando las tablas, sus campos y las relaciones entre ellas; mientras que en la vista SQL se muestra el código SQL perteneciente a la consulta diseñada.

Con el fin de obtener información de los sistemas de gestión a través de utilizar plantillas como vía más rápida para diseñar y gestionar los reportes, se define la estructura interna de cada uno posibilitando entre otras funciones, su creación, exportación e importación en el diseñador de reportes al igual que la visualización de los ya existentes.

El administrador de reportes se encarga de administrar las suscripciones y reportes por categorías, logrando la organización y actualización del estado de cada uno mediante el envío de correos electrónicos dado un tiempo establecido por el usuario.

El visor de reportes permite realizar varias funcionalidades como la visualización y exportación del reporte a diferentes formatos. El Generador Dinámico de Reportes 2.0 implementa el proceso de entrega en el diseñador de reportes y en el visor de reportes, pues ambos módulos posibilitan la visualización y exportación del informe a diferentes formatos.

1.3 Servicio web

La necesidad de estandarizar la comunicación entre diversas plataformas y lenguajes de programación provoca el surgimiento de los servicios web. Estos tienen múltiples definiciones, lo que demuestra su complejidad para brindar una adecuada definición que englobe todo lo que son e implican.

El *World Wide Web Consortium*[5](W3C) define que: *"Los servicios web son un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la web."* (32) Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer una serie de servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web. A su vez proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario; el caso común de uso se refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar SOAP. (33)

[5] **W3C:** es una comunidad internacional formada por un grupo de organizaciones que trabajan conjuntamente para desarrollar estándares Web.

Los servicios web permiten que aplicaciones ubicadas en lugares distintos geográficamente puedan utilizar distintas tecnologías para comunicarse e integrarse. De esta forma se pueden desarrollar aplicaciones que hagan uso de otras aplicaciones que estén disponibles en Internet interactuando con ellas; apoyando el desacoplamiento en un sistema distribuido al minimizar las dependencias entre servicios. Algunas de las ventajas que traen consigo los servicios web y que los convierten en muy utilizados son:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Fomentan los estándares y protocolos basados en texto haciendo más fácil el acceso y la comprensión de su contenido.
- Al apoyarse en HTTP, los servicios web pueden aprovecharse de los sistemas de seguridad *firewall* sin necesidad de cambiar las reglas de filtrado.

1.3.1 Estándares para servicios web

Un punto clave en los servicios web es la interoperabilidad donde las distintas aplicaciones, en lenguajes de programación diferentes puedan utilizar estos para intercambiar datos. Debido a que se asientan sobre protocolos y estándares abiertos ya existentes y muy difundidos (HTTP, XML, SOAP, WSDL).

Los principales estándares para el desarrollo de servicios web son los siguientes:

- SOAP (Simple Object Access Protocol)
- WSDL (Web Services Description Language)
- UDDI (Universal Description Discovery and Integration)

SOAP es un protocolo de mensajería XML que forma la base de los servicios web. Proporciona un mecanismo simple y consistente que permite a una aplicación enviar mensajes XML a otra aplicación. Un mensaje SOAP es una transmisión desde un emisor a un receptor, y cualquier aplicación puede participar en este intercambio. (34) Estos mensajes se pueden combinar para soportar varios comportamientos de comunicación, incluyendo solicitud/respuesta, respuesta solicitada, mensajería asíncrona de una vía, o incluso notificación.

Es un protocolo de alto nivel que sólo define la estructura del mensaje y unas pocas reglas para su procesamiento. Es completamente independiente del protocolo de transporte subyacente, por eso los mensajes SOAP se pueden intercambiar sobre HTTP o protocolos de transporte de correo. Dentro del paradigma orientado a objetos, usar un servicio web es igual que usar cualquier otra clase. Y esto significa instanciarlo, y llamar a sus métodos, pasándoles los parámetros que sean necesarios, y obteniendo a su vez el resultado que se retorne. SOAP define precisamente cómo se debe codificar las llamadas a los métodos de un servicio web, y cómo debe el servicio web codificar el resultado para que se pueda interpretar. Estos mensajes son los que transportarán los protocolos de transporte, por lo general, HTTP.

Ventajas de SOAP:

- Aprovecha los estándares existentes en la industria.
- Permite la interoperabilidad entre múltiples entornos.
- No está asociado a ningún lenguaje.
- No se encuentra fuertemente asociado a ningún protocolo de transporte.
- No está atado a ninguna infraestructura de objeto distribuido.
- Tanto los datos como las funciones se describen en XML, lo que permite que el protocolo no sólo sea más fácil de utilizar sino que también sea muy sólido.
- El Lenguaje de Descripción de Servicios de Web (WSDL, por sus siglas en inglés) contiene y describe el conjunto de normas comunes para definir los mensajes, los enlaces, las operaciones y la ubicación del servicio Web. WSDL es un tipo de contrato formal para definir la interfaz que ofrece el servicio Web.
- Es más seguro debido a que su implementación siempre o la mayoría de las veces se hace del lado del servidor.

1.4 Marco de trabajo

Un marco de trabajo (*framework*) en el desarrollo de software, es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un

proyecto. Un marco de trabajo representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. (35) Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

El Generador Dinámico de Reportes 2.0 utiliza Symfony2.0 y ExtJS 3.4 como marcos de trabajo, por ello son los definidos para la solución a implementar.

1.4.1 Symfony 2.0

Symfony 2.0 es un marco de trabajo de código abierto, rápido, flexible y fácil de aprender que permite a los desarrolladores construir aplicaciones web mantenibles. Ha sido desarrollado teniendo en cuenta el rendimiento como mayor prioridad. Se construye a base de *bundles* que permiten configurar y personalizar el sistema de una forma limpia. (36). Está construido utilizando un contenedor de inyección de dependencias. Todos los detalles de implementación están ocultos detrás de un buen sistema de configuración que permite personalizar todo a través de archivos .yml o .xml o a través de código PHP. (37) Incluye la librería Doctrine que proporciona herramientas para simplificar el acceso y manejo de la información de la base de datos. (38)

1.4.2 ExtJS 3.4

ExtJS es un marco de trabajo escrito en JavaScript con la finalidad de asistir el desarrollo de Aplicaciones Enriquecidas para Internet (RIA, por sus siglas en inglés). Tiene dos tipos de licencias, Licencia Pública General de GNU (GPL, por sus siglas en inglés) y comercial. Se basa en componentes soportados por recursos para la programación orientada a objetos en JavaScript que facilitan la implementación de extensiones y aplicaciones complejas. (30)

1.4.3 ORM Doctrine

Doctrine es un potente y completo sistema de Mapeo Objeto-Relacional (ORM, por sus siglas en inglés) para PHP 5.2 con una Capa de Abstracción de Bases de Datos (DBAL, por sus siglas en inglés) incorporada. Brinda la posibilidad de exportar una base de datos a las clases PHP correspondientes y también realizar el proceso inverso. Doctrine es un ORM para PHP 5.2.3 y posterior. Cuenta con su Lenguaje de Consultas de Doctrine (DQL, por sus siglas en inglés) siendo esta una de sus principales ventajas.

1.4.4 Lycan

Lycan es una librería de ExtJS desarrollada por DATEC. La misma surge por la necesidad de diseñar componentes de ExtJS de manera intuitiva, rápida y cómoda, promoviendo la usabilidad y elevando la productividad de los desarrolladores. Esta librería fue pensada para contribuir también con otros proyectos externos al centro que también trabajaran con ExtJS en la UCI colaborando así con el éxito de los mismos. Además implementa buenas prácticas de arquitectura y diseño contribuyendo a la calidad del desarrollo de las aplicaciones web y se ha optimizado para su ejecución en Mozilla Firefox. (31)

1.5 Ambiente de desarrollo

En el universo del desarrollo de aplicaciones web se utilizan disímiles tecnologías para la construcción de aplicaciones informáticas. La metodología de desarrollo de software a seguir, las herramientas, los lenguajes de programación y el marco de trabajo a utilizar para el desarrollo del componente de conexión a MongoDB como sistema de base de datos no relacional fueron previamente definidas por el equipo de arquitectura del GDR 2.0. Con el fin de brindar nuevas oportunidades y potenciar al máximo las prestaciones de las mismas se realiza el siguiente estudio.

1.5.1 Metodología de desarrollo OpenUP

Roger S. Pressman en “Ingeniería del Software. Un Enfoque práctico” caracteriza el proceso de desarrollo de software como un marco de trabajo de las tareas que se requieren para construir software de alta calidad. (39) En un proyecto de desarrollo de software, la metodología es el proceso que define Quién debe hacer, Qué, Cuándo y Cómo debe hacerlo; guiando a los desarrolladores en la realización de las actividades. (40)

La metodología que se utilizará para desarrollar el componente de conexión a MongoDB para el Generador Dinámico de Reportes 2.0 es OpenUP por decisión del proyecto. La misma se define como un proceso de desarrollo de software unificado dentro de un ciclo de vida estructurado basado en Proceso Racional Unificado (RUP, por sus siglas en inglés). Es dirigida por casos de uso, centrada en la arquitectura, iterativa e incremental.

1.5.2 Lenguaje de modelado UML 2.0

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es el lenguaje estándar de propósito general más utilizado para especificar y documentar cualquier sistema de

forma precisa. Proporciona una gran flexibilidad y expresividad a la hora de modelar sistemas. UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. (41)

1.5.3 Herramienta de modelado Visual Paradigm 6.1

Visual Paradigm soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El UML facilita la construcción de aplicaciones de calidad con un costo relativamente pequeño. Permite diseñar diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es fácil de usar, soporta ingeniería inversa y exportación/importación XML con el objetivo de permitir un intercambio de metainformación. (42)

1.5.4 Lenguajes de programación

Java 6.1

Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. De modo que al ejecutar el código en una plataforma no tiene que ser recompilado para correr en otra. A partir del 2012 es uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones cliente-servidor. (43)

JavaScript 1.8.5

JavaScript es un lenguaje interpretado que se utiliza principalmente en su forma del lado del cliente. Fue implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. (44) Es un lenguaje basado en objetos y es además orientado a eventos. Esto implica que gran parte de la programación en JavaScript se centra en describir objetos y escribir funciones que respondan a movimientos del ratón, pulsación de teclas, apertura y cerrado de ventanas o carga de una página, entre otros eventos. (45)

PHP 5.3.10

Es un lenguaje script de código abierto que posee un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene para el desarrollo de páginas web

dinámicas del lado del servidor, cuyos fragmentos de código se intercalan fácilmente en páginas HTML; por ello es el más extendido en la Web. (46)

PHP incluye muchas ventajas entre las que se encuentran las siguientes:

- **Integración de base de datos**: PHP dispone de una conexión a diferentes tipos de servidores de bases de datos tanto SQL como NoSQL; tales como MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird, SQLite, MongoDB (47), Hyperwave, Informix, InterBase y Sybase, entre otras. (46)

- **Bibliotecas incorporadas**: como se ha diseñado para su uso en la Web, PHP incorpora una gran cantidad de funciones integradas para realizar útiles tareas relacionadas con la Web.

- **Portabilidad**: PHP está disponible para una gran cantidad de sistemas operativos diferentes, por ejemplo: Linux, FreeBSD, IRIX, y *Windows*.

