Universidad de las Ciencias Informáticas
FACULTAD 6

Título: Módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor:

Julio César Brito Rodríguez

Tutor:

Ing. Aldis Joan Abreu Medina

Co-tutor:

Ing. Jorge Bedoya Rusenko

La Habana, Junio del 2012

"Año 54 de la Revolución"

"Lo que sabemos es una gota de agua; lo que ignoramos es el océano."

Isaac Newton

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Julio César Brito Rodríguez

_____
Firma del Autor

Ing Aldis Joan Abreu Medina

_____
Firma del Tutor

Ing Jorge Bedoya Rusenko

_____
Firma del Co-tutor

DATOS DE CONTACTO

Tutor:

Ing. Aldis Joan Abreu Medina

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: ajabreu@uci.cu

Co-tutor:

Ing. Jorge Bedoya Rusenko

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Email: jbedoya@uci.cu

AGRADECIMIENTOS

Muchas han sido las personas y los factores que a lo largo de estos años, han contribuido a que hoy me encuentre aquí. Este es el momento para agradecer a cada uno de ellos:

Primeramente quiero agradecer a mi mamá Gisela Rodríguez Escalona y a mi papá Leonardo Choy Peña, por su amor y cariño, por su apoyo en todas mis decisiones, por el esfuerzo y trabajo con el cual me han criado haciendo de mí lo que ahora soy.

A mi hermano Pablo Choy, a mi abuela Geo, por su infinito amor y por existir.

A mi mentor, guía y amigo Víctor Ramón por todos sus consejos y su ayuda incondicional desde mi llegada a la UCI hasta este día, gracias mi hermano.

A la UCI por las oportunidades ofrecidas.

A mi tutor Aldis por estar siempre pendiente de todos los detalles, guiándome en todo el proceso de desarrollo del trabajo. A mi cotutor Jorge Bedoya por su preocupación y apoyo en los momentos difíciles.

A todos mis compañeros del proyecto GDR por ayudarme y aclarar las dudas surgidas en el fragor de la batalla: Frank, Yasmany, Aurelio, Miguel, Raidel y Marleisy, su ayuda fue muy importante para la realización de este trabajo.

A mi tribunal y a mi oponente, por corregir cada detalle y estar dispuestos a ayudarme.

A todos mis compañeros que estuvieron conmigo en estos cinco años de carrera, en los buenos y en los malos momentos, en especial quiero mencionar a: Reynaldo, Yoandy, Alexander, Marcos, Ismel, Maylen, Yosmany, Yenei y muchos más, tantos, que no cabrían en otras 80 páginas.

Por último agradecer a todos los que de una forma u otra han contribuido a lo largo de estos cinco años a mi preparación profesional y como una mejor persona, a mis compañeros de aula, profesores y a todos con los que he compartido de una forma u otra, todos han puesto su granito de arena. De corazón muchísimas gracias. A todos y cada uno de ustedes muchísimas gracias…

DEDICATORIA

A mi mamá y a mi papá, por ser mi inspiración y el motivo de mí existir.

A mi hermano por estar siempre conmigo y espero ser un buen ejemplo para él.

A mi abuela Geo por su amor y por su fe.

A mi tío el "Rolo", siempre atento y afectuoso, que en paz descanse.

A toda mi familia por esperar siempre lo mejor de mí.

RESUMEN

La presente investigación surge en el departamento de Soluciones Integrales de DATEC, centro productivo cuyo principal objetivo es: desarrollar sistemas informáticos para satisfacer la necesidad de gestión de la información. Entre las tecnologías en desarrollo, se destaca la Plataforma de Ayuda para la Toma de Decisiones y Soluciones Integrales. El sistema Generador Dinámico de Reportes, incluido en dicha plataforma, permite obtener diferentes reportes con el objetivo de tomar decisiones y realizar estudios. Cuenta con un conjunto de módulos que brindan las funcionalidades para dar soporte al ciclo de vida de los reportes. El Diseñador de Modelos es uno de estos módulos, este permite: la gestión de orígenes de datos y el diseño los modelos semánticos que serán empleados en la confección de los reportes. Este estudio se desarrolla por la necesidad de mejorar el Diseñador de Modelos v1.8, con el propósito de actualizar las tecnologías empleadas en su creación y remodelar el diseño de la vista para optimizar la gestión de los componentes de los modelos semánticos. En el presente trabajo de diploma se realiza un estudio y se caracterizan las herramientas, tecnologías y metodología empleada en el desarrollo de la nueva versión, se diseñan e implementan las clases del módulo, aplicando buenas prácticas de los patrones de diseño y la reutilización de componentes. Se elaboran casos de pruebas para validar las funcionalidades del módulo implementado, garantizando la calidad que requiere el proceso de desarrollo de software del módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.

PALABRAS CLAVE

Diseñador de Modelos, Generador Dinámico de Reportes, modelos semánticos, orígenes de datos

TABLA DE CONTENIDOS

INTRODUCCIÓN

El desarrollo actual de las Tecnologías de la Información y las Comunicaciones (TIC) y la rapidez con que fluye la información a nivel mundial, ha propiciado que esta última sea considerada como uno de los principales activos de cualquier organización. Su valor, depende en gran medida, de la forma en que ayuda a los individuos de la organización para que tomen las decisiones que los conduzcan a lograr los objetivos y metas propuestas.

Un sistema de información es un conjunto de elementos interrelacionados con el propósito de prestar atención a las demandas de información de una organización. Desde esa perspectiva, eleva el nivel de conocimientos, lo que permite un mejor apoyo a la toma de decisiones y al desarrollo de acciones (1). Este desempeña un papel primordial en la vida de las empresas, mejorando procesos, reduciendo el tiempo de desarrollo y ayudando a centrarse en tareas que agreguen valor a la entidad donde se aplican.

En la actualidad las empresas manejan grandes volúmenes de información. Estas necesitan tener almacenados todos los datos concernientes a sus negocios en bases de datos, para gestionarlos mediante una aplicación profesional. Sin esta funcionalidad, resultaría imposible manejar en su totalidad la información que se genera en la empresa, ocasionando pérdidas de tiempo y dinero. El análisis de los datos existentes en una entidad no se basa únicamente en la recopilación y acumulación de información; sino que se necesita saber interpretarlos y emplearlos de forma adecuada.

La mayoría de las organizaciones utilizan reportes para visualizar análisis y resultados. De esta manera, los directivos de las empresas pueden seguir la marcha del negocio a partir de los reportes de información, facilitando la identificación de nuevas oportunidades de negocio o servicios. Como consecuencia, los reportes son considerados una necesidad principal en la Inteligencia de Negocio (2). Estos organizan y exhiben la información contenida en las bases de datos**,** aplicando un formato determinado a los mismos. Para luego mostrarlos por medio de un diseño atractivo y fácil de interpretar por los usuarios. De esta forma, le confiere una mayor utilidad a la información de las organizaciones.

Los generadores de reportes son herramientas complementarias a los sistemas de información. Utilizan una especie de lenguaje transparente para el usuario, por medio del cual este realiza consultas a la base de datos y obtiene información de ella en forma de reporte (3).

Cuba se encuentra en medio del proceso competitivo empresarial actual. En este sentido ha puesto empeño en la informatización de sus entidades y en la preparación de personal capacitado en la rama de la informática. En medio de esta revolución digital surge la Universidad de las Ciencias Informáticas (UCI). Con la acertada visión del compañero Fidel Castro Ruz y la misión de convertirse en una casa

de altos estudios y una entidad capaz de producir software de calidad.

El Centro de Tecnologías de Gestión de Datos (DATEC), es uno de los centros de desarrollo de software de la UCI. Este anuncia el inicio de un conjunto de proyectos que tributan al desarrollo de herramientas y servicios informáticos, especializados en el almacenamiento y análisis de datos. El departamento de Soluciones Integrales de DATEC, se encarga del desarrollo de sistemas informáticos con el fin de satisfacer las necesidades de gestión de la información en cualquier entidad que maneje un proceso de negocio. Entre las tecnologías en construcción se destaca la Plataforma de Ayuda para la Toma de Decisiones y Soluciones Integrales (PATDSI).

El sistema Generador Dinámico de Reportes (GDR) incluido en dicha plataforma es uno de sus módulos de inteligencia de negocio. El mismo permite obtener diferentes reportes con el objetivo de tomar decisiones y realizar estudios. Está conformado por un conjunto de módulos que brindan las funcionalidades para dar soporte a todo el ciclo de vida de los reportes.

El Diseñador de Modelos (DM) es uno de estos módulos, este permite gestionar orígenes de datos en varios gestores de base de datos y diseñar los modelos semánticos que serán utilizados en la confección de los reportes. En el ámbito del GDR, un modelo semántico es: "…una abstracción de la base de datos que contiene las entidades (tablas, vistas y rutinas) de la misma. Almacena en forma de fichero XML toda la información de los metadatos de los objetos de la base de datos…" (4).

La versión 1.8 del DM resuelve correctamente los modelos semánticos, pero han sido identificadas serias deficiencias que se deben resolver para mejorar su funcionamiento. Entre ellas figuran que las tecnologías utilizadas para su creación, muchas han caducado, y otras necesitan reemplazo. Existen problemas arquitectónicos en su diseño de clases, incorrecta separación de responsabilidades, indebida aplicación de patrones de diseño, desorganización de código y poca documentación del mismo. Estos problemas influyen en el rendimiento de la aplicación, imposibilitando la integración de nuevas funcionalidades por la dificultad de interpretación y manipulación de código. También requiere que la capa de presentación de datos de dicho módulo mejore su apariencia externa, para facilitar el diseño de los modelos semánticos al usuario.

De la situación anteriormente planteada surge el siguiente **problema de la investigación:** ¿Cómo diseñar los modelos semánticos para el Generador Dinámico de Reportes v2.0?

Se define como **objeto de estudio:** El proceso de desarrollo de software en los sistemas generadores de reportes.

Enmarcado en el **campo de acción**: Diseñador de Modelos para el Generador Dinámico de Reportes.

Para solucionar el problema planteado se define como **objetivo general:** Desarrollar el módulo Diseñador de Modelos para el sistema Generador Dinámico de Reportes v2.0.

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

- Diseñar el módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.
- Implementar las clases del módulo diseñado.
- Validar el correcto funcionamiento del módulo implementado.

Para dar cumplimiento a los objetivos se realizaron esencialmente las siguientes **tareas de la investigación**:

- Diseño del modelo de clases que realizarán las peticiones del cliente.
- Diseño de las clases que conformarán las respuestas del servidor.
- Implementación del diseño de clases de la capa de presentación de datos.
- Implementación del diseño para satisfacer los requerimientos funcionales del módulo.
- Diseño de los casos de pruebas para determinar el correcto funcionamiento de los requisitos.
- Realización de pruebas al módulo para validar su calidad.

El documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, anexos y glosario de términos.

CAPÍTULO I: FUNDAMENTOS TEÓRICOS

En este capítulo se hace un estudio bibliográfico de los sistemas existentes para la generación de reportes. Se fundamenta el uso de las distintas herramientas, tecnologías y metodología empleada con el objetivo de explotar al máximo sus potencialidades. Además, se definen los componentes implicados en la etapa de diseño de un modelo semántico.

CAPÍTULO II: DISEÑO DEL MÓDULO

En este capítulo se realiza el modelo de clases del diseño del módulo, aplicando buenas prácticas de patrones de diseño y la reutilización de componentes. Se realiza el diagrama de despliegue teniendo en cuenta la configuración y distribución de los nodos de cómputo.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO

Se hace referencia a la implementación de la solución, abordando como se realizan los procesos de gestión de orígenes de datos, diseño y administración de los modelos semánticos. De igual manera, se explica el proceso de validación del sistema a través los casos de pruebas realizados al módulo.

1

CAPÍTULO I: FUNDAMENTOS TEÓRICOS

Introducción

En este capítulo se realiza un estudio bibliográfico de los sistemas existentes para la generación de reportes. Se fundamenta el uso de las distintas herramientas, tecnologías y metodología empleada, con el objetivo de explotar al máximo sus potencialidades. Se definen los componentes implicados en la etapa de diseño de un modelo semántico en el Diseñador de Modelos para el GDR v2.0.

1.1 Sistemas generadores de reportes

Los sistemas generadores de reportes complementan a los sistemas de información, estos permiten obtener la información en forma de reporte. De ahí, que ofrezca a los trabajadores calificados un mayor nivel de detalle y flexibilidad. Además, tienen la capacidad de interactuar con los resultados obtenidos basándose en los datos para tomar sus propias decisiones (5). La generación de reportes es fundamental para reflejar el comportamiento de los diferentes componentes que integran cualquier organización. Es importante para el reconocimiento de las diversas problemáticas que existen en las entidades, y en esa medida, apoyan a los individuos en la correcta toma de decisiones para resolver dichas problemáticas.

1.2 Herramientas de código abierto para la generación de reportes.

En la actualidad existen muchos sistemas que facilitan el proceso de generación de reportes, pero la mayoría de ellos son de carácter privativo. El software libre, trabaja de forma rigurosa para desarrollar tecnologías de calidad que se encuentren al alcance de todos. Entre las principales herramientas de reportes de código abierto que más se utilizan en el mercado se encuentran: Pentaho Reporting, Eclipse Birt y Jasper Reports.

1.2.1 Pentaho Reporting

Es una solución basada en el proyecto JFreeReports[1] que permite generar informes de manera ágil. Adquirida e integrada en la suite de código abierto para la inteligencia de negocio de Pentaho (BIOSP por sus siglas en inglés). Pentaho Reporting es la herramienta proporcionada por Pentaho e integrada en su suite para el desarrollo de informes (6).

[1] **JFreeReports:** es una librería de código abierto para la generación de reportes. Está desarrollada en el lenguaje Java.

Permite la distribución de los resultados de los análisis en múltiples formatos y provee el acceso a fuentes de datos relacionales, procesamientos analíticos en línea (OLAP por sus siglas en inglés) o basadas en XML. Además, todos los informes incluyen la opción de imprimir o exportar a formato PDF, XLS, HTML y texto plano. Los reportes de Pentaho permiten la programación de tareas y la ejecución automática de informes con una determinada periodicidad.

Características

- Posee un asistente integrado que guía a los diseñadores de reportes durante el proceso de diseño.
- Dispone de un diseñador gráfico que provee el completo control de acceso a los datos, agrupaciones, círculos, gráficas y los formatos para reportes de alta resolución.
- Contiene plantillas de reportes que aceleran el proceso de generación, proporcionando un aspecto consistente y atractivo (6).

1.2.2 Eclipse Birt

Eclipse BIRT acrónimo de (*Business Intelligence and Reporting Tools*- herramienta para el reporte y la inteligencia de negocio*)* es el software líder mundial en código abierto para la elaboración de cuadros de mando en empresas e instituciones. Es un sistema utilizado para desarrollar aplicaciones web, especialmente las basadas en Java y Java EE. Está conformado por dos componentes principales: un diseñador de informes basado en Eclipse y un componente en tiempo de ejecución que se puede agregar a un servidor de aplicaciones. Dispone de un motor de gráficos que le permite agregar los gráficos que posteriormente serán empleados en los diseños de los reportes. Basada en el entorno de desarrollo Eclipse con capacidad de ser embebida en proyectos J2EE de manera independiente.

1.2.3 Jasper Reports

Herramienta para dar respuesta a las necesidades de generación de reportes. Forma parte de la suite propia conjuntamente con Jasper Server. Presenta el editor de informes IReports, escrito en lenguaje Java, que permite agilizar la creación de plantillas para reportes. Jasper Reports facilita y agiliza la generación, previsualización e impresión de los reportes. Es un poderoso sistema informático para generar informes con las potencialidades de producir contenido completo para la pantalla, directo para la impresora o en diferentes formatos de archivos como PDF, HTML, XLS, CSV y XML.

Características que pueden ser embebidas

- Scriptlets, que pueden acompañar a la definición del informe, y pueden ser invocados en cualquier momento, como un procesamiento adicional. Los scriptlets son fragmentos de código Java embebido en HTML, que se pueden invocar en el proceso de generación de informes.

- Sub-informes, que constituyen informes que se insertan en otros informes (6).

1.2.4 Comparación entre las herramientas analizadas

Existen diferentes aspectos a comparar en estas herramientas de reportes para determinar sus potencialidades. De esta manera, se pueden tener en cuenta para realizar una comparación exhaustiva los siguientes elementos: la conectividad y las propiedades de las fuentes de datos, así como los formatos de salida de cada una de estas soluciones. Algunas de estas características se recopilan en las siguientes tablas (ver tablas 1, 2 y 3).

Herramientas de reportes	**Birt**	**Pentaho Reporting**	**Jasper Reports**
Fuentes de datos			
JDBC	**Si**	**Si**	**Si**
XML	**Si**	**Si**	**Si**
Servicio web	**Si**	**No**	**No**
Hibernate	**No**	**No**	**Si**
EJB	**No**	**No**	**Si**

Tabla 1. Comparación de las fuentes de datos

Propiedades			
Múltiples origen de datos	**Si**	**No**	**Si**
Orígenes de datos combinados	**Si**	**No**	**No**
Transformaciones de datos	**Si**	**Si**	**Si**

Tabla 2. Comparación de las propiedades

Formato de salida			
PDF	**Si**	**Si**	**Si**
HMTL	**Si**	**Si**	**Si**
EXCEL	**Si**	**Si**	**Si**
ODT	**No**	**No**	**No**
DOC	**No**	**No**	**No**

Tabla 3. Comparación de los formatos de salida

Las herramientas analizadas resuelven correctamente el ciclo de vida de generación de los reportes. De igual manera, aportan información valiosa para la nueva concepción del módulo, desde la perspectiva en que: gestionan varios orígenes de datos, modifican los mismos y marcan las pautas para incursionar en la ambiciosa temática de los orígenes de datos combinados. A pesar de estas ventajas existen inconvenientes que obligan a utilizar otro tipo de solución.

Desventajas

- Se necesitaría capacitar al equipo de proyecto, aumentando el tiempo de desarrollo de la solución informática.
- Los productos mencionados son aplicaciones de escritorio que no se integran en la plataforma web.

