Universidad de las Ciencias Informáticas
Facultad 6

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

Título: "Herramienta para la migración de la base de datos del Generador Dinámico de Reportes (GDR) V1.8 a V2.0".

Autores:

Yailema de la Caridad Medinilla Santo

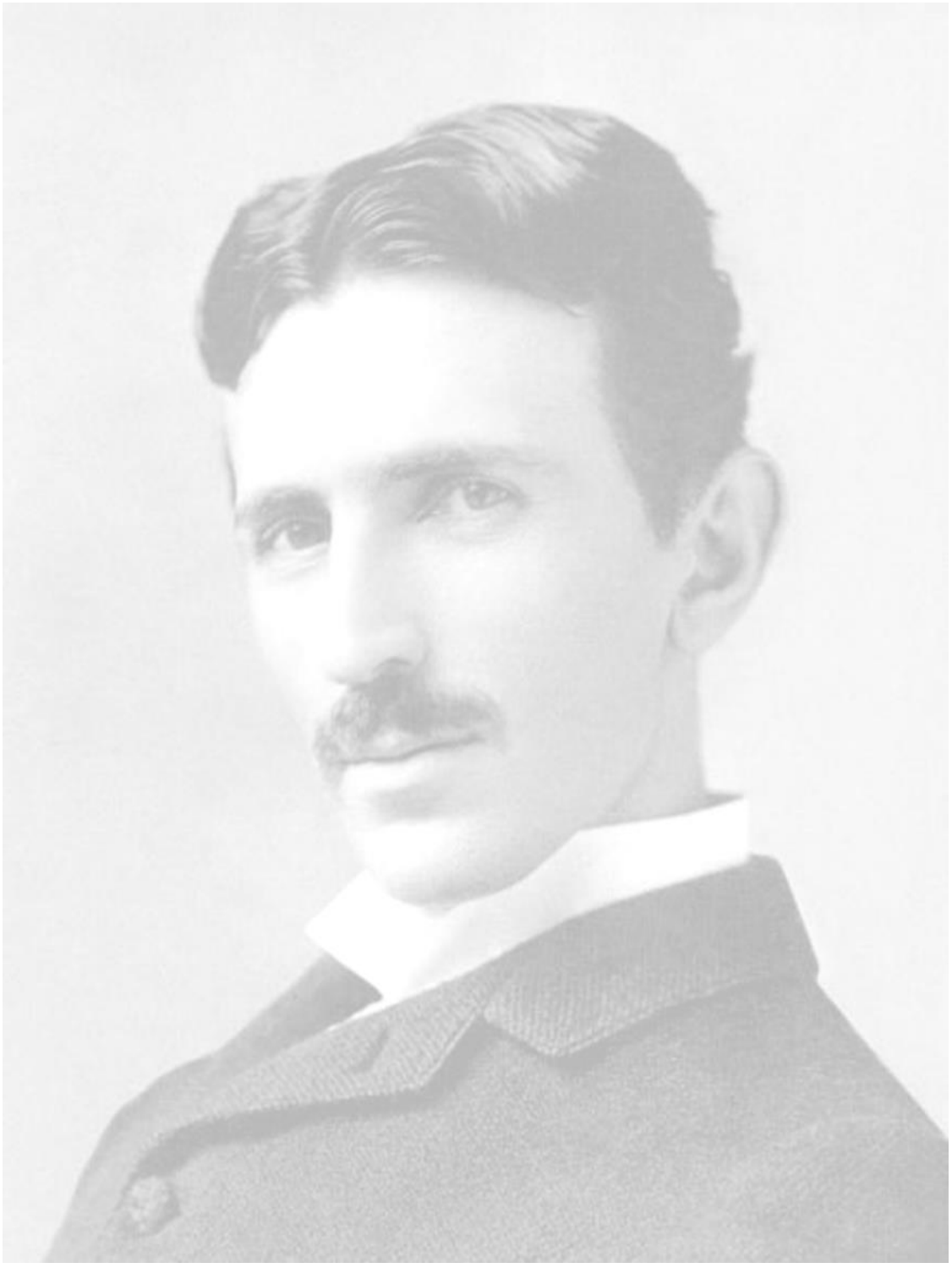
Marvel Álvarez Rojas

Tutores:

Ing. Claudia García Suarez del Villar.

Ing. Yasmany Hernández Hernández.

La Habana, mayo de 2013
"Año 55 de la Revolución"



"El futuro mostrará los resultados y juzgará a cada uno de acuerdo a sus logros"…

Nikola Tesla

Declaración de autoría

Declaramos que somos los únicos autores del trabajo titulado: “Herramienta para la migración de la base de datos del Generador Dinámico de Reportes (GDR) V1.8 a V2.0.”, y otorgamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Yailema de la Caridad Medinilla Santo

_____
Firma del autor

Marvel Álvarez Rojas

_____
Firma del autor

Ing. Claudia García Suarez del Villar

_____
Firma del tutor

Ing. Yasmany Hernández Hernández

_____
Firma del tutor

Datos de contacto

Autores

Autora: Yailema de la Caridad Medinilla Santo
Universidad de las Ciencias Informáticas
La Habana, Cuba
E-mail: ylmedinilla@estudiantes.uci.cu

Autor: Marvel Álvarez Rojas
Universidad de las Ciencias Informáticas
La Habana, Cuba
E-mail: marojas@estudiantes.uci.cu

Tutores

Tutora: Ing. Claudia García Suarez del Villar
Universidad de las Ciencias Informáticas
La Habana, Cuba
E-mail: cgarcias@uci.cu

Tutor: Ing. Yasmany Hernández Hernández
Universidad de las Ciencias Informáticas
La Habana, Cuba
E-mail: yhhdez@uci.cu

Agradecimientos

Hoy es un día muy especial y lleno de sentimientos encontrados, nostalgia porque se cierra un capítulo muy importante de mi vida, y alegría porque se abre uno nuevo. En este día se cumple uno de mis sueños y quisiera agradecerles a todas aquellas personas que de una forma u otra formaron parte de él y lograron que se hiciera realidad.

Primeramente quisiera agradecerle a mi madre por ser tan especial, por ser mi guía, mi ejemplo a seguir y la persona más importante de mi vida, te amo mami.

A mi otra madre, mi hermana Damaris por ser mi motor de inspiración, por estar conmigo en todo momento que la necesito, por enseñarme a enfrentar la realidad, por quererme como su hija, su hermana, su amiga, eres lo más grande que tengo en la vida, te adoro.

A mi papá por darme su apoyo y respetar siempre mis decisiones, a mi hermana Mayelin por todo su cariño, a mi cuñado mongui por ser como un padre para mí, a mi hermano Tomás, a mis sobrinos que los amo a todos, y a toda la familia en general por confiar siempre en mí y darme su apoyo.

A mi novio Odilen por darme todo su amor, su apoyo incondicional, su comprensión y soportarme en los momentos tristes y alegres, por estar al pendiente de mí en todo, por ser mi tutor de tesis, mi amor gracias por formar parte de mi vida, te amo.

A mis suegros por cuidar de mí en estos últimos meses como mis padres.

A mi dúo de tesis Marvel por estar siempre que lo necesité y trabajar juntos para llegar a esta meta.

A todos los compañeros con los que he compartido momentos inolvidables, en especial Jeamny y Geidy que aunque no están hoy aquí, no han dejado de preocuparse por mí, a Yuned, Popi, Sureny, Jorge, Migue, Dasley, Poti, que han sido mi familia en la UCI.

A mis nuevos compañeros Graciela, Yosbel y Yanet con los que he compartido buenos momentos en estos últimos años.

Agradecimientos

A mis tutores, a la profe Diana por dedicarle parte de su tiempo a revisar mi documento, a mi oponente Viviana que siempre estuvo dispuesta a ayudarnos. A todos quienes me brindaron su amistad e hicieron que mi vida universitaria sea una experiencia que no olvidaré jamás.

A todos muchas gracias ***Yailema****.*

Primero quiero agradecer a mi mamá por permitirme con su sacrificio llegar hasta aquí.

A mi familia que siempre me ha apoyado y brindado su mano cuando lo he necesitado.

A mis amigos los de antes, y los de la UCI, los que han estado en los malos y buenos momentos, de los que he necesitado y sin interés me han ayudado, a todos los que saben que puedo llamar amigo.

A mis compañeros del grupo 6506, con los que he pasado momentos inolvidables, a mi incomparable compañera de tesis y a mi novia, que me enseñó que siempre se puede un poco más.

A mis tutores por su ayuda y preocupación.

En general a todas la personas que contribuyeron con mi formación personal y profesional en todo el transcurso de mi vida en la universidad.

A todos muchas gracias ***Marvel****.*

Dedicatoria

A mi hermana Damaris por ayudarme a realizar mi sueño.

A mi mamá por todo su apoyo y dedicación.

A mi novio por hacerme feliz.

Yailema

A mi mamá por ser la mejor madre del mundo.

A mi familia por su preocupación.

A mis amigos, los de verdad.

Marvel

Resumen

El Generador Dinámico de Reportes forma parte de una solución integral de apoyo a la toma de decisiones desarrollada en la Universidad de las Ciencias Informáticas. Dicha herramienta permite generar reportes de forma dinámica, partiendo de información persistente en algún origen de datos soportado por el sistema. Actualmente la base de datos y los reportes generados con la versión 1.8 de la herramienta, son incompatibles con los de la versión 2.0, debido a que la versión 1.8 del Generador Dinámico de Reportes utiliza PHPReports como motor de generación de reportes y la versión 2.0 utiliza JasperReports, además en aras de agregar nuevas funcionalidades ha sido modificado el diseño de la base de datos del sistema. Esto trae como consecuencia que en las entidades donde se encuentra instalado dicho sistema no es posible actualizar a su nueva versión y reutilizar los reportes y demás datos generados con anterioridad. El objetivo del presente trabajo de diploma fue desarrollar una herramienta que lograra migrar la base de datos utilizada en la versión 1.8 del Generador Dinámico de Reportes a la nueva base de datos compatible con la versión 2.0. Permitiendo así, que usuarios de la versión 1.8 puedan actualizar sus sistemas a la versión 2.0 de la forma más rápida y transparente posible, beneficiándose de todas las nuevas características brindadas por esta sin la necesidad de diseñar nuevamente todos los reportes, modelos, consultas y demás recursos que se gestionan con la herramienta.

Palabras claves: base de datos**,** migrar base de datos, motor de reportes, origen de datos, reportes.

Índice

DECLARACIÓN DE AUTORÍA I
DATOS DEL CONTACTO II
AGRADECIMIENTOS III
DEDICATORIA V
RESUMEN VI
ÍNDICE VII
ÍNDICE DE FIGURAS X
INTRODUCCIÓN 1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5
Introducción 5
1.1 Conceptos y herramientas relacionados con el dominio del problema 5
1.1.1 Reporte 5
1.1.2 Sistemas generadores de reportes 5
1.1.3 Generador Dinámico de Reportes 6
1.1.4 Base de Datos 8
1.1.5 Transformación de Base de Datos 8
1.1.6 ETL 8
1.2 Análisis de soluciones existentes de herramientas ETL 10
1.2.1 Pentaho Data Integration-Kettle 10
1.2.2 Talend 12
1.2.3 CloverETL 13
1.3 Metodología de Desarrollo de Software 16
1.3.1 Proceso Unificado Abierto 17
1.4 Herramientas y Tecnologías 17
1.4.1 Lenguaje Unificado de Modelado 18
1.4.2 Herramienta CASE 18
1.4.3 Sistema Gestor de Base de Datos 19
1.4.4 Entorno de desarrollo 20
1.4.5 Lenguaje de Programación 20

1.4.6 Lenguaje de Marca 21
1.5 Conclusiones parciales 22
CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA 23
Introducción 23
2.1 Modelo de Dominio 23
2.2 Requisitos del sistema 24
2.2.1 Requisitos funcionales 25
2.2.2 Requisitos no funcionales 26
2.3 Diagrama de Caso de Uso del Sistema 27
2.4 Modelo de diseño 32
2.4.1 Diagrama de Clase del Diseño 33
2.4.2 Diagrama de Interacción del Diseño 34
2.5 Patrón Arquitectónico 36
2.6 Patrones de Diseño 38
2.7 Modelo de Despliegue 39
2.8 Conclusiones Parciales 40
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA 41
Introducción 41
3.1 Modelo de Implementación 41
3.1.1 Diagrama de componentes 41
3.2 Código fuente 43
3.2.1 Estándar de codificación 43
3.3 Prueba de software 44
3.3.1 Niveles de prueba 45
3.3.2 Desarrollo de pruebas de caja blanca 48
3.3.3 Desarrollo de pruebas de caja negra 49
3.3.4 Resultados generales de las pruebas 51
3.4 Conclusiones parciales 53
CONCLUSIONES GENERALES 54
RECOMENDACIONES 55

BIBLIOGRAFÍA 56
REFERENCIAS BIBLIOGRÁFICAS 60
ANEXOS 63
GLOSARIO DE TÉRMINOS 67

Índice de Figuras

Figura 1. Capas PHPReports....7
Figura 2. Descripción de ETL, tomado de (Proal, 2010).....10
Figura 3.Pentaho Data Integration (Kettle), tomado de (Comparativa B.I. Open Source, 2010). 11
Figura 4.Talend, Tomado de (Gracia Luis, 2010).....13
Figura 5.CloverETL, tomado de (Softpedia, 2013). 14
Figura 6. Modelo de Dominio 23
Figura 7. Diagrama de Casos de Uso del Sistema.....27
Figura 8. Diagrama de Clase de Diseño del CU Migrar Base de Datos.....33
Figura 9. Diagrama de Clase de Diseño del CUS Conectar a origen y destino de los datos. 34
Figura 10. Diagrama de Secuencia de Migrar Base de Datos.....35
Figura 11. Diagrama de Secuencia de Conectar a origen y destino de los datos.....36
Figura 12. Arquitectura por capas, tomado de (Arevalo, 2010). 37
Figura 13. Diagrama de despliegue. 40
Figura 14. Diagrama de Componentes. 42
Figura 15. Ejemplo de estándar de codificación.....44
Figura 16. Función “getXmlTemplate”.....48
Figura 17. Camino Básico de la Función “getXmlTemplate”.....48
Figura 18. Resumen de las no conformidades encontradas.....53
Figura 19. Interfaz principal de la herramienta. 63
Figura 20. Patrón Creador y Singleton. 64
Figura 21. Patrón Experto. 65
Figura 22. Patrón Controlador.....66

Introducción

Actualmente las Tecnologías de la Información y las Comunicaciones (TICs), constituyen un factor decisivo en el desarrollo económico y productivo del hombre. Estas brindan un conjunto de herramientas que permiten la automatización de tareas comunes, posibilitando la obtención de mejores resultados en el menor tiempo posible. Las mismas permiten a las empresas que cuentan con un gran cúmulo de información, una forma de recopilarlas para su posterior uso. Así surge el término de Base de Datos *(*BD*)* que mantiene la información almacenada y se puede acceder a ella de forma directa. Hoy día la mayoría de la entidades que cuentan con base de datos para almacenar su información utilizan reportes para visualizar esos datos de una forma más organizada.

Los reportes son informes que se encargan de organizar y mostrar la información contenida en una base de datos, tienen diversos niveles de complejidad a la hora de visualizarlos ya que pueden ser desde listas o enumeraciones, hasta gráficos muchos más desarrollados. Permiten realizar ciertas operaciones como, imprimirlos, enviarlos por correo electrónico y guardarlos en un archivo que puede tener varios formatos. Existen diferentes tecnologías que permiten generar reportes de forma dinámica y así lograr una mayor efectividad a la hora de hacer uso de ellos. Estas tecnologías son poderosos motores de reportes que pueden generar informes resumidos y detallados en tiempo real desde diferentes bases de datos.

El país no se encuentra ajeno al uso y desarrollo de estas tecnologías tanto es así que se encuentra inmerso en un proceso de transformaciones desde el punto de vista tecnológico, dirigidas a lograr la informatización de la sociedad. La mayoría de las empresas existentes en el país están sustentadas en medios informáticos que ayudan a evitar errores y deficiencias en el trabajo. Por tales motivos se hace necesario tener cada vez herramientas más seguras y que les brinden a los usuarios toda la información que estos necesiten, posibilitando optimizar el trabajo y el uso de los recursos pertenecientes a las empresas. Siendo la Universidad de la Ciencias Informáticas (UCI) el pilar fundamental y punto de partida para el desarrollo de este proceso de transformaciones.

El departamento de Integración de Soluciones perteneciente al Centro de Tecnologías de Gestión de Datos (DATEC) de la universidad, cuenta actualmente con el Generador Dinámico de Reportes (GDR) en su versión 1.8. Esta herramienta permite generar reportes de forma dinámica, partiendo de datos persistentes en algún origen de datos soportado por el sistema. La misma es una aplicación multiplataforma, con tecnologías web, que permite la creación y edición de reportes utilizando una amplia gama de fuentes de datos. Actualmente este sistema presenta una serie de módulos con diferentes

funcionalidades posibilitando un mejor trabajo con los reportes los cuales son: Diseñador de Modelos, Diseñador de Reportes, Visor de Reportes, Administrador de Reportes y Diseñador de Consultas.
Con el paso del tiempo han surgido nuevas tecnologías, ideas, funcionalidades y análisis estadísticos, la tecnología utilizada en esta versión para la generación de reportes se basa en PHPReports, el cual no brinda soporte para realizar análisis estadísticos y se va quedando obsoleto frente a otros motores de generación de reportes. PHPReports, solamente utiliza código PHP, es software libre y provee un alto nivel de productividad en este tipo de aplicaciones, pero se encuentra en desventaja con respecto a las características que presentan otras de su tipo, por ejemplo, esta tecnología solamente puede generar reportes desde bases de datos MySQL, PostgreSQL y SQLite aunque el equipo de desarrollo de GDR ha implementado su compatibilidad con otros gestores como Oracle y MSSQL. Las gráficas realizadas en GDR 1.8 que utiliza PHPReports como generador de reportes son mediante un Chart Server (Servidor de Gráficos), ya que con este motor no se puede graficar directamente.

Para reducir estos inconvenientes se decidió desarrollar la 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 utilizarlo en cualquier otra institución que requiera los servicios de un sistema como este. El producto contará con JasperReports como motor de reportes, que hoy día es una de las tecnologías más avanzadas en este campo y cuenta con un conjunto de librerías que facilitan la generación de reportes con gráficos y análisis estadísticos integrados. Debido a las distintas características que presenta el motor de reportes PHPReports, explicados anteriormente, con respecto a JasperReports, las bases de datos y los reportes generados por estas herramientas son incompatibles. Por lo que se hace imposible utilizar con el Generador Dinámico de Reportes versión 2.0, los reportes diseñados en la versión anterior y los demás datos almacenados en la base de datos del sistema. Este inconveniente trae como consecuencia que en las entidades en que se utiliza actualmente el GDR 1.8 y existe gran cantidad de reportes, modelos y consultas diseñadas, es necesario rediseñar y crear de forma manual cada uno de estos recursos nuevamente utilizando GDR 2.0, siendo este un factor que podría ser determinante a la hora de decidir actualizar o no a la nueva versión del sistema con el objetivo de poder utilizar todas las ventajas y nuevas funcionalidades disponibles en esta.

De acuerdo a la problemática expuesta, en el siguiente trabajo se plantea como **problema de la Investigación**: ¿Cómo utilizar los reportes diseñados y almacenados con GDR v1.8 en GDR v2.0?

El problema planteado lleva al siguiente **objeto de estudio**: Los procesos de Extracción, Transformación y Carga, el cual está delimitado por el **campo de acción**: Proceso de migración de la base de datos de GDR v1.8 a v2.0.

