Universidad de las Ciencias Informáticas

Facultad 6

Título: “Editor 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: Sonia Hidalgo García.

Yoan Gainza Labrada.

Tutores: Ing. Miguel Lezcano Ramos.

Ing. Mirna Martínez Labrador.

Co-Tutor: Ing. Adrián Quintero Henríquez.

La Habana, 2015
“Año 57 de la Revolución”

"La idea es convertir la informática en una de las ramas más productivas y aportadoras de recursos para la nación. Es el empleo a fondo de la inteligencia y del capital humano que tenemos y principalmente del que podemos crear casi como espina dorsal de la economía"

Fidel Castro Ruz

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 _____.

Sonia Hidalgo García **Yoan Gainza Labrada**

_____ _____

Firma del Autor Firma del Autor

Ing. Miguel Lezcano Ramos **Ing. Mirna Martínez Labrador**

_____ _____

Firma del Tutor Firma del Tutor

Ing. Adrián Quintero Henríquez

_____

Firma del Co-Tutor

Datos de contacto

Tutores:

Ing. Miguel Lezcano Ramos.

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

E-mail: mlezcano@uci.cu.

Ing. Mirna Martínez Labrador.

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

E-mail: mirnaml@uci.cu.

Ing. Adrián Quintero Henríquez.

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

E-mail: aquintero@uci.cu.

Agradecimientos de Sonia

A mi mamá por todo su apoyo y cariño sobre todo en esta etapa tan difícil de mi vida y por siempre confiar en mí y escuchar de ella las más hermosas palabra de aliento porque para ella todo siempre iba a salir bien, aunque no supiera por lo yo estaba pasando.

A mi papá por estar siempre preocupado por mí, por su confianza, y por su sacrificio para ayudarme durante estos 5 años.

A mis hermanos Rene, Julito y Yoan por todo su cariño y momentos felices compartidos. A toda mi familia por la preocupación y alentarme para convertirme en la primera ingeniera de la familia.

A mi novio Alain por brindarme todo su amor estos 5 años en la UCI, por ser tan paciente conmigo y darme las fuerzas cuando pensé que todo estaba perdido, por la comprensión y por ser la persona más lindas que conocí en la universidad, Te Quiero.

A mis suegros por su dulzura y hacerme sentir parte de la familia tan hermosa que tienen.

A mi dúo de tesis Yoan por todos los momentos difíciles que pasamos juntos, por su total entrega y dedicación en este sueño y porque sin ti no hubiera sido posible.

A mis tutores Miguel, Mirna y Adrián por estar siempre pendiente de todos los detalles, por sus consejos, por el apoyo y por la guía.

A Arian por ser tan buen amigo y siempre estar ahí cuando lo necesité incondicionalmente, un millón de gracias para ti.

A mis compañeras de apartamento que se han convertido en hermanas para mí, haciéndome sonreír ante los problemas Jessie, Glennis, Laritza, Yailin, Yanet y Arianna. A otras grandes amigas fuera de la universidad que nunca han dejado de preocuparse en especial para Marian y Helen. A mis amigos del alma Yasel, Frank, Guillermo, Carlos, Marcos por estar siempre.

Agradecimientos de Yoan

Quiero empezar agradeciendo mi familia, porque todos pusieron al menos un pequeño grano de arena para que esté hoy aquí justamente donde estoy. Por supuesto agradecer a mi mamá que se las ha arreglado para ser una luz extremadamente positiva que me ha alumbrado el camino durante cada día. Agradecer a mi papá que siempre ha estado ahí, que es mi modelo a seguir, mi orgullo, en incontables veces mi guía. A mi hermana que como hermana mayor ha cumplido con creces su papel y me ha ayudado durante todo el trayecto a ser positivo y emprendedor.

A mis abuelitas Nenita y Cachita por su preocupación y amor incondicional, a mis abuelos Amador y Pajulio.

A mi tía Amadita, a mi primo Jorgito, a mi tía Carmen que debe estar comiéndose las uñas y preocupada por mí, a mis primos de Baracoa, a mis primas Milena y Melina que saben que las quiero y sé que me quieren.

A mis tutores que han sabido aconsejarnos y guiarnos durante todo el proceso para lograr el mejor resultado posible.

Agradecer desde el centro de la Tierra hasta las estrellas a mi dúo de tesis Sonia, que no solo ha formado parte de todo este proceso, sino que desde los inicios de la universidad ha sido una buena amiga, a ella porque hemos pasado por duros momentos y sin embargo aquí estamos, uno al lado del otro y mirando al futuro.

Quiero agradecer también a todos mis amigos y amistades que ellos saben quiénes son. A todos ellos que primeramente vivieron en el edificio 111, los que luego vivieron en el 95, los que llegamos hasta el 106, que hemos pasado todo tipo de stress, problemas, que hemos compartido, que nos hemos divertido por todo lo alto, que nos hemos aburrido de manera indescriptible.

Dedicatoria

Le dedico mi trabajo a mis padres por ser la inspiración siempre, por su apoyo y ayuda incondicional, ser los padres más cariñosos y especiales del mundo, por confiar tanto en mí y estar seguros que no los defraudaría. Los quiero mucho y espero estén muy orgullosos de mi porque gran parte de todo este sacrificio por 5 años fue pensando en ustedes.

Sonia

Dedicatoria

A mi mamá por ser ella misma, por aconsejarme y estar ahí para mí, por ser parte de mi realidad, por quererme, mimarme. A ti por ser mi luz, mi conciencia, por ser uno de los eslabones más fuertes y esenciales de mi propia persona, por ser parte y todo, por conquistar el mundo solo para regalármelo, por hacer y deshacer futuros solo para mí. Te amo.

A mi papá por ser un ejemplo, por quererme porque si, por ser fiel a mis causas y mis sueños, por resaltar mis virtudes y de vez en cuando recordarme mis defectos, por estar ahí. A ti porque siempre me guías, me animas en mis sueños, me muestras los caminos, por conspirar junto a mi mamá para conseguirme un mundo mejor, por esperarme los fines de semana, por preocuparte, por criarme, por quererme. Te amo.

A mi hermana que es parte clave de quien soy y quien represento para el mundo, a ella que contribuye a construirme poco a poco, a ella por tenerme siempre en su mente, por ser ella misma, por ser una de las piezas centrales del rompecabezas que compone mi mundo. A ti porque te quiero, porque somos los mejores hermanos del mundo, porque me apoyas en mis sueños, me sonríes en días que necesito sonrisas y me animas en días tristes. Te amo.

Yoan

RESUMEN

En la Universidad de las Ciencias Informáticas se trabaja en el desarrollo de un conjunto de proyectos destinados a la informatización de los principales procesos de la sociedad cubana. El Generador Dinámico de Reportes (GDR) implementado por el Centro de Tecnologías de Gestión de Datos (DATEC), ofrece una solución para la generación de reportes destinada a mejorar la rapidez y calidad de las decisiones a tomar en todos los niveles de una organización. El GDR actualmente carece del soporte OLAP necesario para realizar reportes provenientes de orígenes de datos multidimensionales. El presente trabajo de diploma tiene como objetivo principal desarrollar un editor de consultas MDX que permita la generación de reportes OLAP en el Generador Dinámico de Reportes v2.0. Para estructurar el proceso de desarrollo, se seleccionó la metodología de desarrollo Open Up. Además se modela el diseño siguiendo el patrón arquitectónico Modelo-Vista-Controlador manteniendo la arquitectura que presenta el sistema. Para atender las peticiones del lado del servidor se utilizó el marco de trabajo Symfony2 y para el lado del cliente ExtJS. Se realizaron pruebas al editor, con el objetivo de probar la corrección de sus funcionalidades y su calidad. El editor de consultas MDX para el Generador Dinámico de Reportes v2.0 brinda el soporte OLAP requerido y permite realizar consultas a los modelos creados a partir de los orígenes de datos multidimensionales.

Palabras claves

Consultas, editor, GDR, MDX, multidimensionales, OLAP, reportes

ABSTRACT

The University of Information Science is working on the development of a set of projects for the computerization of the main processes of Cuban society. The Dynamic Report Generator (GDR) implemented by the Center for Data Management Technologies (DATEC) offers a reporting solution to improve the speed and quality of decisions to make at all levels of an organization. The GDR currently lacks the support needed to perform OLAP reports from multidimensional data sources. This diploma thesis main objective is to develop an MDX query editor that allows viewing OLAP reports from Dynamic Report Generator v2.0. To structure the development process, development methodology Open up was selected. Besides the Model-View-Controller architectural pattern was selected, maintaining the system architecture. To attempt the requests on the server side Symfony2 framework was used and for the client side ExtJS. Tests were performed to the editor, in order to check the correctness of its functionality and its quality. The MDX query editor for Dynamic Report Generator v2.0 provides the required support OLAP and allows queries to models created from multidimensional data sources.

Key words:

Editor, GDR, MDX, multidimensional, OLAP, query, reports

ÍNDICE

Índice de Figuras

Índice de tablas

INTRODUCCIÓN

En el mundo actual, a raíz del surgimiento y avance acelerado de las tecnologías, la información se ha convertido en uno de los eslabones más importantes para el desarrollo de las empresas a nivel mundial. La conducta innovadora en el ámbito político, económico, cultural y tecnológico ha originado que el uso de la información tenga mayor repercusión en el progreso de las empresas. Su manejo eficiente permite, apoyándose en experiencias pasadas realizar pronósticos que contribuyan a tomar decisiones adecuadas.

El cúmulo de datos que se genera en las empresas por su actividad de negocio crece de forma exponencial. Esa información es almacenada tanto en bases de datos de las aplicaciones de negocio como en archivos de múltiples formatos. Las bases de datos multidimensionales que funcionan a través de Aplicaciones de Procesamiento Analítico en Línea OLAP (por sus siglas en inglés *On-Line Analytical Processing*), se han convertido en una de las tendencias tecnológicas más usadas por las empresas. Las principales ventajas de este tipo de bases de datos son la versatilidad para cruzar información y la alta velocidad de respuesta donde el análisis de los datos resulta crucial. Representa una solución para el desarrollo de la inteligencia de negocios, las herramientas OLAP proporcionan un sistema confiable para procesar datos que luego serán utilizados para llevar a cabo análisis e informes.

El país se encuentra enfrascado en el proceso de informatización de los principales sectores económicos, para una mayor agilidad de todas sus funciones. La Universidad de las Ciencias Informáticas (UCI) juega un papel muy importante en la producción de software a escala nacional, fomentando el avance en el camino de la independencia tecnológica. Cuenta con el Centro de Tecnologías de Gestión de Datos (DATEC) especializado en el almacenamiento y análisis de datos donde se desarrolla el Generador Dinámico de Reportes (GDR), uno de los proyectos cuya finalidad es garantizar la generación de reportes desde los gestores de base de datos: PostgreSQL, MySQL, SQLite, Oracle y Microsoft SQL Server. El GDR, es una aplicación web que permite al usuario realizar consultas a bases de datos y obtener información de ella en forma de reporte. El sistema tiene como objetivo principal facilitar a los usuarios abstraerse de conocimientos relacionados con los gestores de bases de datos y generar reportes en varios formatos. (Generador dinámico de reportes, 2012)

Actualmente el GDR carece del soporte requerido para realizar consultas MDX. Clientes como la Oficina Nacional de Estadística e Información (ONEI), la Aduana General de la República de Cuba y la Contraloría General de la República de Cuba (CGR) que utilizan el sistema manejan enormes volúmenes de datos

contenidos en Almacenes de Datos (AD)[1]. Para el análisis multidimensional de estos datos se requiere el uso de la herramienta Pentaho Bi-Server[2] que permite realizar consultas multidimensionales y generar reportes. Crear los reportes multidimensionales de esta forma representa un mayor consumo de tiempo, requiere la instalación y configuración de una aplicación externa al sistema. Además impone mayores prestaciones de hardware y es indispensable el dominio preciso de dicha herramienta para realizar estas funciones.

Dada la situación problemática anteriormente planteada, se define como ***problema científico:*** ¿cómo garantizar la gestión de consultas MDX para reportes OLAP desde el Generador Dinámico de Reportes v2.0?

El problema científico se enmarca en el ***objeto de estudio:*** gestión de consultas MDX y se determina como ***campo de acción:*** gestión de consultas MDX en los generadores de reportes.

Para dar solución al problema planteado se define como ***objetivo general:*** desarrollar un editor de consultas MDX que permita la generación de reportes OLAP en el Generador Dinámico de Reportes v2.0.

En correspondencia con el objetivo general se precisan los siguientes **objetivos específicos**:

1. Analizar los conceptos asociados al campo de acción, selección de herramientas, tecnologías y metodología necesaria en el desarrollo del editor de consultas MDX para el Generador Dinámico de Reportes v2.0.
2. Realizar el análisis y diseño del editor de consultas MDX para el Generador Dinámico de Reportes v2.0.
3. Realizar la implementación y pruebas del editor de consultas MDX para el Generador Dinámico de Reportes v2.0.

Tareas de la investigación:

1. Análisis de los conceptos básicos y técnicos implicados en el desarrollo del editor de consultas MDX para elaborar el marco teórico de la investigación.
2. Selección de la metodología, herramientas y tecnologías a utilizar en el desarrollo del editor de consultas MDX.

[1] AD: es una colección de datos orientada a temas o materias, integrada, variable en el tiempo y no volátil.

[2] Pentaho: herramienta open Source, provee una alternativa de soluciones de inteligencia de negocio.

3. Identificación de las necesidades funcionales y tecnológicas del editor de consultas MDX.
4. Identificación de los principios arquitectónicos del editor de consultas MDX para definir su estructura global.
5. Identificación de los principios de diseño y funcionamiento del editor de consultas MDX para guiar el proceso de implementación.
6. Implementación de los principios de diseño y funcionamiento para satisfacer las necesidades funcionales y tecnológicas del editor de consultas MDX.
7. Realización de las pruebas de software para comprobar el correcto funcionamiento del editor de consultas MDX.

Preguntas científicas

1. ¿Cuáles son los fundamentos teóricos necesarios en el desarrollo del editor de consultas MDX?
2. ¿Cuáles son las funcionalidades que debe cumplir el editor para que permita la generación de consultas MDX?
3. ¿Cómo se debe estructurar el proceso de desarrollo del editor de consultas MDX para que permita la gestión de consultas MDX en el GDR v2.0?
4. ¿Cómo garantizar la obtención de los datos y el correcto funcionamiento del editor de consultas MDX?

Métodos de la Investigación

Métodos teóricos:

Analítico–Sintético: Se utiliza en el estudio y análisis de diferentes fuentes bibliográficas consultas con el objetivo de realizar una amplia investigación sobre el análisis multidimensional de los datos y el lenguaje MDX. Luego se sintetiza el contenido analizado logrando plasmar las ideas fundamentales.

Modelación: Empleado en la etapa de diseño, permite modelar diagramas para crear abstracciones que representarán el proceso de desarrollo y a través de un lenguaje de modelado desarrollar un modelo del editor de consultas MDX.

Resultados esperados.

Se espera obtener un editor de consultas MDX integrado al Generador Dinámico de Reportes v2.0 que permita la edición de consultas multidimensionales a partir de la selección de un modelo. Dicho modelo es creado con anterioridad proveniente de la fuente de datos OLAP elegida. Posteriormente se hace posible la generación de reportes multidimensionales con el objetivo de satisfacer las necesidades de los clientes que requieran el uso de esta solución como apoyo a la toma de decisiones de su empresa.

El Trabajo de Diploma está estructurado por tres capítulos de la siguiente manera:

Capítulo 1: Fundamentación teórica.

En este capítulo se realiza un análisis de los principales sistemas generadores de reportes existentes, haciendo énfasis en los editores de consulta MDX que tienen incorporados cada uno. Se hace un estudio de los conceptos fundamentales manejados durante el desarrollo de la investigación. Además se define la metodología de software que guiará el proceso de desarrollo, las distintas herramientas y tecnologías a emplear, con el objetivo de explotar al máximo sus potencialidades.

Capítulo 2: Análisis y diseño de la solución.

En este capítulo se comienza con la modelación del modelo de dominio y se hace un desglose de los requisitos funcionales y no funcionales con que cumplirá la aplicación. Además de la realización del diagrama de casos de uso del sistema y clases del diseño, a través de los cuales se muestra la estructura estática del sistema.

Capítulo 3: Implementación y pruebas.

En el capítulo se describen las fases de implementación y prueba del sistema. Se modela el diagrama de componente que detalla la forma en que estará estructurado el editor de consultas, reflejando la transformación de los elementos del modelo del diseño en términos de componentes. Además se diseñan y aplican las pruebas de software con el objetivo de validar el correcto funcionamiento del editor.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se define el marco teórico de la investigación. Se realiza un estudio de los conceptos fundamentales manejados durante el desarrollo de la investigación, siendo de vital importancia la claridad de su significado. Además se expone un análisis de las herramientas OLAP integradas a los generadores de reportes para la gestión de consultas MDX. Se define la metodología de software que guiará el proceso de desarrollo, las distintas herramientas y tecnologías a emplear, con el objetivo de explotar al máximo sus potencialidades.

