Universidad de las Ciencias Informáticas

Facultad 3

Título: “Diseño del módulo para la Administración Contable y Financiera de los Registros y Notarías de la República Bolivariana de Venezuela”

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autores: Juan Carlos Montané Izaguirre y Henrik Pestano Pino

Tutor: Ing. Marbys Marante Valdivia

Consultante: Dr. Pedro Yobanis Piñero Pérez

Mayo 2007

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos a la Infraestructura Productiva de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Juan Carlos Montané Izaguirre **Henrik Pestano Pino** **Ing. Marbys Marante Valdivia**

_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado **Diseño del modulo para la Administración Contable y Financiera de los Registros y Notarías de Venezuela**, fue realizado en La Universidad de las Ciencias Informáticas (UCI). Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____
_____
_____
_____
_____
_____
_____

Como resultado de la implantación de este trabajo se reportará un efecto económico considerable.
Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____.

_____ _____
Representante de la entidad Cargo

_____ _____
Firma Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Sobre el Trabajo de Diploma Presentado para Optar por el Título de Ingeniero Informático

Título: **Diseño del módulo para la Administración Contable y Financiera de los Registros y Notarías de Venezuela**

Autores: **Juan Carlos Montane Izaguirre.**

Henrik Pestano Pino

El tutor del presente Trabajo de Diploma considera que durante su ejecución los diplomantes han mostrado total independencia, creatividad y responsabilidad en la solución a situaciones inéditas para ellos.

Han aplicado conocimientos adquiridos en su formación, basados en las experiencias adquiridas durante el desarrollo del proyecto Registros y Notarías demostrando dominio e integralidad en el manejo de técnicas de investigación y análisis de problemáticas.

El documento presenta calidad, lo han desarrollado en menos tiempo del establecido por estar parte de este en el cumplimiento de una misión colaborativa, en la república bolivariana de Venezuela, además conjugando el desarrollo de este trabajo de diploma con las tareas de desarrollo de nuevos módulos encomendados desde su regreso, así como las actividades propias de la Universidad.

El trabajo desarrollado es de utilidad y cumple con los requisitos necesarios y objetivos trazados. Las experiencias reflejadas serán de gran importancia para la construcción del polo productivo: Sistemas Legales.

Por todo lo anteriormente expresado se puede plantear que los diplomantes han cumplido los objetivos propuestos, por lo que le propongo el título de Ingeniero Informático, con una calificación máxima de 5 puntos.

_____ _____

Ing. Marbys Marante Valdivia Fecha

“Al poseedor de las riquezas no le hace dichoso el tenerlas, sino el gastarlas, y no el gastarlas como quiera, sino el saberlas gastar.”

Miguel de Cervantes Saavedra

Dedicatoria

***A**ida, la luz que ilumina mis días… el mapa que le da sentido a mi vida… motor que le da fuerzas a mi alma... y sin dudas mi mayor inspiración.*

A mis padres, quienes supieron forjar mi espíritu sincero y honrado a pesar de todo…

A mi hermana la primera persona en la que pienso cuando tengo algo que ofrecer…

A Belkita el punto de apoyo que nunca me ha faltado en los momentos difíciles…

A todos mis amigos, porque sin ellos no seriamos nada, en especial a JuanK, Pocholo, René, Danieluco y Yosvany, que han soportado bien de cerca mi testarudez y mi espíritu de obstinada negación…

A la divina coincidencia que me hizo recorrer el largo trayecto para llegar hasta aquí y a todos los que de una forma u otra tuvieron que ver con ello.

Henrik Pestano Pino

***A** mis padres: A mi mamá que ha inspirado en mí, la fuerza y el deseo de hacerme alguien en la vida. A mi papá que cada día se ha preocupado porque no me falte lo imprescindible para lograr el sueño de ser un buen profesional, de veras los quiero con todo mi corazón.*

A mis segundos padres: A Esperanza y Robert que me han dado los mejores consejos de mi vida, me han soportado todas mis malacrianzas y me han ayudado a tomar decisiones importantes en la vida.

A mi hermana por apoyarme en cada momento.

Al Pito y a mi hermano Juan Javier también le dedico esta tesis de graduación como ejemplo de esfuerzo y sacrificio.

A mi tarequito, por guiarme, aconsejarme y darme las fuerzas y el motivo de luchar en la vida y saber que no lo estoy haciendo en vano, gracias mi amor por quererme como lo haces cada día, te amo mucho.

A mis amigos: en especial René y Pocholo que han sido los amigos con los que un día soñé, han sido mis hermanos, gracias, sin ustedes mi camino hasta aquí hubiera sido más difícil.

Juan Carlos M.

Agradecimientos

A nuestra Revolución Cubana por darnos la posibilidad de graduarnos en la primera universidad de la Batalla de Ideas.

A todos los profes que contribuyeron a nuestra formación como ingeniero, en especial a Pedro Piñero.

A nuestros compañeros de aula que son los que más nos han enseñado, en especial a: René, Pocholo, Danieluco.

A Marbys, Yosvany e Ismary por las ideas que nos aportaron y sacrificarse junto con nosotros para que pudiera salir este trabajo.

Resumen

El presente trabajo hace un análisis crítico de cómo eliminar los problemas de los procesos económicos en los Registros Inmobiliarios y Mercantiles del Ministerio del Interior y Justicia de Venezuela a partir del diseño del sistema software de Administración Contable y Financiera como parte del sistema que se propone desplegar en todos los registros de este tipo en el país, posibilitando la estandarización y control de los precios de los servicios que estas oficinas prestan a la sociedad venezolana, así como el ingreso a la banca del estado. Los sistemas informáticos son cada vez más complejos debido al desarrollo del mundo actual, se impone la necesidad de hacer diseños de sistemas cada vez más flexibles y fáciles de mantener de acuerdo con los cambios continuos que vive la humanidad. El diseño que se propone se desarrolló sobre esta base. Para ello se hace un estudio de las técnicas del desarrollo del diseño del software en algunas de las metodologías de desarrollo, así como los patrones de diseño que hoy nos permiten resolver problemas que suelen encontrarse en repetidas ocasiones.

Palabras Clave:

Contabilidad, Finanzas, Modelo de Diseño, Requisitos, Capas, Subsistemas, Clases, Realizaciones, Métricas, Trazabilidad.

Índice

Figuras

Tablas

Introducción

Antecedentes y problemática existente

En el Ministerio del Poder Popular para Relaciones Interiores y Justicia de La República Bolivariana de Venezuela existe una dirección, la cual se encarga de gestionar, organizar y dirigir los registros mercantiles e inmobiliarios de todo el país. La función de estos registros es prestar servicio de procesos legales a la población venezolana, a través de la realización de trámites. Actualmente existen tarifas según conceptos de pago y recaudo pero dada la descentralización de los procesos y la independencia de estos registros, no existe un control por parte de la dirección de Registros y Notarías de cuanto se ingresa en cada uno de ellos, cuanto se gasta y cuanto devengan los trabajadores según su nómina y cargo que ocupan. El pueblo es el que sufre las alteraciones de los precios en los trámites a realizar y el ministerio no ingresa el dinero correspondiente a estos servicios. Muchos funcionarios de estos registros se enriquecen ya que el dinero queda en sus bolsillos y no se ingresa a las cuentas del Ministerio Popular del Interior y Justicia de Venezuela. No se tiene la relación de trámite contra ingreso que existe en cada uno de los registros.

Dicho ministerio decidió crear un Servicio Autónomo, que se encargue de planificar el presupuesto, definir los proveedores, el plan único de cuenta, los conceptos de gasto, los bancos donde los registros pueden crear sus cuentas, así como todas las normativas necesarias para el propio proceso registral. La realización de estos procesos de forma manual se convierte en una tarea muy engorrosa y complicada. Con este trabajo se pretende realizar el Diseño de uno de los módulos que conforman este paquete de gestión, específicamente lo relacionado con la gestión contable y financiera.

Como antecedente más importante vale destacar lo realizado en el marco de la modernización de las finanzas públicas realizada en el año 2000 en la República Bolivariana de Venezuela. En ese momento se comenzó a trabajar en una herramienta informática llamada Sigecof (Sistema Integrado de Gestión y Control de las Finanzas Públicas), que tuvo asociado todo un proceso de cambios en normas, reglamentos e incluso en Leyes. La principal función de este sistema fue que las finanzas públicas se ejecutaran bajo la estructura de control que establecen las leyes, en general controla más de cien artículos de la Ley y más de cuatrocientos tomando en cuenta los Reglamentos establecidos.[1]

Las ideas básicas y los principales conceptos empleados en el diseño de este sistema, fueron aportadas por varios especialistas: Miguel Cabrera González licenciado en Economía y ganador del Premio Nacional de Economía 2005 en la especialidad de Contabilidad, el Ingeniero Informático Yosvany Márquez Ruiz y la Ingeniera Industrial Ismary Lorenzo López. Todos ellos formaron parte de un sistema cubano de gestión económica y contable, conocido como Versat-Sarasola. El cual comenzó a construirse en 1998, dentro del Ministerio del Azúcar (MINAZ). Este sistema económico integrado esta certificado, por el Ministerio de Finanzas y Precios de nuestro país, tiene varios módulos que incluyen contabilidad general, análisis económico empresarial y control de activos fijos, finanzas y cajas, planificación y presupuestos, control de inventarios, paquete de gestión, contratación y facturación, entre otros.

Formulación del problema

¿Cómo elaborar el diseño de un sistema de gestión que automatice los procesos económicos en los Registros Inmobiliarios y Mercantiles del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela?

Objeto de estudio

Procesos económicos automatizables de los Registros Inmobiliarios y Mercantiles del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela.

Campo de acción

Diseño del módulo de Administración Contable para la automatización de los procesos económicos y financieros de los Registros Inmobiliarios y Mercantiles del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela.

Hipótesis

Si se diseña una solución de software para la gestión de los procesos contables en los Registros y Notarias del Ministerio del Poder Popular para Relaciones Interiores y Justicia de Venezuela, entonces se logrará un eficiente análisis económico de sus estados financieros, permitiendo conocer y controlar los recursos económicos actuales y a partir de estos facilitar la toma de decisiones.

Objetivos

El objetivo general de este trabajo es realizar el diseño de un sistema capaz de homogenizar y automatizar la gestión económica en los Registros Inmobiliarios y Mercantiles del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela, garantizando el correcto cumplimiento de lo estipulado para los procesos Contables y Financieros; de conformidad con las disposiciones legales que los regulan.

Partiendo de este objetivo general se derivan los siguientes objetivos específicos:

Objetivo 1. Analizar los casos de usos de sistema y los requisitos asociados, así como las reglas del negocio a tener en cuenta para la construcción del diseño.

Objetivo 2. Desarrollar la actividad de diseño del subsistema Gestión Contable.

Objetivo 3. Desarrollar la actividad de diseño del subsistema de Finanzas.

Objetivo 4. Desarrollar la actividad de diseño del subsistema de Reglas Contables.

Objetivo 5. Evaluar la calidad del diseño obtenido a través de la aplicación de métricas.

Metodología de la Investigación

Para realizar la parte investigativa de este trabajo se siguieron los siguientes pasos:

1. Recopilar información necesaria para el proyecto.
    - Análisis de la bibliografía disponible.
    - Consulta de libros con relación al tema.
    - Búsqueda de textos disponibles en Internet.
    - Entrevistas con los funcionarios y especialistas del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela.
2. Seleccionar y organizar datos necesarios para el proyecto.

Método utilizado para recopilar información

Para recopilar la información se utilizaron:

- Análisis Bibliográfico.
- Entrevistas.

Revisión bibliográfica

Se basó fundamentalmente en consultar un conjunto de fuentes de información referidas al tema, como fueron libros, artículos, revistas, publicaciones, boletines y toda una variedad de materiales escritos y digitales, además se localizaron lecturas especializadas sobre el tema para documentar la base teórica de este trabajo. Toda esta fuente de información fue debidamente referenciada para ulteriores consultas, pues la misma servirá a aquellas personas que deseen estudiar más a fondo el resto de los capítulos de este trabajo, aunque es bueno destacar que no representa en manera alguna una bibliografía profunda sobre el tema.

Técnica de Entrevista

Mediante esta forma específica de interacción social, se logró obtener información y detalles del sistema que deseaban obtener finalmente los clientes. A partir de las repuestas y aclaraciones que brindaron los especialistas del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela se obtuvieron las informaciones que se buscaban. Gracias a que esta técnica permite reunir datos primarios, generalmente con un carácter más práctico y menos teórico, se logró definir todos los detalles requeridos para el diseño del sistema.

Método utilizado para seleccionar y organizar los datos

La selección de información de la bibliografía consultada se realizó considerando a aquellas que tuviesen relación directa con el problema en cuestión. Esta se organizó utilizando el método Inductivo – deductivo, partiendo de la idea de comprender los problemas más generalizadores y a partir de estos resultados interpretar conceptos con menor nivel de generalización.

Tareas de la Investigación

1. Estudio de los principios económicos que rigen el negocio de la contabilidad estatal en los Registros Inmobiliarios y Mercantiles del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela.
2. Estudio del estado del arte del las principales tendencia del diseño software.
3. Estudio del estado del arte de los principales software de gestión empresarial.

4. Investigación sobre una propuesta de diseño, acorde con las condiciones de los Registros Inmobiliarios y Mercantiles del Ministro del Poder Popular para Relaciones Interiores y Justicia de Venezuela, que automatice los procesos contables y financieros a un costo razonable y en los plazos establecidos.
5. Selección y adaptación de la metodología para el diseño de sistema atendiendo a la tecnología seleccionada y al problema específico a resolver.
6. Aplicación de patrones de arquitectura y diseño.

Resultados Esperados

- Definición de la estrategia para el diseño de un sistema de gestión económica.
- Identificación de los elementos y mecanismo de diseño.
- Fundamentación del uso de patrones.
- Construcción de los artefactos inherentes al diseño.
    - Análisis de los Requisitos.
        - Principales Casos de Uso del Sistema y requisitos asociados.
    - Modelo de Diseño.
        - Capas y Subsistemas del Diseño.
        - Clases del Diseño.
        - Realización de Casos de Uso de Diseño.
    - Modelo de Despliegue.
    - Modelo de Datos.
- Verificación y revisión formal de los resultados obtenidos.

El presente trabajo cuenta de cuatro capítulos:

En el capítulo 1 se analizan los conceptos y evolución de la Contabilidad y las Finanzas como herramienta de gestión. Se profundiza sobre los procesos de diseño para sistemas de software, desde el enfoque de diferentes metodologías de desarrollo, así como los principios que guían todo el proceso, se aborda el

tema de las métricas más utilizadas para medir la calidad del diseño de un software, se hace un resumen de la evolución de los sistemas ERP.

El capítulo 2 resume un estudio de las principales herramientas CASE utilizadas para documentar y realizar el diseño de un proyecto software. Se presenta un análisis de las tecnologías actuales y las tendencias en la construcción de Software de Gestión especialmente para este entorno de negocio.

El capítulo 3 muestra una selección de los principales artefactos que representan los resultados de este flujo de trabajo. Se desarrollan estos artefactos, partiendo de la fundamentación del tema y el estudio del arte, de las diferentes herramientas, metodologías, frameworks y tecnologías existentes.

En el capítulo 4 se realiza una vista de los resultados una vez concluido todo el flujo de trabajo documentado en el capítulo anterior. Se tienen en cuenta además los indicadores del proyecto productivo relacionado con el Rol de Diseñador de Sistema así como una serie de métricas que miden el alcance y la calidad de los resultados obtenidos.

Capítulo 1: Fundamentación del tema

1.1. Introducción

En el presente capítulo se analizan algunos conceptos y evolución de la Contabilidad y las Finanzas como herramienta de gestión. Se hace alusión en este capítulo al surgimiento y evolución de las empresas élites en el mercado de los sistemas ERP y sus principales productos. Se mencionan además algunas consideraciones sobre el flujo de trabajo relacionado específicamente con el Diseño según los puntos de vista de algunas de las principales metodologías de desarrollo, tendencias y tecnologías actuales para este entorno de negocio, así como su evolución.

1.2. La Contabilidad

1.2.1. Conceptos y evolución

El concepto de la contabilidad se maneja desde el comienzo de la civilización misma y hacer referencia a toda esta historia, en pocos párrafos, realmente es una tarea difícil. Baste con decir que mucho antes de Cristo, famosos y grandes imperios de la antigüedad como el Egipcio, el Griego y el Romano, marcaron los primeros pasos en este mundo como resultado de sus intercambios comerciales.

El inicio de la literatura contable se enmarca por primera vez de manera formal, en la obra de Francisco Fray Lucca Paccioli de 1494 titulada *“Suma de aritmética, geométrica, proporciones y proporcionalidad “*, material donde por primera ves se emplea el término de partida doble. A comienzos del siglo XIX, comienza la Revolución Industrial y con ella Adam Smith, David Ricardo y otros economistas clásicos de la época, crean las bases del liberalismo y comienza una revolución en el pensamiento comercial. La contabilidad comienza a experimentar modificaciones y cambios en sus concepciones, con el nombre de "Principios de Contabilidad", en 1887 se funda la "American Association of Public Accountants", antes, en 1854 "The Institute of Chartered Accountants of Scotland", en 1880 "The Institute of Chartered Accountants of England and Wales", otros organismos similares se constituyen en Francia en 1881, Austria en 1885, Holanda en 1895 y Alemania en 1896. A raíz de la crisis 1930, en Estados Unidos, se crea el Instituto Americano de Contadores Públicos.

Sobre la base de estas experiencias y con el devenir de los años, este concepto ha evolucionado, de manera que el grado de especialización de la disciplina ha tomado fuerzas. La contabilidad ahora es el lenguaje que se utiliza en el mundo de los negocios. Cualquier empresa o institución pública que pretenda prosperar y tener resultados positivos, tiene que saber comunicarse con otras e interpretar la información que recibe, procedente de diferentes ámbitos. Sobre ella existen diversas definiciones con diferentes enfoques, a continuación relacionamos algunas de las más importantes:

1. “Es el arte de ***registrar, clasificar y resumir*** en forma significativa y en términos de dinero, las operaciones y los hechos que son cuanto menos de carácter financiero, así como el de interpretar sus resultados.”[2].
2. “Es el proceso de, ***identificar, medir y comunicar*** la información económica que permite formular juicios basados en información y la toma de decisiones, por aquellos que se sirven de la información.”[3]
3. “Es la ciencia que se encarga del ***estudio cualitativo y cuantitativo*** del patrimonio, tanto en su aspecto estático como dinámico, con la finalidad de lograr la dirección adecuada de las riquezas que la integran.”[4]
4. “Es una técnica en constante evolución, basada en conocimientos razonables y lógicos que tiene como objetivo fundamental, ***registrar y sintetizar las operaciones financieras*** de una entidad e interpretar sus resultados.”[5]
5. "La contabilidad es el sistema que ***mide*** las actividades del negocio, ***procesa*** esa información convirtiéndola en informes y ***comunica*** estos hallazgos a los encargados de tomar las decisiones." [6]
6. "La contabilidad es el arte de ***interpretar, medir y describir*** la actividad económica." [7]
7. "La contabilidad es el lenguaje que utilizan los empresarios para ***poder medir y presentar*** los resultados obtenidos en el ejercicio económico, la situación financiera de las empresas, los cambios en la posición financiera y/o en el flujo de efectivo."[8]

En la actualidad, los sistemas de gestión empresarial incluyen el concepto de contabilidad, el cual se erige como uno de los más importantes y eficaces para dar a conocer los diversos ámbitos de la información económica.

1.2.2. Principios

Los principios de la Contabilidad, se pueden definir como guías de acuerdo con las cuales se ha de manejar la información. El resultado de aplicar sistemáticamente estos principios, es que la información que se obtiene muestra la imagen fiel de la realidad a que se refiere.

Los principios de contabilidad generalmente aceptados, son los conceptos básicos que se reconocen como esenciales para la cuantificación y adecuado registro de los estados contables y sus informes financieros y de gestión complementarias, de manera tal que los mismos registren en el tiempo en forma uniforme las variaciones patrimoniales y el resultado de las operaciones, siendo necesario entonces, el conocimiento de los criterios seguidos para su preparación, lo cual facilita entre otros aspectos, el fluido accionar de los órganos de control público.

Los principios de contabilidad deben aplicarse de manera conjunta y relacionada entre si. Las bases conceptuales que los conforman guardan relación tanto con el proceso económico financiero, como con el flujo continuo de operaciones a los fines de identificarlas y cuantificarlas, de manera tal que satisfacen la necesidad de información de los responsables de la conducción de la entidad, como así también a terceros interesados y por lo tanto, les permitan adoptar decisiones sobre la gestión del mismo.
Los diferentes principios contables públicos recogidos y definidos en los Principios de Contabilidad Generalmente Aceptados (P. C. G. A.) son los siguientes:

- Principio de entidad contable.
- Principio de uniformidad.
- Principio de prudencia.
- Principio de gestión continuada.
- Principio de importancia relativa.
- Principio de devengo.
- Principio de registro.
- Principio de precio de adquisición.
- Principio de correlación de ingreso y gastos.
- Principio de imputación de la transacción.

- Principio de desafectación.
- Principio de no compensación.

1.2.3. Importancia de la Contabilidad como herramienta de gestión

La contabilidad representa una herramienta muy importante en nuestros días, entendida como un sistema de información que expresa en términos cuantitativos y monetarios las transacciones que realiza una entidad económica o que le afectan, con el fin de proporcionar información útil y segura a usuarios internos y externos a la organización para su toma de decisiones.

La contabilidad no es solamente una de las obligaciones de una organización sino un instrumento indispensable para su gerencia, pues a través de sus análisis se puede conocer la situación financiera y económica de la organización; al mismo tiempo que se extraen conclusiones vitales para la toma de decisiones en la organización.

Hoy en día una organización no puede ser competitiva si no cuenta con sistemas de información eficientes de todo tipo, incluyendo un sistema de contabilidad. Ante un ambiente de globalización, donde la competencia es intensa entre todo tipo de organizaciones del tamaño que sean, se necesita un flujo de datos constante y preciso para tomar las decisiones correctas y conducir la entidad al logro de sus objetivos.

La importancia de la contabilidad no es tan solo generar la información, sino que esta sea aprovechada para lograr la misión de la organización y para realizar los objetivos, planes y proyectos de los diferentes usuarios de la misma, tanto internos como externos. Solo así se puede dar a la contabilidad un sentido y un uso verdaderamente útil e importante, un uso estratégico.

A la contabilidad se le considera como el registro en los libros contables de cada uno de los documentos que intervienen en las transacciones que realiza la institución. Posteriormente se limita a los estados contables que se reflejan el resultado que habría obtenido en un determinado período. Los estados contables usualmente, son procesados y emitidos al final del período.

La información contable de gestión se utiliza en tres funciones gerenciales fundamentales, las cuales se describen a continuación:

Control. La gerencia debe verificar que se cumplen las diversas metas asignadas a las distintas gerencias departamentales u oficinas, quienes son responsables, de la realización de las operaciones de la institución; asimismo debe verificar que sean ejecutadas de acuerdo a los estándares establecidos. El

control de las operaciones facilita el cumplimiento de objetivos porque permite corregir deficiencias o problemas oportunamente, facilitando la solución de los mismos al ser detectados con prontitud. El control sirve como medio de comunicación porque permite informar previamente al personal, a cerca de los planes y políticas de la dirección y en general a las acciones que desea cumplir la institución. La información gerencial sirve come medio de evaluación, porque permite conocer el grado de desempeño de las unidades ejecutoras permitiendo incentivar a los empleados, reasignar cargos y también servir como medio de llamada de atención al observarse incumplimiento de los objetivos y resultados.

Coordinación. Las distintas áreas de actividad de la institución deben trabajar coordinadamente bajo objetivos comunes, como son el cumplimiento de las políticas establecidas por la gerencia. En este sentido, las distintas actividades que se ejecuten deben estar debidamente entrelazadas y coordinadas en su ejecución, esta situación permite minimizar errores, evitar la duplicidad de funciones, etc. La contabilidad como herramienta de gestión ayuda a este proceso de coordinación.

Planeamiento. La institución desde su constitución e inclusive antes de la misma, siempre debe estar planeando sus actividades futuras. En este orden de ideas, “planear” consiste en el proceso de decidir con antelación, el curso de acción a seguir en el futuro, lo que equivale a decir: Planear implica tomar decisiones. Cuando la institución planea sus actividades a seguir en un período determinado, le permite observar ahora lo que le puede suceder en el futuro, detectará la existencia de problemas pudiendo establecer distintas alternativas de solución, analizando las consecuencias con el fin de decidir por la mejor de ellas. Para estas tres funciones gerenciales se utiliza la contabilidad gerencial constituyendo una herramienta de gestión porque su utilización es permanente y constante, permitiendo tomar decisiones oportunas, claras y coherentes, buscando el beneficio de la institución y de los socios que la conforman. En definitiva, es necesario que la información sea confiable, clara, oportuna, veraz y razonable, que permita generar diversas alternativas de cursos de acción, de financiamiento, de inversión, de solución de problemas, etc.

El objetivo de todo sistema de gestión contable, es suministrar información útil y aunque las necesidades de información son muy diversas, tanto en cantidad de información a suministrar como en su presentación, dicha información debe necesariamente mostrar una imagen fiel de la realidad de la cual es reflejo. Para ello, y junto con otras premisas el suministro de información debe cumplir una serie de requisitos como son:

- Identificabilidad.

- Claridad.
- Relevancia.
- Imparcialidad.
- Verificabilidad.
- Oportunidad.
- Razonabilidad.
- Economía.
- Objetividad.

1.3. Las Finanzas

1.3.1. Conceptos y evolución

El llamado modelo clásico surge de la mano de Adam Smith en su libro "La riqueza de las naciones", en 1776, donde analizó el modo en que los mercados organizaban la vida económica y conseguían un rápido crecimiento económico, demostrando además que un sistema de precios y de mercados es capaz de coordinar los individuos y a las empresas sin la presencia de una dirección central. A partir de ahí le secundaron en estas ideas otros importantes economistas como Malthus, Mill o David Ricardo, Walras, Pareto, Wicksell y Marshall. [9]

Hasta principios del siglo XIX en el ámbito de las finanzas, los gerentes financieros se dedicaban a llevar libros de contabilidad o a controlar la teneduría, teniendo como principal tarea buscar financiación en el momento oportuno. Ya a mediados del siglo XX, adquiere una importancia relevante la planificación y control de los recursos con que contaban las empresas y con ello la implantación de presupuestos y controles de capital y tesorería.

El principio económico más importante que utilizan las finanzas para la administración es el **Análisis Marginal**, el cual establece que es necesario **Tomar Decisiones Financieras** e intervenir en la Economía solo cuando los beneficios adicionales excedan a los costos agregados y teniendo como consecuencia que todas las decisiones financieras se orientan a un cálculo de sus beneficios y costos marginales.[10]

Actualmente, la metodología basada en el descuento de los flujos de caja parece indiscutible y es la más congruente y sólida en cuanto a sus fundamentos teóricos. Por esta línea han pretendido progresar Martín Marín y Trujillo Ponce (2000) en su obra *"Manual de valoración de empresas"*. Respecto al tema de las

empresas de nueva economía o economía virtual relacionada con Internet estos autores según sus propias palabras prefieren darse un plazo de espera antes de abordar tan espinoso asunto. Estas empresas parecen escapar de la lógica de los modelos hasta ahora desarrollados de valoración.

1. En sentido general se define como parte de la economía que estudia lo relativo a la **obtención** y **gestión** del dinero y de otros valores como títulos, bonos, etc. [11]
2. En un sentido más práctico las finanzas se refieren a la **obtención** y **gestión**, por parte de una compañía o del Estado, de los **fondos** que necesita para sus operaciones y de los criterios conque dispone de sus activos. [11]

Las finanzas tratan, por lo tanto, de las **condiciones** y **oportunidad** en que se consigue el capital, de los usos de éste y de los pagos e intereses que se cargan a las transacciones en dinero.

1.3.2. Algunas consideraciones

Un sistema financiero que trabaje en función de la distribución del dinero hacia el sistema productivo y que genere beneficios tanto para la institución, el gobierno y público en general, puede marcar la diferencia entre tener o no tener desarrollo y eficiencia. Las finanzas son una poderosa fuerza para el desarrollo siempre y cuando haya una planificación y un control adecuado. La dimensión e importancia de una herramienta para el manejo de las Finanzas en la Administración de una entidad cualquiera, dependen en gran medida del tamaño de la empresa y los niveles administrativos con los que esta cuente.

Básicamente existen dos diferencias entre las Finanzas y la Contabilidad: el flujo de efectivo y la toma de decisiones. La función principal del contador es generar y proporcionar información para medir el rendimiento de la empresa, evaluar su posición financiera y pagar los impuestos respectivos, lógicamente mediante el uso de ciertos principios. El contador prepara los Estados Financieros que registran los ingresos y los gastos cuando se incurren en ellos, llamándole a estos el método de **Acumulaciones**. Por el otro lado, el encargado de las Finanzas destaca sobre todo los flujos de efectivo, es decir, las entradas y salidas de efectivo, mientras la solvencia de la empresa se mantiene mediante la planeación de los flujos de efectivo requeridos para hacer frente a sus obligaciones y adquirir todo lo necesario para lograr los objetivos de la entidad. Utiliza para este fin, el método del **Efectivo** para registrar los ingresos y gastos solo con respecto a los flujos reales de entradas y salidas de efectivo. La toma de decisiones, representa la segunda diferencia entre las Finanzas y la Contabilidad, tiene que ver con la utilización esta información, pues esta es analizada y ajustada.

1.4. Marco Legal

La contabilidad internacional se desarrolla a través de organismos de carácter mundial que avalados por organizaciones del mismo peso, promueven la emisión de estándares de contabilidad, dándole un direccionamiento hacia la uniformidad a todos los procesos económicos, pero ante todo hacia la calidad de la información contable. El IASB (International Accounting standard Board) es un ejemplo dentro del desarrollo de la contabilidad internacional, fue creado como un organismo de carácter profesional y de ámbito mundial, mediante un acuerdo realizado por organizaciones profesionales de nueve países dentro de los que se encontraban Australia, Canadá, Francia, Alemania, Japón, México, Holanda, Reino Unido, Irlanda y Estados Unidos. Uno de sus objetivos fundamentales, expresamente señalados en su constitución, era formular y publicar buscando el interés público, normas contables que fueran observadas en la presentación de los Estados Financieros, así como promover su aceptación y seguimiento en todo el mundo.

