Universidad de las Ciencias Informáticas
Facultad 6

Título: "Diseñador gráfico de consultas MDX para el Generador Dinámico de Reportes v2.0"

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

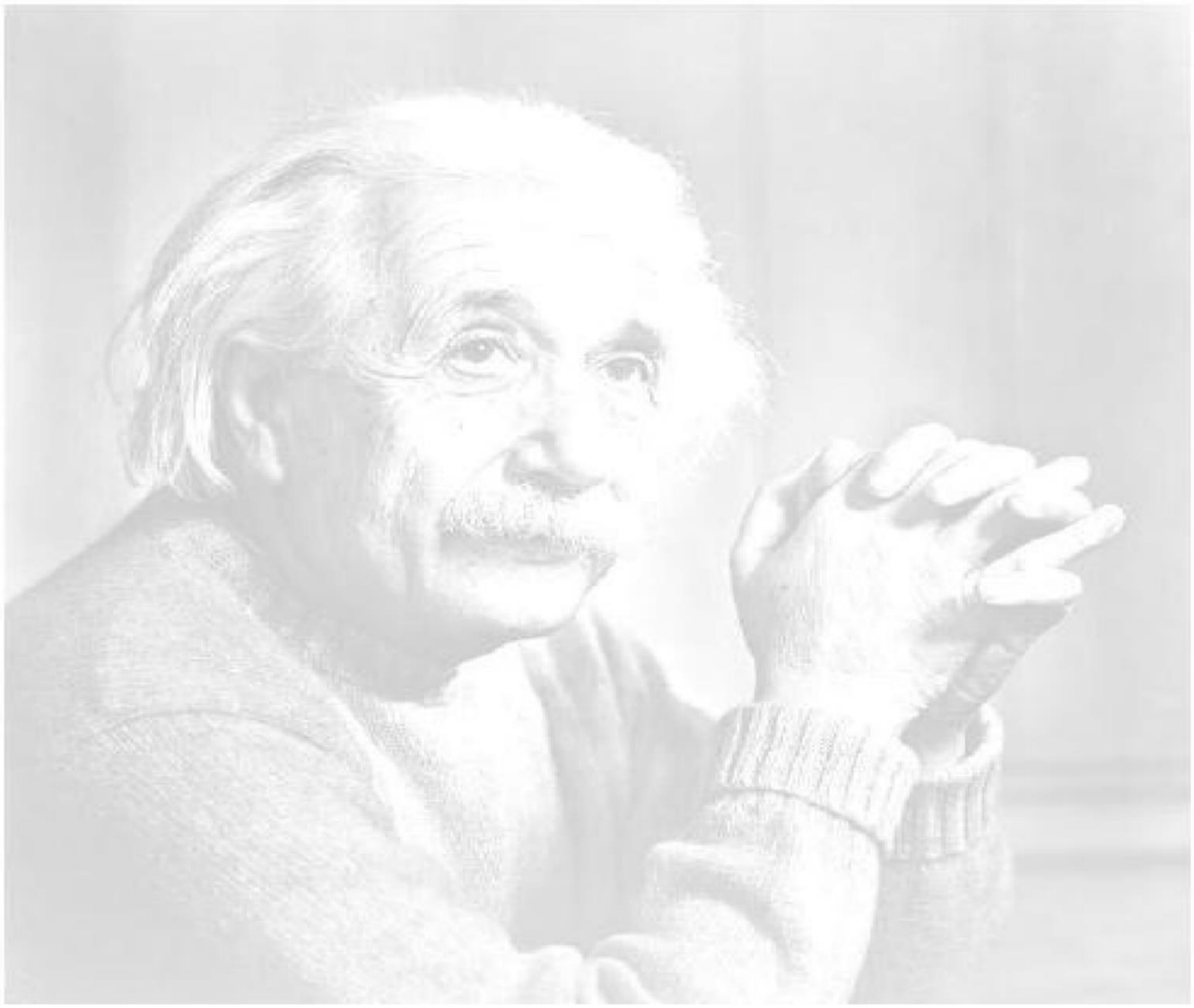
Autores: Daynelis Brito Morales

Carlos Fortún Manzano

Tutores: Ing. Glennis Tamayo Morales

Ing. Yoander Iñiguez Bermúdez

La Habana, 2016



" Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad" .

Albert Einstein

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 firmamos la presente a los ____ días del mes de _____ del año _____.

Daynelis Brito Morales

_____

Firma de Autor

Carlos Fortún Manzano

_____

Firma de Autor

Ing. Glennis Tamayo Morales

_____

Firma de Tutor

Ing. Yoander Iñiguez Bermúdez

_____

Firma de Tutor

Datos de contacto

Tutores:

Ing. Glennis Tamayo Morales.

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

E-mail: gtamayo@uci.cu

Ing. Yoander Iñiguez Bermúdez.

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

E-mail: yiniguez@uci.cu

Agradecimientos: Dayne

Agradecerle a dios por darme la posibilidad de estar hoy cumpliendo mi mayor sueño.

A mi papá **Pedro** por siempre estar ahí para mí, por ser tu niña consentida, por tus buenos consejos, por guiarme, por ser mi ejemplo a seguir, por ayudarme a confiar más en mí y levantarme cuando me he caído, por ser mi apoyo en los momentos más difíciles, no sabes lo que soñé con este día para poder decirte: "Papi la primera Ingeniera de la familia".

A mi mamá **Maritza** por darme la vida, por quererme tanto, por apoyarme, por confiar en mí, por soportarme y ayudarme con todo, por ser una amiga para mí. Eres la mejor mamá del mundo y el universo.

Gracias a los dos por siempre poder contar con ustedes, por su sacrificio constante, hoy podemos decir que valió la pena, por ser mi motor a seguir, por demostrarme que si se puede, por ser los mejores padres que la vida me pudo dar, todo lo que hago es para que se sientan orgullosos de su niña.

Agradecerle mi abuelita **Lidia** por cuidarme desde el cielo, gracias por haber sido la mejor abuela del mundo y una segunda mamá para mí. Yo sé que hoy estas feliz por todo lo que he logrado.

Gracias a mi hermano **Daykel** que aunque todavía nos fajamos y a veces ni nos soportamos te quiero con la vida. Hoy quiero que estas orgulloso de tu princesita de biscuit.

A mi tía **Estelita** por ser la mejor tía del mundo y siempre hacerme reír hasta en los peores momentos.

A mis primos, a mi abuelito **Pedro** en fin a toda mi familia ya sea materna o paterna, gracias porque sin ser la familia perfecta para mí son la mejor.

Quiero agradecerles a mis dos grandes amigas de la universidad:

Wendy: Por ser mi amiga desde el minuto cero que llegué aquí, por compartir tantos momentos juntas, por ser mi confidente, por cuidarme, por ser mi apoyo en los momentos más difíciles, por siempre darme los mejores consejos, aunque nunca te perdonaré que te hayas ido, hoy este triunfo es por las dos.

Agradecimientos

Kielam: gracias por apoyarme cuando más lo necesité, por estar ahí siempre para mí, por ser mi familia aquí. Por hacer mejores y divertidos mis días, por soportarme, aguantar mis repeticiones incansables y mis peleaderas, por aconsejarme. Nunca voy a olvidar lo que hiciste por mí, me diste la mano cuando ni cuarto tenía eso no lo hace nadie. Gracias por hacerme parte de tu maravillosa familia y hacerme sentir como si tú casa fuera la mía.

Gracias a las dos por ser unas amigas incondicionales, mis compañeras de fiesta, de cuarto y de estudio, en verdad hoy las considero las hermanitas hembras que nunca tuve, saben que siempre estaré aquí para ustedes en las buenas y en las malas aunque la distancia nos llegara a separar.

Agradecerle a toda la familia de Kielam en especial a su mamá **Daysi** por acogerme en su casa y hacerme sentir como si fuera la mía, además de tratarme como una hija más.

A mis tutores **Glennis** y **Yoander** por haber sido más que eso, por guiarme, aconsejarme, ayudarme y hasta por pelearme cuando era necesario, sin ustedes no sé qué hubiese sido de mí. Gracias a los dos por su esmerada dedicación.

Como no darle las gracias al mejor dúo de tesis del mundo, no pude tener otro mejor que tú, **Carlos** gracias por tenerme paciencia, por confiar en mí, por dar lo mejor de ti, por siempre estar dispuesto y siempre responsable, en fin por convertirte en excelente amigo.

Agradecerle a una persona muy especial a mi hermano del alma **Elvis** gracias por siempre hacerme reír con tus cosas y estar siempre pendiente de mí, por ser un amigo incondicional.

Darles las gracias a las mejores compañeras y amigas de apartamento que puede tener en estos 5 años la **Tere**, **Ivonsiña**, **Yanarita** y **Yanitza** decirles que fueron para mí como una familia, gracias por su preocupación y por pasar tantos momentos especiales juntas.

A **Yanelis**, **Dunia**, **Yeline**, **Claudia** y a nuestro valor agregado **Dionne** gracias amigas por siempre poder contar con ustedes en todo momento.

A mi grupo de primer año en especial a **Victor** por haber sido una persona súper importante y especial en mi vida, gracias por siempre poder contar contigo en todo momento, sabes que nunca te olvidaré. A **Yamil**, **Juampi**, **Blanco**, **Barra**, al **calvo**, **Yosbel** y **Cusito** chicos espero siempre contar con su amistad en verdad son personas maravillosas, los quiero mucho y nunca los olvidaré.

Agradecimientos

A mi grupo actual y a los del 4 en especial a **Rony**, **Alejandro**, **Michel**, **Lionel**, **Elian**, **Randy** al **Migue**, los **Leo** y **Lijandy** chicos son lo mejor de esta escuela gracias por hacer mis días en la uci con sus chistes y jodederas mucho mejor, se han ganado todos un espacio muy importante a mi vida.

Como no agradecerles a mis amigos de fiestas y coffee **Manu**, **Joyce**, **Orli**, **Bermu**, al **Peke** y al piquete **la guara** gracias por compartir tantos momentos juntos, por cuidarme y hacer mis días aquí diferentes.

Al **Rojo**, **Yaisel** y **Dayron** por brindarme su amistad y siempre estar dispuestos a ayudarme con todo lo que necesité.

A **Karell** por siempre estar pendiente a la evolución de mi tesis, gracias por tu preocupación a pesar de estar lejos.

A mis amigas de antes de llegar a la universidad **Idelis**, **Adis** y **Yairis** gracias por mantener la amistad a pesar de la distancia, esto ha sido una prueba para ratificar que cuando las amistades son verdaderas son para toda la vida.

A todos mis compañeros de año que estuvieron conmigo en estos 5 años de carrera.

A mis amigos de Placetas por siempre preguntarme como me iba con la tesis gracias por su preocupación.

A todos los profes que de una forma u otra me ayudaron y confiar en mí.

A los especialistas del proyecto Componente del centro DATEC por su ayuda y apoyo en la realización de esta tesis.

Al tribunal por sus críticas constructivas y por el tiempo que incondicionalmente nos dedicaron.

En fin a todos mis amigos y amigas que conocí en estos 5 maravillosos años de todas las facultades que son muchísimos y que no me alcanzaría estas 80 páginas para mencionarlos a todos.

Agradecimientos: Carlos

Agradezco a todas las personas que de una forma u otra hicieron posible la culminación de esta tesis.

Dedicatoria: Dayne

A mis **padres** por su amor, preocupación, por siempre confiar y estar pendientes de mí, en fin por haber sido mi motor a seguir.

A mi **hermano** porque esta escuela era su sueño, por eso este triunfo lo quiero compartir con él.

Dedicatoria: Carlos

A toda mi familia que se preocupan siempre por mí y me ayudan en todo lo posible, hasta en las cosas más insignificantes.

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se trabaja en el desarrollo de varios proyectos destinados a la informatización de los principales procesos de la sociedad cubana y de otros países del mundo. Formando parte de este conjunto de proyectos se encuentra el Generador Dinámico de Reportes v2.0 (GDR) que surge como estrategia trazada por el Centro de Tecnologías de Gestión de Datos (DATEC). En este proyecto se desarrolla una herramienta con este mismo nombre que permite, partiendo de información persistente en algún origen de datos, generar reportes de forma dinámica apoyando así la toma de decisiones. Actualmente las funcionalidades que posee el Editor de consultas MDX con que cuenta GDR v2.0 no permite a los usuarios que carecen de conocimientos sobre este lenguaje realizar consultas. El presente trabajo de diploma tiene como objetivo desarrollar un Diseñador gráfico de consultas MDX para GDR v2.0 que facilite esta tarea. Con el estudio realizado sobre los diseñadores de consultas existentes en Cuba y en el mundo se definió la estructura básica y principales funcionalidades del diseñador gráfico a realizar. Se seleccionó la arquitectura Modelo Vista Controlador como arquitectura del sistema y OpenUp como metodología de software, así como las distintas herramientas y tecnologías que fueron empleadas en el desarrollo de la solución. Se definió una estrategia de prueba quedando plasmados los niveles de prueba de unidad, aceptación e integración, así como los tipos de prueba funcionales y aceptación, y el método de prueba escogido fue caja negra con la técnica partición equivalente, todo con el objetivo de encontrar y corregir posibles errores.

Palabras Claves

Consultas, diseñador, MDX, multidimensionales, reportes.

Abstract

The University of Informatics Sciences (UCI) is working on the development of several projects aimed at computerization of main processes of Cuban society. Being part of this group of projects is the Dynamic Generator of Reports v2.0 (GDR) that arises as strategy traced by the Center of Technologies of Data Administration (DATEC). In this project a tool was developed with the same name that it allows, leaving of persistent information in some origin of data, to generate reports in a dynamic way to support decision making. At the moment the functionalities that possesses the MDX Query editor with which counts GDR v2.0 don't allow to the users that lack knowledge on this language a simple edition of MDX querys. The present diploma work has as main objective to develop a Graphic designer of MDX querys for GDR v2.0 that facilitates this task. With the study carried out on the existent query designers in Cuba and in the world was defined the basic structure and main functionalities of a graphic query designer. Model View Controller architecture was defined as system architecture and OpenUp as software methodology to guide the development process, as well as the various tools and technologies used in the development of the solution. Was defined a test strategy being choosen the test levels of unit , acceptance and integration, as well as the test types functional and acceptance, and the selected test method was black box with the equivalent partition technique, everything with the objective of to find and correct possible errors.

Keywords:

Queries, designer, MDX, multidimensional, reports.

Índice

Índice de figuras

Índice de tablas

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) han evolucionado a una velocidad increíble en las últimas décadas, convirtiéndose en uno de los eslabones más importantes para el desarrollo empresarial en el mundo. El uso de estas tecnologías en las empresas, es fundamental para lograr una mayor productividad, ya que mediante su empleo es posible recopilar información y llevar a cabo su tratamiento y análisis. Dicha información es almacenada tanto en bases de datos relacionales, orientadas a objetos o multidimensionales como en archivos de múltiples formatos. Una de las tendencias más utilizadas por las empresas que hacen uso de bases de datos multidimensionales, es el Procesamiento Analítico en Línea (OLAP), el cual permite procesar información que luego será utilizada para apoyar al usuario en el proceso de toma de decisiones (Lanzillotta, 2012).

Cuba no ha quedado exenta de estas transformaciones por lo que ha apostado por la informatización de sus principales sectores con el fin de contribuir al desarrollo del país. Entre las instituciones encargadas de llevar a cabo dichas transformaciones se encuentra la Universidad de las Ciencias Informáticas (UCI), en la cual se desarrolla el proyecto Generador Dinámico de Reportes (GDR) perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC).

En este proyecto se desarrolla una aplicación web, con el mismo nombre, que provee una forma transparente al usuario para realizar consultas a bases de datos y obtener información de ella en forma de reporte. Hoy día es cada vez mayor la necesidad de generar reportes para las empresas e instituciones con el objetivo de ayudar a la toma de decisiones, medir el trabajo que se está realizando y consultar los datos existentes en las bases de datos, permitiendo la formulación de reportes en diferentes formatos y modelos. GDR v2.0 es uno de los productos insignes de DATEC, empleado como parte de disímiles soluciones, pues permite la generación dinámica de reportes a partir del diseño de consultas SQL o MDX, sobre datos de bases de datos PostgreSQL, MySQL y SQLite (gespro, 2015).

El realizar consultas MDX desde GDR v2.0 tiene un alto impacto, ya que simplifica el proceso de generación de reportes multidimensionales realizados sobre almacenes de datos, lo que anteriormente, clientes como la Oficina Nacional de Estadísticas e Información, la Aduana General de la República de Cuba y la Contraloría General de la República de Cuba, que manejan enormes volúmenes de datos,

requerían el uso del Pentaho Bi-Server[1]. Esta situación no solo implicaba el consumo de tiempo extra sino que requería la instalación y configuración de una aplicación externa al sistema. Además, era indispensable el dominio preciso de dicha herramienta para realizar estas funciones. Debido a esto en el 2015 fue desarrollado un Editor de consultas MDX el cual permite la realización de estas sobre cubos OLAP.

Esta solución previamente desarrollada, aun cuando fue integrada satisfactoriamente a GDR v2.0, no cuenta con un diseñador gráfico por lo que la edición de consultas se realiza a nivel de código, implicando que los usuarios requieran de conocimientos adecuados de MDX para el diseño de consultas.

Dada la situación problemática anteriormente planteada, se define como **problema de la investigación:** Los usuarios que utilizan el Editor de consultas MDX de GDR v2.0 requieren de conocimientos avanzados de este lenguaje para el diseño de las mismas. Para dar cumplimiento al problema planteado se define como **objeto de estudio:** diseñadores de consultas, enmarcado en el **campo de acción:** diseñadores gráficos de consultas MDX para GDR v2.0.

El presente trabajo tiene como **objetivo general:** Desarrollar un diseñador gráfico de consultas MDX para GDR v2.0 que permita la generación de reportes sobre bases de datos multidimensionales.

Para garantizar el cumplimiento del objetivo general, el mismo se ha desglosado en los siguientes **objetivos específicos**:

- ✓ Identificar los elementos teóricos necesarios para el desarrollo de la investigación.
- ✓ Realizar el análisis y diseño del Diseñador gráfico de consultas MDX para GDR v2.0.
- ✓ Realizar la implementación del Diseñador gráfico de consultas MDX para GDR v2.0.
- ✓ Validar el Diseñador gráfico de consultas MDX para GDR v2.0.

Para lograr un correcto cumplimiento de los objetivos específicos y obtener los resultados esperados se plantean las siguientes **tareas de investigación:**

1. Análisis de los fundamentos teóricos sobre los diseñadores de las consultas MDX existentes en el mundo y en Cuba.
2. Caracterización de la metodología, herramientas y tecnologías a utilizar en el desarrollo de la solución.

[1] Herramienta open source, provee una alternativa de soluciones de inteligencia de negocio.

3. Identificación de los requisitos de software para el desarrollo de la solución.
4. Identificación de los principales elementos del diseño para definir la arquitectura de la solución.
5. Implementación de las funcionalidades identificadas para el Diseñador gráfico de consultas MDX.
6. Integración de la solución al módulo Diseñador de consultas de GDR v2.0.
7. Definición de una estrategia para aplicar las pruebas de software al Diseñador gráfico de consultas MDX para GDR v2.0.
8. Ejecución de las pruebas definidas para detectar y corregir los errores encontrados en el Diseñador gráfico de consultas MDX para GDR v2.0.