1.5.5 Lenguajes de marcado

Un lenguaje de marcado es un conjunto de reglas que establecen qué tipo de marcas han de ser utilizadas, de qué modo se distinguirán las marcas del texto del documento, cómo se insertarán estas y cuáles son las marcas permitidas en cada una de las partes del texto. (48)

JSON

Notación de Objetos de JavaScript (JSON, por sus siglas en inglés) es un formato para el intercambios de datos ligero. Básicamente describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Surgió como alternativa a XML y su fácil utilización en JavaScript ha hecho aumentar su uso actualmente. Una de las mayores ventajas que brinda el uso de JSON es que puede ser leído por cualquier lenguaje de programación; además es fácil de escribir y de fácil de analizar. Su procesamiento por parte de los ordenadores es rápido pues se necesitan librerías muy pequeñas para trabajar con él y dada su naturaleza es ideal para entornos AJAX. Es usado para el intercambio de información entre distintas tecnologías. Es nativo de las base de datos MongoDB. (49)

XML

El Lenguaje de Marcado eXtensible (XML, por sus siglas en inglés) es un metalenguaje; es decir un lenguaje para definir un lenguaje de marcado. Los documentos escritos en XML pueden leerse por medio de aplicaciones personalizadas utilizando diferentes objetos de análisis gramatical o pueden combinarse con el Lenguaje de Estilo eXtensible (XLS, por sus siglas en inglés) para lograr mostrarse en un navegador. (50)

1.5.6 Herramienta de desarrollo NetBeans 8.0

NetBeans IDE es un entorno de desarrollo escrito en Java. Es un editor de código multilenguaje con sugerencias de código, acceso a clases ejecutándose en el código, control de versiones, localización de ubicación de la clase actual, herramientas de refactorización, comprobaciones sintácticas y semánticas. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto libre y gratuito sin restricciones de uso. (51)

1.6 Conclusiones parciales

En este capítulo se analizaron diferentes tipos de sistemas gestores de bases de datos no relacionales realizando una comparación entre ellos para justificar la selección del que se va a incorporar al GDR 2.0. Mediante un estudio se concluyó que debe emplearse un sistema gestor de base de datos orientado a documentos y entre ellos se seleccionó a MongoDB. Posteriormente se definieron las herramientas, tecnologías, marcos de trabajo y metodología a utilizar para el desarrollo del componente de conexión en correspondencia con las utilizadas en el desarrollo del GDR 2.0.

CAPÍTULO 2

Análisis de la solución propuesta.

Capítulo II: Análisis de la solución propuesta.

Introducción

En el presente capítulo se desarrollan los artefactos del flujo de trabajo análisis y diseño, correspondiente a la metodología de trabajo OpenUP. Se realizan los diagramas de clases del diseño, mediante los cuales se muestra la estructura del sistema y se describen las clases más relevantes. Se desarrollan los diagramas de interacción y se describen los requisitos funcionales y no funcionales del componente de conexión a MongoDB para el GDR 2.0 al igual que los patrones de diseño usados en la solución del mismo.

2.1 Modelo de dominio

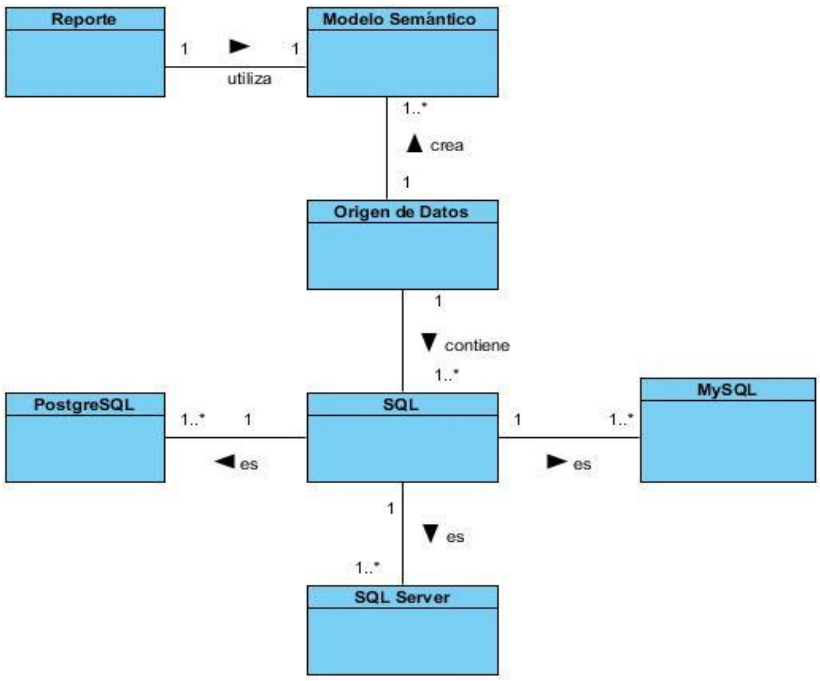


Fig. 2. Modelo conceptual del sistema

El Modelo de Dominio es una representación visual del entorno real de los objetos del proyecto o de las clases conceptuales que se centra en una parte del negocio, la relacionada con el ámbito del proyecto. Es un diagrama con los objetos reales relacionados con el sistema que se va a desarrollar y las relaciones que existen entre ellos (52). Este modelo se crea para documentar el vocabulario del sistema, que ayuda a comprender los conceptos que utilizan los usuarios, con los que trabajan y con los que deberá trabajar la aplicación.

En este diagrama las clases Reporte, Modelo Semántico y Origen de Datos son recursos del negocio del GDR. La clase Reporte utiliza la información contenida en los modelos semánticos para la confección de los reportes por lo que es imprescindible el modelo semántico. En el proceso de generación de un modelo se requiere un origen de datos que lo provea. Mientras que a partir de un origen de datos se pueden construir varios modelos. Un origen de datos contiene bases de datos. Hasta el momento el GDR 2 soporta los SGBD relacionales PostgreSQL, SQLServer y MySQL como se representa en la Figura 2.

2.1.1 Descripción de las clases del dominio

Reporte: Utiliza la información contenida en el modelo semántico para la confección de reportes, que serán exportados en formatos PDF, HTML, XLS, CSV, XLSX, PPTX, ODS, ODT, RTF, TXT, XML, XHTML, DOCX.

Modelo Semántico: Es un esquema de datos usado para almacenar en formato XML toda la información de los metadatos de las entidades de la base de datos, que serán utilizados en los reportes y previamente cargada en el origen de datos creado.

Origen de Datos: Contiene los datos que permiten conectar al Generador Dinámico de Reportes con el Sistema Gestor de Bases de Datos que se seleccione para obtener los datos de una de sus bases de datos. Las variables que maneja son: tipo de gestor de base de datos, dirección IP del servidor, puerto de conexión al servidor, usuario, contraseña y base de datos.

SQL: Esta clase representa los Sistemas Gestores de Bases de Datos relacionales a los que se les brinda soporte en el GDR 2.0.

PostgreSQL: Esta clase representa el Sistema Gestor de Bases de Datos PostgreSQL. **SQLServer:** Esta clase representa el Sistema Gestor de Bases de Datos SQLServer. **MySQL:** Esta clase representa el Sistema Gestor de Bases de Datos MySQL.

2.2 Requisitos del sistema

Son la descripción de los servicios y restricciones de un sistema de software, es decir, lo que el software debe hacer y bajo qué circunstancias debe hacerlo. Los requisitos son condiciones o capacidades que el sistema debe cumplir. Para ello se identifican las funcionalidades requeridas y las restricciones que se imponen clasificándose en funcionales y no funcionales. (53)

2.2.1 Requisitos funcionales

Los requisitos funcionales (RF) definen el comportamiento interno de un software. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. El componente de conexión a MongoDB como sistema de base de datos para el GDR 2.0 debe cumplir con los requisitos funcionales que a continuación se describen:

- **RF1**. Adicionar origen de datos MongoDB.

Descripción: se adiciona al sistema un origen de datos MongoDB.

Entrada: se especifica el nombre, servidor, puerto, usuario, contraseña y base de datos MongoDB a utilizar para crear el modelo.

Salida: se muestra la lista de orígenes de datos actualizada.

- **RF2**. Listar bases de datos MongoDB.

Descripción: el sistema deberá listar las bases de datos MongoDB existentes.

Entrada: recibe los datos necesarios para la conexión (tipo de gestor de base de datos, dirección IP del servidor, puerto de conexión al servidor, usuario, contraseña y base de datos) y le asigna un nombre al origen de datos.

Salida: se muestra las bases de datos pertenecientes al origen de datos MongoDB.

- **RF3**. Modificar origen de datos MongoDB.

Descripción: el sistema deberá permitir modificar el origen de datos MongoDB.

Entrada: recibe los datos de los parámetros a modificar del origen de datos seleccionado.

Salida: se modifica el origen de datos MongoDB.

- **FR4**. Listar colecciones.

Descripción: el sistema deberá brindar la funcionalidad de obtener las colecciones de la base de datos seleccionada.

Entrada: recibe como parámetro el origen de datos seleccionado para acceder a sus colecciones.

Salida: se muestra una lista de las colecciones de la base de datos.

- **FR5**. Listar claves de los documentos.

Descripción: el sistema deberá permitir obtener la clave de los documentos de las colecciones.

Entrada: recibe como parámetro las colecciones seleccionadas para acceder a sus documentos.

Salida: se muestra un listado con las claves asociadas a cada valor de los documentos que componen las colecciones.

- **RF6**. Adicionar modelo de datos.

Descripción: se adiciona al sistema un modelo de datos.

Entrada: recibe las colecciones y documentos seleccionados y se le proporcionan un nombre y su correspondiente descripción.

Salida: se muestra la lista de los modelos actualizada.

- **RF7**. Modificar modelo de datos.

Descripción: el sistema deberá permitir modificar el modelo de datos.

Entrada: recibe el nombre del modelo de datos a modificar.

Salida: se modifica el modelo de datos seleccionado.

- **RF8**. Ejecutar la consulta de MongoDB en formato SQL.

Descripción: el sistema deberá mostrar la respuesta de la consulta realizada a la base de datos MongoDB en el módulo Diseñador de Consultas.

Entrada: recibe la consulta SQL realizada a la base de datos MongoDB que le envía el servicio web.

Salida: se muestra la respuesta de ejecutar la consulta realizada a la base de datos MongoDB en el módulo Diseñador de Consultas.

- **RF9**. Exportar reporte.

Descripción: se obtienen los datos para la construcción del reporte.

Entrada: recibe como parámetro el identificador del modelo, el JRXML diseñado y el formato en el que se exportará.

Salida: se exporta el reporte seleccionado al formato predefinido.

2.2.2 Requisitos no funcionales (RNF)

Los RNF especifican criterios que pueden utilizarse para juzgar las operaciones que realiza un sistema. Constituyen propiedades o cualidades que el producto debe tener, tales como la fiabilidad, el tiempo de respuesta o precisión, capacidad de almacenamiento o tipo de plataforma, características estas que hacen al producto atractivo, usable, rápido y confiable.

Software

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux Ubuntu 12.04 o superior, Debian 7GNU/Linux .
- Paquetes: apache2, php5, php5-pgsql, php5-mysql, php5-sqlite, php5-curl, php5-cli, php5-gd, php5-odbc, php5-xsl, php5-ldap, php5-json, php5-cgi, libapache2-mod-php5, php5-intl, php5-xsl, php5-common, php5-dbg, php-apc, php5-mongo, tomcat7.

- Navegador Firefox en su versión 18.0 a la 29.0.1.
- Usuario con privilegios de administración.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos:

- Sistema Operativo: GNU/Linux Ubuntu 14.04, Debian 7GNU/Linux.
- PostgreSQL versión 9.1.
- PGAdmin III u otro administrador compatible con PostgreSQL.
- Usuario con privilegios para instalar la base de datos.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP utilizando el método de autenticación por md5.