1.3 Generadores de reportes en la UCI

En la UCI, como parte del proceso productivo, para brindar soporte y solución a problemas existentes, se han desarrollado varias herramientas para la generación de reportes. Estas persiguen alcanzar la eficiente interacción de los usuarios de los proyectos con la información almacenada en sus bases de datos. Se trabaja en elevar la eficacia de los sistemas de generación de reportes (7). Ejemplos de estas herramientas son: Akademos y el GDR v1.8.

1.3.1 Akademos

El sistema de gestión de pregrado Akademos es una aplicación desarrollada en plataforma .NET a partir del año 2006, en la Universidad de las Ciencias Informáticas. Este sistema brinda actualmente importantes servicios de gestión académica tanto a los estudiantes como a los departamentos docentes. Cuenta con un módulo para el diseño de reportes personalizados. Utiliza Active Reports[2] para la confección y generación de los reportes con eficiencia. Desde el punto de vista del dinamismo, se encuentra adaptado solamente al negocio de la gestión académica en la UCI. Lo cual imposibilita que esta herramienta sea de propósito general.

1.3.2 Generador Dinámico de Reportes v1.8

Con el objetivo de desarrollar un Sistema de Gestión de Reportes Dinámicos (SGRD), que permitiese cubrir las necesidades de reportes de cualquier empresa o institución, se concibe el sistema GDR. Herramienta que brinda la posibilidad de controlar el funcionamiento periódico de una o varias entidades, mediante la formulación de reportes en diferentes formatos y modelos personalizados. Proporciona a los usuarios la ventaja de agilizar el proceso de la toma de decisiones.

[2] **Active Reports:** es un componente de informes .NET para aplicaciones Windows Forms y formularios web.

Características

- Aplicación desarrollada sobre el marco de trabajo Symfony.
- Escrita en el lenguaje de desarrollo PHP.
- Multiplataforma.
- Soporta imágenes y gráficas.
- Soporta varios orígenes de datos.

La última versión estable de este sistema es GDR v1.8. La misma cubre el ciclo básico de la generación de reportes y soluciona el problema de obtener los diferentes informes en los sistemas de gestión de la información que se desarrollan en cualquier entorno empresarial, incluyendo la UCI. Es una herramienta web que permite a sus clientes consultar los gestores de bases de datos de sus organizaciones y generar reportes con la información que se manejan en sus negocios.
A manera de resumen se puede argumentar que el GDR v1.8 es capaz de realizar estudios de investigación o consultas de información de cualquier negocio con el objetivo de tomar decisiones. Puede ser utilizado tanto en el ámbito de la universidad como en cualquier otra entidad que maneje un proceso de negocio. Aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, el usuario aún debe poseer conocimientos básicos para emplear esta herramienta. Su entorno de trabajo está estructurado de forma que es difícil guiarse para la creación y generación de reportes. Necesita una transición hacia una versión superior por la imperiosa necesidad de actualización de las tecnologías empleadas en su desarrollo. Requiere el uso de patrones de diseño que optimicen su código.

1.4 Diseñador de Modelos

Los modelos de datos son herramientas conceptuales que describen la información de las bases de datos, las relaciones entre los datos de la misma, su semántica y sus limitantes. Representan esquemas de bases de datos mediante entidades y asociaciones. Describen la información de forma sencilla y global. El nivel de diseño de un modelo de datos es aquel que se le presenta al usuario final y que puede tener combinaciones o relaciones entre los objetos que conforman la base de datos global. Puede definirse como la forma en la que el usuario aprecia la información y sus relaciones.

1.4.1 Diseñador de Modelos del GDR v1.8

Es el módulo encargado de conectarse a un origen de datos, utilizando gestores de base de datos PostgreSQL, Oracle, SQLite, MySQL o SQL Server para luego diseñar los modelos semánticos que serán utilizados en la confección de los reportes. El DM permite visualizar, adicionar, modificar y

eliminar los orígenes de datos. Además, muestra los modelos previamente creados y los elementos que los componen (tablas, vistas y rutinas) (8).

1.4.2 Origen de datos

Es la unidad que contiene la información necesaria para obtener acceso a las fuentes de datos. Este término se emplea para referirse al objeto utilizado para establecer las conexiones con las bases de datos. Un origen de datos especifica un proveedor de datos y la configuración del resto de las propiedades de la cadena de conexión utilizada para obtener el acceso a su información.

1.4.3 Origen de datos en el Diseñador de Modelos para GDR v1.8

Gestiona las posibles conexiones con varios gestores de base de datos. Actualmente se brinda soporte a los siguientes gestores: Microsoft SQL Server (mssql), MySQL Server (mysql), Oracle Server (oracle), PostgreSQL (postgresql) y SQLite (sqlite). Los parámetros requeridos para crear un nuevo origen de datos son:

Parámetros del origen de datos (4)

- Nombre con que se registra.
- Tipo de gestor.
- La dirección IP del servidor de bases de datos.
- Puerto que usa el servidor.
- Usuario para conectarse al servidor.
- Clave del usuario.
- Base de datos de la cual se extraerán la información.

1.4.4 Modelo semántico

Forma de representar una base de datos de manera conceptual. Permiten captar mejor el significado o la semántica de la información contenida en la base de datos. Un objeto semántico es una representación de algunos elementos identificables en el ambiente de trabajo de los usuarios. Es un conjunto de atributos que describen con eficacia una identidad bien determinada.

1.4.5 Modelo semántico en el Diseñador de Modelos del GDR v1.8

Contiene las entidades con la información más relevante de las bases de datos. El DM v1.8 permite registrar el nombre con el que deberá ser identificado el modelo que persistirá en la base de datos del sistema para ser gestionado posteriormente. Asimismo posibilita la búsqueda de los modelos semánticos, para lo cual se debe escribir el criterio de búsqueda en un campo de texto, para que se

filtren aquellos modelos que cumplan con los criterios de búsqueda introducidos por el usuario. También brinda la posibilidad de renombrar y modificar la información de los modelos.

Se puede concluir que tras realizar un estudio a cada uno de los componentes que intervienen en el proceso de diseño de los modelos de semánticos se evidenció que el DM v1.8 los gestiona de forma correcta, pero requiere:

- Un proceso de mejora y optimización en el que se alcance un menor grado de interacción entre usuario, sistema y recursos de las bases de datos involucradas.
- Cambios sobre la arquitectura para que pueda ser utilizada por otros sistemas, logrando la modularidad, escalabilidad e independencia.
- Alcanzar mayor rendimiento y usabilidad en la gestión sus componentes.

1.5 Herramientas y tecnologías

En el mundo del desarrollo de aplicaciones web se utilizan diferentes tecnologías para la construcción de este tipo de aplicación informática. Las herramientas que se utilizarán para el desarrollo del módulo fueron previamente definidas por el equipo de arquitectura del GDR v2.0. Con el objetivo de definir ventajas, oportunidades y con la disposición de potenciar al máximo las prestaciones de las mismas se realiza el siguiente estudio.

1.5.1 Lenguajes de programación

Un lenguaje de programación describe un conjunto de acciones consecutivas que se deben ejecutar y que permiten crear programas a través de operadores y reglas de sintaxis. Es un modo práctico para dar instrucciones a un equipo de cómputo, que le permite al desarrollador comunicarse con los dispositivos de hardware y software existentes.

PHP

PHP, acrónimo de (*Hypertext Pre Processor)* constituye un lenguaje de programación interpretado, de propósito general y ampliamente difundido en el mundo. Diseñado originalmente para la creación de páginas web dinámicas, principalmente en interpretación del lado del servidor aunque puede ser incrustado dentro de código HTML (9).

Diseñado especialmente para incrementar el dinamismo de las páginas web, aprovechando los recursos de las redes informáticas y los escasos requerimientos de hardware que solicita el lenguaje para que sus aplicaciones funcionen correctamente. Además, permite aplicar técnicas de programación orientada a objetos. Es un lenguaje completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. No requiere

definición de tipos de variables aunque se pueden evaluar por el tipo que estén manejando en tiempo de ejecución. El código fuente escrito en este lenguaje es invisible al navegador ya que es el servidor el encargado de ejecutar el código y enviar su resultado HTML al cliente, logrando una programación segura y confiable. Posee una amplia documentación, dado que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

XHTML

XHTML, acrónimo de (*Extensible Hypertext Markup Language)* es un lenguaje de marcado pensado para sustituir al HTML como estándar para las páginas web. Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. Al utilizar este lenguaje, el navegador no necesita implementar heurísticas para detectar qué quiso poner el autor, por lo que el parser puede ser mucho más sencillo. Como integra las potencialidades de XML se pueden manejar fácilmente herramientas creadas para procesamiento de documentos XML genéricos.

JS

JS, acrónimo de (*JavaScript)***,** establece un lenguaje basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de internet. Lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Los programas JavaScript van incrustados en los documentos XHTML o en un fichero JS, estos se encargan de realizar acciones en el cliente, como puede ser: pedir datos, confirmar, mostrar mensajes, crear animaciones o comprobar campos (10).

Los programas escritos en este lenguaje no requieren de mucha memoria ni tiempo adicional de transmisión, por ser pequeños y compactos. No requieren un tiempo de compilación por ser un lenguaje interpretado. Además es independiente de la plataforma de hardware o sistema operativo, y funcionan correctamente siempre y cuando exista un navegador que lo soporte. También, asegura la permanencia de una operación realizada, y aunque falle el sistema esta no podrá deshacerse.

CSS

CSS, acrónimo de (*Cascading Style Sheets)* es lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con los lenguajes HTML o XHTML. Entre las características de este lenguaje se encuentra la separación de los contenidos de su presentación, siendo esto imprescindible para crear páginas web complejas (11).

CSS tiene una sintaxis sencilla y fácil de utilizar. Para su empleo se requiere el uso de una serie de reglas que consisten en uno o varios selectores y un bloque de estilos para los elementos del

documento. En ese sentido, cada bloque de estilos se define entre llaves, y está formado por una o varias declaraciones. De forma general; CSS le facilita el trabajo al diseñador, usuario o dispositivo electrónico que muestre la página, ya que propicia la modificación de la visualización del documento sin alterar el contenido del mismo, solo configurando algunos de sus parámetros.

JSON

JSON, acrónimo de (*JavaScript Object Notation*) genera un formato ligero para el intercambio de datos. Subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Se emplea habitualmente en entornos donde el flujo de información entre cliente y servidor es de vital importancia. Además es fácil de leer y escribir para los seres humanos, mientras que para las máquinas es simple de interpretar y generarlo.

Su simplicidad ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Desde esa perspectiva una de sus ventajas sobre XML como formato de intercambio de datos está en, que es mucho más sencillo a la hora de escribir un analizador semántico del mismo (12).

XML

XML, acrónimo de (*Extensible Markup Language)* es un lenguaje conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Este permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades. Contiene tres características muy importantes: extensibilidad, estructura y validación (13).

Con su empleo las aplicaciones se pueden generar rápidamente y su mantenimiento es sencillo. Separa los datos de la presentación y del proceso, lo que permite mostrar y procesar los datos al gusto deseado con solo aplicar distintas hojas de estilo y aplicaciones. La información es más accesible y reutilizable, por la flexibilidad de sus etiquetas que permiten su utilización sin tener que amoldarse a reglas específicas de un fabricante. Asimismo, ofrece un formato para la descripción de datos estructurados, facilitando declaraciones de contenido más precisas y resultados de búsquedas más significativos en varias plataformas.

AJAX

AJAX, acrónimo de (*Asynchronous JavaScript And XML)* es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Esta se ejecuta en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas. Por lo que implica un aumento de la interactividad, velocidad y usabilidad en las aplicaciones. AJAX es una tecnología

asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de AJAX mientras que el acceso a los datos se realiza mediante XMLHttpRequest[3], objeto disponible en los navegadores actuales. Es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores. Puesto que, está basado en estándares abiertos como JavaScript y en el Modelo de Objetos del Documento (*DOM*).

En resumen, AJAX combina cuatro tecnologías ya existentes. Las peticiones del lado del cliente se realizarán utilizando este tipo de tecnología. Se utilizará JSON como formato para la transferencia de datos solicitados al servidor. Además se empleará XHTML y las hojas de estilos en cascada (CSS) para el diseño que acompaña a la información. Asimismo, el acceso al DOM con el lenguaje JavaScript permitirá mostrar e interactuar dinámicamente con la información presentada.

1.5.2 Entorno de desarrollo

IDE, acrónimo de (*Integrated Development Environment)*. Constituyen aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos sistemas informáticos han sido empaquetados como programas de aplicaciones y generalmente están compuestos por un conjunto de herramientas de programación: un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Pueden manejar uno o varios lenguajes de programación.

NetBeans 7.0

NetBeans 7.0 es un entorno de desarrollo integrado distribuido por SUN Microsystems bajo licencia dual: la Licencia Pública General (*GPL*) y la Licencia Común de Desarrollo y Distribución (*CDDL*). Esto significa que es gratuito y de código abierto para desarrolladores de software. Permite a los programadores escribir, compilar, depurar y ejecutar programas. Es apoyado por una gran comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación (14).

Además, introduce soporte de idiomas para el desarrollo de la especificación Java Standard Edition (*Java SE*) 7 con las características del lenguaje Java Development Kit (*JDK*) 7. Ofrece integración mejorada con el servidor Oracle WebLogic, así como soporte para Oracle Database y GlassFish 3.1. Incluye un diseñador de GridBagLayout[4] para mejorar el desarrollo de interfaz gráfica de usuario.

[3] **XMLHttpRequest:** es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores web.
[4] **GridBagLayout:** es uno de los administradores de diseño más flexible que proporciona la plataforma Java.

1.5.3 Gestor de base de datos

Un gestor de base de datos es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, asegurando su integridad, confidencialidad y seguridad. De esta manera, manipulan de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización. Proveen facilidades para la manipulación de grandes volúmenes de datos. Disminuyen drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.

PostgreSQL 9.0

Tiene como propósito general manejar de forma clara, sencilla y ordenada el conjunto de datos que se convierte en información relevante para una organización. Proporciona una notable mejora en el tiempo de ejecución y hace más sencillo el análisis de datos avanzados. Esta nueva versión cuenta con un enfoque que se centra en la administración y vigilancia. La instalación es ilimitada y proporciona mejor soporte que los proveedores comerciales. Además favorece un ahorro considerable en costos de operación. Es multiplataforma y está diseñado para ambientes de alto volumen.

PgAdmin III

Es una herramienta para la administración del gestor de base de datos PostgreSQL. Esta permite crear simples consultas SQL o consultas de una alta complejidad de una manera más fácil, debido a que se le ha incorporado una nueva interfaz gráfica, la cual resulta de gran ayuda en el momento de gestionar las bases de datos. Es una herramienta de código abierto respaldada por una amplia comunidad de desarrolladores que encaminan sus esfuerzos al perfeccionamiento de la misma.

Esta aplicación incluye un editor de resaltado de sintaxis SQL, una interfaz administrativa gráfica y una herramienta de consulta SQL. La conexión al servidor se puede hacer a través de protocolo TCP/IP. No se requieren controladores adicionales para comunicarse con el servidor de base de datos (15).

1.5.4 Lenguaje de Modelado

El modelado de sistemas software es una técnica que ayuda al ingeniero de software a visualizar el sistema a construir. De ahí que los modelos pueden utilizarse para la comunicación con el cliente, grupo de desarrollo y otros factores que interviene en el proceso de desarrollo.

UML

Lenguaje Unificado de Modelado (UML) por sus siglas en inglés, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar,

especificar, construir y documentar un sistema. Además, se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software. Ha sido aprobado por el Grupo de Gestión de Objetos (OMG) por sus siglas en inglés, como la notación estándar para el desarrollo de proyectos informáticos. Igualmente es útil para el desarrollo de modelado visual de cualquier proyecto no sólo informático, conjuntamente promueve la reutilización.

1.5.5 Herramienta de modelado

CASE, acrónimo de (*Computer Aided Software Engineering)* es el tipo de herramienta destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos y hasta la implementación de parte del código automáticamente a partir de un diseño.

Visual Paradigm

Es una herramienta de ingeniería de software asistida por computación que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, generación del código fuente de los programas y la documentación de los mismos. Su propósito general es soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas.

Está dispone en múltiples plataformas (Windows, Linux). Genera diseños centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. Usa un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Mantiene capacidades de ingeniería directa e inversa. Posibilita que el modelo y el código permanezcan sincronizados en todo el ciclo de desarrollo. Soporta aplicaciones web (17).

1.5.6 Marcos de trabajo

En el desarrollo de software, un marco de trabajo, es una estructura conceptual y tecnológica de soporte definido con módulos de software concretos, sirviendo de base para que otro proyecto de software pueda ser fácilmente organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y otras herramientas, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Symfony 1.4.8

Esta solución informática está diseñada para optimizar el desarrollo de las aplicaciones web. Desarrollada completamente con PHP 5, sencilla de usar pero lo suficientemente flexible como para

adaptarse a los casos más complejos. Fácil de extender, lo que permite su integración con las librerías de otros fabricantes. Separa 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. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en sistemas Windows y Unix estándares. La versión 1.4 fue liberada al mercado en noviembre de 2009. Esta posee similares características a las de Symfony 1.2, pero todas las características obsoletas, incluyendo la capa de compatibilidad completa, han sido eliminadas.

Ventajas

- Independiente del sistema gestor de bases de datos.
- Acceso a datos a través del mapeo objeto-relacional (ORM).
- Incluye soporte para AJAX.
- Soporta la instalación de extensiones para añadir nuevas funcionalidades.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Multiplataforma.

Ext-Js 3.3

Es una librería de JavaScript para el desarrollo de aplicaciones web enriquecidas, haciendo un uso intensivo de las tecnologías AJAX, XHTML, DHTML y DOM. Originalmente fue creado como una extensión de Yahoo User Interface (YUI). Incluye interoperabilidad con jQuery, Prototype y Scriptaculo.US. La versión 3.0 fue liberada el 3 de junio de 2009 con grandes mejoras.

Mejoras (18)

- Nuevos ejemplos y componentes (incluye un componente para gráficas).
- Administración de memoria mejorada para el navegador Internet Explorer 6.
- Interfaz de programación de aplicaciones (*API*[5]) documentada y CSS refactorizado.