Para darle solución a este problema se ha propuesto como **objetivo general**: Desarrollar una herramienta para migrar la base de datos de la versión 1.8 del Generador Dinámico de Reportes a su versión 2.0.

Del **objetivo general** se desglosan los siguientes **objetivos específicos**:

- ✓ Análisis de las herramientas de transformación de bases de datos y selección de la adecuada para el desarrollo del software.
- ✓ Realizar el análisis y diseño de la herramienta para la migración de la base de datos desde la versión 1.8 del Generador Dinámico de Reportes a su versión 2.0.
- ✓ Realizar la implementación y prueba de la herramienta para la migración de la base de datos desde la versión 1.8 del Generador Dinámico de Reportes a su versión 2.0.

Con el fin de resolver el problema de la investigación y darle cumplimiento al objetivo planteado con anterioridad de forma sistemática y ascendente se exponen las siguientes **tareas de la investigación**:

- ✓ Análisis de los reportes generados por los motores PHPReports y JasperReports.
- ✓ Revisión bibliográfica y selección de las herramientas y metodología a utilizar.
- ✓ Definición de los requisitos funcionales y no funcionales de la solución.
- ✓ Identificación de la arquitectura.
- ✓ Diseño de la herramienta para la migración de reportes.
- ✓ Implementación de la solución.
- ✓ Ejecución de las pruebas.
- ✓ Documentación y corrección de las no conformidades identificadas en la ejecución de las pruebas.

Posibles resultados

- ✓ Herramienta que permita migrar la base de datos de la versión 1.8 del Generador Dinámico de Reportes (GDR) basado en PHPReports a su versión 2.0 que utilizará JasperReports como motor de reportes.

Estructura del Trabajo de Diploma

- **Capítulo 1: Fundamento Teórico**

Este capítulo contiene los fundamentos teóricos para entender el problema a solucionar y conceptos fundamentales. Un análisis de las herramientas y tecnologías posibles a utilizar, la definición de cuáles serán las seleccionadas para lograr tal propósito y un estudio sobre la metodología para desarrollar el sistema.

- **Capítulo 2: Análisis y Diseño del Sistema**

En este capítulo se realizará el modelo de dominio con su descripción correspondiente, una especificación bien detallada de los requisitos funcionales y no funcionales. Además se realizará el modelo de casos de uso del sistema que incluye todos los diagramas de casos de uso con su respectiva descripción y un modelo de diseño con los diagramas de clases de diseño y de secuencia de la aplicación a realizar.

- **Capítulo 3: Implementación y Prueba**

Comienza con el resultado del diseño, implementando el sistema en términos de componentes, así como una revisión final de las especificaciones del diseño y de la codificación mediante la realización de pruebas, garantizando la calidad del software.

Capítulo 1: Fundamentación Teórica

Introducción

En este capítulo se abordarán temas relacionados con las diferentes herramientas que existen para transformar bases de datos. Se profundizará en sus características y se realizará un análisis crítico de las mismas a través de una comparación, teniendo en cuenta los aspectos más relevantes y haciendo referencia al proceso de desarrollo de software. Se estudiarán los motores de reportes, la metodología de desarrollo, tecnologías y tendencias actuales que pueden ser útiles en la propuesta de solución, argumentando el por qué de su selección.

1.1 Conceptos y herramientas relacionados con el dominio del problema

A continuación se abordarán conceptos fundamentales sobre la investigación para una mejor comprensión de los temas que aquí se tratan, los cuales son imprescindibles en la realización del presente trabajo. Se explicarán algunas herramientas que se deben conocer y están estrechamente vinculados con el dominio del problema.

1.1.1 Reporte

Un reporte es un informe o una noticia. Este tipo de documento puede ser impreso, digital o audiovisual, pretende transmitir una información, aunque puede tener diversos objetivos. Existen reportes divulgativos y persuasivos, en el ámbito de la informática *"son informes que organizan y exhiben la información contenida en una base de datos"*. Su función es aplicar un formato determinado a los datos para mostrarlos por medio de un diseño atractivo y que sea fácil de interpretar por los usuarios. Los reportes tienen diversos niveles de complejidad, desde una lista o enumeración hasta gráficos mucho más desarrollados. Según la herramienta informática y la base de datos en cuestión, *"permiten la creación de etiquetas y la elaboración de facturas, entre otras tareas"*, (Definicion.De, 2008). Los reportes son visualizados por sistemas generadores que nos permiten obtener la información de manera estructurada y/o resumida.

1.1.2 Sistemas generadores de reportes

Los sistemas generadores de reportes son herramientas complementarias de los sistemas de información. Utilizan una especie de lenguaje transparente para el usuario, por medio del cual éste realiza consultas a

la base de datos y obtiene información de ella en forma de reporte. En la actualidad son muy usados en los productos de software empresariales, debido a la gran cantidad de información existente. Tienen la capacidad de interactuar con los resultados obtenidos basándose en los datos para tomar sus propias decisiones (Silva Hernández, 2003).

Los generadores de reportes son programas diseñados para crear y administrar informes en una amplia variedad de formatos. Estos proporcionan más control, pueden manejar datos de cálculos y lógica compleja antes de darles la salida. Se caracterizan por dos elementos básicos: un diseñador o editor de informes y un motor de generación. A continuación se explica el sistema generador de reportes más usado en la universidad.

1.1.3 Generador Dinámico de Reportes

El Generador Dinámico de Reportes es una aplicación web multiplataforma, para la gestión de la información de cualquier empresa o institución. Esta herramienta brinda la posibilidad de controlar el funcionamiento periódico de una o varias entidades, mediante la formulación de reportes en diferentes formatos y modelos personalizados. Además *"ofrece una solución de reporte que le brinda a los negocios una herramienta útil e inteligente, destinada a mejorar la rapidez y calidad de la adopción de decisiones en todos los niveles de una organización."* (Abreu Medina, y otros, 2012). Permite diseñar los reportes en forma de tabla y gráficos obteniendo la información para estos desde diferentes gestores de Base de Datos: SQL Server, SQLite, Oracle, MySQL y PostgreSQL. El GDR permite que un usuario con pocos conocimientos de base de datos pueda generar este tipo de reportes de forma dinámica. Esta fue desarrollada en la Universidad de las Ciencias Informáticas, específicamente en el departamento de Integración de Soluciones perteneciente al Centro DATEC. A continuación se caracterizan los motores de reportes utilizados por GDR en las versiones 1.8 y 2.0.

1.1.3.1 PHPReports

PHPReports es un conjunto de clases PHP, instrucciones XML y XSLT utilizadas para la trasformación de archivos XML en código PHP. El código generado será usado por dichas clases PHP.

Un reporte, en PHPReports está dividido por capas:

- ✓ Capa del documento
- ✓ Capa de páginas
- ✓ Capa de grupos

Todas estas capas recogen la información en los reportes como secuencias de números, estadísticas u otro tipo de información. Cada capa tiene su propio encabezado y pie de página. La capa de grupo tiene una división más donde se muestra la información de los datos (Rangel, 2006).

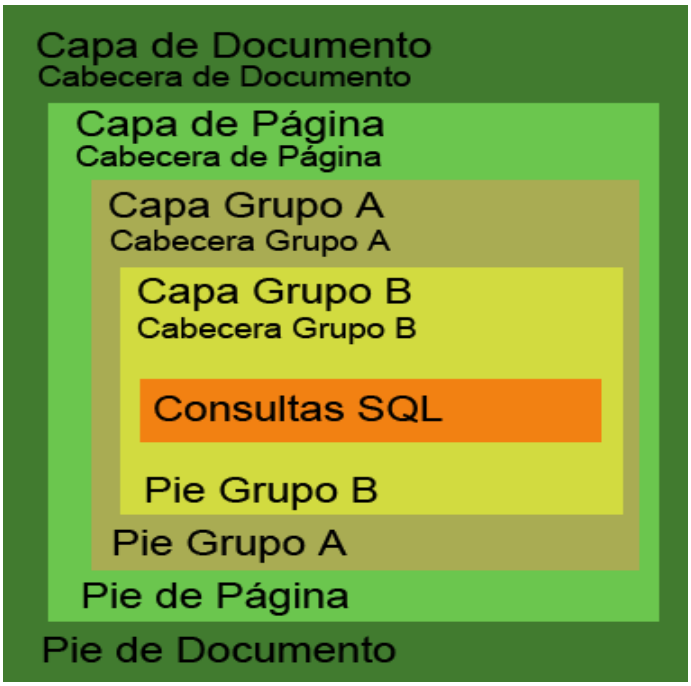


Figura 1. Capas PHPReports

1.1.3.2 JasperReports

JasperReports es una librería de código abierto escrita en java con el propósito de ayudar a los desarrolladores a generar reportes para las aplicaciones. Dado que no es una herramienta independiente no se puede instalar por sí sola, en su lugar se integran aplicaciones de java mediante la inclusión de su biblioteca, no está destinado para los usuarios finales, pero es bastante dirigido a los desarrolladores que necesiten añadir capacidades de mostrar información a sus aplicaciones. JasperReports es capaz de generar reportes profesionales incluyendo imágenes y gráficos, algunas de sus mejores características incluyen tener un posicionamiento flexible de los reportes, siendo capaz de presentar datos en forma de textos o gráficos, permite diferentes orígenes de datos así como generar marca de agua y exportar estos reportes en varios formatos de archivos como PDF, XML, HTML, CSV, XLS, RTF, TXT (Heffelfinger, 2006).

1.1.4 Base de Datos

Una Base de Datos (BD), *"se define como un conjunto de información estructurada en registros y almacenada en un soporte electrónico legible desde un ordenador. Cada registro constituye una unidad autónoma de información que puede estar a su vez estructurada en diferentes campos o tipos de datos que se recogen en dicha BD"* (Yunta, 2001). Las BD ayudan a recuperar la información rápida y flexiblemente, permiten la ejecución de diversas técnicas estadísticas sobre ellas, las cuales aportan grandes funcionalidades para el trabajo con los datos.

1.1.5 Transformación de Base de Datos

La transformación de base de datos se encarga de las inconsistencias en los formatos de datos y la codificación, que pueden existir dentro de una base de datos única. Estas transformaciones de BD "*consiste básicamente en convertir los datos de un formato de archivo a otro. Cada formato es creado por uno o varios programas siguiendo ciertas normas, y se codifica de modo que sólo podrá abrirse o leerse siguiendo esas normas"* (mastermagazine, 2010). Hoy día existen varios programas para convertir cualquier dato o formato, estos programas permiten obtener datos desde una fuente, transformarlos y volcarlos a otro soporte. A los programas que realizan estos procesos se les conoce como ETL, que a continuación se explicará detalladamente en que consiste y como es el proceso.

1.1.6 ETL

ETL son las siglas en inglés de Extraer, Transformar y Cargar (Extract, Transform and Load), estas herramientas permiten capturar y adaptar los datos de una base de datos a otra. Se puede definir como "*el proceso que permite a las organizaciones mover datos desde múltiples fuentes, reformatearlos, limpiarlos y cargarlos en otra base de datos para analizar, o en otro sistema operacional para apoyar un proceso de negocio"* (Espinosa, 2010). A continuación se exponen las funcionalidades del proceso de ETL.

Proceso de Extracción con Software ETL

- ✓ Cada sistema separado puede usar una organización diferente de los datos o formatos distintos.
- ✓ Los formatos de las fuentes normalmente se encuentran en bases de datos relacionales o ficheros planos.
- ✓ La extracción convierte los datos a un formato preparado para iniciar el proceso de transformación.
- ✓ Debe causar un impacto mínimo en la fuente de origen, si los datos a extraer son muchos, el sistema de origen se podría ralentizar e incluso colapsar.

Proceso de Transformación con Software ETL

La fase de transformación aplica una serie de reglas de negocios o funciones sobre los datos extraídos para convertirlos en datos que serán cargados.

- ✓ Seleccionar sólo ciertas columnas para su carga (por ejemplo, que las columnas con valores nulos no se carguen).
- ✓ Traducir códigos (por ejemplo, si la fuente almacena una “H” para Hombre y “M” para Mujer entonces el destino tiene que guardar “1″ para Hombre y “2″ para Mujer).
- ✓ Codificar valores libres (por ejemplo, convertir “Hombre” en “H” o “Sr” en “1″).
- ✓ Obtener nuevos valores calculados (por ejemplo, total_venta = cantidad * precio).
- ✓ Unir datos de múltiples fuentes (por ejemplo: búsquedas, combinaciones.).
- ✓ Calcular totales de múltiples filas de datos (por ejemplo, ventas totales de cada región).
- ✓ Generación de campos claves en el destino.
- ✓ Transponer o pivotar (girando múltiples columnas en filas o viceversa).
- ✓ Dividir una columna en varias (por ejemplo, columna “Nombre: García, Miguel”; pasar a dos columnas “Nombre: Miguel” y “Apellido: García”).

Proceso de Carga con Software ETL

La fase de carga es el momento en el cual los datos de la fase anterior (transformación) son cargados en el sistema de destino. Dependiendo de los requerimientos de la organización, este proceso puede abarcar una amplia variedad de acciones diferentes. A continuación se muestra gráficamente el proceso ETL anteriormente descrito:

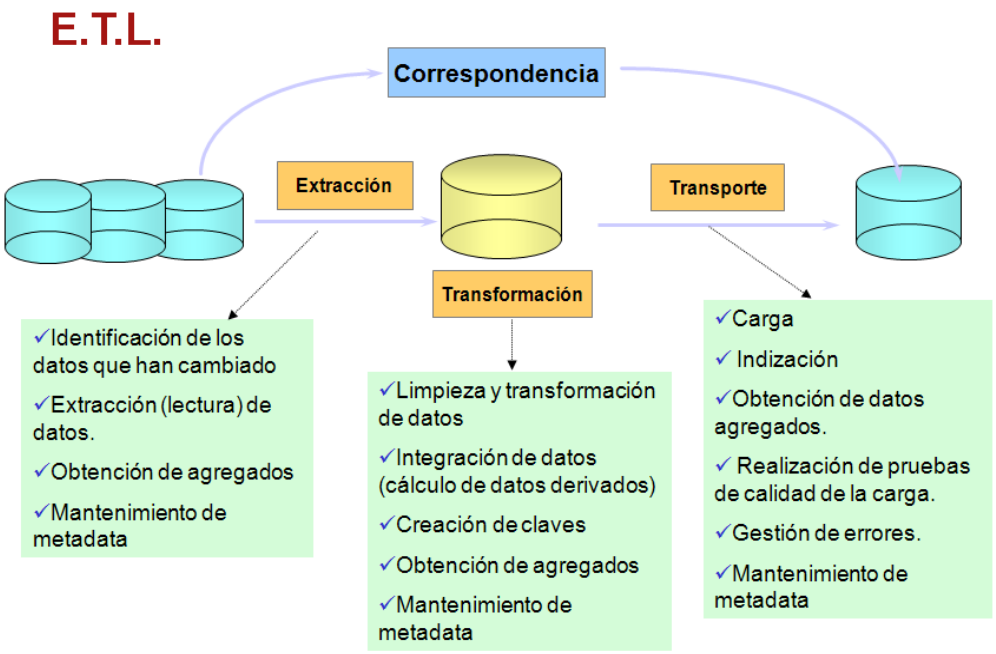


Figura 2. Descripción de ETL, tomado de (Proal, 2010).

1.2 Análisis de soluciones existentes de herramientas ETL

Se analizarán temas relacionados con soluciones para este tipo de herramientas existentes en el mundo. Las cuales pueden servir de base para el desarrollo de este trabajo ya sea por deficiencias de estas tecnologías, como características positivas, que ayudarían al mejor desempeño y resultados de la investigación.

1.2.1 Pentaho Data Integration-Kettle

Pentaho, creada en el 2004 es el actual líder en cuanto a soluciones de Inteligencia de Negocios (por sus siglas en ingles Business Intelligence B.I) de código abierto (Open Source). Ofrece, con soluciones propias, todo el espectro de recursos para desarrollar, mantener y explotar un proyecto de B.I. Desde las ETL con Data Integration hasta los cuadros de mando con el Diseñador de paneles(por sus siglas en inglés Dashboard Designer). La forma como Pentaho ha construido su solución de B.I. es integrando diferentes proyectos ya existentes y de solvencia reconocida. Data Integration anteriormente era Kettle, de hecho sigue conservando su antiguo nombre como nombre coloquial.

Pentaho Data Integration (Kettle) es una de las soluciones ETL más extendidas y mejor valoradas del mercado. Cuenta con una larga historia, una solidez y robustez que le hace una herramienta altamente

recomendable. Permite realizar transformaciones, que son una serie de pasos en el proceso ETL y trabajos, que se encargan de coleccionar estas transformaciones, de una forma muy sencilla e intuitiva, además de ser multiplataforma, totalmente gratis y de código abierto. El uso de esta herramienta puede evitar grandes cargas de trabajo manual frecuentemente difícil de mantener y de desplegar, utiliza varios estándares de tecnologías: Java, XML, JavaScript. La misma posee un gran número de seguidores que comparten consejos en diversos foros, por lo que esta comunidad puede ayudar con las dudas que puedan surgir (Comparativa B.I. Open Source, 2010).

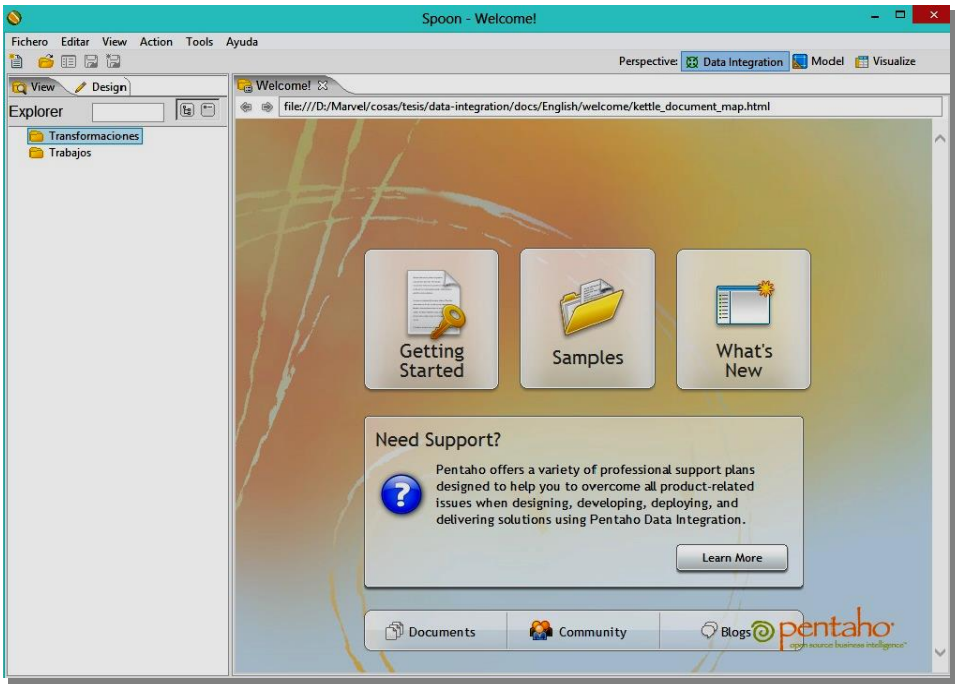


Figura 3.Pentaho Data Integration (Kettle), tomado de (Comparativa B.I. Open Source, 2010).

La imagen anterior muestra la interfaz de la herramienta gráfica (Spoon) que nos permite el diseño de las transformaciones y trabajos. Incluye opciones para previsualizar y testear los elementos desarrollados. Es la principal herramienta de trabajo y con la que se diseñan y validan los procesos ETL.

