Universidad de las Ciencias Informáticas

Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores

Aylienn Aquino Leiva.

Yasser Linares Domínguez.

Tutor

Ing. Joiser Bruzón Estrada.

La Habana. Mayo, 2009

"Año 50 de la Revolución"

Pensamiento

Las manos matan, sudan, tiemblan, las manos cambian, las manos logran, hay manos obreras, hay manos que tocan producen sonidos, hay manos amigas, hay manos de ciencia, hay manos que influyen, empujan y cambian países…

yo decidí ser manos de ciencia…

yo decidí crear con mis manos…yo decidí vencer…

Datos del Contacto

Síntesis del Tutor: Ing. Joiser Bruzón Estrada

Profesión: Ingeniero en Ciencias Informáticas

***Años de graduado*:** 2

Agradecimientos

Aylienn Aquino Leiva

A todas las personas que me ayudaron y guiaron en la realización de este trabajo, en especial a Yadira por ser esa amiga incondicional, a Yasser por su apoyo, a nuestro tutor Joiser (Lelo), quien se preocupó y nos guió siempre, a Yorji por ayudarme tanto, por su comprensión, por estar allí cuando verdaderamente lo necesitaba.

A la familia ERP, que me ha ido enseñando a ser cada día más profesional.

A mis padres por ser siempre tan dedicados y preocupados por mis estudios y darme su voto de apoyo en mis decisiones.

A Yasmany por su cariño.

A mi familia en general por sus consejos.

A todos mis amigos, todos, los que de una forma u otra me enseñaron tantas cosas de la vida, de todo corazón gracias, un beso para todos, los quiero y los extrañaré mucho.

Yasser Linares Domínguez

A todos aquellos hombres que el paso de nuestra historia cubrió de gloria.

A la Revolución, que nos dio la posibilidad de lo alcanzado.

A mis padres y a mi hermano por el apoyo que siempre me han dado a lo largo de mi vida.

A mi tío Julio, por verme alumbrado los caminos de la informática.

A Aylienn: siempre presente, gracias por tu incondicional apoyo que solo así supimos sacar el trabajo a adelante.

A mis amigos Yadier, Jorge Luis, Alieski, Eraldo, Julio, Victor, Lelo, Yanelis, Yunei por su apoyo incondicional.

En fin a todos aquellos que de una forma u otra contribuyeron a mi educación como ingeniero informática y dieron razones para seguir adelante.

Dedicatoria

Aylienn Aquino Leiva

A mis papitos por su amor, su comprensión, su apoyo incondicional en todas mis decisiones, por ser tan lindos conmigo…..este trabajo se los dedico a ustedes que son mi fuente de inspiración…..los quiero con la vida……

Yasser Linares Domínguez

A mis padres, mi hermano y mi familia que son el motivo de inspiración en este trabajo. Así como a todas aquellas personas que de un modo u otro han formado y forman parte de mi vida y mi corazón....

Resumen

El desarrollo eficiente de una empresa se ve influenciado mayormente por el uso correcto de los Sistemas Integrales de Gestión Empresarial, pues posibilitan la optimización de los procesos empresariales (área de finanzas, logística, comercial, producción, entre otros), el acceso a toda la información de forma confiable, precisa y oportuna, viabilizando de esta manera la disponibilidad de la misma.

La gestión de la Contabilidad General es uno de los módulos que componen un Sistema Integral de Gestión, la misma se acompaña de diferentes procesos que involucran su total desarrollo, es quizás el elemento más importante en toda empresa o negocio, ya que permite conocer la realidad económica y financiera de la empresa, su evolución, sus tendencias y lo que se puede esperar de ella. La Contabilidad General no sólo permite conocer el pasado y el presente de una empresa, sino predecir el futuro.

El presente trabajo tiene como propósito informatizar dicho módulo. Se espera que el sistema sea usado por todas las empresas del país, además de poder comercializarlo con otras fuera del mismo, posibilitando entre otras ventajas que la información que se maneje sea verdadera, exacta, completa y clara, para que pueda ser bien utilizada en la toma de decisiones, facilitando el trabajo eficiente de las mismas.

Palabras Claves: procesos empresariales, Contabilidad General, empresa, Sistemas Integrales de Gestión Empresarial.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Introducción

La informatización de los procesos empresariales es uno de los mecanismos indispensables para el desarrollo económico de una empresa, en búsqueda del aumento en los niveles de eficiencia, organización de las actividades y optimización de los procesos.

Hoy los Sistemas Integrales de Gestión Empresarial son un eslabón importante y estratégico para todas aquellas empresas que pretenden desarrollarse. Cuba en aras de su desarrollo informático no se encuentra ajeno a lo antes mencionado, pues desde hace algunos años se inició en el proceso de informatización de la actividad económica. En este sentido se comenzaron a elaborar y explotar disímiles productos nacionales, pero la mayoría no respondían completamente a las exigencias de la economía cubana, pues generalmente estos estaban enfocados a un sector específico, de igual forma pasaba con los productos extranjeros, debido a que estos habían sido desarrollados en el marco de otra economía.

En la economía cubana se han producido una serie de cambios, uno de ellos es la existencia de dos monedas que coexisten y comparten legalmente las funciones del dinero, denominado dualidad monetaria. En Cuba, el peso cubano (CUP) es la moneda oficial y la que se reconoce como moneda base de los registros económicos, y el peso convertible (CUC) es el que comparte la situación de dualidad monetaria. De acuerdo a las normas cubanas de información financiera la moneda de presentación y de registro de los estados financieros en Cuba es en pesos cubanos (CUP), por eso se hace necesario conservar la moneda original y hacer la conversión utilizando la tasa de cambio a la moneda contable (CUP). Es importante aclarar que todas las transacciones económicas que se realicen se harán en la moneda original.

Otro de los cambios realizados, es la utilización de varias monedas en las transacciones económicas, conocido como multimoneda, una de sus características principales es que las transacciones originales se conservan en la moneda en que se realizaron y es en los procesos de registro donde se homogeniza la moneda aplicando la tasa de cambio según las normas establecidas, aspectos necesarios a tener en cuenta para un mejor desarrollo de los procesos contables.

Es importante resaltar aparte de lo citado anteriormente que dentro de las nuevas concepciones de informatización que se ha propuesto el país, está el desarrollo de productos informáticos relacionados con el empleo de herramientas libres y de tecnologías multiplataforma.

Debido a las características antes mencionadas sobre la economía cubana, es que se hace necesario por parte del Ministerio de Finanzas y Precios (MFP), en especial por la Dirección de Política Contable de conjunto con los demás Ministerios y Organismos del Estado, unido a estudiantes y profesores de la Universidad de las Ciencias Informáticas (UCI) la realización de un Sistema Integral de Gestión que unifique y estandarice toda la información que se genera, que sirva para mejorar la gestión económica de los recursos, y que además ayude a los directivos a la toma de decisiones; una solución que tiene particularidades que responden a la economía cubana y logre armonizar las normas cubanas con las normas internacionales, una solución única, integral y actualizada con las nuevas normas existentes.

Dicho sistema está constituido por una serie de módulos integrados, la gestión de la Contabilidad General constituye uno de ellos, siendo uno de los más indispensables pues es la base sobre la cual se sustentan el resto de las funcionalidades de una empresa, debido a que responde al control estricto de la actividad económica financiera de una entidad, así como de ser capaz de emitir un resumen de la información que refleje los elementos primordiales que sintetizan la situación económica de la misma; módulo que necesita adaptarse a las nuevas concepciones de informatización en el país, y resolver los problemas de la dualidad monetaria y multimoneda, dificultades que no se solucionaban de la mejor manera en el módulo Contabilidad General en los Sistemas Integrales de Gestión utilizados en Cuba.

Teniendo en cuenta lo expuesto anteriormente se plantea el siguiente ***problema****:*
Carencia de un sistema de gestión que aborde los procesos de la “Contabilidad General” que proporcione una solución a la dualidad monetaria y multimoneda, así como las nuevas concepciones de informatización en el país.

En consecuencia el ***objeto de estudio*** de la presente investigación es: Procesos Contables que pertenecen a Contabilidad General y específicamente el ***campo de acción*** se centra en la Implementación de los procesos de Contabilidad General.

Basado en la idea anteriormente expuesta se define como ***objetivo general*** del presente trabajo:
Desarrollar una solución informática para la Contabilidad General que proporcione una solución a la dualidad monetaria y multimoneda, así como su adaptabilidad a las nuevas concepciones de informatización en el país.

Para lograr dicho objetivo se plantearon las siguientes ***tareas investigativas***:

- Realizar un estudio de los principales sistemas de Contabilidad General existentes, siendo más específicos en los procesos que recoge dicho sistema, identificando sus características fundamentales y deficiencias.
- Realizar un estudio de las tecnologías, lenguajes y herramientas propuestas.
- Realizar un análisis de los artefactos[1] entregados por el flujo de Análisis y Diseño.
- Implementar el módulo Contabilidad General para la informatización de los procesos que en él se desarrollan.

Para desarrollar este trabajo se ha planteado la siguiente ***idea a defender:***
Si se implementa el módulo Contabilidad General, se contribuirá a la informatización de los procesos que se encuentran dentro del mismo, adaptándolo a las nuevas concepciones de informatización del país, y dándole una solución a la dualidad monetaria y la multimoneda.

La culminación e implantación de esta solución informática proporcionará a las entidades que la empleen los siguientes ***aportes prácticos*:**

- Disponer de una solución, que cumpla con todos los requerimientos establecidos.
- Dar la posibilidad de configurar sus Estados Financieros de acuerdo al tipo de entidad.
- Tener en cuenta lo establecido en la Resolución 14/2007[2] del MFP en cuanto a los datos de uso obligatorio de Mayor y Submayor.

[1] ***Artefactos:*** *Herramientas organizativas identificadas para la confección organizada y correcta del Software en cuestión.*

[2] *Consultor del DISAIC. Resolución 14/2007 del MFP. [Online] 3 2007.*

- Dar la posibilidad de la recuperación de los reportes mayor, submayor, balance de comprobación de saldos, utilidad acumulada y comprobante de operaciones en la moneda original y en cualquier otra moneda (moneda de registro y moneda contable).
- Permitir la revaluación de partidas monetarias de cuentas no controladas por otros módulos.
- Emitir el comprobante de operaciones en moneda contable y original, cumpliendo con lo establecido en la Resolución 235/2005[3] y 294/2005[4] y la resolución 9/2007[5] del MFP.
- Registrar asientos tipos de todos los módulos que agilizan los procesos contables.
- Solucionar los problemas relacionados con la dualidad monetaria, multimoneda, así como las nuevas concepciones de informatización en el país, referentes a la idea de la utilización de herramientas libre y tecnología multiplataforma.
- Contar con una fuerte integración con otros módulos de vital importancia para el correcto funcionamiento del sistema, tales como: costo y proceso, configuración, estructura y composición, entre otros.
- Permitir realizar todas las operaciones contables de forma rápida, precisa y segura.
- Permitir tener un control de los procesos dentro del mismo evitando la pérdida de datos por labores manuales, lentitud en los procesos, documentos repetidos.

[3] *Consultor del DISAIC. Resolución 235/2005 del MFP. [Online] 3 2007.*

[4] *Consultor del DISAIC. Resolución 294/2005 del MFP. [Online] 3 2007.*

[5] *Consultor del DISAIC. Resolución 9/2007 del MFP. [Online] 3 2007.*

La estructura del documento está constituida de la siguiente manera:

- ***En el Capítulo 1. Fundamentación teórica:*** Se desarrolla un estudio de los principales sistemas ERP existentes, tanto en Cuba como en el resto del mundo, además de realizar un mayor hincapié en el módulo de Contabilidad General presente en dichos sistemas. Se describen los lenguajes, las herramientas y metodologías a utilizar para el desarrollo de la aplicación.

- ***En el Capítulo 2. Descripción y análisis de la solución propuesta:*** Se desarrolla una valoración crítica del diseño propuesto por el analista, un análisis de posibles implementaciones, componentes o módulos ya existentes y que puedan ser reutilizados, estrategias de integración, descripción de los algoritmos no triviales a implementar, análisis de complejidad de los mismos, selección de las estructuras de datos apropiadas para la implementación de estos algoritmos y descripción de las clases utilizadas.

- ***En el Capítulo 3. Validación de la solución propuesta:*** Se desarrolla una búsqueda de los test de unidades que permitan validar la solución propuesta, se realiza una descripción del mismo teniendo en cuenta: objetivo del test, alcance y tipo de test, se describen los valores utilizados y se evalúa la ejecución del test y los resultados obtenidos. Además se aplican métricas con el objetivo de evaluar el diseño propuesto por el analista.

Capítulo 1: Fundamentacion Teórica

1.1 Introducción.

En el presente capítulo se realiza una síntesis acerca de los sistemas ERP[6] existentes tanto en Cuba como en el resto del mundo, su evolución a lo largo de la historia así como las principales empresas que se dedican a su creación y explotación, además de caracterizar los procesos referentes a la gestión de la Contabilidad General presentes en dichos sistemas, considerados también como Sistemas Integrales de Gestión Empresarial.

Otro aspecto tratado es todo lo referente a como se caracterizan las herramientas, tecnologías y metodologías usadas para el desarrollo de la aplicación, las cuales ya habían sido escogidas con anterioridad siguiendo un riguroso trabajo de selección, basándose en que las mismas se encuentra en constante evolución, por lo que se hace necesario tener un conocimiento avanzado y actualizado de estas a la hora de desarrollar aplicaciones informáticas, ya que muchas veces las herramientas usadas pueden quedar obsoletas, trayendo consigo una considerable disminución en la calidad de dichas aplicaciones.

1.2 Antecedentes y Estado del Arte.

El surgimiento de los sistemas ERP como herramientas estratégicas de trabajo se registra aproximadamente hace cuatro décadas atrás. La evolución de estos sistemas ha estado muy relacionada con el desarrollo que actualmente manifiestan las tecnologías de la información y las técnicas de gestión.

Los sistemas ERP que existen en la actualidad tuvieron sus antecedentes, los cuales a pesar de ser sistemas mejorables, fueron de gran utilidad en la Contabilidad General. Se tiene conocimiento de sistemas como el EQQ (Orden Económica de Cantidades) utilizado un poco antes de la década del 60. A partir de los años 60, surge una nueva técnica con el nombre MRP (Planificación de Pedidos de Material), técnica que era simple, pero en una situación real, generaba una enorme cantidad de datos, lo que lo hacía muy incómodo ser implantado en un ordenador. Con la llegada de los años 70 aparece una nueva técnica semejante al MRP, CL MRP (Planificación de Pedidos de Material de Ciclo Cerrado), en esta nueva técnica, la capacidad de producción pasó a ser tomada en cuenta.

[6] ***ERP:*** *Planificación de Recursos Empresariales.*

Alrededor de la década del 80 surge una versión del MRP conocida como MRP-II, que abordaba el concepto de la planificación de recursos de producción y la necesidad de ensanchar la gestión a otras áreas de la empresa.

A partir de este momento es que surge como tal el concepto de ERP asociado a términos como la planificación de negocios, la planificación de producción, tablas de tiempos de producción, la planificación de material y requisitos, planificación de capacidades y el funcionamiento del sistema para capacidades y prioridades. Pero el MRP-II sufrió también ciertos contratiempos producto de asumir elementos con carácter erróneo, como son los tiempos de producción fijos y las capacidades infinitas.

Alrededor de los años 90 y ya con todas las innovaciones tecnológicas de la época y su necesidad de expansión hacia otras áreas tan distinguidas como la Ingeniería, Finanzas, Recursos Humanos, Gestión de Proyectos, Servicios y Banca, surge el Enterprise Resource Planning (Software de Gestión Empresarial).

"Los ERP evolucionaron y continúan evolucionando hasta la actualidad, pues intentan acompañar el desarrollo de las propias tecnologías computacionales. Todo esto provocó que las empresas comenzaran a ver a los ERP como sistemas capaces de hacerlas mejorar." (1)

En la actualidad, la implantación de sistemas de gestión, sirven de soporte para la realización de una administración eficiente, brindando soluciones prácticas e integrales a problemas reales. Son sistemas estructurados que buscan satisfacer las demandas de soluciones de gestión empresarial, basados en el concepto de una solución completa que permita a las empresas unificar las diferentes áreas de productividad de la misma.

Existen proveedores a nivel mundial que marcan pauta en el mercado ERP, algunas de estas grandes empresas proveedoras de sistemas ERP a nivel mundial son:

- ***SAP:*** Fue fundada en 1972 en Alemania por cinco ingenieros de la IBM[7], siendo hoy la mayor empresa de su rama. "Su nombre se forma con las siglas en alemán: "Sistemas, Aplicaciones

[7] ***IBM:*** *International Business Machines (conocida coloquialmente como el Gigante Azul) es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática.*

y Procesamiento de datos". El primer producto que SAP desarrolló, fue comercializado bajo el nombre R/2. El dos significa los "niveles" en los que se implantaba el sistema: 1) servidor 2) cliente." (2) Siguiendo la evolución normal de cualquier sistema y atendiendo a las necesidades de sus clientes, en la década de los 80, el R/2 se mejora para dar como resultado el R/3[8] el cual fue optimizado para gestionar los procesos de producción y gestión, logística y recursos humanos. Hoy día, pasados más de 30 años, cuenta ya con más de 12 millones de usuarios, 64.500 instalaciones, 1.500 socios y 23 soluciones informáticas. Es considerada la mayor empresa proveedora de ERP a nivel mundial. En Cuba existen compañías como la Empresa de Telecomunicaciones de Cuba, Sociedad Anónima (ETECSA) que usan sus soluciones implementadas.

- ***PeopleSoft:*** Es el segundo mayor proveedor mundial, siendo su arma más fuerte los módulos de gestión de recursos humanos. La compañía PeopleSoft está actualmente por direccionar sus productos para las áreas de los servicios, con productos de control de costos. SAP y la Peoplesoft han mantenido un éxito continuo debido a la oferta de nuevas potencialidades a sus clientes, así como el constante aumento de clientes que son empresas conocidas mundialmente.
- ***Oracle:*** Produce y vende aplicaciones ERP desde 1987, siendo la mayoría de sus clientes empresas relacionadas a la producción y consumo de productos, siendo así un adversario directo de SAP. Curiosamente en cerca de un 80% de los casos, el software de SAP opera sobre una base de datos Oracle.
- ***Baan:*** Es una empresa holandesa y una fuerte competidora de SAP. Recientemente, tal como otros proveedores, ha dedicado especial atención al mercado de pequeñas y medianas empresas, hecho que tiene resultado en una enorme variedad de productos que ofrece así como un rápido retorno financiero.
- ***JDEdwards:*** A pesar de vender software ya hace muchos años, sólo se hicieron conocidos mundialmente hace muy poco. Desde que lanzaron el OneWorld como software ERP, consiguieron una importante cuota dentro del mercado mundial de ERP.

Es importante destacar que en los sistemas ERP existentes, la *gestión de la* ***Contabilidad General*** se convierte en una herramienta fundamental para el avance de una empresa, pues esta constituye la base sobre la cual se sustenta el resto de sus procesos. Este proceso tiene como objetivo

[8] ***R/3:*** *El número 3, indicativo de que ahora el sistema opera en tres niveles o capas: 1) servidor de base de datos, 2) servidor de aplicación (donde residirá el programa exclusivamente) y 3) cliente.*

fundamental que los involucrados en dicha actividad cuenten con un sistema que le permita incidir con mayor rigor en el control de la actividad económica financiera partiendo de que esta es el proceso sistemático de registrar, clasificar, medir, resumir e interpretar en términos monetarios los procesos que realizan las entidades, en caso de que no se cuente con los procesos es capaz de gestionar las funcionalidades de la entidad, la cual se sustenta en el cumplimiento eficaz del ciclo contable que comienza con el registro de la documentación primaria y concluye con la elaboración y presentación de los informes financieros.

Dentro de las 26 soluciones empresariales con las que cuenta Cuba tenemos el sistema integral económico creado por CITMATEL llamado **RODAS XXI**, el cual entró en el mercado en el año 1999. RODAS XXI posibilita informatizar el funcionamiento de cualquier empresa o unidad presupuestada, es un sistema en constante desarrollo que tiene muy en cuenta la opinión de los clientes para perfeccionarse. Es un sistema multimoneda o sea pueden realizarse operaciones tanto en moneda nacional como extranjera.
Este sistema cuenta con la integración de varios módulos entre lo que se encuentran: Inventario, Activos fijos, Nómina, Finanzas, Facturación, Recursos Humanos, Telecobranza y Contabilidad.