Métodos Científicos de la Investigación

Métodos Teóricos

- ✓ **Análisis y síntesis:** Está integrado por el desarrollo del análisis y la síntesis, mediante el cual se descompone un objeto, fenómeno o proceso en los principales elementos que lo integran para analizar, valorar y conocer sus particularidades, y simultáneamente a través de la síntesis. El mismo se utilizó para el estudio y análisis de diferentes fuentes bibliográficas con el objetivo de obtener un amplio conocimiento acerca del lenguaje MDX, sobre el análisis multidimensional de los datos y la estructura y funciones principales de un diseñador de consultas.
- ✓ **Modelación:** Es justamente el método mediante el cual se crean abstracciones con vistas a explicar la realidad. La modelación es el método que opera en forma práctica o teórica con un objeto, no en forma directa, sino utilizando cierto sistema intermedio, auxiliar, natural o artificial. Este método se utilizó para modelar los diagramas, la arquitectura y los distintos procesos que se realizan en la construcción del Diseñador gráfico de consultas MDX para GDR v2.0.

Métodos Empíricos:

- ✓ **Observación:** Con este método es posible distinguir directamente los hechos de la realidad objetiva. Permite conocer el proceso delimitado como objeto de estudio, lo cual contribuyó a tener un registro visual más detallado de lo que se quiere y hace falta hacer; y cómo hay que hacerlo. Mediante este método se observó el módulo Diseñador de Consultas el cual contiene el Editor de consultas MDX anteriormente desarrollado en él se pudieron detectar algunas deficiencias a ser corregidas además de obtener una guía para el desarrollo de una nueva solución.
- ✓ **Entrevista:** Este método es una técnica de recopilación de información mediante una conversación profesional, con el que se adquiere información acerca de lo que se investiga. Esta

puede estar estructurada o no mediante un cuestionario previamente elaborado donde la información que se obtiene resulta fácil de procesar. Mediante este método se pudieron definir parte de los requisitos funcionales del sistema, en una entrevista con los clientes (Ver anexo 1)

El presente trabajo de diploma está estructurado de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía y anexos.

Capítulo 1. Fundamentación Teórica del Diseñador gráfico de consultas MDX para GDR v2.0: este capítulo comprende el estudio de los conceptos asociados al problema planteado, además de realizar un análisis de los principales diseñadores de consultas existentes, haciendo énfasis en los diseñadores gráficos de consulta MDX. Se define la metodología de software, así como las distintas herramientas y tecnologías que serán empleadas en el desarrollo de la solución.

Capítulo 2. Análisis y Diseño del Diseñador gráfico de consultas MDX para GDR v2.0: en este capítulo para un mejor entendimiento del dominio se realizó el modelo de dominio en el cual se relacionan los conceptos fundamentales de la investigación. Se definieron los requisitos del sistema los cuales se clasifican en funcionales y no funcionales. Además, se definió como arquitectura del sistema la arquitectura Modelo Vista Controlador (MVC).

Capítulo 3. Implementación y Prueba del Diseñador gráfico de consultas MDX para GDR v2.0: este capítulo está enfocado a la implementación de la aplicación para dar solución a los requisitos especificados. Se elaboran los diagramas de componentes que forman parte del modelo de implementación. Se definen el estándar de codificación a utilizar en la implementación de la solución y se realiza el diagrama de despliegue. Además se realizan pruebas de software con el objetivo de descubrir y corregir errores.

Capítulo 1: Fundamentación Teórica del Diseñador gráfico de consultas MDX para el GDR v2.0

En el presente capítulo se comprende el estudio de los conceptos asociados al problema planteado, además de realizar un análisis de los principales diseñadores de consultas existentes, haciendo énfasis en los diseñadores gráficos de consultas MDX. Se define la metodología de software, así como las distintas herramientas y tecnologías que serán empleadas en el desarrollo de la solución.

1.1 Conceptos asociados a la investigación

1.1.1 OLAP

Procesamiento Analítico en Línea, (OLAP según sus siglas en inglés *Online Analitical Processing)* define una tecnología que se basa en el análisis multidimensional de los datos, permitiéndole al usuario tener una visión más rápida e interactiva de los mismos (Lanzillotta, 2012). Este análisis suele implicar, generalmente, la lectura de grandes cantidades de datos para llegar a extraer algún tipo de información útil. El acceso a los datos suele ser de sólo lectura. La acción más común es la consulta, con muy pocas inserciones, actualizaciones o eliminaciones (Sinnexus, 2015). Para su funcionamiento, OLAP hace uso de estructuras multidimensionales**,** denominadas también Cubos OLAP para ofrecer la posibilidad de crear una base completa y de rápido acceso a la información.

1.1.2 Cubos OLAP

Los cubos OLAP son una tecnología que proporciona rápido acceso a los datos de un almacén de datos. Un cubo es un conjunto de datos que normalmente se construye a partir de un subconjunto de un almacén de datos y se organiza y resume en una estructura multidimensional definida por un conjunto de dimensiones y medidas.

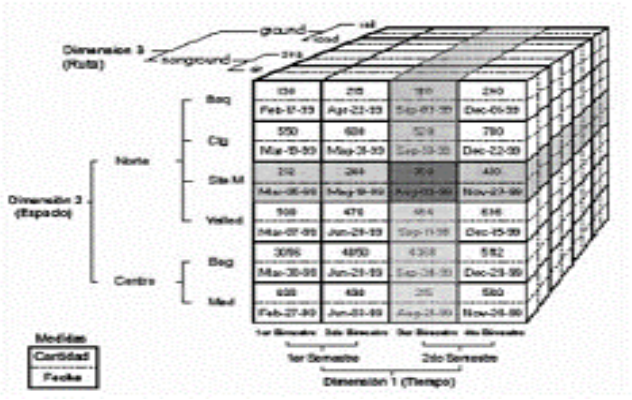


Fig. 1 Cubo OLAP (Tomado de (Sinnexus, 2015))

Dimensiones: Las dimensiones son un atributo[2] estructural de los cubos y sirven como un mecanismo de selección de datos. Están organizadas en jerarquías[3] de categorías y niveles[4] que describen los datos de las tablas de hechos.

Medidas: Dentro del modelo de datos multidimensional las medidas o atributos numéricos describen un cierto proceso del mundo real, el cual va a ser objeto de un análisis (GUEVARA , y otros, 2007).

1.1.3 Lenguaje MDX

MDX (*Multi-Dimensional Expressions*) es un acrónimo de *Multidimensional Query Expression*. Este lenguaje fue creado en 1997 por Microsoft. Es a través del cual se puede explotar la información que reside en los motores OLAP. La principal diferencia del mundo OLAP respecto al mundo relacional radica en que las estructuras dimensionales están jerarquizadas y se representan en forma de árbol por lo que existen relaciones entre los diferentes miembros de las dimensiones. Estas estructuras jerarquizadas son especialmente útiles para poder visualizar los datos de forma comparada a nivel temporal, pero de las dimensiones temporales y sus particularidades. A continuación, se muestra un ejemplo de la sintaxis de una consulta referente al lenguaje y para un mejor entendimiento del mismo se establece una analogía con SQL:

[2]Atributo: provee información adicional acerca de los datos.
[3]Jerarquías: conjunto de miembros de una dimensión, es decir una forma de organizar los datos en diferentes niveles de agregación.
[4]Niveles: elementos de una jerarquía de dimensiones, se definen en una dimensión para especificar el contenido y la estructura de la jerarquía de dimensiones.

WITH <expresión de fórmula> **SELECT** < expresión de eje> on columns, < expresión de eje> on rows **FROM** < expresión de cubo> **WHERE** <expresión de filtrado>

Se tiene la estructura de la consulta idéntica al SQL con las cláusulas **SELECT... FROM... WHERE** pero en MDX se especifica en la cláusula **SELECT** los datos que se necesitan visualizar en cada eje (Columnas, Filas); en el **FROM** se indica el cubo del que se extrae la información y en el **WHERE** (opcional) es usado para filtrar una pequeña sección de datos de los cuales se quieren sacar los resultados. Además tiene la cláusula opcional **WITH** la cual sirve para crear cálculos o conjuntos que se usarán en las consultas sucesivas a esta cláusula. Las **{ }** son para permitir listas de elementos en las selecciones y los **[ ]** encapsulan elementos de las dimensiones y niveles. La información de este lenguaje es devuelta en forma de tabla (Biosca, 2009).

Ejemplos de funciones y operadores MDX

Las funcionalidades que lo distinguen como lenguaje son los accesos a los elementos utilizando estructura de árbol. Para un determinado nivel de una dimensión tenemos comandos como:

- ✓ **CurrentMember**: permite acceder al miembro actual.
- ✓ **prevMember**: permite acceder al miembro anterior de la dimensión.

Existen diferentes funciones que permiten realizar cálculos y complementar las consultas por ejemplo:

- ✓ **CrossJoin** (conjunto_a, conjunto_b): obtener el producto cartesiano de dos conjuntos de datos.

En este lenguaje también están implementadas muchas funciones matemáticas y estadísticas que permiten enriquecer los análisis, tales como: **AVG, COUNT, ORDER, SUM, MIN y MAX.**
Además está implementado el operador **NON EMPTY** mediante el cual los datos definidos en los ejes se generan primero y luego se retiran las tuplas que contienen todas sus celdas vacías.

1.2 Análisis de soluciones existentes

El Generador Dinámico de Reportes v2.0 es una herramienta web que permite a sus clientes consultar los gestores de bases de datos de sus organizaciones y generar reportes con la información que estos manejan de forma rápida y con una amplia gama de opciones que facilitan el diseño de los reportes por los usuarios. Uno de los elementos más importantes en la generación de los reportes son las consultas asociadas a estos. Para la creación de estas GDR v2.0 posee un Diseñador de consultas que permite crear, editar y ejecutar consultas del tipo especificado por el usuario, sin necesidad de conocer el gestor de base de datos ni el lenguaje de consulta. Este Diseñador cuenta con dos herramientas básicas: un

diseñador gráfico, representado en la vista de Diseño y un editor de consultas, representado en la vista Código. El diseñador permite la construcción de las consultas de forma gráfica, donde se muestran los campos, tablas y sus relaciones. Por su parte el editor muestra el código perteneciente a la consulta. Estas dos herramientas están estrechamente relacionadas teniendo en cuenta que las modificaciones que se hagan en una de ellas se actualizan automáticamente en la otra.

Actualmente este Diseñador de consultas que posee GDR v2.0 cuenta con editores de consulta para los lenguajes SQL y MDX, sin embargo en el caso de los diseñadores gráficos solo se ha desarrollado uno para el lenguaje SQL, es por esta razón que se decide desarrollar un Diseñador gráfico de consultas MDX que al igual que el de SQL, deberá relacionarse con el editor de consultas para MDX que actualmente tiene GDR v2.0.

Para una mayor comprensión y entendimiento con el objetivo de conocer las características que posee un diseñador gráfico de consultas y teniendo en cuenta además, la estrecha relación que debe existir entre este y el editor de consultas MDX existente en GDR v2.0, se expone, a continuación, el estudio realizado sobre el Editor de consultas MDX de GDR v2.0, el Diseñador gráfico de consultas SQL de GDR v2.0 y el Diseñador gráfico de consultas MDX de Analysis Services (PowerPivot).

Editor de consultas MDX de GDR v2.0

El editor de consultas MDX para GDR v2.0 posee soporte OLAP y permite realizar consultas a los modelos creados a partir de los orígenes de datos multidimensionales. GDR v2.0 brinda al editor los modelos que contienen los cubos OLAP ya creados y mostrándose en forma jerárquica con las dimensiones y medidas en el explorador de modelos del Diseñador de consultas. A partir de estos es posible la creación de consultas en el lenguaje MDX posibilitando la ejecución de estas y mostrando los resultados obtenidos en un panel definido para ello. Estas consultas pueden ser salvadas y posteriormente mostradas en el explorador de modelos. Este editor desarrollado en el 2015, no cuenta con un diseñador gráfico, por lo que la edición de consultas se realiza a nivel de código, implicando que los usuarios requieran de conocimientos adecuados de MDX para el diseño de consultas. Además, las consultas realizadas en el editor no permiten el autocompletamiento de código y la diferenciación de palabras reservadas del lenguaje, lo que no facilita la adopción de la herramienta y el trabajo con comodidad con la misma (Editor de consultas MDX para GDR v2.0, 2015).

Diseñador gráfico de consultas SQL de GDR v2.0

El Diseñador gráfico de consultas SQL de GDR v2.0 fue desarrollado para diseñar las consultas que

posteriormente serán empleadas para el diseño de los reportes. Desde el explorador de modelos se selecciona el modelo y las respectivas tablas con las que se desea trabajar y estas son arrastradas hasta la vista de Diseño donde quedan representadas gráficamente permitiendo seleccionar los campos y definir las relaciones entre las tablas. La ejecución de esta consulta diseñada devuelve un resultado que se muestra en el panel con dicho nombre.

Sus principales funcionalidades son:

- ✓ **Diseñar consulta visualmente:** mediante la opción de arrastrar y soltar, permite colocar libremente dentro del área de diseño las tablas y/o vistas que conformarán la consulta deseada. Para seleccionar los campos que se quieren consultar, basta con marcarlos dentro de la entidad a la que pertenecen.
- ✓ **Modificar consulta visual:** carga una consulta previamente diseñada, la cual puede ser modificada.
- ✓ **Incluir funciones de agregación:** adiciona funciones de agregación a campos de la consulta (suma, cantidad, máximo, mínimo y promedio).
- ✓ **Los operadores de comparación que soporta son:** igual a, mayor que, menor que, mayor-igual a, menor-igual a y distinto a.
- ✓ **Establecer criterios de ordenamiento:** permite ordenar los datos de salida de la consulta seleccionando el o los campos por los cuales se desea ordenar.
- ✓ **Ejecutar consulta diseñada:** ejecuta la consulta diseñada mostrando los resultados de la misma.
- ✓ **Limitar la consulta:** especifica la cantidad de datos que se obtendrán en la consulta diseñada.

Diseñador gráfico de consultas MDX de *Analysis Services* (PowerPivot)

El diseñador gráfico de consultas de expresiones multidimensionales (MDX) de *Analysis Services* proporciona una interfaz gráfica de usuario para ayudarle a crear consultas en este lenguaje hacia un origen de datos de *Microsoft SQL Server Analysis Services*. Este diseñador gráfico tiene dos modos: modo de diseño y modo de consulta. Cada modo proporciona un panel de metadatos desde el que se pueden arrastrar miembros de los cubos seleccionados para crear una consulta MDX que recupera los datos que desee usar. En la siguiente figura se indican los nombres de los paneles del modo de diseño:

Fig. 2 Diseñador de consultas MDX en modo diseño.

En la siguiente tabla se muestran los paneles de este modo:

Tabla. 1 Paneles del modo diseño.

Panel	Función
Botón Selección del cubo (…)	Muestra el cubo seleccionado actualmente.
Panel Metadatos	Muestra una lista jerárquica de medidas, indicadores clave de rendimiento (KPI) y dimensiones definidas en el cubo seleccionado.
Panel Miembros calculados	Muestra los miembros calculados definidos actualmente que se encuentran disponibles para utilizarse en la consulta.
Panel Filtro	Se usa a fin de elegir dimensiones y jerarquías relacionadas para filtrar datos en el origen y limitar datos devueltos.
Panel Datos	Muestra los encabezados de columna del conjunto de resultados mientras arrastra elementos de los paneles Metadatos y Miembros calculados. Actualiza automáticamente el conjunto de resultados si el botón **Ejecución automática** está seleccionado.

En este modo de diseño se pueden arrastrar dimensiones, medidas y KPI desde el panel Metadatos, y miembros calculados desde el panel Miembros calculados al panel Datos. En el panel Filtro, puede seleccionar dimensiones y jerarquías relacionadas, y establecer expresiones de filtro para limitar los datos disponibles en la consulta. Si el botón de alternancia **Ejecución automática** de la barra de herramientas

está seleccionado, el diseñador de consultas ejecuta la consulta cada vez que coloque un objeto de metadatos en el panel Datos. También puede ejecutar la consulta manualmente mediante el botón **Ejecutar** de la barra de herramientas.

Diseñador gráfico de consultas MDX en modo de consulta

Para cambiar el diseñador gráfico de consultas al modo Consulta, se selecciona el botón del Modo de consulta de la barra de herramientas.

En la siguiente figura se indican los nombres de los paneles del modo de consulta:

Fig. 3 Diseñador de consultas MDX en modo consulta.

En la siguiente tabla se muestran los paneles de este modo:

Tabla. 2 Paneles del modo consulta.

Panel	Función
Botón Selección de cubo	Muestra el cubo seleccionado actualmente.
Panel Metadatos/Funciones/Plantillas	Muestra una lista jerárquica de medidas, KPI y dimensiones definidas en el cubo seleccionado.
Panel de consulta	Muestra el texto de consulta.
Panel Resultado	Muestra los resultados de la ejecución de la consulta.

El panel Metadatos muestra pestañas de Metadatos, Funciones y Plantillas. Desde la pestaña Metadatos, puede arrastrar dimensiones, jerarquías, KPI y medidas al panel Consulta MDX. Desde la pestaña

Funciones, puede arrastrar funciones hasta el panel Consulta MDX. Desde la pestaña Plantillas, puede agregar plantillas al panel Consulta MDX. Cuando ejecute la consulta, en el panel Resultado se mostrarán los resultados de la consulta (Microsoft, 2014).

Conclusiones parciales

Después de haber realizado un análisis de las soluciones existentes se decide desarrollar un Diseñador gráfico de consultas MDX estructurado por:

- ✓ Un panel de selección del cubo con el que se trabajará posteriormente.
- ✓ Un panel de metadatos donde se muestren según el cubo seleccionado las medidas y dimensiones de este en forma de árbol donde se define su jerarquía.
- ✓ Un panel de diagrama en el que se muestren las representaciones gráficas de las tablas seleccionadas en el panel de metadatos, se seleccionen los campos y se definan las relaciones entre tablas.
- ✓ Un panel de consulta donde se muestre el código de la consulta MDX representada mediante el panel de diagrama. Se usará este panel para escribir o actualizar una consulta y estos cambios realizados se actualizarán en el panel de diagrama.
- ✓ Un panel de resultados donde se muestre la tabla con los resultados de la consulta.

Además, se definen como elementos fundamentales los siguientes:

- ✓ Permite mostrar y ocultar celdas vacías en el panel de resultados.
- ✓ Ejecutar una consulta desde el diseño realizado.
- ✓ Mostrar resultados de la consulta ejecutada.
- ✓ Mostrar según el cubo seleccionado los metadatos.
- ✓ Actualizar automáticamente la vista Código a partir del diseño realizado en la vista de Diseño.
- ✓ Actualizar automáticamente la vista de Diseño a partir de los cambios realizados en el código de la consulta de la vista Código.
- ✓ Diseñar gráficamente la consulta.

1.3 Herramientas, tecnologías y metodología

Desarrollar un *software* implica que sea necesario definir las tecnologías, herramientas y metodología de desarrollo que se van a emplear para lograr el objetivo planteado (Cendejas, 2011). Para la selección de

las herramientas, tecnologías y metodología de desarrollo a emplear en el desarrollo de la solución se tuvieron en cuenta las definidas en el marco del proyecto GDR v2.0.

OpenUp

Proceso Unificado de Desarrollo (OpenUp) es una metodología ágil dirigida por casos de uso, centrada en la arquitectura, iterativo e incremental (Barranco, 2002). El objetivo de OpenUp es ayudar al equipo de desarrollo, a lo largo de todo el ciclo de vida de las iteraciones, para que sea capaz de añadir valor de negocio a los clientes, de una forma predecible, con la entrega de un *software* operativo y funcional al final de cada iteración. El ciclo de vida del proyecto provee a los clientes de: una visión del proyecto, transparencia y los medios para que controlen la financiación, el riesgo, el ámbito, el valor de retorno esperado. Todo proyecto en OpenUp consta de cuatro fases: inicio, elaboración, construcción y transición.