Ventajas:

- ✓ Gran facilidad de uso para instalar y configurar la herramienta.
- ✓ Los módulos incluidos por Kettle, pueden utilizarse de manera conjunta o de forma separada según las necesidades de la organización.

- ✓ Gran flexibilidad a la hora de realizar transformaciones ya que esta herramienta extrae la información, la limpia e integra permitiéndole al usuario la interacción con la información.

Desventajas:

- ✓ Es una solución desarrollada completamente en java por lo que:
    - El aspecto visual no es su mejor virtud
    - El rendimiento, a pesar de ser muy bueno, siempre será menor si lo comparamos contra scripts sql.

1.2.2 Talend

Talend es "la otra gran solución ETL", sigue siendo una empresa independiente que ofrece sus productos de forma autónoma. De hecho, aunque Talend Studio es su producto estrella, tiene otros productos interesantes como puede ser la Gestión de Datos Maestro (por sus siglas en ingles Master Data Management MDM). La forma de trabajar con Talend es también visual y bastante intuitiva, aunque a nivel interno el enfoque es completamente distinto. Talend en realidad es un generador de código es decir, el resultado de un proyecto ETL de Talend es código Java o Perl nativo (Per[1]). Talend está más orientado a un tipo de usuario programador con un nivel de conocimientos técnicos superior al requerido por Kettle. En contrapartida la flexibilidad que ofrece es absoluta (Comparativa B.I. Open Source, 2010).

Ventajas:

- ✓ Las ETL son código Java / Perl nativo por lo que en el momento de ejecución el rendimiento es muy bueno.
- ✓ Se pueden exportar los proyectos como servicios web.
- ✓ Pueden generar un ejecutable por lo que no dependen de las versiones del motor ETL.

Desventajas:

- ✓ Su principal desventaja es el entorno de desarrollo. Está basado en Eclipse y tiene una exigencia de maquina muy alta para el entorno de desarrollo.
- ✓ La curva de aprendizaje resulta más difícil que en otras herramientas de su tipo.
- ✓ Es más complicado o más oscuro de depurar el flujo que Kettle:

[1] Perl es un lenguaje de programación de propósito general originalmente desarrollado para la manipulación de texto y ahora se utiliza para una gran variedad de tareas, incluyendo la administración de sistemas, desarrollo web, programación en red, desarrollo de GUI y más.

- Para depurar el código se puede hacer con el debugger de eclipse, poniendo breakpoints (puntos de ruptura) y demás.

✓ Paralelismo: muy reducido en las versiones abiertas. Funcionalidad avanzada en las versiones de pago.

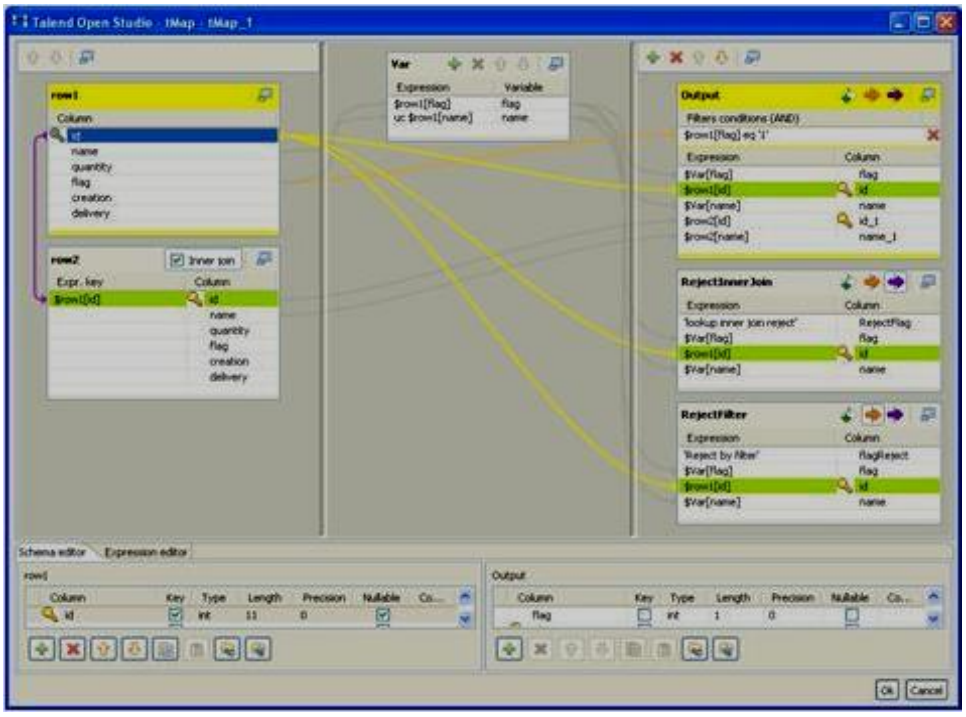


Figura 4.Talend, Tomado de (Gracia Luis, 2010).

La imagen anterior representa el Diseñador Visual que posee Talend, el cual proporciona una vista gráfica de los procesos, con componentes desplazables y presenta capacidades de depuración, que permite seguir en tiempo real los datos que fluyen a través de los procesos de transformación. También ofrece varias versiones del producto, desde una gratuita, a las profesionales. Sus principales deficiencias, están centradas en la necesidad de tener instalada la máquina virtual de java ya que está desarrollada sobre esta tecnología, también es necesario adquirir una versión no gratuita para poder disfrutar de las mejores prestaciones de esta aplicación.

1.2.3 CloverETL

CloverETL, es una aplicación programada en Java, que funciona como un marco de trabajo ETL y que puede utilizarse para transformar datos estructurados. Ejecuta las transformaciones de forma paralela,

soporta todos los principales estándares de la industria de bases de datos (Oracle, MSSQL, DB2, Infromix, Sybase) y otras variantes de código abierto (MySQL, PostgreSQL). Se puede utilizar independiente, como una aplicación de servidor o puede ser incorporado en otras aplicaciones como una biblioteca de la transformación. CloverETL es liberado bajo doble licencia, comercial y pública general reducida de GNU (LGPL, por sus siglas en inglés) (Softpedia, 2013).

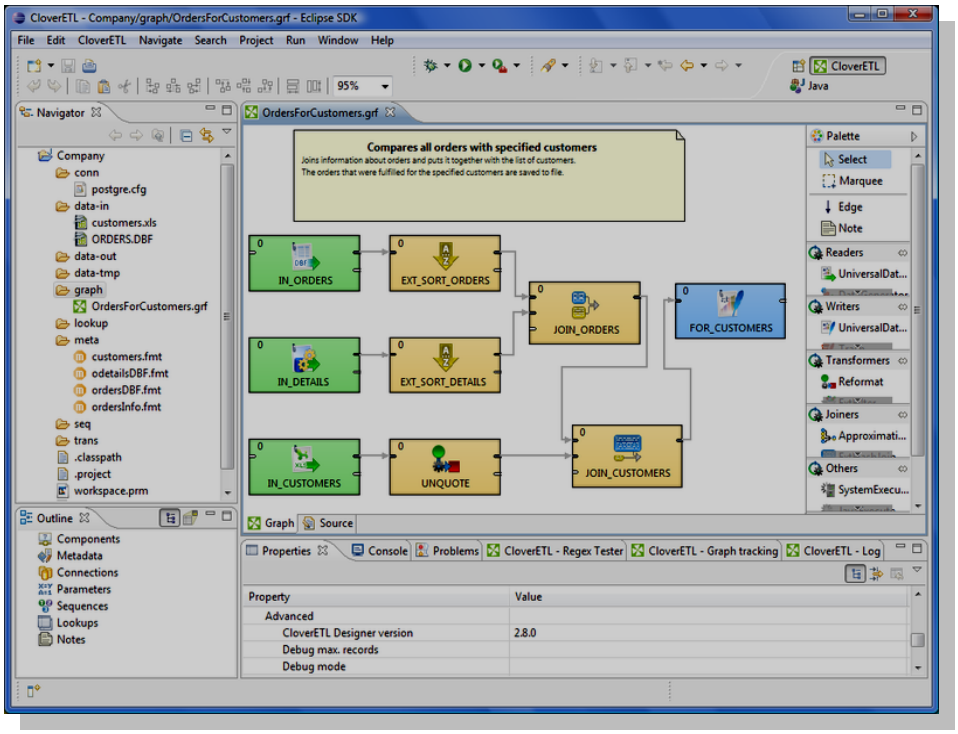


Figura 5.CloverETL, tomado de (Softpedia, 2013).

Ventajas:

- ✓ Internamente representa a todos los datos de caracteres Unicode como - cualquier carácter de cualquier códigos pueden ser representados - ASCII, LATINA, ASIA, entre otros.
- ✓ Convierte a partir de los datos comunes y conjuntos de caracteres (ASCII, UTF-8, ISO-8859-1, ISO-8859-2).
- ✓ Contiene la paleta de más de 40 componentes especializados de transformación.
- ✓ Lee y escribe datos XML / Excel y XLS.
- ✓ Apoya distancia de lectura / escritura de datos a través de protocolos FTP / SFTP / HTTP / HTTPS.

Desventajas:

- ✓ Es parcialmente gratis ya que su interfaz Clover GUI no lo es.

A continuación se realiza una comparación entre las herramientas Pentaho-Kettle, Talend y CloverETL:

Características	Talend	Pentaho - Kettle	CloverETL
Fabricante	Talend – Francia.	Pentaho – Estados Unidos.	Opensys.com.
Lenguaje de desarrollo	Java.	Java.	Java.
Plataforma	Windows, Unix y Linux.	Windows, Unix y Linux.	Windows, Unix y Linux.
Paralelismo	Muy reducido en las versiones Open. Funcionalidad avanzada en las versiones de pago.	Es muy sencillo realizar paralelismo de proceso.	Ejecuta tubería-paralelismo supuesto - cuando el expediente de datos es procesado por el componente, se envía inmediatamente al componente siguiente para el proceso adicional.
Costo	Las mejores versiones hay que pagarlas.	Totalmente gratis.	No es gratis.
Código fuente	El código fuente está disponible.	El código fuente está disponible.	El código fuente de motor está disponible.
Dependencia	Necesita de Java Database Connectivity (API que permite la realización de operaciones sobre bases de datos) para acceder a las fuentes.	Ninguna.	Ninguna.

Conectividad	Con Oracle, DB2, MySQL, Sybase y PostgreSQL. Conectividad ODBC para otras bases de datos.	Soporta Oracle, DB2, SQL Server y Sybase. Compatible también con MySQL, PostgreSQL, Hypersonic, FireBird SQL e Ingres. Soporta conectividad con SAP R/3 si se paga la licencia.	Oracle, MSSQL, DB2, Infromix, Sybase, MySQL, PostgreSQL.

Tabla 1: Comparación de herramientas ETL.

Después de un amplio estudio de las soluciones existentes a nivel mundial y las características de un centro como la Universidad de las Ciencias Informáticas que opta por el uso del software libre, se llega a la conclusión que la herramienta ETL más recomendable para el desarrollo de este trabajo es la conocida como Pentaho Data Integration. Una herramienta muy intuitiva y fácil de utilizar, siendo más rápida que las de su tipo en algunos aspectos. Tomando en cuenta estas ventajas se hará uso de sus funcionalidades para realizar una herramienta mucho más fácil y sencilla para migrar la base de datos de GDR 1.8 hacia GDR 2.0 en el menor tiempo posible y con las mayores garantías de éxito.

1.3 Metodología de Desarrollo de Software

Para definir en qué consiste la Metodología de Desarrollo de Software y en qué puede ayudar al desarrollo de este trabajo, se parte del concepto de metodología que brinda la Real Academia Española (RAE), la cual la define como *"Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal"* (RAE, 2012). Por lo tanto la metodología de desarrollo de software es un conjunto de pasos que guían el proceso de construcción de un sistema informático, para que el mismo logre tener la calidad requerida.

En los últimos tiempos la industria del software ha evolucionado de tal manera, que ha sido necesario desarrollar metodologías, para sostener la demanda de producción de sistemas cada vez mayores en complejidad y tamaño, logrando su construcción de forma óptima y eficiente. Para el desarrollo de este trabajo se decidió optar por la metodología del Proceso Unificado Abierto (OpenUP).

1.3.1 Proceso Unificado Abierto

La metodología de desarrollo de software OpenUP, generalmente es utilizada para proyectos pequeños de corto plazo, al igual que RUP mantiene las principales características de ser orientada por caso de uso, centrada en la arquitectura y es iterativo e incremental, además de ser una metodología ágil, genera pocos artefactos. *"OpenUP está organizado en dos dimensiones diferentes pero interrelacionadas: el método y el proceso. El contenido del método es donde los elementos del método (roles, tareas, artefactos y lineamientos) son definidos, sin tener en cuenta como son utilizados en el ciclo de vida del proyecto. El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Muchos ciclos de vida para diferentes proyectos pueden ser creados a partir del mismo conjunto de elementos del método"* (INGENIERIA DE SOFTWARE II, 2010). Estructura el ciclo de vida del proyecto en cuatro fases: Concepción, Elaboración, Construcción, y Transición, a continuación se explican cada una de las fases:

1. Inicio o Conceptualización: Obtener los objetivos del ciclo de vida del software.
2. Elaboración: Obtener la arquitectura candidata del ciclo de vida del software.
3. Construcción: Obtener la capacidad operativa inicial del ciclo de vida del software.
4. Transición: Obtener el reléase del producto.

Se decidió utilizar como metodología para el desarrollo de esta investigación OpenUP, pues el tiempo estimado para el desarrollo de la herramienta no excede de 6 meses y el equipo de trabajo es pequeño, lo que permite disminuir las probabilidades de fracaso e incrementar las probabilidades de éxito. Se detectan errores en fechas tempranas del desarrollo a través de un ciclo iterativo, evitando la elaboración de documentos, diagramas e iteraciones innecesarias.

1.4 Herramientas y Tecnologías

Las herramientas y tecnologías que apoyan al desarrollo de una aplicación, desempeñan un papel fundamental en su evolución. Constituyen el soporte en la administración del proyecto, adecuándose a las características y objetivos propios de la organización. Con el fin de realizar un software con la calidad requerida se realizó un análisis de las tecnologías y herramientas necesarias para el desarrollo de la aplicación.

1.4.1 Lenguaje Unificado de Modelado

El lenguaje unificado de modelado (UML), "*es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software."* (Dugarte, y otros, 2009). Dentro de las principales características del lenguaje se puede mencionar que permite especificar las funcionalidades de un sistema antes de su construcción, expresa de forma gráfica el sistema de modo que el equipo de desarrollo lo pueda entender, otra de estas características es que a partir de los modelos especificados se pueden construir los sistemas diseñados y por último, los elementos gráficos generados sirven como documentación del sistema, facilitando cualquier cambio que sea necesario realizar.

La utilización de UML versión 2.1 en este trabajo se hace necesaria para construir muchos de los artefactos que necesita OpenUP la metodología de desarrollo de software seleccionada, entre los cuales destacan, los diagramas: de Casos de Uso del Sistema(CUS), de clase del diseño, de secuencia, de componentes y despliegue. También debido a que como es un lenguaje visual es fácil de entender por cualquier persona del equipo de desarrollo y permite toda la documentación del ciclo de vida del sistema, con vista a facilitar el desarrollo de futuras versiones o mejoras.

1.4.2 Herramienta CASE

Las herramientas CASE (CASE, sigla en inglés de *Computer Aided Software Engineering*), *"son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero"* (Reyna, 2010). Estas aplicaciones ayudan al proceso de desarrollo de software permitiendo realizar el diseño del sistema y automatizar algunas actividades, entre las cuales destacan: implementación de parte del código fuente y detección de errores. De este tipo de herramientas una de las más destacadas y con la cual se desarrolla este trabajo es Visual Paradigm en su versión 8.0.

Visual Paradigm 8.0

Visual Paradigm es una herramienta CASE que soporta UML como lenguaje de modelado y la cual soporta el ciclo de vida completo de desarrollo de software indicado por las metodologías y en particular por OpenUP, a través de la representación de los diagramas necesarios. Esta herramienta acelera el desarrollo de aplicaciones, ya que sirve de puente visual entre arquitectos, analistas y diseñadores de sistemas de información. Está destinada a aumentar la calidad del software, mejorando la productividad y mantenimiento del software, permitir la reutilización, portabilidad y estandarización de la documentación.

1.4.3 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD), es una interfaz para que el usuario se pueda comunicar dinámicamente con la base datos. Estos sistemas manejan de forma clara la información de modo que su uso es bastante transparente, disminuyendo el tiempo de desarrollo y aumentando la calidad del sistema. Entre los principales SGBD que existen en la actualidad a nivel mundial se encuentra el privativo Oracle y los libre MySQL y PostgreSQL, tales gestores brindan similares prestaciones las cuales son necesarias para la transformación de BD.

PostgreSQL 9.1

Es un SGBD distribuido bajo la licencia Berkeley Software Distribution (BSD2) el cual dispone libremente su código fuente. Entre los SGBD ya sean de código abierto o totalmente propietario que se encuentran en la actualidad, es uno de los más potentes. Esta aplicación utiliza multiprocesos para garantizar la estabilidad del sistema, ya que de esta manera, un fallo, en uno de los procesos no afectará el resto y el sistema continuará su funcionamiento. También permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Posee una gran comunidad de seguidores los cuales pueden servir de referencia ante cualquier duda que pueda surgir durante la implementación del sistema.

PgAdmin III

Es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas, de una manera más fácil, debido a que se le ha incorporado una nueva interfaz gráfica, la cual resulta de gran ayuda en el momento de gestionar las bases de datos. Esta interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. Es desarrollado en todo el mundo por una comunidad de expertos de PostgreSQL y está disponible en más de una docena idiomas. Es de software libre, liberado bajo la licencia de PostgreSQL y multiplataforma. Permite el diseñado para las versiones de PostgreSQL múltiples y derivados. (pgAdmin PostgreSQL Tools, 2012).

[2] BSD es una licencia de software libre permisiva, con pocas restricciones que permite el uso del código fuente en software propietario.

1.4.4 Entorno de desarrollo

Un entorno de desarrollo integrado (IDE) es una aplicación de software que proporciona servicios integrales a los programadores para el desarrollo de software . Un IDE normalmente está compuesto por un editor de código fuente, un compilador o un intérprete y un depurador. En el desarrollo del software se utilizará como IDE NetBeans en su versión 7.2.

NetBeans 7.2

El Entorno de Desarrollo Integrado (IDE, sigla en inglés de *Integrated Development Environment*) NetBeans es *"una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso"* (Parra, 2011). Es apoyado por una gran comunidad de desarrolladores y ofrece una amplia documentación y recursos de capacitación. Incluye características notables, como el apoyo a marcos de PHP múltiples y se encuentra disponible en varios idiomas como: inglés, portugués brasileño, japonés, ruso y chino simplificado.

1.4.5 Lenguaje de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis, que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes (Arias Marin , 2008).

Java

"*Java es un lenguaje de programación orientado a objetos desarrollado por la Sun Microsystems. Java fue proyectado con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos, sea a nivel de código fuente que a nivel de código binario; lo que significa que los programas Java pueden ser ejecutados sobre cualquier sistema en la cual sea instalada la máquina virtual. Esta es una iniciativa de la compañía para guiar a los desarrolladores en la escritura, la certificación y la comercialización de las aplicaciones*" (Babylon Ltd, 2012).