En el caso específico de La Republica Bolivariana de Venezuela, existen una serie de leyes que rigen la administración pública, las cuales se tuvieron en cuenta para el diseño de este sistema y se listan a continuación:

- Ley Orgánica de Régimen Presupuestario.
- Reglamento de la Ley Orgánica de Régimen Presupuestario.
- Ley Orgánica de la Hacienda Pública Nacional.
- Ley Orgánica de la Contraloría General de la República.
- Reglamento de la Ley Orgánica de la Contraloría General de la República.
- Constitución de la República Boliviariana de Venezuela.
- Ley Orgánica de Salvaguarda del Patrimonio Público. TITULO II. “Del Control de la Administración Central”. Capítulo IV. Del Control de los Bienes Nacionales. Artículo 24.
- Ley Organica de la Administración Financiera del Sector Publico.
- Reglamento de la Ley Organica de la Administración Financiera del Sector Publico.
- Código Orgánico Tributario.
- Impuesto al Valor Agregado.
- Impuesto sobre la Renta.

1.5. Evolución y estado actual de los ERP

1.5.1. ¿Qué es un ERP? ¿Cual es su principal función?

Por sus siglas en ingles: Enterprise Resource Planning lo cual quiere decir en español, Aplicaciones de Planificación de Recursos Empresariales.

Los ERP son sistemas transaccionales, están diseñados para trabajar de forma parcial o total, las áreas funcionales de la empresa. Involucra diseñar los procesos de la empresa, soportarlos, procesar los datos y obtener de ellos información específica. Así puede haber un seguimiento y control de los procesos de negocio como son: **finanzas** y **contabilidad**, **compras**, **manufacturas**, **logística**, **recursos humanos** y **mercadotecnia**.

Los ERP gestionan de manera integrada y eficiente la información de la empresa, comunicando las diferentes áreas de negocio mediante procesos electrónicos. La función principal es organizar y estandarizar procesos y datos internos de la empresa, transformándolos en información útil para la toma de decisiones. Aunque los ERP apoyan en la toma de decisiones no quiere decir que ellos lo hagan, sino que los administradores o contadores (humanos) tienen el poder final para tomar decisiones.

1.5.2. Historia y Evolución

Los ERP surgen a partir de la Segunda Guerra Mundial, hasta la fecha estos sistemas han sufrido un largo proceso de cambios y adaptaciones.

A principios de la década de los sesenta, estos sistemas incursionan en el sector productivo, principalmente en EE.UU. Durante los 60 y 70 tuvieron un desarrollo importante ya que permitían reducir los inventarios al planear sus insumos en base a la demanda real. Estos sistemas fueron un apoyo fundamental en el crecimiento de esta industria, por lo que aumentan los recursos destinados a la investigación y desarrollo de estos y de las tecnologías informáticas.

En los 80 estos sistemas evolucionaron completamente lo que dio lugar a los MRP II, aunque el acrónimo cambia de manera radical a Manufacturing Resource Planning, estos nuevos sistemas permitían cuidar factores relacionados con las capacidades de manufactura desde producción hasta logística, apoyados fuertemente en los avances que se habían logrado en la industria tecnológica.

En la década de los 90 se veía un panorama disperso, por un lado los sistemas especializados en factores de requerimiento y por otro los sistemas orientados hacia la planeación de procesos de manufactura.

Dado el contexto de negocio que se comenzó a vivir, regido por un marco de competencia global que exige mayores niveles de eficiencia y productividad dentro de los procesos y las operaciones de la empresa para poder alcanzar los niveles óptimos de servicio; las empresas necesitaban soluciones de tecnologías integrales que les permitiera alcanzar estos niveles. Debido a los requerimientos la industria del software desarrolló varias aplicaciones con el fin de interconectar los sistemas MRP II con los sistemas MRP existentes, a fin de integrar ese panorama disperso.

Poco después la simple conexión se transformó en un sistema empresarial integrado. Cuando sus alcances se llevaron a las áreas de **finanzas**, **recursos humanos, compras**, **ventas** y **cobros** entre otras, los ERP habían nacido.[12]. La **figura 1** muestra un esquema de la evolución de estos sistemas.

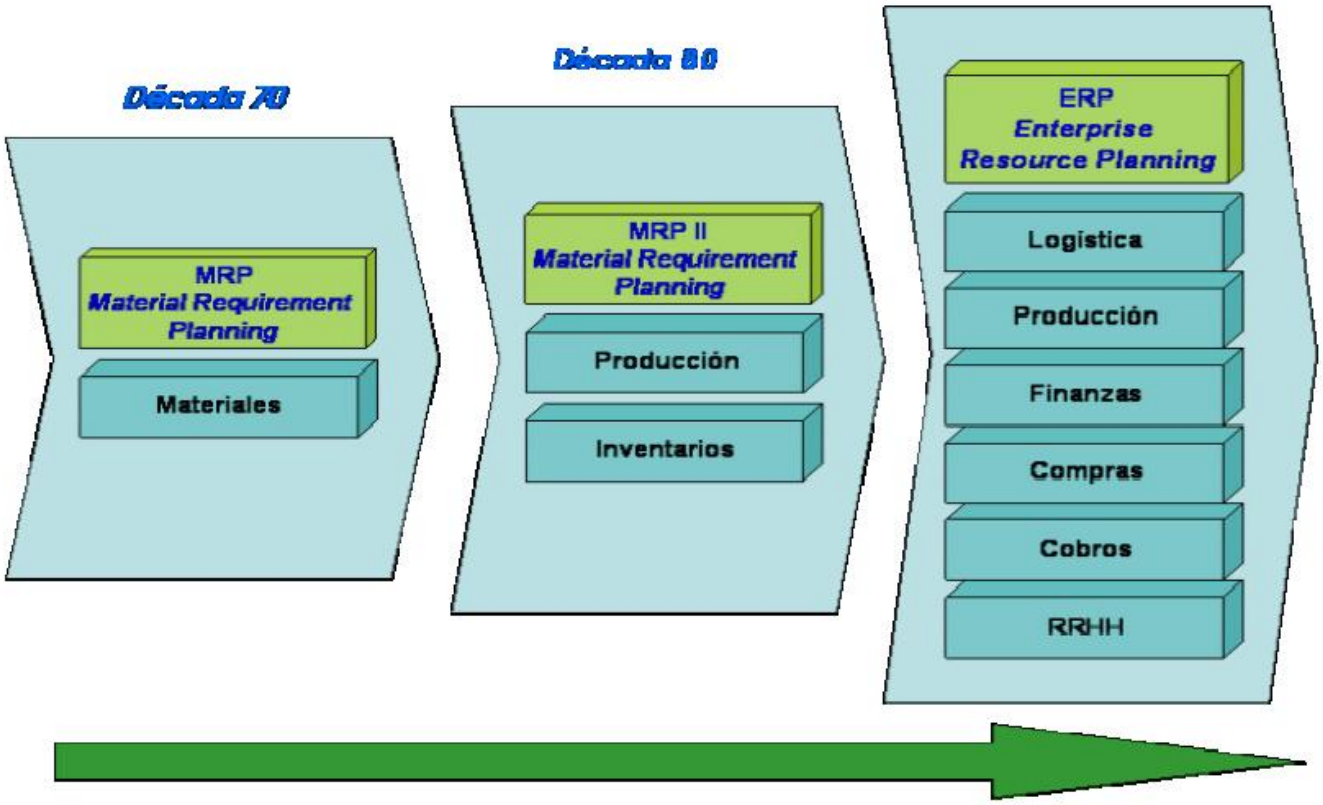

Figura 1. Evolución de los sistemas ERP.

1.5.3. Principales Empresas productoras de Sistemas ERP

✓ SAP- R/3

La compañía alemana SAP, mundialmente conocida por su producto R/3, ofrece soluciones para administrar la cadena de logística (SCM), administración de las relaciones con proveedores (CRM), Businness Intelligence (BI), portales y marketplaces, entre otras.
R/3, en el que la **R** significa *procesamiento en tiempo real* y el número **3** se refiere a las tres capas de la arquitectura de proceso: bases de datos, servidor de aplicaciones y cliente. El predecesor de R/3 fue R/2.

✓ PeopleSoft

Otra empresas conocida mundialmente por sus aportes en el campo de los sistemas ERP es PeopleSoft que se define actualmente como proveedor de soluciones de negocios, esta famosa empresa cuenta con una oferta compuesta por ERP-HR, CRM, Bussines Intelligence, orientado tanto a empresas medianas o grandes (sobre los 50 millones en facturación), como también las pequeñas (que facturan menos de 50 millones de dólares). Con un ERP compuesto por más de 168 módulos integrados totalmente, PeopleSoft es considerado uno de los mejores en el mundo.

✓ Microsoft Corporation

El gigante Microsoft no podía dejar de poner su firma en el mundo de la gestión integral. Desde el año 2000, con la compra de la empresa de software de contabilidad Great Plains, Microsoft dio su señal de ataque a un mercado que, hasta entonces, parecía alejado de sus intereses. Entre los ERP más importantes que esta compañía ha adquirido hasta el momento se encuentran Microsoft Dynamics Navision y Microsoft Dynamics Axapta.

1.6. Diseño del software

1.6.1. Estado actual del diseño del software

El diseño del software en la actualidad esta orientado a diferentes vertientes específicas, según las necesidades y las arquitecturas que los problemas del mundo real requieran, los estilos arquitectónicos han ido evolucionando para acercar cada vez más la computadora a los problemas reales de la vida. Según Pressman “El diseño del software, al igual que los enfoques de diseño de ingeniería en otras disciplinas, va cambiando continuamente a medida que se desarrollan métodos nuevos, análisis mejores y se amplia el conocimiento. Las metodologías de diseño del software carecen de la profundidad, flexibilidad y naturaleza cuantitativa que se asocian normalmente a las disciplinas de diseño de ingeniería más clásicas. Sin embargo, si existen métodos para el diseño del software; también se dispone de calidad de diseño y se pueden aplicar notaciones de diseño”.

“El diseño del software se encuentra en el núcleo técnico de la ingeniería del software y se aplica independientemente del modelo de diseño de software que se utilice. Una vez que se analizan y especifican los requisitos del software, el diseño del software es la primera de las tres actividades técnicas - diseño, generación de código y pruebas - que se requieren para construir y verificar el software. Cada actividad transforma la información de manera que de lugar por último a un software de computadora validado.”[13]

Pressman hace referencia a la importancia del diseño en la calidad del software. “El diseño es el lugar en donde se fomentan la calidad en la ingeniería del software. El diseño proporciona las representaciones del software que se pueden evaluar en cuanto a calidad. El diseño es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado. El diseño del software sirve como fundamento para todos los pasos siguientes del soporte del software y de la ingeniería del software. Sin un diseño, corremos el riesgo de construir un sistema inestable un sistema que falla cuando se lleven a cabo cambios; un sistema que puede resultar difícil de comprobar; y un sistema cuya calidad no puede evaluarse hasta muy avanzado el proceso, sin tiempo suficiente y con mucho dinero gastado” [13]

1.6.2. Evolución del diseño del software

La evolución del diseño del software es un proceso continuo que ha abarcado las últimas cuatro décadas. El primer trabajo de diseño se concentraba en criterios para el desarrollo de programas modulares y métodos para refinar las estructuras del software de manera descendente. Los aspectos procedimentales de la definición de diseño evolucionaron en una filosofía denominada programación estructurada. Un trabajo posterior propuso métodos para la conversión del flujo de datos o estructura de datos en una definición de diseño. Enfoques de diseños más recientes hacia la derivación de diseño proponen un método orientado a objetos. Hoy en día, se ha hecho hincapié en un diseño de software basado en la arquitectura del software.

Independientemente del modelo de diseño que se utilice, un ingeniero del software deberá aplicar un conjunto de principios fundamentales y conceptos básicos para el diseño a nivel de componentes, de interfaz, arquitectónico y de datos.

1.6.3. Principios del diseño

El diseño de software es tanto un proceso como un modelo. El proceso de diseño es una secuencia de pasos que hacen posible que el diseñador describa todos los aspectos del software que se va construir. Sin embargo, es importante destacar que el proceso de diseño simplemente no es un recetario. Un conocimiento creativo, experiencia en el tema, un sentido de lo que hace que un software sea bueno y un compromiso general con calidad son factores críticos de éxito para un diseño competente.

El modelo de diseño es el equivalente a los planes de un arquitecto para una casa. Comienza representando la totalidad de todo lo que se va a construir (por ejemplo, una representación en tres dimensiones de la casa) y refina lentamente lo que va a proporcionar la guía para construir cada detalle (por ejemplo, el diseño de fontanería). De manera similar, el modelo de diseño que se crea para el software proporciona diversas visiones diferentes de software de computadora.

Los principios básicos de diseño hacen posible que el ingeniero del software navegue por el proceso de diseño. Algunos autores sugieren un conjunto de principios para el diseño del software, los cuales han sido adaptados y ampliados en la lista siguiente: [13]

En el proceso de diseño no deberá utilizarse "orejeras". Un buen diseñador deberá tener en cuenta enfoques alternativos, juzgando todos los que se basan en los requisitos del problema, los recursos disponibles para realizar el trabajo y los conceptos de diseño.

El diseño deberá poderse rastrear hasta el modelo de análisis. Dado que un solo elemento del modelo de diseño suele hacer un seguimiento de los múltiples requisitos, es necesario tener un medio de rastrear como se han satisfecho los requisitos por el modelo de diseño.

El diseño no deberá inventar nada que ya esté inventado. Los sistemas se construyen utilizando un conjunto de patrones de diseño, muchos de los cuales probablemente ya se han encontrado antes. Estos patrones deberán elegirse siempre como una alternativa para reinventar. Hay poco tiempo y los recursos son limitados. El tiempo de diseño se deberá invertir en la representación verdadera de ideas nuevas y en la integración de esos patrones que ya existen.

El diseño deberá minimizar la distancia intelectual entre el software y el problema como si de la misma vida real se tratara. Es decir, la estructura del diseño del software (siempre que sea posible) imita la estructura del dominio del problema.

El diseño deberá presentar uniformidad e integración. Un diseño es uniforme si parece que fue una persona la que lo desarrolló por completo. Las reglas de estilo y de formato deberán definirse para un equipo de diseño antes de comenzar el trabajo sobre el diseño. Un diseño se integra si se tiene cuidado a la hora de definir interfaces entre los componentes del diseño.

El diseño deberá estructurarse para admitir cambios. Los conceptos de diseño estudiados hacen posible un diseño que logra este principio.

El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operación aberrantes. Un software bien diseñado no deberá nunca explotar, deberá diseñarse para adaptarse a circunstancias inusuales y si debe terminar de funcionar, que lo haga de forma suave.

El diseño no es escribir código y escribir código no es diseñar. Incluso cuando se crean diseños procedimentales para componentes de programas, el nivel de abstracción del modelo de diseño es mayor que el código fuente. Las únicas decisiones de diseño realizadas a nivel de codificación se enfrentan con pequeños datos de implementación que posibilitan codificar el diseño procedimental.

El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminarlo. Para ayudar al diseñador en la evaluación de la calidad se dispone de conceptos de diseño y de medidas de diseño.

El diseño deberá revisarse para minimizar los errores conceptuales. A veces existe la tendencia de centrarse en minucias cuando se revisa el diseño, olvidándose del bosque por culpa de los árboles. Un

equipo de diseñadores deberá asegurarse de haber afrontado los elementos conceptuales principales antes de preocuparse por la sintaxis del modelo del diseño.

1.6.4. Patrones de diseño y frameworks

Los Patrones de Diseño nos hablan de como construir software, de como utilizar las clases y los objetos de forma conocida.

Los precedentes a los patrones de diseño vienen del campo de la Arquitectura, Christopher Alexander a finales de los 70 escribe varios libros acerca de urbanismo y construcción de edificios, y se plantea reutilizar diseños ya aplicados en otras construcciones que cataloga como modelos a seguir.

En 1987 Ward Cunningham y Kent Beck utilizan las ideas de Alexander para desarrollar un lenguaje de patrones como guía para los programadores de Smaltalk, dando lugar al libro "Using Pattern Languajes for Object-Oriented Programs".

Posteriormente en 1991 Jim Coplien publica el libro "Advanced C++ Programming Styles and Idioms", donde realiza un catalogo de "idioms" (especie de patrones).

Entre 1990 y 1994, Erich Gamma, Richard Helm, Ralph Johnson y Hohn Vlissides (conocidos como el grupo de los cuatro) realizan el primer catálogo de patrones de diseño, que publican en el libro "Design Patterns: Elements of Reusable Object-Oriented Software" (Gang of Four).

Los patrones se clasifican según su propósito en:

1. **Patrones de Creación:** Tratan la creación de instancias.
2. **Patrones Estructurales:** Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad.
3. **Patrones de Comportamiento:** Tratan la interacción y cooperación entre clases.

Los **patrones de creación** abstraen la forma en que se crean los objetos, de forma que permite tratar las clases a crear de forma genérica apartando la decisión de qué clases crear o como crearlas.

Pero los Patrones de Diseño son conceptos aplicables directamente en la producción de software, cualquier abstracción no se queda en el aire como una entelequia que solo sirve para dar discursos. Los patrones de creación se listan a continuación:

1. Patrones de Creación de Clase:
    - Factoría Abstracta.
    - Builder.
2. Patrones de Creación de Objeto:
    - Método Factoría.
    - Prototipo.
    - Singleton.
    - Object Pool.

También existen los Patrones Estructurales, como antes se había mencionado tratan la relación entre clases, la combinación y la formación de estructuras de alta complejidad en un sistema, alguno de estos patrones son:

1. Patrones de Estructurales de Clase:
    - Herencia Múltiple.
    - Class Adapter.
2. Patrones de Estructurales de Objeto:
    - Object Adapter.
    - Bridge.
    - Composite.
    - Decorador.
    - Facade.
    - Flyweight.
    - Proxy.

Los **patrones de comportamiento** hablan de como interaccionan entre si los objetos para conseguir ciertos resultados.

Los principales patrones de comportamiento son:

1. Experto.

2. Comando.
3. Interprete.
4. Iterator.
5. Mediador.
6. Memento.
7. Observador.
8. Estado.
9. Estrategia.
10. Plantilla.
11. Visitante.
12. Vista/Controlador.

Por otra parte los patrones están bastante relacionados con los marcos de trabajos, comúnmente conocidos como frameworks. En el desarrollo de software, un framework es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio. Los frameworks orientados a objetos (llámense simplemente frameworks) son la piedra angular de la moderna ingeniería del software. El desarrollo del framework está ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente. Los frameworks son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.

“La capacidad de reutilización del código y del diseño de frameworks orientados al objeto permite una productividad mayor y un tiempo de mercado breve en el desarrollo de aplicaciones, en comparación con el desarrollo tradicional de los sistemas de software. La configuración flexible de frameworks, permite la reutilización del núcleo. El desarrollo del framework ha sido exitoso en muchos dominios.”[14]

1.6.5. Estilos Arquitectónicos. El papel del diseño en la arquitectura del software

El tópico más urgente y exitoso en arquitectura de software en los últimos cuatro o cinco años es, sin duda, el de los patrones, tanto en lo que concierne a los patrones de diseño como a los de arquitectura. Inmediatamente después, en una relación a veces de complementariedad, otras de oposición, se encuentra la sistematización de los llamados estilos arquitectónicos. Cada vez que alguien celebra la mayoría de edad de la arquitectura de software, y aunque señale otros logros como las técnicas de refinamiento, esos dos temas se destacan más que cualesquiera otros. Sin embargo, sólo en contadas ocasiones la literatura técnica existente se ocupa de analizar el vínculo entre estilos y patrones. Se los yuxtapone cada vez que se enumeran las ideas y herramientas disponibles, se señala con frecuencia su aire de familia, pero no se articula formal y sistemáticamente su relación.

Habrá que admitir que ambos asuntos preocupan y tienen como destinatarios a distintas clases de profesionales, o diferentes stakeholders, como ahora se recomienda llamar, quienes trabajan con estilos favorecen un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, mientras que quienes se ocupan de patrones se centran en cuestiones que están más cerca del diseño, la práctica, la implementación, el proceso, el refinamiento, el código. Los patrones coronan una práctica de diseño que se origina antes que la arquitectura de software se distinguiera como discurso en perpetuo estado de formación y proclamara su independencia de la ingeniería en general y el modelado en particular. Los estilos, en cambio, expresan la arquitectura en el sentido más formal y teórico, constituyendo un tópico esencial de lo que Goguen ha llamado el campo “seco” de la disciplina.

“Conviene caracterizar el escenario que ha motivado la aparición del concepto de estilo, antes siquiera de intentar definirlo. Desde los inicios de la arquitectura de software, se observó que en la práctica del diseño y la implementación ciertas regularidades de configuración aparecían una y otra vez como respuesta a similares demandas. El número de esas formas no parecía ser muy grande. Muy pronto se las llamó estilos, por analogía con el uso del término en arquitectura de edificios. Un estilo describe entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes. Una vez que se han identificado los estilos, es lógico y natural pensar en re-utilizarlos en situaciones semejantes que se presenten en el futuro. Igual que los patrones de arquitectura y diseño,

todos los estilos tienen un nombre: cliente-servidor, modelo-vista-controlador, tubería-filtros, arquitectura en capas, etc.” [15]

En un estudio comparativo de los estilos, Mary Shaw considera los siguientes, mezclando referencias a las mismas entidades a veces en términos de “arquitecturas”, otras invocando “modelos de diseño”: [15]

1. Arquitecturas orientadas a objeto.
2. Arquitecturas basadas en estados.
3. Arquitecturas de flujo de datos: Arquitecturas de control de realimentación.
4. Arquitecturas de tiempo real.
5. Modelo de diseño de descomposición funcional.
6. Modelo de diseño orientado por eventos.
7. Modelo de diseño de control de procesos.
8. Modelo de diseño de tabla de decisión.
9. Modelo de diseño de estructura de datos.

El mismo año, Mary Shaw, junto con David Garlan, propone una taxonomía diferente, en la que se entremezclan lo que antes llamaba “arquitecturas” con los “modelos de diseño”:

1. Tubería-filtros.
2. Organización de abstracción de datos y orientación a objetos.
3. Invocación implícita, basada en eventos.
4. Sistemas en capas.
5. Repositorios.
6. Intérpretes orientados por tablas.
7. Procesos distribuidos, ya sea en función de la topología (anillo, estrella, etc.) o de los protocolos entre procesos (p. ej. algoritmo de pulsación o heartbeat). Una forma particular de proceso distribuido es, por ejemplo, la arquitectura cliente-servidor.
8. Organizaciones programa principal / subrutina.
9. Arquitecturas de software específicas de dominio.
10. Sistemas de transición de estado.

11. Sistemas de procesos de control.
12. Estilos heterogéneos.

No es objetivo del presente trabajo detallar cada uno de estos estilos, para más información se recomienda la referencia bibliográfica [15].

1.7. Diseño en diferentes metodologías

En el presente epígrafe se hace un estudio del análisis y diseño en algunas de las metodologías más usadas para el desarrollo de software a nivel mundial, así como el estudio de este rol a lo largo de su historia.

1.7.1. Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. **MSF** se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Figura 2. Metodología MSF

MSF tiene las siguientes características:

1. **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

2. **Escalable**: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas o más.
3. **Flexible:** es utilizada en el ambiente de desarrollo de cualquier cliente.
4. **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología[16].

Para el diseño de soluciones de negocios con Arquitectura Microsoft.NET se usa el Proceso de Desarrollo **Microsoft Solutions Framework (MSF)**, teniendo en cuenta que, el MSF se caracteriza por:

1. **MSF** proporciona un conjunto de modelos, principios, y pautas para diseñar y desarrollar soluciones empresariales, que aseguran manejar todos los elementos de un proyecto, como las personas, procesos, y herramientas, con éxito.
2. **MSF** también provee las prácticas comprobadas para planear, **diseñar**, desarrollar y desplegar soluciones empresariales exitosas.
3. Está compuesto de distintas fases para el desarrollo de un proyecto, en cada fase se definen responsabilidades asignadas a cada miembro del equipo.
4. Cada fase esta compuesta por actividades que permitirán diseñar una aplicación y entregables asociadas a cada actividad.
5. Hay un control al final de cada fase (hitos), considerando que al final de cada control se pueden incorporar nuevas funcionalidades.
6. Al final de cada fase se obtienen documentos formales denominados entregables.
7. Este proceso de desarrollo es una combinación del modelo de desarrollo en cascada y en espiral que toma en cuenta dos dimensiones de desarrollo, el eje vertical (los fallos) y el eje horizontal (tiempo), toma como referencia el RUP (Proceso Unificado del Rational).

El Modelo **MSF** está compuesto de las siguientes fases:

Fase	Descripción
Envisioning	El éxito de un proyecto depende de la habilidad de los miembros de equipo del proyecto y los clientes, de compartir una visión clara de las metas y objetivos del proyecto. En esta fase se define la visión del proyecto.
Planning	Durante la fase de la planificación, el equipo determina qué desarrollar y planea cómo crear la solución. El equipo prepara la

	especificación funcional, crea un diseño de la solución, prepara planes de trabajo, estimaciones del costo y cronograma para el cumplimiento de los entregables. Esta fase involucra el análisis de requisitos, estos requisitos pueden categorizarse como: requisitos comerciales, requisitos del usuario, requisitos operacionales, y requisitos del sistema. Estos requisitos se usan para diseñar la solución y sus rasgos y validar la exactitud del plan.
Developing	Durante esta fase, el equipo del proyecto crea la solución. Este proceso incluye la creación del código que será implementado para la solución y la documentación del código, el equipo desarrolla también la infraestructura para la solución.
Stabilizing	En esta fase el equipo realiza la integración, carga, y prototipos que prueban la solución. Se aprueba los escenarios de la solución. El equipo se enfoca en identificar, priorizar, y resolver los problemas para que la solución pueda prepararse para el "release". Además, la solución está lista para el despliegue al negocio.
Deploying	Durante esta fase, el equipo despliega la tecnología de la solución y componentes del sitio, estabiliza el despliegue, transfiere el proyecto a las funcionalidades, y obtiene la última aprobación del cliente del proyecto. Después del despliegue, el equipo dirige una revisión del proyecto y un estudio de satisfacción de cliente. La fase culmina en el despliegue el hito completo.

Tabla 1. Fases de la metodología Microsoft Solutions Framework

Antes de finalizar el diseño lógico, el equipo tiene modelos UML para objetos, servicios, atributos y relaciones en la solución. Típicamente, el equipo usa los artefactos que mejor capturen y manejen las partes complejas del proyecto. Esto incluye el siguiente juego de entregables:

1. Inventarios de Objetos y Servicios.
2. Diagramas de Clases.
3. Diagramas de Secuencia.
4. Diagramas de Actividades.
5. Diagramas de Componentes.

Durante el diseño físico, el equipo refina estos modelos generados en el diseño lógico del sistema en si.[17]

1.7.2. Extreme Programming (XP)

La **programación extrema** o *eXtreme Programming* (XP) es una aproximación a la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change*. Se trata de un proceso ágil de desarrollo de software. Es una metodología que esta basada en los principios del manifiesto ágil que es el punto de partida de XP y de otras metodologías ágiles que están apareciendo:

1. La metodología XP esta diseñado para proyectos de corta duración.
2. Los individuos e interacciones son más importantes que los procesos y herramientas.
3. Software que funcione es más importante que documentación exhaustiva.
4. La colaboración con el cliente es más importante que la negociación de contratos.
5. La respuesta ante el cambio es más importante que el seguimiento de un plan.
6. XP se basa en 12 prácticas ya conocidas y que se refuerzan entre sí:

    - Retroalimentación a escala fina.
        - Desarrollo dirigido por pruebas.
        - El juego de la planificación.
        - Cliente in-situ.
        - Programación en parejas.
    - Proceso continuo en lugar de por lotes.
        - Entregas pequeñas.
        - Refactorización sin piedad.
        - Integración continúa.
    - Entendimiento compartido.
        - Diseño Simple (lo más simple que funcione, ¿seguro que es necesario?, simplificar vigorosamente).
        - Metáfora del sistema.
        - Propiedad colectiva del código.
        - Convenciones de código. Estándares de programación.

- Bienestar del programador.
  - 40 horas por semana.

Diseño Simple: Como fue mencionado anteriormente, el Diseño Simple o “Simple Design” establece que siempre deben ser realizadas las tareas de la forma más simple posible. Para lograr un diseño simple, nunca serán adicionadas características funcionales que no formen parte de la propuesta de requisitos analizada en la planificación del juego. En la Programación Extrema, un buen diseño es esencial para asegurar el éxito esperado.

1.7.3. Rational Unified Process (RUP)

Un proceso de desarrollo del software es un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo el proceso unificado es más que un simple proceso, es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software.

El Proceso Unificado utiliza el ***Lenguaje Unificado de Modelado*** (UML) para preparar todos los esquemas de un sistema software. De hecho, UML es una parte esencial del Proceso Unificado.

En el **diseño** modelamos el sistema y encontramos su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones que se le suponen. En concreto los propósitos del diseño son:

1. Adquirir una compresión en profundidad de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnología de destrucción y concurrencia, tecnología de interfaz de usuario, tecnologías de gestión de transacciones, etc.
2. Crear una entrada apropiada y un punto de partida para actividades de implementación subsiguiente, capturando los requisitos y subsistemas individuales, interfaces y clases.

El modelo que propone RUP esta muy cercano al de implementación lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software. Esto es especialmente cierto en la ingeniería de ida y vuelta, donde el modelo de diseño se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica.

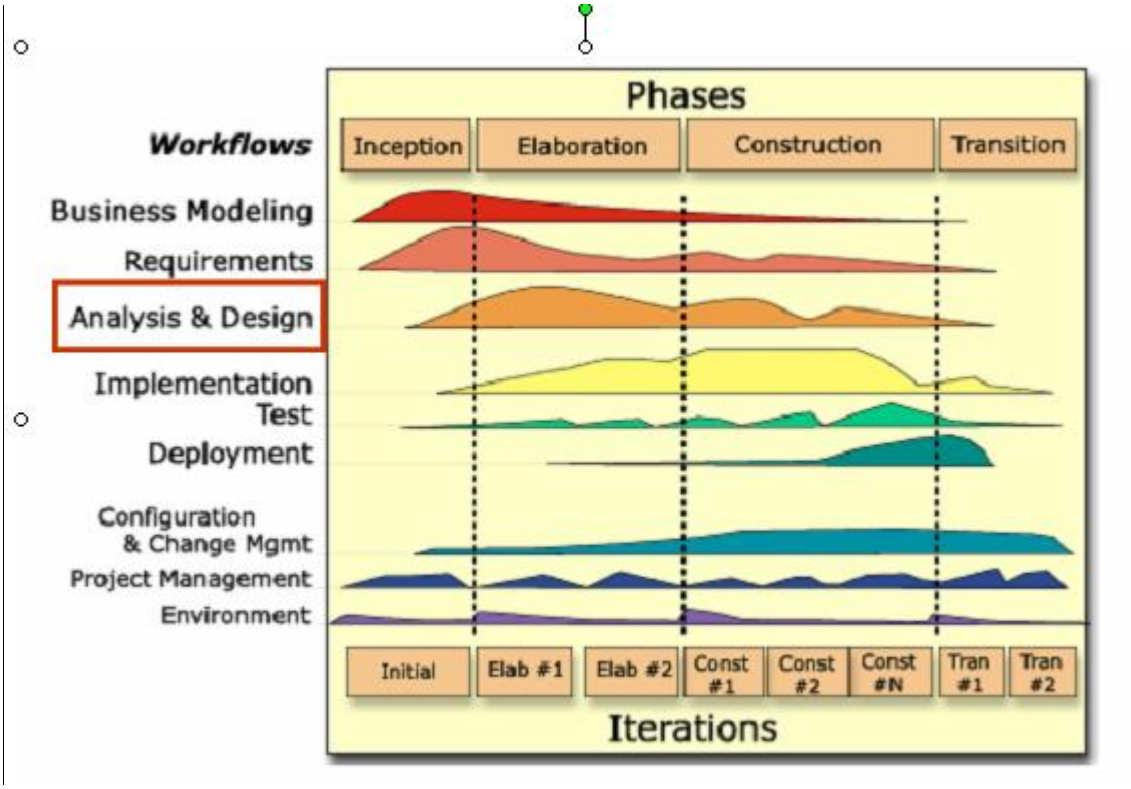

Figura 3. Vista en 2 dimensiones de RUP.

