Universidad de las Ciencias Informáticas

Facultad 6.

Título: “Generador Dinámico de Reportes V 2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes.”

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

Autor: Mabel Valle Laborde.

Tutores: Ing. Javier Alfonso Valdés.
Ing. Luis Ernesto Saballo López.

Ciudad de La Habana, Cuba

Junio 2011.

“…Lo que caracteriza al hombre de ciencia no es la posesión del conocimiento o de verdades irrefutables, sino la investigación desinteresada e incesante de la verdad…”

KARL POPPER.

Declaración de Autoría

Declaro que soy la única autora del trabajo titulado “Generador Dinámico de Reportes V2.0: Análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes” y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

_____

Firma del Autor

Mabel Valle Laborde

_____ _____

Firma del Tutor Firma del Tutor

Ing. Luis Ernesto Saballo López Ing. Javier Alfonso Valdés

Datos de Contacto

Tutores:

Tutor: Ing. Javier Alfonso Valdés
Especialidad de graduación: Ingeniero en Ciencias Informáticas
Categoría Científica: Ingeniero
Años de experiencia en el tema: 1
Años de graduado: 1
Correo Electrónico: jalfonso@uci.cu

Tutor: Ing. Luis Ernesto Saballo López
Especialidad de graduación: Ingeniero en Ciencias Informáticas
Categoría docente: Instructor
Categoría Científica: Ingeniero
Años de experiencia en el tema: 5
Años de graduado: 5
Correo Electrónico: lsaballo@uci.cu

Agradecimientos

Muchas han sido las personas que a lo largo de estos años, han contribuido a que hoy yo esté cumpliendo mi sueño.

Primeramente quiero agradecer a mis padres que durante toda mi vida me han alentado a que haga de mí una mejor persona, que han estado al lado mío a pesar de todo, dándome cariño y soporte.

A mi abuelita y a mi hermano querido que son toda mi vida.

A mi novio, que me ha consolado cuando nadie ha podido, que me ha dado las fuerzas para seguir adelante, que me ha levantado cuando me he caído, que me ha hecho reír cuando lloraba desconsoladamente y que me ha dado el amor que nunca había conocido.

A mis amigos que han sido muy importantes para mí, que de igual manera me han ayudado a llegar tan lejos, que me han dado su amistad sin pedir nada a cambio, a los que tanto quiero y aprecio.

A Marleysi López Duque me extendió su mano cuando más la necesitaba, muchas gracias.

Por último agradecer a los que de una forma u otra han contribuido a lo largo de todo este tiempo, a mis compañeros de aula, profesores y a todos los que se han cruzado en mi vida y de una forma u otra han puesto su granito de arena. De todo corazón muchísimas gracias.

A todos y cada uno de ustedes gracias…… los amo.

Dedicatoria

……A mis abuelos Pepe y Pedro, que descansen en paz….

…A mi abuelita Victoria…

…A mis padres, en especial…

…A mi hermano…

… A mi novio…

Resumen

El presente trabajo propone un análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes del Sistema Generador Dinámico de Reportes V2.0, para dar solución a muchas de las deficiencias que actualmente presenta el sistema y alcanzar mayores funcionalidades para el trabajo con reportes. Para resolver el trabajo investigativo, se realizó un estudio de las tendencias mundiales de los generadores de reportes, así como de las prestaciones que los mismos ofrecen. Se desarrolló una propuesta de solución al problema, representando el modelo del dominio que expresa la relación de las entidades de los módulos, se realizó un análisis de cómo funcionará el sistema evidenciándose en los diagramas que propone la metodología, como los diagramas de Casos de Uso, los diagrama de clases y de colaboración del análisis. Se presentan las descripciones de los Casos de Uso más significativos y se validan los resultados de la solución para verificar que posean la calidad requerida; con el objetivo de que a partir del análisis realizado se logre implementar un sistema con mayores funcionalidades y mejores resultados que la versión precedente.

Palabras Clave: reportes, modelos, orígenes de datos.

Tabla de Contenido

Agradecimientos I

Dedicatoria II

Resumen III

Introducción 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

Introducción. 5

1.1 Sistemas Generadores de Reportes. 5

1.1.1 Herramientas de generación de reportes en el mundo. 5

1.1.2 Herramientas de generación de reportes en la UCI. 9

Conclusiones Parciales. 10

1.2 Metodología y Herramientas para la modelación del software. 10

1.2.1 Metodología. 10

1.2.2 Lenguaje de Modelado. 11

1.2.3 Herramientas CASE. 11

1.2.4 Herramientas para la Gestión de Requisitos. 12

1.2.5 Herramientas de Estimación. 13

1.3 Ingeniería de Requisitos. 14

1.3.1 Características de los Requisitos. 15

1.3.2 Estudio de Viabilidad. 16

1.3.3 Captura de requisitos. 16

1.3.3 Análisis de requisitos. 17

1.3.4 Especificación de requisitos. 17

1.3.5 Técnicas de validación de requisitos. 17

1.4 Características del rol Analista. 18

1.4.1 Artefactos generados por el Analista. 19

1.5 Patrones de Casos de Uso. 21
1.6 Conclusiones. 23
CAPÍTULO 2: ANÁLISIS DEL SISTEMA 25
Introducción. 25
2.1 Descripción del Sistema. 25
2.2 Modelo de Dominio. 27
2.2.1 Diagrama Conceptual del Modelo de Dominio. 27
2.3 Especificación de Requisitos Software. 28
2.3.1 Requisitos Funcionales. 28
2.3.2 Requisitos No Funcionales. 38
2.4 Definición de Caso de Uso del Sistema. 42
2.4.1 Definición de los actores. 42
2.4.2 Diagramas de Caso de Uso del Sistema. 43
2.5 Diagramas de Clases y de Colaboración del Análisis. 53
2.6 Gestión de Requisitos. 56
2.7 Estimación de costo del proyecto 58
2.8 Conclusiones. 58
CAPITULO 3: VALIDACIÓN DE LOS REQUISITOS 59
Introducción. 59
3.1 Validación de requisitos. 59
3.2 Validación por fases. 60
3.2.1 Fase I: Validación del listado de requisitos de software. 60
3.2.2 Fase II: Validación del documento de especificación de requisitos. 61
3.2.3 Fase III: Validación del Modelo de Caso de Uso del Sistema. 63
3.3 Conclusiones. 64

Índice de Tablas y Figuras.

Tabla 1: Descripción de los actores. 1 42

Tabla 2: Descripción del Caso de Uso Diseñar Modelo. 1 43

Tabla 3: Descripción del Caso de Uso Diseñar Reporte. 1 48

Figura 1: Flujo de generación de reporte 1 25

Figura 2: Modelo de Dominio. 1 27

Figura 3: Diagrama de caso de uso del sistema para el módulo Diseñador de Modelo. 1 43

Figura 4: Diagrama de caso de uso del sistema para el módulo Diseñador de Reportes. 1 47

Figura 5: Diagrama de Clases del Análisis del módulo Diseñador de Modelos (CUS Diseñar Modelo) 1 53

Figura 6: Diagrama de Clases del Análisis del módulo Diseñador de Reportes (CUS Diseñar Reporte) 1 54

Figura 7: Diagrama de Colaboración del módulo Diseñador de Modelos (CUS Diseñar Modelo, escenario Adicionar Modelo) 1 55

Figura 8: Diagrama de Colaboración del módulo Diseñador de Reportes (CUS Diseñar Reporte, escenario Crear reporte columnar en blanco). 1 55

Figura 9: Matriz de Trazabilidad. 1 57

Figura 10: Elementos definidos en la lista de chequeo 1 62

Figura 11: Elementos definidos en la lista de chequeo 1 63

Introducción

La información es un elemento vital para cualquier empresa o institución, influye de manera directa en la forma en que estas operan. Los sistemas de información están conformados por un conjunto de componentes capaces de realizar operaciones de procesamiento con múltiples datos para generar información.

En todo sistema de información es fundamental contar con una herramienta cuya función sea la de gestionar reportes, que son objetos que entregan información en un formato particular y que permiten realizar ciertas operaciones como: imprimirlos, enviarlos por correo electrónico y guardarlos a un archivo, a partir de los datos almacenados en una base de datos. En este sentido, los generadores de reportes son herramientas adicionales a los sistemas de información. Los generadores de reportes tienen definido un mecanismo para realizar consultas a la base de datos y obtener información de ella en forma de reporte. A diferencia de las consultas tradicionales, con los generadores de reportes se tiene mayor control sobre el diseño y la forma en que la información será visualizada.

La mayoría de las organizaciones utilizan reportes para registrar y visualizar análisis y resultados. Debido a ello, los reportes son considerados una necesidad principal del negocio, acorde con esto existen en el mundo diferentes herramientas que viabilizan el proceso de generación de reportes.

El Centro de Tecnologías de Gestión de Datos de la Universidad de las Ciencias Informáticas (DATEC), como parte del proceso de informatización de la sociedad ha creado un sistema de Gestión Dinámica de Reportes, cuya finalidad es obtener un grupo de aplicaciones capaces de crear, administrar y entregar reportes dinámicos en tiempo real o programado.

En la actualidad dicho sistema cuenta con una serie de módulos que brindan funcionalidades para el trabajo con reportes tales como:

- Diseñador de Modelos.
- Diseñador de Reportes.
- Visor de Reportes.
- Administrador de Reportes.
- Diseñador de Consulta.

Siendo una de sus principales ventajas el dar soporte para todo el ciclo de vida del reporte. Sin embargo presenta serias deficiencias que se deben resolver para lograr un mejor producto, ellas son que la tecnología usada para su creación, ha caducado, debido a que a la librería de reportes no ha recibido soporte, presentando una ausencia de potencialidades necesarias presentes en otros sistemas de este tipo, restándole eficiencia y eficacia. No permite la generación de sub-reportes ni la división de reportes por tuplas. Además las malas prácticas de programación durante su desarrollo, hace imposible agregarle nuevas funcionalidades provocando descontento en los clientes y no poder realizar una mejor comercialización del producto.

Como consecuencia de estos fallos y la necesidad de brindarle a los usuarios finales un producto confiable, seguro, eficiente, con todas las funcionalidades y potencialidades necesarias para la creación de un reporte que ayude a la toma de decisiones en una determinada empresa, DATEC se propone realizar una versión 2.0 del Generador Dinámico de Reportes con el objetivo de aumentar la estabilidad, confiabilidad y seguridad de los reportes dinámicos creados con este producto y ser lo suficientemente genérico para ser utilizado en cualquier otra institución que requiera los servicios de un sistema de Gestión Dinámica de Reportes.

Por lo antes expuesto, surge el presente trabajo, centrándose en el siguiente **problema de la investigación**:

¿Cómo definir los elementos necesarios para el desarrollo de los módulos Diseñador de Modelos y Diseñador de Reportes del Generador Dinámico de Reportes V2.0?

Para obtener una solución al problema planteado se define como **objeto de estudio** el proceso de desarrollo de software del sistema de Gestión Dinámica de Reportes, enmarcado en el **campo de acción** el análisis de los módulos Diseñador de Modelos V2.0 y Diseñador de Reportes V2.0.

Se define como **Objetivo General:**

Realizar el análisis de los módulos Diseñador de Modelos y Diseñador de Reportes del Generador Dinámico de Reportes V2.0.

Desglosado en los siguientes **Objetivos Específicos:**

Realizar el análisis de las herramientas para la creación de reportes.
Realizar el análisis de la solución.
Validar el análisis propuesto.

Para alcanzar dichos objetivos, se planteó realizar las siguientes **tareas**:

- Análisis del marco conceptual teórico acerca de sistemas para la generación de reportes desarrollados.
- Selección de la herramienta generadora de reportes a utilizar.
- Definición de una metodología de desarrollo a emplear.
- Entendimiento del negocio.
- Definición de Modelo de Dominio o del Negocio, según corresponda.
- Elaboración de los diagrama de casos de uso del negocio.
- Definición de los actores del sistema.
- Especificación de los casos de uso del sistema.
- Definición de los requisitos funcionales y no funcionales.
- Elaboración del diagrama de caso de uso del sistema.
- Desarrollo del diagrama de clases del análisis.
- Validación de los resultados mediante prototipos funcionales.
- Validación de los resultados mediante revisiones.
- Validación de los resultados obtenidos mediante métricas definidas.

Estructura de la Tesis

La tesis quedó estructurada en 3 capítulos:

Capítulo 1 Fundamentación Teórica: Estudio bibliográfico del tema a tratar a nivel internacional, nacional y de la universidad, así como la fundamentación de las técnicas, metodologías, herramientas y lenguajes para el posterior análisis.

Capítulo 2 Análisis del Sistema: Se establecen los requisitos funcionales y no funcionales, se realiza una descripción de los casos de uso del sistema, se muestra el diagrama de casos de uso del sistema, así como el diagrama de clases del análisis.

Capítulo 3 Validación por Métricas: Se realiza el análisis de los resultados que incluye los pasos para la validación de requisitos, técnica de validación y las pruebas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción.

En este capítulo se abordarán temas relacionados con las diferentes herramientas que existen en el mundo para la generación de reportes, sus características y un análisis crítico de las mismas a través de una comparación, teniendo en cuenta los aspectos más relevantes. Se hará referencia al proceso de desarrollo de software, así como las metodologías de desarrollo, tecnologías y tendencias actuales que pueden ser útiles en la propuesta de solución, argumentando el por qué de su selección.

1.1 Sistemas Generadores de Reportes.

Antiguamente los reportes se confeccionaban en formato duro, no existía una herramienta capaz de realizarlos. En la actualidad el empleo del formato digital ha venido ganando terreno debido a su facilidad y la presencia de aplicaciones que son capaces de generarlos automáticamente. Las aplicaciones de gestión de información tienen cada vez más tendencia al uso de las tecnologías web y de manera progresiva, en los últimos años han ido aumentando los desarrollos bajo estas arquitecturas. Dichos programas, denominados generadores de reportes, permiten al usuario realizar de forma transparente, consultas a la base de datos y obtener información de ella en un determinado formato, con una mayor rapidez y un mayor nivel de detalle y flexibilidad.

1.1.1 Herramientas de generación de reportes en el mundo.

En el mundo existen diferentes herramientas que viabilizan el proceso de generación de reportes, tales son los casos de Active Reports, Agata Reports, Crystal Reports, Jasper Reports, cuyas principales características son:

Active Reports.

Active Reports es un componente de informes .NET para aplicaciones Windows Forms y de formularios Web. Entre las características claves figuran la personalización, un rendimiento rápido, alta calidad y prestaciones multilingües: todas ellas contrastadas mediante su uso en decenas de miles de aplicaciones en todo el mundo. Admite exportaciones de datos a todos los formatos de archivo habituales, como PDF, Excel, RTF, TIFF, etc. Incluye un diseñador de informes Visual Studio .NET fácil de usar y una potente API. Posee soporte multilingüe listo para ser utilizado. También ofrece un despliegue de tiempo de ejecución armonizado y libre de derechos.

Características Principales:

- El diseñador de informes se integra perfectamente en los entornos de desarrollo Visual Studio .NET. Una vez instalado el producto en el equipo del desarrollador, la adición de un informe a un proyecto es tan fácil como añadir una clase o un formulario.
- Los informes se crean dentro de Visual Studio .NET y se compilan directamente en el ejecutable. Por consiguiente, los ensamblados pueden distribuirse mediante despliegue Xcopy o colocarse en la Caché de ensamblados global (Global Assembly Cache: GAC). El modelo de objetos proporciona el motor de generación de informes como un único ensamblado administrado con nombre seguro. No es necesario realizar ninguna otra configuración adicional en el servidor o el equipo cliente.
- Dado que Active Reports es totalmente administrado, no presenta dependencias de aplicaciones de terceros.