El módulo de Contabilidad de RODAS XXI: "Incluye importación, realización y reversión de comprobantes de operaciones. Establece cinco niveles de análisis contables, contando a las cuentas principales, subcuentas y tres análisis dentro de cada subcuenta. Contiene una gran variedad de informes que permiten obtener desde las facturas por edades de un cliente en particular hasta el comprobante específico donde se contabilizó una factura de un proveedor, además permite la configuración de estos informes. Ofrece gran variedad de reportes como son balance general, balance de comprobación, estado de resultados, análisis de cuentas por pagar y cuentas por cobrar." (3) Brinda la posibilidad de volver a períodos contables ya cerrados e incluso a períodos de años anteriores para visualizar informes correspondientes a dichos períodos, aunque por supuesto, en períodos ya cerrados no le permite realizar ninguna operación. Informatiza el proceso de cierre del año mediante una opción configurable que permite la realización del comprobante de cancelación de las cuentas de gastos e ingresos transfiriendo sus saldos a la cuenta resultado, quedando solamente por parte del usuario la realización del comprobante de traspaso de saldo de esta cuenta a las cuentas que correspondan antes de cerrar el año. Admite el registro de cheques tanto emitidos, como recibidos, así como de otros instrumentos y efectúa operaciones de cobros y de pagos con los mismos.

En la actualidad este sistema integral RODAS XXI se encuentra en más de 400 entidades del país en las que hay aproximadamente 1000 computadoras con el sistema instalado.

Otra solución nacional es el **SISCONT5**, sistema integral de contabilidad elaborado por el MINBAS, el cual se aplica en cientos de entidades en el país. Se encuentra en su versión número 5 y cuenta con 8 módulos integrados: Tesorería, Caja chica, Créditos y cobranzas, Gestión de negocios, Presupuestos, Informes gerenciales, Información compartida entre sistemas y Contabilidad.

El módulo de Contabilidad de SISCONT5: Fue el primer módulo que se creó. Cuenta con fácil ingreso de datos (en una sola pantalla se ingresan todos los datos del documento). Actualiza en línea. Exporta al Excel 2003 para generar informes y formatos. Genera la diferencia de cambio y asientos automáticos. Al realizar el cierre del mes de los procesos contables de cobro y pago si existe diferencia entre submayor, histórico y cuenta, SISCONT5 posibilita al cliente dar solución a este problema. Además cuando se comienzan a realizar operaciones de cobros anticipados y pagos anticipados este sistema integral de contabilidad emite un mensaje si la apertura no ha sido cerrada previamente. No se requiere de conocimientos contables para el manejo de este módulo.

Con SISCONT5 se puede trabajar en forma monousuario, multiusuario y por Internet, dependiendo de el tipo de hardware (dispositivo).

El **VERSAT - Sarasola** fue el primer sistema cubano que logró certificarse en el país. Es un sistema integrado de gestión económica, diseñado para ser utilizado por el sector empresarial cubano. Sistema que es utilizado por diversas entidades, de cada una de las provincias del país, tanto del sector público, empresas mixtas, productores agropecuarios (UBPC y CPA), además del Ministerio de la Azúcar (MINAZ) y la Oficina Nacional de Administración Tributaria (ONAT). Resulta ser un sistema económico integrado, constituido por 12 módulos que incluyen Configuración y Seguridad, Costos y Procesos, Análisis Económico Empresarial, Control de Activos Fijos, Gastos y Contabilidad General.

El módulo de Contabilidad General del VERSAT – Sarasola: Constituye el corazón del sistema, rector de todo el VERSAT, al resumir las necesidades y requerimientos del registro contable. Hacia este módulo tributan todas las operaciones contables de los restantes subsistemas. En este

subsistema, los comprobantes de operaciones a procesar manualmente son mínimos, pues la casi totalidad de los mismos serán generados por los restantes módulos. El módulo posee cinco funciones básicas: comprobantes de operaciones, clasificador de cuentas, análisis de las cuentas, costos y procesos y estados financieros.

También existen ERP de software libre que realiza la gestión de la Contabilidad General entre los que podemos citar:

- ***AbanQ***

Es un software innovador y muy útil para las empresas, creado por Infosial[9]. Sistema orientado a automatizar distintos procesos o actividades en la PYME[10], que pueden pasar desde lo financiero hasta lo comercial. Entre sus ventajas, están la de ofrecer a las PYMES su propia plataforma tecnológica, es decir, sin licencias y pagos por sistemas automatizados comerciales. Dispone de varios módulos que permiten dar funcionalidad a un sin número de actividades y procesos: Facturación, Almacén del Área de Facturación, Informes del Área de Facturación, Informes, Contabilidad, Informes de Contabilidad.

El módulo de Contabilidad: Esta relacionado con todo lo que tenga que ver con el área contable. Esta integrada con los módulos de facturación y tesorería, aunque puede utilizarse independientemente.
El módulo de Informes de Contabilidad: Brinda reportes y datos enfocados principalmente en el módulo contable, integra de manera sencilla y rápida todo un proceso amplio en la empresa.

- ***Openbravo***

Es un "sistema de gestión empresarial, integrado y modular. Basado en web y con el valor añadido de ser software libre. Específicamente diseñado para PYMES, es idóneo para las empresas que buscan una solución que puedan implementar rápidamente, aprendiendo a utilizarla fácilmente y personalizándola a sus necesidades, siendo propietario real de la aplicación y solo pagando por el soporte profesional." (4)
Sistema que permite realizar un análisis financiero de toda la obra comparando los costos estimados y reales e identificando las variaciones y fuentes de los mismos, con el fin último de optimizar todos y

[9] ***Infosial:*** *Empresa informática dedicada a la investigación y desarrollo de software, servicios de Internet, servicios informáticos y formación.*

[10] ***Pyme:*** *Es el término técnico con el que se les conoce a la Pequeña y Mediana Empresa, o Small Business en inglés.*

cada uno de los recursos materiales, técnicos, humanos y de tiempo destinados a la realización de la obra. Openbravo ERP dispone una serie de funcionalidades para dar soporte a todas las áreas funcionales de una PYME sofisticada: Datos Maestros, Gestión de Aprovisionamiento, Gestión de Almacenes, Gestión de Proyectos, Gestión de Servicios, Gestión de la Producción, Gestión Comercial y Gestión Avanzada de Clientes, Gestión Económico-Financiera, Inteligencia de Negocio.

La Gestión económico-financiera: Orientada a la minimización de la introducción manual de datos actuando como recolector automático de todas las actividades generadas en el resto de áreas de gestión.
La Inteligencia de negocio: Facilita toda la información relevante para la toma de decisiones mediante la monitorización de una serie de indicadores clave para la organización.

- ***OpenERP***

Es una herramienta potente y flexible, que cuenta con una organización formada por varias empresas españolas conocedoras y distribuidoras del producto, creada con el objetivo de difundir y facilitar la implantación a las áreas de negocio alrededor del mismo. Es un sistema de gestión empresarial que cubre las necesidades de las aéreas de contabilidad, ventas, compras, y almacén e inventario, entre otras. OpenERP soporta múltiples monedas, múltiples compañías y múltiples contabilidades; además incorpora funcionalidades de gestión de documentos para agilizar la colaboración entre departamentos y equipos en la empresa; y permite trabajar remotamente mediante una interfaz web desde una computadora conectada a Internet. Entre sus principales funcionalidades tenemos: Gestión de Compras, Gestión de Ventas, Gestión de Inventario, Gestión de Casos y Solicitudes, Gestión Contable y Financiera.

El módulo Gestión Contable y Financiera: Provee de una contabilidad general, analítica y presupuestaria, y cuenta con todas las funcionalidades para llevar los libros contables en forma rigurosa. Este módulo permite definir centros de costos, gestionando de una manera eficiente la contabilidad analítica en su organización, dispone de contabilidad automática de doble entrada, que se combina con una gran cantidad de herramientas de informes y análisis totalmente integradas. El módulo de contabilidad financiera está pensado para gestionar los datos económicos de su empresa, siendo posible la utilización de múltiples planes de cuentas de manera simultánea. Este módulo permite la generación de presupuestos, informes, entre otros. Con este módulo la tesorería puede gestionar los flujos de caja y el efectivo con un alto nivel de trazabilidad.

ASSETS NS es un sistema comercializado por la firma panameña D´MARCO S.A. y distribuido en Cuba por INFOMASTER, entidad informática perteneciente a la empresa nacional de producción y servicios a la educación superior del mes. Este es un sistema de gestión integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos. Como sistema integral todos sus módulos trabajan en estrecha relación, generando, automáticamente, al ***Módulo de Contabilidad*** los Comprobantes de Operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de contabilidad al día. ASSETS NS está diseñado para multi-compañía, con una estructura organizativa a varios niveles, en la que podrán existir: Grupo Corporativo, Corporativo, Grupo de Agrupaciones, Agrupación, Almacenes y Centros de Costos. Para entidades con esta estructura se brinda un módulo de Comunicaciones que facilita poder intercambiar información entre ellas, con el fin de consolidar información sobre la Gestión Comercial y Contable, pudiéndose obtener los estados financieros, resúmenes de compras, ventas, entre otros a distintos niveles.

Los sistemas citados anteriormente abarcan la mayoría de los procesos de una empresa, encontrándose el módulo Contabilidad General responsable de alguno de estos procesos; pero estos tenían sus inconvenientes pues sufrían el efecto de decisiones que no se habían previsto en sus configuraciones, además de que no todos los sistemas analizados como parte de la investigación cumplen con la condición de dualidad monetaria y multimoneda, en el caso de los sistemas extranjeros solo cumplen con la multimoneda, y los sistemas nacionales si cumplen con ambas funcionalidades ya que esto fue un requerimiento que les impuso el Ministerio de Finanzas y Precios para poder certificar dichos sistemas en el país, por lo que este tema no se trató con la profundidad que lo requería, no cumpliéndose así requisitos indispensables entre los que se encuentran el de ser configurable, previendo posibles cambios.

1.3 Tecnologías, lenguajes de programación y Librería

En la actualidad con el gran avance que va teniendo el mundo de la informática se han ido desarrollando nuevas tecnologías, se han consolidados los lenguajes de programación y aparejado con el desarrollo y crecimiento de internet, se consolida todo lo relacionado con la creación de aplicaciones web, las cuales ofrecen grandes facilidades para establecer comunicación con el usuario

mediante las páginas Web, por su capacidad para ser visualizadas desde cualquier parte del mundo haciendo uso de un navegador.

1.3.1 Metodología de Desarrollo

Todo desarrollo de software es riesgoso y difícil de controlar, si no se utiliza una metodología que guíe el proceso se obtienen clientes insatisfechos con los resultados que ellos esperan alcanzar del producto final, además de que no va a existir una guía sólida que le facilite el trabajo a los integrantes del equipo de trabajo.

1.3.1.1 Proceso de desarrollo de software.

Un proceso de desarrollo de software define QUIÉN está haciendo QUÉ, CUÁNDO y CÓMO alcanzar un objetivo. En la ingeniería de software el objetivo es construir un producto de software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad. Captura y presenta las mejores prácticas que el estado actual de las tecnologías permite. En consecuencia reduce el riesgo y hace el proyecto más predecible.

No existe un Proceso de Software Universal, esto se debe a las características de cada proyecto, los equipos de desarrollo, los recursos, los riesgos, entre otros, por lo tanto se exige que los procesos sean configurables.

Es necesario usar la metodología adecuada para reducir el tiempo de desarrollo del software y aumentar la calidad del producto, obviamente se debe haber llegado a un acuerdo formal con el cliente al inicio del proyecto, de tal manera que cada cambio no represente un problema. La metodología utilizada en este trabajo se explica a continuación.

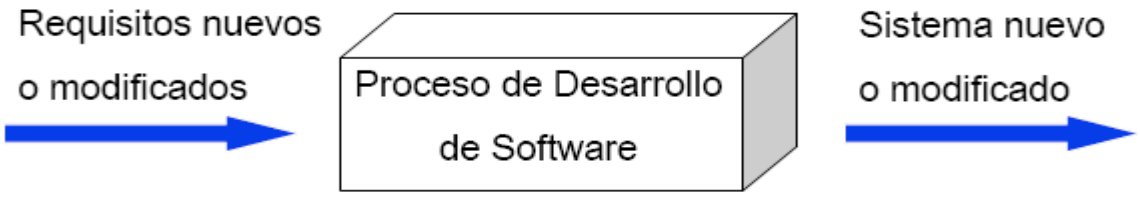

Figura 1 Entrada y salida de un proceso de desarrollo de software.

1.3.1.2 Modelo de Desarrollo

La metodología de desarrollo utilizada para el avance de la aplicación se caracteriza por no ser una metodología pura, sino un híbrido de estas. En este modelo de desarrollo se definió como estaría

estructurada la línea de desarrollo, se delimitaron los roles y las responsabilidades por cada uno, así como los artefactos que producirían, se decidió como se iba a comportar el flujo de actividades, se planteó la descripción de las misma, así como quienes participarían en su cumplimento. A continuación se muestra dicho Modelo de Desarrollo.

1.3.1.2.1 Organigrama de las Líneas de Desarrollo.

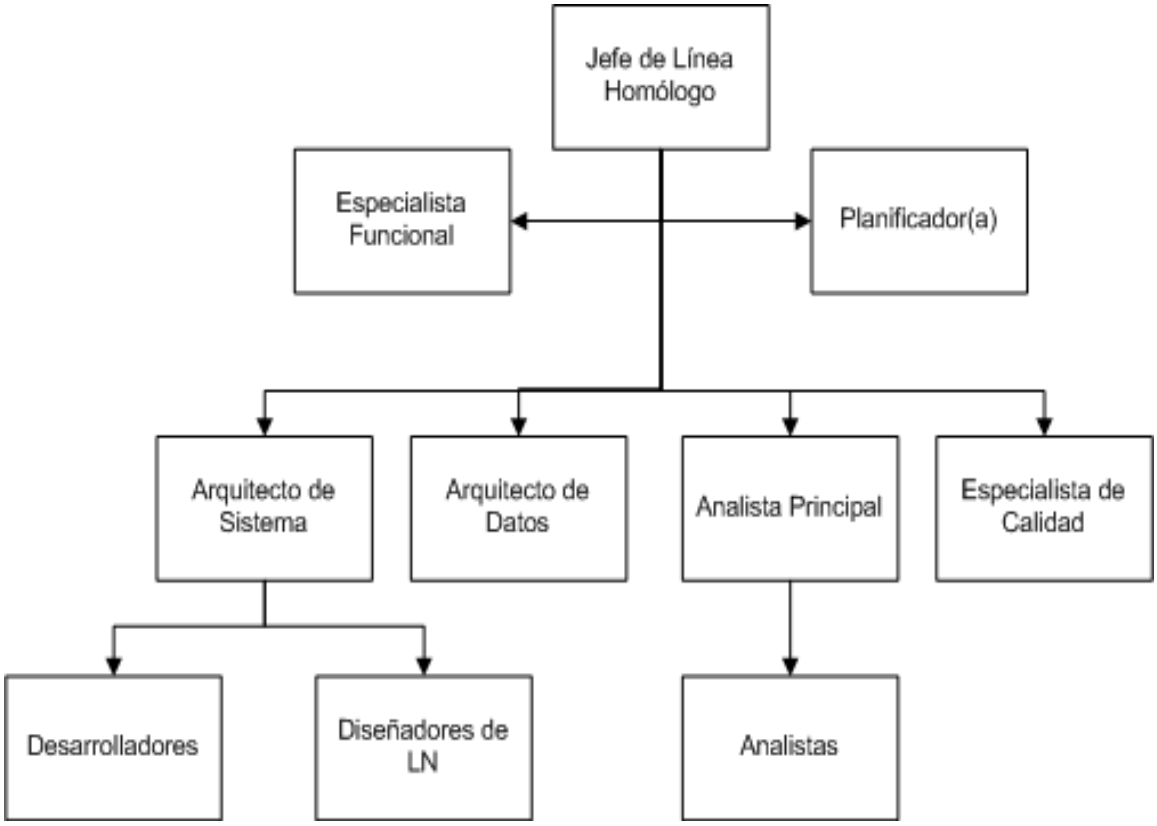

Figura 2 Organigrama de la Línea de Desarrollo.

1.3.1.2.2 Distribución de responsabilidades por rol.

Roles	Responsabilidades
Jefe de Línea de Desarrollo	Responsable de garantizar los cronogramas y compromisos de la línea Supervisar el proceso de desarrollo Organiza y controla el trabajo de los miembros de su línea Controla los indicadores de eficiencia
Planificador	Mantener actualizado el cronograma Mantener actualizada la plantilla de Capital Humano. Planificar y controlar las tareas de los miembros del equipo, según las prioridades Controlar los horarios de trabajo y distribución de máquinas. Llevar las actas de las reuniones y talleres. Controlar los planes de trabajo Individuales
Arquitecto de Sistema	Que se cumplan las políticas y estándares definidos en la Arquitectura. Las decisiones de integración en el proyecto y la Arquitectura del Sistema. Modera el Taller de Diseño.
Arquitecto de Datos	Construye y actualiza el Modelo de Datos, además responde por el manejo y recuperación de la información del mismo.
Analista Principal	Dirigir y organizar el trabajo del grupo de analistas de la Línea. Elaborar el Mapa de Procesos de la Línea según los estándares. Participar en la definición y construcción de la Arquitectura de Negocio del ERP. Mantener actualizados el seguimiento a los requerimientos de su línea de desarrollo.
Especialista de Calidad	Revisar, controlar las normas y estándares que establece el grupo de aseguramiento de la calidad incluyendo el proceso de desarrollo. Guiar al grupo de auditoría y revisiones Coordinar el proceso de diseño de casos de prueba. Coordinar las pruebas de aceptación o liberación.
Especialista Funcional	Participar en las sesiones de trabajo para identificar, describir y validar los procesos de negocio y los requisitos de software Validar, desde el punto de vista funcional, los procesos de negocio y

	requisitos de software Elaborar Casos de Prueba según los estándares establecidos para ello
Analista	Participar en las sesiones de trabajo para identificar, describir y validar los procesos de negocio y los requisitos de software Elaborar la Descripción de Procesos de Negocio, Especificación de Requisitos y Casos de Prueba según los estándares establecidos para ello Participar en el Taller de Diseño
Desarrollador	Diseña y Construye los componentes de software de la línea

Tabla 1 Distribución de responsabilidades por rol en el Modelo de Desarrollo.

1.3.1.2.3 Flujo de Actividades

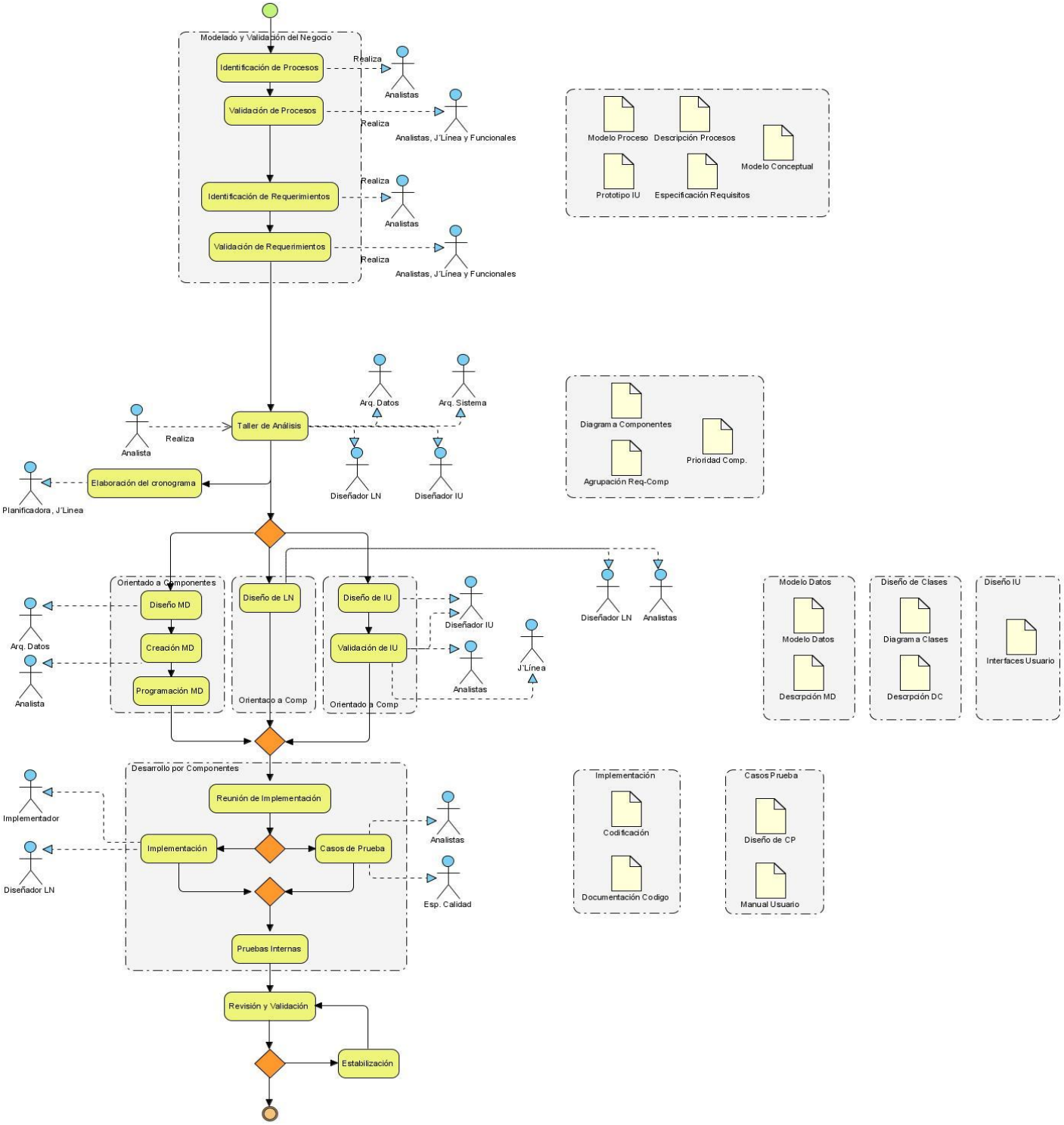

Figura 3 Flujo de Actividades del Modelo de Desarrollo.