Este lenguaje en muy usado en varias herramientas debido a las características importantes que tiene, una de ellas es que permite la ejecución de procesos concurrentemente o lo que se le conoce también

como hilo de ejecución. Es simple y fácil de aprender tanto así que los programadores experimentados de C++ pueden migrar fácilmente para java y ser productivos en poco tiempo. Permite la reutilización de código y se pueden crear programas modulares. Varias de las herramientas de este trabajo utilizan el lenguaje java, como por ejemplo en la versión 2.0 de GDR el motor reporteador JasperReports cuenta con un conjunto de librerías de este tipo que permite realizar varias funcionalidades que lo hace ser un potente generador de reportes.

XSTL

XSLT o XSL de Transformación es un estándar del Consorcio Internacional de la Red de Redes (W3C) que presenta una forma de transformar documentos XML en otros, incluso a formatos que no son XML. Las hojas de estilo XSLT realizan la transformación del documento utilizando una o varias reglas de plantilla unidas al documento fuente a transformar. Es un lenguaje declarativo, por lo que las hojas de estilo XSLT no se escriben como una secuencia de instrucciones, sino como una colección de plantillas en la que cada plantilla establece cómo se transforma un determinado elemento, definido mediante expresiones XPath que es el lenguaje para direccionar partes del XML (Clark, 1999).

1.4.6 Lenguaje de Marca

Un lenguaje de marcas es un lenguaje que incorpora etiquetas que contienen información adicional acerca de la estructura del texto de modo que el ordenador puede manipularlo. La mayoría de los lenguajes de marcas son legibles debido a que las anotaciones están escritas de forma tal que se puedan distinguir de los textos (Menéndez-Barzanallana Asensio, 2012).

XML

EL Lenguaje de Marcas Extensible (XML) es creado a partir del Lenguaje de marcado Generalizado (GML) diseñado especialmente para los documentos de la web. En el cual los diseñadores crean sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos. XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades. El Lenguaje Extensible de Marcado de Hipertexto (XHTML), el Lenguaje de Marcado Matemático (MathML) y los Gráficos Vectoriales Redimensionables (SVG) son algunos de los lenguajes que el XML puede definir. Los campos de aplicación del XML son las bases de datos, los documentos de texto, las hojas de cálculo y las páginas web. Entre los lenguajes creados con XML, se destacan el Lenguaje de Hoja de Estilo

Extensible (XSL) y el lenguaje de enlace XML (XLINK) que intenta trascender las limitaciones de los enlaces de hipertexto en HTML. La validez de los documentos depende la relación especificada entre los distintos elementos a partir de una Definición de Tipo de Documento (DTD).

1.5 Conclusiones parciales

En el presente capítulo se plantearon los principales conceptos relacionados con el dominio del problema que son de vital importancia para entender posteriormente la solución propuesta. Se realizó un estudio sobre todo el proceso ETL y las diferentes herramientas que existen de este tipo, haciendo un análisis crítico a través de una comparación, que ayudó a seleccionar la herramienta Kettle para el desarrollo del sistema. Se decidió utilizar como metodología de desarrollo OpenUP para guiar el proceso del software, como herramienta de modelado Visual Paradigm 8.0, que utiliza el lenguaje de modelado UML. El gestor de base de datos elegido PostgreSQL 9.1 y como administrador el PgAdmin III. Para el desarrollo de la aplicación se utiliza como lenguaje de programación java y XSLT, utilizando el entorno integrado de desarrollo NetBeans 7.2, y como lenguaje de marca XML.

Capítulo 2: Análisis y Diseño del Sistema

Introducción

En el presente capítulo se describe el proceso de desarrollado de la aplicación mediante el análisis y diseño del software. El mismo incluye el modelo de dominio del sistema a partir del cual se definen una serie de requisitos funcionales y no funcionales con que contará la solución. Se diseñarán los diagramas de clases y de interacción para los casos de uso identificados en la aplicación, así como también se definirá con precisión la estructura física que presentará el sistema en un modelo de despliegue.

2.1 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. Este modelo ayuda a comprender los conceptos que utilizan los usuarios, es decir, los conceptos con los que trabajan y con los que deberá trabajar la aplicación. Su objetivo es contribuir a la comprensión del contexto y de los aspectos básicos del dominio de la aplicación. Se describe mediante diagramas de UML, especialmente mediante diagramas de clases. A continuación se presenta el modelo de dominio propuesto (ver Figura 6) para un mejor entendimiento de la aplicación:

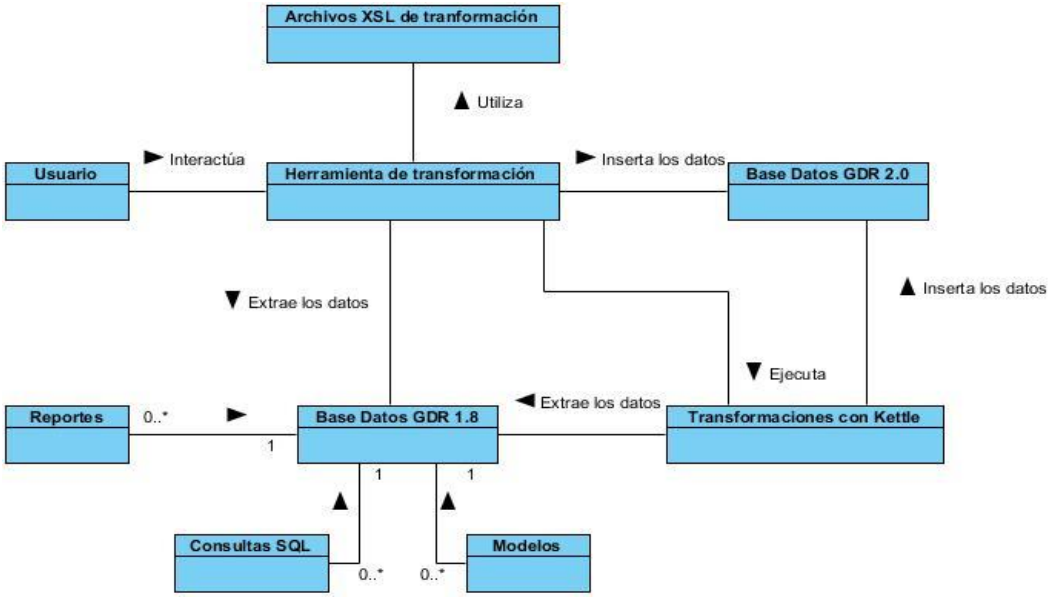


Figura 6. Modelo de Dominio

Conceptos del Modelo de Dominio:

Usuario: Persona capacitada para utilizar la herramienta de transformación de la base de datos de GDR.

Base de datos de GDR 1.8: Origen de datos donde se encuentran almacenados los datos y reportes del sistema GDR 1.8.

Herramienta de Migración: Aplicación encargada de extraer los reportes y transformarlos utilizando archivos XSL de transformación e insertarlos en la base de datos de GDR 2.0. Además de ejecutar las transformaciones utilizando librerías de la herramienta Kettle sobre ambas bases de datos.

Base de datos de GDR2.0: Destino de los datos, donde se almacenarán los datos previamente transformados de la versión 1.8 de GDR.

Archivos XSL de transformación: Archivos encargados de transformar los reportes con formato PHPReports que es el motor de reportes que utiliza la versión 1.8, a formato JasperReports el cual es el generador de reportes utilizado en la versión 2.0 de GDR.

Transformaciones con Kettle: Procedimientos encargados de cargar los datos almacenados en la base de datos de la versión 1.8 de GDR, e insertarlos en la base de datos de GDR 2.0 utilizando los trabajos y transformaciones previamente configurados las librerías de la herramienta Kettle.

Reporte: Es un archivo XML, generado por el motor de reportes PHPReports adoptando su formato.

Consultas SQL: Es un archivo XML, generado por el motor de reportes PHPReports adoptando su formato, que describe las consultas almacenadas.

Modelos: Es un archivo XML, generado por el motor de reportes PHPReports adoptando su formato, que contiene los datos de conexión a un origen de datos del sistema.

2.2 Requisitos del sistema

Los requisitos de un sistema representan los servicios que ha de ofrecer la aplicación y las limitaciones asociadas a su funcionamiento. Los requisitos se pueden clasificar en dos grupos: los funcionales y los no funcionales, donde los primeros indican qué debe hacer el sistema y los segundos cómo debe ser el sistema. Para la obtención de estos requisitos se pueden utilizar algunas técnicas como son las entrevistas con los clientes y los talleres, donde pueden participar varios factores del entorno donde se desarrolla el sistema, para aclarar y abordar las principales necesidades que se identifiquen.

2.2.1 Requisitos funcionales

Los requisitos funcionales (RF) son capacidades o condiciones que el sistema debe cumplir. Se mantienen invariables sin importar con qué propiedades o cualidades se relacionen. Los requisitos funcionales definen las funciones que el sistema será capaz de realizar. En la aplicación a desarrollar se identificaron los siguientes requisitos funcionales:

RF1 Conectar la herramienta de transformación con el origen de datos.

Entrada:

Servidor: Este campo describe la dirección del servidor donde se encuentra el origen de datos.

Base de Datos: Este campo describe el nombre de la base de datos del GDR 1.8 a la cual se conecta el sistema.

Puerto: Este campo describe el número del puerto por el cual se conectará al gestor.

Usuario: Este campo describe el nombre del usuario con permiso para acceder al sistema.

Contraseña: Este campo describe la contraseña del usuario.

Descripción: Establece la conexión de la herramienta con la base de datos fuente utilizando el valor de los campos anteriormente insertados.

Salida: De ser correctos los datos se establece la conexión a la base de datos especificada.

RF2 Conectar la herramienta de transformación con el destino de los datos.

Entrada:

Servidor: Este campo describe la dirección del servidor donde se encuentra el destino de los datos.

Base de Datos: Este campo describe el nombre de la base de datos de GDR 2.0 a la cual se conecta el sistema.

Puerto: Este campo describe el número del puerto por el cual se conectará al gestor.

Usuario: Este campo describe el nombre del usuario con permiso para acceder al sistema.

Contraseña: Este campo describe la contraseña del usuario.

Descripción: Establece la conexión de la herramienta con la base de datos destino utilizando el valor de los campos anteriormente insertados.

Salida: De ser correctos los datos se establece la conexión a la base de datos especificada.

RF3 Transformar reportes de la versión 1.8 de GDR a la versión 2.0 de GDR.

Entrada:

Los reportes XML con formato PHPReports almacenados en el origen de datos.

Descripción: Selecciona los reportes y configuraciones del origen de datos a los cuales se les van a realizar la transformaciones del formato PHPReports a JasperReports utilizando los XSL de Transformación, insertándolos posteriormente en el destino de los datos.

Salida: Los reportes XML en formato JasperReports almacenados en la base de datos destino y visualización del proceso de migración.

RF4 Importar los de datos de la versión 1.8 de GDR a la versión 2.0 de GDR.

Entrada:

Datos almacenados en la base de datos origen.

Descripción: Se conecta con el origen de datos, luego utilizando las librerías de la herramienta Kettle ejecuta las transformaciones de las demás tablas de la base de datos origen y posteriormente su inserción en la base de datos destino.

Salida: Datos transformados almacenados en la base de datos destino.

2.2.2 Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el producto debe tener, que se refieren a las características que hacen al producto usable, rápido o confiable. Estos requisitos son importantes para que los clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con todas las funcionalidades requeridas, entonces las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación.

Usabilidad:

RNF1: El sistema podrá ser usado por cualquier persona que tenga conocimientos básicos de informática.

RNF2: La aplicación es una solución de escritorio, accesible desde una computadora, a través de una conexión a las bases de datos de GDR.

RNF3: La aplicación tiene como objetivo transformar la base de datos de GDR 1.8 a la versión 2.0 de forma ágil y rápida, simplemente brindando los datos de conexión y seleccionando la opción MigrarBD.

Interfaz:

RNF4: La aplicación será diseñada con una interfaz amigable, fácil de usar por el usuario, de manera que agilice y facilite el trabajo con el software.

Software:

RNF5: Tener previamente creadas las bases de datos de GDR 1.8 y GDR 2.0.

Hardware:

RNF6: La PC cliente debe cumplir con los siguientes requisitos de hardware:

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

2.3 Diagrama de Caso de Uso del Sistema

Los diagramas de Casos de Uso del Sistema (CUS), son los encargados de recoger el comportamiento de un sistema desde el punto de vista de los usuarios. De esta manera luego de tener especificados los requisitos funcionales e identificado los actores del sistema, se puede dar paso a la realización del diagrama, el cual muestra la relación que existe entre el usuario final y las funcionalidades. La siguiente figura muestra el diagrama de CUS, el cual representa la relación que existe entre el actor en este caso el usuario y las diferentes funcionalidades que debe cumplir la aplicación.

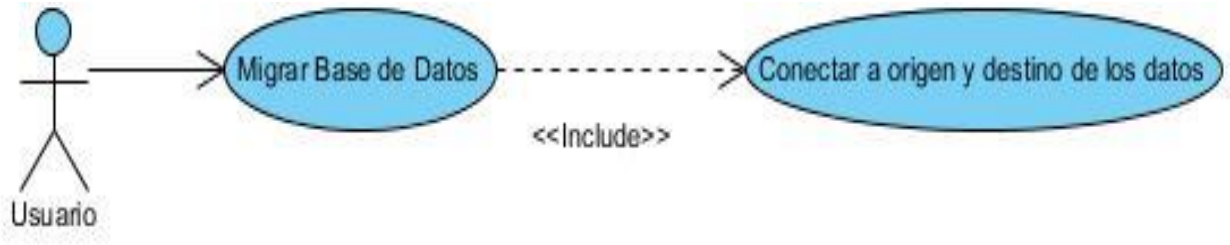


Figura 7. Diagrama de Casos de Uso del Sistema.

Descripción del Diagrama de Casos de Uso del Sistema

En el Diagrama de Casos de uso de Sistema (DCUS) el **actor** que en este caso es el usuario es el encargado de migrar la base de datos de GDR 1.8 a la versión 2.0 de GDR, mediante la conexión a ambas bases de datos dando inicio al caso de uso Migrar Base de Datos.

CUS Migrar Base de Datos: Mediante la conexión al origen y destino de los datos con los valores introducidos por el usuario, permite migrar la base de datos del GDR 1.8 a la versión 2.0.

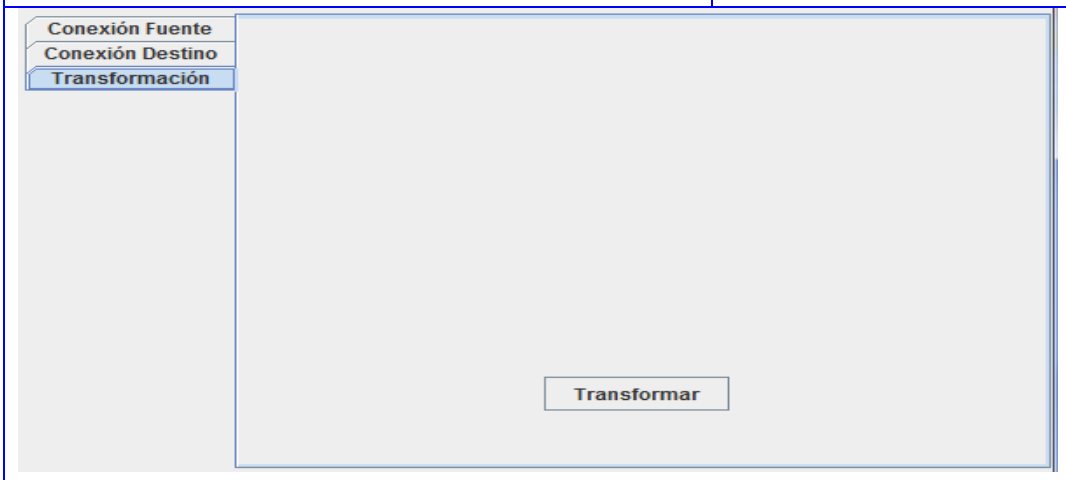
CUS Conectar a origen y destino de los datos: Establece la conexión con ambas bases de datos, utilizando los valores brindados por el usuario.

CU1 Migrar Base de Datos.

Objetivo	Transformar reportes y configuraciones almacenados en GDR 1.8 a su nueva versión 2.0.
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona la opción MigrarBD, donde se transforman los reportes y demás datos de la base de datos origen, y se actualizan en el destino de los datos. El caso de uso termina una vez realizado este proceso.
Precondiciones	Estar conectado previamente a la base de datos origen. Tener al menos un reporte almacenado. Tener creada la base de datos destino.
Referencias	RF3, RF4, CU Conectar a origen y destino de los datos <incluido>
Prioridad	Crítico

Flujo Normal de Eventos	
Sección1 "Migrar Base de Datos"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción MigrarBD.	2. El sistema selecciona los reportes y configuraciones almacenados en la base de datos origen. Transforma los reportes y configuraciones anteriormente seleccionados utilizando archivos XSL de Transformación. Configura las conexiones de las transformaciones utilizando librerías de Kettle previamente

	especificados y los ejecuta, seleccionando los datos de la base de datos origen e insertándolos en la base de datos destino. Inserta los reportes y configuraciones en la base de datos destino. Mensaje de migración realizada con éxito. Terminando así el caso de uso.



Prototipo de Interfaz

Flujos Alternos al paso 1	
No. Evento Cerrar interfaz	
Acción del Actor	**Respuesta del Sistema**
1a. El usuario selecciona la opción cerrar	1b. El sistema cierra la aplicación. Terminando así el caso de uso.
Flujos Alternos al paso 2	
No. Evento Fuente de datos vacía	

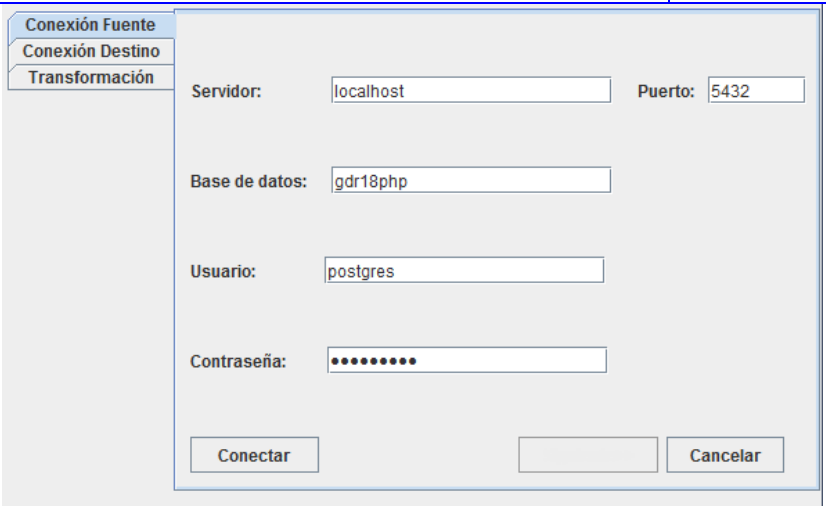
Acción del Actor	**Respuesta del Sistema**
	2a. El sistema muestra un mensaje de error indicando que la base de datos origen no contiene datos. Terminando así el caso de uso.
Poscondiciones	Se insertan los datos en la base de datos destino.

Tabla 2: Descripción del caso de uso Migrar Base de Datos.

CU2 Conectar a origen y destino de los datos.

