Universidad de las Ciencias Informáticas

Facultad 8

Sistema para la gestión de los artefactos manejados en los procesos de Extracción, Transformación y Carga de los Datos al Almacén de Datos.

(DATEC)

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Jailen Irán Cuadrado Pardillo, Raúl Antonio Alonso Benavidez.

Tutor: Ing. Omar Ahmed García Pérez.

Cotutor: Ing. Osniel Hernández Calvo.

Curso docente: 2009-2010

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 ___ del año____.

_____	_____
Firma del Autor	Firma del Autor
Jailen Irán Cuadrado Pardillo	Raúl Antonio Alonso Benavides.
_____	_____
Firma del Tutor	Firma del Cotutor
Ing. Omar Ahmed García Pérez	Ing. Osniel Hernández Calvo.

"… lo que da al hombre el poder no es el mero conocimiento que viene del uso de los sentidos, sino ese otro conocimiento más profundo que se llama Ciencia."

José Martí

Agradecimientos

Quisiera agradecer a mis padres por la educación que me han brindado. A mi madre por su paciencia y sabiduría infinita a la hora de corregir mis errores, a mi padre por mostrarme la importancia de ser honesto en la vida. A mi hermana, mi única y querida hermana que ha sabido ser una madre para mí por poner su granito de arena en mi educación con su ejemplo y por siempre estar pendiente de mi. A mi abuelita linda, esa viejita imparable a pesar de los años que me ha mostrado como vivir una vida plena y feliz. A todos mis tíos y tías de los que siempre he recibido amor y cariño. A mi abuelo David por enseñarme a ser una buena persona y atento con los demás al cual le doy gracias también porque es gracias a sus manos que puedo caminar sin problemas. A mi tía Alicia por transmitirme su amor por los animales y por enseñarme a enfrentar a la muerte con dignidad. A la gente del barrio por acogerme como si fuera un hijo mas a mi hermano Alejandro por compartir tantos momentos juntos y aprender uno del otro. A mis amigas y amigos de la infancia Soe, Yeni, Bárbaro, Daniel, Wendy etc. por mantenerse fieles a pesar de la distancia. A mi ahijado Jean Pablo y a sus padres por darme ese honor. Y por supuesto a mis hermanos y hermanas de la universidad en especial a la gente del Voltaje XXL a Lara, Yuri, Denis, Danais, Isabel, Daniel, El filio, Yoan, Mario, Matojo, Pedro, Lucas, el Gorock, a Indirin a Yoandra, Yone y a todos los demás. A la gente de mi nueva aula también al Trujo y a Lisbe, a Yanio por sus consejos. A mis compañeros de madrugas y madrugas de andanzas por el mundo, A Celia y a la gente de calidad sin las que la culminación de este trabajo no sería posible, a las muchachitas de segundo que entraron en mi vida y ya tienen un huequito en mi corazón. Al tribunal por sus críticas constructivas, a la gente del centro que nos ayudaron a Frank y por supuesto al compañero Omar tutor de este trabajo y no podían faltar a mis amigos de siempre Abel y el Rene el cual se convirtió en nuestro salvador cada vez que se nos presentaba algún problema con la aplicación. Y dejar para el final a cierta personita intranquila y atolondrada a la cual conocí en segundo año y se convirtió en un hermano más para mí y que este curso fue mi compañero de tesis a Jailen el cual me ha soportado estos años gracias por todo.

Dedicatoria

A mi padres a mi hermana a mi abuela y a toda aquella gente que siempre ha tenido fe en mi.

Resumen

Con el desarrollo de las Tecnologías de la información y las comunicaciones en especial aquellas relacionadas con el almacenamiento y gestión de la información, la utilización de aplicaciones que automaticen el trabajo con las mismas se convierte en una necesidad para empresas, instituciones sociales y demás entidades por los beneficios que aportan al cliente, entregando al mismo información actualizada así como rapidez en la manipulación del contenido que sea de su interés. El objetivo del siguiente trabajo diploma es la implementación de un sistema que permita la gestión de los artefactos generados en el proceso de extracción, transformación y carga de los datos (ETL). El desarrollo de la misma estuvo guiado por las especificaciones que propone la metodología RUP, obteniéndose una serie de artefactos que se generan en cada uno de los flujos de trabajo por los que se transita. Para realizar su implementación fueron utilizadas diversas herramientas, las cuales fueron fruto de la investigación previamente realizada para poder asimilar el desarrollo de las tecnologías y tendencias actuales. Una vez completado el sistema posibilitará un adecuado proceso de gestión de la información de los artefactos que se generan en el proceso ETL de los almacenes de datos pertenecientes a los proyectos productivos de la universidad particularmente el Centro de tecnologías de gestión de datos (DATEC) mediante una aplicación rápida y de fácil manejo que le permita a los arquitectos ETL manejar la información y obtener beneficios de esto según sus necesidades.

Índice

Introducción

A lo largo de la historia el hombre siempre ha querido simplificar su modo de vida, por esta razón los grandes pensadores de todos los tiempos han dedicado gran parte de su vida a desarrollar teorías matemáticas para construir máquinas que faciliten las tareas de la vida diaria. En la actualidad la humanidad ha alcanzado un notable nivel de desarrollo técnico y social en gran medida a partir de la segunda mitad del siglo XX, es innegable que la informática ha pasado a comandar en cuanto a avances tecnológicos se refiere y que rápidamente se ha convertido en una ciencia clave para el progreso de la misma. Gracias a ella enormes distancias quedan reducidas a solo un clic, *y* es posible procesar operaciones muy complejas y manejar gran cantidad de información prácticamente sin esfuerzo y en un tiempo mínimo. Para poder trabajar con este cúmulo de datos son muy útiles los almacenes de datos (del inglés *data warehouse*).

Un almacén de datos es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas, para luego procesarla permitiendo su análisis desde infinidad de perspectivas y con grandes velocidades de respuesta.

Se caracteriza por ser:

- Integrado: Los datos almacenados en el almacén de datos deben integrarse en una estructura consistente, por lo que las inconsistencias existentes entre los diversos sistemas operacionales deben ser eliminadas.
- Temático: Sólo los datos necesarios para el proceso de generación del conocimiento del negocio se integran desde el entorno operacional. Los datos se organizan por temas para facilitar su acceso y entendimiento por parte de los usuarios finales. Por ejemplo, todos los datos sobre clientes pueden ser consolidados en una única tabla.
- Histórico: Se carga con los distintos valores que toma una variable en el tiempo para permitir comparaciones.
- No volátil: La información es permanente, significando la actualización del almacén de

datos, la incorporación de los últimos valores que tomaron las distintas variables contenidas en él sin ningún tipo de acción sobre lo que ya existía.

Para comprender íntegramente el concepto de almacén de datos es importante entender cuál es el proceso de construcción del mismo, denominado ETL (Extracción, Transformación y Carga), a partir de los sistemas operaciones de una compañía:

- Extracción: obtención de información de las distintas fuentes tanto internas como externas.
- Transformación: filtrado, limpieza, depuración, homogeneización y agrupación de la información.
- Carga: organización y actualización de los datos y los metadatos en la base de datos.

Cuba no está exenta de esta tecnología, empresas como la Empresa de Telecomunicaciones de Cuba (ETECSA), la Oficina Nacional de Estadística (ONE) hacen uso de estos almacenes de datos debido a que les facilita enormemente el trabajo con el volumen de información que manejan. La Universidad de las Ciencias Informáticas (UCI) se destaca en el uso de estos almacenes de datos por el manejo información el cual es cada vez mayor, apreciándose claramente en los centros de desarrollo como el Centro de Almacenamiento de Datos (DATEC).

El Centro de Almacenamiento de Datos (DATEC), fue creado en el curso 2008-2009, y se especializa en proveer soluciones integrales y consultorías relacionadas con tecnologías de bases de datos y análisis de información e implementa un modelo de desarrollo organizado en líneas o departamentos de producción. Una de ellas es la Línea de Almacenes, la cual genera una serie de artefactos que son los encargados de registrar los datos necesarios para el trabajo en el centro.

Todos estos artefactos son llenados de forma manual por parte del equipo de desarrollo y almacenado en forma de documentos, lo cual dificulta enormemente el trabajo con los mismos puesto que algunos se encuentran relacionados estrechamente y además pueden llegar a tener un volumen de información considerable. Por lo que el **problema a resolver** consiste en:

La insuficiencia de las herramienta que automatizan la gestión de los artefactos resultantes de los procesos de extracción, transformación y carga de datos de un almacén de datos está afectando la productividad de las soluciones de los proyectos de almacenes de datos del DATEC en relación a la calidad de los entregables y el tiempo de realización de estos por el equipo de desarrollo.

Objetivo General:

El objetivo de la investigación es desarrollar un sistema de gestión que sea capaz de manejar los artefactos que se construyen en el proceso de extracción, transformación y carga de datos al almacén de datos, así como las relaciones entre los mismos, y además exporte a un formato acordado previamente con el grupo de calidad del centro.

El **objeto de estudio** consiste en el proceso de extracción, transformación y carga de datos.

Los **objetivos específicos** que abarca la investigación son:

- Realizar un estudio sobre los artefactos que se construyen en el proceso de integración de datos y las relaciones entre los mismos.
- Definir con el grupo de calidad el formato en el que se exportarán los artefactos por el sistema a implementar y que los artefactos generados por el sistema sean aceptados por el expediente de proyecto de almacenes de datos delel Centro de Almacenamiento de Datos (DATEC).
- Desarrollar el sistema a través del análisis, diseño, implementación y prueba del mismo.

El **campo de acción** de la investigación es la gestión documental de los procesos de extracción, transformación y carga de los datos en sistemas de almacenes de datos.

Idea a Defender:

Con la automatización de la gestión de los artefactos resultantes de los procesos de extracción, transformación y carga de datos de un almacén de datos, el equipo de desarrollo podrá

aumentar la productividad de las soluciones de los proyectos de almacenes de datos del centro, la calidad de los entregables y disminuir el tiempo de realización.

Métodos de Investigación:

Fue necesario apoyarse en estrategias y métodos de investigación para obtener puntos de vista claves una vez abordada la realidad. Los métodos científicos de investigación empleados fueron los siguientes: como métodos teóricos se hizo necesario utilizar el método de Análisis histórico-lógico y el método Analítico-sintético, además, como Método Empírico para la obtención de información se utilizó la entrevista.

La aplicación de su conjunto moldeó una solución sólida al problema planteado, partiendo desde el enfoque histórico lógico del estudio del desarrollo de sistemas que permitan gestionar los artefactos manejados en el proceso de extracción, transformación y carga de los datos al almacén de datos y transitando por el planteamiento de objetivos específicos y una hipótesis que termina consolidándose mientras la investigación se profundiza. Gracias al análisis y la síntesis y al procesamiento de todo el cúmulo de información disponible se produjo el arribo a importantes conclusiones acerca del tema.

Estructura capitular

En el primer capítulo, denominado **Fundamentación Teórica** comprende un análisis de los sistemas que existen en la actualidad y se vinculan con la investigación, y el estado del arte de las tecnologías y herramientas a utilizar en el desarrollo de la aplicación. En el segundo capítulo: **Análisis y Diseño del Sistema,** se realiza el análisis del sistema a desarrollar, con el propósito de refinar y estructurar los requisitos obtenidos con anterioridad para facilitar la comprensión, preparación, modificación y mantenimiento de los mismos. Se describen los aspectos relacionados al diseño de la solución propuesta, se muestran los diagramas de clases del diseño y se especifican los principios para el diseño gráfico, finalmente el capítulo, **Implementación y Prueba** trata los aspectos relacionados con la construcción de la solución propuesta, se ilustran

los diagramas de componentes y despliegue, se aborda la descripción de los estándares de diseño, codificación y además del tratamiento de errores en la solución del sistema, así como una vez terminado el software probar que los artefactos que el mismo genera son aceptados por el expediente de calidad del centro.

Capitulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se hace referencia de los elementos teóricos que fundamentan el desarrollo del sistema informático propuesto. También contempla un análisis efectuado sobre el estudio del arte de las posibles herramientas y tecnologías que pudieran ser utilizadas para el desarrollo del sistema, posibilitando la toma de decisiones sobre qué es más factible utilizar en función de cumplimentar los objetivos generales de la investigación y solución del problema planteado.

1.2 Gestión de los artefactos generados en el proceso ETL

Para el sistema a desarrollar es necesario tener en cuenta el método utilizado en la organización para la que se realiza el producto, este consiste en la elaboración de una serie de artefactos que permiten almacenar toda la información relacionada con el proceso de extracción, transformación y carga del almacén de datos del centro.

1.2.1 Artefactos

Estos artefactos constituyen una manera de almacenar toda la información que se obtiene al realizar el proceso de extracción transformación y carga para lograr un buen control y organización.

Se clasifican en 5 tipos:

1.2.1.1 Diccionario de datos: Documentación detallada de la fuente de datos, Conceptos de funcionamiento, recoge el significado de cada una de las variables con los posibles valores correctos y erróneos y las abreviaturas con las cuales se identifica en la fuente. (Doc. Pedro Yobanis Piñero, 2009)

1.2.1.2 Registro de sistemas fuente: Información referente a cada una de las fuentes de datos, características técnicas como el tipo de Sistema de Administración de Base de Datos con que se administran los sistemas fuentes, el tamaño de la Base de datos (en GB) o aspectos de orden general como a cuál departamento pertenece esta Base de Datos o quiénes el administrador de la misma. (Doc. Pedro Yobanis Piñero, 2009)

1.2.1.3 Reglas del negocio: Reglas de negocio con los detalles de las transformaciones que deberán ser llevadas a cabo por los procesos de extracción, modificación y carga y a quién se debe reportar los errores encontrados en los datos. (Doc. Pedro Yobanis Piñero, 2009)

1.2.1.4 Mapa Lógico de datos: Especificación del mapeo de fuente a destino de los datos, o sea de que Base de datos, Tabla y Columna se debe sacar cada uno de los datos que se necesitan para poblar el almacén. Es de mucha importancia y con él trabajará el arquitecto ETL y el modelador de datos del almacén para desarrollar los *mapping* de fuente a destino. (Doc. Pedro Yobanis Piñero, 2009)

1.2.1.5 Perfil de Datos: La limpieza de los datos debe comenzar realmente antes de dar el primer paso hacia la construcción del sistema ETL. El análisis del perfil de los datos debe llevarse a cabo durante las fases de planificación y diseño. El reporte del perfil de los datos adquiere la forma de un repositorio de metadatos específicos. (Doc. Pedro Yobanis Piñero, 2009)

1.2.2Sistemas de gestión de los artefactos generados en el proceso de ETL

La información acumulada en los artefactos posee un volumen demasiado grande y si se le añade que es necesario estar constantemente realizando consultas y cambios a la misma se llega a la conclusión de que su trabajo de forma manual es algo engorroso y consume una buena cantidad de tiempo, tal y como sucede hoy en día en el centro. Es por esto que actualmente existen sistemas que automatizan la gestión de los artefactos, ejemplo de ello lo constituyen:

1.2.2.1 Octopus

Octopus es una herramienta de ETL basada en Java y que se puede conectar a cualquier fuente

JDBC (*Java Database Connectivity*) y realizar la transformación que se encuentra definida en un archivo XML.