Requerimientos del Programa:

- Procesador: Pentium® II-class procesador 450 MHz (Pentium® III 600 MHz recomendado).
- RAM: 200 MB.
- Espacio en disco duro: 50 MB disponible.
- Sistema Operativo: Windows 2000, Windows XP, Windows NT 4.0, Windows Vista, Windows 7, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2.
- Microsoft .NET Framework Versión: 2.0 o superior
- Microsoft Visual Studio: 2005, 2008, ó 2010. Note: La edición express de Visual Studio 2005 no trabaja Active Reports.
- Para desarrollo web: IIS 5.0, 5.1, 6.0, 7.0 o 7.5 and ASP.NET.

Licencia: Se distribuye bajo licencia privativa perteneciente a GrapeCity, inc.

Agata Reports.

El Agata Report es una herramienta libre, liberada bajo la licencia GNU GPL y escrita en PHP GTK que se caracteriza por ser multiplataforma (Windows y Linux), por soportar gráficos, imágenes y sub-reportes, así como varios orígenes de datos. Permite extraer datos de PostgreSQL, MySQL**,** Oracle, DB2, MS-SQL, Informix, InterBase, Sybase, o Frontbase y exportarlas a PostScript, texte, HTML, XML, PDF, o CSV. Permite establecer los niveles de datos, subtotales y totales para el reporte. Entre otras funcionalidades interesantes, Agata Report da la posibilidad de generar un completo diagrama ER en formato DIA, proporciona un módulo para crear gráficas en curvas o en bastones y permite cruzar

informes con otras bases de datos. Además, el programa soporta parámetros temporales de ejecución (te consulta estos valores antes de correr la aplicación). [1]

Crystal Reports.

Es una herramienta que permite ordenar, filtrar y dar formato a los informes de manera dinámica dentro de los visualizadores de informes (compatibles en .NET Webform y .NET Winform), todo sin forzar una actualización de la base de datos, además transformar rápidamente cualquier fuente de datos en contenido interactivo, integrar estrechamente capacidades de diseño, modificación y visualización en aplicaciones .NET, Java o COM y permitir a los usuarios finales acceder e interactuar con los reportes a través de portales Web, dispositivos móviles y documentos de Microsoft Office®.

Los visores Web avanzados habilitan a los usuarios finales para realizar búsquedas dentro de los datos de un reporte y exportarlas posteriormente a Microsoft Excel, Word y páginas HTML con el vínculo dinámico al reporte original. Adicionalmente, el reporte completo puede ser exportado a una variedad de formatos incluyendo XML, PDF, HTML y Microsoft Excel.

Con flexibles SDKs (Software Development Kits) para aplicaciones .NET, Java y COM y controles para que incluso el usuario final pueda hacer cambios a los reportes, Crystal Reports habilita la estrecha integración tanto en aplicaciones Cliente/Servidor como en aplicaciones Web.

Licencias de publicación Web flexibles - Capacidades de procesamiento en cola, que hacen a un mejor aprovechamiento de sus licencias. [2]

Jasper Reports.

Es una librería de clases de Java de código abierto desarrollada por Teodor Danciu que está diseñada para facilitar el agregar capacidades de reporte a las aplicaciones Java. No es una herramienta por sí sola, por lo que no se puede instalar. Para utilizar Jasper Reports es necesario añadirlo a las aplicaciones Java por medio de la inclusión de su librería al classpath de la aplicación. Aún cuando Jasper Reports fue hecho con el propósito principal de añadir características de generación de reportes a aplicaciones web desarrolladas bajo Java, ésta no tiene ningún tipo de dependencia con las librerías de Java asociada a las aplicaciones web por lo que es posible utilizar Jasper Reports para aplicaciones Java de escritorios, para aplicaciones por línea de comando o inclusive para aplicaciones web hechas en PHP a través de la librería PHP/Java Bridge.

Además de los datos en texto, Jasper Reports permite incluir en los reportes imágenes, gráficos, etc, para que los mismos tengan un aspecto profesional. Algunas de las características que provee Jasper Reports son las siguientes:

- Permite una diagramación flexible de los reportes: Los reportes se pueden dividir en secciones opcionales que son: título del reporte, el encabezado de página, una sección para los detalles del reporte, el pie de página y una sección de resumen que aparece al final del reporte.
- Permite que los desarrolladores le surtan datos en varias formas: esto es que los desarrolladores pueden pasar datos a los reportes por medio del paso de parámetros. Estos parámetros de reportes pueden ser instancia de cualquier clase de Java.
- Pueden generar sub-reportes: Jasper Reports permite la creación de reportes dentro de reportes lo que facilita bastante el diseño porque es posible usar estos sub-reportes en otros reportes.
- Los reportes son capaces de presentar los datos de manera textual o a través de gráficos: no sólo son capaces de mostrar los datos que le son pasados sino que pueden generar o calcular con esos datos otros datos de forma dinámica y mostrarlos.
- Pueden generar marcas de agua: Jasper Reports permite generar textos o imágenes de fondo para utilizarlo como marcas de agua con el propósito de identificar el reporte o simplemente por motivos de seguridad.
- Se pueden exportar los reportes a una multitud de formatos: Los reportes generados con Jasper Reports pueden ser exportados a una multitud de formatos como PDF, XLS, RTF, HTML, XML, CVS (valores separados por coma) y texto plano.

Licencia: GNU Library General Public License versión 3.0 (LGPLv3). [3]

Variadas son las ofertas de herramientas con este fin que existen en el mercado. No obstante, la elección de la más conveniente depende de las necesidades de los clientes así como de las funcionalidades que ofrezcan cada una de ellas como base para soportar una implementación que responda a dichas necesidades.

Active Reports es un sistema que cumple con algunos de los requerimientos de los clientes, pero que una de las desventajas contrasta, precisamente, con los principales intereses del departamento de soberanía tecnológica, y es el hecho de ser software privativo. De igual manera sucede con la herramienta Crystal Reports.

En cambio Jasper Reports es una herramienta que se distribuye bajo licencia pública, es multiplataforma, que maneja muy fácilmente los sub-reportes y que satisface las necesidades de los clientes, pero se hace necesario analizar las herramientas creadas en la Universidad, para determinar si existe algún generador de reportes que cumpla con los requisitos que posee Jasper Reports y que satisfaga las necesidades de los clientes.

1.1.2 Herramientas de generación de reportes en la UCI.

En la Universidad como parte del proceso productivo, para soporte y solución a problemas existentes, se han desarrollado varias herramientas para la generación de reportes, en busca de una eficiente interacción de los usuarios de dichos proyectos con la información almacenada en sus bases de datos, persiguiendo la idea de elevar la eficiencia y eficacia de los mismos. Ejemplos de esas herramientas son el Generador Dinámico de Reporte V1.0 y Akademos.

Generador Dinámico de Reportes V 1.0 (GDR).

GDR es una aplicación desarrollada sobre el framework Symfony y escrita en PHP. Es multiplataforma. Soporta imágenes, gráficas, así como varios orígenes de datos. Proporciona a los usuarios, entre otras opciones, agilizar la toma de decisiones, generar reportes en varios formatos y con gran variedad de opciones en su diseño, marcando una diferencia entre los reportes tradicionales y los reportes dinámicos, objetos de este producto. Está compuesto por varias aplicaciones entre las que se encuentran el Visor de reportes, Diseñador de modelos y el Diseñador de reporte. Aunque permite abstraerse en parte de los conocimientos relacionados con los gestores de bases de datos, el usuario aún debe poseer conocimientos básicos. Además, su entorno de trabajo está estructurado de forma que es difícil guiarse en la creación y generación de reportes. La aplicación no permite la generación de sub-reportes y debido al motor de generación de reportes con la que fue creada, no es posible agregarle nuevas funcionalidades al sistema.

Akademos.

Akademos es una aplicación desarrollada en plataforma .NET, en el 2006, en la Universidad de las Ciencias Informáticas, la cual brinda actualmente importantes servicios académicos tanto a los estudiantes como a los departamentos docentes. Cuenta con una herramienta de generación de reportes, la cual los genera con eficiencia, pero desde el punto de vista del dinamismo de la generación, se encuentra adaptada solamente al negocio de la gestión académica en la UCI, restringiendo que sean elaborados con la información contenida en la base de datos específicamente

utilizada por el software, lo cual dificulta que esta herramienta sea de propósito general. Además, está diseñada sólo para la Web.

Conclusiones Parciales.

Estudiadas las herramientas se decidió desarrollar la versión 2.0 del GDR, debido a que la versión anterior presenta una compleja conexión a los diferentes gestores de base de datos, no permite la generación de sub-reportes, ni la división de reportes por tuplas, funcionalidades altamente demandadas por los clientes. Se utilizará como motor de generación de reportes Jasper Reports que ofrece soluciones a todos los problemas que presenta el GDR actual y porque además es una herramienta que se distribuye bajo los términos de licencia pública, que posee una gran comunidad que mantiene y desarrolla la librería.

1.2 Metodología y Herramientas para la modelación del software.

Existen diversas metodologías, herramientas y lenguajes de modelado que se utilizan en el proceso de desarrollo de un software de manera que se pueda obtener una breve representación del sistema. A continuación se realizará un estudio sobre estas metodologías y herramientas.

1.2.1 Metodología.

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. Existen diversas metodologías, se podrían nombrar algunas como: RUP, que es completa, XP (Extreme Programming), que es una metodología para proyectos de corto plazo y OpenUP.

La finalidad de una metodología de desarrollo es garantizar la eficacia, cumpliendo los requisitos iniciales y la eficiencia, minimizando las pérdidas de tiempo en el proceso de generación de software.

OpenUP.

Se determinó por políticas del departamento Integración de Soluciones utilizar OpenUP, por ser un proceso completo, práctico y muy eficiente.

Presenta las siguientes características:

- Proceso de desarrollo del software. Es completo en el sentido que puede ser manifestado como todo el proceso para construir un sistema.
- Es extensible ya que en el proceso se puede agregar o adaptar según lo vayan requiriendo 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.
- Se centra en una arquitectura temprana para reducir al mínimo los riesgos y organizar el desarrollo.
- Es la metodología utilizada por desarrolladores de alto nivel en casi todo el mundo por sus altas cualidades administrativas (Flores, 2008).

Además ofrece la administración de diferentes áreas del proyecto, manejando el ciclo de vida de manera apropiada. [4]

1.2.2 Lenguaje de Modelado.

Por políticas del departamento Integración de Soluciones se decide utilizar el Lenguaje de Modelación Unificado (UML), por ser un lenguaje expresivo, claro y uniforme para el diseño Orientado a Objetos, que permite una fuerte integración entre las herramientas, los procesos y los dominios.

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de la industria. [5]

Es un lenguaje más expresivo, claro y uniforme para el diseño Orientado a Objetos, que no garantiza el éxito de los proyectos pero si mejora notablemente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios.

1.2.3 Herramientas CASE.

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costos de las mismas en términos de tiempo y de dinero. Estas herramientas son de mucha ayuda en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores, entre otras.

Visual Paradigm.

Se determinó por políticas del departamento Integración de Soluciones utilizar el Visual Paradigm para visualizar y diseñar los elementos de software debido a que es multiplataforma, por las facilidades que

brinda para el diseño de los diagramas necesarios y su documentación y por contar la Universidad con la licencia para su uso.

A continuación se brindan algunas características de esta herramienta: [6]

- Es una potente herramienta CASE, para visualizar y diseñar elementos de software, para ello utiliza el Lenguaje Unificado de Modelado (UML).
- 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 diez lenguajes de programación, Java, C++, CORBA IDL, PHP, XML Schema y ADA.
- Se integra con el Visio para importar imágenes del mismo para realizar los diagramas de despliegue.
- Genera documentación para el proyecto en HTML, MS Word y PDF.
- Es gratis en su edición Community.

1.2.4 Herramientas para la Gestión de Requisitos.

La Gestión de Requisitos es un componente vital en el desarrollo de un proyecto ya que provee la dirección y alcance del proyecto.

“La Gestión de Requisitos en Ingeniería de Sistemas es el proceso encargado de la identificación, asignación y seguimiento de sus requisitos, incluyendo la interfaz, verificación, modificación y control del estatus a lo largo del ciclo de vida” [7].

Actividades de la Gestión de Requisitos: [8]

- Recolección de requisitos.
- Documentación de requisitos.
- Verificación de requisitos.
- Gestión de cambio de requisitos.

Open Source Requirements Management Tool (OSRMT).

Por políticas del departamento Integración de Soluciones y por ser una herramienta libre, se decide utilizar esta herramienta.

OSRMT es una herramienta de gerencia abierta de los requisitos de la fuente, diseñada para servir al ciclo de vida completo del desarrollo de software para el funcionamiento de proyectos de la empresa, bajo control del diseño. Trabaja en arquitectura cliente/servidor, desarrollado bajo Java.

Incorpora un módulo para manejar la trazabilidad de los requisitos y un módulo para el control de cambios, genera además la documentación de los requisitos tratados. Posee una interfaz agradable y seguridad duplicada. Permite la trazabilidad entre los documentos de trabajo, personalización y configuración, representación jerárquica de los documentos de trabajo, definición de casos de prueba mediante pruebas para cada uno de los pasos del caso de uso, entre otras funcionalidades.

1.2.5 Herramientas de Estimación.

Cuando se planifica un proyecto se obtienen estimaciones del esfuerzo humano requerido, de duración cronológica del proyecto y del costo.

Existen diferentes herramientas de estimación del costo y esfuerzo, estas son una forma de resolución de problemas en donde la mayoría de los casos, el problema a resolver es demasiado complejo para considerarlo como una sola parte, por eso se descompone en un conjunto de pequeños problemas.

Las técnicas de estimación se dividen en dos grandes categorías: [9]

1. Descomposición: requieren un bosquejo de las principales funciones del software, algunas de las técnicas que agrupa son:
    - Tamaño del Software.
    - Líneas de Código.
    - Puntos de Función.
    - Basada en el Proceso.
    - Casos de Uso.
2. Modelo empírico: usan expresiones para esfuerzo y tiempo obtenidas empíricamente para predecir estas cantidades del proyecto, en este grupo se encuentra:
    - Cocomo II
    - La ecuación del software

Por políticas de la Universidad se utiliza la planilla de estimación de Calisoft, pues la misma está actualmente involucrada en un proceso de mejora (SPI, por sus siglas en inglés “Software Process Improvement”) basado en el modelo CMMI (Capability Madurity Model Integration), cuyos objetivos se han enfocado en:

- Mejorar los procesos, métodos, tecnologías y la calidad de los proyectos a partir de la incorporación de buenas prácticas propuestas por el modelo CMMI.

- Obtener una evaluación del nivel 2 de CMMI.

El desarrollo del método de estimación parte de la necesidad de estimar tamaño, costo y esfuerzo requerido para desarrollar los proyectos en la Universidad. Dado que la misma no cuenta con una base histórica, se tuvo en cuenta los datos dispersos de algunos proyectos y la evaluación de algunos factores que, según criterio de expertos, pueden influir en las estimaciones de los proyectos. Para la elaboración del método de estimación se realizó un estudio de otros métodos de la literatura como COCOMO I, II y III, Puntos de Función y Puntos de Caso de Uso, así como algunas buenas prácticas de proyectos en la UCI.

1.3 Ingeniería de Requisitos.

Sobre el proceso de ingeniería de requisitos, varios autores plantean diferentes conceptos o significados según su nivel de experiencia o por su manera de ver los requisitos respecto al desarrollo de un determinado proyecto. En la ingeniería de requisitos principalmente se identifican dos aspectos muy importantes, el primero, que es el propósito del sistema que se va a desarrollar, y el segundo, el contexto en el que será usado. En base a estas características, se definen algunos conceptos como:

La ingeniería de requisitos o los requisitos en sí, constituyen el enlace entre las necesidades reales de los clientes, usuarios y otros participantes vinculados al sistema. La ingeniería de requisitos consiste en un conjunto de actividades y transformaciones que pretenden comprender las necesidades de un sistema software y convertir la declaración de estas necesidades en una descripción completa, precisa y documentada de los requisitos del sistema siguiendo un determinado estándar. [10]

La ingeniería de requisitos es un área de investigación que procura atacar un punto fundamental en el proceso, que es la definición de lo que se quiere producir. [11]

Sobre el tema se han publicado numerosos artículos y cada uno de estos actores propone diferentes procesos de ingeniería de requisitos.