1.3.1.2.4 Descripción de las Actividades

Actividades del Desarrollo	Descripción	Participan
Identificación de Procesos	Se debe identificar, analizar y describir los procesos que se llevan a cabo en el negocio que se desea automatizar, con el objetivo de organizar y documentar todas las acciones a tener en cuenta en el análisis para el desarrollo del Software	- Analistas - Especialistas Funcionales - J´de Línea de Desarrollo
Validación de Procesos	Se aprueban que la identificación de los procesos se hiciera correctamente y que el equipo tenga plena claridad del negocio a automatizar.	- Analistas - Especialistas Funcionales - J´de Línea de Desarrollo
Identificación de Requerimientos	A partir de los procesos identificados se realiza para cada uno de ellos la identificación de los requisitos o funcionalidades que debe cumplir, para que pueda ser realizado dicho proceso.	- Analistas - Especialistas Funcionales - J´de Línea de Desarrollo
Validación de Requerimientos	Se validan todos y cada uno de los requisitos identificados para cada uno de los procesos que intervienen en la automatización del negocio deseado.	- Analistas - Especialistas Funcionales - J´de Línea de Desarrollo
Taller de Análisis	Se evalúan cada uno de los requerimientos y procesos identificados y validados, a partir de los cuales se desarrollan el mapa de procesos a través del cual: Se agrupan los requerimientos y procesos por componentes. Se identifican las dependencia entre los componentes, así como los contratos de los mismo en el mismo modulo.	- Analistas - Arquitecto de Datos - Arquitecto de Sistema - Diseñador de LN - Desarrollador de IU

	Se establecen las prioridades de desarrollo de cada uno de los componentes, dependiendo de cuan críticos y complejos sean.	
Elaboración del Cronograma	Teniendo claro el alcance del módulo a partir de la identificación de componentes y la prioridad de desarrollo de los mismos se elabora el cronograma de desarrollo teniendo en cuenta los riesgos, costos, personal y productividad del equipo.	- Planificadora - J´de Línea de Desarrollo
Diseño del MD	Partiendo de un modelo lógico obtenido durante la captura de requisitos, se definen las estructuras de base de datos que darán soporte de persistencia a la solución de software orientada a los componentes identificados.	- Arquitecto de Datos - Analistas
Creación del MD	Creación de las estructuras y objetos de base de datos en el sistema de gestión seleccionado orientada a los componentes identificados..	- Arquitecto de Datos - Analistas
Programación del MD	Consiste en implementar la capa de acceso a datos orientada a los componentes identificados. Tiene que estar creada la base de datos y las entidades de dominio. - Realizar los ficheros de mapeo: Consiste en crear los ficheros de mapeo mediante el Doctrine. - Programación de interfaces: Consiste en crear las clases que implementan las interfaces de los DAOs. - Realizar pruebas unitarias: Realizar pruebas unitarias a las implementaciones de los DAOs.	- Arquitecto de Datos - Analistas
Diseño de LN	Se diseñan los métodos y clases para dar solución a todas las necesidades detectadas durante la identificación de componentes, ajustándose a las funcionalidades previstas.	- Diseñadores de LN - Analistas
Diseño IU	Se diseñan las interfaces de interacción con el usuario	- Diseñadores de IU

	en dependencia de las funcionalidades y componentes detectados	- Analistas
Validación IU	Se valida que todas las interfaces elaboradas están acorde con las funcionalidades necesarias a automatizar.	- Diseñadores de IU - J´de Línea de Desarrollo
Reunión de Implementación	Se realiza una breve descripción con los implementadores explicándole las órdenes de desarrollo y explicándole de forma operativa la Lógica de Negocio.	- Diseñadores de LN - Analistas - Desarrolladores
Implementación	Implementan toda la lógica de negocio diseñada (orientada a componentes) orientándose por la priorización de los componentes detectados en el taller de análisis	- Implementadores - Diseñadores de LN
Casos de Prueba	Construcción de todos los posibles caminos de ejecución, o escenarios, de cada componente desarrollado. Se obtiene como resultado un listado final con los casos de prueba identificados a partir de los posibles escenarios, los resultados esperados para cada caso y las condiciones o valores requeridos para la ejecución de los distintos escenarios.	- Analistas - Especialista de Calidad
Pruebas Internas	Se realizan pruebas internas del sistema antes de incorporarlo al equipo central de calidad para que realice la pruebas de liberación, tratando de garantizar las detecciones de la menor cantidad de no conformidades.	Especialista de Calidad
Revisión y Validación	Se realizan pruebas de calidad dirigidas por el equipo central de calidad, prueban cada uno de los juegos de datos así como los casos de prueba desarrollados por las líneas. Se realizan además tantas iteraciones como sean necesarias para garantizar la calidad del sistema a desplegar.	Equipo central de Calidad
Estabilización	Se dan respuesta a las no conformidades detectadas en las pruebas de calidad.	Línea de Desarrollo

Tabla 2 Descripción de Actividades en el Modelo de Desarrollo.

1.3.1.2.5 Representación de los artefactos por roles

Roles	Artefactos
Jefe de Línea de Desarrollo	Plan de Iteración Plan de Gestión de Riesgos Plan de Trabajo individual de los integrantes de la Línea
Planificador	Plan de Iteración Plan de Trabajo Individual de los profesionales Definición del cronograma de desarrollo
Arquitecto de Sistema	Plan de Trabajo Individual Diagrama de componentes Prioridad de los componentes Agrupación Requerimientos - Componentes Informe de Integración
Arquitecto de Datos	Plan de Trabajo Individual Modelo de Datos Descripción del Modelo de Dato
Analista Principal	Plan de Trabajo Individual Mapa de Procesos de la Línea
Analista	Plan de Trabajo Individual Modelo de procesos de negocio Descripción de procesos de negocio Modelo Conceptual Prototipo de IU Especificación de requisitos. Casos de Prueba
Especialista de Calidad	Plan de Trabajo Individual Plan de pruebas Casos de Prueba Registro de No Conformidades
Especialista Funcional	Plan de Trabajo Individual Casos de Prueba Validación de Procesos y Requisitos

Desarrollador	Plan de Trabajo Individual Implementación de componentes Descripción de los componentes
Diseñador de LN	Diagrama de Clases Descripción del Diseño de Clases

Tabla 3 Representación de los artefactos por roles en el Modelo de Desarrollo.

1.3.2 Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

1.3.2.1 Programación Orientada a Objetos.

La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). La POO expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

"La POO introduce nuevos conceptos, que superan y amplían percepciones antiguas ya conocidas, maneja definiciones nuevas tales como: Clase, Objeto, Método, Evento, Mensaje, Atributo, Estado Interno, Componentes de un Objeto y Representación de un Objeto, además de un conjunto de características propias como son: Abstracción, Encapsulamiento, Principio de Ocultación, y Polimorfismo que debido a la importancia que representan las mismas se hace necesario mencionar en este trabajo." (5)

1.3.2.2 PHP 5.2

"PHP es un lenguaje de programación interpretado[11], diseñado especialmente para la creación de páginas web dinámicas y puede ser embebido dentro de código HTML. Generalmente se ejecuta en

[11] *Un **lenguaje de programación interpretado** requiere de un programa auxiliar (el intérprete), que traduce los comandos de los programas según sea necesario.*

un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores." (6)

Características de PHP:

- Dispone de una conexión propia a varios sistemas de base de datos como: MySQL, PostgreSQL y Oracle.
- Incorpora bibliotecas que contienen funciones integradas para realizar útiles tareas relacionadas con la Web. Puede generar imágenes GIF al instante, establecer conexiones a otros servicios de red, enviar correos electrónicos, trabajar con cookies y generar documentos PDF, todo con unas pocas líneas de código.
- Es un producto de código abierto[12], soportado por una gran comunidad de desarrolladores que se encargan de encontrar y reparar los fallos de funcionamiento.
- Es un lenguaje multiplataforma.
- Permite las técnicas de Programación Orientada a Objetos.
- No requiere definición de tipos de variables.
- Posee tratamiento de errores.

PHP 5.2 es una versión de PHP que además incluye:

- Soporte sólido para Programación Orientada a Objetos (OOP) con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte a XML.

Es un lenguaje sencillo, de sintaxis cómoda y dispone de muchas librerías que facilitan en gran medida el desarrollo de las aplicaciones; convirtiéndolo en el favorito de millones de programadores en todo el mundo.

1.3.2.3 Librería ExtJS 2.2

ExtJS es una librería de componentes que facilita las herramientas necesarias para la creación de aplicaciones Web con excelentes gráficos; ya que posee una considerable colección de elementos para el diseño de interfaces, ventanas, pestañas, menús, tablas, entre otros.

Brinda soporte para:

- Construir interfaces gráficas complejas y dinámicas.
- Comunicar datos de forma asíncrona con el servidor.

[12] ***Código Abierto:*** *Es el término con el que se conoce al software distribuido y desarrollado libremente.*

- Diversos navegadores como: Internet Explorer, Firefox, Safari y Opera.

Actualmente ExtJS es considerado un FrameWork independiente; ya que a principios del 2007 se creó una compañía para comercializar y dar soporte al mismo, dicha compañía proporciona los servicios de consultoría necesarios para ayudar a los clientes en el aprovechamiento máximo de las ventajas de ExtJS. Es importante señalar que la ExtJS 2.2 tiene dos tipos de licencias, LGPL[13] y la comercial, esta última es obligatoria si se desea obtener soporte.

1.3.2.3.1 AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML) facilita la creación de aplicaciones interactivas en la Web que se ejecutan en el navegador de los usuarios y mantienen comunicación asíncrona con el servidor; posibilitando realizar cambios sobre una página sin necesidad de recargarla, aumentando de esta forma la interactividad, velocidad y usabilidad de la misma.

"AJAX no es una tecnología en sí mismo. En realidad, se trata de la unión de varias tecnologías que se desarrollan de forma autónoma y que se unen de formas nuevas y sorprendentes." (7)

AJAX está conformado por:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM[14], para la interacción y manipulación dinámica de la presentación.
- XML, XSLT[15] y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest[16], para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

Provee de un mecanismo para mezclar y hacer coincidir XML con XHTML, sus aplicaciones son más rápidas e interactivas al estilo de aplicaciones de escritorio, reduce de manera significativa tener que cargar información continuamente del servidor actualizando solamente porciones de la página,

[13] ***Licencia LGPL:*** *Permite que los desarrolladores utilicen programas bajo la GPL o LGPL sin estar obligados a someter el programa final bajo dichas licencias.(Ver Glosario)*

[14] ***DOM:*** *Es una abreviatura de Document Objet Model. En español Modelo de Objeto de Documento. El DOM es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.*

[15] ***XSLT*** *o Transformaciones XSL es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML.*

[16] *Un objeto* ***XMLHttpRequest*** *es una instancia de una API que nos permite la transferencia de datos en formato XML desde los script del navegador (JavaScript, JScrip, VBScript ...) a los del servidor (PHP, Perl, ASp, Java ...) e inversamente.*

además de que si se utiliza adecuadamente en el desarrollo de una aplicación, se reduce de manera significativa los tiempos de carga inicial.

1.3.2.3.2 XML

Es el estándar de Extensible Markup Language (Lenguaje de Etiquetado Extensible), conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que son: extensibilidad, estructura y validación.

Ventajas de XML:

- Las aplicaciones se pueden generar rápidamente y su mantenimiento es más sencillo.
- Separa los datos de la presentación y del proceso, lo que permite mostrar y procesar los datos al gusto deseado con sólo aplicar distintas hojas de estilo y aplicaciones.
- La información es más accesible y reutilizable, por la flexibilidad de las etiquetas de XML que permiten su utilización sin tener que amoldarse a reglas específicas de un fabricante.

Además, ofrece un formato para la descripción de datos estructurados, facilitando declaraciones de contenido más precisas y resultados de búsquedas más significativos en varias plataformas.

1.3.2.3.3 JSON

“JSON, es el acrónimo en inglés de JavaScript Object Notation (notación de objetos JavaScript). Es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador semántico de JSON. En JavaScript, JSON puede ser analizado trivialmente usando el procedimiento eval(), lo cual ha sido fundamental para la aceptación de JSON por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web.” (8)

1.3.2.3.4 XHTML

"XHTML es el acrónimo en inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto). Es una versión más estricta y limpia de HTML[17], que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos." (9)

XHTML reúne la capacidad de formato de HTML y se consolida con la formalidad del XML a la hora de estructurar documentos para la portación de datos. Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. *Algunos de estos requisitos son:* 1) Elementos correctamente anidados, 2) Etiquetas en minúsculas, 3) Elementos cerrados correctamente y 4) Atributos de valores entrecomillados.

1.3.2.3.5 CSS

Es un lenguaje de hojas de estilos (Cascading Style Sheets) creado para controlar la presentación de documentos estructurados y escritos en HTML y XHTML, aspectos como: el color, el tamaño, el tipo de letra, la separación entre párrafos y la tabulación con la que se muestran los elementos de una lista. El propósito del desarrollo de CSS es separar la estructura y el contenido de la presentación estética en un documento, esto permite un control mayor del documento y sus atributos, convirtiendo al HTML en un documento muy versátil y liviano.

"Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes." (10)

Entre los beneficios concretos de CSS se encuentran: (11)

- Control de la presentación de muchos documentos desde una única hoja de estilo.
- Control más preciso de la presentación.
- Aplicación de diferentes presentaciones a diferentes tipos de medios (pantalla, impresión, entre otros.)

[17] ***HTML:*** *Lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto que es el formato estándar de las páginas web. (Ver Glosario)*

1.3.2.3.6 JavaScript

"Es un lenguaje de tipo script compacto, basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet. Los programas JavaScript van incrustados en los documentos HMTL, y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos." (12) Los programas escritos con este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios, convirtiéndolo en un lenguaje interpretado.

Ventajas de JavaScript:

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

1.3.3 Herramientas utilizadas

1.3.3.1 Herramientas CASE

"CASE es una sigla, que corresponde a las iniciales de: **C**omputer **A**ided **S**oftware **E**ngineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computación. Las herramientas CASE representan una forma que permite Modelar los Procesos de Negocios. Un elemento importante conveniente de destacar, es que las herramientas CASE, son eso: "HERRAMIENTAS", y que como tales permiten aumentar la productividad en el desarrollo de un proyecto y como herramientas que son, deben ser aplicadas a una metodología determinada." (13)

1.3.3.1.1 Visual Paradigm

"Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación." (14)

Visual Paradigm forma parte del IDE[18] de Eclipse. Está diseñado para desarrollar software con Programación Orientada a Objetos, examina reducir la duración del ciclo de desarrollo brindando ayuda tanto a arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores.

Dentro de sus características fundamentales están:

- ***Multiplataforma:*** Soportada en plataformas Java para Sistemas Operativos Windows, Linux y Mac OS X.
- ***Interoperabilidad:*** Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XMI[19], XML y archivos Excel. Importa archivos de proyectos de Rational Rose. Integración con Microsoft Office Visio.
- ***Modelamiento de los Requisitos:*** Captura de requisitos con diagrama de requisitos, modelamiento de casos de uso y análisis textual.
- ***Colaboración de Equipo:*** Realiza el modelado en colaboración y simultáneamente con el Visual Paradigm TeamWork Server y Subversion.
- ***Generación de Documentación:*** Comparte y genera los diagramas y diseños en formatos como PDF[20], HTML y Microsoft Word.
- ***Editor de Detalles de Casos de Uso:*** Entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ***Ingeniería de Código:*** Permite generación de código e ingeniería inversa en lenguajes como Java, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash ActionScript, Delphi, Perl y Rugby.
- ***Modelado de Procesos de Negocio:*** Visualiza, comprende y mejora los procesos de negocio con la herramienta muy completa para estos procesos.
- ***Integración con Entornos de Desarrollo:*** Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación, en IDE como Eclipse, Microsoft Visual Studio, NetBeans, Sun ONE, Oracle JDeveloper, JBuilder y otros.

[18] ***IDE:*** *Entorno de Desarrollo Integrado del inglés Integrated Development Environment.*

[19] ***XMI:*** *Es un estándar para el intercambio de información y metainformación entre herramientas, repositorios y aplicaciones, que proporciona un formato de intercambio para entornos distribuidos usando XML.*

[20] ***PDF:*** *(acrónimo del inglés **P**ortable **D**ocument **F**ormat, formato de documento portátil) es un formato de almacenamiento de documentos*

- ***Modelamiento de Bases de Datos:*** Generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.

1.3.3.2 Servidor Web

Apache 2.0 es un servidor Web potente, flexible y disponible para distintas plataformas y entornos. Es altamente configurable de diseño modular, posibilitando que los administradores de sitios Web puedan elegir los módulos que serán incluidos y ejecutados en el servidor.

Características de Apache:

- Es una tecnología gratuita y de código abierto, lo que proporciona transparencia en todo el proceso de instalación.
- Es prácticamente universal, por su disponibilidad en multitud de sistemas operativos.
- Posee una alta configurabilidad en la creación y gestión de logs, de este modo es posible tener un mayor control sobre lo que sucede en el servidor.

Este servidor Web tiene una fácil integración con varios lenguajes de programación como: Java, Perl y especialmente PHP. Dicha relación a dado a lugar el desarrollo de aplicaciones como el APPSERV y XAMPP los cuales instalan el Apache y el PHP configurados para su uso.

1.3.3.3 Sistema Gestor de Base de Datos

PostgreSQL 8.3 es un gestor de bases de datos relacional orientada a objetos, libre y gratuito. Está liberado bajo la licencia BSD, lo que significa que se puede disponer de su código fuente, modificarlo a voluntad y redistribuirlo libremente.

Presenta las siguientes propiedades:

- ***Atomicidad:*** Asegura la realización de una operación, por lo que ante un fallo del sistema esta no queda a medias.
- ***Consistencia:*** Posibilita la ejecución de aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.
- ***Aislamiento:*** Mediante un sistema denominado MVCC (Acceso concurrente multiversión) asegura que una operación no pueda afectar a otras, de esta manera dos transacciones sobre la misma información no genera error.

1.3.3.4 PGAdmin III

"Es una aplicación gráfica usada para la gestión de PostgreSQL, siendo la más completa y popular con licencia Open Source[21]. PGAdmin está escrito en C++ y utiliza la librería gráfica multiplataforma wxWidgets, permitiendo que se pueda usar en sistemas operativos como: GNU/Linux[22], FreeBSD, Solaris, Mac OS y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3, ejecutándose en cualquier plataforma.

PGAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración." (15)

1.3.3.5 ZendStudio for Eclipse

ZendStudio 6.0 es un IDE de desarrollo para aplicaciones de la Web 2.0, gratuito y de código fuente abierto. Está basado en el conocido entorno de desarrollo Eclipse, también de código abierto; pero mientras que Eclipse está focalizado en el desarrollo para Java, ZendStudio es una distribución focalizada en el desarrollo Web, con soporte a HTML, CSS y JavaScript.

Soporta varias librerías como: ExtJS, Yahoo UI y JQuery[23] para la presentación así como la inclusión de FrameWork para el desarrollo de la web pudiendo combinarlas fácilmente en una aplicación. ZendStudio está disponible como una aplicación independiente, se puede encontrar para dos plataformas fundamentales: Windows y GNU/Linux.