Objetivo	Establecer conexión con el origen y destino de los datos.
Actores	Usuario
Resumen	Establece la conexión con el origen y destino de datos mediante los valores previamente insertados por el usuario.
Precondiciones	Servidor de PostgreSQL con las base de datos de GDR v1.8 y v2.0.
Referencias	RF1, RF2.
Prioridad	Crítico
Flujo Normal de Eventos	
Sección "Conectar a origen y destino de los datos"	
Acción del Actor	**Respuesta del Sistema**
1. El usuario introduce los datos para realizar la conexión con la base de datos origen.	2. El sistema valida los datos de conexión.
3. El usuario selecciona la opción Probar Conexión.	4. El sistema se conecta y muestra un mensaje de conexión realizada con éxito.

5. El usuario introduce los datos para realizar la conexión con la base de datos destino.	6. El sistema valida los datos de conexión.
7. El usuario selecciona la opción Probar Conexión.	8. El sistema se conecta y muestra un mensaje de conexión realizada con éxito. Terminando así el caso de uso.



Prototipo de Interfaz

Flujos Alternos al paso 1	
No. Evento Cerrar interfaz	
Acción del Actor	**Respuesta del Sistema**
1a. El usuario selecciona la opción cerrar	1b. El sistema cierra la aplicación. Terminando así el caso de uso.
Flujos Alternos al paso 4	
No. Evento Datos incorrectos	
Acción del Actor	**Respuesta del Sistema**

	4a. El sistema muestra un mensaje de error informando que los datos de conexión son incorrectos. Terminando así el caso de uso.
Flujos Alternos al paso 5	
No. Evento Cerrar interfaz	
Acción del Actor	**Respuesta del Sistema**
5a. El usuario selecciona la opción cerrar	5b. El sistema cierra la aplicación. Terminando así el caso de uso.
Flujos Alternos al paso 8	
No. Evento Datos incorrectos	
Acción del Actor	**Respuesta del Sistema**
	8a. El sistema muestra un mensaje de error informando que los datos de conexión son incorrectos. Terminando así el caso de uso.
Poscondiciones	Se establece la conexión entre ambas bases de datos.

Tabla 3: Descripción del caso de uso Conectar a origen y destino de los datos.

2.4 Modelo de diseño

En el modelo de diseño se guía el sistema de manera que soporte todos los requisitos, tanto funcionales como no funcionales. Mediante el modelo de diseño se elaboran diagramas que muestran gráficamente cómo los objetos se comunican entre ellos a fin de cumplir con los requerimientos. Entre estos diagramas se pueden encontrar los diagramas de clases del diseño, los cuáles resumen la definición de las clases que se deben implementar en el software.

2.4.1 Diagrama de Clase del Diseño

En el diagrama de clases de diseño se representan las principales clases y métodos del caso de uso Migrar Base de Datos, siendo las clases InterfazPrincipal, jLayeredPanel1 y jPanel1 componentes principales de la presentación. La figura 8 representa la funcionalidad de migrar BD de la clase principal HMDBPrincipal, la cual es la encargada de realizar todos los métodos implicados en la migración tales como ConfigTransformaciones (), EjecutarTrabajos (), entre otros, utilizando la clase ObjectController. Esta clase se encarga de inicializar los demás controladores pasándole como parámetro la conexión perteneciente a cada objeto para acceder a sus métodos de forma sencilla. Siendo HMBDPrincipal , ObjectController y las que se relacionan con esta, parte de la capa del negocio. La clase Conexión de la capa acceso a datos es la encargada de comunicarse con la base de datos fuente y destino.

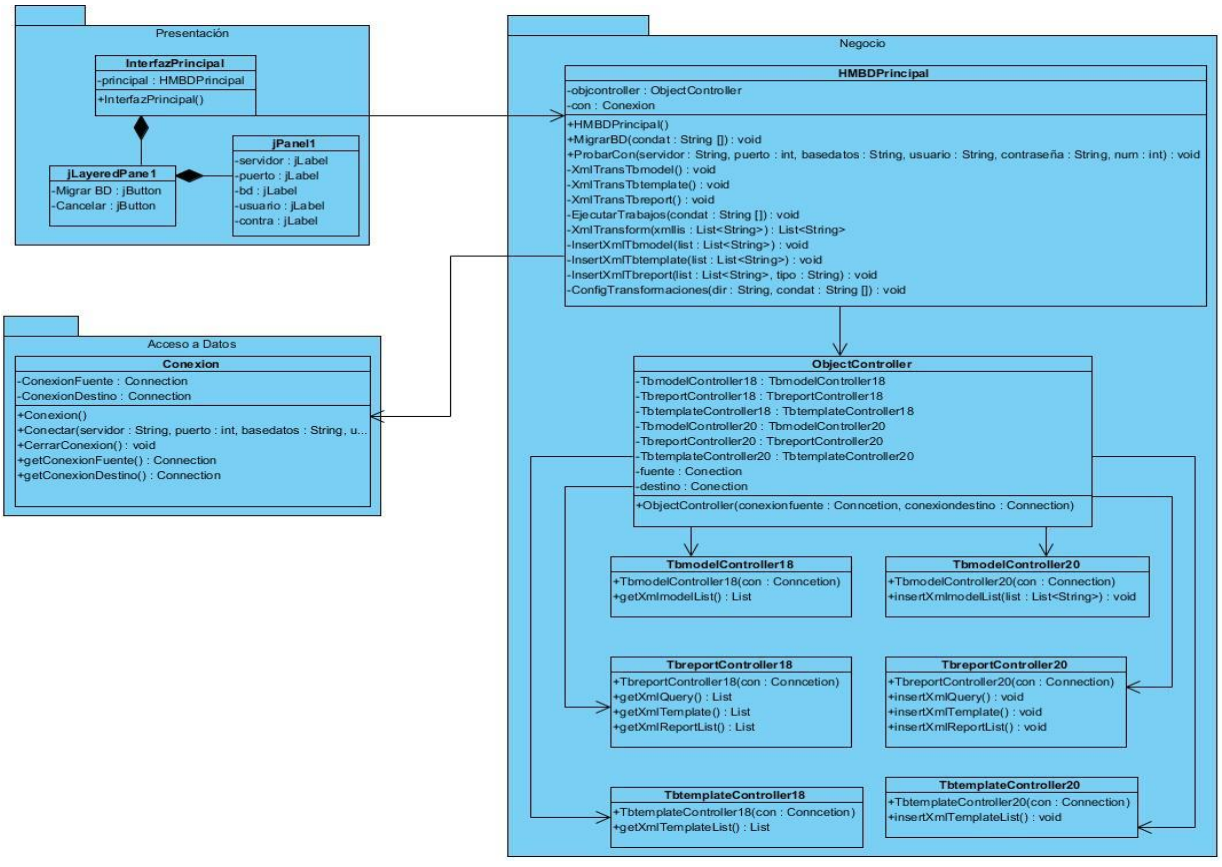


Figura 8. Diagrama de Clase de Diseño del CU Migrar Base de Datos.

En el diagrama de clases de diseño se representa las principales clases y métodos del caso de uso Conectar a origen y destino de los datos. La clase InterfazPrincipal está compuesta por varios componentes entre los que se encuentran jLayeredPanel1 y jPanel1 encargados de proporcionar la presentación para la entrada de datos de conexión. Estos datos son utilizados mediante la clase HMBDPrincipal que forma parte de la capa del negocio por el método Conectar () de la clase Conexión, ver figura 9.

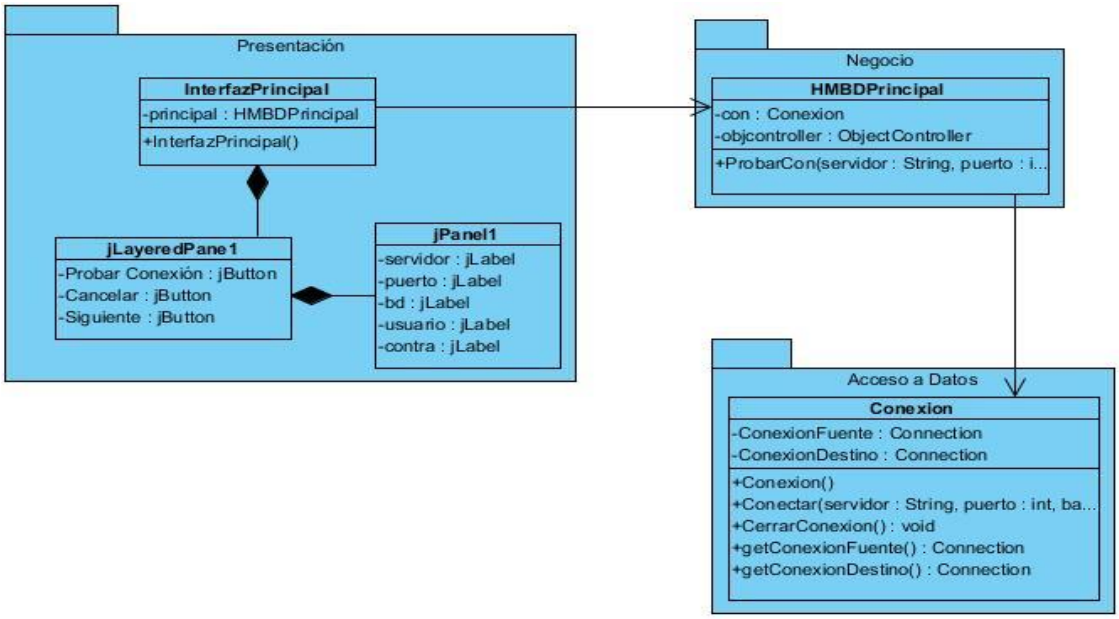


Figura 9. Diagrama de Clase de Diseño del CUS Conectar a origen y destino de los datos.

2.4.2 Diagrama de Interacción del Diseño

El diagrama de interacción representa la forma en que el usuario y los objetos de la aplicación se comunican, al ocurrir un determinado evento. Esta secuencia de pasos se realiza mediante llamadas, y se visualiza a través de una transferencia de mensajes, obteniéndose de esta manera las responsabilidades de cada objeto, por lo que este tipo de diagramas representan aspectos dinámicos de un sistema. Dentro de los diagramas de interacción se encuentran los diagramas de colaboración y de secuencia que son semánticamente equivalentes pero sin embargo los diagramas de colaboración destacan el orden estructural de los objetos que interactúan y los de secuencia destacan el orden temporal de los mensajes. Para la realización de los casos de uso del diseño es más factible el empleo de los diagramas de secuencia ya que representan con más claridad el flujo de las acciones que debe realizar el sistema.

El siguiente diagrama de secuencia representa las acciones que se ejecutan para realizar el caso de uso Migrar base de datos. El usuario selecciona la opción MigrarBD que se encuentra en la interfaz principal. Luego el sistema envía esta petición a la clase controladora HMBDPrincipal la cual va hacer la encargada de ejecutar el método dándole las responsabilidades a cada una de las clases controladoras. Una vez que ocurra toda esta secuencia de pasos se muestra un mensaje en la interfaz principal indicando que se realizó con éxito la operación. A continuación se representan el diagrama de Secuencia del Caso de Uso "Migrar base de datos":

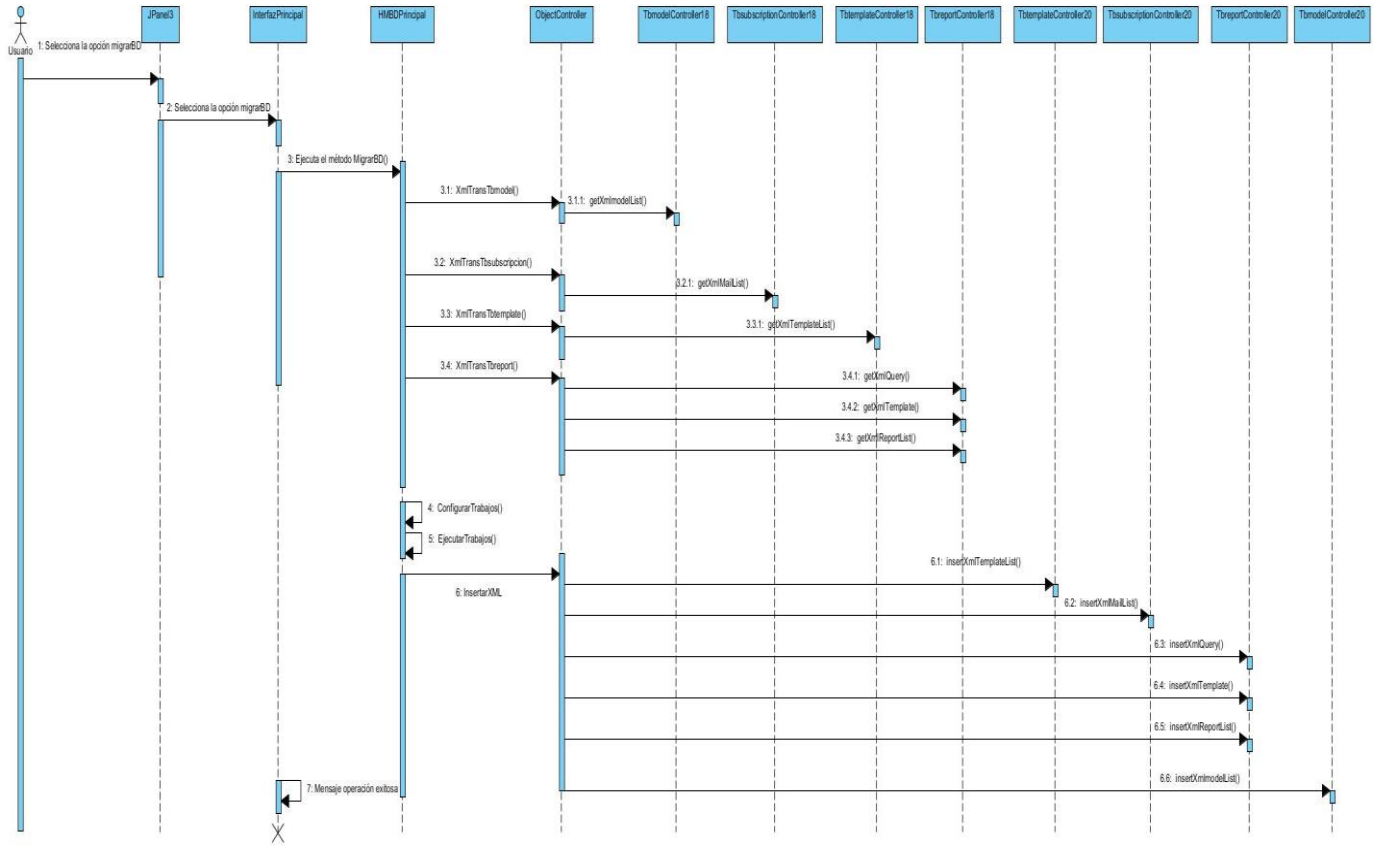


Figura 10. Diagrama de Secuencia de Migrar Base de Datos.

El siguiente diagrama de secuencia representa las acciones que se ejecutan para realizar el caso de uso Conectar a origen y destino de los datos. El usuario introduce los datos en la InterfazPrincipal para conectarse con el origen de los datos, si estos datos son correctos selecciona la opción Probar Conexión. Una vez realizada la conexión a la base de datos fuente, se efectúa la misma secuencia de pasos para la

conexión destino. Luego el sistema envía esta petición a la clase controladora HMDBPrincipal la cual va hacer la encargada de ejecutar el método Conectar con los datos previamente introducidos por el usuario y esta también le envía la petición a la clase conexión. Después que ocurra toda esta cadena de pasos se muestra un mensaje en la interfaz principal indicando que se realizó con éxito la operación. A continuación se representan el diagrama de Secuencia del Caso de Uso “Conectar a origen y destino de los datos”:

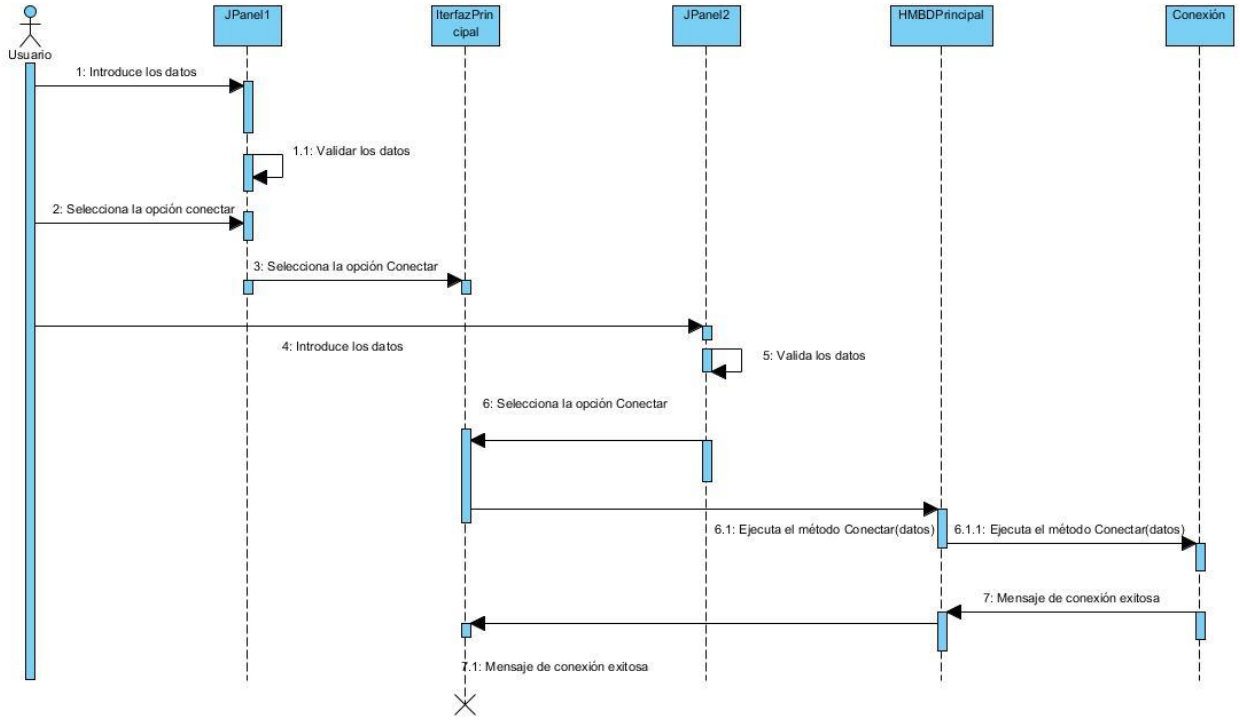


Figura 11. Diagrama de Secuencia de Conectar a origen y destino de los datos.

2.5 Patrón Arquitectónico

Los patrones arquitectónicos actualmente están muy difundidos, existen una gran variedad de ellos, todos aportan sus peculiaridades con el objetivo de resolver una problemática determinada. Sin embargo no siempre es beneficioso utilizar alguno, esto está condicionado a las características del software a implementar, pues un mal uso de un patrón, puede causar pérdidas lamentables en cuanto a la complejidad del diseño y el mantenimiento del mismo.

Es por esto que en el presente trabajo se aprovechan solo las ventajas que brinda el patrón arquitectónico por capas, en este caso tres capas. Este patrón es una de las técnicas más comunes que los arquitectos

de software utilizan para dividir sistemas de software complicados. Al pensar en un sistema en términos de capas, se imaginan los principales subsistemas de software ubicados de la misma forma que las capas de un pastel, donde cada capa descansa sobre la inferior. En este esquema la capa más alta utiliza varios servicios definidos por la inferior, pero la última es inconsciente de la superior.

