Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.6, Mes Junio, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-15
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: Soluciones informáticas
Recibido: 12/03/2019 | Aceptado: 18/04/2019 | Publicado: 20/06/2019

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

Graphic designer of MDX querys for Dynamic Generator of Reports v2.0

Daynelis Brito Morales [1*], Carlos Fortún Manzano [2]

[1]Centro de Tecnologías y Gestión de Datos, Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. Cuba. dbmorales@uci.cu

[2]Centro de Tecnologías y Gestión de Datos, Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas. Cuba. cfortun@uci.cu

*Autor para correspondencia: dbmorales@uci.cu

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se trabaja en el desarrollo de varios proyectos destinados a la informatización de los principales procesos de la sociedad cubana y de otros países del mundo. Formando parte de este conjunto de proyectos se encuentra el Generador Dinámico de Reportes v2.0 (GDR) que surge como estrategia trazada por el Centro de Tecnologías de Gestión de Datos (DATEC). En este proyecto se desarrolla una herramienta con este mismo nombre que permite, partiendo de información persistente en algún origen de datos, generar reportes de forma dinámica apoyando así la toma de decisiones. Actualmente las funcionalidades que posee el Editor de consultas MDX con que cuenta GDR v2.0 no permite a los usuarios que carecen de conocimientos sobre este lenguaje realizar consultas. El presente trabajo de diploma tiene como objetivo desarrollar un Diseñador gráfico de consultas MDX para GDR v2.0 que facilite esta tarea. Con el estudio realizado sobre los diseñadores de consultas existentes en

Cuba y en el mundo se definió la estructura básica y principales funcionalidades del diseñador gráfico a realizar. Se seleccionó la arquitectura Modelo Vista Controlador como arquitectura del sistema y OpenUp como metodología de software, así como las distintas herramientas y tecnologías que fueron empleadas en el desarrollo de la solución. Se definió una estrategia de prueba quedando plasmados los niveles de prueba de unidad, aceptación e integración, así como los tipos de prueba funcionales y aceptación, y el método de prueba escogido fue caja negra con la técnica partición equivalente, todo con el objetivo de encontrar y corregir posibles errores.

Palabras clave: consultas; diseñador; MDX; multidimensionales; reportes.

Abstract

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

Keywords: *queries; designer; MDX; multidimensional; reports.*

Introducción

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

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

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

El realizar consultas MDX desde GDR v2.0 tiene un alto impacto, ya que simplifica el proceso de generación de reportes multidimensionales realizados sobre almacenes de datos, lo que anteriormente, clientes como la Oficina Nacional de Estadísticas e Información, la Aduana General de la República de Cuba y la Contraloría General de la República de Cuba, que manejan enormes volúmenes de datos, requerían el uso del Pentaho Bi-Server[1]. Esta situación no solo implicaba el consumo de tiempo extra sino que requería la instalación y configuración de una aplicación externa al sistema. Además, era indispensable el dominio preciso de dicha herramienta para realizar estas funciones. Debido a esto en el 2015 fue desarrollado un Editor de consultas MDX el cual permite la realización de estas sobre cubos OLAP.

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

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

Dada la situación problemática anteriormente planteada, se define como **problema de la investigación:** Los usuarios que utilizan el Editor de consultas MDX de GDR v2.0 requieren de conocimientos avanzados de este lenguaje para el diseño de las mismas. Para dar cumplimiento al problema planteado se define como **objeto de estudio:** diseñadores de consultas, enmarcado en el **campo de acción:** diseñadores gráficos de consultas MDX para GDR v2.0.
El presente trabajo tiene como **objetivo general:** Desarrollar un diseñador gráfico de consultas MDX para GDR v2.0 que permita la generación de reportes sobre bases de datos multidimensionales.

Materiales y métodos o Metodología computacional

Desarrollar un *software* implica que sea necesario definir las tecnologías, herramientas y metodología de desarrollo que se van a emplear para lograr el objetivo planteado (Cendejas, 2011). Para la selección de las herramientas, tecnologías y metodología de desarrollo a emplear en el desarrollo de la solución se tuvieron en cuenta las definidas en el marco del proyecto GDR v2.0.