Hardware

El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos:

- Procesador Intel Pentium 4 1.7 GHz como mínimo o AMD similar.
- Como mínimo requiere 1 GB de memoria RAM.
- Necesita espacio en disco duro igual o superior a 40 GB.

El servidor donde se instalará la base de datos debe cumplir con los siguientes requisitos:

- Procesador Intel Pentium 4 1.7 GHz o superior, o AMD similar.
- Necesita memoria RAM igual o superior a 1 GB.
- Como mínimo requiere 40 GB de espacio en disco duro.

La máquina utilizada para acceder a la aplicación debe cumplir con los siguientes requisitos:

- Como mínimo procesador Intel Pentium 4 1.7 GHz, o AMD similar.
- 256 MB RAM como mínimo.
- 20 GB de espacio en disco duro o superior.

Restricciones de Diseño e Implementación

- El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3.10.
- Se utilizará el framework de desarrollo Symfony en su versión 2.0 y la librería de JavaScript ExtJS versión 3.4.
- Se empleará la herramienta de desarrollo NetBeans 8.0 y el sistema gestor de base de datos postgreSQL 9.1.

Interfaz

- La interfaz debe ser amigable y fácil de usar, de manera que no sea una dificultad para los usuarios el trabajo con la misma.

Usabilidad

- El sistema debe permitir que usuarios sin mucha experiencia informática, puedan utilizarlo en su totalidad en un margen de tiempo relativamente corto.

Eficiencia

- El tiempo promedio de respuesta del sistema para la interacción de cualquier usuario con el mismo es de seis a ocho segundos.

2.3 Diagrama de caso de usos del sistema

El diagrama de casos de uso del sistema (DCUS) se utiliza para describir las funcionalidades de un software y documentar su comportamiento.

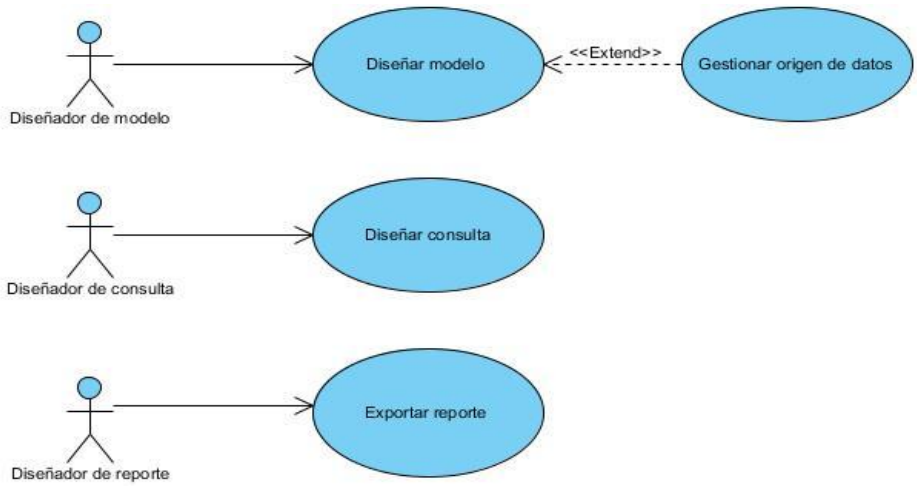


Fig. 3. Diagrama de casos de uso del sistema.

Los casos de uso del sistema (CUS) son operaciones o tareas específicas que se realizan tras una orden de algún agente externo, ya sea desde una petición de un actor o desde la invocación de otro caso de uso. Estos engloban los requisitos funcionales de un sistema, representando las funciones que la aplicación puede ejecutar.

En este DCUS el actor Diseñador de modelo es el encargado de inicializar el caso de uso Diseñar modelo, considerado crítico por su impacto significativo en la arquitectura del componente. En el diseño de un modelo semántico este actor tiene la posibilidad de gestionar orígenes de datos ya que son elementos necesarios para esta función, por tal razón puede inicializar el caso de uso Gestionar origen de datos. El actor Diseñador de Consulta inicializa el caso de uso Diseñar consulta mientras el actor Consultor de Reporte es el encargado de inicializar el caso de uso Exportar Reporte.

En la elaboración de este diagrama se aplican varios patrones de casos de uso, primeramente se aplica el patrón **Extensión Concreta** pues existe una relación de extensión entre el caso de uso base Diseñar modelo y el caso de uso extendido Gestionar origen de datos donde este último no altera el funcionamiento del primero, sino que se incorpora como parte integral del caso de uso base. También se ponen en práctica otros patrones de casos de uso tales como **CRUD completo** en el caso de uso Gestionar origen de datos pues se realizan todas las funcionalidades que propone este patrón (crear, leer, modificar y eliminar) y **CRUD parcial** en el caso de uso Diseñar modelo ya que se realizan solo algunas de las funciones (adicionar, leer, modificar) de un CRUD completo.

2.3.1 Descripción de los casos de uso

- **Gestionar origen de datos**: este CUS permite mostrar los orígenes de datos creados, modificar un origen de datos existente así como adicionar un origen de datos nuevo, luego de listar las bases de datos de ese origen y seleccionar una, desde donde se extraerá la información necesaria para la realización de los modelos y posteriormente los reportes.
- **Diseñar modelo**: este CUS permite adicionar un nuevo modelo a partir de listar las colecciones y los documentos que conforman el origen de datos seleccionado además de buscar o modificar un modelo específico de la lista de los modelos que han sido creados previamente.
- **Diseñar consulta**: este CUS permite diseñar una nueva consulta a partir de un modelo semántico seleccionado, obtener su resultado luego de ejecutarla y salvarla de modo que pueda ser utilizada en el módulo Diseñador de reporte.
- **Exportar Reporte**: este CUS permite exportar los reportes realizados en diferentes formatos: PDF, HTML, XLS, CSV, XLSX, PPTX, ODS, ODT, RTF, TXT, XML, XHTML, DOCX.

2.3.2 Especificación del Caso de Uso Diseñar modelo

Tabla 1. Especificación del Caso de Uso Diseñar modelo.

Objetivo	Diseñar un modelo
Actores	Diseñador de modelo.
Resumen	El caso de uso comienza al seleccionar el módulo "*Diseñador de modelos*", muestra los modelos y orígenes de datos que se encuentren creados.

	Además permite crear nuevos modelos, a partir de un origen de datos seleccionado así como buscar o modificar un modelo específico en la lista de los modelos previamente creados. Termina al crearse y guardarse un nuevo modelo.
Complejidad	Alta
Prioridad	Primaria
Precondiciones	Tiene que existir al menos un origen de datos creado. El actor debe estar autenticado y tener los privilegios asociados a este módulo.
Postcondiciones	Se crea un nuevo modelo y se actualiza el listado de modelos del sistema.

Flujo de eventos

Flujo básico Diseñar modelo

	Actor	**Sistema**
1.	Selecciona el módulo *"Diseñador de modelos".*	
2.		Muestra una interfaz con los modelos y orígenes de datos existentes, ver Figura 3. Interfaz principal del módulo Diseñador de modelos. Permite realizar acciones sobre un modelo tales como: -Crear un nuevo modelo. -Buscar un modelo. Ver expediente de proyecto: DATEC_GDR2.0_0114_ECU_2DM Sección 1: "Buscar modelo". Flujo básico Diseñar modelo. -Modificar un modelo. Ver Sección 1 "Modificar modelo".
3.	Selecciona un origen de datos.	

4.		Resalta los modelos que utilizan el origen de datos seleccionado.
5.	Selecciona la opción “Siguiente”.	
6.		Muestra una interfaz con las entidades pertenecientes al origen de datos seleccionado, agrupadas por colecciones. *En caso de que el actor seleccione la opción “Anterior”, ver Flujo Alterno 1.*
7.	Selecciona las entidades que conformarán el modelo de datos.	
8.		Habilita la opción “Siguiente”. *En caso de que el actor seleccione la opción “Anterior”, ver Flujo Alterno 1.*
9.	El actor selecciona la opción “Siguiente”.	
10.		Muestra los documentos asociados a cada entidad previamente seleccionada. *En caso de que el actor seleccione la opción “Anterior”, ver Flujo Alterno 2.*
11.	Selecciona la opción “Finalizar”.	
12.		Muestra un formulario para introducir el nombre y de forma opcional la descripción del modelo que se desea generar, ver Figura 4. Interfaz para adicionar modelo. *En caso de que el actor seleccione la opción “Cancelar”, ver Flujo Alterno 3.*
13.	El actor introduce un nombre para el nuevo modelo a generar.	

14.		Habilita la opción “Aceptar”. *En caso de que el actor seleccione la opción “Cancelar”, ver Flujo Alterno 3.*
15.	El actor selecciona el botón “Aceptar”.	
16.		Muestra la interfaz con el listado de los orígenes de datos y actualiza el listado de modelos terminando así el caso de uso.
Flujos alternos		
Nº Evento < Flujo alterno 1 : En caso de que se seleccione la opción “Anterior”>		
	Actor	**Sistema**
6.1.		El sistema retorna al paso 2 del Flujo básico Diseñar modelo.
Nº Evento < Flujo alterno 2 : En caso de seleccionar la opción “Anterior”>		
	Actor	**Sistema**
10.1		Retorna al paso 6 del Flujo básico Diseñar modelo.
Nº Evento < Flujo alterno 3 : En caso de seleccionar la opción “Cancelar”>		
	Actor	**Sistema**
12.1		Cierra el formulario donde se debe introducir el nombre del modelo a generar y retorna al paso 2 del Flujo básico Diseñar modelo.
Sección 1: “Modificar modelo”		
Flujo básico Diseñar modelo		
	Actor	**Sistema**
1	Selecciona el modelo que desea modificar.	

2		Resalta el origen de datos al cual está asociado el modelo seleccionado.
3	Selecciona la opción “Modificar”.	
4		Muestra una interfaz para modificar el nombre o la descripción del modelo. En caso de no modificar algún campo no se habilita el botón aceptar. En caso que se seleccione la opción “Cancelar”, ver flujo alterno 1.
5	Selecciona la opción “Aceptar”.	
6		Modifica el modelo actualizando la lista de modelos.
Flujos alternos		
Nº Evento <Flujo Alterno 1: En caso de seleccionar la opción “Cancelar” >		
	Actor	**Sistema**
2.1		Ignora la solicitud de modificación del modelo, volviendo al paso 2 del flujo básico Diseñar modelo.

Relaciones	**CU Incluidos**	-
	CU Extendidos	Gestionar origen de datos.
Requisitos funcionales	RF4, RF5, RF6, RF7.	
Requisitos no funcionales	Usabilidad	

Asuntos pendientes	-

2.3.3 Prototipo de interfaz de usuario

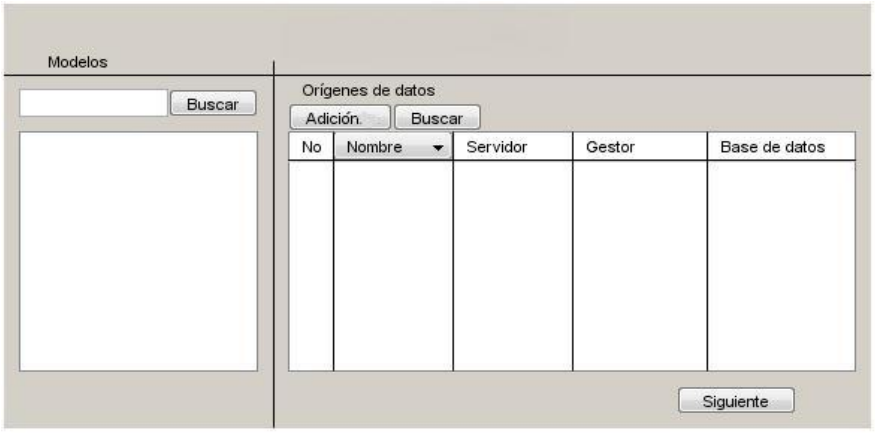


Fig. 4. Interfaz principal del módulo Diseñador de modelos.