Diferentes bases de datos pueden ser mezcladas de forma simultánea por Octopus (*Microsoft SQL Server*, *Oracle*, con *Excel* y *Access*, *MySQL*, entre otras). Octopus soporta a la *Ant and JUnit* para automatizar los procesos como: crear base de datos y tablas y los procesos de extracción y carga, en producción o en pruebas.

Esta herramienta se centra más en el proceso de ETL que en la gestión de los artefactos generados por el mismo por lo que no es factible para la utilización del centro. (ACIS, 2005)

1.2.2.2 IBM InfoSphere Master Data Management Server

La familia de productos IBM InfoSphere proporciona una plataforma de información unificada para integrar, conciliar, gestionar y analizar, desde diferentes silos o sistemas de distintos proveedores, datos y contenido, independientemente de su tipo, volumen o complejidad, de forma tan rápida como sea necesario, y después ofrecerlos en contexto a los usuarios, a las aplicaciones y a los procesos empresariales. Así se crea una única versión de los datos reales que se puede utilizar para todo lo relativo a los clientes, productos, cuentas o cualquier otro dominio de información. Los productos IBM InfoSphere tienen la capacidad de gestionar el ciclo de vida completo de todos los tipos de información, ya sea histórica o transaccional, lo que incluye datos y contenido, así como de integrar fácilmente nuevas entidades empresariales, fuentes de información, definiciones y usos a medida que se van añadiendo o cambiando.

Aunque esta herramienta es muy útil para el manejo de los metadatos de cualquier empresa constituye un software de carácter propietario, además el manejo de datos es de carácter global, es decir no se centra en ninguno de manera específica, lo cual resulta en un inconveniente para el centro, ya que cada línea maneja sus artefactos de manera independiente. (IBM-España)

1.3 Metodologías, herramientas y tecnologías para el sistema

Para proceder a la construcción de la herramienta es necesario realizar un análisis de las

metodologías, tecnologías y herramientas existentes que pueden contribuir al desarrollo de la misma. Si se desea lograr una selección óptima es fundamental el estudio profundo de cada una de ellas, poniendo especial atención a aquellas características que la conviertan en una candidata factible a utilizar respetando y manteniendo siempre las políticas de desarrollo del centro.

1.3.1 Metodología de desarrollo de software

Se entiende por metodología de desarrollo a una colección de pasos a seguir los cuales guían el trabajo referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la calidad del trabajo, logrando con ello el cumplimiento de los requisitos iniciales, minimizando las pérdidas de tiempo en el proceso de generación de software.

1.3.1.1 Programación Extrema

La Programación Extrema (XP), o *eXtreme Programming*, intenta como metodología ágil reducir la complejidad del software por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

En todas las iteraciones del ciclo de vida de XP tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración. El ciclo de vida ideal de XP consiste de seis fases: Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento

y Muerte del Proyecto. Lo fundamental en este tipo de metodología es la comunicación, entre los usuarios y los desarrolladores, la simplicidad al desarrollar y codificar los módulos del sistema y la retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales. (Letelier, 2006)

1.3.1.2 El Proceso Unificado de Desarrollo

El Proceso Unificado de Desarrollo (RUP) es una metodología para la ingeniería de software que va más allá del mero análisis y diseño orientado a objetos, para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por casos de uso, iterativo e incremental y centrado en la arquitectura. El Proceso Unificado se basa en componentes, lo que significa que el sistema en construcción está hecho de componentes de software interconectados por medio de interfaces bien definidas. Usa el Lenguaje Unificado de Modelado (UML) en la preparación de todos los planos del sistema. Presenta como características que unifica los mejores elementos de metodologías anteriores, está preparado para desarrollar proyectos grandes y complejos y es orientado a objetos. Como es un proceso, en su modelación describe entre sus principales elementos a los trabajadores, artefactos, actividades y flujo de actividades, o sea, plantea quién va a hacer qué, cómo y cuándo para lograr un determinado objetivo.

Para desarrollar la propuesta de solución que presenta este trabajo, se ha decidido utilizar esta metodología dado que representa una propuesta de proceso cualitativamente superior a XP puesto que combina lo mejor de esta y algunas otras metodologías para garantizar la elaboración de todas las fases de desarrollo de un producto de software orientado a objeto.(Molpeceres, 2002)

1.3.2 Lenguaje de Modelado

El lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño.

UML es un lenguaje de modelado para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Entrega una forma de modelar elementos conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. Es un sistema notacional destinado a los sistemas de modelado que utilizan conceptos orientados a objetos. Puede usarse para modelar distintos tipos de sistemas: de software, de hardware y organizaciones del mundo real. Divide cada proyecto en un número de diagramas que representan las distintas vistas del proyecto y juntos representan la arquitectura del mismo. Permite describir un sistema en diferentes niveles de abstracción, simplificando la complejidad sin perder información, para que los usuarios y desarrolladores comprendan las características de la aplicación.

UML es un lenguaje expresivo, claro y uniforme, que no garantiza el éxito de los proyectos, pero si mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios. (Cornejo)

1.3.3 Herramienta CASE de modelado

CASE es una sigla, que corresponde a las iniciales *Computer Aided Software Engineering*; y en su traducción al español significa Ingeniería de Software Asistido por Computación. El concepto de CASE es muy amplio; y una buena definición genérica sería la de considerar a la Ingeniería de Software Asistida por Computación (CASE), como la aplicación de métodos y técnicas a través de las cuales se les hace útil a las personas comprender las capacidades de las computadoras por medio de programas, de procedimientos y su respectiva documentación. Las herramientas CASE representan una forma que permite modelar los procesos del negocio de las empresas y desarrollar los Sistemas de Información Gerenciales. (Cyta)

1.3.3.1 Rational Rose Enterprise Edition

Es una de las más poderosas herramientas de modelado visual para el análisis y diseño de

sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto se encuentra presente en la concepción y formalización del modelo, en la construcción de los componentes, en la transición a los usuarios y además, en la certificación de las distintas fases.

Es la herramienta CASE que comercializan los desarrolladores de UML, la cual permite el completamiento de una gran parte de los flujos fundamentales del RUP como son el modelado del negocio, captura de requisitos, análisis y diseño, implementación, gestión de configuración y control de cambios.

Características:

- Permite especificar, analizar, diseñar el sistema antes de codificarlo.
- Mantiene la consistencia de los modelos del sistema de software.
- Chequeo de la sintaxis UML.
- Generación y documentación automática.
- Generación de código a partir de los modelos.
- Ingeniería Inversa.

1.3.3.2 Visual Paradigm:

El Visual Paradigm es una herramienta CASE que utiliza UML como lenguaje de modelado, con el uso del acercamiento orientado al objeto. Esta herramienta apoya los estándares más altos de las notaciones de Java y de UML. Está dotada de una buena cantidad de productos o módulos para facilitar el trabajo durante la confección de un software, lo cual garantiza la calidad del producto final.

Características.

- Diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.

- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Disponibilidad en múltiples plataformas.
- Contiene facilidades para redactar Especificaciones de Casos de Uso del Sistema.
- Sincronización entre Diagramas de Entidad Relación y Diagramas de Clases.
- Generación de documentos.
- Integración con distintos Ambientes de Desarrollo Integrados (IDE).

Se concreta la utilización de Visual Paradigm como herramienta CASE por su compatibilidad con UML, amigable en su uso y posee interoperabilidad con otras aplicaciones e integración con distintos Ambientes de Desarrollo Integrado (IDE).

1.3.4 Lenguajes de Programación para el proceso de Desarrollo:

Actualmente existen diferentes lenguajes de programación que han ido surgiendo debido a las tendencias y necesidades que presentan las plataformas. Desde los inicios de la programación fueron surgiendo diferentes demandas por los usuarios y se dieron soluciones mediante lenguajes estáticos. Con el transcurso del tiempo las tecnologías han evolucionado trayendo consigo nuevos problemas lo que nos ha obligado a seguir evolucionando de manera continua en este tema y a ir creando nuevos lenguajes como C, C++,C#, JAVA, entre otros.

1.3.4.1 Java:

Java es un lenguaje de programación que se utiliza para la creación de aplicaciones informáticas, ha sido diseñado con el fin de eliminar las complejidades de otros lenguajes como C y C++.

Presenta como ventaja que es un lenguaje orientado a objetos, es multiplataforma, robusto y combina la flexibilidad, robustez y legibilidad gracias a una revisión de tipos durante la compilación y durante la ejecución. Además, puede tener apuntadores a objetos de un tipo

específico y también se pueden tener apuntadores a objetos de cualquier tipo. Administra automáticamente la memoria porque posee un recolector de basuras el cual determina cuando se puede liberar la memoria ocupada por un objeto y puede ser interpretado y compilado a la vez.

A pesar de lo anterior expuesto, Java presenta algunas deficiencias, como son su velocidad de procesamiento que es lenta debido a que la máquina virtual de java es un intérprete y redunda en una falta de rendimiento con relación a aplicaciones equivalentes escritas en código máquina nativo, el recolector de basura es una sobrecarga adicional al procesador, para su comprensión y entendimiento con facilidad se requiere conocimientos básicos de C

1.3.4.2 C++:

C++ es un potente lenguaje de programación que apareció en 1980, continuando con las ventajas, flexibilidad y eficacia del C. Además de ser un lenguaje de programación que permite programar desde sistemas operativos, compiladores, aplicaciones de bases de datos, procesadores de texto, juegos, etc. En este lenguaje posee puntos a favor, ya que constituye un lenguaje de programación orientado a objetos, didáctico y muy potente en lo que se refiere a creación de sistemas complejos. Actualmente, en C++ se puede compilar y ejecutar código de C, ya viene con librerías para realizar esta labor.

No obstante posee también debilidades tales como:

- Uso de DLLs (librerías dinámicas) muy complejo. Java y .Net han evolucionado estos conceptos manipulando las DLLs mediante los *frameworks* que proveen. En cambio, en C++ el desarrollador debe encargarse de cargar y liberar de memoria estas librerías, y correr los riesgos por el manejo de esta memoria.
- Elaborar un sistema en C++ es como construir un rascacielos: tiene buen soporte y es robusto, pero si existen errores en los pisos inferiores toda la parte superior se desploma.
- Manejo de punteros y memoria respecto a ello. Claro, ésta también es una gran ventaja porque permite un mejor control de la memoria y una buena administración de recursos de

computadora, pero la inexperiencia de los desarrolladores o la pérdida de costumbre con este tipo de variables (sobre todo cuando son dobles o triples punteros, inclusive de mayor orden) puede traer serias consecuencias.

- No es recomendable para desarrollo de páginas Web.(Tala)

1.3.4.3 C#:

C# es el lenguaje estrella de .NET, construido especialmente para adaptarse de manera natural al *framework* y aprovechar al máximo todas sus características. Con él se pretende desplazar a Java del mundo de Internet y hacerse con este importante mercado. C# combina algunas de las características más avanzadas de Java con algunas de las más potentes de C y C++ , acercándose a convertirse en el nuevo lenguaje de Internet y, por supuesto, en el lenguaje nativo para acceder a todos los servicios que en el futuro brindará .NET. C#, al igual que C y C++, permite programar fácilmente a bajo nivel. Gracias a esto, acceder a las características avanzadas de la plataforma sobre la que se trabaja, crear código muy eficiente en aquellos puntos de la aplicación que son críticos y acceder a las interfaces de programación de aplicaciones (APIs) existentes es perfectamente posible.(GeNeura, 2000)

Para el desarrollo de la aplicación, de los lenguajes antes mencionados, aunque con similitudes, se selecciona C#, para esto se tuvo en cuenta que con respecto al C++ presenta una serie de ventajas entre las que se encuentran:

- Eliminación del uso punteros, en C# no se necesitan
- Capacidades de reflexión
- No hay que preocuparse por archivos de cabecera ".h"
- No importa el orden en que hayan sido definidas las clases ni las funciones
- No hay necesidad de declarar funciones y clases antes de definirlas
- No existen las dependencias circulares

- Soporta definición de clases dentro de otras
- No existen funciones, ni variables globales, todo pertenece a una clase
- Todos los valores son inicializados antes de ser usados (automáticamente se inicializan al valor estandarizado, o manualmente se pueden inicializar desde constructores estáticos).

También se tomó en cuenta a Java, que constituye un excelente lenguaje de programación pero el C# presenta ciertas ventajas en cuanto a que:

- El rendimiento es, por lo general, mucho mejor
- CIL (el lenguaje intermedio de .NET) está estandarizado, mientras que los bytecodes de java no lo están
- Soporta bastantes más tipos primitivos (*value types*), incluyendo tipos numéricos sin signo
- Indicadores que permiten acceder a cualquier objeto como si se tratase de un *array*
- Compilación condicional
- Aplicaciones multi-hilo simplificadas
- Soporta la sobrecarga de operadores, que aunque pueden complicar el desarrollo son opcionales y algunas veces muy útiles
- Permite el uso (limitado) de punteros cuando realmente se necesiten, como al acceder a librerías nativas que no se ejecuten sobre la máquina virtual.

  Aunque C# entra en conflicto con la política del centro por ser un software privativo gracias al proyecto Mono de la empresa Ximian, permite una implementación "*open source*" del framework de desarrollo de .NET a través de la inclusión de un compilador de C#, un sistema de ejecución para el "*Common Language Infrastructure*" (CLR), y un conjunto de librerías de clase haciendo posible su utilización en software libre.

1.3.5 Visual Studio .NET

Visual Studio .NET es la herramienta que Microsoft distribuye junto a la plataforma que permite construir y desarrollar aplicaciones .NET. Esta nueva versión no revoluciona la anterior, sino que se limita a añadir una serie de nuevas características y funciones. Es una mezcla de los diferentes entornos que Microsoft utilizaba hasta ahora (Visual Basic 6 IDE, Visual InterDev...).

La principal diferencia respecto a versiones anteriores es que Microsoft utiliza exactamente el mismo entorno para todos los lenguajes incluidos en la plataforma. De hecho, este entorno está creado para poder manejar proyectos que usen más de un lenguaje a la vez, teniendo en cuenta la característica multilenguaje de la plataforma.

1.3.6Sistemas de Gestión de Bases de Datos:

Un Sistema de Gestión de Bases de Datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos. Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan, su propósito general es manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para un buen manejo de los datos.

1.3.6.1 MySQL:

MySQL es un sistema de gestión de bases de datos relacional, licenciado bajo la GPL de la GNU. Su diseño multi-hilo le permite soportar una gran carga de forma muy eficiente. Fue creada por la empresa sueca MySQL AB, que mantiene el copyright del código fuente del servidor SQL, así como también de la marca.

Este gestor es, probablemente, el más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de

programación, además de su fácil instalación y configuración.(MYSQL)

Ventajas:

- Aprovecha la potencia de sistemas multiprocesador gracias a su implementación multi-hilo.
- Soporta gran cantidad de tipos de datos para las columnas.
- Dispone de APIs en gran cantidad de lenguajes (C, C++, Java, PHP).
- Gran portabilidad entre sistemas.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

Desventajas:

- Carece de soporte para transacciones, *rollback's* y subconsultas.
- El hecho de que no maneje la integridad referencial, hace de este gestor una solución pobre para muchos campos de aplicación, sobre todo para aquellos programadores que provienen de otros gestores que sí que poseen esta característica.
- No es viable para su uso con grandes bases de datos a las que se acceda continuamente, ya que no implementa una buena escalabilidad.