- ✓ **Fase de inicio:** En esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los casos de uso críticos, una estimación inicial del coste y un boceto de la planificación.
- ✓ **Fase de elaboración:** En esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. En esta fase se especifican en detalle el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo. Al final de la fase se debe tener una definición clara y precisa de los casos de uso, los actores, la arquitectura del sistema y un prototipo ejecutable de la misma.
- ✓ **Fase de construcción:** Todos los componentes y funcionalidades del sistema que falten por implementar son realizados, probados e integrados en esta fase. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado.
- ✓ **Fase de transición:** Esta fase corresponde a la introducción del producto en la comunidad de usuarios, cuando el producto está lo suficientemente maduro. La fase de la transición consta de las subfases de pruebas de versiones beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema. En función de la respuesta obtenida por los usuarios puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más.

OpenUp se enfoca en las siguientes disciplinas: Requerimientos, Arquitectura, Desarrollo, Prueba, Administración de Configuración y Cambio y Administración de Proyecto.

- ✓ **Requerimientos (Requirements):** Explica cómo elicitar, analizar, validar y manejar los requerimientos para el sistema a desarrollar. Es importante entender la definición y alcance del problema que se intenta resolver, identificar los interesados y definir el problema a resolver. Durante el ciclo de vida se administran los cambios de los requerimientos.
- ✓ **Arquitectura (Architecture):** El propósito es lograr evolucionar a una arquitectura robusta para el sistema. Explica cómo crear una arquitectura a partir de requerimientos arquitectónicamente (estructuralmente) importantes. Es en la Disciplina de desarrollo donde se construye la arquitectura.
- ✓ **Desarrollo (Development):** Explica cómo diseñar e implementar una solución técnica que esté conforme a la arquitectura y sustente los requerimientos
- ✓ **Prueba (Test):** Explica cómo proveer retroalimentación sobre la madurez del sistema diseñando, implementando, ejecutando y evaluando pruebas. Es iterativa e incremental. Aplica la estrategia de "prueba lo antes posible y con la mayor frecuencia posible" para replegar los riesgos tan pronto como sea posible dentro del ciclo de vida del proyecto.
- ✓ **Administración de configuración y cambio (Configuration and change management**): Explica cómo controlar los cambios de los artefactos, asegurando una evolución sincronizada del conjunto de productos (*Work Products*) que compone un sistema software. Esta disciplina se expande durante todo el ciclo de vida.
- ✓ **Administración del Proyecto (Project Management)**: Explica cómo entrenar, ayudar y apoyar al equipo, ayudándolo a manejar los riesgos y obstáculos que se encuentren al construir el software (Gimson, y otros, 2012).

Características fundamentales de OpenUp:

- ✓ Colaboración para unificar intereses y compartir conocimientos.
- ✓ Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- ✓ Permite detectar errores tempranos a través de un ciclo iterativo (Barranco, 2002).

ExtJS 3.4

ExtJS es una biblioteca de JavaScript desarrollada por Sencha, para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. ExtJS es el marco más amplio de JavaScript para crear aplicaciones web multiplataforma con múltiples funciones. Esta aprovecha las funciones de HTML5 en los navegadores modernos manteniendo la compatibilidad y funcionalidades para navegadores antiguos. Cuenta con cientos de widgets[5] de interfaz de usuario de alto rendimiento que están meticulosamente diseñados para adaptarse a las necesidades de los más simples, así como las aplicaciones web más complejas. El marco incluye un paquete de datos robusto que puede consumir datos desde cualquier fuente de datos (Sencha ExtJS, 2015).

Symfony 2.0.18

Symfony 2 está desarrollado completamente con PHP. Está diseñado para optimizar el desarrollo de aplicaciones Web, entre sus características presenta que separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web, es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft, se puede ejecutar tanto en plataformas UNIX, como en plataformas Windows. Emplea el tradicional patrón de diseño MVC para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista (Macías, y otros, 2016).

Apache Tomcat 7.0

Apache Tomcat es una herramienta código abierto[6] utilizada como servidor web. Se desarrolla en un entorno abierto y participativo, además es liberado bajo la licencia Apache versión 2. Apache Tomcat versión 7.0 implementa las especificaciones de la comunidad de procesos de Java sobre Servlet 3.0 y JavaServer Pages 2.2, e incluye muchas características adicionales que lo convierten en una plataforma útil para el desarrollo y despliegue de aplicaciones y servicios web (Apache Tomcat, 2015).

Apache 2.0

[5] Widgets: es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets.*

[6] Código abierto (en idioma inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

Apache es el servidor web más utilizado en sistemas Linux. Los servidores web son utilizados para servir páginas web solicitadas por equipos cliente. Los clientes normalmente solicitan y visualizan páginas web usando navegadores como Firefox, Opera, o Mozilla. Los servidores web Apache a menudo se usan en combinación con el motor de bases de datos MySQL, el lenguaje de scripting[7] PHP, y otros lenguajes de scripting populares como Python y Perl (Ubuntu, 2014).

PostgreSQL 9.4

PostgreSQL es un sistema gestor de bases de datos objeto-relacional, distribuidos bajo licencia *Berkeley Software Distribution* (BSD) y con su código fuente disponible libremente. Es el sistema gestor de bases datos de código abierto más potente del mercado. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

Sus características técnicas lo hacen una de las bases de datos más potentes y robustas del mercado. Estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las que más se han tenido en cuenta durante su desarrollo. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (Martinez , 2013).

PHP 5.0

PHP (acrónimo recursivo de PHP: del inglés *Hypertext Preprocessor*) es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que posee. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP. Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales. El desarrollo de PHP está centrado en la programación de scripts del lado del servidor (php, 2015).

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es

[7] Un lenguaje scripting o lenguaje de guión es un tipo de lenguaje de programación que es generalmente interpretado.

necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java (Eguiluz, 2008).

NetBeans 8.0

El IDE NetBeans es un entorno galardonado de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear aplicaciones web, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript, Ajax, Groovy y C / C + +. El proyecto de NetBeans está apoyado por una comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación, así como una gran cantidad de plugins de terceros. NetBeans funciona en sistemas operativos compatibles con la máquina virtual de Java (Windows XP, Windows Vista, Windows7, Ubuntu, Solaris, Mac OS X 10.5 o superior) (netbeans, 2012).

Visual Paradigm 8.0

Visual Paradigm para UML es una herramienta para desarrollo de aplicaciones utilizando modelado UML. Es ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de Sistemas que tienen como objetivo la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Visual Paradigm también ofrece:

- ✓ Navegación intuitiva entre la escritura del código y su visualización.
- ✓ Potente generador de informes en formato PDF/HTML.
- ✓ Ambiente visualmente superior de modelado.
- ✓ Sincronización de código fuente en tiempo real.

Esta herramienta CASE es multiplataforma y su licencia se encuentra disponible de forma gratuita solo con fines académicos y no comerciales. Además brinda un completo conjunto de herramientas de equipos de desarrollo de software necesario para la captura de requisitos, software de planificación y el modelado de datos (Software, 2013).

UML 2.0

El Lenguaje Unificado de Modelado establece un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y

símbolos significan. Incluye aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (UML, 2012).

Conclusiones del capítulo

El estudio de los diferentes diseñadores de consulta encontrados permitió definir las principales funcionalidades que debe poseer la solución propuesta tomando como base el Diseñador gráfico de consultas SQL para GDR v2.0. Se seleccionó como metodología de desarrollo de software OpenUp; Symfony 2.0.18 y ExtJS 3.4 como marcos de trabajo; UML 2.0 como lenguaje de modelado; Visual Paradigm 8.0 para UML como herramienta de modelado; como lenguaje de programación JavaScript y PHP 5; PostgreSQL 9.4 como sistema gestor de bases de datos; NetBeans 8.0 como entorno integrado de desarrollo; Apache Tomcat 7 y Apache2 como servidor de aplicaciones.

Capítulo 2: Análisis y Diseño del Diseñador gráfico de consultas MDX para GDR v2.0

En el presente capítulo, para un mejor entendimiento del dominio, se realizó el modelo de dominio en el cual se relacionan los conceptos fundamentales de la investigación. Se definieron los requisitos del sistema los cuales se clasifican en funcionales y no funcionales. Además, se definió como arquitectura del sistema la arquitectura Modelo Vista Controlador (MVC).

2.1 Modelo de Dominio

El modelo de dominio, es el punto de partida para lograr un mejor diseño del sistema a desarrollar y una comprensión más clara del problema a resolver. En él se muestran las clases conceptuales más significativas asociadas al dominio del problema. Está considerado como parte del modelo de negocio, pues se centra en una parte del mismo relacionada con el ambiente del proyecto (Sosa, 2013).

En la siguiente figura se muestra el modelo de dominio de la solución del sistema:

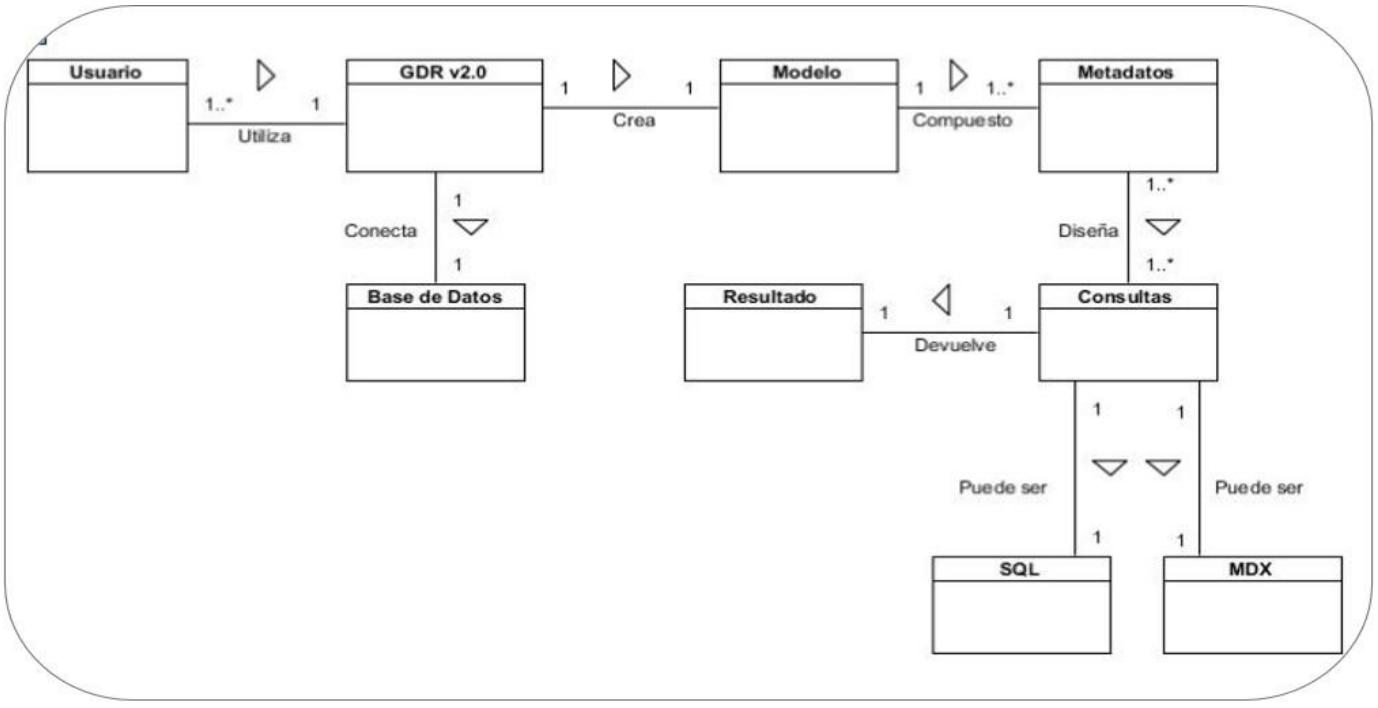


Fig. 4 Modelo de Dominio.

Definición de las clases del modelo de dominio

1. **Usuario:** Persona que interactúa con el sistema GDR v2.0.

2. **GDR v2.0:** Representa al sistema Generador Dinámico de Reportes v2.0.
3. **Base de Datos:** Contiene los datos que permiten conectar al GDR v2.0 con el origen de datos.
4. **Modelo:** Describe la estructura de la Base de datos.
5. **Metadatos:** Datos contenidos en una Base de datos.
6. **Consultas:** Se utiliza para acceder a datos específicos contenidos en un origen de datos.
7. **SQL:** Lenguaje de programación diseñado para almacenar, manipular y recuperar datos almacenados en bases de datos relacionales.
8. **MDX:** Lenguaje utilizado para consultar datos multidimensionales o crear expresiones MDX en un cubo.
9. **Resultado:** Información que devuelve la consulta.

2.2 Requisitos del Sistema

La especificación de requisitos es el proceso de desarrollo de software donde se analizan las necesidades exactas de los usuarios y del sistema. Un correcto levantamiento de requisitos permite posteriormente la construcción de un software de calidad (Pressman, 2010). Los mismos se clasifican en requisitos funcionales y no funcionales.

Requisitos Funcionales

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Además, se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los requisitos funcionales definen las funciones que el sistema será capaz de realizar (Sommerville, 2011). A continuación, se describen los requisitos funcionales que debe cumplir el Diseñador gráfico de consultas MDX para GDR v2.0:

RF1: Ejecutar consulta MDX desde la vista de Diseño.
Descripción: Permite ejecutar una consulta diseñada desde la vista de Diseño.

RF2: Mostrar resultados de la consulta MDX.
Descripción: Permite visualizar una tabla con los resultados de la consulta MDX una vez ejecutada correctamente.

RF3: Modificar código de la consulta MDX.
Descripción: Permite editar el código de la consulta MDX con que se está trabajando.

RF4: Ocultar celdas vacías en el panel de Resultados.
Descripción: Al generarse la tabla resultados si esta contiene campos vacíos permite ocultar los mismos.

RF5: Exportar en formato xls el resultado de la consulta.
Descripción: Permite almacenar el resultado de la consulta en un fichero xls en la carpeta seleccionada por el usuario.

RF6: Diseñar gráficamente la consulta MDX.
Descripción: Muestra una representación gráfica del cubo una vez arrastrada alguna dimensión o medida desde el Explorador de modelos.

RF7: Actualizar automáticamente la vista Código a partir del diseño realizado en la vista de Diseño.
Descripción: Permitirá que cualquier cambio realizado en la vista de Diseño se actualice automáticamente en la vista Código.

RF8: Actualizar automáticamente la vista de Diseño a partir de los cambios realizados en el código de la vista Código.
Descripción: Permitirá que cualquier cambio realizado en el código de la vista Código se actualice automáticamente en la vista de Diseño.

RF9: Salvar el diseño gráfico de la consulta.
Descripción: Permite almacenar el diseño gráfico realizado en la base de datos del sistema mostrándose al usuario en el Explorador de Modelos en la sección Consulta.

RF10: Abrir diseño gráfico de la consulta**.**
Descripción: Permite abrir el diseño gráfico una vez que este fue salvado en el sistema.

RF11: Autocompletar el código MDX de las consultas a partir de las palabras reservadas del lenguaje.
Descripción: Permite una escritura más sencilla de las consultas al brindar las palabras reservadas del lenguaje MDX según sean requeridas por el usuario.

Requisitos No Funcionales

Los requisitos no funcionales (NF) se refieren a funciones específicas que proporciona el sistema. Son restricciones de los servicios o funciones ofrecidas por el sistema (fiabilidad, tiempo de respuestas, capacidad de almacenamiento). Generalmente se aplican al sistema en su totalidad. Surgen de las

necesidades del usuario (restricciones de presupuesto, políticas de la organización, necesidad de interoperabilidad). En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Están vinculados a requisitos funcionales, es decir, una vez que se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser (Pressman, 2010).

A continuación, se describen los requisitos no funcionales que debe cumplir el Diseñador gráfico de consultas MDX para el GDR v2.0.

Requisitos de software

RNF 1 Requisitos de software para la PC cliente.

Para el uso del sistema la PC cliente debe contar con un navegador web Firefox versión 34 o una superior, recomendando el uso de la versión 44.0.

RNF 2 Servidor para instalar la aplicación.

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

- ✓ SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.
- ✓ Paquetes: apache2, php5, libapache2-mod-php5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-sybase, php5-xsl, php5-gd, php-apc.
- ✓ Usuario con privilegios de administración del SO.

RNF 3 Servidor para instalar la Base de Datos.

El servidor donde se instalará la base de datos del sistema debe cumplir con los siguientes requisitos de software (puede ser el mismo donde estará la aplicación):

- ✓ SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.
- ✓ PostgreSQL versión 8.3 o superior.
- ✓ PGAdmin III o algún administrador para 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.

Requisitos de hardware.

RNF 4 PC Cliente

La PC cliente debe cumplir con los siguientes requisitos de hardware:

- ✓ Ordenador Dual Core o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 1GB.

RNF 5 Servidor de aplicaciones

La PC donde se instalará el servidor de la aplicación debe cumplir con los siguientes requisitos de hardware:

- ✓ Ordenador Dual Core o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 2GB.
- ✓ Disco Duro con 80Gb de capacidad.

RNF 6 Servidor de Base de Datos

La PC donde se instalará la base de datos del sistema debe cumplir con los siguientes requisitos de hardware:

- ✓ Ordenador Dual Core o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 2GB.
- ✓ Disco Duro con 80Gb de capacidad.

Restricciones de diseño

RNF 7 Lenguaje y marco de trabajo «framework» para el desarrollo del sistema del lado del servidor.

El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.2 o superior. Como framework de desarrollo se usará Symfony el cual propone una arquitectura modular en tres capas: el modelo, la vista y el controlador.

RNF 8 Lenguaje y marco de trabajo «framework» para el desarrollo del sistema del lado del cliente.

Unas de las bibliotecas fundamentales en el desarrollo de la herramienta es ExtJS la cual es una librería en JavaScript que permite el diseño de interfaces visuales interactivas usando metodologías como AJAX y permite crear aplicaciones WEB con la apariencia de escritorio.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF 9 Documentación de usuarios

Se entregará a los clientes un manual de usuario que guía paso a paso las acciones a seguir para trabajar con el sistema. Este manual estará escrito en español.

Requisitos de Interfaz

RNF 10 Interfaz

La interfaz de la aplicación da continuidad a los patrones de diseño, mantiene los colores y la distribución de los contenidos, basándose en los estándares empleados en GDR v2.0.

Requisitos de Seguridad

RNF 11 Seguridad

Las funcionalidades desarrolladas siguieron los estándares de implementación definidos para mantener las funcionalidades seguras y para que se acoplen al módulo de seguridad del GDR v2.0 el cual es el encargado de gestionar los niveles de acceso por usuarios y roles del sistema; para lo cual se deben tener en cuenta las respectivas anotaciones de seguridad en las funcionalidades adicionadas al sistema.

2.3 Patrones de Casos de Uso

Como mejores prácticas para modelar casos de uso son utilizados los patrones de casos de uso, los cuales permiten reflejar los requisitos reales de un sistema con mayor facilidad (Cuesta, 2005). A continuación, se explica el patrón de caso de uso utilizado.

Extensión concreta: Consiste en dos casos de uso y una relación extendida entre ellos. Este patrón se aplica cuando un flujo puede extender el flujo de otro caso de uso así como ser realizado en sí mismo (Craig, 2003). El patrón extensión concreta es evidenciado en los casos de uso Autocompletar código MDX y Diseñar consulta ya que estos son una extensión del caso de uso Administrar consulta (caso base), le incorporan funcionalidades al caso base sin alterar ni modificar a este, no siendo indispensable para su completamiento.