Fig. 5. Interfaz para adicionar modelo.

2.4 Diagrama de clases del diseño para el CUS Diseñar modelo

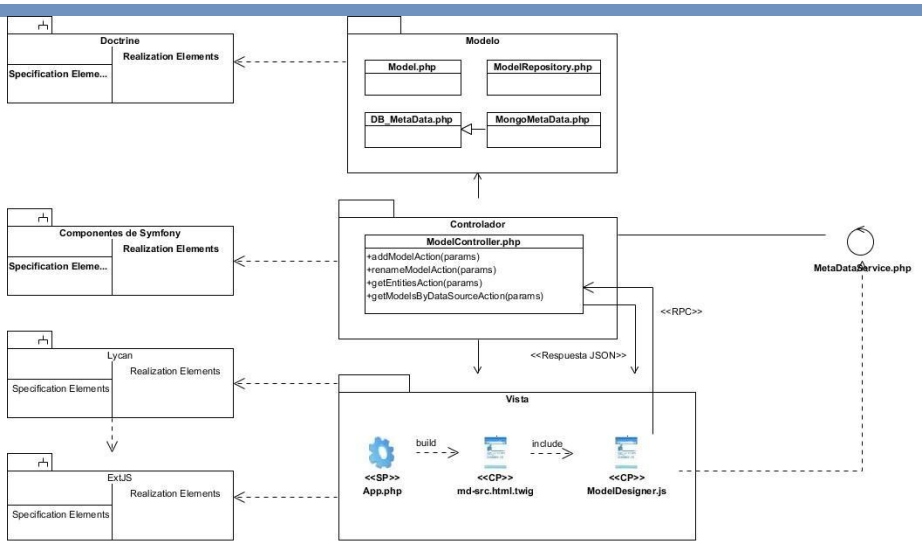


Fig. 6. Diagrama de clases del diseño para el caso de uso Diseñar modelo.

El diagrama de clases del diseño (DCD) muestra las propiedades y funcionalidades de cada clase en el lenguaje de desarrollo y tecnologías seleccionados. (54) Expresa la colaboración y responsabilidades de cada clase entorno al sistema que conforman y muestra cómo quedaría implementada toda la aplicación en términos lógicos.

El diagrama de clases del diseño que se presenta en la Figura 6 contiene las principales clases para el CUS Diseñar modelo. En el mismo las clases están agrupadas por paquetes en correspondencia con el patrón arquitectónico utilizado. En el paquete Vista se encuentran los ficheros JavaScript que interactúan en este CUS. El paquete Controlador contiene la clase PHP que manejan la lógica del negocio. Las clases necesarias para interactuar con la base de datos en Symfony2 se encuentran en el paquete Modelo, donde sus elementos corresponden a las clases generadas por el completo sistema de mapeo objeto-relacional Doctrine, modelado por el subsistema de igual nombre. El mismo genera las clases *Model.php* y *ModelRepository.php*.

En este paquete se encuentran además, las clases *BDMetadata.php* y *MongoMetaData.php* que contienen las acciones necesarias para realizar la conexión a la base de datos MongoDB al igual que todas las funcionalidades que permiten obtener la información de la misma para el correcto funcionamiento del sistema; permitiendo manejar orígenes de datos MongoDB e integrar las funciones del componente con la lógica de negocio establecida por el GDR. Las acciones son los elementos que construyen las respuestas del servidor, estas

se encuentran agrupadas según el correspondiente caso de uso al que pertenecen e implementadas en lenguaje PHP, con la dependencia del subsistema Symfony.

Las solicitudes se realizan por medio del protocolo Llamada a Procedimiento Remoto (RPC, por sus siglas en inglés) y las respuestas son devueltas en formato JSON. Los elementos del lado del cliente corresponden a componentes específicos del negocio y a los componentes genéricos reutilizables e implementados en lenguaje JavaScript utilizando los componentes del subsistema ExtJS. El sistema interactúa además con el subsistema Lycan.

2.5 Diagrama de secuencias

Un diagrama de interacción (secuencias o colaboración) explica gráficamente las relaciones existentes entre las instancias y las clases; es decir, se utilizan para organizar los eventos con la sucesión temporal en que se desencadenan. Este tipo de diagrama muestra los objetos que participan en la interacción mediante las líneas de vida y los mensajes que intercambian. (52)

El siguiente diagrama de secuencias pertenece al escenario Listar colecciones del CU Diseñar modelo.

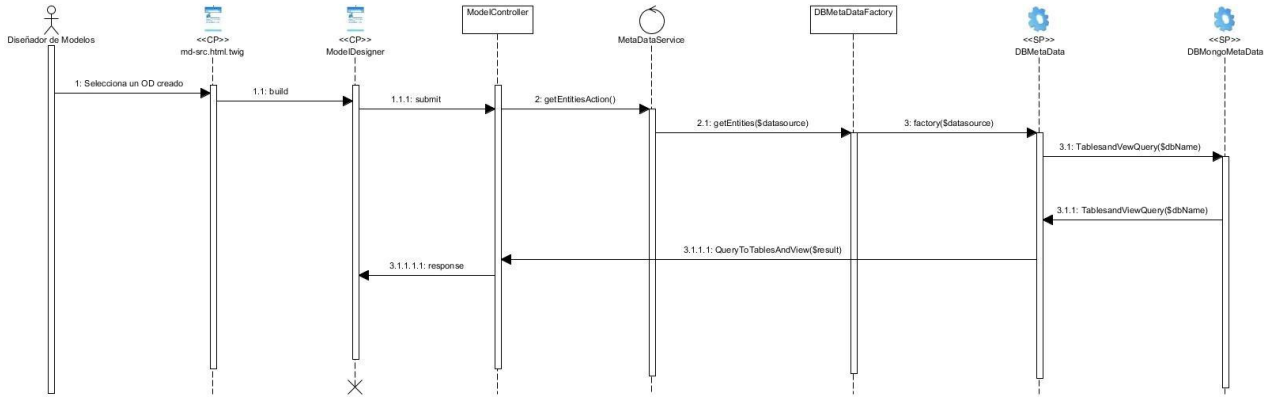


Fig. 7. Diagrama de secuencias para el escenario Listar colecciones del CU Diseñar modelo

En el diagrama de secuencias el actor Diseñador de Modelos selecciona uno de los orígenes de datos existentes en el sistema para listar las colecciones que contiene. En la clase *ModelDesigner.js* se registran todas las acciones referentes al módulo en cuestión, se selecciona el origen de datos a través de la operación “submit” al controlador ModelController.php, en este se ejecuta la función getEntitiesAction($params) que permite listar las colecciones del origen de datos seleccionado siendo necesario obtener el listado

de las entidades del OD utilizando el motor de inyecciones de dependencia de Symfony2 que ejecuta la función getEntities($dataSource) a través de *DBMetaDataService.php*. Internamente este crea una instancia del SGBD MongoDB en el método *factory($datasource)* de la clase *DBMetaDataFactory.php* pasando por parámetro el *datasource* seleccionado. Posteriormente, la clase *DBMetaData.php* delega la responsabilidad a la clase *DBMongoMetaData.php* de ejecutar la función TablesandViewQuery($dbName) que se encarga de proveer la consulta para obtener los metadatos de las entidades del OD seleccionado, luego el método queryToTablesAndViews($result) transforma el resultado al formato establecido y lo envía al cliente para que los muestre.

2.6 Patrones de software

Los patrones de software estandarizan principios y buenas prácticas en la solución de sistemas informáticos. Estos han permitido agrupar soluciones a problemas existentes en la construcción de aplicaciones y generar una respuesta común que resuelva de forma genérica las principales deficiencias en la creación de software.

2.6.1 Patrones de diseño

Los patrones de diseño constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Además son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. (55)

En la solución del sistema se aplicaron principalmente los siguientes patrones:

- **Controlador:** El uso de este patrón se evidencia con la existencia de la clase controladora *ModelController.php* que tiene la responsabilidad de interactuar con las clases de acceso a datos, obtener la información necesaria y dar respuesta a las peticiones del cliente controlando así el flujo central de las acciones en el sistema.
- **Experto:** El uso de este patrón se evidencia con la existencia de la clase *ModelDesigner.js* indicando que la responsabilidad de la creación del componente debe recaer sobre la misma, ya que posee toda la información de la lógica del CU.
- **Creador:** El uso de este patrón se evidencia con la existencia de la clase *MongoMetaData.php* para crear objetos de este tipo en la clase *DBMetaDataFactory.php.*

- **Bajo acoplamiento:** El uso de este patrón se evidencia con la existencia de la clase *DBMetaData.php* lo que permite reducir el impacto en modificaciones que intervengan en la misma buscando posibilidades de reutilización. Esto trae como ventaja que solo se realicen acciones sobre el tipo de entidad que se solicite y no sobre todo el conjunto.
- **Alta cohesión:** El uso de este patrón se evidencia con la existencia de la clase *MongoMetaData.php* al asignarle sus responsabilidades procurando que la cohesión sea lo más alta posible.
- **DAO (Objeto de Acceso a Datos):** El uso de este patrón se evidencia con la existencia de la clase *GdrConexionService.php* en la cual se tiene implementado el método *getGdrPDO()*. En este método es donde se configura la conexión a los distintos gestores de base de datos relacionales que soporta GDR y se hace una instancia de este en las clases de acceso a datos para reconfigurar la conexión según el gestor que sea.
- **Factory:** El uso de este patrón se evidencia con la existencia de la clase *DBMetaDataFactory.php* en la que en el método *factory ()* decide cuál de las implementaciones instanciar y la crea a partir de tener el origen de datos; es decir, recibiendo el origen de datos como parámetro.
- **Singleton:** Se evidencia el uso de este patrón en la clase *DBMetaData.php* la cual contiene el método *configureConnection()* para configurar la conexión a los diferentes gestores de base de datos que el GDR soporta.

2.6.2 Patrones de arquitectura

Los patrones arquitectónicos ofrecen soluciones a problemas de arquitectura en la ingeniería de software. Estos expresan un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. Las arquitecturas más utilizadas para el desarrollo de software son: Monolítica, Cliente-Servidor y Tres Capas. Esta última constituye una especialización de la arquitectura Cliente-Servidor, donde la carga se divide en tres capas, con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (modelado el negocio) y una tercera para el almacenamiento (persistencia), una capa solamente tiene relación con la siguiente.

MVC

Modelo Vista Controlador acrónimo de (MVC, por sus siglas en inglés) es un patrón de arquitectura que pertenece a la familia de los estilos arquitectónicos de Llamada y Retorno. El mismo separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos: Modelo, Vista y Controlador. Por ello el modelo encapsula los datos y la funcionalidad de la aplicación. La vista despliega la información contenida en el modelo. El controlador está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del modelo o de la vista. De ahí que su empleo provee claridad en el diseño y facilita una mayor escalabilidad.