Después de analizar las principales características y ventajas de los marcos de trabajos que se utilizarán en el desarrollo de la aplicación. Se concluye que, el uso de los mismos, permitirá definir una arquitectura adaptada a las particularidades del módulo; estableciendo reglas y mecanismos de interacción entre los componentes del DM. Igualmente posibilitará completar la arquitectura con nuevos componentes dentro de las relaciones estructurales que bridan los propios marcos de trabajo,

[5] **API:** es el conjunto de métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

al quedar en responsabilidad del desarrollador solo aquellos aspectos relevantes en la implementación. El uso de Symfony simplificará el desarrollo de la aplicación, permitiendo la automatización de las tareas, proporcionando estructura al código fuente, creando código más legible y fácil de mantener. El empleo de la tecnología Ext-Js permitirá incluir componentes predefinidos para la interfaz visual del módulo debido a la reutilización de su código fuente.

1.5.7 Servidor de aplicaciones

Se denomina servidor de aplicaciones a una computadora que provee servicios a otras computadoras. Un servidor de aplicaciones gestiona la mayor parte o la totalidad de las funciones de lógica de negocio y de acceso a los datos de una aplicación. Los principales beneficios de las tecnologías de servidores de aplicación son la centralización y la disminución de la complejidad en el desarrollo de aplicaciones.

Apache 2.2

Apache 2.2 constituye una tecnología gratuita de código abierto. Es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Permite la creación de sitios web dinámicos mediante el uso de Server Side Includes (SSI) por sus siglas en inglés, de lenguajes de scripting como PHP, JavaScript, Python, entre otros. Se ejecuta en varios sistemas operativos. Posee una arquitectura modular que le permite ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos. Tiene una alta configurabilidad en la creación y gestión de logs. Es eficiente y flexible; permitiendo personalizar la respuesta ante los posibles errores que se puedan generar en el servidor.

1.6 Patrones de software

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

1.6.1 Patrones de arquitectura

Los patrones arquitectónicos ofrecen soluciones a problemas de arquitectura en la ingeniería de software. Estos expresan un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. Asimismo, describen los

elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre su empleo. Las arquitecturas más utilizadas para el desarrollo de software son: Monolítica, Cliente-Servidor y Tres Capas. Esta última constituye una especialización de la arquitectura Cliente-Servidor, donde la carga se divide en tres capas, con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (modelado el negocio) y una tercera para el almacenamiento (persistencia); una capa solamente tiene relación con la siguiente.

MVC

MVC, acrónimo de (*Model View Controller)* es un patrón de arquitectura pertenece a la familia de los estilos arquitectónicos de Llamada y Retorno. El mismo separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos: Modelo, Vista y Controlador. Por ende, el modelo encapsula los datos y la funcionalidad de la aplicación. La vista despliega la información contenida en el modelo. El controlador está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del modelo o de vista. De ahí que su empleo garantiza una mayor claridad en el diseño y facilita una mayor escalabilidad. Asimismo, fomenta la reutilización de los componentes. Simplifica el mantenimiento de los sistemas. Igualmente, separa la interfaz, lógica de negocio y de presentación.

1.6.2 Patrones de diseño

Los patrones de diseño constituyen el esqueleto de las soluciones a problemas comunes en el desarrollo de software. Estos brindan una solución probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño son clasificados según su finalidad en: creacionales, estructurales y de comportamiento, mientras que respecto a su ámbito se especifican en clases y objetos (19).

Clasificación

- **Creacionales**: resuelven problemas relativos a la creación de objetos.

Patrón de Fábrica Abstracta (*Abstract Factory*), Patrón Constructor (*Builder*), Patrón de Instancia Única (*Singleton*), entre otros.

- **Estructurales**: solucionan problemas relativos a la composición de objetos.

Patrón Adaptador (*Adapter*), Patrón Decorador (*Decorator*), Patrón de Fachada (*Facade*), entre otros.

- **De Comportamiento**: resuelven problemas relativos a la interacción entre objetos.

Patrón de Comando (*Command*), Patrón Mediador (Mediator), Patrón Observador (*Observer*).

1.6.3 Patrones GRASP

GRASP, acrónimo de (*General Responsibility Assignment Software Patterns)* son una serie de buenas prácticas de aplicación recomendable en el diseño de software. Los mismos describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades (20). A continuación se describen brevemente los principales patrones de esta clasificación.

Experto

Indica que la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo.

Creador

Identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases.

Alta cohesión

Plantea que la información que almacena una clase debe ser coherente y estar en la mayor medida relacionada con la clase.

Bajo acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. 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 clases.

Controlador

Asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas, facilitando la centralización de las actividades.

Después de haber realizado la investigación de los patrones de diseños y de asignación de responsabilidades anteriormente expuestos, se llega a la siguiente conclusión. En la solución del sistema se deben aplicar los patrones Controlador Frontal e Instancia Única para garantizar la seguridad y consistencia del módulo. Así como el patrón Orden para encapsular las operaciones y permitir su ejecución sin necesidad de conocer el contenido de la misma. Se debe potenciar la aplicación de los patrones Experto, Creador, Controlador, Alta cohesión y Bajo acoplamiento para concebir un módulo con una arquitectura sólida y lo más escalable posible.

1.7 Metodología de desarrollo de software

Una metodología de desarrollo de software constituye una guía que define las tareas y actividades que se deben realizar para obtener un producto informático con una buena calidad. Esta indica cuál es el personal que debe participar en el desarrollo de las distintas actividades así como el papel que deberán enfrentar. Además, muestra toda la información necesaria para culminar o iniciar alguna actividad que se genere como resultado de los diferentes procesos por los cuales transcurre un

software. Las metodologías de desarrollo tienen dos enfoques: enfoque tradicional/metodologías robustas y enfoque ágil/metodologías ágiles.

Metodologías robustas

Las metodologías tradicionales o robustas imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Estas se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. No se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar (21). Entre ellas se destacan: RUP[6] (Rational Unified Procces), MSF (Microsoft Solution Framework), Iconix, entre otras.

Metodologías ágiles

Las metodologías ágiles combinan un conjunto de directrices de desarrollo donde resaltan la entrega sobre el análisis y el diseño, además de buscar la satisfacción del cliente y entrega temprana del software incremental, equipos de proyecto pequeño y con alta motivación. Son menos orientadas al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. Más orientadas al código, siguiendo un camino que indica que la parte más importante de la documentación es el código fuente. Entre ellas se destacan: XP (eXtreme Programming), Scrum, ASD (Adaptive Software Development) y OpenUP.

Actualmente no existe una metodología universal que le haga frente con éxito a cualquier proyecto de desarrollo de software. Para la selección de la metodología se ha de tener en cuenta las características del equipo de desarrollo. El equipo de GDR v2.0 tiene experiencia en el desarrollo de proyectos de manera conjunta. Por lo que desempeñan tareas diferentes y aportan distintas habilidades al proyecto, pero persiguen un objetivo común. Además, se ejecutan actividades y se genera información que ayuda a entender el trabajo del equipo por lo que existe una fuerte colaboración entre los miembros, adquiriendo una responsabilidad compartida.

Teniendo en cuenta estas características, el grupo de arquitectura del GDR v2.0 seleccionó a OpenUp como la mejor opción para enfrentar este proceso de desarrollo de software. Es una metodología ágil, orientadas al código y que se ajusta a los continuos cambios del proceso de desarrollo.

[6] **RUP:** constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

1.7.1 OpenUP

OpenUP mantiene las mismas características que RUP: contiene el desarrollo iterativo, casos de uso y escenarios de conducción de desarrollo, gestión de riesgos y el enfoque centrado en la arquitectura. Tiene los componentes básicos que pueden servir de modelo a procesos específicos. Además, la mayoría de los elementos están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

Es un proceso iterativo para el desarrollo de software que es mínimo porque solo incluye el contenido del proceso fundamental. El mismo, se caracteriza por ser un proceso completo ya que puede ser manifestado como proceso entero para construir un sistema. Igualmente, se comporta de forma extensible porque puede ser utilizado como base para agregar o para adaptar más procesos.

La metodología OpenUp, divide en cuatro fases el desarrollo del software:

- **Inicio:** se determinar la visión del proyecto.
- **Elaboración:** el objetivo es determinar la arquitectura óptima.
- **Construcción:** en esta etapa el objetivo es obtener la capacidad operacional.
- **Transición**: se obtiene la integración del proyecto.

Durante la fase de inicio se presenta una arquitectura candidata y se determinan algunos de los casos de uso arquitectónicamente significativos. En la fase de elaboración el objetivo fundamental es la construcción de la línea base de la arquitectura del sistema. Después en la fase de construcción se genera el modelo de implementación y se lleva a cabo el proceso de integración.

El presente trabajo de diploma se enmarcará en el desarrollo de los flujos de trabajo de Diseño e Implementación, de mayor peso en las fases de Elaboración y Construcción. Con el desarrollo del diseño del sistema se persigue adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario. De este modo, crea una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases. Asimismo, Descompone los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Por otra parte en el flujo de implementación es donde se define cómo se organizarán las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, así como la estructura de capas de la aplicación.

Se desarrollarán esencialmente los roles de diseñador y desarrollador de software en el Diseñador de Modelos para GDR v2.0, debido a que el mayor peso de la solución recae sobre estas etapas del desarrollo del software.

Diseñador en la metodología OpenUP

El diseñador tiene la misión de generar el diseño del sistema. Este debe ser capaz de concebir el diseño arquitectónico detallado del sistema, basándose en los requisitos y en la generación de prototipos rápidos para verificar dichos requisitos. El propósito del diseñador es crear una estructura interna limpia y relativamente simple. Desde esa perspectiva, una de las metas que persigue es derivar la arquitectura del sistema. Esta arquitectura sirve como marco desde el cual se conducen más actividades de diseño detallado. Al finalizar el flujo de trabajo, el diseño debe contener una organización jerárquica que haga un uso inteligente del control entre los elementos del software y debe conducir a procedimientos que muestren características funcionales independientes.

Desarrollador en la metodología OpenUP

El desarrollador debe tener experiencia en el desarrollo de aplicaciones en el ambiente seleccionado y debe dominar diferentes paradigmas de programación y estilos. Deberá conocer perfectamente las técnicas de diseño empleadas por el diseñador. De ahí que, el desarrollador es quien escribe el código fuente de los componentes, reutiliza código, compila, realiza prueba de unidad e implementa los elementos del modelo de diseño. Además, si encuentra defectos en el diseño, regresa a esta fase y los corrige, también arregla defectos de código y realiza pruebas de unidad para verificar los cambios.

Artefactos que se deberán generar

En correspondencia de la metodología definida y los roles involucrados en el desarrollo de la solución del sistema se generarán los siguientes artefactos.

Clase del diseño: se especifican utilizando la sintaxis del lenguaje de programación elegido. Definen la visibilidad de los atributos y operaciones. Los métodos tienen correspondencia directa con los métodos en la implementación.

Subsistema de diseño: es una forma de organizar los artefactos del modelo de diseño en piezas más manejables. Puede contar con clases del diseño, realización de casos de uso, interfaces, u otros subsistemas.

Paquetes de Diseño: se emplea para estructurar el modelo de diseño dividiéndolo en componentes más pequeños. Se utiliza como herramienta organizativa del modelo, para agrupar relaciones,

ejecuciones de guión de uso, diagramas y otros paquetes. Su contenido es responsabilidad de un único rol: diseñador.

Realización de Casos de Uso del Diseño: es una colaboración en el modelo de diseño que describe cómo se realiza un caso de uso específico en términos de clases de diseño. Posee una descripción de flujo de eventos textual.

Colaboración: sociedad de clases y otros elementos que colaborarán para realizar el comportamiento expresado en un caso de uso.

Modelo de Implementación: para su realización se establece que estructura contendrá, se crean los subsistemas de implementación y se definen las dependencias entre ellos. Finalmente, se actualiza en la Vista de Implementación de la arquitectura.

1.8 Conclusiones del Capítulo I

En este capítulo quedaron definidas las bases arquitectónicas para el desarrollo del módulo. Desde esa perspectiva, se realizó un estudio de las tecnologías y herramientas seleccionadas por el equipo de arquitectura del proyecto Generador Dinámico de Reportes v2.0. Se seleccionó el lenguaje PHP como lenguaje de desarrollo, para incrementar el dinamismo de la aplicación y utilizar las potencialidades de la red. Se utilizará el IDE NetBeans 7.0 por ser libre y tener gran integración con PHP. También se seleccionó el marco de trabajo Symfony 1.4.8, para que atienda las peticiones del servidor y Ext-Js como tecnología del lado del cliente, dado que posibilita la reutilización de componentes y la validación interfaces. Se empleará el Visual Paradigm como herramienta de modelado, por su rapidez en la construcción de aplicaciones de calidad y para documentar y visualizar los resultados se hará uso del lenguaje de modelado UML. En todo momento, se seguirán las pautas que propone la metodología de desarrollo OpenUp ya que esta propone procesos ágiles y ligeros para construir software con buenas prácticas.

2

CAPÍTULO II: DISEÑO DEL MÓDULO

Introducción

En el presente capítulo se describe el diseño de clases del sistema. Para ello se desarrollan los artefactos correspondientes al flujo de trabajo diseño. En este sentido, se realizan los diagramas de clases del diseño, por medio de los cuales se muestra la estructura estática del sistema. Para modelar los aspectos dinámicos, se realizan los diagramas de interacción donde se representan las clases y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellas. Además, se especifica la estructura física de la solución que se propone mediante el diagrama de casos de uso del sistema, la vista lógica y el diagrama de despliegue.

2.1 Modelo de Dominio

El modelo de Dominio es una representación visual estática del entorno real de los objetos del proyecto. De ahí que, se modelan los términos más relevantes del entorno donde se desarrolla el software y los elementos relacionados con este. Se centra en una parte del negocio, la relacionada con el ámbito del proyecto. Ayuda a comprender los conceptos que utilizan los usuarios, con los que trabajan y con los que deberá trabajar la aplicación. A continuación se presenta el Modelo de Dominio (ver figura 1) del módulo Diseñador de Modelos y se describen sus clases.

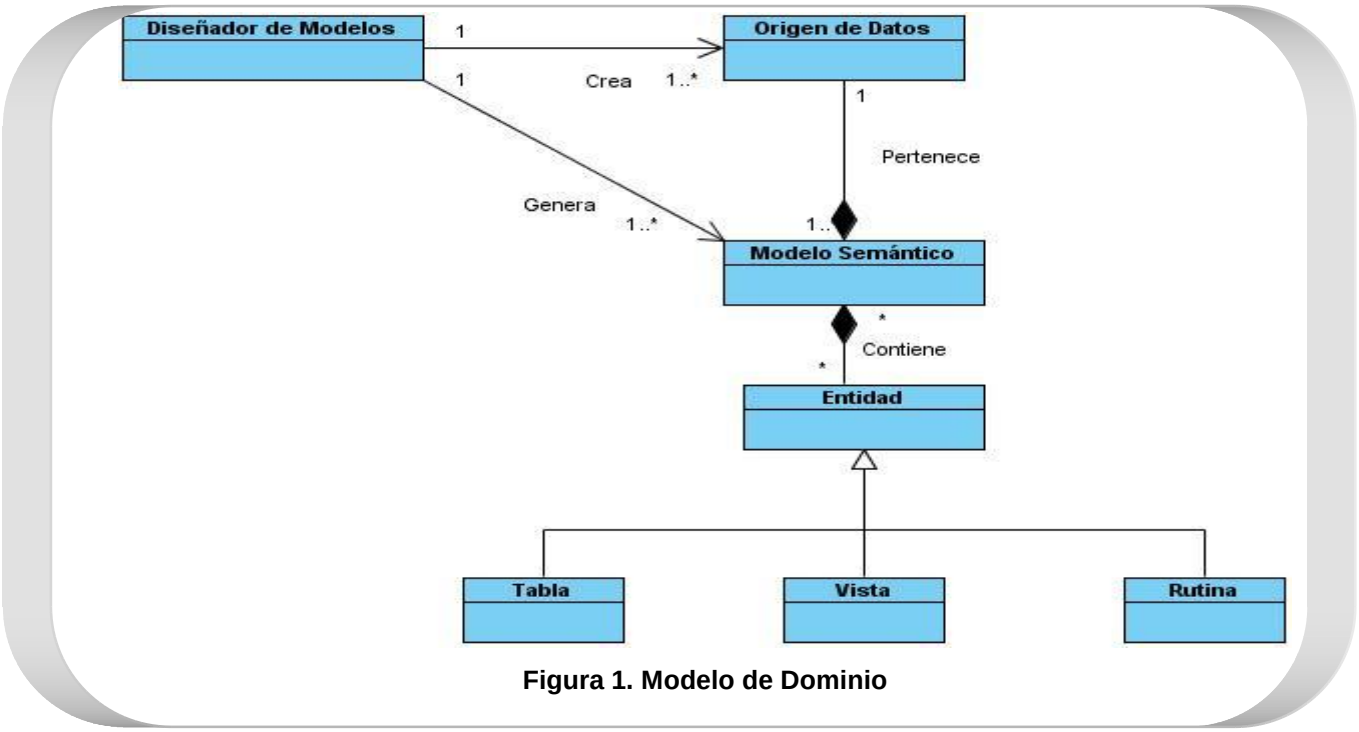

Figura 1. Modelo de Dominio

En este diagrama la clase Diseñador de Modelos representa al usuario GDR, el cual puede crear los orígenes de datos o generar los modelos semánticos. En este contexto, es imprescindible destacar que en el proceso de generación de un modelo se requiera de un origen de datos que lo provea, es decir, que estará asociado a un único origen de datos. Mientras que a partir de un origen de datos se pueden construir varios modelos. Estos últimos contienen los diferentes tipos de entidades: Tabla, Vista y Rutina.

Descripción de las clases del dominio

Origen de Datos: contiene los parámetros necesarios para conectarse a la fuente de los datos.

Modelo Semántico: contiene las entidades que serán utilizadas en los reportes y previamente cargadas en los orígenes de datos.

Entidad: representa una clase contenedora de información del negocio, puede ser una tabla, vista o rutina.

Diseñador de Modelos: encargada de diseñar los modelos semánticos, asociando un origen de datos para el diseño de los modelos semánticos que serán utilizados en los reportes.

Tabla: tipo de entidad que contiene las tablas de las bases de datos.

Vista: tipo de entidad que contiene las vistas de las bases de datos.

Rutina: tipo de entidad que contiene las rutinas de las bases de datos.