- Wiegers propone que los procesos de la Ingeniería de Requisitos (IR) se divida en 2 procesos Desarrollo de los requisitos y Gestión de los requisitos. El primero contemplará los procesos de Obtención, Análisis, Especificación y Validación de requisitos.
- Pressman propone que los procesos sean: Inicio, Obtención, Elaboración, Negociación, Especificación, Validación y Gestión de requisitos.
- Sommerville propone que sean: Estudio de Viabilidad, Obtención y Análisis, Especificación y Validación de requisitos.

Después de realizado un estudio de los diferentes procesos de IR anteriormente mencionados, se decide resumir las mejores prácticas de estos procesos para obtener aquellos por los cuales se guiará el análisis del trabajo.

- Estudio de Viabilidad.
- Obtención/Elicitación.
- Análisis
- Especificación.
- Validación.
- Administración.

1.3.1 Características de los Requisitos.

La etapa de definición de requisitos es de vital importancia para el proceso de desarrollo de un software, es la actividad donde el equipo de desarrollo de un sistema de software identifica las necesidades que debe cubrir este sistema, este proceso puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas.

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) en el glosario estándar de términos de la ingeniería de software define un requisitos como:

- Una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.
- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Una representación documentada de una condición o capacidad como en I o II.

Requisitos funcionales y no funcionales.

Los requisitos se clasifican en funcionales y no funcionales.

Requisitos funcionales: Son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

Requisitos no funcionales: Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto.

1.3.2 Estudio de Viabilidad.

Factibilidad se refiere a la disponibilidad de los recursos necesarios para llevar a cabo los objetivos o metas señalados, la factibilidad se apoya en 3 aspectos básicos:

- Operativo.
- Técnico.
- Económico.

El éxito de un proyecto está determinado por el grado de factibilidad que se presente en cada una de los tres aspectos anteriores. Este estudio sirve para recopilar datos relevantes sobre el desarrollo de un proyecto y en base a ello tomar la mejor decisión, si procede su estudio, desarrollo o implementación.

1.3.3 Captura de requisitos.

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo, extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir el sistema que se desea crear.
Algunas técnicas para la captura de requisitos:

Entrevistas.

Técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendida. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. [12]

Brainstorming (Tormenta de ideas):

Técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. [12]

Casos de Uso.

Aunque inicialmente se desarrollaron como técnica para la definición de requisitos [13] algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los *actores* del mismo para realizar una determinada función.

Introspección.

Esta técnica recomienda que el analista se ponga en el lugar del interesado y trate de imaginar cómo desearía éste la aplicación de software. Basado en estas suposiciones, el analista entrega recomendaciones, al interesado sobre la funcionalidad que debería tener dicha aplicación (Guguer y Linde, 1993). Dicha técnica aplicada a los software, consiste en la exploración y utilización de productos, con el fin de detectar requisitos ausentes en el producto a desarrollar.

Cuestionarios y Checklists:

Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (Checklist). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.

1.3.3 Análisis de requisitos.

Este proceso permite obtener una idea real de lo que se quiere y se pude lograr con los requisitos levantados. Es la fase donde se determina el alcance y la verdadera aplicación de los requisitos obtenidos. Se aceptan los mismos.

1.3.4 Especificación de requisitos.

Es una de los procesos más importante, es donde por primera vez se definirán clara y abundantemente en lo que consiste cada requisito detectado. Es donde además se reconocerán las entradas y las salidas que dicho requisito requiere. Es el proceso de documentación formal de cada condición, capacidad y característica que el sistema deberá poseer.

1.3.5 Técnicas de validación de requisitos.

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea.

Algunas técnicas que serán aplicadas:

Prototipos.

Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final.

Generación de casos de prueba.

Esta técnica consiste en comprobar que el producto cumpla con todos los requisitos que puedan ser verificados antes de continuar con el desarrollo de las disciplinas subsecuentes del proceso de desarrollo del software. Basada en la premisa de que cada requisito debe ser posible de probar, la técnica propone el diseño de casos de prueba donde se evalúen todos los requisitos. Un caso de prueba para requisitos funcionales generalmente consiste en definir las entradas y las salidas del sistema, además de acciones del usuario que posibilitan completar lo que expresa el requisito.

Revisiones.

Esta técnica es muy fácil de utilizar. En general se revisan los requisitos y se utiliza un mecanismo de validación llamado lista de chequeo, que es una lista de preguntas que el analista debe usar para evaluar cada requisito, verificando y marcando los puntos de la lista, mientras se lee el documento de requisitos. Cuando se descubren problemas potenciales deben ser anotados, ya sea en los márgenes del documento o en una lista de análisis. Las listas brindan un recordatorio de lo que se debe buscar y reducen la posibilidad de obviar alguna verificación importante.

1.3.6 Administración de requisitos.

Es el último proceso de IR y es donde finalmente se gestionan cada uno de los requisitos funcionales y no funcionales del sistema, con el fin de comprobar que el producto que se diseñará e implementará cumple con cada uno de ellos.

1.4 Características del rol Analista.

Un analista es una persona imprescindible en cualquier departamento de Informática. El analista es quien determina la problemática concreta que debe solucionar una aplicación y las líneas centrales de cómo debe desarrollarse dicha aplicación para resolver el problema.

El rol de un analista es fundamentalmente evaluar de manera sistemática el funcionamiento de un negocio, conducir y coordinar el levantamiento de requisitos y el modelado de Casos de Uso, estableciendo la funcionalidad del sistema y el alcance del mismo, y además, validar los requisitos levantados con los usuarios o clientes, todas estas actividades con el propósito de mejorar los procesos organizacionales.

El Analista agrupa los roles que están involucrados fundamentalmente en la captura y gestión de los requisitos del sistema, que pueden estar representados por una o varias personas entre los que se encuentran: Analista de Procesos del Negocio, Diseñador del Negocio, Analista del Sistema y Especificador de Requisitos.

- Analista de Procesos del Negocio.
  Es el responsable de definir la arquitectura del negocio; los casos de uso del negocio y actores, así como sus relaciones.

- Diseñador del Negocio.
  Es el encargado de detallar la especificación de la organización y especificar el flujo de trabajo de los casos de usos del negocio en términos de trabajadores del negocio y entidades del negocio.

- Analista del sistema.
  Es el responsable de dirigir y coordinar el proceso de captura de requisitos y desarrollo del modelo de casos de uso, definiendo las funcionalidades y límites del sistema.

- Especificador de Requisitos.
  Es el encargado de especificar los detalles de una o varias partes de la funcionalidad del sistema, describiendo uno o varios aspectos de los requisitos, además de agrupar los casos de usos en paquetes.

1.4.1 Artefactos generados por el Analista.

Los artefactos que son generados o que modifica el analista del sistema son:

- Modelo de casos de uso del negocio.
  Es un modelo que describe los procesos de un negocio y su interacción con elementos externos.

- Modelo de análisis del negocio.

Describe la realización de los casos de uso del negocio por la interacción de los trabajadores y entidades del negocio.

- Glosario del negocio.
  Es un documento que define los principales términos usados en una parte del proyecto.

- Actor del negocio.
  Es el rol que algo o alguien juega cuando interactúa con el negocio para beneficiarse de sus resultados.

- Casos de uso del negocio.
  Representa a un proceso de negocio, por lo que se corresponde con una secuencia de acciones que producen un resultado observable para los actores del negocio.

- Realización de los casos de uso del negocio.
  Describe cómo los trabajadores del negocio, entidades del negocio y los eventos del negocio interactúan en la realización de un caso del uso del negocio.

- Trabajador del negocio.
  Es una abstracción de una persona o sistema automatizado, que realiza una o varias actividades en el negocio. Es el encargado de manipular las entidades del negocio.

- Entidades del negocio.
  Representan a los objetos que los trabajadores del negocio toman, inspeccionan, manipulan y utilizan durante la realización de los casos de uso del negocio. Comúnmente representan un documento o una parte esencial de un producto.

- Modelo de casos de uso.
  Es un modelo del sistema que contiene actores, casos de uso y sus relaciones.

- Glosario.
  Es un documento que define los términos comunes que se utilizan para describir el proyecto.

- Actor
  Representa terceros fuera del sistema que colabora con éste.

- Casos de uso.

Fragmentos de funcionalidad que el sistema ofrece para brindar un resultado de valor a sus actores.

- Paquetes de casos de uso.
  Es una colección de casos del uso, actores, relaciones, diagramas y otros paquetes; que se usan para estructurar el modelo de casos de uso en partes más pequeñas.

- Especificación de requisitos del software.
  Es la captura de los requisitos del software para el sistema o una parte de éste.

- Descripción de la arquitectura (vista del modelo de casos de uso).
  Este artefacto es realizado por el Arquitecto, representa los casos de uso significativos para la arquitectura, debido a que describen alguna funcionalidad importante y crítica que debe priorizarse dentro del ciclo de vida del software.

- Prototipo de interfaz de usuario.
  Este artefacto es realizado por el Diseñador de interfaz de usuario, representa un ejemplo visual de la interfaz de usuario que se construye para explorar y/o validar el diseño de interfaz de usuario, permitiéndole al cliente verificar que el sistema satisfaga sus necesidades.

Debido a que el análisis de la solución se encuentra inmerso en un Proceso de Mejora no es necesaria la realización de todos los artefactos anteriormente mencionados. Los artefactos que se generarán en esta etapa son: Modelo de Dominio, Especificación de Requisitos y Especificación de Caso de Uso.

1.5 Patrones de Casos de Uso.

La experiencia en la utilización de casos de uso ha evolucionado en un conjunto de patrones que permiten con más precisión reflejar los requerimientos reales, haciendo más fácil el trabajo con los sistemas, y mucho más simple su mantenimiento.

¿Por qué usar Patrones?

- Producción de Software más resistente al cambio.
- Establece problemas Pareja-Solución.
- Ayudan a especificar interfaces.
- Reutilización del Código.
- Uso de Documentación Estándar.

Patrones de Caso de uso:

Concordancia (Commonality).

Extrae una subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso y es expresado por separado.

Reusabilidad: consta de 3 casos de uso. El primero llamado subsecuencia común, modela una secuencia de acciones que aparecerán en múltiples casos de uso en el modelo. Los otros casos de uso modelan el uso del sistema que comparte la subsecuencia común de acciones. De manera que deben existir al menos dos de ellos.

Especialización: contiene casos de uso del mismo tipo, estos son modelados como una especialización de casos de uso de tipo de uso común. Todas las acciones en estos casos de uso son heredadas por los casos de uso hijos, donde otras acciones serán adicionadas o acciones heredadas que serán especializadas. Este patrón es aplicable cuando la utilización de los casos de uso que han sido modelados son del mismo tipo, y este tipo debe hacerse visible en el modelo.

Adición: en el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema.

CRUD (Creating, Reading, Updating, Deleting).

Este patrón se basa en la fusión de casos de uso simples para formar una unidad conceptual.

Completo: consta de un caso de uso, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples.

Parcial: 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.

Extensión Concreta.

Consiste en dos casos de uso y una relación extendida entre ellos. Puede ser instalado en sí mismo, así como extendido en el caso de uso base. El referente puede ser concreto o abstracto. Este patrón se aplica cuando un flujo puede extender el flujo de otro caso de uso así como ser realizado en sí mismo.

Inclusión.

Es un patrón de estructura, consiste en dos casos de uso y una relación de inclusión entre el caso de uso base y el caso de uso incluido. Este último puede ser instanciado por sí solo. El caso de uso base puede ser concreto o abstracto. Este patrón se utiliza cuando un flujo de datos puede ser incluido en el flujo de otro caso de uso y también puede ejecutarse por sí solo (Gunnar Overgaard, 2004).

Múltiples Actores.

Roles diferentes Captura la concordancia entre actores manteniendo roles separados. Consiste de un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso (Gunnar Overgaard, 2004).

Reglas del Negocio.

Se basan en la extracción de información originada de las políticas, reglas y regulaciones del negocio de la descripción del flujo y describe la información como una colección de reglas del negocio referenciadas a partir de las descripciones de los casos de uso. Definición Estática El patrón Reglas de Negocio: Definición Estática es de tipo descripción, por lo que no influye sobre la estructura del modelo de casos de uso. Este patrón es aplicado a todos los casos de uso que modelan servicios que son afectados por reglas de negocio definidas en la organización. Las reglas son descritas en un documento por separado, referenciado por la descripción del caso de uso. Es apropiado cuando no se necesita cambios dinámicos en las reglas del negocio cuando el sistema está en uso (Gunnar Overgaard, 2004).

Generalización/Especialización.

Un caso de uso generalización es una relación de un caso de uso hijo a un caso de uso padre que especifica cómo el hijo puede especializar todo el comportamiento y características descritas para el padre. Se utiliza para mostrar que los flujos comparten la estructura, objetivo y comportamiento.

1.6 Conclusiones.

En este capítulo se analizaron los principales conceptos necesarios para entender la importancia de este trabajo, se realizó una reseña del marco conceptual teórico del tema a tratar. Se hizo un estudio de algunos sistemas informáticos existentes para la generación de reportes, las herramientas y metodologías que debe utilizar un analista, definiendo las bases teóricas para un correcto análisis. Se utilizará como generador de reporte la librería Jasper Report, pues permite la creación de sub-reportes, que los desarrolladores le surtan datos a los reportes en varias formas y por sus características en particular, se empleará como metodología a utilizar, OpenUP, como herramienta de modelado el Visual Paradigm, como herramienta de gestión de requisitos, OSRMT. Luego de haber estudiado los diferentes patrones de casos de uso, se eligirán aquellos que permitan la elaboración de los casos de uso del sistema, para así dar solución a la problemática propuesta.

CAPÍTULO 2: ANÁLISIS DEL SISTEMA

Introducción.

En el presente capítulo se ofrece una descripción del sistema propuesto por esta investigación, se realiza el modelo conceptual definiendo los diversos elementos que serán manejados, así como las relaciones que entre estos se establecen. Se especifican los actores y casos de uso. Para el desarrollo del análisis del software se obtienen los requisitos funcionales y no funcionales, a partir de los cuales se identifican los casos de uso y posteriormente se describen los mismos.

2.1 Descripción del Sistema.

El Generador Dinámico de Reportes (GDR), pretende dar solución al problema de obtener los diferentes reportes en los sistemas de gestión de la información que se desarrollan en cualquier entorno empresarial, incluyendo la Universidad de las Ciencias Informáticas. Consta de 5 módulos: Diseñador de Modelos, Diseñador de Reportes, Diseñador de Consultas, Visor de Reportes, y Administrador de Reportes. Uno de los pasos esenciales para lograr dicho objetivo es desarrollar un producto que sea capaz de mostrar informes o reportes, con el objetivo de tomar decisiones, realizar estudios investigativos, o sencillamente consultar información propia del negocio para el cual fue concebido. En la versión anterior los procesos se distribuyen entre varios componentes que se pueden ampliar e integrar en soluciones personalizadas. Una típica aplicación para la generación de informes atraviesa por las tres etapas del ciclo de vida de los reportes: creación, administración y entrega; GDR ofrece todas las herramientas necesarias para soportar estos procesos, la siguiente figura muestra la integración de las diferentes aplicaciones del sistema en el ciclo de vida.