1.1. Conceptos asociados a la investigación

1.1.1. OLAP

OLAP (Procesamiento Analítico en Línea) es una tecnología de gran alcance para la búsqueda de datos, incluyendo las capacidades de visualización de informes sin límites y cálculos analíticos complejos. OLAP realiza análisis multidimensional de datos de negocio y proporciona la capacidad para realizar cálculos complejos, análisis de tendencias y sofisticado modelado de datos. Es la tecnología detrás de muchas aplicaciones de BI (Inteligencia de negocio), permitiendo a los usuarios finales realizar el análisis de los datos en múltiples dimensiones, proporcionando así el conocimiento y la comprensión que necesitan para tomar mejores decisiones. (Lanzillotta, 2012)

Tradicionalmente, los sistemas OLAP se clasifican según el tipo de motor en el que se almacenan los datos en las siguientes categorías:

Sistemas MOLAP: La arquitectura MOLAP usa las bases de datos multidimensionales para proporcionar el análisis, su principal premisa es que el OLAP está mejor implantado almacenando los datos multidimensionalmente. El sistema utiliza una arquitectura de dos niveles: la base de datos multidimensional y el motor analítico. La base de datos multidimensional es la encargada del manejo, acceso y obtención del dato. El nivel de aplicación es el responsable de la ejecución de los requerimientos OLAP y el nivel de presentación se integra con el de aplicación y proporciona una interfaz a través de la cual los usuarios finales visualizan los análisis OLAP. Una arquitectura cliente/servidor permite a varios usuarios acceder a la misma base de datos multidimensional. (Sinnexus, 2012)

Sistemas ROLAP La arquitectura ROLAP, accede a los datos almacenados en un almacén de datos para proporcionar los análisis OLAP. La premisa de los sistemas ROLAP es que las capacidades OLAP se soportan mejor contra las bases de datos relacionales. El sistema ROLAP utiliza una arquitectura de tres niveles. El nivel de base de datos usa bases de datos relacionales para el manejo, acceso y obtención del dato. El nivel de aplicación es el motor que ejecuta las consultas multidimensionales de los usuarios. El motor ROLAP se integra con niveles de presentación, a través de los cuáles los usuarios realizan los análisis OLAP. (Sinnexus, 2012)

Sistemas HOLAP: La solución OLAP híbrida (HOLAP) combina las arquitecturas ROLAP y MOLAP para brindar una solución con las mejores características de ambas: desempeño superior y gran escalabilidad. Un tipo de HOLAP mantiene los registros de detalle (los volúmenes más grandes) en la base de datos relacional, mientras que mantiene las agregaciones en un almacén MOLAP separado. (Sinnexus, 2012)

1.1.2. Cubo OLAP

Un cubo OLAP es una estructura de datos que supera las limitaciones de las bases de datos relacionales y proporciona un análisis rápido de los datos. Los cubos pueden mostrar y sumar grandes volúmenes de datos, a la vez que proporcionan a los usuarios acceso mediante búsqueda a los puntos de datos. De este modo, los datos se pueden resumir o reorganizar según sea necesario, para procesar la variedad más amplia de consultas pertinentes al área de interés del usuario. Las herramientas de soluciones para sistemas de inteligencia de negocio han avanzado notablemente en cuanto a las prestaciones que estas aplicaciones brindan a las empresas, donde la información confiable, precisa y en el momento oportuno, son uno de los bienes más preciados. Una base de datos multidimensional puede contener varios cubos que extenderán las posibilidades del sistema OLAP con el cual se trabaja. (Sinnexus, 2012).

Los cubos OLAP se estructuran de la siguiente manera:

Dimensiones: Las dimensiones son categorías descriptivas por las cuales los datos numéricos en un cubo son separados para su análisis. Una dimensión puede ser creada para usarse en un cubo individual o en múltiples cubos, para un cubo individual, es llamada dimensión privada. Por el contrario si esta puede ser usada por múltiples cubos, se le llama dimensión compartida. Estas podrán ser usadas dentro de todo cubo, en la base de datos, así se optimiza el tiempo y se evita duplicar dimensiones privadas. Las dimensiones compartidas, también habilitan la estandarización de las métricas de negocios entre cubos.

Medidas: Las medidas son los datos numéricos de interés primario para los usuarios. Son usadas por el procedimiento de agregación de los servicios de OLAP y almacenadas para su rápida respuesta a las

peticiones de los usuarios. Se puede crear una medida calculada y computar miembros de dimensiones, combinando expresiones multidimensionales (MDX), fórmulas matemáticas y funciones definidas por el usuario. Esta facilidad, permite definir nuevas medidas y miembros de dimensión, basados sobre una sintaxis de fórmulas sencillas. (Sinnexus, 2012)

1.1.3. Lenguaje MDX

MDX (*Multidimensional Expression*). A través del lenguaje MDX es posible explotar la información que reside en los motores OLAP y satisfacer peticiones de consultas analíticas. La principal diferencia de OLAP respecto al lenguaje relacional radica en que las estructuras dimensionales están jerarquizadas y se representan en forma de árbol y por lo tanto existen relaciones entre los diferentes miembros de las dimensiones. Al ejecutar una consulta de este tipo se devuelve un conjunto de celdas que es el resultado de tomar un subconjunto de las celdas del cubo original.

La sintaxis básica es:

SELECT <especificación de eje> on columns, <especificación de eje> on rows

FROM <especificación de cubo>

WHERE <especificación Slicer (rebanador)> (Restringe los resultados a cierta rebanada del cubo)

1.1.4. Analogía con SQL

Al igual que en SQL, en MDX se pueden utilizar expresiones para manipular los datos, ordenarlos, filtrarlos, agruparlos y realizar cálculos con ellos, dichas expresiones deben escribirse con la sintaxis del lenguaje MDX. Analizando la sintaxis del lenguaje MDX es posible establecer una analogía con el SQL. En la estructura general de la consulta MDX las cláusulas SELECT, FROM, WHERE se usan idénticamente que en el SQL, especificando en el SELECT el conjunto de elementos que se deseen visualizar, en el FROM se indica el cubo del que se extraen los datos y en el WHERE las condiciones de filtrado. En la cláusula SELECT se usa la cláusula on columns para separar los elementos a visualizar en las filas de los elementos de las columnas.

1.1.5. Principales funciones, opciones y elementos MDX

A nivel de funcionalidades y potencia cuando se realizan consultas, el MDX es potente como el SQL aunque su objetivo está orientado a temas de comparaciones y relaciones jerárquicas entre elementos. Una de las funcionalidades que distinguen al MDX es el acceso a los elementos utilizando una estructura de árbol. Así para un determinado nivel de una dimensión se tienen comandos como:

[Familia].[X].CurrentMember: Permite acceder al miembro actual.

[Familia].[X].Children: Permite acceder a los hijos de una familia.

[Familia].[X].prevMember: Permite acceder al miembro anterior de la dimensión.

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

CrossJoin (conjunto_a, conjunto_b): Obtiene el producto cartesiano de dos conjuntos de datos. **BottomCount** (conjunto_datos, N): Obtener un número determinado de elementos de un conjunto, empezando por abajo, opcionalmente ordenado.

BottomSum (conjunto_datos, N, S): Obtener de un conjunto ordenado los N elementos cuyo total es como mínimo el especificado(S).

Except (conjunto_a, conjunto_b): Obtener la diferencia entre dos conjuntos.

Además de las funcionalidades expuestas, en MDX también están implementadas funciones matemáticas y estadísticas que permiten enriquecer los análisis como: AVG, COUNT, VARIANCE y todas las funciones trigonométricas (Seno, Coseno, Tangente, entre otras).

Las funcionalidades más destacadas del lenguaje MDX son las funciones especiales de tratamiento del tiempo. En el análisis multidimensional es muy recurrente comparar los valores de las métricas con los valores de periodos anteriores. También es común ejecutar operaciones acumuladas con periodos móviles, es decir de una fecha a otra.

El lenguaje MDX permite identificar y tratar de forma especial las dimensiones temporales, para poder aplicar las fórmulas que permiten obtener los cálculos mencionados. Entre sus principales fórmulas relacionadas con las dimensiones temporales y su sintaxis se encuentran:

PeriodsToDate

Funcionalidad: Devuelve un conjunto de miembros del mismo nivel que un miembro determinado, empezando por el primer miembro del mismo nivel y acabando con el miembro en cuestión, de acuerdo con la restricción del nivel especificado en la dimensión de tiempo.

Sintaxis: <Conjunto> PeriodsToDate (<Nivel>, <Miembro>)

Existen funciones especiales para simplificar el uso de PeriodsToDate en sus versiones más comunes y son las siguientes:

WTD (<Miembro>): Devuelve los miembros de la misma semana del miembro especificado.

MTD (<Miembro>): Devuelve los miembros del mismo mes del miembro especificado.

QTY (<Miembro>): Devuelve los miembros del mismo trimestre del miembro especificado.

YTD (<Miembro>): Devuelve los miembros del mismo año del miembro especificado.

ParallelPeriod

Funcionalidad: Devuelve un miembro de un periodo anterior en la misma posición relativa que el miembro especificado.

Sintaxis: <Miembro> ParallelPeriod (<Nivel>, <Expresión numérica>, <Miembro>)

1.2. Gestión de consultas MDX en Generadores de Reportes.

Los generadores de reportes son herramientas de apoyo a los sistemas de información, que permiten a los usuarios acceder de la manera más simple y rápida a los datos o bases de datos guardados en forma de reportes. Para el trabajo con bases de datos multidimensionales, los generadores de reportes han integrado entre sus funcionalidades herramientas OLAP que permitan la gestión de consultas MDX.

1.2.1. Gestión de consultas MDX en la herramienta iReport Designer

Para la creación de consultas MDX, el iReport está integrado con Rex (Explorador de Almacenes), un diseñador visual de código abierto para consultas MDX. Rex requiere una conexión XML/A para trabajar, por lo que no puede ser usado directamente con una conexión Mondrian. Rex es un constructor visual de consultas MDX, permite explorar las dimensiones brindadas por el cubo OLAP, y arrastrar las medidas deseadas hacia el centro de la pantalla. Se puede probar una consulta, ver el resultado, y cuando esté listo llevarlo al iReport para usarlo en el reporte deseado.(Herrera 2005)

1.2.2. Gestión de consultas MDX en la herramienta Crystal Reports

Crystal Reports es una herramienta de diseño permite elaborar informes a partir de datos de SAP. El controlador MDX Query brinda la posibilidad de, a partir de consultas que contienen variables de jerarquía y variables de nodo de jerarquía crear campos específicos que permiten listar selecciones para las variables en Crystal Reports. Iniciando el Query Designer de SAP desde Crystal Reports, la consulta que se defina se seleccionará automáticamente como el origen de datos de un nuevo informe de Crystal Query Designer mostrando automáticamente en el modo plano (un modo de visualización recomendado para

diseñar informes de Crystal con el controlador BW Query). Se define la consulta seleccionando las medidas y dimensiones, arrastrándolas al área de consultas. (SAP Business Objects Enterprise , 2010)

1.2.3. Gestión de consultas MDX en la herramienta Pentaho

La herramienta Pentaho Reporting utiliza el conjunto de librerías JPivot que permite explorar cubos OLAP y mostrar esa información en forma de tablas y gráficos, soportando la funcionalidad típica de los entornos OLAP. Como servidor OLAP utiliza Mondrian, siendo quien recibe las solicitudes de información de JPivot y realiza las consultas a la base de datos, devolviendo la información solicitada. Luego JPivot utiliza dichos resultados para construir una página HTML que permita visualizar la información. La herramienta posee un Navegador OLAP que realiza una consulta personalizada, bridando las siguientes opciones: pasar a filtro, pasar a columna, cambiar el orden de las dimensiones en el eje, configurar dimensión/medida entre otras. Además permite mostrar y configurar el gráfico correspondiente a la vista generada por el navegador OLAP. Para la edición de consultas MDX de JPivot permite introducir la consulta en el Editor MDX y al ejecutarla, es posible ver la tabla JPivot con el resultado correspondiente. (Glenda, 2011)

El estudio realizado a las herramientas OLAP integradas a los generadores de reportes para la gestión de consultas MDX posibilitó el entendimiento de su funcionamiento, a través de qué bibliotecas gestionan los datos necesarios para ejecutar y validar las consultas, la cadena de conexión requerida para efectuar la conexión a la base de datos seleccionada, para luego iniciar la fase de implementación del editor de consultas MDX.

1.3. Metodología de desarrollo de software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Las metodologías se dividen en dos grandes grupos, las pesadas o tradicionales y las ágiles. Las metodologías tradicionales se basan en la idea de que el éxito del producto se puede lograr si se tiene todo correctamente documentado, mientras las ágiles defienden la idea de que el proceso de desarrollo del software, se centra en el software y no en la documentación alrededor de este, por lo que le da mayor importancia a la programación que a la documentación, aunque no la obvia por completo, sino que toma en cuenta sólo la documentación necesaria y de forma muy sencilla.

Atendiendo a las necesidades del editor de consultas MDX de enfocar el proceso de desarrollo hacia obtener un resultado funcional y no una documentación detallada, por el período de tiempo con que se dispone y la cantidad de miembros en el equipo, se decide utilizar una metodología ágil. Se selecciona Open Up para guiar el desarrollo del editor de consultas MDX para el GDR v2.0, ya que se ajusta al ambiente que se presenta, se centra en grupos pequeños de trabajo, tiempo limitado, genera una cantidad mínima de artefactos y además es la metodología de desarrollo definida por el equipo del GDR.

1.3.1. Open Up

Open Up es un proceso unificado ágil y liviano, que aplica un enfoque iterativo e incremental dentro de un ciclo de vida estructurado y contiene un conjunto mínimo de prácticas que ayuda al equipo a ser más efectivo desarrollando software. Mantiene las características esenciales de RUP, en el cual se incluyen las siguientes:

- ✓ Desarrollo incremental.
- ✓ Uso de casos de uso y escenarios.
- ✓ Manejo de riesgos.
- ✓ Diseño basado en la arquitectura.

Su proceso puede ser personalizado y extendido para distintas necesidades, que aparecen a lo largo del ciclo de vida del desarrollo de software, dado que su modelo de desarrollo es incremental iterativo.

El esfuerzo personal en un proyecto Open Up se organiza en micro-incrementos. Estas representan unidades cortas de trabajo que producen un ritmo estable y medible de progreso del proyecto (generalmente medido en horas o unos pocos días). Este proceso aplica colaboración intensiva a medida que el sistema se va desarrollando incrementalmente por un equipo comprometido y auto-organizado.

Open UP es un proceso iterativo con iteraciones distribuidas en las siguientes cuatro fases:

- ✓ **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. Se debe elaborar un plan de proyecto, estableciendo, unos requisitos y una arquitectura estables. Por otro lado, el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase.
- ✓ **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 sub-fases 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. (Balduino, 2007)

Ventajas de su uso

- ✓ Metodología de desarrollo de software de código abierto diseñado para pequeños equipos organizados.
- ✓ Proceso iterativo e incremental que es mínimo, completo y extensible.
- ✓ No define un modelo de negocio ni de dominio necesario.
- ✓ Permite detectar errores tempranos a través de un ciclo iterativo.
- ✓ Evita la elaboración de documentación, diagramas e iteraciones innecesarios requeridos en la metodología RUP.
- ✓ Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas.

1.4. Herramienta de Modelado

El UML (*Unified Modeling Language*) se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es un sistema de notación destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. UML es un lenguaje para construir modelos, no

guía al desarrollador en la forma de realizar el análisis y diseño orientados a objetos ni le indica cual proceso de desarrollo adoptar. Se compone de elementos de esquematización que representan las diferentes partes de un sistema de software. (Craig, 1999)

1.4.1. Visual Paradigm 8.0 para UML

Visual Paradigm para UML es una herramienta CASE[3] que soporta el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo del software. Permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas y generar documentación. (Reuse, 2013)

Las ventajas que proporciona Visual Paradigm para UML 8.0 son:

- Dibujo: Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Coherencia entre diagramas: Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Generación de código: Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- Generación de informes: Permite generar diversos informes a partir de la información introducida en la herramienta. (Reuse, 2013)

1.5. Marco de trabajo y bibliotecas

Un marco de trabajo o framework es un diseño abstracto orientado a objetos para un determinado tipo de aplicación, es un patrón arquitectónico que proporciona una plantilla extensible para un tipo específico de aplicaciones. “Un framework o "esquema" es un subsistema expandible de un conjunto de servicios, es un conjunto cohesivo de interfaces y clases que colaboran para proporcionar los servicios de la parte central e invariable de un subsistema lógico”. (Craig, 1999)

[3] **CASE:** *Computer-Aided Software Engineering* (Ingeniería Asistida por Computadora).

1.5.1. Symfony 2.0

Symfony2 es un marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web. Las principales funcionalidades que provee son: separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (Potencier, 2007)

Symfony2 es la versión más reciente de Symfony y supone un cambio radical tanto en la arquitectura interna como en la filosofía de trabajo respecto a sus versiones anteriores. La nueva versión ha sido ideada para aprovechar al máximo las nuevas características de PHP 5.3, convirtiéndose en uno de los frameworks PHP con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en un proyecto. (Eguiluz, 2014)

Es compatible con la mayoría de los gestores de bases de datos, entre ellos MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Otras de sus características más novedosas son:

- ✓ Fácil de instalar y configurar en la mayoría de las plataformas.
- ✓ Independiente del sistema gestor de bases de datos.
- ✓ Sencillo de usar, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- ✓ Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- ✓ Preparado para aplicaciones empresariales, y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

1.5.2. ExtJS 3.4

ExtJS es una biblioteca de JavaScript desarrollado 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. ExtJS aprovecha las funciones de HTML5 en los navegadores modernos manteniendo la compatibilidad y funcionalidad para navegadores

antiguos. Cuenta con cientos de widgets[4] 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)

1.5.3. Lycan

Lycan es una biblioteca de ExtJS, surge por la necesidad de diseñar componentes de ExtJS de manera intuitiva, rápida y cómoda, promoviendo la usabilidad y elevando la productividad de los desarrolladores. Esta biblioteca fue desarrollada por el departamento DATEC, pensada para contribuir también con otros proyectos externos al centro que también trabajaran con ExtJS en la UCI colaborando así con el éxito de los mismos. Además implementa buenas prácticas de arquitectura y diseño contribuyendo a la calidad del desarrollo de las aplicaciones web y se ha optimizado para su ejecución en Mozilla Firefox. (Lobo, 2010)

1.6. Lenguaje de programación

Un lenguaje de programación es un lenguaje artificial que dentro de la informática permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; permitiendo al desarrollador comunicarse con los dispositivos de hardware y software existentes.