2.2 Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno de un software, son condiciones que el sistema ha de cumplir. Estos muestran las funcionalidades que deben satisfacerse para cumplir con las especificaciones de software. El módulo Diseñador de Modelos del Generador Dinámico de Reportes v2.0 debe cumplir con los requisitos funcionales que a continuación se describen:

- **RF1 Visualizar orígenes de datos:** permitirá visualizar las bases de datos disponibles.
- **RF2 Seleccionar un origen de datos:** se selecciona un origen de datos y se muestran los modelos semánticos creados a partir de este.
- **RF3 Mostrar las entidades del origen de datos seleccionado:** el sistema debe mostrar todas las tablas, vistas y rutinas presentes en el origen de datos seleccionado.
- **RF4 Seleccionar entidades desde el origen de datos seleccionado:** el sistema debe brindar la posibilidad de seleccionar las tablas, vistas y rutinas para diseñar un modelo semántico.
- **RF5 Modificar orígenes de datos:** permitirá modificar cualquiera de los parámetros utilizados para la conexión con la base de datos.

- **RF6 Adicionar origen de datos:** se registra en el sistema un nuevo origen de datos, el mismo cuenta con los datos de conexión al gestor de bases de datos.
- **RF7 Eliminar origen de datos:** se elimina del servidor un origen de datos previamente registrado y seleccionado por el usuario, siempre que no tenga asociado a él un modelo semántico.
- **RF8 Crear modelos semánticos:** se crea un modelo semántico asociado a un origen de datos previamente registrado en el sistema. El mismo tiene asociadas las entidades pertenecientes a un origen de datos.
- **RF9 Buscar los modelos semánticos:** el sistema debe brindar la posibilidad de buscar los modelos semánticos a partir de un criterio de búsqueda introducido, especificando una parte del nombre o el nombre completo del modelo semántico.
- **RF10 Modificar modelos semánticos:** es modificado el modelo semántico seleccionado del listado de modelos disponibles. Permitirá agregar o quitar entidades de un modelo específico, así como cambiarle el alias a las mismas.
- **RF11 Eliminar modelos semánticos:** es eliminado del sistema un modelo semántico del listado de modelos disponibles.
- **RF12 Obtener entidades relacionadas:** el sistema debe brindar la funcionalidad de obtener las entidades que se relacionan con las entidades seleccionadas.
- **RF13 Renombrar un modelo semántico:** el sistema debe brindar la funcionalidad de cambiar el nombre a un modelo semántico seleccionado.
- **RF14 Probar conexión a un origen de datos:** se prueba establecer la conexión a un origen de datos para verificar que la conexión funciona correctamente, antes de registrar o modificar el origen de datos en el sistema.
- **RF15 Visualizar modelos semánticos:** se visualizan los modelos semánticos con sus respectivas (tablas, vistas y rutinas) disponibles en el sistema previamente creados por el usuario.
- **RF16 Renombrar las entidades del origen de datos seleccionado y los atributos de esas entidades:** el sistema deberá permitir al usuario asignarle alias a las entidades del origen de datos y a los atributos de estas entidades, para crear un modelo semántico personalizado.
- **RF17 Mostrar las entidades del modelo semántico seleccionado:** el sistema debe mostrar todas las tablas, vistas y rutinas presentes en el modelo semántico seleccionado.
- **RF18 Seleccionar entidades del modelo semántico seleccionado:** el sistema deberá permitir seleccionar tablas, vistas y rutinas presentes en el modelo semántico seleccionado.

- **RF19 Mostrar el origen de datos desde el cual fue diseñado el modelo semántico:** el sistema deberá permitir que al seleccionar un modelo semántico se muestre el origen de datos desde el cual fue diseñado dicho modelo semántico.

2.3 Requisitos no funcionales

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar las operaciones que un sistema realiza. Constituyen propiedades o cualidades que el producto debe tener. El documento especificación de requisitos del DM para el GDR v2.0 recoge los requerimientos no funcionales que deben cumplirse en el proceso de desarrollo del sistema. Algunos de ellos se describen a continuación.

Usabilidad

- Se desarrollará para la web, pero con características muy similares a las aplicaciones de escritorio, en cuanto al diseño de las interfaces visuales y los tiempos de respuesta.

Eficiencia

- El sistema debe mantener tiempos de respuestas en un marco razonable de diez segundos, permitiendo que existan al menos 10 usuarios conectados de forma simultánea.

Interfaz

- El usuario deberá acceder a la aplicación a través del protocolo HTTP usando el navegador Firefox en su versión 4.0 o superior.

Restricciones de Diseño e Implementación

- El sistema deberá ser implementado en el lenguaje de programación PHP versión 5.3 o superior.
- Se utilizará el framework de desarrollo Symfony en su versión 1.4.8 y la librería de JavaScript Ext-Js versión 3.3.
- Se empleará la herramienta de desarrollo NetBeans 7.0 y el sistema gestor de base de datos PostgreSQL 9.0.

Software

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4.0 GNU/Linux o superior.
- Paquetes: apache2, php5, libapache2-modphp5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-xsl, php5-gd.

- Usuario con privilegios de administración.

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

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4.
- PostgreSQL versión 8.3 o superior.
- PGAdmin III u otro administrador compatible con PostgreSQL.
- Usuario con privilegios para instalar la base de datos.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP.

Hardware

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

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- Como mínimo requiere 512 MB de memoria RAM.
- Necesita espacio en disco duro igual o superior a 40 GB.

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

- Procesador Intel Pentium 4 1.7 GHz o AMD similar.
- Necesita memoria RAM igual o superior a 512 MB.
- Como mínimo requiere 40 GB de espacio en disco duro.

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

- Procesador Intel Pentium 4 1.7 GHz, o AMD similar.
- 256 MB RAM.
- 20 GB de espacio en disco duro.

2.4 Diagrama de caso de usos del sistema

El diagrama de casos de uso del sistema (DCUS) se utiliza para describir las funcionalidades de un software y documentar su comportamiento. Los casos de uso (CU) engloban los requisitos funcionales de un sistema, representando las funciones que la aplicación puede ejecutar. A continuación se presenta el DCUS (ver figura 2) del módulo en desarrollo.

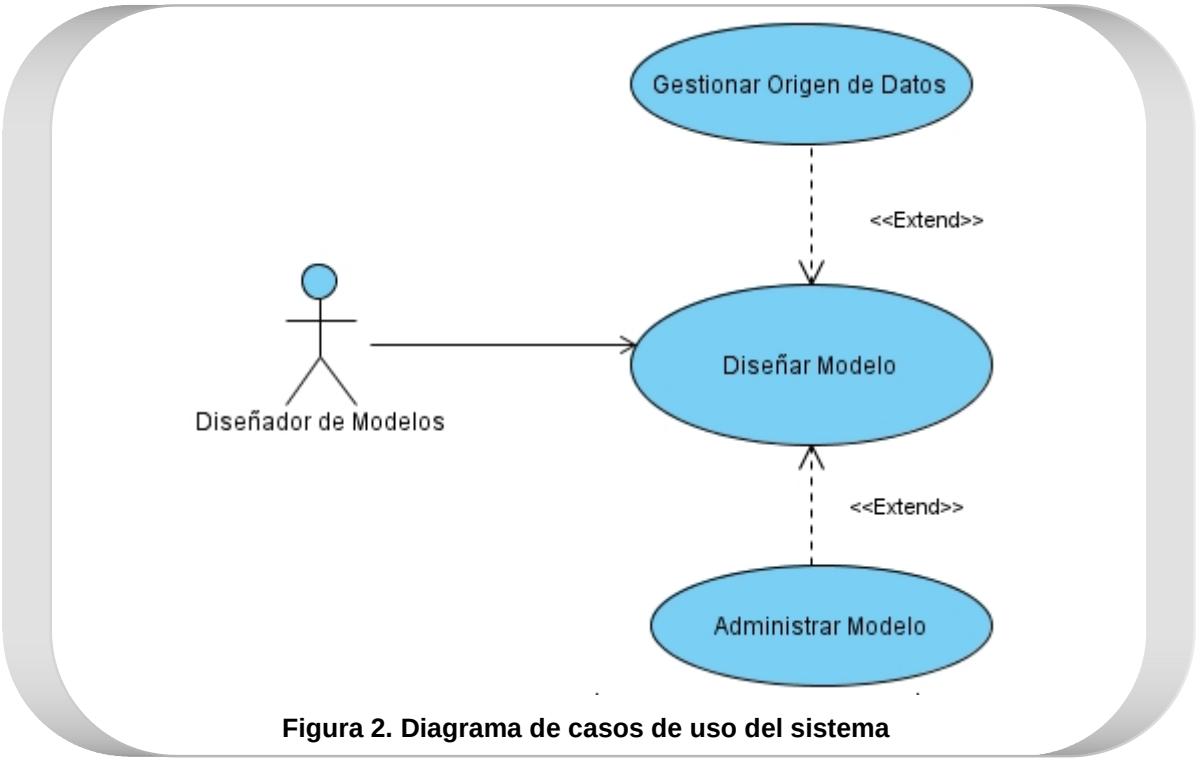

Figura 2. Diagrama de casos de uso del sistema

En este DCUS, el actor Diseñador de Modelos es el encargado de inicializar el caso de uso Diseñar Modelo, considerado crítico por su impacto en la arquitectura del módulo. Como función opcional este actor tiene la posibilidad de gestionar orígenes de datos, elementos imprescindibles para el diseño de un modelo semántico. Esta actividad es crítica por el impacto que proporciona en la operatividad del sistema. La administración de los modelos es una tarea que puede o no realizarse, es por ello por lo que este caso de uso extiende del principal. De esta forma, se lograron agrupar los diecinueve requisitos funcionales en tres casos de uso, todos de alta prioridad para la arquitectura del DM.

Descripción de los casos de uso

Diseñar Modelo: permite crear nuevos modelos semánticos, a partir de un origen de datos.

Gestionar Origen de Datos: permite adicionar, modificar y eliminar un origen de datos.

Administrar Modelo: permite modificar, buscar, renombrar y eliminar un modelo existente en el listado de modelos semánticos.

Trazabilidad

La trazabilidad es un aspecto indispensable en el proceso de gestión de cambio del software. En un proceso de desarrollo de software se necesita que los requisitos sean trazables, es decir, que se puedan describir y seguir su ciclo de vida en ambos sentidos, hacia sus orígenes o hacia su implementación. Por lo que se afirma, que un requisito es trazable si se puede identificar todas las partes del producto final relacionadas con este requisito. Se persigue entender el alcance del proyecto

y ser capaz de gestionar los cambios de los requerimientos. En consecuencia, permite determinar el impacto que provoca en el proyecto, un cambio en un requerimiento. Determina el impacto de la falta de una prueba en un requerimiento, es decir, si una prueba falta, pueda que el requerimiento no sea satisfecho. Verifica que todos los requisitos del sistema sean satisfechos mediante la implementación y que la aplicación haga solo lo que debe hacer. Para facilitar el trabajo de determinar las relaciones entre los requisitos y el seguimiento de los mismos, se utiliza la matriz de trazabilidad[7] de requisitos funcionales contra casos de usos (ver Anexo 2).

2.5 Diagrama de clases del diseño

En la fase de diseño se persigue el desarrollo de una arquitectura estable y sólida. Este flujo de trabajo es un refinamiento del análisis, que se preocupa en cómo van a ser implementados los requerimientos. Sus principales objetivos son: transformar los requisitos en un diseño del sistema, evolucionar hacia una arquitectura sólida y adaptar el diseño para que se ajuste al entorno de implementación.

El diagrama de clases del diseño (DCD) muestra las propiedades y funcionalidades de cada clase en el lenguaje de desarrollo y tecnologías seleccionados. Expresa la colaboración y responsabilidades de cada clase entorno al sistema que conforman. El siguiente diagrama de clase del diseño (ver figura 3) contiene las principales clase con sus respectivos métodos y atributos para el caso de uso Diseñar Modelo.

[7] **Matriz de trazabilidad:** es una herramienta para saber que requerimientos quedan cubiertos con una prueba.

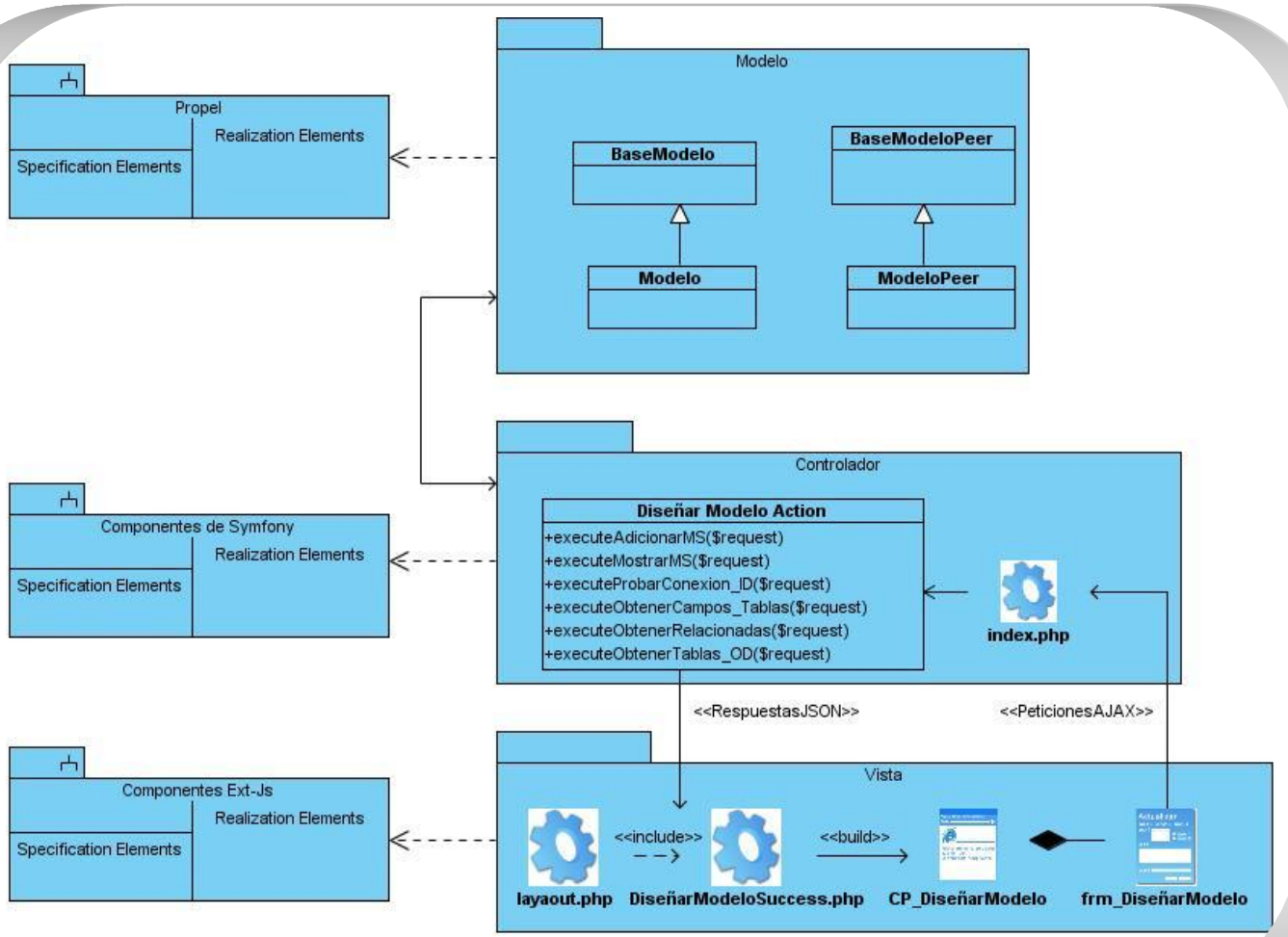

Figura 3. Diagrama de clases del diseño para el caso de uso Diseñar Modelo

En este diagrama los elementos del modelo corresponden a las clases generadas por el ORM Propel modelado por el subsistema de igual nombre. El mismo genera cuatro clase por cada tabla de la bases de datos, esto permite refactorizar las funcionalidades sin perder la información contenida hasta ese momento. Las acciones son los elementos que construyen las respuestas del servidor, estas se encuentran agrupadas según el correspondiente caso de uso al que pertenecen e implementadas en el lenguaje PHP, con la dependencia del subsistema “Componentes de Symfony” y comprobadas por el controlador frontal “index.php”. Se destaca la relación de flujo de información que existe entre el componente caso de uso del cliente y el componente módulo del servidor. Las solicitudes se realizan por medio de la tecnología AJAX y las respuestas son devueltas específicamente en formato JSON. Los elementos del lado del cliente corresponden a componentes específicos del negocio como paneles de trabajos asociados al caso de uso y a los componentes genéricos fuertemente reutilizables e implementados en lenguaje JavaScript utilizando los componentes del subsistema “Componentes Ext-Js”.

2.6 Diagrama de interacción del diseño

Un diagrama de interacción (secuencia o colaboración), representa la forma en que un cliente (actor) y otros objetos (clases) se comunican entre sí, en respuesta a un determinado evento. Recorriendo toda la secuencia de llamadas se obtienen las responsabilidades, por lo que modelan aspectos dinámicos de un sistema.

En el presente trabajo se utilizan diagramas de secuencia, para organizar los eventos con la sucesión temporal en que se desencadenan. Este tipo de diagrama, muestra los objetos que participan en la interacción mediante las líneas de vida y los mensajes que intercambian. A continuación se representa el escenario “Eliminar Origen de Datos” (ver figura 4) perteneciente al caso de uso Gestionar Origen de Datos. Los demás diagramas de interacción se pueden consultar en el Anexo 1.