La gestión de proyectos, vista previa, autocompletado de código y gestión de documentación, son algunas de las características similares que presenta con otros entornos de desarrollo integrado (Eclipse, C++ Builder, Visual Studio. NET), pero además, visualiza los errores de sintaxis a medida que se escribe, mejora los Debuger[24] y ayudantes de código, y da soporte a extensiones de PHP.

---

[21] ***Open Source:*** *Código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente.*

[22] ***GNU/Linux****: Es el término empleado para referirse al sistema operativo Unix-like que utiliza como base las herramientas de sistema de GNU y el núcleo Linux.*

[23] ***jQuery:*** *Es un una biblioteca o framework de Javascript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.*

[24] ***Debuger:*** *Un depurador es un programa que permite depurar o limpiar los errores de otro programa informático.*

1.3.3.6 Spket

IDE basado en eclipse Spket IDE, es una excelente aplicación que ofrece la posibilidad de editar en lenguaje de programación JavaScript. Dentro de las numerosas características de Spket IDE, podemos destacar algunas como el autocompletado de comandos, diferenciación por colores de la sintaxis, la incorporación de librerías por ejemplo Extjs, Air, prototype, jQuery entre otras. Cuenta con un funcionamiento totalmente sencillo para todo aquel programador profesional o aficionado y posee una interfaz gráfica verdaderamente eficiente y completa para la edición de las aplicaciones.

1.3.3.7 Navegador

Mozilla Firefox 3.* es un navegador de Internet libre y de código abierto. Es usado para visualizar páginas web. Incluye navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. Además se pueden añadir funciones a través de complementos desarrollados por terceros.

Las características de Mozilla Firefox 3.* son las siguientes:

- Multiplataforma.
- Cuenta con una protección antiphishing, antimalware e integración con el antivirus.
- La navegación por pestañas.
- Bloqueador de ventanas emergentes.
- Múltiples Extensiones.
- Incluye un buscador integrado en la interfaz que hace búsquedas en Google.
- Posee gestor de descargas.
- Utiliza el sistema SSL[25] para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS[26].

1.3.3.8 SVN (SubVersion)

SVN (SubVersion) es un sistema de control de versiones, que mantiene los registros de todos los cambios que se han realizado a los archivos de un software, lo que permite el trabajo de distintos

[25] ***SSL:*** *Sistema de cifrado. Secure Sockets Layer que hace posible que sólo el servidor y el cliente entiendan un texto .(Ver Glosario)*

[26] ***HTTPS:*** *URL creada por Netscape Communications Corporation para designar documentos que llegan desde un servidor WWW seguro.(Ver Glosario)*

desarrolladores en un mismo proyecto, está herramienta es muy usada por los programadores de software libre.

Existen dos razones fundamentales para el uso de esta herramienta:

1. Gestiona las modificaciones durante el desarrollo.
2. Permite que varias personas trabajen sobre los mismos ficheros.

1.3.3.9 TortoiseSVN

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion, implementado como una extensión al shell de Windows[27]. Es software libre liberado bajo la licencia GNU GPL[28].

Entre sus características fundamentales podemos citar:

- ***Integración con el shell de Windows:*** TortoiseSVN se integra perfectamente en el shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya conoce y que no tiene que cambiar a una aplicación diferente cada vez que necesite las funciones del control de versiones.
- ***Iconos sobreimpresionados:*** El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.
- ***Fácil acceso a los comandos de Subversion:*** TortoiseSVN añade su propio submenú al explorador.
- ***Versionado de carpetas:*** Control de la historia de ficheros individuales
- ***Confirmaciones atómicas:*** Esto permite a los desarrolladores construir y confirmar cambios como unidades lógicas.
- ***Metadatos versionados:*** Cada fichero y directorio tiene un conjunto invisible de "propiedades" adjuntos. Puede inventarse y almacenar cualquier par de clave/valor que desee. Las propiedades se versionan en el tiempo, igual que el contenido de los ficheros.
- ***Elección de capas de red:*** Subversion tiene una noción abstracta del acceso al repositorio, proporciona varias características importantes gratis: autentificación, autorización, compresión de la transmisión y navegación del repositorio

[27] ***Shell de Windows:*** Es el programa que determina el aspecto del escritorio. (Ver Glosario)

[28] ***GNU GPL:*** *La Licencia Pública General (inglés: General Public License o GPL) otorga al usuario la libertad de compartir el software licenciado bajo ella, así como realizar cambios en él. Es decir, el usuario tiene derecho a usar un programa licenciado bajo GPL, modificarlo y distribuir las versiones modificadas de éste. (Ver Glosario)*

- ***Manejo de datos consistente:*** Subversion expresa las diferencias entre ficheros usando un algoritmo de diferenciación binario, que funciona exactamente igual tanto en ficheros de texto (legibles por los humanos) como en ficheros binarios (que no son legibles por nosotros). Ambos tipos de ficheros se almacenan igualmente comprimidos en el repositorio, y las diferencias se transmiten en ambas direcciones por la red.
- ***Etiquetado y creación de ramas eficiente:*** Subversion crea ramas y etiquetas simplemente copiando el proyecto, utilizando un mecanismo similar a los vínculos duros.
- ***Extensibilidad:*** Subversion no tiene lastre histórico; está implementado como una colección de librerías C compartidas con APIS bien definidas. Esto hace que Subversión sea extremadamente mantenible y se pueda utilizar por otras aplicaciones y lenguajes.

1.3.3.10 Sistema Operativo

Debian GNU/Linux es un sistema operativo libre que soporta un total de once arquitecturas de procesador e incluye los entornos KDE, Xfce y GNOME, este último es el entorno de escritorio predeterminado. Viene con más de 25113 paquetes (programas precompilados y empaquetados en un formato amigable para una instalación sencilla en su máquina).

Debian GNU/Linux 5.0 incluye más de 7700 paquetes nuevos, para un total de más de 23200 paquetes. La mayor parte de los programas que se distribuyen se han actualizado, entre ellos se puede citar la de 13400 paquetes de programas. La gestión de paquetes desde la consola es **aptitude** que ha probado tener una mejor resolución de dependencias que las funciones de gestión de paquetes **apt-get.**

1.3.4 Frameworks

“Un framework es una estructura real o conceptual que pretende servir como soporte o guía para la construcción de algo que expande su estructura en algo usable. En sistemas computarizados, un framework es con frecuencia una estructura que indica cuales tipos de programas pueden o deben ser construidos y como estos se interrelacionaran. Algunos pueden además incluir programas existentes, interfaces de programación específicas u ofrecer herramientas de programación para usar el framework. Un framework puede ser para un conjunto de funciones dentro de un sistema y como ellos se interrelacionan, las capas de un sistema operativo, de un subsistema de aplicación, como la comunicación debe ser estandarizada en algún nivel de la red de trabajo. Es generalmente más comprensivo que un protocolo y más prescriptivo que una estructura.” (16)

1.3.4.1 ExtJS Framework

"Es un framework JavaScript del lado del cliente para el desarrollo de aplicaciones Web. Tiene un sistema dual de licencia: Comercial y Open Source." (17)

ExtJS posee una librería inmensa que permite configurar las interfaces Web de manera semejante a aplicaciones desktop, incluye la mayoría de los controles de los formularios Web basándose en Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería contiene componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX.

Usar un motor de render como ExtJS permite entre otros beneficios un balance entre Cliente – Servidor, la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo; una comunicación asíncrona, en este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se de cuenta, además de que facilita eficiencia de la red, el tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

A continuación se muestra la descripción de las clases pertenecientes a este framework, además del diagrama referente a las mismas donde se representa el uso de ExtJS en la vista de la aplicación: (18)

- ***ext-base:*** Encargada del manejo de las solicitudes y respuestas, trabajo con ajax y manejo de componentes de Ext. Está incluida en el paquete original.
- ***ext-all:*** Es la encargada de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae ExtJS.
- ***Vista***: Representa la vista que se muestra al usuario.
- ***js_vista:*** Fichero js con las funciones JavaScript asociadas a la vista. Aquí se establece la referencia a las clases de Ext.

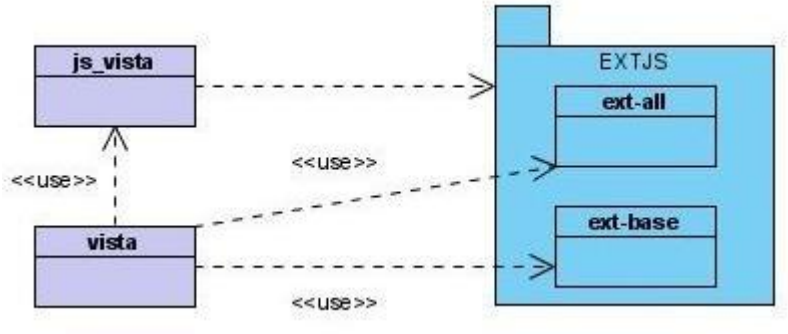

Figura 4 Diagrama de Clases Genérico para EXTJS.

1.3.4.2 Doctrine Framework

Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado, el mismo cuenta con disimiles funcionalidades, una de ellas es que da la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO), debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente.

El patrón "Active Record es una extensión del patrón *Domain Model* ("Modelo de Dominio"), que se entiende como una clase o un grupo de clases que representan a "objetos" o responsabilidades particulares en la aplicación" (19), de forma general es un enfoque al problema de acceder a los datos de una base de datos. Una fila en la tabla de la base de datos (o vista) se envuelve en una clase, de manera que se asocian filas únicas de la base de datos con objetos del lenguaje de programación usado. Cuando se crea uno de estos objetos, se añade una fila a la tabla de la base de datos. Cuando se modifican los atributos del objeto, se actualiza la fila de la base de datos.
Para una aplicación que despliega un catálogo de productos, por ejemplo, cada producto podría ser una instancia de la clase "Producto" (el catálogo mismo podría ser, a su vez, un objeto que contiene muchos objetos "Producto").

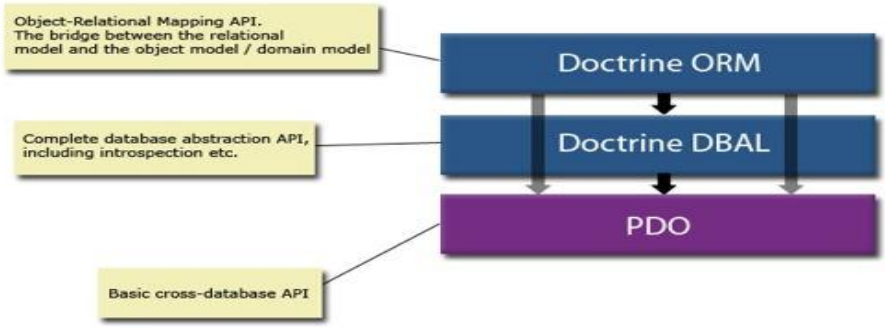

Figura 5 Doctrine ORM.

A continuación se muestra el diagrama de clases referente a este framework, las mismas son generadas automáticamente.

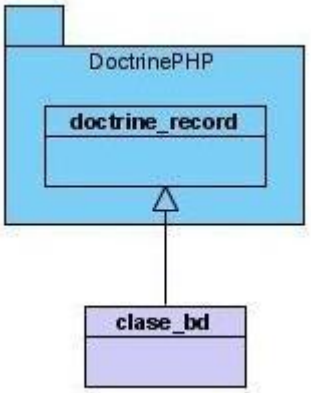

Figura 6 Diagrama de Clases Genérico para Doctrine.

1.3.4.3 Zend Framework

"Es un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5." (20)

Zend Framework utiliza el estilo MVC como base de su funcionamiento. Es fácilmente integrable a las aplicaciones debido a su composición y a que contiene diferentes clases de gran utilidad, como por ejemplo en la búsqueda dinámica de ficheros a incluir o utilizar. Cuenta con un importante mecanismo de manejo de controladores y vistas.

Dentro de sus principales características están:

- Proporciona los componentes que forma la infraestructura del patrón MVC.
- Proporciona una capa de acceso a base de datos, construida sobre PDO[29] pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.
- Proporciona capacidades de búsqueda sobre documentos y contenidos.
- Permite consumir y proveer servicios web.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar la base de datos.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.
- Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, entre otros.

Zend Framework en su implementación incluye una serie de patrones de diseño que se explicarán brevemente a continuación: (18)

- **Vista:** Implementa el patrón **Decorator** en la clase Zend_View, encargada de asignarle responsabilidades a objetos de manera dinámica y configurarlos con nuevos atributos.
- **Controlador:** Zend Framework tiene implementado el patrón **Front Controller** que implica que todas las solicitudes son dirigidas a un único script PHP que se encarga de instanciar al controlador frontal y redirigir las llamadas. Además tiene una instancia única del controlador frontal disponible mediante el patrón **Singleton** para lograr una vía de entrada única a las solicitudes.
- **Modelo:** Zend Framework provee una API para el acceso a dato conformado por un conjunto de clases que implementan los patrones **Factory Table Data Gateway** y **Row Data Gateway**.

[29] ***PDO:*** *PHP Data Objects es una extensión que provee una capa de abstracción de acceso a datos para PHP 5, con lo cual se consigue hacer uso de las mismas funciones para hacer consultas y obtener datos de distintos manejadores de bases de datos.*

A continuación se muestra la descripción de las clases pertenecientes a este framework, además del diagrama referente a las mismas: (18)

- **Zend_Controller_ Action:** De esta clase deben heredar todos los controladores de la aplicación, en ella se incluyen numerosas funcionalidades comunes.
- **vistaController:** Representa el controlador del Caso de Uso en cuestión.
- **modelController:** Es un intermediario entre el controlador y la clase del modelo. No debe heredar de Zend_Controller_Action, incluye las principales funciones para el manejo de los datos.

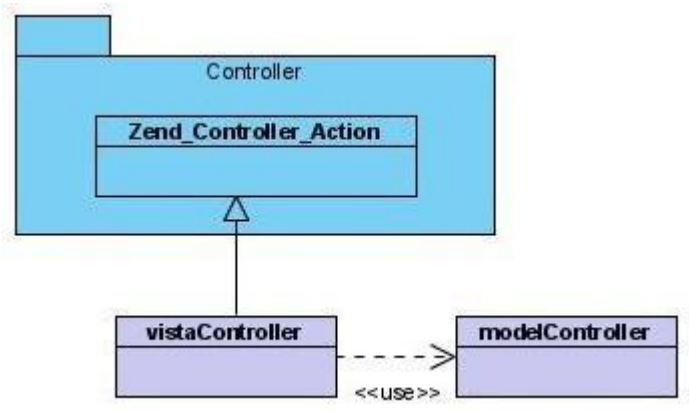

Figura 7 Diagrama de Clases Genérico para Zend Framework.

En la representación general e implementación se utilizan además otras dos clases de Zend Framework muy importantes que son Zend_Controller_Front y Zend_Loader que son el controlador frontal y la encargada de búsquedas automáticas respectivamente. Sin ellas el funcionamiento sería imposible.

1.3.4.3.1 Zend_Ext Framework

Es un framework open Source, que esta diseñado para php 5 y buenas capacidades de ampliación. Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizada por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor. Se le incluyó el IoC para la comunicación en entre los módulos o componentes, la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos, el ExtJS Framework para el desarrollo de las vistas y un controlador de trazas para controlar las acciones del sistema (acción, excepciones, rendimiento, integración, y excepción de integración).

1.3.4.4 UCID Framework

Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJS Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación y el multilenguaje.

1.3.5 Arquitectura

"Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles." (21)

Uno de los elementos bases del proceso de desarrollo de software es diseñar la Arquitectura de Software. Esencialmente sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. De la cohesión, utilidad y flexibilidad de los componentes de la arquitectura dependerán la calidad final y la utilidad del software. La correcta definición del estilo arquitectónico a utilizar, patrones y mecanismos de diseño es la raíz de lo anteriormente descrito.

1.3.5.1 Estilo Basado en Capas

Es un estilo de programación, su objetivo primordial es organizar la estructura lógica de gran escala de un sistema, en capas separadas de responsabilidades distintas y relacionadas, o sea lograr la separación de la capa de presentación, capa de negocio y la capa de datos, con una separación clara y cohesiva de intereses como que las capas "más bajas" son servicios generales de bajo nivel, y las capas más altas son más específicas de la aplicación. La colaboración y el acoplamiento se manejan desde las capas más altas hacia las más bajas.

Figura 8 Estilo basado en tres capas.

Dentro de sus principales ventajas podemos citar: (22)

- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que codificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.
- La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

1.3.5.2 Patrón Modelo-Vista-Controlador (MVC)

"El Modelo-Vista-Controlador se creó para Smalltalk[30] a finales de los setenta. A partir de entonces su uso se ha ido extendiendo cada día más para la construcción de sistemas software con interfaz gráfica." (23)

[30] ***Smalltalk**: Es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.*

MVC es un patrón de arquitectura utilizado en sistemas Web para separar los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, permitiendo flexibilidad y facilidad a la hora de hacer futuros cambios.

La **Vista** es la información presentada al usuario. Una vista puede ser una página Web o una parte de una página.

El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página, es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo.

El **Modelo** representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos.

En los años en los que se creó este patrón, los patrones como se entienden hoy en el mundo de la informática no existían, pero a lo largo de estos años, y debido a su gran aplicación en el mundo de las aplicaciones web sobre todo, se ha convertido en uno de los patrones más conocidos.

"MVC es particularmente apropiada para aplicaciones web interactivas, aplicaciones donde un usuario web interacciona con un sitio web."(23)

Figura 9 Representación del patrón MVC.

1.3.5.3 Arquitectura Cliente/Servidor

La arquitectura cliente-servidor es una nueva tendencia en el desarrollo de redes locales, que tiene como objetivo optimizar el uso tanto del hardware como del software a través de la separación de funciones: el cliente, que maneja la porción de la aplicación y el servidor, que administra los procesos de almacenamiento y recuperación de los datos.

"Puede presentarse como uno a varios clientes y uno o más servidores, junto con un sistema operativo y una plataforma de comunicación para formar un sistema cooperativo que permita la

computación distribuida, el análisis y la presentación de datos. Un único servidor típicamente sirve a una multitud de clientes, ahorrando a cada uno de ellos el problema de tener la información instalada y almacenada localmente.” (24)

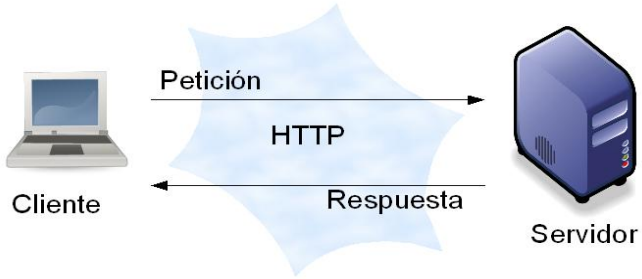

Figura 10 Arquitectura Cliente/Servidor.

Características de la arquitectura Cliente/Servidor:

- El servidor presenta una interfaz única y bien definida a todos sus clientes.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor no afectan al cliente.

1.3.6 IoC

Inversión de control (Inversion of Control en inglés, IoC) es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos (procedure calls) o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretos, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. El flujo habitual se da cuando es el código del usuario quien invoca a un procedimiento de una librería. La inversión de control sucede cuando es la librería la que invoca el código del usuario. Típicamente sucede cuando la librería es la que implementa las estructuras de alto nivel y es el código del usuario el que implementa las tareas de

bajo nivel. Su principal ventaja es que permite “reducir el acople entre una clase y las clases de las cuales depende.” (25)

1.4 Conclusiones del Capítulo 1

Después de realizar el estudio de los Sistemas Integrales de Gestión Empresarial utilizados en el mundo y en Cuba, llegamos a la conclusión de que los mismos presentan una serie de características interesantes, pero también limitaciones que impiden controlar adecuadamente todos los procesos dentro de la contabilidad y específicamente la gestión de la Contabilidad General. Por tanto se decidió elaborar un software de fácil de uso, multiplataforma, que reúna todos los requisitos propuestos por el cliente y lleve un control de todos los procesos dentro de la Contabilidad General.

En la confección de dicha aplicación se utilizará como lenguaje de programación del lado del servidor PHP 5.2, como servidor web Apache 2.0, para una mejor interacción entre el usuario y la aplicación se decidió utilizar la tecnología AJAX, como gestor de base de datos PostgreSQL 8.3, la programación por el lado del cliente XHTML, Java Script, CCS. Además la librería ExtJS 2.2 para el diseño de la interfaz auxiliándose de la herramienta ZendStudio 6.0.