Dentro de las ventajas que presenta esta arquitectura se encuentra que una capa se puede entender como un todo, sin considerar las otras. Estas pueden ser sustituidas con implementaciones alternativas de los mismos servicios básicos, se minimizan dependencias entre capas y posibilitan la estandarización de servicios. Luego de tener una capa construida, puede ser utilizada por muchos servicios de mayor nivel. La imagen que se muestra a continuación representa el esquema de la arquitectura que sigue este patrón (Arevalo, 2010):

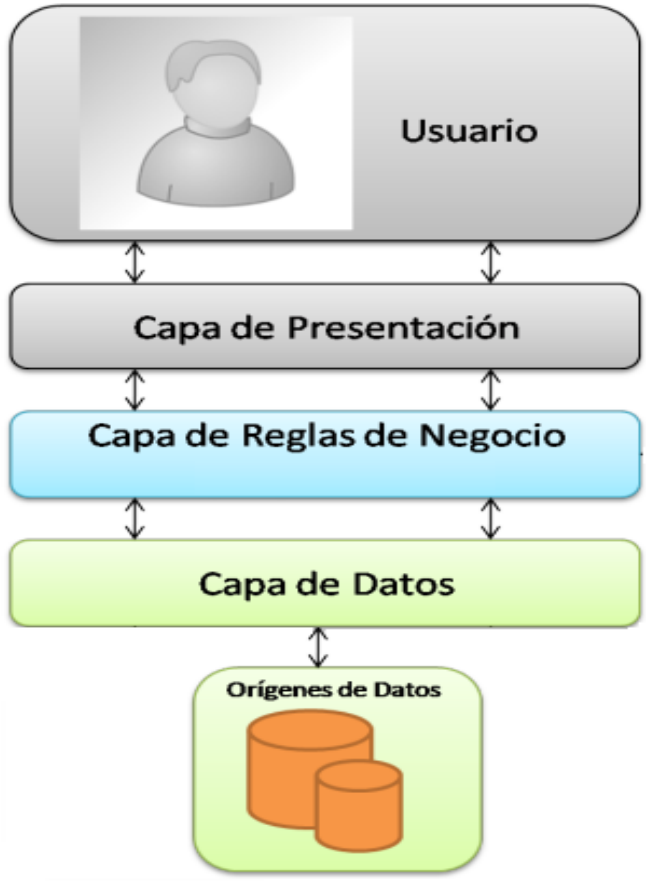


Figura 12. Arquitectura por capas, tomado de (Arevalo, 2010).

1. Capa de Presentación: Referente a la interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su principal responsabilidad es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados. En esta aplicación se encuentran los formularios y sus componentes, es decir la InterfazPrincipal con sus JPanel formando parte de dicha capa.

2. Capa de Reglas de Negocio: También denominada Lógica de Dominio, esta capa contiene la funcionalidad que implementa la aplicación. Entre sus principales características se encuentran cálculos basados en la información dada por el usuario, ya sean los datos almacenados, como las validaciones. Además se encarga de controlar la ejecución de la capa de acceso a datos y servicios externos. En la aplicación esta capa está compuesta por la clase controladora HMBDPrincipal la cual es la encargada de delegar funcionalidades al resto de las clases del nivel.

3. Capa de Datos: Esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación. Estos pueden ser monitores transaccionales, otras aplicaciones, sistemas de mensajerías, etc. Para el caso de esta aplicaciones, está representado por la clase Conexión que es la encargada de conectarse con las bases de datos, esta capa debe abstraer completamente a las capas superiores (negocio).

2.6 Patrones de Diseño

"*Los patrones de diseño solucionan problemas que existen en varios niveles de abstracción. Con el uso de estos se evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Permiten formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño*" (LARMAN, 1999). Las soluciones exitosas tienen sus raíces en patrones. Los patrones son una descripción de un problema y la esencia de su solución, de forma que esta pueda ser reutilizada en diferentes situaciones.

Patrones de diseño GRASP

Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades) (LARMAN, 1999).

Creador: La clase ObjectController tiene la responsabilidad de crear una instancia de las clases controladoras que se encargan de implementar los métodos de inserción y selección de reportes en la aplicación (Anexo 2**: Patrones de diseño GOF Singleton y GRASP Creador.**).

Experto: La Clase InterfazPrincipal tiene la responsabilidad de crear los objetos de las clases que se utilizan en la aplicación, esta crea las conexiones que se utilizan como parámetro en la Clase HMBDPrincipal y se utilizan para conectar con ambos orígenes de datos y realizar todo el trabajo de migración (Anexo 3**: Patrón GRASP Experto.**).

Controlador: Funciona como intermediario entre la interfaz principal y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control (Anexo 4**: Patrón GRASP Controlador.**).

Patrones de diseño GoF

Los patrones GoF (Gang of Four, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento (LARMAN, 1999).

Patrones creacionales:

Singleton: La clase ObjectController garantiza la existencia de una única instancia para las clases que se encargan de realizar la inserción y selección de los datos en la aplicación, además de la creación de un mecanismo de acceso global a dicha instancia (Anexo 2**: Patrones de diseño GOF Singleton y GRASP Creador.**).

2.7 Modelo de Despliegue

El modelo de despliegue es un modelo de objeto que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre nodos de la aplicación. La siguiente figura representa el modelo de despliegue de la herramienta para migrar la base de dato de GDR 1.8 a la versión 2.0 de GDR, donde se representa los nodos físicos de la aplicación así como su ubicación y relación.

Figura 13. Diagrama de despliegue.

Para el despliegue de la aplicación es necesario una PC cliente la cual ejecutará la aplicación de forma local usando la máquina virtual de java. La aplicación se conecta a un servidor donde se encuentran las base de datos del sistema GDR v1.8 y v2.0 a través del protocolo TCP/IP.

2.8 Conclusiones Parciales

En el presente capítulo se describió el análisis y diseño de la aplicación, comenzando con el modelo de dominio, en el cual se representaron los principales conceptos y objetos reales para un mejor entendimiento del problema. Se identificaron cuatro requisitos funcionales que deben cubrir el sistema y seis no funcionales que describen como debe ser el sistema, dando inicio a la elaboración del diagrama de caso de uso del sistema, el cual está compuesto por dos casos de uso. También se describieron diagramas de clases de diseño, con el objetivo de guiar la implementación del software, diagramas de secuencia para visualizar el aspecto dinámico del sistema y patrones de diseño que indican la esencia de la solución de un problema. Finalmente se realizó el diagrama de despliegue que muestra la distribución física del sistema.

Capítulo 3: implementación y prueba

Introducción

En esta disciplina de implementación y prueba se analizarán los artefactos principales de la misma, así como el modelo de implementación que incluye el diagrama de componentes donde se ejemplifica la distribución física del sistema basado en la arquitectura utilizada. Como todo sistema de software este debe ser probado antes de su entrega o puesta en funcionamiento, a este proceso o fase se le denomina prueba o testeo de la aplicación. Es una de las etapas más importantes en la vida del software y en ocasiones a la que menos recursos se le asignan, siendo una mala práctica que conlleva en muchos casos al fracaso del producto o a la no satisfacción del cliente.

3.1 Modelo de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Rodríguez, 2012).

3.1.1 Diagrama de componentes

Dentro del Modelo de Implementación se encuentran los diagramas de componentes. Un diagrama de componentes muestra como el sistema está dividido en componentes y las dependencias lógicas que existen entre ellos, sean éstos componentes una representación de una unidad de código(fuentes, binarios o ejecutables) que permite mostrar las dependencias en tiempo de compilación y ejecución. Estos diagramas proveen una vista arquitectónica de alto nivel del sistema, brindando una ayuda a los desarrolladores para visualizar el camino de la implementación, y permitiendo tomar decisiones respecto a las tareas a desarrollar (Figueroa , 2010).

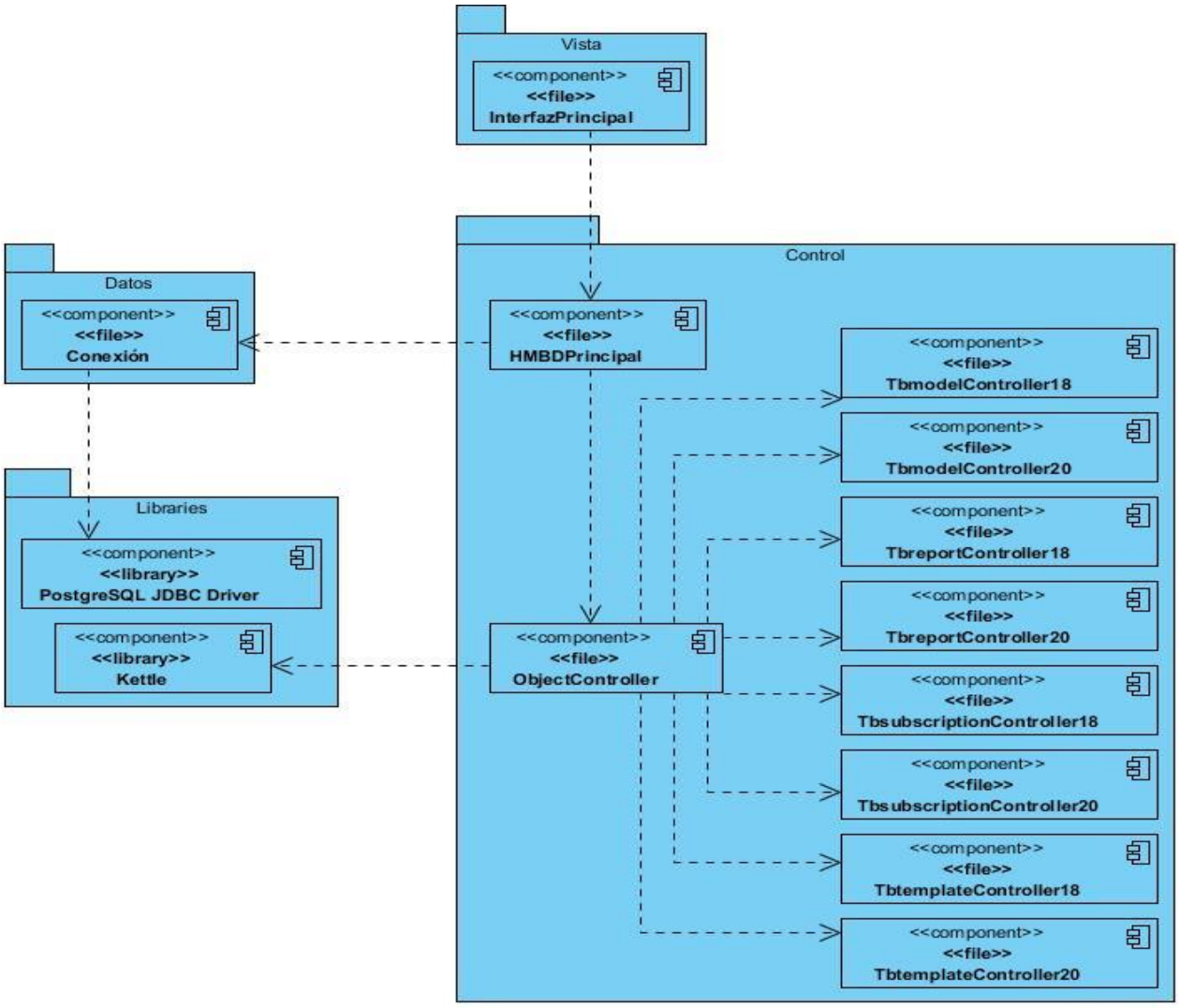


Figura 14. Diagrama de Componentes.

Este diagrama de componentes indica la distribución física de los elementos del sistema. En este caso el diagrama muestra los tres paquetes fundamentales: el paquete de la vista, donde están ubicados los componentes de la interfaz principal. El paquete llamado Control en el cual se encuentran las clases controladoras encargadas del funcionamiento de la aplicación y el paquete de datos que contiene la clase conexión, la que se encarga de conectarse a las bases de datos utilizando el driver de conexión PostgreSQL, que se encuentra dentro del paquete de librerías, junto con las librerías Kettle, que son utilizadas por la clase principal para realiza la migración.

3.2 Código fuente

El código fuente, también llamado código base, es un texto que se ha escrito en un lenguaje de programación concreto, y que sólo puede ser leído por un experto o programador. Estos caracteres deben ser traducidos a un lenguaje que se denomina código máquina, el cual podrá ejecutar cualquier ordenador. También puede ser traducido a un lenguaje llamado códigos de *bytes*, el cual podrá ser traducido por un intérprete. Este tipo de transferencias y traducciones se llaman compilación (ethek, 2010).

3.2.1 Estándar de codificación

La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. La mantenibilidad del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento. Aunque la legibilidad y la mantenibilidad son el resultado de muchos factores, una faceta del desarrollo de software en la que todos los programadores influyen especialmente es en la técnica de codificación. El mejor método para asegurarse de que un equipo de programadores mantenga un código de calidad es establecer un estándar de codificación sobre el que se efectuarán luego revisiones. El estándar de codificación seleccionado para el desarrollo de la aplicación es el conocido como CamelCase.

CamelCase: Es un estilo de escritura que se aplica a frases o palabra compuesta, dentro de este estilo existen dos tipos **lowerCamelCase** que define la primera letra de cada palabra en mayúscula excepto la primera que la pone en minúscula completa y el **UpperCamelCase** define el comienzo de cada palabra en mayúscula, este último es el seleccionado para el desarrollo de la aplicación**.**

Para cumplir con estos paradigmas se establece el estilo siguiente:

Comentarios

- ✓ Para los comentarios de más de una líneas se escriben comenzando con “/*” y terminando con “*/”, y “//” para los de una sola línea.
- ✓ Deben escribirse comentarios al principio de cada clase y método brindando una breve descripción de los propósitos generales de cada funcionalidad.

Declaraciones

- ✓ Las variables y atributos deben ser explícitas para saber realmente lo que significan y utilizando letra minúscula.

- ✓ Para declarar una clase se comienza con letra mayúscula y en caso de necesitar escribir otra palabra para hacer la misma más entendible se comienza también con mayúscula para lograr una mejor legibilidad de la palabra. Ejemplo: InterfazPrincipal.
- ✓ Las constantes se declaran en mayúsculas todas las letras que la componga. Ejemplo: PI = 3.14.

Operadores lógicos y aritméticos

- ✓ Colocar espacios en blanco entre operadores lógicos-aritméticos y sus operandos. Ejemplo: a + b, c = a * 3

Llaves

- ✓ Colocar al lado de la declaración y al final del código, ya sean clases, métodos o instrucciones.

Ejemplo:

if (a != 1) {

…

}

```
/**
 *
 * @author mar
 */
public class HMBDPrincipal {

    Conexion con;
    ObjectController objcontroller;

    public HMBDPrincipal() {
        con = new Conexion();

    }

    public void MigrarBD(String[] condat) throws Exception {
        objcontroller = new ObjectController(con.getConexionFuente(), con.getConexionDestino());

        EjecutarTrabajos(condat);
        XmlTransTbmodel();
        XmlTransTbtemplate();
        XmlTransTbreport();


    }
```

Figura 15. Ejemplo de estándar de codificación

3.3 Prueba de software

"El único instrumento adecuado para determinar el status de la calidad de un producto de software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los

requerimientos" (PRUEBASDESOFTWARE, 2005). Las pruebas son de gran importancia en la garantía de la calidad del software. Estas son un proceso de ejecución de un programa con la intención de descubrir errores. Una buena prueba es aquella que tiene una alta probabilidad de cumplir con sus principales objetivos.

Los objetivos principales de realizar una prueba son:

- ✓ Detectar un error
- ✓ Tener un buen caso de prueba
- ✓ Descubrir un error no descubierto antes.

3.3.1 Niveles de prueba

Los niveles de prueba son diferentes ángulos de verificar y validar un producto de software. Las pruebas se aplican durante todo el ciclo de desarrollo del software para diferentes objetivos y en distintos niveles de trabajo, dentro de estas se distinguen diferentes niveles de prueba: Prueba a nivel de desarrollador, unidad, integración, sistema y aceptación (Carrillo, 2011). En el desarrollo de la herramienta de migración de base de datos de GDR v1.8 a la versión 2.0 se aplicarán las pruebas a nivel de desarrollador, la cual está diseñada e implementada por el equipo de desarrollo.

Tipos de prueba

Existen diferentes tipos de pruebas que se pueden aplicar para verificar que la aplicación cumple con todos los requisitos identificados durante el proceso de análisis. Dentro de los tipos de pruebas seleccionados se encuentran las pruebas unitarias y funcionales.

Prueba Unitaria: Se focaliza en aislar cada parte del programa y mostrar que las partes individuales son correctas. Busca asegurar que el código funciona de acuerdo con las especificaciones y que el módulo lógico es válido.

Prueba de Funcionales: Mediante las pruebas funcionales validamos el cumplimiento de la aplicación desarrollada contra las funcionalidades detalladas, fijando su atención en la validación de las funciones, métodos, caso de uso.

Se decide realizar las pruebas de tipo unitarias y funcionales ya que con el desarrollo de estas pruebas se validan todos los requisitos definidos en la fase de análisis y diseño satisfactoriamente. Las pruebas funcionales tienen como objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, estas responden a los requisitos funcionales en la que a

partir de casos de pruebas a través del método de caja negra se pretende mostrar los errores que presenta el sistema. Las pruebas unitarias tienen como objetivo asegurar la calidad del código entregado, además estas pruebas facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten realizar pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Métodos de prueba

"Los métodos de prueba del software tienen el objetivo de diseñar pruebas que descubran diferentes tipos de errores con menor tiempo y esfuerzo" (Codetel, 1998). Existen métodos de pruebas independientemente del nivel en que se enmarquen los tipos de pruebas, en los que se encuentran el método de caja blanca y el de caja negra los cuales serán usados en el presente trabajo, a continuación se explicarán cada uno de estos métodos.

Pruebas de caja blanca: La prueba de caja blanca, denominada a veces prueba de caja de cristal, da comienzo con la observación de que un programa difícilmente puede considerarse como probado por completo si su código contiene partes que nunca han sido ejecutadas. Este método analiza la estructura lógica del programa y, para cada alternativa que puede presentarse, los datos de prueba ideados conducirán a ella. Se procura escoger los que verifiquen cada posibilidad en las proposiciones **case**, las cláusulas de cada proposición **if** y la condición de terminación de cada ciclo. En un programa extenso, este método es inusual, pero en un módulo pequeño constituye un excelente medio de prueba y depuración (Codetel, 1998).

Uno de los métodos de prueba de caja blanca es el camino básico, la cual determina la complejidad ciclomática de una porción de código. La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa el camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar. Esta complejidad se puede calcular de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
2. La complejidad ciclomática, V(G), de un grafo de flujo G, se define como: V(G)=A-N+2. Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3. La complejidad ciclomática, V(G), de un grafo de flujo G también se define como V(G) = P+1. Donde P es el número de nodos predicado[3] contenidos en el grafo de flujo G.

Pruebas de caja negra: La prueba de la caja negra, también denominada prueba de comportamiento se centra en los requisitos fundamentales del software y permite obtener entradas que prueben todos los requisitos funcionales del programa (Codetel, 1998). Con estas pruebas se intenta identificar errores de las siguientes categorías:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las bases de datos externas
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Con la aplicación de este método se obtiene un conjunto de pruebas que, reduce el número de casos de pruebas y nos dicen algo sobre la presencia o ausencia de errores. Existen varios métodos de caja negra entre los que se pueden mencionar los métodos de prueba basados en grafo, partición equivalente y análisis de valores al límite.

Una partición equivalente es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos. El diseño de casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia. Las cuales se pueden definir de acuerdo con las siguientes directrices:

1. Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y una no válida.
4. Si una condición de entrada es lógica, se define una clase de equivalencia válida y una no válida.

[3] Predicado: Nodos a partir de los que se puede tomar más de un camino.