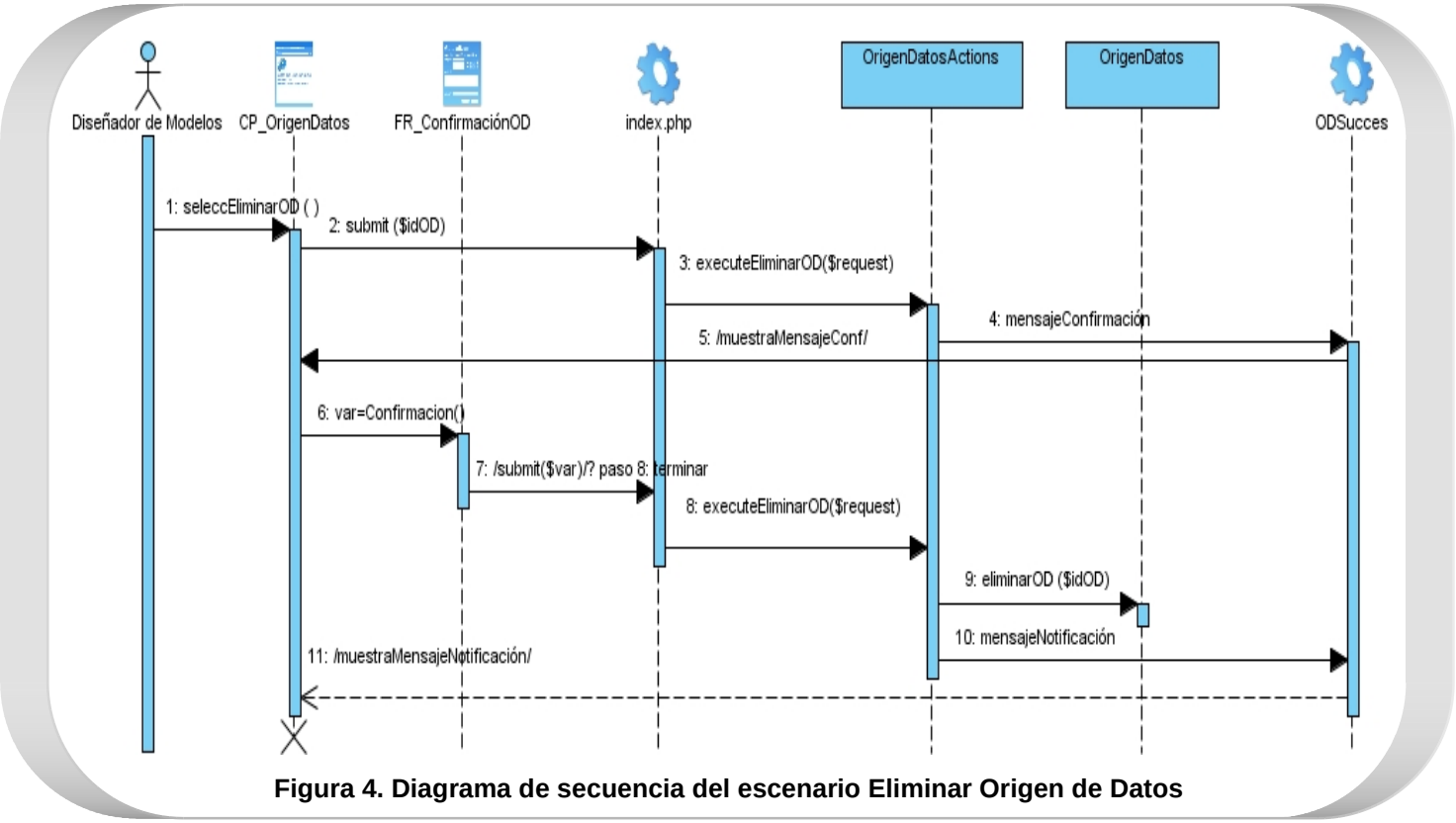

Figura 4. Diagrama de secuencia del escenario Eliminar Origen de Datos

En este diagrama de secuencia el actor Diseñador de Modelos selecciona un origen de datos y ejecuta la opción “Eliminar Origen de Datos”. La página cliente del origen de datos se encarga de enviar a través de la operación “submit”, el identificador del origen seleccionado al controlador frontal. Este tiene la tarea de ejecutar la acción específica para cumplir esta orden, en este caso sería “executeEliminarOD ($request)”. Posteriormente se le envía un formulario de confirmación al usuario para saber si realmente desea eliminar el origen de datos en cuestión. De ahí que si se procede a

eliminar, el actor debe aceptar la confirmación y esta vez se ejecutaría la acción sobre la clase entidad “Origen_Datos”, eliminando definitivamente el origen de los registros de esta clase. Finalmente el sistema envía un mensaje de notificación al usuario para que conozca del éxito de la operación.

2.7 Vista Lógica

Esta vista constituye un subconjunto del artefacto Modelo de Diseño, donde se representan los elementos de diseño más relevantes para la arquitectura del sistema. Describe las clases más importantes, su organización en paquetes y subsistemas de diseño. También describe las realizaciones de casos de uso más críticos como por ejemplo las que describen aspectos dinámicos del sistema. La Vista Lógica muestra el Modelo de Diseño cuando se usa un método de diseño orientado a objetos. Esta proporciona una base para comprender la estructura y la organización del diseño del sistema. Describe las funcionalidades del sistema en sus dos aspectos esenciales: la estructura de los componentes que lo integran, y su comportamiento. A continuación se detalla la Vista Lógica del DM v2.0 (ver figura 5), aplicando la arquitectura definida por el Ing. Armando Robert Lobo, en su trabajo de diploma “Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas”.

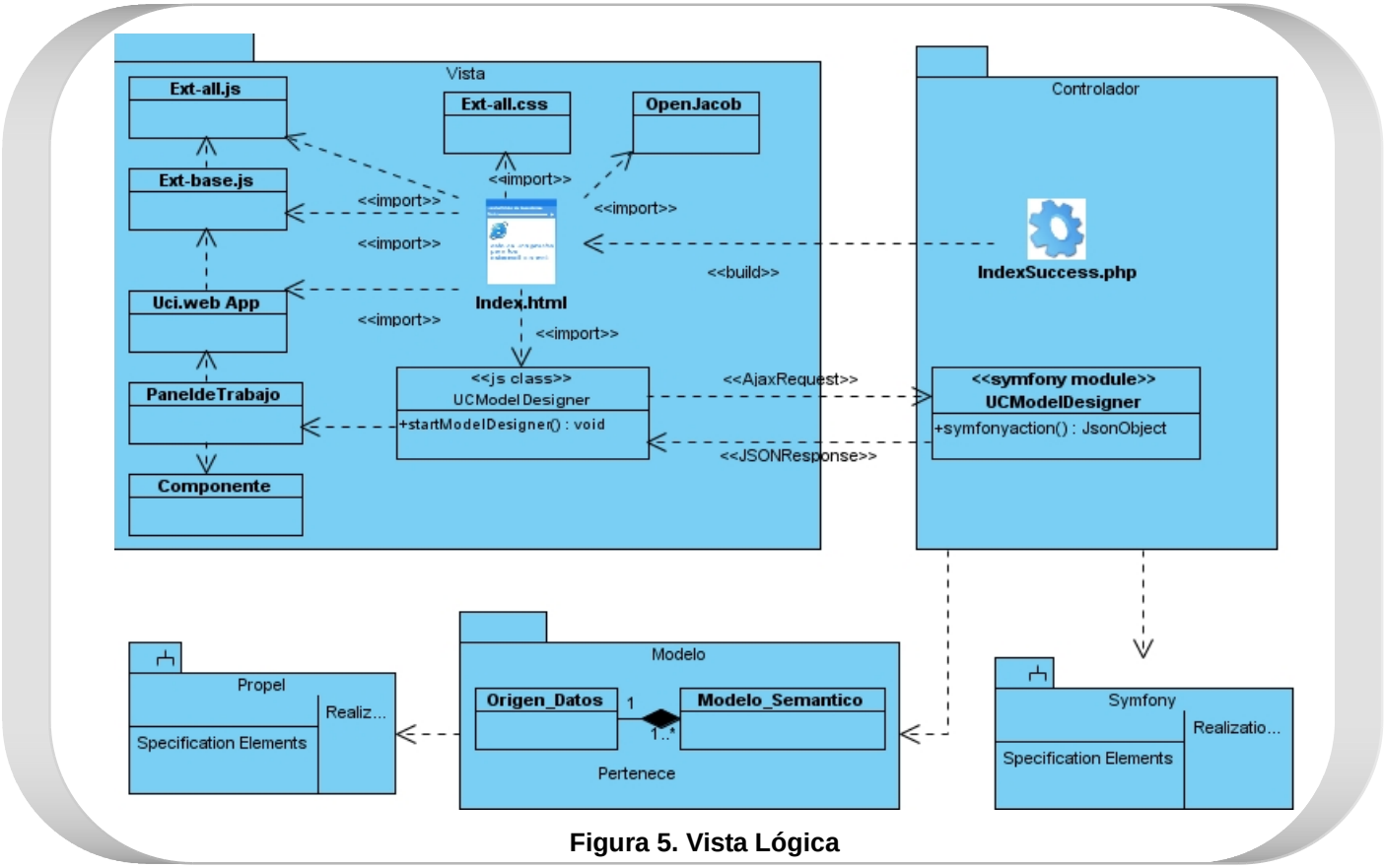

Figura 5. Vista Lógica

En este diagrama los componentes ubicados en la capa del cliente corresponden a las librerías “ext-base.js”, “ext-all.js”, “uci.webApp”, componentes específicos del negocio como paneles de trabajos, servicios asociados al caso de uso y a los componentes genéricos fuertemente reutilizables. Los elementos del controlador corresponden a las acciones, agrupadas en su correspondiente caso de uso en dependencia de la finalidad de las mismas. Estas son las encargadas de construir las respuestas del servidor (representada con el estereotipo “symfonyaction ()”) agrupadas en el correspondiente caso de uso (representado con el estereotipo “<<symfony module>>”) al que pertenecen. Se destaca la relación de flujo de información que existe entre el componente caso de uso del cliente y el componente módulo del servidor. Las solicitudes se realizan por medio de AJAX y las respuestas son devueltas específicamente en formato JSON.

2.8 Patrones utilizados en la solución

Los patrones estandarizan buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. En la solución del sistema se aplicaron principalmente los siguientes patrones:

Experto: se hace uso del mismo en la clase controladora UCModelDesigner (ver figura 6), indicando que la responsabilidad de la creación del componente “Panel de Trabajo” debe recaer sobre la misma, ya que posee toda la información de la lógica del caso de uso.

Controlador: utilizado para separar la lógica de negocio de la capa de presentación, controlando el flujo de eventos mediante las acciones de Symfony. En la arquitectura del módulo se definió una clase para cada acción en aras de alcanzar un bajo acoplamiento. Esto garantiza que los procesos de dominio sean manejados por la capa de los objetos de dominio y no por la capa de presentación.

Controlador Frontal: se aplica en la creación de un único punto de acceso para todas las peticiones realizadas al módulo, en este caso el “index.php” (ver figura 7). Escucha las peticiones que vienen desde una URL, luego se encarga de llamar al controlador específico, el cual maneja la acción requerida para satisfacer la petición realizada.

Bajo acoplamiento y Alta cohesión: se hace uso de los mismos en la creación de clases independientes para la lógica de los diferentes tipos de entidades (tablas, vistas y rutinas), así como para los atributos de cada una de estas. Esto trae como ventaja que solo se realicen acciones sobre el tipo de entidad que se solicite y no sobre todo el conjunto.

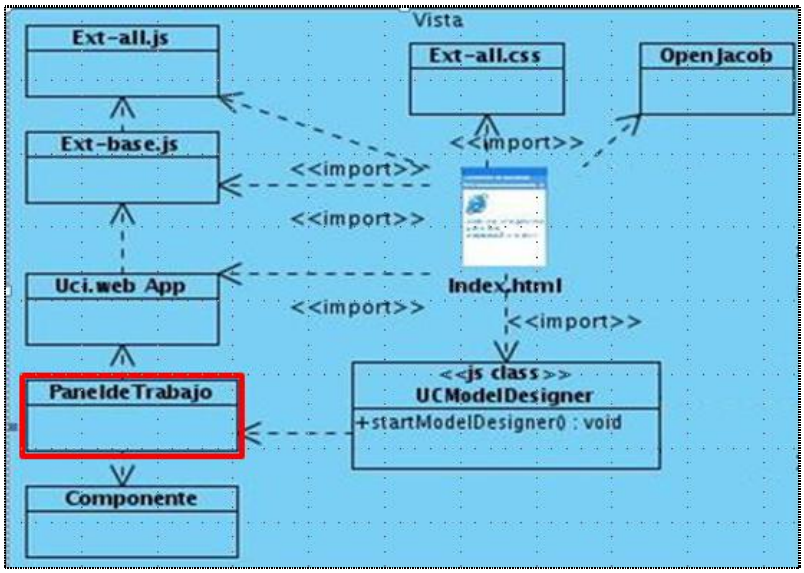

Figura 6. Patrón Experto

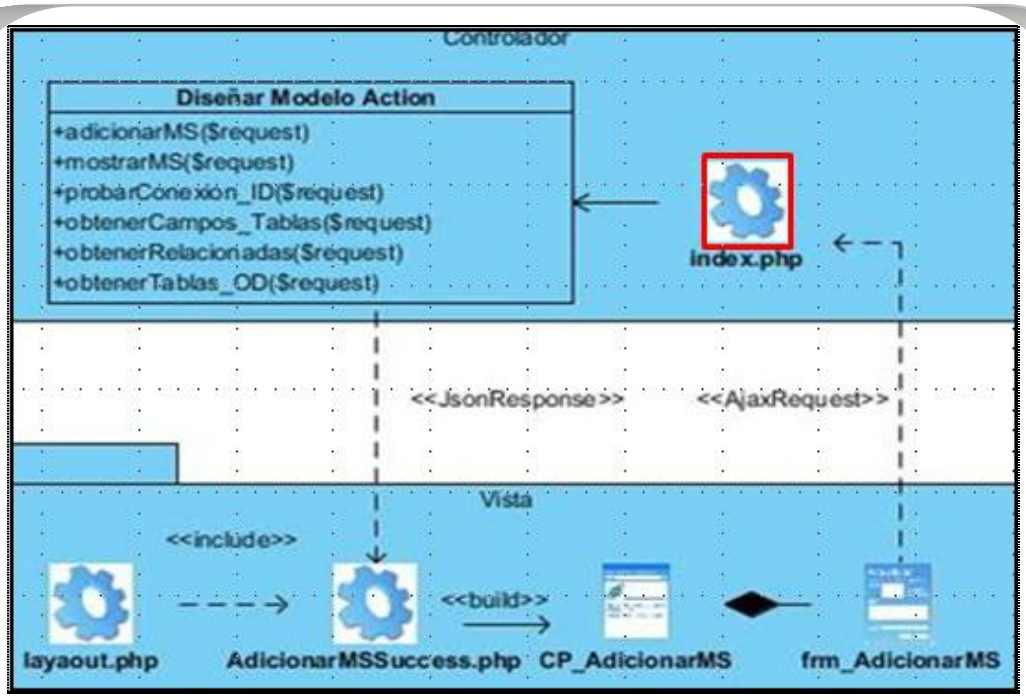

Figura 7. Patrón Controlador Frontal

2.9 Vista de Despliegue

La Vista de Despliegue se emplea para modelar el hardware y el software utilizado en los nodos del sistema. El Diseñador de Modelos es un módulo del Generador Dinámico de Reportes, por lo que asume las características generales de este. De ahí que pueda ser usado directamente o mediante un sistema externo que establezca una conexión con el mismo. A continuación se muestra la Vista

Despliegue para el GDR v2.0, entorno en el cual se encuentra ubicado la nueva versión del Diseñador de Modelos, por lo que los componentes resultantes del mismo deberán estar distribuidos en sus correspondientes nodos (ver figura 8).

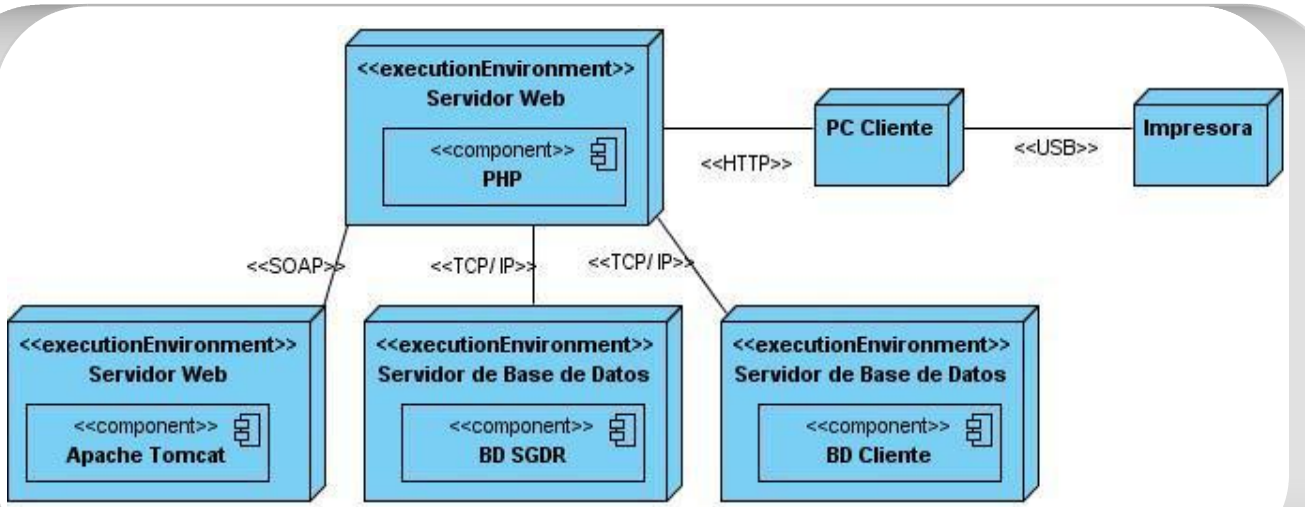

Figura 8. **Vista de Despliegue**

En este caso se necesitan dos servidores web: uno como entorno de ejecución con Apache 2 con soporte para PHP 5.3 y sistema operativo basado en GNU/Linux en cualquiera de sus distribuciones y el otro servidor como entorno de ejecución con Apache Tomcat para la instalación de Jasper Report 4.0. Esto es debido a que la aplicación está desarrollada en PHP, pero la librería para la generación de reportes utilizada (Jasper Report) que está desarrollada en Java por lo que se hace necesario vincular PHP/Java mediante servicios utilizando SOAP como protocolo de comunicación entre ambos servidores para el intercambio de información. La base de datos puede estar instalada en el mismo servidor Apache2 o en uno distinto en dependencia de las posibilidades de la organización, siendo necesario PostgreSQL, versión 9. Los usuarios podrán conectarse a la aplicación desde estaciones clientes e incluso desde el mismo servidor a través del navegador web Mozilla Firefox o cualquier otro que utilice el motor Gecko como es el caso de Epiphany Web Browser del escritorio Gnome (22).