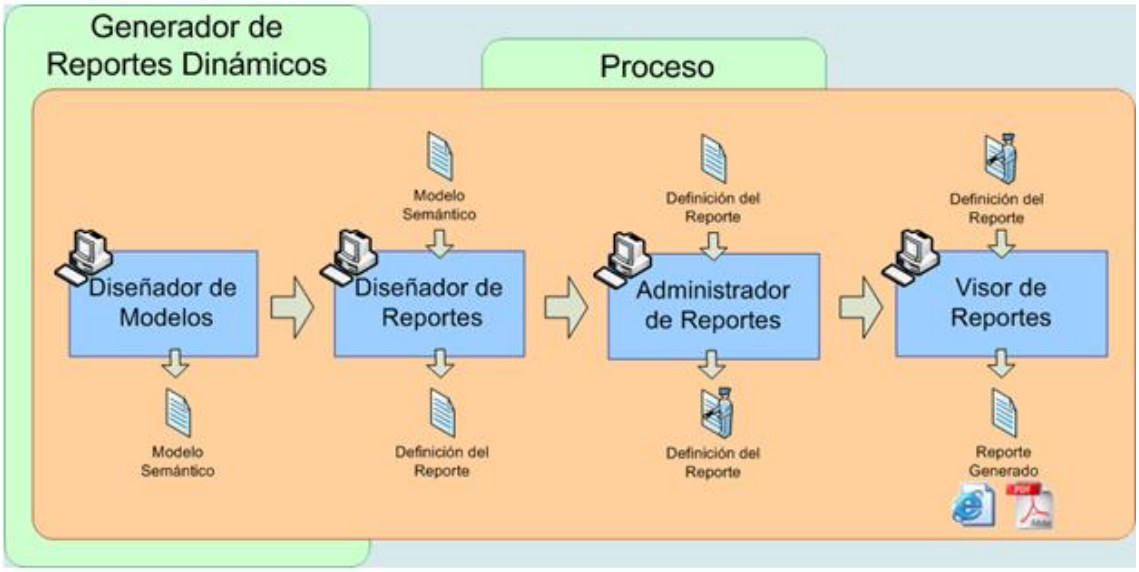

Figura 1: Flujo de generación de reporte 1

Dentro de la nueva versión, los módulos que se estarán analizando son Diseñador de Modelos (DM) y Diseñador de Reportes (DR), quienes en versiones anteriores realizan una serie de funciones que se explicarán a continuación. El DM permite diseñar modelos semánticos (MS), que son una porción determinada de la base de datos de donde potencialmente se necesita extraer información para la conformación de los reportes, permitiendo a los usuarios no usar directamente las tablas de la base de datos del sistema, sino solamente aquellas que contienen información relevante para ellos, esta abstracción mejora la eficiencia, por cuanto la aplicación no trabajará directamente con la base de datos de donde deberá recuperar datos, sino con una representación de ella que contiene los metadatos de la porción representada. El segundo modelo, el DR, permite diseñar reportes y plantillas de reportes que luego pueden ser salvados hacia la base de datos del Sistema de Generación Dinámica de Reportes para poder ser abiertos y utilizados en el momento que se les necesite, esta herramienta tiene un área de trabajo hacia la cual se pueden arrastrar componentes que se encuentran en una paleta y a los cuales se les puede configurar sus propiedades en un inspector de propiedades, a los reportes se les puede realizar una vista previa desde su propia concepción y de esta forma poder ir acomodando el diseño del reporte a las necesidades de los usuarios.

La aplicación consta de 3 grandes subsistemas, los cuales conforman los principales elementos del sistema:

- Un Servidor de Reportes que administra y procesa los reportes en diferentes formatos. Las salidas incluidas pueden ser HTML, PDF, XLS, entre otras.
- Un Conjunto de Aplicaciones que se utilizan para diseñar, administrar y visualizar reportes.
- Una Interfaz de Comunicación que permita a terceros programas generar reportes y utilizarlos.

Actualmente el GDR permite elaborar informes sobre orígenes de datos que pueden tener varios cientos de miles de tuplas, obteniendo reportes de más de 3500 páginas en un tiempo razonable. Posee 5 niveles de integración con otras aplicaciones que lo deseen usar. Sus módulos son independientes entre sí y por tanto desacoplables. La salida de un módulo constituye la entrada del siguiente.

2.2 Modelo de Dominio.

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema. Representa conceptos del mundo real, no de los componentes de software. El modelo del dominio se crea para documentar los conceptos dominantes y el vocabulario del sistema, identifica las relaciones entre todas las entidades importantes dentro del sistema e identifica generalmente sus métodos y cualidades importantes. [14]

2.2.1 Diagrama Conceptual del Modelo de Dominio.

El Modelo de Dominio que se presenta a continuación tiene como objetivo contribuir a la comprensión del análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes.

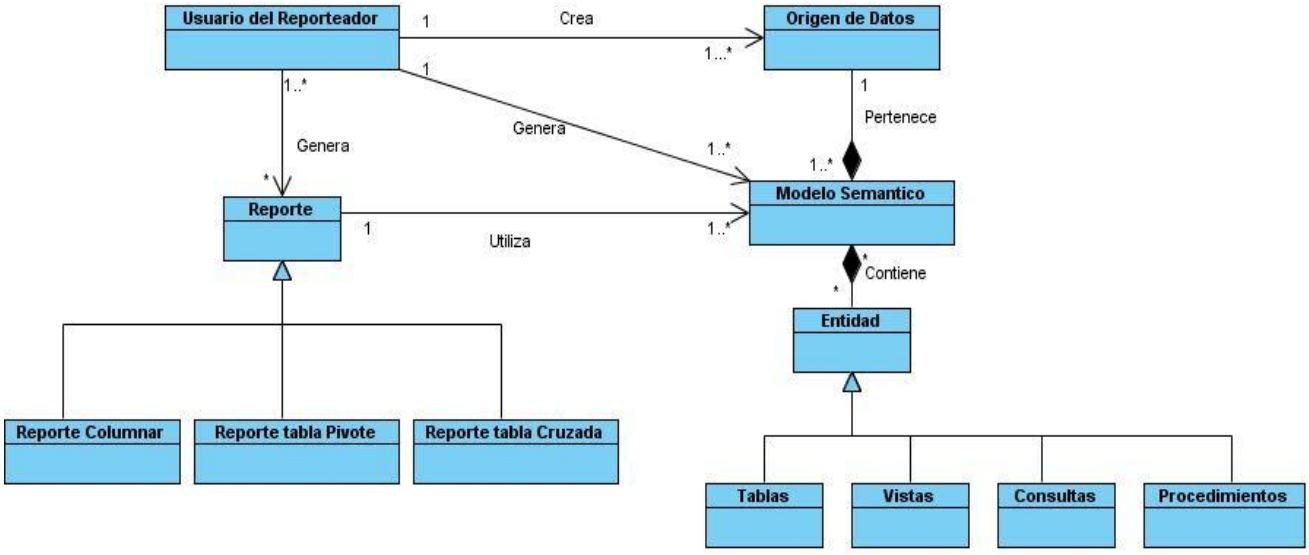

Figura 2: Modelo de Dominio. 1

En el diagrama anterior se puede apreciar que el usuario del reporteador es quien crea el Origen de datos, y este último pertenece al Modelo Semántico. Este modelo contiene una Entidad que puede ser tablas, vistas, procedimientos y consultas. Se puede observar además que los reportes son generados por el usuario y que ambos utilizan el modelo anteriormente mencionado. Los reportes pueden ser de tres tipos: tabulares, tipo tablas pivotes y tipo tablas cruzadas.

Se identifican como conceptos utilizados en el diagrama a los siguientes términos del glosario:

Reporteador: Se refiere al producto Generador Dinámico de Reportes, a la aplicación en sí.

Reportes: Son objetos que entregan información en un formato particular y que permiten realizar ciertas operaciones como, imprimirlos, enviarlos por email, guardarlos a un archivo, a partir de los datos almacenados en una base de datos.

Origen de Datos: 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.

Modelo: Es un modelo de datos usado para almacenar en forma de fichero XML toda la información de los metadatos de los objetos de la base de datos, ejemplo: tablas, vistas, funciones.

Entidades: Se refiere a las vistas, consultas, tablas, funciones y procedimientos de una base de datos, o sea las acciones que se pueden realizar en esta.

2.3 Especificación de Requisitos Software.

La Especificación de Requisitos Software (ERS) es una descripción completa del comportamiento del sistema que se va a desarrollar. La ERS también contiene requisitos no funcionales (o complementarios). Los requisitos no funcionales son requisitos que imponen restricciones en el diseño o la implementación. [15]

2.3.1 Requisitos Funcionales.

Para la realización de esta versión, se utilizaron los requisitos levantados en versiones anteriores, mediante las técnicas de entrevistas con el usuario, tormentas de ideas y discusiones, además para enriquecer la nueva versión se aplicó la técnica de introspección a las herramientas generadoras de reportes estudiadas en el capítulo anterior, con el objetivo de identificar aquellos requisitos que no estaban en versiones anteriores y agregarlos a las nuevas. Dicha técnica consiste en la utilización y análisis del entorno de trabajo de estas herramientas para la detección de requisitos no existentes en nuestra solución.

A continuación se presenta el listado de los requisitos fundamentales que fueron obtenidos, el resto se podrán encontrar en el documento de especificación de requisitos, anexo a la tesis:

Módulo Diseñador de Modelos.

RF1 Seleccionar un origen de datos.

Permitirá seleccionar un origen de datos y mostrará todos los modelos semánticos creados a partir de ella.

Entradas:

- Nombre: Este campo será el nombre del origen de datos seleccionado y debe ser una cadena entre 1 y 255 caracteres (obligatorio).

Salidas:

- Se focalizará el origen de datos seleccionado y se marcarán los modelos semánticos diseñados a partir de esta.

RF2 Seleccionar objetos y entidades desde el origen de datos seleccionado.

El sistema debe permitir seleccionar tablas, vistas, funciones o procedimientos almacenados para diseñar un modelo semántico.

Entradas:

- Listado de tablas: Este campo será un arreglo de cadenas entre 1 y 255 caracteres que representa el nombre de las tablas en la base de datos que pueden incluirse en un modelo semántico (opcional).
- Listado de vistas: Este campo será un arreglo de cadenas entre 1 y 255 caracteres que representa el nombre de las vistas de la base de datos que pueden incluirse en un modelo semántico (opcional).
- Listado de funciones o procedimientos almacenados: Este campo será un arreglo de cadenas entre 1 y 255 caracteres que representa el nombre de las funciones o procedimientos almacenados de la base de datos que pueden incluirse en un modelo semántico (opcional).

Salidas:

- Mostrará las tablas, vistas, funciones o procedimientos almacenados seleccionados.

RF3 Crear un nuevo origen de datos.

El sistema debe guardar los datos de conexión a un servidor de base de datos, creando de esta forma una nueva fuente de datos.

Entradas:

- Dirección IP: Este campo será una cadena de caracteres formada por 4 números separados por punto, donde cada número debe estar entre 0 y 253 (obligatorio).
- Puerto: Este campo será un número entre 1 y 65536 e indicará el puerto de conexión al servidor de Base de Datos (obligatorio).

- Tipo de gestor: Este campo indicará el tipo de gestor de Base de Datos sobre el que se encuentra la base de datos utilizada como origen de datos (obligatorio).
- Usuario: Este campo será una cadena de caracteres formada por números y letras e indicará el nombre de usuario que utilizará el sistema para conectarse a la base de datos (opcional).
- Contraseña: Este campo será una cadena de caracteres que podrá estar formada por letras, números, caracteres especiales o cualquier combinación de estos, e indicará la contraseña del usuario que utilizará el sistema para conectarse a la base de datos (opcional).
- Nombre: Este campo indicará el nombre de la nueva fuente de datos a conectar y será una cadena entre 1 y 255 caracteres (obligatorio).
- Base de Datos: Este campo indicará el nombre de la base de datos presente en el servidor especificado y que será utilizada como fuente de datos (obligatorio).

Salidas:

- Origen de datos: Mostrará la nueva fuente de datos creada la cual será una cadena entre 0 y 255 caracteres.

RF4 Crear modelos semánticos.

Permitirá guardar los metadatos de las tablas, vistas, funciones o procedimientos almacenados y consultas seleccionados por el usuario.

Entradas:

- Nombre: Este campo indicará el nombre del modelo semántico a crear y será una cadena entre 1 y 255 caracteres (obligatorio).
- Nombres de las tablas: Este campo indicará un arreglo de nombres de tablas y será una cadena entre 1 y 255 caracteres (opcional).
- Nombres de las vistas: Este campo indicará un arreglo de nombres de vistas y será una cadena entre 1 y 255 caracteres (opcional).
- Nombres de las funciones o procedimientos almacenados: Este campo indicará un arreglo de nombres de funciones o procedimientos almacenados creados en el gestor y que hayan sido previamente seleccionados y será una cadena entre 1 y 255 caracteres (opcional).
- Nombres de columnas: Este campo indicará los nombres de columnas de cada entidad y será una cadena entre 1 y 255 caracteres (opcional).

- Tipo de las columnas: Este campo indicará el tipo de dato de las columnas, estos están en dependencia de los tipos de datos soportados por cada gestor (opcional).
- Parámetros: Este campo representará un arreglo de nombres de parámetros usados por cada una de las funciones o procedimientos almacenados y será una cadena entre 1 y 255 caracteres (opcional).
- Tipos de los parámetros: Este campo representará un arreglo con los tipos de dato de los parámetros usados por cada una de las funciones o procedimientos almacenados y será una cadena entre 1 y 255 caracteres (opcional).

Salidas:

- Mostrará el modelo semántico creado.

Módulo Diseñador de Reportes.

RF5 Diseñar reportes columnares.

Permitirá diseñar reportes columnares a modo de listados de personas, carros, etc., permitiendo realizar agrupaciones por cualquier campo y con varios niveles de agrupamiento.

Entradas:

- Modelo semántico: Este campo será el modelo semántico que se usará para diseñar el reporte y será una cadena entre 1 y 255 caracteres (obligatorio).
- Tipo de reporte: Este campo será una cadena de caracteres entre 1 y 255 caracteres, usado para indicar el tipo de reporte que se va a crear, en este caso debe especificarse: reporte tabular (obligatorio).
- Tabla: Este campo será el nombre de alguna tabla en caso que el reporte se vaya a generar a partir de ella y será en una cadena entre 1 y 255 caracteres (opcional).
- Vista: Este campo será el nombre de alguna vista en caso que el reporte se vaya a generar a partir de ella y será en una cadena entre 1 y 255 caracteres (opcional).
- Función o procedimiento almacenado: Este campo será el nombre de alguna función o procedimiento almacenado en caso que el reporte se vaya a generar a partir de ella y será una cadena entre 1 y 255 caracteres (opcional).
- Consulta: Este campo será el nombre de alguna consulta en caso que el reporte se vaya a generar a partir de ella y será una cadena entre 1 y 255 caracteres (opcional).

- Campos**:** Esta entrada serán los campos que se desean incluir en el reporte y será una cadena de entre 1 y 255 caracteres (obligatorio).
- Etiquetas: Este campo consistirá en una cadena entre 1 y 255 caracteres que se usará para poner texto personalizado a los reportes tales como: título, cabeceras, pie de páginas o cualquier otro texto que se desee visualizar (opcional).
- Imagen: Este campo será en una imagen que se desee adicionar al reporte, esta imagen estará referenciada por una cadena entre 1 y 255 caracteres que representará la ubicación de la misma (opcional).
- Número de página: Este campo será un número entero usado para numerar las páginas que contendrá el reporte (opcional).

Salidas:

- El diseño del reporte en un fichero XML que contendrá la estructura del reporte columnar diseñado.

RF6 Diseñar reportes de tipo tablas pivotes.

Permitirá diseñar reportes matriciales a modo de análisis estadísticos de la información, obteniendo como resultado las llamadas tablas pivotes.

Entradas:

- Modelo semántico: Este campo será el modelo semántico que se usará para diseñar el reporte y será en una cadena entre 1 y 255 caracteres (obligatorio).
- Tipo de reporte: Este campo será en una cadena de caracteres entre 1 y 255 caracteres, usado para indicar el tipo de reporte que se va a crear, en este caso debe especificarse: reporte de tabla cruzada (obligatorio).
- Tabla: Este campo será el nombre de alguna tabla en caso que el reporte se vaya a generar a partir de ella y será en una cadena entre 1 y 255 caracteres (opcional).
- Vista: Este campo será el nombre de alguna vista en caso que el reporte se vaya a generar a partir de ella y consistirá en una cadena entre 1 y 255 caracteres (opcional).
- Función: Este campo será el nombre de alguna función en caso que el reporte se vaya a generar a partir de ella y consistirá en una cadena entre 1 y 255 caracteres (opcional).
- Consulta: Este campo será el nombre de alguna consulta en caso que el reporte se vaya a generar a partir de ella y consistirá en una cadena entre 1 y 255 caracteres (opcional).
- Filas**:** Esta entrada será un listado de campos que se deseen incluir en las filas de la tabla pivote y será un arreglo de cadena entre 1 y 255 caracteres (obligatorio).