3.3.2 Desarrollo de pruebas de caja blanca

Como se mencionó en acápites anteriores uno de los métodos de prueba de caja blanca es la del camino simple, que se aplica a fragmentos de código de la aplicación. Se determinó aplicar este método a la función que devuelve la Plantilla de los Modelos.

Para la función "getXmlTemplate" se identificaron los bloques de ejecución y se enumeraron para identificarlos como se muestra en la siguiente Figura 16. Se obtuvieron 4 bloques y se determinó el camino básico ilustrado en la Figura 17. Identificando en cada sentencia condicional un nodo predicado del cual se derivan más de un camino a seguir, tal es el caso del nodo 2.

```
public List getXmlTemplate() throws SQLException {
    LinkedList<String> reslist = new LinkedList<String>();

    stat = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
            ResultSet.CONCUR_READ_ONLY);

    result = stat.executeQuery("select xmltemplate "
            + "from mod_recuperaciones.tbtemplate");
    while (result.next()) {
        reslist.add(result.getString(1));
    }
    return reslist;
}
```

Figura 16. Función "getXmlTemplate".

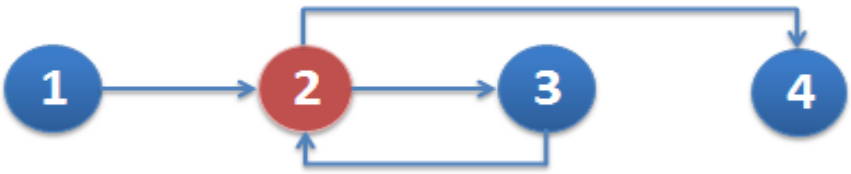


Figura 17. Camino Básico de la Función "getXmlTemplate".

Con el camino básico determinado, se aplica una de las tres formas para calcular la complejidad ciclomática, se utilizó la fórmula V(G) = A - N + 2, para la cual se obtuvo 4 artistas y 4 nodos, por lo tanto: V(G) = 4 - 4 + 2, quedando V(G) = 2. De la misma forma se pueden comprobar que las otras variantes

explicadas de calcular la complejidad ciclomática arriban al mismo resultado. La complejidad ciclomática indica los posibles casos de ejecución para la función.

3.3.3 Desarrollo de pruebas de caja negra

Diseño de caso de prueba

Antes de ejecutar un caso de prueba es necesario definir una serie de variables que serán utilizadas. Para el CU: “Conectar a base de datos” se definieron cinco variables que se describen a continuación.

N0	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Servidor	campo de texto	No	Permite letras, números y los siguientes caracteres:(“.“,”/”).
2	Puerto	campo de texto	No	Número mayor que 0 y menor que 10000.
3	Base de datos	campo de texto	No	Secuencia de caracteres alfanuméricos
4	Usuario	campo de texto	No	Secuencia de caracteres alfanuméricos.
5	Contraseña	campo de texto	No	Secuencia de caracteres alfanuméricos.

Tabla 4. Descripción de las variables para el CU Conectar a base de datos.

CP 1 Conectar a base de datos

Escenario	Variables (Enumeradas según descripción de la variable)					Respuesta del sistema	Resultado de la Prueba	Flujo Central
	1	2	3	4	5			
Conectar a bases de datos	V	V	V	V	V	El sistema se conecta correctamente con la base de datos.	Satisfactorio.	1. Se llenan los campos con datos correctos para la conexión

	Localhost 123.142.10.2	5432	GDR18	Postgres	Postgres1	Muestra un mensaje avisando “Conexión realizada con éxito”.		de la BD Fuente. 2. Selecciona la opción “Probar Conexión”. 3. Selecciona la opción “Siguiente”. 4. Se llenan los campos con datos correctos para la conexión de la BD Destino. 5. Por último selecciona la opción “Probar Conexión”.
El sistema muestra un mensaje que los campos son incorrectos.	**I**	**I**	**I**	**I**	**I**	El sistema muestra el siguiente mensaje “Fallo de conexión”	Insatisfactorio	1. Se llenan los campos con datos incorrectos para la conexión de la BD Fuente. 2. Selecciona la opción “Probar Conexión”, el sistema muestra un mensaje indicando el error.

El sistema muestra un mensaje que faltan campos por llenar.	I	I	I	I	I	El sistema muestra el siguiente mensaje “Fallo de conexión”	Insatisfactorio	1. Se llenan los campos dejando algunos con datos en blanco para la conexión de la BD Fuente. 2. Selecciona la opción “Probar Conexión”, el sistema muestra un mensaje indicando el error.

Tabla 6. Validación de las variables de entrada.

3.3.4 Resultados generales de las pruebas

Caja Blanca

Como resultado de la prueba de caja blanca tenemos que la complejidad ciclomática del sistema es 2 por lo que como mínimo se deberán diseñar dos casos de prueba. Donde se logró que cada sentencia del método escogido se ejecutara al menos una vez. Los caminos de pruebas elegidos son los siguientes:

Camino 1: 1, 2, 4

Camino 2: 1, 2, 3, 2, 4.

Para estos caminos independientes se conformaron los siguientes casos de prueba:

Caso de prueba para el Camino 1

Descripción:

Se quiere obtener un listado que contenga los XML de las consultas, para esto se invoca al método createStatement() y las consultas creadas se guardan en una variable llamada stat. Luego se ejecutan estas consultas guardándolas en otra variable llamada result, a través de un ciclo se recorre la variable result para obtener el resultado de las consultas y guardarlos en una lista que será retornada, en este caso la lista retornada estará vacía porque no llega nunca a adicionar el valor obtenido. Siguiendo estas especificaciones se garantiza que se ejecute dicho camino independiente en su totalidad.

Resultado esperado:

Al tener la consulta creada y luego ejecutarla, se guarda en una variable que se recorre a través de un ciclo, pero este resultado nunca se almacena en la lista que debe devolver el XML de la consulta.

Caso de prueba para el Camino 2

Descripción:

Se quiere obtener un listado que contenga los XML de las consultas, para esto se invoca al método createStatement() y las consultas creadas se guardan en una variable llamada stat. Luego se ejecutan estas consultas guardándolas en otra variable llamada result, a través de un ciclo se recorre la variable result para obtener el resultado de las consultas y adicionar los XML de estas consultas en una lista que será retornada. Siguiendo estas especificaciones se garantiza que se ejecute dicho camino independiente en su totalidad.

Resultado esperado:

Al tener la consulta creada y luego ejecutarla, se guarda en una variable que se recorre a través de un ciclo, luego se adiciona el resultado en una lista que se retorna con los XML de las consultas.

Caja Negra

Al aplicar las pruebas funcionales a la aplicación, las cuales se dividieron en tres iteraciones, se encontraron cinco no conformidades significativas, la mayoría precisamente por problemas de validación, en la 1era iteración se lograron resolver dos no conformidades de las cinco encontradas, en la 2da iteración se encontraron tres no conformidades las cuales fueron resueltas en la 3ra iteración, de forma general se excluyeron todos los errores culminando las pruebas con resultados satisfactorios, como se muestra en la Figura 18.

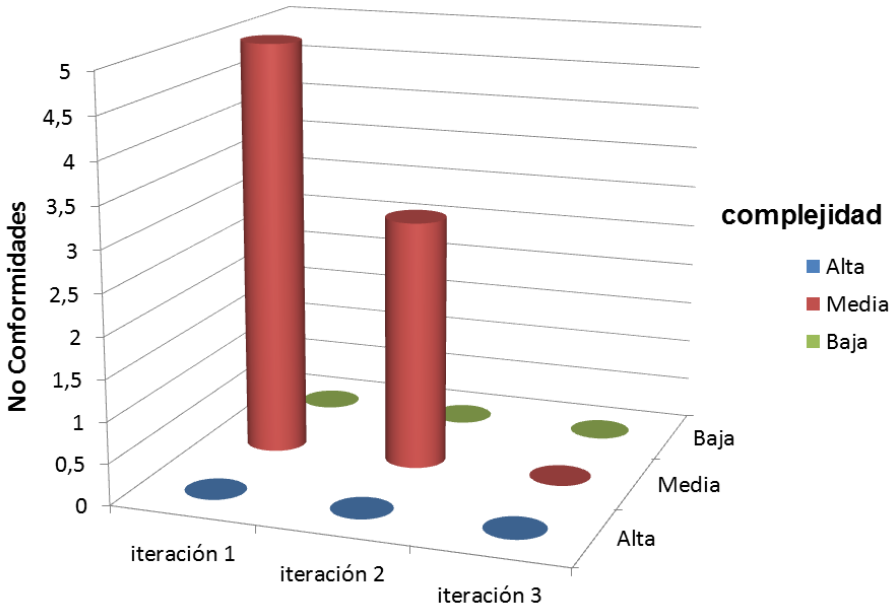


Figura 18. Resumen de las no conformidades encontradas.

3.4 Conclusiones parciales

Como resultado de este capítulo se han modelado los artefactos relacionados con los flujos de trabajo implementación y pruebas, quedando definido el diagrama de componentes del sistema. Se explicaron los estándares de codificación que se tendrán en cuenta en el desarrollo de la aplicación. Se realizaron las pruebas para validar las entradas y salidas de los datos sobre la interfaz del sistema, denominadas pruebas funcionales utilizando el método de caja negra, donde se encontraron cinco no conformidades que fueron resueltas demostrándose que cumple con la calidad y las restricciones de diseño, especificadas para este sistema. Se realizaron pruebas unitarias sobre el código a través del método de caja blanca, las cuales brindan una medición cuantitativa de la complejidad lógica de un programa.

Conclusiones generales

Con el desarrollo de este trabajo se arribó a las siguientes conclusiones:

- ✓ Con la investigación realizada sobre las herramientas de migración de base de datos se logró seleccionar Pentaho Data Integration como la herramienta indicada para ayudar con el desarrollo de la aplicación.
- ✓ Mediante el proceso de análisis y diseño de la herramienta de migración de base de datos se realizaron los diagramas correspondientes guiado por la metodología de desarrollo OpenUP, proporcionando el punto de partida para el desarrollo de la aplicación.
- ✓ Con la implementación realizada se obtuvo el software que soluciona los cuatro requisitos funcionales y seis no funcionales definidos. Esto se validó mediante pruebas funcionales y unitarias que se le realizaron al sistema.

Esta investigación logró satisfacer el problema de la investigación mediante el desarrollo de una herramienta para migrar la base de datos de la versión 1.8 del Generador Dinámico de Reportes a su versión 2.0.

Recomendaciones

Al finalizar la investigación, la implementación y las pruebas de la Herramienta de Migración de Base de Datos de GDR v1.8 a la v2.0 se recomienda:

- ✓ Actualizar el trabajo y las transformaciones diseñadas con Kettle que utiliza la aplicación, así como los XSL de Transformación, a medida de que surjan nuevos cambios en la base de datos y los reportes de GDR 2.0.

Bibliografía

1. **Abreu Medina, Aldis Joan, y otros. 2012.** Generador dinámico de reportes. [En línea] 2012. http://publicaciones.uci.cu/index.php/SC/article/view/889.
2. **Arevalo, Maria. 2010.** Introducción al Patrón de Arquitectura por Capas. [En línea] 2 de 12 de 2010. [Citado el: 4 de 4 de 2013.] http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/.
3. **Arias Marin , Marvin David. 2008.** Definición de lenguaje de programación. Tipos. Ejemplos. [En línea] 16 de 10 de 2008. [Citado el: 10 de 12 de 2012.] http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.
4. **Babylon Ltd. 2012.** Babylon 9. *Glosario de Informatica.* [En línea] 2012. [Citado el: 10 de 12 de 2012.] http://diccionario.babylon.com/java/.
5. **Bosque Viejo. 2009.** ETL: Extracción, Transformación y Carga | Bosque Viejo. [En línea] 12 de 3 de 2009. [Citado el: 9 de 12 de 2012.] http://bosqueviejo.net/2009/03/12/etl-extraccion-transformacion-y-carga/.
6. **Carrillo , Fernando. 2011.** Pruebas del Software. [En línea] 3 de 6 de 2011. http://ingenieriadepruebasdelsoftware.blogspot.com/2011/01/niveles-de-prueba-del-software.html.
7. **Clark, James . 1999.** W3C. *XSL Transformations (XSLT).* [En línea] 1999. [Citado el: 12 de 12 de 2012.] http://www.w3.org/TR/xslt.
8. **Codetel, Djheric O. 1998.** Métodos de prueba. [En línea] 1998. [Citado el: 10 de 4 de 2013.] http://farm.plista.com.
9. **Comparativa B.I. Open Source. 2010.** *Comparativa B.I. Open Source.* 2010.
10. **DaptaProERP. 2010.** DaptaProERP. *DaptaProERP.* [En línea] 2010. [Citado el: 22 de 11 de 2012.] http://www.datapronet.com/index.php?id_seccion=110&tipo_seccion=section.
11. **Date, Christopher. 2003.** *Introducción a los Sistemas de Base de Datos.* La Habana : Primera Parte, 2003.
12. **Definicion.De. 2008.** definicion.de. *Definición de reporte - Qué es, Significado y Concepto.* [En línea] 2008. [Citado el: 21 de 11 de 2012.] http://definicion.de/reporte/.
13. **Definición.DE;. 2008.** http://definicion.de/reporte/. *Definición de reporte - Qué es, Significado y Concepto.* [En línea] 2008. [Citado el: 9 de 12 de 2012.] definicion.de/reporte/.

14. **Dugarte, Ana, y otros. 2009.** Lenguaje Unificado de Modelado. [En línea] 2009. http://www.oocities.org/es/annadugarte/ads1/PRINCIPAL.htm.
15. **Espinosa, Roberto. 2010.** DataPrix. *Herramientas ETL. ¿Que son, para que valen?. Productos mas conocidos. ETL´s Open Source.* [En línea] 2010. [Citado el: 7 de 12 de 2012.] http://www.dataprix.com/blogs/respinosamilla/herramientas-etl-que-son-para-que-valen-productos-mas-conocidos-etl-s-open-sour.
16. **Espinosa, Roberto. 2010.** El Rincon del BI. [En línea] 1 de 6 de 2010. [Citado el: 14 de 12 de 2012.] http://churriwifi.wordpress.com/category/talend/ .
17. **Espinosa, Roberto. 2010.** Webminar sobre Talend Open Profiler. [En línea] 2010. [Citado el: 10 de 12 de 2012.] http://churriwifi.wordpress.com/category/talend/.
18. **ethek. 2010.** Definición del código fuente. [En línea] 2010. [Citado el: 5 de 4 de 2013.] http://www.ethek.com/definicion-del-codigo-fuente/.
19. **Figueroa , Pablo. 2010.** Conceptos en un Diagrama de Implementación. [En línea] 2010. [Citado el: 4 de 4 de 2013.] http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/implementacion01.html.
20. **Gazquez Martínez, Orelvi y Serrano García, Yadrian. 2011.** *Diseño e Implementación del módulo Diseñador de Consultas del Generador Dinámico de Reportes.* 2011.
21. **Gracia Luis, Luis Miguel. 2010.** Un poco de java. *Talend Open Studio: Herramienta para transformaciones de datos.* [En línea] 21 de 6 de 2010. [Citado el: 12 de 12 de 2012.] http://unpocodejava.files.wordpress.com/2010/06/image00411.jpg.
22. **Heffelfinger, David R. 2006.** Jasperreports Reporting for Java Developers. Birmingham,UK : Packt Publishing Ltd, 2006.
23. **INGENIERIA DE SOFTWARE II. 2010.** Desarrollo de Software. *Metodologias,Frameworks y Modelos de Desarrollo De Software.* [En línea] 7 de 2 de 2010. [Citado el: 8 de 12 de 2012.] http://softwareiimarfrednarvaez.blogspot.com/2012_02_01_archive.html.
24. **jaspersoft community. 2000.** jaspersoft community. [En línea] 2000. [Citado el: 18 de 12 de 2012.] http://community.jaspersoft.com/.
25. **Lamarca Lapuente, María Jesús.** Bases de Datos. *Bases de Datos.* [En línea] [Citado el: 10 de 12 de 2012.] http://www.hipertexto.info/documentos/b_datos.htm.
26. **LARMAN, CRAIG. 1999.** *UML Y PATRONES.Introducción al análisis y diseño orientado a objetos.* s.l. : PRENTICE HAL, 1999. 970-17-0261-1..

27. **mastermagazine. 2010.** Definición de Conversión. *Definición de Conversión.* [En línea] 2010. [Citado el: 11 de 12 de 2012.] http://www.mastermagazine.info/termino/4432.php.
28. **Menéndez-Barzanallana Asensio, Rafael. 2012.** Departamento Informática y Sistemas. Universidad de Murcia. [En línea] 3 de 11 de 2012. [Citado el: 12 de 12 de 2012.] http://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Que-son-lenguajes-marcado.html.
29. **Parra, Eduardo. 2011.** Portal Ubuntu. [En línea] 2011. http://www.portalubuntu.com/2011/04/instalar-netbeans-70-en-espanol-en.html.
30. **Pentaho Corporation. 2011.** Pentaho. [En línea] 2011. [Citado el: 12 de 12 de 2012.] http://forums.pentaho.com/showthread.php?68345-CloverTL-comparison-CloverETL-Talend-and-PDI.
31. **Pérez, Alberto. 2012.** Introducción a Talend. [En línea] 26 de 3 de 2012. http://www.brujuleo.es/introduccion-a-talend/.
32. **pgAdmin PostgreSQL Tools. 2012.** [En línea] 2012. [Citado el: 12 de 12 de 10.] http://www.pgadmin.org/.
33. **Pressman, Roger S. 1998.** *Ingeniería del software. Un enfoque práctico.* Mc Graw Hill : s.n., 1998. 614.
34. **Proal, Carlos. 2010.** Herramientas ETL. *Herramientas ETL.* [En línea] 2010. [Citado el: 12 de 12 de 2012.] http://www.carlosproal.com/dw/dw05.html.
35. **PRUEBASDESOFTWARE. 2005.** Gestión de Calidad y Pruebas de Software. [En línea] 2005. [Citado el: 4 de 4 de 2013.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.
36. **RAE. 2012.** RAE. [En línea] 2012. [Citado el: 27 de 11 de 2012.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=metodolog%C3%ADa.
37. **Rangel, Eustaquio. 2006.** *Manual de PHPReports.* 2006.
38. **Reyna, Rafael. 2010.** Herramientas Case. [En línea] 9 de 5 de 2010. [Citado el: 12 de 12 de 2012.] http://es.scribd.com/doc/36908565/Herramientas-Case.
39. **Rodríguez, Jose. 2012.** Modelo de Implementación. [En línea] 2012. [Citado el: 4 de 4 de 2013.] merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=96&Itemid=297.

40. **Silva Hernández, Iliana Amabely. 2003.** *Generador Automático de Reportes Dinámicos.* [En línea] 2003. [Citado el: 10 de 12 de 2012.] http://www.cs.cinvestav.mx/tesisgraduados/2003/resumenIlianaAma.
41. **Softpedia. 2013.** Softpedia. [En línea] 2013. [Citado el: 27 de 1 de 2013.] http://webscripts.softpedia.com/scriptScreenshots/CloverETL-Screenshots-52008.html.
42. **Somerville, IAN . 2005.** *Ingeniería del software .* Madrid : Séptima Edición, 2005.
43. **Yunta, Luis Rodríguez. 2001.** *Bases de Datos Documentales.* MALDONADO,Angeles : CINDOC, 2001.