Artefactos:

1. Modelo de diseño.
2. Clases del diseño.
3. Realización de caso de uso del diseño.
4. Subsistema de diseño.
5. Interfaz.
6. Descripción de la arquitectura.
7. Modelo de despliegue.

El principal resultado del diseño es el modelo del diseño que se esfuerza en conservar la estructura del sistema impuesta por el modelo de análisis y que sirve como esquema para la implementación.

1.8. Métricas del diseño del software

El presente epígrafe hace un análisis de las formas y medidas para obtener la calidad de un diseño software. A nivel mundial se han definido métricas para medir las potencialidades y cualidades del diseño.
Las métricas de diseño para el software, como otras métricas del software, no son perfectas. Continúa el debate sobre la eficacia y cómo deberían aplicarse. Muchos expertos argumentan que se necesita más experimentación hasta que se puedan emplear las métricas de diseño. Y sin embargo, el diseño sin medición es una alternativa inaceptable[13, 18].

1.8.1. Métricas del diseño arquitectónico

Las métricas de diseño de alto nivel se concentran en las características de la arquitectura del programa, con especial énfasis en la estructura arquitectónica y en la eficiencia de los módulos. Estas métricas son de caja negra en el sentido que no requieren ningún conocimiento del trabajo interno de un módulo en particular del sistema.
Algunos expertos definen tres medidas de la complejidad del diseño del software: **complejidad estructural, complejidad de datos y complejidad del sistema.**

La complejidad estructural, $S(i)$, de un módulo i se define de la siguiente manera: $S(i) = f_{out}^{2}(i)$ donde $f_{out}(i)$ es la expansión del módulo i.

La complejidad de datos, $D(i)$, proporciona una indicación de la complejidad de la interfaz interna de un módulo i y se define como: $D(i) = V(i) / \left\lfloor f_{out}(i) + 1 \right\rfloor$ donde $V(i)$ es el número de variable de entrada y de salida que entran y salen del módulo i.

La complejidad del sistema, $C(i)$, se define como las sumas de las complejidades estructural y de datos, y se define como: $C(i) = S(i) + D(i)$.

A medida que crecen los valores de complejidad, la complejidad arquitectónica o global del sistema también aumenta. Esto lleva a una mayor probabilidad de que aumente el esfuerzo necesario para la integración y las pruebas[13].

1.8.2. Métricas de diseño a nivel de componentes

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen medidas de las “3Cs” -la cohesión, acoplamiento y complejidad del módulo-. Estas tres medidas pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes. Estas métricas son de caja blanca en el sentido de que requieren conocimiento del trabajo interno del módulo en cuestión. Las métricas de diseño de los componentes se pueden aplicar una vez que se ha desarrollado un diseño procedimental. También se pueden retrasar hasta tener disponible el código fuente.

1.8.3. Métricas orientadas a clase

Las medidas y métricas para una clase individual, la jerarquía de clases y las colaboraciones de clases poseen un valor incalculable, para el ingeniero del software que ha de evaluar la calidad del diseño. Las clases encapsulan operaciones (procesamiento) y atributos (datos).

Así mismo, la clase <<padre>> es de la que heredan las subclases (algunas veces llamadas hijas), sus atributos y operaciones. La clase, normalmente, colabora con otras clases. Cada una de estas características puede usarse como base de la medición.

Actualmente se debate en la literatura técnica. Aquellos que abanderan la teoría de medición requieren de un grado de formalismo que algunas de las métricas OO no proporcionan. De cualquier manera, es razonable declarar que todas las métricas proporcionan una visión útil para el ingeniero de software[13].

Algunas de las métricas que se han definido a nivel de clases por algunos expertos son:

La serie de métricas CK

Uno de los conjuntos de métricas OO más ampliamente referenciados, ha sido el propuesto por Chidamber y Kemerer. Normalmente conocidas como la serie de métricas CK, los autores han propuesto seis métricas basadas en clases para sistemas OO:

1. Métodos ponderados por clase (**MPC**).
2. Árbol de profundidad de herencia (**APH**).
3. Número de descendiente (**NDD**).
4. Acoplamiento entre clases objeto (**ACO**).
5. Respuesta para una clase (**RPC**).

6. Carencia de cohesión en los métodos (**CCM**).

En su libro sobre métricas OO, Lorenz y Kidd separan las métricas basadas en clases en cuatro amplias categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas al tamaño para las clases OO se centran en el recuento de atributos y operaciones para cada clase individual, y los valores promedio para el sistema OO como un todo. Las métricas basadas en la herencia se centran en la forma en que las operaciones se reutilizan en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y los aspectos orientados al código; las métricas orientadas a valores externos, examinan el acoplamiento y la reutilización[13].

1. Tamaño de clase (**MPC**).
2. Número de operaciones redefinidas por una subclase (**NOR**).
3. Número de operaciones añadidas por una subclase (**NOA**).

1.9. Conclusiones

En este capítulo se ha descrito el objeto de estudio y se ha hecho alusión al campo de acción del entorno de negocio en cuestión, se hizo un análisis de los actuales competidores en este campo así como de sus principales productos. Se definió el modelo de proceso de Diseño a seguir y se fundamentaron las métricas correspondientes.

Se arribaron a las siguientes conclusiones:

- La Contabilidad es la base de todo proceso económico, por ende recae un peso muy importante sobre la misma y las Finanzas son las responsables de manejar las condiciones y oportunidades en que se obtiene y gasta el capital.
- Emplear RUP como metodología de desarrollo, específicamente lo inherente al flujo de trabajo de Análisis y Diseño, más concretamente la segunda parte de este.
- Utilizar métricas definidas por Pressman para el chequeo y validación del Modelo de Diseño obtenido, en tres niveles: Diseño Arquitectónico, a nivel de componentes y a nivel de clases.

Capítulo 2: Herramientas CASE y Tecnologías actuales

2.1. Introducción

En el presente capítulo se hace un estudio de las herramientas case más utilizadas para documentar un proyecto software, las principalmente actividades del diseño, así como las de generación de cogido ejecutable que se deben tener en cuenta para que el diseño se aplique con productividad y eficiencia. También se realiza un análisis de las tecnologías actuales y las tendencias en las ciencias de la computación. El capítulo nos sirve para determinar cueles de estas tecnologías nos sirven para nuestro caso y en próximos capítulos hacer un análisis minucioso de los resultados.

2.2. Herramientas Case

2.2.1. Herramientas CASE para diseño y construcción de sistemas

Hoy en día, muchas empresas se han extendido a la adquisición de herramientas CASE (Ingeniería Asistida por Computadora), con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema, desde el principio hasta el final e incrementar su posición en el mercado competitivo, pero obteniendo algunas veces elevados costos en la adquisición de la herramienta y costos de entrenamiento de personal así como la falta de adaptación de la herramienta a la arquitectura de la información y a las metodologías de desarrollo utilizadas por la organización. Por otra parte, algunas herramientas CASE no ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas o virtualmente no llevan a cabo ningún análisis de los requerimientos de la aplicación.

Sin embargo, CASE proporciona un conjunto de herramientas semi-automatizadas y automatizadas que están desarrollando una cultura de ingeniería nueva para muchas empresas. Uno de los objetivos más importante del CASE (a largo plazo) es conseguir la generación automática de programas desde una especificación a nivel de diseño. [19]

✓ Rational XDE Professional para Microsoft Visual Studio .Net

XDE es la primera herramienta (Case) en acelerar el desarrollo de software al eliminar la brecha que existe entre el diseño de un sistema y su desarrollo en el IDEs de Microsoft Visual Studio .NET. Rational XDE Professional para Visual Studio .Net permite al desarrollador el programar y diseñar directamente desde su IDE - Microsoft Visual Studio .NET sin tener que estar brincando entre dos herramientas diferentes y sin integración real, entre otras facilidades la herramienta case permite:

1. Desarrollo basado en modelos con soporte para UML.
2. Ingeniería bidireccional para lenguajes C++, Visual Studio y .NET.
3. Sincronización de código-modelo automática o bajo petición.
4. Patrones y plantillas de código definibles por el usuario.
5. Modelado asistido con edición sensible al lenguaje.
6. Soporte para múltiples modelos para la Arquitectura basada en modelos.
7. Diagramas de formas libres.
8. Diseño de bases de datos lógicas y físicas.
9. Publicación web y generación de informes.

Los sistemas operativos apropiados para Rational XDE Profesional:

1. Windows 2000.
2. Windows NT.
3. Windows XP.

✓ Enterprise Architect 6.1

Enterprise Architect es una herramienta CASE para el diseño y construcción de sistemas de software. EA soporta la especificación de UML 2.0, que describe un lenguaje visual por el cual se pueden definir modelos de un proyecto.

EA es una herramienta progresiva que cubre todos los aspectos del ciclo de desarrollo, proporcionando una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento. También provee soporte para pruebas, mantenimiento y control de cambio. Algunas de las características claves de Enterprise Architect son:

1. Crear elementos del modelo UML para un amplio alcance de objetivos.
2. Ubicar esos elementos en diagramas y paquetes.
3. Crear conectores entre elementos.
4. Documentar los elementos que ha creado.
5. Generar código para el software que está construyendo.
6. Realizar ingeniería reversa del código existente en varios lenguajes.

Usando EA, permite realizar ingeniería directa y reversa de código C++, C#, Delphi, Java, Python, PHP, VB.NET y clases de Visual Basic, sincronizar códigos y elementos del modelo, diseñar y generar elementos de base de datos. La documentación de alta calidad puede ser rápidamente exportada desde sus modelos en industria estándar .formato RTF e importar a Word para una personalización y presentación final.

Enterprise Architect sustenta todos los diagramas y modelos UML. Puede modelar procesos de negocio, sitios web, interfaces de usuario, redes, configuraciones de hardware, mensajes y más. Estimar el tamaño de su proyecto en esfuerzo de trabajo en horas. Capturar y trazar requisitos, recursos, planes de prueba, solicitudes de cambio y defectos. Desde los conceptos iniciales hasta el mantenimiento y soporte, Enterprise Architect tiene las características precisa para diseñar y administrar su desarrollo e implementación.

✓ Rational Rose Enterprise Edition 2003

Rational Rose es la herramienta CASE desarrollada por los creadores de UML (Booch, Rumbaugh y Jacobson), que cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases y entregables.

El navegador UML de Rational Rose nos permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable. Facilita el desarrollo de un proceso cooperativo en el que todos los agentes tienen sus propias vistas de información (vista de Casos de Uso, vista Lógica, vista de Componentes y vista de Despliegue), pero utilizan un lenguaje común para comprender y comunicar la estructura y la funcionalidad del sistema en construcción.

Rational Rose Enterprise Edition es una herramienta que esta dentro del grupo de herramientas más técnicas debido a que se encarga de llevar a cabo tanto la automatización de los sistemas para la posterior generación de código (esto es, realización de los distintos diagramas y generación del código posterior), como para labores de ingeniería inversa(es decir, realización de los diagramas una vez conocido el código). La siguiente tabla muestra características de ingeniería de código en Rose.

Lenguaje	Rose
ANSI C++	Si
Visual C++	Si
Visual Basic 6	Si
Java	Si
C#	No
VB.NET	No
Delphi	No
J2EE/EJB	Si
CORBA	Si
Ada 83, Ada 95	Si
Bases de datos	Si. DB2, Oracle, SQL 92, SQL Server, Sybase
COM	Si. Solamente ingeniería reversa
Aplicaciones Web	Si

Tabla 2. Ingeniería de código en Rose

2.3. Tecnologías actuales

En el presente epígrafe se hace un análisis de las tendencias y tecnologías más utilizadas actualmente a nivel mundial en el desarrollo de aplicaciones de escritorio, así como sistemas gestores de bases de datos, metodologías de desarrollo de software, entornos de desarrollo y herramientas CASE.

2.3.1. Microsoft .NET

Microsoft .NET es un proyecto para crear una nueva plataforma de desarrollo de software con énfasis en transparencia de redes, con independencia de plataforma y que permita un rápido desarrollo de aplicaciones. Basado en esta plataforma, Microsoft intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el Sistema Operativo hasta las herramientas de mercado [20].

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Sun Microsystems.

A largo plazo Microsoft pretende reemplazar la Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) Win32 o Windows API con la plataforma .NET. Esto debido a que la API Win32 o Windows API fue desarrollada sobre la marcha, careciendo de documentación detallada, uniformidad y cohesión entre sus distintos componentes, provocando múltiples problemas en el desarrollo de aplicaciones para el sistema operativo Windows. La plataforma .NET pretende solventar la mayoría de estos problemas brindando un conjunto único y expansible con facilidad, de bloques interconectados, diseñados de forma uniforme y bien documentados, que permitan a los desarrolladores tener a mano todo lo que necesitan para producir aplicaciones sólidas.

Debido a las ventajas que la disponibilidad de una plataforma de este tipo puede darle a las empresas de tecnología y al público en general, muchas otras empresas e instituciones se han unido a Microsoft en el desarrollo y fortalecimiento de la plataforma .NET, ya sea por medio de la implementación de la plataforma para otros sistemas operativos aparte de Windows (Proyecto Mono de Ximian/Novell para Linux/MacOS X/BSD/Solaris [21]), el desarrollo de lenguajes de programación adicionales para la plataforma (ANSI C de la Universidad de Princeton, NetCOBOL de Fujitsu, Delphi de Borland, entre otros) o la creación de bloques adicionales para la plataforma (como controles, componentes y bibliotecas de clases adicionales); siendo algunas de ellas iniciativas de distribución gratuita bajo la licencia GNU.

Con esta plataforma Microsoft incursiona de lleno en el campo de los Servicios Web y establece el XML como norma en el transporte de información en sus productos y lo promociona como tal en los sistemas desarrollados utilizando sus herramientas.

.NET intenta ofrecer una manera rápida y económica pero a la vez segura y robusta de desarrollar aplicaciones - o como la misma plataforma las denomina, soluciones - permitiendo a su vez una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

✓ NET Framework

El "framework" o marco de trabajo, constituye la base de la plataforma .Net y denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones en entorno de ejecución distribuido.

Bajo el nombre .NET Framework o Marco de trabajo .NET se encuentran reunidas una serie de normas impulsadas por varias compañías además de Microsoft (como Hewlett-Packard , Intel, IBM, Fujitsu Software, Plum Hall, la Universidad de Monash e ISE), entre las cuales se encuentran:

- La norma que define las reglas que debe seguir un lenguaje de programación para ser considerado compatible con el marco de trabajo .NET (ECMA-335 [22], ISO/IEC 23271 [23])
  Por medio de esta norma se garantiza que todos los lenguajes desarrollados para la plataforma ofrezcan al programador un conjunto mínimo de funcionalidad, y compatibilidad con todos los demás lenguajes de la plataforma.
- La norma que define el lenguaje C# (ECMA-334 [24], ISO/IEC 23270 [25])
  Este es el lenguaje insignia del marco de trabajo .NET, y pretende reunir las ventajas de lenguajes como C/C++ y Visual Basic en un solo lenguaje.
- La norma que define el conjunto de funciones que debe implementar la librería de clases base (BCL por sus siglas en inglés) (incluido en ECMA-335 [22], ISO/IEC 23271 [23])
  Tal vez el más importante de los componentes de la plataforma, esta norma define un conjunto funcional mínimo que debe implementarse para que el marco de trabajo sea soportado por un sistema operativo. Aunque Microsoft implementó esta norma para su sistema operativo Windows, la publicación de la norma abre la posibilidad de que sea implementada para cualquier otro sistema operativo existente o futuro, permitiendo que las aplicaciones corran sobre la plataforma independientemente del sistema operativo para el cual haya sido implementada. El Proyecto Mono emprendido por Ximian pretende realizar la implementación de la norma para varios sistemas operativos adicionales bajo el marco de código abierto [26].

Los principales componentes del marco de trabajo son:

- El conjunto de lenguajes de programación.
- La Biblioteca de Clases Base o BCL.
- El Entorno Común de Ejecución para Lenguajes o CLR por sus siglas en inglés.

Debido a la publicación de la norma para la infraestructura común de lenguajes (CLI por sus siglas en inglés), el desarrollo de lenguajes se facilita, por lo que el marco de trabajo .NET soporta ya más de 20 lenguajes de programación y es posible desarrollar cualquiera de los tipos de aplicaciones soportados en la plataforma con cualquiera de ellos, lo que elimina las diferencias que existían entre lo que era posible hacer con uno u otro lenguaje.

Algunos de los lenguajes desarrollados para el marco de trabajo .NET son: C#, Visual Basic, C++, J#, Perl, Python, Fortran y Cobol.NET.

Figura 4. Diagrama detallado del Marco de Trabajo .NET.

✓ Common Language Runtime (CLR)

El CLR es el verdadero núcleo del Framework de .NET, entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes, ampliando el conjunto de servicios del sistema operativo (Windows 2000 y Windows 2003 Server).

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .NET en un código intermedio (MSIL, Microsoft Intermediate Lenguaje), similar al BYTECODE de Java. Para

generar dicho código el compilador se basa en el Common Language Specification (CLS) que determina las reglas necesarias para crear ese código MSIL compatible con el CLR.

Para ejecutarse se necesita un segundo paso, un compilador JIT (Just-In-Time) es el que genera el código máquina real que se ejecuta en la plataforma del cliente.

De esta forma se consigue con .NET independencia de la plataforma hardware, que no de sistema operativo.

La compilación JIT la realiza el CLR a medida que el programa invoca métodos, el código ejecutable obtenido, se almacena en la memoria caché del ordenador, siendo recompilado de nuevo sólo en el caso de producirse algún cambio en el código fuente.

Figura 5. Diagrama de la estructura interna del Entorno Común de Ejecución para Lenguajes.

✓ Biblioteca de Clases Base de .NET

La Biblioteca de Clases Base (BCL por sus siglas en inglés) maneja la mayoría de las operaciones básicas que se encuentran involucradas en el desarrollo de aplicaciones, incluyendo entre otras:

- Interacción con los dispositivos periféricos.
- Manejo de datos (ADO.NET).
- Administración de memoria.
- Cifrado de datos.
- Transmisión y recepción de datos por distintos medios (XML, TCP/IP).
- Administración de componentes Web que corren tanto en el servidor como en el cliente (ASP.NET).
- Manejo y administración de excepciones.
- Manejo del sistema de ventanas.
- Herramientas de despliegue de gráficos (GDI+).
- Herramientas de seguridad e integración con la seguridad del sistema operativo.
- Manejo de tipos de datos unificado.
- Interacción con otras aplicaciones.
- Manejo de cadenas de caracteres y expresiones regulares.
- Operaciones aritméticas.
- Manipulación de fechas, zonas horarias y periodos de tiempo.
- Manejo de arreglos de datos y colecciones.
- Manipulación de archivos de imágenes.
- Aleatoriedad.
- Generación de código.
- Manejo de idiomas.
- Auto descripción de código.
- Interacción con el API Win32 o Windows API.
- Compilación de código.

Esta funcionalidad se encuentra organizada por medio de espacios de nombres jerárquicos.

La Biblioteca de Clases Base se clasifica, en tres grupos clave:

- ASP.NET y Servicios Web XML.
- Windows Forms.
- ADO.NET.

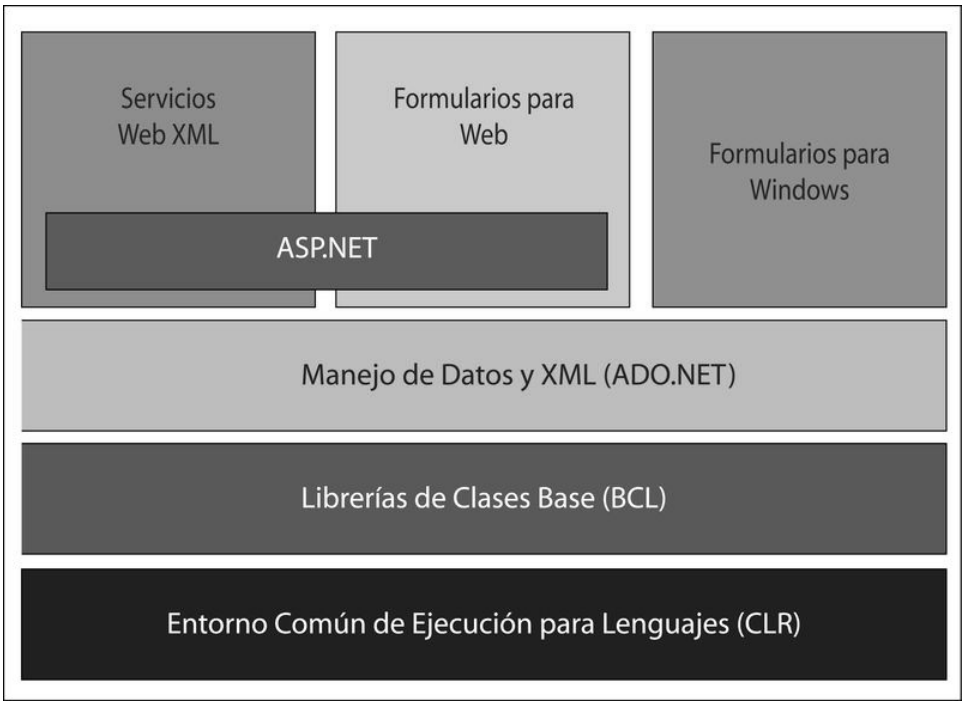

Figura 6. Diagrama básico de la Biblioteca de Clases Base.

✓ Ensamblados

Los ensamblados son ficheros con forma de EXE o DLL que contienen toda la funcionalidad de la aplicación de forma encapsulada.

Con los ensamblados ya no es necesario registrar los componentes de la aplicación.

Figura 7. Diagrama interno de un Ensamble .NET.

2.3.2. Java 2, Enterprise Edition

J2EE son las siglas de Java 2, Enterprise Edition que es la edición empresarial del paquete Java creada y distribuida por Sun Microsystems. Comprenden un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales. Debido a que J2EE no deja de ser un estándar, existen otros productos desarrollados a partir de ella aunque no exclusivamente [20].

Algunas de sus funcionalidades más importantes son:

- Acceso a base de datos (JDBC).
- Utilizado por BEA, IBM, Oracle, Sun, y Apache Tomcat entre otros.
- Utilización de directorios distribuidos (JNDI).
- Acceso a métodos remotos (RMI/CORBA).
- Funciones de correo electrónico (JavaMail).
- Aplicaciones Web (JSP y Servlet).

- Uso de Beans, etc.

La plataforma Java 2, Enterprise Edition es fruto de la colaboración de Sun con los líderes del sector del software empresarial (IBM, Apple, Bea Systems, Oracle, Inprise, Hewlett-Packard, Novell, etc.) para definir una plataforma robusta y flexible orientada a cubrir las necesidades empresariales en e-business y business-to-business.

✓ Especificaciones J2EE

Java 2, Enterprise Edition, aprovecha muchas de las características de la plataforma Java, como la portabilidad "Write Once, Run Anywhere", el Application Program Interface (API) JDBC para el acceso a bases de datos, las tecnología CORBA para la interacción con los recursos existentes de la empresa y un modelo de seguridad que protege los datos incluso en la aplicaciones para Internet. Sobre esta base, Java 2, Enterprise Edition añade el soporte completo para componentes Enterprise Java Beans, el API Java Servlets y la tecnología JavaServer Pages. El estándar J2EE incluye todas las especificaciones y pruebas de conformidad que permiten la portabilidad de las aplicaciones a través de la amplia gama de sistemas empresariales compatibles con J2EE.

J2EE esta basado en la arquitectura del lado del servidor (Served-based). Este tipo de arquitectura concentra la mayoría de los procesos de la aplicación en el servidor o en un pedazo de este. Este tipo de arquitectura tiene dos ventajas críticas en comparación con los otros tipos, estos son:

- Múltiples Clientes: Una arquitectura basada en el servidor requiere una clara separación entre la capa cliente (interfaz) y la capa servidor, en la cual se realizan los procesos de la aplicación. Esto permite que una simple aplicación soporte simultáneamente clientes con distintos tipos de interfaces, incluyendo poderosas interfaces gráficas para equipos corporativos, interfaces multimedia interactivas para usuarios con conexiones de alta velocidad, interfaces eficientes basadas en texto para usuarios con conexiones de baja velocidad, etc.
- Operaciones robustas: Una arquitectura basada en el servidor soporta escalabilidad, confiabilidad, disponibilidad y recuperabilidad. Aplicaciones basadas en el servidor pueden ser divididas y distribuidas en múltiples procesadores. Componentes de la aplicación pueden ser replicados para dar soporte a caídas instantáneamente.

La plataforma de J2EE provee un conjunto de APIs de java y servicios necesarios para el soporte de aplicaciones para empresas. La plataforma completa puede ser implementada en un solo sistema, o la

plataforma de servicios puede ser distribuida a través de varios sistemas, pero todas las APIs especificadas deben ser incluidas en alguna parte del sistema completo.

2.3.3. Microsoft SQL Server

Microsoft SQL Server es un sistema de gestión de bases de datos relacionales (SGBD) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea [20].

Entre sus características figuran:

- Soporte de transacciones.
- Gran estabilidad.
- Gran seguridad.
- Escalabilidad.
- Soporta procedimientos almacenados.
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
- Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo accedan a la información.
- Además permite administrar información de otros servidores de datos.

Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños.

Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle o Sybase.

Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa una potente base de datos (Microsoft SQL Server) con un entorno de desarrollo cómodo y de alto rendimiento (VBA Access) a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows.

Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para la mayoría de las plataformas de desarrollo, incluyendo .NET.

Microsoft SQL Server, al contrario de su más cercana competencia, no es multiplataforma, ya que sólo está disponible en Sistemas Operativos de Microsoft.

2.3.4. Oracle

Oracle es un sistema de administración de base de datos (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), fabricado por Oracle Corporation.
Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando su:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Es multiplataforma.

Su mayor defecto es su enorme precio, que es de varios miles de euros (según versiones y licencias). Otro aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.
Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySql o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo Linux.

2.3.5. NHibernate

NHibernate es un framework de Object-Relational-Mapping open-source que resuelve en forma automática la persistencia de mis objetos de dominio .NET. NHibernate esta basado en el popular framework open-source Hibernate surgido en la comunidad Java en el año 2002. [27]

NHibernate es la conversión de Hibernate de lenguaje Java a C# para su integración en la plataforma .NET. Al igual que muchas otras herramientas libres para esta plataforma, NHibernate también funciona en Mono.

NHibernate permite el acceso a datos asegurándole al desarrollador de que su aplicación es agnóstica en cuanto al motor de base de datos a utilizar en producción, pues NHibernate soporta los más habituales en

el mercado: MySQL, PostgreSQL, Oracle, MS SQL Server, etc. Sólo se necesita cambiar una línea en el fichero de configuración para que podamos utilizar una base de datos distinta.

2.3.6. TierDeveloper 4.0

TierDeveloper es una herramienta de generación de código para mapeos de objetos en la plataforma .NET tanto para aplicaciones ASP.NET como para aplicaciones Windows Forms. TierDeveloper permite disminuir sustancialmente el tiempo de desarrollo en dependencia de de la complejidad de la aplicación que se vaya a desarrollar. Esta herramienta se integra a VS. Net 2003 y 2005.

2.4. Conclusiones

En este capítulo, se ha tratado de hacer un análisis de las principales herramientas de desarrollo, que apoyan la productividad y la automatización de la codificación. Se definieron las tecnologías y herramientas a utilizar en el diseño de la solución.

Se arribaron a las siguientes conclusiones:

- Utilizar la plataforma Microsoft .NET, específicamente Microsoft Visual Studio .NET 2003 y como lenguaje de programación Microsoft Visual C# .NET los cuales brindan una gran productividad elemento necesario dado el corto periodo de entrega del software.
- Utilizar el sistema gestor de base de datos Oracle 10g Standard Edition One debido a que brinda un rendimiento máximo, su escalabilidad y capacidad de almacenamiento, respuesta rápida y seguridad.
- Emplear la herramienta CASE Enterprise Arquitect 6.1 para crear los diagramas del diseño del sistema, utilizando el lenguaje de modelado UML 2.0.
- Utilizar la herramienta TierDeveloper 4.0 para generar las capas de acceso a datos.

Capítulo 3: Diseño del Sistema

3.1. Introducción

En capítulos anteriores se llego a la conclusión de adoptar la metodología RUP como guía para la construcción del diseño de este sistema, que forma parte de la solución de automatización planteada para los Registros Mercantiles e Inmobiliario del Ministerio del Poder Popular para Relaciones Interiores y Justicia de la República Bolivariana de Venezuela. El objetivo de este capítulo es desarrollar los artefactos más importantes para la actividad de diseño, partiendo de la fundamentación del tema y el estudio del arte de las diferentes herramientas, metodologías, frameworks y tecnologías existentes.

¿Pero exactamente que es el Diseño de Software Orientado a Objetos? Según define Pressman, requiere la definición de una arquitectura de software multicapa, la especificación de subsistemas que realizan funciones necesarias y proveen soporte de infraestructura, una descripción de objetos (clases), que son los bloques de construcción del sistema y una descripción de los mecanismos de comunicación, que permiten que los datos fluyan entre las capas, subsistemas y objetos.

3.2. Entradas del Proceso de Diseño

El punto de partida para la elaboración de este diseño fueron las especificaciones de los casos de uso del sistema, los requisitos asociados a estos y las reglas generales del negocio. El análisis de los requisitos es una tarea vital dentro del proceso de desarrollo de software porque cubre el espacio que existe entre las ideas que forman la definición del software a nivel de procesos de negocio y el diseño del software. Este análisis permite detallar en las características funcionales del software así como en algunas restricciones que deba cumplir el sistema.

3.2.1. Principales Casos de Uso del Sistema y requisitos asociados

El trabajo mostrado en el presente epígrafe constituye una entrada en el desarrollo del diseño propuesto para las oficinas legales de Venezuela. Debido a la cantidad de casos de usos definidos en su totalidad para el sistema, solamente se relacionan aquí los casos de usos arquitectónicamente significativos (son aquellos que presentan un alto grado de prioridad para el cliente y los más complejos) y los requisitos asociados, identificados por los analistas, así como las reglas del negocio.