Capítulo 2: Descripción y Análisis de la solución propuesta.

2.1 Introducción

En este capítulo se tratan varios aspectos ya más inmersos en el desarrollo del módulo. Se comienza realizando una profunda valoración del diseño propuesto por los analistas del sistema exponiendo las principales ventajas y deficiencias del mismo. Se pretende mostrar como está aprovechada la arquitectura y las posibilidades que proporciona el framework y las librerías utilizadas en la programación de la aplicación, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. También se describe y examina un algoritmo no trivial de los que son considerados como más importantes dentro del sistema, incluyendo el análisis de la complejidad del mismo, se explica el funcionamiento de aquellas implementaciones, componentes o módulos existentes que sean reutilizables, así como la estrategia de integración llevada a cabo, se detallan los estándares de codificación utilizados y la estructura de dato apropiada a utilizar para la implementación, por último se hace una descripción de clases y operaciones utilizadas.

2.2 Valoración crítica del diseño propuesto por el analista

A partir de la propuesta de diseño elaborada por los analistas del proyecto, se decide llevar a análisis la misma. A continuación se describe cómo resulta para los desarrolladores el trabajo a partir de dicha propuesta: La obtención del diseño propuesto, resultó de gran importancia, pues permitió una mejor comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, tecnologías de distribución y tecnologías de interfaz de usuario. Posibilitó crear una entrada apropiada y un punto de partida para las actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.

Una de las principales preocupaciones de los analistas de hoy día es cómo reutilizar modelos ya elaborados, cada analista del sistema establece sus propios mecanismos, y es el total responsable de sus modelos, siempre y cuando no viole los patrones y estilos seleccionados. Es por ello que una buena práctica del uso de los patrones, garantizarían beneficios indispensables como es el de mantener la homogeneidad en la solución propuesta. De esta manera el uso de los patrones de diseño se lleva a cabo buscando proporcionar catálogos de elementos reusables en el diseño de de la solución, evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, formalizar un vocabulario común entre los diseñadores y estandarizar el

modo en que se realiza el diseño. Es así que los patrones constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, permitiendo llevar a cabo la implementación clara y limpia del módulo bajo patrones como los GRASP.

En el diseño propuesto se utilizaron del tipo de patrones antes mencionado los cuales fueron aplicados a las clases definidas, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones y que no sea sobrecargada de métodos una clase en específico, un ejemplo pudiera ser el uso el patrón Alta Cohesión y Bajo Acoplamiento, los cuales así unidos mantienen la complejidad dentro de límites manejables, permiten además obtener clases que pueden ser reutilizadas y que no sean vulnerables a los cambios. Entre los patrones usados en la solución se encuentran:

- **Creador:** Se encarga de identificar la clase responsable de la creación de nuevos objetos. Este patrón se aplica generalmente en relaciones asociativas o de composición.
- **Alta cohesión:** Indica que la información que almacena una clase debe ser coherente, de manera que todos sus métodos tengan un comportamiento bien definido. Es utilizado para la construcción del módulo debido a que cada clase implementa las funcionalidades que le son correspondidas.
- **Bajo acoplamiento:** Su objetivo es tratar de mantener las clases lo menos ligadas entre sí, de tal forma que en caso de producirse una modificación en alguna de ellas se tenga la mínima repercusión posible en el resto, potenciando la reutilización y disminuyendo sus dependencias.
- **Experto:** Es el encargado de asignar una responsabilidad al experto en información: indica que la creación de un objeto debe recaer sobre la clase que conoce todo lo referente para su realización.
- **Controlador:** Se encarga de gestionar los eventos generados en capas anteriores ya a partir de dichos eventos tomar las decisiones apropiadas, pudiendo invocar funcionalidades contenidas en capas más profundas como el acceso a datos. Su función es de mediador o intermediario, del controlador del negocio asociados.

Con una sólida arquitectura base, un diseño flexible y escalable y el uso de las ventajas y utilidades que brindan los frameworks implementados y disponibles para el desarrollo Web es posible diseñar y posteriormente desarrollar aplicaciones con una considerable rapidez y calidad. El correcto uso de estilos, patrones y mecanismos de diseño en la generación de los artefactos necesarios garantizaron

la base necesaria para lograr calidad en la propuesta hecha por los analistas del sistema. El uso del Zend Framework, unido a las potencialidades que brinda EXT y un excelente manejo de datos con Doctrine son muestras de esto.

2.3 Implementaciones, componentes o módulos ya existentes reutilizables.

- **Arbol_nomcuenta:** Es un componente implementado con el objetivo de mostrar el árbol de cuentas contables desde cualquier interfaz del sistema que la necesite.
- **Arbol_nomgrupo:** Es un componente implementado con el objetivo de mostrar el árbol de grupos contables desde el propio gestor de contenidos, hasta la configuración de los contenidos económicos.
- **Arbol_nom_estructura:** Es un componente implementado con el objetivo de mostrar el árbol de estructuras económicas, con el nivel de jerarquía que el usuario posee del sistema para la ver el nivel de consolidación de la información desde los reportes.
- **Ver_comprobante:** Es un componente implementado con el objetivo de mostrar el comprobante de operaciones correspondiente a cualquier pase mostrado en las diferentes recuperaciones.
- **Ext_printer:** Este componente es usado para la impresión de reportes en formato HTML y PDF, usa las librerías DOMPDF para la impresión de los mismos, es usado en el componente de recuperaciones y en el comprobante de operaciones.

2.3.1 Estrategias de integración

La aplicación está definida por capas: capa de Presentación (view), Negocio (controller) y Acceso a Datos (models), esta arquitectura posibilita un trabajo más seguro, rápido y eficiente pues antes se programaba de una manera engorrosa todo detrás del botón, lo que hace una aplicación más lenta e insegura, ahora, ¿cómo trabajan estas tres capas?, la integración vertical o llamada arquitectura en 3 capas consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatros nodos de integración, el que encontramos entre la vista y el controlador, el que está entre el controlador y el modelo, el que vincula el modelo con el framework Doctrine y el que se encuentra entre el Doctrine y la base de datos. Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa.

Cada componente tiene su registro de los datos de los módulos en un fichero xml que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IoC, registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema permitiendo la comunicación entre los diferentes módulos y componentes. La base de datos es accedida de forma directa mediante controladoras y los componentes rehusados son integrados mediantes interfaces sencillas, garantizando así una total integración de las capas en el sistema.

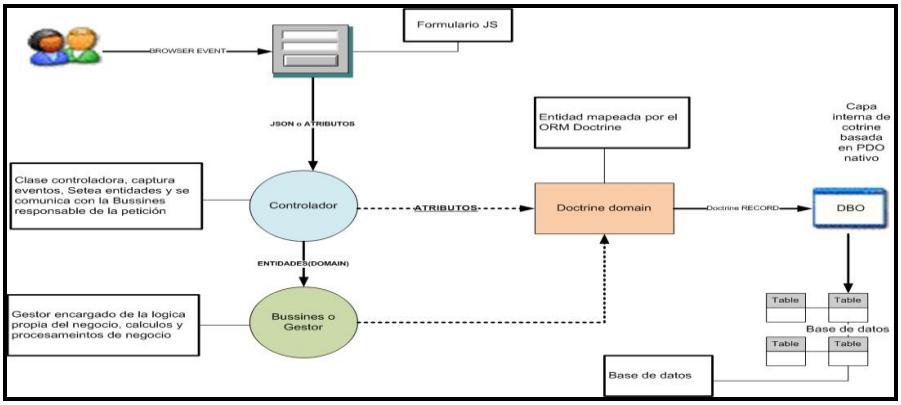

Figura 11 Colaboración entre clases en la arquitectura.

2.4 Estándares de código

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. “Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito.

Siguiendo esta idea, se definen 3 partes principales dentro de un estándar de programación:

- **Convención de nomenclatura:** Como nombrar variables, funciones, métodos, entre otros.
- **Convenciones de legibilidad de código:** Como identar el código.
- **Convenciones de documentación:** Como establecer comentarios, archivos de ayuda, entre otros.” (26)

Los estándares de codificación permiten una mejor integración entre las líneas de producción y establece pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumenta su mantenibilidad a lo largo del tiempo.

Los estándares de codificación utilizados fueron:

1. ***Notación húngara: "***Esta convención se basa en definir prefijos para cada tipo de datos y según el ámbito de las variables. También es conocida como notación REDDICK (por el nombre de su creador). La idea de esta notación es la de dar mayor información al nombre de la variable, método o función definiendo en ella un prefijo que identifique su tipo de dato y ámbito." (26) Esta notación se utilizó para los nombres de las variables.
   A continuación un ejemplo: ***intEdad:*** Según la definición se observa que esta variable es de tipo INTEGER y que representa la edad de alguna persona.

Los prefijos a utilizar en la creación de variables serán los siguientes:

Tipos de Datos	**Prefijos**
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
Float	flt
Boolean	boo

Tabla 4 Prefijos a utilizar en la creación de variables.

2. ***Notación PascalCasing: "***Es como la notación húngara pero sin prefijos. En este caso, los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula." (26) Esta notación se utilizó para los nombres de las clases.
   A continuación un ejemplo: ***GestionarUsuario:*** Este nombre de clase esta compuesto por 2 palabras, ambas iniciando con letra mayúscula.
3. ***Notación CamelCasing: "***Es parecido al Pascal-Casing con la excepción que la letra inicial del identificador no debe estar en mayúscula." (26) Esta notación se utilizó para el nombre de las funciones y el nombre de los atributos.
   A continuación ejemplos:
   insertarMoneda: Este nombre de método esta compuesto por 2 palabras, la primera todo en minúsculas y la segunda iniciando con letra mayúscula.

dineroM: Este nombre del atributo esta compuesto por 2 palabras, la primera todo en minúsculas y la segunda iniciando con letra mayúscula.

2.5 Descripción del algoritmo no trivial a implementar.

Uno de los componentes fundamentales de la Contabilidad General es el clasificador cuentas pues allí se definen las cuentas contables de una entidad, crearaperturatreeAction() es un método de este componente que surge debido a la necesidad de crear cuentas comprendidas por otro concepto, por ejemplo, crear cuentas para órganos determinados. Al tener la mayoría de estos conceptos una estructura árbolica se hacen unas series de pasos que a continuación se describen:

1. Recibe la lista de aperturas[31], la cuenta, y el carácter separador.
2. Se recorre la lista de aperturas creando el objeto Cuenta[32] y salvando en la BD[33] las aperturas que son raíz y además guardando en una lista el identificador que tenía y el que se le dio al ser insertado. Si no es raíz entonces se busca en la lista de raíces guardadas el padre de cada apertura para ser salvado. En caso de que en esa gestión el código de la cuenta a crear exista se guarda en una lista y se emite un mensaje con el código de las repetidas.

2.5.1 Análisis de complejidad del algoritmo no trivial.

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del algoritmo, para hacer dicho cálculo es necesario primero tener el código o el diseño del mismo, luego enmarcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo, a continuación se representa el código con sus instrucciones enmarcadas:

[31] ***Apertura:*** *Incorporar una cuenta de acuerdo al nomenclador según las legislación vigente.*

[32] ***Objeto Cuenta:*** *Es una abstracción de una entidad en la que se almacena todo lo referente a la cuenta.*

[33] ***BD:*** *Una base de datos es un "almacén" que nos permite guardar grandes cantidades de información de forma organizada para que luego podamos encontrar y utilizar fácilmente.*

```
public function crearaperturatreeAction()
{
 $aperturasArray = json_decode( stripslashes( $this->_request->getPost('aperturasArray'))); 1
 $cuentaDatos = json_decode( stripslashes( $this->_request->getPost('cuentaDatos'))); 1
 $separador = $this->dameFormatoPartes(0)->separador; 1
 $smlExisteApertura = array(); 1
 $arrayAprtNomExistentes = array(); 1
 $arrayCuentas = array(); 1
  foreach($aperturasArray as $objApertura){ 2
        $objCuenta = new NomCuenta(); 3
        $objRefApertura = new DatRefapertura(); 3
        $objCuentaModel = new conmCuentaModel(); 3
        $objApertRefModel = new RefaperturaModel();  3
        $objCuenta->nivel = $objApertura->data->nivel; 3
        $objCuenta->clavecuenta = $objApertura->data->codigo; 3
        $objCuenta->descripcion = $cuentaDatos->descripcionl.'  '.$objApertura->data->descripcion; 3
        $objCuenta->fechainicio = $cuentaDatos->fechainicio; 3
        if($cuentaDatos->fechafin) 4
            $objCuenta->fechafin = $cuentaDatos->fechafin;  5
        $objCuenta->idcontenidoe = $cuentaDatos->idcontenidoe; 6
        $objCuenta->idnaturaleza = $cuentaDatos->idnaturaleza; 6
        $objCuenta->idtipocuenta = $cuentaDatos->idtipocuenta; 6
        $objCuenta->idestructurae = $this->global->EstructuraEconomica->idestructurae; 6
        $cantidadX = (int)$objCuenta->nivel - (int)$cuentaDatos->nivel; 6
        if($objApertura->data->identificador == $objApertura->data->idparend){ 7
            $arrayAprtNomExistentes = array(); 8
            $objCuenta->idcuentapadre = $cuentaDatos->idcuenta; 8
            $objCuenta->concatcta = $cuentaDatos->concatcta.$separador.$objApertura->data->codigo; 8
    if($cantidadX > 1){  9
     $concacX = conmCuentaModel::llenarNivelXCuenta($separador, $cantidadX); 10
     $concatctax = $cuentaDatos->concatctax.$separador.$concacX.$separador.$objApertura->data->codigo; 10
     $objCuenta->concatctax = $concatctax; 10
            }else 11
                $objCuenta->concatctax = $cuentaDatos->concatctax.$separador.$objApertura->data->codigo; 11
        }else{ 12
            foreach($arrayAprtNomExistentes as $obj){ 13
                if($obj->indetificador == $objApertura->data->idparend){ 14
                    $objCuenta->idcuentapadre = $obj->idcuenta; 15
                    $objCuenta->concatcta = $obj->concatcta.$separador.$objApertura->data->codigo; 15
                    $objCuenta->concatctax = $obj->concatctax.$separador.$objApertura->data->codigo; 15
```

```
            break; 15
            } 15
        } 16
} 17
if(NomCuenta::existeApertura($objCuenta->concatcta, $objCuenta->idestructurae)){ 17
   $objCuentaE = new NomCuenta(); 18
   $objCuentaE = NomCuenta::obtenerCuentasPorConcat($objCuenta->idestructurae, $objCuenta->concatcta); 18
   if(!$this->tieneOperaciones($objCuentaE->idcuenta)){ 19
    $objc = new stdClass(); 20
    $objc->idcuenta = $objCuentaE->idcuenta; 20
    $objc->concatcta = $objCuentaE->concatcta; 20
    $objc->concatctax = $objCuentaE->concatctax; 20
    $objc->indetificador = $objApertura->data->identificador; 20
    $objc->idparend = $objApertura->data->idparend; 20
    conmCuentaModel::adicionaElementoSiNoExist(&$arrayAprtNomExistentes, $objc); 20
   }else{ 21
        $concat = $objCuentaE->concatcta; 21
        echo("{'codMsg':-1, 'concat': '$concat'}"); 21
        return; 21
   } 21
}elseif(!NomCuenta::existeConcat($objCuenta->concatcta, $objCuenta->idestructurae)){ 22
    $objCuentaModel->insertarCuenta($objCuenta); 23
    $objc = new stdClass(); 23
    $objc->idcuenta = $objCuenta->idcuenta; 23
    $objc->concatcta = $objCuenta->concatcta; 23
    $objc->concatctax = $objCuenta->concatctax; 23
    $objc->indetificador = $objApertura->data->identificador; 23
    $objc->idparend = $objApertura->data->idparend; 23
    conmCuentaModel::adicionaElementoSiNoExist(&$arrayAprtNomExistentes, $objc); 23
    $objRefApertura->idcuenta = $objCuenta->idcuenta; 23
    $objRefApertura->codigo = $objApertura->data->identificador; 23
    $objRefApertura->idconfapertura = $objApertura->data->idconfapertura; 23
    $objRefApertura->idestructurae = $this->global->EstructuraEconomica->idestructurae; 23
    $objApertRefModel->addRefApert($objRefApertura); 23
    if($this->_request->getPost('buscando')){ 24
        $objc = new stdClass(); 25
        $objc->nivel = $objCuenta->nivel; 25
        $objc->concatcta = $objCuenta->concatcta; 25
        $objc->concatctax = $objCuenta->concatctax; 25
        $objc->clavecuenta  = $objCuenta->clavecuenta; 25
```

```
                $objc->descripcion = $objCuenta->descripcion; 25
                $objc->idcuentapadre = $objCuenta->idcuentapadre; 25
                $objc->idcuenta = $objCuenta->idcuenta; 25
                $objc->idcontenidoe = $objCuenta->idcontenidoe; 25
                $objc->idnaturaleza = $objCuenta->idnaturaleza; 25
                $objc->idestructurae = $objCuenta->idestructurae; 25
                $objc->idtipocuenta = $objCuenta->idtipocuenta; 25
                $objc->fechainicio = $objCuenta->fechainicio; 25
                if($cuentaDatos->fechafin) 26
                    $objc->fechafin = $cuentaDatos->fechafin; 27
                else{ 28
                    $objc->fechafin = ''; 28
                $objc->descripnaturaleza = $this->dameDescripcionObj(0, $objCuenta->idnaturaleza); 28
                $objc->descripcontenidoe = $this->dameDescripcionObj(1, $objCuenta->idcontenidoe); 28
                array_push($arrayCuentas, $objc); 28
                unset($objc); 28
                } 28
            } 28
        }else{ 29
            $smlExisteApertura[] = $objCuenta->concatcta; 29
            break; 29
        } 29
     } 30
    if(count($smlExisteApertura) > 0){ 31
        $smlExisteApertura = json_encode($smlExisteApertura); 32
        echo("{'codMsg':-2, 'smlExisteApertura':$smlExisteApertura)"); 32
    }else{ 33
        $d = json_encode($arrayCuentas); 33
        echo ("{'codMsg':1, 'arrayCuentas':$d)"); 33
    } 33
} 34
```

Figura 12 Representación del algoritmo no trivial (crearaperturatreeAction()).

Después de este paso, es necesario representar el grafo de flujo asociado, en el cual se representan distintos componentes como es el caso de:

Nodo: Son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: Son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

Regiones: Son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

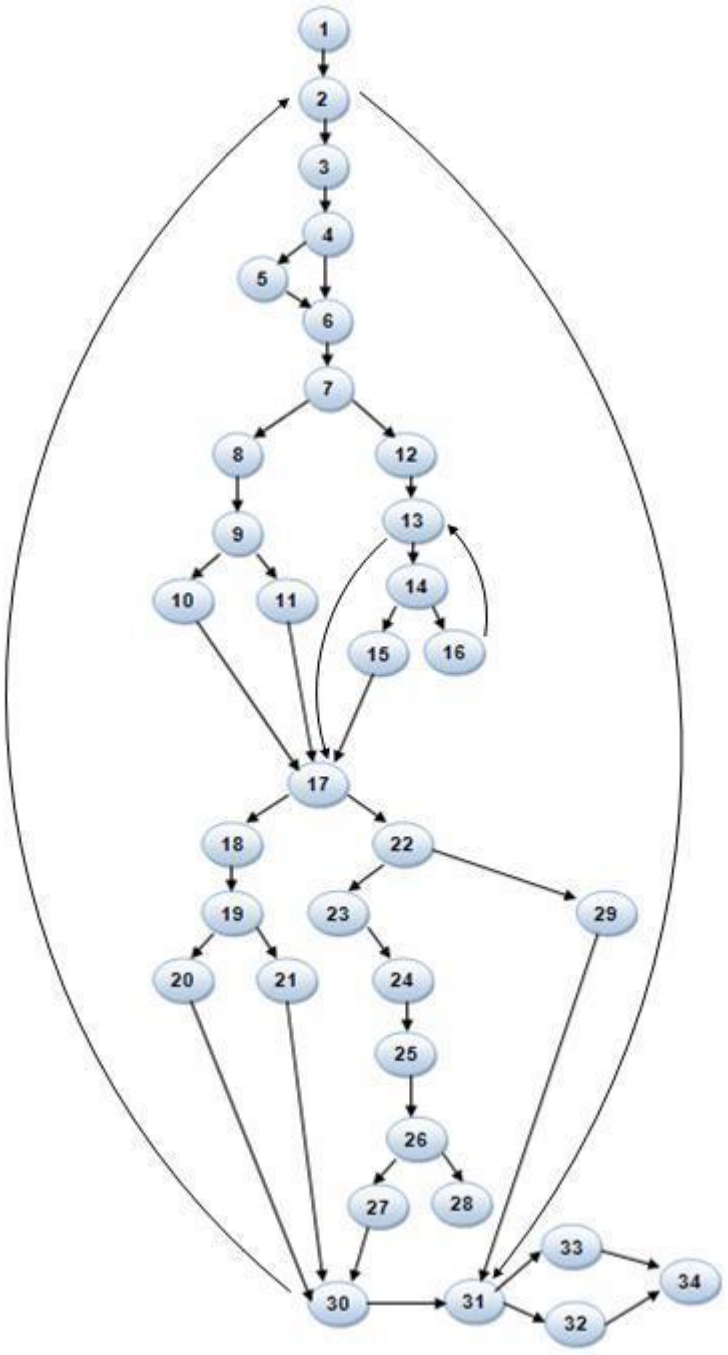

Figura 13 Grafo de flujo asociado al algoritmo no trivial (crearaperturatreeAction()).

