Universidad de las Ciencias Informáticas

FACULTAD 6

Título: Componente para la exportación e importación del diseño de reportes para el Generador Dinámico de Reportes 1.8.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor: José Antonio Torreblanca Alvarez.

Tutor(es):

MsC. Asnay Guirola González.

Ing. Aurelio Rodríguez Durán.

Ing. Andry Daniel Díaz León.

La Habana, Junio 2014
"Año 56 de la Revolución"

Declaración de autoría

Declaro ser el único autor de este trabajo y cedo 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 2014.

_____
Firma del autor

José Antonio Torreblanca Alvarez

_____	_____	_____
Firma del tutor	Firma del tutor	Firma del tutor
MsC. Asnay Guirola González	Ing. Aurelio Rodríguez Durán	Ing. Andy Daniel Díaz León

Datos de contacto

Autor:

José Antonio Torreblanca Alvarez.

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

Tutores:

MsC. Asnay Guirola González

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

E-mail: aguirola@uci.cu

Ing. Aurelio Rodríguez Durán

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

E-mail: arduran@uci.cu

Ing. Andry Daniel Díaz León

Universidad de las Ciencias Informáticas.

La Habana, Cuba.

E-mail: addiaz@uci.cu

Agradecimientos

Agradecer a todas las personas que aportaron lo mínimo para mi formación tanto personal como profesional.

A mis tutores, Asnay, Aurelio y Andry por todo su apoyo y conocimiento brindado en el desarrollo de este sueño profesional.

A Yoanni por las tantas horas dedicadas de su tiempo, cuyo esfuerzo lo hace dueño también del resultado obtenido.

A mis padres, Rosa y Antonio por delegarme toda la confianza y brindarme su apoyo en cada paso que doy, por ser los maestros de mi vida y el tesoro más grande que llevo siempre conmigo.

A mi hermano Reynier por quererme y apoyarme tanto.

A mi tía Raquel por darme mucho más que todo su corazón.

A mis amigos Leodán, Yuneiry, Rolando, Manu y Roxana por permitirme formar parte de sus vidas.

Agradecer a mis amigos Ana Belkis y Alberto, no sabría cómo agradecer cada minuto de apoyo, tantas cosas que le debo a nuestra amistad que no caben palabras para describirlo, ustedes han sido lo máximo, en las buenas y en las malas siempre han estado ahí y espero que continúe así por el resto de nuestras vidas.

A todos…Mil Gracias

Dedicatoria

A mis padres, Rosa y Antonio, para ustedes es este logro.

A mi familia, a mis amigos.

Resumen

La Oficina Nacional de Estadística e Información (ONEI) utiliza el Sistema Generador Dinámico de Reportes (GDR) para las generar los reportes del proceso de producción y difusión de estadísticas. El presente trabajo tiene como objetivo desarrollar un componente que permita la exportación e importación del diseño de reportes para el sistema Generador Dinámico de Reportes V (1.8), permitiendo el intercambio del diseño de los mismos. Para desarrollar el componente se utilizó la metodología OpenUP, el *framework* Symfony para la aplicación del lado del servidor y para el lado del cliente el *framework* ExtJS. Para el modelado de la solución se utilizó Visual Paradigm y como entorno integrado de desarrollo NetBeans. Se realizaron pruebas unitarias y funcionales para comprobar el correcto funcionamiento del sistema, obteniéndose como resultado el componente para la exportación e importación del diseño de reportes con las nuevas funcionalidades incluidas, las cuales son de gran importancia para el GDR.

Palabras claves: componente, diseño, exportación, importación, reporte.

Abstract

The National Office of Statistics and Information (ONEI) use Dynamic Reporting System (GDR) to generate reports of the process of production and diffusion of statistics. The objective of this diploma paper is to develop a component that allows the export and import of design reports for GDR V (1.8), allowing the exchange the same design. To develop the component is used OpenUp as software methodology, the framework Symfony for implementing the server-side and for the client-side the framework ExtJs. For modeling the solution is used Visual Paradigm, and NetBeans as integrated development environment. Unit and functional tests were performed to check the correct operation of the system, obtaining as a result the component to export and import the report design with the new included features, which ones are very important to the GDR.

Keywords: *component, design, exportation, importation, report.*

Índice

Índice de figuras

Índice de tablas

Introducción.

La información es un elemento vital para cualquier empresa, influye de manera directa en la forma en que estas operan. En la actualidad estas empresas necesitan recolectar información y almacenarla para evitar la pérdida o deterioro de la misma. A raíz de ello se hace necesaria una alternativa eficiente que les proporcione información útil y una opción efectiva es en forma de reportes, así es posible conservar y mostrar los análisis y resultados de las empresas pudiendo servir además para la ayuda a la toma de decisiones.

En Cuba, con el objetivo de alcanzar una mejora en la industria productora de *software* y convertirla en una fuente de ingresos para la economía cubana, surge la Universidad de Ciencias Informáticas (UCI). Su principal objetivo: formar profesionales capaces de producir *software* y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación.

Esta casa de altos estudios cuenta con varios centros de desarrollo, entre ellos se encuentra el Centro de Tecnologías y Gestión de Datos (DATEC). Entre los departamentos que lo componen está el departamento de Integración de Soluciones cuya función principal consiste en el desarrollo de las herramientas necesarias para crear soluciones integrales de análisis de datos. El Generador Dinámico de Reportes (GDR) en su versión 1.8 es uno de los sistemas desarrollados en dicho departamento, el mismo tiene como propósito garantizar la generación de reportes partiendo de datos persistentes en algún origen de datos soportado por el sistema (PostgreSQL, MySQL Server, SQLlite, Microsoft SQL Server, Oracle).

El GDR es utilizado por una amplia gama de entidades de la administración pública de nuestro país, entre ellas la Oficina Nacional de Estadísticas e Información (ONEI), encargada de garantizar la producción y difusión de estadísticas, en dependencia de las necesidades del país. La ONEI utiliza para la gestión de sus reportes el Generador Dinámico de Reportes que se encuentra instalado en cada una de sus oficinas.

La distribución física del GDR para la ONEI implica tener aplicaciones instaladas por todas las oficinas del país. La infraestructura de red presente entre estas oficinas, no permite tener las aplicaciones clientes conectadas entre sí, y por tanto sus respectivas bases de datos tampoco lo están.

En ocasiones los diseños de los reportes, o parte de estos, pueden resultar comunes en varias oficinas

debido a la semejanza del modelo de datos que se desea procesar y la forma en que se presenta la misma. Se hace necesario, a partir de la reutilización de los diseños ya existentes, el intercambio de los mismos en aras de ahorrar tiempo y recursos en la elaboración de nuevos reportes.

Ante estas nuevas necesidades de información los especialistas de la ONEI realizan el diseño de nuevos reportes desde alguna de las aplicaciones instaladas. El intercambio de estos nuevos diseños de una aplicación a otra se realiza manualmente por especialistas con conocimientos en sistemas de base de datos.

Para realizar esta operación un especialista necesita hacer un resguardo de la información deseada del diseño de los reportes y que esta sea almacenada en un formato adecuado para ser enviada hasta donde se necesita. Para ello utiliza el PgAdminIII o alguna otra herramienta para administrar bases de datos. Luego, por el mismo u otro especialista, se procesa esa información a través de técnicas de inserción en bases de datos, dicho proceso se realiza manualmente, verificando e introduciendo cada uno de los campos presentes en las tablas que contienen la información del diseño de los reportes, de esta forma la base de datos destino recibe los nuevos diseños que se necesitan utilizar.

El uso de intermediarios provoca en primer lugar una deficiente asignación de recursos humanos a realizar tareas monótonas y repetitivas: se requiere un consumo de una mayor cantidad de tiempo y personal que implica además que la necesidad de trasladar el diseño de esos nuevos reportes se realice de forma personal por alguno de los especialistas. Además provoca duplicidades o incongruencias en la información recibida: al realizarse de forma manual aumenta el margen de error en la ejecución de la misma, quedando vulnerable la integridad de los reportes a la hora de introducir los nuevos diseños de reportes en las bases de datos destino.

A partir de lo anteriormente expuesto el presente trabajo de diploma identifica como **Problema de la investigación:** ¿Cómo contribuir a la mejora del intercambio del diseño de reportes entre instancias de aplicaciones de GDR sin afectar la integridad de los mismos? estableciendo como **Objeto de Estudio:** Intercambio de datos entre aplicaciones informáticas, delimitado por el **Campo de acción:** Exportación e importación del diseño de reportes del Generador Dinámico de Reportes.

Para resolver el problema planteado se define como **Objetivo general:** Desarrollar un componente que

contribuya a mejorar la exportación e importación del diseño de reportes entre instancias de aplicaciones de GDR sin afectar la integridad de los mismos.

Preguntas de Investigación:

1. ¿Cuáles son los fundamentos teóricos para la exportación e importación del diseño de reportes?
2. ¿Qué elementos caracterizan intercambio del diseño de reportes del GDR?
3. ¿Cómo realizar una representación efectiva del software que debe ser producido para guiar el proceso de exportación e importación del diseño de reportes?
4. ¿El desarrollo del componente para la exportación e importación del diseño de reportes contribuye a la mejora del intercambio del diseño de reportes entre instancias de aplicaciones de GDR?

Tareas de Investigación:

1. Revisión bibliográfica relacionada con la exportación e importación de reportes para tener mayor comprensión del componente a desarrollar.
2. Caracterización de las herramientas, tecnologías y metodologías a utilizar para guiar el proceso de desarrollo del componente.
3. Obtención y especificación de requisitos funcionales y no funcionales para definir las necesidades del cliente.
4. Descripción de los casos de uso del sistema para determinar cada una de las funcionalidades del componente.
5. Descripción del flujo de datos durante los procesos de exportación e importación del diseño de reportes para identificar las pautas de estos procesos.
6. Realización del diseño del componente para brindar una panorámica de cómo el usuario interactúa con el mismo.
7. Implementación del componente que permita la exportación e importación del diseño de reportes.
8. Diseño de las pruebas del componente con el objetivo de establecer el conjunto de herramientas, técnicas y métodos para probar el correcto funcionamiento del mismo.
9. Ejecución de las pruebas del componente para validar que el sistema cumpla las funcionalidades requeridas.

Métodos científicos de investigación.

Analítico - sintético: Permite la realización del análisis del proceso de exportación e importación del diseño

de reportes llevado a cabo en las oficinas de la ONEI y la división de dicho proceso en sus múltiples relaciones y componentes, para así poder entender mejor sus características y funciones. Permite además descubrir sus características generales y las relaciones entre dichos procesos, a partir del análisis previamente hecho.

Inductivo - deductivo: A partir del estudio de diferentes sistemas generadores de reportes y su proceso de exportación e importación de reportes, es posible arribar al modo en que se realiza dicho proceso en el presente trabajo de diploma para dar solución al presente problema de investigación. Se aplica en el análisis de la información obtenida, pues a partir de la adaptación y generalización de los contenidos se determina la forma en que se realizara este proceso para la situación problemática presente.

Modelación: Permite crear abstracciones con el objetivo de explicar la realidad. La necesidad práctica para la cual se ejecuta la modelación y la posible solución del problema de investigación da la medida en que se logra la relación entre el modelo y el objeto que se modela. Dicha relación es determinada por el sujeto (el investigador) escogiendo una alternativa de acuerdo con su criterio. Este método se puede observar durante el análisis y diseño de la solución, a través de la modelación de los procesos exportar e importar el diseño de reportes, donde es posible predecir la respuesta de dicho proceso a variaciones de algunos de sus parámetros sin tener que ejecutar el proceso en la realidad.

Técnica de recogida de datos

Tormenta de ideas: Esta técnica se utiliza para designar un pensamiento creativo, generativo y libre. Consiste en la generación de nuevas ideas en reuniones de grupos de trabajo o de manera independiente. Es una técnica altamente productiva tanto en el esfuerzo individual como en el trabajo de grupo. En el desarrollo de esta técnica intervienen el investigador y otros especialistas relacionados en el ámbito donde ocurre la situación problémica. Para dar cumplimiento al objetivo de este método los miembros involucrados aportan, durante un tiempo previamente establecido el mayor número de ideas sobre el problema planteado.

Estructura del trabajo de diploma:

Capítulo 1: Fundamentos teóricos sobre el proceso de exportación e importación del diseño de reportes.

Este capítulo contiene los fundamentos teóricos para entender el problema sobre la exportación e importación del diseño de reportes y los conceptos fundamentales. Se realiza un análisis y selección de las tecnologías, las herramientas y metodologías a utilizar para desarrollar el sistema.

Capítulo 2: Análisis y Diseño del componente para la exportación e importación del diseño de reportes.

En este capítulo se realiza el modelo de clases del diseño del sistema, aplicando buenas prácticas de patrones de diseño y la reutilización de componentes. Se presenta además el diseño del componente a través de la elaboración de los artefactos de ingeniería de *software* como resultado de las disciplinas análisis y diseño requeridos para la elaboración y construcción del mismo.

Capítulo 3: Implementación y pruebas del componente para la exportación e importación del diseño de reportes.

En este capítulo se hace referencia a la implementación de la solución, abordando cómo se realiza el proceso de exportación e importación del diseño de reportes. De igual manera, se explica el proceso de validación del sistema a través las pruebas realizadas al componente.

Capítulo 1: Fundamentos teóricos sobre el proceso de exportación e importación del diseño de reportes.

Introducción.

En este capítulo se describen los principales aspectos teóricos relacionados con la exportación e importación del diseño de reportes a partir de los conceptos de reporte, diseño y componente. Además se describen las herramientas, metodología, lenguajes de programación y marcos de trabajo seleccionados para el desarrollo de la solución.

1.1 Herramientas generadoras de reportes.

Un reporte es una noticia o informe que brinda información con algún propósito. Combina tres tipos de información:

- Datos y su estructura.
- Diseño o información de formato que describe cómo serán presentados los datos.
- Propiedades del reporte, como autor, parámetros, imágenes dentro del reporte. (*González, 2010*)

Este tipo de documento puede ser impreso, digital o audiovisual, pretende transmitir una información, aunque puede tener diversos objetivos. En el ámbito de la informática los reportes son informes que organizan y muestran la información contenida en una base de datos. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados.

Los reportes facilitan a los usuarios observar el desarrollo del negocio y que de esta forma logren identificar nuevas oportunidades de negocio o servicios. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios. Según la herramienta informática y la base de datos en cuestión, los reportes permiten la creación de etiquetas y la elaboración de facturas, entre otras tareas. (*Definicion.de, 2013*)

Los generadores de reportes son herramientas complementarias de los sistemas de información capaces de realizar consultas a las bases de datos obteniendo así un mejor dominio en el diseño de visualización de los resultados obtenidos (*Infante, 2009*). Además de los beneficios de disponer de información válida para tomar decisiones, otra de las ventajas de los sistemas de reporte es la automatización de la

generación de reportes. Los generadores de reportes están compuestos principalmente por dos elementos básicos, un diseñador o editor de informes y un motor de generación. El diseñador visual es un programa que ayuda a los usuarios y desarrolladores para diseñar reportes visualmente. A través de una interfaz simple de usar provee las funciones más importantes para crear reportes en poco tiempo.

Una vez concluida la etapa de diseño, ocurre la generación del reporte a través de un motor de reportes, completándolo con el contenido directamente de la base de datos y elaborando el reporte final con la apariencia previamente diseñada.

1.2 Diseño de reportes

El diseño de reportes representa una plantilla que va a ser utilizada por un motor para armar el reporte de acuerdo con su estructura, y completándolo con la información obtenida desde la base de datos. De manera que el usuario puede agregar componentes y demás características, en las distintas secciones, que se irán almacenando en la plantilla para en el momento de la generación del reporte el mismo presente el diseño personalizado que se desea y además con los datos dispuestos en su interior.