1.6.1. PHP 5.3

PHP (*PHP Hypertext Preprocessor*) es un lenguaje de programación de código abierto que está especialmente pensado para el desarrollo web y que puede ser embebido en páginas HTML. Su sintaxis recurre a C, Java y Perl, siendo así sencillo de aprender. PHP está enfocado principalmente a la programación de scripts del lado del servidor. Puede emplearse en todos los sistemas operativos principales, incluyendo Linux, muchas variantes de Unix, Windows, Mac OS X entre otros. Admite la mayoría de servidores web como Apache, IIS, entre otros. Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página web con acceso a una base de datos se hace simple utilizando una de las extensiones específicas de bases de datos, o utilizar una capa de abstracción. (PHP Group, 2001)

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

1.6.2. JavaScript

JavaScript, se conoce como lenguaje script, es decir, se trata de código de programación que se inserta dentro de un documento. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa JavaScript fue desarrollado por la empresa Netscape con la idea de potenciar la creación de páginas web dinámicas para el navegador. JavaScript es más sencillo porque lo único que permite es insertar código especial dentro del HTML de una página, su función es ampliar las posibilidades de HTML. Este lenguaje no crea programas independientes, dependen por completo del código HTML de la página. Sin embargo es interpretado directamente por el navegador. Su ventaja fundamental es que su aprendizaje y uso son muy sencillos y que permite realizar labores complejas en una página. (Sánchez, 2003)

1.6.3. Java 7.0

Java es un lenguaje de programación orientado a objetos diseñado para ser portable en diversas plataformas. Fue diseñado tomando como patrón el lenguaje de programación C++. La característica predominante de Java es su adecuación a internet, la cual le permitió crear programas, gráficas interactivas y otros efectos en las páginas web.

Características de Java

Simple: Basado en el lenguaje C++ pero donde se eliminan muchas de las características orientadas a objeto que se utilizan esporádicamente y que creaban frecuentes problemas a los programadores.

Orientado al objeto: Java brinda soporte a las técnicas de desarrollo orientado a objeto.

Distribuido: Java se ha diseñado para trabajar en ambiente de redes y contienen una gran biblioteca de clases para la utilización del protocolo TCP/IP, incluyendo HTTP y FTP.

Interpretado: El compilador Java traduce cada fichero fuente de clases a código de bytes (Bytecode), que puede ser interpretado por todas las máquinas que provean soporte a un visualizador que funcione con Java.

Multihilos: Java puede aplicarse al desarrollo de aplicaciones en las que ocurra más de una evento a la vez.

Dinámico: Exige que se compile de nuevo la aplicación al cambiar una clase madre Java, utiliza un sistema de interfaces que permite aligerar esta dependencia.

1.7. Entorno integrado de desarrollo

Un IDE (*Integrated Development Environment*), es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Los IDE proveen un marco de trabajo amigable dedicándose en exclusiva a un sólo lenguaje de programación o como la mayoría, a varios lenguajes de programación como C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc.

1.7.1. NetBeans 8.0

NetBeans es un entorno de desarrollo integrado (IDE) que permite editar programas en Java, compilarlos, ejecutarlos, depurarlos y construir rápidamente la interfaz gráfica de una aplicación, eligiendo diferentes componentes. Es posible ejecutarlo tanto en Windows, Linux, Mac OS X como en Solaris. NetBeans es un entorno de programación para varios lenguajes, incluyendo Java y C++ .Permite el uso de un amplio rango de tecnologías de desarrollo tanto para escritorio, aplicaciones Web, como dispositivos móviles. Da soporte a tecnologías como: Java, PHP, Groovy, C/C++, HTML5. Este entorno de desarrollo integrado es de fuente abierta es decir, se proporciona el código fuente del entorno para que se pueda modificar de acuerdo a ciertos parámetros de licencia. El proyecto NetBeans es apoyado por una gran comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación. El código fuente está disponible para su reutilización de acuerdo con la CDDL (*Common Development and Distribution License*). (Oracle Corporation, 2014)

1.8. Servidor de Aplicaciones

1.8.1. Apache2

Apache es el servidor web utilizado para servir páginas web solicitadas por equipos cliente. Los clientes normalmente solicitan y muestran páginas web mediante el uso de navegadores web 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 PHP, y otros lenguajes de scripting populares como Python y Perl. Esta configuración se denomina LAMP (Linux, Apache, MySQL y Perl/Python/PHP) y conforma una potente y robusta plataforma para el desarrollo y distribución de aplicaciones basadas en la web.

La licencia Apache es descendiente de la licencias BSD, no es GPL. Esta licencia permite un código fuente abierto. (Una Introducción a APACHE, 2014)

Principales características:

- ✓ Puede ser instalado en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- ✓ Es una tecnología gratuita de código abierto.
- ✓ Servidor altamente configurable de diseño modular.

1.8.2. Apache Tomcat 8.0

Apache Tomcat también conocido como Tomcat o Jakarta Tomcat, es un servidor web multiplataforma que funciona como contenedor de servlets[5]y que se desarrolla bajo el proyecto denominado Jackarta. Implementa las especificaciones de los servlets y permite el libre acceso al código fuente bajo los términos establecidos por la *Apache Software Foundation* bajo la licencia Apache 2.0. El hecho de que Tomcat fue escrito en Java, hace posible que funcione en cualquier sistema operativo que disponga de la máquina virtual Java. Tomcat es usado como servidor web independiente en entornos con alto nivel de tráfico y alta disponibilidad. (The Apache Software Foundation , 2015).

1.9. Gestor de Base de Datos

Los gestores de base de datos tienen como objetivo servir de interfaz entre la base de datos, el usuario y las aplicaciones. Las características de un Gestor de Base de Datos son: abstracción de la información, independencia, redundancia mínima, consistencia, seguridad, integridad, respaldo y recuperación, control de la concurrencia. Entre los gestores de base de datos más utilizados se encuentra: Oracle, PostgreSQL, MySQL, MS SQL Server, entre otros.

1.9.1. PostgreSQL 9.3

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. PostgreSQL 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. Su estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema.

[5] Servlets: Pequeño programa de Java que se ejecuta dentro de un servidor web, recibe y responde las peticiones de clientes web a través de HTTP.

A continuación alguna de las características más importantes y soportadas por PostgreSQL:

- ✓ Es una base de datos totalmente ACID.
- ✓ Integridad referencial.
- ✓ Replicación asincrónica/sincrónica.
- ✓ Completa documentación.
- ✓ Licencia BSD.
- ✓ Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit. (Sobre PostgreSQL, 2009)

1.10. Administrador de Bases de Datos

Un administrador de base de datos dirige o lleva a cabo todas las actividades relacionadas con el mantenimiento de un gestor de base de datos. Las responsabilidades incluyen el diseño, implementación y mantenimiento del sistema de base de datos; el establecimiento de políticas y procedimientos relativos a la gestión y la seguridad. Se selecciona utilizar como Administrador de Base de Datos PgAdmin III.

1.10.1. PgAdmin III

PgAdmin III es una aplicación gráfica de código abierto para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia *Open Source*. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL.

La herramienta está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La conexión al servidor puede hacerse mediante conexión TCP/IP o *Unix Domain Sockets* y se puede cifrar mediante SSL para mayor seguridad. (PgAdmin III, 2008)

1.11. Olap4J

Olap4 es una API de código abierto de Java para construir aplicaciones OLAP. Olap4j representa para los datos multidimensionales el equivalente de los JDBC para los datos relacionales. Tiene un modelo de

programación similar al JDBC, comparte alguna de sus clases básicas. Es posible programar una aplicación OLAP en Java para un servidor (Mondrian por ejemplo) y fácilmente cambiarlo por otro. Una aplicación OLAP interactúa con un servidor OLAP por medio de sentencias MDX. Las sentencias son definidas en términos de metadatos y validadas de acuerdo al tipo de sistema, y algunas aplicaciones manipulan árboles de análisis sintáctico MDX y define complejas consultas en términos que el usuario del negocio pueda comprender, la API olap4j provee todas estas facilidades. En el nivel más bajo, olap4j tiene un frameworks para registrar drivers y gestionar el ciclo de vida de las conexiones. Además del analizador sintáctico MDX y operaciones en el árbol de análisis sintáctico MDX existe un modelo de consulta de nivel superior que incluye las operaciones para transformar las consultas y las instalaciones de la disposición de los resultados multidimensionales como tablas HTML. (Hyde, 2011)

1.12. Servicio Web

El consorcio W3C[6] define a los Servicios Web como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer una serie de servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la web. A su vez proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. (CONSORTIUM, 2014)

REST (*Representational State Transfer*) es un estilo de arquitectura de software para sistemas de hipermedias distribuidos. El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP. REST se refiere estrictamente a una colección de principios para el diseño de arquitecturas en red. Estos principios resumen como los recursos son definidos y diseccionados. (Marset, 2007)

Los objetivos de este estilo de arquitectura se listan a continuación:

- ✓ **Escalabilidad de la interacción con los componentes**. La variedad de clientes que pueden acceder a través de la Web: estaciones de trabajo, sistemas industriales y dispositivos móviles.
- ✓ **Generalidad de interfaces.** El uso del protocolo HTTP permite a cualquier cliente interactuar con cualquier servidor HTTP sin ninguna configuración especial.

[6] El Consorcio World Wide Web (W3C) es una comunidad internacional donde las organizaciones Miembro, trabajan conjuntamente para desarrollar estándares Web

- ✓ **Puesta en funcionamiento independiente**. Los clientes y servidores pueden ser puestas en funcionamiento durante años. Por tanto, los servidores antiguos deben ser capaces de entenderse con clientes actuales y viceversa. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URIs para crear nuevos métodos y tipos de contenido.

Se escoge al servicio web REST para garantizar el consumo de las librerías OLAP debido a las siguientes características:

- ✓ HTTP es un protocolo que sigue los principios REST, por lo tanto se aprovecha toda la infraestructura de la web ya existente: cache, proxis y firewall.
- ✓ La posibilidad de usar múltiples representaciones o formatos de datos, de forma natural, es una ventaja indiscutible. No hay que extender ningún protocolo, o limitarse a XML, el protocolo HTTP ya lo soporta de forma natural.
- ✓ Los servicios REST tienen menor acoplamiento que los servicios basados en SOAP.

Conclusiones del capítulo

En el presente capítulo se realiza un estudio de los conceptos fundamentales de la investigación y un análisis de las herramientas OLAP integradas a los generadores de reportes para la gestión de consultas MDX. Se seleccionó como metodología de desarrollo Open Up, para facilitar la planificación del proceso desarrollo. Se utilizarán como marco de trabajo Symfony 2.0 del lado del servidor y ExtJS 3.4 con Lycan del lado del cliente y como lenguaje de modelado UML. Se decidió utilizar como herramienta de modelado Visual Paradigm 8.0 para UML, como lenguaje de programación JavaScript y PHP 5.3, como IDE de desarrollo NetBeans 8.0. Como gestor de base de datos se usará PostgreSQL 9.3 y para su administración gráfica el PgAdmin III; Apache2 será utilizado como servidor web. Para la implementación del servicio web se empleará el lenguaje Java 7.0, la librería Olap4j utilizada para el desarrollo de aplicaciones OLAP y será desplegado en el servidor web Apache Tomcat 8.0. Estas herramientas serán las utilizadas en el desarrollo del editor de consultas MDX para el GDR v2.0, brindando facilidades a los desarrolladores en cuanto a rapidez y variedad de funciones.

CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA SOLUCIÓN.

Introducción

En este capítulo se describe el análisis y diseño del editor de consultas MDX para el Generador Dinámico de Reportes v2.0. Siendo el punto de partida el modelo de dominio donde se relacionan los conceptos fundamentales de la investigación. Se hace un desglose de los requisitos funcionales y no funcionales con que cumplirá la aplicación. Además de la realización del diagrama de casos de uso del sistema, clases del diseño, a través de los cuales se muestra la estructura estática del sistema y la vista de despliegue donde se muestra la estructura física de la solución.

2.1. Modelo de dominio

Un modelo del dominio tiene por finalidad tratar una especificación del dominio del problema y los requerimientos desde la perspectiva de la clasificación por objetos y desde el punto de vista de entender los términos empleados en el dominio. Para descomponer el dominio del problema hay que identificar los conceptos, los atributos y las asociaciones del dominio que se juzgan importantes. El resultado puede expresarse en un **modelo conceptual,** el cual se muestra gráficamente en un grupo de diagramas que describen los conceptos (Craig, 1999). A continuación se presenta en la figura 1 el modelo de dominio del editor de consultas MDX para el Generador Dinámico de Reportes v2.0.

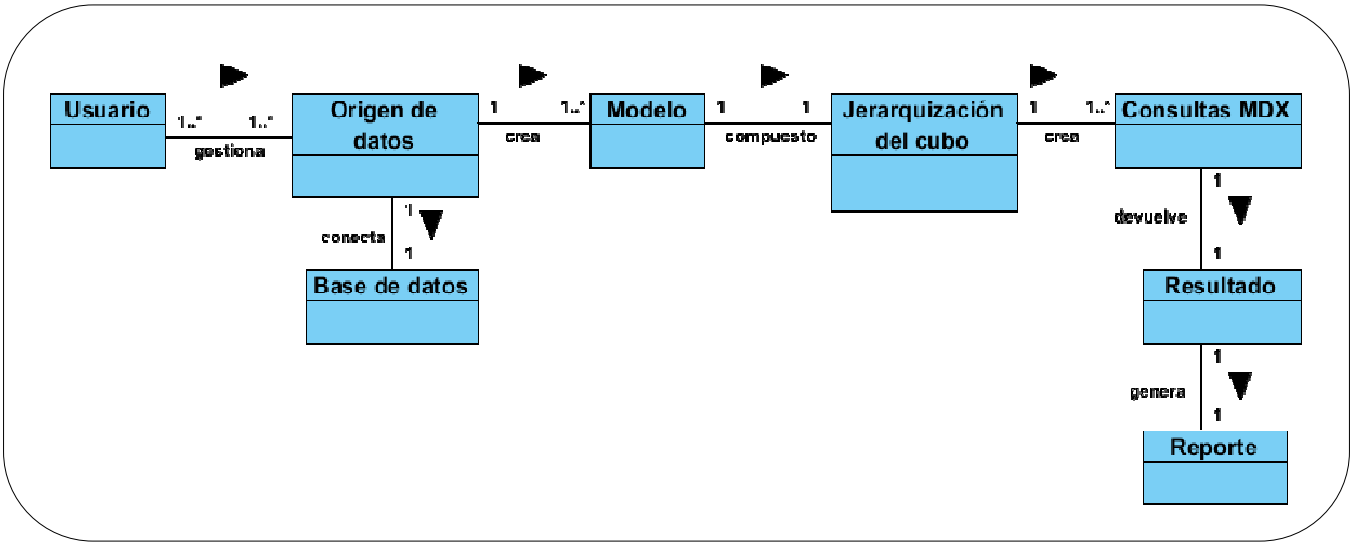


Fig. 1 Modelo de dominio

Definición de las clases del modelo de dominio

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

Origen de datos: Contiene los datos que permiten conectar al GRD v2.0 con el servidor Mondrian. Las variables que maneja son: servidor, puerto, gestor, cubo de datos y base de datos asociada al cubo de datos.

Base de dato: Datos que maneja la estructura de cubos de datos.

Modelo: Abstracción de la base de datos de los clientes que contienen los metadatos de las tablas.

Jerarquización del cubo: Estructura jerárquica por niveles de cubos de datos.

Consulta MDX: Método para acceder a datos específicos contenidos en cubos de datos.

Resultado: Información que devuelve la consulta MDX.

Reporte: Utiliza la información contenida en los modelos para la confección de reportes, los cuales van a ser exportados en diferentes formatos.

2.2. Requisitos de Software

La ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento, modelado y especificación. Se refinan en detalle los requisitos del sistema y el papel asignado al software. Tanto el desarrollador como el cliente tienen un papel activo en la ingeniería de requisitos un conjunto de actividades que son denominadas análisis. El análisis de requerimientos permite especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software. (Pressman, 2011)

2.2.1. Requisitos Funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares describen con detalle su función, entradas y salidas. (Somerville, 2005). A continuación se describen los requisitos funcionales que debe cumplir el editor de consultas MDX para el Generador Dinámico de Reportes v2.0.

RF 1: Añadir origen de datos OLAP.

Descripción: Permite añadir el nuevo origen de datos a la lista de orígenes de datos.

Entrada: Nombre, servidor, puerto, gestor, usuario, contraseña, base de datos asociada al cubo de datos y cubo de datos.

Salida: Adiciona el origen de datos OLAP a la lista de orígenes de datos y muestra un mensaje de notificación.

RF 2: Modificar origen de datos OLAP.

Descripción: Permite la modificación de los parámetros definidos para el origen de datos.

Entrada: Nombre, servidor, puerto, gestor, usuario, contraseña, base de datos asociada al cubo de datos y cubo de datos.

Salida: Origen de datos modificado y mensaje de notificación.

RF 3: Eliminar origen de datos OLAP.

Descripción: Permite eliminar el origen de datos de la lista de orígenes de datos.

Entrada: Origen de datos.

Salida: Origen de datos eliminado de la lista de orígenes de datos.

RF 4: Adicionar modelo.

Descripción: Permite añadir un modelo a partir de un origen de datos creado.

Entrada: Nombre y descripción del modelo.

Salida: Modelo añadido a la lista de modelos y mensaje de notificación.

RF 5: Modificar modelo.

Descripción: Permite modificar un modelo.

Entrada: Nombre y descripción del modelo.

Salida: Modelo modificado y mensaje de notificación.

RF 6: Mostrar jerarquía del cubo de datos.

Descripción: Permite explorar la jerarquización conformada por hechos, medidas y dimensiones del origen de datos añadido al sistema.

Entrada: El cubo de datos.

Salida: Vista por los niveles de la jerarquía.

RF 7: Crear consulta MDX

Descripción: Permite la creación de consultas a partir de un modelo creado.

Entrada: No aplica.

Salida: Añadir la consulta al modelo.

RF 8: Abrir consulta MDX.

Descripción: Permite abrir una consulta registrada en el sistema.

Entrada: Nombre de la consulta.

Salida: La consulta MDX.

RF 9: Salvar consulta MDX.

Descripción: Permite salvar la consulta.

Entrada: La consulta MDX.

Salida: Consulta salvada en el modelo correspondiente.

RF 10: Exportar consulta MDX.

Descripción: Permite exportar la consulta seleccionada.

Entrada: La consulta MDX que se desee exportar.

Salida: Fichero con la consulta MDX.

RF 11: Importar consulta MDX.

Descripción: Permite importar el código MDX de la consulta, adicionándola al modelo asociado a esta.

Entrada: Fichero que contiene la consulta MDX.

Salida: Visualización de la consulta en el editor de consultas.

RF 12: Ejecutar consulta MDX.

Descripción: Permite visualizar el resultado de la consulta MDX realizada al cubo de datos.

Entrada: La consulta MDX.

Salida: Resultado de la consulta MDX.

RF13 Exportar reporte.