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías o fórmulas que para concluir que el cálculo fue correcto es necesario que por las tres vías el resultado sea el mismo, las fórmulas para calcular son las siguientes:

1. **V (G) = (A – N) + 2**

Siendo "**A**" la cantidad total de aristas y "**N**" la cantidad total de nodos.

V (G) = (43 – 34) + 2

V (G) = 11

2. **V (G) = P + 1**

Siendo "**P**" la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 10 +1

V (G) =11

3. **V (G) = R**

Siendo "**R**" la cantidad total de regiones, para cada formula "V (G)" representa el valor del calculo.

V (G) = 11

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 11 que da la visión de que existen a lo sumo once caminos lógicos por donde recorrer el algoritmo.

2.6 Estructuras de datos apropiadas para la implementación de los algoritmos.

En programación, una estructura de datos es una forma de organizar un conjunto de datos elementales con el objetivo de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema. Cada estructura ofrece ventajas y desventajas en relación a la simplicidad y eficiencia para la realización de cada operación. De esta forma, la elección de la estructura de datos apropiada para cada problema depende de factores como la frecuencia y el orden en que se realiza cada operación sobre los datos.

En la realización de un script[34] en PHP en múltiples ocasiones existen variables que tienen información similar y se procesan de forma semejante. Para ello PHP (y otros lenguajes) poseen un elemento denominado array. Un array es un conjunto de variables agrupadas bajo un único nombre. Cada variable dentro de la matriz se denomina elemento. Dentro de la misma matriz pueden existir variables de diferentes tipos y no es necesario que sean todas del mismo tipo.

Una matriz en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves. Este tipo es optimizado en varias formas, de modo que puede usarlo como una matriz real, o una lista (vector), tabla asociativa (caso particular de implementación de un mapa), diccionario, colección, pila, cola y probablemente más. Ya que puede tener otra matriz PHP como valor, es realmente fácil para simular árboles.

2.7 Descripción de las nuevas clases u operaciones necesarias.

El módulo Contabilidad General es el rector del Sistema Integral de Gestión Cedrux, ya que el contenido del resto de los módulos, está subordinado a las necesidades y requerimientos del registro contable, que se resumen en la Contabilidad General.

Para definir ese registro contable citado anteriormente es necesario definir en Contabilidad General la estructura contable de la organización por lo que se divide en una serie de componentes, los cuales a su vez en su interior contienen una serie de procesos integrados.

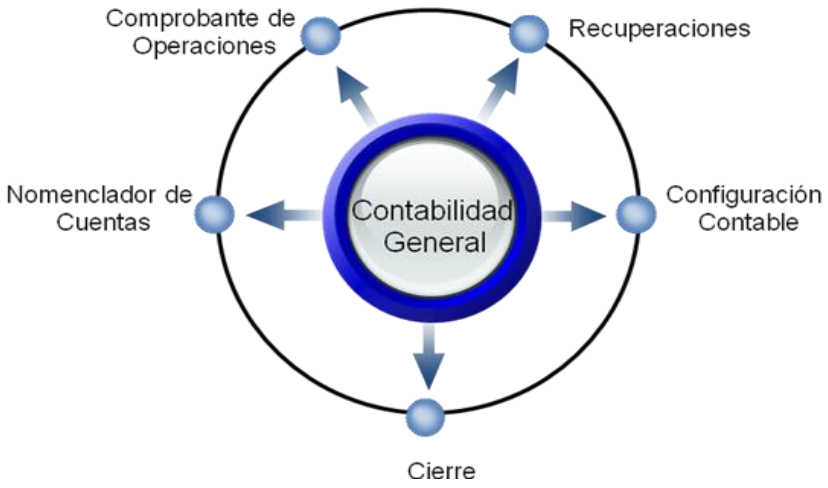

Figura 14 Componentes del proceso Contabilidad General.

[34] ***Script:*** *Guión o conjunto de instrucciones que manejan la lógica de programación.*

A continuación se muestra la descripción de cada una de las clases pertenecientes a cada uno de los procesos de cada componente:

2.7.1 Clases modelo del Componente: Configuración.

Nombre: NomGrupoModel.	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
obtenerArbol	Responsabilidad que se encarga de construir el árbol de grupos contables dado el código y la descripción de un grupo.
construirArbol	Responsabilidad que se encarga de construir el árbol de grupos contables.
buscaPadreEnArbol	Responsabilidad que se encarga de buscar el grupo padre.
buscaPadreEnArray	Responsabilidad que se encarga de buscar el grupo padre dentro de un arreglo de grupos.
compruebo ExistenciaGrupo	Responsabilidad que se encarga de verificar la existencia de un grupo contable.
registrargrupo eliminargrupo buscahijoarreglobd convierteadicionahijo buscaraizarreglobd convierteadicionahijo adicionararreglo salvanodorraiz salvanodohijos	Responsabilidades que se encargan de la gestión de un grupo contable.

Tabla 5 Descripción de la clase modelo NomGrupoModel.

Nombre: DatClasificadorcuentasConfModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD[35].

Tabla 6 Descripción de la clase modelo DatClasificadorcuentasConfModel.

Nombre: DatEctructuraEModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 7 Descripción de la clase modelo DatEctructuraEModel.

[35] ***CRUD:*** *Acrónimo de Crear, Obtener, Actualizar y Borrar (Create, Read, Update y Delete en inglés).*

Nombre: NomContenidoeModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 8 Descripción de la clase modelo NomContenidoeModel.

Nombre: NomEstadofModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 9 Descripción de la clase modelo NomEstadofModel.

2.7.2 Clases domain del Componente: Configuración.

Nombre: ConfCuentaredondeo.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
cargarCtaRedondeo	Responsabilidad que se encarga de cargar las cuentas para redondeo.
ObtenerCtaRedondeo	Responsabilidad que se encarga de obtener las cuentas de redondeo para una entidad dada.

Tabla 10 Descripción de la clase domain ConfCuentaredondeo.

Nombre: ConfCuentaredondeo.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
mostrarestadof	Responsabilidad que se encarga de cargar los estados financieros
mostrarconcepto	Responsabilidad que se encarga de obtener los conceptos de un estado financiero.

Tabla 11 Descripción de la clase domain ConfCuentaredondeo.

Nombre: DatEstructurae.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
Existeest	Responsabilidad que se encarga de buscar la existencia de alguna relación de estructura económica con una estructura común.
cargarnom	Responsabilidad que se encarga de obtener el nomenclador de estructura económica.
getEstructuraE	Responsabilidad que se encarga de obtener la estructura económica correspondiente a una estructura común.
getEstructura	Responsabilidad que se encarga de obtener una estructura económica.
ObtenerPadreRector	Responsabilidad que se encarga de obtener el padre rector de una estructura económica
ObtenerSubArbol	Responsabilidad que se encarga de obtener un subárbol de estructura económica dado una estructura.

Tabla 12 Descripción de la clase domain DatEstructurae.

Nombre: DatGrupoconcepto.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
obteneridgrupoconcepto	Responsabilidad que se encarga de obtener el grupo de concepto dado un nombre del mismo.
buscargrupo	Responsabilidad que se encarga de obtener los grupos de conceptos.
Mostrarestadof buscavalores	Responsabilidad que se encarga de obtener los datos necesarios para mostrar un estado financiero.

Tabla 13 Descripción de la clase domain DatGrupoconcepto.

Nombre: DatGrupoconcepto.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
Cargarnomcont cantidadFilasTotal getArrayContenido	Responsabilidades que se encargan de cargar el nomenclador de contenido económico para la realización del CRUD.
Buscavalores buscarContenido	Responsabilidades que se encargan de realizar la búsqueda de los contenidos económicos por varios criterios.
cargarcontenidodadoid	Responsabilidad que se encarga de obtener un contenido económico dado un id del mismo.

Tabla 14 Descripción de la clase domain DatGrupoconcepto.

Nombre: NomGrupo.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
obtenerFormatoTabla	Responsabilidad que se encarga de cargar el nomenclador de contenido económico para la realización del CRUD
cargargrupos cargargrupostodos	Responsabilidades que se encargan de realizar la búsqueda de los contenidos económicos por varios criterios
buscarDenominacion	Responsabilidad que se encarga de hacer la búsqueda de los grupos.
obtenerPadre	Responsabilidad que se encarga de obtener el grupo padre.
verificaExistenciaDatos	Responsabilidad que se encarga de verificar la existencia de datos como el concat y la descripción del grupo.

Tabla 15 Descripción de la clase domain NomGrupo.

2.7.3 Clases modelo del Componente: Nomenclador de Cuentas.

Nombre: conmCuentaModel.	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar Cuenta modificarCuenta eliminarCuenta modificarTipoCuentaDesdeCosto	Responsabilidades que se encargan de ejercer el CRUD.
llenarNivelXCuenta	Responsabilidad que se encarga de llenar con X los niveles vacíos de las cuentas.
buscaElemento	Responsabilidad que se encarga de buscar un elemento
adicionaElementoSiNoExist	Responsabilidad que se encarga de adicionar un elemento si no existe en una relación dada.
obtenerArbolApertura	Responsabilidad que se encarga de obtener el árbol de las cuentas ya aperturadas.
construirArbolApertura	Responsabilidad que se encarga de construir el árbol de cuentas sin obtener los fueras de virgencia.
buscaPadreEnArbolApertura	Responsabilidad que se encarga de obtener el padre supremo dentro de la jerarquía de cuentas.
salvaNodoRaiz salvaNodoHijos	Responsabilidad que se encarga de adicionar una rama de un árbol.
obtenerAperturasInmediatas	Responsabilidad que se encarga de obtener las aperturas inmediatas dado una cuenta padre.
obtenerAperturasSuperiores	Responsabilidad que se encarga de obtener las aperturas superiores dado una cuenta hoja.

Tabla 16 Descripción de la clase modelo conmCuentaModel.

Nombre: RefaperturaModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 17 Descripción de la clase modelo RefaperturaModel.

Nombre: ConfaperturaModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 18 Descripción de la clase modelo ConfaperturaModel.

Nombre: AsigcuentasubsistemaModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 19 Descripción de la clase modelo AsigcuentasubsistemaModel.

Nombre: DatClasificadorcuentasConfModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 20 Descripción de la clase modelo DatClasificadorcuentasConfModel.

Nombre: DatConfcuentaentidadeModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 21 Descripción de la clase modelo DatConfcuentaentidadeModel.

Nombre: RefaperturaModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 22 Descripción de la clase modelo RefaperturaModel.

2.7.4 Clases domain del Componente: Nomenclador de Cuentas.

Nombre: Confapertura.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
CargarNomApertura	Responsabilidad que se encarga de cargar el nomenclador de configuración de aperturas para la realización del CRUD.
CargarNomAperturaParaCuenta	Responsabilidad que se encarga de cargar las configuraciones de aperturas.
BuscaValorDenominacion Buscar BuscaValorIdentificador	Responsabilidades que se encargan de hacer las búsquedas de las referencias.
PoderAdicionar PoderAdicionarr	Responsabilidades que se encarga de verificar si es posible adicionar una nueva estructura.
CantidadRecords	Responsabilidad que se encarga de obtener la cantidad de cuentas para una estructura económica dada.

Tabla 23 Descripción de la clase domain Confapertura.

Nombre: DatConfcuentaentidade.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
estaArbolPadreDefinido	Responsabilidad que se encarga de verificar si hay definido cuentas del padre para una entidad hija.
obtenerArbolPadre obtenerArbolPadreDefinido	Responsabilidades que se encargan de obtener el árbol definido por la entidad padre para la entidad hija.

Tabla 24 Descripción de la clase domain DatConfcuentaentidade.

Nombre: DatRefapertura.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
BuscarValor	Responsabilidad que se encarga de buscar los valores definidos para la apertura de una entidad.
EstaIdCuenta	Responsabilidad que se encarga de obtener si para una cuenta hay configurada apertura.

Tabla 25 Descripción de la clase domain DatRefapertura.

Nombre: NomCuenta.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
buscarDenominacion	Responsabilidad que se encarga de buscar las cuentas por denominación.
obtenerAperturas	Responsabilidad que se encarga de obtener las aperturas de una cuenta.
obtenerAperturasInmediatas	Responsabilidad que se encarga de obtener las aperturas inmediatas.
obtenerRefAperturas	Responsabilidad que se encarga de obtener la referencia de apertura de una cuenta.
obtenerAperturaTreeHoja obtenerRefApertTreeHoja cambiarRefApertTreeHoja esRefAperturaTreeNoHoja	Responsabilidades que se encargan del control del CRUD con referencia de tipo arbolea.
obtenerPadre	Responsabilidad que se encarga de obtener el padre de una cuenta.
obtenerCuentasDadoEstructura obtenerCuentasDadoEstructuraChecked obtenerArrayDeCuentas obtenerCuentaPorId obtenerCuentasPorId obtenerCuentasPadres obtenerCuentasPadresAperturadas obtenerCuentasHijas obtenerCuentasHojas obtenerCuentasHojasP obtenerCuentasHojasSL	Responsabilidades que se encargan de devolver las cuentas por diferentes criterios.

obtenerHojas obtenerCuentasPorConcat obtenerCuentaPorNom obtenerArrayPadresAperturar ObtenerCuentaFinanciera	
soyHijo	Responsabilidad que se encarga de verificar si una cuenta es hija de otra dada.
cargarCuentasSinTipo cargarCuentaTipo	Responsabilidades que se encargan de cargar las cuentas para la asignación de tipos.
esPadre	Responsabilidad que se encarga de verificar si una cuenta es padre o no.
existeCuenta	Responsabilidad que se encarga de verificar si una cuenta existe.
esControlada	Responsabilidad que se encarga de saber si una cuenta es controlada o no.
obtenerContenidoCuenta	Responsabilidad que se encarga de obtener el contenido económico correspondiente a una cuenta contable.
obtenerCuentaCliente esCuentaCliente	Responsabilidades que se encargan del control de las cuentas y los clientes
existeEstructuraEconomica	Responsabilidad que se encarga de saber si existen cuentas para una estructura dada.
existeApertura	Responsabilidad que se encarga de verificar si una cuenta ha sido aperturada por otro nomenclador.
existeConcat	Responsabilidad que se encarga de verificar la existencia del un concat dado.
estaCuentaEnEntidadHija	Responsabilidad que se encarga de verificar si existe una cuenta configurada para una entidad hija.

Tabla 26 Descripción de la clase domain NomCuenta.

2.7.5 Clases modelo del Componente: Comprobante de Operaciones.

Nombre: DatComprobanteModel.	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertarComprobante modificarComprobante eliminaComprobante	Responsabilidades que se encargan de la realización del CRUD.
GuardarComprobante	Responsabilidad que se encarga de guardar un comprobante de operaciones desde otro subsistema (comprobante tipo).

Tabla 27 Descripción de la clase modelo DatComprobanteModel.

Nombre: DatAsientoModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 28 Descripción de la clase modelo DatAsientoModel.

Nombre: DatPaseModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 29 Descripción de la clase modelo DatPaseModel.

Nombre: DatRegistroanexoModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 30 Descripción de la clase modelo DatRegistroanexoModel.

Nombre: ResCentrocuentaElementoPeriodoModel	
Tipo de clase: modelo.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertar(& $instance) modificar (& $instance) elimina (& $instance)	Responsabilidades que se encargan de la realización del CRUD.

Tabla 31 Descripción de la clase modelo ResCentrocuentaElementoPeriodoModel.

2.7.6 Clases domain del Componente: Comprobante de Operaciones.

Nombre: DatAsiento.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
BuscaId	Responsabilidad que se encarga de buscar un asiento por su id.
ExisteCodigo	Responsabilidad que se encarga de verificar si existe un código de documento.
cargarGridAsiento cantidadFilas	Responsabilidades que se encargan de cargar los asientos para el desglose débito-crédito.

Tabla 32 Descripción de la clase domain DatAsiento.

Nombre: ResumenPaseterminado.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
sumacuenta	Responsabilidad que se encarga de obtener la suma del débito y el crédito de una cuenta dada para la realización de su cierre contable.

Tabla 33 Descripción de la clase domain ResumenPaseterminado.

Nombre: ResumenPase.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
getSaldosCuentas	Responsabilidad que se encarga de obtener la suma del débito y el crédito de una cuenta dada para la realización de su cierre contable.
getSaldosCuentasEjercicio	Responsabilidad que se encarga de obtener la suma del débito y el crédito a una cuenta dada para la realización de su cierre de ejercicio contable

Tabla 34 Descripción de la clase domain ResumenPase.

Nombre: ResCentrocuentaElementoPeriodo.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
SaldoCuentaCentroElemento SaldoCuentaCentroElementoPeriodoAnterior	Responsabilidades que se encargan de obtener la suma del débito y el crédito de una cuenta perteneciente a un centro o un elemento al cual este asociado.

Tabla 35 Descripción de la clase domain ResCentrocuentaElementoPeriodo.

Nombre: DatRegistroAnexo.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
Existe	Responsabilidad que se encarga de verificar si existe un anexo a un pase dado.
Obtenertodo	Responsabilidad que se encarga de obtener todos los pases dados.

Tabla 36 Descripción de la clase domain DatRegistroAnexo.

Nombre: DatPase.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
cargarGridMC	Responsabilidad que se encarga de cargar todos los pases de un comprobante.
Dame_pase	Responsabilidad que se encarga de obtener el pase por un id.
Dame_paseContra	Responsabilidad que se encarga de obtener el pase de la cuenta de contrapartida.
Dame_paseCantPase	Responsabilidad que se encarga de obtener la cantidad de pases de un asiento dado.
Dame_Pase_Doc	Responsabilidad que se encarga de obtener los pases de un documento dado.

Tabla 37 Descripción de la clase domain DatPase.

Nombre: DatComprobante.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
genNumero	Responsabilidad que se encarga de generar el número del comprobante para una entidad dada.
cantidadFilas CargarNomCo Cargargrid Buscarcomp Buscarporperiodo buscarporejercicio	Responsabilidades que se encargan de mostrar los comprobantes en el registro de comprobantes.
verificanumero	Responsabilidad que se encarga de verificar la existencia de un número de comprobante.
estadoComprobantes	Responsabilidad que se encarga de obtener todos los estados de los comprobantes en un rango de fecha o en un período dado.
CO	Responsabilidad que se encarga de obtener los comprobantes en una fecha dada.
ObtenerCoCompletoImprimir	Responsabilidad que se encarga de obtener el comprobante completo para realizar su impresión.
ExisteDocPCA	Responsabilidad que se encarga de chequear si existe un documento dado en un comprobante sin asentar.

Tabla 38 Descripción de la clase domain DatComprobante.

2.7.7 Clases domain del Componente: Cierre.

Nombre: CierreContableModel.	
Tipo de clase: domain.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
CierreContableModel	Responsabilidad que se encarga de ejercer el cierre del período contable.
CerrarEjercicio	Responsabilidad que se encarga de ejercer el cierre de ejercicio contable.

Tabla 39 Descripción de la clase domain CierreContableModel.

2.8 Conclusiones del Capítulo 2

Con la realización de este capítulo se arribó a la conclusión de que la obtención del diseño propuesto por el analista resultó ser muy importante, pues permitió adquirir una comprensión de todo lo relacionado con los requisitos no funcionales y con las restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.

Se realiza un análisis de los procesos utilizados que cumplan la función de ser reutilizables, así como la estrategia de integración empleada, permitiendo de esta manera entender el funcionamiento del módulo. Se ofrecen estándares de codificación, por los que se rigen los programadores del proyecto, logrando de esta forma hacer más legible el programa fuente así como lograr un mejor entendimiento del mismo, queda de parte de quienes la extiendan tratar de mantener un equilibrio, permitiendo una codificación homogénea y evitar las redundancias.