2.10 Conclusiones del Capítulo II

En el desarrollo del presente capítulo se describieron los requisitos funcionales y no funcionales que el sistema debe cumplir. De igual manera, se realizó el diagrama de caso de uso del sistema, donde se agruparon diecinueve requisitos funcionales en tres casos de usos, considerados críticos para la arquitectura del módulo. Además se utilizó la matriz de trazabilidad de requerimientos y su dependencia con los respectivos casos de usos para seguir la línea de vida de los primeros y saber el impacto de cada uno en la realización de los casos de usos. Igualmente se modeló la Vista Lógica

como subconjunto del Modelo de Diseño, que a partir de los diagramas de clases servirán de base para la implementación del sistema. De igual manera, los diagramas de secuencia se utilizaron para inspeccionar los aspectos dinámicos del módulo. Se modeló la Vista de Despliegue teniendo en cuenta que el sistema es un módulo que se enmarca en la solución GDR y asume sus características y requerimientos de tecnológicos.

CAPÍTULO III: IMPLEMENTACIÓN Y PRUEBA DEL MÓDULO 3

Introducción

En el presente capítulo se generan los artefactos pertenecientes a las fases de implementación y prueba de un software. Se modela el sistema en términos de componentes, que serán distribuidos por los nodos de configuración. También se especifican los casos de pruebas realizados al módulo para validar su correcto funcionamiento.

3.1 Implementación

La implementación es el centro de atención en las iteraciones de la fase de construcción. En este flujo de trabajo se organizan y realizan las pruebas unitarias y se integran los componentes implementados, basándose en las especificaciones de diseño. Se define la organización del código, en términos de los subsistemas de implementación, organizados en capas. Se implementan los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros). La implementación brinda la posibilidad de probar y desarrollar componentes como unidades, que finalmente serán integrados como un sistema ejecutable.

3.2 Mecanismos de implementación

Los mecanismos de implementación definen los procedimientos y las normas que han de cumplirse en el proceso de generación de código. A continuación se listan los definidos por el equipo de arquitectura del GDR para la versión 2.0 del mismo sistema, los mismos estuvieron presentes en la construcción del Diseñador de Modelos v2.0.

- El sistema se desarrolló utilizando una arquitectura cliente-servidor.
- La lógica del servidor se implementó en el lenguaje de programación PHP5, utilizando Symfony 1.4.8 como framework de desarrollo para dicho lenguaje.
- Se mantiene el estilo arquitectónico MVC propuesto por Symfony.
- La capa del Modelo se genera utilizando Propel como herramienta de mapeo objeto relacional (ORM) por sus siglas en inglés, que se encuentra incluida en la versión de Symfony empleada.
- Cada acción de la capa del controlador se escribe en un fichero independiente según la estructura que define Symfony para este propósito.
- Todos los métodos, nombres de clases y variables se escribirán en estructura camelCase.

- El acceso a las bases de datos de los orígenes de datos se realiza mediante la capa de abstracción PDO[8] pero de forma unificada haciendo uso de la clase gdrPDO.
- Todas las librerías y clases de apoyo implementadas están incluidas en la carpeta /lib del proyecto, haciendo una estructura de carpetas lógica con previa aprobación por el equipo de arquitectura.
- El estándar de codificación será el propuesto por el IDE NetBeans 7.0 para PHP.
- Las respuestas que se envían desde el servidor mantienen el formato JSON.

3.3 Diagrama de componentes

Un diagrama de componentes representa la vista estática y dinámica de un sistema. Muestra como este se estructura en componentes, señalando la organización y las dependencias que existen entre ellos. Normalmente este tipo de diagrama contienen los componentes que representan la parte física de un sistema (módulo, base de datos, programa ejecutable). Se pueden agrupar en paquetes, y entre ellos pueden existir relaciones de dependencia como: generalización, asociación, agregación o realización. Además contienen las interfaces que constituyen la unión entre varios componentes. Los componentes pueden agruparse en paquetes y pueden contener subsistemas.

En un diagrama de componentes no se incluyen todos los componentes de un sistema, normalmente se modelan por partes; cada diagrama describe un apartado del sistema. Se utiliza para mostrar qué componentes pueden compartirse entre sistemas o entre diferentes partes de un sistema. A continuación se representa el diagrama de componentes perteneciente al caso de uso Diseñar Modelo (ver figura 9). Los demás diagramas se pueden consultar en el Anexo 3.

[8] **PDO:** es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos gestores de bases de datos.

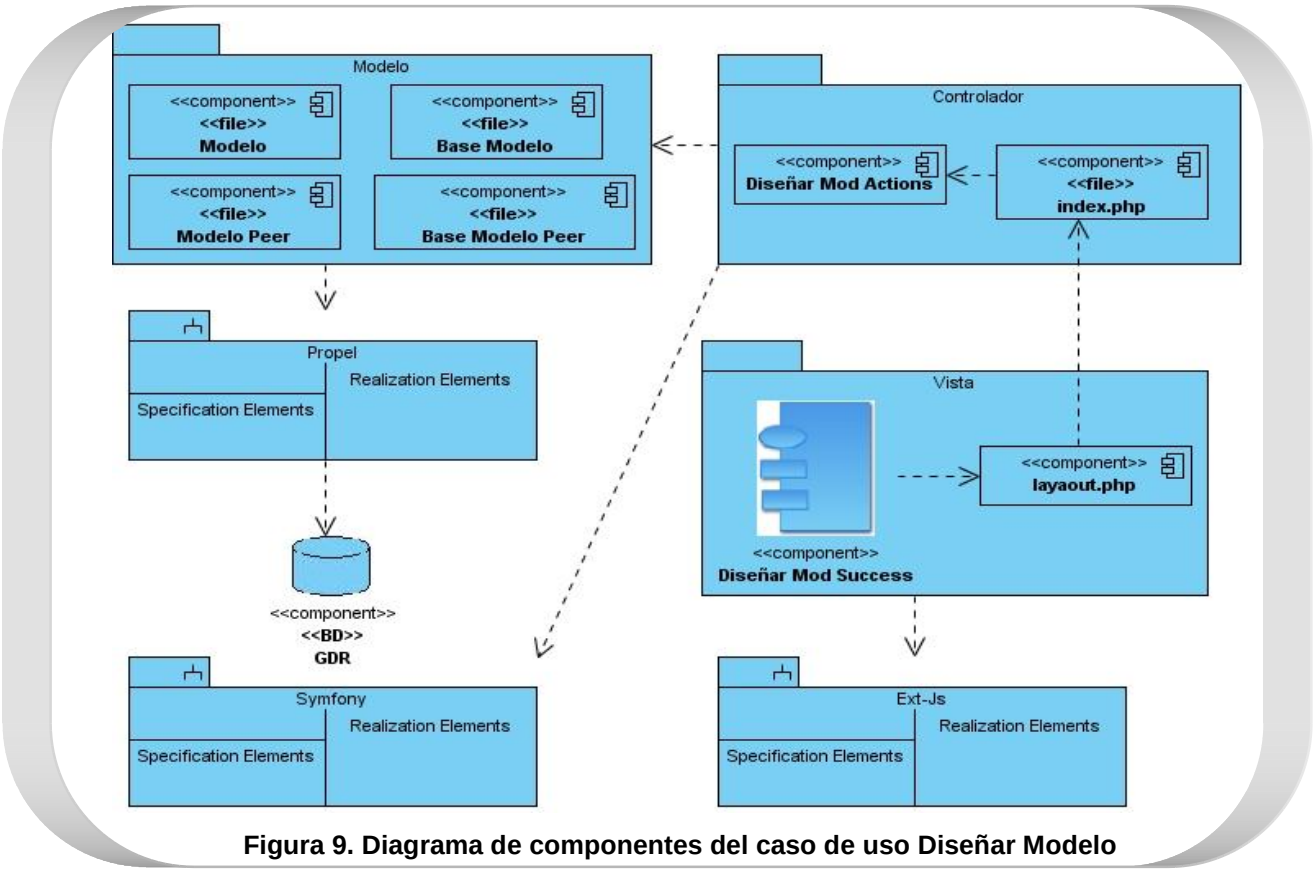

Figura 9. Diagrama de componentes del caso de uso Diseñar Modelo

En este diagrama, en el componente "**Diseñar_Mod_Actions**" se agrupan todas las acciones que intervienen en el caso de uso Diseñar Modelo, cada una de estas en su respectivo fichero PHP. Los componentes del modelo corresponden a las cuatro clases generadas por el subsistema Propel por cada una de las tablas de la base de datos del sistema. Mientras que en el componente "**Diseñar_Mod_Success**" se engloban las diferentes vistas que intervienen en este caso de uso.

3.4 Implementaciones relevantes

Dentro del grupo de mejoras realizadas al módulo se encuentra: independizar el trabajo con las diferentes entidades contenidas en los orígenes de datos. Se requerían funcionalidades que lograran separar la lógica de cada entidad (tabla, vista o rutina). Con el fin de dar cumplimiento a este requerimiento, específicamente para las entidades de tipo vistas se crea la clase **getViewsByDataSourceAction** (ver figura 10). Esta tiene que la responsabilidad de obtener todas las vistas que contiene el origen de datos seleccionado para diseñar los modelos semánticos. Esta funcionalidad recibe el parámetro "**id**" del origen de datos, de esta forma se obtiene dicho origen. Luego se obtienen las entidades asociadas a este origen y finalmente se escogen sólo las de tipo vista. De forma similar se procede para entidades de tipo tabla y de tipo rutina.

```php
<?php
class getViewsByDataSourceAction extends sfAction {

    public function execute($request) {
        $error = 'Error';
        $id = $request->getParameter('id');
        $dataSource = DataSourcePeer::retrieveByPK($id);
        if (!($dataSource instanceof DataSource))
            return $this->renderText(
                "{success:false, errors:{ Message:'" . ModelMessages::$DSEliminado . "'}}"
            );
        $this->getUser()->setAttribute('dataSource', $dataSource);
        $mCreator = new ModelCreator();
        $tables = $mCreator->getTablesByDS($dataSource);
        $json = array();
        foreach ($tables as $table) {
            $array = array();
            $type = $table->getType();
            if ($type === 'Vista') {
                $array['table'] = $table->getName();
                $array['alias'] = $table->getAlias();
                $array['type'] = $table->getType();
                $array['schema'] = $table->getSchema();
                $json[] = $array;
            }
        }
        $result = array();
        $result['data'] = $json;
        return $this->renderText(json_encode($result));
    }
```

Figura 10. **Clase para obtener las vistas de un origen de datos**

También se requería, independizar el trabajo con los atributos de las diferentes entidades seleccionadas atendiendo a su clasificación. Se necesitaba obtener los campos de los diferentes tipos de entidades, con el fin de modificarlos o simplemente visualizarlos. Para ello, específicamente para las entidades de tipo rutina, se crea la clase **getFieldsByFuntionsAction** (ver figura 11). La misma procede de forma similar a la anterior analizada hasta el punto donde se obtienen las entidades de tipo rutina. Luego se obtienen todos los atributos de cada una de las rutinas seleccionadas por el usuario y finalmente se visualizan en la interfaz “Editar atributos”. El usuario puede modificar estos atributos o simplemente consultarlos.

```php
<?php
class getFieldsByfunctionsAction extends sfAction {

    public function execute($request) {
        try {
            $tables = $request->getParameter('tables');
            $idM = $request->getParameter('idmodel');
            $dataSource = DataSourcePeer::retrieveByPK($idM);
            $fields = $this->SaveAtSession($tables, $dataSource);
            $json = array();
            foreach ($fields as $field) {
                $array = array();
                $array ['alias'] = $field->getAlias();
                $array ['name'] = $field->getName();
                $array ['type'] = $field->getType();
                $array ['key'] = $field->getKey();
                $array ['autoLoad'] = false;
                $array ['tableName'] = $field->getTableName();
                $array ['schema'] = $field->getSchema();
                $array ['tableView'] = $field->getTableView() . ": " . $field->getParentFullName();
                $json [] = $array;
            }
            $result ['data'] = $json;
            return $this->renderText(json_encode($result));
        } catch (Exception $e) {
            return $this->renderText(
                '{success:false, errors:{ Message:'" . ModelMessages::$OrigenDeDatosNoValido . "'}}"
                );
        }
```

Figura 11. **Clase para obtener los atributos de las rutinas de un origen de datos**

Como parte del proceso de optimización de código requerido en el desarrollo del Diseñador de Modelos v2.0 se crean componentes de tipo EditorGridPanel para los diferentes tipos de entidades y atributos de las mismas. Esta práctica reduce en gran medida el número de líneas de código del módulo, y promueve la reutilización del mismo. Permite que estos componentes se puedan utilizar tanto en el escenario de “Diseñar Modelo” como en el de “Modificar Modelo”. En el caso particular de las entidades de tipo tabla, se crea el componente **GridTabla** (ver figura 12) con la responsabilidad de obtener todas las instancias de este tipo pertenecientes a un origen de datos específico. Cada vez que se necesite utilizar el grid de las entidades “Tabla” solo se requiere instanciar un objeto de esta clase y definirle por parámetros el identificador del origen de datos del cual se extraerán las tablas.

```
Ext.namespace('patdsi.report_generator.model_designer');
patdsi.report_generator.model_designer.GridTabla= Ext.extend(Ext.grid.EditorGridPanel, {
    constructor: function(config) {
        config = config || {};
        var store = new Ext.data.GroupingStore({
            autoLoad:true,
            proxy: new Ext.data.HttpProxy({
                url:'/report_generator.php/model_designer/getTablesByDataSource',
                method: 'POST'
            }),
            baseParams: {
                id:config.datasource
            },
            reader: new Ext.data.JsonReader({
                root: 'data',
                totalProperty: 'allColum',
                id:'idt'
            },[
            {
                name: 'table',
                type: 'string'
            },
            {
                name: 'alias',
                type: 'string'
```

Figura 12. **Componente GridTabla**

3.5 Estándar de codificación

Un estándar de codificación completo comprende todos los aspectos que intervienen en la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiese escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurarse que todos los programadores del proyecto trabajen de forma coordinada. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones del código de rutinas.

Se utilizó el estándar de codificación que define el Centro de Tecnologías de Gestión de Datos, con las siguientes especificaciones:

- El estándar de codificación será el propuesto por el IDE Netbeans 7.0 para PHP con la excepción de que deberá cambiarse la configuración de las llaves y en todos los casos estas se colocarán en una nueva línea.

- Todos los métodos, nombres de clases y variables se escribirán en estructura camelCase, comenzando siempre en minúscula.

3.6 Diagrama de despliegue actualizado

Se presenta una vista actualizada del diagrama de despliegue, mostrando los principales artefactos y componentes de la aplicación, distribuidos en sus correspondientes nodos, así como las dependencias que existen entre cada uno de ellos.

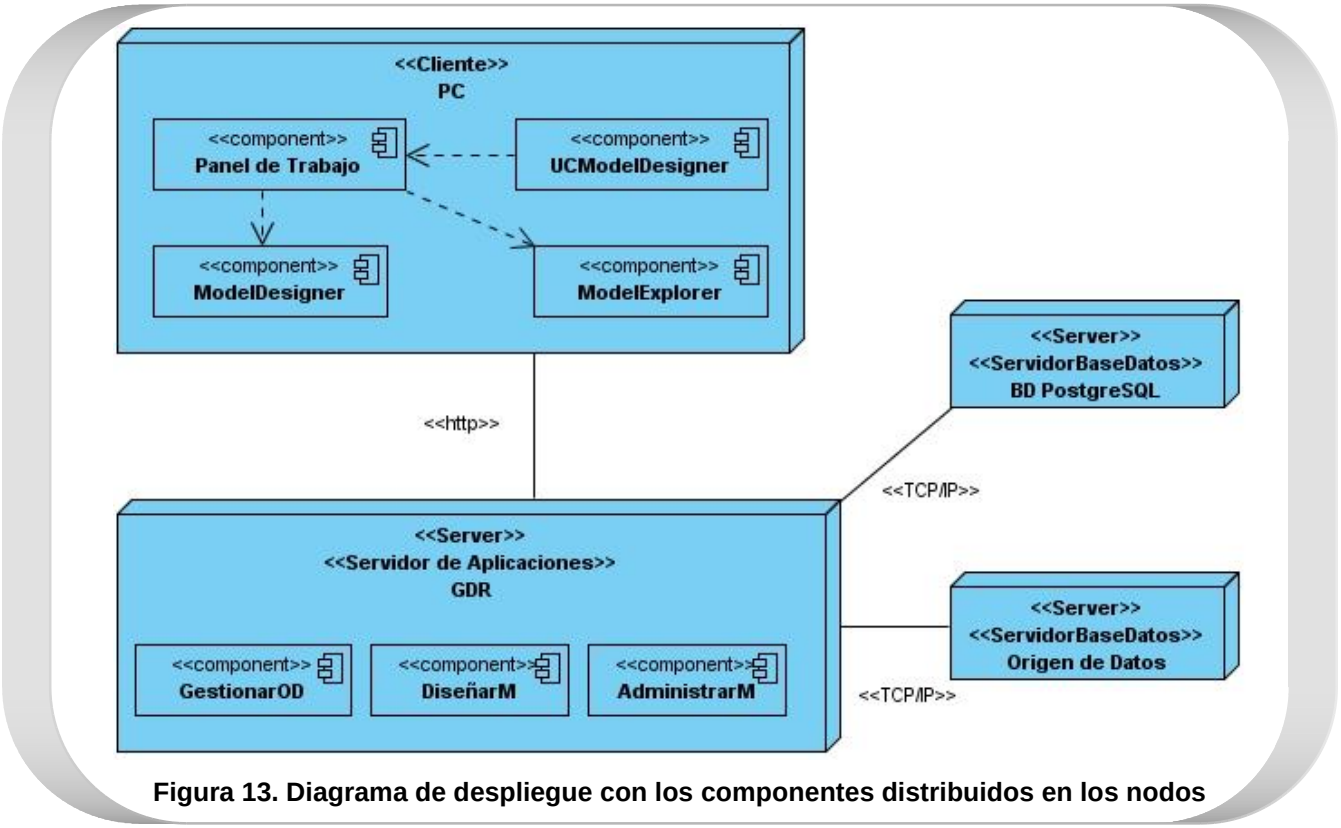

Figura 13. Diagrama de despliegue con los componentes distribuidos en los nodos

En este diagrama de despliegue los componentes de la vista del módulo se encuentran ubicados en el nodo que representa a la máquina del cliente "PC". Los componentes del controlador, donde se agrupar las acciones empaquetadas por casos de usos, se ubican en el nodo servidor de aplicaciones "GDR", conectado al cliente a través del protocolo "HTTP". También se utilizan dos servidores de bases de datos: uno donde se ubica la base de datos del sistema y otro donde se encuentran los datos del cliente, los cuales pueden ser consultados para la generación de los modelos. Ambos servidores de bases de datos se conectan al servidor de aplicaciones a través del protocolo "TCP/IP".