✓ Gestionar Estados de Cuenta

Este caso de uso consiste en gestionar los estados de cuentas recibidos por el banco para cada una de las cuentas bancarias. Intervienen en este proceso los funcionarios de Administración Contable de los Registros o el funcionario de Administración Contable de la oficina nacional del Servicio Autónomo.

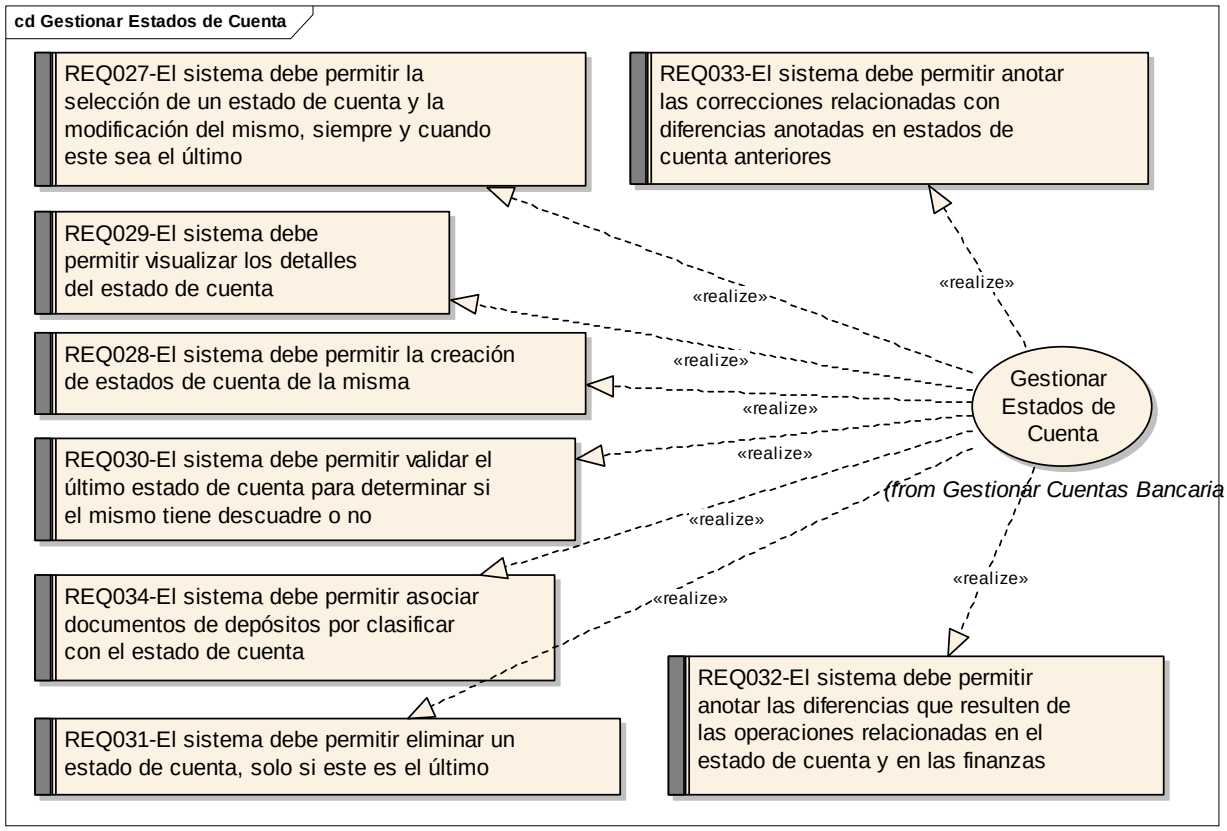

Figura 8. Caso de uso Gestionar Estado de Cuenta y los requisitos asociados.

✓ Cerrar Cuentas Nominales

Este caso de uso consiste en generar de forma automática un comprobante para llevar el saldo de las cuentas nominales a valor cero. Intervienen en este proceso los funcionarios de Administración Contable de los Registros o el funcionario de Administración Contable de la oficina nacional del Servicio Autónomo.

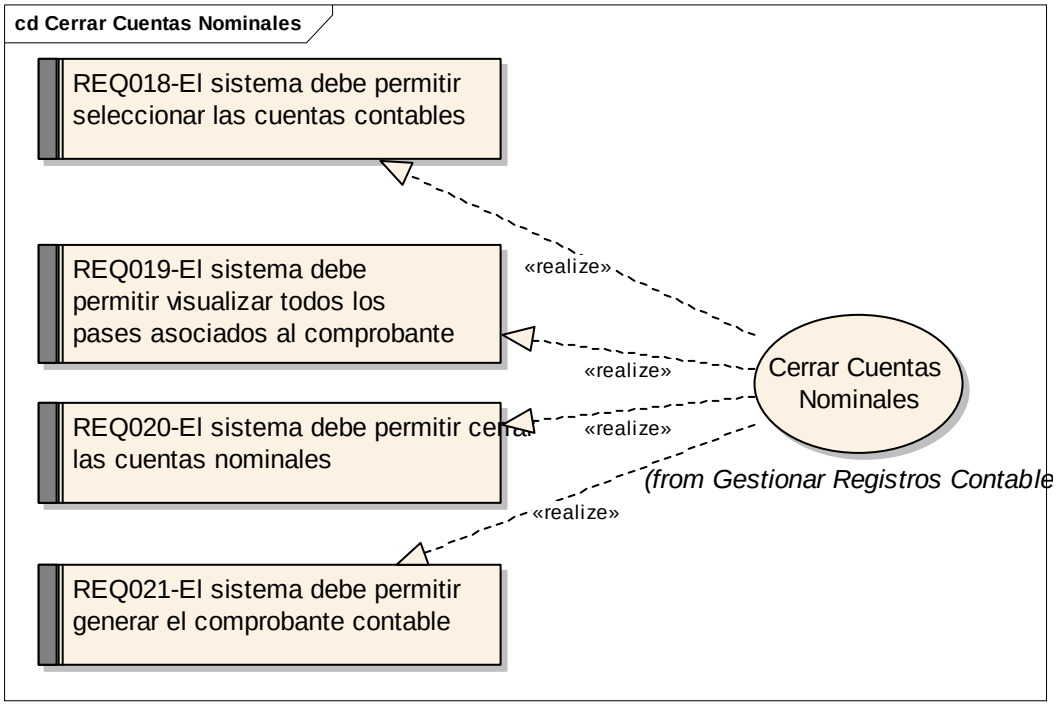

Figura 9. Caso de uso Cerrar Estado de Cuenta y los requisitos asociados.

✓ Gestionar Registro de Comprobantes

Este caso de uso consiste en mostrar la relación de todos los comprobantes contables existentes en el sistema en cada uno de los períodos seleccionados y en dependencia además del Origen. En él se pueden realizar las funcionalidades de visualizar la relación de los comprobantes para ser impresa o exportada hacia otro formato, crear nuevos comprobantes, ver el contenido de un comprobante para ser impreso o exportado, abrir un comprobante de origen contabilidad para ser modificado y anular un comprobante de origen Contabilidad si fuese necesario. Es iniciado por el funcionario de Administración Contable de la oficina o el funcionario de Administración Contable de Servicio Autónomo.

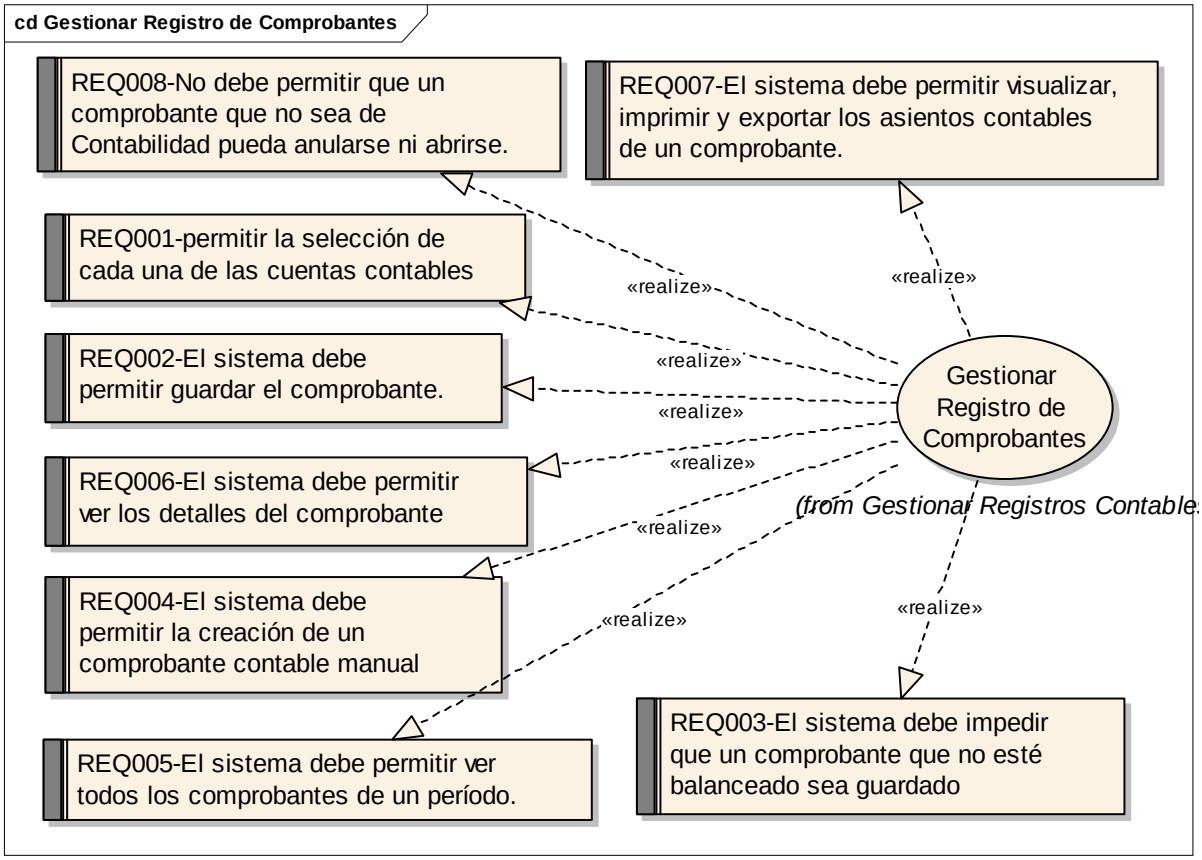

Figura 10. Caso de uso Gestionar Registro de Comprobantes y los requisitos asociados.

✓ Configurar Reglas Contables

Este caso de uso consiste en configurar diferentes reglas contables necesarias para efectuar las operaciones del módulo de Contabilidad y Finanzas. Es iniciado por el funcionario de Administración Contable de la oficina o el funcionario de Administración Contable del Servicio Autónomo. Este caso de uso está asociado a la oficina de Registros Públicos, Mercantil y al Servicio Autónomo.

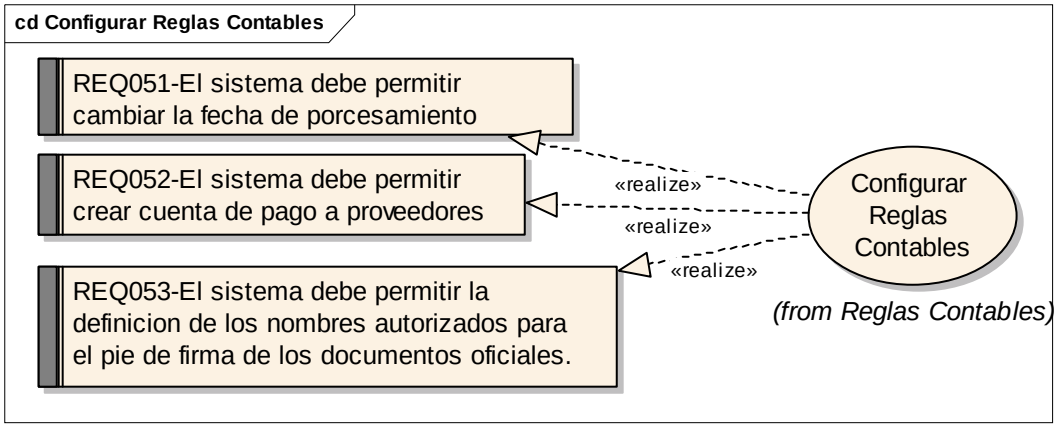

Figura 11. Caso de uso Configurar Reglas Contables y los requisitos asociados.

✓ Gestionar Factura

Este caso de uso consiste en fijar las obligaciones de pago por factura recibida de los proveedores cuando se hace una compra en los registros, este caso de uso permite que sea generado el comprobante contable asociado a esta operación financiera. Es iniciado por el funcionario de Administración Contable de las oficinas regionales o del Servicio Autónomo.

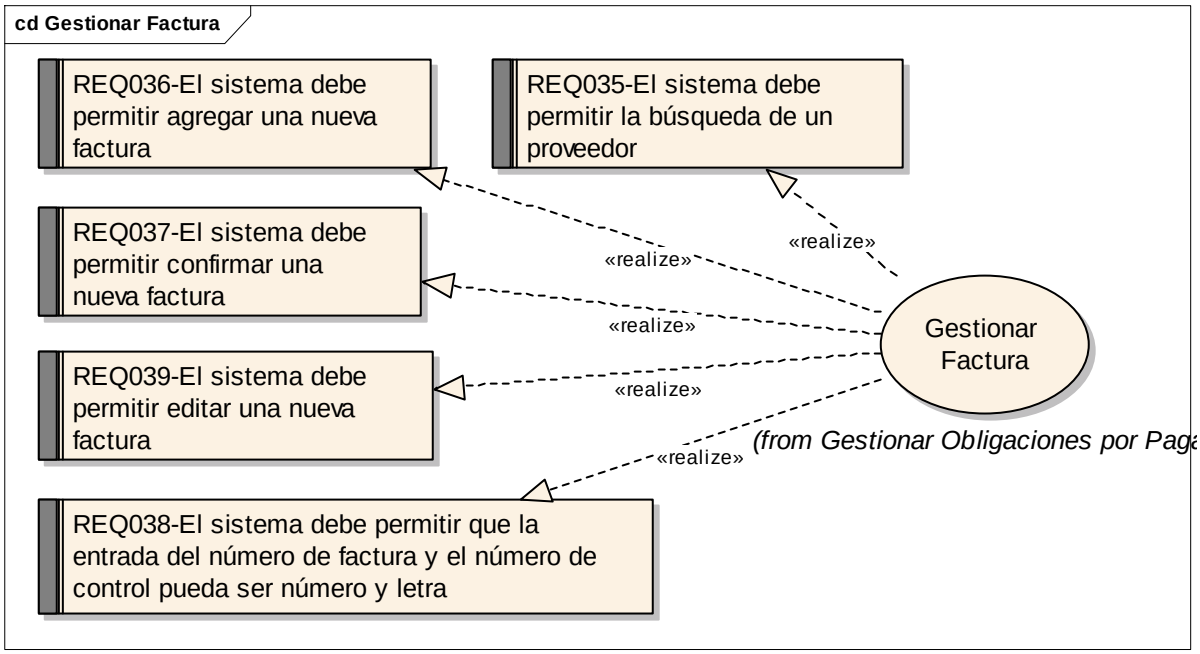

Figura 12. Caso de uso Gestionar Factura y los requisitos asociados.

✓ Registrar Asignación de Presupuesto

Este caso de uso consiste en registrar en el sistema las operaciones de Asignación de Presupuesto en base al aprobado para cada entidad en el presupuesto. Es iniciado por el funcionario de Administración Contable de la oficina o el funcionario de Administración Contable del Servicio Autónomo. Este caso de uso está asociado a las oficinas de los Registros Públicos, Mercantil, Inmobiliarios y Servicio Autónomo.

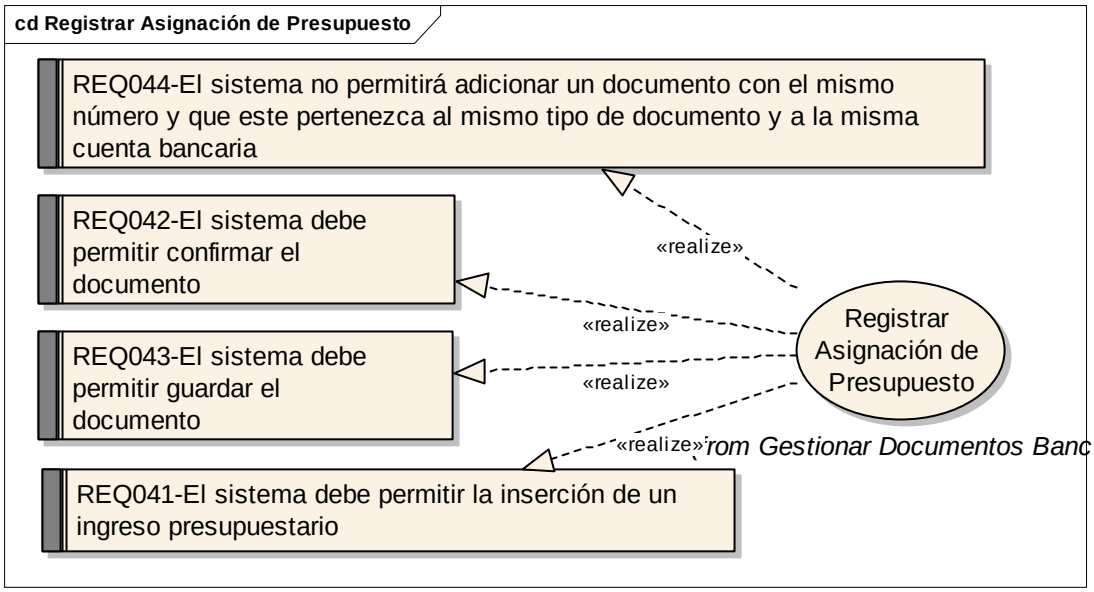

Figura 13. Caso de uso Registrar Asignación de Presupuesto y los requisitos asociados.

✓ Registrar Egresos de Caja

Este caso de uso consiste en realizar todas las operaciones de egresos que se ejecutan en la caja chica donde se encuentra el dinero en efectivo, las mismas están relacionadas únicamente con pagos menores. Es iniciado por el funcionario de Administración Contable de la oficina o el funcionario de Administración Contable del Servicio Autónomo.

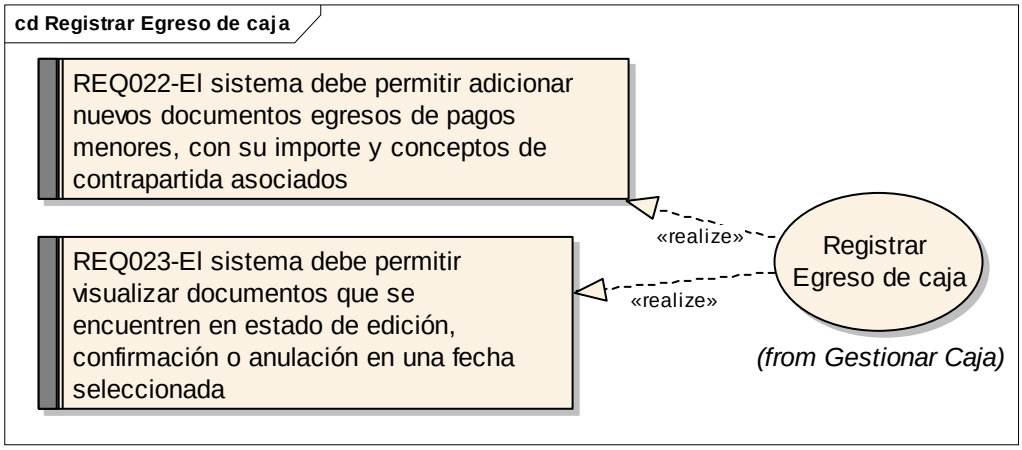

Figura 14. Caso de uso Registrar Egreso de Caja y los requisitos asociados.

✓ Registrar Pago de Obligaciones

Este caso de uso consiste en la emisión de documentos de Pago de Obligaciones ya sea de los proveedores o de los Entes Recaudadores. Es iniciado por el funcionario de Administración Contable de la oficina o el funcionario de Administración Contable del Servicio Autónomo.

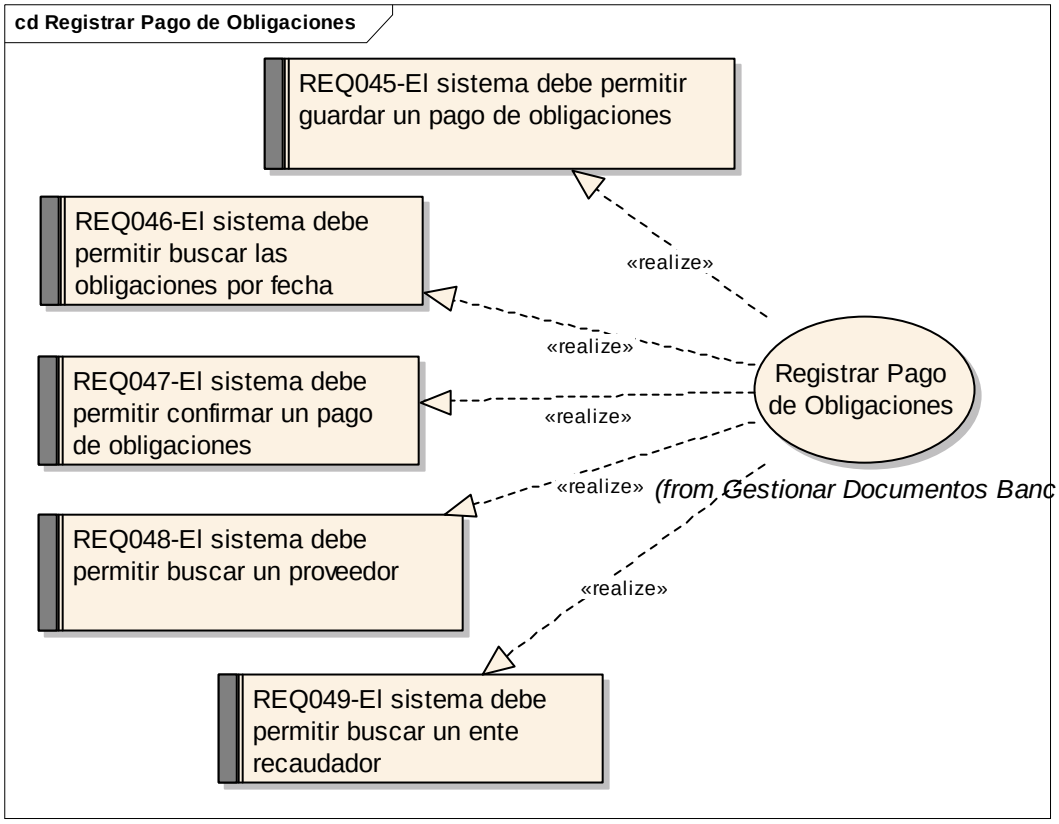

Figura 15. Caso de uso Registrar Pago de Obligaciones y los requisitos asociados.

✓ Visualizar Reporte de Balance General

Este caso de uso consiste en realizar un Reporte Contable donde se muestran las cuentas que pertenecen a los activos, los pasivos, y las de patrimonio. El saldo de los activos debe ser igual a la suma del saldo de los pasivos más las del patrimonio, partiendo de la información que van generando los comprobantes. Este reporte se puede visualizar también en proyección, donde se muestra el estado de Ganancia o Pérdida, cuando no se hayan cerrado las cuentas nominales. En este caso de uso se contempla la posibilidad de visualizar e imprimir el reporte. Es iniciado por el funcionario de Administración Contable de la oficina o el funcionario de Administración Contable del Servicio Autónomo.

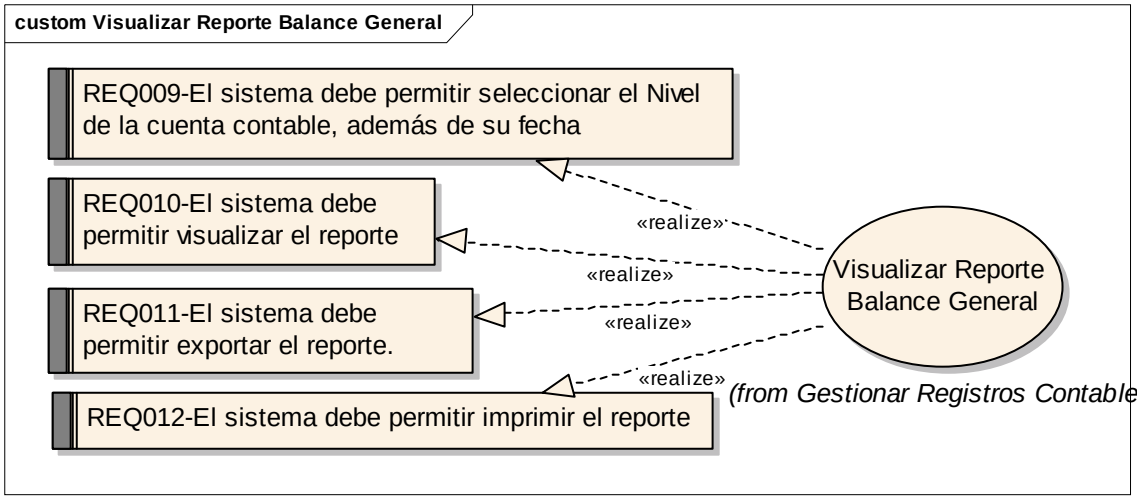

Figura 16. Caso de uso Visualizar Reporte Balance General y los requisitos asociados.

✓ Registrar Depósitos por Clasificar

Este caso de uso se refiere a los depósitos por clasificar (Comprobantes bancarios que hayan sido emitidos pero que no han sido presentados). Es iniciado por el funcionario contabilidad de oficina o el funcionario contabilidad de Servicio Autónomo. Este caso de uso está asociado a la oficina de Registros Públicos, Mercantil y Servicio Autónomo.

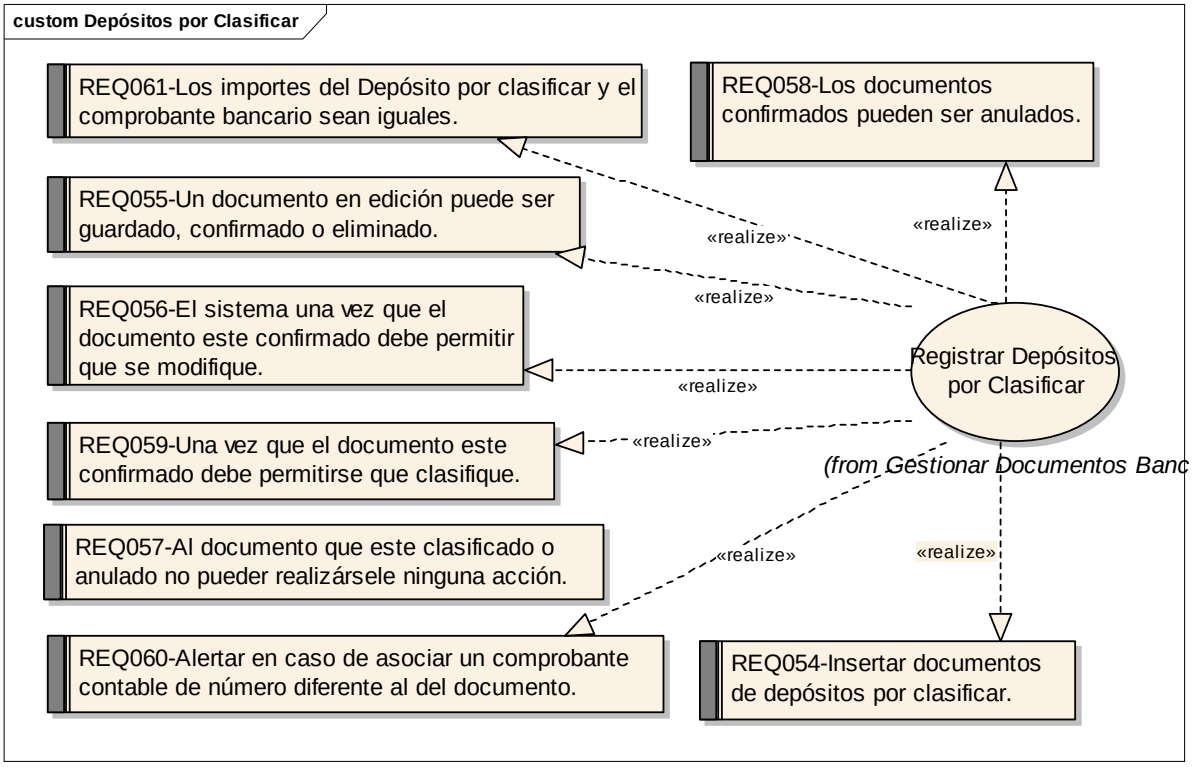

Figura 17. Caso de uso Depósitos por Clasificar y los requisitos asociados.

Gestionar Notas de Crédito y Débito

Este caso de uso consiste en Gestionar las Notas de Crédito y Débito. Es iniciado por el funcionario de Administración Contable de oficina o el funcionario de Administración Contable de Servicio Autónomo. Este caso de uso está asociado a la oficina de Registros Públicos, Mercantil y Servicio Autónomo.

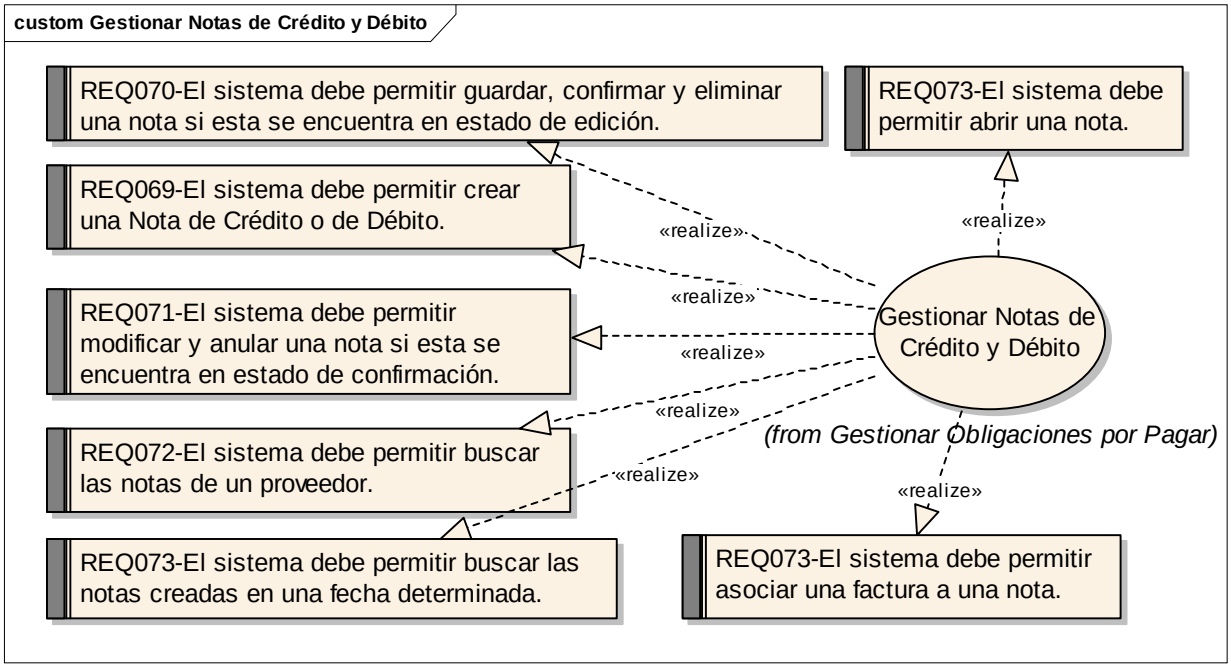

Figura 18. Caso de uso Gestionar Notas de Crédito y Débito y los requisitos asociados

✓ Registrar Créditos Adicionales

Este caso de uso consiste en registrar en el sistema los Créditos Adicionales que se asignan a Servicio Autónomo en cualquier momento del ejercicio fiscal. Es iniciado por el funcionario de Administración Contable del Servicio Autónomo. Este caso de uso está asociado a la oficina de Servicio Autónomo.