2.4 Diagrama de Casos de Uso del sistema

Una vez aplicados los patrones de casos de uso se realizó el Diagrama de Casos de Uso del Sistema (DCUS). Este modelo captura todos los requisitos funcionales del sistema (Jacobson, y otros, 2000). Está

conformado por actores, casos de uso y las relaciones entre ellos. En la siguiente figura se muestra el diagrama de casos de uso del sistema:

Fig. 5 Diagrama de Casos de Uso del Sistema.

Descripción textual de los casos de uso y del actor del sistema

Los actores del sistema son aquellos individuos u otros sistemas externos que interactúan con el sistema (Svitla Systems, 2016). Para el desarrollo de este diagrama se ha identificado el siguiente actor:

Tabla. 3 Descripción del actor del sistema.

Actor	Descripción
Diseñador	Usuario encargado de interactuar con el módulo Diseñador de Consultas de GDR v2.0, con la responsabilidad de diseñar consultas, administrar consultas y autocompletar código MDX en caso que así lo desee.

Un caso de uso (CU) se utiliza para modelar y representar las funcionalidades o los servicios prestados por un sistema a los usuarios. Es decir son interacciones o diálogos entre la aplicación y los actores, incluyendo los mensajes intercambiados y las acciones realizadas (Svitla Systems, 2016). A continuación, se muestran una descripción de los casos de uso del sistema:

CU Diseñar consulta: Permite al diseñador diseñar gráficamente las consultas MDX, actualizar automáticamente la vista Código a partir del diseño realizado en la vista de Diseño o actualizar automáticamente la vista de Diseño a partir de los cambios realizados en el código de la vista Código. Además, salvar el diseño gráfico de la consulta en el sistema, al igual que abrir el diseño gráfico de la consulta una vez que fue salvado.

CU Administrar consulta: Permite al diseñador realizar las acciones: ejecutar una consulta MDX desde la vista de Diseño, mostrar resultados de la consulta, modificar código de la consulta, ocultar campos vacíos en el resultado y exportar el resultado de la consulta.

CU Autocompletar código MDX: Permite al diseñador una escritura más sencilla de las consultas al brindar las palabras reservadas del lenguaje MDX según sean requeridas.

En la tabla 4 se muestran los casos de uso identificados, así como los requisitos funcionales que cada uno de estos agrupan.

Tabla. 4 Relación entre los casos de uso y los requisitos funcionales.

CU	Referencia RF
Diseñar consulta	RF6, RF7, RF8, RF9, RF10
Administrar consulta	RF1, RF2, RF3, RF4, RF5
Autocompletar código MDX	RF11

Descripción del Caso de Uso "Diseñar consulta"

El comportamiento de un caso de uso se puede especificar describiendo un flujo de eventos de forma textual, lo suficientemente claro que lo entienda fácilmente el cliente (Sabio, y otros, 2009). A continuación, se muestra en la tabla 5 la descripción del caso de uso **"Diseñar consulta":**

Tabla. 5 Especificación del Caso de Uso: "Diseñar consulta".

Objetivo	Diseñar gráficamente una consulta en el sistema.
Actores	Diseñador: (Inicia) Diseña gráficamente la consulta MDX, salva el diseño gráfico de la consulta y abre el diseño gráfico de la consulta una vez salvada en el sistema.
Resumen	El caso de uso se inicia cuando el usuario realiza alguna de las acciones tales como: diseñar gráficamente la consulta MDX, salvar el diseño gráfico, abrir diseño gráfico de la consulta, actualizar automáticamente la vista Código a partir del diseño realizado en la vista de Diseño o actualizar automáticamente la vista de Diseño a partir de los cambios realizados en el código de la vista Código. El caso de uso termina con una o varias de estas acciones realizadas.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	✓ Es necesario que exista al menos un modelo OLAP. ✓ Es necesario que exista al menos un cubo en el modelo.

	✓ Para salvar el diseño gráfico y actualizar automáticamente la vista Código a partir del diseño realizado en la vista de Diseño es necesario que exista previamente un diseño en la vista de Diseño. ✓ Para abrir el diseño gráfico de una consulta es necesario que exista una consulta salvada en el modelo.
Referencias	RF6, RF7, RF8, RF9, RF10
Postcondiciones	Un diseño gráfico.

Flujo de eventos		
Flujo básico < Diseñar consulta >		
	Actor	**Sistema**
1.	Selecciona el módulo "Diseñador de consultas".	
		Habilita el diseño gráfico permitiendo en el módulo Diseñador de consultas: ✓ Diseñar gráficamente la consulta MDX, ver **Sección 1: "Diseñar gráficamente la consulta MDX".** ✓ Salvar el diseño gráfico de la consulta, ver **Sección 2: "Salvar el diseño gráfico de la consulta".** ✓ Actualizar automáticamente la vista Código a partir del diseño realizado en la vista de Diseño, ver **Sección 3: "Actualizar la vista Código a partir del diseño de la vista de Diseño".** ✓ Actualizar automáticamente la vista de Diseño a partir de los cambios realizados en el código de la vista Código ver **Sección 4: "Actualizar la vista de Diseño a partir de los cambios en el código de la vista Código".** ✓ Abrir diseño gráfico de la consulta **Sección 5: "Abrir diseño gráfico de la consulta".**
3.		Termina el caso de uso.
Sección 1: "Diseñar gráficamente la consulta MDX"		

Flujo básico: "Diseñar gráficamente la consulta MDX"		
	Actor	**Sistema**
1.	Selecciona un modelo **OLAP**.	
2.		Despliega toda la información del modelo.
3.	Selecciona el cubo con que desee trabajar.	
4.		Despliega las dimensiones y medidas del cubo seleccionado.
5.	Selecciona de este cubo cualquier medida o dimensión.	
6.	Arrastra la medida o la dimensión seleccionada para la vista de Diseño.	
7.		Muestra una tabla conformada por filas, columnas y filtro donde en este último aparecerán todos los datos del cubo tales como las dimensiones con sus respectivos niveles y las medidas.
8.	Selecciona de la tabla en la sección filtro la(s) dimensión(es) con los niveles con que desee trabajar y la(s) medida(s).	
9.	Selecciona el eje (columnas o filas) donde quiere que se visualicen las dimensiones o medidas en la tabla de resultados.	
10.		Muestra en el diseño en las secciones filas y columnas los campos seleccionados por el usuario.
Flujos alternos: "Arrastrar para la vista de Diseño otro metadato del mismo cubo".		
Nº Evento < Flujo alterno 1.1 >: "Arrastrar para la vista de Diseño otro metadato del mismo cubo".		
6.1	Arrastra para vista de Diseño otra dimensión o medida del mismo cubo.	
		Muestra una notificación: "El cubo seleccionado ya se encuentra en la vista de Diseño".
Flujos alternos: "Intentar hacer diseños simultáneos de cubos diferentes".		
Nº Evento < Flujo alterno 1.2 >: "Intentar hacer diseños simultáneos de cubos diferentes".		
6.1	Arrastra para vista de Diseño otra dimensión	

	o medida de otro cubo.	
6.2		Muestra una notificación: “Solo se puede trabajar con un cubo a la vez en la vista de Diseño”.
Sección 2: “Salvar el diseño gráfico de la consulta”		
Flujo básico: “Salvar el diseño gráfico de la consulta”		
	Actor	**Sistema**
1.	Selecciona el botón **“Salvar”** en el menú “Gestionar Consultas”.	
2.		Muestra una ventana indicando al usuario que introduzca el nombre del diseño.
3.	Introduce el nombre del diseño.	
4.	Selecciona el botón **“Aceptar”.**	
5.		Almacena el diseño en el sistema y lo muestra en el Explorador de Modelos en la sección Consultas.
Flujos alternos: “Cancelar salva de diseño gráfico”.		
Nº Evento < Flujo alterno 2.1 >: “Cancelar Salva”.		
4.1	En caso de que el usuario no desee salvar el diseño gráfico selecciona el botón **“Cancelar”.**	
4.2		El sistema cierra la ventana Salvar.
4.3		Muestra la vista Código o vista de Diseño según donde se encontraba trabajando en ese momento.
Sección 3: “Actualizar la vista Código a partir del diseño de la vista de Diseño”.		
Flujo básico: “Actualizar la vista Código a partir del diseño de la vista de Diseño”.		
	Actor	**Sistema**
1.	Realiza algún cambio en el diseño de la vista de Diseño ya sea remover o adicionar algún campo de las secciones columnas y filas.	
2.		Actualiza en la vista Código los cambios realizados en la vista de Diseño.
Sección 4: “Actualizar la vista de Diseño a partir de los cambios en el código de la vista Código”.		

Flujo básico: "Actualizar la vista de Diseño a partir de los cambios en el código de la vista Código".		
	Actor	**Sistema**
1.	Realiza algún cambio en el código de la consulta de la vista Código ya sea borrar o ingresar nuevos campos.	
2.		Actualiza en la vista de Diseño los cambios realizados en el código de la consulta en la vista Código.
Sección 5: "Abrir diseño gráfico de la consulta"		
Flujo básico: "Abrir diseño gráfico de la consulta"		
1.	Selecciona una consulta salvada en el sistema en el Explorador de Modelos en la sección Consultas.	
2.	Selecciona la opción Abrir en el menú "Gestionar Consultas" o da clic derecho sobre la consulta salvada y selecciona la opción **"Abrir"**.	
3.		Muestra el diseño gráfico de la consulta en la Vista de Diseño o el código de la consulta en la vista Código según donde se encontraba trabajando en ese momento.
Flujos alternos: "Abrir diseño gráfico existiendo otra ya en la vista de Diseño".		
Nº Evento < Flujo alterno 5.1 >: "Abrir diseño gráfico existiendo otra ya en la vista de Diseño".		
2.1	Abre el diseño gráfico de una consulta existiendo ya uno en la vista de Diseño.	
2.2		Muestra una notificación: El diseño ha sido modificado. ¿Que desea hacer?". Este contiene tres botones con las opciones.1-Guardar.2-Descartar cambios.3-Cancelar.
Flujos alternos: "Abrir diseño gráfico existiendo otra ya en la vista de Diseño".		
Nº Evento < Flujo alterno 5.1.1 >: "Seleccionar opción "Guardar" cuando abres el diseño gráfico existiendo otro en la vista de Diseño".		
2.2.2.	Selecciona el botón **"Guardar"** de la interfaz de la notificación.	

2.2.3		Salva el diseño gráfico de la consulta que existe en la vista de Diseño en el sistema.
Flujos alternos: "Abrir diseño gráfico existiendo otra ya en la vista de Diseño".		
Nº Evento < Flujo alterno 5.1.2 >: "Seleccionar opción "Descartar cambios" cuando abres el diseño gráfico existiendo otro en la vista de Diseño".		
2.2.2	Selecciona el botón **"Descartar cambios"** de la interfaz de la notificación.	
2.2.3		Elimina el diseño gráfico de la consulta existente en la vista de Diseño abriendo el nuevo diseño en esta vista.
Flujos alternos: "Abrir diseño gráfico existiendo otra ya en la vista de Diseño".		
Nº Evento < Flujo alterno 5.1.3 >: "Seleccionar opción "Cancelar" cuando abres el diseño gráfico existiendo otro en la vista de Diseño".		
2.2.2	Selecciona el botón **"Cancelar"** de la interfaz de la notificación.	
2.2.3		Cancela la opción de abrir diseño gráfico de la consulta manteniendo el diseño actual en la vista.

Para consultar la descripción de los restantes casos de uso debe remitirse al expediente de proyecto, específicamente a la plantilla 0114_ Especificaciones_de_casos_de_uso.doc

2.5 Patrones de arquitectura

Un patrón de arquitectura de *software* es un esquema genérico probado para solucionar un problema particular, el cual es recurrente dentro de un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades y relaciones (Rivera , y otros, 2010). Para el desarrollo del diseñador gráfico de consultas se utilizará el patrón arquitectónico Modelo-Vista–Controlador que es el utilizado en el desarrollo de GDR v2.0.

Modelo-Vista–Controlador (MVC): el patrón proporciona grandes ventajas como la organización de código, reutilización y flexibilidad. La programación es organizada pues separa los datos de la aplicación, la interfaz de usuario y la lógica de los datos en tres componentes diferentes, en el cual cada capa se especializa en una función específica.

- ✓ **Modelo:** representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.

- ✓ **Vista:** muestra el modelo en un formato adecuado para interactuar. Maneja la presentación visual de los datos representados por el modelo.
- ✓ **Controlador:** responde a eventos e invoca cambios en el modelo y probablemente en la vista.

Ventajas del MVC:

- ✓ Soporte para múltiples vistas pues no existe dependencia directa entre la vista y el modelo.
- ✓ Proporciona un mecanismo de configuración a componentes complejos mucho más tratable que el puramente basado en eventos.
- ✓ Mayor soporte a los cambios, pues los requisitos de interfaz tienden a cambiar más rápidamente que las reglas de negocio. Puesto que el modelo no depende de las vistas, la adición de nuevos tipos de vistas al sistema generalmente no afecta al modelo (Salazar, 2012).

2.6 Diagramas de clases de diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de software y de las interfaces de una aplicación. Además, muestra las definiciones de entidades de software más que conceptos del mundo real y contiene información acerca de las clases, asociaciones, atributos, interfaces, métodos y relaciones (Unad, 2006). Se realizó un diagrama de clases de diseño por cada caso de uso, en la siguiente figura se muestra el diagrama de clases del diseño perteneciente al caso de uso **"Diseñar consulta"**.

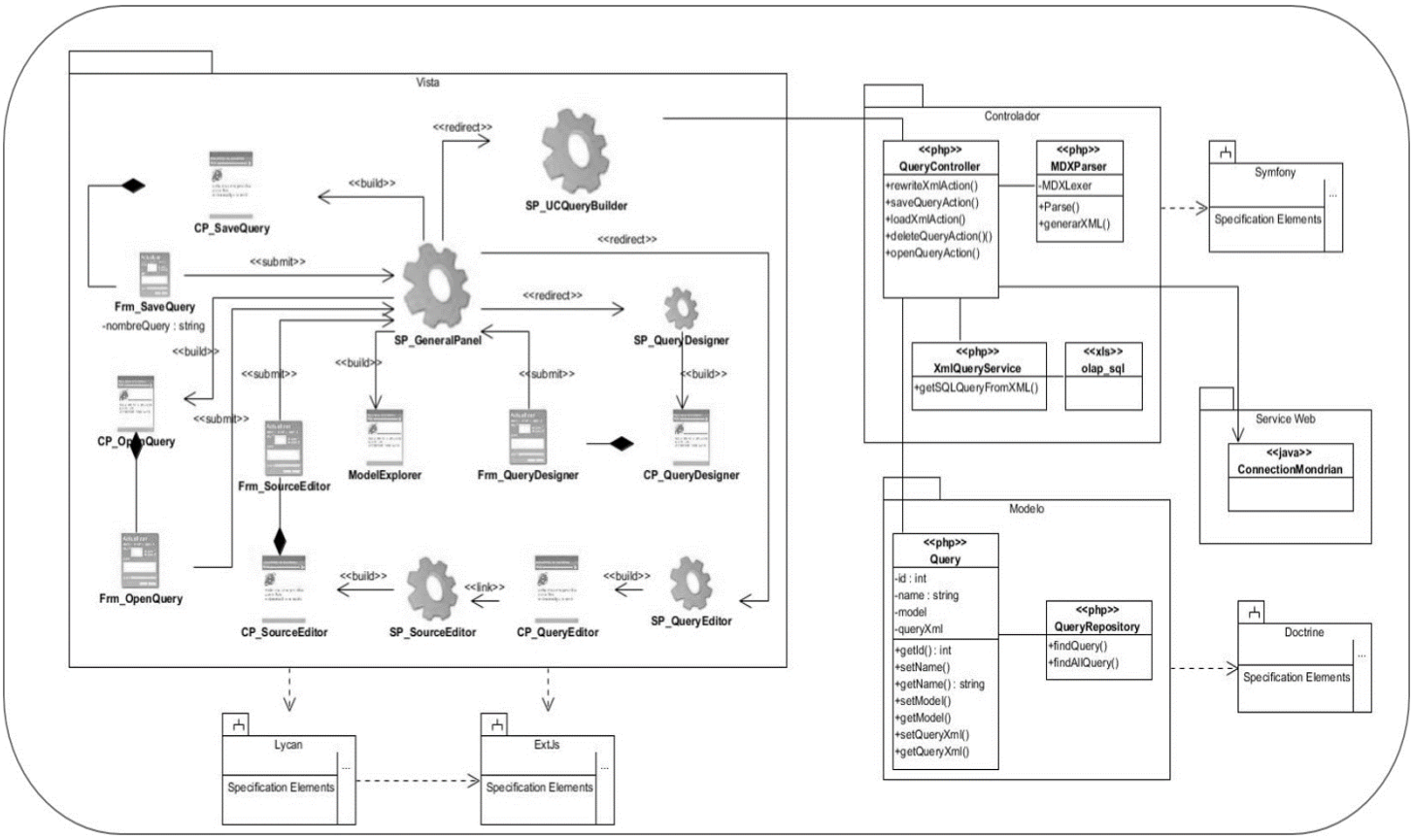


Fig. 6 Diagrama de clases del diseño "Diseñar consulta".

Para consultar la descripción de los restantes diagramas de clases del diseño debe remitirse al expediente de proyecto, específicamente a la plantilla 010215_Modelo_de_diseño.doc

Descripción de las clases del Diagrama de clases de diseño

En el diagrama de clases de diseño de la figura 6, se representaron las principales clases que se utilizaron para la realización del caso de uso "Diseñar consulta", estas quedaron representadas en una arquitectura MVC ya que GDR v2.0 utiliza el framework Symfony v2.0.18 el cual emplea este tradicional patrón de diseño. A continuación, se describen las clases del Diagrama de clases de diseño:

En el Modelo se encuentran las clases:

- ✓ La clase **Query** contiene la estructura con la que se guarda la consulta en la base de datos.
- ✓ La clase **QueryRepository** ayuda a buscar consultas guardadas en un modelo y representa de manera general la información y consultas realizadas por las clases.

En la Vista se encuentran las clases:

- ✓ La clase **UCQueryBuilder** gestiona las acciones del lado del cliente y de ser necesario hace llamadas a la clase controladora QueryController.
- ✓ La clase **GeneralPanel** es la encargada de construir la interfaz visual del módulo Diseñador de Consultas utilizando las clases ModelExplorer, QueryDesigner y QueryEditor.
- ✓ La clase **ModelExplorer** muestra el explorador de modelos donde se encuentran todos los metadatos en forma de árbol jerárquico una vez extraídos del origen de datos seleccionado.
- ✓ La clase **QueryDesigner** es la que permite que en la vista de Diseño se pueda diseñar gráficamente la consulta.
- ✓ La clase **QueryEditor** se encarga de construir en la vista Código el área de edición de consultas y el panel de Resultados.
- ✓ La clase **SourceEditor** está contenida dentro de la clase QueryEditor y permite escribir o modificar el código de la consulta.

En el Controlador se encuentran las clases:

- ✓ La clase **QueryController** es la encargada de responder las peticiones realizadas del lado del cliente a través de métodos tales como saveQueryAction(), deleteQueryAction(), loadQueryAction(), resultQueryAction() y openQueryAction().
- ✓ La clase **MDXParser** comprueba si la estructura de la consulta cumple con la estructura definida por la gramática, a este proceso se le denomina análisis sintáctico en el cual se examina una secuencia de tokens para determinar si el orden de esa secuencia es correcta de acuerdo a ciertas convenciones estructurales de las reglas de la definición sintáctica del lenguaje. La entrada del analizador sintáctico o parser es la secuencia de tokens generada por el analizador léxico (MDXLexer.php) y la salida es la indicación del cumplimiento de las reglas gramaticales que definen al lenguaje.
- ✓ La clase **XmlQueryService** transforma un XML a una consulta en el lenguaje del modelo especificado.
- ✓ La clase **olap_sql** es utilizada por el servicio XmlQueryService para transformar un XML a una consulta en el lenguaje MDX.