2.7 Diagrama de Despliegue

El diagrama de despliegue permite mostrar la arquitectura, en tiempo de ejecución, del sistema respecto al hardware y software. Este se utiliza en el diseño y la implementación, es más limitado que el diagrama de componentes, en el sentido de que representa la estructura del sistema solo en tiempo de ejecución, pero no en tiempo de desarrollo o compilación. (56)

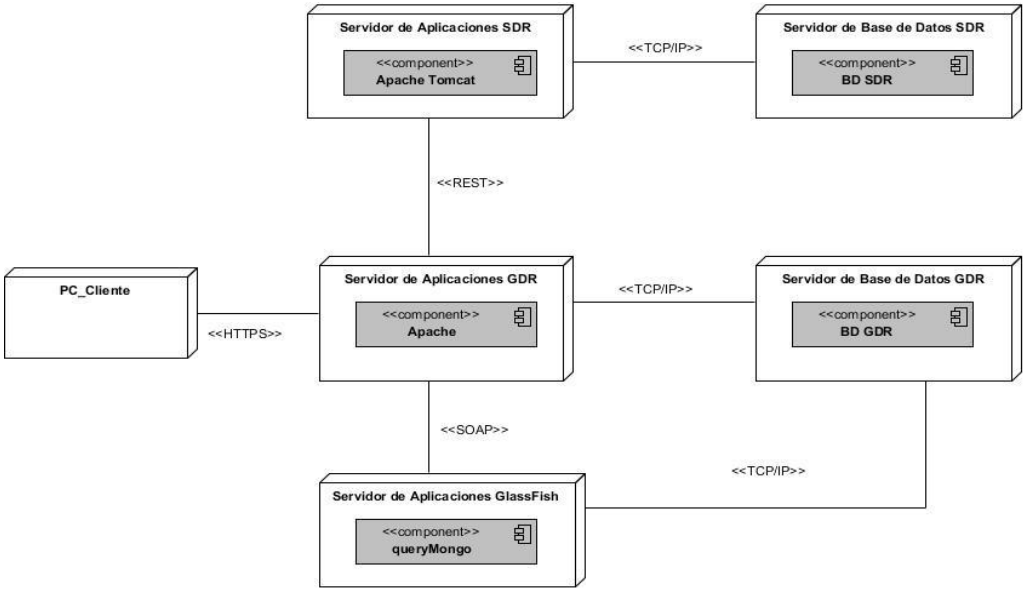


Fig. 8. Diagrama de Despliegue.

En la solución el diagrama de despliegue se distribuye por una estación de trabajo para el cliente desde la cual se realizarán peticiones por el protocolo HTTPS al servidor de aplicaciones web representado por el nodo Generador Dinámico de Reportes (GDR) que utiliza el servidor de aplicaciones Servidor Dinámico de Reportes (SDR) mediante el

protocolo de comunicación REST y también el Servidor de Aplicaciones GlassFish a través del protocolo SOAP para ejecutar el servicio web queryMongo. Los servidores de aplicaciones GDR y SDR acceden de forma local o remota a los orígenes de datos especificados por el cliente mediante el protocolo TCP/IP así como también el servicio web queryMongo. A continuación se muestra el diagrama de despliegue y se describen los elementos que lo conforman:

- **PC_Cliente:** nodo de no procesamiento desde donde se ejecuta la *url* para acceder a la aplicación. Debe tener un procesador Intel Pentium 4 1.7 GHz, 512 GB RAM y 80 GB de espacio en disco duro.
- **Servidor de Aplicaciones GDR:** nodo de procesamiento donde está publicada la aplicación. Debe tener un procesador Intel Pentium 4 1.7 GHz, 2 GB de memoria RAM y un disco duro de 10 GB.
- **Servidor de Aplicaciones SDR:** nodo de procesamiento que contiene al motor de reportes SDR.
- **Servidor de Bases de Datos GDR:** nodo de procesamiento que contiene toda la información del GDR, contiene con procesador Intel Pentium 4 1.7 GHz, una memoria RAM de 2 GB y 6 GB de espacio en disco duro.
- **Servidor de Bases de Datos SDR:** nodo de procesamiento que contiene toda la información del SDR. Debe tener un procesador Intel Pentium 4 1.7 GHz, una memoria RAM de 1 GB y 2 GB de espacio en disco duro.
- **Servidor de Aplicaciones Glassfish:** nodo de procesamiento que contiene el servicio web queryMongo aplicación en java que ejecuta la consulta SQL y devuelve el resultado al Servidor de Aplicaciones GDR a partir de utilizar el JDBC de MongoDB.
- **HTTPS:** protocolo que se utiliza para conectar la computadora del cliente y el servidor donde está la aplicación de forma segura.
- **TCP/IP:** protocolo que se utiliza para conectar el servidor de aplicaciones con las bases de datos.
- **REST:** protocolo de comunicación que se utiliza para conectar los servidores de aplicaciones Servidor Aplicaciones SDR y Servidor Aplicaciones GDR.
- **SOAP:** protocolo simple de acceso a objetos que forma la base de los servicios web, permite ejecutar consultas en formato SQL sobre una base de datos MongoBD.

2.8 Conclusiones parciales

En el desarrollo del presente capítulo se realizó modeló la solución propuesta definiendo el CUS Diseñar Modelo como el más crítico por su impacto en la arquitectura de la solución, se identificaron nueve requisitos funcionales agrupados en cuatro casos de uso y seis requisitos no funcionales con los cuales el sistema debe cumplir. También se elaboró el diagrama de clases del diseño para mostrar las principales propiedades y funcionalidades de las clases que conforman el caso de uso Diseñar Modelo y la relación entre ellas. De igual manera, se confeccionó el diagrama de secuencias que permite inspeccionar los aspectos dinámicos del caso de uso en cuestión y finalmente, se modeló el diagrama de despliegue teniendo en cuenta que la solución se incorporará al Generador Dinámico de Reportes 2.0.

CAPÍTULO 3

Implementación y pruebas

Capítulo lll: Implementación y pruebas.

Introducción

Una vez concluida la fase de diseño, en este capítulo, se realizan las actividades que se llevan a cabo durante la fase de implementación y pruebas. Se modela el sistema en términos de componentes, se especifican los tipos, niveles, métodos y casos de prueba que se realizarán al componente de conexión. Y además, se muestran ejemplos de las implementaciones más relevantes.

3.1 Modelo de implementación

El modelo de implementación, describe cómo los elementos del modelo del diseño y las clases se implementan en términos de componentes, como ficheros de código fuente y ejecutables. Describe también, como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles, en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y como dependen los componentes unos de otros. (57)

3.1.1 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes, mostrando las dependencias que existen entre ellos. Los componentes físicos incluyen: archivos, cabeceras, bibliotecas compartidas, módulos, ejecutables y paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema; estos son utilizados para modelar la vista estática y dinámica de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se modelan por partes; cada diagrama describe un apartado del sistema. En él, se situarán librerías, tablas, archivos, ejecutables y documentos que formen parte del sistema. Uno de

los usos principales es que pueden servir para mostrar qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. (57)

En la figura se muestra, el diagrama de componentes perteneciente al CU Diseñar Modelo:

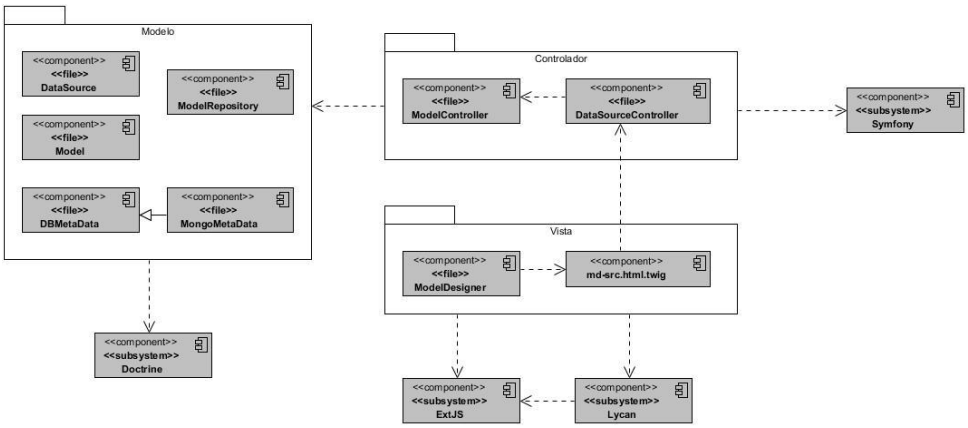


Fig. 9. Diagrama de componentes para el CU Diseñar modelo.

En el diagrama de componentes de la Figura 9 el componente *ModelController.php* agrupa todas las acciones que intervienen en el caso de uso Diseñar Modelo, cada una de estas en su respectivo fichero PHP. Los componentes *Model.php* y *ModelRepository.php* del modelo corresponden a las dos clases generadas por el subsistema Doctrine por cada una de las tablas de la base de datos del sistema y los componentes *DBMetaData.php* y *MongoMetaData.php* engloban las funcionalidades mediante las cuales se accede a las bases de datos MongoDB y se manipula la información correspondiente. Mientras que los componentes *ModelDesigner.js* y *md-src.html.twig* representan las diferentes vistas que intervienen en este caso de uso.

3.2 Estándar de codificación

El estándar de codificación es el método que permite convertir un carácter de un lenguaje natural en un símbolo de otro sistema de representación, aplicando normas o reglas de codificación para definir la forma en la que se codifica dado el sistema de representación. (58) Es necesario definirlo pues contar con un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo fortaleciendo el desarrollo del mismo y proporcionando que su código sea mantenible.

Un estándar de codificación comprende todos los aspectos de la generación de código. (59) Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento.

Se utilizó el estándar de codificación que define DATEC de los cuales se muestran algunos de los más utilizados:

- Todos los métodos, nombres de clases y variables se escribirán en estructura “camelCase”, comenzando siempre en minúscula.
- Los comentarios deben ser escritos correctamente y claros. Generalmente deben usarse comentarios de una sola línea.
- Cada variable debe de ser declarada en una línea y el nombre debe de comenzar con letras minúsculas.
- Las constantes pueden contener tanto caracteres alfanuméricos como guiones bajos.

3.3 Implementaciones relevantes

```php
public function configureConnection() {

    $dbms = $this->getDbms();
    $host = $this->getDataSource()->getHost();
    $port = $this->getDataSource()->getPort();
    $user = $this->getDataSource()->getUsername();
    $db = $this->getDataSource()->getDatabase();
    $pass = $this->getDataSource()->getPassword();
    $dsn = $this->createDSN($dbms, $host, $port, $db);

    if (!$dsn) {
        throw new \Exception('Error');
    } else {
        return new \MongoClient($dsn);
    }
}
```

Fig. 10. Instancia de conexión a una base de datos MongoDB.

El GDR utiliza PDO para conectarse a los SGBD relacionales pero el mismo no brinda soporte para MongoDB, por lo que fue necesario redefinir el método de conexión al gestor utilizando el driver de PHP para este caso específico.

```php
public function query($query, $con) {
    if (is_array($query)) {
        return $query;
    }
    if ($this->sql_query($query)) {
        try {
            return $this->sql_to_mongo_query($query);
        } catch (\Exception $exc) {
            $this->setErrorInfo(TRUE);
            return $exc->getTraceAsString();
        }
    }
    return $con->$query();
}
```

Fig. 11. Instancia de método query () sobre una base de datos MongoDB.

El método *query()* recibe por parámetro la instancia de conexión a la base de datos y la consulta a ejecutar sobre ella. En caso de que la consulta tenga formato SQL, se retorna el resultado de esta mediante el servicio web implementado que permite ejecutar consultas en formato SQL sobre una base de datos MongoDB.

```php
public function sql_to_mongo_query($query) {
    $sqlTransform = '';
    //obteniendo host del servicio web para ejecutar la consulta
    $config = parse_ini_file("jdbcWebServer.ini");
    $soap = new \SoapClient($config['host'].$config['url_function']);
    $resul = $soap->QueryMongo(array('host' => $this->getDataSource()->getHost(), 'port' => $this->getDataSource()->getPort(),
                                     'db' => $this->getDataSource()->getDatabase(), 'user' => $this->getDataSource()->getUsername(),
                                     'passw' => $this->getDataSource()->getPassword(), 'query' => $query));
    $resul = $resul->return;
    $sqlTransform = rtrim($resul, "\0");
    $sqlTransform = $this->array_objet_to_array(json_decode($sqlTransform));
    return $sqlTransform;
}
```

Fig. 12. Método para ejecutar consultas SQL sobre una base de datos MongoDB.