3.7 Prueba

Las pruebas de software permiten verificar y revelar la calidad de un software. Son utilizadas para identificar posibles fallos durante el proceso de desarrollo. Los casos de prueba son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos específicos. Los resultados son observados y registrados, realizando una evaluación de algún aspecto o componente del sistema. Permiten encontrar y documentar los defectos que puedan afectar la calidad del software. Verifican que el software trabaje como fue diseñado. Validan y prueban cada uno de los requisitos que debe cumplir el software y determina si estos fueron implementados correctamente.

3.7.1 Estrategia de prueba

Una estrategia de prueba describe el enfoque y los objetivos generales de este tipo de actividad. Incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos. Define qué técnicas (manual o automática) y herramientas serán usadas. Delimita los criterios de éxitos y culminación de las pruebas. También define consideraciones especiales relacionadas con los recursos necesarios para realizar esta tarea.
Teniendo en cuenta que el Generador Dinámico de Reportes v2.0 será revisado por el grupo de calidad de la universidad, en el presente trabajo solo se realizaron pruebas a nivel de desarrollador, utilizando las de tipo funcional y aplicando el método de caja negra.

Nivel de prueba

Los niveles de prueba especifican diferentes ángulos para verificar y validar un producto de software. Es como el tomar una radiografía a un cuerpo humano desde diferentes vistas y buscar donde hay un problema en los huesos. Existen diferentes niveles de prueba de software, uno de los más importantes es el nivel de desarrollador, que es la que realizan los desarrolladores de software en su código fuente. Esta prueba es totalmente diseñada e implementada por el equipo de implementadores del proyecto.

Tipo de prueba

Se denominan pruebas funcionales, a las pruebas que tienen por objetivo demostrar que los sistemas desarrollados, cumplen con las funciones específicas para los cuales han sido creados. Este tipo de prueba es común que sea desarrollada por los analistas de pruebas con apoyo de algunos usuarios finales. Están basadas en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Se realizan mediante el diseño de modelos que buscan evaluar cada una de las opciones con las que cuenta el paquete informático.

Método de prueba

El método de prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Con estos casos de prueba se pretende demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

3.7.2 Diseños de casos de pruebas

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Se parte de la descripción de estos últimos, como apoyo para las revisiones. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión.

Para detallar el caso de uso se utiliza una tabla, donde se desglosa esta funcionalidad en secciones y estas a su vez en escenarios, para hacer más fructífera la ejecución de las pruebas. A continuación se presentan las tablas de las secciones probadas para el caso de uso Gestionar Origen de Datos.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
SC1 construirOrigenDatosAction	SC 1.1: Adicionar origen de datos.	El diseñador de modelos decide introducir un nuevo origen de datos, el sistema envía un mensaje indicando que la petición se realizó.
SC2 eliminarOrigendeDatosAction	SC 2.1: Eliminar origen de datos.	El diseñador de modelos elimina el origen de datos seleccionado y el sistema envía un mensaje indicando que la petición se ejecutó.
SC3 modificarOrigenDatosAction	SC 3.1: Modificar origen de datos.	El diseñador de modelos decide cambiar elementos del origen de datos, el sistema envía un mensaje indicando que la petición se realizó.
SC4 mostrarOrigenDatosAction	SC 4.1: Mostrar origen de datos.	El diseñador de modelos decide ver todos los orígenes de datos, el sistema muestra en pantalla todos los orígenes de datos existentes.

Tabla 4. Secciones de prueba para el caso de uso Gestionar Origen de Datos

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

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	campo de texto	No	Es el nombre del origen de datos a registrar, para el cual solo se permiten letras, números y los siguientes caracteres:(“-“,”_”,”.”).
2	Tipo de gestor	campo de selección	No	Se selecciona el tipo de gestor de bases de datos al cual se conectará el sistema para realizar los reportes.
3	Servidor	campo de texto	No	Es el nombre o la dirección IP del servidor de bases de datos.
4	Puerto	campo de texto	No	Es el puerto que usa el servidor de bases de datos.
5	Usuario	campo de texto	No	Es el usuario para conectarse al servidor de bases de datos.
6	Clave	campo de texto	No	Es la clave del usuario para conectarse al servidor de bases de datos.
7	Base de datos	campo de selección	No	Se selecciona las bases de datos asociadas al servidor definido.
8	id	automático	No	Identificador de un origen de datos.

Tabla 5. Descripción de las variables

Esta descripción posibilitó que se realizara una matriz de datos, donde se evaluó y probó la validez de cada uno de los datos introducidos en el sistema, específicamente en la sección que se estuvo probando. Utilizando un juego de datos válidos e inválidos se registraron, con el empleo de la técnica de partición de equivalencia, los resultados de las pruebas. La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Matriz de Datos

Escenario	Variables (Enumeradas según descripción de la variable)							Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	1	2	3	4	5	6	7			
Adicionar origen de datos.	V	V	V	V	V	V	V	El sistema guarda los nuevos cambios del origen de datos en la base de datos, enviando un mensaje al usuario.	Satisfactorio.	El sistema envía los datos codificados en formato JSON, retornando al cliente la respuesta de confirmación de la ejecución.
	I	V	V	V	V	V	V	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con los datos del nuevo origen de datos.	Satisfactorio.	

Tabla 6. Sección construirOrigenDatosAction

Escenario	Respuesta del Sistema		Resultado de la Prueba	Flujo Central
Eliminar origen de datos.	V	El sistema elimina el origen de datos de la base de datos, enviando un mensaje de confirmación al usuario.	Satisfactorio.	El sistema envía los datos codificados en formato JSON, retornando al cliente la respuesta de confirmación de la ejecución.
	I	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con los datos del origen de datos.	Satisfactorio.	

Tabla 7. Sección eliminarOrigenDatosAction

Escenario	Variables (Enumeradas según descripción de la variable)				Respuesta del Sistema	Resultado de la Prueba	Flujo Central
	1	3	5	6			
Modificar origen de datos.	V	V	V	V	El sistema guarda los cambios del origen de datos en la base de datos, enviando un mensaje al usuario.	Satisfactorio.	El sistema envía los datos codificados en formato JSON, retornando al cliente la respuesta de confirmación de la ejecución.
	V	V	V	I	El sistema devuelve un mensaje tipo Exception debido a que ha ocurrido un error con los datos del nuevo origen de datos.	Satisfactorio.	

Tabla 8. Sección modificarOrigenDatosAction

Escenario	Respuesta del Sistema	Resultado de la Prueba	Flujo Central
Mostrar origen de datos.	El sistema muestra todos los orígenes de la base de datos.	Satisfactorio.	El sistema muestra todos los orígenes de datos existentes en la base de datos.

Tabla 9. Sección mostrarOrigenDatosAction

3.8 Conclusiones del Capítulo III

El desarrollo del presente capítulo permitió la realización de la descripción de la implementación del módulo, representando las dependencias que existen entre los principales componentes, a través del diagrama de componentes correspondiente a cada caso de uso. Se realizó el diagrama de despliegue de la aplicación, con los componentes distribuidos en los nodos de configuración del sistema. Se desarrollaron pruebas de caja negra utilizando la técnica de particiones de equivalencia para revelar la calidad del módulo implementado. Finalmente se mostraron los resultados obtenidos durante la realización de las pruebas, para determinar que el software posee una calidad aceptable.

CONCLUSIONES

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

- Se definió el diseño de clases del Diseñador de Modelos para el Generador Dinámico Reportes v2.0.
- Se realizó la implementación del diseño propuesto, para ello se modeló el sistema en términos de componentes, con los cuales se construyó el Diseñador de Modelos v2.0.
- Se aplicaron pruebas funcionales de tipo caja negra, descritas en los casos de pruebas realizados, validando con estas el correcto funcionamiento del módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0.

Con el desarrollo de esta investigación se alcanzó satisfactoriamente el objetivo propuesto, pues se logró la implementación del módulo Diseñador de Modelos para el Generador Dinámico de Reportes v2.0. Se aplicaron patrones de diseño y elementos de programación basada en componentes, lo que permitió aumentar la usabilidad y la reutilización en el proceso de diseño de los modelos semánticos para el diseño de los reportes.

RECOMENDACIONES

Al concluir este trabajo se recomienda:

- Desarrollar la lógica del servidor con la versión 2.0 del framework Symfony.
- Actualizar la capa de presentación con la versión 4.0 de la librería JavaScript Ext-Js.

REFERENCIAS BIBLIOGRÁFICAS

1. **Peña Ayala, Alejandro.** *Ingeniería de Software:una guía para crear Sistemas de Información.* México : s.n., 2006. 970-94797-0-9.

2. **Ingeniería en Computación y Sistamas**. *Inteligencia de Negocios* [En línea] (2010). [Citado el: 12 de noviembre de 2011]

Disponible en: <http://ingenieraensistema.blogspot.com/2010/06/inteligencia-de-negocios.html>

3. **Silva Hernández, Ing Iliana.** *Generador Automático de Reportes Dinámicos.* Tesis Ciudad México. Departamento de Computación, Centro de Investigación y de estudios avanzados del IPN, 2003.

4. **Universidad de las Ciencias Informáticas.** *Manual de Usuarios v1.7 del Generador Dinámico de Reportes.*

5. **Lafaurie Olivares, José Rolando.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2008.

6. **Curto Díaz, Josep.** *Comparativa herramientas reporting open source.* 2007.

7. **Valle Laborde, Mabel.** *Generador Dinámico de Reportes v2.0: Análisis del Módulo Diseñador de Modelos y Diseñador de Reportes.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2011.

8. **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software v2.0 del Generador Dinámico de Reportes*.

9. **Hernández Hernández, Yasmany.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2009.

10. **Eguíluz Pérez, Javier.** *Introducción a Java Script*.

Disponible en: <http://www.htmlpoint.com/javascript/corso/index.html>

11. **Álvarez, Miguel Ángel.** *Desarrollo Web.* [En línea] 2009. [Citado el: 28 de noviembre de 2011]. Disponible en: <http://www.desarrolloweb.com/articulos/modelo-entidad-relacion.html>

12. **json.org.** *json.org* [En línea] [Citado el: 2 de diciembre de 2011].

Disponible en: http://www.json.org/json-es.html

13. **O'Really** *XML.com.* [En línea] 2010. [Citado el: 2 de dicimbre de 2011].

Disponible en: < http://www.xml.com/ >

14. **netbeans.org.** *netbeans.org.* [En línea] [Citado el: 2 de diciembre de 2011].

Disponible en: <www.netbeans.org>

15. **pgadmin.org.** *pgadmin.org*. [En línea] [Citado el: 3 de diciembre de 2011].

Disponible en: <http://www.pgadmin.org/index.php>

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

17. **Conran, Aaron.** Sencha. [En línea] 3 de mayo de 2009. [Citado el: 2 de diciembre de 2011]. Disponible en: <http://www.sencha.com>

18. **Lobo, Armando Robert.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2009.

19. **Reynoso, Carlos. Kiccillof, Nicolas.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Versión 1.0*. [Citado el: 15 de diciembre de 2011].
Disponible en: <http://www.willydev.net/descargas/prev/Estiloypatron.pdf>

20. **Veloso Hernández, Pedro.** *Uso de patrones de arquitectura.*

21. **G. Figueroa, Roberth, J. SOLÍS, Camilo y A. Cabrera, Armando**. *Metodologías tradicionales vs. Metodologías ágiles.* S.l.: Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación, 2008.

22. **Yanes León, Vania Elena.** *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2011.

BIBRIOGRAFÍA

1. **Veloso Hernández, Pedro.** *Uso de patrones de arquitectura.*
2. **Hernández Hernández, Yasmany.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos. La Habana* : s.n., 2009.
3. **Hernández Hernández, Yasmany.** *Pautas de arquitectura para el desarrollo de la versión 2.0 del GDR*.
4. **Grupo de desarrollo del generador dinámico de reportes.** *Pautas para el desarrollo basado en componentes para la capa de presentación.*
5. **Grupo de desarrollo del generador dinámico de reportes.** *Pautas para la Internalización del Generador Dinámico de Reportes.*
6. **Corp, IBM.** *Rational Software Architect.* 2006.
7. **Universidad de las Ciencias Informáticas.** *Especificación de Requisitos de Software v2.0 del Generador Dinámico de Reportes.*
8. **Universidad de las Ciencias Informáticas.** Tesis.uci. [En línea] 2010. http://tesis.uci.cu.
9. **Lafaurie Olivares, José Rolando.** *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo Reportador. Ciudad de la Habana* : s.n., 2008.
10. **Valle Laborde, Mabel.** *Generador Dinámico de Reportes v2.0: Análisis del Módulo Diseñador de Modelos y Diseñador de Reportes.* La Habana : s.n., 2011.
11. **Serrano Yadrían, Gazquez Orelvis.** *Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes*. La Habana : s.n., 2011.
12. **López Quintana, Abelardo**. *Diseño e implementación del módulo de Gestión de indicadores del proyecto del Ministerio del Poder Popular de la Comunicación y la Información.* La Habana : s.n., 2010.
13. **Curto Díaz, Josep.** *Comparativa herramientas reporting open source.* 2007.
14. **Gamma Erich, Helm Richard, Johnson Ralph, Vlissides, John.** *Design Patterns (Elements of Reusable Object-Oriented Software).*
15. **Peña Ayala, Alejandro.** *Ingeniería de Software:una guía para crear Sistemas de Información.* México : s.n., 2006. 970-94797-0-9.
16. **Ingeniería en Computación y Sistemas**. *Inteligencia de Negocios* [En línea] (2010). [Citado el: 12 de noviembre de 2011]
    Disponible en: <http://ingenieraensistema.blogspot.com/2010/06/inteligencia-de-negocios.html>

17. **Silva Hernández, Ing Iliana.** *Generador Automático de Reportes Dinámicos.* Tesis Ciudad México. Departamento de Computación, Centro de Investigación y de estudios avanzados del IPN, 2003.
18. **Universidad de las Ciencias Informáticas.** *Manual de Usuarios v1.7 del Generador Dinámico de Reportes.*
19. **Hernández Hernández, Yasmany.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2009.
20. **Eguíluz Pérez, Javier.** *Introducción a Java Script*.
Disponible en: <http://www.htmlpoint.com/javascript/corso/index.html>
21. **Álvarez, Miguel Ángel.** *Desarrollo Web.* [En línea] 2009. [Citado el: 28 de noviembre de 2011].
Disponible en: <http://www.desarrolloweb.com/articulos/modelo-entidad-relacion.html>
22. **json.org.** *json.org* [En línea] [Citado el: 2 de diciembre de 2011].
Disponible en: <http://www.json.org/json-es.html>
23. **O'Really** *XML.com.* [En línea] 2010. [Citado el: 2 de dicimbre de 2011].
Disponible en: < http://www.xml.com/ >
24. **netbeans.org.** *netbeans.org.* [En línea] [Citado el: 2 de diciembre de 2011].
Disponible en: <www.netbeans.org>
25. **pgadmin.org.** *pgadmin.org*. [En línea] [Citado el: 3 de diciembre de 2011].
Disponible en: <http://www.pgadmin.org/index.php>
26. **Conran, Aaron.** Sencha. [En línea] 3 de mayo de 2009. [Citado el: 2 de diciembre de 2011].
Disponible en: <http://www.sencha.com>
27. **Lobo, Armando Robert.** *Arquitectura de Software para el Sistema Integrado de Gestión Estadística 2.0 Nuragas.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2009.
28. **Reynoso, Carlos. Kiccillof, Nicolas.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. Versión 1.0*. [Citado el: 15 de diciembre de 2011].
Disponible en: <http://www.willydev.net/descargas/prev/Estiloypatron.pdf>
29. **G. Figueroa, Roberth, J. Solís, Camilo y A. Cabrera, Armando**. *Metodologías tradicionales vs. Metodologías ágiles.* S.l.: Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación, 2008.
30. **Yanes León, Vania Elena.** *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0.* Tesis La Habana. Universidad de las Ciencias Informáticas, 2011.