Además quedaron representados los subsistemas y el servicio web:

- ✓ El subsistema **Lycan** es una librería de ExtJS utilizado para diseñar componentes de este.
- ✓ El subsistema **ExtJS** se utiliza para atender las peticiones del lado del cliente.
- ✓ El subsistema **Symfony** se emplea para atender las peticiones del lado del servidor.
- ✓ El subsistema **Doctrine** es usado para el manejo de los datos y la comunicación con los distintos gestores de bases de datos.
- ✓ **ConnectionMondrian** es un Servicio web externo utilizado por la aplicación para parsear el código de la consulta.

2.6.1 Patrones de Diseño

Con el uso de patrones de diseño se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Permite formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño. *"Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema general de diseño en un contexto particular."* (Gamma, 2011). Para el diseño de la aplicación se emplearon distintos patrones de diseño clasificados en dos grupos, patrones GRASP y patrones GoF:

Patrones GRASP

Los patrones para Asignar Responsabilidades (GRASP del inglés *Responsability Assignment Patterns*) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño del objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. En el diseño de la aplicación se emplearon patrones de diseño GRASP tales como:

Creador: Soluciona el problema de ¿quién debería ser responsable de crear una nueva instancia? El patrón creador guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. En la clase GeneralPanel.js se evidencia la utilización de este patrón ya que esta es la encargada de la creación de las instancias de las clases QueryDesigner, ModelExplorer y QueryEditor.

Experto: El patrón experto en información soluciona el problema ¿de qué forma podemos saber qué responsabilidad delegar a cada objeto? Es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. En la clase QueryEditor se evidencia la

utilización de este patrón ya que esta clase delega toda la responsabilidad del trabajo con el código de las consultas a la clase SourceEditor la cual cuenta con la información y los métodos necesarios para cumplir esta responsabilidad.

Controlador: Resuelve el problema de ¿quién gestiona un evento del sistema? Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón se evidencia en la clase QueryController que es la encargada de gestionar las principales acciones que se solicitan en la interfaz de la aplicación.

Alta Cohesión: Soluciona el problema de ¿cómo mantener manejable la complejidad?, ya que asigna responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. Este patrón se evidencia en todas las clases de la interfaz ya que la clase GeneralPanel para no estar cargada de responsabilidades crea instancias de las clases ModelExplorer, encargada de los modelos, QueryDesigner, encargada del diseño de las consultas, y QueryEditor, encargada de la edición de las consultas. Esta última a su vez le delega todo el trabajo relacionado con el código a la clase SourceEditor. De esta manera cada una de estas clases se especializan en una determinada función y así se mantiene estable la complejidad de la aplicación.

Bajo Acoplamiento: Soluciona el problema de ¿cómo dar soporte a las bajas dependencias y al incremento de la reutilización? Plantea tener las clases lo menos ligadas entre sí, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las mismas (Grosso, 2011). Este patrón es utilizado en toda la interfaz de la aplicación ya que las clases ModelExplorer, QueryDesigner y QueryEditor sólo están relacionadas con la clase GeneralPanel, por lo que para comunicarse entre sí sólo lo pueden hacer a través de esta clase, evitándose así que haya mucha dependencia entre ellas y que un cambio en una afecte a todas las demás.

Patrones GOF

Otro conjunto de patrones de diseño bien conocidos son los patrones conocidos como grupo de los cuatro GOF (*Gang of Four*) son una serie de patrones que permiten ampliar el lenguaje, aprender nuevos estilos de diseño e introducir más notación UML. Estos patrones GoF se encuentran agrupados en tres categorías: de creación, de estructura y de comportamiento (Scielo, 2013).

De los diferentes patrones que ofrece GOF se han tenido en cuenta:

- ✓ **Patrones de creación:**

Creación (Singleton): Este patrón garantiza que solamente se cree una instancia de la clase y provee un punto de acceso global a él. Está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Este patrón se evidencia en la clase SourceEditor.js ya que el constructor de esta clase se crea privado, asegurándose de esta forma que cada vez que se haga una llamada a esta clase se estará utilizando esa única instancia que se creó anteriormente.

- ✓ **Patrones de estructura:**

Fachada (Facade): Es un patrón que define un único punto de conexión de un subsistema, este objeto fachada presenta una única interfaz unificada y es responsable de colaborar con los clientes. (Controlador de fachada) (Mühlrad, 2008). Un ejemplo de este patrón es la interfaz del módulo Diseñador de Consultas, la cual a pesar de estar dividida en varias subclases, cada una con sus respectivas responsabilidades, todas se encuentran agrupadas y se utilizan en una misma interfaz.

- ✓ **Patrones de comportamiento:**

Iterador (Iterator): Proporciona una forma de acceder a los elementos de una colección de objetos de manera secuencial sin revelar su representación interna (Gamma, 2011). Este patrón se evidencia en la clase MDXParser ya que esta clase es la encargada de recorrer la colección de tokens que forman el código de la consulta para a partir de este construir un XML con el que se trabajará.

Mediador (Mediator): Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente y permite variar la interacción entre ellos de forma independiente. Este patrón se utiliza en la clase UCQueryBuilder la cual está encargada de conectar las clases de la interfaz con la clase controladora. En clases como QueryDesigner o QueryEditor que se necesite llamar a acciones de la clase controladora, se le hace una llamada a la clase UCQueryBuilder la cual llama a su vez al método necesario de la clase QueryController, recoge la respuesta y la envía a la clase que la solicitó.

Visitante (Visitor): Permite aplicar una o más operaciones a un conjunto de objetos en tiempo de ejecución, representa una operación a realizar sobre los elementos de una estructura de objetos. Visitante permite definir una nueva operación sin cambiar las clases de los elementos en los que opera (SourceMaking, 2013). Este patrón se evidencia en la clase MDXParser la cual en tiempo de ejecución

visita al código de la consulta y realiza cambios sobre este como por ejemplo en el requisito Ocultar campos vacíos en el resultado en el cual se le añade el operador NON EMPTY al código de la consulta.

2.7 Diagrama de Secuencia

Un diagrama de secuencia muestra una interacción que representa la secuencia de mensajes entre instancias de clases, componentes, subsistemas o actores. El tiempo fluye por el diagrama y muestra el flujo de control de un participante a otro. En el diagrama, puede aparecer más de una instancia del mismo tipo. También puede haber más de una ocurrencia del mismo mensaje (Microsoft, 2016). En la siguiente figura se muestra el diagrama de secuencia del caso de uso **"Diseñar consulta"** del escenario diseñar gráficamente una consulta MDX y una breve descripción del mismo:

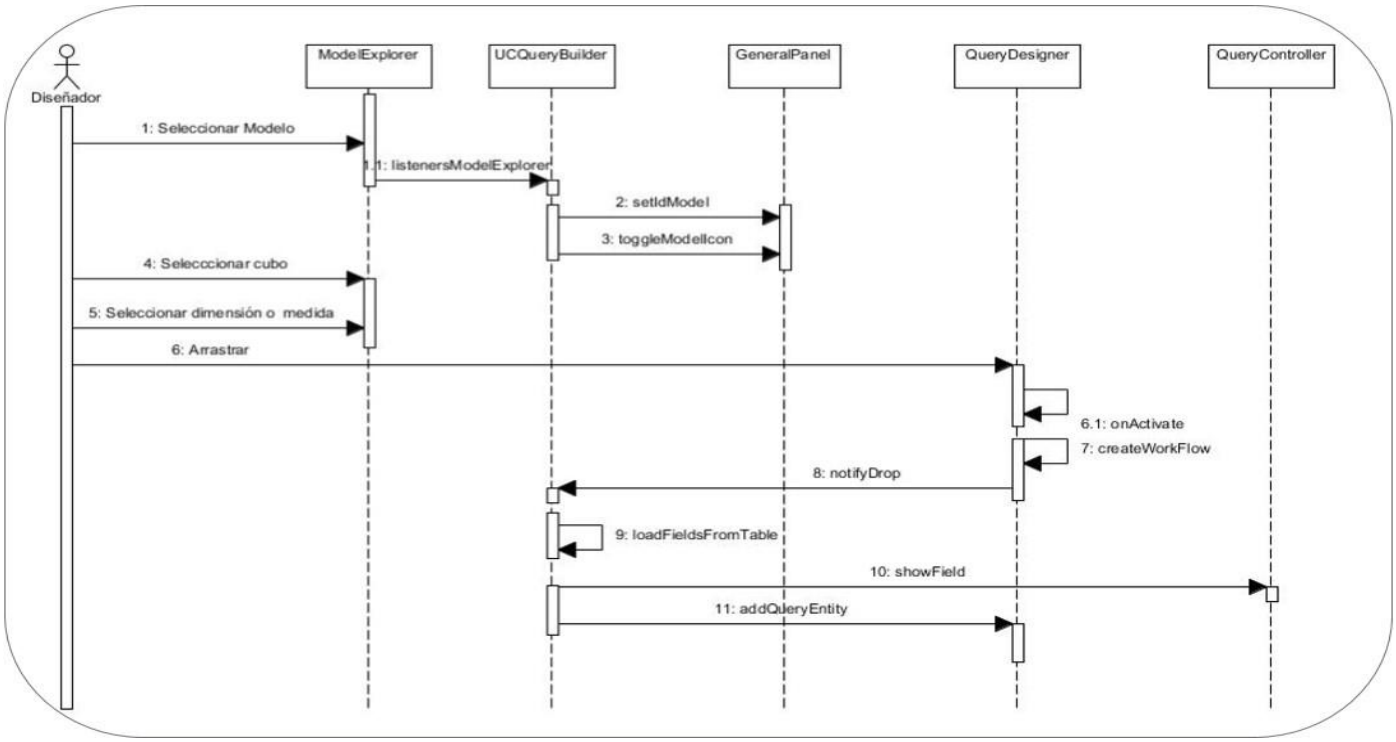


Fig. 7 Diagrama de secuencia "Diseñar consulta" del escenario "Diseñar gráficamente una consulta MDX".

En el diagrama de secuencia de la figura anterior el actor diseñador de consulta selecciona un Modelo OLAP en la interfaz ModelExplorer, al hacer esto como parte del método listenersModelExplorer se guarda el (id) del Modelo con el método setIdModel y se resalta el nodo seleccionado en negrita con el método toggleModelIcon, una vez seleccionado el modelo el actor pasa a seleccionar el cubo con que desea

trabajar, luego selecciona cualquier medida o dimensión de este cubo y lo arrastra al QueryDesigner, donde como parte del listener onActivate llama al método createWorkFlow el cual dibuja el área donde se va a trabajar y a su vez llama al método notifyDrop de la clase UCQueryBuilder creando un objeto entidad con los datos del cubo seleccionado el cual se le pasa como parámetro en la llamada al método loadFieldsFromTable. En este se hace una llamada al método showField de la clase QueryController el cual devuelve toda la información de las dimensiones, niveles y medidas que posee este cubo. Con esta información se crea un objeto entityField que se le pasa por parámetro al método addQueryEntity quedando mostrado este en el QueryDesigner en forma de tabla que contiene todos los datos del cubo con los cuales se diseñará la consulta.

Conclusiones del capítulo

Una vez concluido el presente capítulo se le dio fin a la etapa de análisis y diseño de la solución quedando relacionados todos los conceptos fundamentales de la investigación en el Modelo de Dominio donde se definieron nueve clases con sus respectivas descripciones. Se identificaron 11 requisitos agrupados en tres casos de uso de los cuales dos son arquitectónicamente significativos quedando representados en el Diagrama de casos de uso del sistema. Se elaboraron los diagramas de clases del diseño utilizando el patrón arquitectónico MVC y los patrones de diseño GRASP y GOF quedando definida la arquitectura a seguir en el desarrollo del diseñador.

Capítulo 3: Implementación y Pruebas del Diseñador gráfico de consultas MDX para GDR v2.0

El presente capítulo está enfocado a la implementación de la aplicación para dar solución a los requisitos especificados. Se elaboran los diagramas de componentes que forman parte del modelo de implementación. Se definen el estándar de codificación a utilizar en la implementación de la solución y se realiza el diagrama de despliegue. Además se realizan pruebas de software con el objetivo de descubrir y corregir errores.

3.1 Modelo de implementación

Como parte de la metodología OpenUp una vez concluida el levantamiento de los requisitos se procede a la implementación de la aplicación, por lo que se elabora el Modelo de implementación. Este modelo está conformado por los diagramas de componentes, describiendo cómo los elementos del modelo de diseño se implementan en términos de componentes, ficheros de código fuente y ejecutables. El modelo de implementación describe además cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización, disponibles en el entorno de implementación y en el lenguaje de programación utilizado y cómo dependen los componentes unos de otros (Presman, 2001).

3.1.1 Diagrama de componentes

Vinculado al Modelo de Implementación se encuentran los diagramas de componentes. Un componente es el empaquetamiento físico de los elementos de un modelo, como las clases en el modelo de diseño. El diagrama de componentes describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (Jacobson, y otros, 2000). Se realizó un diagrama de componentes por cada caso de uso, en la siguiente figura se muestra el diagrama de componentes del caso de uso **“Diseñar consulta”**:

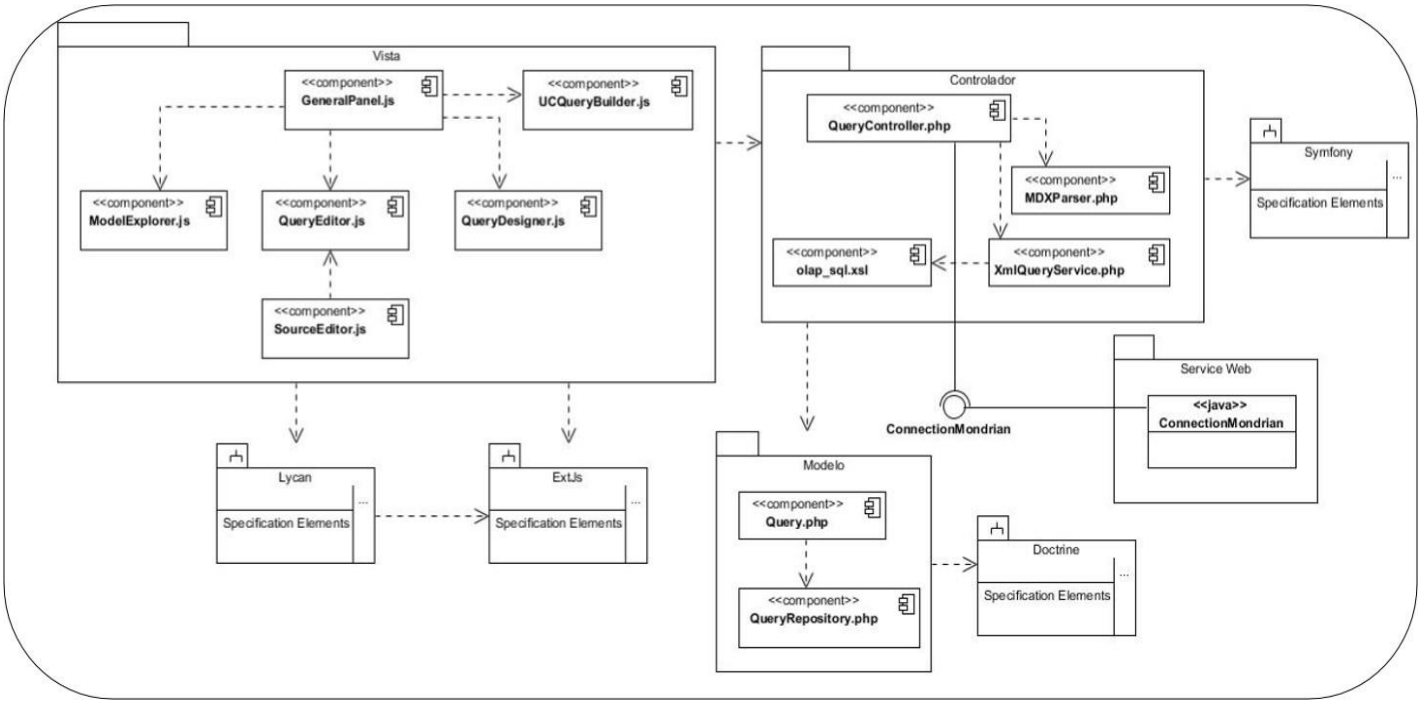


Fig. 8 Diagrama de componente "Diseñar consulta".

Descripción de los componentes

En el diagrama de componentes anterior se representa el caso de uso **"Diseñar consulta"** según el patrón arquitectónico MVC, incluyendo los framework Symfony, Lycan y ExtJS, el ORM[8] Doctrine y el Servicio web ConnectionMondrian utilizados en la implementación de la herramienta, dicho diagrama queda estructurado de la siguiente forma:

El paquete **"Modelo"** contiene los componentes referentes a la estructura de las consultas.

En el paquete **"Vista"** están ubicados los componentes de la interfaz principal con los que interactúa el usuario.

En el paquete **"Controlador"** se encuentra el componente encargado del funcionamiento de la aplicación.

3.1.2 Estándar de codificación

Para comenzar con la implementación del Diseñador gráfico de consultas MDX para GDR v2.0 los programadores deben establecer un estándar de codificación para que el trabajo con respecto a la generación de código se haga de forma coordinada. Un código fuente completo debe reflejar un estilo

[8] Mapeador de Objetos-Relacional

armonioso, como si un único programador hubiera escrito todo el código de una sola vez (Microsoft, 2016).

Se decidió continuar con la misma línea de trabajo definida por el equipo de desarrollo del GDR v2.0 ya que para implementar la solución se utilizó como base el Diseñador gráfico de consultas SQL.

A continuación, se muestran el estándar de codificación que se utilizó para la implementación del Diseñador gráfico de consultas MDX para GDR v2.0 continuando con la línea del proyecto:

UpperCamelCase: Las palabras que forman el nombre se escriben juntas y la primera letra de cada una de ellas en mayúscula.

LowerCamelCase: Es la práctica de escribir frases o palabras compuestas eliminando los espacios y poniendo en mayúscula la primera letra de cada palabra, excepto la de la primera que es en minúscula. Es utilizado para nombrar las funciones, variables y constantes.

En la siguiente figura se muestra un ejemplo de un fragmento de un método del Diseñador gráfico de consultas MDX que utiliza este estándar:

```
},
showSelectedField0fEntity: function(xmlDoc) {
    var type = this.getGeneralPanel().getQueryType();
    if (type === "mdx") {
        var arrayEntitys = this.getGeneralPanel().getQueryDesigner().getQueryEntity()[0];
        var columnasDesign = arrayEntitys.getColumns();
        var filasDesign = arrayEntitys.getRows();
        var deselectedFieldColumns = arrayEntitys.getDeselectedFields(columnasDesign);
        var deselectedFieldRows = arrayEntitys.getDeselectedFields(filasDesign);
        arrayEntitys.addFieldsInFilter(deselectedFieldColumns);
        arrayEntitys.addFieldsInFilter(deselectedFieldRows);
        var ejes = xmlDoc.getElementsByTagName("QUERY")[0].getElementsByTagName("SELECT")[0].getElementsByTagName("EJE");
        for (var i = 0; i < ejes.length; i++) {
            var columnasCode = ejes[0].getElementsByTagName("column");
```

Fig. 9 Fragmento del Método showSelectedField0fEntity.

Clases: Los nombres de las clases así como sus atributos o métodos deben estar en UpperCamelCase, como se explica anteriormente.