El método *sql_to_mongo_query()* permite ejecutar una consulta en formato SQL en una base de datos MongoDB utilizando el servicio web implementado, al que se conecta mediante SOAP pasando por parámetros todos los datos necesarios para ejecutar la consulta.

3.4 Pruebas de software

Las pruebas de software son un elemento esencial para la garantía de calidad del software y representan una revisión final de las especificaciones del diseño y de la codificación. Estas consisten en la dinámica de la verificación del comportamiento de un programa en un

conjunto finito de casos de prueba. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo (60).

3.4.1 Estrategia de prueba

Una estrategia de prueba describe el enfoque y los objetivos generales de este tipo de actividad. Incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos. Define qué técnicas (manual o automática) y qué herramientas serán usadas. Delimita los criterios de éxitos y culminación de las pruebas. También define consideraciones especiales relacionadas con los recursos necesarios para realizar esta tarea. (52)

3.4.2 Nivel de Prueba

Los niveles de prueba especifican diferentes ángulos para verificar y validar un producto de software. Existen varios niveles de pruebas como: pruebas de desarrollador, independiente, unidad, integración, sistema y aceptación, donde cada nivel contiene una técnica de prueba específica según los atributos de calidad que se deseen verificar. En el desarrollo del componente se aplicarán solamente las pruebas siguientes para comprobar que el sistema da respuesta a los requisitos funcionales definidos anteriormente:

- **Nivel de Desarrollador:** es el primer nivel de pruebas, diseñadas e implementadas por el equipo de desarrollo, donde el programador es quien revisa su código fuente.
- **Nivel de Sistema:** en este nivel las pruebas tienen como propósito ejercitar profundamente el sistema para verificar que se han integrado adecuadamente todos los elementos del sistema (hardware, software) y que realizan las funciones adecuadas.
- **Nivel de Integración:** en este nivel se prueba los componentes combinados para ejecutar un CUS. Además se realiza la prueba para descubrir errores en las especificaciones de las interfaces de las clases.
- **Nivel de Aceptación:** en este nivel las pruebas se realizan con el objetivo de detectar fallas en la implementación del sistema.

3.4.3 Tipos de Pruebas

Para verificar que el componente de conexión cumple con todos los requisitos identificados en el proceso de análisis y comprobar su correcto funcionamiento se pueden realizar varios tipos de prueba, a continuación se explica en qué consiste cada uno de los seleccionados.

- **Pruebas Funcionales:** Las pruebas funcionales están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Se elaboran mediante el diseño de modelos de prueba que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.
- **Pruebas de Rendimiento (Carga y *stress*):** están diseñadas para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. Se realizan basándose en la funcionalidad del sistema bajo cargas pesadas, un gran número de repeticiones, manejo de grandes datos y demasiadas preguntas a bases de datos grandes.
- **Pruebas de Integración:** Las pruebas de integración constituyen un conjunto de pruebas unitarias, funcionales, de regresión y aceptación que se realizan para probar el software. Enmarcan su atención en la integración de los componentes de una aplicación y su objetivo es verificar que el sistema funcione correctamente una vez integrado. Según Roger Pressman, existen dos tipos de integración, no incremental e incremental. En la integración no incremental se combinan todos los módulos y se prueba el sistema en su conjunto; mientras que la integración incremental es aquella en la que el sistema se prueba a medida que se construye, dividiendo el ciclo de prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir. (61). En la presente solución se aplicaron pruebas de integración incremental.
- **Pruebas de Aceptación:** Estas pruebas se realizan en conjunto con el cliente para de este modo definir su aceptación con respecto al sistema. En esencia son pruebas funcionales y de sistema sobre todo el producto y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas se realizan una vez pasadas todas las pruebas por parte del desarrollador para garantizar la mayor calidad posible del sistema.

3.4.4 Método de Prueba

Las pruebas funcionales utilizan el método de Caja Negra el cual describe las pruebas que se aplican sobre la interfaz del software utilizando los casos de prueba. Con estos, se pretende demostrar que las funciones del software son operativas, se definen las entradas al sistema y los resultados esperados de estas. Son diseñados para validar los requisitos funcionales sin detenerse en el funcionamiento interno del programa.

Técnica de prueba:

Partición equivalente: es una técnica del método de prueba de caja negra que divide el campo de entrada de un programa en clases de equivalencia de las que se pueden derivar casos de prueba.

3.5 Diseño de casos de prueba

Un caso de prueba se diseña según las funcionalidades descritas en los casos de uso. Se parte de la descripción de estos últimos, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión. Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y estas a su vez en escenarios, para hacer más fructífera la ejecución de las pruebas. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. A continuación se presentan las tablas de las secciones probadas para el caso de uso Diseñar Modelo.

Tabla 2. Sección de prueba para el CUS Diseñar Modelo

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1 diseñar Modelo	SC 1.1: Seleccionar origen de datos.	El diseñador de modelos decide seleccionar un origen de datos y el

		sistema muestra todas las entidades que existen en el origen de datos.
	SC 1.2: Seleccionar entidades.	El diseñador de modelos escoge las entidades que desea para el reporte y el sistema muestra todos los atributos de las entidades seleccionadas.
	SC 1.3: Editar atributos	El diseñador puede modificar los atributos de las entidades seleccionadas y el sistema muestra un formulario para el nombre del modelo.
	SC 1.4: Nombrar el modelo	El diseñador de modelos introduce el nombre del modelo y el sistema crea y muestra el modelo en la lista de modelos.

A partir de esta descripción se detallan las variables que se encuentran asociadas al caso de uso Diseñar Modelo.

Tabla 3. Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Origen de datos	entidad de tipo origen de datos	no	Origen de datos a partir del cual se crea el modelo.
2	Entidades	casilla de verificación	no	Entidades del origen de datos seleccionado que pasan a formar parte del modelo.
3	Nombre	campo de texto	no	Nombre que se le da al nuevo modelo de datos.

4	Descripción	área de texto	si	Breve descripción del modelo de datos que se renombra.

Luego de haber descrito las variables se realizó la matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se registraron los resultados de las pruebas, con el empleo de la técnica de partición de equivalencia. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

3.5.1 Matriz de datos

Tabla 4. Matriz de datos. Escenario seleccionar origen de datos.

Escenario	**Variables (enumeradas según descripción de las variable)**				**Descripción**	**Respuesta del Sistema**	**Resultado de la Prueba**	**Flujo Central**
	1	**2**	**3**	**4**				
EC 1.1: Nombrar el modelo correctamente	NA	NA	V	NA	En este escenario se introduce un nombre al modelo de manera correcta.	El sistema adiciona un nuevo modelo de datos y muestra el mensaje: “Operación realizada correctamente.”	Satisfactorio.	1- Acceder al módulo Diseñador de Modelos. 2- Seleccionar un origen de datos. 3- Pulsar botón Siguiente. 4- Seleccionar entidades para el modelo. 5- Pulsar botón Siguiente. 6- Pulsar botón Finalizar. 7- Introducir nombre. 8- Pulsar botón Aceptar.

EC 1.2: Nombrar el modelo incorrectamente.	NA	NA	I	NA	En este escenario se introduce un nombre no permitido al modelo.	Destaca de color rojo el marco del campo de texto y no habilita el botón aceptar para crear el modelo.	Satisfactorio.	1- Acceder al módulo Diseñador de Modelos. 2- Seleccionar un origen de datos. 3- Pulsar botón Siguiente. 4- Seleccionar entidades para el modelo. 5- Pulsar botón Siguiente. 6- Pulsar botón Finalizar. 7- Introducir nombre. 8- Pulsar botón Aceptar.

3.6 Resultados de las pruebas

3.6.1 Pruebas funcionales de Caja Negra

Una vez realizadas las pruebas funcionales mediante el método de Caja Negra a través de la técnica de partición equivalente, definiéndose los casos de prueba asociados a cada CUS, se demostró el correcto funcionamiento del componente, así como la correcta validación de los campos comprobando que sean aceptados todos los caracteres válidos que les corresponde a cada uno. En el proceso de pruebas se detectaron 12 no conformidades, las cuales fueron corregidas y gestionadas correctamente en cada iteración. La siguiente tabla muestra un resumen de las deficiencias encontradas luego de realizar el caso de prueba Diseñar modelo, en el cual se verificó que el sistema permite diseñar modelos, chequeando que sólo se acepten los caracteres válidos para cada campo.

Tabla 5: No conformidades asociadas al CU Diseñar Modelo.

Fecha	Iteración	Caso de Prueba	Tipo de Error	Descripción

20/05/2015	1	CU Diseñar modelo	Ortografía	El campo Nombre no aceptaba las tildes.
20/05/2015	1	CU Diseñar modelo	Validación	El campo Descripción permitía caracteres numéricos como inicio de cadena.
22/05/2015	2	CU Diseñar modelo	Validación	El campo nombre no tenía definido un límite de caracteres de entrada.
22/05/2015	2	CU Diseñar modelo	Programación	El sistema no notifica la ocurrencia de errores cometidos.
25/05/2015	3	CU Diseñar modelo	_	

A continuación se muestran las no conformidades que se identificaron en cada iteración, asociadas a cada caso de uso. En la primera iteración se identificaron siete, de las cuales tres fueron de validación dos de ortografía y dos de programación, en la segunda iteración cinco, dos de validación, dos de ortografía y una de programación y en la tercera iteración no se encontraron no conformidades.

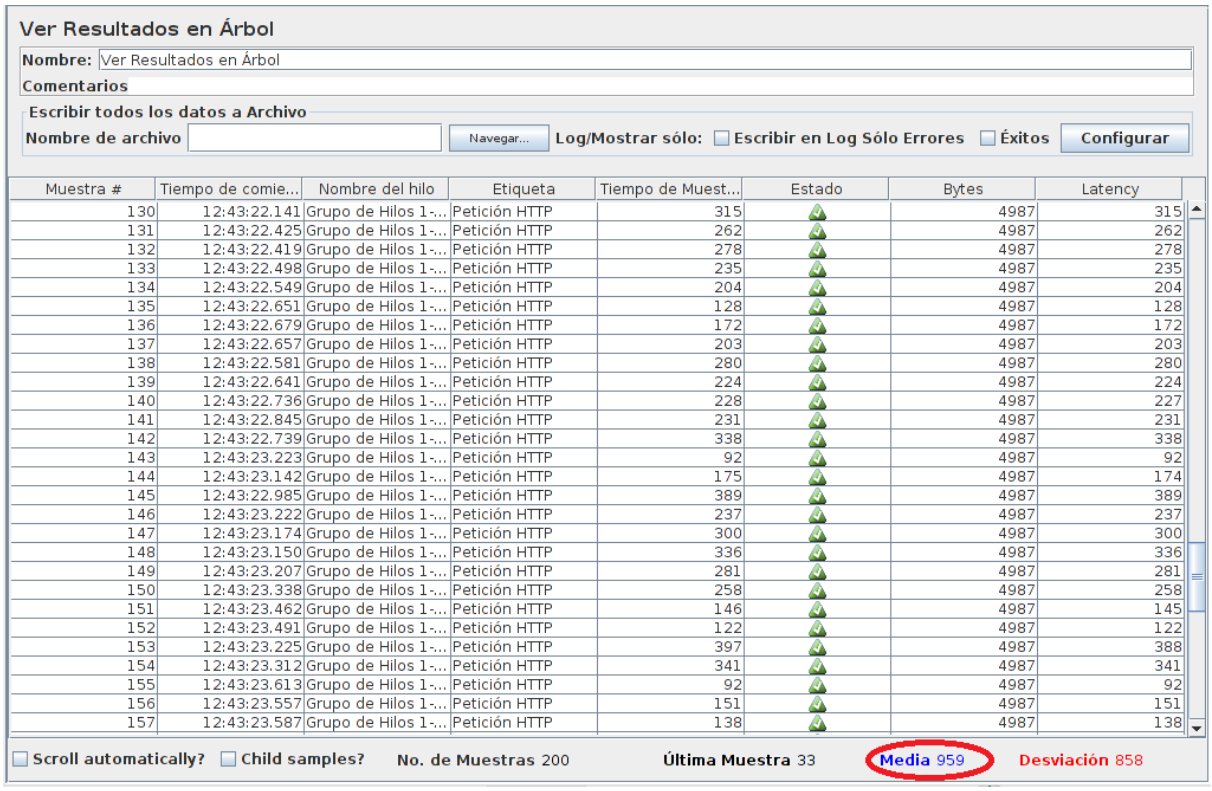
Tabla 6: Resumen de las no conformidades asociadas a los casos de uso.