Metodología de desarrollo de software:

OpenUp

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

ExtJS 3.4

ExtJS es una biblioteca de JavaScript desarrollada por Sencha, para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. ExtJS es el marco más amplio de JavaScript para crear aplicaciones web multiplataforma con múltiples funciones. Esta aprovecha las funciones de HTML5 en los navegadores modernos manteniendo la compatibilidad y funcionalidades para navegadores antiguos. Cuenta con cientos de widgets[2] de interfaz de usuario de alto rendimiento que están meticulosamente diseñados para adaptarse a las necesidades de los más simples, así como las aplicaciones web más complejas. El marco incluye un paquete de datos robusto que puede consumir datos desde cualquier fuente de datos (Sencha ExtJS, 2015).

Symfony 2.0.18

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

Apache Tomcat 7.0

Apache Tomcat es una herramienta código abierto[3] utilizada como servidor web. Se desarrolla en un entorno abierto y participativo, además es liberado bajo la licencia Apache versión 2. Apache Tomcat versión 7.0 implementa las especificaciones de la comunidad de procesos de Java sobre Servlet 3.0 y JavaServer Pages 2.2, e incluye muchas características adicionales que lo convierten en una plataforma útil para el desarrollo y despliegue de aplicaciones y servicios web (Apache Tomcat, 2015).

[2] Widgets: es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets.*
[3] Código abierto (en idioma inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

Apache 2.0

Apache es el servidor web más utilizado en sistemas Linux. Los servidores web son utilizados para servir páginas web solicitadas por equipos cliente. Los clientes normalmente solicitan y visualizan páginas web usando navegadores como Firefox, Opera, o Mozilla. Los servidores web Apache a menudo se usan en combinación con el motor de bases de datos MySQL, el lenguaje de scripting[4] PHP, y otros lenguajes de scripting populares como Python y Perl (Ubuntu, 2014).

PostgreSQL 9.4

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

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

PHP 5.0

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

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones

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

que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java (Eguiluz, 2008).

NetBeans 8.0

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

Visual Paradigm 8.0

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

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

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

UML 2.0

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

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

Patrón arquitectónico:

Para el desarrollo del diseñador gráfico de consultas se utilizará el patrón arquitectónico Modelo-Vista–Controlador que es el utilizado en el desarrollo de GDR v2.0.

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

- ✓ **Modelo:** representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- ✓ **Vista:** muestra el modelo en un formato adecuado para interactuar. Maneja la presentación visual de los datos representados por el modelo.
- ✓ **Controlador:** responde a eventos e invoca cambios en el modelo y probablemente en la vista.

Patrones GRASP

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

Experto: El patrón experto en información soluciona el problema ¿de qué forma podemos saber qué responsabilidad delegar a cada objeto? Es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. En la clase QueryEditor se evidencia la utilización de este patrón ya que esta clase delega toda la responsabilidad del trabajo con el código de las consultas a la clase SourceEditor la cual cuenta con la información y los métodos necesarios para cumplir esta responsabilidad.

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

QueryController que es la encargada de gestionar las principales acciones que se solicitan en la interfaz de la aplicación.

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

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

Patrones GOF

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

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

- ✓ **Patrones de creación:**

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

- ✓ **Patrones de estructura:**

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

- ✓ **Patrones de comportamiento:**

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

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

Visitante (Visitor): Permite aplicar una o más operaciones a un conjunto de objetos en tiempo de ejecución, representa una operación a realizar sobre los elementos de una estructura de objetos. Visitante permite definir una nueva operación sin cambiar las clases de los elementos en los que opera (SourceMaking, 2013). Este patrón se evidencia en la clase MDXParser la cual en tiempo de ejecución visita al código de la consulta y realiza cambios sobre este como por ejemplo en el requisito Ocultar campos vacíos en el resultado en el cual se le añade el operador NON EMPTY al código de la consulta.

Resultados

Interfaces de la aplicación

Interfaz de la vista de Diseño

La aplicación desarrollada consta con una interfaz para la vista de Diseño. En ella se encuentra un menú de Gestionar consultas con las acciones de Nueva, Renombrar, Exportar, Importar, Guardar, Abrir y Ejecutar. También cuenta con