Funciones: Los nombres de las funciones o métodos deben usar LowerCamelCase y además deben existir los métodos como modificador de acceso para cada atributo de la clase, aunque para los atributos públicos no es necesario.

Variables: Las variables deben usar LowerCamelCase y además tener nombres coherentes y que la identifiquen, o sea que no se le deben poner nombres como “a” o “b”, solo en caso de que la variable sea

demasiado genérica como para no tener un nombre concreto, como por ejemplo el índice de un bucle (un ciclo for o while) se declarará con nombre “i”, “j”, “k” y así sucesivamente en el caso de varios bucles anidados.

3.1.3 Diagrama de despliegue

Es un diagrama que muestra la configuración del hardware que participa en la ejecución de los componentes en los que se encuentra distribuido el sistema. Gráficamente un diagrama de despliegue es una colección de nodos físicos en tiempo de ejecución, que pueden representar dispositivos o computadoras con capacidad de procesamiento. Estos se relacionan a través de arcos que definen los protocolos de comunicación para el intercambio de información. Es un modelo de objetos que describe la distribución física de la aplicación, representando cómo se distribuyen los recursos de cómputo formados por dispositivos de hardware. Además se representan las relaciones entre ellos (Jacobson, y otros, 2000). En la siguiente figura se muestra el diagrama de despliegue del sistema correspondiente a GDR v2.0:

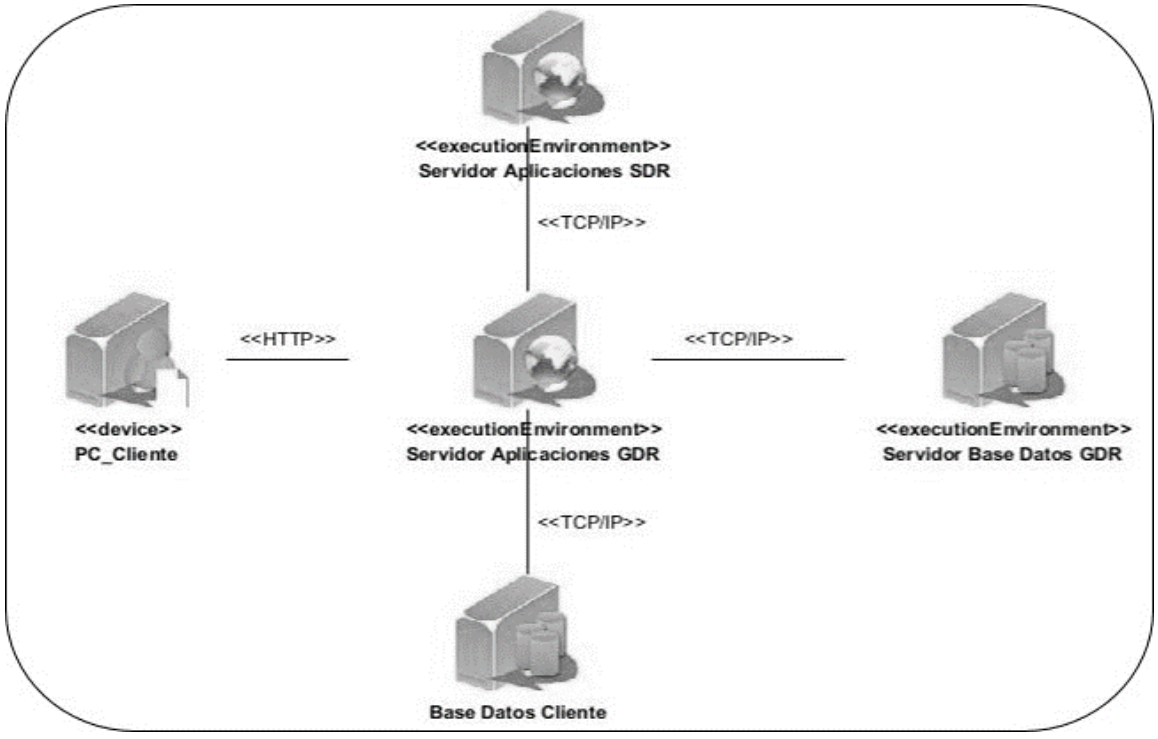


Fig. 10 Diagrama de despliegue del sistema.

Descripción de los nodos

- ✓ **PC_Cliente:** computadora desde donde se ejecuta la aplicación.

- ✓ **Servidor Aplicaciones GDR:** es donde se encuentra el servidor web con el GDR v2.0.
- ✓ **Servidor Bases Datos GDR:** contiene la base de datos de GDR v2.0.
- ✓ **Base Datos Cliente:** es donde se encuentran los orígenes de datos que serán cargados por el cliente.
- ✓ **Servidor Aplicaciones SDR:** contiene al servidor dinámico de reportes (SDR) que es el encargado de la generación de los reportes.
- ✓ **HTTP:** se utiliza para conectar la computadora del cliente y el servidor donde está la aplicación.
- ✓ **TCP/IP:** protocolo para conectar el servidor de aplicaciones con las bases de datos.

3.2 Interfaces de la aplicación

Interfaz de la vista de Diseño

La aplicación desarrollada consta con una interfaz para la vista de Diseño. En ella se encuentra un menú de Gestionar consultas con las acciones de Nueva, Renombrar, Exportar, Importar, Guardar, Abrir y Ejecutar. También cuenta con un Explorador de Modelos donde se encuentran los modelos, en el caso del presente trabajo sólo se utilizan los modelos OLAP. Estos están compuestos por cubos y estos a su vez contienen dimensiones y medidas. Además, todo el contenido del modelo es representado en forma de árbol jerárquico. Una vez seleccionado el cubo el usuario puede arrastrar cualquier dimensión o medida del mismo hacia la vista de Diseño donde quedará diseñado gráficamente con toda la información referente a él. En este diseño el usuario podrá escoger los campos que desea que aparezcan en la consulta y en que eje desea visualizarlo, en columnas o en filas. Las consultas salvadas son almacenadas en el sistema en la sección Consultas del cubo correspondiente. Una vez salvada una consulta en el sistema está podrá ser abierta en caso de que el usuario lo desee. En la siguiente figura se muestra la Interfaz de la vista de Diseño:

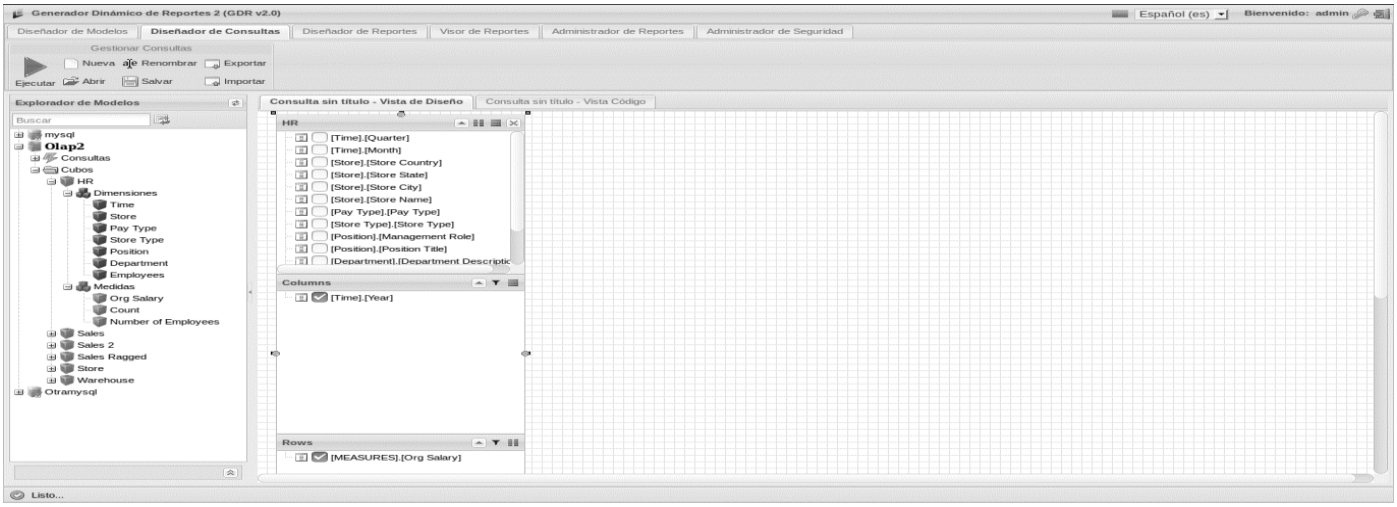


Fig. 11 Interfaz de la vista de Diseño.

Interfaz de la vista Código

La otra interfaz en la que se puede trabajar es la vista Código, la cual cuenta con un área de trabajo para la edición de las consultas. En ella se puede modificar el código de la consulta diseñada desde la vista de Diseño, ya sea abriendo una nueva consulta o escribiéndola, además de contar con la opción de autocompletamiento donde se muestran las palabras reservadas del lenguaje MDX, el cual puede usarse como apoyo en todo este proceso. Esta vista además cuenta con un panel donde se muestra el resultado de la consulta una vez sea ejecutada satisfactoriamente y en caso de contener algunos errores estos se muestran en el panel Errores. En el panel "Vista previa de los datos" se encuentran los botones de "Ocultar" en caso que el resultado de la consulta ejecutada contenga campos en blanco y se desee removerlos, el botón de "Guardar" donde se podrá exportar el resultado de la consulta a un archivo Excel el cual se almacenará en la carpeta seleccionada por el usuario, además de la opción "Actualizar" la vista previa de los datos. En la siguiente figura se muestra la Interfaz de la vista Código:

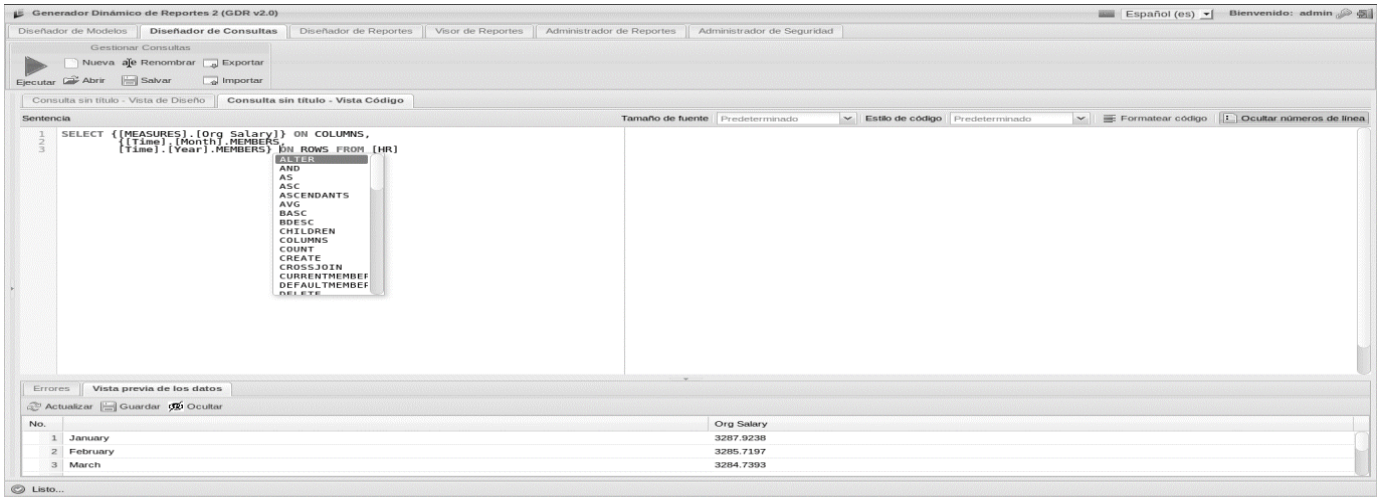


Fig. 12 Interfaz de la vista Código.

3.3 Pruebas de Software

Siguiendo la metodología OpenUp se realizó una serie de pruebas al producto una vez finalizada la implementación de este. Ya que es necesario verificar el funcionamiento, la calidad del mismo y garantizar que cumpla con los requisitos planteados por los clientes y sean detectados y corregidos todos los posibles errores (Pressman, 2012).

Estrategia de Prueba

Para darle inicio a las pruebas de un software lo primero es definir una estrategia de prueba donde queden plasmados los niveles de prueba a tratar, así como los tipos de prueba, el método de prueba y la técnica aplicada en este método.

Una estrategia de prueba de software integra los métodos de diseño de casos de pruebas de software en una serie bien planeada de pasos que desembocará en la construcción del mismo. Debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas y la recolección y evaluación de los datos resultantes. Esta tiene que ser lo suficientemente flexible para promover un enfoque personalizado. Al mismo tiempo, debe ser lo adecuadamente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del proyecto (Pressman, 2011).

A continuación, se describen los niveles de pruebas que serán aplicados para verificar y validar un producto de software.

Nivel de Unidad

La prueba de unidad, es donde se ponen a prueba unidades de programa o clases de objetos individuales. Estas deben enfocarse en comprobar la funcionalidad de objetos o métodos (Sommerville, 2011). Con las pruebas a nivel de unidad se comprobaron individualmente las funcionalidades y métodos verificando el correcto funcionamiento de los mismos. Esto se evidencia en algunos métodos de la clase MDXParser.php como son Select () y From (), así como el método getNiveles () de la clase controladora los cuales arrojaron resultados satisfactorios en cortos plazos de tiempo.

Nivel de Integración

La prueba de integración es una técnica sistémica para construir la arquitectura del software mientras, al mismo tiempo, se aplican pruebas para descubrir errores asociados con la interfaz. El objetivo es tomar componentes a los que se aplicó una prueba de unidad y construir una estructura de programa que determine el diseño (Pressman, 2011). Para aplicar este tipo de pruebas se aplicará un enfoque incremental aplicando la estrategia incremental ascendente.

Integración Incremental: Es la antítesis del enfoque del "big bang". El programa se construye y prueba en pequeños incrementos, en los cuales resulta más fácil aislar y corregir los errores.

- ✓ **Integración Ascendente (*Bottom-Up*):**

La prueba de integración ascendente se aplica con la construcción y prueba de módulos atómicos. Debido a que los componentes se integran de abajo hacia arriba, siempre está disponible el procesamiento requerido para los componentes subordinados a un determinado nivel (Pressman, 2011).

Nivel de aceptación

Esta es la etapa final en el proceso de pruebas, antes de que el sistema se acepte para uso operacional. El sistema se pone a prueba con datos suministrados por el cliente del sistema, en vez de datos de prueba simulados. Las pruebas de aceptación revelan los errores y las omisiones en la definición de requerimientos del sistema, ya que los datos reales ejercitan el sistema en diferentes formas a partir de los datos de prueba. Las pruebas de aceptación revelan problemas de requerimientos, donde las instalaciones del sistema en realidad no cumplan las necesidades del usuario o cuando sea inaceptable el rendimiento del sistema (Sommerville, 2011).

Tipos de Pruebas

Las pruebas son diseñadas para encontrar el mayor número de errores. Cada tipo de prueba tiene un objetivo específico y una técnica que lo soporte (Cabrera , y otros, 2008). Existen diferentes tipos de pruebas que se pueden aplicar para verificar que el Diseñador gráfico de consultas MDX cumple con todos los requisitos identificados y se logre un correcto funcionamiento de este. A continuación, se explican los tipos de pruebas utilizados:

Prueba de Aceptación de tipo alfa: Estas pruebas son realizadas por el cliente (Jefa de proyecto GDR) para certificar que el sistema es válido para él. Son básicamente pruebas funcionales sobre el sistema una vez terminado y buscan comprobar que se satisfacen los requisitos establecidos. Son realizadas en presencia del desarrollador, este en caso de que existan errores los registrará y posteriormente les dará solución. Esta prueba es ejecutada nuevamente, repitiendo el proceso hasta lograr que el producto quede libre de errores y el cliente obtenga el producto que solicitó (Valdez, y otros, 2013).

Prueba Funcional: Son aquellas que tienen por objetivo demostrar que los sistemas desarrollados, cumplen con las funciones específicas para los cuales han sido creados. Es común que sea desarrollada por los analistas de pruebas con apoyo de algunos usuarios finales y están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software (Sánchez, y otros, 2013). Utiliza el método el Caja Negra

Métodos de Prueba

Los Métodos de Prueba de Software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con un menor tiempo y esfuerzo (González, 2016). Existen dos fundamentales: Caja Blanca para verificar la estructura y Caja Negra para la funcionalidad del software. En el proceso de pruebas en cuestión se hace uso del método de prueba de Caja Negra.

Método de prueba: Caja Negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. Esta prueba permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación. La técnica aplicada en este método es la **Partición equivalente** la cual divide el campo de entrada de un programa en clases de datos de los que

se pueden derivar casos de prueba. 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 (Pressman, 2012).

Diseño del Caso de Prueba

Los casos de prueba son un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema (Aristegui, 2010).

Para realizar las pruebas se utilizaron juegos de datos válidos e inválidos haciendo uso de la técnica de partición de equivalencia. A continuación, se muestra el caso de prueba diseñar gráficamente una consulta MDX asociado al caso de uso **"Diseñar consulta".**

Descripción de las variables

Tabla. 6 Descripción de las variables del diseño de casos de prueba del caso de uso Diseñar consulta

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
V1	Cubo	campo de selección	No	Estructura de datos que supera las limitaciones de las bases de datos relacionales y proporciona un análisis rápido de datos.
V2	Dimensión	campo de selección	No	Atributos estructurales del cubo.
V3	Niveles	campo de selección	No	Contenidas dentro de las dimensiones.
V4	Medidas	campo de selección	No	Conjunto de valores basados en una columna de la tabla de hechos del cubo.
V5	Consulta	campo de texto o campo de selección	No	Estructura de una consulta MDX.
V6	Nombre	campo de texto	No	Este campo de texto admite letras minúsculas, letras mayúsculas, números y caracteres especiales.

Tabla. 7 Caso de prueba Diseñar gráficamente un consulta MDX del caso de uso Diseñar consulta

Escenario	Descripción	V1	V2	V3	V4	Respuesta del sistema	Flujo central

EC 1.1 Diseñar consulta correctamente	Permite diseñar una consulta mdx de forma gráfica.	V	V	V	V	Una vez arrastrado para la vista de Diseño o una dimensión o una medida el sistema muestra una tabla conformada por filas, columnas y filtros.	1-Autenticarse.2-Seleccionar pestaña de Diseñador de Consultas.3-Seleccionar modelo OLAP en el Explorador de Modelos.4-Seleccionar el cubo con que desee trabajar.5-Seleccionar de este cubo cualquier medida o dimensión.6-Arrastrar la medida o la dimensión seleccionada para la vista de Diseño. 7-Seleccionar de la tabla que aparecerá en la vista de Diseño de la sección filtro las dimensiones con los niveles que desee de esta y las medidas. 8-Seleccionar en que sección poner las dimensiones y medidas en filas o columnas mediante los botones en la parte superior de la tabla.
		HR	Time	Year	Count		

EC 1.2 Intentar hacer diseños simultáneos de cubos diferentes.	Al existir un diseño de un cubo en el área diseño y el usuario arrastrar un campo de otro cubo para esta área el sistema muestra una notificación “Solo se puede trabajar con un cubo a la vez en la vista de Diseño”.	I Sales	I Store	I Store City	I Unit Sales	Si existe un diseño de un cubo en el área diseño y el usuario arrastra un campo de otro cubo para esta área el sistema muestra la siguiente notificación: “Solo se puede trabajar con un cubo a la vez en la vista de Diseño ".	1-Autenticarse.2-Seleccionar pestaña de Diseñador de Consultas.3-Seleccionar modelo OLAP en el Explorador de Modelos.4-Seleccionar el cubo con que desee trabajar.5-Seleccionar de este cubo cualquier medida o dimensión.6-Arrastrar la medida o la dimensión seleccionada para la vista de Diseño.7-Arrastrar cualquier medida o dimensión de otro cubo del Explorador de Modelos.8-El sistema muestra la siguiente notificación: “Solo se puede trabajar con un cubo a la vez en la vista de Diseño “.