1.3.6.2 PostgreSQL:

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (*ORDBMS*) basado en el proyecto POSTGRES de la Universidad de Berkeley. Es una derivación libre (*Open Source*) de este proyecto y utiliza el lenguaje SQL92/SQL99. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido más tarde en otros sistemas de gestión comerciales. Incluye características de la orientación a objetos, como puede ser: la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional. A pesar

de esto, no es un sistema de gestión de bases de datos puramente orientado a objetos.

Ventajas:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos, además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos *array*.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes.
- Permite la declaración de funciones propias, así como la definición de disparadores entre ellas:
    - Vistas.
    - Integridad transaccional.
    - Herencia de tablas.
    - Tipos de datos y operaciones geométricas.
- Soporta el uso de índices, reglas y vistas.
- Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que se le incluye entre los gestores objeto-relacionales.
- Permite la gestión de diferentes usuarios, así como los permisos asignados a cada uno de ellos.

Desventajas:

- Consume gran cantidad de recursos.

- Tiene un límite de 8K por fila, aunque se puede aumentar a 32K, con una disminución considerable del rendimiento.
- Es de 2 a 3 veces más lento que MySQL.

Se define PostgreSQL como Sistema Gestor de Base de Datos por sus características en la orientación a objetos, sus tipos de datos, sus funciones, restricciones, disparadores e integridad transaccional. Porque soporta el uso de índices, reglas y vistas. Además, permite la gestión de diferentes usuarios, así como los permisos asignados a cada uno de ellos, a pesar de consumir más recursos y ser más lento que el MySQL su condición de ser libre lo hace más sustentable.(Pecos)

1.3.7 Propuesta de Solución:

Como propuesta de solución y partiendo del análisis previo realizado sobre las diferentes herramientas y tecnologías que se pueden utilizar para el desarrollo de la aplicación y como premisa para construir un software que se adecue a las necesidades del Centro de Tecnologías de Almacenamiento y Análisis de Datos, se define el uso de PostgreSQL como gestor de base de datos por las características anteriormente mencionadas. Visual Paradigm que es CASE, RUP como metodología de desarrollo, teniendo en cuenta las ventajas y características expuestas anteriormente. Se utiliza UML como lenguaje de modelado, también se puntualiza el uso de C# como lenguaje de programación partiendo de las características que posee como lenguaje, integrado al Visual Studio.NET, el cual es portador de características que ayudan al desarrollador, brindando la posibilidad de que este centre su atención en las funcionalidades esenciales de la aplicación.

1.4 Conclusiones

Todo lo expuesto en este capítulo constituye la base sobre la que se trabajará para desarrollar la aplicación como vía de solución al problema planteado. Se realizó un estudio de los artefactos generados en el proceso de extracción, transformación y carga al almacén de datos permitiendo

el análisis de una serie de características y cualidades que son esenciales en la aplicación a desarrollar. Además, se realizó una investigación de todas las tecnologías, herramientas y metodologías que contribuirán al desarrollo de la aplicación, permitiendo elaborar una propuesta para la selección de aquellas que resulten ser las más acertadas para utilizar en la aplicación.

Capitulo 2: Análisis y Diseño.

2.1 Introducción

En el presente capítulo quedará definido el modelo de dominio, para poder entender el sistema. También se definirán todas las características del sistema para garantizar su funcionamiento, especificándose sus requerimientos, así como los diagramas de casos de usos y las especificaciones textuales de cada uno de ellos. Se describirán también las clases del diseño, mostrando una definición de la arquitectura del sistema, con sus estilos arquitectónicos, el modelo de despliegue y el modelo de Base de Datos.

2.2 Modelo de Dominio y Características del Sistema:

2.2.1Modelo de Dominio

Un Modelo del Dominio es una representación de las clases conceptuales o entidades del mundo real, no de componentes software. Esto ayuda a los usuarios, clientes y desarrolladores e interesados; a utilizar un lenguaje común para poder entender el contexto en que se desarrolla el sistema. Es decir, captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema y no incluyen las responsabilidades de las personas que ejecutan las actividades.

Basado en esta definición, se muestra a continuación el Modelo de Dominio del sistema de gestión de los artefactos resultantes de los procesos de ETL de datos en el almacén de datos.

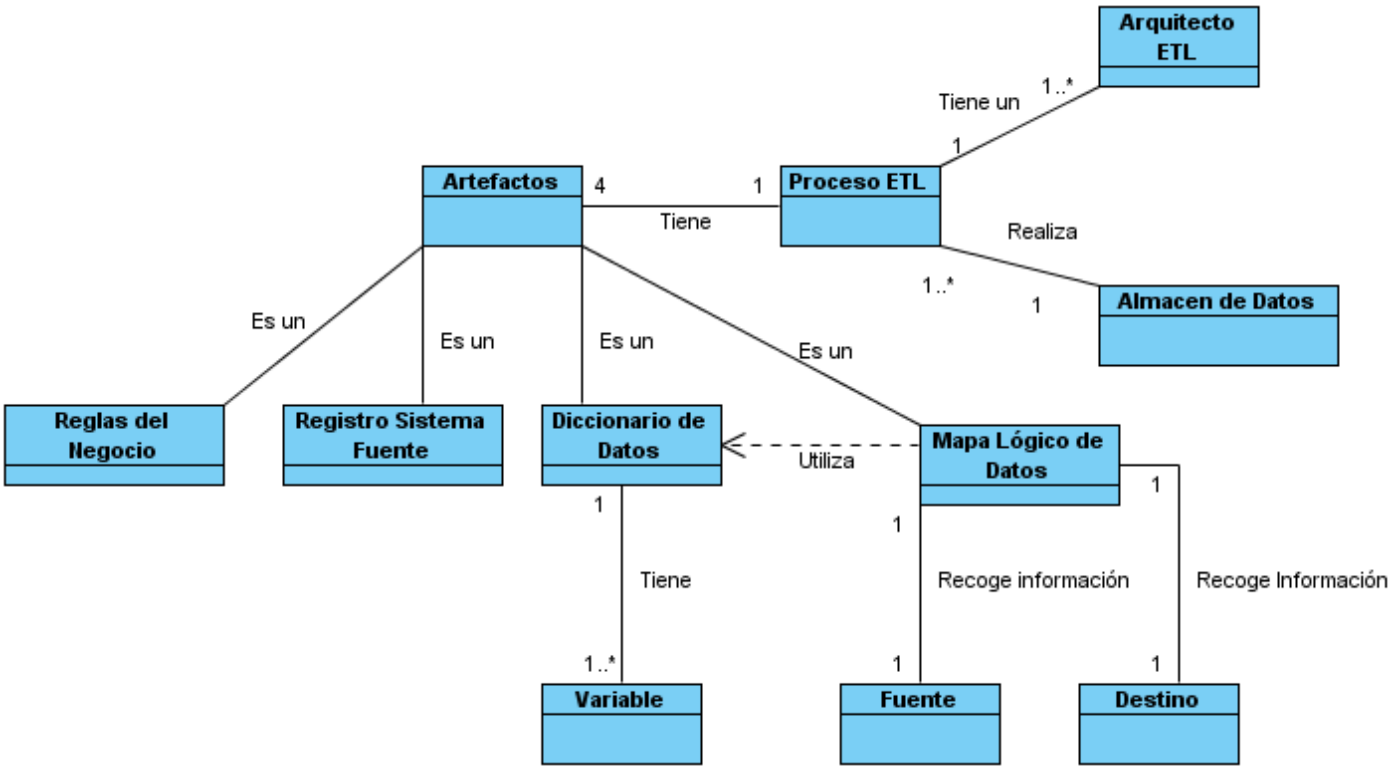

Figura 1 Modelo de Dominio

2.2.1.1 Descripción de los conceptos del Modelo de Dominio

A continuación se realiza una breve descripción del Modelo de Dominio explicando cada uno de los conceptos que se encuentran representados en el modelo anterior:

- **ETL:** Proceso de Extracción, Transformación y Carga de Datos.
- **Arquitecto ETL:** Es el encargado de diseñar tanto los flujos de extracción, transformación y carga como también la arquitectura ETL a utilizar.
- **Artefacto:** Modelo o resumen de información generado como resultado de un proceso.

- **Diccionario de Datos:** Documentación obtenida de la fuente de datos, conceptos de funcionamiento, recoge el significado de cada una de las variables con los posibles valores.
- **Mapa Lógico de los Datos:** Recoge la especificación del mapeo de fuente a destino de los datos, o sea de que base de datos, tabla y columna se debe sacar cada uno de los datos que se necesitan para poblar el almacén.
- **Reglas del Negocio:** Artefacto que recoge las reglas de negocio con los detalles de las transformaciones que deberán ser llevadas a cabo por los procesos de extracción, transformación y carga y a quién se debe reportar los errores encontrados en los datos.
- **Registro Sistema Fuente:** Artefacto que recoge la información referente a cada una de las fuentes de datos, características técnicas como el tipo de Sistema de Administración de Base de Datos con que se administran los sistemas fuentes, el tamaño de la Base de datos (*en Gigabytes*) o aspectos de orden general como a cuál departamento pertenece esta Base de Datos o quiénes son administradores de la misma.
- **Fuente:** Todo lo que se constituya como sistemas orígenes de donde se extrae información ya sean bases de datos, ficheros Excel, u otros.
- **Destino:** Son los sistemas hacia donde se cargan los datos extraídos de la fuente ya sean sistemas integrados o almacenes de datos u otros.
- **Variable:** Una variable es un símbolo que representa un elemento no especificado de un conjunto dado.
- **Almacén de Datos:** Un almacén de datos es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes, para luego procesarla permitiendo su análisis con grandes velocidades de respuesta.

2.2.2 Especificación de los requisitos del software.

Todo sistema de software tiene asociado una serie de requerimientos que deben ser cumplidos durante el ciclo de vida del mismo. Actualmente los requerimientos se clasifican en dos grandes

grupos, requerimientos funcionales y requerimientos no funcionales. Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, para su funcionamiento. En el sistema a implementar se cuenta con los siguientes requerimientos funcionales:

1. Conectar a Orígenes de Datos

   El sistema debe permitir que el usuario se conecte a un origen de datos.

   Entradas: El usuario debe de poder:

   1.1 Definir el Tipo de gestor de BD.

   1.2 Definir el nombre de la BD.

   1.3 Definir el la dirección IP.

   1.4 Definir el puerto.

   1.5 Definir el usuario.

   1.6 Definir la contraseña.

   Salida:

   1.7 Establecimiento de una conexión.

2. Elaborar Registro de Sistema Fuente.

   El sistema debe permitir describir los componentes de un registro de sistema fuente.

   Entradas: El usuario debe de poder

   2.1 Definir el nombre del director técnico.

   2.2 Definir el nombre del director del negocio.

   2.3 Definir el nombre.

   2.4 Definir el teléfono.

   2.5 Definir la dirección de correo electrónico.

   2.6 Definir el área subordinada.

   2.7 Definir el identificador de registro de sistema fuente.

   2.8 Definir el DBMS.

   2.9 Definir el número de usuarios/día.

   2.10 Definir el número de transacciones/día.

   2.11 Definir el nombre de la interfaz.

2.12 Definir el departamento.

2.13 El sistema debe permitir que el usuario defina el SP/SO

2.14 Definir el tamaño de la BD (GB).

2.15 El sistema debe permitir que el usuario defina la prioridad.

2.16 Definir los comentarios.

Salidas:

2.17 El sistema debe guardar todas las entradas a la base de datos

3. Gestionar Artefacto Reglas del negocio.

El sistema debe permitir al usuario:

3.1 Gestionar un listado de reglas.

3.1.1 Cargar un listado de reglas y mostrarlas en una ventana.

Salidas:

3.1.2 Listado de reglas.

3.1.3 Eliminar reglas del listado de reglas.

3.2 Añadir o modificar una regla.

Entrada:

3.2.1 Definir un nombre de regla.

3.2.1 Definir un identificador.

3.2.2 Definir una resolución.

3.2.3 Definir los detalles de la limpieza de los datos.

3.2.4 Definir los detalles de ETL.

3.2.5 Definir un informe de auditoria.

3.2.6 Definir a quien se le envía el reporte.

Salidas:

3.2.7 El sistema debe guardar todas las entradas en la base de datos.

3.2.8 El sistema debe cargar los datos de las reglas previamente definidas.

4. Conformar Diccionario de datos.

El sistema debe permitirle al usuario:

4.1 Cargar el listado de variables

Salidas:

4.1.1 Listado de variable

4.2 Gestionar variables.

4.2.1 Eliminar variables.

4.2.2 El sistema debe permitir adicionar o modificar una variable.

Entradas:

4.2.3 Definir un nombre de variable.

4.2.4 Definir el ID de la variable en la fuente.

4.2.5 Definir la descripción e la variable.

4.2.6 Adicionar los posibles valores que puede tomar la variable.

4.2.7 Filtrar los posibles valores.

4.2.8 Eliminar los posibles valores.

4.2.9 Definir el significado intrínseco que posee la variable.

Salida:

4.2.8 El sistema debe guardar los datos en la BD.

4.3 El sistema debe mostrar los datos de la variable seleccionada.

Salida:

4.3.1 Id de la variable.

4.3.2 Descripción de la variable.

4.3.3 Posibles valores que puede tomar la variable.

4.3.4 Significado intrínseco.

5. Conformar un Mapa lógico de datos.

El sistema debe permitir al el usuario:

5.1 Cargar las variables desde la BD correspondiente y mostrarlas en un listado.

5.2 Cargar valores de la variable seleccionada.

Entrada:

5.2.1 Definir el tipo de SDC para la variable.

Salidas:

5.3 Listado de variable.

5.4 Nombre de la tabla.

5.5 Nombre de la columna.

5.6 Tipo de dato de la variable.

5.7 Tipo de tabla donde se encuentra

5.8 Mostrar los datos de la variable seleccionada en la base de datos fuente:

5.8.1 Nombre de la BD.

5.8.2 Nombre de la tabla.

5.8.3 Nombre de la columna.

5.8.4 Tipo de dato de la variable.

5.9 Detalles de la transformación de la variable.

5.10 El sistema debe guardar los datos.

6. Exportar artefactos.

6.1 El sistema debe permitir al usuario seleccionar que artefacto desea exportar.

6.2 El sistema debe ser capaz de exportar el registro de sistema fuente en formato PDF, WORD, EXCEL, HTML.

Salidas:

6.3 Vista previa del artefacto a exportar.

6.4 Artefacto exportado.

Los requerimientos no funcionales constituyen un conjunto de propiedades o cualidades, que debe presentar el software. Se pueden clasificar en diferentes categorías como por ejemplo: Requerimientos no Funcionales de Software, Restricciones de diseño e Implementación, de Soporte, de Interfaz de Usuario, de Funcionalidad, Portabilidad, Disponibilidad, entre otros.
El sistema que se propone cuenta con los siguientes Requerimientos No Funcionales:

Restricción de Diseño e implementación:

- Servidor de bases de datos PostgreSQL.
- Sistema Operativo Microsoft Windows.
- Tener previamente instalado el framework 2.0 del Visual Studio.