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

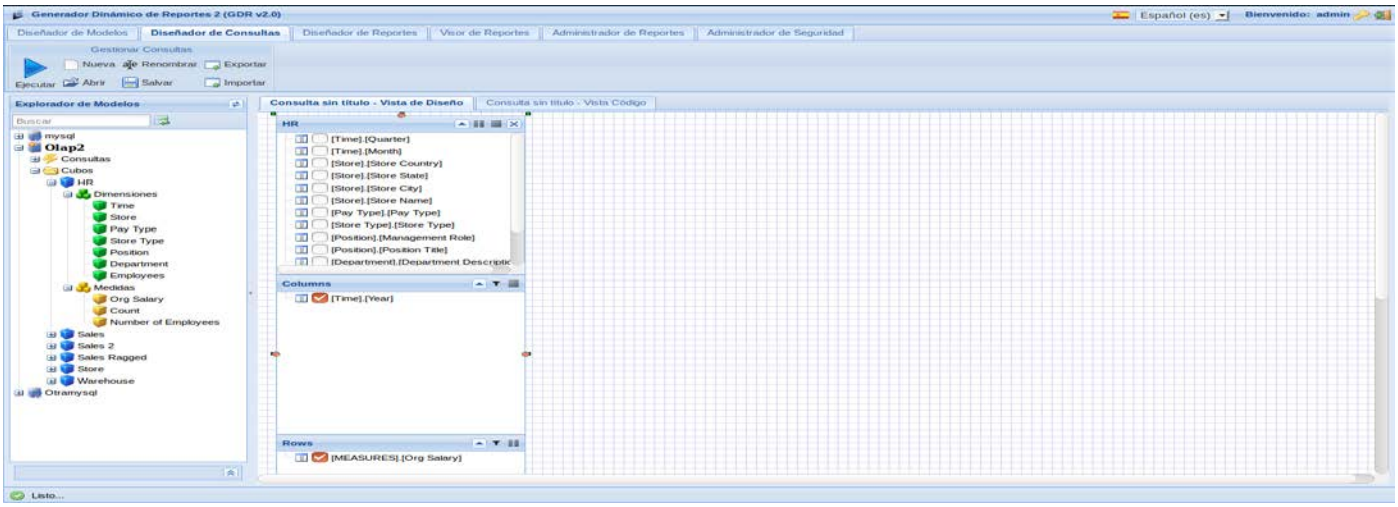

Fig. 1 Interfaz de la vista de Diseño.

Interfaz de la vista Código

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

almacenará en la carpeta seleccionada por el usuario, además de la opción “Actualizar” la vista previa de los datos. En la siguiente figura se muestra la Interfaz de la vista Código:

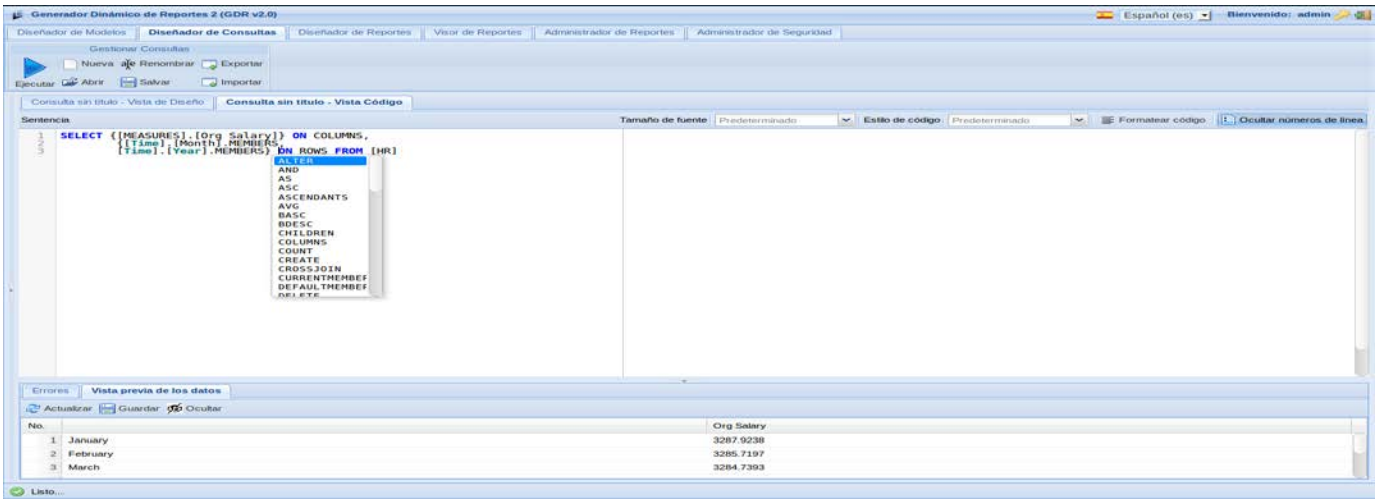

Fig. 1 Interfaz de la vista Código.

Conclusiones

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

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

Referencias

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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