EC 1.3 Arrastrar para la vista de Diseño un metadato del mismo cubo que ya se encuentra diseñado.	Al existir un diseño de un cubo en el área diseño y el usuario arrastrar un campo de este mismo cubo para esta área el sistema muestra la siguiente notificación: "El cubo seleccionado ya se encuentra en la vista de Diseño".	I	I	I	I	Si existe un diseño de un cubo en el área diseño y el usuario arrastra un campo de ese mismo cubo sistema muestra la siguiente notificación: "El cubo seleccionado ya se encuentra en la vista de Diseño".	1-Autenticarse.2-Seleccionar pestaña de Diseñador de Consultas.3-Seleccionar modelo OLAP en el Explorador de Modelos.4-Seleccionar el cubo con que desee trabajar.5-Seleccionar de este cubo cualquier medida o dimensión.6-Arrastrar la medida o la dimensión seleccionada para la vista de Diseño.7-Arrastrar cualquier medida o dimensión ese mismo cubo.8-El sistema muestra la siguiente notificación: "El cubo seleccionado ya se encuentra en la vista de Diseño".
		HR	Time	Month	Count		

Para consultar la descripción de los restantes Casos de prueba debe remitirse al expediente de proyecto, específicamente a la plantilla Diseño_de_casos_de_prueba.xls

Este proceso permitió verificar el cumplimiento de los requisitos funcionales del sistema, donde los resultados de las pruebas que no fueron satisfactorios pasaron a ser No conformidades (NC) y se resolvieron posteriormente.

Resultados de las pruebas funcionales aplicando el método caja negra

Para validar el correcto funcionamiento del software se realizó la prueba de caja negra a través de los casos de pruebas asociados a cada caso de uso. Fueron detectadas un total de ocho no conformidades

con la clasificación: cinco de funcionalidad, una opción que no funciona, una de error de interfaz y una recomendación durante tres iteraciones, en la primera iteración se detectaron seis no conformidades, en la segunda iteración dos y en una tercera iteración no se encontró ninguna no conformidad. De forma general se excluyeron todos los errores culminando las pruebas con resultados satisfactorios. A continuación, se muestra en la siguiente figura las No conformidades detectadas y su clasificación:

Fig. 13 Gráficas de No Conformidades

Entiéndase por NC de funcionalidad cuando los resultados mostrados al realizar una acción no son los esperados, NC de interfaz a los errores visuales relacionados con el diseño de las interfaces, NC de recomendaciones son sugerencias hechas por los probadores y NC de opciones que no funcionan son las que no muestran ningún resultado al realizar alguna acción.

A continuación, en el **"Registro de defectos y dificultades detectados"** se muestran ejemplos de las no conformidades detectadas durante las pruebas a la aplicación así como su estado (Pendiente (PD) o Resuelto (RA)) y la respuesta del equipo de desarrollo:

Tabla. 8 Registro de defectos y dificultades detectados

Clasificación	No.	No conformidad (NC)	Estado NC	Resp. equipo de desarrollo

NC funcionalidad	1	Al Ejecutar una consulta desde la vista de Diseño muestra un error en la línea 1 donde detecta el token incorrecto (<), en cambio se ejecuta la misma desde la vista Código y se muestra el resultado satisfactoriamente.	PD 1/5/16 RA 3/5/16	Se especificó como parámetro del método ejecutar consulta el código de la consulta en vez del XML.
NC opciones que no funcionan	2	Cuando el usuario selecciona el botón “Ocultar” no remueve los campos en blanco del resultado.	PD 3/5/16 RA 4/5/16	Se corrigió el nombre del parámetro IdModel necesario para la ejecución de este método.
NC funcionalidad	3	Cuando se arrastra para la vista de Diseño un campo de otro cubo el sistema no muestra la notificación “Solo se puede trabajar con un cubo a la vez en la vista de Diseño "	PD 5/5/16 RA 6/5/16	Se agregó una nueva validación para mostrar el mensaje “Solo se puede trabajar con un cubo a la vez en la vista de Diseño "
NC funcionalidad	4	Cuando se arrastra para la vista de Diseño un campo del mismo cubo que ya se encuentra diseñado el sistema no muestra la notificación "El cubo seleccionado ya se encuentra en la vista de Diseño".	PD 5/5/16 RA 6/5/16	Se agregó una nueva validación para mostrar el mensaje “El cubo seleccionado ya se encuentra en la vista de Diseño".
NC interfaz	5	En el diseño de la vista de Diseño la imagen del Filtro no se define bien.	PD 9/5/16 RA 10/5/16	Se agregó una imagen con mejor resolución.
NC recomendación	6	Al autocompletar el código de la consulta se mostraba un listado con las palabras reservadas tanto del lenguaje SQL como de MDX.	PD 11/5/16 RA 12/5/16	Se le añadió una validación y dependiendo del resultado se muestran las palabras de MDX o SQL.
NC funcionalidad	7	Al añadir manualmente un campo al código de la consulta este no actualiza el diseño de la vista de	PD 13/5/16 RA 15/5/16	Se comparan los nombres de los campos en el código con los nombres de los campos en

		Diseño.		el filtro y si coinciden se añade a columnas o filas según sea el caso.
NC funcionalidad	8	Se prueba Abrir una consulta de las salvadas en el sistema existiendo un diseño en la vista de Diseño y esta se abre sin eliminar la que anteriormente existía.	PD 16/5/16 RA 18/5/16	Se remueve el diseño existente en la vista de Diseño.

Resultados de las pruebas de unidad

Para desarrollar esta prueba se crearon en la clase QueryController distintos métodos con el nombre Test con el objetivo de comprobar individualmente el correcto funcionamiento de varios métodos de la solución. Cada uno de estos conteniendo el código íntegro del método a probar y junto con el retorno de cada uno se utilizó la función var_dump para visualizar el valor de alguna variable de interés. Se definió en la clase routing.yml la ruta hacia el método en la clase controladora y el nombre del test, el cual al escribirlo en el navegador se ejecutó y mostró la respuesta correcta o los errores cometidos señalados. A continuación, se muestra en la siguiente figura un fragmento del resultado del test del método getNiveles ().

Fig. 14 Fragmento del resultado del test del método getNiveles()

Resultados de las pruebas de integración

Una vez terminado el Diseñador gráfico de consultas MDX para GDR v2.0 se integra al módulo Diseñador de Consultas hasta que funciona como un todo. A continuación, se explican los pasos para la integración:

1. Agregar los ficheros que contienen las nuevas clases desarrolladas para el módulo Diseñador de consultas:

- ✓ MDXQueryService.php, MDXParser.php, MDXLexer.php, MDXTokenKind.php en la ruta GDR2/src/GDR2/QDBundle/lib/common/ManipulateQuery/.
- ✓ olap_sql.xsl en la ruta GDR2/src/GDR2/QDBundle/lib/XSLtrans/
- ✓ QueryEntity.js, QueryDesigner.js, UCQueryBuilder.js en la ruta GDR2/src/GDR2/QDBundle/Resources/public/js/query_designer/view/query/.

2. Incluir los nuevos ficheros .js de la vista en la plantilla twig del módulo para que estos sean cargados.
3. Realizar cambios en las clases QueryEditor.js, GeneralPanel.js, QueryController.php, NewQuery.js, RunQuery.js, SaveQuery.js y SourceEditor.js para adaptarlas a los escenarios donde se comparten funcionalidades y comportamientos entre los diseñadores de consultas MDX y SQL.
4. Luego se le realizan pruebas funcionales a la aplicación para comprobar que los nuevos cambios funcionen correctamente y que estos no hayan afectado ninguna de las otras funcionalidades del sistema.

Resultados de las pruebas de aceptación

Las pruebas de aceptación se realizan con el objetivo de validar la solución a través de un encuentro con el Jefe de proyecto del GDR, el cual comprueba que el sistema cumple con los requerimientos planteados y da muestra de su conformidad en el documento oficial Carta de Aceptación del cliente donde queda plasmado que el Diseñador gráfico de consultas MDX para GDR v2.0 está apto para ser utilizado y cumple con las expectativas planteadas. (Ver anexo 2)

Conclusiones del capítulo

Una vez finalizado el presente capítulo se concluye que la realización de los diagramas de componentes correspondientes a cada uno de los casos de uso permitió organizar las clases a nivel de componente para su posterior implementación. Para describir la distribución física de la aplicación se elaboró el diagrama de despliegue. La aplicación de la estrategia de prueba definida permitió identificar un total de ocho no conformidades que quedaron solucionadas en tres iteraciones lo cual permitió validar la solución implementada.

Conclusiones generales

Una vez concluida la investigación se arribó a las siguientes conclusiones:

- ✓ El estudio de los principales conceptos asociados a la investigación, permitió definir la estructura básica del Diseñador gráfico de consultas MDX, así como sus principales funcionalidades.
- ✓ El estudio del marco teórico de la investigación posibilitó la selección de la metodología, herramientas y tecnologías a utilizar para la implementación de la solución propuesta.
- ✓ La definición de los requisitos funcionales y no funcionales que debe cumplir el sistema permitió realizar un correcto análisis y diseño del mismo mediante el uso del patrón arquitectónico MVC así como los patrones GRASP y GOF.
- ✓ La implementación de la solución desarrollada permitió dar cumplimiento a la totalidad de los requisitos de software identificados con el cliente durante la etapa de análisis.
- ✓ La solución fue validada a partir de la ejecución de pruebas funcionales y de aceptación las cuales permitieron comprobar el correcto funcionamiento del Diseñador gráfico de consultas MDX para GDR v2.0.

Recomendaciones

Después de haber logrado los objetivos definidos en este trabajo, se plantea la siguiente recomendación:

- ✓ Agregar al Diseñador gráfico de consultas MDX las funcionalidades que dan soporte a las sentencias whit y where propias del lenguaje, aumentando así la complejidad de las consultas a diseñar lo que permite obtener datos más específicos.

Referencias Bibliográficas

Abreu , Aldis Joan, y otros. 2012. *Generador dinámico de reportes.* 2012.

Agut, R,M. 2011. *Especificación de Requisitos Software según el estándar IEEE 830.* 2011.

Alegsa, Leandro. 2010. Sitio Web Alegsa.com. *Sitio Web Alegsa.com.* [En línea] 12 de 5 de 2010. [Citado el: 12 de 11 de 2015.] http://www.alegsa.com.ar/Dic/servidor%20de%20aplicaciones.php.

Apache Tomcat. 2015. Sitio web Apache Tomcat. *Sitio web Apache Tomcat.* [En línea] 2015. [Citado el: 12 de 11 de 2015.] http://tomcat.apache.org/.

Ariste, María Cecilia, y otros. *Un Marco de Trabajo para Enseñanza del paradigma MD.*

Aristegui, José Luis. 2010. *LOS CASOS DE PRUEBA EN LA PRUEBA DEL SOFTWARE.* 2010.

Barranco, Jesus. 2002. *Metodologías Del Análisis Estructurado del Sistema.* 2002.

Biosca, Enric. 2009. *Tutorial MDX.* 2009.

Blank, Isabel , Herrera, Larissa y Ortiz, Miguel. 2005. Sitio Web Pruebas Funcionales – Software II. *Sitio Web Pruebas Funcionales – Software II.* [En línea] Mayo de 2005. [Citado el: 9 de 4 de 2016.] http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf.

Booch , Grady , Jacobson , Ivar y Rumbaugh, Jim. 2000. *El Lenguaje Unificado de Modelado.* s.l. : Addinson-Wesley, 2000.

Cabrera , Yoan Manuel , Gonzalez , Yosbel Ernesto y Cruz , Claribel Lucy. 2008. *El Proceso de Prueba de del proyecto Sistema de Información para la Salud (SISalud).* 2008.

Cendejas. 2011. Sitio web eumed. *Sitio web eumed.* [En línea] 2011. [Citado el: 1 de 10 de 2015.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

Craig, Larman. 2003. *UML y Patrones.* 2003.

Cuesta. 2005. Sitio web SG Buzz. *Sitio web SG Buzz.* [En línea] 11 de 2005. [Citado el: 4 de 10 de 2015.] https://sg.com.mx/revista/6/patrones-casos-uso.

Cuesta, Saúl. 2005. Sitio Web SG Buzz conocimiento para crear software grandioso. *Sitio Web SG Buzz conocimiento para crear software grandioso.* [En línea] 2005. [Citado el: 10 de 3 de 2016.] http://sg.com.mx/content/view/510.

Dataprix. 2007. Sitio Web Dataprix. [En línea] 2007. [Citado el: 1 de octubre de 2015.] http://www.dataprix.com/manual-mdx/ti/introduccion-consulta-lenguaje-multidimensional-mdx.

Diseño de consulta. 2015. Sitio web Diseño de consulta. *Sitio web Diseño de consulta.* [En línea] 27 de 9 de 2015. [Citado el: 4 de 11 de 2015.] https://help.libreoffice.org/Common/Query_Design/es.

Editor de consultas MDX para GDR v2.0. 2015. *Editor de consultas MDX para GDR v2.0.* La Habana : s.n., 2015.

Eguiluz, Javier. 2008. Sitio web Introducción a JavaScript. *Sitio web Introducción a JavaScript.* [En línea] 7 de 6 de 2008. [Citado el: 12 de 11 de 2015.] http://librosweb.es/libro/javascript/capitulo_1.html.

Gamma, Erich . 2011. *Design Patterns.* 2011.

gespro. 2015. gespro Suite de Gestión de Proyectos. *gespro Suite de Gestión de Proyectos.* [En línea] 2015. [Citado el: 4 de 9 de 2015.] https://gespro.datec.prod.uci.cu/login?back_url=https%3A%2F%2Fgespro.datec.prod.uci.cu%2F.

Gimson, Loraine, Gil, Daniel Gustavo y Rossi, Gustavo . 2012. *Metodologías ágiles y desarrollo basado en conocimiento.* 2012.

González, Héctor. 2016. Sitio Web Plataforma Virtual Educativa ITCA. *Sitio Web Plataforma Virtual Educativa ITCA.* [En línea] 2016. [Citado el: 3 de 5 de 2016.] http://virtual.itca.edu.sv/Mediadores/stis/43___visiones_interna_y_externa_de_las_pruebas.html.

Grosso, Andrés. 2011. sitio web Prácticas de software. *sitio web Prácticas de software.* [En línea] 21 de 3 de 2011. [Citado el: 26 de 1 de 2016.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

GUEVARA , JORGE EDUARDO y VALENCIA, JANETH DEL CARMEN . 2007. *DATA WAREHOUSE PARA EL ANÁLISIS ACADÉMICO DE LA ESCUELA POLITECNICA NACIONAL.* 2007.

Herramientas Case . 2011. Sitio web Herramientas Case . *Sitio web Herramientas Case .* [En línea] 2 de 4 de 2011. [Citado el: 12 de 11 de 2015.] http://fds-herramientascase.blogspot.com/.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison-Wesley, 2000.

Krall , César . 2016. Sitio Web apr. *Sitio Web apr.* [En línea] 2016. [Citado el: 1 de 1 de 2016.] http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

Lafosse, Jérome. 2012. *Expert IT Struts 2-El framework de desarrollo de aplicaciones JAVA EE.* 2012.

Lanzillotta, Analía. 2012. Sitio Web Master Magazine. [En línea] 2012. http://www.mastermagazine.info/termino/6841.php.

LibrosWeb. 2016. *Symfony 1.2, la guía definitiva Capítulo 2. El patrón MVC .* 2016.

Lopez, Jose Manuel. 2001. [En línea] 15 de 10 de 2001. [Citado el: 25 de 11 de 2015.] http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node21.html.

Macías, Luis Ernesto, Aluija, Víctor Emilio y Monteagudo, Yamila. 2016. *SISTEMA DE GESTIÓN Y GESTIÓN Y PROMOCIÓN DE PROYECTOS DE INVESTIGACIÓN Y DESARROLLO.* La Habana,Cuba : s.n., 2016.

Martinez , Rafael. 2013. Sitio web PostgreSQL-es. *Sitio web PostgreSQL-es.* [En línea] 2013. [Citado el: 12 de 11 de 2015.] http://www.postgresql.org.es/sobre_postgresql.

Martínez, Jorge Alfredo. 2007. Sitio Web Gestiopolis. [En línea] 2007. [Citado el: 20 de septiembre de 2015.] http://www.gestiopolis.com/olap-y-el-diseno-de-cubos/.

Microsoft. 2016. Sitio Web Microsoft. *Sitio Web Microsoft.* [En línea] 2016. [Citado el: 24 de 4 de 2016.] https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

—. 2014. Sitio Web Microsoft . [En línea] 2014. https://msdn.microsoft.com/es-es/library/gg399173%28v=sql.120%29.aspx.

Mühlrad, Daniel. 2008. *Patrones de Diseño.* 2008.

netbeans. 2012. Sitio Web netbeans.org. *Sitio Web netbeans.org.* [En línea] 20 de 11 de 2012. [Citado el: 12 de 11 de 2015.] http://netbeans.org/community/releases/72/relnotes.html..

PATRONES DE DISEÑO. 2011. PATRONES DE DISEÑO. *PATRONES DE DISEÑO.* [En línea] 28 de 3 de 2011. [Citado el: 8 de 3 de 2016.]

php. 2015. Sitio web php. *Sitio web php.* [En línea] 2015. [Citado el: 12 de 11 de 2015.] http://php.net/manual/es/intro-whatis.php.

Potencier, Fabien y Zaninotto, Francois. 2009. *Symfony 1.4,la guía definitiva.* 2009.

Presman, Roger. 2001. *Ingeniería de Software, un enfoque práctico.* 2001.

Pressman, Roger S. 2012. *Ingeniería de Software:Un Enfoque Práctico.5ta edición.* 2012.

—. 2011. *Ingeniería de Software:Un Enfoque Práctico.6ta edición.* 2011.

Pressman, Roger S. 2010. *Software engineering: a practitioner's approach - 7th ed.* s.l. : Palgrave Macmillan, 2010.

Programación de Computadoras. 2011. *Programación de Computadoras.* 2011.

Rivera , Luis Alberto y Tixe , Juan Pablo. 2010. *Integración de la Metodología Agil XP el Estudio de de Windows Communication Foundation como Aternativa Metodológica para el Desasrrollo de Software Orientado a Servicios.* 2010.

Sabio, Gustavo y Pinciroli, Fernando. 2009. *Descripcion de CU.* 2009.

Salazar, Lianet Labrada. 2012. *Administrador de Reportes para la versión 2.0 del Generador Dinámico de Reportes.* 2012.

Sánchez, Sureny Orihuela y López, Liandry Monteslier. 2013. *Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes.* La Habana : s.n., 2013.

Scielo. 2013. Sitio web Scielo. *Sitio web Scielo.* [En línea] 2013. [Citado el: 1 de 4 de 2016.] http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext.

Sencha ExtJS. 2015. *Sencha ExtJS.* 2015.

Sinnexus. 2015. Sitio Web Sinnexus. [En línea] 2015. [Citado el: 1 de octubre de 2015.] http://www.sinnexus.com/business_intelligence/olap_vs_oltp.aspx.

Software. 2013. Sitio web Software. *Sitio web Software.* [En línea] 11 de 12 de 2013. [Citado el: 12 de 11 de 2015.] http://www.software.com.ar/visual-paradigm-para-uml.html..