Descripción: Se obtiene los datos para la construcción del reporte.

Entrada: Reporte publicado.

Salida: Fichero con el reporte generado.

2.2.2. Requisitos no Funcionales

Los requerimientos no funcionales no se refieren a las funcionalidades específicas que proporciona el sistema, sino a sus propiedades emergentes como la disponibilidad del sistema, rendimiento fiabilidad, tiempo de respuesta y la capacidad de almacenamiento. De forma alternativa definen las restricciones del sistema como la capacidad de los dispositivos de entrada/salida y las representaciones de los datos que utilizan las interfaces del sistema (Somerville, 2005). A continuación se describen los requisitos no funcionales que debe cumplir el editor de consultas MDX para el Generador Dinámico de Reportes v2.0.

Facilidad de uso

RNF 1 El usuario en el módulo Diseñador de Modelos tendrá la facilidad a través de un paginado de tres clic de crear un modelo seleccionando los cubos que desee para conformarlo. Luego en la parte izquierda de la interfaz se mostrará el explorador de modelos brindándole al usuario la visualización de los elementos que lo componen para un mejor trabajo.

RNF 2 En el módulo Diseñador de Reportes a través de la propiedad emptytext se le informa al usuario cuando adiciona los campos para diseñar un reporte, que cuando las consultas son MDX el campo Descripción es obligatorio llenarlo.

RNF 3 En el módulo Diseñador de Reportes al adicionar los campos, la propiedad tooltip le notifica al usuario final a través de un evento en el campo Descripción, la estructura que debe seguir para añadir el campo necesario en la generación del reporte.

Interfaz

Las interfaces de usuario serán diseñadas a modo de aplicaciones RIA (Rich Internet Application, en español Aplicaciones de Internet Enriquecidas), lo que permite a los usuarios contar con aplicaciones web con una experiencia de usuario similar a la de las aplicaciones de escritorios.

RNF 4 El usuario deberá acceder a la aplicación a través del protocolo HTTPS usando el navegador Firefox 28.0 o superior.

Eficiencia

La eficiencia del sistema depende en gran medida de la velocidad de conexión a las base de datos donde se encuentre, así como del volumen de información contenido en las mismas. Sin embargo el sistema debe mantener tiempos de respuestas en un marco razonable

RNF 5 El sistema para mostrar el resultado de las consultas MDX debe mantener un tiempo de respuesta de hasta 30 segundos.

RNF 6 El sistema debe permitir la conexión simultánea de 10 usuarios.

Restricciones de Implementación

Para la implementación de la aplicación se requiere del uso de las siguientes herramientas:

RNF 7 El sistema para atender las peticiones del lado del servidor utiliza el marco de trabajo Symfony 2.0, basado en el lenguaje de desarrollo PHP 5.3 o superior. La librería de JavaScript ExtJS 3.4 para el lado del cliente. Como IDE de desarrollo se usará NetBeans 8.0 y como sistema gestor de base de datos se empleará PostgreSQL 9.3.

RNF 8 Para la implementación del servicio web se empleó el lenguaje Java 7.0 y fue desplegado en el servidor web Apache Tomcat 8.0. Para gestionar las consultas MDX en el servicio web se utiliza la librería Olap4j utilizada para el desarrollo de aplicaciones OLAP.

RNF 9 Requerimientos de Software

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

- ✓ Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 12.04, Debian 4 GNU/Linux.
- ✓ Paquetes: apache2, php5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-intl, php5-xsl, php5-gd, php5-curl, apache tomcat.

- ✓ Usuario con privilegios de administración del Sistema Operativo.

El servidor donde se instalará la Base de Datos del sistema debe cumplir con los siguientes requisitos:

- ✓ Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 12.4.
- ✓ PostgreSQL versión 9.3.
- ✓ 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.

RNF 10 Requerimientos de hardware

Las PC clientes deben cumplir con los siguientes requisitos de hardware:

- ✓ Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 256 MB.

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

- ✓ Procesador Intel Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ Memoria RAM mínimo 1Gb.
- ✓ 10 GB de espacio en disco duro.

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

- ✓ Procesador Intel Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- ✓ 512 MB de RAM como mínimo.
- ✓ 40 GB de espacio en disco duro.

2.3. Diagrama de Casos de Uso del sistema

Los diagramas de casos de uso especifican la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. Representan la relación entre los actores y los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito. Las principales relaciones entre los elementos del modelo son la especialización y la generalización. La figura 2 muestra el diagrama de Casos de Uso del editor de consultas MDX para GDR v2.0.

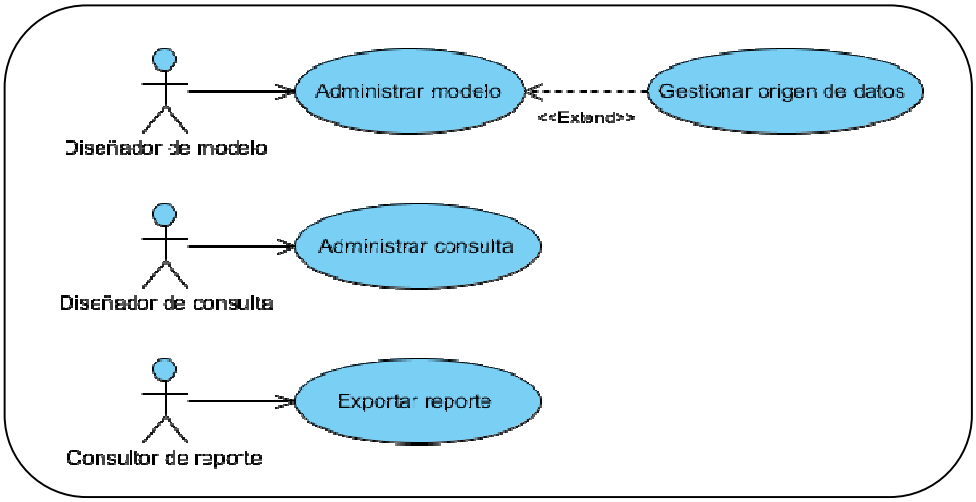


Fig. 2 Diagrama de Casos de Uso del Sistema

Descripción de los actores y casos de uso del sistema

Diseñador de modelo: Usuario encargado de interactuar con el módulo Diseñador de Modelos de GDR, con la responsabilidad de Administrar modelo y Gestionar origen de datos.

Diseñador de consulta: Usuario encargado de interactuar con el módulo Diseñador de Consultas de GDR, con la responsabilidad de Administrar consulta.

Consultor de reporte: Usuario encargado de interactuar con el módulo Diseñador de Reportes de GDR, con la responsabilidad de Exportar reporte.

Gestionar origen de datos: Permite al diseñador de modelo realizar las acciones: crear, modificar y eliminar orígenes de datos a partir de los cuales se crearán posteriormente modelos.

Administrar modelo: Permite al diseñador de modelo modificar modelos y mostrar la estructura jerárquica por niveles de cubos de datos.

Administrar consulta: Permite al diseñador de consultas realizar las acciones: crear, abrir, salvar, exportar e importar consultas.

Exportar reporte: Permite al consultor de reportes exportar los reportes realizados en formato PDF.

2.3.1. Patrones de Casos de Uso usados en la solución

CRUD parcial: El patrón se evidencia en el CU Administrar consulta pues en él se agrupan los siguientes requisitos funcionales: crear, abrir, salvar, ejecutar, exportar e importar consulta.

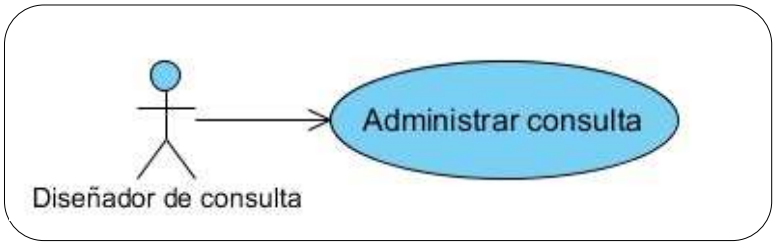


Fig. 3 Evidencia del patrón CRUD parcial

Extensión concreta: El patrón se evidencia en la relación extendida entre el CU base Administrar modelo y el CU extendido Gestionar origen de datos, donde este último no altera el funcionamiento del primero, sino que se incorpora como parte integral del caso de uso base.

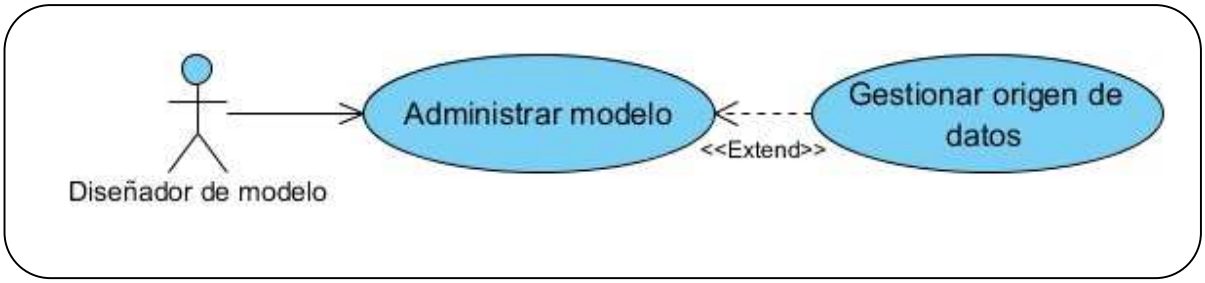


Fig. 4 Evidencia del patrón Extensión concreta

Matriz de trazabilidad

La matriz de trazabilidad permite tener una visión rápida de las relaciones de dependencia entre casos de uso del sistema y los requisitos funcionales, posibilitando comprobar si todos los casos de uso tienen algún requisito asociado y si todos los requisitos están justificados por un determinado caso de uso.

Tabla 1 Matriz de trazabilidad

RF/CU	Gestionar origen de datos	Administrar modelo	Administrar consulta	Exportar reporte
RF1	X			
RF2	X			
RF3	X			
RF4		X		

RF5		X		
RF6		X		
RF7			X	
RF8			X	
RF9			X	
RF10			X	
RF11			X	
RF12			X	
RF13				X

A través de la matriz de trazabilidad queda evidenciada la correcta correspondencia entre los 13 requisitos funcionales definidos y los 4 casos de uso en que fueron agrupados.

2.4. Descripción del Caso de Uso arquitectónicamente significativo del sistema CU. Administrar consulta.

A continuación se muestra la especificación del caso de uso arquitectónicamente significativo para el sistema Administrar consulta, se puede consultar el expediente de proyecto del GDR versión 2.0 para ver las especificaciones de los restantes casos de uso.

Tabla 2 Especificación del Caso de Uso: Administrar consulta

Objetivo	Administrar consulta
Actores	Diseñador de consulta
Resumen	El caso de uso se inicia cuando el Diseñador de consulta selecciona la opción “Nueva” para construir una consulta MDX. Termina una vez finalizada alguna de las acciones (crear, abrir, salvar, eliminar, exportar e importar consulta).
Complejidad	Alta
Prioridad	Crítico

Precondiciones	Es necesario tener al menos un modelo creado. Para crear, abrir, exportar e importar una consulta MDX es necesario tener seleccionado un modelo. Para abrir, salvar, ejecutar, exportar consulta es necesario que exista previamente una consulta creada. Para ejecutar y exportar la consulta es necesario antes abrirla.
Postcondiciones	Administrar una consulta.

Flujo de eventos		
Flujo básico < Administrar consulta>		
	Actor	**Sistema**
1.	Selecciona el módulo “Diseñador de Consultas”	
2.		Habilita el menú “Gestionar consultas” permitiendo en el área de edición de consultas: - Crear consulta, ver **Sección1: “Crear consulta”.** - Abrir consulta, ver **Seccion2: “Abrir consulta”.** - Salvar consulta, ver **Seccion3: “Salvar consulta”.** - Exportar consulta, ver **Sección4: “Exportar consulta”.** - Importar consulta, ver **Sección5: “Importar consulta”.** - Ejecutar consulta, ver **Sección6: “Ejecutar consulta”.**
3.		Termina el caso de uso.
Sección 1: “Crear consulta”		
Flujo básico: “Crear consulta”		

	Actor	Sistema
1.	Selecciona el modelo asociado.	
2.	Selecciona la opción “Nueva” del menú “Gestionar consultas”.	
3.		Muestra un menú con los lenguajes de consultas MDX y SQL para seleccionar.
4.	Selecciona el lenguaje de consulta MDX.	
5.		Si el editor tiene abierta una consulta previamente. Muestra una solicitud para salvar dicha consulta.
6.	Presiona el botón “Salvar”, **ver Sección 3**	
	Presionar el botón de “Descartar”, **ver Flujo Alterno 1.**	
	Presionar el botón de “Cancelar”, **ver Flujo Alterno 2.**	
7.		Muestra la interfaz para construir la consulta MDX.
8.	Introduce la sentencia de la consulta MDX.	
Flujos alternos		
Nº Evento < Flujo alterno 1: Descartar opción “Crear consulta”>		
	Actor	**Sistema**
1.1		Permite crear la nueva consulta sin salvar los cambios de la consulta actual.
Flujos alternos		
Nº Evento < Flujo alterno 2: Cancelar opción “Crear consulta”>		
	Actor	Sistema
2.1		Cancela la solicitud de abrir y cierra la interfaz.
Sección 2: “Abrir consulta”		

Flujo básico: “Abrir consulta”		
	Actor	**Sistema**
1.	Selecciona la opción “Abrir” del menú “Gestionar consultas”.	
2.		Si el editor tiene abierta una consulta previamente. Muestra una solicitud para salvar dicha consulta.
3.	Presiona el botón “Salvar”, ver **Sección 3**	
	Presionar el botón de “Descartar”, ver **Flujo Alterno 1.**	
	Presionar el botón de “Cancelar”, ver **Flujo Alterno 2.**	
4.		Muestra una interfaz que visualiza un listado con las consultas registradas en el sistema agrupadas por modelo.
5.	Selecciona la consulta a abrir.	
6.	Selecciona el botón “Aceptar”	
	Selecciona el botón “Cancelar”, ver **Flujo Alterno 1.**	
7.		Muestra la consulta en el editor de consulta
Flujos alternos		
Nº Evento < Flujo alterno 1: Descartar opción “Abrir consulta”>		
	Actor	**Sistema**
1.1		Abre la nueva consulta sin salvar los cambios de la consulta actual.
Flujos alternos		
Nº Evento < Flujo alterno 2: Cancelar opción “Abrir consulta”>		
	Actor	**Sistema**

2.1		Cancela la solicitud de abrir y cierra la interfaz.
Sección 3: "Salvar consulta"		
Flujo básico: "Salvar **consulta"**		
	Actor	**Sistema**
1.	Selecciona la opción "Salvar" del menú "Gestionar consultas".	
2.		Verifica que exista en el editor de consulta una consulta asociada a un modelo.
3.		Muestra una interfaz solicitando al actor que introduzca el nombre identificativo de la consulta.
4.	Introduce el nombre identificativo de la consulta.	
5.	Selecciona el botón "Salvar",	
	Selecciona el botón "Cancelar", ver **Flujo Alterno 1**	
6.		Almacena la consulta en la base de datos del sistema.
Flujos alternos		
Nº Evento < Flujo alterno 1: Cancelar opción "Salvar consulta">		
	Actor	**Sistema**
1.		Cancela la solicitud de salvar y cierra la interfaz.
Sección 4: "Exportar consulta"		
Flujo básico: "Exportar consulta"		
	Actor	**Sistema**
1.	Abre la consulta, ver **Sección 2**	
2.	Selecciona la opción "Exportar" del menú	

	"Gestionar consultas".	
3.		Muestra una interfaz para seleccionar la URL hacia donde se va a exportar la consulta.
4.	Selecciona la opción "Salvar"	
	Selecciona la opción "Cancelar" ver **Flujo Alterno 1.**	
5.		Exporta el archivo a la URL indicada.
Flujos alternos		
Nº Evento < Flujo alterno 1: Cancelar opción "Exportar consulta">		
	Actor	**Sistema**
1.1		Cancela la solicitud de exportar y cierra la interfaz.
Sección 5: "Importar consulta"		
Flujo básico: "Importar consulta"		
	Actor	**Sistema**
1.	Selecciona la opción "Importar" del menú "Gestionar consultas".	
2.		Muestra una interfaz para seleccionar la ubicación de la consulta que se agrega al sistema.
3.	Selecciona la consulta que se agrega al sistema y da clic en el botón "Aceptar".	
	Selecciona la opción "Cancelar" ver **Flujo Alterno 1**.	
4.		Muestra la consulta en el área de edición de consultas.
Flujos alternos		
Nº Evento < Flujo alterno 1: Cancelar opción "Importar consulta">		

	Actor	Sistema
1.		Cancela la solicitud de importar y cierra la interfaz.

Sección 6: “Ejecutar consulta”		
Flujo básico: “Ejecutar consulta”		
	Actor	**Sistema**
1.	Abre la consulta, ver **Sección 2**	
2.		Muestra la interfaz para la edición de consultas con la sentencia de la consulta seleccionada.
3.	Presiona la opción “Ejecutar” en el editor de consultas.	
4.		Muestra una interfaz con el resultado que arroja la consulta. En caso de error se muestra en parte inferior del editor.
Flujos alternos		
Nº Evento < Flujo alterno 1: Cancelar opción “Exportar consulta”>		
	Actor	**Sistema**
1.1		Cancela la solicitud de exportar y cierra la interfaz.

Relaciones	**CU Incluidos:** -
	CU Excluidos: -
Requisitos no funcionales	-
Asuntos pendientes	-

Prototipo de interfaz

Fig. 5 Prototipo de interfaz

2.5. Modelo de diseño

En el diseño se modela el sistema para que soporte todos sus requisitos funcionales, no funcionales y otras restricciones que le suponen. Una entrada esencial en el diseño es el resultado del análisis, proporcionando una comprensión detallada de los requisitos. El diseño es la atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida para crear un plano del modelo de implementación.

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como el entono de la implementación tienen impacto en el sistema a considerar. El modelo sirve de abstracción de la implementación del sistema, usada como una entrada fundamental de las actividades de implementación. (Ivar Jacobson, 2000).