Hardware:

- Mínimo Requerido: RAM 1GB.
- Recomendado: RAM 2 GB.

Eficiencia:

- Debe ser rápida, eficiente tanto en los tiempos de respuesta como en la velocidad de procesamiento de 0.25 segundos y 0.45 segundos respectivamente.

2.2.3Especificación de los Casos de Uso del sistema.

Los casos de uso describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.
En todo sistema se deben definir los casos de usos del sistema así como los actores cada uno.
En el presente documento se describen los actores con cada una de sus responsabilidades en el sistema, así como los casos de usos asociados a ellos:

2.2.3.1 Definición de actores del sistema.

Actor	Descripción
Arquitecto ETL	Es el encargado de gestionar los artefactos generados en el proceso de ETL, además participa en el diseño de la infraestructura del ambiente ETL y es el encargado de resolver problemas técnicos complejos para el equipo de desarrollo.

2.2.3.2 Definición de Casos de Uso del sistema.

Caso de Uso:	CU Elaborar Registro de sistema fuente.
Actores:	Arquitecto ETL.
Resumen:	Se crea o modifica el Registro de sistema fuente.

Caso de Uso:	CU Elaborar Reglas del Negocio.
Actores:	Arquitecto ETL.
Resumen:	Se crean o modifican las Reglas del Negocio.

Caso de Uso:	CU Conectar a origen de Datos.
Actores:	Arquitecto ETL.
Resumen:	Se conecta a un origen de datos especificando los parámetros necesarios para realizar la conexión.

Caso de Uso:	CU Editar Diccionario de datos.

Actores:	Arquitecto ETL.
Resumen:	Se gestiona tanto las variables como sus características para poder conformar el Diccionario de Datos.

Caso de Uso:	CU Editar Mapa lógico de datos.
Actores:	Arquitecto ETL.
Resumen:	Se especifican de qué base de datos, tabla y columna se deben sacar cada uno de los datos para conformar un Mapa lógico.

2.2.3.3 Diagrama de Casos de Uso:

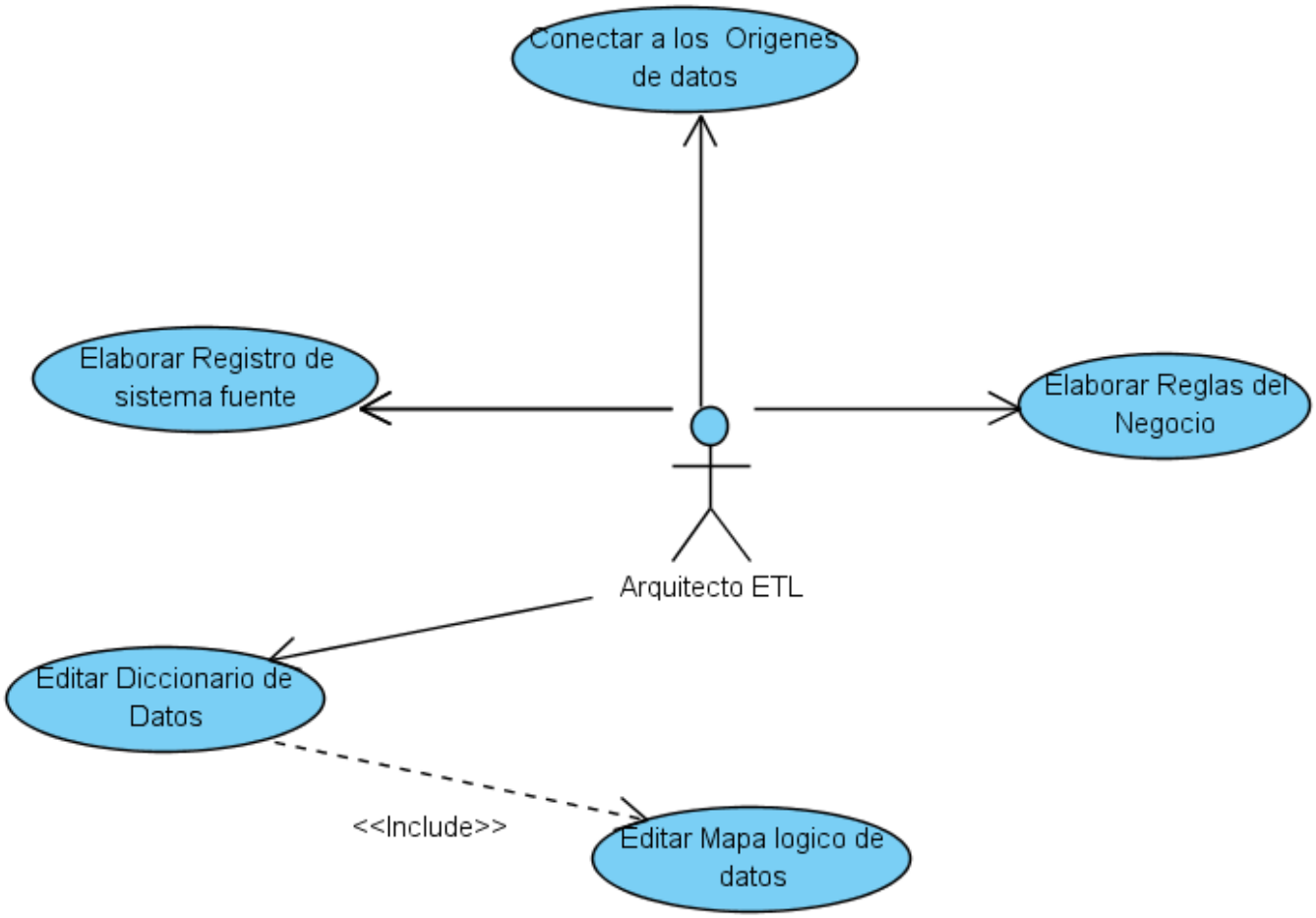

Figura 2: Diagrama de Casos de Uso del Sistema

2.2.3.4 Descripción de los Casos de Uso del sistema.

Para comprender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del Diagrama de casos de uso. El prototipo del sistema que se construye en este punto da una visión de las pantallas diseñadas para cada caso de uso, pero con comportamiento estático, que se presenta al usuario para verificar los requerimientos funcionales.

2.2.3.4.1 CU Elaborar Registro de sistema fuente.

Caso de Uso:	CU Elaborar Registro de sistema fuente.
Actores:	Arquitecto ETL
Resumen:	El caso de uso inicia cuando el actor accede a la aplicación y selecciona la opción de Archivo y luego las de Artefactos y registro de sistemas fuente habilitándose de esta manera un formulario con todas las opciones necesarias para la conformación de un registro de sistema fuente.
Precondiciones:	El arquitecto tiene que haber inicializado la aplicación y estar conectado al origen de datos.
Referencias	Ver requisito funcional 2: Elaborar Registro de Sistema Fuente. Ver requisito funcional 6: Exportar artefactos.
Prioridad	Critico

Flujo Normal de Eventos	
Acción del Actor	Respuesta del Sistema
1 El actor selecciona la opción de Registro de sistema fuente.	2 El sistema muestra una serie de opciones validas: • Descripción de componentes. • Salir. • Exportar Registro de Sistema Fuente.

3 Selecciona la opción Descripción de componentes.	4 Habilita una serie de campos que al ser llenados permiten tener una descripción del registro de sistema fuente:
	• Director Técnico
	  ○ Nombre
	  ○ Correo

	○ Teléfono • Área subordinada • Nombre comercial • DBMS • Usuario/día • Transacciones/día • Director del negocio ○ Nombre ○ Correo ○ Teléfono • Nombre de la interfaz • Departamento • SP/SO • Tamaño de la BD (GB) • Prioridad • Comentarios.
5 Llena los campos y selecciona la opción de guardar.	6 Comprueba y guarda los datos llenados por el actor y muestra el mensaje “Se insertó correctamente.”
7 Selecciona la opción de Exportar registro de sistema fuente.	8 El sistema muestra una ventana con la vista previa del documento y una serie de opciones de formatos a exportar: PDF Excel Word HTML
9 Selecciona el tipo de formato que desea	10 El sistema exporta los datos en el

exportar y después la opción de exportar	formato seleccionado y cierra la ventana de vista previa.
11 Selecciona la opción de salir	12 El sistema cierra la ventana que era objeto de trabajo y vuelve a mostrar la interfaz principal terminando así el caso de uso.

Prototipo de Interfaz

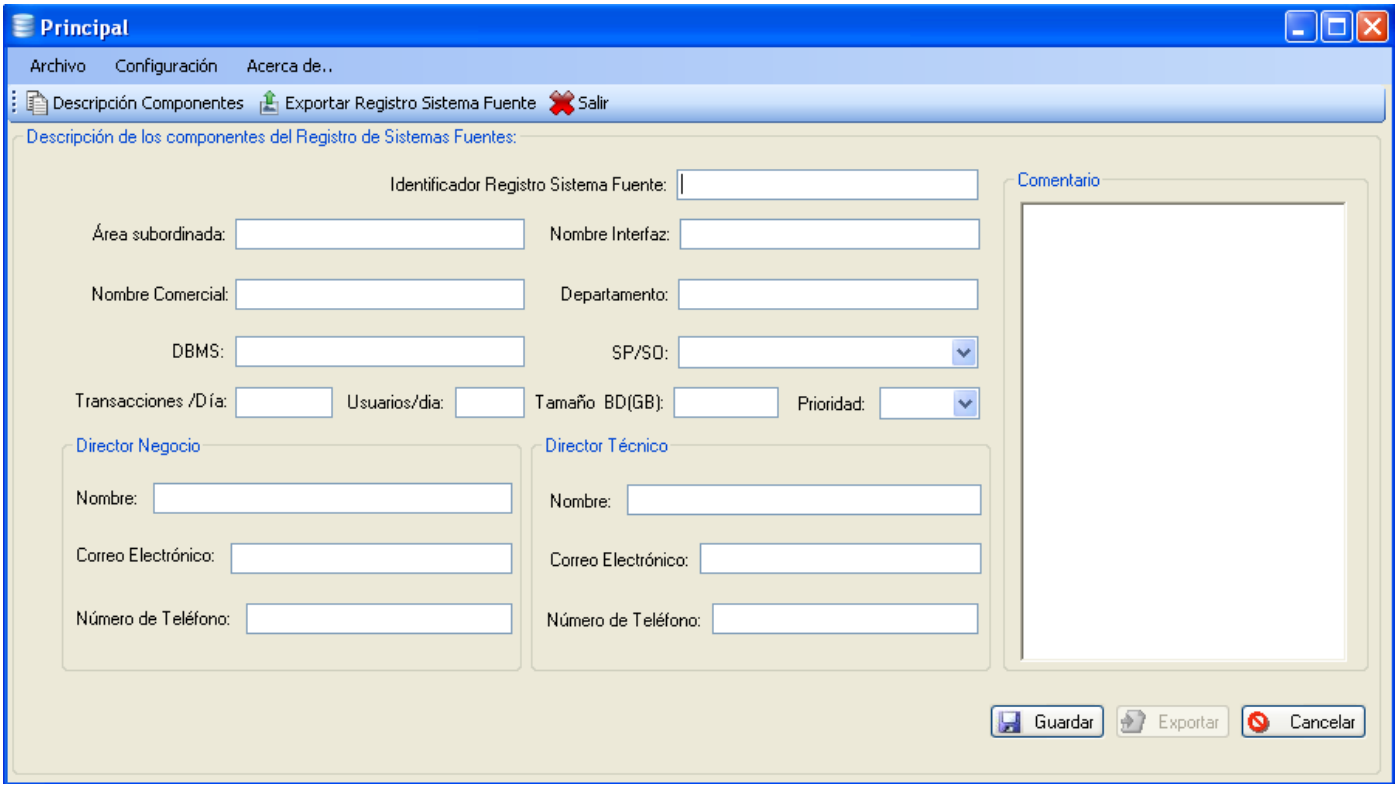

Figura 3: Interfaz Registro Sistema Fuente

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
3.1 Selecciona la opción de Cancelar	3.2Cierra la ventana en la que se esta

	trabajando y vuelve a la interfaz principal.
Flujos Alternos	
Acción del Actor	Respuesta del Sistema
5.1 Llena los campos de manera incorrecta.	5.2 Comprueba los datos y muestra un mensaje de error “Datos incorrectos”
Poscondiciones	Se elabora un nuevo registro de sistema fuente el cual es exportado en formato definido por el usuario.

El resto de las descripciones de los casos de uso pueden ser consultadas en la Plantilla Modelo de los Casos de Uso del Sistema que se encuentra en el expediente de proyecto.

2.3 Análisis del Sistema:

Para una mejor comprensión de esta parte, se mostrarán cada uno de los diagramas generados dentro del análisis, específicamente los diagramas de colaboración, compuestos por clases interfaz, clases controladoras y clases entidades.

Clases interfaz: Modelan la interacción entre el sistema y sus actores.

Clases controladoras: Coordinan la realización de uno o unos casos de uso controlando las actividades de los objetos que implementan la funcionalidad del caso de uso y la interacción entre las clases interfaz y las clases entidades.

Clases entidades: Modelan información que posee larga vida y que es a menudo persistente.

2.3.1 Diagrama de clase de Análisis CU Elaborar Registro Sistema Fuente:

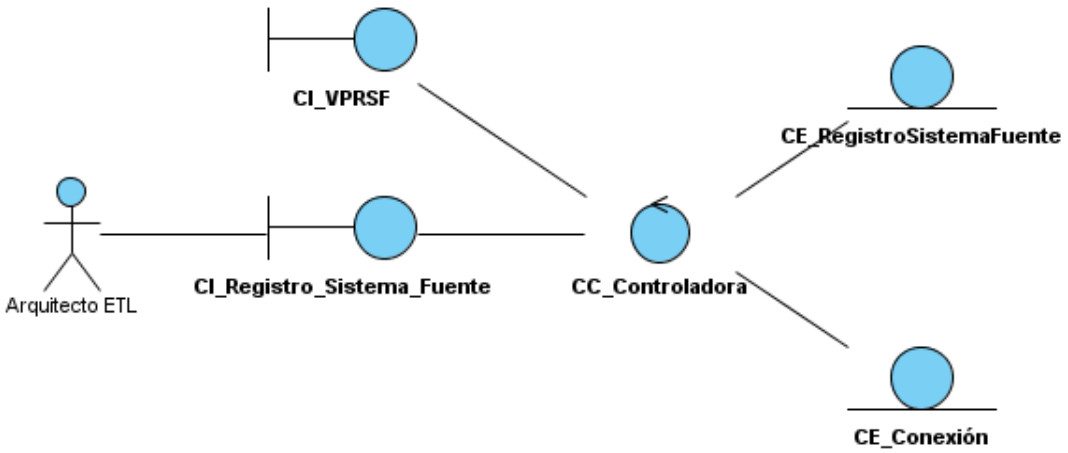

Figura 4: Diagrama de clase de Análisis CU Elaborar Registro Sistema Fuente.