Sommerville, Ian. 2011. *Ingeniería de Software:9na Edición.* México : s.n., 2011.

—. 2011. *Ingeniería del software.7ma Edicion.* 2011.

Sosa, Dailyn. 2013. Sitio Web Eumed.net. *Sitio Web Eumed.net.* [En línea] 2013. [Citado el: 10 de 3 de 2016.] http://www.eumed.net/libros-gratis/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm.

SourceMaking. 2013. Sitio web SourceMaking:Mediator Design Pattern. *Sitio web SourceMaking:Mediator Design Pattern.* [En línea] 2013. [Citado el: 9 de 3 de 2016.] https://sourcemaking.com/design_patterns/mediator.

Svitla Systems. 2016. Sitio web Methods & Tools Software Development Magazine. *Sitio web Methods & Tools Software Development Magazine.* [En línea] 2016. [Citado el: 4 de 5 de 2016.] http://www.methodsandtools.com/archive/archive.php?id=24.

Ubuntu. 2014. Sitio web Ubuntu documentation. *Sitio web Ubuntu documentation.* [En línea] 2014. [Citado el: 12 de 11 de 2015.] https://help.ubuntu.com/lts/serverguide/httpd.html.

UML. 2012. sitio web Modelado de Sistemas con UML. *sitio web Modelado de Sistemas con UML.* [En línea] 2012. [Citado el: 3 de 10 de 2015.] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

Unad. 2006. Sitio Web Unad. *Sitio Web Unad.* [En línea] 9 de 2006. [Citado el: 16 de 3 de 2016.] http://datateca.unad.edu.co/contenidos/200609/exeuml/leccin_39_diagrama_de_clases_de_diseo.html.

Valdez, Abner Gerardo, y otros. 2013. Sitio Web Pruebas de sistemas y pruebas de aceptación. *Sitio Web Pruebas de sistemas y pruebas de aceptación.* [En línea] 29 de junio de 2013. [Citado el: 25 de abril de 2016.] http://es.slideshare.net/abnergerardo/pruebas-de-sistemas-y-aceptacion-23663195.

Bibliografía

Abreu , Aldis Joan, y otros. 2012. *Generador dinámico de reportes.* 2012.

Agut, R,M. 2011. *Especificación de Requisitos Software según el estándar IEEE 830.* 2011.

Alegsa, Leandro. 2010. Sitio Web Alegsa.com. *Sitio Web Alegsa.com.* [En línea] 12 de 5 de 2010. [Citado el: 12 de 11 de 2015.] http://www.alegsa.com.ar/Dic/servidor%20de%20aplicaciones.php.

Apache Tomcat. 2015. Sitio web Apache Tomcat. *Sitio web Apache Tomcat.* [En línea] 2015. [Citado el: 12 de 11 de 2015.] http://tomcat.apache.org/.

Ariste, María Cecilia, y otros. *Un Marco de Trabajo para Enseñanza del paradigma MD.*

Aristegui, José Luis. 2010. *LOS CASOS DE PRUEBA EN LA PRUEBA DEL SOFTWARE.* 2010.

Barranco, Jesus. 2002. *Metodologías Del Análisis Estructurado del Sistema.* 2002.

Biosca, Enric. 2009. *Tutorial MDX.* 2009.

Blank, Isabel , Herrera, Larissa y Ortiz, Miguel. 2005. Sitio Web Pruebas Funcionales – Software II. *Sitio Web Pruebas Funcionales – Software II.* [En línea] Mayo de 2005. [Citado el: 9 de 4 de 2016.] http://carolina.terna.net/ingsw3/datos/Pruebas_Funcionales.pdf.

Booch , Grady , Jacobson , Ivar y Rumbaugh, Jim. 2000. *El Lenguaje Unificado de Modelado.* s.l. : Addinson-Wesley, 2000.

Cabrera , Yoan Manuel , Gonzalez , Yosbel Ernesto y Cruz , Claribel Lucy. 2008. *El Proceso de Prueba de del proyecto Sistema de Información para la Salud (SISalud).* 2008.

Cendejas. 2011. Sitio web eumed. *Sitio web eumed.* [En línea] 2011. [Citado el: 1 de 10 de 2015.] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm.

Craig, Larman. 2003. *UML y Patrones.* 2003.

Cuesta. 2005. Sitio web SG Buzz. *Sitio web SG Buzz.* [En línea] 11 de 2005. [Citado el: 4 de 10 de 2015.] https://sg.com.mx/revista/6/patrones-casos-uso.

Cuesta, Saúl. 2005. Sitio Web SG Buzz conocimiento para crear software grandioso. *Sitio Web SG Buzz conocimiento para crear software grandioso.* [En línea] 2005. [Citado el: 10 de 3 de 2016.] http://sg.com.mx/content/view/510.

Dataprix. 2007. Sitio Web Dataprix. [En línea] 2007. [Citado el: 1 de octubre de 2015.] http://www.dataprix.com/manual-mdx/ti/introduccion-consulta-lenguaje-multidimensional-mdx.

Diseño de consulta. 2015. Sitio web Diseño de consulta. *Sitio web Diseño de consulta.* [En línea] 27 de 9 de 2015. [Citado el: 4 de 11 de 2015.] https://help.libreoffice.org/Common/Query_Design/es.

Editor de consultas MDX para GDR v2.0. 2015. *Editor de consultas MDX para GDR v2.0.* La Habana : s.n., 2015.

Eguiluz, Javier. 2008. Sitio web Introducción a JavaScript. *Sitio web Introducción a JavaScript.* [En línea] 7 de 6 de 2008. [Citado el: 12 de 11 de 2015.] http://librosweb.es/libro/javascript/capitulo_1.html.

Gamma, Erich . 2011. *Design Patterns.* 2011.

gespro. 2015. gespro Suite de Gestión de Proyectos. *gespro Suite de Gestión de Proyectos.* [En línea] 2015. [Citado el: 4 de 9 de 2015.] https://gespro.datec.prod.uci.cu/login?back_url=https%3A%2F%2Fgespro.datec.prod.uci.cu%2F.

Gimson, Loraine, Gil, Daniel Gustavo y Rossi, Gustavo . 2012. *Metodologías ágiles y desarrollo basado en conocimiento.* 2012.

González, Héctor. 2016. Sitio Web Plataforma Virtual Educativa ITCA. *Sitio Web Plataforma Virtual Educativa ITCA.* [En línea] 2016. [Citado el: 3 de 5 de 2016.] http://virtual.itca.edu.sv/Mediadores/stis/43___visiones_interna_y_externa_de_las_pruebas.html.

Grosso, Andrés. 2011. sitio web Prácticas de software. *sitio web Prácticas de software.* [En línea] 21 de 3 de 2011. [Citado el: 26 de 1 de 2016.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

GUEVARA , JORGE EDUARDO y VALENCIA, JANETH DEL CARMEN . 2007. *DATA WAREHOUSE PARA EL ANÁLISIS ACADÉMICO DE LA ESCUELA POLITECNICA NACIONAL.* 2007.

Herramientas Case . 2011. Sitio web Herramientas Case . *Sitio web Herramientas Case .* [En línea] 2 de 4 de 2011. [Citado el: 12 de 11 de 2015.] http://fds-herramientascase.blogspot.com/.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison-Wesley, 2000.

Krall , César . 2016. Sitio Web apr. *Sitio Web apr.* [En línea] 2016. [Citado el: 1 de 1 de 2016.] http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=688:ique-es-y-para-que-sirve-uml-versiones-de-uml-lenguaje-unificado-de-modelado-tipos-de-diagramas-uml&catid=46:lenguajes-y-entornos&Itemid=163.

Lafosse, Jérome. 2012. *Expert IT Struts 2-El framework de desarrollo de aplicaciones JAVA EE.* 2012.

Lanzillotta, Analía. 2012. Sitio Web Master Magazine. [En línea] 2012. http://www.mastermagazine.info/termino/6841.php.

LibrosWeb. 2016. *Symfony 1.2, la guía definitiva Capítulo 2. El patrón MVC .* 2016.

Lopez, Jose Manuel. 2001. [En línea] 15 de 10 de 2001. [Citado el: 25 de 11 de 2015.] http://trevinca.ei.uvigo.es/~txapi/espanol/proyecto/superior/memoria/node21.html.

Macías, Luis Ernesto, Aluija, Víctor Emilio y Monteagudo, Yamila. 2016. *SISTEMA DE GESTIÓN Y GESTIÓN Y PROMOCIÓN DE PROYECTOS DE INVESTIGACIÓN Y DESARROLLO.* La Habana,Cuba : s.n., 2016.

Martinez , Rafael. 2013. Sitio web PostgreSQL-es. *Sitio web PostgreSQL-es.* [En línea] 2013. [Citado el: 12 de 11 de 2015.] http://www.postgresql.org.es/sobre_postgresql.

Martínez, Jorge Alfredo. 2007. Sitio Web Gestiopolis. [En línea] 2007. [Citado el: 20 de septiembre de 2015.] http://www.gestiopolis.com/olap-y-el-diseno-de-cubos/.

Microsoft. 2016. Sitio Web Microsoft. *Sitio Web Microsoft.* [En línea] 2016. [Citado el: 24 de 4 de 2016.] https://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

—. 2014. Sitio Web Microsoft . [En línea] 2014. https://msdn.microsoft.com/es-es/library/gg399173%28v=sql.120%29.aspx.

Mühlrad, Daniel. 2008. *Patrones de Diseño.* 2008.

netbeans. 2012. Sitio Web netbeans.org. *Sitio Web netbeans.org.* [En línea] 20 de 11 de 2012. [Citado el: 12 de 11 de 2015.] http://netbeans.org/community/releases/72/relnotes.html..

PATRONES DE DISEÑO. 2011. PATRONES DE DISEÑO. *PATRONES DE DISEÑO.* [En línea] 28 de 3 de 2011. [Citado el: 8 de 3 de 2016.]

php. 2015. Sitio web php. *Sitio web php.* [En línea] 2015. [Citado el: 12 de 11 de 2015.] http://php.net/manual/es/intro-whatis.php.

Potencier, Fabien y Zaninotto, Francois. 2009. *Symfony 1.4,la guía definitiva.* 2009.

Presman, Roger. 2001. *Ingeniería de Software, un enfoque práctico.* 2001.

Pressman, Roger S. 2012. *Ingeniería de Software:Un Enfoque Práctico.5ta edición.* 2012.

—. 2011. *Ingeniería de Software:Un Enfoque Práctico.6ta edición.* 2011.

Pressman, Roger S. 2010. *Software engineering: a practitioner's approach - 7th ed.* s.l. : Palgrave Macmillan, 2010.

Programación de Computadoras. 2011. *Programación de Computadoras.* 2011.

Rivera , Luis Alberto y Tixe , Juan Pablo. 2010. *Integración de la Metodología Agil XP el Estudio de de Windows Communication Foundation como Aternativa Metodológica para el Desasrrollo de Software Orientado a Servicios.* 2010.

Sabio, Gustavo y Pinciroli, Fernando. 2009. *Descripcion de CU.* 2009.

Salazar, Lianet Labrada. 2012. *Administrador de Reportes para la versión 2.0 del Generador Dinámico de Reportes.* 2012.

Sánchez, Sureny Orihuela y López, Liandry Monteslier. 2013. *Componente de conexión para la generación de reportes mediante un servicio web en el Generador Dinámico de Reportes.* La Habana : s.n., 2013.

Scielo. 2013. Sitio web Scielo. *Sitio web Scielo.* [En línea] 2013. [Citado el: 1 de 4 de 2016.] http://www.scielo.cl/scielo.php?pid=S0718-07642013000300012&script=sci_arttext.

Sencha ExtJS. 2015. *Sencha ExtJS.* 2015.

Sinnexus. 2015. Sitio Web Sinnexus. [En línea] 2015. [Citado el: 1 de octubre de 2015.] http://www.sinnexus.com/business_intelligence/olap_vs_oltp.aspx.

Software. 2013. Sitio web Software. *Sitio web Software.* [En línea] 11 de 12 de 2013. [Citado el: 12 de 11 de 2015.] http://www.software.com.ar/visual-paradigm-para-uml.html..

Sommerville, Ian. 2011. *Ingeniería de Software:9na Edición.* México : s.n., 2011.

—. 2011. *Ingeniería del software.7ma Edicion.* 2011.

Sosa, Dailyn. 2013. Sitio Web Eumed.net. *Sitio Web Eumed.net.* [En línea] 2013. [Citado el: 10 de 3 de 2016.] http://www.eumed.net/libros-gratis/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm.

SourceMaking. 2013. Sitio web SourceMaking:Mediator Design Pattern. *Sitio web SourceMaking:Mediator Design Pattern.* [En línea] 2013. [Citado el: 9 de 3 de 2016.] https://sourcemaking.com/design_patterns/mediator.

Svitla Systems. 2016. Sitio web Methods & Tools Software Development Magazine. *Sitio web Methods & Tools Software Development Magazine.* [En línea] 2016. [Citado el: 4 de 5 de 2016.] http://www.methodsandtools.com/archive/archive.php?id=24.

Ubuntu. 2014. Sitio web Ubuntu documentation. *Sitio web Ubuntu documentation.* [En línea] 2014. [Citado el: 12 de 11 de 2015.] https://help.ubuntu.com/lts/serverguide/httpd.html.

UML. 2012. sitio web Modelado de Sistemas con UML. *sitio web Modelado de Sistemas con UML.* [En línea] 2012. [Citado el: 3 de 10 de 2015.] http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/doc-modelado-sistemas-uml.pdf.

Unad. 2006. Sitio Web Unad. *Sitio Web Unad.* [En línea] 9 de 2006. [Citado el: 16 de 3 de 2016.] http://datateca.unad.edu.co/contenidos/200609/exeuml/leccin_39_diagrama_de_clases_de_diseo.html.

Valdez, Abner Gerardo, y otros. 2013. Sitio Web Pruebas de sistemas y pruebas de aceptación. *Sitio Web Pruebas de sistemas y pruebas de aceptación.* [En línea] 29 de junio de 2013. [Citado el: 25 de abril de 2016.] http://es.slideshare.net/abnergerardo/pruebas-de-sistemas-y-aceptacion-23663195.

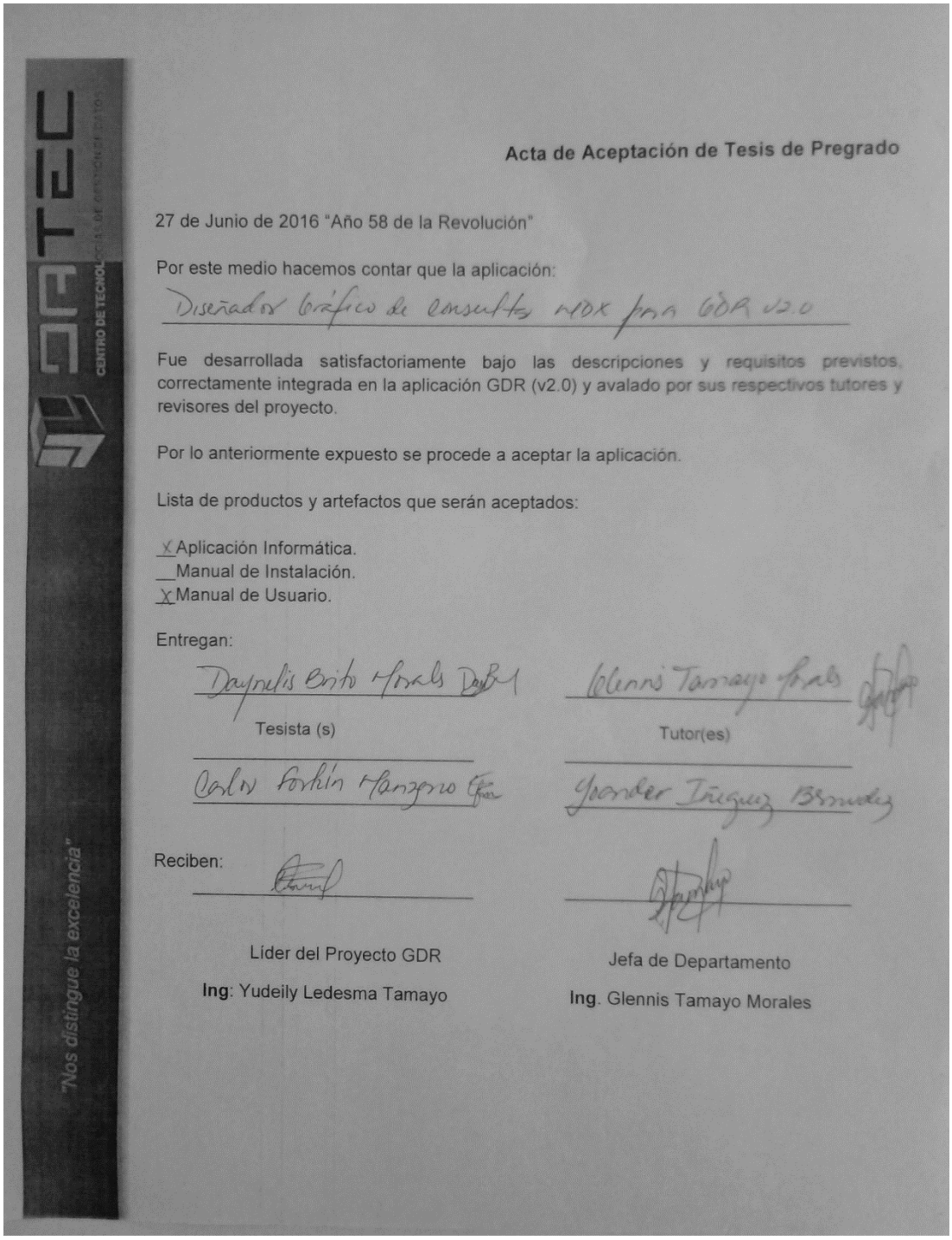
Anexos

Anexo 1 Preguntas realizadas al cliente como parte de la entrevista no estructurada con el objetivo de recopilar información referente a la realización del Diseñador de consultas MDX para GDR v2.0.

Entrevistado:

1. ¿Cuáles son las principales deficiencias que existen en el Editor de consulta MDX de GDR v2.0?
2. ¿Cuáles son las principales funcionalidades que deben ser implementadas en el diseñador gráfico?
3. ¿Cuáles son los requerimientos de software que necesita GDR v2.0 tener instalados para llevar a cabo la implementación del diseñador gráfico?
4. ¿Cuáles son los requerimientos de hardware que necesita GDR v2.0 tener instalados para llevar a cabo la implementación del diseñador gráfico?

Anexo 2. Carta de Aceptación



CENTRO DE TECNOLOGÍAS DE AVANZADA

"Nos distingue la excelencia"

Acta de Aceptación de Tesis de Pregrado

27 de Junio de 2016 "Año 58 de la Revolución"

Por este medio hacemos contar que la aplicación:

Diseñador gráfico de consultas MDX para GDR v2.0

Fue desarrollada satisfactoriamente bajo las descripciones y requisitos previstos, correctamente integrada en la aplicación GDR (v2.0) y avalado por sus respectivos tutores y revisores del proyecto.

Por lo anteriormente expuesto se procede a aceptar la aplicación.

Lista de productos y artefactos que serán aceptados:

X Aplicación Informática.
__Manual de Instalación.
X Manual de Usuario.

Entregan:

Daynelis Brito Morales — Tesista (s)

Glennis Tamayo Morales — Tutor(es)

Carlos Yorkin Manzano Cruz

Yoander Iñiguez Bermudez

Reciben:

Líder del Proyecto GDR

Ing: Yudeily Ledesma Tamayo

Jefa de Departamento

Ing. Glennis Tamayo Morales