Referencias Bibliográficas

1. **Arias Marin , Marvin David. 2008.** Definición de lenguaje de programación. Tipos. Ejemplos. [En línea] 16 de 10 de 2008. [Citado el: 10 de 12 de 2012.] http://catedraprogramacion.foroactivos.net/t83-definicion-de-lenguaje-de-programacion-tipos-ejemplos.
2. **Babylon Ltd. 2012.** Babylon 9. *Glosario de Informatica.* [En línea] 2012. [Citado el: 10 de 12 de 2012.] http://diccionario.babylon.com/java/.
3. **Clark, James . 1999.** W3C. *XSL Transformations (XSLT).* [En línea] 1999. [Citado el: 12 de 12 de 2012.] http://www.w3.org/TR/xslt.
4. **Codetel, Djheric O. 1998.** Métodos de prueba. [En línea] 1998. [Citado el: 10 de 4 de 2013.] http://farm.plista.com.
5. **Figueroa , Pablo. 2010.** Conceptos en un Diagrama de Implementación. [En línea] 2010. [Citado el: 4 de 4 de 2013.] http://webdocs.cs.ualberta.ca/~pfiguero/soo/uml/implementacion01.html.
6. **Lamarca Lapuente, María Jesús.** Bases de Datos. *Bases de Datos.* [En línea] [Citado el: 10 de 12 de 2012.] http://www.hipertexto.info/documentos/b_datos.htm.
7. **Menéndez-Barzanallana Asensio, Rafael. 2012.** Departamento Informática y Sistemas. Universidad de Murcia. [En línea] 3 de 11 de 2012. [Citado el: 12 de 12 de 2012.] http://www.um.es/docencia/barzana/DIVULGACION/INFORMATICA/Que-son-lenguajes-marcado.html.
8. **Reyna, Rafael. 2010.** Herramientas Case. [En línea] 9 de 5 de 2010. [Citado el: 12 de 12 de 2012.] http://es.scribd.com/doc/36908565/Herramientas-Case.
9. **Silva Hernández, Iliana Amabely. 2003.** *Generador Automático de Reportes Dinámicos.* [En línea] 2003. [Citado el: 10 de 12 de 2012.] http://www.cs.cinvestav.mx/tesisgraduados/2003/resumenIlianaAma.
10. **Abreu Medina, Aldis Joan, y otros. 2012.** Generador dinámico de reportes. [En línea] 2012. http://publicaciones.uci.cu/index.php/SC/article/view/889. [Citado el: 10 de 12 de 2012.].
11. **Arevalo, Maria. 2010.** Introducción al Patrón de Arquitectura por Capas. [En línea] 2 de 12 de 2010. [Citado el: 4 de 4 de 2013.] http://arevalomaria.wordpress.com/2010/12/02/introduccion-al-patron-de-arquitectura-por-capas/.

12. **Bosque Viejo. 2009.** ETL: Extracción, Transformación y Carga | Bosque Viejo. [En línea] 12 de 3 de 2009. [Citado el: 9 de 12 de 2012.] http://bosqueviejo.net/2009/03/12/etl-extraccion-transformacion-y-carga/.
13. **Comparativa B.I. Open Source. 2010.** *Comparativa B.I. Open Source.* 2010. [Citado el: 9 de 12 de 2012.].
14. **DaptaProERP. 2010.** DaptaProERP. *DaptaProERP.* [En línea] 2010. [Citado el: 22 de 11 de 2012.] http://www.datapronet.com/index.php?id_seccion=110&tipo_seccion=section.
15. **Definicion.De. 2008.** definicion.de. *Definición de reporte - Qué es, Significado y Concepto.* [En línea] 2008. [Citado el: 21 de 11 de 2012.] http://definicion.de/reporte/.
16. **Definición.DE;. 2008.** http://definicion.de/reporte/. *Definición de reporte - Qué es, Significado y Concepto.* [En línea] 2008. [Citado el: 9 de 12 de 2012.] definicion.de/reporte/.
17. **Dugarte, Ana, y otros. 2009.** Lenguaje Unificado de Modelado. [En línea] 2009. http://www.oocities.org/es/annadugarte/ads1/PRINCIPAL.htm. [Citado el: 9 de 12 de 2012.].
18. **Espinosa, Roberto. 2010.** DataPrix. *Herramientas ETL. ¿Que son, para que valen?. Productos mas conocidos. ETL´s Open Source.* [En línea] 2010. [Citado el: 7 de 12 de 2012.] http://www.dataprix.com/blogs/respinosamilla/herramientas-etl-que-son-para-que-valen-productos-mas-conocidos-etl-s-open-sour.
19. **Espinosa, Roberto. 2010.** El Rincon del BI. [En línea] 1 de 6 de 2010. [Citado el: 14 de 12 de 2012.] http://churriwifi.wordpress.com/category/talend/ .
20. **Espinosa, Roberto. 2010.** Webminar sobre Talend Open Profiler. [En línea] 2010. [Citado el: 10 de 12 de 2012.] http://churriwifi.wordpress.com/category/talend/.
21. **ethek. 2010.** Definición del código fuente. [En línea] 2010. [Citado el: 5 de 4 de 2013.] http://www.ethek.com/definicion-del-codigo-fuente/.
22. **Gracia Luis, Luis Miguel. 2010.** Un poco de java. *Talend Open Studio: Herramienta para transformaciones de datos.* [En línea] 21 de 6 de 2010. [Citado el: 12 de 12 de 2012.] http://unpocodejava.files.wordpress.com/2010/06/image00411.jpg.
23. **INGENIERIA DE SOFTWARE II. 2010.** Desarrollo de Software. *Metodologias,Frameworks y Modelos de Desarrollo De Software.* [En línea] 7 de 2 de 2010. [Citado el: 8 de 12 de 2012.] http://softwareiimarfrednarvaez.blogspot.com/2012_02_01_archive.html.

24. **jaspersoft community. 2000.** jaspersoft community. [En línea] 2000. [Citado el: 18 de 12 de 2012.] http://community.jaspersoft.com/.
25. **LARMAN, CRAIG. 1999.** *UML Y PATRONES.Introducción al análisis y diseño orientado a objetos.* s.l. : PRENTICE HAL, 1999. 970-17-0261-1..
26. **mastermagazine. 2010.** Definición de Conversión. *Definición de Conversión.* [En línea] 2010. [Citado el: 11 de 12 de 2012.] http://www.mastermagazine.info/termino/4432.php.
27. **Parra, Eduardo. 2011.** Portal Ubuntu. [En línea] 2011. http://www.portalubuntu.com/2011/04/instalar-netbeans-70-en-espanol-en.html. [Citado el: 11 de 12 de 2012.].
28. **Pentaho Corporation. 2011.** Pentaho. [En línea] 2011. [Citado el: 12 de 12 de 2012.] http://forums.pentaho.com/showthread.php?68345-CloverTL-comparison-CloverETL-Talend-and-PDI.
29. **pgAdmin PostgreSQL Tools. 2012.** [En línea] 2012. [Citado el: 12 de 12 de 10.] http://www.pgadmin.org/.
30. **Proal, Carlos. 2010.** Herramientas ETL. *Herramientas ETL.* [En línea] 2010. [Citado el: 12 de 12 de 2012.] http://www.carlosproal.com/dw/dw05.html.
31. **PRUEBASDESOFTWARE. 2005.** Gestión de Calidad y Pruebas de Software. [En línea] 2005. [Citado el: 4 de 4 de 2013.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.
32. **RAE. 2012.** RAE. [En línea] 2012. [Citado el: 27 de 11 de 2012.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=metodolog%C3%ADa.
33. **Rangel, Eustaquio. 2006.** *Manual de PHPReports.* 2006. [Citado el: 27 de 11 de 2012.].
34. **Rodríguez, Jose. 2012.** Modelo de Implementación. [En línea] 2012. [Citado el: 4 de 4 de 2013.] merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=96&Itemid=297.
35. **Softpedia. 2013.** Softpedia. [En línea] 2013. [Citado el: 27 de 1 de 2013.] http://webscripts.softpedia.com/scriptScreenshots/CloverETL-Screenshots-52008.html.

Anexos

Anexo 1: Interfaz principal de la herramienta.

Figura 19. Interfaz principal de la herramienta.

Anexo 2: Patrones de diseño GOF Singleton y GRASP Creador.

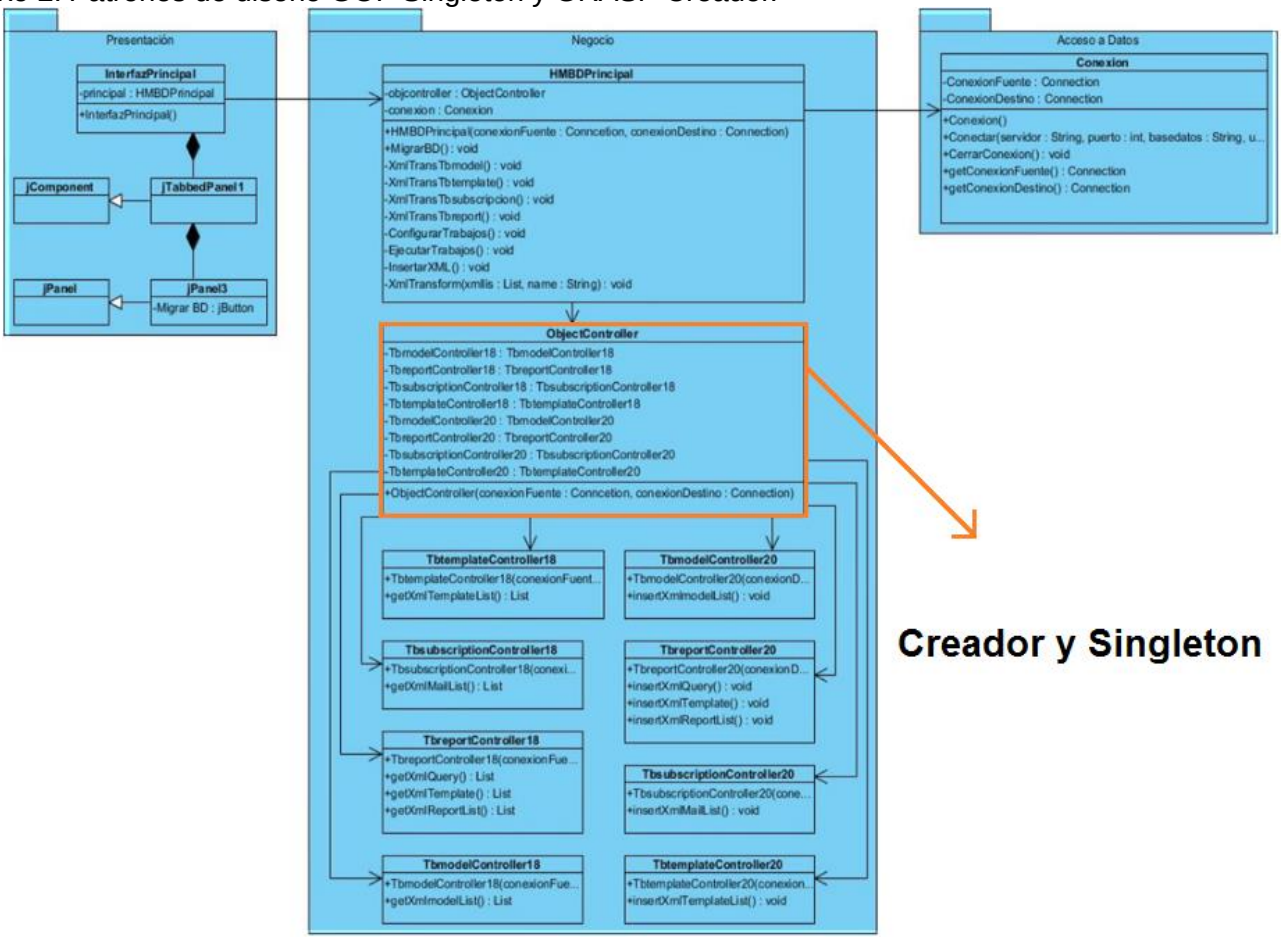


Figura 20. Patrón Creador y Singleton.

Anexo 3: Patrón GRASP Experto.

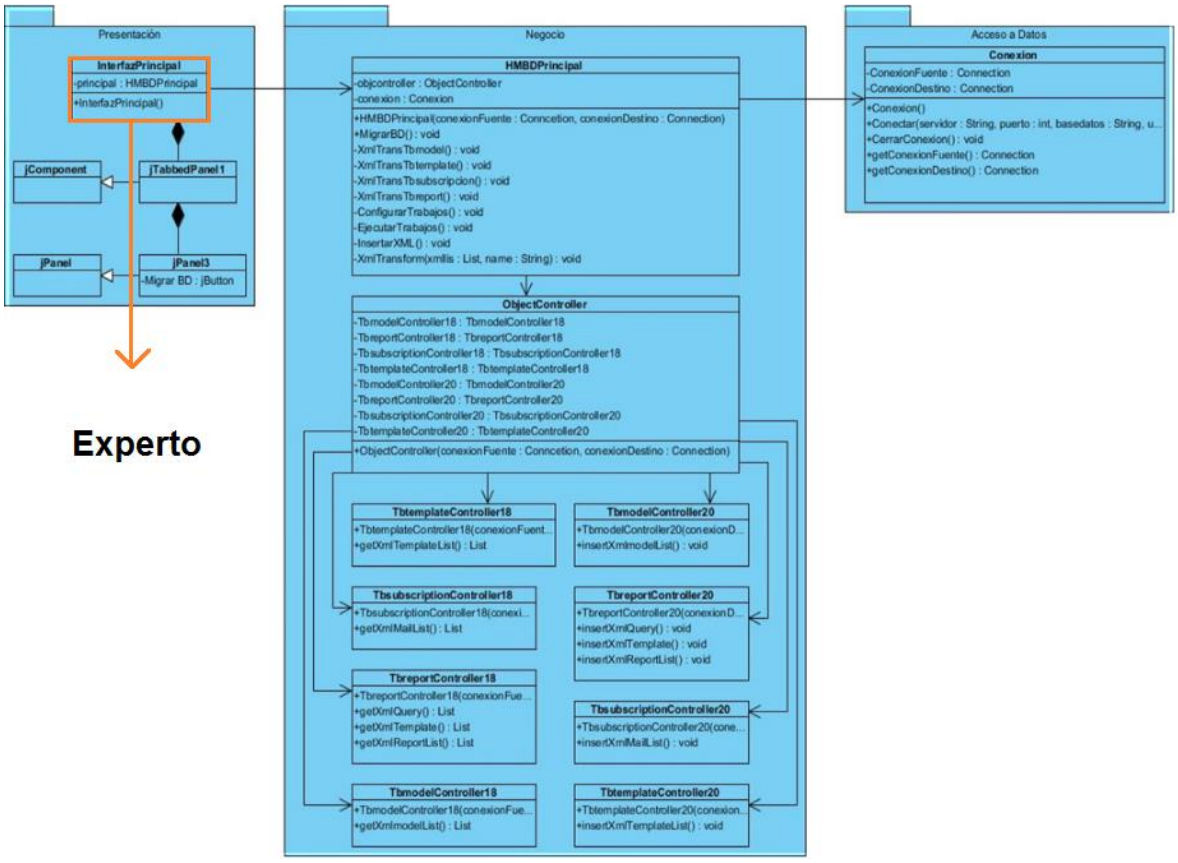


Figura 21. Patrón Experto.

Anexo 4: Patrón GRASP Controlador.

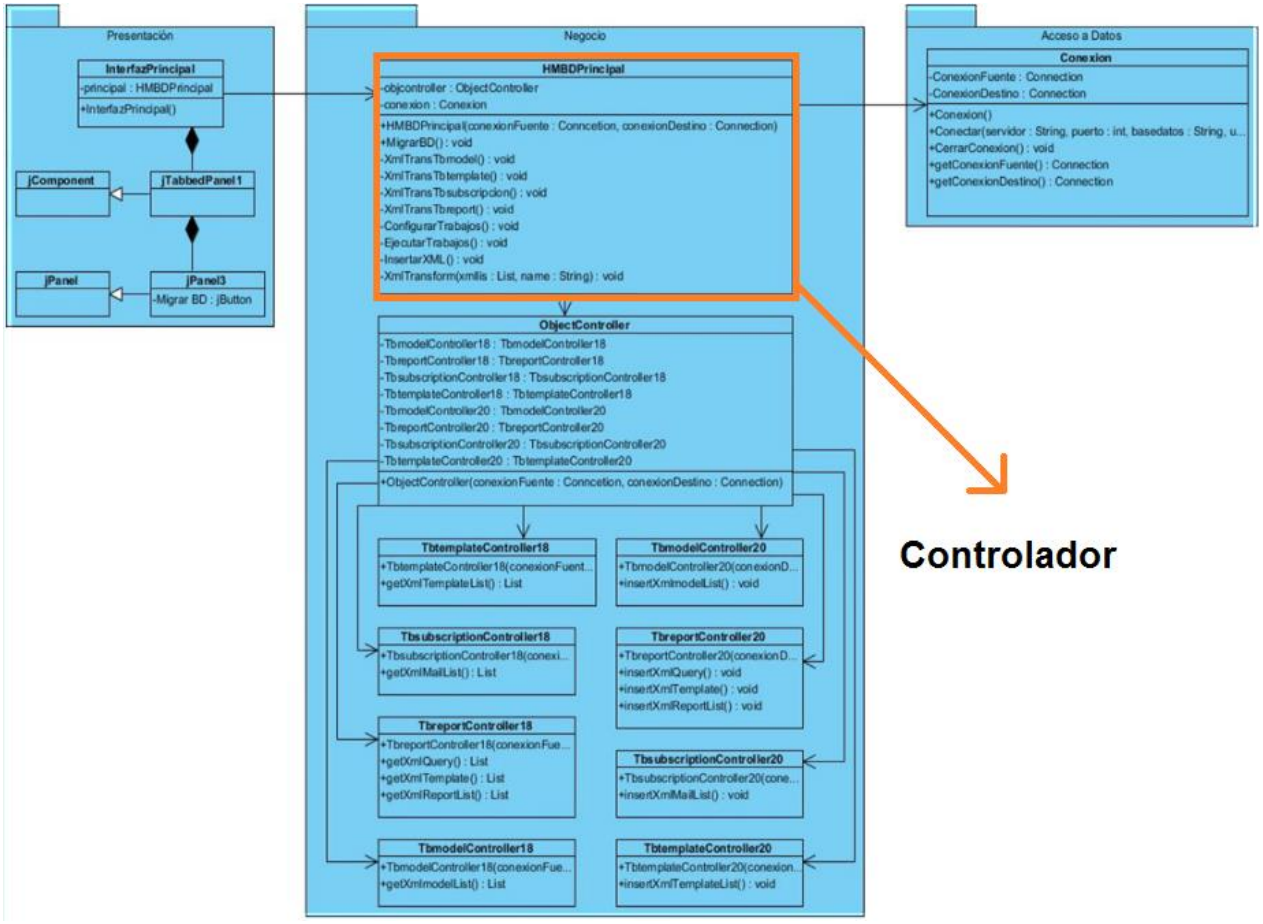


Figura 22. Patrón Controlador.

1

Glosario de Términos

BD: Base de Datos.

CASE: Ingeniería de Software Asistida por Computación.

CU: Caso de Uso.

CUS: Caso de Uso del Sistema.

DATEC: Centro de Tecnologías de Gestión de Datos.

GDR: Generador dinámico de reportes.

GOF: Gang of Four.

GRASP: General Responsability Assignment Software Patterns (Patrones de Software para la asignación General de Responsabilidades).

IDE: Entorno desarrollo integrado**.**

OpenUP: Proceso Unificado Abierto.

UML: Unified modeling language(Lenguaje Unificado de Modelado).

XML: Lenguaje de Marcas Extensible.