Se muestra uno de los principales algoritmos utilizados en la solución así como su descripción, tratando de lograr que se entienda como fue pensado y desarrollado, además de realizar el análisis de la complejidad ciclomática del mismo. De la misma forma quedó descrito la estructura de dato que será necesaria para la implementación del módulo, y las clases más importantes definidas en el diseño; tales como las clases entidad, interfaz y controladoras.

Capítulo 3: Validación de la solución propuesta.

3.1 Introducción

En el desarrollo del software, las posibilidades de error son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, entre otras. Para resolver este problema, se incluyó a nivel metódico un nuevo proceso en la confección de los sistemas informáticos: el proceso de prueba. “Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa ya que requiere, un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80% siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.” (27) El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del software.

La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuando se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

En este capítulo se valida la solución propuesta en el capítulo anterior, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas, por lo que se presenta la descripción de clases de prueba que verifican la validez de las funcionalidades del módulo Contabilidad General, así como verificar el rendimiento interno de las mismas, además de evaluar mediante métricas el diseño propuesto por los analistas.

3.2 Pruebas de Software

Las pruebas de software es el conjunto de técnicas que permiten determinar la calidad de un producto. Se integran a las diferentes fases del ciclo de desarrollo dentro de la ingeniería de software, manifestándose mediante la ejecución de un programa o mediante técnicas experimentales con el propósito de descubrir errores. “Un buen caso de prueba es aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.” (28)

La fase de pruebas es la que añade al producto final el valor para afirmar que ya se ha alcanzado la calidad requerida. Gran porciento de los programas que se desarrollan tienen errores, y es en la fase de pruebas donde se descubren, ese es el valor que añade esta etapa, el objetivo específico de la fase de pruebas es encontrar cuantos más errores mejor. Es por ello que probar es una de las fases más importantes para que un producto salga con la calidad máxima y sin errores.

3.2.1 Objetivos

"La prueba del software es un elemento crítico para la garantía de la calidad del software. El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Además, esta etapa implica:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo." (29)

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: probar la funcionalidad de los primeros prototipos, probar la estabilidad, cobertura y rendimiento de la arquitectura, probar el producto final, entre otras. Lo que conduce al principal beneficio de la prueba: proporcionar feedback[36] mientras hay todavía tiempo y recursos para hacer algo.

3.2.2 Alcance

Se lleva a cabo la prueba y se evalúan los resultados obtenidos frente a los resultados esperados. Si se descubren datos erróneos implica que hay un error y hay que corregirlo y empieza el proceso de depuración de errores. Se basa en las estructuras de control del diseño procedimental para generar los casos de prueba que:

[36] ***Feedback:*** *Conocido como retroalimentación, es un sistema de comunicación que se refiere a la capacidad del emisor para recoger las reacciones de los receptores y de acuerdo con la actitud de estos modificar su mensaje.*

- Garanticen que se recorren por lo menos una vez todos los caminos independientes de cada módulo.
- Se ejecutan todas las decisiones lógicas en su parte verdadera y en su parte falsa.
- Se recorren todos los bucles.
- Se utilizan las estructuras de datos internas para garantizar su validez.

Se invierte tiempo y esfuerzo en los detalles de control debido a que:

- Los errores suelen estar en situaciones fuera de las normales.
- A menudo caminos que se piensa que tienen pocas posibilidades de recorrerse, son recorridos regularmente.
- Los errores tipográficos son aleatorios. Puede que no sean detectados por los procesadores de la sintaxis del lenguaje particular y estar presentes en cualquier camino lógico.

3.3 Descripción de los test de Unidad

Las pruebas de unidad es la manera de comprobar el funcionamiento correcto de determinado módulo de código y ayuda a independizar el mismo, significa esto que se pueden probar los módulos independientemente uno de otros.

“Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido.” (30)

Los “test de unidades” son orientados casi siempre a las pruebas de “caja blanca” aunque para realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no se comportan correctamente al ser introducidos en la aplicación, todas las demás pruebas no cumplen función hacerlas.

3.3.1 Prueba de Caja Blanca o Estructurales

A este tipo de técnicas se le conoce también como Técnicas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento.

El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

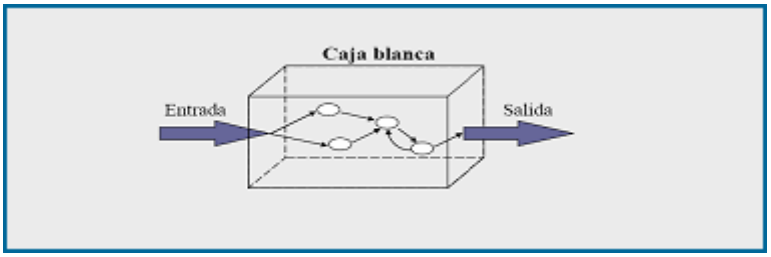

Figura 15 Representación de pruebas de Caja Blanca.

En resumen, mediante la prueba de la caja blanca se puede obtener casos de prueba que:

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Algunas técnicas de prueba de Caja Blanca son:

- ***Prueba de Condición:*** Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
- ***Prueba de Flujo de Datos:*** Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
- ***Prueba de Bucles:*** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.
- ***Prueba del Camino Básico:*** Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico.

  La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

  Los pasos que se siguen para aplicar esta técnica son:

  1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
  2. Se calcula la complejidad ciclomática del grafo.
  3. Se determina un conjunto básico de caminos independientes.
  4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

3.3.2 Pruebas de Caja Negra o Funcionales

También conocidas como Pruebas de Comportamiento o Pruebas Inducidas por los Datos, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable, se selecciona un conjunto de ellas sobre las que se realizan las pruebas. Para seleccionar el conjunto de entradas y salidas sobre las que trabajar, hay que tener en cuenta que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es pues, encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible.

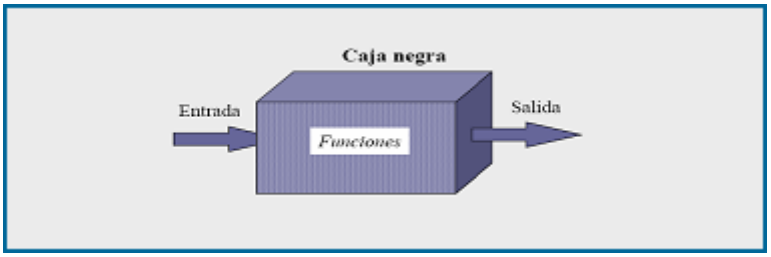

Figura 16 Representación de pruebas de Caja Negra.

Algunas técnicas utilizadas en la prueba de caja negra son:

- ***Partición de Equivalencia:*** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

  Ejemplo: Si una condición de entrada especifica un rango de valores, o sea, si un contador puede ir de 1 a 999, la clase válida sería "1 <= contador <= 999. Mientras que las clases no válidas serían "contador < 1" y "contador > 999".

- ***Análisis de Valores Límite:*** La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hayan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica nos lleva a elegir los casos de prueba que ejerciten los valores límite.

  Las pautas para desarrollar casos de prueba con esta técnica son:

  - Si una condición de entrada especifica un rango de valores, se diseñarán casos de prueba para los dos límites del rango, y otros dos casos para situaciones justo por debajo y por encima de los extremos.
  - Si una condición de entrada especifica un número de valores, se diseñan dos casos de prueba para los valores mínimo y máximo, además de otros dos casos de prueba para valores justo por encima del máximo y justo por debajo del mínimo.
  - Aplicar las reglas anteriores a los datos de salida.
  - Si la entrada o salida de un programa es un conjunto ordenado, habrá que prestar atención a los elementos primero y último del conjunto.
- ***Grafos de Causa-Efecto:*** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

3.4 Aplicación de pruebas de caja blanca

Para realizar el test es necesario realizar primeramente los procesos descritos en el capítulo 2, epígrafe 2.5.1 “Análisis de complejidad del algoritmo no trivial”, con el propósito de calcular los valores de la complejidad ciclomática del algoritmo al cual se le va a aplicar la prueba.

A continuación se enumera las sentencias de código del procedimiento antes mencionado:

```
public function ValidaFormatoAction(){
    $flag = true; 1
    $error = 'error';  $ok = 'ok'; 1
    $idfp = $this->_request->getPost ( 'idfp' ); 1
    $idfh = $this->_request->getPost ( 'idfh' ); 1
    $formatoP = $this->integrator->parametros->BuscarFormatoPorID ($idfp); 1
    $formatoH = $this->integrator->parametros->BuscarFormatoPorID ($idfh); 1
    $parteP = $this->integrator->parametros->BuscarParteFormato ($idfp); 1
    $parteH = $this->integrator->parametros->BuscarParteFormato ($idfh); 1
    if($formatoP->separador == $formatoH->separador){ 2
       if(count($parteP) <= count($parteH)){ 3
         for($i = 0; $i<count($parteP);$i++){ 4
            if(((int)$parteP[$i]->longitud) != ((int)$parteH[$i]->longitud)){ 5
               $flag = false;  break;} 6
             } 7
            if(!$flag){ 8
               $error.=' en la longitud de la parte'; echo  ("{res:'$error'}"); 9
              }else echo  ("{res:'$ok'}"); 10

   }else{$error.=' en la longitud';  echo ("{res:'$error'}");} 11
    }else{$error.=' en el separador'; echo ("{res:''$error''}");} 12
  } 13
```

Figura 17 Representación del algoritmo ValidaFormatoAction().

Seguidamente se construye el grafo de flujo asociado al código anterior:

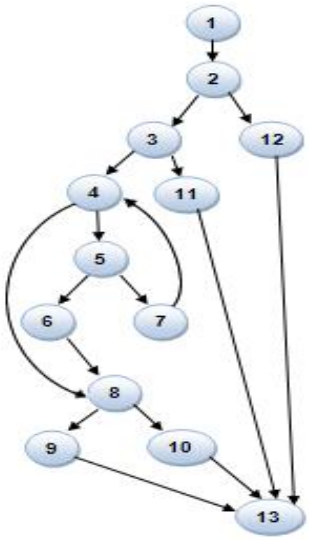

Figura 18 Grafo de flujo asociado al algoritmo ValidaFormatoAction().

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas descritas en el capítulo 2, epígrafe 2.5.1 “Análisis de complejidad del algoritmo no trivial”, las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Fórmulas para calcular complejidad ciclomática:

1. **V (G) = (A – N) + 2.**
2. **V (G) = P + 1.**
3. **V (G) = R.**

Aplicando estas fórmulas al grafo de flujo de la figura 18 se obtienen los siguientes resultados:

Calculando mediante la fórmula 1:

V (G) = (17 - 13) + 2

V (G) = 6.

Calculando mediante la fórmula 2:

V (G) = 5 + 1

V (G) = 6.

Calculando mediante la fórmula 3:

V (G) = 6.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 6, lo que significa que existen seis posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo:

Camino básico #1:

1 - 2 - 12 -13

Camino básico #2:

1 - 2 – 3 – 11 – 13

Camino básico #3:

1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 13

Camino básico #4:

1 – 2 – 3 – 4 – 5 – 6 – 8 – 10 – 13

Camino básico #5:

1 – 2 – 3 – 4 – 5 – 7 – 4 – 8 – 9 – 13

Camino básico #6:

1 – 2 – 3 – 4 – 5 – 7 – 4 – 8 – 10 – 13

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.

Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:

Descripción*:* Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución*:* Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.

Entrada: Se muestran los parámetros que entran al procedimiento

Resultados Esperados: Se expone el resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico # 1:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del formato padre, el id del formato hijo son números enteros.

Condición de ejecución: El id del formato padre será igual 900000012, el id del formato hijo será igual 900000020.

Entrada: $idfp = 900000012, $idfh = 900000020.

Resultados esperados: Se espera que los formatos sean iguales.

Los formatos difieren del separador.

Caso de prueba para el camino básico # 2:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del formato padre, el id del formato hijo son números enteros.

Condición de ejecución: El id del formato padre será igual 900000012, el id del formato hijo será igual 900000019.
Entrada: $idfp = 900000012, $idfh = 900000019.
Resultados esperados: Se espera que los formatos sean iguales.
Los formatos difieren en la cantidad de niveles, el formato de la entidad padre posee más niveles que el formato escogido para la entidad hija.

Caso de prueba para el camino básico # 3:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del formato padre, el id del formato hijo son números enteros.
Condición de ejecución: El id del formato padre será igual 900000012, el id del formato hijo será igual 900000018.
Entrada: $idfp = 900000012, $idfh = 900000018.
Resultados esperados: Se espera que los formatos sean iguales.
Los formatos difieren en la longitud de los niveles, el formato de la entidad padre difiere en la longitud de los niveles del formato escogido para la entidad hija.

Caso de prueba para el camino básico # 4:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del formato padre, el id del formato hijo son números enteros.
Condición de ejecución: El id del formato padre será igual 900000012, el id del formato hijo será igual 900000013.
Entrada: $idfp = 900000012, $idfh = 900000013.
Resultados esperados: Se espera que los formatos sean iguales.
Los formatos son válidos por los que se puede realizar la asignación a la entidad hija.

Caso de prueba para el camino básico # 5:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del formato padre, el id del formato hijo son números enteros.
Condición de ejecución: El id del formato padre será igual 900000012, el id del formato hijo será igual 900000013.
Entrada: $idfp = 900000012, $idfh = 900000013.
Resultados esperados: Se espera que los formatos sean iguales.
Los formatos difieren en la longitud de los niveles, el formato de la entidad padre difiere en la longitud de los niveles del formato escogido para la entidad hija.

Caso de prueba para el camino básico # 6:

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El id del formato padre, el id del formato hijo son números enteros.
Condición de ejecución: El id del formato padre será igual 900000012, el id del formato hijo será igual 900000013.
Entrada: $idfp = 900000012, $idfh = 900000013.
Resultados esperados: Se espera que los formatos sean iguales.
Los formatos son válidos por los que se puede realizar la asignación a la entidad hija.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función está correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.5 Aplicación de pruebas de caja negra

Para la aplicación de este tipo de prueba se usará el requisito Adicionar Estructura Económica. El objetivo de este requisito es definir la organización contable de una entidad, teniendo en cuenta el nombre de la entidad y el criterio económico, el cual puede ser: Centro Rector, Centro no Rector, Centro que define Análisis, Centro que no define Análisis y Centro Agrupador, además de la estructura física a la que pertenece y de seleccionar el formato (nomenclador de cuentas[37]) por el cual se regirá la entidad.

[37] *Nomenclador de Cuentas, proceso conocido también como Clasificador o Catálogo de Cuentas.*

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar estructura económica.	El sistema permite adicionar una nueva Estructura económica, teniendo en cuenta los criterios Descripción, Formato, Criterio económico, que son atributos para su creación.	EP 1.1: Adicionar una Estructura económica introduciendo los datos correctamente al presionar el botón **Aceptar.**	– Se presiona el botón **Adicionar.** – Se introducen los datos correctamente. – Se presiona el botón **Aceptar.** – Se muestra la notificación "Se ha insertado satisfactoriamente la Estructura económica.". – Se presiona el botón **Aceptar** de la
		EP 1.2: Adicionar una Estructura económica introduciendo datos inválidos al presionar el botón **Aceptar.**	– Se presiona el botón **Adicionar.** – Se introducen los datos correspondientes insertando algunos datos inválidos. – Se presiona el botón **Aceptar.**
		EP 1.3: Adicionar una Estructura económica dejando campos requeridos en blanco al presionar el botón **Aceptar.**	– Se presiona el botón **Adicionar**. – Se introducen los datos correspondientes en el

			formulario y deje al menos un campo requerido en blanco. – Se presiona el botón **Aceptar.**
		EP 1.4: Cancelar.	– Se presiona el botón **Adicionar**. – Se introducen o no los datos en los campos. – Se presiona el botón **Cancelar**.
		EP 1.5: Adicionar una Estructura económica introduciendo los datos correctamente al presionar el botón **Aplicar.**	– Se presiona el botón **Adicionar.** – Se introducen los datos correctamente. – Se presiona el botón **Aplicar.** – Se muestra la notificación “Se ha insertado satisfactoriamente la Entidad económica.”. – Se presiona el botón **Aceptar** de la notificación.
		EP 1.6: Adicionar una Estructura económica introduciendo datos inválidos al presionar el botón **Aplicar.**	– Se presiona el botón **Adicionar.** – Se introducen los datos correspondientes insertando algunos

			datos inválidos. – Se presiona el botón **Aplicar.**
		EP 1.7: Adicionar una Estructura económica dejando campos requeridos en blanco al presionar el botón **Aplicar.**	– Se presiona el botón **Adicionar**. – Se introducen los datos correspondientes en el formulario y deje al menos un campo requerido en blanco. – Se presiona el botón **Aplicar.**

Tabla 40 Descripción del requisito Adicionar estructura económica.

3.6 Evaluación del modelo de diseño propuesto

Son varios los puntos de vista relacionados con la calidad del software. Desde metodologías hasta las distintas normas de calidad, que pueden estar orientados tanto a los procesos de desarrollo como a los productos de software. No es objetivo de este epígrafe abundar sobre los temas de calidad, pero si desarrollar una evaluación del diseño propuesto del módulo Contabilidad General del Sistema Integral de Gestión Cedrux.

“Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes.” (31)

Atributos de calidad que se abarcan:

- ***Responsabilidad (Cohesión)****:* Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ***Complejidad del mantenimiento:*** Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- ***Complejidad de implementación****:* Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ***Reutilización****:* Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.
- ***Acoplamiento****:* Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, esta muy ligada a la característica de Reutilización.
- ***Cantidad de pruebas****:* Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas empleadas para evaluar la calidad del diseño de los componentes y su relación son las siguientes:

- ***Tamaño operacional de clase (TOC):*** Está dado por el número de métodos asignados una clase. Los atributos que afecta son la Responsabilidad, Complejidad de implementación y la Reutilización. De manera que mientras mayor sea el tamaño operacional de clase mayor será la Responsabilidad y Complejidad de implementación, mientras que su Reutilización disminuye.
- ***Relaciones entre clases (bajo) (RC):*** Esta dado por el número de relaciones de uso de una clase. Los atributos que afecta son el Acoplamiento, la Complejidad de mantenimiento, la Reutilización y la Cantidad de pruebas. De manera que mientras mayor sean las relaciones entre las clases mayor será el Acoplamiento, la Complejidad de mantenimiento y la Cantidad de pruebas, mientras que su Reutilización disminuye.

Existen otras métricas como **Profundidad de herencia** (PH), **Número de descendientes** (ND), **Número de operaciones redefinidas para una clase hija** (NOR) que se emplean también para medir la calidad del diseño pero, que son enfocadas a las herencia entre clases y en el modelo de diseño

propuesto no existe esta relación, por lo que se decidió no reflejarlas en la investigación y hacer un análisis más profundo en las demás métricas, exceptuando la métrica de **Volatilidad**, encargada de medir el grado de adaptabilidad del diseño propuesto ante los cambios de requisitos o de partes de diseño.