Las plantillas generalmente contienen las siguientes secciones o bandas:

Tabla 1: Secciones de la plantilla de un reporte

title	Título del reporte
pageHeader	Encabezado del documento
columnHeader	Encabezado de las columnas
detail	Detalle del documento. Cuerpo
columnFooter	Pie de columna
pageFooter	Pie del documento
summary	Resumen. Cierre del documento

El diseño de los reportes comprende además las definiciones de las fuentes de datos con el cual se construyen los mismos (tablas, vistas, funciones), su estructura y la forma en que se presentan los mismos así como otras propiedades del reporte como autor, parámetros o imágenes.

1.3 Diseñadores de reportes

Existen diferentes herramientas diseñadoras de reportes que presentan la funcionalidad de exportar los reportes diseñados en diferentes formatos como por ejemplo HTML[1], PDF[2], CSV[3], XLS[4] y XML[5]. Este último se utiliza además como fuente que contiene información necesaria para importar reportes previamente generados por alguna de estas herramientas. Partiendo de la estructura de los formatos de salida de este proceso y la validación de la misma, se realiza una comparación a partir de las características de algunas de ellas para la selección del propio formato de salida de la presente solución.

1.3.1 iReport

iReport es una herramienta visual de diseño de informes para JasperReports[6]. Es un poderoso sistema informático para generar reportes 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. (*iReport, 2013*)

Las plantillas XML en JasperReports son archivos XML estándar, que tienen una extensión .jrxml. Todos los archivos .jrxml contienen una etiqueta <jasperReport> como elemento raíz. Este a su vez, contiene muchos subelementos, todos ellos son opcionales:

- <*queryString*>: Usualmente contiene la sentencia SQL para la construcción del reporte.
- <*fieldname*>: Este elemento se utiliza para asignar los datos de fuentes de datos o consultas dentro de la plantilla del reporte.
- <*fieldDescription*>: Este elemento asigna el nombre del campo con el elemento correspondiente dentro del archivo XML.
- <*staticText*>: En este elemento se define el texto estático que no depende de ninguna expresión,

[1]HTML:(«lenguaje de marcas de hipertexto»), hace referencia al lenguaje de marcado para la elaboración de páginas web.
[2]PDF: (sigla del inglés *portable document format*, formato de documento portátil) es un formato de almacenamiento de documentos digitales independiente de plataformas de *software* o *hardware*.
[3]CSV (del inglés *comma-separated values*) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas.
[4]XLS: formato de ***Microsoft Excel*** es una aplicación distribuida por Microsoft Office para hojas de cálculo.
[5]XML: siglas en inglés de ***eXtensible Markup Language*** ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible.
[6]JasperReport: herramienta libre de Java para la creación de informes que facilita y agiliza la generación, visualización e impresión de los reportes.

variable o parámetro del reporte.

- <*textFieldExpression*>: Este elemento define el aspecto del campo resultado.
- <*$F{country}*>: Esta es una variable que contiene el valor del campo predefinido resultado en la etiqueta <*fieldname*>.
- <band>: Secciones que contienen los datos que se muestran en el reporte. (*iReport, 2013)*

Para la importación de dicho reporte iReport presenta un asistente que permite localizar el archivo JRXML que almacena estos datos. Para procesar un archivo XML y extraer información de ella, JasperReports utiliza XPath, que es un lenguaje de consulta popular para filtrar los datos y permite construir expresiones que recorren y procesan un documento XML.

1.3.2 Oracle Reports

Oracle Reports es un componente de Oracle Fusion Middleware, es una herramienta de generación de reportes empresariales de alta fidelidad. Contiene una poderosa herramienta de declarativa de reportes de la forma WYSIWYG[7] y está basada en Oracle Reports Server con la arquitectura de varios niveles para acceder a cualquier fuente de datos, genera informes en cualquier formato para la web y el papel (HTML, RTF, PDF, XML, XLS). (*Oracle, 2013*)

El archivo de salida XML para Oracle Reports presenta la siguiente estructura.

- <*report*>: Contiene el nombre del reporte y otros parámetros como la versión del DTD[8] del archivo XML.
- <*layout*>: Este elemento define la plantilla del reporte, el resto de las secciones estarán contenida dentro de esta.
- <*section*>: Esta etiqueta define las secciones o bandas del reporte, contiene los campos a mostrar.
- <*field*>: Contiene los campos a mostrar en el reporte, su fuente de datos y parámetros como el color, formato.
- <*customize*>: Este elemento contiene los elementos repetitivos y lógicos del reporte. (*OracleAS,*

[7] **WYSIWYG** es el acrónimo de ***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et* (en español, "lo que ves es lo que obtienes"). Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final

[8] **DTD** (siglas en inglés de *document type definition*) definición de tipo de documento es una descripción de estructura y sintaxis de un documento XML.

2013)

En Oracle Reports la funcionalidad de importar un reporte se realiza mediante un asistente. El archivo XML incluye un valor *hash*[9] que se utiliza cuando el archivo se importa para determinar si el archivo se ha modificado externamente. Si el archivo ha sido modificado, no está disponible para la importación, y se muestra un mensaje de error donde se indica que el archivo no es válido. (*Communities, 2013*)

1.3.3 Eclipse BIRT

Eclipse BIRT acrónimo de (*Business Intelligence and Reporting Tools*). Es un sistema utilizado para desarrollar aplicaciones web y 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[10] de manera independiente. (*BIRT, 2013*)

El contenido del archivo de salida de un reporte en formato XML muestra las principales secciones:

- <*report*>: Esta sección contiene el nombre del reporte y otros parámetros de su configuración.
- <*image*>: En caso de que el reporte contenga una imagen se almacena en esta sección
- <*label*>: Sección que almacena las etiquetas que se muestran en el reporte.
- <*data*>: Contiene los datos que se muestran por cada una de las etiquetas. (*Developer.com, 2013*)

Para importar estos reportes Eclipse BIRT proporciona una herramienta visual para ayudar a crear una asignación de XML con datos relacionales. Para ello el usuario debe proporcionar el archivo de muestra y así la herramienta visual también proporcionará la vista previa de los datos. Para extraer esta información se utiliza el lenguaje XPath.

1.3.4 Sistema Generador Dinámico de Reportes (GDR).

El Sistema Generador Dinámico de Reportes es una herramienta que permite a sus clientes consultar las

[9] Hash: Una función hash es una función computable que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte (mapea) en un rango de salida finito, normalmente cadenas de longitud fija.

[10] **Java Platform, Enterprise Edition** o **Java EE** (anteriormente conocido como Java 2 *Platform, Enterprise Edition* o J2EE hasta la versión 1.4; traducido informalmente como **Java Empresarial**), es una plataforma de programación para desarrollar y ejecutar *software* de aplicaciones en el lenguaje de programación Java.

bases de datos de sus organizaciones y poder generar reportes con la información que estos manejan.

Está estructurado en seis módulos principales de los cuales dos evidencian el diseño de un reporte.

- Diseñador de reportes: En este módulo se realizan estos diseños personalizados de los reportes de una manera muy sencilla e intuitiva para posteriormente ser visualizados.
- Visor de reportes: El módulo permite al usuario visualizar los reportes previamente diseñados, así como exportarlos a diferentes formatos de salida tales como: HTML, PDF, EXCEL y CSV. (*Rodríguez, 2012*)

Dentro del GDR los reportes contienen las siguientes secciones o bandas:

- Encabezado del documento.
- Encabezado de página.
- Cuerpo.
- Pie del documento.
- Pie de página.

Esta información se almacena en la base de datos del sistema en un conjunto de tablas, entre ellas la que almacena los reportes. Hasta el momento el sistema no permite la exportación del diseño de reportes en formato XML, y por tanto no es posible la importación de los mismos.

1.4 Comparación y selección de las características a incluir en el GDR.

El estudio de las herramientas anteriores permitió revelar que las mismas poseen propiedades similares a la hora de exportar e importar lo reportes, permitiendo la reutilización del diseño de los mismos. Los diseñadores de reportes estudiados presentan licencias de código abierto siendo esta una de las ventajas principales. La mayoría de estas herramientas no están orientadas a la web, sino al escritorio, siendo esto una limitante excepto el GDR, el cual no presenta la funcionalidad de exportar e importar los reportes a partir de un archivo XML, principal funcionalidad a incorporar en el GDR. La estructura de los archivos XML no comprenden la definición de la fuente de datos para la construcción de reportes, siendo esto una limitante para importar los reportes ya que dichas definiciones no persistan en el sistema hacia donde se desean importar los mismos; se hace necesario la incorporación de dicha funcionalidad. La posibilidad de

exportar e importar reportes en la mayoría de estas herramientas están diseñadas para un único reporte a la vez, lo que requiere la realización de actividades repetitivas cuando se desee más de un reporte a procesar, por lo que se decide incorporar esta funcionalidad al GDR. La adaptación de las características mencionadas anteriormente y su inclusión dentro del GDR, permitirá un mejor manejo del proceso de exportación e importación del diseño de reportes para esta herramienta.

1.5 ¿Cómo realizar el intercambio del diseño de reportes en el GDR?

La forma en que persisten los reportes en el GDR hace imprescindible que ocurra un intercambio de la información entre las respectivas bases de datos de cada una de las oficinas, teniendo en cuenta la infraestructura de red presente dichas oficinas, que no permite que las bases de datos estén conectadas entre sí y por tanto no estarían disponibles durante el proceso de intercambio de información.

Ante esta situación es necesario optar por una variante que se adapte a la situación existente y para llevar a cabo este proceso se debe tener en cuenta que:

- En el momento de insertar los nuevos diseños en las bases de datos destino, se respete el criterio de unicidad que establece que no se debe violar una restricción de integridad ya sea por llave primaria o única, por ejemplo cuando durante alguna transacción se inserta algún registro, en su respectiva tabla, con el mismo valor de la clave primaria, o sea que ya exista en la base de datos destino.
- Para la creación de un reporte las posibles fuentes de datos pueden ser tablas, vistas o funciones procedurales, elementos del modelo que serán necesarios manipular en el proceso de exportación e importación del diseño de los reportes.
- Las definiciones de datos compuesto generados por las funciones procedurales utilizadas como fuentes de datos para la creación de los reportes constituye un elemento a tener en cuenta para realizar el proceso. Las bases de datos destino necesitan conocer las definiciones de los tipos de datos compuestos y que estas persistan en el sistema.
- Los reportes llevan asociados un origen y un modelo de datos, además de estar organizados por categorías. Esto implica que estos elementos deben ser tomados en cuenta para la realización del proceso.

Se hace necesario, para guiar este proceso, implementación de un componente adicional al GDR capaz

de realizar dichas funcionalidades y guiándose por los criterios antes expuestos.

Un componente es aquello que forma parte de la composición de un todo. Se trata de elementos que, a través de algún tipo de asociación o contigüidad, dan lugar a un conjunto uniforme. (*Definicion.de, 2013*) Según Szyperski[11] "*Un componente de software es una unidad de composición con interfaces especificadas contractualmente y solamente dependencias explícitas de contexto. Un componente de software puede ser desplegado independientemente y está sujeto a composición por terceros*." (*Montilva, 2003*)

Según Philippe Krutchen[12] "*Un componente de software es una parte no trivial, casi independiente y reemplazable de un sistema que cumple una función dentro del contexto de una arquitectura bien definida. Un componente cumple con un conjunto de interfaces y provee la realización física de ellas*". (*Montilva, 2003*)

Del análisis de las anteriores definiciones se deduce que un componente de *software* es: "una pieza de *software* casi independiente que ejecuta funciones cuyos servicios son conocidos mediante sus interfaces y requisitos".

Con la solución propuesta se pretende crear un componente que se integre con el GDR, facilite la exportación e importación del diseño de los reportes y respete las normas que rigen este proceso.

1.6 Metodología y herramientas para el desarrollo de software.

A continuación se hace una descripción y selección de la metodología y las herramientas de desarrollo de *software* utilizadas para la implementación del componente a desarrollar. Por investigaciones realizadas anteriormente a nivel de proyecto ya se encontraban definidas las tecnologías que formarán parte del ambiente de desarrollo a utilizar para dar solución al problema en cuanto a metodología, herramientas, lenguajes y *framework*. La selección de las mismas sigue los principios del uso de herramientas libres y de código abierto orientadas en la arquitectura diseñada por el equipo de desarrollo de *software* del GDR.

[11]Arquitecto de *software* en la división de sistemas conectados y afiliado con Microsoft Research.
[12] Ingeniero y profesor de ingeniería de *software* en la Universidad de British Columbia en Vancouver, Canadá. Director de desarrollo de proceso (RUP) en *Rational Software*.

1.6.1 Metodología de desarrollo de software.

Una metodología de desarrollo de *software* es un marco de trabajo que muestra las tareas que se requieren para construir un *software* de alta calidad, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener. Guía a los desarrolladores en la realización de las actividades, durante todo el proceso de creación de un producto. Las metodologías se pueden clasificar en robustas que son usadas en grandes proyectos y las ágiles se utilizan para proyectos de corto plazo de menor tamaño. (*Jacobson, 2000*)

Proceso Unificado Abierto (OpenUP).

OpenUP es una metodología ágil dirigida a la gestión y desarrollo de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos. Es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. 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 mínimo, completo y extensible, pues sólo incluye el contenido del proceso fundamental, puede ser manifestado como proceso entero para construir un sistema y puede ser utilizado como base para agregar o para adaptar más procesos.

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

- Concepción: se determina 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.

OpenUp presenta las siguientes características:

- Es extensible pues en el proceso se puede adaptar según la necesidad de los sistemas.
- OpenUp es un proceso ágil.
- Es ligero y proporciona una comprensión detallada del proyecto, beneficiando a clientes y desarrolladores sobre productos a entregar y su formalidad.

- Permite detectar errores tempranos a través de un ciclo iterativo.
- Por ser una metodología ágil tiene un enfoque centrado al cliente y con iteraciones cortas. (*Gestión, 2008)*

Por las características anteriormente expuestas se decide utilizar OpenUp como metodología para guiar el proceso de desarrollo de software.

1.6.2 Lenguajes de Programación.

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (*Ecured, 2013*)

PHP 5.3.10.

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. Es usado principalmente en interpretación del lado del servidor (*server-side scripting*) pero actualmente puede ser utilizado desde una interfaz de línea de comandos.

Es un lenguaje multiplataforma orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos. (*Php.net, 2013*)

Ventajas:

- El código fuente escrito en PHP 5.x.x hace que la programación en PHP sea segura y confiable porque es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones bastante amplia e incluida.
- Es completamente extensible. Está compuesto de un sistema principal, un conjunto de módulos y

una variedad de extensiones de código.

JavaScript.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. (*Eguíliz, 2009*)

JavaScript es un lenguaje de tipo *script* compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor. Los programas JavaScript van incrustados en los documentos XHMTL o en un fichero .js y se encargan de realizar acciones en el cliente.

Ventajas:

- Los programas escritos en este lenguaje no requieren de mucha memoria ni tiempo adicional de transmisión, por ser pequeños y compactos.
- No requiere un tiempo de compilación, pues los *scripts* se pueden desarrollar en un período de tiempo relativamente corto.
- Es independiente de la plataforma *hardware* o sistema operativo, este funciona correctamente siempre y cuando exista un navegador que lo soporte.
- Asegura la permanencia de una operación realizada y aunque falle el sistema esta no podrá deshacerse. (*Eguíliz, 2009*)

1.6.3 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. (*Ecured, 2013*)

Un marco de trabajo simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

Symfony 1.1.7.

Symfony es un completo marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web, el mismo se utilizó en la implementación del GDR 1.8. Para empezar, 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 de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. (*Potencier, 2008*)

Symfony está desarrollado completamente con PHP 5. Este marcode trabajo compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux) como en plataformas Windows.