- Columnas: Esta entrada será un listado de campos que se deseen incluir en las columnas de la tabla pivote y será un arreglo de cadena entre 1 y 255 caracteres (obligatorio).
- Pivote: Este entrada será un campo que aparecerá llenando las celdas interiores de la tabla y es una cadena entre 1 y 255 caracteres (obligatorio).
- Etiquetas: Esta entrada será una cadena entre 1 y 255 caracteres que se usa para poner texto personalizado a los reportes tales como: título, cabeceras, pie de páginas o cualquier otro texto que se desee aparezca (opcional).
- Imagen: Este campo será una imagen que pueda ser usada como logotipo y que se desee adicionar al reporte, esta imagen estará referenciada por una cadena entre 1 y 255 caracteres que representará un camino en el lado del cliente (opcional).
- Número de página: Este campo será un número entero usado para numerar las páginas que contendrá el reporte (opcional).

Salidas:

- El diseño del reporte en un fichero XML que contendrá la estructura del reporte de tipo tabla pivote diseñado.

RF7 Diseñar reportes de tipo tablas cruzadas.

Permitirá diseñar reportes matriciales a modo de análisis estadísticos de la información, obteniendo como resultado las llamadas tablas cruzadas.

Entradas:

- Modelo semántico: Este campo será el modelo semántico que se usará para diseñar el reporte y será en una cadena entre 1 y 255 caracteres (obligatorio).
- Tipo de reporte: Este campo será en una cadena de caracteres entre 1 y 255 caracteres, usado para indicar el tipo de reporte que se va a crear, en este caso debe especificarse: reporte de tabla cruzada (obligatorio).
- Tabla: Este campo será el nombre de alguna tabla en caso que el reporte se vaya a generar a partir de ella y será en una cadena entre 1 y 255 caracteres (opcional).
- Vista: Este campo será el nombre de alguna vista en caso que el reporte se vaya a generar a partir de ella y consistirá en una cadena entre 1 y 255 caracteres (opcional).
- Función: Este campo será el nombre de alguna función en caso que el reporte se vaya a generar a partir de ella y consistirá en una cadena entre 1 y 255 caracteres (opcional).

- Consulta: Este campo será el nombre de alguna consulta en caso que el reporte se vaya a generar a partir de ella y consistirá en una cadena entre 1 y 255 caracteres (opcional).
- Filas**:** Esta entrada será un listado de campos que se deseen incluir en las filas de la tabla cruzada y será un arreglo de cadena entre 1 y 255 caracteres (obligatorio).
- Columnas: Esta entrada será un listado de campos que se deseen incluir en las columnas de la tabla cruzada y será un arreglo de cadena entre 1 y 255 caracteres (obligatorio).
- Etiquetas: Esta entrada será una cadena entre 1 y 255 caracteres que se usa para poner texto personalizado a los reportes tales como: título, cabeceras, pie de páginas o cualquier otro texto que se desee aparezca (opcional).
- Imagen: Este campo será una imagen que pueda ser usada como logotipo y que se desee adicionar al reporte, esta imagen estará referenciada por una cadena entre 1 y 255 caracteres que representará un camino en el lado del cliente (opcional).
- Número de página: Este campo será un número entero usado para numerar las páginas que contendrá el reporte (opcional).

Salidas:

- El diseño del reporte en un fichero XML que contendrá la estructura del reporte de tipo tabla cruzada diseñado.

RF8 Diseñar sub-reportes.

Descripción: Permitirá diseñar sub-reportes a modo de listados de reportes. Permitirá realizar reportes tabulares, de tipo tablas pivotes y de tipo tablas cruzadas.

Entradas:

- Nombre del reporte: Este campo indicará el nombre del reporte principal y será una cadena entre 1 y 255 caracteres (obligatorio).
- Nombre del sub-reporte: Este campo indicará el nombre del sub-reporte a crear y será una cadena entre 1 y 255 caracteres (obligatorio).
- Modelo semántico: Este campo será el modelo semántico que se usará para diseñar el reporte y será una cadena entre 1 y 255 caracteres (obligatorio).
- Tipo de reporte: Este campo será una cadena de caracteres entre 1 y 255 caracteres, usado para indicar el tipo de reporte que se va a crear.

- Etiquetas: Esta entrada será una cadena entre 1 y 255 caracteres que se usa para poner texto personalizado a los reportes tales como: título, cabeceras, pie de páginas o cualquier otro texto que se desee aparezca (opcional).
- Imagen: Este campo será una imagen que pueda ser usada como logotipo y que se desee adicionar al reporte, esta imagen estará referenciada por una cadena entre 1 y 255 caracteres que representará un camino en el lado del cliente (opcional).
- Número de Página: Este campo será un número entero usado para numerar las páginas que contendrá el reporte (opcional).

Salidas:

- Se mostrará el menú de creación de reportes según el tipo seleccionado pro el usuario.

RF9 Seleccionar y personalizar componentes.

Permitirá seleccionar y personalizar cada uno de los componentes que forman el reporte incluyendo imágenes, configuración de página (orientación del papel: vertical u horizontal), tipo de papel, márgenes de página y otros recursos como gráficos.

Entradas:

- Propiedades: Este campo permitirá configurarle las propiedades a cada uno de los componentes seleccionados, y sus valores varían según el componente y la propiedad que se desea configurar.
- Ubicación: Este campo será una coordenada x , y en la hoja del reporte donde se desea que este el componente, la cual será introducida y modificada arrastrando y soltando el componente en el área de diseño.

Salidas:

- Componente personalizado y ubicado en una posición de la hoja del reporte.

RF10 Filtros y parámetros del reporte.

Permitirá pasarle parámetros y filtros a los reportes, para esto en el caso de los procedimientos almacenados definidos por la oficina de tecnología e informática es necesario que la aplicación identifique que un procedimiento almacenado recibe o devuelve cursores y así se pueda hacer los pasos necesarios para obtener la información contenida en dichos cursores.

Entradas:

- Nombre del parámetro: Este campo será una cadena entre 1 y 255 caracteres (obligatorio).

- Valores manuales: Este campo será un arreglo de valores ya sean números de cualquier tipo o cadenas de entre 1 y 255 caracteres, separados por coma que especifican los posibles valores que puede tomar un parámetro determinado (opcional).
- Valores desde tabla: Este campo será una cadena entre 1 y 255 caracteres que indicará el nombre de la tabla y el campo desde donde se extraerán los posibles valores que se le pasarán a un determinado parámetro (opcional).

Salidas:

- Filtro que será la estructura interna necesaria para poder aplicar la condición impuesta sobre el campo.

RF11 Usar modelo semántico para diseñar el reporte.

El sistema debe permitir elegir un modelo semántico existente desde el cual se desea generar un reporte y seleccionar una tabla, una vista, una consulta, una función o procedimiento almacenado para extraer los datos a mostrar en el reporte.

Entradas:

- Modelo semántico: Este campo será el modelo semántico que se usará para diseñar el reporte y será una cadena entre 1 y 255 caracteres (obligatorio).

Salidas:

- Mostrará las tablas, vistas, funciones y procedimientos almacenados en el modelo semántico seleccionado.

RF12 Guardar reporte.

Permitirá guardar un reporte.

Entradas:

- Nombre del Reporte: Este campo será el nombre del reporte y consistirá en una cadena entre 1 y 255 caracteres (obligatorio).
- Categoría: Este campo será una cadena entre 1 y 255 caracteres e indicará la categoría en la cual se desea guardar el reporte (obligatorio).

Salidas:

- Mostrará un mensaje indicando que el reporte fue salvado satisfactoriamente o un mensaje de error en caso que no pueda ser salvado.

RF13 Guardar reportes como plantillas.

Permitirá guardar el diseño de un reporte sin asociarlo a ningún origen de datos para reutilizar el diseño de la plantilla.

Entradas:

- Nombre del Reporte: Este campo será el nombre del reporte y consistirá en una cadena entre 1 y 255 caracteres (obligatorio).
- Categoría: Este campo será una cadena entre 1 y 255 caracteres e indicará la categoría en la cual se desea guardar el reporte (obligatorio).

Salidas:

- Mostrará un mensaje indicando que la plantilla fue salvada satisfactoriamente o un mensaje de error en caso que no pueda ser salvado.

RF14 Importar plantillas de reportes.

Permitirá cargar una plantilla previamente diseñada para usarla y modificarla en caso necesario.

Entradas:

- Nombre de la plantilla: Este campo será el nombre con el que se importara la plantilla consistirá en una cadena entre 1 y 255 caracteres (obligatorio).

Salidas:

- Mostrará un mensaje indicando que la plantilla fue salvada satisfactoriamente o un mensaje de error en caso que no pueda ser salvada.

RF15 Seleccionar campos disponibles para el diseño del reporte.

Permitirá seleccionar los campos de las tablas, vistas, funciones procedimientos o consultas que se podrán mostrar en el reporte.

Entradas:

- Tablas: Este campo será un arreglo de cadenas entre 1 y 255 caracteres cada una que representarán los nombres de la tablas que forman parte del modelo semántico (opcional).
- Vistas: Este campo será un arreglo de cadenas entre 1 y 255 caracteres cada una que representarán los nombres de la vistas que forman parte del modelo semántico (opcional).
- Consultas: Este campo será un arreglo de cadenas entre 1 y 255 caracteres cada una que representarán los nombres de las consultas que forman parte del modelo semántico (opcional).

- Funciones o procedimientos almacenados: Este campo será un arreglo de cadenas entre 1 y 255 caracteres cada una que representarán los nombres de la funciones o procedimientos almacenados que forman parte del modelo semántico (opcional).

Salidas:

- Mostrará los campos seleccionados que formarán parte del diseño del reporte.

RF16 Realizar una vista previa del reporte.

Permitirá pre-visualizar el reporte para ver las características actuales y aspectos visuales del reporte.

Entradas:

Nombre del reporte: Este campo será una cadena de caracteres entre 1 y 255 caracteres que representará el nombre del reporte (obligatorio).

XML: Este consistirá en una cadena que contiene la estructura del reporte a visualizarse (obligatorio).

Consulta: Este campo será una cadena SQL, la cual debe ejecutarse para extraer la información del reporte (obligatorio).

Salidas:

Código HTML: Consistirá en un código HTML que representará una vista previa del reporte.

Luego de obtenidos y especificados los requisitos y de haber hecho un amplio estudio en el capítulo anterior sobre los patrones de caso de uso, se procede a la aplicación de los mismo. Se aplican los patrones: CRUD Completo y Parcial, Concordancia Reuso, Generalización/Especialización y Extensión Concreta, obteniendo de esta forma un total de 12 Casos de Uso del Sistema.

2.3.2 Requisitos No Funcionales.

RNF1 Requisitos de usabilidad.

El tiempo de entrenamiento requerido para que usuarios normales sean productivos operando el sistema debe ser entre 15 y 30 días. En el caso de usuarios avanzados aproximadamente 15 días máximo. La herramienta debe ser WEB, pero con características muy similares a las aplicaciones de escritorio en cuanto al diseño de las interfaces visuales y los tiempos de respuesta de la interacción del usuario con el sistema.

- **Permitir personalización de los reportes.**

El usuario podrá diseñar un reporte lo más ajustado posible a sus necesidades, se permitirá cambiar la tipografía del texto, agregar imágenes, ubicar en el área de diseño los campos en la sección que el usuario desee y usando características de arrastrar y soltar.

- **Buscar de manera rápida y sencilla un reporte que se desee visualizar.**

El usuario podrá localizar un reporte de manera rápida, si es posible tenerlo ubicado en alguna categoría que permita tener la posibilidad de organizarlos por áreas temáticas comunes.

- **Diseñar una consulta SQL de manera sencilla y ágil.**

El usuario podrá diseñar una consulta SQL de manera ágil y sencilla sin necesidad de ser un experto en el lenguaje SQL.

RNF2 Requisitos de fiabilidad.

El sistema debe estar disponible las 24 horas del día. En caso de fallo, pudiera estar fuera de servicio por un período de 72 horas máximo. La precisión y exactitud de las salidas del sistema se corresponden con la calidad y exactitud de la información contenida en las base de datos desde donde se extraigan los reportes. El sistema no será responsable por la falta de veracidad de dicha información. Algunos errores que pueden resultar críticos son:

- Que salgan de funcionamiento las bases de datos desde donde se extraen los reportes o que no exista conectividad hacia ellas.
- Que falle el servidor donde se despliegue la solución.

RNF3 Requisitos de eficiencia.

La eficiencia del sistema depende en gran medida de la velocidad de conexión a las base de datos donde se encuentre, así como del volumen de información contenido en las mismas. Sin embargo el Sistema debe mantener tiempos de respuestas en un marco razonable, con tal fin se implementará un mecanismo de paginación de los reportes que permitirá que los mismos sean obtenidos de manera progresiva disminuyendo así el tráfico por la red de la información y las consultas que devuelven grandes cantidades de registros.

- **Tiempo de máximo de respuesta para la obtención de un reporte.**

El tiempo máximo para la obtención de un reporte no debe sobrepasar los 5 minutos.

- **Tiempo de promedio de respuesta para la obtención de un reporte.**

Como promedio un reporte debe demorar alrededor de 10 segundos.

- **Cantidad de usuarios conectados de forma simultánea.**

El sistema debe permitir que existan al menos 100 usuarios conectados de forma simultánea.

RNF4 Requisitos de arquitectura y plataforma.
El sistema debe ser implementado en el lenguaje de programación PHP versión 5.2 o superior. Como marco de trabajo se usará Symfony el cual propone una arquitectura modular en tres capas: el modelo, la vista y el controlador.

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

Otra librería importante y necesaria en el desarrollo de la herramienta será Jasper Reports la cual constituye el núcleo del proceso de generación de reportes.

RNF5 Requisitos de soporte del sistema.
El sistema contará con una plataforma para el soporte que consta de un chat online así como de un sistema de gestión de incidencias el cual es descrito detalladamente en el Proyecto Técnico.

RNF6 Requisitos para la documentación y ayuda del sistema.
El sistema contará con una ayuda, la misma se entregará en formato HTML y se ubicará como un enlace dentro del sistema. La ayuda debe cubrir todos y cada uno de los subsistemas, se escribirá en lenguaje español lo más clara posible. Además se entregarán como materiales adicionales los cursos de capacitación y transferencia, que incluyen presentaciones y documentos con actividades para el aprendizaje de la herramienta.

RNF7 Requisitos de adquisición de componentes.
Durante el proceso de análisis de requisitos se detectó que no todos los componentes que se utilizarán son libres, así es el caso de la librería ExtJS. Dicha librería cuenta hoy en día con dos licencias la comercial y la libre (LGPL), pero por problemas externos al equipo de desarrollo, solamente se cuenta con la licencia LGPL, con la cual se trabajará para la realización de esta versión.

RNF8 Requisitos de Interfaz.
El usuario debe acceder a la aplicación a través del protocolo HTTPS usando el navegador Firefox versión 2.0 o superior.

- **Interfaces de usuario.**

Las interfaces de usuario serán diseñadas a modo de aplicaciones RIA (Rich Internet Application) lo que permite a los usuarios contar con aplicaciones web con una experiencia de usuario similar a la de

las aplicaciones de escritorio. Para lograr este fin se usará la librería javascript Extjs la cual conjuga una serie de componentes visuales que proveen funcionalidades, que ayudan al diseño de este tipo de aplicaciones WEB con apariencia de escritorio.

- **Interfaces de hardware.**

Por su naturaleza, el sistema podrá interactuar solamente con una interfaz de hardware que es la impresora. Esta interacción tendrá lugar cuando se necesite imprimir algún reporte en formato duro.