2.5.1. Diagrama de clases del diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Normalmente contiene la siguiente información: clases, asociaciones y atributos, interfaces, con sus operaciones y constantes métodos información sobre los tipos de los atributos, navegabilidad y dependencias. A diferencia del modelo conceptual, un diagrama de este tipo contiene las definiciones de las entidades del software en vez de conceptos del mundo real (Craig, 1999)

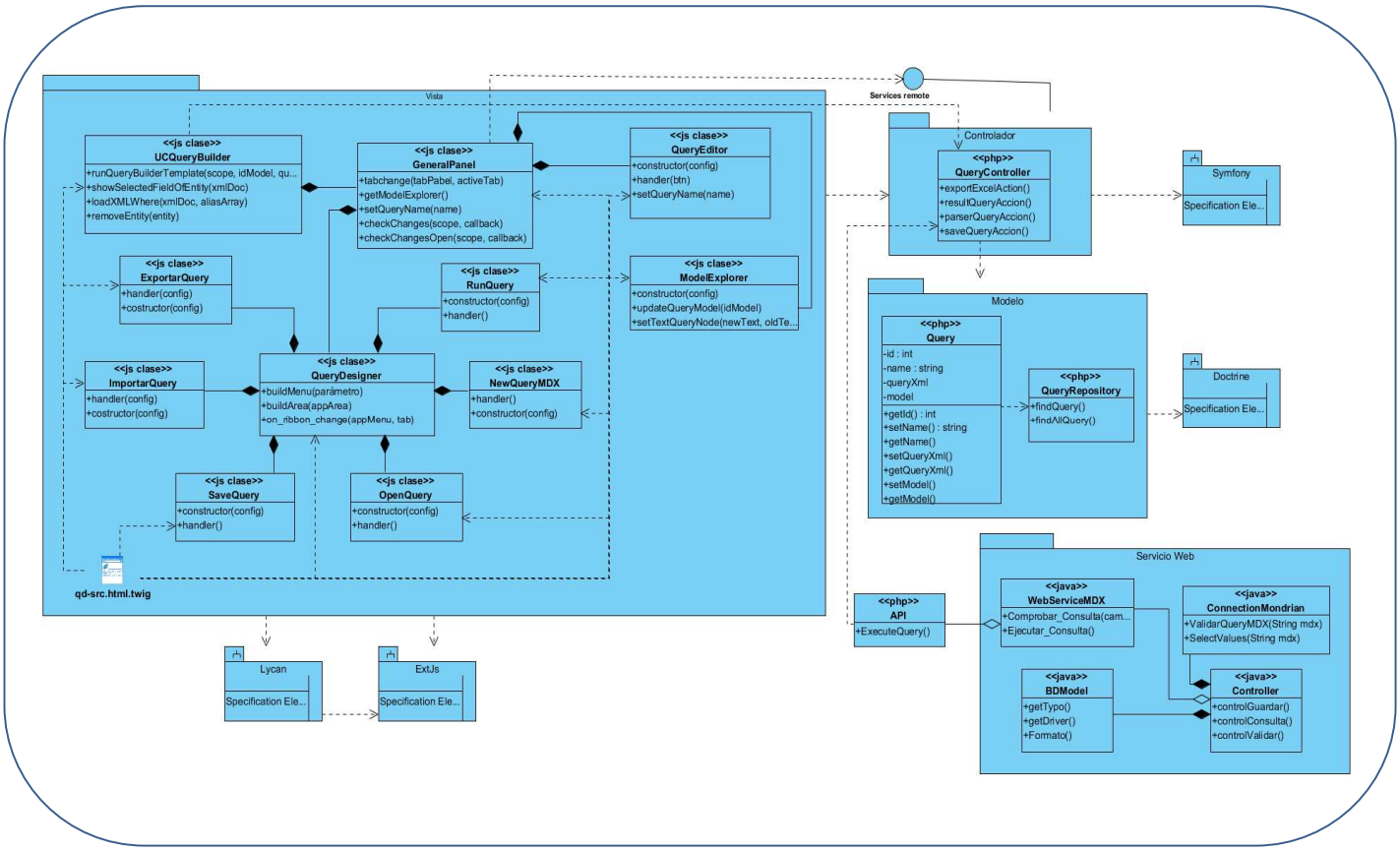


Fig. 6 Diagrama de clases del diseño

La figura 6 muestra el diagrama de clases del diseño del caso de uso arquitectónicamente significativo para el sistema Administrar Consulta, estructurado según el patrón arquitectónico Modelo-Vista-Controlador. En el paquete Vista se muestran todos los ficheros JavaScript que permiten la interacción con el usuario. El paquete Controlador contiene la clase controladora del módulo Diseñador de Consultas, se encarga de procesar las peticiones del usuario y realiza los cambios apropiados en el modelo y la vista.

Las clases entidades necesarias para interactuar con la base de datos que manejan la lógica del negocio se encuentran en el paquete modelo.

La clase *GeneralPanel.js* es la encargada de construir la interfaz visual del módulo Diseñador de Consultas utilizando las clases *ModelExplorer.js, QueryDesigner.js y QueryEditor.js*. La clase *ModelExplorer.js* visualiza el explorador de modelos construido en el módulo Diseñador de Modelos a partir de los metadatos extraídos del origen de datos seleccionado. El menú Gestionar Consultas que contiene las acciones *RunQuery.js, OpenQuery.js, NewQueryMDX.js, SaveQuery.js, RenameQuery.js, ExportarQuery.js e ImportarQuery.js* posicionado en la parte superior del módulo, es construido por la clase *QueryDesigner.js*. En la parte central se encuentra situada el área de edición de consultas que es construida por la *QueryEditor.js*.

La clase *Query.php* contiene la estructura con la que se guarda la consulta en la base de datos y la *QueryRepository.php* representa de manera general la información y consultas realizadas por las clases. La clase *UCQueryBuilder.js* gestiona todas las acciones y los métodos de la controladora *QueryController.php.* Dicha clase contiene métodos como saveQueryBuilderTemplate(), que llama a la acción saveQueryAction() para salvar las consultas que se construyen y loadXMLQuery() que utiliza la acción loadXmlAction() para abrir las consultas anteriormente guardadas. La clase *QueryController.php* es la encargada de responder las peticiones realizadas del lado del cliente a través de métodos como saveQueryAction(), renameQueryAction(), deleteQueryAction(), entre otros.

2.5.2. Patrones de Diseño usados en la solución

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Craig, 1999).

Controlador

Se evidencia en la clase controladora *QueryController.php* cuya responsabilidad es interactuar con las clases de acceso a dato, obtener la información necesaria y dar respuesta a las peticiones del cliente controlando así el flujo central de acciones del sistema.

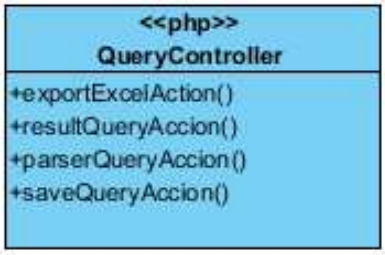


Fig. 7 Evidencia del patrón Controlador

Experto

El patrón se evidencia en la clase *UCQueryBuilder.js*, contiene toda la información necesaria para recibir las peticiones y comunicarse con la clase controladora, siendo la experta en cumplir las funcionalidades para las que fue implementada.

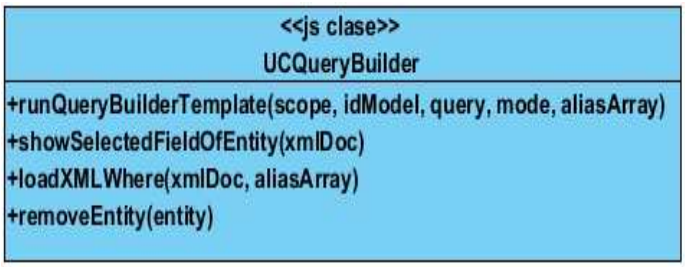


Fig. 8 Evidencia del patrón Experto

Patrón Alta Cohesión

Se puede apreciar el patrón alta cohesión en la relación existente entre *GeneralPanel.js* y las clases *QueryDEsigner.js, ModelExplorer.js y QueryEditor.js* que tienen responsabilidades moderadas en un área funcional y colaboran entre ellas para llevar a cabo las tareas para las que fueron implementadas.

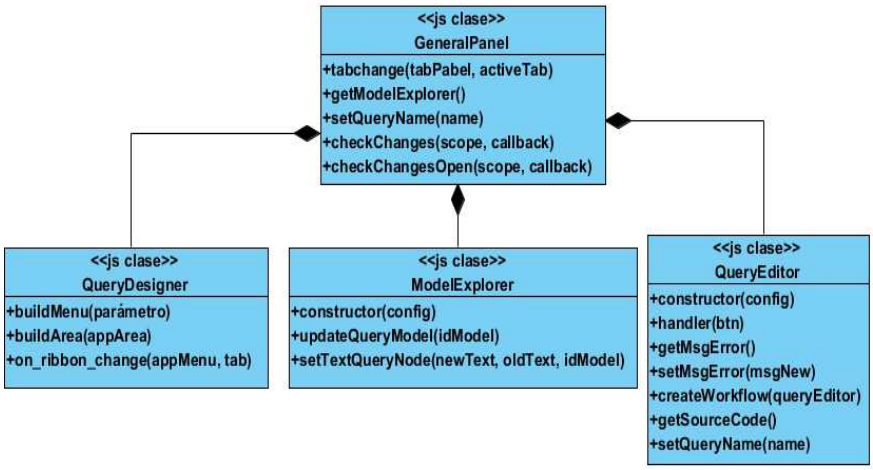


Fig. 9 Evidencia del patrón alta cohesión

Patrón Bajo Acoplamiento

El patrón bajo acoplamiento significa asignar una responsabilidad para mantener pocas dependencias entre las clases, evidenciándose a través de las clases *QueryDesigner.js, ModelExplorer.js y QueryEditor.js* que solo se relacionan con *GeneralPanel.js.* Garantizándose minimizar las afectaciones por cambios en otros componentes y su reutilización.

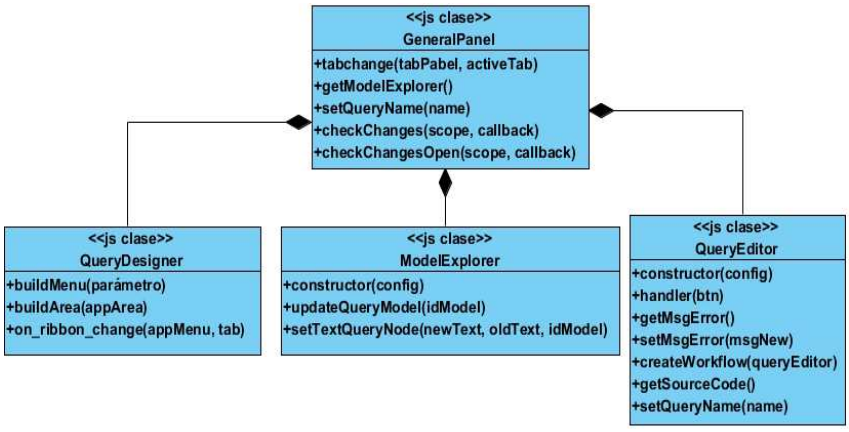


Fig. 10 Evidencia del patrón Bajo Acoplamiento

Los patrones GOF, también conocidos como los patrones de la pandilla de los cuatro (GoF, gang of four)

Fachada: Se evidencia el uso de este patrón en la clase *GeneralPanel.js* pues unifica las clases visuales y las muestra en una sola interfaz.

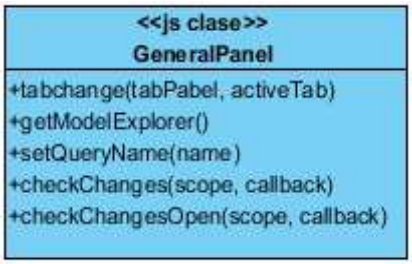


Fig. 11 Evidencia del patrón Fachada

Patrones de Arquitectura

El patrón Modelo- Vista- Controlador:

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por 3 niveles:

- ✓ El modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- ✓ La vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- ✓ El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (Potencier, 2007)

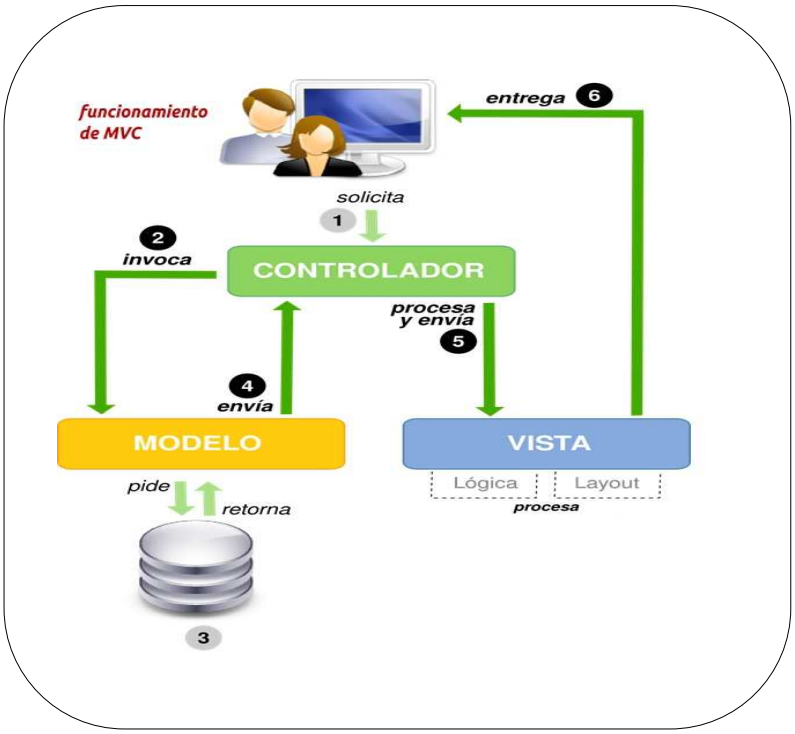


Fig. 12 Patrón arquitectónico Modelo -Vista –Controlador

2.5.3. Diagrama de secuencia

Los diagramas de secuencia en UML, indican cómo los eventos causan transiciones de objeto a objeto. Un vez que se han identificado los eventos al examinar un caso de uso, se crea el diagrama de secuencia: una representación de como los eventos causan un flujo de un evento a otro como una función de tiempo. (Pressman, 2011)

La figura 13 muestra el diagrama de componentes del Caso de Uso Administrar Consulta. En el expediente de proyecto del GDR versión 2.0 se encuentran los restantes diagramas.

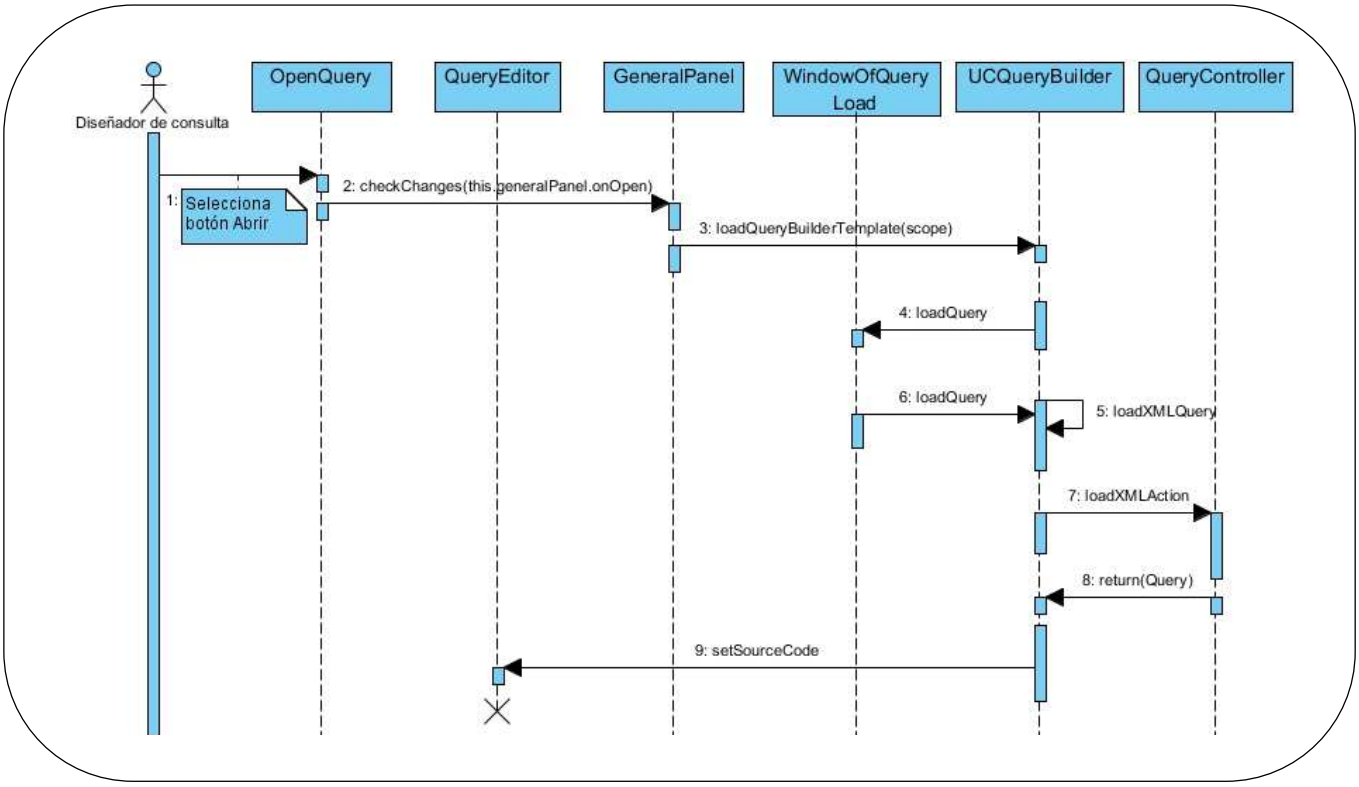


Fig. 13 Diagrama de secuencia del CU Administrar Consulta sección Abrir Consulta

En el diagrama de secuencia de la figura anterior el actor diseñador de consulta selecciona la opción abrir en el módulo Diseñador de Consultas. Seguidamente se ejecuta el método handler() de la clase *OpenQuery.js* que llama al método checkChanges(this.generalPanel.onOpen) de la clase *GeneralPanel.js*. Luego se invoca al método onOpen() de la *GeneralPanel.js* que llama al método loadQueryBuilderTemplate() de la clase *UCQueryBuilder.js* para obtener una instancia de la clase *WindowOfQueryLoad.js*. Dicha clase contiene el explorador de modelos donde se escoge la consulta que se quiere abrir. Después en el método handler() se hace una llamada al método loadQuery() de la clase *UCQueryBuilder.js* y este a su vez llama al método loadXMLQuery() de la propia clase que se encarga de llamar directamente a la acción loadXml() de la clase controladora *QueryController.php* retornando el código de la consulta.