Características:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Preparado para aplicaciones empresariales y es adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

ExtJS 2.2.

ExtJS es una librería de interfaz de usuario con todas las funciones capaz de crear interfaces dinámicas y fluidas dirigidas a datos estructurados que a su vez están relacionados con las fuentes de datos del servidor. Utilizada para la implementación del GDR 1.8 proporciona una forma fácil de usar para el usuario de la misma forma que una aplicación de escritorio. Esto le permite a los desarrolladores web centrarse en la funcionalidad de las aplicaciones en lugar de las advertencias técnicas. (*Frederick, 2012*)

Principales características de ExtJS:

- Alto rendimiento en ejecución debido a la optimización de código Java Script.
- Controles de usuario personalizables.

- Modelo orientado a componentes, bien diseñado y extensible.
- Posee una interfaz de programación de aplicaciones intuitiva y fácil de utilizar.
- Es distribuido bajo licencias *open source* y comerciales.

1.6.4 Herramienta de modelado.

CASE, acrónimo de (*Computer Aided Software Engineering*) es el tipo de herramienta destinada a aumentar la productividad en el desarrollo de *software* reduciendo el costo de las mismas en términos de tiempo y dinero. Son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

El empleo de herramientas Case permiten integrar el proceso de ciclo de vida:

- Análisis de datos y procesos integrados mediante un repositorio.
- Generación de interfaces entre el análisis y el diseño.
- Generación del código a partir del diseño.
- Control de mantenimiento. (*UNIDAD, 2013*)

Visual Paradigm 6.4.

La herramienta de modelado Visual Paradigm soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Utiliza como lenguaje de modelado el Lenguaje Unificado de Modelado (UML) que ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. También es uno de los lenguajes de modelado de sistemas de *software* más conocido y utilizado en la actualidad. Además Visual Paradigm es una potente herramienta CASE la cual permite dibujar todos los tipos de diagramas de clases y generar código desde diagramas. El mismo es multiplataforma, por lo que brinda facilidades para el diseño de los diagramas necesarios y su documentación. (*UNIDAD, 2013*)

Características Visual Paradigm:

- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Proporciona a los desarrolladores una plataforma que les permite diseñar un producto rápidamente y con la calidad requerida.

- Facilita la interoperabilidad con otras herramientas CASE y se integra con múltiples herramientas de desarrollo, como Eclipse/IBMWebSphere, Jbuilder, NetBeans IDE, Oracle Jdeveloper.
- Genera código y realiza ingeniería inversa para varios lenguajes de programación, Java, CCS, PHP y XML Schema. Código a modelo, código a diagrama.
- Genera documentación para el proyecto en HTML, MS Word y PDF.
- Licencia: gratuita y comercial.

1.6.5 Gestores de bases de datos.

Un sistema gestor de base de datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Permite almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las aplicaciones más usuales son para la gestión de empresas e instituciones públicas, además de ser ampliamente utilizado en entornos científicos con el objeto de almacenar la información experimental. (*Yanes, 2011*)

PostgreSQL 9.1.

Este gestor de base de datos 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. Favorece un ahorro considerable en costos de operación. (*Postgresql, 2013*)

Entre sus principales ventajas se destacan las siguientes:

- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- Posee una gran escalabilidad, haciéndolo idóneo para su uso en sitios web que atienden un gran número de solicitudes.
- Puede ser instalado un número ilimitado de veces sin temor de sobrepasar la licencia.
- Es extensible a través del código fuente disponible sin costos adicionales.
- Es multiplataforma, disponible en la mayoría de los sistemas operativos.
- Permite implementar reglas, vistas, disparadores, sub-consultas y funciones. (*Postgresql, 2013*)

1.6.6 Cliente para la gestión de bases de datos.

Un cliente para la gestión de bases de datos es una herramienta que permita al usuario administrar una o varias bases de datos de forma visual y amigable, permitiéndole una mejor organización y visualización de los datos a administrar. Estas herramientas, cuya interfaz puede ser web o de escritorio, permiten realizar consultas sobre las bases datos, realizar copias de seguridad y restauración de las mismas.

PgAdminIII 1.14.

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, con licencia *open source*. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. PgAdminIII está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL[13] simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración del mismo. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor y un agente para lanzar *scripts* programados. La conexión al servidor puede hacerse mediante conexión TCP/IP. (*Guía Ubuntu, 2013*).

1.6.7 Entorno Integrado de Desarrollo.

Un entorno de desarrollo informático IDE (*Integrated Development Environment* por sus siglas en inglés) es un programa informático compuesto por un conjunto de herramientas de programación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI[14]). Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como PHP, C++, Python, Java, C#, Delphi, Visual Basic. (*Yanes, 2011*)

NetBeans 7.2.

NetBeans es un entorno de desarrollo integrado de código abierto y gratuito para desarrolladores de *software*. El proyecto NetBeans es apoyado por una gran comunidad de desarrolladores y ofrece una

[13]SQL: El **lenguaje de consulta estructurado** o **SQL** (por sus siglas en inglés *Structured Query Language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

[14] GUI:*Graphic User Interface* o Interfaces Gráficas de Usuario

amplia documentación y recursos de capacitación. Además, ofrece todas las herramientas necesarias para crear aplicaciones de escritorio, empresariales, web y móviles con el lenguaje Java, JavaFX, C/C ++ y lenguajes dinámicos como PHP, JavaScript, Groovy y Ruby. (*NetBeans, 2013*)

Características generales:

- Integración con Symfony: Una de las ventajas de utilizar NetBeans es justamente su integración con este marco de trabajo de PHP, lo que hace posible dejar a un lado la consola de comandos de Symfony y centrarse en desarrollar en el IDE, además se encuentran cargadas las todas las clases y ayuda en línea.
- Editor de código fuente: El editor de PHP, es más ágil y robusto, contiene ayuda en línea y reconocimiento de sintaxis.
- Depuración de PHP: NetBeans integra muy bien la utilización de Xdebug, gracias a esto se puede inspeccionar y examinar cada variable local, establecer puntos de interrupción y evaluar el código. (*NetBeans, 2013*)

1.7 Patrones de Arquitectura.

Un patrón de arquitectura de *software* es un esquema genérico probado para solucionar un problema particular recurrente que surge en un cierto contexto. Este esquema se especifica describiendo los componentes con sus responsabilidades, relaciones y formas en que colaboran.

Cliente – Servidor: es un modelo de aplicación distribuida en el que las tareas se reparten entre los servidores y los clientes. Un cliente realiza peticiones al *software* y el servidor da la respuesta, los cuales no tienen que ejecutarse en la misma computadora.

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

- **Modelo:** representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos.
- **Vista:** muestra el modelo en un formato adecuado para interactuar. Maneja la presentación visual

de los datos representados por el Modelo.

- **Controlador:** responde a eventos e invoca cambios en el modelo y probablemente en la vista. (*Ivanek, 2013*)

1.8 Patrones de Diseño.

Un patrón de diseño nombra, abstrae, e identifica los aspectos claves de una estructura de diseño común que hacen que sea útil para la creación de un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases o instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema de diseño orientado a objeto en particular; en él se describe cuándo se aplica, si se puede aplicar a la vista de otras restricciones de diseño, y las consecuencias y compensaciones de su uso.(*Gamma, 2013*)

1.8.1 Patrones GOF (Gang of Four "Banda de los Cuatro")

Los patrones de diseño GOF dan una descripción de clases y objetos que se comunican entre sí, adaptada para resolver un problema general de diseño en un contexto particular. Se clasifican en:

- **Creacionales:** resuelven problemas relativos a la creación de objetos.
- **Estructurales:** resuelven problemas relativos a la composición de objetos.
- **De Comportamiento:** resuelven problemas relativos a la interacción entre objetos. (*Gamma, 2013*)

1.9 Patrones GRASP (General Responsability Assignment Software Patterns, "Patrones de Software para la Asignación General de Responsabilidad")

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Se pueden destacar 5 patrones principales que son: Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador. Además de otros patrones adicionales que son: Fabricación Pura, Polimorfismo, Indirección.

- **Bajo Acoplamiento:** debe haber pocas dependencias entre las clases. De manera que en caso de producirse una modificación en alguna de las clases se tenga la mínima repercusión posible en el resto de las clases. Estas clases no se afectan por los cambios en otros componentes, fáciles de entender por separado y de fácil reutilización.

- **Alta Cohesión:** cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. La información que almacena una clase debe ser coherente y estar en mayor medida relacionada con la clase. Estas clases mejoran la claridad y la facilidad con que se entiende el diseño.
- **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. Es el principio básico de asignación de responsabilidades.
- **Creador:** identifica quien debe ser el responsable de la creación o instanciación de nuevos objetos o clases. Se brinda un soporte al bajo acoplamiento.
- **Controlador:** asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión.
- **Polimorfismo:** cuando se identifican variaciones en un comportamiento, asignar la clase (interfaz) al comportamiento y utilizar polimorfismo para implementar los comportamientos alternativos.
- **Fabricación Pura:** cuando los problemas se complican se deben construir clases que se encarguen de construir los objetos adecuados en cada momento (factorías).
- **Indirección:** crear clases intermedias para desacoplar clientes de servicio y servicios. (*Hernández, 2013*)

1.10 Conclusiones parciales del capítulo.

A partir de la bibliografía consultada y el estudio realizado de la misma se agruparon los aspectos teóricos imprescindibles para el desarrollo de la investigación, lo que posibilitó identificar y agrupar los elementos necesarios para realizar la exportación e importación del diseño de reportes. El estudio de las herramientas, tecnologías, y metodología para el desarrollo del componente de exportación e importación del diseño de reporte permitió realizar un análisis que culminó con la selección de las mismas para guiar el proceso de desarrollo de software. OpenUp como metodología para guiar dicho proceso, el IDE NetBeans 7.2 que presenta integración con Symfony 1.1.7 y ExtJS 2.2 como marcos de trabajo. PostgreSQL 9.1 como gestor de bases de datos y PgAdminIII para administrar este gestor. Como herramienta de modelado Visual Paradigm 6.4 de gran utilidad para la generación de los artefactos durante el diseño de la solución.

Capítulo 2: Análisis y Diseño del componente para la exportación e importación del diseño de reportes.

Introducción.

En este capítulo se describe el proceso llevado a cabo durante la fase de análisis y diseño, en el cual se definen los requisitos funcionales y no funcionales del *software*. A partir de los requisitos funcionales se identifican y describen los casos de uso y como parte del proceso que se realiza durante esta fase se generan los diferentes diagramas y artefactos correspondientes para dar cumplimiento a los objetivos trazados.

2.1 Modelo de dominio.

El modelo de dominio constituye una representación de las clases conceptuales del mundo real, no de componentes *software*. Este permite mostrar a los modeladores clases conceptuales significativas en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos y se utiliza con frecuencia como fuente de inspiración para el diseño de los objetos *software*. La identificación de las clases conceptuales forma parte del estudio del dominio del problema. El lenguaje UML contiene notación, en forma de diagramas de clases, para representar los modelos del dominio. (*Larman, 1999*). A continuación se muestra el modelo de dominio del componente y la descripción de sus clases.

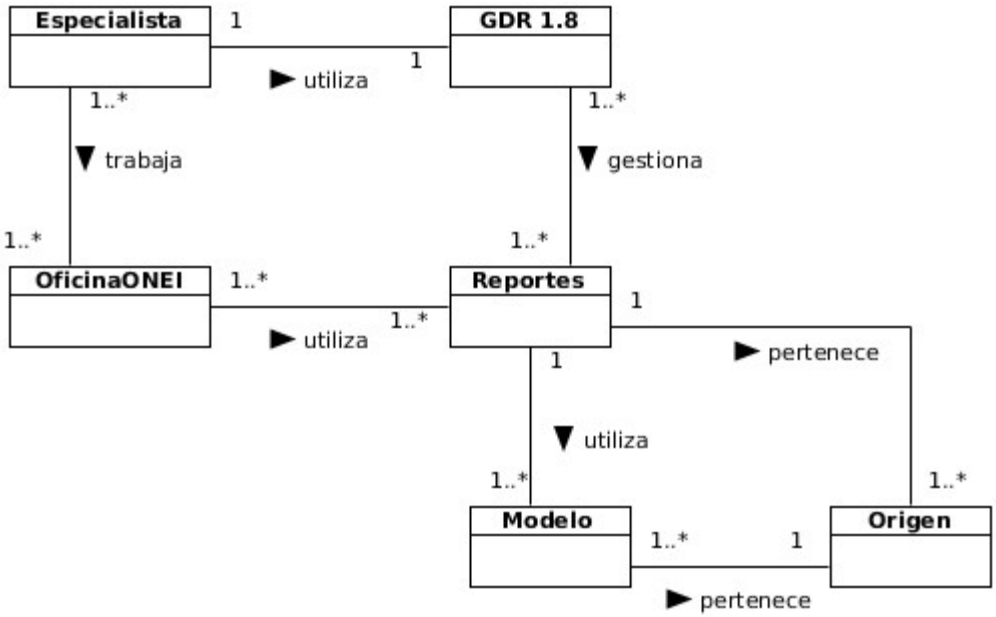


Fig. 1: Modelo de dominio

Especialista: Especialista de la ONEI responsable y capacitado para utilizar el Generador Dinámico de Reportes.

GDR 1.8: Sistema Informático que garantiza la generación dinámica de reportes partiendo de datos persistentes en algún origen de datos soportado por el sistema. Ofrece la posibilidad de controlar el funcionamiento periódico de una o varias entidades, a través de la creación de reportes en distintos formatos.

OficinaONEI: Oficina que se encarga de producir y difundir estadísticas

Origen: Contiene los datos que permiten conectar al Generador Dinámico de Reportes con los Sistemas Gestores de Bases de Datos. Las variables que maneja son: tipo de gestor de base de datos, dirección IP del servidor, puerto de conexión al servidor, usuario, clave y base de datos. Luego de obtenidos los datos permite generar el modelo de datos para obtener los reportes.

Modelo: Se compone por un conjunto de entidades asociadas a un origen de datos registrado en el sistema. Las mismas se utilizan para crear los datos a procesar en la creación del reporte.

Reporte: Es un archivo, generado por un motor de reportes, con diferentes estructuras que contiene los datos de acuerdo a un interés específico.

2.2 Requisitos funcionales.

Los requisitos funcionales son condiciones que el sistema debe cumplir. El componente para la exportación e importación del diseño de reportes debe cumplir con los requisitos funcionales que a continuación se describen:

RF 1: Mostrar los reportes existentes en la base de datos

- Descripción: el componente debe permitir mostrar la lista de reportes almacenados tal y como éstos se encuentran ubicados por categorías que permite agruparlos por áreas temáticas.
- Entrada: solicitud por parte del sistema de visualización de los reportes almacenados.
- Salida: se muestra una lista de reportes organizados y almacenados en la base de datos.

RF 2: Seleccionar reportes existentes en la base de datos

- Descripción: el componente debe permitir seleccionar los reportes deseados para transferir y en caso de error en la selección revertir la misma
- Entrada: selección de reportes deseados mediante *clics*
- Salida: se muestra una lista con los reportes seleccionados previamente**.**

RF 3: Exportar reportes

- Descripción: el componente debe permitir exportar los reportes previamente seleccionados hacia un archivo determinado por el usuario en extensión XML, y le permite además ubicar la ruta donde se desea almacenar.
- Entrada: solicitud de exportación de reportes a un formato determinado
- Salida: fichero almacenado que contiene los datos de los reportes almacenados en el sistema, y la definición del las vistas, funciones y tipos de datos compuestos, exportados hacia una dirección especificada.