- **Interfaces de software.**

El sistema contará con un API como mecanismo de integración con otros sistemas, este mecanismo debe ser independiente del lenguaje de programación en el cual se encuentre la aplicación que lo utilizará.

- **Interfaces de comunicación.**

El sistema puede ser desplegado sobre red LAN, MAN, o WAN siempre y cuando la velocidad de conexión sea mayor que 1 Mbit/s.

RNF9 Software requerido para desplegar y utilizar la aplicación.
Servidor para instalar la aplicación

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

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

Servidor para instalar la de Base de Datos

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

- SO: GNU/Linux preferentemente Ubuntu GNU/Linux 8.04 o superior, Debian 4 GNU/Linux o superior.
- PostgreSQL versión 8.3 o superior.
- PGAdmin III o algún administrador para postgreSQL.
- Usuario con privilegios para instalar la base de datos.
- PostgreSQL debe estar correctamente configurado para aceptar conexiones vía TCP/IP utilizando el método de autenticación por md5.

RNF10 Hardware requerido para desplegar y utilizar la aplicación.

PC Cliente

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

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

PC Servidor

- 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.

2.4 Definición de Caso de Uso del Sistema.

Un caso de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas.

2.4.1 Definición de los actores.

A continuación, en la **Tabla 1** se muestra la descripción de los actores obtenidos de los Módulos Diseñador de Modelos y Diseñador de Reportes; y en la **Figura 3** se representa el diagrama de los mismos.

Tabla 1: Descripción de los actores. 1

Actor	Descripción
Diseñador de modelo	Actor humano. Interactúa con el módulo “Diseñador de modelo”. Responsable de la Gestión de los Orígenes de Datos y los Modelos de Datos.
Diseñador de reporte	Actor humano. Interactúa con el módulo “Diseñador de reporte”. Responsable de los diseños de los reportes tabulares o de tablas personalizadas o cruzadas.

2.4.2 Diagramas de Caso de Uso del Sistema.

Las siguientes figuras representan los casos de uso referentes a los módulos Diseñador de Modelos y Diseñador de Reportes.

A continuación de cada diagrama se presenta la descripción del caso de uso correspondiente.

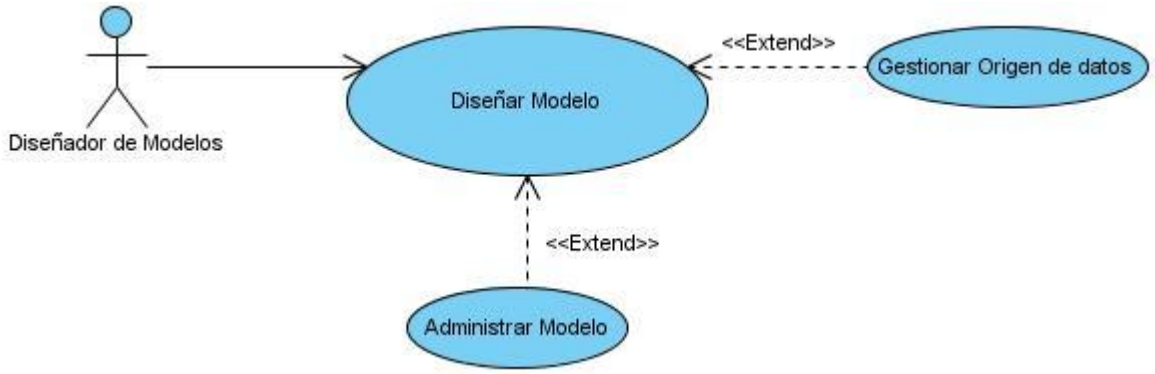

Figura 3: Diagrama de caso de uso del sistema para el módulo Diseñador de Modelo. 1

Tabla 2: Descripción del Caso de Uso Diseñar Modelo. 1

Objetivo	Diseñar un modelo
Actores	Diseñador de Modelo: Inicia el CU mostrando los modelos y orígenes de datos y crea nuevos orígenes de datos para crear modelos.
Resumen	El caso de uso comienza al seleccionar la opción “Diseñador de modelos”, muestra los modelos y orígenes de datos que se encuentren creados. Además permite crear nuevos modelos, a partir de un origen de datos seleccionado y buscar un modelo específico en la lista de los modelos previamente creados. Termina al crearse y guardarse un nuevo modelo.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	Tiene que existir al menos un origen de datos creado. La ejecución de las secciones “ Buscar”, “Modificar”, “Renombrar”, “Eliminar” modelo dependen de que se haya ejecutado el Flujo Básico del Caso de

	Uso.	
Postcondiciones	Se carga la interfaz de diseñar de modelo.	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	. El actor selecciona la opción “Diseñador de modelos”.	2. El sistema muestra una interfaz brindándole un listado de modelos. Si el actor decide crear un nuevo modelo, ver Sección: “Adicionar modelo”; si decide buscar un modelo, ver Sección: “Buscar modelo”; si decide modificar un modelo existente, ver**¡Error! No se encuentra el origen de la referencia.**, Sección: “Modificar modelo”; si decide renombrar un modelo existente, ver**¡Error! No se encuentra el origen de la referencia.**, Sección: “Renombrar modelo”; si decide eliminar un modelo existente, ver**¡Error! No se encuentra el origen de la referencia.**, Sección: “Eliminar modelo”; si el actor decide adicionar un nuevo origen de datos, ver Sección: “Adicionar origen de datos”; si el actor decide eliminar un origen de datos ver Sección: “Eliminar origen de datos”; si el actor decide modificar un origen de datos ver Sección: “modificar origen de datos”**¡Error! No se encuentra el origen de la referencia.**; si el actor decide cerrar el Diseñador de modelo, ver Sección: “Cerrar el diseñador de

		modelos”.
Sección 1: “Adicionar modelo”		
Flujo básico Diseñar Modelo		
	Actor	**Sistema**
1.	El actor selecciona un origen de datos.	2. El sistema resalta el origen de datos seleccionado.
3.	El actor selecciona la opción “Siguiente”.	4. El sistema muestra las entidades pertenecientes al origen de datos, agrupadas por tablas, vistas y rutinas.
5.	El actor selecciona las entidades que conformarán el modelo semántico.	6. El sistema habilita la opción “Siguiente”. En caso de que el actor seleccione la opción “Anterior”, ver Flujo Alterno 4.1.
7.	El actor selecciona la opción “Siguiente”.	8. El sistema muestra los campos asociados a cada entidad previamente seleccionada, ver Figura 6: Panel de visualización de entidades para asociarlas a un modelo y el botón “Aceptar” para finalizar la construcción del modelo. En caso de que el actor seleccione la opción “Anterior”, ver Flujo Alterno 8.1.
9.	El actor selecciona la opción “Aceptar”.	10. El sistema muestra una ventana para introducir el nombre del modelo que se desea generar. En caso de que el actor seleccione la opción “Cancelar”, ver Flujo

		Alterno 10.1.
11.	El actor introduce un nombre para el nuevo modelo a generar.	12. El sistema habilita la opción “Aceptar”. En caso de que el actor seleccione la opción “Cancelar”, ver Flujo Alterno 10.1.
13.	El actor selecciona el botón “Aceptar”.	14. El sistema genera el modelo semántico y carga la interfaz con el listado de los orígenes de datos.
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		
	Actor	**Sistema**
		4.1. El sistema muestra la lista de orígenes de datos existentes.
		8.1. Retorno al paso 4 del Flujo Normal de Eventos.
		10.1. Cierra la ventana donde se debe introducir el nombre del modelo a generar.
Sección 1: “Buscar modelo”		
1.	El actor escribe el nombre del modelo que desea buscar.	2. El sistema muestra el modelo buscado en la lista de modelos. Si el actor introduce un nombre incorrecto o el listado de modelos se encuentra vacío, ver Flujo Alterno 2.1.
Flujos alternos		
Nº Evento <Condición que dio lugar a la extensión>		

	Actor	Sistema
		2.1. El sistema muestra el listado de modelos vacío.
Sección 1: "Cerrar diseñador de modelo"		
		1. El sistema destruye todas las instancias creadas en la interfaz de "Diseñar modelo" terminando así el caso de uso.

Relaciones	**CU Incluidos**	
	CU Extendidos	CUS Gestionar Origen de datos <extendido>, CUS Administrar Modelo <extendido>.

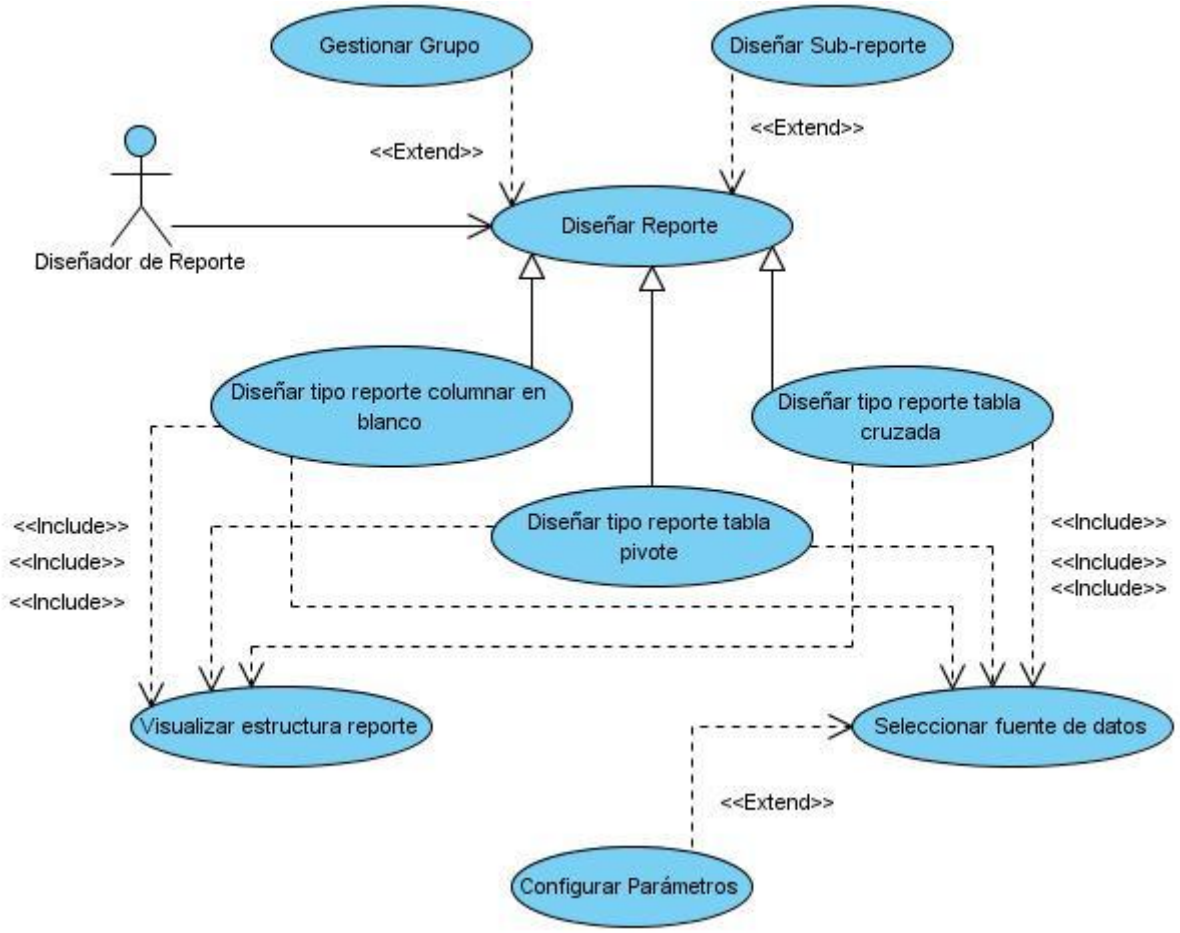

Figura 4: Diagrama de caso de uso del sistema para el módulo Diseñador de Reportes. 1

Tabla 3: Descripción del Caso de Uso Diseñar Reporte. 1

Objetivo	Diseñar reporte.
Actores	Diseñador de Reporte: Crea reportes.
Resumen	El presente caso de uso permite al actor realizar el diseño de reportes columnares, con tablas cruzadas o tablas pivotes, comienza al seleccionar la opción "*Diseñar reporte*". El mismo ha de estar compuesto por campos asociados a un modelo registrado en el sistema. El reporte puede ser visualizado en formato HTML. El módulo permite salvar y cargar reportes y plantillas. El caso de uso finaliza al guardar un nuevo reporte.

Complejidad	Alta
Prioridad	Crítica
Precondiciones	Debe existir al menos un “Modelo” de datos registrado en el sistema para poder especificar la “Fuente de datos”. Para el diseño del reporte, se debe asignar al mismo de manera obligatoria una “Fuente de datos” partiendo de un “Modelo” existente. Debe existir al menos un campo de la “Fuente de datos” seleccionada en el “Área de diseño de reporte” para poder visualizar la “Vista previa” del reporte. Debe existir al menos un campo de la “Fuente de datos” seleccionada en el “Área de diseño de reporte” para poder “Salvar como Reporte. Debe existir al menos un campo de la “Fuente de datos” seleccionada en el “Área de diseño de reporte” para poder “Salvar como Plantilla”. La propiedad “Tipo de reporte” de la “Tabla de propiedades” tiene como valor por defecto: Reporte tabular.
Postcondiciones	Es diseñado un nuevo reporte.
Flujo de eventos	
Flujo básico Diseñar Reporte	

	Actor	**Sistema**
1.	El caso de uso inicia cuando el actor selecciona la opción de *“Diseñar reporte”*.	2. El sistema muestra un interfaz que le permitirá al actor seleccionar la opción que desea acometer en el caso de uso. De manera predeterminada el sistema mostrará seleccionada la opción de “Crear un reporte columnar en blanco”. En caso que el actor

		oprima el botón “Cancelar”, ver *Flujo Alterno 2.1*.
3.	El actor selecciona la opción que desea realizar.	4. En caso de haber seleccionado la opción de “Crear un reporte estadístico de:” el sistema cargará en la lista desplegable los dos tipos de análisis estadístico permitidos: “tabla cruzada” y “tabla pivote”. En caso de haber seleccionado la opción “Abrir un elemento existente”, ver *Flujo Alterno 4.1*.
5.	El actor oprime el botón “Aceptar”	6. En caso que el actor haya seleccionado la opción: “Crear un reporte columnar en blanco”, ver Sección: “Diseñar reporte en blanco”; en caso que haya seleccionado la opción: “Crear un reporte estadístico de tabla cruzada”, ver caso de uso especializado **¡Error! No se encuentra el origen de la referencia.**; en caso que haya seleccionado la opción: “Crear un reporte estadístico de tabla pivote”, ver caso de uso especializado **¡Error! No se encuentra el origen de la referencia.**.
Flujos alternos		
		2.1 Termina el caso de uso.
		El sistema carga la lista de reportes y plantillas existentes para que el actor seleccione uno de ellos, se utilizan los

		componentes de las secciones “Abrir reporte”, ver Sección: "Abrir reporte" y “Abrir plantilla”, ver Sección: "Abrir plantilla"
Sección 1: “Crear reporte columnar en blanco”		
	Actor	**Sistema**
1.		• El sistema construye la interfaz que permitirá la creación de un reporte. En la misma se visualiza la “Paleta de componentes”, se visualiza el “Inspector” de reportes y de fuente de datos.
Flujos alternos		
	Actor	**Sistema**
Sección 1: “Abrir reporte”		
1.	El actor selecciona la opción de “Abrir reporte”.	2. El sistema muestra una interfaz para seleccionar el reporte a cargar. En caso que no haya reportes registrados en el sistema, ver *Flujo Alterno 2.1*. La interfaz muestra un botón “Cancelar” para ignorar la realización de la operación, en caso que el actor presione dicho botón, ver *Flujo Alterno 2.1*.