2.5.4. Vista de despliegue

El modelo de despliegue describe la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos de software. (Ivar Jacobson, 2000). A continuación se muestra el diagrama de despliegue correspondiente al editor de consultas MDX para el GDR v2.0.

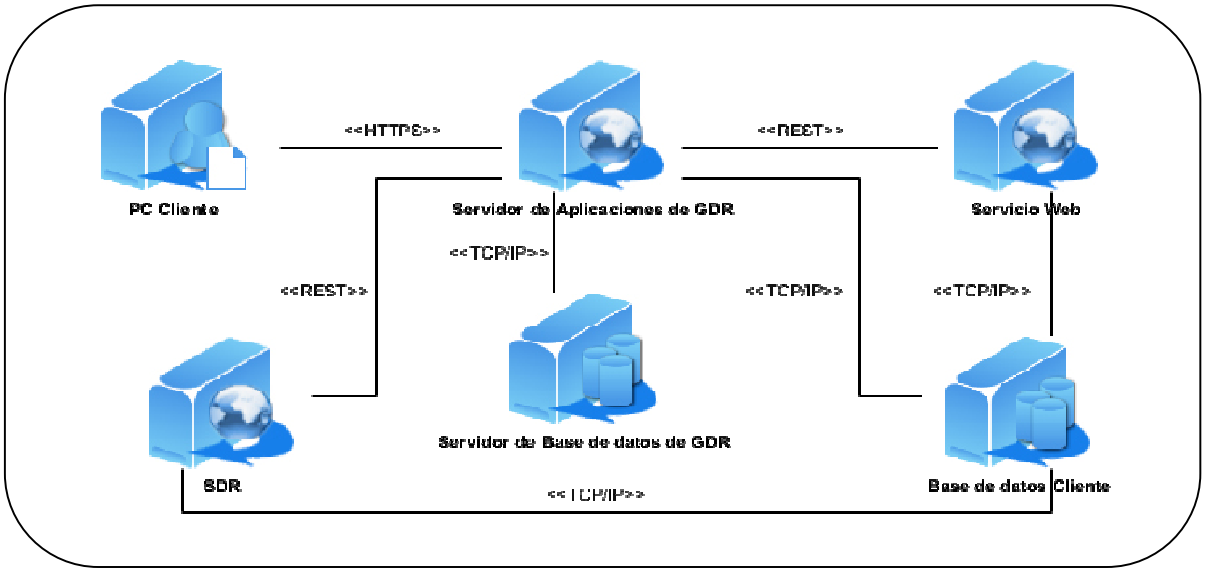


Fig. 14 Diagrama de despliegue

El diagrama de despliegue que se muestra en la figura 14 representa los seis nodos principales con sus respectivas conexiones:

- ✓ **PC Cliente:** computadora desde donde se ejecuta la URL, para acceder a la aplicación.
- ✓ **Servidor Aplicaciones GDR:** computadora donde encuentra publicada la aplicación.
- ✓ **Servidor de Base de datos de GDR:** contiene toda la información del GDR.
- ✓ **Base de datos Cliente:** contiene toda la información del cliente.
- ✓ **Servicio web:** contiene las librerías OLAP para administrar las consultas.
- ✓ **SDR:** servidor dinámico de reportes para generar los reportes diseñados en el GDR.
- ✓ **HTTPS:** protocolo de transferencia de hipertexto, conecta la computadora del cliente y el servidor de aplicaciones de GDR, donde se encuentra la aplicación.

- ✓ **TCP/IP:** protocolo de comunicación, conecta al servidor de aplicaciones y los nodos servidor de base de datos de GDR y servidor de base de datos cliente.
- ✓ **REST:** protocolo que representa una especificación del protocolo HTTP, conecta al servidor de aplicaciones de GDR y el nodo servidor web del SDR, además al servicio web con el de base de datos cliente.

Conclusiones del capítulo

En el presente capítulo se describen las fases de análisis y diseño del sistema. Comenzando por el modelo de dominio, el cual permite explicar los conceptos significativos en el dominio del problema. Se definieron 13 requisitos funcionales y 10 no funcionales, con el objetivo de precisar las funcionalidades que deberá realizar el editor de consultas MDX para GDR v2.0. Los requisitos funcionales se agruparon en cuatro casos de uso, los cuales se muestran en el diagrama de casos de uso del sistema. También se modelaron los diagramas de clases del diseño y de secuencia, representando la estructura estática y dinámica del sistema siguiendo el patrón arquitectónico Modelo-Vista-Controlador. Fueron identificados los patrones de diseño GRAPS y GOF presentes en el diseño e implementación de la aplicación. Además se modeló la vista de despliegue donde se muestra la estructura física de la solución.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Introducción

En el presente capítulo se describen las fases de implementación y prueba del sistema. Se modela el diagrama de componente que detalla la forma en que estará estructurado el editor de consultas, reflejando la transformación de los elementos del modelo del diseño en términos de componentes. Además se diseñan y aplican las pruebas de software con el objetivo de validar el correcto funcionamiento del editor.

3.1. Diagrama de componentes

El conjunto completo de componentes se define durante el diseño arquitectónico y especifica las estructuras de datos, los algoritmos, las características de la interfaz y los mecanismos de comunicación asignados a cada componente de un software. Muestra como el sistema está dividido en componentes y las dependencias entre ellos proveyendo una vista arquitectónica de alto nivel del sistema y ayudando a los desarrolladores a visualizar el camino de la implementación.

La figura 15 muestra el diagrama de componentes del Caso de Uso Administrar Consulta. En el expediente de proyecto del GDR versión 2.0 se encuentran los restantes diagramas.

El diagrama se estructura siguiendo la arquitectura Modelo-Vista-Controlador donde:

- ✓ **Vista**: contiene las interfaces de usuario asociadas al caso de uso Gestionar Modelo, convertidas en componentes de tipo <<file>>. La Vista tiene relación de dependencia con el <<subsystem>> ExtJS y Lycan.
- ✓ **Controlador**: contiene las acciones asociadas al caso de uso Gestionar Modelo, con las cuales se le da respuesta a las peticiones del usuario, convertidas en componentes de tipo <<file>>. El Controlador tiene relación de dependencia con el <<subsystem>> Symfony.
- ✓ **Modelo**: contiene las tablas de la base de datos asociadas al caso de uso Gestionar Modelo, convertidas en componentes de tipo <<file>>. El Modelo tiene relación de dependencia con el <<subsystem>>Doctrine.

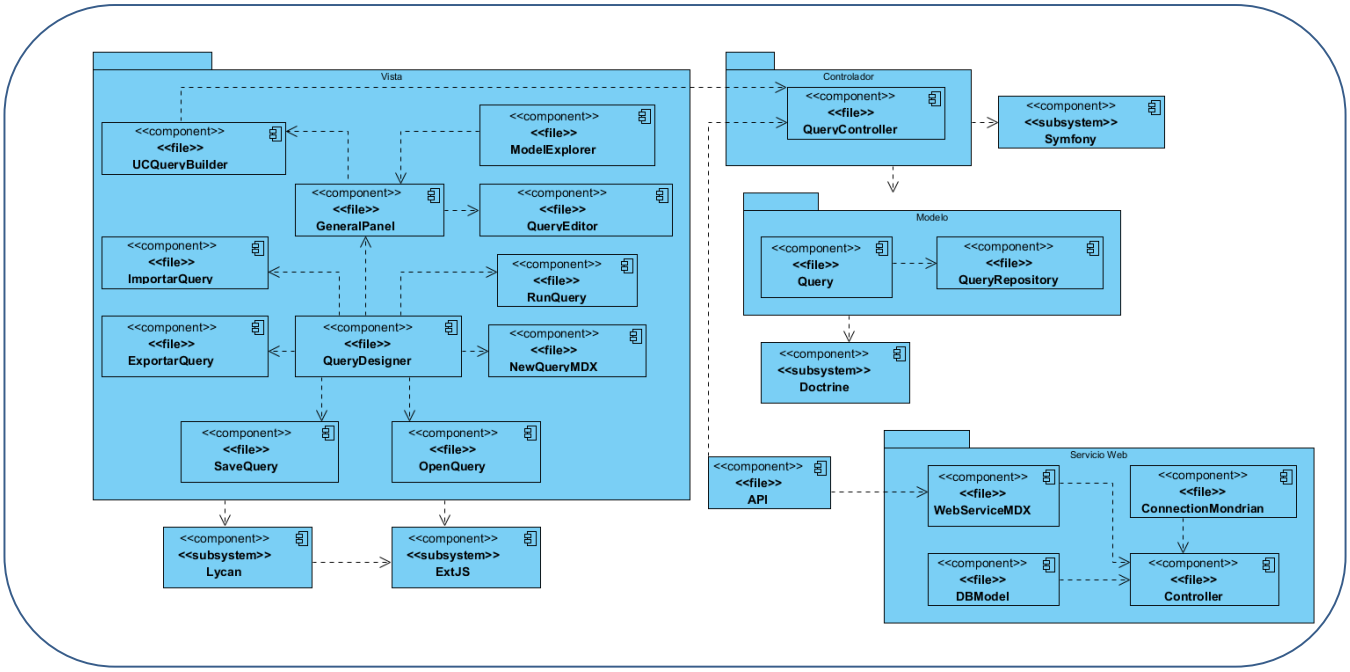


Fig. 15 Diagrama de Componentes del Caso de Uso Administrar Consulta

3.2. Estándares de codificación

Al programar en un lenguaje específico se deben seguir reglas que permitirán al desarrollador poder interpretar de manera eficiente la escritura del código. Un estándar de codificación completo comprende todos los aspectos de la generación de código.

El estándar de codificación seleccionado para el desarrollo de la aplicación es el CamelCase.

CamelCase: Es un estilo de escritura que se aplica a frases o palabras compuestas, dentro de este estilo existen dos tipos:

- ✓ **UpperCamelCase**: Las palabras que forman el nombre se escriben juntas y la primera letra de cada una de ellas en mayúscula.
- ✓ **LowerCamelCase**: Las palabras que forman el nombre se escriben juntas, la primera letra de la primera palabra en minúscula y del resto de las palabras, la primera letra en mayúscula.

Los estilos CamelCase se utilizaron durante la implementación del editor de consulta siguiendo la codificación que presenta el proyecto GDR. El uso del estándar UpperCamelCase se evidencia, al

nombrar las clases utilizadas en el desarrollo de la aplicación y LowerCamelCase, es utilizado para nombrar las funciones, variables y constantes.

3.2.1. Implementación del servicio web

Para la ejecución de consultas MDX se integra un servicio web al GDR v2.0. Las peticiones remotas al servicio desplegado en el servidor Apache Tomcat se realizaron utilizando la librería cURL de PHP empleando el método POST.

```
$file_dir = $this->container->getParameter('cubos.olap.directory');
$context = $service->ExecuteQuery(
        $meta->driver,
        $dataSource->getHost(),
        $dataSource->getPort(),
        $dataSource->getDatabase(),
        $dataSource->getUsername(),
        $dataSource->getPassword(),
        $codeSQL, $file_dir . $fileName, ""
);
```

Fig. 16 Código de integración

3.2.2. Implementaciones Relevantes

El siguiente fragmento de código pertenece a la clase *QueryController.php*. El método exportMdxAction() pertenece a dicha clase, fue implementado con el propósito de exportar las consultas MDX.

```
public function exportMdxAction() {
    try {
        $content = $this->getRequest()->get('query');
        $name = $this->getRequest()->get('name');

        $response = new Response($content);

        $response->headers->set('Content-Type', 'application/octet-stream');
        $response->headers->set('Content-Description', 'File Transfer');
        $response->headers->set('Cache-Control', 'private, must-revalidate, post-check=0, pre-check=0, max-age=1');
        $response->headers->set('Pragma', 'public');
        $response->headers->set('Last-Modified', gmdate('D, d M Y H:i:s') . ' GMT');

        $response->headers->set('Content-Disposition', 'attachment; filename="' . $name . '.mdx"');
        $response->headers->set('Content-Transfer-Encoding', 'binary');
        $response->setStatusCode(202);

        return $response;
    } catch (\Exception $e) {
        print $e->getMessage();
    }
}
```

Fig. 17 Ejemplo de código relevante

3.3. Resultado de la implementación

Fueron implementados los 13 requisitos funcionales definidos en la etapa de diseño. El editor de consultas MDX requiere adicionar un nuevo origen de datos OLAP en el módulo Diseñador de Modelos para la edición de consultas MDX. Partiendo de un origen de datos OLAP es posible construir un modelo, que permite seleccionar los cubos de datos con los que se desea trabajar. El modelo creado muestra una jerarquía con las dimensiones y medidas de los cubos seleccionados. En el módulo Diseñador de Consultas se crea una consulta MDX que puede ser guardada en el paquete Consultas y el resultado mostrado en la tabla de resultados. A raíz del modelo se procede a crear un reporte multidimensional en el módulo Diseñador de Reportes utilizando la consulta seleccionada para luego exportarlo en el módulo Visor de reportes. En la figura 18 se evidencia la gestión de consultas MDX en el Generador Dinámico de Reportes v2.0.

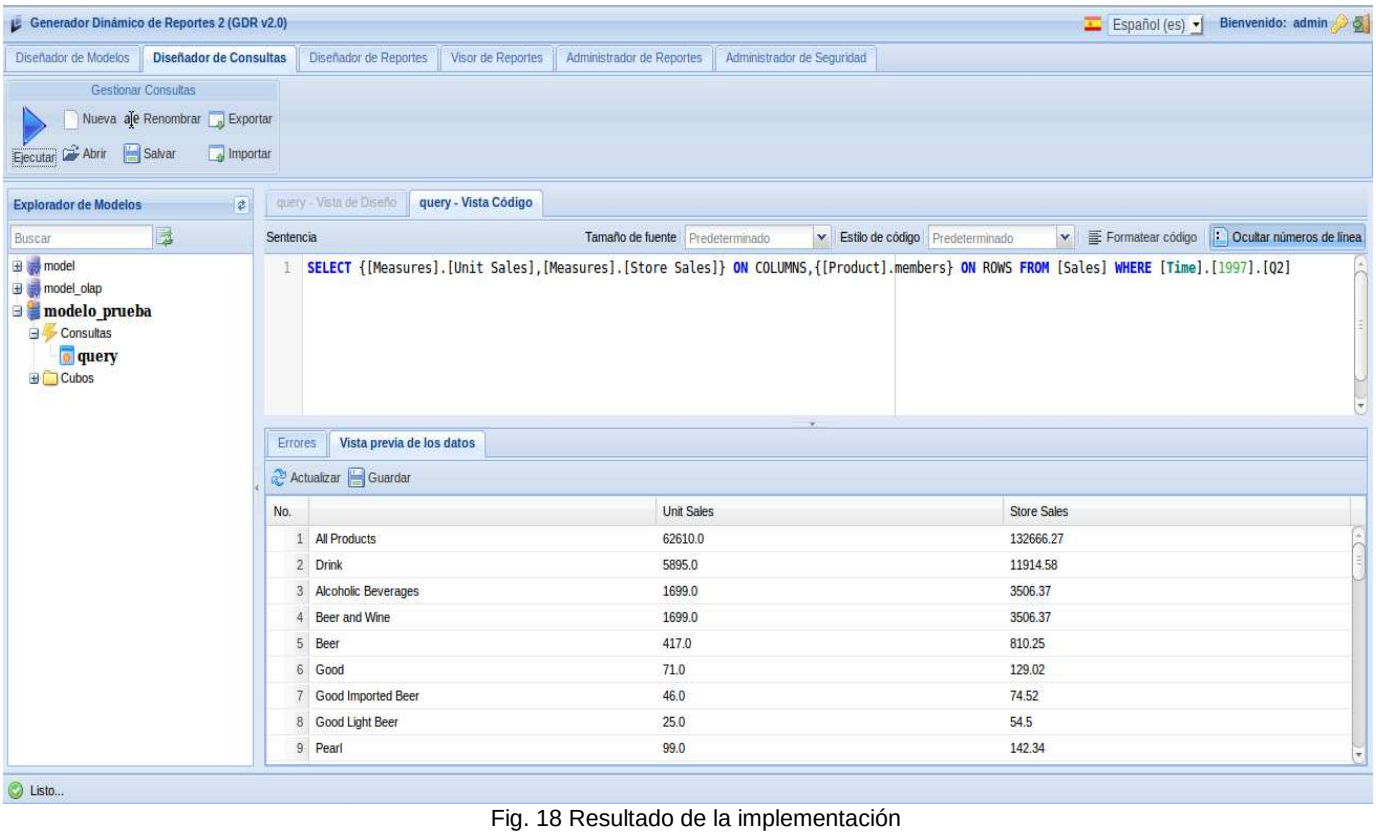


Fig. 18 Resultado de la implementación

3.4. Pruebas del software

Es el conjunto de actividades que se planean con anticipación y se realizan de manera sistemática. Por tanto es necesario definir un conjunto de pasos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba. Las pruebas son el último bastión para la evaluación de la calidad y de manera más pragmática, el descubrimiento de errores. (Pressman, 2011)

3.4.1. Niveles de Prueba

Las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos escenarios por ello se distinguen los siguientes niveles de pruebas: prueba de desarrollador, unidad, integración, sistema y aceptación. El editor de consultas MDX para el GDR v2.0 fue probado aplicando las pruebas a nivel de desarrollador, integración, sistema y aceptación.

Pruebas de desarrollador: se realizan con el objetivo de detectar errores en la implementación de requerimientos.