RF 4: Seleccionar el fichero a importar.

- Descripción: el componente debe permitir seleccionar el fichero XML que contiene los datos de los reportes a importar a partir de la ruta donde éste se encuentra almacenado.
- Entrada: ruta o directorio donde se encuentra almacenado del fichero XML deseado.
- Salida: ruta de un fichero XML compatible con el sistema.

RF 5: Mostrar los reportes existentes en el fichero

- Descripción: el componente debe permitir mostrar una lista de los reportes almacenados en el

fichero XML, organizados por categorías

- Entrada: solicitud del sistema de visualización de los reportes existente en el fichero XML
- Salida: muestra una lista con los reportes existentes en el fichero organizados por categorías.

RF 6: Seleccionar los reportes del fichero a importar

- Descripción: el componente debe permitir seleccionar los reportes que se desean importar y de la misma forma revertir la acción en caso de error en el momento de selección.
- Entrada: selección de reportes deseados mediante *clics*.
- Salida: muestra una lista con los reportes seleccionados previamente organizados según su categoría.

RF 7: Importar reportes.

- Descripción: el componente debe permitir importar hacia la base de datos los reportes previamente seleccionados.
- Entrada: solicitud de importación de reportes.
- Salida: entrada de los nuevos reportes almacenados en la base de datos.

RF 8: Editar el nombre de elementos repetidos

- Descripción: el componente debe permitir renombrar los orígenes, modelos, reportes, vistas y funciones en caso de que ya exista alguno con el mismo nombre en la base de datos destino.
- Entrada: nombre del elemento repetido que se muestra en una ventana
- Salida: nombre del elemento actualizado que se muestra en un cuadro de texto editable.

2.3 Requisitos no funcionales.

Los requisitos no funcionales son las propiedades o cualidades que el producto debe tener. El componente para la exportación e importación del diseño de los reportes cuenta con los siguientes requisitos no funcionales:

Usabilidad.

RNF 1: Facilidad de uso al usuario para el manejo de la información del sistema.

El componente debe presentar una vista sencilla, descriptiva y fácil de usar. Los botones deben estar alineados a la derecha, y de organizados de forma intuitiva.

RNF 2: Seleccionar de manera sencilla los reportes que se desean exportar.

El usuario podrá localizar y seleccionar un reporte de manera rápida. Los reportes estarán organizados por categorías lo que hará más fácil su localización en el sistema.

RNF 3: Exporta los reportes de manera sencilla y ágil.

El usuario podrá exportar los reportes de una forma sencilla e intuitiva sin necesidad de ser un experto en el uso de la herramienta. Los botones de la interfaz estarán alineados a la derecha con íconos representativos de la acción a realizar

RNF 4: Seleccionar de manera sencilla los reportes que se desean importar.

Luego de ubicar el archivo en el sistema, el usuario podrá localizar y seleccionar un reporte de manera rápida, éstos estarán ubicados en alguna categoría que permita organizarlos en áreas temáticas comunes.

RNF 5 Importar los reportes de manera sencilla y ágil.

El usuario podrá importar los reportes de una forma sencilla e intuitiva sin necesidad de ser un experto en el uso de la herramienta. Los botones de la interfaz estarán alineados a la derecha con íconos representativos de la acción a realizar.

Interfaz.

RNF 6 El usuario debe acceder a la aplicación a través del protocolo HTTP o HTTPS en dependencia de la configuración del navegador, utilizando Mozilla Firefox desde la versión 2.0 hasta la 28.0. El sistema se desarrollará para la web, pero con características muy similares a las aplicaciones de escritorio, en cuanto al diseño de las interfaces visuales.

Requisitos de *Software*.

RNF 7 El servidor donde se instalará la aplicación debe cumplir con los siguientes requisitos de *software*:

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 hasta la versión 12.04 o Debian 4 GNU/Linux hasta la versión 6.0.
- Paquetes: apache2, php5, libapache2-mod-php5, php5-cli, php5-mysql, php5-pgsql, php5-sqlite, php5-sybase, php5-xsl, php5-gd, php-apc.
- Usuario con privilegios de administración del Sistema Operativo.

RNF 8 El servidor donde se instalará la Base de Datos del sistema debe cumplir con los siguientes requisitos de *software* (puede ser el mismo donde estará la aplicación):

- Sistema Operativo: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 hasta 12.04 o Debian 4 GNU/Linux hasta la versión 6.0
- PostgreSQL versión 8.3 hasta la 9.1.

- Usuario con privilegios para instalar la base de datos.
- PGAdminIII o algún administrador para PostgreSQL.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP utilizando el método de autenticación por md5.

Requisitos de *hardware*.

RNF 9 Las PC clientes deben cumplir con los siguientes requisitos de *hardware*:

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 256MB.
- Disco Duro con 10 Gb de capacidad mínima

RNF 10 Las PC servidor deben cumplir con los siguientes requisitos del *hardware*:

- Ordenador Pentium IV o superior, con 1.7 GHz de velocidad de microprocesador.
- Memoria RAM mínimo 1Gb.
- Disco Duro con 10Gb de capacidad para instalar el sistema.

Licencia

RNF 11: No se posee ningún requisito de licencia o restricción en el uso del *software* puesto que se desarrollará con la utilización de herramientas libres como: Symfony 1.1.7, PostgreSQL 9.1, PgAdminIII 1.14, Netbeans 7.2 y ExtJS 2.2.

Seguridad

RNF 12: La información manejada por el sistema deberá estar protegida de acceso no autorizado y encontrarse disponible sólo para los usuarios que pertenezcan al sistema. Durante el desarrollo de la solución se cuenta con un sistema de control de versiones alojado en un servidor con acceso mediante autenticación autorizada.

2.4 Modelo de caso de uso del sistema.

El modelo de casos de uso documenta el comportamiento de un sistema desde el punto de vista del usuario. Por tanto los casos de uso determinan los requisitos funcionales del sistema, los cuales representan las funcionalidades que un sistema puede ejecutar. Luego de realizar el análisis se identificaron ocho requisitos funcionales agrupados en dos casos de uso. El patrón de caso de uso

presente es CRUD[15] Parcial. Este patrón alternativo modela una de las vías de los casos de uso como un caso de uso separado. Es preferiblemente utilizado cuando una de las alternativas de los casos de uso es más significativa, larga o más compleja que las otras. En la siguiente figura se representan las relaciones entre el actor y los casos de uso del sistema, evidenciándose los principales procesos del sistema.

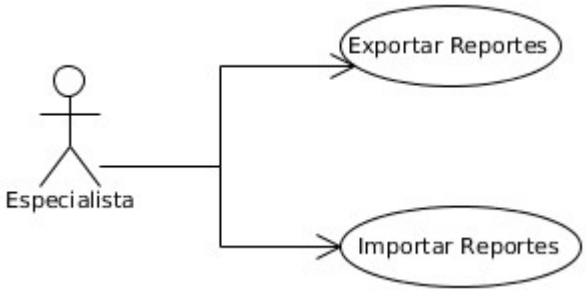


Fig. 2: Diagrama de casos de uso del sistema

Tabla 2: Descripción del actor del sistema

Actor	Descripción
Especialista	Ente capacitado en el uso del sistema GDR encargado de exportar e importar los reportes de una aplicación a otra.

Caso de uso Importar Reportes.

Tabla 3: Descripción del caso de uso Importar Reportes.

Caso de Uso	Importar Reportes
Actor	Especialista

[15]CRUD: Create, Read, Update y Delete por sus siglas en inglés

Resumen	El caso de uso se inicia cuando el especialista selecciona la opción “Importar” del componente para la exportación e importación del diseño de reportes. Permite seleccionar e importar los reportes almacenados en un archivo hacia el sistema. Culmina cuando termina de realizar estas operaciones.
Precondiciones	Debe existir al menos un archivo exportado que contenga los reportes. El archivo debe contener al menos un reporte almacenado.
Referencias	RF 4, RF 5, RF 6, RF 8
Prioridad	Alta

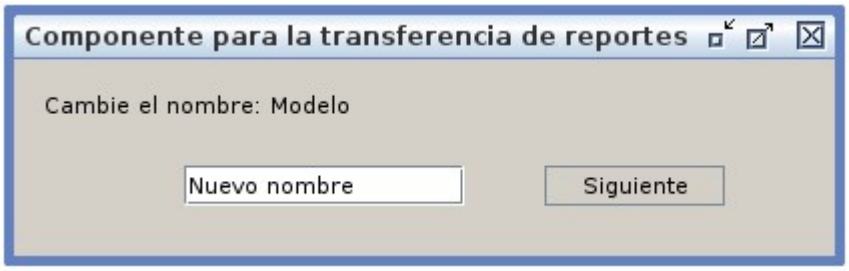
Flujo de eventos		
Flujo básico Importar Reportes		
	Actor	**Sistema**
1	El especialista selecciona la opción “Importar Reportes” desde la interfaz inicial del componente.	
Prototipo de Interfaz		

2		Se muestra una interfaz que permite localizar el archivo.
3	El especialista selecciona el archivo desde el cual se desea importar los reportes.	

Prototipo de Interfaz

4		Se muestra una interfaz con los reportes almacenados en el archivo.

5	El especialista selecciona los reportes que desea importar.	
6		Se muestra una interfaz con los reportes seleccionados.
7	El especialista selecciona la opción “Importar”.	
Prototipo de Interfaz		
		
8		Se muestra un mensaje indicando el éxito de la acción. Termina el caso de uso.
Flujos alternos: En caso de que ya existan los elementos a importar		
	Actor	**Sistema**
7a		Se muestra una interfaz que permite renombrar los elementos ya existentes.
7b	Renombra los elementos ya existentes	
Prototipo de Interfaz		



Postcondiciones	Se importan los reportes seleccionados hacia la base de datos.

2.5 Modelado de la exportación e importación de reportes.

En el presente epígrafe se explican las peculiaridades de los procesos de exportación e importación del diseño de los reportes a partir de las entradas, descripciones y salidas de estos procesos. Se describe la síntesis de la estructura del formato de salida XML a partir del flujo de datos para conformar el mismo y cómo se validan estos para su importación en el sistema.

2.5.1 Exportación de reportes.

La selección del formato de salida XML parte de continuar con la tendencia de las herramientas estudiadas a la hora de exportar sus reportes, estructurándose de una forma organizada que sirva de guía en el proceso inverso de importación. Para ello se tiene como **entrada** la selección de los reportes que se desean exportar. A partir de ahí se seleccionan las categorías por las que están organizadas, los modelos y orígenes de datos asociados y las relaciones entre ellos según la forma en que estos persisten en la base de datos del sistema. Se incluye además las definiciones de las fuentes de datos que construyen el reporte en caso de que sean vistas o funciones procedurales, y en el caso de éstas últimas las definiciones de sus tipos de datos compuestos.

Los datos obtenidos se organizan a través de la creación de un documento XML que organiza sus etiquetas unas dentro de otras en dependencia de la selección realizada, siguiendo el siguiente orden:

1. Por cada reporte seleccionado se agrupan todas las categorías en las que están organizados con

sus características.

2. Se agrupan los reportes seleccionados con cada una de sus características.
3. Se agrupan los modelos asociados a los reportes seleccionados con sus características.
4. Se agrupan los orígenes de datos asociados a los reportes seleccionados con sus características.
5. Se agrupan las relaciones entre reportes y categorías.
6. Se agrupan las relaciones entre reportes y orígenes de datos.
7. Se agrupan las relaciones entre reportes y modelos de datos
8. Si la fuente de datos es una vista o una función procedural.
    a. Se agrupa la definición nombre de la vista o función procedural.
    b. Si es una función procedural
        i. Se agrupa la definición tipo de dato compuesto.

Como **salida** se obtendría un documento XML con las siguientes etiquetas:

- <*categoría*>: Este elemento contiene los campos de la tabla que almacena las categorías por las cuales están organizados los reportes.
- <*reportes*>: Este elemento contiene los datos asociados a los reportes y sus atributos.
- <*model*>: Contiene los datos de los modelos y sus atributos
- <*datasource*>: Este elemento contiene las características de los orígenes asociados y sus atributos
- <*reportecategoria*>: Contiene los atributos de las relaciones entre los reportes y las categorías.
- <*reportedatasource*>: Contiene los atributos de las relaciones entre los reportes y los orígenes de datos.
- <*reportemodel*>: Contiene los atributos de las relaciones entre los reportes y los modelos de datos.
- <*fuente*>: Este elemento contiene las definiciones de las fuentes de datos para la construcción de reportes.

2.5.1 Importación de reportes.

Para realizar la importación de los reportes se tiene como **entrada** el archivo XML generado por el proceso Exportar Reportes, que a partir de un conjunto de peticiones y modificaciones permitirán la incorporación de los nuevos reportes al sistema.

1. En una primera petición se localiza el archivo deseado y se comprueba que cumple con la estructura previamente diseñada en el proceso de Exportar Reportes. Se obtienen los nombres de los reportes que están presenten en dicho archivos y se almacenan los mismos de forma temporal, se muestran los reportes al usuario para que este los seleccione.

2. Una segunda petición ocurre una vez que el usuario ha seleccionado los reportes deseados, a partir de ahí se determinan cuáles son los reportes cuyo nombre ya existían previamente en el sistema y se procede a renombrarlos, de la misma forma ocurre con las fuentes de datos para la creación de los mismos en caso de que estas fueran vistas o funciones. Se actualiza el archivo temporal con los nuevos nombres de los elementos.

3. La tercera y última petición incorpora desde el archivo temporal los reportes y sus elementos correspondientes hacia la base de datos del sistema, y se obtiene como **salida** los nuevos reportes incorporados en la base de datos del sistema.

2.6 Modelo de diseño.

El diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución que se prepara para la implementación y prueba del sistema. Tiene como finalidad:

- Transformar los requisitos en un diseño del sistema.
- Evolucionar una arquitectura sólida para el sistema.
- Adaptar el diseño para que se ajuste al entorno de implementación. (*Jacobson, 2000*)

2.6.1 Diagrama de clases del diseño.

Un diagrama de clases describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Éstos son utilizados durante el proceso de análisis y diseño de los sistemas, donde

se crea el diseño conceptual de la información que se manejará en el sistema. Además de los componentes que se encargarán del funcionamiento y la relación entre uno y otro. (*Jacobson, 2000*) A continuación se muestra el diagrama de clases del diseño del caso de uso Importar Reportes.