Casos de uso	Iteración 1	Iteración 2	Iteración 3
Diseñar modelo	2	2	_
Gestionar origen de datos	2	1	_
Diseñar consulta	3	1	_
Exportar reporte	_	_	_

3.6.2 Prueba de rendimiento de carga y stress

Para observar el comportamiento del servicio web implementado bajo una cantidad de peticiones esperadas se realizaron las pruebas de rendimiento. En la ejecución de estas pruebas se empleó la herramienta JMeter para comprobar la velocidad con la que el mismo opera con 100 usuarios conectados de forma concurrente, el sistema presenta un tiempo de respuesta de 0.95 segundos, resultado satisfactorio que demuestra la eficiencia del servicio web ante situaciones de carga y stress.

En la siguiente figura se muestra el resultado de realizar la prueba de rendimiento de carga y *stress* al servicio web implementado:



Muestra #	Tiempo de comie...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Latency
130	12:43:22.141	Grupo de Hilos 1-...	Petición HTTP	315		4987	315
131	12:43:22.425	Grupo de Hilos 1-...	Petición HTTP	262		4987	262
132	12:43:22.419	Grupo de Hilos 1-...	Petición HTTP	278		4987	278
133	12:43:22.498	Grupo de Hilos 1-...	Petición HTTP	235		4987	235
134	12:43:22.549	Grupo de Hilos 1-...	Petición HTTP	204		4987	204
135	12:43:22.651	Grupo de Hilos 1-...	Petición HTTP	128		4987	128
136	12:43:22.679	Grupo de Hilos 1-...	Petición HTTP	172		4987	172
137	12:43:22.657	Grupo de Hilos 1-...	Petición HTTP	203		4987	203
138	12:43:22.581	Grupo de Hilos 1-...	Petición HTTP	280		4987	280
139	12:43:22.641	Grupo de Hilos 1-...	Petición HTTP	224		4987	224
140	12:43:22.736	Grupo de Hilos 1-...	Petición HTTP	228		4987	227
141	12:43:22.845	Grupo de Hilos 1-...	Petición HTTP	231		4987	231
142	12:43:22.739	Grupo de Hilos 1-...	Petición HTTP	338		4987	338
143	12:43:23.223	Grupo de Hilos 1-...	Petición HTTP	92		4987	92
144	12:43:23.142	Grupo de Hilos 1-...	Petición HTTP	175		4987	174
145	12:43:22.985	Grupo de Hilos 1-...	Petición HTTP	389		4987	389
146	12:43:23.222	Grupo de Hilos 1-...	Petición HTTP	237		4987	237
147	12:43:23.174	Grupo de Hilos 1-...	Petición HTTP	300		4987	300
148	12:43:23.150	Grupo de Hilos 1-...	Petición HTTP	336		4987	336
149	12:43:23.207	Grupo de Hilos 1-...	Petición HTTP	281		4987	281
150	12:43:23.338	Grupo de Hilos 1-...	Petición HTTP	258		4987	258
151	12:43:23.462	Grupo de Hilos 1-...	Petición HTTP	146		4987	145
152	12:43:23.491	Grupo de Hilos 1-...	Petición HTTP	122		4987	122
153	12:43:23.225	Grupo de Hilos 1-...	Petición HTTP	397		4987	388
154	12:43:23.312	Grupo de Hilos 1-...	Petición HTTP	341		4987	341
155	12:43:23.613	Grupo de Hilos 1-...	Petición HTTP	92		4987	92
156	12:43:23.557	Grupo de Hilos 1-...	Petición HTTP	151		4987	151
157	12:43:23.587	Grupo de Hilos 1-...	Petición HTTP	138		4987	138

Fig. 13. Prueba de rendimiento realizada al sistema utilizando la herramienta JMeter

3.6.3 Prueba de integración

Una vez integrado el componente de conexión al Generador Dinámico de Reportes, se pudo apreciar que funciona correctamente en cuanto a la interacción entre sus diferentes módulos. Una vez el usuario haya adicionado el origen de datos puede realizar todas las operaciones siguientes como listar las bases de datos, obtener las entidades de las bases

de datos y obtener sus campos para crear el modelo. Ya creado el modelo, el usuario puede acceder al módulo Diseñador de Consultas para crear la consulta asociada a la información solicitada por el usuario y seguidamente pasar al módulo Diseñador de Reportes para elaborar el informe deseado. Luego en el módulo Visor de Reportes, exportar dicho reporte usando como fuente de datos la información contenida en los modelos semánticos. También permite administrar los reportes diseñados previamente, a través del módulo Administrador de Reportes y garantizar la seguridad del sistema mediante el módulo Administrador de Seguridad.

3.6.4 Prueba de aceptación

Como constancia de la aceptación del producto, se realizó la prueba por el cliente junto a especialistas del equipo de desarrollo del GDR 2.0, cuya aceptación del componente de conexión a MongoDB como gestor de bases de datos NoSQL se evidencian a través una carta de aceptación por parte del cliente hacia el producto desarrollado.

3.7 Conclusiones Parciales

En el desarrollo del presente capítulo se realizó la descripción de la implementación del componente de conexión, y se representó su estructura en el diagrama de componentes. Se definieron los estándares de codificación que fueron utilizados. Además se definieron como niveles de pruebas usados el nivel de desarrollador, el de sistema, el de integración y el de aceptación utilizando las pruebas funcionales, de rendimiento con carga y *stress*, de integración y de aceptación empleando el método de caja negra y utilizando la técnica de partición de equivalencia. Una vez desarrolladas estas pruebas fueron resueltas todas las no conformidades detectadas en cada iteración.

Conclusiones generales

Culminando el desarrollo de la presente solución se arriba a las siguientes conclusiones:

- El estudio y desarrollo de la fundamentación teórica garantizó definir el sistema de base de datos a utilizar entre los sistemas gestores de bases de datos no relacionales.
- Se le incorporó soporte al Generador Dinámico de Reportes para el SGBD no relacional MongoDB permitiendo a sus usuarios interactuar con el sistema sin tener conocimientos previos sobre este, pues la solución se comporta de igual forma que con los demás SGBD relacionales soportados anteriormente.
- Se implementó un servicio web que permite obtener la información resultante de realizar las consultas a bases de datos MongoDB en formato SQL aprovechando las potencialidades del JDBC utilizado.
- Se aplicaron pruebas funcionales de tipo caja negra descritas en los casos de pruebas realizados además de pruebas de rendimiento a través de carga y *sstres* y pruebas de integración, validando con estas el correcto funcionamiento del componente de conexión a MongoDB para el Generador Dinámico de Reportes.

Recomendaciones

Al concluir este trabajo se recomienda:

- Incorporar la funcionalidad de ejecutar la consulta a una base de datos MongoDB en formato SQL en el Servidor Dinámico de Reportes.
- Incorporarle al GDR el soporte a otros gestores de bases de datos no relacionales.

Bibliografías

1. **Universidad de las Ciencias Informáticas.** Universidad de las Ciencias Informáticas. [En línea] 10 de septimbre de 2012. [Citado el: 15 de octubre de 2014.] http://www.uci.cu/?q=mision.

2. **GESPRO.** XEDRO gespro.Suite de Gestión de proyectos. [En línea] 19 de octubre de 2013. [Citado el: 28 de octubre de 2014.] http://gespro.datec.prod.uci.cu/.

3. **Cobo, Angel.** *Diseño de programacion de base de datos.* s.l. : Didactica escolar. s.l. :Vision Libros, 2007.

4. **Portal de la Innovación en Euskadi.** Portal de la Innovación en Euskadi: Euskadi+innova. [En línea] 15 de noviembre de 2014. [Citado el: 26 de octubre de 2014.] http://www.euskadinnova.net/es/enpresa-digitala/agenda/mongodb-bbdd-nosql-popular-mercado/7623.aspx.

5. **MappingGIS.** RDBMS SQL vs NonRDBMS NoSQL. [En línea] 15 de octubre de 2014. [Citado el: 13 de noviembre de 2014.] http://mappinggis.com/2014/07/mongodb-y-gis/.

6. **GENBETA_dev: Desarrollo y Software.** GENBETA:dev_Desarrollo y Software. [En línea] 27 de agosto de 2013. [Citado el: 17 de noviembre de 2014.] http://www.genbetadev.com/productos/bases-de-datos/mongodb.

7. **Leavitt, Neal.** *Will NoSQL Databases Live Up to Their Promise.* Cambridge : s.n., 2011.

8. **QuantumMode.** [En línea] 27 de abril de 2014. [Citado el: 1 de junio de 2015.] http://quantummode.com/las-bases-de-datos-nosql-ii-el-modelo-de-datos/.

9. WebMining Consultores. Webmining. [En línea] 8 de 06 de 2012. http://www.webmining.cl/2012/06/como-elegir-una-base-de-datos-para-su-empresa/.

10. Amazon.com, Inc. aws.Amazon. [En línea] 3 de 03 de 2012. http://aws.amazon.com/es/dynamodb/.

11. Amazon DynamoDB. Amazon. [En línea] 25 de junio de 2014. [Citado el: 7 de noviembre de 2014.] http://aws.amazon.com/es/dynamodb/.

12. Benítez, Juan;. redis-TecnoPedia.net. [En línea] 3 de marzo de 2012. http://www.tecnopedia.net/software/5-base-de-datos-nosql-mejorar-performance-aplicaciones/.

13. Borja. Antweb. ***Sistema de base de datos Redis: Todo lo que debes saber.*** [En línea] 18 de 10 de 2012. http://www.antweb.es/servidores/redis-todo-lo-que-debes-saber.

14. Sg. Sg. [En línea] 7 de 11 de 2014. http://sg.com.mx/revista/42/nosql-la-evolucion-las-bases-datos.

15. Josek. josek. [En línea] 2 de 03 de 2010. http://www.josek.net/2010/03/cassandra-nsql/.

16. Cassandra Apache. cassandra.apache. [En línea] 5 de julio de 2010. [Citado el: 12 de noviembre de 2014.] http://cassandra.apache.org.

17. NoSQL Cassandra. NoSQL Cassandra. [En línea] 17 de diciembre de 2013. [Citado el: 27 de octubre de 2014.] http://www.nosql.es/blog/nosql/cassandra.html.