Pruebas de integración: se realizan con el objetivo de detectar errores de interfaces y relaciones entre componentes.

Pruebas de sistema: se realizan con el objetivo de detectar fallas en el cubrimiento de los requerimientos.

Pruebas de aceptación: se realizan con el objetivo de detectar fallas en la implementación del sistema.

3.4.2. Tipos de prueba

Pruebas funcionales: Las pruebas funcionales tienen como objetivo probar que los sistemas desarrollados cumplan con las funciones específicas para los cuales han sido creados.

Pruebas de integración: Las pruebas de integración tienen como objetivo tomar componentes individualmente probados y verificar que en conjunto las partes del software funcionan correctamente integradas.

Pruebas de rendimiento: Las pruebas de rendimiento se basan en comprobar que el sistema puede soportar el volumen de carga definido en la especificación de eficiencia.

Para medir el rendimiento se realizan diferentes tipos de pruebas:

Pruebas de carga: Una prueba de carga se realiza generalmente para observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas.

Las pruebas de aceptación: Las pruebas de aceptación son realizadas con el cliente y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos y del manual del usuario. Estas pruebas no se realizan durante el desarrollo, sino una vez pasadas todas las pruebas por parte del desarrollador.

3.4.3. Método y técnica

Método Caja negra: Las pruebas de caja negra son aplicadas a la interfaz del software. Una prueba de este tipo examina algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software. (Pressman, 2011)

Técnica Partición Equivalente: Esta técnica de prueba de caja negra divide el dominio de entrada de un programa en clases de datos, a partir de las cuales deriva los casos de prueba.

3.4.4. Diseño de Caso de Prueba

En la siguiente tabla se detallan las variables que se encuentran asociadas al Caso de Uso Administrar Consulta.

Tabla 3 Descripción de las variables

No.	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Nombre de la consulta	Campo de texto	No	Admite la entrada de letras mayúsculas y minúsculas, números y guión bajo, comenzando siempre por caracter alfabético.
2	Consulta	Campo de texto	No	Cadena MDX. Cadena entre 1 y 255 caracteres.
3	Exportar archivo de consulta	Campo de exportar archivo	N/A	Fichero que contiene el código de la consulta MDX que se va a exportar.
4	Importar archivo de consulta	Campo de cargar archivo	No	Fichero que contiene el código de la consulta MDX que se va a importar.

5	Exportar archivo resultado de consulta	Campo de exportar archivo	No	Fichero que contiene el resultado de la consulta MDX

Se realizó una matriz de datos donde se evaluó y probó la validez de las entradas del editor de consultas utilizando un juego de datos válidos e inválidos para crear los diferentes casos de prueba. En la Tabla 4 se ilustra el escenario Abrir Consulta.

Tabla 4 Matriz de datos

Escenario	**Variables**					**Descripción**	**Respuesta del sistema**	**Flujo central**
	1	**2**	**3**	**4**	**5**			
EC 2.1 Abrir consulta existente.	NA	V	NA	NA	NA	El sistema debe permitir abrir una consulta	El sistema muestra la sentencia MDX de la consulta solicitada por el actor.	1. El actor selecciona el módulo “Diseñador de Consultas”. 2. El actor selecciona la opción “Abrir” del menú Gestionar Consultas. 3. El sistema muestra una interfaz que visualiza un listado con las consultas. 4. El actor selecciona la consulta a abrir y oprime el botón “Aceptar”.
EC 2.2 Cancelar operación	NA	NA	NA	NA	NA	Se cancela la operación de abrir la consulta.	El sistema cierra la interfaz para abrir una consulta y muestra nuevamente la interfaz de edición de	1. El actor selecciona el módulo “Diseñador de Consultas”. 2. El actor selecciona la opción “Abrir consulta” del menú Gestionar Consultas. 3. El sistema muestra una interfaz que visualiza un listado

							consulta	con las consultas registradas. 4. El actor oprime el botón "Cancelar"

3.4.5. Resultados de las pruebas

Se realizaron tres iteraciones de pruebas funcionales, aplicadas a los cuatro casos de prueba diseñados arrojando un total de 11 NC (No Conformidades). En la primera iteración se probaron diez RF, detectándose un total de ocho NC, de ellas, dos fueron funcionales, tres de validación y tres de ortografía. En la segunda iteración se detectaron tres NC, de ellas dos de validación y una de ortografía. En una tercera iteración se comprobó que las NC de las iteraciones anteriores estuvieran resueltas, no se detectaron no conformidades. A continuación se muestra un gráfico con el número de iteraciones realizadas y la cantidad de NC detectadas en cada iteración.

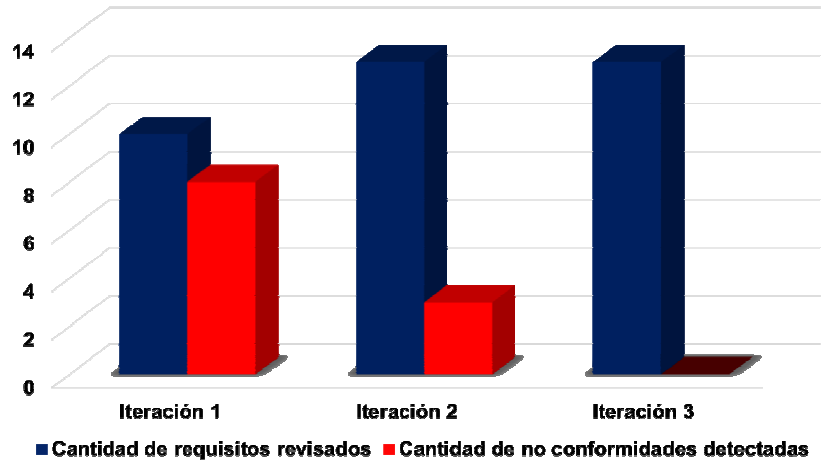


Fig. 19 Iteraciones de pruebas de desarrollador

Para ejecutar la prueba de integración del componente servicio web al GDR v2.0 se decide realizar una prueba de integración incremental, que permite probar en pequeños incrementos, en los cuales resulta más fácil aislar y corregir los errores. La integración incremental ascendente empieza la construcción y la prueba con los componentes más bajos de la estructura del programa. El servicio web se fue probando gradualmente con respecto a los requisitos funcionales que lo utilizan. En un primer momento se integró con el requisito funcional salvar consulta, en dicho proceso se fue adaptando la conexión al servicio y luego el formato de retorno de respuesta que se visualiza. Para conseguir la integración con el requisito

funcional ejecutar consulta se trabajó en la visualización de los datos de la consulta enviados desde el servicio, visualizándolos en la tabla de resultados definida en el módulo Administrar Consulta.

Una vez integrado el componente de conexión al Generador Dinámico de Reportes, se puede comprobar su correcto funcionamiento. Cuando el usuario ejecuta una consulta, el sistema muestra el resultado del análisis solicitado, la acción se realiza enviando una petición al servicio web implementado, que tiene la función de ejecutar la consulta y devolver los datos al sistema de forma tal que el flujo no sea interrumpido. Una vez creado el modelo y guardada alguna consulta el usuario puede acceder al módulo Diseñador de Reportes para crear el reporte y seguidamente pasar al módulo Visor de Reportes para exportarlo.

Se realizaron pruebas de rendimiento utilizando la herramienta JMeter 2.9, simulando las peticiones de consultas MDX hechas al editor por varios usuarios conectados concurrentemente. Estas pruebas fueron realizadas utilizando una computadora haciendo función de PC cliente, Servidor de Aplicaciones, Servidor de base de datos de GDR y Servidor de base de datos del cliente. A continuación se presentan las prestaciones de las mismas:

- ✓ Ordenador Intel Core i3, con 2.53 GHz de velocidad de microprocesador.
- ✓ Memoria RAM de 2 GB.

Tabla 5 Resultado de la prueba de rendimiento

Cantidad de usuarios conectados concurrentemente	Tiempo(s)
10	25,728
30	35,556
50	49,904
80	70,785
100	79,368

Se concluye que la aplicación responde en 25.728 segundos a las peticiones de 10 usuarios concurrentes que fueron definidos en los requisitos no funcionales, siendo este óptimo pues los tiempos de respuesta

deben estar por debajo de los 30 segundos. Se midió el tiempo de respuesta de la aplicación para una mayor cantidad de usuarios, obteniéndose que no responde adecuadamente pues excede los 30 segundos.

Para que conste la aceptación del producto, se lleva a cabo la prueba en el departamento de Desarrollo de Componentes, por el cliente y con especialistas que laboran en el desarrollo del GDR v2.0. La aceptación del editor de consultas MDX se materializó a través de la carta de aceptación del cliente al producto desarrollado. Ver Anexo 4.

Conclusiones del capítulo

En el presente capítulo se realizó la descripción de la implementación del editor de consultas MDX para el GDR v2.0. Como parte de este se diseñó el diagrama de componentes asociado al caso de uso Administrar Consulta para mostrar una representación de los componentes implementados. Se especificó el uso de los estándares de codificación para lograr un estilo claro y organizado del código durante la fase de construcción. Se realizaron pruebas de desarrollador a través del método de caja negra, aplicando la técnica de partición de equivalente. Además de comprobar la integración del componente servicio web al GDR v2.0 y la aceptación dada por el cliente; garantizando un producto sin errores y con un correcto funcionamiento.

CONCLUSIONES GENERALES

Una vez culminada la investigación se puede afirmar que se le dio cumplimiento a los objetivos planteados, arribando a las siguientes conclusiones:

- ✓ El análisis de los conceptos fundamentales de la investigación y el estado del arte de las herramientas OLAP integradas a los generadores de reportes para la gestión de consultas MDX, posibilitaron adquirir los conocimientos necesarios para dar comienzo a la fase de elaboración.
- ✓ A partir del análisis de tecnologías actuales para el desarrollo de software y bajo el principio de emplear tecnologías libres, se seleccionaron lenguajes, marco de trabajo, entorno de desarrollo y metodología a utilizar para la implementación del editor de consultas MDX para GDR v2.0
- ✓ A partir de la realización del análisis y diseño del editor de consultas MDX para GDR v2.0 se obtuvo como resultado los diagramas y artefactos necesarios para guiar el desarrollo del mismo.
- ✓ La implementación del editor de consultas MDX para GDR v2.0 dio cumplimiento a los 13 requisitos funcionales identificados en las fases de análisis y diseño.
- ✓ El diseño y ejecución de las pruebas permitió comprobar el correcto funcionamiento del editor de consultas MDX para GDR v2.0
- ✓ Como resultado se obtuvo el editor de consultas MDX para GDR v2.0 que permite efectuar la gestión de consultas MDX.

RECOMENDACIONES

Luego de haber analizado los resultados de la investigación se recomienda:

- ✓ Incorporar un diseñador visual de consultas MDX al módulo Diseñador de Consultas de GDR v2.0.
- ✓ Agregar la funcionalidad Leer Campos para el diseño de reportes multidimensionales en el módulo Diseñador de Reportes de GDR v2.0.
- ✓ Integrar el servicio web implementado al SDR como parte de los servicios consumidos por el GDR v2.0.

REFERENCIAS BIBLIGRÁFICAS

1. **Beltrá, Carlos Patricio López. 2007.** *Análisis, Diseño e implementación de un Data Mart para la Dirección Financiera y Recursos Humanos de la Escuela Politécnica del Ejército para una toma de decisión efectiva.* 2007.
2. **Bernabue, Ing. Ricardo Dario Córdoba. 2009.** *Data Warehousing: Investigación y Sistematización de Conceptos – Hefesto: Metodología propia para la Construcción de un Data Warehouse.* Argentina : s.n., 2009.
3. **Craig, Larman. 1999.** *UML y Patrones.* Mexico : Pearson, 1999.
4. **Eguiluz, Javier. 2014.** *Desarrollo aguil de symfony.* 2014.
5. **Espinosa, Roberto. 2010.** Pentaho Shema Worbench. [En línea] 2010. [Citado el: 29 de Octubre de 2013.] http://churriwifi.wordpress.com/2010/07/04/17-3-preparando-el-analisis-dimensional-definicion-de-cubos-utilizando-schema-workbench.
6. **Fernández, Carlos. 2013.** Dataprix. OLAP, MOLAP y ROLAP. [En línea] 2013. [Citado el: 1 de Diciembre de 2013.] http://www.dataprix.com/olap-rolap-molap.
7. **Gallardo, Erith Eduardo Pérez. 2012.** Data Warehouse, Modelo, Conceptos e Implementación orientada a SQL Server. [En línea] 2012. [Citado el: 15 de Noviembre de 2013.] http://www.monografias.com/trabajos57/data-warehouse-sql/data-warehouse-sql2.shtml.
8. **Herrera, Cristhian. 2007.** Adictos al trabajo. [En línea] 2007. [Citado el: 10 de Noviembre de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=datawarehouse4.
9. **IBM. 2013.** Creación de modelos dimensionales lógicos y físicos. [En línea] 2013. [Citado el: 16 de Noviembre de 2013.] http://pic.dhe.ibm.com/infocenter/idm/docv3/index.jsp?topic=%2Fcom.ibm.datatools.dimensional.ui.doc%2Ftopics%2Ft_ldm_pdm_dim_not_wizard.html.
10. **Inmon, Willian Bill. 2002.** *Building Data Warehouse.* 2002.
11. **Kimball, Ralph. 2002.** *The Data Warehouse ETL Toolkit. Second Edition.* 2002.
12. **Lobo, Armando Robert. 2010.** *Lycan IDE.* 2010.

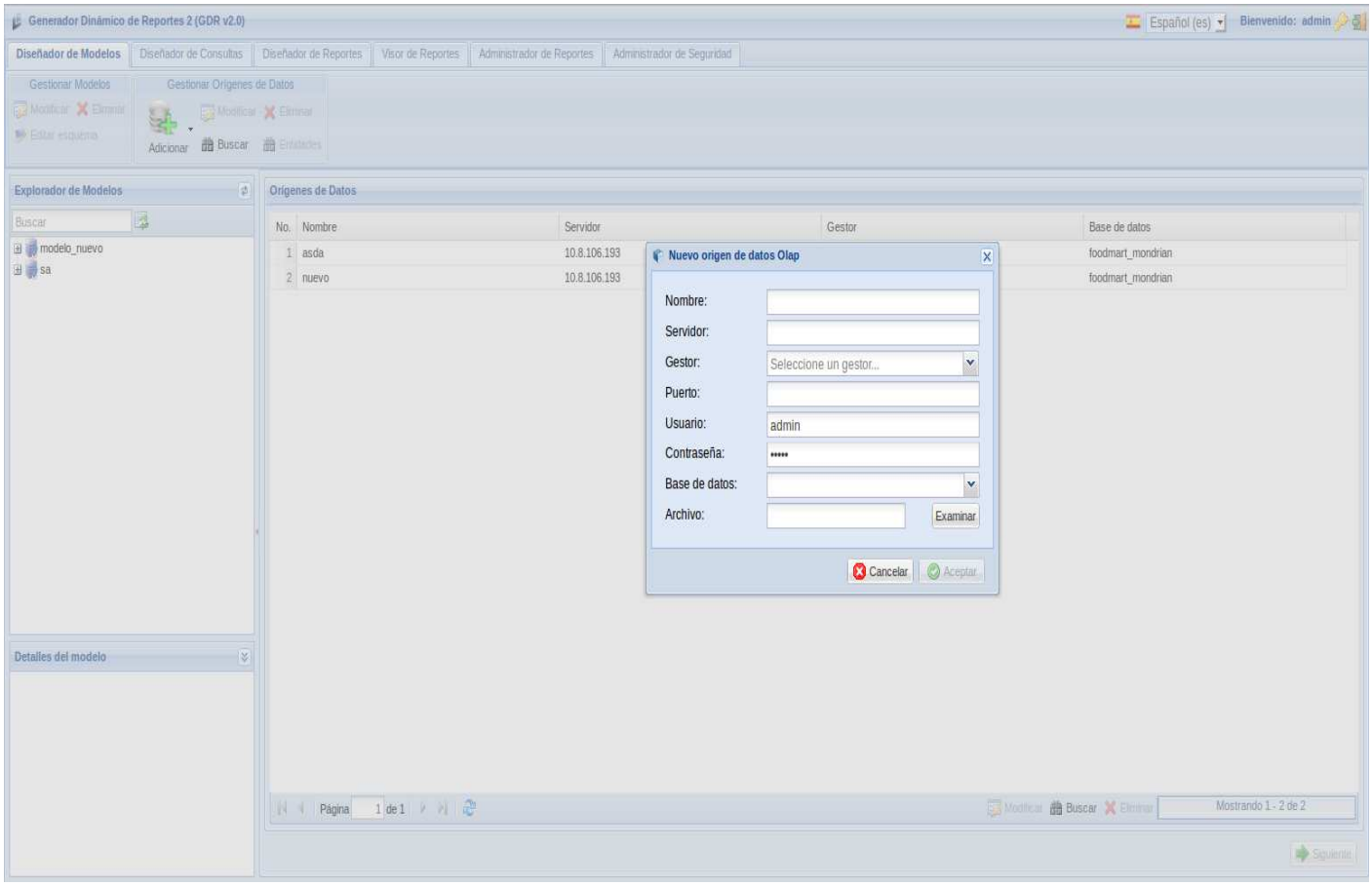
13. **Medina, Cdte. Gustavo Moya Pérez - Darisleidy Arcia. 2012.** *Modelación, Implementación e Integración de la Herramienta de Gestión del XML para Almacenes de Datos.* La Habana : s.n., 2012.
14. *OpenUp.* **OpenUp. 2012.** 2012.
15. **Oracle Corporation. 2014.** *NetBeans IDE (Entorno Desarrollo de Integrado).* 2014.
16. **paradigm, Visual. 2010.** UML and Business Process modeling tool for software development project -Visual Paradigm for UML. [En línea] 2010. [Citado el: 20 de Noviembre de 2013.] http://www.visual-paradigm.com/product/vpuml/index.jsp.
17. *PgAdmin III.* **Givology. 2008.** 2008.
18. **PHP Group. 2001.** *Manual PHP pagina oficial.* 2001.
19. **Potencier, Fabien. 2007.** *Symfony la guia definitiva.* 2007.
20. **Reuse, Knowledge. 2013.** Madrid : s.n., 2013.
21. **Sánchez, Jorge. 2003.** *Manual JavaScript.* 2003.
22. **SAP Business Objects Enterprise . 2010.** *Manual del usuario de la Integración de SAP BusinessObjects Enterprise para soluciones SAP.* 2010.
23. **Sinnexus. 2012.** [En línea] 2012. [Citado el: 28 de Noviembre de 2013.] http://www.sinnexus.com/business_intelligence/olap_avanzado.aspx.
24. *Sobre PostgreSQL.* **Martinez, Rafael. 2009.** 2009.
25. **Somerville, Ian. 2005.** *Ingenieria de software 7ma edicion.* Madrid : Pearson, 2005.
26. **Summan. 2014.** [En línea] 2014. [Citado el: 4 de Abril de 2014.] http://www.summan.com/pentaho/pentaho-bi-platform-server.
27. *Una Introducción a APACHE.* **González, Guillermo. 2014.** 2014.
28. **Zorrilla, Marta. 2008.** *Data warehouse y OLAP.* Universidad de Cantabria : s.n., 2008.