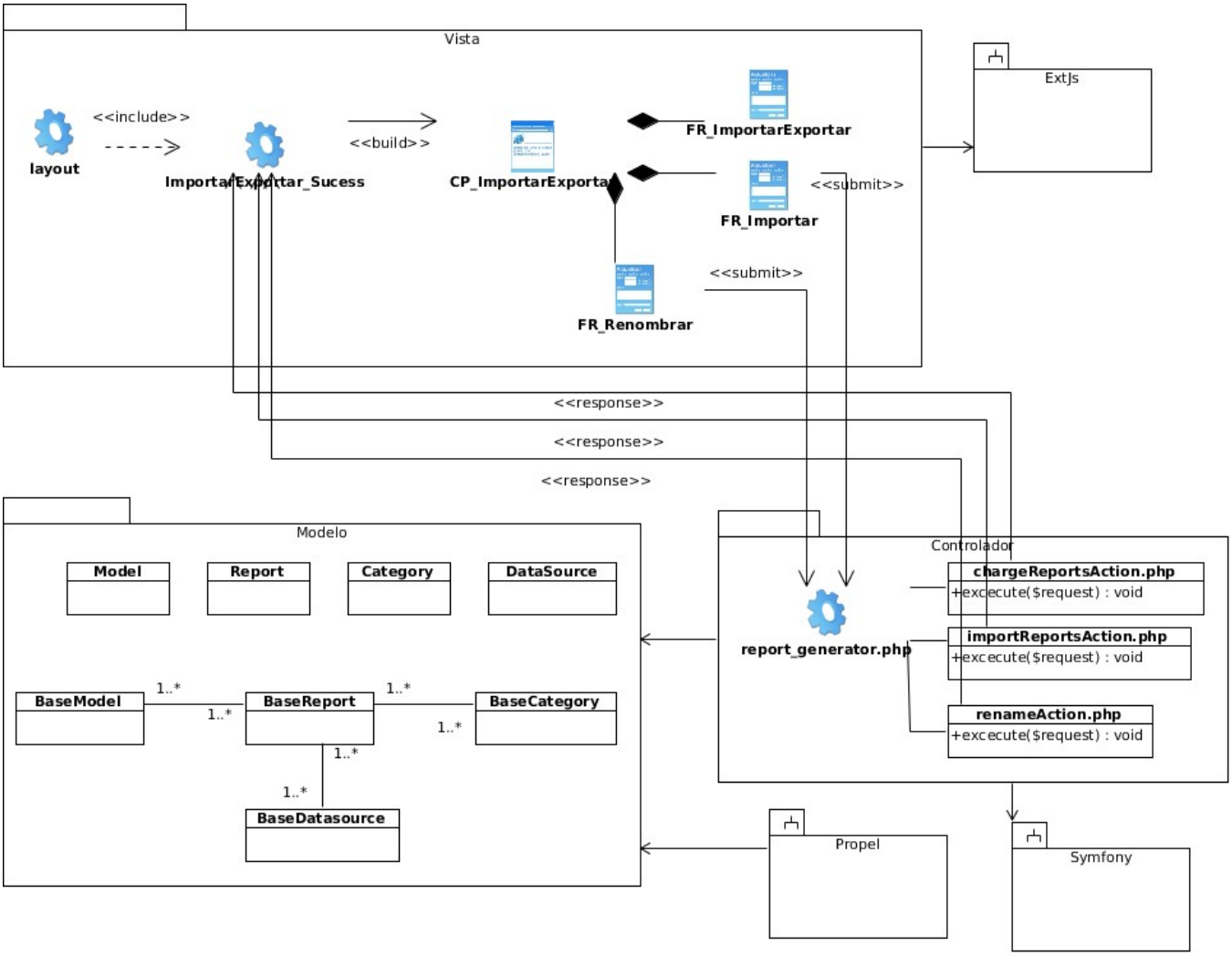


Fig. 3: Diagrama de clases del diseño CU Importar Reportes.

Modelo: esta capa contiene todas las clases de la base de datos, se utiliza Propel como ORM[16]. Cada una con sus funciones específicas. Contiene las clases de abstracción de datos, las cuales son encargadas de la abstracción de datos y responsables de realizar las operaciones con la base de datos.

[16]ORM: es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional

Controlador: esta capa es la que recibe las entradas, traducidas a solicitudes de servicios para el modelo. Es un bloque de código que realiza llamadas al modelo para obtener los datos y se los pasa a la vista para que los muestre al usuario.
Vista: en esta capa se encuentran las interfaces de usuario de la aplicación, con las cuales los usuarios pueden interactuar.

2.7 Patrones de Arquitectura

Modelo-Vista-Controlador (MVC).

Modelo-Vista-Controlador: este patrón se utiliza para modelar los diferentes diagramas los cuales están agrupados por paquetes que representan las capas, en la que cada una realiza una función específica. En la Vista se encuentran todas las interfaces con las que el usuario puede interactuar. En el Controlador se encuentran todas las acciones, con las cuales se le da respuesta a las peticiones del usuario que llegan a través del controlador frontal. En el Modelo se encuentran las tablas de la base de datos, en este caso se usa Propel como ORM, el cual crea cuatro clases por cada tabla. Este patrón se evidencia en la imagen del diagrama de clases del diseño del caso de uso Importar Reportes.

Cliente – Servidor: este patrón se evidencia, pues se separa al cliente del servidor. Lo que significa que el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa.

2.8 Patrones GRASP.

Controlador: utilizado para separar la lógica de negocio de la capa de presentación, controlando el flujo de eventos mediante las acciones de Symfony. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la URL entrada por el usuario. Se evidencia en el archivo report_generator.php.

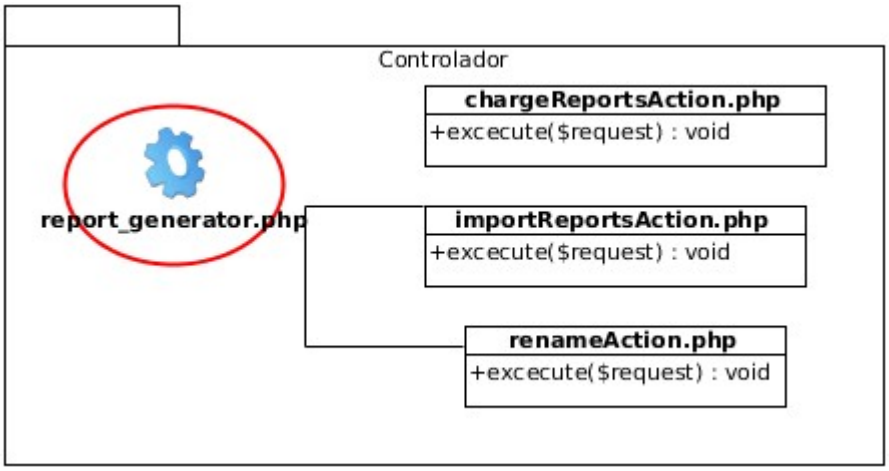


Fig. 4: Patrón Controlador

Experto: este patrón se evidencia en la capa modelo donde existen dos tipos de clases, las de abstracción de datos y las de acceso a datos. Symfony utiliza para el GDR1.8 Propel como ORM, el cual crea 4 clases. Las de acceso a datos que trabajan directamente con la base de datos utilizando Propel y las de abstracción de datos que son las que tienen los atributos necesarios para realizar dicha función. Por tanto deben implementar la responsabilidad de realizar las acciones directamente con la base de datos y ahí es donde se aplica el patrón. Por ejemplo la clase Model no implementa los métodos para comenzar a interactuar con la base de datos, tan sólo le avisa a la BaseModel lo que tiene que hacer, pues esta si tiene todos los datos necesarios para ejecutar la acción.

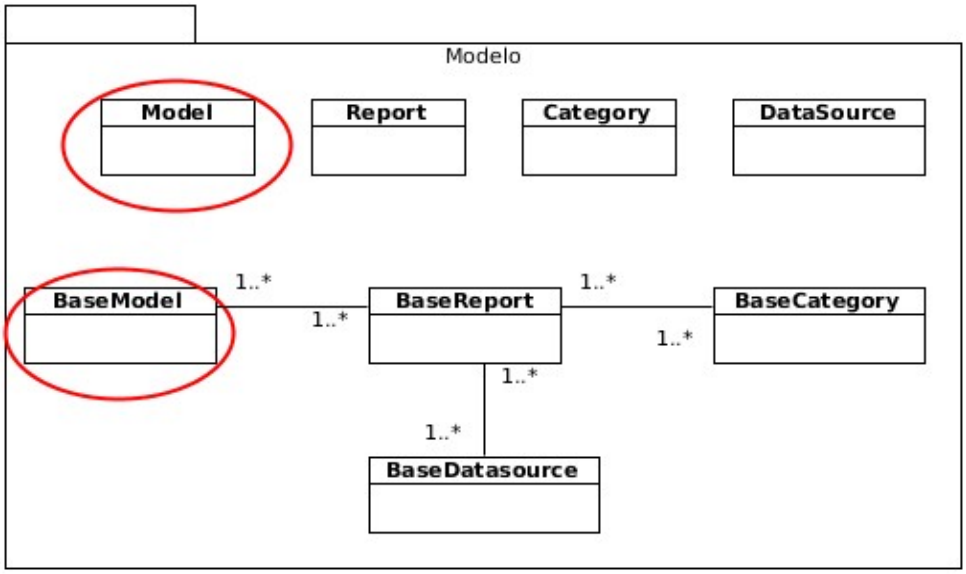


Fig. 5: Patrón Experto

2.9 Patrones GOF.

Decorator (Decorador): está presente en la layout, este archivo, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

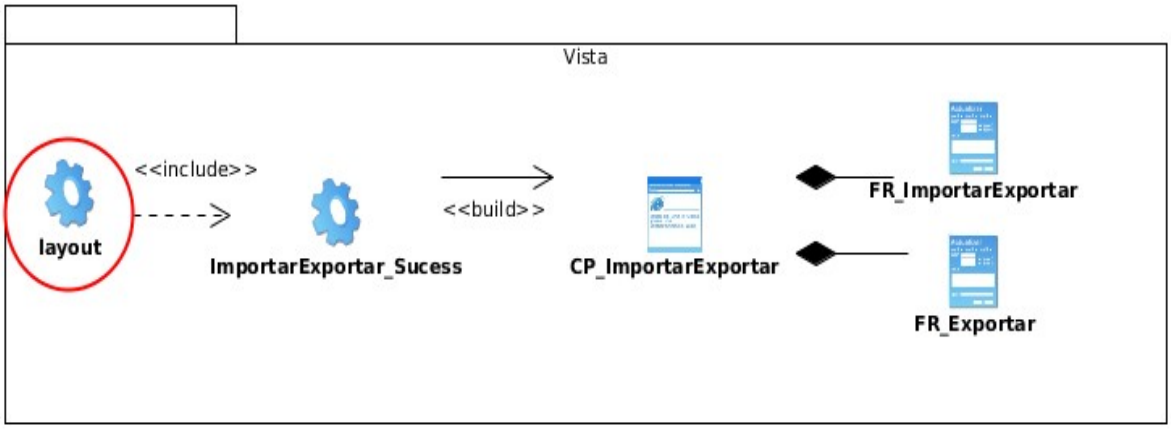


Fig. 6: Patrón Decorador

Command (Orden): en el *framework* Symfony se utiliza este patrón, el cual está representado por las acciones, por tanto cada acción, encapsula una petición en un objeto. Se evidencia en la clase importReportAction.php.

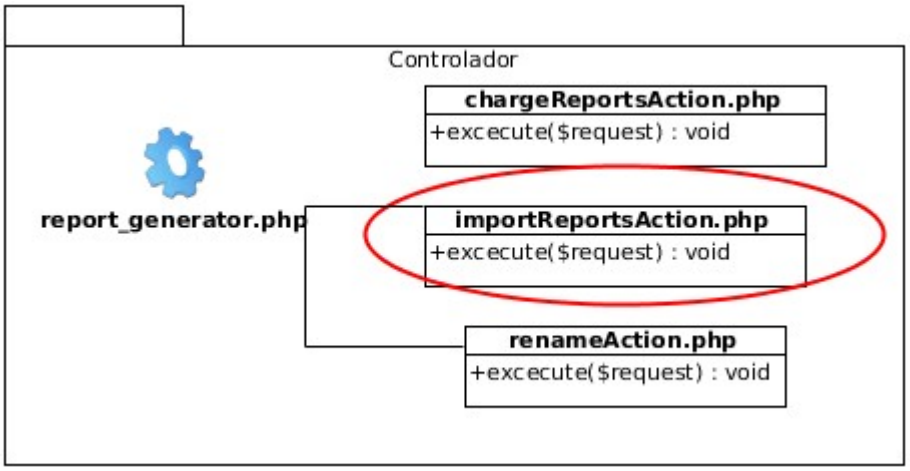


Fig. 7: Patrón Comand (Orden)

2.10 Diagramas de secuencia.

Los diagramas de secuencia ofrecen las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto. Describen el curso particular de los eventos de un caso de uso, proporcionando una realización física de comportamiento de los casos de uso en términos de clases del diseño. (*Larman, 1999*)

El escenario importar reportes representa las acciones que se ejecutan para realizar dicha acción. El usuario selecciona el fichero desde donde desea importar los reportes, luego el sistema analiza el fichero y muestra al usuario para que este seleccione los reportes que se encuentran almacenados. Seguidamente el usuario selecciona la opción Importar y de esta forma se incorporan los nuevos reportes, previamente seleccionados al sistema. El flujo alterno ocurre en caso de que exista algún elemento repetido se muestra una interfaz para renombrar los mismos antes de importarlos. A continuación se muestra el diagrama de secuencias del caso de uso Importar Reportes.

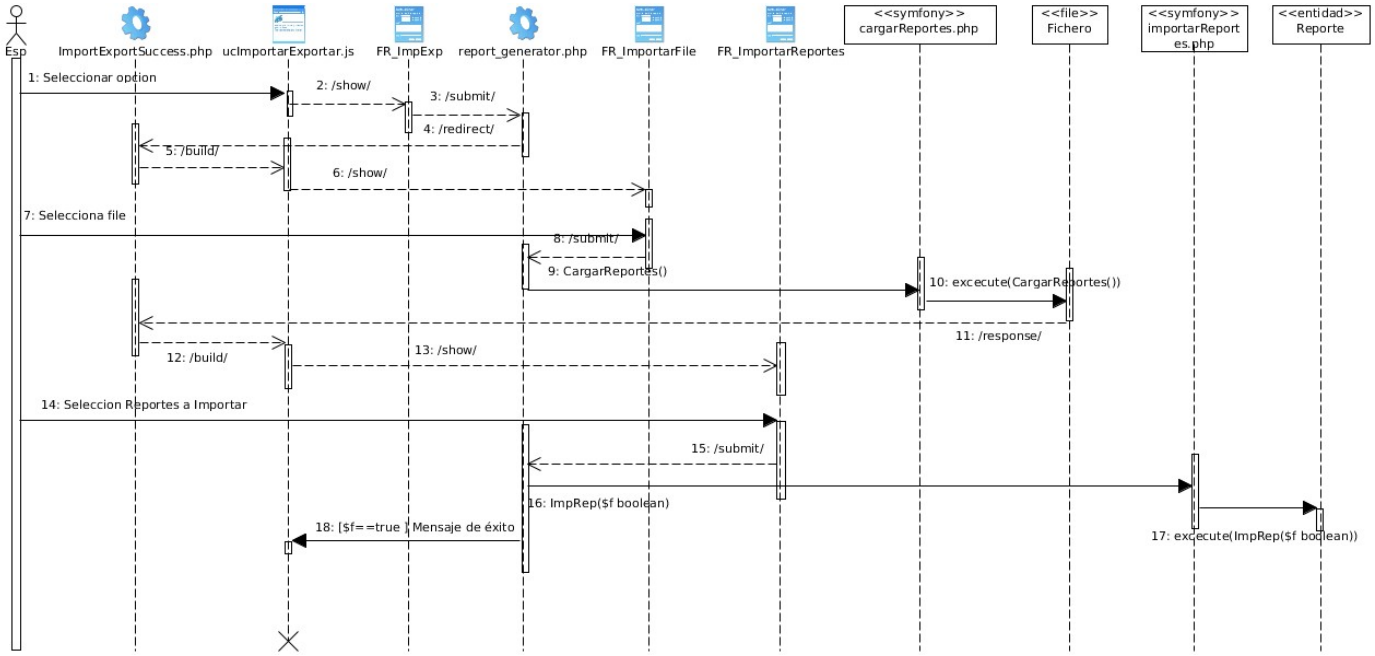


Fig. 8: Diagrama de secuencia del CU Importar Reportes

2.11 Modelo de datos.

Un modelo de datos es la estructura o representación física de las tablas de la base de datos obtenido a partir del diagrama de clases persistentes. Aporta la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos. De igual forma, permiten describir las estructuras de la base de datos, las restricciones de integridad y las operaciones de manipulación de los datos. (*MundoGeek, 2014*) A continuación se muestra el diagrama entidad-relación del componente de la solución.