ANEXOS

Anexo 1: Diagramas de secuencias del caso de uso Administrar Modelo

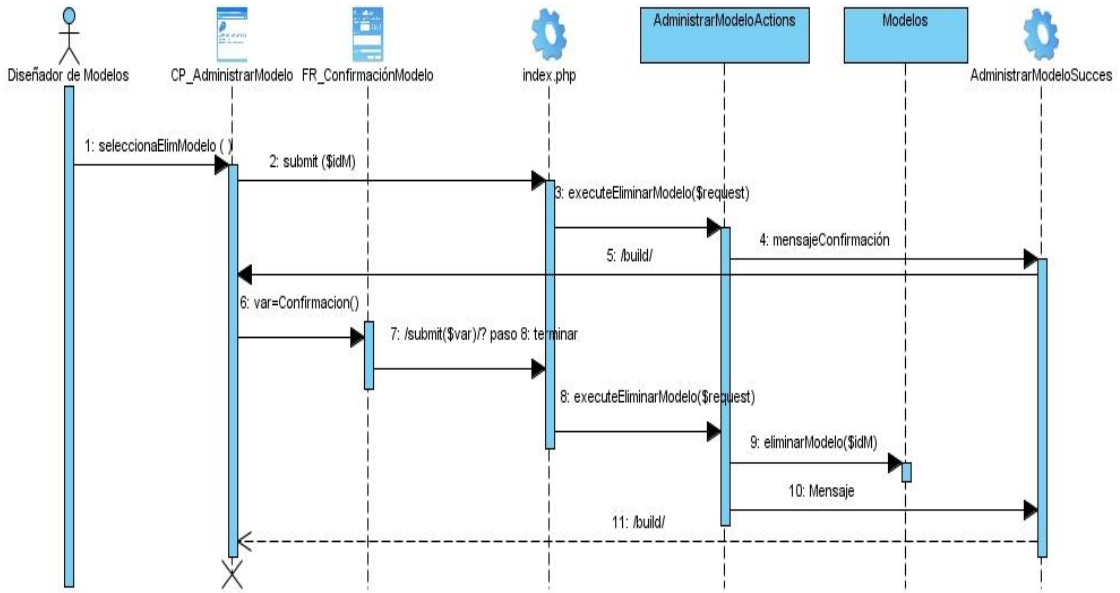

Figura 14. Escenario Eliminar Modelo

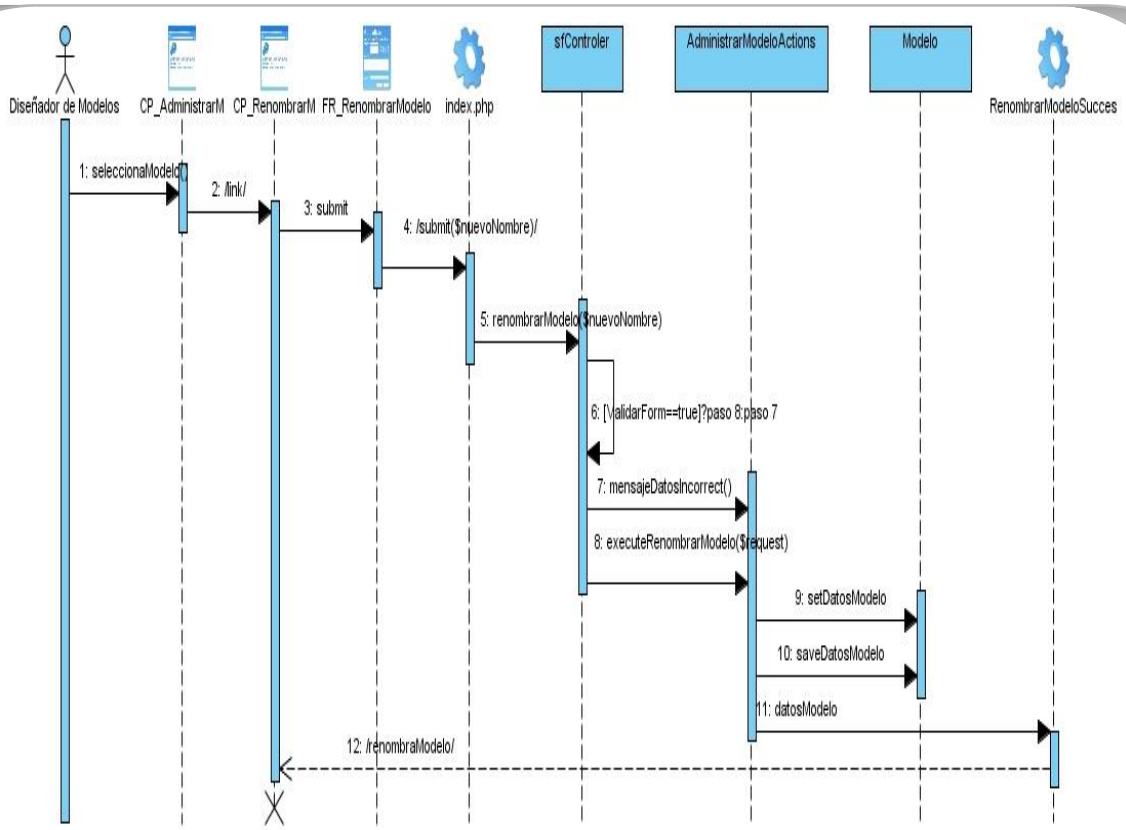

Figura 15. Escenario Renombrar Modelo

Anexo 2: Matriz de trazabilidad

RF vs CU	Diseñar modelo	Administrar modelo	Gestionar origen de datos
Visualizar OD			X
Seleccionar OD			X
Mostrar entidades OD			X
Seleccionar objetos del OD			X
Modificar OD			X
Adicionar OD			X
Eliminar OD			X
Obtener entidades relacionadas	X		
Renombrar MS		X	
Probar conexión a OD	X		
Visualizar MS	X		
Renombrar las entidades y atributos de un OD	X		
Mostrar objetos y entidades del MS	X		
Seleccionar entidades del MS	X		
Mostrar el OD desde el cual fue diseñado el MS		X	
Crear MS	X		
Eliminar MS		X	
Modificar MS		X	
Buscar MS		X	

Tabla 10. Matriz de trazabilidad de requisitos funcionales y casos de uso

Anexo 3: Diagramas de componentes

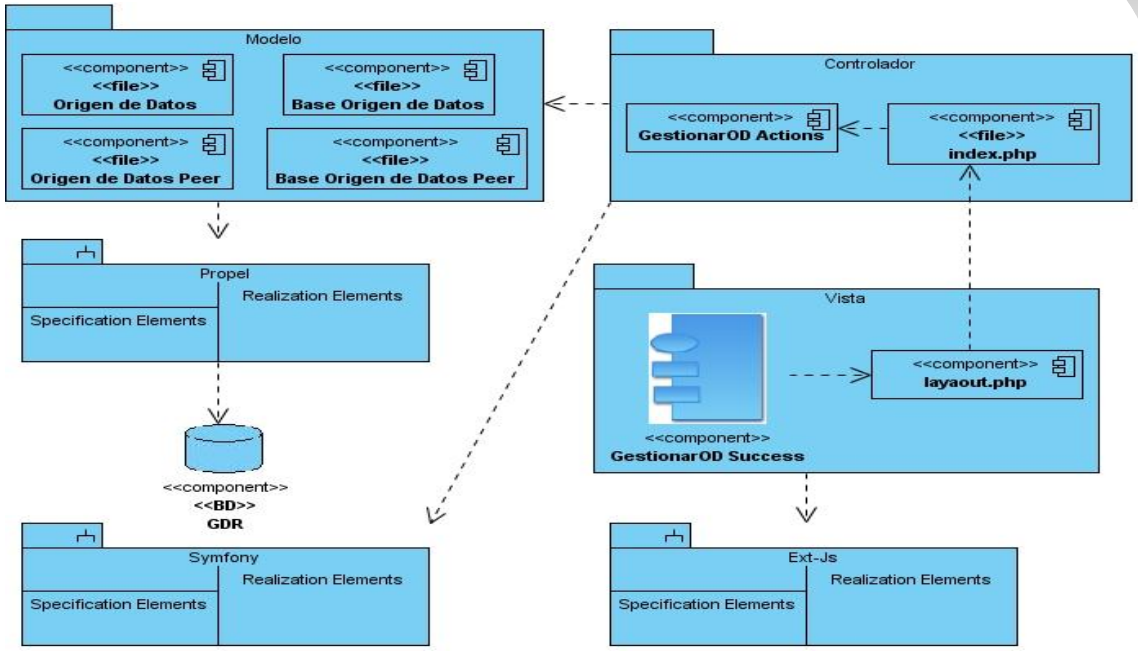

Figura 16. Diagrama de componentes para el caso de uso Gestionar Origen de Datos

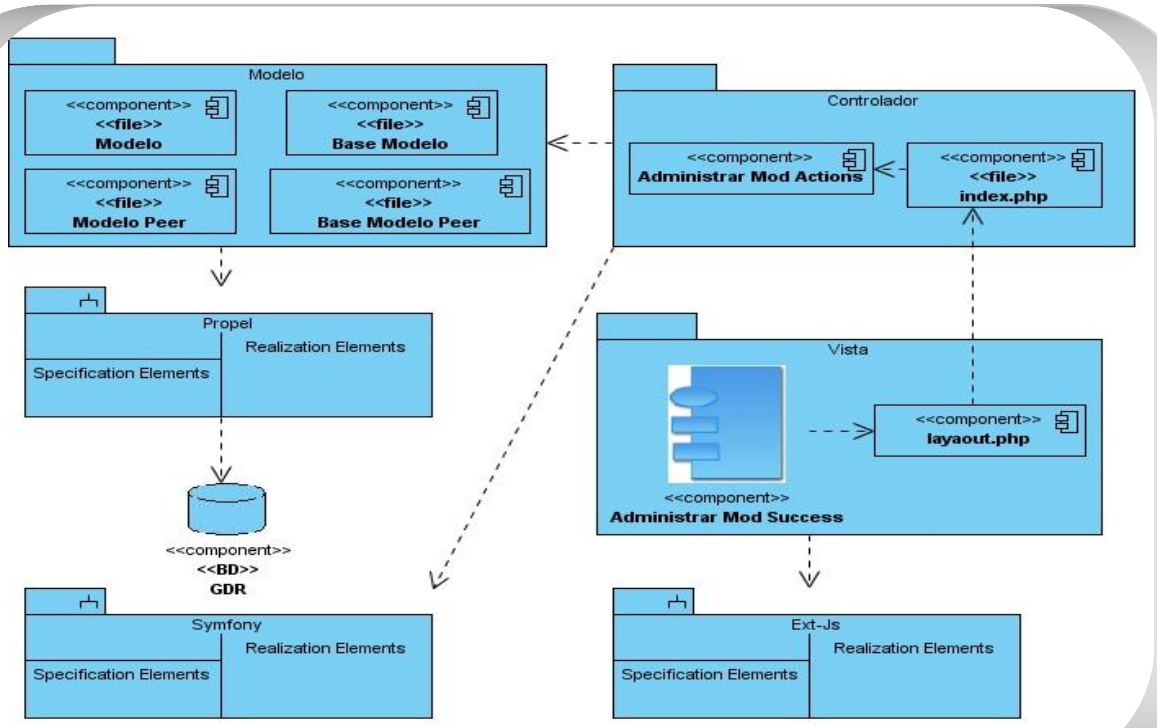

Figura 17. Diagrama de componentes para el caso de uso Administrar Modelo

Anexo 4: Interfaces del Diseñador de Modelos v2.0

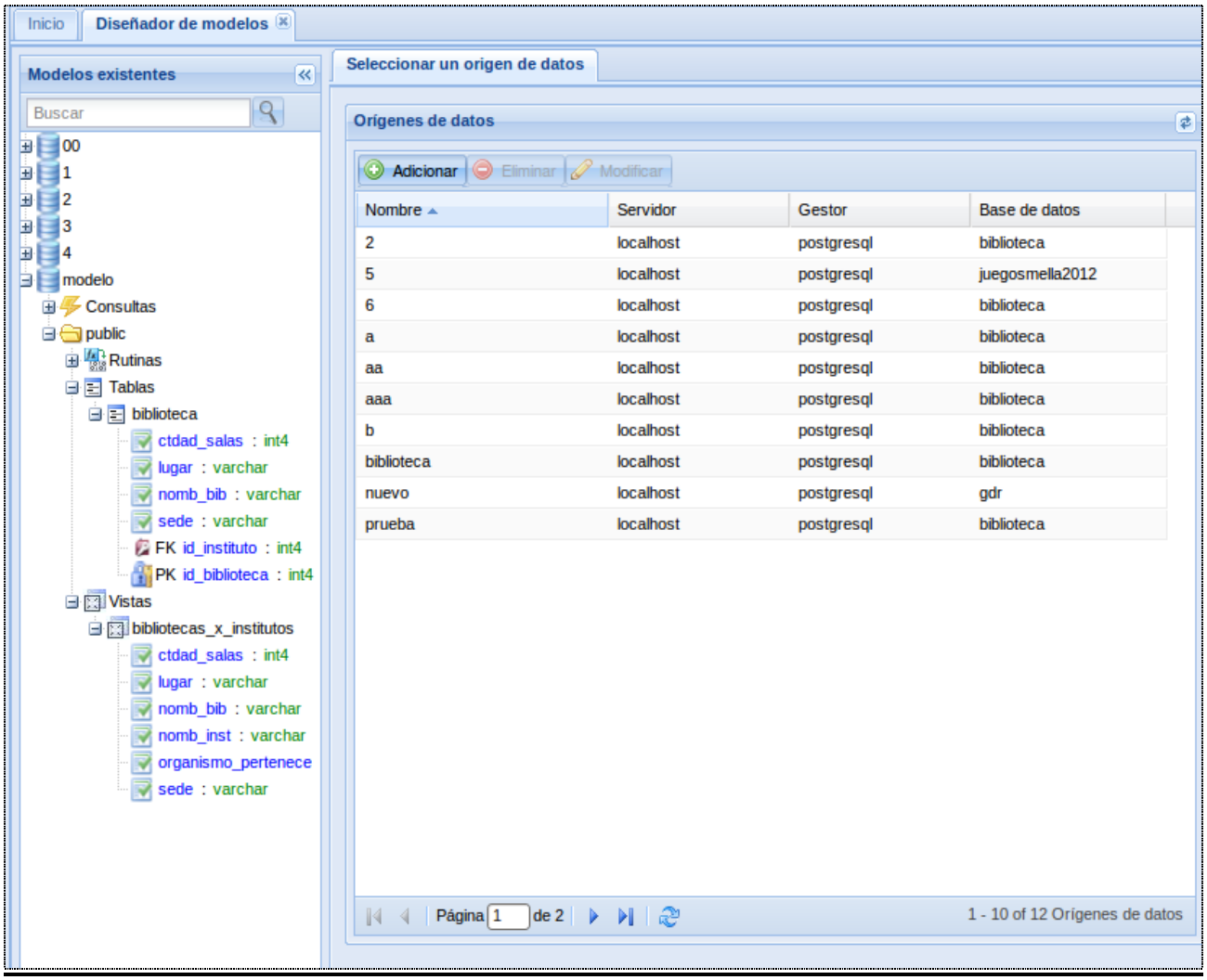

Figura 18. Nivel de atributos en las entidades de los modelos y paginador para mostrar los orígenes de datos

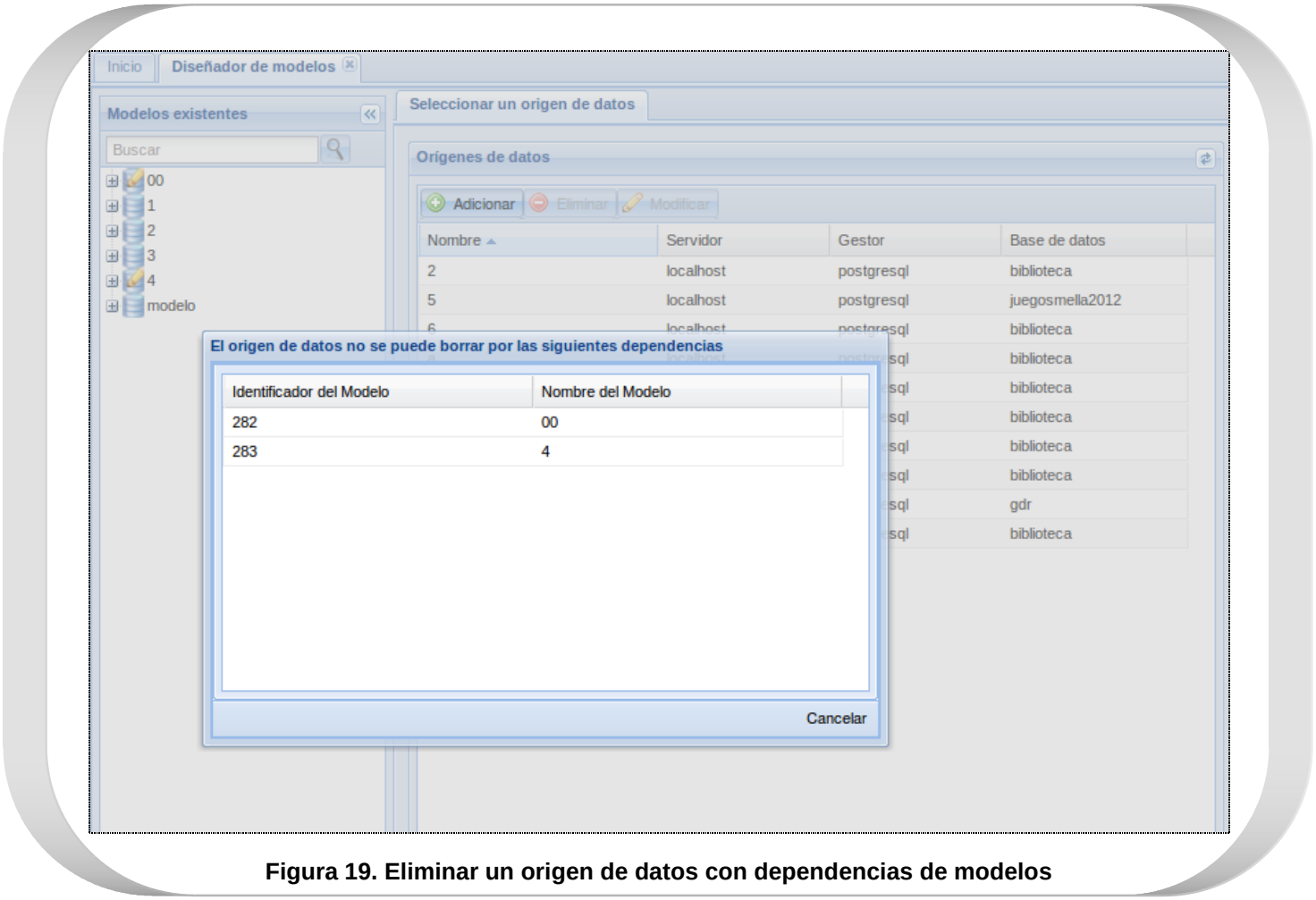

Figura 19. Eliminar un origen de datos con dependencias de modelos

Figura 20. Diseño de modelos con varias entidades

GLOSARIO DE TÉRMINOS

API: (*Application Programming Interface* - Interfaz de Programación de Aplicaciones) es un conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.

Entidad: Es la representación de un objeto o concepto del mundo real que se describe en una base de datos.

Frameworks: Marco de trabajo.

ORM: Object Relational Mapper es una técnica de programación que permite generar clases a través de las tablas de la base de datos.

Reporte: Documento caracterizado por contener información u otra materia reflejando el resultado de una investigación adaptado al contexto de una situación.