2.3.2 Diagrama de clase de Análisis CU Elaborar Reglas del Negocio:

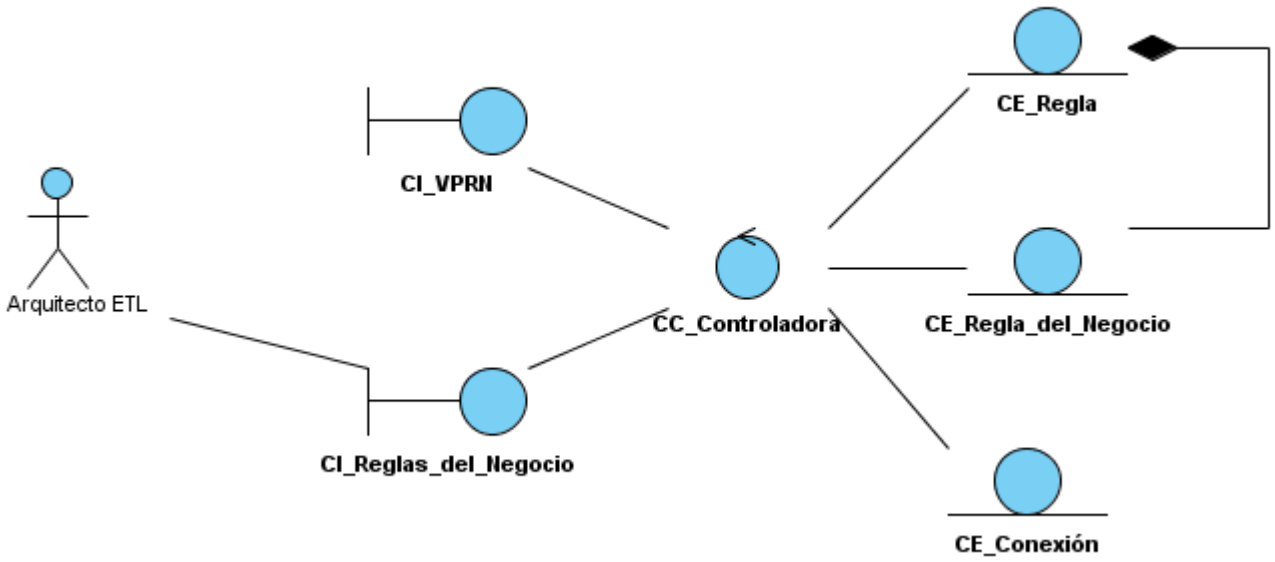

Figura 5: Diagrama de clase de Análisis CU Elaborar Reglas del Negocio.

2.3.4 Diagrama de clase de Análisis CU Editar Diccionario de datos:

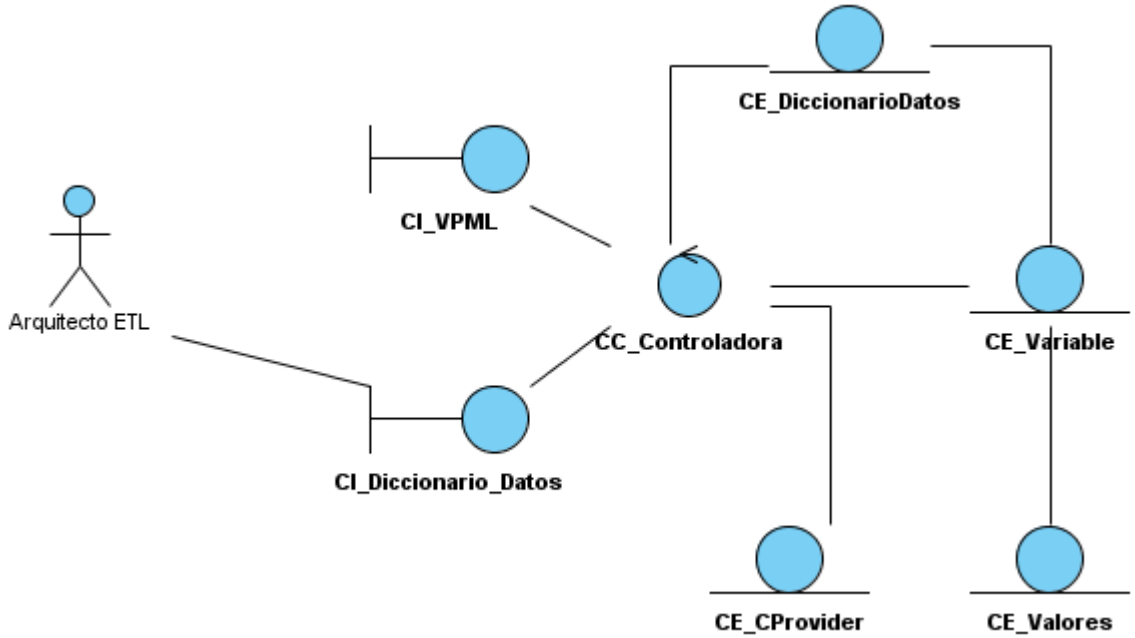

Figura 6: Diagrama de clase de Análisis CU Editar Diccionario de datos.

2.3.5 Diagrama de clase de Análisis CU Editar Mapa lógico de datos:

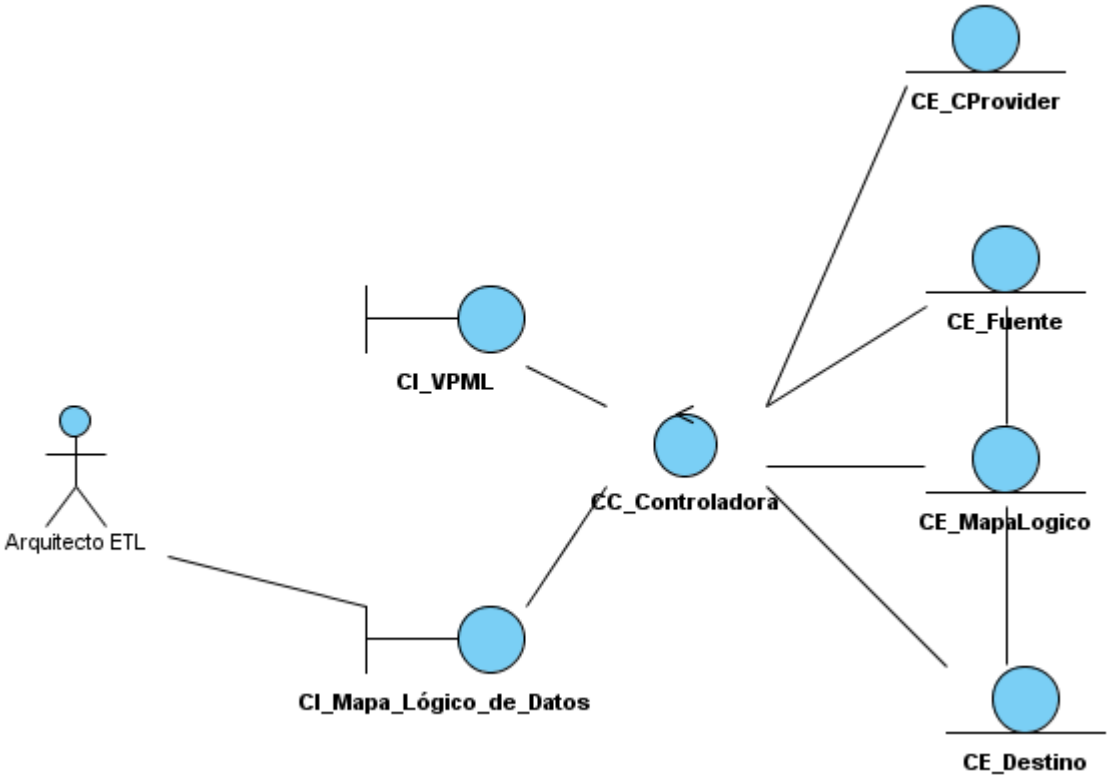

Figura 7: Diagrama de clase de Análisis CU Editar Mapa lógico de datos.

2.3.6 Diagrama de clase de Análisis CU Conectar a los orígenes de Datos:

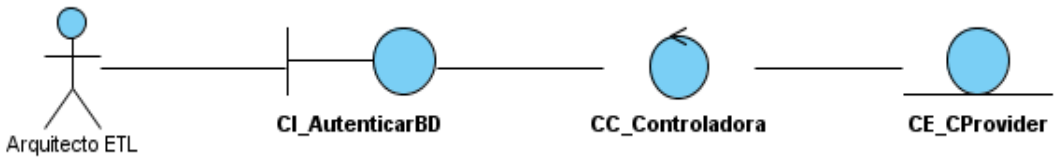

Figura 8: Diagrama de clase de Análisis CU Conectar a los orígenes de Datos.

.2.4.2 Caso de Uso Elaborar Reglas del Negocio:

2.4 Diseño:

El Flujo de Trabajo Análisis y Diseño de RUP tiene el propósito analizar si es posible dar una solución que satisfaga a los requerimientos significativos de la arquitectura, así como traducir los requisitos a una especificación que describa cómo implementar el sistema propuesto.

2.4.1 Diagramas de clases de diseño:

2.4.1.1 CU Elaborar Registro Sistema Fuente:

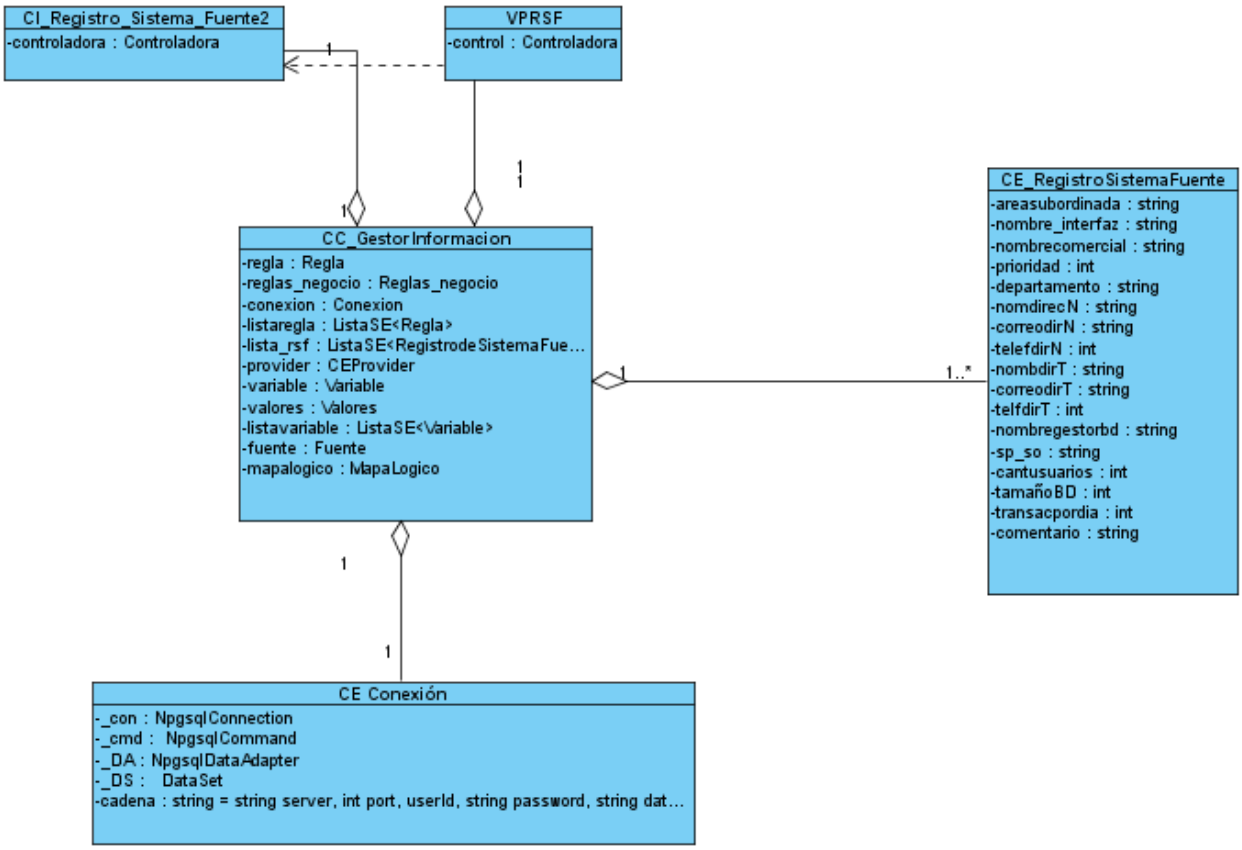

Figura 9: Diagrama de clase de diseño del CU Elaborar Registro de sistema fuente.

2.4.1.2 CU Elaborar Reglas del Negocio:

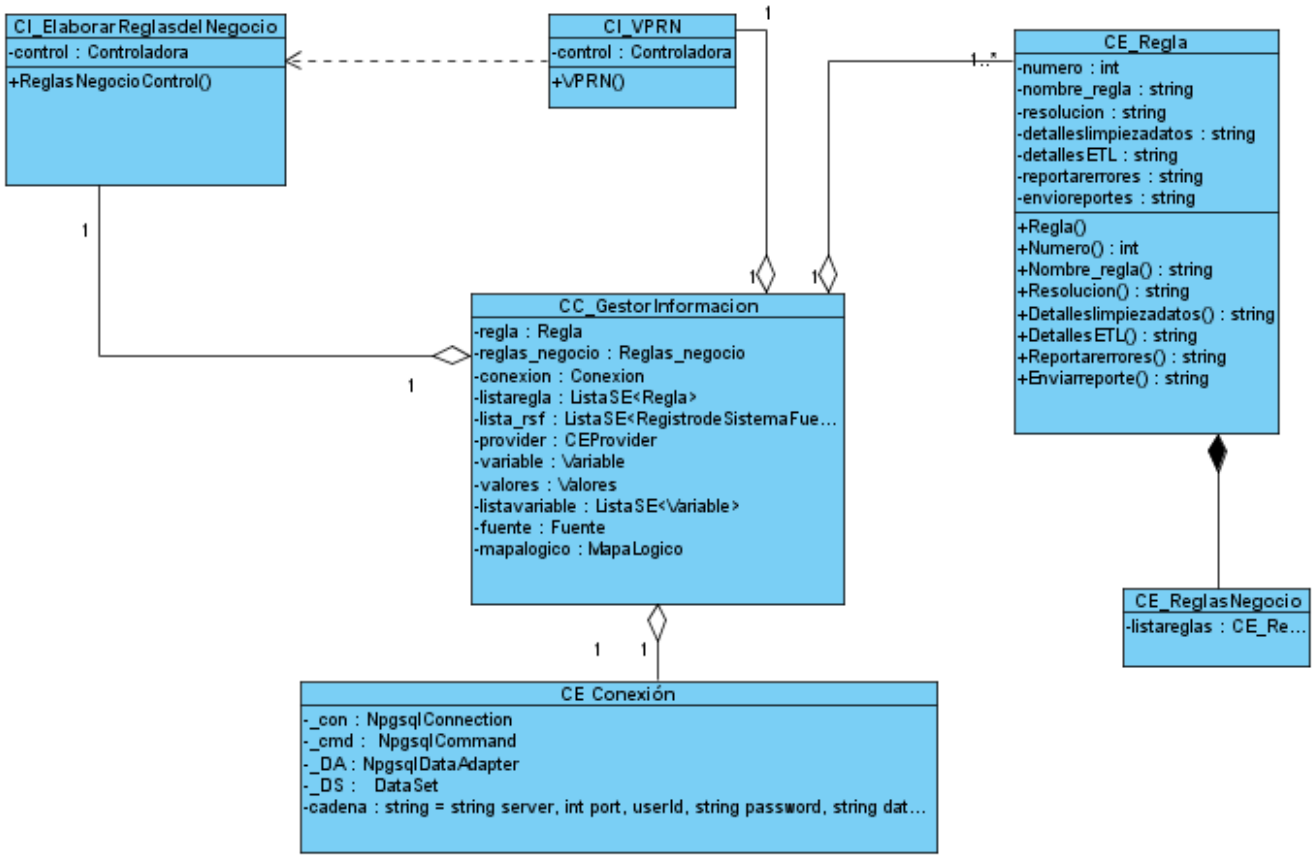

Figura 10: Diagrama de clase de diseño del CU Elaborar Reglas del Negocio.