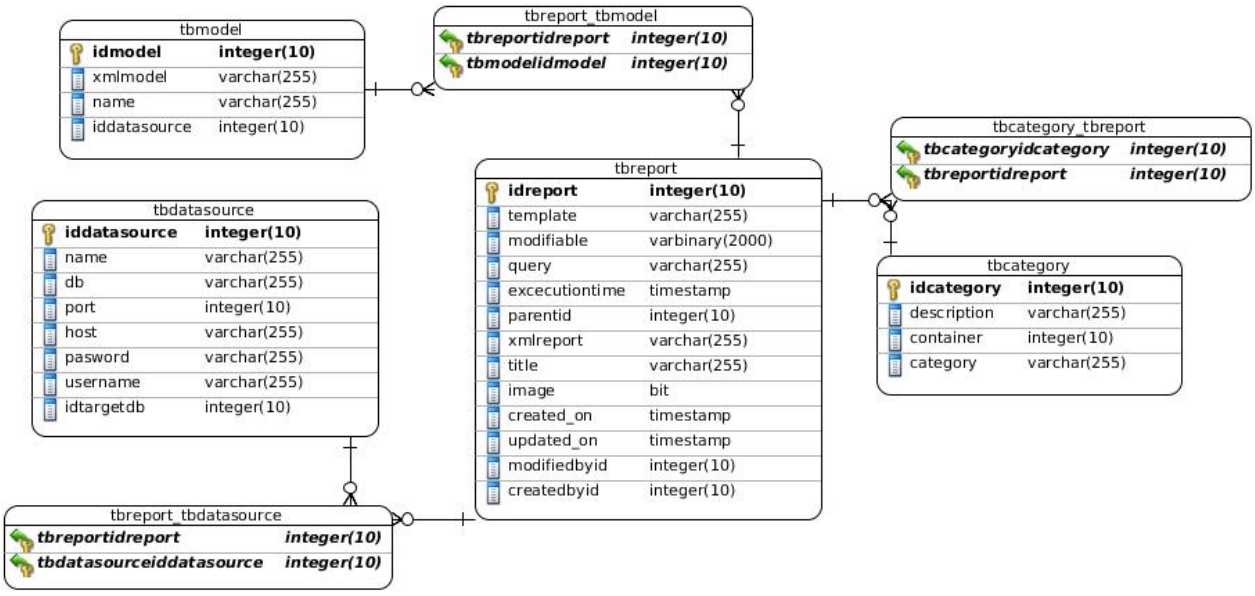


Fig. 9 Diagrama entidad-relación

Descripción de las tablas.

Tabla 4: Descripción de la tabla tbreport

Nombre: tbreport		
Descripción: Almacena las características de los reportes.		
Atributo	Tipo	Descripción
Idreport	integer	Identificador auto-incremental.
template	varchar	Contiene las plantillas de los reportes.
modifiable	varbinary	Contiene las propiedades de modificación de los reportes.
query	varchar	Contiene las consultas de los reportes.
excecutiontime	timestamp	Contiene el tiempo de ejecución de un reporte.
parentid	integer	Contiene el id padre o raíz de los Idreport
xmlreport	varchar	Contiene el XML del cuerpo del reporte.
title	varchar	Contiene el titulo del reporte.
image	bit	Contiene las imágenes que tiene el reporte.

created-on	timestamp	Tiempo de creación del reporte.
updated-on	timestamp	Tiempo de modificación del reporte.
modifiedbyid	integer	Contiene los datos de modificación.
createbyid	integer	Contiene los datos de creación.

Tabla 5: Descripción de la tabla tbdatasource

Nombre: tbdatasource		
Descripción: Almacena los datos del nodo origen para el cual se almacenarán las informaciones de los reportes.		
Atributo	Tipo	Descripción
Iddatasource	integer	Identificador auto-incremental.
Name	varchar	Contiene el nombre del nodo origen donde estarán ubicados los reportes.
Db	varchar	Contiene el nombre de la base de datos.
Port	integer	Puerto por la cual se conectará el servidor central.
Host	varchar	Número de IP del nodo origen.
Username	varchar	Usuario por el cual se conectara al nodo origen
Password	varchar	Contraseña del usuario.
Idtargetdb	integer	Identificador auto-incremental.

Tabla 6: Descripción de la tabla tbmodel

Nombre: tbmodel		
Descripción: Almacena las características de los modelos de datos.		
Atributo	Tipo	Descripción
Idmodel	integer	Identificador auto-incremental.
Xmlmodel	varchar	Contiene el xml del cuerpo del modelo de datos.
Name	varchar	Contiene el nombre del modelo de datos almacenado en la base de datos.
Iddatasource	integer	Identificador auto-incremental.

Tabla 7: Descripción de la tabla tbcategory

Nombre: tbcategory		
Descripción: Almacena las características de las categorías de reportes		
Atributo	Tipo	Descripción
Idcategory	integer	Identificador auto-incremental.
Description	varchar	Contiene la descripción textual de la categoría.
Container	integer	Contiene el id padre o raíz de las categorías.
Catgory	varchar	Contiene el nombre de la categoría.

Tabla 8: Descripción de la tabla tbreportdatasource

Nombre: tbreportdatasource Descripción: Almacena los reportes y los orígenes de datos.		
Atributo	Tipo	Descripción
Idreport	integer	Identificador auto-incremental.
Iddatasource	integer	Identificador auto-incremental.

Tabla 9: Descripción de la tabla tbreportmodel

Nombre: tbreportmodel Descripción: Almacena los reportes y los modelos de datos.		
Atributo	Tipo	Descripción
Idreport	integer	Identificador auto-incremental.
Idmodel	integer	Identificador auto-incremental.

Tabla 10: Descripción de la tabla tbcategoryreport

Nombre: tbcatgoryreport		
Descripción: Almacena los reportes y las categorías		
Atributo	Tipo	Descripción
Idreport	integer	Identificador auto-incremental.
Idcategory	integer	Identificador auto-incremental.

2.12 Modelo de despliegue.

Los diagramas de despliegue son los complementos de los diagramas de componentes que, unidos,

proveen la vista de implementación del sistema. Describen la topología del sistema y la estructura de los elementos de *hardware* y *software* que ejecuta cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Los nodos son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. Los diagramas de despliegue muestran la configuración en funcionamiento del sistema incluyendo su *software* y su *hardware*. (*Ingenieriasoftwaredos, 2014*) A continuación se muestra la estructura de los componentes que se desplegarán a lo largo de la infraestructura del sistema.

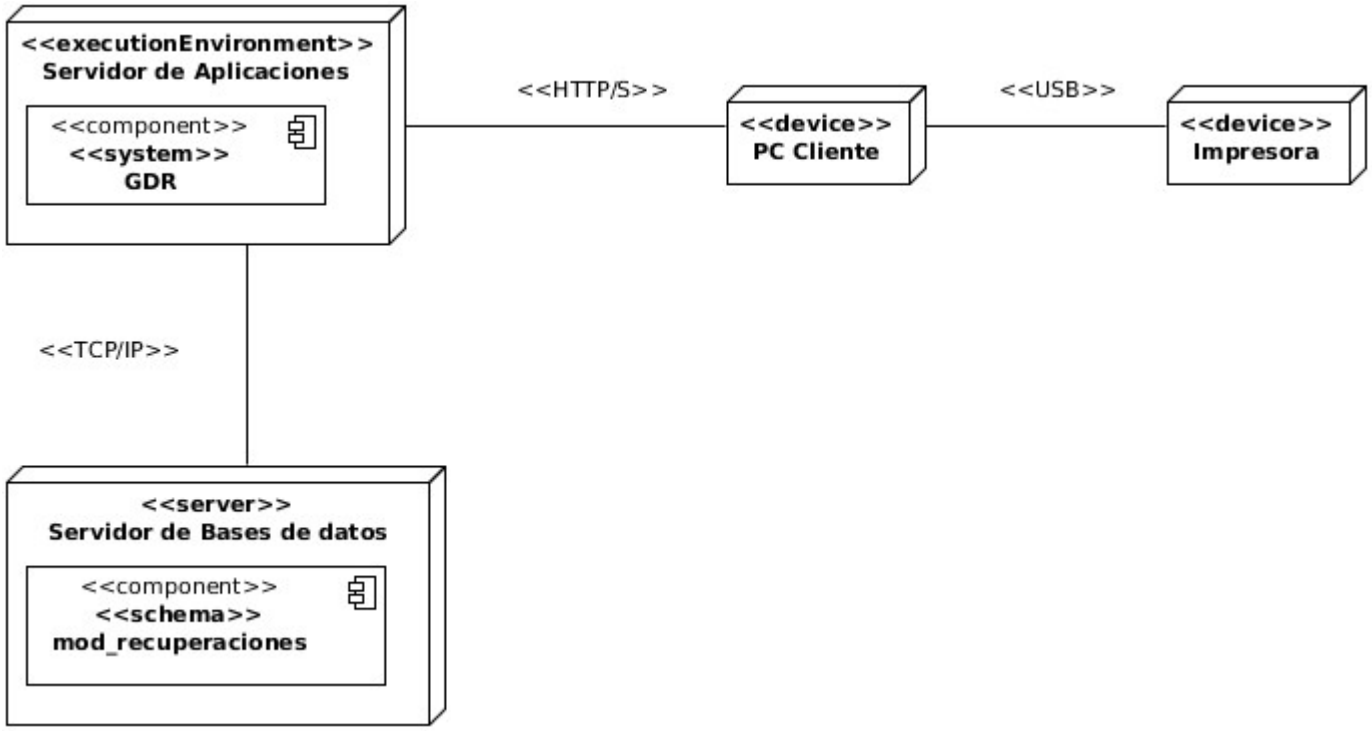


Fig. 10: Diagrama de despliegue

- **Servidor de aplicaciones:** Servidor de aplicación utilizado para la publicación de la aplicación; y para lograr la conexión del sistema con la PC Cliente se utiliza protocolo de transferencia HTTP/S como protocolo de comunicación. Es la herramienta principal para ejecutar la lógica de negocio en el lado del servidor. Es el responsable de ejecutar el código de las páginas servidor. Se utiliza el servidor de aplicación Apache Tomcat.
- **Servidor de Bases de Datos:** Representa a un servidor que radica en cada nodo regional en el

cual el cliente define que sean guardados los datos. En el servidor central estarán almacenados todos los datos recopilados por todos los nodos regionales. Contiene el esquema “mod_recuperaciones” del GDR y actúa como base de datos cliente.

- **PC Cliente:** Representa a las estaciones de trabajo que el usuario utilizará para acceder a la aplicación Web y transcribir sus datos.
- **Impresora:** Representa a las impresoras conectadas a las PC clientes en caso que se desee imprimir un reporte exportado a formato PDF.
- **HTTP/S:** protocolo que se utiliza para conectar la computadora del cliente y el servidor donde está la aplicación.
- **TCP/IP:** protocolo para conectar el servidor de aplicaciones con las bases de datos.
- **USB:** protocolo para conectar la impresora con la PC cliente.

2. 13 Conclusiones parciales del capítulo.

En el presente capítulo, con la elaboración del modelo de dominio, fue posible representar las clases conceptuales significativas para el dominio del problema. La captura de requisitos y su agrupación en casos de uso, evidenció las funcionalidades que el sistema debe cumplir y el modelado del diseño permitió evidenciar las clases que intervienen, así como el uso de los patrones de arquitectura MVC y los patrones GRASP y GOF. Los diagramas de secuencia elaborados ofrecen las interacciones entre objetos, ordenadas en secuencia temporal durante un escenario concreto y el diagrama entidad-relación permitió representar las entidades que intervienen en el proceso y la relación entre ellas. Por último, el modelo de despliegue permitió describir la topología del sistema y la estructura de los elementos de *hardware* y *software* que ejecuta cada uno de ellos.

Capítulo 3: Implementación y prueba del componente para la exportación e importación del diseño de reportes.

Introducción.

Luego de obtener el resultado del diseño, en este capítulo se realizan las actividades que se llevan a cabo durante la fase de implementación y prueba. Entre las cuales se encuentran generar los artefactos pertenecientes a ambas fases. Se modela el sistema en términos de componentes, se especifican los tipos, niveles, métodos y casos de pruebas que se realizarán al componente. Además se muestran ejemplos de las implementaciones más relevantes.

Objetivos de la Implementación

Esta disciplina explica cómo desarrollar, organizar, realizar pruebas de unidad e integrar los componentes implementados basándose en las especificaciones del diseño. Tiene como finalidad:

- Definir la organización del código en términos de los subsistemas de implementación, organizados en capas.
- Implementar los elementos del diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros).
- Probar y desarrollar componentes como unidades.
- Integrar los resultados producidos por los implementadores individuales o equipos en un sistema ejecutable.

3.1 Mecanismos de implementación

- El sistema se desarrolla utilizando la arquitectura Modelo-Vista-Controlador.
- Las implementaciones del lado del cliente se realizan haciendo uso generalizado de la librería para el desarrollo de interfaces de usuario ExtJS, versión 2.2.
- Las implementaciones del lado del servidor se implementará en el lenguaje de programación PHP 5.3.10, utilizando Symfony 1.1.7 como *framework* de desarrollo para dicho lenguaje.
- La capa del modelo (acceso a datos) se genera utilizando Propel como ORM incluido en la versión de Symfony a utilizar.

- El intercambio de información entre el cliente y el servidor se hace únicamente en formato JSON[17].
- Todos los métodos, nombres de clases y variables se escribirán en estructura camelCase[18].

3.2 Diagrama de Componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema, muestran y organizan las relaciones de dependencia entre los componentes y subsistemas que lo conforman. Los componentes representan los elementos de un modelo dentro de un paquete, como son las clases en el modelo de diseño. (*Jacobson, 2000*) Normalmente los diagramas de componentes contienen: componentes, interfaces, relaciones de dependencia, generalización, asociación y realización, paquetes y subsistemas.

Estructura general del sistema

La aplicación está compuesta por 3 paquetes de componentes: Web, Action y Model. La carpeta Web presenta la librería ExtJS, y las clases relacionadas con la vista del sistema a desarrollar. En la carpeta Action se encuentran los ficheros php representando las funcionalidades o acciones con los objetos del negocio que le dan funcionalidad al sistema. El componente Model está compuesto por las clases relacionadas con la abstracción y el acceso a datos de la aplicación.

La siguiente imagen muestra el diagrama de componentes del sistema.

[17] **JSON**. Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos, es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

[18] **camelCase:** es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en camelCase se asemejan a las jorobas de un camello.