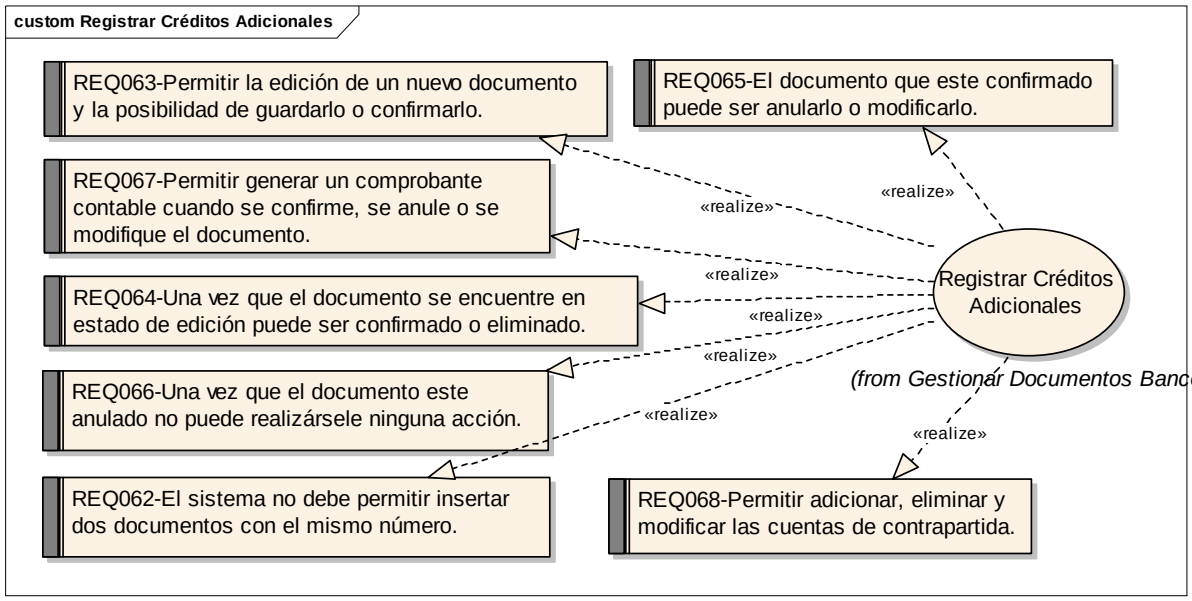

Figura 19. Caso de uso Registrar Créditos Adicionales y requisitos asociados.

3.2.2. Reglas del Negocio

Las reglas del negocio constituyeron un artefacto de entrada importante para la construcción de este diseño. Especifican detalles de como debe funcionar el sistema correctamente. Estas reglas del negocio permitieron identificar y gestionar correctamente algunas entidades claves en este entorno de negocio, los estados por el que transitan y los comportamientos y restricciones del diseño asociadas a ellas. A continuación una muestra de algunas de las reglas más importantes:

- ✓ Todo asiento contable que se registre debe estar correctamente balanceado, es decir que la suma de todas las cuentas por el Crédito deben ser igual a la suma de todas las cuentas por el Débito.
- ✓ Un asiento que se registre no puede ser eliminado, sino que se revierte cualquier error a partir de otro asiento contable.
- ✓ La contabilización debe estar al día.
- ✓ No se puede extraer más dinero del que tiene el fondo de caja.
- ✓ Cada operación financiera debe emitir y registrar de manera inmediata un asiento contable.

- ✓ Se deben mantener conciliadas las Cuentas Bancarias.
- ✓ Una cuenta contable correspondiente al efectivo en banco no puede estar asociada a más de una Cuenta Bancaria y estas cuentas contables no pueden ser registradas en operaciones que no estén involucradas en las operaciones financieras.
- ✓ En las operaciones de egresos bancarios no pueden existir sobregiros.

3.3. Modelo de Diseño

3.3.1. Capas y Subsistemas del Diseño

Partiendo de la idea de Pressman, de que el Modelo de Diseño sirve como anteproyecto para la construcción del software, podemos deducir fácilmente que su realización se torna bastante complicada, atendiendo a diversos factores y restricciones del entorno en que se este desarrollando. Gamma ofrece una idea sobre el tema:

El diseño de software orientado a objetos es difícil, y el diseño de software reusable orientado a objetos es aun más difícil. Se deben identificar los objetos pertinentes, clasificarlos dentro de las clases en la granularidad correcta, definir interfaces de clases y jerarquías de herencia y establecer relaciones clave entre ellos. El diseño debe ser específico al problema que se tiene entre manos, pero suficientemente general para adaptarse a problemas y requerimientos futuros. Además se debería evitar el rediseño, o por lo menos minimizarlo. Los diseñadores experimentados en OO dicen que un diseño reusable y flexible es difícil, si no imposible de obtener bien, la primera vez. Antes de que un diseño sea terminado, usualmente tratan de reutilizarlo varias veces, modificándolo cada vez. [28]

Siguiendo la línea de esta definición, fue concebido un Diseño estructurado en cinco capas fundamentales:

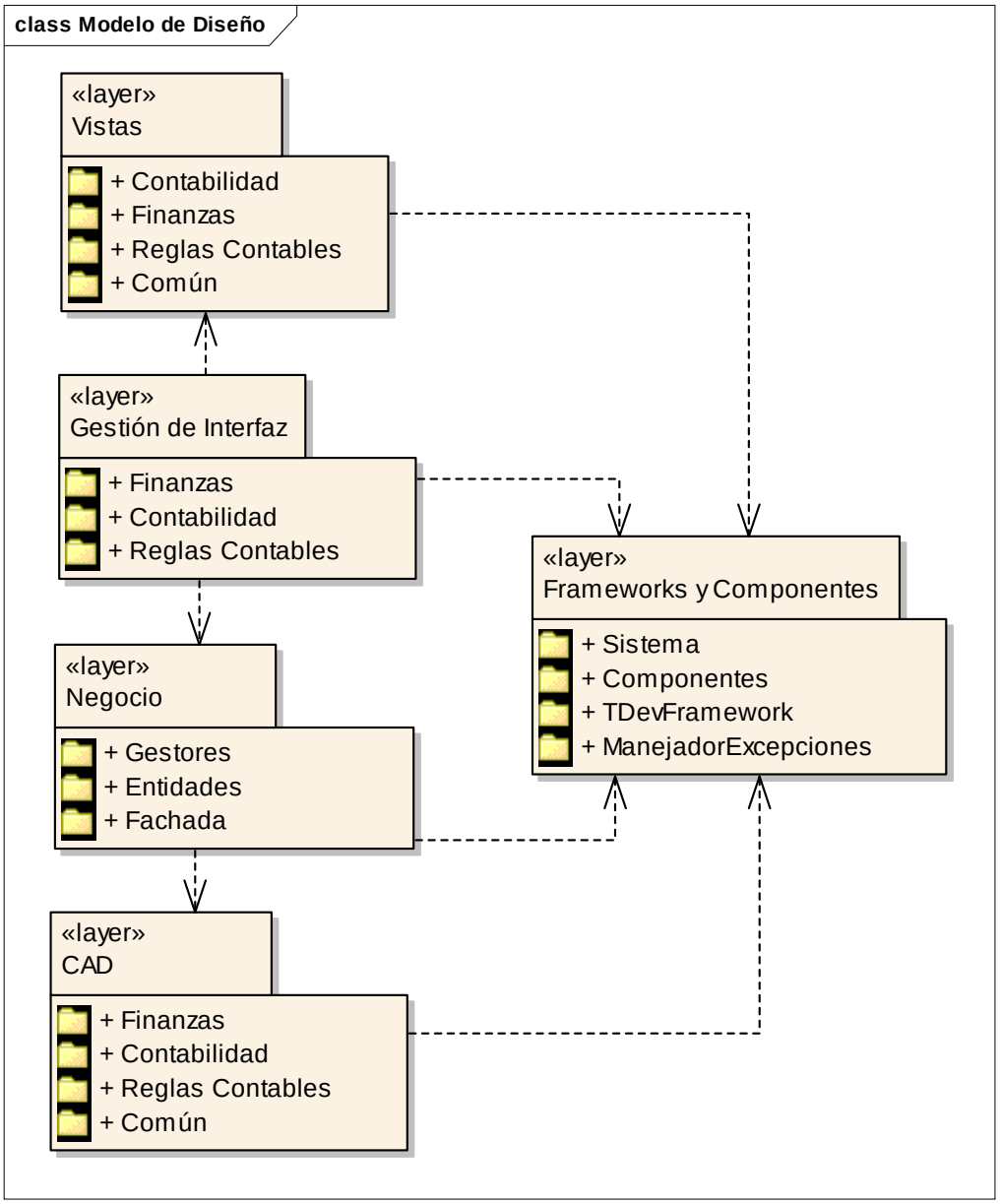

Figura 20. Modelo de diseño. Estructura en Capas.

✓ Capa Vistas

La capa de Vistas responde a un patrón estándar, todos los subsistemas tienen prácticamente la misma forma con vistas a facilitar el trabajo con ellos y la estandarización del sistema completo. Su uso se basa en la explotación de la componente interfaz que realmente provee un framework facilitando el desarrollo

del sistema. Este framework se basa en acciones y cada acción que se corresponda con funcionalidades de captura o visualización de datos tiene asociado un formulario cuya forma depende de la funcionalidad específica para la cual fue concebida dicha acción.

✓ Capa Gestión de interfaz

Su uso se basa en la explotación de la componente interfaz que realmente provee un framework facilitando el desarrollo del sistema basado en acciones.

¿Qué es una acción?

Cada acción debe tener sentido semántico propio y completo es por eso que estas deben ser atómicas. Las acciones básicamente definen un bloque de código reusable por varias operaciones sobre el sistema. Las acciones pueden estar asociadas a vistas del sistema. Cada acción puede representar un estado del sistema que puede o no persistir. Una acción es una clase que representa precisamente la ejecución de alguna tarea concreta. Estas tareas no deben ser excesivamente complejas.

✓ Capa Negocio

Dentro de la capa de Negocio específicamente, existen otras tres capas de manera horizontal que completan y organizan toda la lógica de negocio y los datos que se manejan en el sistema. La capa de Gestores engloba todas las clases encargadas de los cálculos y procesos económicos necesarios para que el negocio funcione; La capa de Entidades incluye todas las clases que definen algún tipo de documento u otra información que se maneje en un entorno del negocio determinado; La Fachada aísla la Capa del Negocio de la persistencia de los Datos, brindándole independencia total al diseño de cualquier tecnología que se utilice para dicho proceso y un bajo acoplamiento en sentido general.

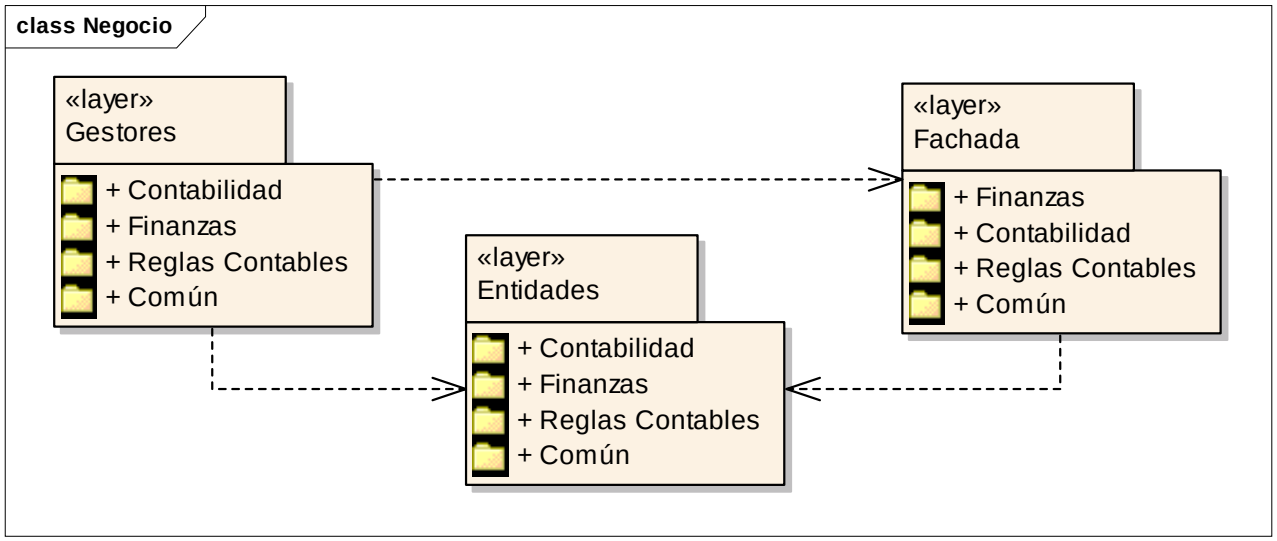

Figura 21. Capas definida dentro la capa de negocio.

✓ Capa Acceso a Datos

Esta capa fue generada con la herramienta TierDeveloper versión 4.0. Garantiza que la información fluya desde el negocio hasta la base de datos y viceversa. Mayormente están formadas por clases que representan los objetos mapeados de la Base de Datos y una Factoría que se encarga de gestionarlos. A continuación se ilustra un ejemplo de la estructura de una de estas Capas de Acceso a Datos.

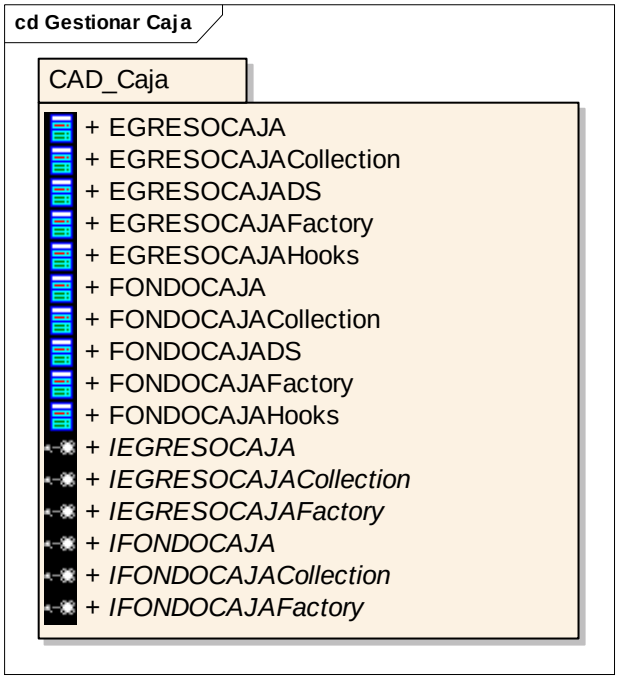

Figura 22. Clase Gestionar Caja de la capa acceso a datos.

✓ Frameworks y Componentes

En esta capa se ubican una serie de frameworks y componentes que se utilizan en el diseño. El framework **Sistema** es la base de casi de toda las capas pues en el existen funcionalidades que abarcan el manejo de las vistas a través de acciones, la seguridad y el control del acceso, manejo de excepciones a alto nivel, gestión de los temas de ayuda según el contexto del sistema y otras utilidades más especificas. El framework **ManejadorExcepciones** se emplea en todas las capas del sistema, pues captura y trata deforma personalizada todas las excepciones dependiendo del tipo, el contexto donde se haya capturado y las políticas de manejo establecidas previamente. El framework **TDevframework** es necesario para poder utilizar las Capas de Acceso a Datos que genera la herramienta TierDeveloper. En el paquete **Componentes** están todos los componentes diseñados especialmente para ser utilizados en las capas de presentación (Vistas y Gestión de Interfaz).

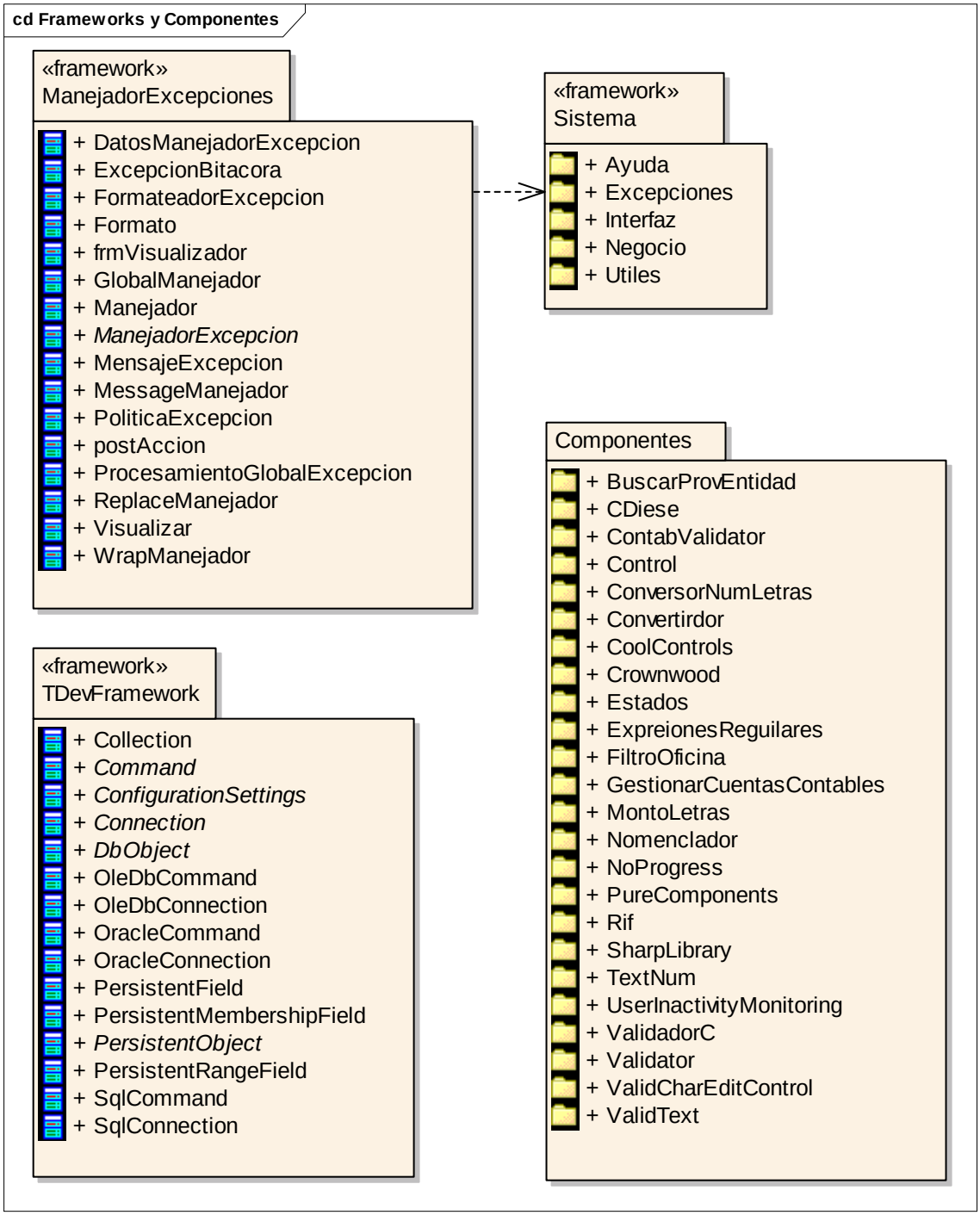

Figura 23. Capa de frameworks y componentes.

✓ Subsistemas

El problema global ha sido dividido según las áreas particulares de la Contabilidad y las Finanzas en varios subsistemas, permitiendo el desarrollo simultáneo y ágil de cada uno de ellos de manera independiente, quedando definidos de esta forma cuatro paquetes fundamentales que engloban a su vez otros más específicos:

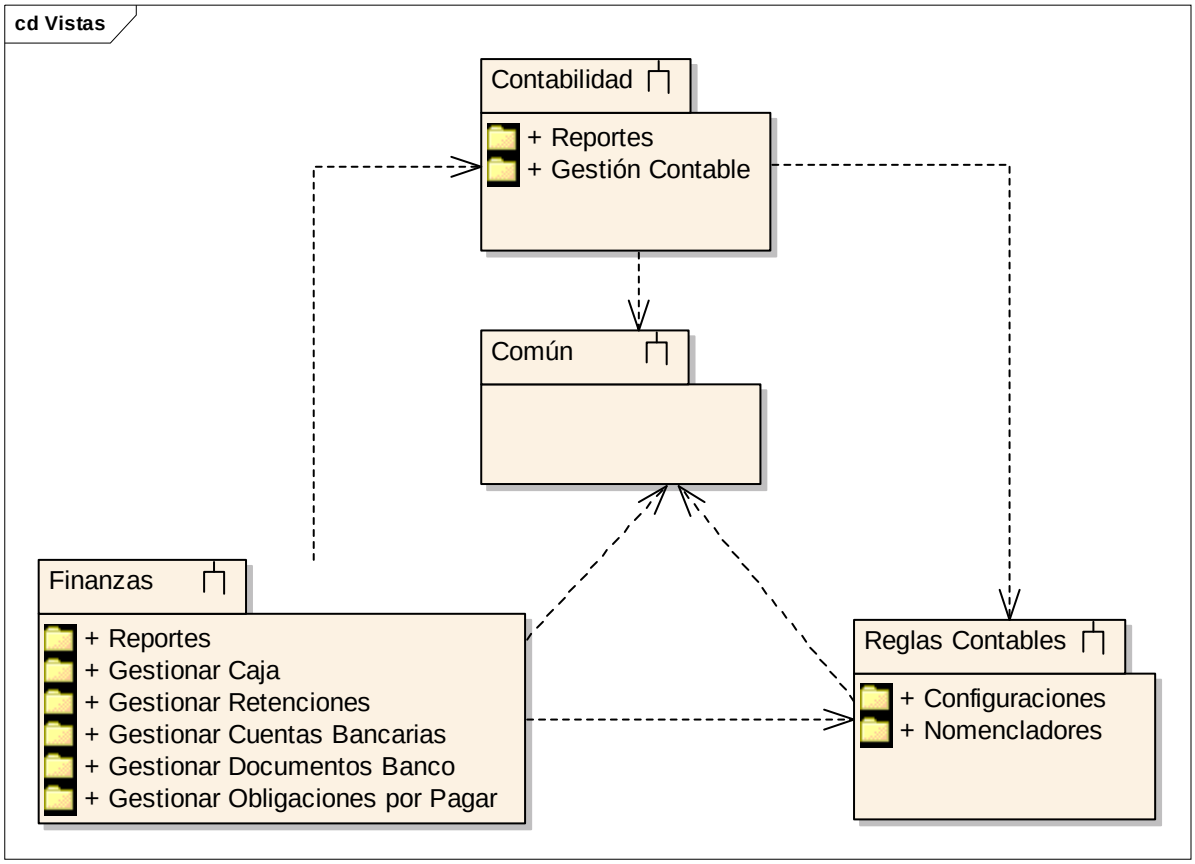

Figura 24. Vista de los subsistemas.

El paquete **Contabilidad** incluye todos los registros que generan las operaciones financieras en el sistema, esta información se recopila y organiza en libros contables que aparecen agrupados en un subsistema llamado **Reportes.** El otro subsistema dentro de este paquete es el de **Gestión Contable** que abarca lo referido a la generación, edición y anulación de Comprobantes Contables, incluyendo una vista general de todo el registro de comprobantes según la naturaleza de los mismos, además de lo inherente al cierre de cuentas nominales.

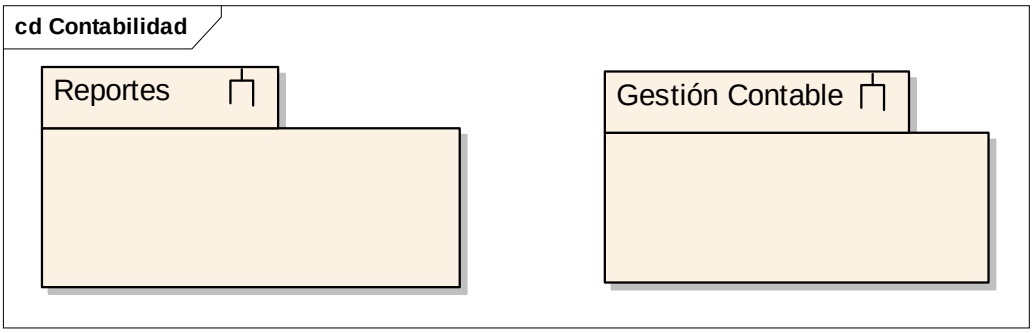

Figura 25. Subsistemas dentro del subsistema de contabilidad.

El paquete de **Finanzas** es un poco más amplio e incluye varios subsistemas como: el de **Reportes** donde aparecen vistas que muestran el registro general de documentos según el origen de estos, su estado y la fecha en que fueron generados; **Gestionar Caja** es otro de los subsistemas que incluye este paquete y tiene que ver con la definición del Fondo de Caja, los Egresos de Caja y el Reporte de los mismo; **Gestionar de Retenciones** es otro de los subsistemas y se encarga de la generación y edición de comprobantes de retención, de la gestión de las retenciones según la forma de pago y del reporte de retenciones por proveedor; **Gestionar Cuentas Bancarias** agrupa la administración y conciliación de las mismas así como la edición de los estados de cuentas de estas; La **Gestión de los Documentos de Banco** es el subsistema más complejo y tiene que ver directamente con todos los documentos primarios que se generan en el sistema tanto de ingresos como de egresos, incluyendo los reportes de los mismo; Por último pero por ello no menos importante la **Gestión de Obligaciones por Pagar** que controla todo el proceso de facturación y seguimiento de las obligación de pago contraídas con los proveedores así como la generación de notas de crédito o debito a estas facturas.

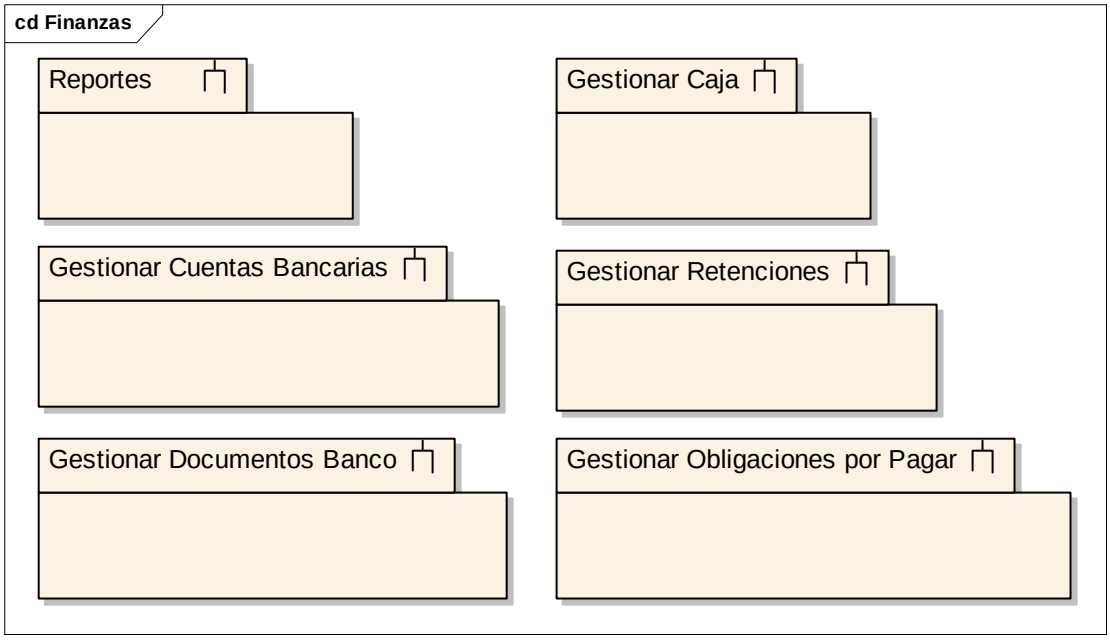

Figura 26. Subsistemas dentro del subsistema de finanzas.

El paquete de **Reglas Contables** tiene dos subsistemas fundamentales: **Las Configuraciones** que tienen que ver con todas las definiciones que se necesitan para que el proceso contable fluya de manera automática y que la gestión económica se haga más eficiente, esto incluye el control de la fecha de procesamiento contable, la configuración de los Ejercicios Fiscales y de los Períodos Contables, la especificación de reglas del negocio y el control del acceso a determinadas cuentas contables.

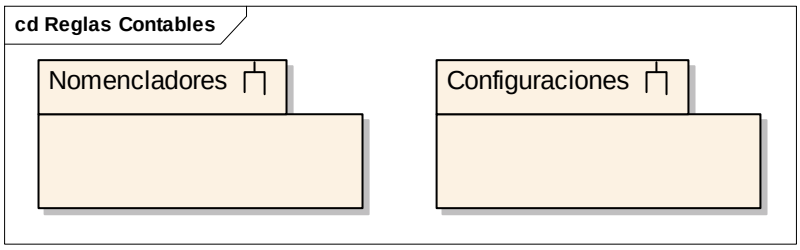

Figura 27. Subsistemas dentro del subsistema de reglas contables

Existe en este Diseño un subsistema especial llamado **Común** donde se agrupan todas clases que son compartidas por los demás subsistemas, facilitando en gran medida la reutilización de las mismas y dándole una alta cohesión a dicho diseño.