2.4.1.3 CU Conectar a los orígenes de Datos:

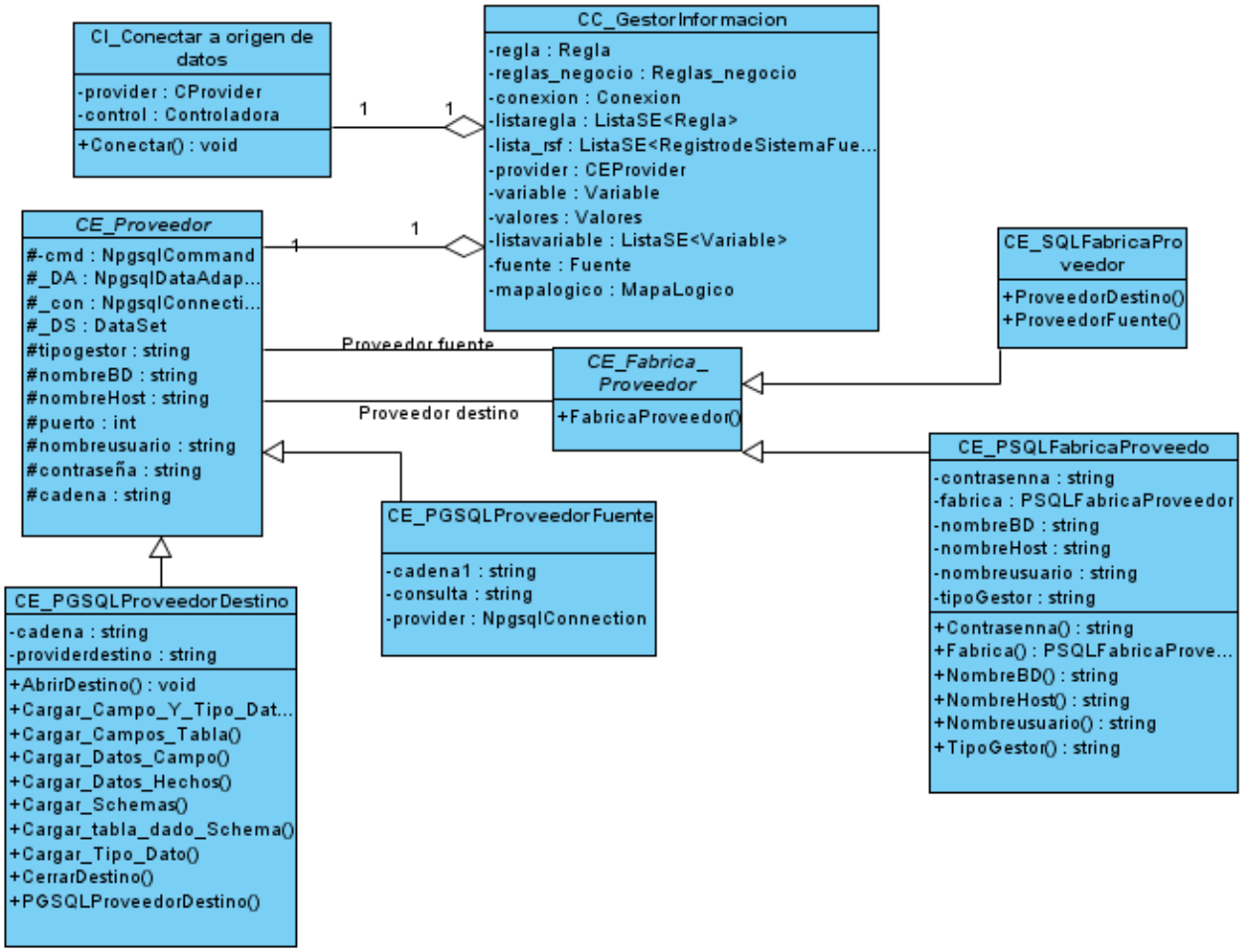

Figura 11: Diagrama de clases de diseño del CU Conectar a los orígenes de Datos.

2.4.1.4 CU Editar Diccionario de datos:

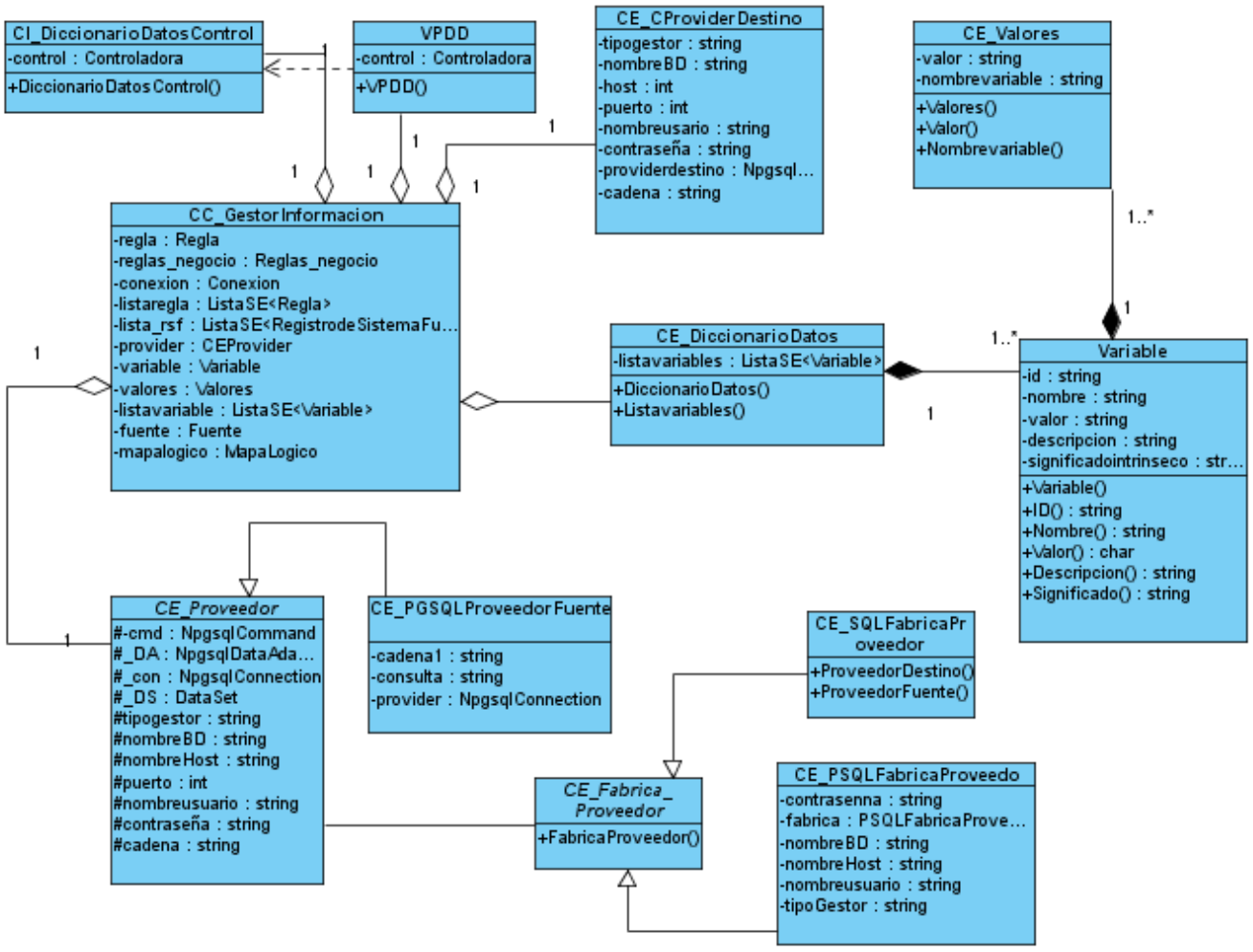

Figura 12: Diagrama de clase de diseño del CU Editar Diccionario de datos.

2.4.1.5 CU Editar Mapa lógico de datos:

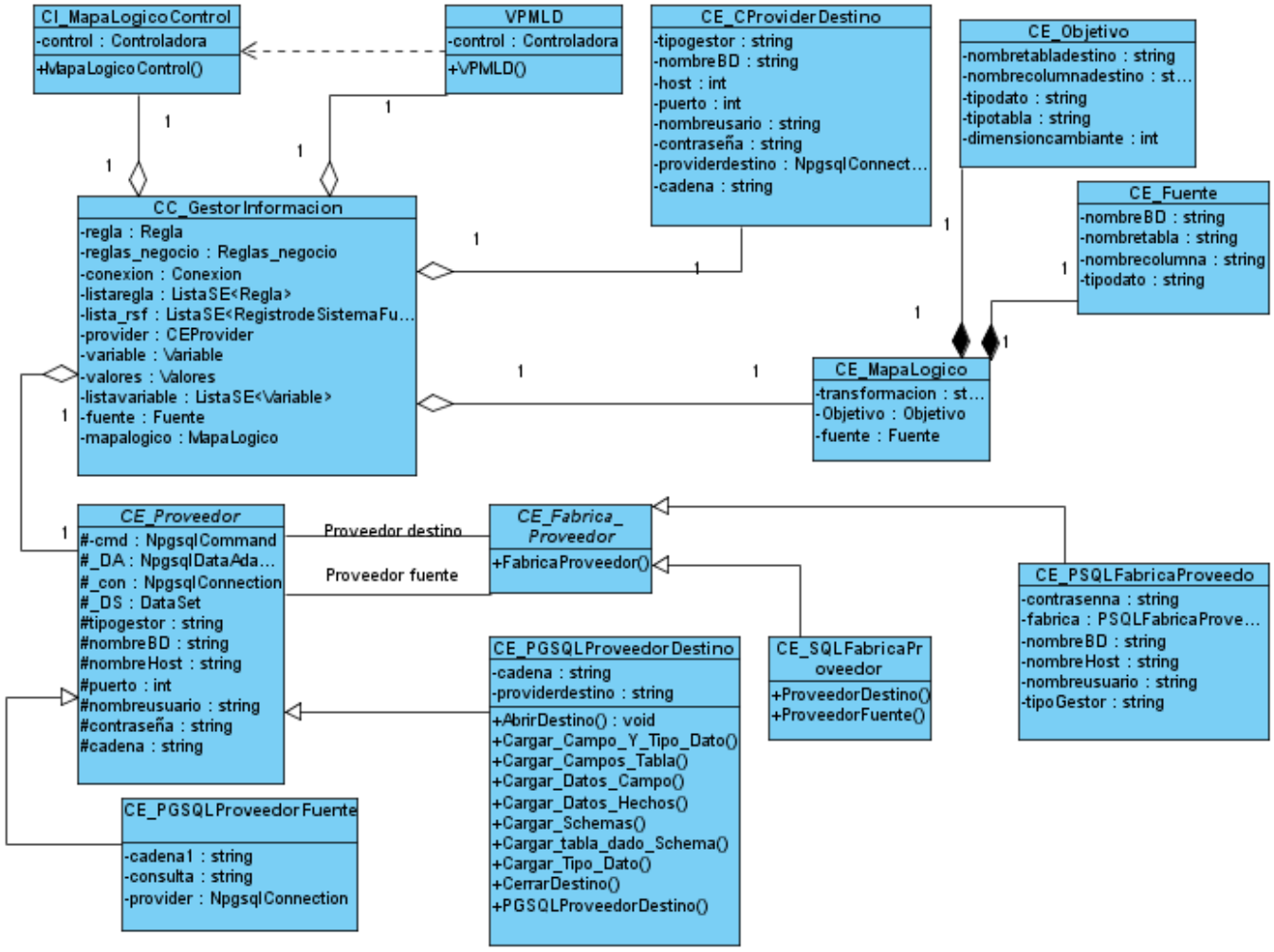

Figura 13: Diagrama de clase de diseño del CU Editar Mapa lógico de datos.

2.4.2Modelo de Despliegue:

En todo sistema software es necesario describir la distribución física de los elementos que lo componen, mostrando la configuración del hardware en la que se desplegará el sistema, identificando los nodos, que pueden ser procesadores o dispositivos, así como los protocolos de comunicación entre cada uno de ellos.

Un procesador es un nodo que tiene capacidad de procesamiento, por ejemplo computadoras y

servidores, mientras que un dispositivo es aquel nodo que no tiene capacidad de procesamiento, entre los que se destacan los siguientes ejemplos Impresora, Scanner, Webcam, Lector de Tarjeta.

El modelo de despliegue del sistema queda constituido de la siguiente manera:

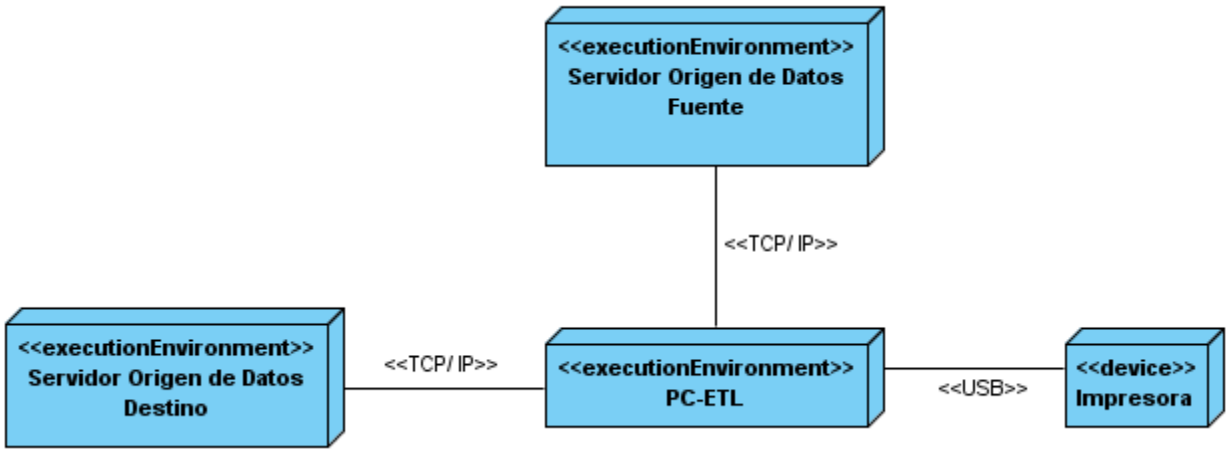

Figura 14: Modelo de despliegue.