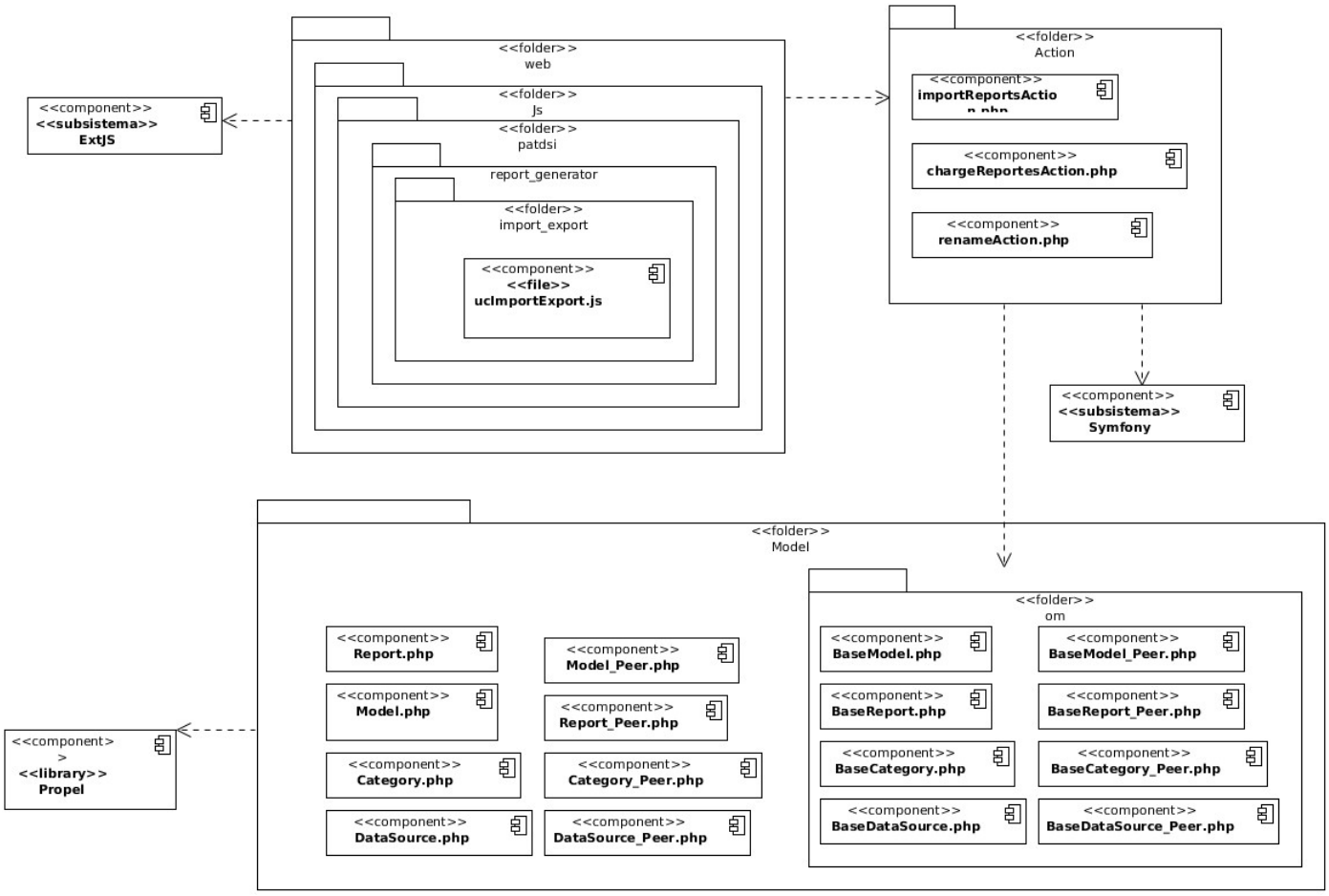


Fig. 11: Diagrama de componentes CU Importar Reportes

3.3 Estándares de codificación

Los estándares de código resultan importantes en cualquier proyecto de desarrollo, pero son especialmente importantes cuando muchos desarrolladores trabajan en el mismo proyecto. Los estándares de código ayudan a asegurar que el código tenga una alta calidad, menos errores, y pueda ser mantenido fácilmente.

PHP

- Para archivos que contengan únicamente código PHP, la etiqueta de cierre ("?>") no está permitida. No es requerida por PHP, y omitirla evita la inyección de espacios en blanco en la respuesta.

- La longitud recomendable para una línea de código es de 80 caracteres.
- Los nombres de clases pueden contener sólo caracteres alfanuméricos.
- La llave "{" deberá escribirse siempre en la línea debajo del nombre de la clase.
- Los nombres de funciones pueden contener únicamente caracteres alfanuméricos. Los guiones bajos (_) no están permitidos.
- Los métodos dentro de clases deben declarar siempre su visibilidad usando un modificador *private, protected, o public*.
- Los nombres de variables pueden contener caracteres alfanuméricos. Los nombres de variables deben empezar siempre con una letra en minúscula y seguir la convención "camelCase".

JavaScript

- Se deben emplear cuatro espacios como unidad de indentación.
- Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.
- Los comentarios de implementación están delimitados por /*...*/, y //, y son para comentar el código o para comentarios acerca de una implementación particular.
- Los nombres de las clases deben ser sustantivos, cuando son compuestos tendrán la primera letra de cada palabra que lo forma en minúscula.
- Las funciones deben ser verbos, cuando son compuestos tendrán la primera letra en minúscula, y la primera letra de las siguientes palabras que lo forma en mayúscula.
- Todas las instancias y variables o funciones empezarán con minúscula.

3.4 Ejemplos de código

PHP

```php
private function reportsByCategory($category)
    {
        $category = $category == NULL ? 0 : $category;
        $id = $this->getUser()->getAttribute('iduser');
        $reports = Report::getByCategory($category, $id);
        $json = array();

        foreach ($reports as $report)
        {
            $array = array();
            $array['id'] = $report->getIdreport() . "#" . $category;
            $array['text'] = $report->getTitle();
            $array['idreport'] = $report->getIdreport();
            $array['leaf'] = true;
            $array['checked'] = false;
            $array['icon'] = '/images/icons/report.ico';
            $array['qtip'] = $report->getTitle();
            $json[] = $array;
        }
        return $json;
    }
```

Fig. 12: Función que organiza los reportes por categorías

JavaScript

```
function ReportsToExport(tree, reports) {
    if (tree === undefined)
        return;
    if (tree.isLeaf()) {
        reports[reports.length] = tree.attributes.id;
    }
    if (tree.hasChildNodes()) {
        tree.eachChild(function(n){
            ReportsToExport(n, reports);
        });
    }
}

function ObtenerReportes(tree, reportes){
    if (tree === undefined)
        return;
    if (tree.isLeaf()) {
        reportes[reportes.length] = tree.attributes.id;
    }
    if (tree.hasChildNodes()) {
        tree.eachChild(function(n){
            ObtenerReportes(n, reportes);
        });
    }
}
```

Fig. 13: Funciones para seleccionar los reportes

3.5 Pruebas de software

Las pruebas de *software* permiten verificar y revelar la calidad de un *software*. Consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados de por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador; probando el comportamiento del mismo. (*DLSI, 2014*)

3.5.1 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. (*DLSI, 2014*) En la presente investigación se propuso realizar pruebas de a nivel de desarrollador, aplicando las de tipo funcional y el método de caja negra, utilizando la técnica partición de equivalencia.

3.5.2 Nivel de prueba

Los niveles de prueba especifican diferentes ángulos para verificar y validar un producto de *software*. 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. (*DLSI, 2014*)

3.5.3 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. (*DLSI, 2014*)

3.5.4 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. (*DLSI, 2014*)

Para desarrolla este método se decide aplicar la técnica de partición de equivalencia. La misma permite examinar los valores válidos en inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores, que de otro modo, requerirían la ejecución de muchos casos de pruebas antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de casos de prueba que hay que desarrollar.

3.6 Casos de Prueba

Un caso de prueba se diseña según las funcionalidades descritas en los casos de uso. El propósito que se persigue con este artefacto es lograr una comprensión común de las condiciones específicas que la solución debe cumplir. Se parte de la descripción de los casos de uso del sistema, 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. Además quedan plasmadas las revisiones realizadas al caso de prueba. Se efectuaron los casos de pruebas a los dos casos de uso del sistema, plasmándose en el expediente de proyecto de la fase de pruebas en la planilla de caso de pruebas del proyecto.

A continuación se presenta la tabla de secciones a probar del caso de uso Exportar Reportes:

Tabla 11: Secciones a probar en el caso de uso Exportar Reportes

Nombre de la Sección	Escenarios de la Sección	Descripción de la funcionalidad
SC1: Exportar Reportes	EC 1.1: Exportar Reportes	El sistema debe mostrar al usuario una lista con los reportes almacenados en el sistema. Debe permitir seleccionarlos para ser exportados. De la misma forma se solicitará el directorio donde se desea almacenar el archivo XML que contiene los reportes

Tabla 12: Descripción de las variables

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Lista de reportes	Listbox	No	Una lista de reportes almacenados en la base de datos del sistema

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 identificó el empleo de la técnica de partición de

equivalencia.

Matriz de datos

Tabla 13: Sección Exportar Reportes

Escenario	Descripción	Variable (1)	Respuesta del sistema	Flujo Central
EC 1.1 Exportar los reportes de forma correcta	Se exportan los reportes correctamente.	V (Se seleccionan los reportes)	Se realiza la exportación de los reportes seleccionados de forma correcta.	1. Se selecciona la opción “Exportar Reportes”. 2. Se seleccionan los reportes 3. Se selecciona la opción “Exportar”. 4. Se selecciona la ruta para guardar el archivo XML.
EC 1.2 Exportar los reportes de forma correcta	Se exportan los reportes sin ser seleccionados. Se muestra un mensaje de error	( )	Se muestra un mensaje que la exportación no se realizó.	1. Selecciona la opción “Exportar Reportes”. 2. No se selecciona ningún reporte. 3. Se selecciona la opción “Exportar 4. Se muestra un mensaje de error en la exportación.

3.6.1 Resultado de la ejecución las pruebas funcionales

Luego de aplicar las pruebas funcionales se arrojan diferentes resultados, pero en esencia, se busca mayor eficacia y eficiencia a la hora de encontrar defectos, verificando la calidad, de forma tal que se

minimicen tiempos, costos, y se obtenga un resultado satisfactorio. El sistema desarrollado después de aplicarle las pruebas arrojó ocho no conformidades significativas, las cuales se le dio seguimiento y fueron resueltas paulatinamente.

Para la primera iteración se detectaron cinco no conformidades (tres altas, una media y una baja). En la segunda se obtuvieron dos no conformidades (dos altas) y en la tercera iteración no se identificaron no conformidades. La siguiente figura muestra la cantidad de no conformidades por cada iteración realizada.

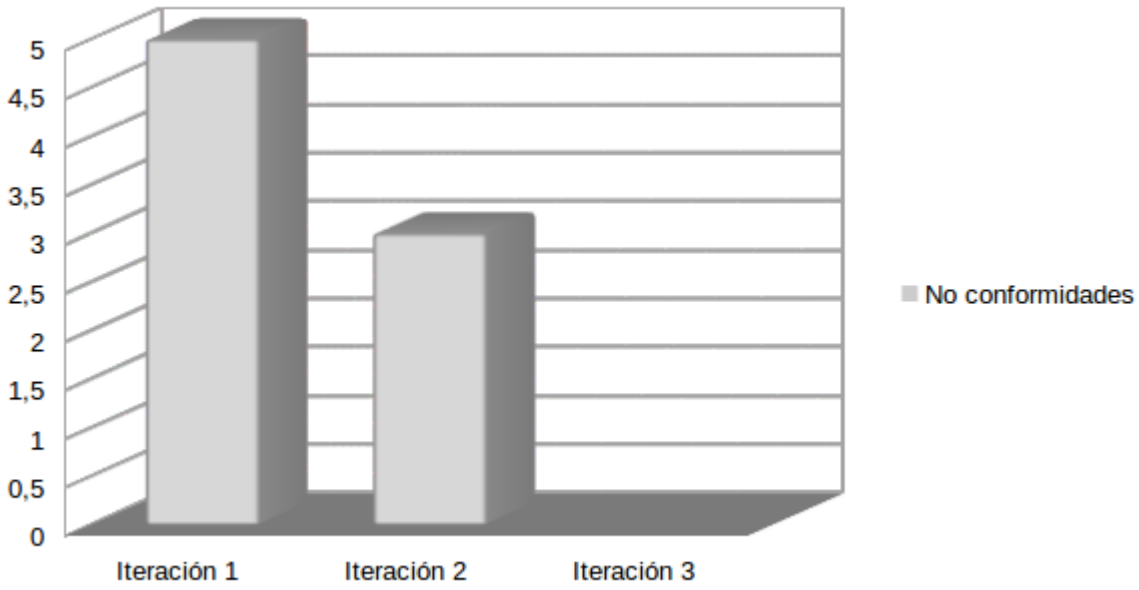


Fig. 14: Pruebas funcionales

3.7 Conclusiones parciales del capítulo

En este capítulo de realizó el modelado de la implementación del componente ofreciendo solución a los requisitos especificados en el capítulo anterior. La elaboración del diagrama de componentes permitió mostrar las relaciones entre sus elementos. Los estándares de codificación presentes en la implementación del componente permitieron asegurar que el código posee una alta calidad, menos errores, y puede ser mantenido fácilmente. Por último, el método de caja negra basado en la técnica de partición de equivalencia, junto con las pruebas funcionales permitieron validar el correcto funcionamiento del componente para exportación e importación del diseño de reportes.

Conclusiones.

Con la realización de este trabajo de diploma se obtuvo una propuesta que da cumplimiento al objetivo general planteado, al lograr desarrollar un componente para la exportación e importación del diseño de reportes para el GDR 1.8, llegando a las siguientes conclusiones:

- El estudio de los principales aspectos relacionados con la exportación e importación del diseño de reportes posibilitó identificar las funcionalidades necesarias que luego fueron implementadas en el componente.
- El análisis y diseño del componente permitió obtener los artefactos para y diagramas necesarios para guiar la implementación del mismo.
- La implementación del componente permitió dar cumplimiento a los 8 requisitos funcionales identificados en la fase de análisis y diseño.
- El diseño y ejecución de pruebas del sistema permitieron comprobar el correcto funcionamiento del componente para la exportación e importación del diseño de reportes.

Recomendaciones.

Después de haber cumplido con los objetivos trazados en el presente trabajo de diploma se propone:

- Desarrollar la lógica del sistema para la versión 2.0 del Generador Dinámico de Reportes e integrarlo con el módulo Administrador de Reportes.

Referencias Bibliográficas.

1. **González Abreu, Luis Eduardo y Fonseca Guzmán, Mónica.** *Módulo de gestión para los reportes estadísticos.* UCI. La Habana: UCI, 2010.

2. Definición.de. [En línea] [Citado el: 11 de Noviembre de 2013.] Disponible en: http://definicion.de/reporte/.

3. **Infante Frómeta, Jenny y Hernández, Yasmany Hernández.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* La Habana: UCI, 2009.

4. iReport. [En línea] [Citado el 11 de Noviembre de 2013] Disponible en: http://community.jaspersoft.com/project/ireport-designer.

5. Oracle. [En línea] [Citado el 11 de Noviembre de 2013] Disponible en: http://www.oracle.com/technetwork/middleware/reports/overview/index-100240.html.

6. OracleAS Reports Services and XML. [En línea] [Citado el 13 de Noviembre de 2013] Disponible en: http://isu.ifmo.ru/docs/IAS904/web.904/b12099/adx14rep.htm.

7. Comunities Right Now**.** [En línea] [Citado el 12 de Noviembre de 2013] Disponible en: http://communities.rightnow.com/posts/279bac570b.

8. BIRT. [En línea][Citado el: 12 de Noviembre de 2013] Disponible en: http://www.eclipse.org/birt/phoenix/project/description.php.

9. Developer.com. [En línea][Citado el 12 de Noviembre de 2013] Disponible en: http://www.developer.com/xml/article.php/10929_3732446_4/Developing-an-Eclipse-BIRT-XML-Report-Rendering-Extension.htm.

10. **Rodríguez Durán, Aurelio.** *Impacto del Generador Dinámico de reportes en los Sistemas de Gestión de Información.* UCI, La Habana: 2012.

11. Definicion.de. [En línea] [Citado el: 13 de Noviembre de 2013] Disponible en: http://definicion.de/componente/

12. **Montilva, Jonas, Arapé, Nelson Y Colmenares, Juan.** Desarrollo de *Software* Basado en Componentes. En: *Actas del IV Congreso de Automatización y Control*, Mérida, nov 2003. 9p

13. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado del Desarrollo de Software.* 2000.