3.	El actor selecciona el reporte que desea cargar y oprime el botón “Aceptar”.	4. El sistema lee las configuraciones del reporte. 5. El sistema actualiza la “Fuente de datos”. 6. El sistema actualiza el “Inspector” de reporte y fuente de datos. 7. El sistema actualiza el “Área de diseño del reporte”. 8. El sistema pasa al paso 3 del *Flujo Básico*. 9. Termina la sección.
Flujos alternos		
	Actor	**Sistema**
		2.1 El sistema muestra la interfaz de selección de reporte vacía y el botón “Aceptar” deshabilitado, pudiendo sólo “Cancelar” la operación, ejecutando el paso 9 del *Flujo Normal de Eventos*.
Sección 1: “Abrir plantilla”		
1.	El actor selecciona la opción de “Abrir plantilla”.	2. El sistema muestra una interfaz para seleccionar la plantilla a cargar. En caso que no haya plantillas registradas en el sistema, ver *Flujo Alterno 2.1*. La interfaz muestra un botón “Cancelar” para ignorar la realización de la operación, en caso que el actor presione dicho botón, ver *Flujo Alterno*

		2.1.
3.	El actor selecciona la plantilla que desea cargar y oprime el botón “Aceptar”.	4. El sistema lee las configuraciones de la plantilla. 5. El sistema actualiza el “Inspector" de reporte. 6. El sistema actualiza el “Área de diseño del reporte”. 7. El sistema pasa al paso 3 del *Flujo Básico*. 8. Termina la sección.
Flujos alternos		
	Actor	**Sistema**
		2.1 El sistema muestra la interfaz de selección de plantilla vacía y el botón “Aceptar” deshabilitado, pudiendo sólo “Cancelar” la operación, ejecutando el paso 8 del *Flujo Normal de Eventos*.

Relaciones	**CU Incluidos**	
	CU Extendidos	CUS Gestionar Grupo CUS Diseñar Sub-reporte
Requisitos no funcionales	Usabilidad.	
Asuntos pendientes	*[Posibles mejoras al caso de uso.]*	

2.5 Diagramas de Clases y de Colaboración del Análisis.

En el documento se han plasmado los diagramas de clases y de colaboración del análisis correspondiente a la descripción de los casos de usos anteriormente descritos con el objetivo de describir la estructura del sistema, los restantes quedan reflejados en los anexos del documento.

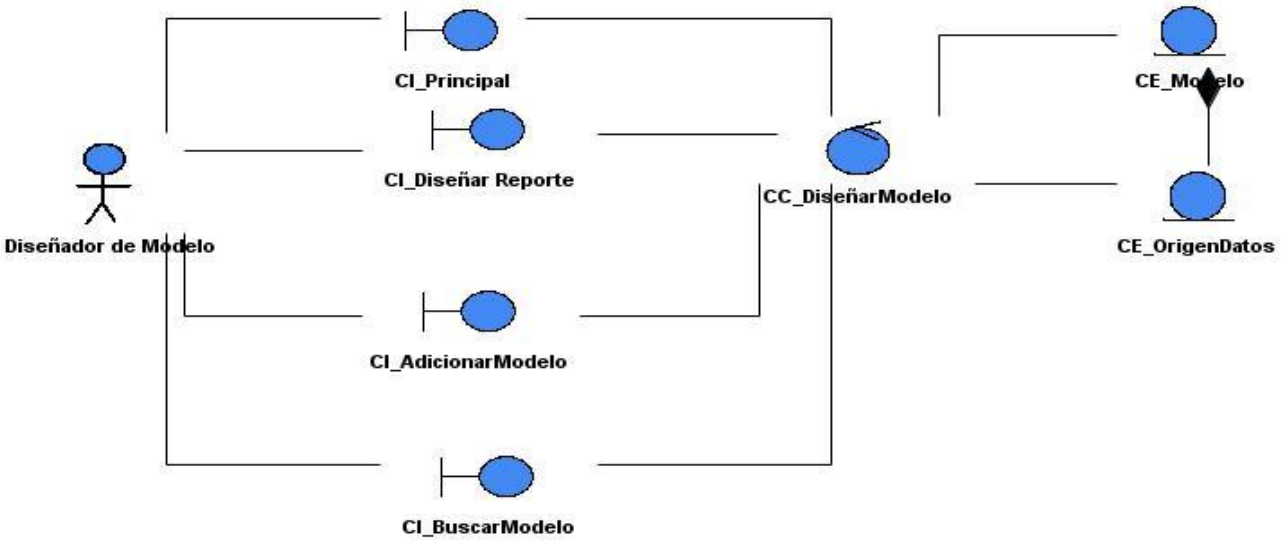

Figura 5: Diagrama de Clases del Análisis del módulo Diseñador de Modelos (CUS Diseñar Modelo) 1

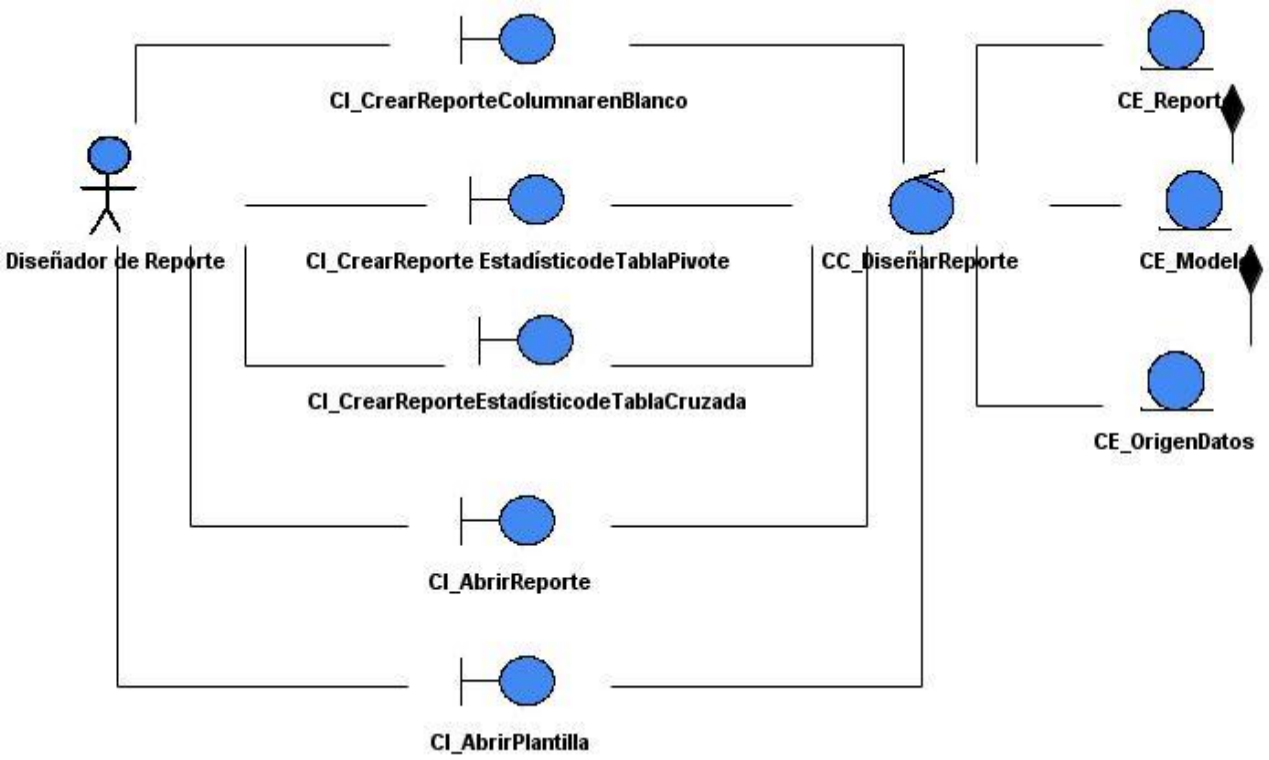

Figura 6: Diagrama de Clases del Análisis del módulo Diseñador de Reportes (CUS Diseñar Reporte) 1

A continuación se representan los diagramas de colaboración de los escenarios más significativos de los casos de uso Diseñar Modelo y Diseñar Reporte respectivamente.

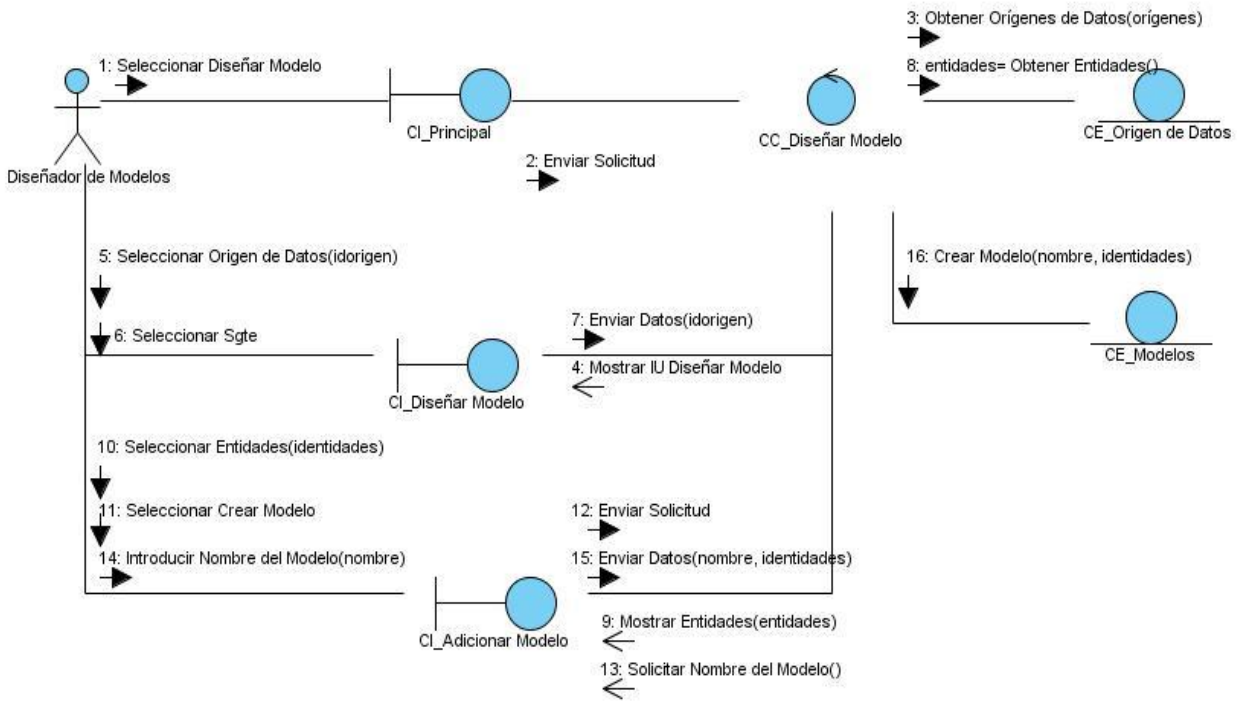

Figura 7: Diagrama de Colaboración del módulo Diseñador de Modelos (CUS Diseñar Modelo, escenario Adicionar Modelo) 1

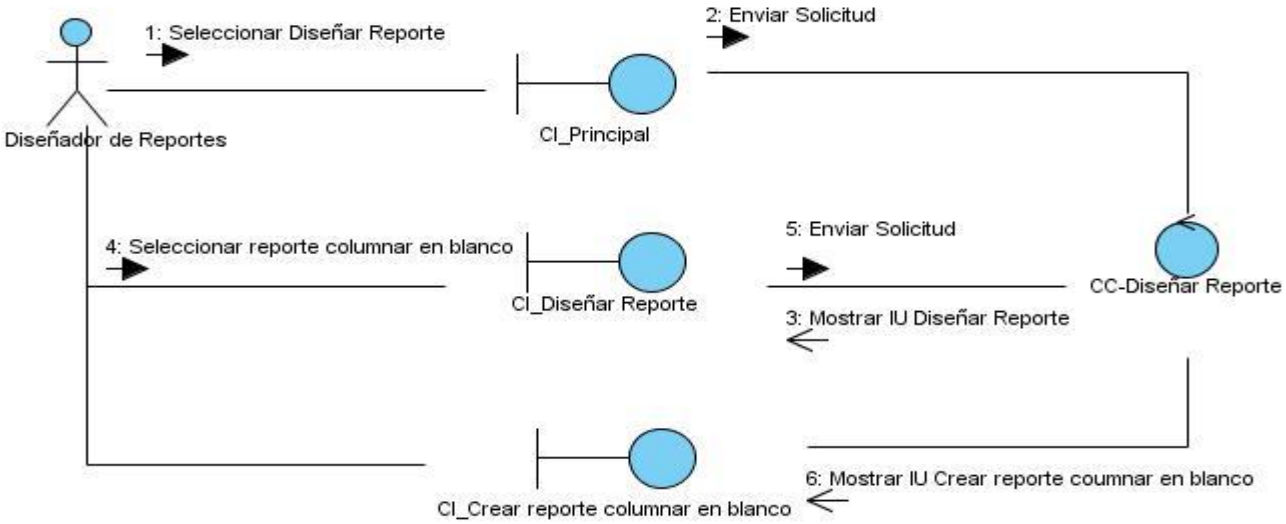

Figura 8: Diagrama de Colaboración del módulo Diseñador de Reportes (CUS Diseñar Reporte, escenario Crear reporte columnar en blanco). 1

2.6 Gestión de Requisitos.

La herramienta OSRMT, permite la administración de requisitos mediante la matriz de trazabilidad, que es una representación gráfica de las relaciones entre dos productos del proceso de desarrollo. Generalmente identificadas en las intersecciones de líneas verticales y horizontales. Para lograr un correcto seguimiento de los requisitos en todo el proceso de desarrollo del proyecto.

La matriz de trazabilidad se usa también para ver si los casos de uso satisfacen todos los requisitos del sistema. Se pueden trazar varios tipos de matrices de trazabilidad. Algunos ejemplos pueden ser:

- Requisitos frente a fases de desarrollo de software.
- Requisitos frente a actores.
- Requisitos frente a procesos del negocio.
- Requisitos frente a casos de uso.

Este último tipo se utiliza para comprobar si los requisitos del proyecto actual se están cumpliendo. Garantiza la integridad de los requisitos al implementar cambios en el sistema.

Requerimientos Funcionales	Diseñador de Modelos		Diseñador de Reportes								
	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11
RF 1	X										
RF 2	X		X								
RF 3	X										
RF 4	X										
RF 5	X		X								
RF 6	X		X								
RF 7	X		X								
RF 8	X	X									
RF 9	X										
RF 10	X	X									
RF 11	X	X									
RF 12	X										
RF 13	X										
RF 14	X		X								
RF 15	X	X									
RF 16	X		X								
RF 17	X	X									
RF 18	X	X									
RF 19	X	X									
RF 20				X							
RF 21					X						
RF 22						X					
RF 23				X	X			X			
RF 24										X	
RF 25								X			
RF 26				X	X		X				
RF 27				X							
RF 28				X							
RF 29				X							
RF 30							X	X			
RF 31								X	X		
RF 32				X	X	X					
RF 33				X	X	X					
RF 34				X	X	X					
RF 35				X	X	X					X

Figura 9: Matriz de Trazabilidad. 1

Como se puede apreciar todos los requisitos están asociados con al menos un caso de uso, validando que todos los requisitos están satisfechos.

2.7 Estimación de costo del proyecto.

Durante el proceso de análisis además se realizó una estimación del presupuesto del proyecto, siguiendo la línea propuesta por la dirección de la Universidad, utilizando para ello la planilla de estimación de costos que propone el Proceso de Mejora, arrojando un presupuesto $52914,80 y en resumen plantea que el proyecto debe tener una duración de 2 años con un total de tiempo de desarrollo de 2200 horas, para la obtención final del producto. Para más detalle referirse al documento Métodos de Estimación anexo a la tesis.