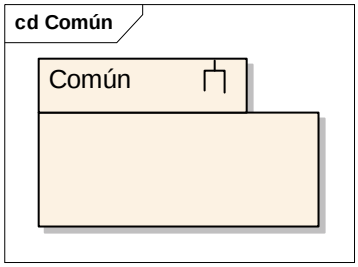

Figura 28. Subsistema Común

3.3.2. Clases del diseño

El Diseño de este sistema se ajusta al marco arquitectónico definido anteriormente, procurando siempre que el acoplamiento entre las Capas y los Subsistemas sea mínimo, pero que la distribución de las clases dentro de estos paquetes le de al sistema una alta cohesión. De esta forma y siguiendo los preceptos del buen diseño quedarían identificados cinco tipos de clases según el estrato de la arquitectura donde se encuentren:

- ✓ **Vistas o Formularios**

Estas son clases que contienen todos los controles y componentes necesarios para construir las vistas que se utilizan para la entrada de datos o para mostrar registros y reportes de ellos. Están nombradas de forma tal que se puedan identificar fácilmente, pues a su nombre se le anteponen las letras **frm**. Todos los atributos de estas clases que como se mencionó eran controles y componentes mayormente, son públicos de manera tal que las acciones que utilizan estas vistas puedan acceder a ellos en un momento determinado. Al carecer estos formularios de cualquier tipo de lógica de negocio o dependencia con este, pueden ser utilizados por varias acciones diferentes, que representan procesos de un entorno de negocio específicos.

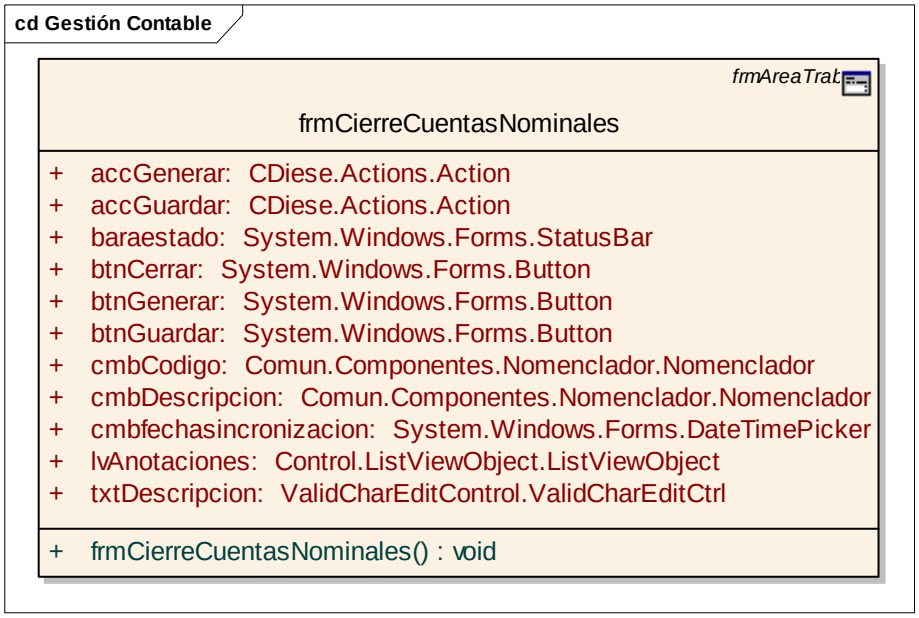

Figura 29. Clase del formulario frmCierreCuentasNominales.

De forma general estas vistas pueden heredar de tres tipos básicos de formularios que se incluyen dentro del framework **Sistema** y aparecen relacionados a continuación:

o **Sistema.Interfaz.frmAreaTrabajo**

Estas Interfaces se utilizan para mostrar registros de documentos u otros tipos de informaciones y constituyen el área de trabajo fundamental. Se muestran empotradas en la forma principal por cuestiones prácticas y de diseño gráfico, tienen una dimensión aproximada de (846, 713) píxel.

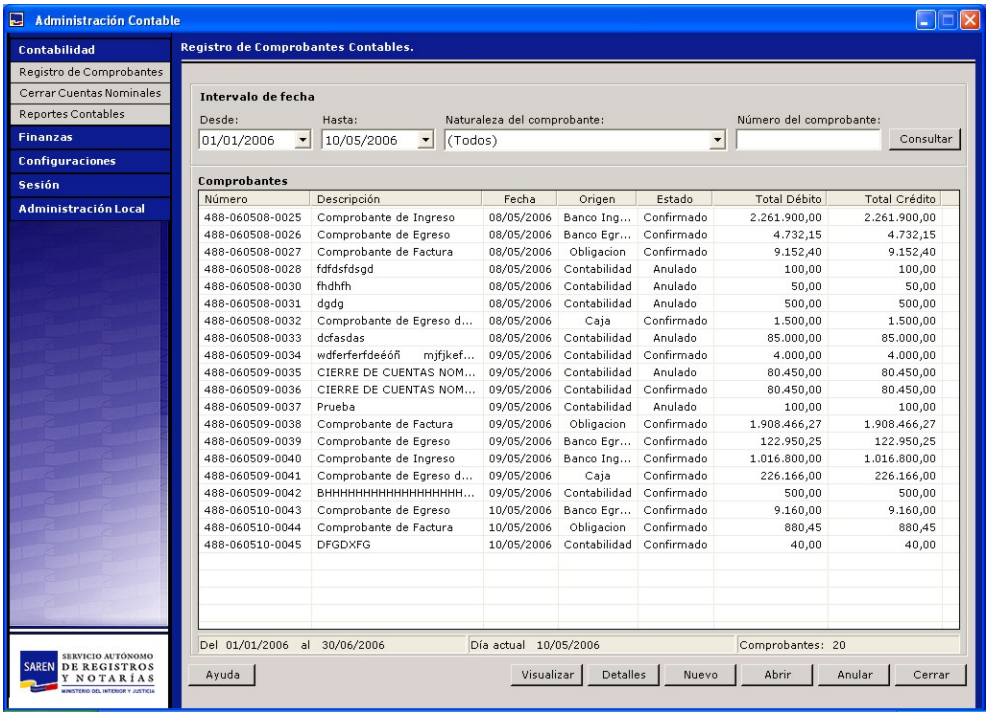

Figura 30. Interfaz Registro de Comprobantes Contables.

- **Sistema.Interfaz.frmConfirmar**

La función principal de este tipo de formularios es proveer el mecanismo para realizar preguntas y solicitar confirmaciones en determinadas situaciones de los procesos del negocio. Pueden ser configurados y adaptados para una situación específica según sea el caso. Sus dimensiones pueden ser variables.

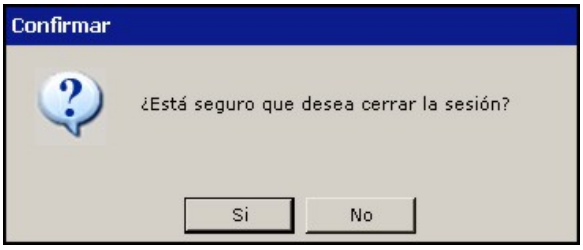

Figura 31. Interfaz frmConfirmar.

- **Sistema.Interfaz.frmMensaje**

Son utilizados con frecuencia para crear pequeños formularios que se superponen al área de trabajo que está activa en ese momento y se subordinan mayormente a estas. Su complejidad y dimensiones pueden variar dependiendo solo del uso especifico que se le de.

Figura 32. Formulario Anular Comprobante Contable.

✓ Acciones

Las acciones representan procesos específicos o pasos dentro de estos, generalmente tienen asociada un área de trabajo particular y pueden usar además, otros formularios auxiliares. Se identifican por su nomenclatura, pues su nombre comienza por las siglas **acc**. Estas clases representan la base del funcionamiento del sistema porque la gestión del acceso y control de este se realiza a través de roles bien definidos que están asociados a estas acciones. Pueden heredar de dos tipos de acciones genéricas:

- **Sistema.Interfaz.Acciones.AccionSegura**

Se realiza un control estricto con el usuario que esta accediendo a ellas y las acciones que realiza sobre la misma. Deben ser asociadas a los roles pertinentes a través de configuraciones de la aplicación.

- **Sistema.Interfaz.Acciones.Accion**

El acceso a estas acciones no se controla, simplemente representan proceso simples del negocio que no requieren restricción alguna.

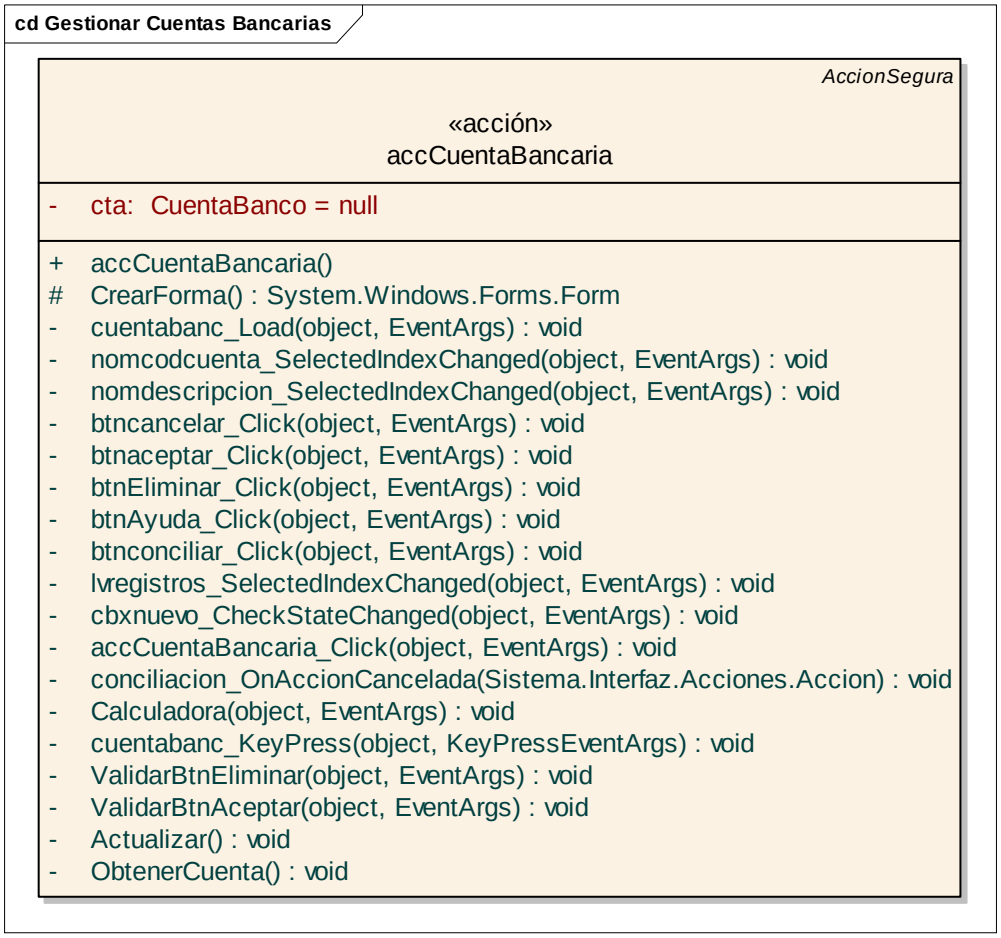

Figura 33.Clase acción Gestionar Cuentas Bancarias.

✓ Gestores

Son las clases que contienen la lógica del negocio y manejan la mayoría de las operaciones contables y financieras que se realizan en el sistema. Su nombre comienza en la mayoría de los casos con la palabra **Gestor** para que puedan ser identificadas fácilmente. A pesar de la diversidad de procesos y subprocesos que se ejecutan en cada uno de los subsistemas, se procuró diseñar un gestor para cada uno de los subsistemas en la mayoría de los casos.

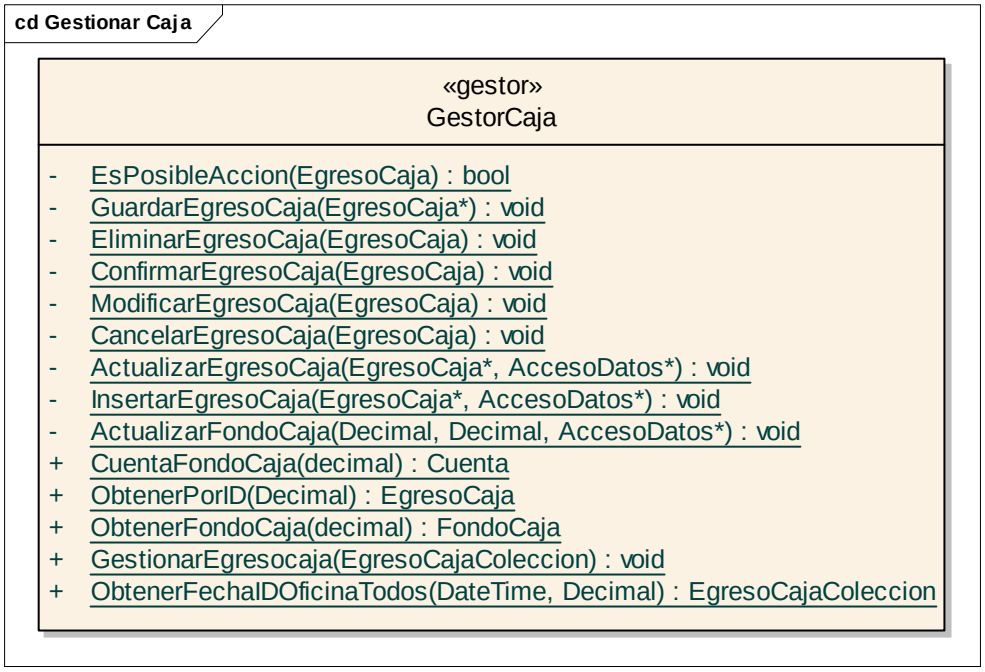

Figura 34. Clase Gestor Caja.

✓ Entidades

Las entidades son las clases que representan objetos o conceptos del negocio y mayormente se identifican por su nombre. Su gran mayoría tiene una clase Colección asociada que representa una lista de sus instancias, con los métodos básicos y otros que se puedan necesitar para manejar estas colecciones. Además de contener toda la información del sistema las entidades realizan algunos cálculos y operaciones sobre los datos que poseen, siempre evitando que sean muy complejos, pues en ese caso seria tarea de alguna clase Gestor desempeñar dicha función. Se integran en ocasiones en grandes jerarquías dado el nivel de complejidad de determinado subsistema y la granularidad del mismo.

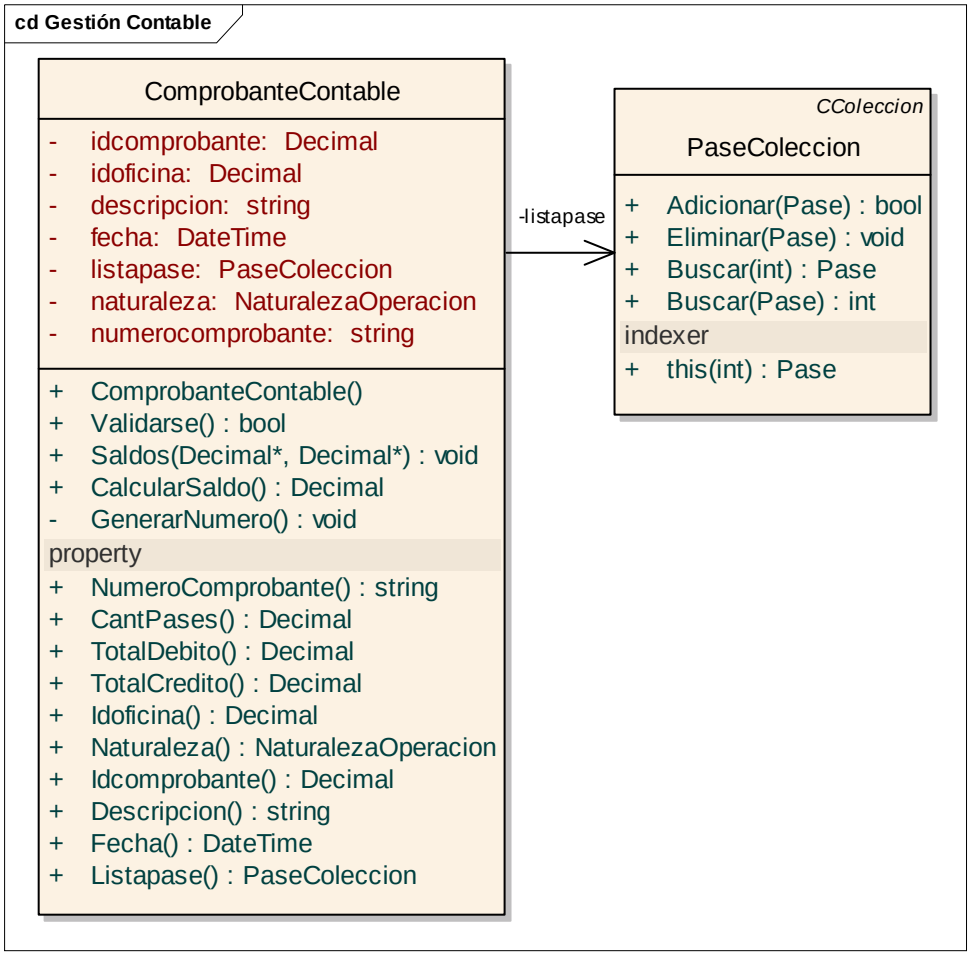

Figura 35. Clase entidad Comprobante Contable.

✓ Fachadas

Las clases que sirven de fachada, como su nombre lo indica, son un punto intermedio entre los objetos del negocio y los objetos persistentes que existen en la Capa de Acceso a Datos, que no necesariamente tienen porque ser iguales, pues cuando estos se materializan desde la Base de Datos, se obtienen objetos de los tipos que genera el TierDeveloper y en muchas ocasiones integran varios conceptos del negocio por razones obvias de rendimiento, este proceso por supuesto funciona a la inversa con los objetos del negocio que tienen que desmaterializarse y convertirse en los tipos persistentes con los que trabajan las Capas de Acceso a Datos. Por convenio se agrega al final del nombre de estas clases la palabra

Fachada. Se construyó en la mayoría de los casos una clase fachada para cada grupo de entidades que forman un proceso del negocio aunque existen excepciones por la complejidad de algunos objetos.

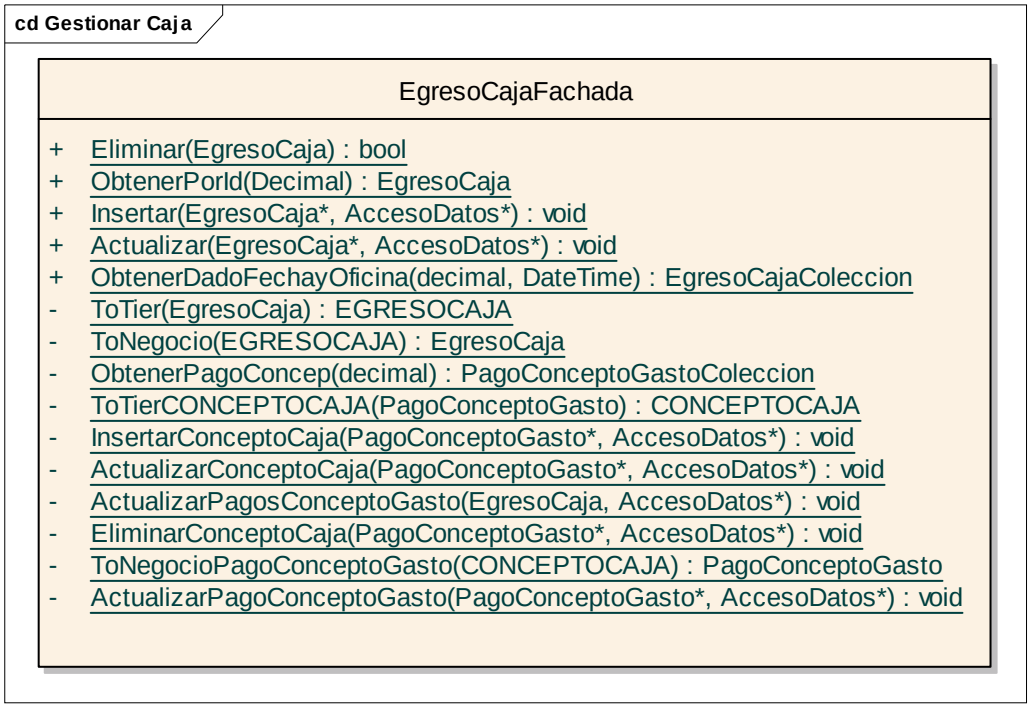

Figura 36. Clase fachada EgresoCajaFachada.

✓ Capas de Acceso a Datos

Estas clases son generadas automáticamente por la herramienta TierDeveloper. Construye un paquete con todas las clases e interfaces necesarias para manejar los objetos mapeados desde la Base de Datos y se le antepone al nombre de dicho paquete las letras **CAD** por convenio. Estas clases se instancian desde la fachada correspondiente al entorno del negocio especifico que se este manipulando, desde allí se realizan todas la trasformaciones de un sistema a otro y se integran en ocasiones varias CAD y varias fachadas para lograr una operación determinada, que generalmente se torna bastante compleja.

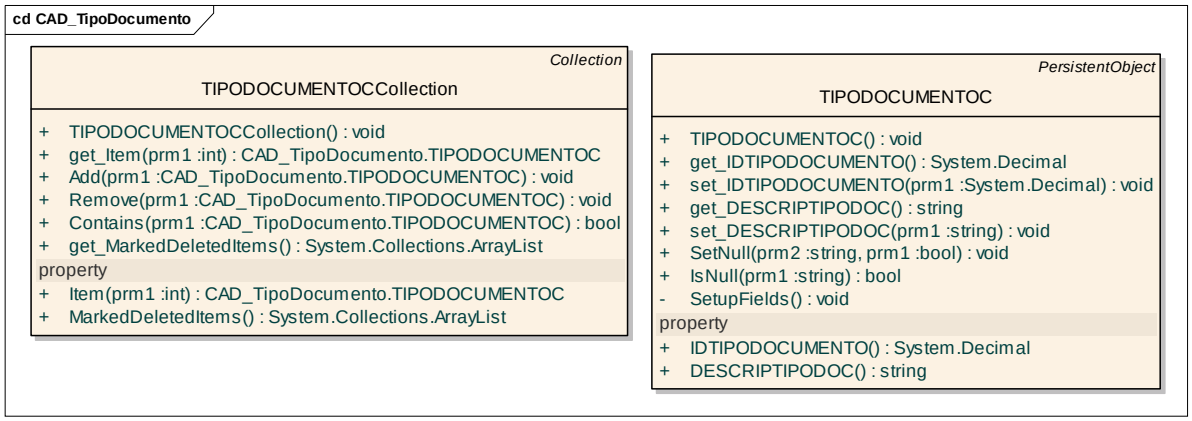

Figura 37. Clase entidad Tipodocumento generada por TierDeveloper.

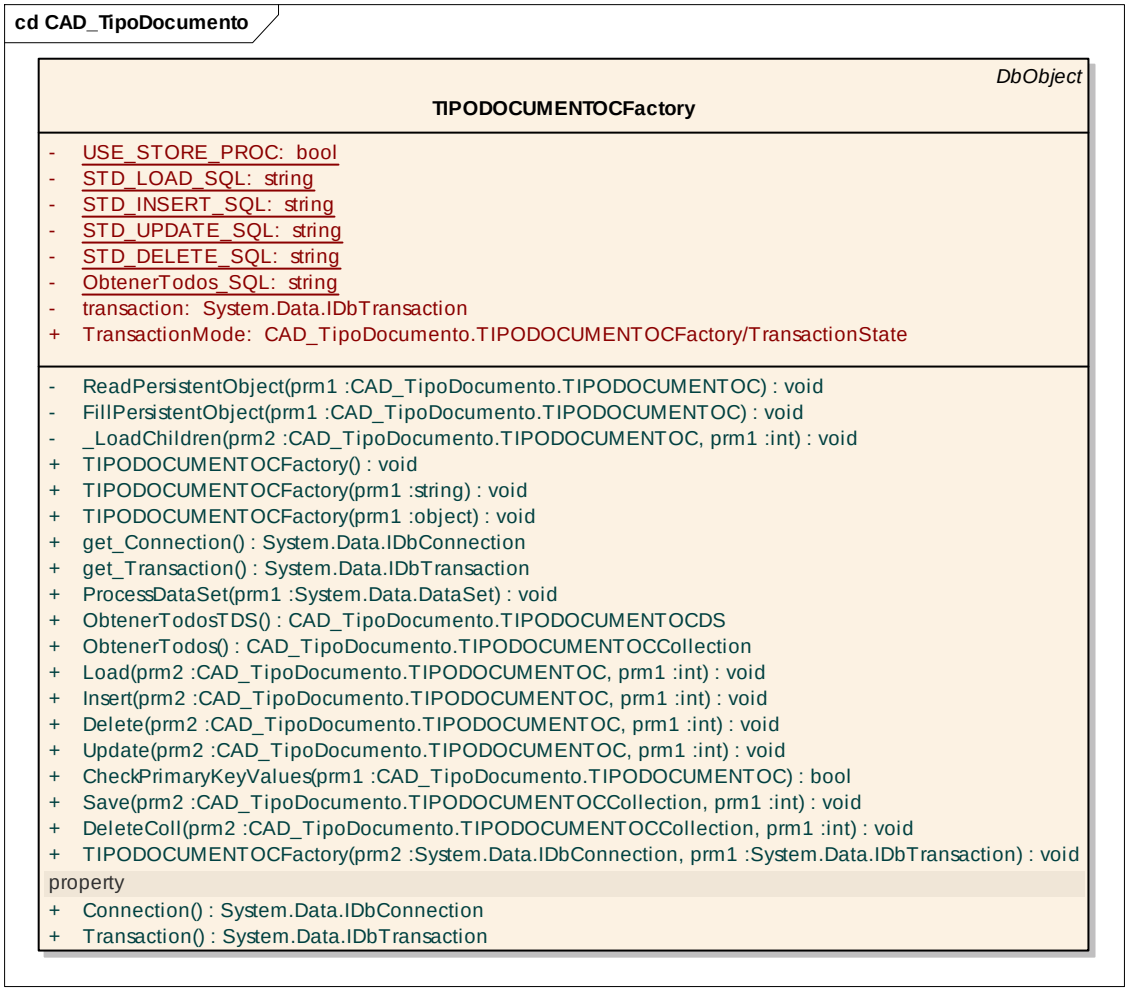

Figura 38. Clase fábrica Tipodocumento generada por TierDeveloper.

3.3.3. Realización de Casos de Uso del Diseño

Según *El Proceso Unificado de Desarrollo de Software* una realización de caso de uso-diseño es una colaboración en el modelo de diseño que describe como se realiza un caso de uso específico y como se ejecuta, en términos de clases de diseño y sus objetos. En este trabajo en particular proporciona una traza directa a los casos de uso del sistema, como ya se analizó anteriormente. A continuación se relacionan las realizaciones más importantes para la arquitectura del sistema:

RCU Gestionar Estados de Cuenta (Ver Anexo 1).

RCU Cerrar Cuentas Nominales (Ver Anexo 2).

RCU Gestionar Registro de Comprobantes (Ver Anexo 3).

RCU Registrar Depósitos por Clasificar (Ver Anexo 4).

RCU Gestionar Factura (Ver Anexo 5).

RCU Registrar Asignación de Presupuesto (Ver Anexo 6).

RCU Registrar Egresos de Caja (Ver Anexo 7).

RCU Registrar Pago de Obligaciones (Ver Anexo 8).

RCU Visualizar Reporte de Balance General (Ver Anexo 9).

RCU Registrar Créditos Adicionales (Ver Anexo 10).

RCU Gestionar Notas de Crédito y Débito (Ver Anexo 11).

3.4 Modelo de despliegue

El artefacto que se realiza en este epígrafe, no pertenece al rol del diseñador según la metodología utilizada (RUP), pero se considera importante ya que el trabajo estuvo sujeto a esta topología y sirve para dar una vista Arquitectónica más visible al modelo de diseño del sistema propuesto en el trabajo.

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software. Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución.

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos o procesos. En general un nodo será una unidad de computación de algún tipo, desde un sensor a un servidor. Las instancias de componentes software pueden estar unidas por relaciones de dependencia, posiblemente a interfaces.

La solución del presente trabajo propone el siguiente diagrama de despliegue:

En los registros mercantiles e inmobiliarios existirán varias **PC** clientes con Windows XP y el sistema como tal, en estas PC se procesa la información que se obtiene del proceso económico de dichos registros, toda esta información se almacena con carácter temporal en los registros de la base de datos que estará en un servidor local con Windows 2003 Server utilizando Oracle 10g Standard Edition One como gestor de bases de datos, estas computadoras estarán conectadas a una red de área local (WAN) en cada oficina, utilizando tecnología inalámbrica. Estos servidores locales se comunicarán a través de una red privada virtual (VPN) al centro de datos, el cual contiene un cluster de base de datos soportado por Oracle Enterprise Edition versión 10g R2 Real Application Cluster. A continuación se muestra el diagrama de despliegue de nuestra propuesta. Este servidor debe estar preparado para soportar un acceso constante e intensivo a través de la red de varias computadoras clientes simultáneamente. En todas las oficinas deberá estar disponible al menos una impresora conectada a la red para imprimir los reportes que se emiten desde el sistema.

En la dirección nacional de estos registros existirá un centro de costo que es el que va a analizar, trazar políticas y dictar medidas económicas y financieras, a partir de la información almacenada a nivel central. Este centro de costo debe tener las PC clientes, donde estará instalado el sistema y necesita de al menos una impresora. En este lugar se gestiona la información directamente del centro de datos.

A continuación se aparecen relacionados algunos elementos que componen el modelo de despliegue concebido para el diseño de este sistema:

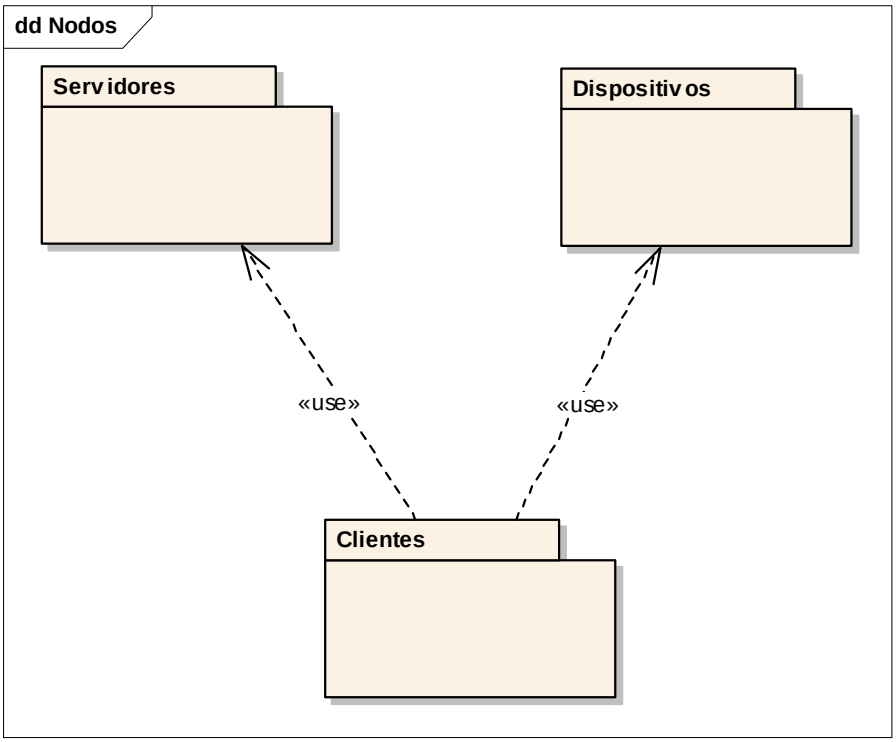

Figura 39. Unidades del modelo de despliegue.