BIBLIOGRAFÍA

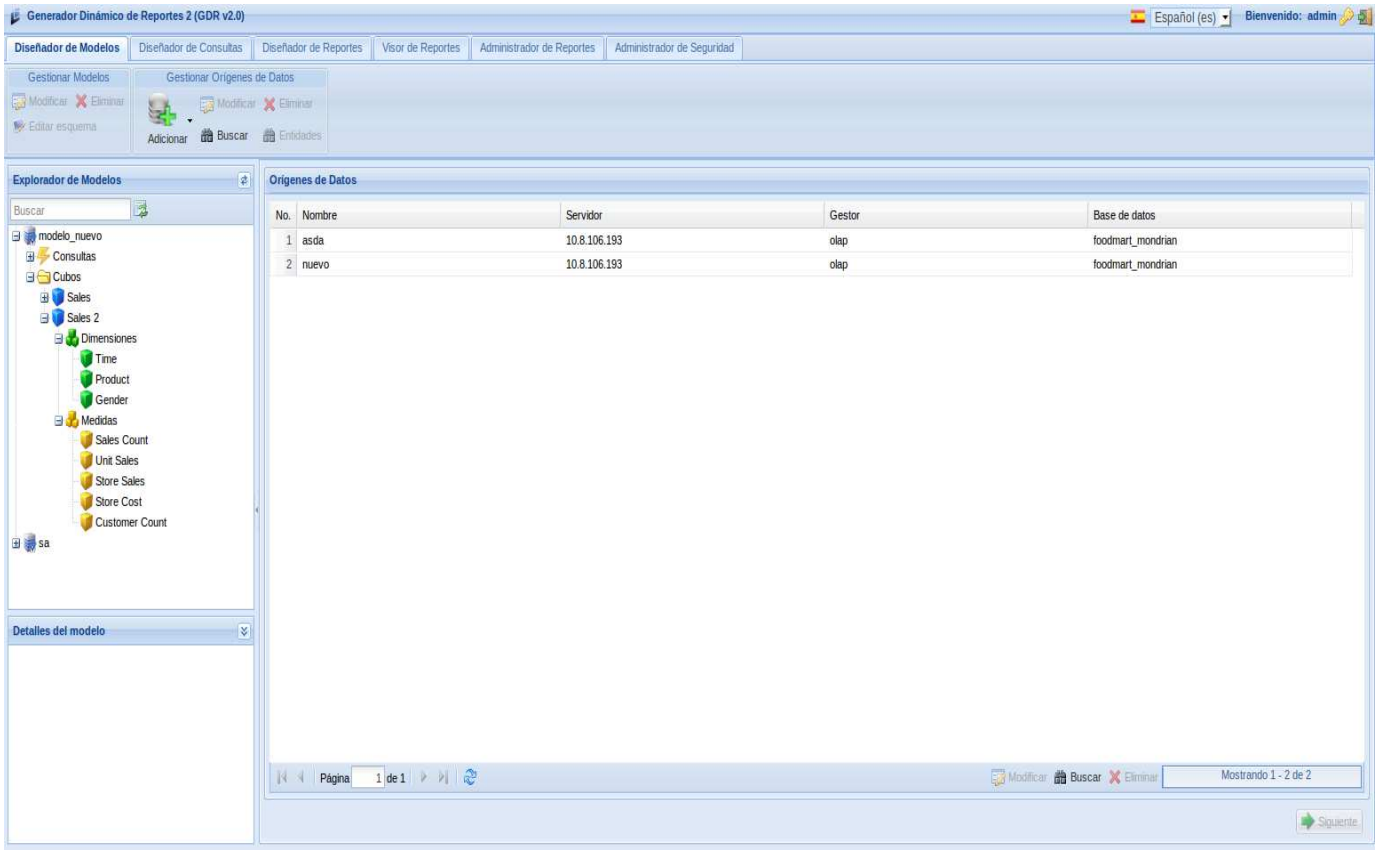
1. **Balduino, Ricardo. 2007.** *Configuracion de la metodologia Open UP.* Sao Paulo : s.n., 2007.
2. **Beltrá, Carlos Patricio López. 2007.** *Análisis, Diseño e implementación de un Data Mart para la Dirección Financiera y Recursos Humanos de la Escuela Politécnica del Ejército para una toma de decisión efectiva.* 2007.
3. **Bernabue, Ing. Ricardo Dario Córdoba. 2009.** *Data Warehousing: Investigación y Sistematización de Conceptos – Hefesto: Metodología propia para la Construcción de un Data Warehouse.* Argentina : s.n., 2009.
4. **Craig, Larman. 1999.** *UML y Patrones.* Mexico : Pearson, 1999.
5. **Eguiluz, Javier. 2014.** *Desarrollo aguil de symfony.* 2014.
6. **Espinosa, Roberto. 2010.** Pentaho Shema Worbench. [En línea] 2010. [Citado el: 29 de Octubre de 2013.] http://churriwifi.wordpress.com/2010/07/04/17-3-preparando-el-analisis-dimensional-definicion-de-cubos-utilizando-schema-workbench.
7. **Fernández, Carlos. 2013.** Dataprix. OLAP, MOLAP y ROLAP. [En línea] 2013. [Citado el: 1 de Diciembre de 2013.] http://www.dataprix.com/olap-rolap-molap.
8. **Gallardo, Erith Eduardo Pérez. 2012.** Data Warehouse, Modelo, Conceptos e Implementación orientada a SQL Server. [En línea] 2012. [Citado el: 15 de Noviembre de 2013.] http://www.monografias.com/trabajos57/data-warehouse-sql/data-warehouse-sql2.shtml.
9. *Generador dinámico de reportes.* **Centro de Tecnologías de Gestión de Datos. Universidad de las Ciencias Informáticas. 2012.** La Habana : s.n., 2012.
10. **Herrera, Cristhian. 2007.** Adictos al trabajo. [En línea] 2007. [Citado el: 10 de Noviembre de 2013.] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=datawarehouse4.
11. **IBM. 2013.** Creación de modelos dimensionales lógicos y físicos. [En línea] 2013. [Citado el: 16 de Noviembre de 2013.] http://pic.dhe.ibm.com/infocenter/idm/docv3/index.jsp?topic=%2Fcom.ibm.datatools.dimensional.ui.doc%2Ftopics%2Ft_ldm_pdm_dim_not_wizard.html.
12. **Inmon, Willian Bill. 2002.** *Building Data Warehouse.* 2002.
13. **Kimball, Ralph. 2002.** *The Data Warehouse ETL Toolkit. Second Edition.* 2002.

14. **Lobo, Armando Robert. 2010.** *Lycan IDE.* 2010.
15. **Marset, Rafael Navarro. 2007.** *Modelado, Diseño e Implementación de Servicios Web.* 2007.
16. **Medina, Cdte. Gustavo Moya Pérez - Darisleidy Arcia. 2012.** *Modelación, Implementación e Integración de la Herramienta de Gestión del XML para Almacenes de Datos.* La Habana : s.n., 2012.
17. *OpenUp.* **OpenUp. 2012.** 2012.
18. **Oracle Corporation. 2014.** *NetBeans IDE (Entorno Desarrollo de Integrado).* 2014.
19. **paradigm, Visual. 2010.** UML and Business Process modeling tool for software development project -Visual Paradigm for UML. [En línea] 2010. [Citado el: 20 de Noviembre de 2013.] http://www.visual-paradigm.com/product/vpuml/index.jsp.
20. *PgAdmin III.* **Givology. 2008.** 2008.
21. **PHP Group. 2001.** *Manual PHP pagina oficial.* 2001.
22. **Potencier, Fabien. 2007.** *Symfony la guia definitiva.* 2007.
23. **Pressman, Roger S. 2011.** Ingenieria de software un enfoque practico. 6ta edici . 2011.
24. **Reuse, Knowledge. 2013.** Madrid : s.n., 2013.
25. **Sánchez, Jorge. 2003.** *Manual JavaScript.* 2003.
26. **SAP Business Objects Enterprise . 2010.** *Manual del usuario de la Integración de SAP BusinessObjects Enterprise para soluciones SAP.* 2010.
27. **Sencha ExtJS. 2015.** *Sencha ExtJS.* 2015.
28. **Sinnexus. 2012.** [En línea] 2012. [Citado el: 28 de Noviembre de 2013.] http://www.sinnexus.com/business_intelligence/olap_avanzado.aspx.
29. *Sobre PostgreSQL.* **Martinez, Rafael. 2009.** 2009.
30. **Somerville, Ian. 2005.** *Ingenieria de software 7ma edicion.* Madrid : Pearson, 2005.
31. **Summan. 2014.** [En línea] 2014. [Citado el: 4 de Abril de 2014.] http://www.summan.com/pentaho/pentaho-bi-platform-server.
32. *Una Introducción a APACHE.* **González, Guillermo. 2014.** 2014.
33. **Zorrilla, Marta. 2008.** *Data warehouse y OLAP.* Universidad de Cantabria : s.n., 2008.

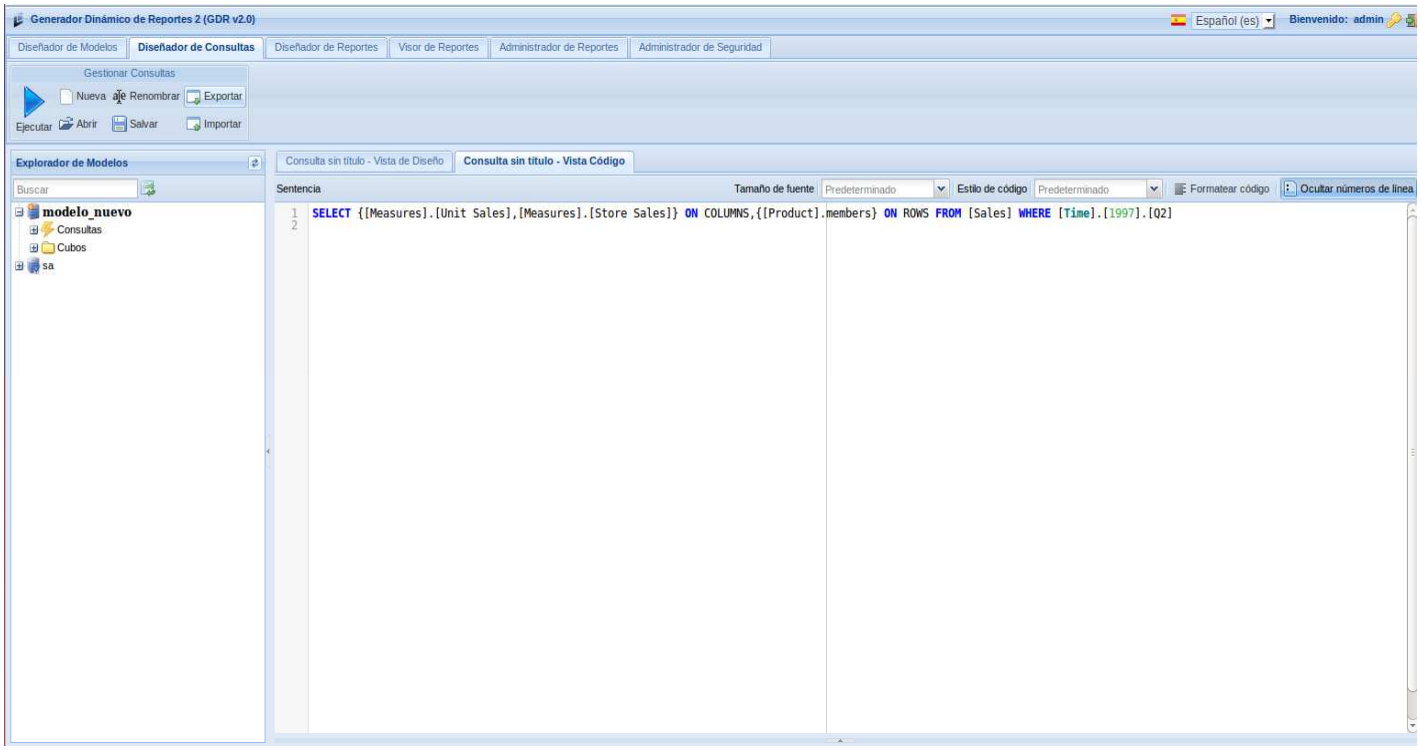
ANEXOS



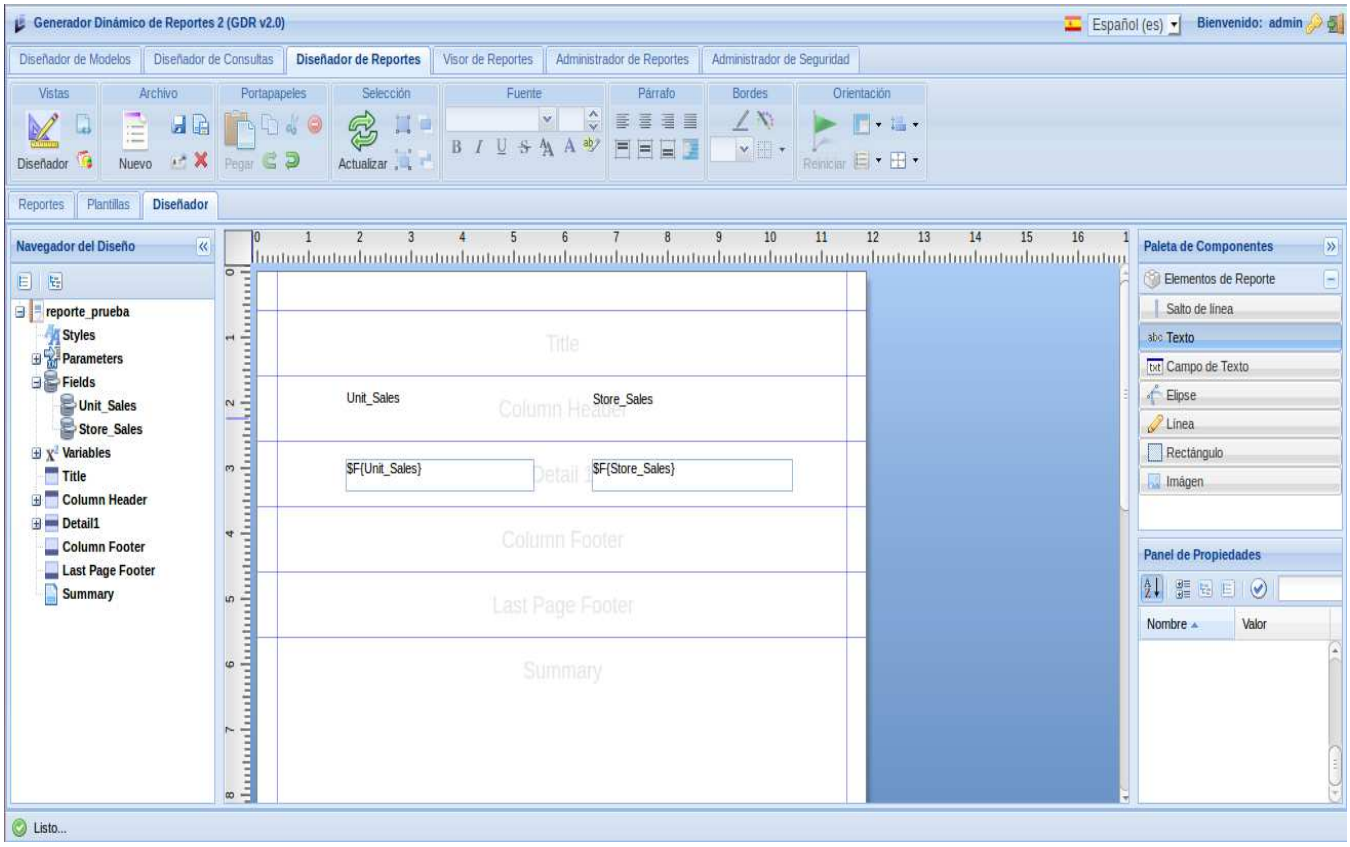
Anexo 1 Interfaz para Gestionar origen de datos



Anexo 2 Interfaz para para la administración del modelo



Anexo 3 Interfaz para administrar consulta



Anexo 3 Interfaz para el diseño de reportes

UCI Universidad de las Ciencias Informáticas

CARTA DE ACEPTACIÓN

En cumplimiento con la fase de desarrollo y en función de la ejecución del proyecto: Generador Dinámico de Reportes (GDR v2.0) para el Departamento de Desarrollo de Componentes del Centro de Tecnología de Gestión de Datos de la facultad 6: DATEC, se hace entrega de los productos que se relacionan a continuación:

-Editor de Consultas MDX para el Generador Dinámico de Reportes 2.0 (código fuente).

Entrega	**Recibe**	**Recibe**
Nombre y apellidos: Sonia Hidalgo García Yoan Gainza Labrada	**Nombre y apellidos**: Glennis Tamayo Morales	**Nombre y apellidos**: Yoander Iñiguez Bermúdez
Cargo: Tesistas	**Cargo:** Jefe de Departamento Desarrollo de Componentes	**Cargo:** Jefe de proyecto GDRv2.0
Firma:	**Firma:**	**Firma:**

UCI Universidad de las Ciencias Informáticas Decanato Facultad 6

Fecha: 15/06/2015

Anexo 4 Acta de aceptación