2.8 Conclusiones.

En el presente capítulo se realizó un profundo análisis de los Módulos Diseñador de Modelos y Diseñador de Reportes, a partir del cual se obtuvo la realización del modelo de dominio. Se especificaron los requisitos del software tanto funcionales como no funcionales y mediante la aplicación de patrones de casos de uso que permitieron englobar los requisitos, se pudieron definir los casos de uso del sistema. También se realizó la matriz de trazabilidad con el objetivo de verificar la correspondencia de los requisitos con al menos un caso de uso, además de estimar el presupuesto total del proyecto. En resumen se definieron todos los elementos necesarios para el correcto análisis de la solución.

CAPITULO 3: VALIDACIÓN DE LOS REQUISITOS

Introducción.

En el presente capítulo se realiza el análisis de los resultados alcanzados en el capítulo anterior, con el objetivo de evaluar los artefactos generados. Además, se plantean los resultados obtenidos y se analizan teniendo en cuenta las diferentes técnicas aplicadas a cada artefacto generado, con el objetivo de verificar la calidad con que fueron realizados.

3.1 Validación de requisitos.

Los requisitos una vez definidos necesitan ser validados. La validación es la actividad que permite demostrar que los requisitos definidos en el sistema son los que realmente quiere el cliente; además revisa que no se haya omitido ninguno, que no sean ambiguos, inconsistentes o redundantes. Además garantiza que todos los requisitos presentes en el documento de especificación sigan los estándares de calidad, que sean medibles, no ambiguos, posibles de probar, por mencionar algunos. [17]

Existen varias técnicas para la realización de la validación, la mayoría se destinan a revisar los modelos obtenidos en la definición de requisitos. A continuación se muestran las que se usaron para verificar la calidad de los artefactos del análisis.

- Revisiones (Reviews o Walk-throughs): Es un proceso manual, en el que se involucran varias personas para verificar el documento final de requisitos. Participan tanto el personal del cliente como los desarrolladores, en la revisión de anomalías y omisiones. El equipo de revisores debe verificar la consistencia de cada requisito y la integridad de los mismos como un todo, también se comprueba que el documento de requisitos cumpla con los criterios establecidos de validación. Los conflictos, contradicciones, errores y omisiones deben señalarse durante la revisión y registrarse formalmente. [18]

- Generación de casos de prueba (test de requisitos): Esta técnica tiene como objetivo comprobar la verificabilidad de los requisitos. Consiste en la definición de casos de prueba que permitan verificar el cumplimiento de los requisitos funcionales. El único método existente para comprobar la verificabilidad de un requisito es que sea posible definir uno o varios casos de prueba para dicho requisito. Los casos de prueba son artefactos bien definidos en el contexto de la prueba del software. En dicho contexto, un caso de prueba es la descripción de una acción bien definida que se debe realizar con el software. Por acción bien definida, debe entenderse que están perfectamente descritos tanto los datos de entrada como las tareas a

realizar y los resultados esperados. Durante la validación de requisitos, los casos de prueba se utilizan del mismo modo que durante la prueba del software; es necesario poder describir, como se ha indicado anteriormente, tanto los datos de entrada como las tareas a realizar y los resultados esperados, para lo cual es necesario que estén perfectamente descritos los requisitos.

- Prototipos: permiten llevarse una idea de la estructura de la interfaz del sistema con el usuario. Un prototipo es una versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y entender mejor el problema y su solución. Un prototipo puede revelar errores u omisiones en los requisitos propuestos, favorece la comunicación entre clientes y desarrolladores, da una primera visión del producto. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final [16].Es una de las propuestas más usadas en la validación de los requisitos actualmente, por los beneficios que reportan a los usuarios y desarrolladores.

3.2 Validación por fases.

Para garantizar la calidad de los artefactos del análisis de los módulos Diseñador de Modelos y Diseñador de Reportes, se pretende validar los documentos que aportan gran flujo de información referente a los requisitos de software, los cuales son:

- Documento de especificación de requisitos de software.
- Modelo de casos de uso del sistema.

La validación de requisitos plantea un conjunto de actividades que sirven de guía para el proceso de validación, se han dividido en tres fases para una mejor organización:

I: Validar los requisitos de software.

II: Validar el documento de especificación de requisitos de software.

III: Validar el modelo de casos de uso del sistema.

3.2.1 Fase I: Validación del listado de requisitos de software.

Durante esta fase se realizaron varias actividades para validar los requisitos, se aplicó la técnica revisiones para corregir la información del listado de los mismos, para comprobar que se encuentren contenidos en un documento tangible tanto para los clientes y desarrolladores. Además de verificar que los requisitos no sean ambiguos, siendo estos posibles de probar, así como que describan de forma concisa qué es lo que debe hacer el sistema, entre otros factores.

Para esta primera fase se aplicó además, las listas de chequeo del Centro de Calidad de Software (Calisoft) como:

- Lista de chequeo Especificación de Requisitos.
- . Criterios para validar requisitos del cliente.
- Criterios para validar requisitos del producto.

Estas listas permitirán aplicar un conjunto de preguntas a los requisitos, para verificar que cumplan con todos los parámetros necesarios. Además de identificar los elementos de entrada y salida y evaluar si su impacto es positivo. En esta fase se desarrollaron 2 revisiones para validar el listado de requisitos, a continuación se presentan los resultados de cada una de ellas.

Revisión 1:

En esta primera revisión se detectó que existían requisitos que no tenían muy bien especificadas las entradas y salidas. También se observó que según las nuevas plantillas para el Proceso de Mejora de Calisoft, los tipos de datos de las entradas y salidas no eran los correctos.

Revisión 2:

Al realizar la segunda revisión se habían solucionado los errores que se detectaron en la anterior. Como resultado de las 2 revisiones, se puede afirmar que los requisitos obtenidos en el análisis realizado y los resultados alcanzados en esta etapa de definición son correctos.

3.2.2 Fase II: Validación del documento de especificación de requisitos.

Para la realización de esta fase, se validó el documento de especificación de requisitos de software (ERS), para verificar la calidad del mismo. Se aplicó la técnica de revisiones con la lista de chequeo especificación de requisitos como se muestra en las siguientes figuras, para verificar una parte del documento.

Estructura del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
crítico	1. ¿Está el documento acorde con a la plantilla estándar del proyecto o del expediente de proyecto?	B		Ninguno	
crítico	2. ¿Contiene las secciones obligatorias definidas en el expediente? (Ver Expediente de Proyecto)	B		Ninguno	

Figura 10: Elementos definidos en la lista de chequeo 1

Semántica del documento					
Peso	**Indicadores a Evaluar**	**Eval**	**(NP)**	**Cantidad de elementos afectados**	**Comentarios**
Crítico	1. ¿Ha identificado errores ortográficos?	B		Ninguno	
Crítico	2. ¿Se entiende claramente lo que se ha especificado en el documento?	B		Ninguno	
	3. ¿El número de página que aparece en el índice coincide con el contenido que se refleja realmente en dicha página?	B		Ninguno	
	4. ¿El total de páginas que aparecen en las reglas de confidencialidad coincide con el total de páginas que tiene el documento?	B	NP	Ninguno	No aparece en el documento de las nuevas planillas de proceso de mejora

Figura 11: Elementos definidos en la lista de chequeo 1

Una vez aplicada la lista de chequeo al documento de Especificación de Requisitos se detectaron 4 No Conformidades, entre las que se encuentran, no homogeneidad en la realización de los prototipos funcionales y no inclusión de las respuestas válidas y no válidas de los valores de entrada. Para resolver estas no conformidades se hizo necesaria la realización de una nueva iteración de análisis y verificación del documento para validar que no existieran errores, luego de lo cual se pudo concluir que el documento alcanza una evaluación de B y con aseguramiento de calidad.

3.2.3 Fase III: Validación del Modelo de Caso de Uso del Sistema.

En esta última fase para corregir los errores que pudiesen quedar, se validó el documento de casos de uso del sistema, para ello se aplicó la técnica prototipos de interfaz y pautas para la descripción de los casos de uso, con el objetivo de verificar la calidad requerida en el proceso de análisis realizado, la completitud y consistencia de los CU.

Se determinó que el artefacto caso de uso del sistema posee la calidad deseada en la validación, mediante la realización de los prototipos de los CU, los que facilitaron visualizar mejor estas descripciones para que los especialistas tuvieran una idea de la estructura de la interfaz del sistema. Esta técnica fue fundamental en la validación y verificación de los requisitos, siendo un modelo que

integró las necesidades de los clientes y los requisitos funcionales capturadas en la etapa de Levantamiento de Requisitos.

Las pautas guiaron a la revisión de las descripciones de los CU, permitiendo verificar que las mismas cumplían con los parámetros definidos. Las que arrojaron un total de 5 No Conformidades, puesto que existían errores en la escritura, lo que pudiera provocar a un incorrecto entendimiento en el momento d implementación de los mismos.

Para la corrección del documento se hizo necesaria, la realización de una nueva revisión de la escritura de los CU, para verificar que las No Conformidades anteriormente detectadas, ya hubiesen sido erradicadas, conclusión a la que se arribó una vez culminada esta última revisión.

3.3 Conclusiones.

En el capítulo se describió los pasos para la validación de los requisitos detectados durante el levantamiento de requisitos, se validaron los mismos mediante diferentes técnicas tales como: revisiones y prototipos funcionales, las que permitieron lograr la validación de todos los artefactos obtenidos en el análisis. La utilización de estas técnicas permitió demostrar que en un proceso de creación de software, siempre ocurren errores técnicos y humanos, pero con una detección válida y temprana, dicho errores pueden corregirse, pudiendo así culminar las diferentes fases de desarrollo de un software con la calidad que el cliente y el grupo de trabajo espera.

Conclusiones.

Del estudio realizado en el presente trabajo se puede constatar que:

- Con el estudio de las diferentes herramientas generadoras de reportes en el mundo se logró detectar nuevas funcionalidades y requisitos útiles y necesarios para el desarrollo de la solución.
- Con el correcto seguimiento de los pasos que propone la metodología se logró la elaboración de los artefactos Modelo del Dominio, Especificación de Requisitos y Especificación de Caso de Uso, con la calidad requerida.
- Con las técnicas de validación y verificación de requisitos aplicadas a los artefactos obtenidos, se puede garantizar que todo el proceso de análisis se ha realizado correctamente y con calidad.
- Todo lo anterior permite reafirmar que el análisis detallado de los Módulos Diseñador de Modelos y Diseñador de Reportes constituye la base para el diseño e implementación del sistema, que sin lugar a dudas redundará en una reducción del tiempo y esfuerzo empleado por parte de los usuarios del sistema y en una mejora en la toma de decisiones.

Recomendaciones.

Al finalizar este trabajo quedan algunas recomendaciones que pueden servir de punto de partida para mejorar aún más el análisis obtenido:

1. Llevar a cabo la implementación del sistema basado en el análisis propuesto.

2. Continuar con las investigaciones para añadir nuevas funcionalidades al sistema y obtener mejoras en futuras versiones, logrando adecuarlo cada vez más a las necesidades de los usuarios.

Bibliografía.

Booch G., Rumbaugh, J., Jacobson, I. (1999). *Unified Modeling Language User Guide*. Addison-Wesley.

Cockburn, Alistair. *Agile Software Development*. Highsmith Series.Kendall & Kendall *Análisis Y Diseño De Sistemas 3ª. Edición* Páginas 15.16.17.18

CrystalReports.com. 2009. CrystalReports.com. [En línea] 2009. [Citado el: 2 de octubre de 2010.] http://www.crystalreports.com/.

Durán A., Bernárdez, B., Ruiz, A., Toro M. (1999). *A Requirements Elicitation Approach Based in Templates and Patterns*. Workshop de Engenharia de Requisitos. Buenos Aires, Argentina.

Escalona, M.J. (2002). *Metodología para el desarrollo de sistemas de información global: análisis comparativo y propuesta*. Department of Language and Computer Science. University of Seville. Seville, January 2002.

IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard Institute of Electrical and Electronics Engineers. 1990.

Ian Sommerville, Ingenieria de Software. Pearson, 2005

Larman, Craig. 2003. *UML y patrones. Introducción al análisis y diseño orientado a objetos. .* s.l. : PEARSON, 2003.

Liu, L., Yu, E. (2001). *From Requirements to Architectural Design using Goals and Scenarios* Proceedings of the 6th Micon Workshop. Canada.

María José Escalona, Nora Kosh. 2009. Ingeniería de requisitos en aplicaciones para la web. [En línea] Universidad de sevilla. españa, 2009. e-amanecer.com/licenciatura/docu/LSI-2002-4-1.pdf.

Michael Arias Chaves. **La Ingeniería de Requerimientos y su importancia en el desarrollo de proyectos de software**. Revista Intersedes, Universidad de Costa Rica, 2006

Murcia, Universidad de. 2010. Ingeniería de Software. [En línea] 2010. http://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html.

Nicolás Davyt Dávila. **Ingeniería de Requerimientos: Una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto**. Universidad ORT Uruguay, 2003

Pan, D., Zhu, D., Johnson, K. (2001). *Requirements Engineering Techniques*. Internal Report. Department of Computer Science. University of Calgary. Canada.

Paradigm, Visual. 2010. Visual Paradigm. [En línea] 2010. http://www.visual-paradigm.com.

Presman, Roger S. 2005. *Ingeniería de Software. Un enfoque práctico. Sexta Edición.* 2005.

Roger Pressman, Ingenieria de Software**: Un enfoque práctico**. Mcgraw Hill, 2006

Referencia Bibliográfica.

[1] Agata Report. *Agata Report.* [En línea] Solis, 2009. [Citado el: 20 de Febrero de 2009.] http://www.agata.org.br.

[2] CrystalReports.com. 2009. CrystalReports.com. [En línea] 2009. [Citado el: 2 de octubre de 2010.] http://www.crystalreports.com/.

[3] JasperForge.org. *JasperForge.org.* [En línea] Jaspersoft Corporation, 2009. [Citado el: 21 de Febrero de 2009.] http://jasperforge.org/website/jasperreportswebsite/trunk/highlights.html?group_id=252.

[4] OpenUp. *OpenUp.* [En línea] 2009. [Citado el: 18 de octubre de 2010.] http://epf.eclipse.org/wikis/openup/.

[5] Robles, L.A. *Sistema Seguidor de Objetos(1). 2003* [Citado el: 12 de noviembre de 2010.] Disponible en: http://ccc.inaoep.mx/~labvision/doo/proy/T72.pdf.

[6] Free Download Manager. 2010. Free Download Manager. [En línea] 2010. [Citado el: 15 de octubre de 2010.] Disponoble en: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/

[7] IEEE Software Requirement Engineering, Second Edition, Editado por Richard H. Thayer y Merlin Dorfman, IEEE Computing Society, New York, NY. 1997.

[8] Guideline for Requirement Management, U.S. Department of Energy, April 2000.

[9] Pressman_Cap_23_Estimacion_Proyectos. Sexta Edición.

[10] Nicolás Davyt Dávila. **Ingeniería de Requerimientos: Una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto**. Universidad ORT Uruguay, 2003.

[11] Roger Pressman, Ingenieria de Software**: Un enfoque práctico**. Mcgraw Hill, 2006.

[12] María José Escalona, Nora Kosh. 2009. Ingeniería de requisitos en aplicaciones para la web. [En línea] Universidad de sevilla. españa, 2009. e-amanecer.com/licenciatura/docu/LSI-2002-4-1.pdf.

[13] Jacobson, Booch y Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de software, UML y patrones. .* s.l. : PEARSON EDUCACION, 2000.

[14] Referencia >Larman, Craig**.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : Prentice Hall, 2004.

[15] Raymond Turner. "The Foundations of Specification". Journal of Logic and Computation, Vol. 15, No. 5 (October 2005).

[16] Ingeniería, Software 1. 2010. Fase de Inicio. Flujo de trabajo de requerimientos. 2009.

[17]Senn, James A. "Análisis y Diseño de Sistemas de Información". Segunda Edición. McGraw Hill. 1992.

[18] Medina, J.C. Análisis Comparativo de Técnicas, Metodologías y Herramientas de Ingeniería de Requerimientos. México : Centro de Investigación y de Estudios Avanzados del IPN, 2004.