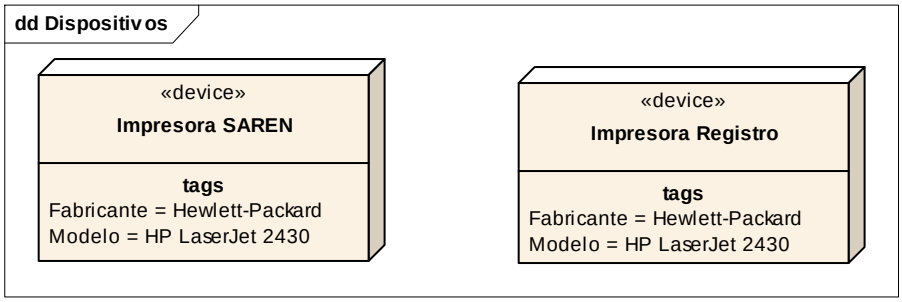

Figura 40. Dispositivos

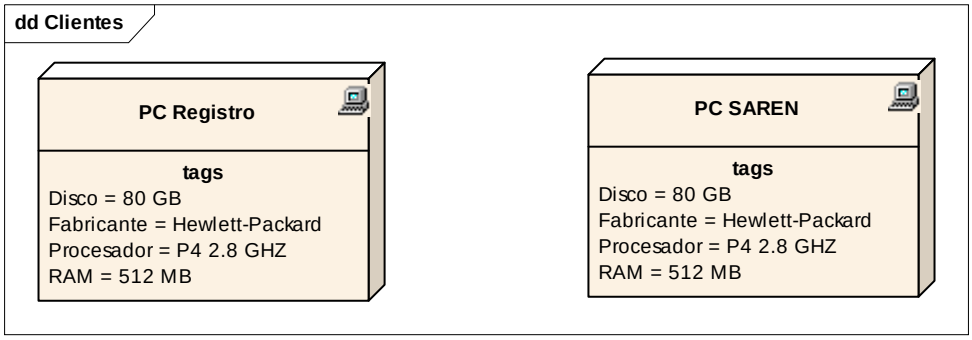

Figura 41. Clientes.

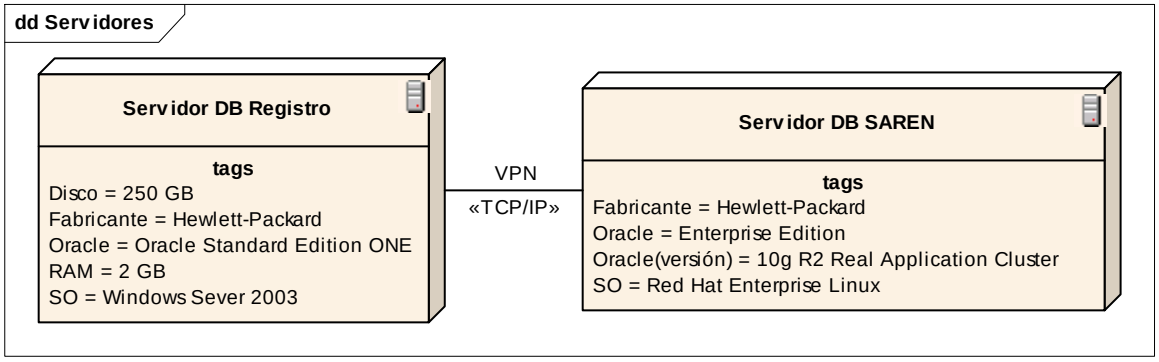

Figura 42. Servidores

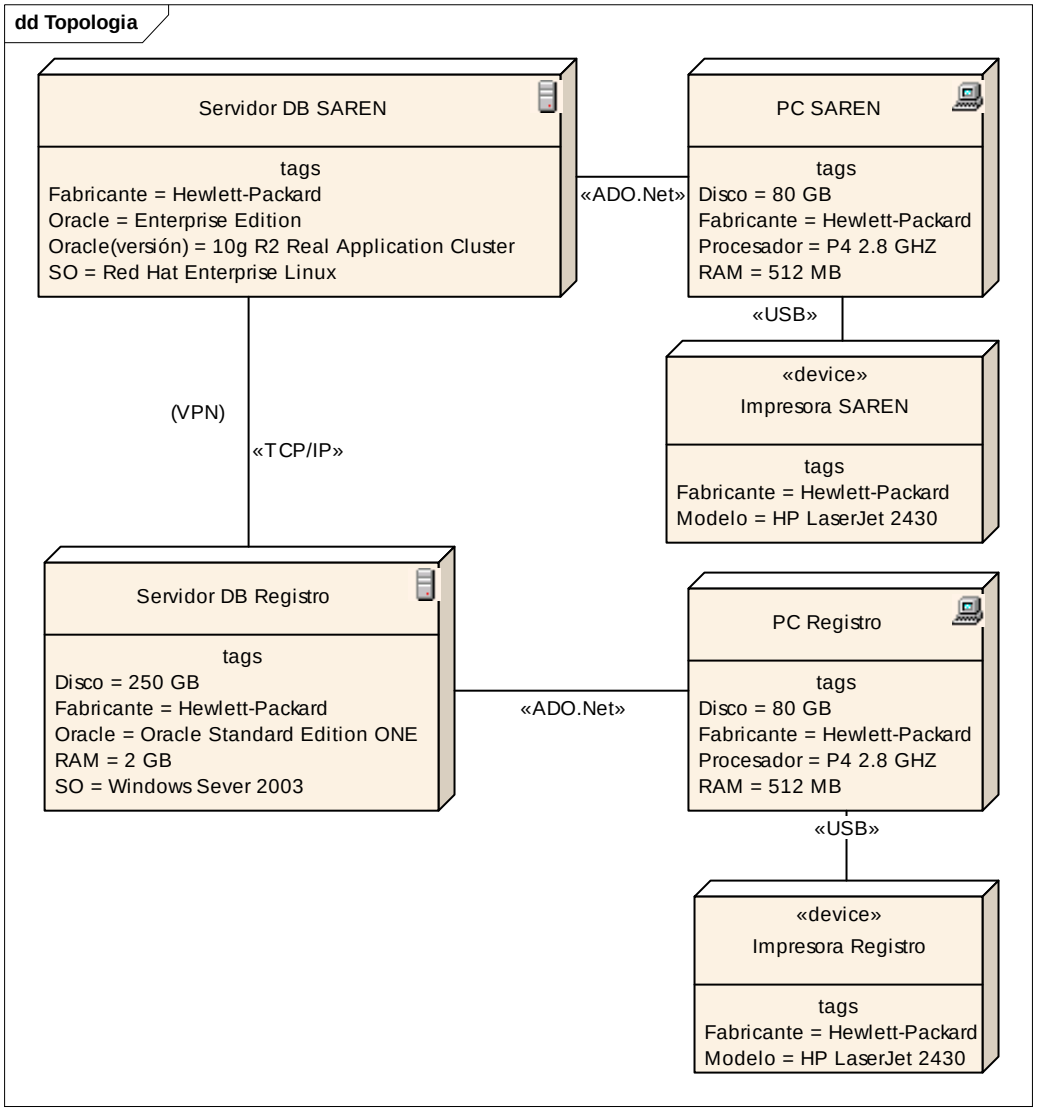

Figura 43. Diagrama de despliegue.

3.5. Modelo de Datos

El modelo de datos del diseño propuesto en su conjunto cuenta con 74 tablas, 436 atributos y 121 llaves. Por su tamaño, se muestra dividido en unidades lógicas de almacenamiento de datos, según los subsistemas definidos en el modelo de diseño y las clases persistentes de cada uno de estos.

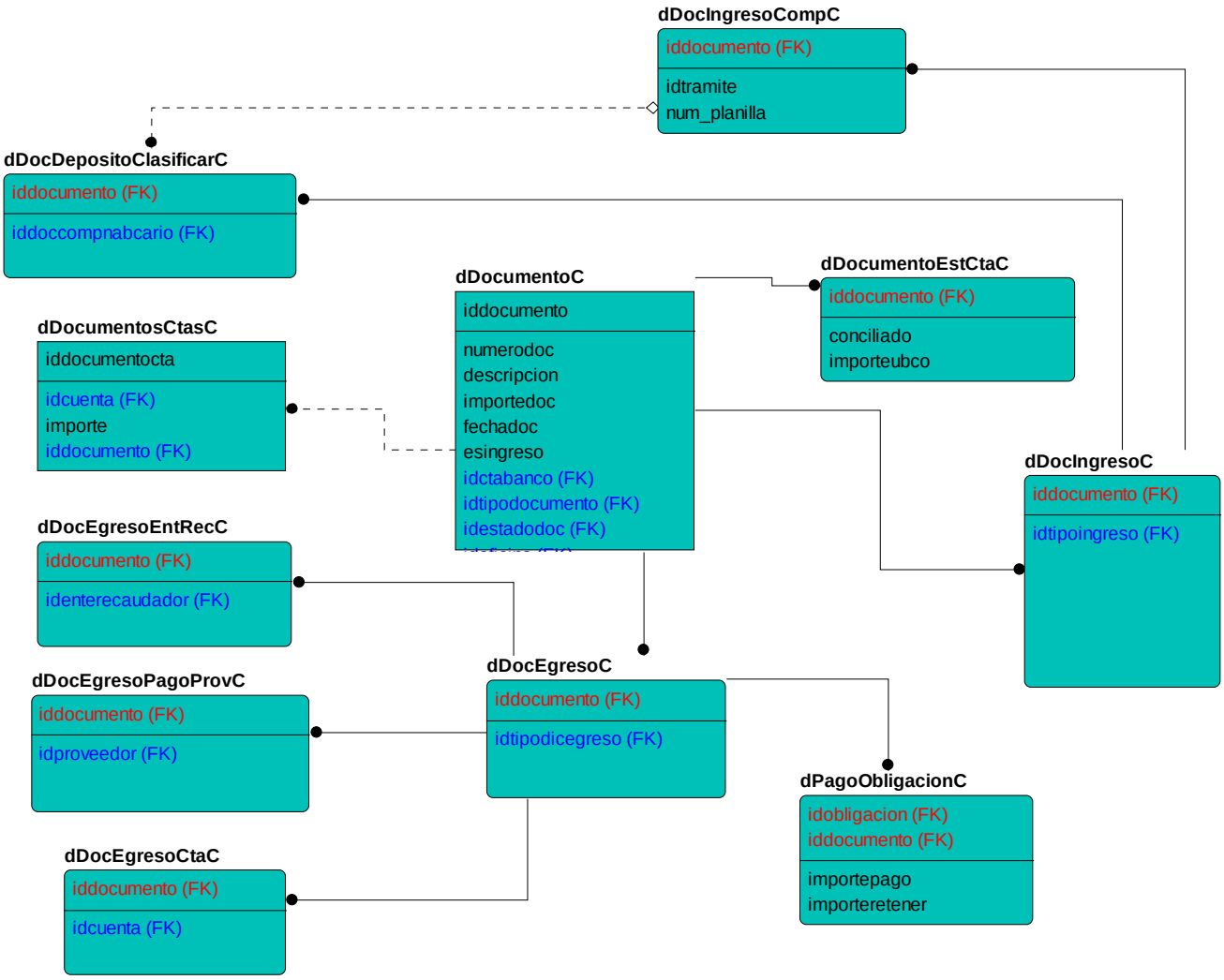

Figura 44. Modelo de datos del subsistema “Gestionar Documentos Banco”.

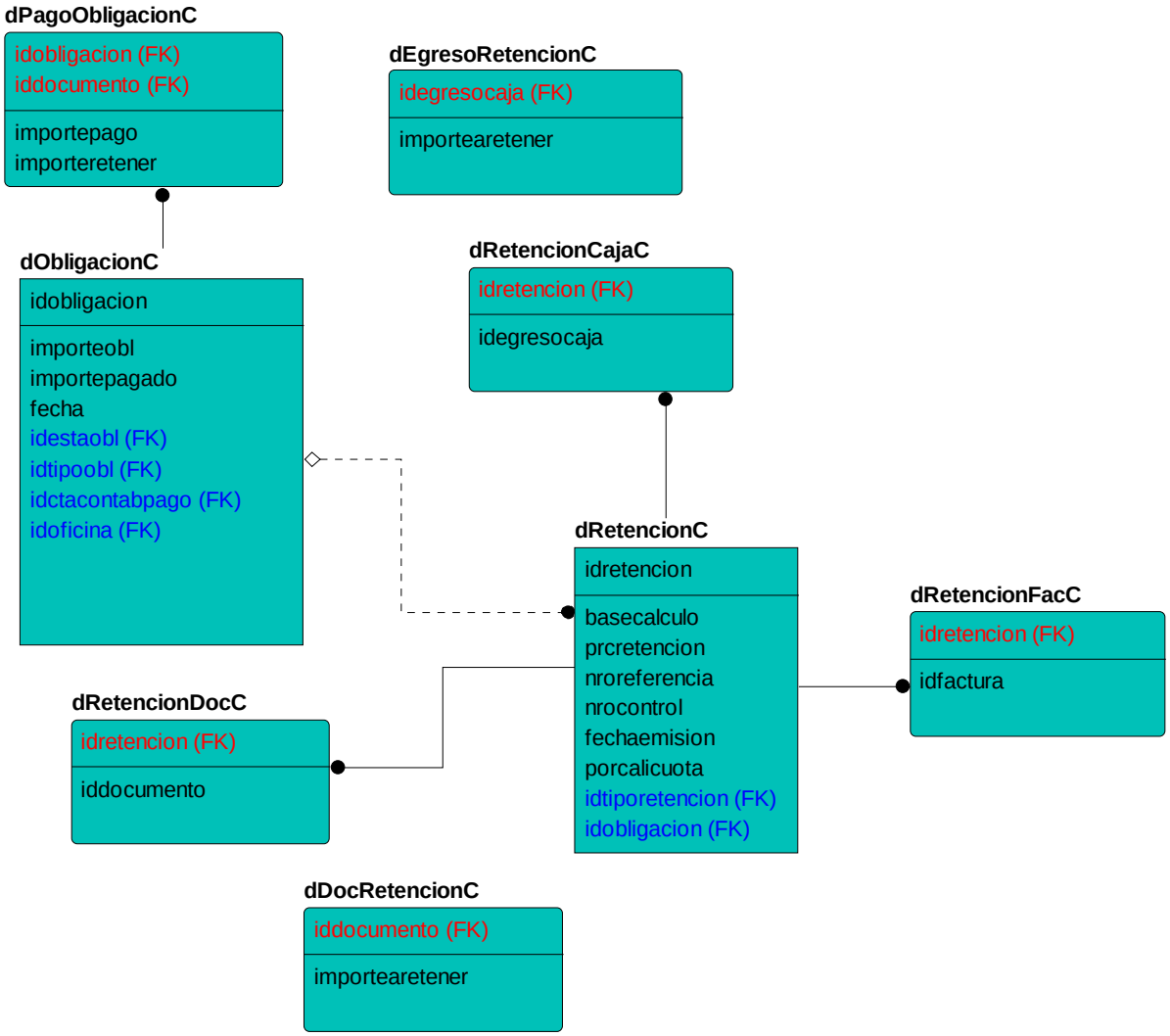

Figura 45. Modelo de datos del subsistema “Gestionar Retenciones”.

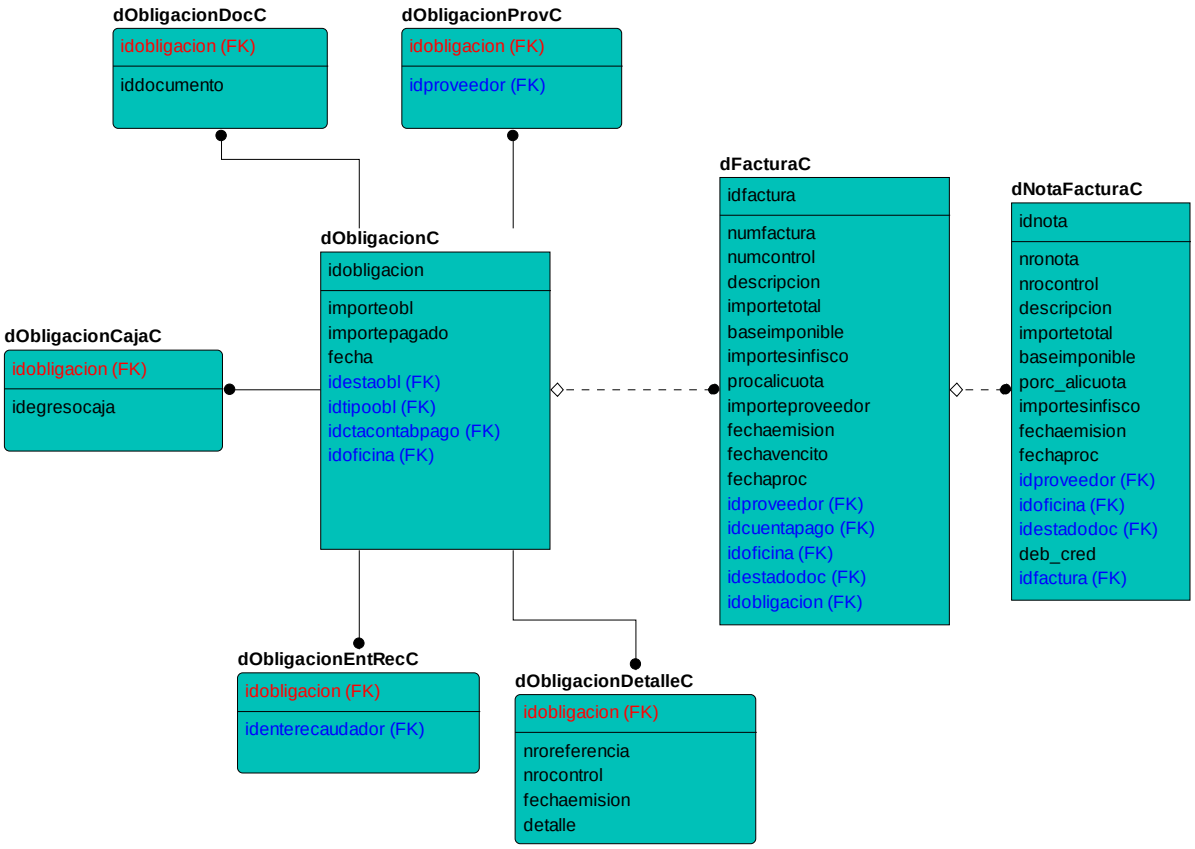

Figura 46. Modelo de datos del subsistema “Gestionar Obligaciones por Pagar”.

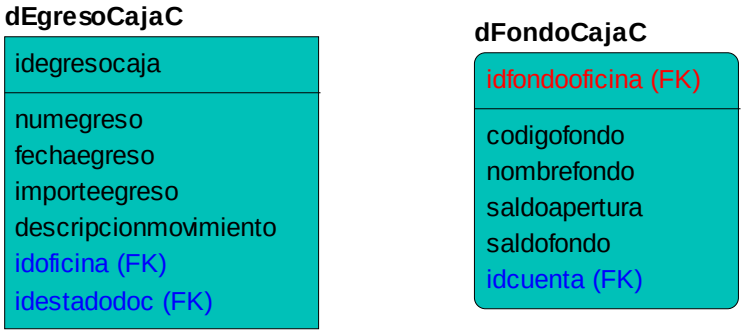

Figura 47. Modelo de datos del subsistema “Gestionar Caja”.

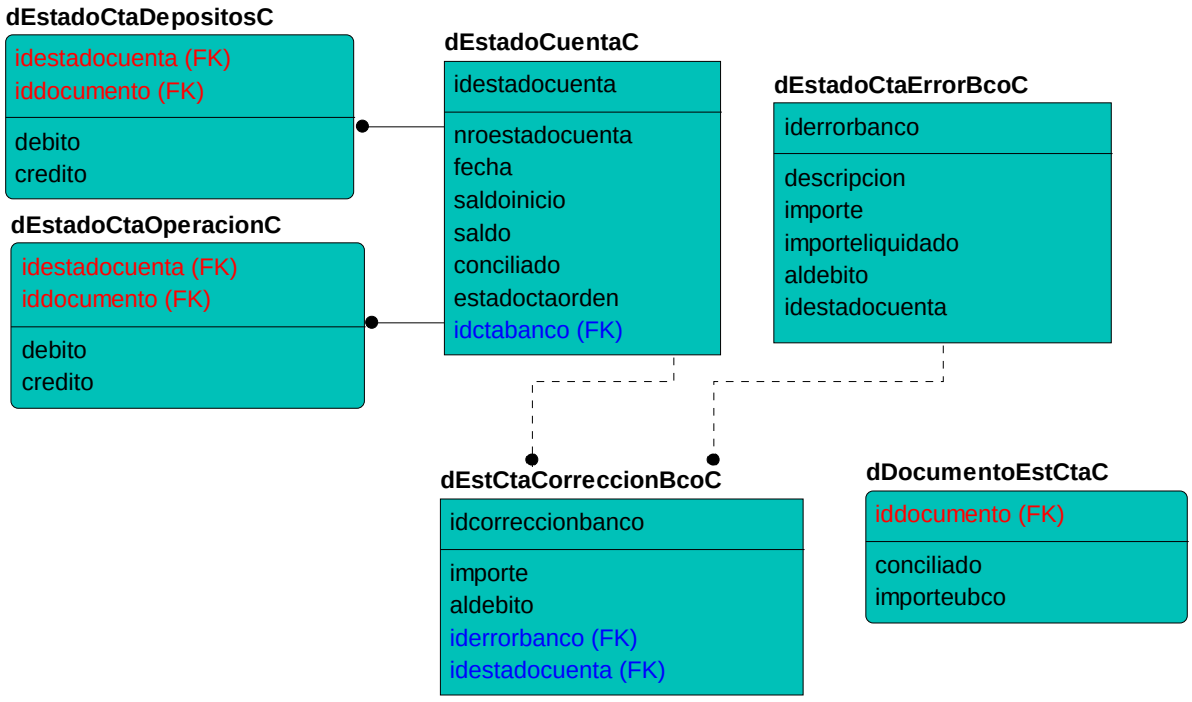

Figura 48. Modelo de datos del subsistema “Gestionar Cuentas Bancarias”.

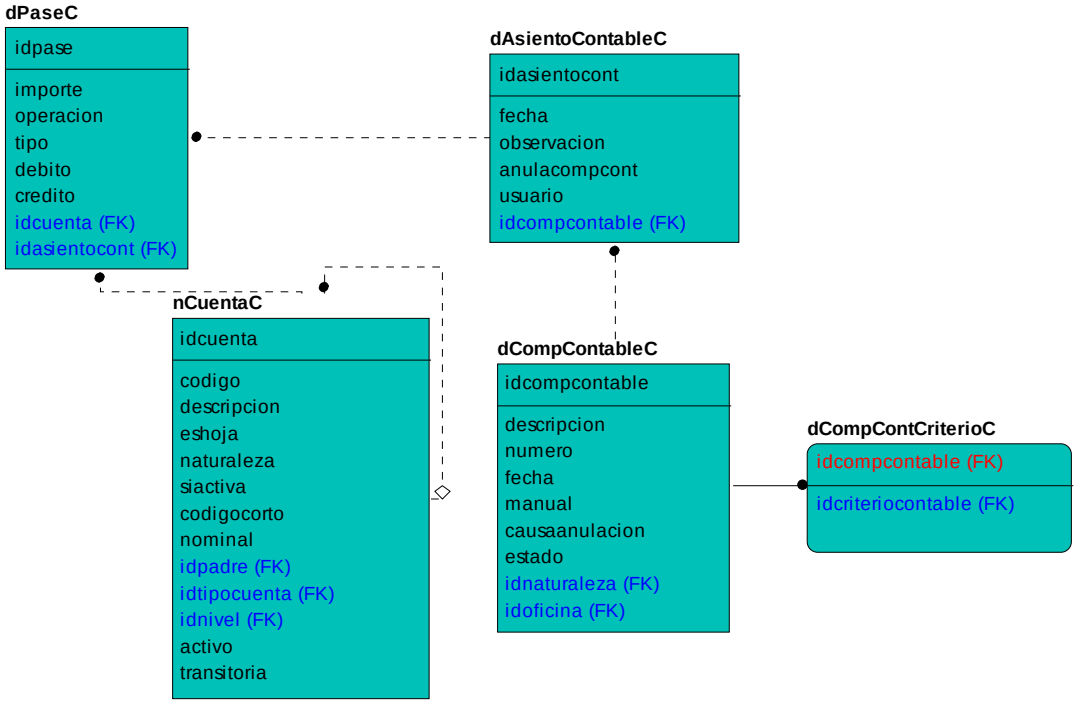

Figura 49. Modelo de datos del subsistema “Gestión Contable”.

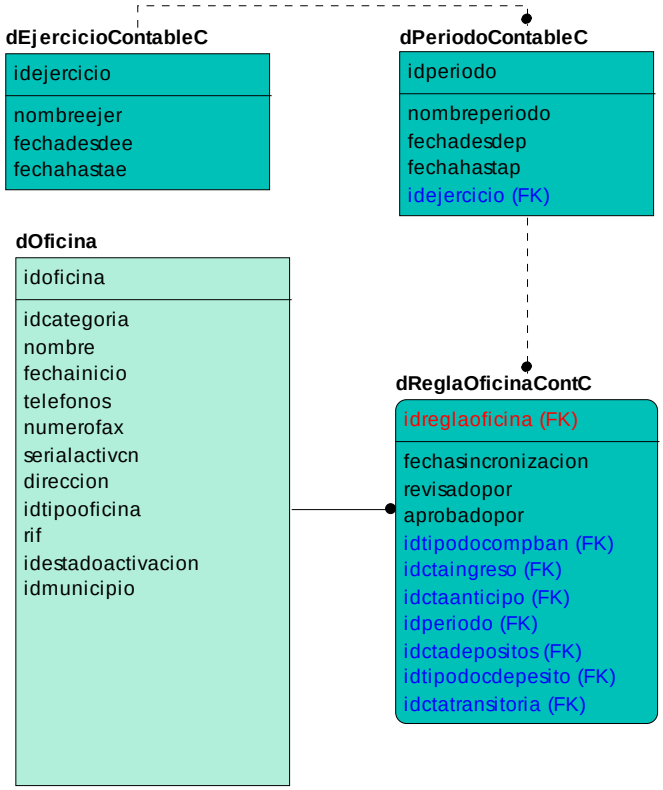

Figura 50. Modelo de datos del subsistema “Configuraciones”.

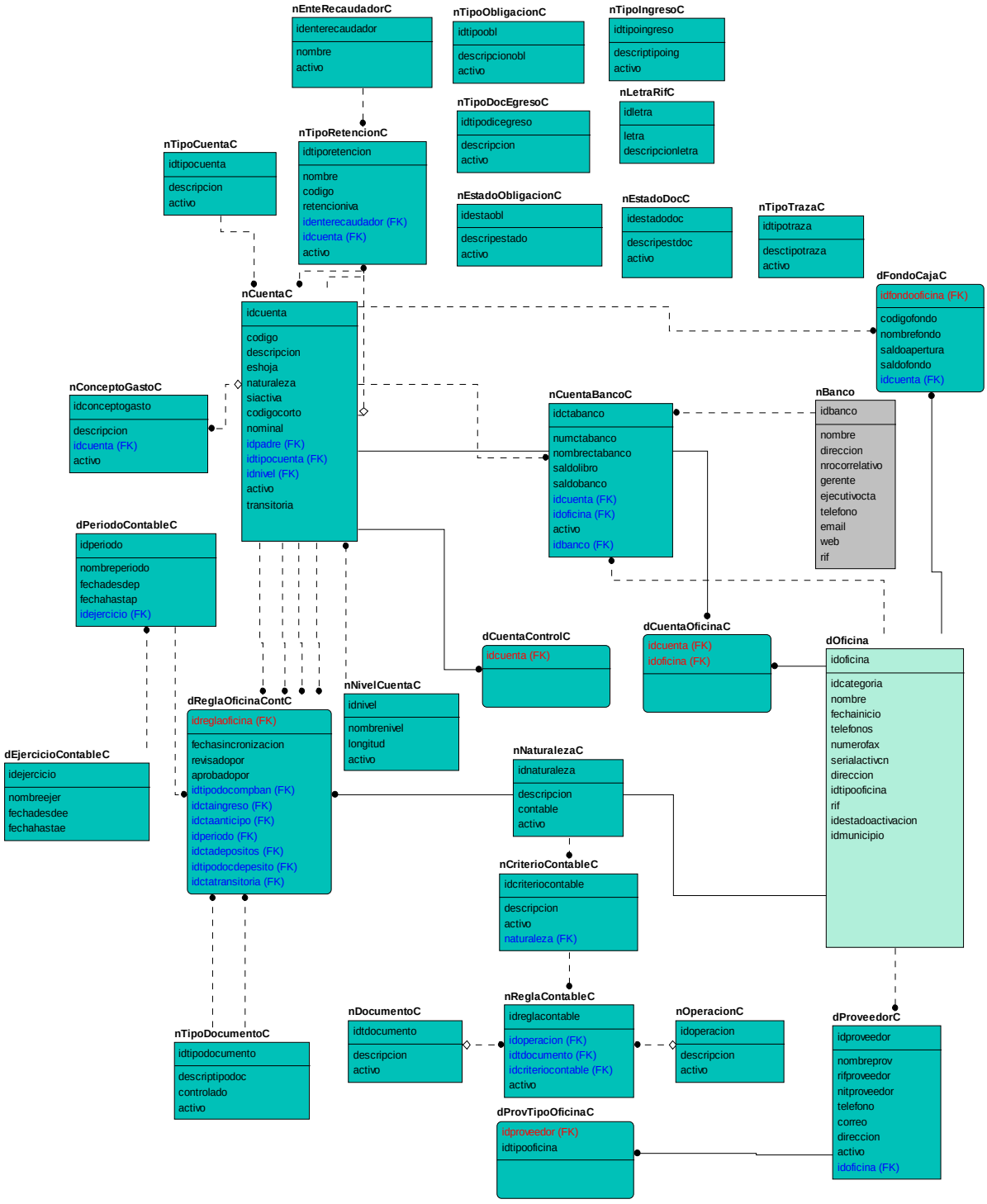

Figura 51. Modelo de datos del subsistema “Nomencladores”.