2.4.3Diseño de la Base de Datos:

Una Base de Datos es:

- Conjunto de datos lógicamente coherentes, con un cierto significado inherente.
- Una BD se diseña, construye y puebla con datos para propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a distintos usuarios.

En la Base de Datos a utilizar se ha querido organizar de una forma sencilla toda la información que debe estar almacena y relacionada para un mejor funcionamiento del sistema.

A continuación se muestra como quedaría organizada la Base de Datos para un mejor funcionamiento del sistema.

2.4.4Diagrama de Clases Persistentes:

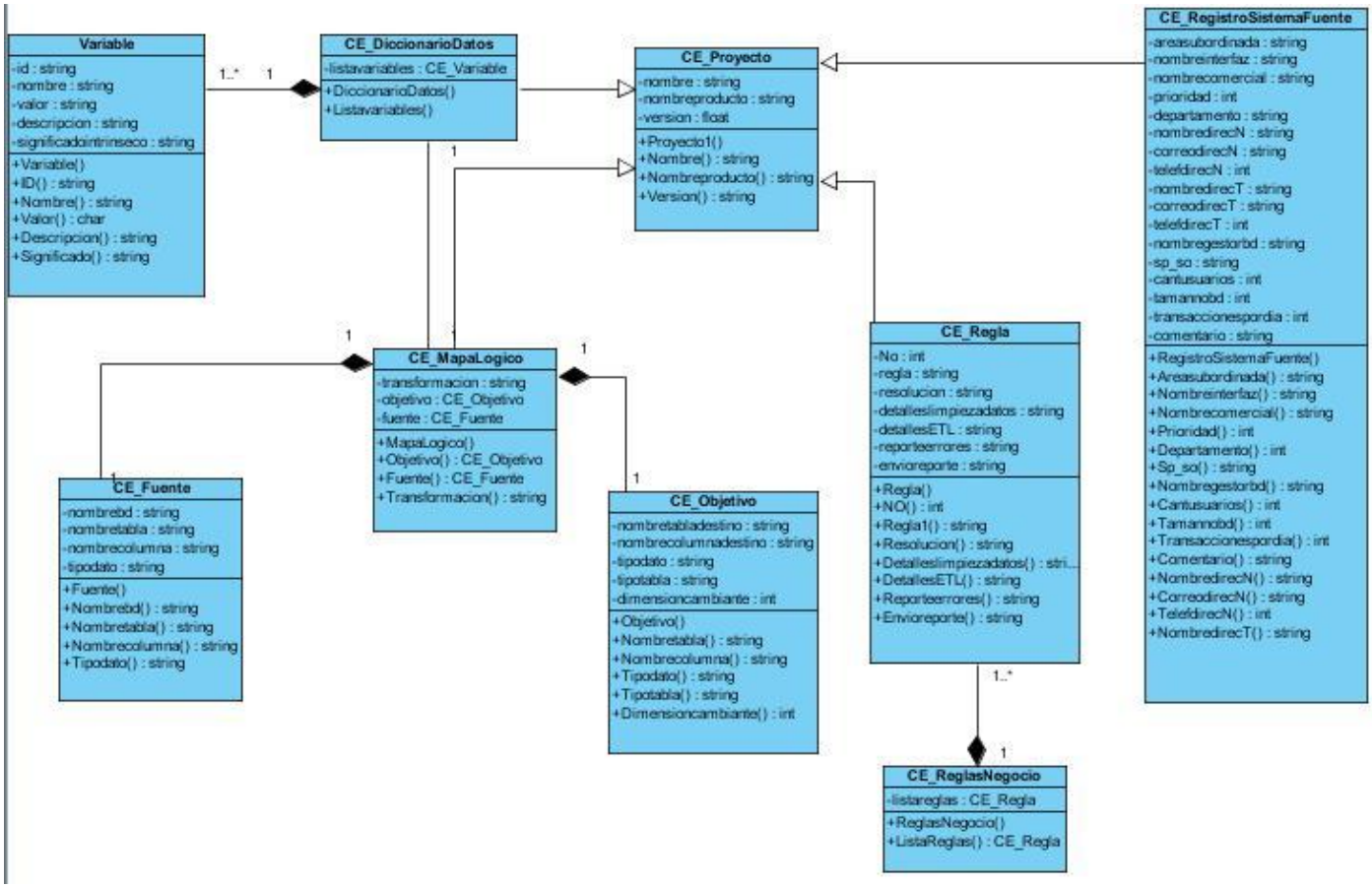

Figura 15: Diagrama de clases persistentes

2.4.5Modelo de Diseño de la Base de Datos:

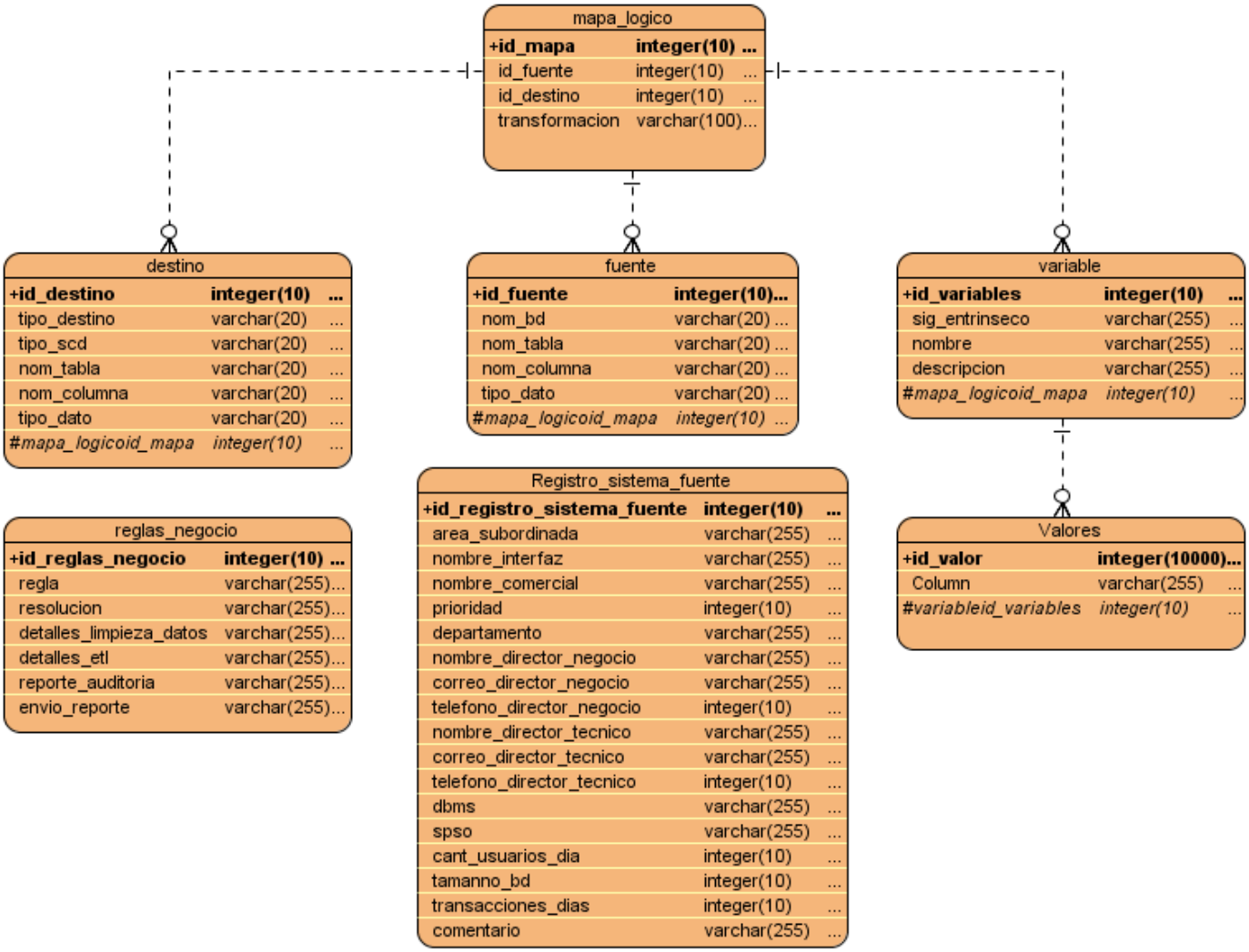

Figura 16 Modelo de diseño de la base de Datos

2.4.6Descripción de las tablas de la Base de Datos:

Nombre: destino		
Descripción: Contiene la información referente a la variable en la base de datos destino.		
Atributo	**Tipo**	**Descripción**

id_destino	integer	Identificador del destino.
tipo_tabla	varchar	Especifica el Tipo de Tabla Destino.
tipo_scd	varchar	Especifica el Tipo de SCD para las tablas Dimensionales que pueden ser 1, 2 ó 3.
nom_tabla	varchar	Especifica el nombre de la Tabla Destino.
nom_columna	varchar	Especifica el nombre de la Columna Destino en la Tabla Destino.
tipo_dato	varchar	Especifica el Tipo de Dato de la Columna Destino en la Tabla Destino

Nombre: fuente		
Descripción: Contiene la información referente a la variable en la base de datos fuente.		
Atributo	**Tipo**	**Descripción**
id_fuente	integer	Identificador de la fuente.
nom_bd	varchar	Especifica el Nombre de la Base de Datos Fuente.
nom_tabla	varchar	Especifica el Nombre de la Tabla Fuente de la Base de Datos Fuente.
nom_columna	varchar	Especifica el Nombre de la Columna Fuente en la Tabla Fuente.
tipo_dato	varchar	Especifica el Tipo de Dato de la Columna Fuente en la Tabla Fuente.

Nombre: mapa_lógico		
Descripción: Contiene los identificadores de la fuente y destino además de la transformación que sufren los datos.		
Atributo	**Tipo**	**Descripción**
id_mapa	integer	Identificador del mapa
id_fuente	integer	Identificador de la fuente
id_destino	integer	Identificador del destino
transformaciones	varchar	Especifica el mecanismo requerido para extraer los Datos Fuentes. Su notación es normalmente en pseudocódigo SQL.

Nombre: registro_sistema_fuente		
Descripción: Contiene los datos relacionados al registro de sistema fuente.		
Atributo	**Tipo**	**Descripción**
Id_registro_sistema_fuente	integer	Identificador del registro sistema fuente
area_subordinada	varchar	Especifica el nombre del proceso que analiza esa Área.
nombre_interfaz	varchar	Especifica el nombre de la Aplicación que soporta el Sistema Fuente.
nombre_comercial	varchar	Especifica el nombre del Sistema Fuente como normalmente es referido por los usuarios empresariales.
prioridad	integer	Especifica el rango o posición ordinal utilizada para determinar futuras fases.
departamento	varchar	Especifica el principal departamento que utiliza la base de datos.
nombre_director_negocio	varchar	Especifica el nombre del director del negocio.
correo_director_negocio	varchar	Especifica el correo del

		director del negocio.
telefono_director_negocio	integer	Especifica el teléfono del director del negocio.
nombre_director_tecnico	varchar	Especifica el nombre del director del técnico.
correo_director_tecnico	varchar	Especifica el correo del director del técnico.
telefono_director_tecnico	integer	Especifica el teléfono del director del técnico.
dbms	varchar	Especifica el nombre del sistema de gestión de base de datos de origen.
spso	varchar	Especifica el nombre físico del servidor donde está la base de datos.
cant_usuarios_dia	integer	Especifica el número de personas operacionales en la organización de los datos.
tamanno_bd	integer	Especifica el tamaño bruto de la fuente de datos.
transacciones_dias	integer	Especifica como la estimación que da una indicación de los requisitos de capacidad para el incremento del proceso de carga.
comentario	varchar	Especifica las observaciones de carácter general

Nombre: reglas_negocio		
Descripción: Contiene lo referente a las reglas del negocio.		
Atributo	**Tipo**	**Descripción**
Id_reglas_negocio	integer	Identificador de la regla del negocio.
regla	varchar	Especifica la Regla del

		Negocio.
resolución	varchar	Propone una vía para dar solución a los datos que no cumplan con esta regla del negocio.
detalles_limpieza_datos	varchar	Exponen detalles de la limpieza de los datos que se ejecutará.
detalles_etl	varchar	Exponen los detalles de la solución ETL dada, con todos los detalles posibles para la solución del problema.
reporte_auditoria	varchar	Incluye los detalles que tendrá el reporte generado por el error que se encontró, una vez se lleve a cabo el proceso ETL.
envio_reporte	varchar	Especifica a quien se envía el reporte.

Nombre: variables		
Descripción: Contiene todo lo referente al Diccionario de Datos.		
Atributo	**Tipo**	**Descripción**
id_variables	integer	Identificador del diccionario de Datos.
valores	varchar	Especifica los posibles valores a tomar por la variable.
sig_entrínseco	varchar	Especifica el significado real de la variable
nombre	varchar	Especifica el nombre de la variable.
descripción	varchar	Describe la variable.

2.6 Arquitectura y Patrones:

2.6.1 Arquitectura y Estilos Arquitectónicos:

El sistema posee una arquitectura de tipo aplicación-escritorio la cual se encuentra compuesta por una capa de presentación en donde se contienen todos los componentes de la interfaz del sistema, otra capa de Negocio la cual contiene la fachada de la aplicación así como las entidades y componentes del negocio y es la capa mediadora de las operaciones entre la capa de presentación y la capa de acceso a datos sin que estas interactúen directamente entre sí. Esta última capa es la encargada de proveer y permitir las operaciones con orígenes de datos con los cuales se desea trabajar y la misma contiene aquellos componentes Npgsql que le permiten realizar dichas operaciones. Las capas en su conjunto formaron la arquitectura de la siguiente manera:

Figura 17 Descripción de la Arquitectura del sistema

Para la conformación de la arquitectura del sistema fue necesario la aplicación de algunos estilos arquitectónicos entre ellos el estilo Arquitectura basada en capas el cual se utiliza cuando se quiere trabajar en un sistema complejo y es necesario manejar esa complejidad dividiendo cada uno de sus elementos en capas donde cada capa se ocupa de un nivel de problema, en el caso de la aplicación esta se fue dividida en tres capas una de presentación, otra de negocio y la tercera de acceso a datos como se describió anteriormente.

Otro de los estilos utilizados fue la Arquitectura orientada a objetos el cual constituye un estilo de programación que se basa en obtener objetos autosuficientes individuales y reusables, donde cada objeto contiene los datos y el comportamiento relevante para sí mismo a partir de la división de las tareas de la aplicación o sistemas viéndose claramente en el objeto de la clase Proveedor el cual permite tener conectado simultáneamente dos orígenes de datos dado a que dicha clase es abstracta y de ella heredan las clases PGSQLProveedorDestino y PGSQLProveedorFuente en las que se define el comportamiento de dicho objeto.

El estilo Cliente-Servidor tambien fue utilizado en la conformacion de la arquitectura dado a que la aplicación necesita de un programa que actue como cliente y que este le haga pedidos a otro programa que constituye el servidor y que en este caso le corresponde al origen de datos con el cual se va a trabajar.

2.6.2 Patrones de Asignación de responsabilidades:

En el desarrollo de la aplicación se utilizaron diferentes patrones para la asignación de responsabilidades (GRASP).