14. Gestión de Proyectos. *OpenUP como alternativa metodológica para proyectos pequeños de software. 2008.* [En línea] [Citado el 13 de Noviembre de 2013] Disponible en:

http://kasyles.blogspot.com/2008/09/openup-como-alternativa-metodolgica.html.

15. Ecured**.** *Lenguaje de Programación.* [En línea][Citado el 13 de Noviembre de 2013] Disponible en: http://www.ecured.cu/index.php/Lenguaje_de_programación.

16. Php.net. [En línea][Citado el 14 de Noviembre de 2013] Disponible en: http://www.php.net/.

17. **Eguíluz Pérez, Javier.** *Introducción a JavaScript.* 2009.

18. Ecured**.** *Framework* [En línea][Citado el 14 de Noviembre de 2013] Disponible en: http://www.ecured.cu/index.php/Framework.

19. **Potencier, Fabien y Zaninotto, Francisco.** *Symfony, la guía definitiva.* 2008.

20. UNIDAD VI. Herramientas CASE. [En línea][Citado el 14 de Noviembre de 2014] Disponible en: http://docente.ucol.mx/al961223/public_html/centro6.htm.

21. **Frederick, Shea.** *Learning ExtJS.* Birmingham, UK: Packt Publishing, 2012.

22. **Yanes León, Vania Elena.** *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0.* UCI, La Habana: 2011.

23. **Postgresql**. Comunidad Técnica de Desarrollo. [En línea][Citado el 21 de Noviembre de 2013] Disponible en: http://postgresql.uci.cu/?page_id=30.

24. Guía Ubuntu**.** PgAdminIII. [En línea][Citado el 21 de Noviembre de 2013] Disponible en: http://www.guia-ubuntu.com/index.php?title=PgAdmin_III.

25. NetBeans. [En línea][Citado el 21 de Noviembre de 2013] Disponible en: https://netbeans.org/index_es.html.

26. Ivanek. [En línea][Citado el 21 de Noviembre de 2013] Disponible en: http://ivanex.wikidot.com/patron-arquitectura.

27. **Gamma, Erich**. What is a Design Pattern? En: *Design Patterns: Elements of Reusable Objected-Oriented Software,* p 12.

28. **Gamma, Erich**. Design Pattern Catalog. En: *Design Patterns: Elements of Reusable Objected-Oriented Software,* p 93.

29. **Hernández, Pedro Veloso.** Uso de patrones de arquitectura. [En línea] [Citado el: 23 de Noviembre de 2013.] Disponible en: http://static2.docstoccdn.com/search/patrones-de-arquitectura.

30. **Larman, Craig**. Modelo Del Dominio: Visualización De Conceptos. En: Dawn Speth White. *UML y Patrones*. México, 1999, p 96-98.

31. **Larman, Craig**. Arquitectura de *Software*. En: Dawn Speth White. *UML y Patrones*. México, 1999, p

356.

32. **Potencier, Fabien y Zaninotto, Francisco.** El patrón MVC. En: *Symfony, la guía definitiva,* 2008, p 24.

33. **Marquina, Ernesto**. Patrones de Diseño. En: *Guía de patrones, prácticas y arquitectura .NET*, 2008, p 5.

34. **Larman, Craig**. GRASP: Patrones de Principios Generales para Asignar Responsabilidades En: Dawn Speth White. *UML y Patrones*. México, 1999, p 163.

35. **Larman, Craig**. Diseño De Las Realizaciones De Casos De Uso Con Los Patrones De Diseño GoF En: Dawn Speth White. *UML y Patrones*. México, 1999, p 271.

36. **Larman, Craig**. Modelo De Casos De Uso: Representación De Los Diagramas De Secuencia Del Sistema. En: Dawn Speth White. *UML y Patrones*. México, 1999, p 89.

37. **Jacobson, Ivar; Rumbaugh, James; Booch, Grady.** *El proceso unificado de desarrollo*. 2000. Addison Wesley. capítulos 9 p 205-254

38. Modelo de datos. [En línea][Citado el 11 de Enero de 2014] Disponible en: http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/.

39. Ingenieriasoftwaredos. [En línea][Citado el 15 de Enero de 2014] Disponible en: http://www.ingenieriasoftwaredos.wikispaces.com/Diagramas+de+Artefactos+y+despliegue

40. Departamento de Lenguajes y Sistemas Informáticos. Universidad de Granada. [En línea] [Citado el 10 de Febrero de 2014] Disponible en: http://lsi.ugr.es/lsi/

Bibliografía.

1. **Barchini, Graciela E**. *Métodos "I+D" de la Informática*. Universidad Nacional de Santiago del Estero. Argentina.
2. **Billy, Carlos R.** *Introducción a la Arquitectura de Software.* Universidad de Buenos Aires. 2004.
3. **Brito Rodríguez, Julio César.** *Módulo Diseñador de Modelos para el Generador*. UCI, La Habana:
4. 2012.
5. **Camacho, Erika.** Arquitecturas de Software. Guía de Estudio. 2004.
6. **Clarke, Charles L. A.** Controlling Overlap in Content-Oriented XML Retrieval. University of Waterloo. Canada. 2010.
7. **Del Toro, Alberto**. *Biblioteca de componentes para la implementación de la interfaz web del Gestor de Documentos Administrativos eXcriba*. UCI. 2012.
8. **Domínguez, Evelyn y Delgado, Ariadne.** *Proceso de Pruebas para el Módulo Mercantil del Proyecto Registros y Notarías*. UCI. 2007
9. **Eguíluz Pérez, Javier.** *Introducción a JavaScript.* 2009.
10. **Frederick, Shea.** *Learning ExtJS.* Birmingham, UK: Packt Publishing, 2012.
11. **Gamma, Erich**. *Design Patterns: Elements of Reusable Objected-Oriented Software.*
12. **García Suárez del Villar, Claudia**. *Herramienta para importar los reportes desde la versión 1.7 del Generador Dinámico de Reportes a su versión 2.0.* UCI, La Habana: 2012.
13. **González, Luis E y Fonseca, Mónica.** *Módulo de gestión para los reportes estadísticos de alas RIS*. UCI. 2010.
14. **Hernández Hernández, Yasmany**. *Pautas de arquitectura para el desarrollo de la versión 2 del GDR*.
15. **Hernández, Rolando Alfredo y Coello González, Sayda.** *El proceso de investigación científica*. Editorial Universitaria. 2011.
16. **Hernández, Roberto Sampieri**. *Metodología de la Investigación.* México. 2006.
17. **Hernández, Juan J.** *Desarrollo del módulo Administrador de Reportes del Generador Dinámico de Reportes*. UCI. 2011.
18. **Infante Frómeta, Jenny y Hernández, Yasmany Hernández.** *Arquitectura de Software para el Sistema de Gestión de Reportes Dinámicos.* La Habana: UCI, 2009.

19. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado del Desarrollo de Software.* 2000.
20. **Larman, Craig**. UML y Patrones. México, 1999.
21. **Letelier, Patricio**. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Universidad Politécnica de Valencia
22. **Lockhart, Thomas.** *Guía del administrador de PostgreSQL*. [En línea] Disponible en: http://sunshine.prod.uci.cu/book/4f26bcfb0571746662000004/.
23. **Lluch, Mayelín.** *Diseñador gráfico de reportes para el Reporteador Dinámico de la plataforma alasGRATO*. UCI. 2010.
24. **Martínez, Yanoski y Lafaurie José R**. *Sistema para la generación de reportes en la plataforma alasGRATO: Desarrollo del módulo "Reportador".* UCI. 2008
25. **Navarro, Ana N. Rodríguez**. *Asistente de Reportes para el módulo Diseñador de Reportes del Generador Dinámico de Reportes versión 2.0*.UCI. 2011
26. **Potencier, Fabien.** *Practical symfony, second edition.* 2009.
27. **Potencier, Fabien.** *Symfony forms in action*. 2009.
28. **Potencier, Fabien.** *The symfony Reference Book.* 2009.
29. **Pressman, Roger S**. *Ingeniería de Software. Un enfoque práctico. 5ta Edición*. 2005.
30. **Reynoso, Carlos y Kiccillof Nicolás**. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Universidad de Buenos Aires. 2004.
31. S**alazar, Lianet.** *Administrador de Reportes para la versión 2.0 del Generador Dinámico de Reportes*. UCI. 2012.
32. **Sotolongo, Anthony R. León.** *Compendio de consultas útiles al catálogo de postgreSQL*. Universidad de las Ciencias Informáticas. La Habana, Cuba. 2011.
33. **Valle, Mabel Laborde.** *Generador Dinámico de Reportes V 2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes*. UCI, 2011
34. **Yanes León, Vania Elena.** *Definición de la arquitectura de software para el Generador Dinámico de Reportes en su versión 2.0.* UCI, La Habana: 2011.

Anexos.

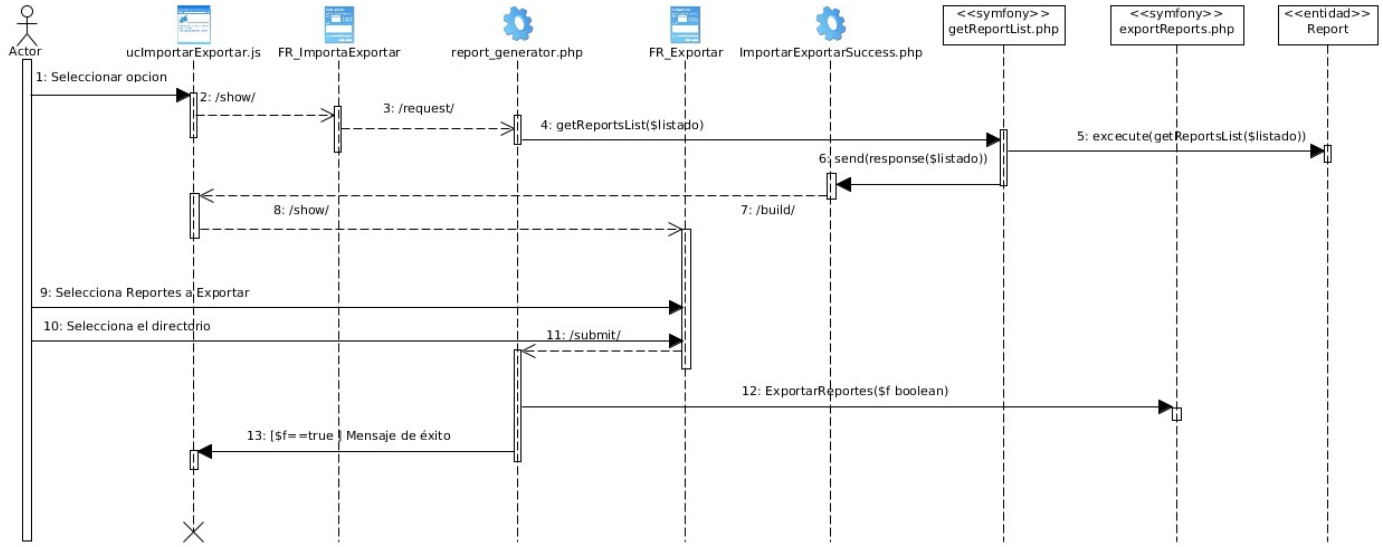


Fig. 15: Diagrama de secuencia CU Exportar Reportes

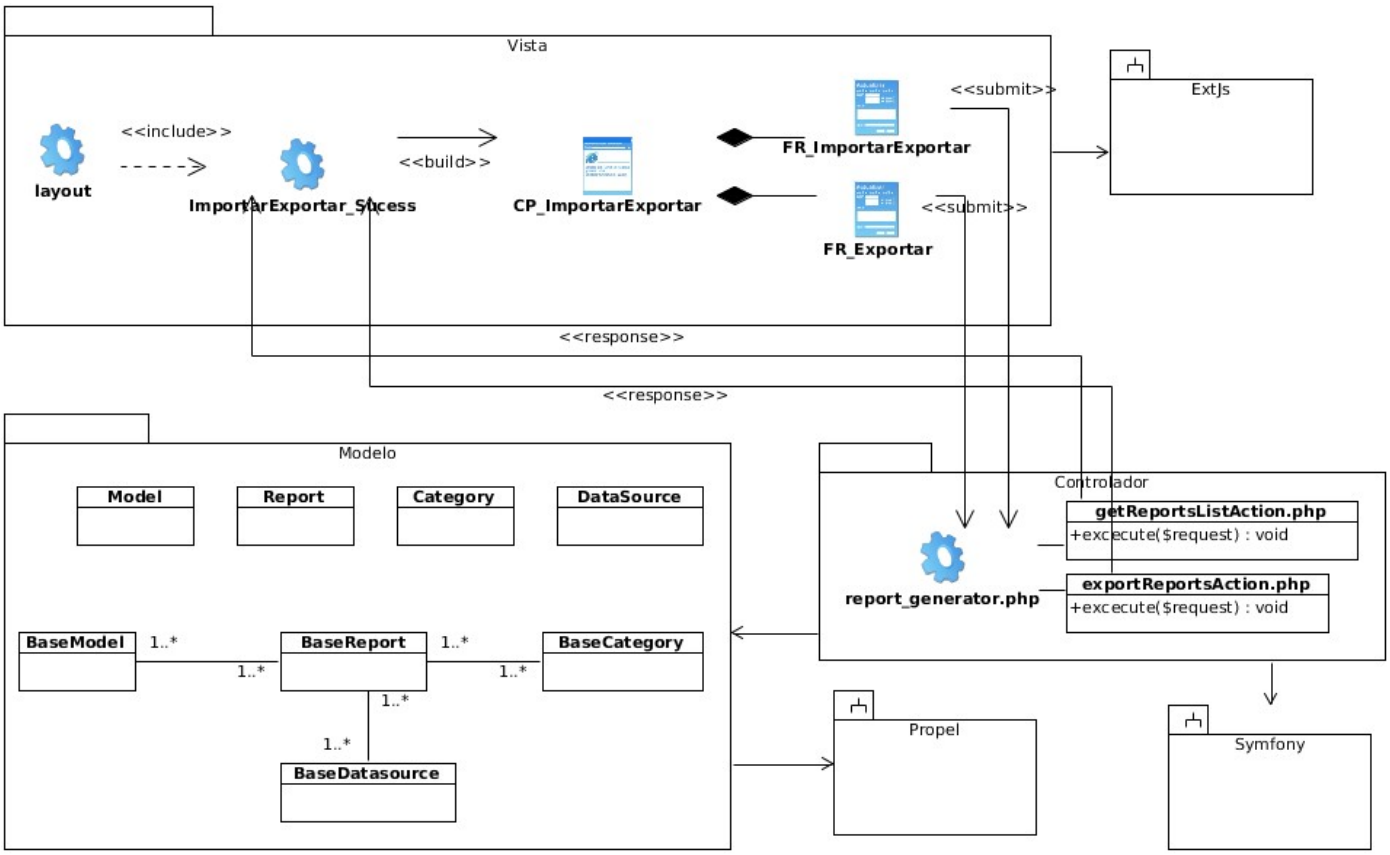


Fig. 16: Diagrama de clases del diseño del CU Exportar Reportes

Glosario de Términos.

CASE: Ingeniería de Software Asistida por Computación.

CU: Caso de Uso.

DATEC: Centro de Tecnologías de Gestión de Datos.

Framework: Marco de Trabajo.

GDR: Generador Dinámico de Reportes.

GOF: Banda de los Cuatro.

GRASP: Patrones de Software para la Asignación General de Responsabilidad.

IDE: Entorno Desarrollo Integrado.

JSON: Notación de Objetos de JavaScript.

MVC: Modelo – Vista – Controlador.

ONEI: Organización Nacional de Estadísticas en Información.

ORM: Mapeo de Objetos a Bases de Datos.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje Unificado de Modelado.

XML: Lenguaje de Marcas Extensible.