3.6. Conclusiones

- Se construyó un Modelo de Diseño ajustado al entorno de negocio concreto, con todos los artefactos inherentes a el y que trata de aprovechar las potencialidades de las tecnologías empleadas.
- El Modelo de Despliegue fue elaborado satisfactoriamente según los requisitos no funcionales establecidos.
- Se desarrollo un Modelo de Datos que responde por el almacenamiento de toda la información del negocio, según los estándares establecidos y las principales potencialidades del gestor utilizado.

Capítulo 4: Análisis de los resultados

4.1. Introducción

Mucho acerca del diseño orientado a objetos es subjetivo un diseñador experimentado *"sabe"* como caracterizar a un sistema OO, para que implemente efectivamente los requerimientos del cliente-. Pero, cuando un modelo de diseño OO crece en tamaño y complejidad, una visión más objetiva de las características del diseño puede beneficiar al diseñador experimentado (que adquiere vista adicional), y al novato (que obtiene indicadores de calidad que de otra manera no estarían disponibles).[13]

En el presente capítulo se realiza un análisis de los resultados, tomando como principal basamento las métricas para determinar la calidad de un diseño de sistema. Varios actores proponen métricas para determinar la calidad de un diseño, en este capítulo se toman las más recomendadas a nivel mundial.

4.2. Resultado de las métricas del modelo de diseño.

4.2.1. Métricas de diseño arquitectónico

A continuación se realizan los cálculos para cada uno de los módulos del sistema aplicando las métricas para el diseño arquitectónico**.** Haciendo referencia a las fórmulas definidas en el capítulo 1.

Algunos actores han propuesto umbrales para estas métricas como se explica a continuación:

Para caracterizar un módulo en no complejo estructuralmente, los valores de los umbrales, según la fórmula definida en el capítulo 1 deben ser de: 1;4;9;16;25, los cuales corresponde con valores de 1;2;3;4;5 para $f_{out}(i)$, donde $f_{out}(i)$ representa la cantidad de relaciones de este módulo con los demás módulos. Así mismo se definen valores para un módulo complejo de 36;49;64 y muy complejo mayor igual que 81.

Para caracterizar un módulo de baja complejidad de datos $D(i)$, debe ser menor igual que 7, para la categoría de complejo mayor que 7 y menor igual que 12 y para muy complejo mayor que 12.

En el caso de la complejidad de sistema $C(i)$, que representa la suma de la complejidad estructural más la de datos, se define que los valores para un sistema no complejo deben estar en el rango de menor igual que 32. Complejo de mayor que 32 y menor igual que 50 y muy complejo los valores deben ser mayor que 50.

A continuación se muestra una tabla con los valores definidos anteriormente:

Complejidad del sistema			
Complejidad	**S(i)**	**D(i)**	**C(i)**
No complejo	1,4,9,16,25	<=7	<=32
Complejo	36,49,64	>7 y <=12	>32 y <=50
Muy complejo	81…	>12	>50

Tabla 3. Umbrales de complejidad.

Los indicadores de calidad que podemos medir con estas métricas corresponden a la integración entre los módulos y la complejidad de las pruebas. En la siguiente tabla se muestran estos indicadores o parámetros de calidad:

Parámetros de calidad	**Sistemas complejos y muy complejos**
Integración	Aumenta el esfuerzo necesario para la integración.
Complejidad de las pruebas	Aumenta la complejidad de las pruebas.

Tabla 4. Parámetros de calidad para sistemas altamente complejos

A continuación se definen los módulos que presentan relaciones y se calculan para cada uno de estos, las complejidades anteriormente definidas:

Gestión Contable

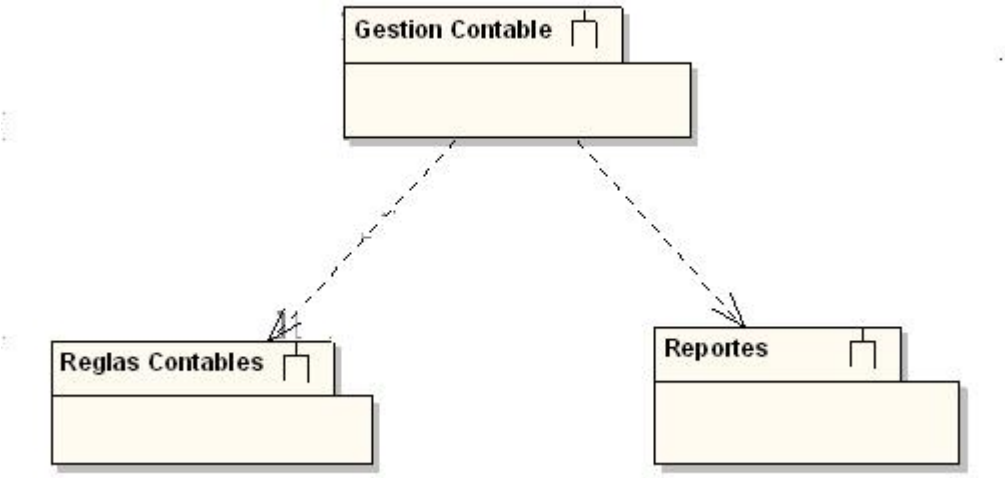

Figura 52. Módulo Gestión Contable y los módulos descendientes.

$$S(i) = 2^2 = 4$$

$$D(i) = 13/[2+1] = 4.33$$

$$C(i) = 4 + 4.33 = 8.33$$

Gestionar Documento de Banco

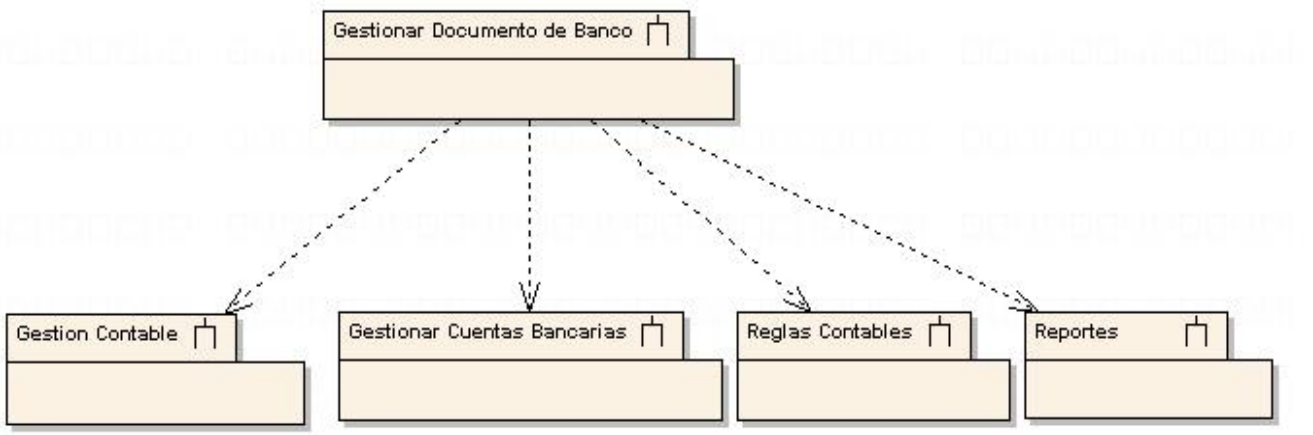

Figura 53. Módulo Gestionar Documento de Banco y los módulos descendientes.

$$S(i) = 4^2 = 16$$

$$D(i) = 15/[4+1] = 3$$

$$C(i) = 16 + 3 = 19$$

Gestionar Cuentas Bancarias

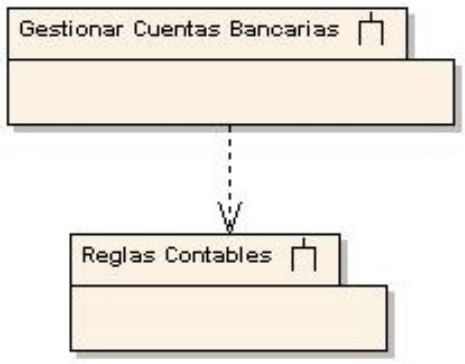

Figura 54. Módulo Gestionar Cuentas Bancarias y los módulos descendiente.

$$S(i) = 1^2 = 1$$

$$D(i) = 8/[1+1] = 4$$

$$C(i) = 1+4 = 5$$

Gestionar Obligaciones por Pagar

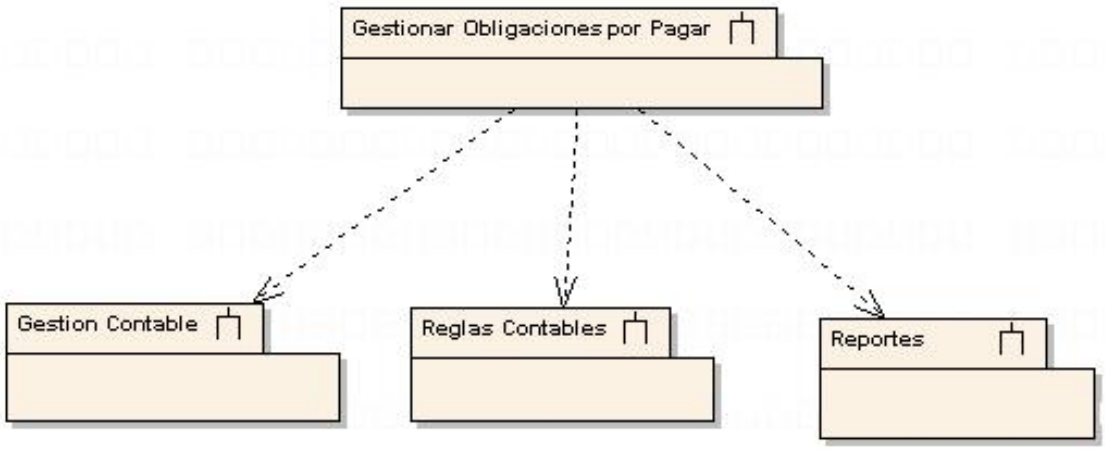

Figura 55. Módulo Gestionar Obligaciones por Pagar y los módulos descendientes.

$$S(i) = 3^2 = 9$$

$$D(i) = 12/[3+1] = 3$$

$$C(i) = 9 + 3 = 12$$

Gestionar Caja.

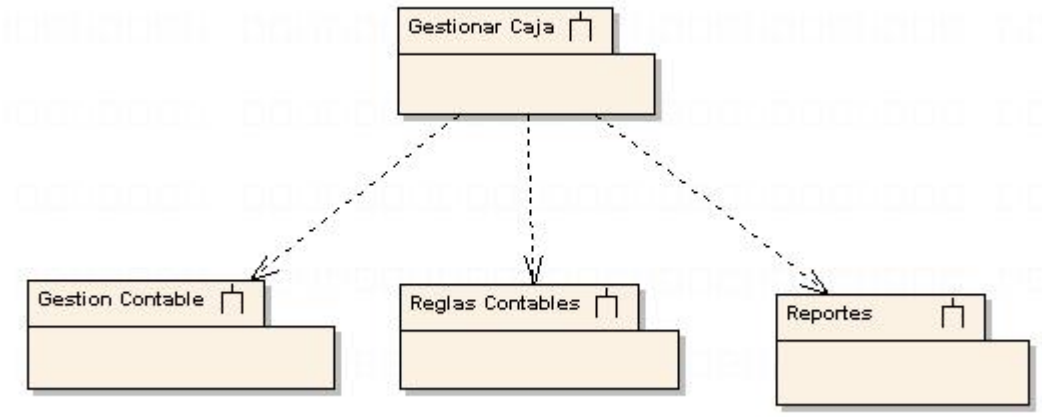

Figura 56. Módulo Gestionar Caja y los módulos descendientes.

$$S(i) = 3^2 = 9$$

$$D(i) = 10/[3+1] = 2.5$$

$$C(i) = 9 + 2.5 = 11.5$$

Resultado: Según los valores obtenidos en los análisis anteriores para los módulos que inician otros y partiendo de que los umbrales definidos con anterioridad, se llegó a los resultados ilustrados en la siguiente gráfica:

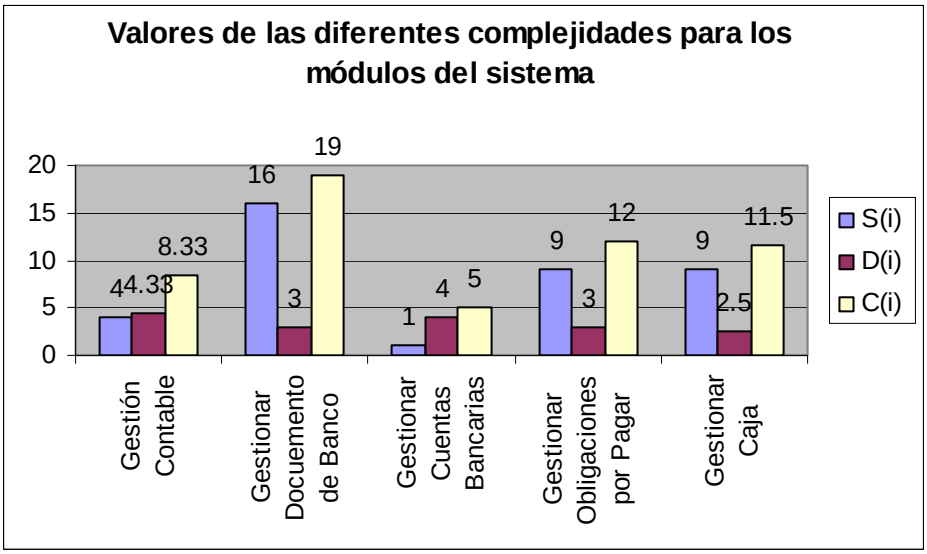

Figura 57. Complejidad de los módulos.

Como se observa en la gráfica el módulo más complejo dentro del sistema es “Gestionar Documento de Banco”. En sentido general el sistema no es altamente complejo después de evaluar los umbrales propuestos para estas métricas.

4.3. Resultado de las métricas orientadas a clases

4.3.1. Métricas propuestas por Lorenz y Kidd. Aplicación al modelo.

A continuación se aplican algunas de las métricas antes mencionadas, para determinar el grado de calidad y fiabilidad del diseño propuesto en este trabajo.

Tamaño de clase (TC)

Esta métrica consiste en medir el tamaño de una clase a partir de las siguientes medidas:

1. Total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
2. Número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.
3. Promedio general de las dos métricas anteriores para el sistema en general.

La métrica **TC** fue aplicada solamente a la capa de negocio porque es la más compleja dentro de las capas y la que engloba la mayoría de la lógica del negocio. Todas las clases de la capa negocio con la cantidad de atributos y operaciones pueden verse en el **Anexo 12.**

Un **TC** grande afecta los indicadores de calidad definidos para esta métrica por los especialistas, ver la tabla 5.

Parámetros de calidad	A valores grande de TC
Reutilización	Reduce la reutilización de la clase
Implementación	Complica la implementación
Complejidad de las pruebas	Hace compleja las pruebas del sistema
Responsalbilidad	La clase debe tener bastante responsabilidad

Tabla 5. Parámetros de calidad para valores grandes de TC.

Las medidas o umbrales para los parámetros de calidad han sido una polémica a nivel mundial en el diseño de sistemas Algunos especialistas plantean umbrales para estas métricas según se muestra en la **tabla 6**, estos fueron los aplicados en el diseño de este sistema.

No de Operaciones y/o Atributos	
TC	**Umbral**
Pequeño	<=20
Medio	>20 y <=30
Grande	>30

Tabla 6. Umbrales para TC

Resultado: Los resultados para esta métrica en el diseño del sistema propuesto fueron los siguientes:

La capa de negocio cuenta con 245 clases como se muestra en el **anexo 12**, para un promedio de cantidad de atributos de 2.1 y un promedio de cantidad de operaciones de 24.9 como se muestra en la **tabla 7**.

Total de Clases	Promedio de Atributos	Promedio de Operaciones
245	2.1	24.9

Tabla 7. Total de clases del negocio y los promedios de atributos y operaciones.

La capa de negocio presenta 32 clases pequeñas, 63 de tamaño medio y 50 de tamaño grande, en la **tabla 8** se puede ver la distribución por tamaño.

Umbral	Tamaño	Cantidad de Clases
<=20	Pequeño	132
>20 y <=30	Medio	63
>30	Grande	50

Tabla 8. Cantidad de clase por tamaño.

Otra forma representativa es el por ciento de clase según los tamaños a partir de los umbrales definidos en la **tabla 58** se puede observar que existe en la capa de negocio un 54% de clase pequeñas, un 26% de clases medianas y un 20% de clases grandes.

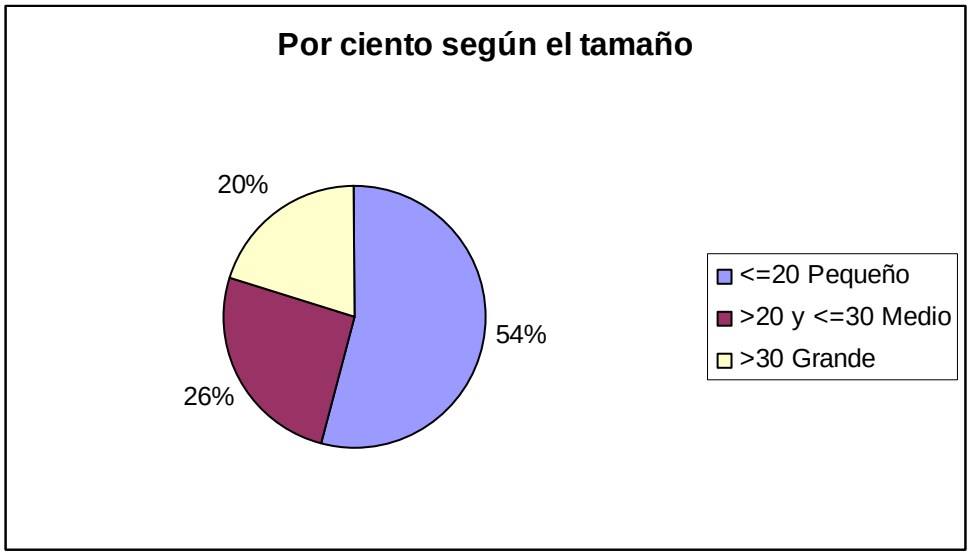

Figura 58. Por ciento de clases por tamaño.

Como se puede observar en las tablas que muestran los resultados la mayor cantidad de clases están clasificadas entre pequeñas y medianas para un resultado positivo según los parámetros de calidad propuesto para la métrica **TC**.

Número de operaciones redefinidas para una sub-clase (NOR)

Existen casos en que una subclase reemplaza una operación heredada de su superclase por una versión especializada para su propio uso. A esto se le llama redefinición. Los valores grandes para el **NOR**, generalmente indican un problema de diseño[13].

Esta es una de las métricas que se aplican para medir la calidad del diseño propuesto para los registros y notarías de Venezuela.

Resultado: A partir de los datos obtenidos después de aplicar al sistema la métrica **NOR** se obtuvo lo siguiente:

Cantidad de clases del sistema	Total de subclases que reemplazan operaciones
2721	286

Tabla 9. Total de clases que reemplazan operaciones.

El número de subclases que remplazan operación de las superclases es mínimo en relación a la cantidad de clases que conforman el sistema. Se puede ver que existe una jerarquía adecuada y también esto permite que el software pueda ser fácilmente probado y modificado en el tiempo.

4.3.2. La serie de métricas CK. Aplicación al modelo.

Uno de los conjuntos de métricas OO más ampliamente referenciados, ha sido el propuesto por Chidamber y Kemerer. Normalmente conocidas como la serie de métricas **CK**, los autores han propuesto seis métricas basadas en clases para sistemas OO[13].

En el presente epígrafe se van a tomar algunas de estas métricas para tener una medida del diseño propuesto en el capítulo anterior.

Árbol de profundidad de herencia (APH)

Esta métrica se define como <<la máxima longitud del nodo a la raíz del árbol >>. A medida que el APH crece, es posible que clases de más bajos niveles hereden muchos métodos. Esto conlleva dificultades potenciales, cuando se intenta predecir el comportamiento de una clase. Una jerarquía de clases

profunda (el APH es largo) también conduce a una complejidad de diseño mayor. Por el lado positivo, los valores APH grandes implican un gran número de métodos que se reutilizarán[13].
Por su parte, algunos autores sugieren un umbral de 6 niveles como indicador de un abuso en la herencia en distintos lenguajes de programación.

Resultado: A partir de los datos obtenidos después de aplicar al sistema la métrica **APH** se obtuvo que los niveles más altos de herencias son de **4** lo cual se encuentra dentro del umbral definidos para determinar que el diseño no es complejo, no existe un alto acoplamiento y no es de difícil mantenimiento.

Número de descendiente (NDD)

La métrica **Número de Descendiente** nos brinda resultados numéricos para determinar la complejidad de prueba del diseño, aplicando esta métrica al diseño propuesto se obtuvieron los siguientes resultados:

Resultado: El sistema presenta como máximo nivel de descendiente 5. Este nivel representa una media para los umbrales que proponen algunos autores en el campo de métricas de diseño.

4.4. Conclusiones

- En este capítulo se logró aplicar las métricas propuestas para la evaluación del diseño del capítulo 3.
- Se determinó que el diseño propuesto no presenta una alta complejidad estructural, de datos, ni del sistema en general permitiendo que las pruebas no sean complejas y no exista un gran esfuerzo para integrar los módulos.
- Las métricas como “Tamaño de Clase” evidencian que la mayoría de las clases son reutilizables, la implementación no es complicada y las pruebas no son complejas.
- El número de subclases que redefinen métodos no es alto posibilitando que el sistema pueda ser fácilmente probado y modificado en el tiempo.
- La profundidad de los niveles de herencia están acordes con los umbrales definidos por algunos autores permitiendo el bajo acoplamiento y que el sistema no sea complejo.

Conclusiones generales

- Quedaron definidas la metodología, tecnologías y herramientas para la construcción del diseño software a partir de un estudio previo.
- Después de analizados los principios y las tendencias actuales del diseño, se definió la estrategia para construir un diseño flexible y escalable, a partir del uso de patrones y frameworks.
- Las herramienta CASE es parte importante dentro de la construcción de este diseño software, pues permitió la automatización de tareas implícitas a los largo de todo el proceso de diseño.
- El estudio de los casos de usos del sistema, requisitos asociados y reglas de negocio, fueron el punto de partida fundamental para la elaboración de este trabajo permitiendo ajustar el modelo de diseño a los requisitos de los usuarios finales.
- Con el modelo de diseño de los subsistemas de Gestión Contable, Finanzas y Reglas Contables, se obtuvieron los artefactos necesarios, según la metodología de desarrollo de software seleccionada a partir del estudio del arte realizado (RUP), para la implementación, flujo que define esta metodología a continuación del diseño.
- Los resultados obtenidos a partir de los análisis del diseño del sistema son positivos, tomando como argumento la aplicación de métricas que permitieron evaluar el nivel de calidad que presenta.

Recomendaciones

- Elaborar el modelo de análisis para este sistema, ya que se obtiene un nivel de abstracción con el diseño, permitiendo el ahorro de tiempo, la disminución del costo y migrar más fácilmente a otra tecnología o lenguaje de programación.
- Incorporar al diseño los módulos propuestos por los ERP para brindar una solución más completa a los procesos financieros de los Registros y Notarías de Venezuela.
- Tomar como referencia este modelo para la creación de un modelo de diseño genérico para aplicaciones contables en el polo productivo de sistemas legales.

Bibliografía

1. Dávila, L.M. (2002) *El Software que Controla*.
2. AICPA, *American Institute of Certified Public Accountants*.
3. *American Accounting Association.*
4. Lázaro, E.L.d.
5. Maldonado.
6. Charles T. Horngren, W.T.H., *Accounting.* 1991.
7. Meigs, R., 1992.
8. Catacora, F., 1998.
9. Suárez, L.A.G. (2005) *Reseña histórica de la evolución de la ciencia financiera.*
10. Carbajal, E., *La importancia de las finanzas.* 2006.
11. Málaga, U.d., *Diccionario de economía y finanzas*.
12. UNMSM, O.G.d.S.d.B.y.B.e.d.l., *Automatización de Conciliciones Bancarias y de Procesos de Facturación, Control de Estados de Cuenta, Linea de Créditos y Riesgos de Canales* 2003.
13. Pressman, R.S., *Ingenieria de software. Un enfoque practico*. Vol. Vol. 1. 1998.
14. Lucena, M.E.M.y.C.J.P.d., *El Desarrollo del Framework Orientado al Objeto.*
15. Reynoso, C., *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft* 2004
16. Sanchez, M.A.M. (2004) *Metodologías de Desarrollo de Software*.
17. Montecinos., M.I. (2007) *Análisis y Diseño para soluciones con Arquitectura Microsoft.NET*.
18. Escorial, J.S. (2004) *CALIDAD DEL SOFTWARE.Métricas para modelos conceptuales*.
19. Huerta, R.P.L.G.y.J.L.D.J., *Herramientas Case.*
20. wikipedia, *Algunos lenguajes de especificación*. 2007.
21. Mono, *Mono Project*. Vol. 2006. 2006.
22. ECMA. *Standard ECMA-335, Common Language Infrastructure (CLI), 3rd Edition*. 2005 June 2005 [cited 2006; 3rd:[Available from: http://www.ecma-international.org/publications/standards/Ecma-335.htm
23. ISO. *ISO/IEC 23271:2003* 2005 2005-06-30 [cited; Available from: http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36769&ICS1=35&ICS2=60
24. ECMA. *Standard ECMA-334 C# Language Specification, 3rd Edition*. 2005 June 2005 [cited 2006; 3rd:[Available from: http://www.ecma-international.org/publications/standards/Ecma-334.htm
25. ISO. *ISO/IEC 23270:2003* 2005 2005-06-30 [cited; Available from: http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36768&ICS1=35&ICS2=60&ICS3=&scopelist
26. Ximian, *Mono Project*. Vol. 2006. 2006.
27. Cabrera, M., *Introducción a NHibernate.* 2005.
28. Gamma, E., *Design Patterns*. 1995: Addison Wesley.

Glosario de términos

A

Activos

Recursos económicos de propiedad de una empresa que se espera beneficie operaciones futuras.

B

Balance General

Es un estado financiero que muestra los activos, pasivos y patrimonio de una entidad comercial específica en una fecha determinada.

C

Contabilidad

Es el arte de interpretar, medir y describir la actividad económica.

Comprobante

Autorización escrita que se utiliza para aprobar una transacción para registro y pago. Contiene pases de diferentes cuentas a la columna del debe y al haber. De forma tal que cuadren estas dos columnas.

Conciliación Bancaria

Análisis que explica la diferencia entre el saldo del efectivo que aparece en el extracto bancario y el saldo de efectivo que aparece en los registros del depositante.

Cuenta Contable

Son registros que se utilizan para hacer asientos de disminuciones o aumentos de los activos, pasivos, patrimonio, ingresos, gastos, etc. Según su naturaleza se clasifican en deudoras y acreedoras.

Crédito

Un valor asentado en el lado derecho de la cuenta, denominado Haber.

Comprobantes Contables.

Documento primario donde se registra las transacciones, reflejándose la fecha, descripción y los asientos contables, los cuales pueden contemplar una o más cuentas al débito y una o más cuentas al crédito. La suma del valor de todos los asientos débitos es igual a la suma del valor de todos los asientos créditos.

Cuenta bancaria

Una cuenta bancaria es donde se deposita el dinero de la empresa. Es una forma de no llevar dinero en efectivo.

Cuenta Deudora

Son aquellas cuentas cuyo saldo está al débito, por ejemplo, activos, gastos.

Cuenta Acreedora

Son aquellas cuentas cuyo saldo está al crédito, por ejemplo, pasivos, ingresos, capital.

Case

Computer Aided Software Engineering

D

Débito

Un valor asentado en el lado izquierdo de una cuenta llamado también Debe. **David Ricardo**

E

ERP

Enterprise Resource Planning

Efectivo

Billetes, monedas, cheques, cheques de gerencia o cualquier otro medio de intercambio que un banco aceptará en depósito.

Ejercicio Fiscal

Es un período de tiempo que agrupa varios períodos contables.

Egresos

Salidas de efectivo.

F

Factura

Comprobante de venta; cuenta detallada que el vendedor entrega al comprador y que muestra todos los detalles de la venta, por ejemplo, la fecha, nombre del comprador y del vendedor, número del vendedor, cantidad y descripción de los artículos, precio unitario, prórrogas, descuentos, si los hubiere, importe total, etc. La factura es en realidad un documento de entrada original y como quiera que se haya generalizado

el sistema de cuentas, con frecuencia se retiene por el vendedor un duplicado de cada factura con el fin de preparar el estado de cuenta mensual. Es una solicitud de pago que le hace el vendedor al comprador. Contiene la Relación pormenorizada de las mercancías que se compran o venden. Muestra las cantidades, precios y condiciones del crédito. Sirve como base para el asiento en los registros contables del vendedor y del comprador puesto que evidencia la transferencia de la propiedad de la mercancía.

Fondo de Caja.

Pequeña suma de dinero que se deja en la oficina para atender los gastos menores para los cuales no es aconsejable girar cheques.

G

Gastos

Costo de las mercancías y servicios utilizados en el proceso de generación del ingreso.

I

Ingresos

Aumento en el capital contable del propietario que se gana al entregarles bienes o servicios a los clientes.

L

Liberalismo

Es una corriente de pensamiento filosófico, social, económico y de acción política, que promueve las libertades civiles y el máximo límite al poder coactivo de los gobiernos sobre los seres humanos. Aboga principalmente por: El desarrollo de las libertades individuales y, a partir de ésta, por el progreso de la sociedad y el establecimiento de un Estado de Derecho, en el que todos los seres humanos incluyendo aquellos que en cada momento formen parte del Gobierno estén sometidos al mismo marco mínimo de leyes.

Libro Contable

Los libros contables son los libros que deben llevar obligatoriamente los comerciantes y en los cuáles se registran en forma sintética las operaciones mercantiles que realizan durante un lapso de tiempo determinado.

N

Nota Crédito

Documento emitido por el vendedor al comprador que indica la voluntad del vendedor para reducir (acreditar) la cuenta por cobrar del comprador como resultado de una devolución o rebaja en ventas.

Nota Débito

Documento emitido por el comprador al vendedor que señala la intención del comprador de reducir (debitar) la cuenta por pagar con el vendedor como resultado de una devolución o rebaja en compra.

P

Período Contable.

Espacio de tiempo al que corresponde un estado de Resultado. Los estados financieros pueden prepararse trimestralmente y mensualmente.

S

Servicio Autónomo

Se refiere a la oficina de registro.

Sistema Contable

Los métodos, procedimientos y mecanismos que una entidad utiliza para seguir la huella de las actividades financieras y resumir estas actividades en una forma útil para quienes toman las decisiones.

Transacción Se refiere a una acción terminada.