El patrón **Controlador** es aplicable en la solución de la investigación específicamente a la clase controladora ya que la misma funciona como intermediaria entre las interfaces y las clases de la capa del acceso a datos, de forma tal que recibe los datos del usuario y los envía a las distintas clases según el método llamado cumpliendo de esta manera la política de la arquitectura del sistema de mantener la capa de presentación separada de la capa de acceso a datos.

En el diseño de la aplicación se tomo en cuenta la utilización del patrón **Alta cohesión** el cual permite que la información que almacena una clase sea coherente y esté relacionada con la misma para que su uso fuera un éxito se dividió el mismo en un conjunto de cohesiones que poseen características especificas según su utilidad:

- ✓ **Cohesión Coincidente:** Este tipo de cohesión permite realizar múltiples tareas, sin ninguna relación entre ellas, el cual es más visible en la clase GestorInformación.

- ✓ **Cohesión Temporal:** Posibilita llevar a cabo un conjunto de tareas por un módulo, las cuales son ejecutadas al mismo tiempo por lo que al igual que el anterior es más viable su utilización en la clase GestorInformación.

- ✓ **Cohesión de Comunicación:** Las tareas corresponden a una secuencia de pasos propia del sistema y todas afectan a los mismos datos, este patrón es visible en la clase GestorInformación.

En base a la arquitectura del sistema uno de los patrones que reviste gran importancia es el **Bajo acoplamiento** el cual mantiene a las clases lo menos ligadas entre sí que sea posible de forma tal de que si se produce un cambio en alguna de las clases, este repercuta lo menos posible en el resto de las clases potenciando de esta manera la reutilización del código y disminuyendo la dependencia entre clases, lo cual lo convierte en el patrón ideal para el estilo arquitectura basada en capas que forma parte de la arquitectura del sistema.

Otro de los patrones utilizados fue el **Polimorfismo** el cual es aplicable en casi todas las clases del sistema ya que es recomendable que cuando se tenga que llevar a cabo una responsabilidad que dependa del tipo, cuando las alternativas o comportamientos relacionados varíen según el tipo o cuando se desea asignar la responsabilidad para el comportamiento se aplique este patrón.

El patrón **Creador** también es utilizado en el diseño debido a que permite identificar quien va a ser el responsable de la creación o instanciación de nuevos objetos o clases. Este patrón es aplicable en la interfaz principal ya que en la misma se crea un único objeto de la clase GestorInformación (clase controladora) para la restantes interfaces.

Otro de los patrones utilizados fue el **Experto** de información el cual se aplico en el diseño de toda las clases del sistema debido a que el mismo constituye el principio básico de asignación de responsabilidades, indica que la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información para crearlo.

2.6.3 Patrones de Diseño

Algunos de los patrones GOF (Gang of four) también fueron aplicados entre ellos se encuentran:

El patrón **Facade** (Fachada) el cual reúne un conjunto de implementaciones e interfaces dispares en una interfaz común e unificada, este patrón es aplicable en la clase GestorInformación la cual constituye la clase controladora del sistema representando la fachada de este entre las capas de interfaz y acceso a datos Este patrón permite ofrecer un acceso sencillo y se desacopla al máximo el sistema y se puede implementar como una interfaz o clase abstracta sin especificar los detalles de la implementación. Es habitual que se implemente la fachada utilizando el patrón **Singleton** el cual admite exactamente una instancia de una clase donde los objetos necesitan un único punto de acceso global.

Abstrac factory (Fabrica Abstracta): Es un patrón que le permite a un sistema determinar la subclase a partir de la cual se va a instanciar un objeto en tiempo de ejecución. (Rojas, 2009)

Para la implementación de este patrón se paso a su división en 4 componentes:

La **fabrica abstracta** aplicada en el sistema a la clase abstracta FabricaProveedor de la cual heredan las clases PGSQLFabricaProveedor y SQLFabricaProveedor formadoras de la **fabrica concreta**, a su vez el **producto abstracto** lo cosntituye la clase Proveedor de la cual heredan las clases PGSQLProveedorDestino y PGSQLProveedorFuente las cuales representan al **producto concreto** y es la union de todas estas clases la que permite proveer un origen de datos para el funcionamiento de la aplicación.

2.7 Conclusiones:

En este capítulo se trató la parte correspondiente al flujo de RUP Análisis y diseño. El análisis consiste en obtener una visión del sistema que se preocupa de ver qué hace, de modo que sólo se interesa por los requisitos funcionales. Se presentó también el Diseño que viene siendo una especie de refinamiento del Análisis que tiene en cuenta también los requisitos no funcionales. Se presentó la distribución física del hardware del sistema; además se mostraron los diferentes diagramas relacionados con la base de datos. Otro aspecto a destacar es que se exhibieron los patrones de asignación de responsabilidades a utilizar.

Capítulo 3: Implementación y prueba.

3.1 Introducción.

En el presente capitulo se aborda la fase de construcción en particular su actividad más importante, la implementación dándose una explicación de la misma puesto que es la que da paso a la terminación de la aplicación a través de una serie de diagramas de componentes que permiten estructurar el modelo de implementación en términos de subsistemas de implementación . Otra actividad no menos importante perteneciente a una de las últimas fases en el desarrollo de una aplicación son las pruebas de software que garantizan la calidad de la aplicación una vez terminada a través de la detección de errores.

3.2 Modelo de Implementación.

Uno de los artefactos que se identifica en la metodología RUP es el modelo de Modelo de Implementación, este modelo representa cómo, los elementos del modelo de diseño y las clases, se implementan en términos de componentes, ficheros de código fuente, ejecutables, entre otros.
Es importante tener en cuenta que un componente del Modelo de Implementación constituye una parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.).
Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando.
A continuación se muestra el diagrama de componentes de la aplicación desarrollada.

3.2.1 Diagrama de Componentes.

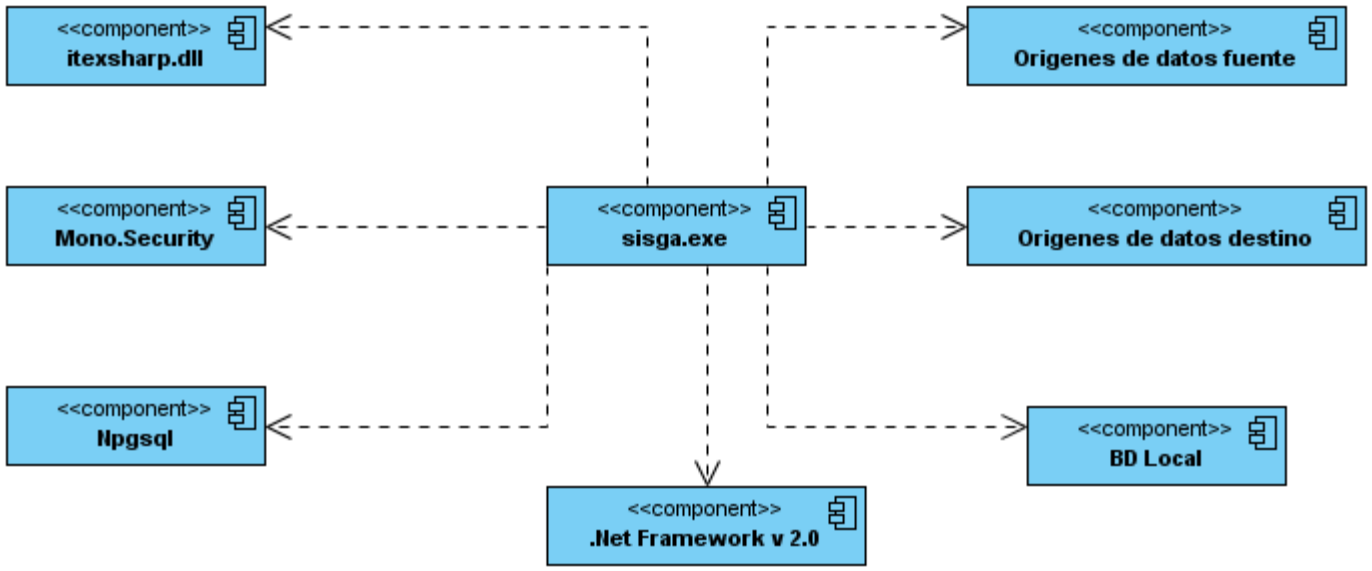

Figura 18 Diagrama de componentes

3.3 Prueba.

Una de las últimas fases de ciclo de vida de desarrollo de software es el flujo de trabajo de pruebas, al cual es necesario dedicarle un importante tiempo pues dicha actividad está encaminada a encontrar errores en el software.

La actividad de probar el sistema y verificar que se encuentra libre de defectos tiene muchos beneficios. Por ejemplo la calidad es una variable muy importante para todo producto y uno de los caminos para garantizarla es siguiendo esta doctrina, además de proporcionar una medida del progreso del trabajo que se despliega.

3.3.1 Pruebas de caja negra

Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos

de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo fundamentalmente del sistema sin tener mucho en cuenta la estructura interna del software.

3.3.2 Casos de pruebas

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada.

Los casos de pruebas deben verificar:

- ✓ Si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos.
- ✓ Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Se realizaron cinco casos de prueba (Uno por cada caso de uso, para un mejor entendimiento remitirse a las planillas de diseño de caso de prueba del expediente de proyecto) y a continuación se procedió a probar cada uno arrojando los siguientes resultados en la primera iteración:

Fueron encontradas diez no conformidades de ellas nueve de funcionalidad y una de interfaz, a su vez las mismas presentaron seis no conformidades de alta prioridad, dos de media y dos recomendaciones como se ilustra en la siguiente figura:

Ticket	Summary	Owner	Type	Priority ▲	Component	Version
1104	CU Conectar a los Origenes de Datos.		Error (Funcionalidad)	Alta	Conectar a Orígenes de Datos	1.0
1106	CU Conectar a los Origenes de Datos.		Error (Funcionalidad)	Alta	Conectar a Orígenes de Datos	1.0
1110	CP Elaborar registro de sistema fuente		Error (Funcionalidad)	Alta	Elaborar Registro de Sistema Fuente	1.0
1111	CP Elaborar registro de sistema fuente		Error (Funcionalidad)	Alta	Elaborar Registro de Sistema Fuente	1.0
1112	CP Elaborar registro de sistema fuente		Error (Funcionalidad)	Alta	Elaborar Registro de Sistema Fuente	1.0
1113	CU Elaborar Reglas del Negocio		Error (Funcionalidad)	Alta	Elaborar Reglas del Negocio	1.0
1105	CP Elaborar registro de sistema fuente		Error (Funcionalidad)	Media	Elaborar Registro de Sistema Fuente	1.0
1109	CP Elaborar registro de sistema fuente		Error (Funcionalidad)	Media	Elaborar Registro de Sistema Fuente	1.0
1107	CU Conectar a los Origenes de Datos.		Error (Interfaz)	Recomendación	Conectar a Orígenes de Datos	1.0
1108	CP Elaborar registro de sistema fuente		Error (Funcionalidad)	Recomendación	Elaborar Registro de Sistema Fuente	1.0

Figura 19 Tabla de No conformidades encontradas en el proceso de prueba.

3.4 Conclusiones

En el capítulo se abarcaron los temas de implementación y prueba. Se definieron cada uno de los componentes asociados a los diagramas de clase de diseño pudiendo apreciarse la trazabilidad directa entre los mismos a través de los diagramas de compontes descritos por casos de uso. También se hace referencia a las pruebas en particular a las pruebas de caja negra y se muestran el diseño de casos de prueba para cada caso de uso.

Conclusiones

Concluida la investigación y el desarrollo del sistema para la gestión de los artefactos ETL y cumplidos a su vez los objetivos planteados así como las tareas de investigación pueden plantearse los resultados alcanzados:

- ✓ Se realizó un estudio detallado sobre los artefactos que se construyen en el proceso de integración de datos en el centro así como se analizaron las relaciones presentes entre los mismos.
- ✓ Se concilió con el grupo de calidad tanto el formato correcto para exportar los artefactos así como la aceptación de estos por parte del expediente de proyecto de almacenes de datos pertenecientes al Centro de Almacenamiento de Datos.
- ✓ Se elaboró el documento correspondiente al desarrollo de la aplicación.
- ✓ Se desarrolló una aplicación que garantiza la gestión de la información los artefactos generados en el proceso ETL.

BIBLIOGRAFÍA

Calvo, J. M. (30 de 11 de 2005). *ACIS*. Recuperado el 3 de 02 de 2010, de ACIS: http://www.acis.org.co/index.php?id=622

Cornejo, J. E. (2007). *DocIRIS*. Retrieved 02 11, 2010, from DocIRIS: http://www.docirs.cl/uml.htm

CYTA. (2006, 01 15). *CYTA.* Retrieved 01 26, 2010, from CYTA: http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm

Doc. Pedro Yobanis Piñero, O. H. (2009). *Metodología para Soluciones de Integración y Análisis de DatosVFinal.* Habana.

IBM-España. (2009). *IBM-España*. Retrieved 01 20, 2010, from http://www-01.ibm.com/software/es/data/infosphere/mdm_server

J.D. Meier, A. H. (2008). *Application Architecture Guide 2.0.* Microsoft Corporation.

Letelier, P. (2006, 01 15). *CyTA*. Retrieved 01 28, 2010, from http://www.cyta.com.ar/ta0502/v5n2a1.htm

Microsoft Corporation. (2010). *msdn*. Retrieved 02 20, 2010, from http://msdn.microsoft.com/es-es/library/aa187916.aspx

Molpeceres, A. (2002, 12 15). *Willydev.* Retrieved 01 26, 2010, from http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF

MYSQL. (2008). *dev.mysql.com*. Retrieved 02 18, 2010, from http://dev.mysql.com/doc/refman/5.0/es/features.html

Pecos, D. (2007). *Netpecos*. Retrieved 02 9, 2010, from http://www.netpecos.org/docs/mysql_postgres/index.html

Rojas, M. J. (2009). *Instituto Tecnologico de Morelia.* Retrieved 05 25, 2010, from Patrones de Diseño: http://antares.itmorelia.edu.mx/~jcolivar/courses/dp07a/patrones.pdf

GLOSARIO DE TÉRMINOS

Homogeneización: Proceso a través del cual se provoca que una mezcla presente las mismas propiedades en todas las sustancias.

XML (*Extensible Markup Languaje*): Lenguaje de marcas extensible, constituye unos metalenguajes extensibles de etiquetas.

Ingeniería Inversa: Proceso de construir especificaciones de un mayor nivel de abstracción partiendo del código fuente de un sistema software o cualquier otro producto.

Dependencia circular: Es una situación que puede ocurrir en lenguajes de programación en donde la definición de un objeto incluye el objeto sí mismo.

Npgsql: Librería del lenguaje C# que permite la interacción con un origen de datos.

Coherente: Cuando se presenta coherencia o relación lógica y adecuada entre las partes que lo forman.

No conformidad: Constituye le no cumpliemento de un requisito o sea cuando se presentan errores.

Trazabilidad: La propiedad del resultado de una medida o del valor de un estándar donde este pueda estar relacionado con referencias especificadas, usualmente estándares nacionales o internacionales, a través de una cadena continúa de comparaciones todas con incertidumbres especificadas.