18. Shroff, Rishit; Fong, Zelaine. Engineering Blog. [En línea] 5 de junio de 2014. [Citado el: 7 de junio de 2015.] https://code.facebook.com/posts/321111638043166/hydrabase-the-evolution-of-hbase-facebook/.

19. YAHOO! Developer Network. [En línea] 7 de junio de 2013. [Citado el: 7 de junio de 2015.] https://developer.yahoo.com/blogs/ydn/apache-hbase-yahoo-multi-tenancy-helm-again-203911418.html.

20. IEEE. IEEExplore. [En línea] 24 de 08 de 2011. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6182030&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6182030.

21. Romero Sánchez, Daniel;. Dbigcloud. ***Dbigcloud.*** [En línea] 27 de 7 de 2013. http://www.dbigcloud.com/bigdata/69-introduccion-a-couchdb.html.

22. Blog, Eduarea's. La Empresa y los Medios de Comunicación Social. [En línea] 24 de 01 de 2013. https://eduarea.wordpress.com/2013/01/11/que-es-google-bigtable/.

23. Sitio Ofical del proyecto MongoDB. [En línea] 6 de febrero de 2012. [Citado el: 12 de noviembre de 2014.] www.mongodb.org.

24. Vázquez Ortíz, MsC. Yudisney y Sotolongo León, MsC. Anthony Rafael. ***Mirada a bases de datos NoSQL de código abierto orientadas a documentos.*** La Habana : s.n., 2014.

25. Godel, Pablo. ***Symfony2 y MongoDB.*** Buenos Aires : s.n., 2013.

26. DB-Engines. [En línea] [Citado el: 6 de junio de 2015.] http://db-engines.com/en/ranking.

27. Silva Hernández. ***Tesis del Departamento de Computación, Centro de Investigación y de estudios avanzados del IPN.*** Ciudad México : s.n., 2003.

28. Dpto de Computación de México. Dpto de Computación de México. [En línea] 28 de octubre de 2013. [Citado el: 21 de noviembre de 2014.] http://www.cs.cinvestav.mx/tesisgraduados/2003/resumenIlianaAma.html.

29. CNet.com. [En línea] 11 de enero de 2014. [Citado el: 9 de marzo de 2015.] http://descargar.cnet.com/iReport-Designer/3000-2383_4-75748748.html.

30. Aurelio Rodríguez Durán. ***IMPACTO DEL GENERADOR DINÁMICO DE REPORTES EN LOS SISTEMAS DE GESTIÓN DE INFORMACIÓN.*** La Habana : s.n., 2011.

31. ***GDRv2.0-UCIENCIA.*** **Brito, Julio César; Hernández, Yasmany ; Lezcano, Miguel; Rodríguez, Aurelio; Abreu, Aldis Joan; Hernández, Beatriz; Yanes, Vania Elena; López, Marleysi; Mendoza, Alberto; Cruz, Viviana; Nicot, Asdrubal Antonio; Alfonso, Javier; Álvarez, Anaibis. La Habana : s.n., 2014.** GDR2.0-UCIENCIA: I Conferencia Científica Internacional.

32. W3C.es. [En línea] [Citado el: 10 de febrero de 2015.] http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb.

33. Navarro Marset, Rafael. ***REST vs Web Services.*** 2011.

34. Meza Martínez, Jorge Iván. ***Servicios web.*** Colombia : s.n., 2014.

35. Dirk, R. Framework Design: A Role Modeling Approach. [En línea] 21 de 11 de 2000. http://www.riehle.org/computer.

36. Eguiluz. ***La guía de Symfony2.*** España : s.n., 2010.

37. Eguiluz, Javier. ***Desarrollo Web Ágil con Symfony2.*** España : s.n., 2011.

38. Ardissone, Juan. ***Desarrollo de Aplicaciones Web con Symfony.*** España : s.n., 2012.

39. Pressman, Roger S.;. ***Ingeniería del Software. Un Enfoque práctico.*** Quinta Edición. 2010.

40. La tecla del escape. La tecla del escape. [En línea] 3 de octubre de 2010. [Citado el: 18 de noviembre de 2014.] http://latecladeescape.com/index.php?start=45.

41. Larman, Craig. ***UML y Patrones.*** México : s.n., 1999.

42. Visual Paradigm. Visual Paradigm for UML. [En línea] 27 de octubre de 2011. [Citado el: 20 de noviembre de 2014.] http://www.visual-paradigm.com/product/vpuml/.

43. «TIOBE Programming Community Index». 2009. [En línea] 06- de 05 de -2009.

44. Flanagan, David y Ferguson, Paula. ***JavaScript: The Definitive Guide.*** 2012. ISBN 0-596-00048-0.

45. Rivas Santos, Víctor Manuel;. Curso de Javascript | Versión 4.0. [En línea] 21 de enero de 2014. [Citado el: 15 de diciembre de 2014.] http://geneura.ugr.es/~victor/cursillos/javascript/js_intro.html.

46. Luke y Thomson, Laura Welling. ***Desarrollo Web con PHP y MySQL.*** Madrid : Anaya Multimedia, 2005.

47. php.net. [En línea] 23 de enero de 2012. [Citado el: 11 de noviembre de 2014.] http://php.net/manual/es/refs.database.php.

48. Fornieles, MZ. ***El futuro de Internet:'XM. Los secretos xml****.* 2011.

49. MongoDB Documentation Project. ***MongoDB Documentation.*** 2013.

50. Hernández, Yasmany. ***Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.*** La Habana : s.n., 2009.

51. NetBeans, Sitio oficial del. Sitio oficial del NetBeans. [En línea] 25 de noviembre de 2014. https://netbeans.org/index_es.html.

52. Brito Rodríguez, Julio César. ***Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.*** s.l. : La Habana, 2012.

53. Pressman, Roger. ***Ingeniería del Software. Un enfoque práctico.*** *5ta. Edición.* s.l. : McGraw-Hill, 2002. ISBN 84-481-3214-9.

54. KDE Documentation. KDE Documentation. [En línea] 14 de diciembre de 2014. [Citado el: marzo de 1 de 2015.] https://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html.

55. Harvey M. Deitel, Paul J. Deitel. ***Cómo programar en Java.*** **s.l**. : Pearson Educación, 2004. 9702605180,9789702605188.

56. Falgueras, Benet Campderrich. ***Ingeniería de software Biblioteca multimedia: Informática.*** s.l. : UOC, 2003. 8484297934.

57. EDWARD FRASER, DUSTIN. CAPITULO-4-IMPLEMENTACION-Y-PRUEBA-DEL-SISTEMA. [En línea] Scribd. [En línea] , 5 de Abril de 2012. http://es.scribd.com/doc. /74894700/44/.

58. MAG. ***Estándares de codificación.*** Costa Rica : s.n., 2014.

59. MSDN.Microsoft. [En línea] [Citado el: 5 de mayo de 2015.] https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

60. Fernando Alonso Amo, Loïc A. Martínez Normand, Francisco Javier Segovia Pérez. Introducción a la ingeniería del software, Delta Publicaciones, 2005. [En línea]

61. Pressman, Roger. 2005.

62. Unal. [En línea] 20 de 11 de 2014. http://www.virtual.unal.edu.co/cursos/IDEA/2007219/lecciones/cap_4/sub8.html.

63. Libros Web - España. [En línea] 14 de 01 de 2010. http://librosweb.es/symfony_2_x/capitulo_8.html.

64. PHP Ya. [En línea] 11 de 09 de 2014. http://www.phpya.com.ar/temarios/descripcion.php?cod=23.

65. Genbetadev. [En línea] 29 de 11 de 2014. http://www.genbetadev.com/herramientas/netbeans-1.

66. Agustín Martínez. Sitio Oficial del Prof. Agustín Martínez. [En línea] 11 de diciembre de 2014. [Citado el: 16 de enero de 2015.] http://profmartinez.es.tl/Metodo-Cientifico.htm.

67. TECNOLOGÍA, DICCIONARIO DE INFORMÁTICA Y. DICCIONARIO DE INFORMÁTICA Y TECNOLOGÍA. [En línea] ALEGSA.com, 25 de octubre de 2009. [Citado el: 17 de enero de 2015.] http://www.alegsa.com.ar/Dic/sgbd.php.

68. ***GDRv2.0-UCIENCIA: I Conferencia Científica Internacional.*** UCI. La Habana : s.n., 2014.

69. Carlos Ruiz,Degli Pino. TuBasedeDatosLibre.org. [En línea] 29 de septiembre de 2014. [Citado el: 21 de enero de 2015.] http://www.euskadinnova.net/es/enpresa-digitala/agenda/mongodb-bbdd-nosql-popular-mercado/7623.aspx.

70. Jaspersoft Community. Jaspersoft Community. [En línea] 4 de enero de 2015. [Citado el: 20 de enero de 2015.] https://community.jaspersoft.com/wiki/jaspersoft-mongodb-query-language.

71. ***IBM WebSphere Application Server V8.5.5 Liberty extended profile.*** WAS855_MongoDB. 2013.

72. ***Rational Software Architect.*** Corp, IBM. 2006.

73. Software Selección. Software Selección: Soluciones Empresariales. [En línea] 2015. [Citado el: 21 de enero de 2015.] http://www.softwareseleccion.com/eclipse+birt-p-1201.

74. Zaninotto, Fabien y François. ***Symfony La Guía Definitiva.*** s.l. : POTENCIER, 2009.

75. Herrera, Cristhian. Adictos al trabajo. [En línea] 29 de abril de 2005. [Citado el: 15 de enero de 2015.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iReport.

76. Navarro Rodríguez, Ana Niuska. ***Asistente de Reportes para el módulo Diseñador de Reportes versión 2.0.*** La Habana : s.n., 2011.

77. Ginacho.Github.io. Ginacho.Github.io. [En línea] 12 de 11 de 2014. [Citado el: 2015 de enero de 10.] http://gitnacho.github.io/symfony-docs-es/bundles/DoctrineMongoDBBundle/index.html.

78. Symfony2, Sitio oficial de. Sitio oficial de Symfony2.es. [En línea] 16 de noviembre de 2013. [Citado el: 14 de noviembre de 2014.] http://sf2-es.net16.net/bundles/DoctrineMongoDBBundle/index.html.

79. Rodríguez Durán. ***IMPACTO DEL GENERADOR DINÁMICO DE REPORTES EN LOS.*** La Habana : s.n., 2011.

80. EDWARD FRASER, DUSTIN. CAPITULO-4-IMPLEMENTACION-Y-PRUEBA-DEL-SISTEMA. [En línea] 5 de Abril de 2012. http://es.scribd.com/doc. /74894700/44.

81. weboPedia. RDBMS - relational database management system. [En línea] 25 de 10 de 2014. http://www.webopedia.com/TERM/R/RDBMS.html.

Glosario de términos

API: *Application Programming Interface*, es un conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.

GDR: Generador Dinámico de Reportes.

SDR: Servidor Dinámico de Reportes.

JavaBeans: es un modelo de componentes para la construcción de aplicaciones en Java.

JDBC: *Java Database Connectivity*, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java.

Map-Reduce: es una funcionalidad de consulta muy potente; se utiliza para desarrollar operaciones de agregación complejas y agrupaciones.

MongoClient: es el *driver* de PHP5 para MongoDB; se utiliza para administrar las conexiones entre ambos.

PDO: *PHP Data Objects,* es una extensión que provee una capa de abstracción de acceso a datos para PHP5 para SGBD relacionales.

Reporte: Documento que contiene la información reflejando el resultado de una investigación.

RPC: *Remote Procedure Call*, es una técnica utilizada para la comunicación entre procesos en una o más computadoras conectadas a una red. Es utilizada por el GDR para realizar las solicitudes al servidor.