Resultados del instrumento de evaluación de la métrica Tamaño Operacional de clase (TOC)

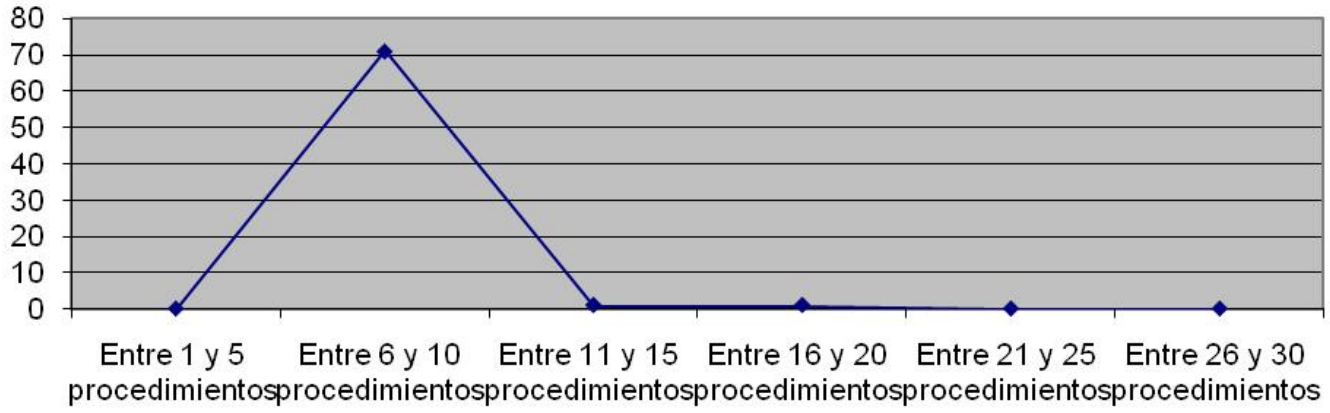

Figura 19 Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definido

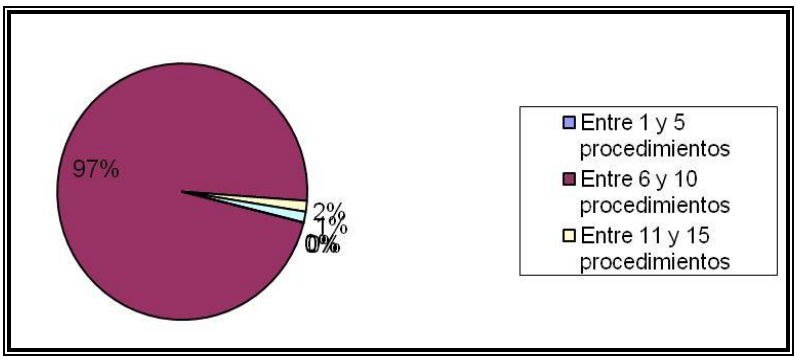

Figura 20 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

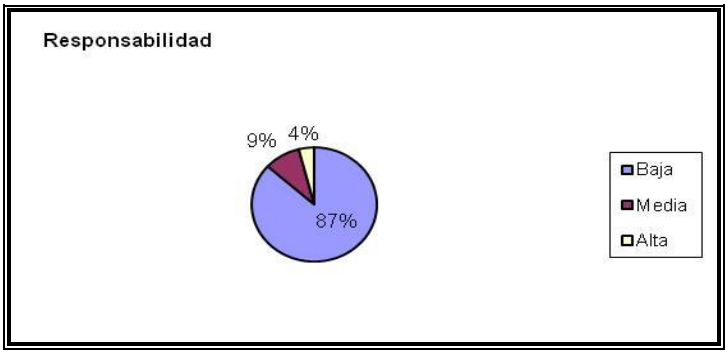

Figura 21 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación

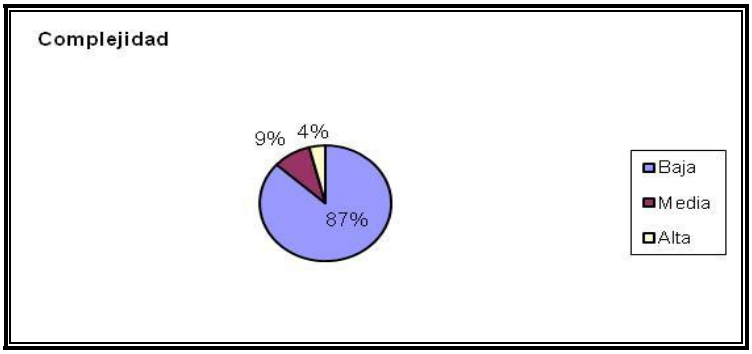

Figura 22 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

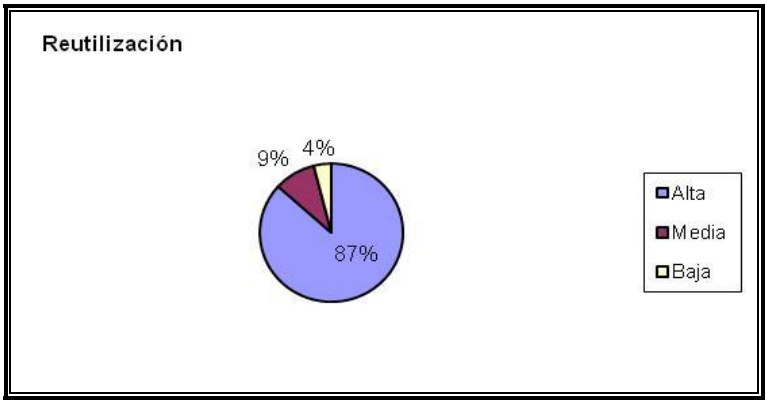

Figura 23 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño de los componentes Cierre, Configuración, Nomenclador de Cuentas, Comprobante de Operaciones y Recuperaciones tienen una calidad aceptable teniendo en cuenta que el 97 % de las clases incluidas en estos componentes posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 97% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de implementación y Reutilización).

Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)

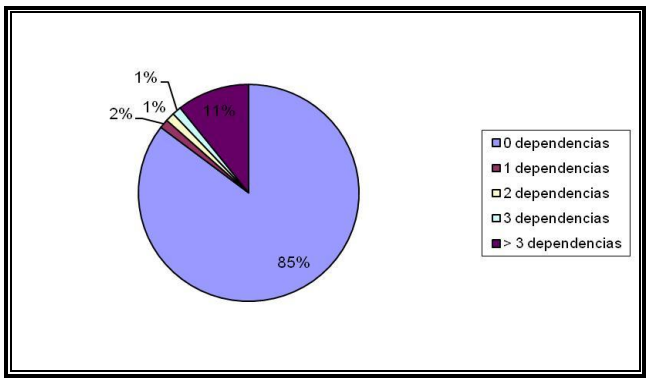

Figura 24 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

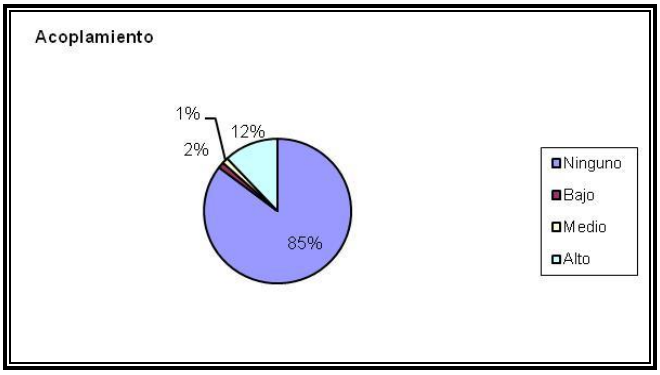

Figura 25 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento

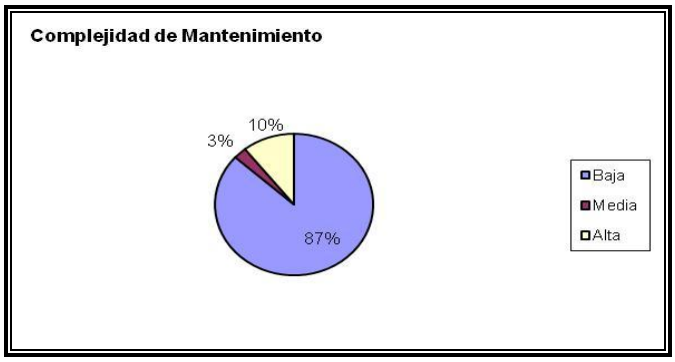

Figura 26 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas

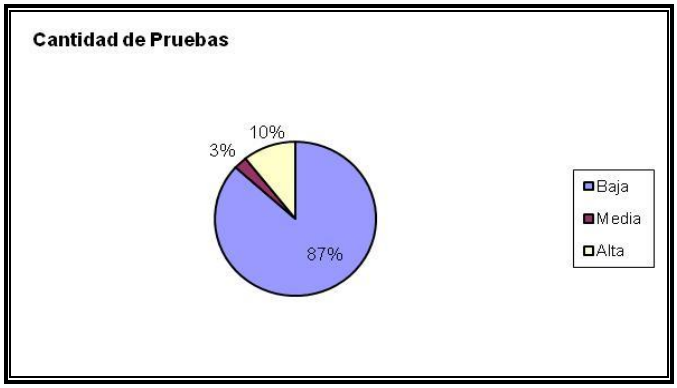

Figura 27 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización

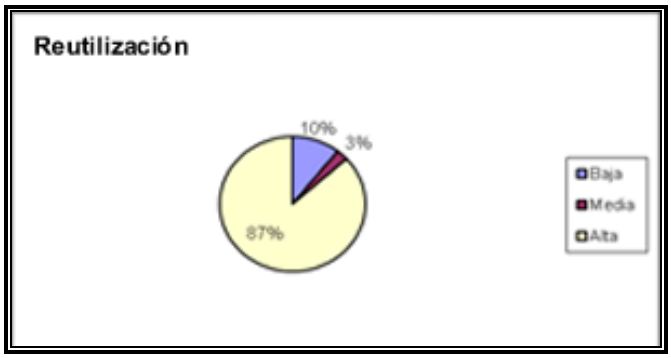

Figura 28 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño de los módulos Cierre, Configuración, Nomenclador de Cuentas, Comprobante de Operaciones y Recuperaciones tienen una calidad aceptable teniendo en cuenta que el 85 % de las clases incluidas en este módulo poseen menos de 3 dependencias de otras clases. Además el 85% de las clases no poseen acoplamiento con otras, así como los atributos de calidad Complejidad de mantenimiento, Cantidad de pruebas y Reutilización se comportan satisfactoriamente en un 87 % de las clases.

3.6.1 Resumen de los resultados

A continuación se muestra la Matriz de Cubrimiento o Matriz de Inferencia de los indicadores de calidad:

	Tamaño operacional de clase	Relaciones entre clases	Total	Promedio
Complejidad de implementación	1		1	
Reutilización	1	1	2	1
Acoplamiento		1	1	1
Complejidad de mantenimiento		1	1	1
Cantidad de pruebas		1	1	1
Cohesión	1		1	1

Tabla 41 Resumen de los resultados

En la tabla de resumen de los resultados queda expuesto como se van evaluando las diferentes métricas en el módulo, se aprecia como cada uno de los indicadores pertenecientes a las métricas evaluadas se comportaron de manera satisfactoria, o sea, se observa como el indicador Reutilización aumenta tanto en el Tamaño operacional de clase (TOC) como en las Relaciones entre clases (RC), no ocurriendo así en los indicadores de Complejidad de implementación, Cohesión, Acoplamiento, Complejidad de mantenimiento y Cantidad de pruebas, los cuales disminuyeron en dependencia de la métrica en la cual fueron evaluadas, beneficiando de esta manera que el diseño realizado se puede valorar de satisfactorio.

3.7 Conclusiones del Capítulo 3

En el desarrollo de este capítulo se comienza con el análisis de diferentes aspectos como el significado de las pruebas de software, sus objetivos y alcance. Se realiza una descripción de los test de unidad, dentro de los cuales se analizaron los tipos de prueba: Caja Blanca y Caja Negra, la ejecución de las mismas y los resultados obtenidos, además se evaluó mediante métricas el diseño propuesto por los analistas.

La ejecución de estas pruebas permitió obtener una medida de la complejidad del algoritmo escogido, y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución a través del concepto de complejidad ciclomática del grafo de flujo de control, así como interesarnos por su forma de interactuar con el medio que le rodea entendiendo, qué es lo que hace, es decir estudiarlo desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce. En cuanto a la evaluación del modelo de diseño propuesto, se pudo apreciar como cada uno de los indicadores pertenecientes a las métricas evaluadas, se comportaron de manera satisfactoria, beneficiando de esta manera que el diseño realizado se valorara de forma satisfactoria.

Conclusiones Generales

Con la finalización del trabajo realizado, se ha confirmado que es verdaderamente importante y necesaria la implementación del módulo Contabilidad General, pues este permitió la informatización de todos los procesos que se encuentran dentro de mismo, además de que se resolvió de forma eficiente uno de los problemas que más afecta a Cuba en la actualidad, los temas relacionados con la dualidad monetaria y multimoneda, así como la adaptación de la solución a las nuevas concepciones de informatización del país, de manera general se concluye con lo siguiente:

- Se estudiaron de forma satisfactoria, sistemas informáticos vinculados con la gestión de la Contabilidad General identificando ventajas y deficiencias de los mismos, permitiendo erradicar los problemas de los sistemas existentes y fusionar las mejores prácticas y funcionalidades.
- Se utilizaron para el correcto funcionamiento de este módulo las metodologías, herramientas y lenguajes acordes a las necesidades del país de migrar al software libre.
- Se realizó un estudio de los estándares de codificación elegidos por la línea de Arquitectura, encargada de seleccionar todo lo relacionado con la arquitectura del proyecto, lo cual permitió entender el por qué de su selección, siendo de gran utilidad en la implementación; ya que sirvió para aprender nuevos método y formas de tener organizado el código, para en futuros mantenimientos o iteraciones, mejorar la aplicación.
- Se le aplicaron métricas al diseño propuesto por los analistas, las cuales arrojaron un resultado positivo, lo que permitió ahorrar tiempo al equipo de desarrollo, pues se partió de un diseño satisfactorio ya existente.
- Se cumplió el objetivo de la investigación, pues se realizó la implementación del módulo Contabilidad General del Sistema Integral de Gestión Cedrux, cumpliendo con todos los requisitos establecidos.
- Se realizaron diferentes tipos de pruebas con el objetivo de comprobar la viabilidad de la solución.

El sistema resultante logró cubrir todas las funcionalidades previstas, facilitando así el trabajo y la toma de decisiones de los directivos y demás trabajadores de la entidad que utilice dicha aplicación.

Recomendaciones

Con la realización del presente trabajo se recomienda:

- Continuar tratando las incidencias provenientes del proceso de implantación en las empresas pilotos con el objetivo de perfeccionar el trabajo realizado.
- Incorporar la funcionalidad de ***Consolidación de estados financieros*** que son documentos que muestran la situación financiera y los resultados de operación de un grupo de empresas interrelacionadas por la propiedad de sus acciones, y que consideradas desde un punto de vista económico, forman todas una sola organización que opera bajo un control común, por tanto, este tipo de estados no muestra la posición financiera ni los resultados de operación de una empresa en particular, ni tampoco los de una entidad legal concreta, sino los de un grupo de empresas que integran una unidad económica.
- Incorporar la funcionalidad de ***Análisis de estados financieros o Revisión analítica*** que es analizar e interpretar los Estados Financieros mediante razones o indicadores con el objetivo de la toma de decisiones.
- Incorporar la funcionalidad de ***Recuperaciones de mayor y Submayor por saldos*** que permite obtener los saldos de una cuenta en un período determinado sin desglosar por movimiento.

Bibliografía

Alvarez, M. A. (2001, Junio 13). *desarrolloweb.com*. Retrieved Enero 10, 2009, from Qué es XML: http://www.desarrolloweb.com/articulos/449.php

aplicaciones empresariales.com. (n.d.). Retrieved Diciembre 22, 2008, from AbanQ: software ERP para la pequeña empresa: http://www.aplicacionesempresariales.com/abanq-software-erp-para-la-pequena-empresa.html

assets. (n.d.). Retrieved Diciembre 19, 2008, from SISTEMA DE GESTIÓN INTEGRAL: http://assets.co.cu/assets.asp

Booch, G. (2005). *EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* (A.-W. Professional, Ed.)

Burbujas en .NET. (n.d.). Retrieved Febrero 20, 2009, from IoC o el poder de ceder el control: http://geeks.ms/blogs/etomas/archive/2008/10/28/ioc-o-el-poder-de-ceder-el-control.aspx

Celis, I. (2005). *ESTADOBETA desarrollo web con estándares*. Retrieved Febrero 12, 2009, from Active Record: http://www.estadobeta.com/2006/05/02/active-record/

Cocchi Guerrero, H. (2007, Agosto 11). *h@nz...el Geek*. Retrieved Diciembre 17, 2008, from Introducción a JSON en JavaScript y .Net: http://hanzcocchi.net/introduccion-a-json-en-javascript-y-net/

Coello Costa, H. R. (2002, Noviembre 27). *informatizate*. Retrieved Febrero 20, 2009, from http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html

CONSOLTIC. (n.d.). Retrieved Diciembre 22, 2008, from ERP OPENBRAVO Software libre para la gestión empresarial: http://www.consoltic.com/productos/consoltic_erp_openbravo/

de San Martin, E. (2008, Enero 7). *ProgramaciónWeb.net*. Retrieved from Que es ajax: http://www.programacionweb.net/articulos/articulo/?num=317

D'Onofrio, D. L. (n.d.). *Probando software* . Retrieved Marzo 2, 2009, from http://www.elguille.info/Clipper/probando.htm

Eguíluz Pérez, J. (n.d.). *Introducción a XHTML*. Retrieved Diciembre 16, 2008, from http://www.librosweb.es/xhtml

Eguíluz Pérez, J. (n.d.). *librosweb.es*. Retrieved Diciembre 16, 2008, from Introducción a CSS: http://www.librosweb.es/css/

Eguíluz Pérez, J. (n.d.). *librosweb.es*. Retrieved Diciembre 17, 2008, from Introducción a AJAX: http://www.librosweb.es/ajax/capitulo1.html

El ERP de SAP: R/3. (n.d.). Retrieved Diciembre 22, 2008, from http://www.tuobra.unam.mx/publicadas/040702105342-El.html

El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. (n.d.). Retrieved Diciembre 24, 2008, from 50. El servidor de web Apache: Introducción práctica: Apache 1.x y 2.0 alpha. [Online] [Cihttp://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html.

Etapa de pruebas . (2007, Septiembre 16). Retrieved Febrero 22, 2009, from http://fabian-quintosemestre.blogspot.com/2007/09/etapa-de-pruebas.html

Ext JS. (n.d.). Retrieved Diciembre 19, 2008, from http://extjs.es/

Forouzan, B. A. (2003). *Introduccion a la ciencia de la Computación.* Cengage Learning Editores.

Free Download Manager. (n.d.). Retrieved Enero 11, 2009, from Paradigma visual para UML: http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

GACETA OFICIAL DE LA REPÚBLICA DE CUBA MINISTERIO DE JUSTICIA. (2008). Retrieved Diciembre 27, 2008, from http://www.gacetaoficial.cu/codedicante.php

Guía Breve de CSS. (n.d.). Retrieved Enero 23, 2009, from http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo

Guía Breve de Tecnologías XML. (1994-2005). Retrieved Enero 11, 2009, from http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML

Guía Breve de XHTML. (n.d.). Retrieved Enero 11, 2009, from http://www.w3c.es/Divulgacion/Guiasbreves/XHTML

HERRAMIENTAS CASE. (n.d.). Retrieved Enero 19, 2009, from http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm

HTML.net. (n.d.). Retrieved Diciembre 17, 2008, from Tutoriales sobre HTML y CSS - Construye tu propio sitio web: http://es.html.net/tutorials/css/lesson1.asp

ih Informática Hoy. (n.d.). Retrieved Diciembre 20, 2008, from Evolución Histórica del Software ERP: http://www.informatica-hoy.com.ar/software-erp/Evolucion-Historica-del-Software-ERP.php

INTEGRADO verSat sarasola. (n.d.). Retrieved Diciembre 19, 2009, from http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc

Introducción a JSON. (n.d.). Retrieved Enero 16, 2009, from http://www.json.org/json-es.html

JohannA. (n.d.). *Modelo N-Capas*. Retrieved Febrero 17, 2009, from http://ncapas.blogspot.com/2008/11/modelo-n-capas_13.html

kynetia. (n.d.). Retrieved Abril 12, 2009, from Tipos de Pruebas: http://www.kynetia.es/calidad/tipos-de-pruebas.html

Leopoldo Magaña, C. (2005-2008). *Zend Framework, una introducción*. Retrieved Enero 29, 2009, from http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/

maestros del web. (n.d.). Retrieved Enero 26, 2009, from ¿Qué es CSS?: http://www.maestrosdelweb.com/editorial/introcss/

Manuales,Tutoriales y Herramientas. (n.d.). Retrieved Diciembre 22, 2008, from Curso de JavaScript : http://max-alva.webs.com/javascript.htm

Martín, S. (n.d.). *BOLETIN ELECTRÓNICO.* Retrieved Marzo 22, 2009, from http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin/boletin_8_Octubre_2007.pdf

Medinilla, N. (n.d.). *Proyecto Práctico de Construcción de un Sistema Software*. Retrieved Enero 7, 2009, from Resumen: http://is.ls.fi.upm.es/udis/docencia/proyecto/docs/Confuso_Modelo_Vista_Controlador.doc.

Molina Díaz, Y. (2007). *Trabajo de diploma para optar por el título de Ingeniería en Informática.* Caracas, Venezuela.

mozilla.org. (1998–2009). Retrieved Febrero 8, 2009, from http://www.mozilla.org/about/

OpenERP Spain. (n.d.). Retrieved Diciembre 21, 2008, from http://www.openerpspain.com/.

Osmosis Latina. (n.d.). Retrieved Febrero 26, 2009, from http://www.osmosislatina.com/subversion/

Patrones de diseño. (n.d.). Retrieved Marzo 22, 2009, from http://mit.ocw.universia.net/6.170/6.170/f01/pdf/lecture-12.pdf

Pérez Hernández, Y. (2008). *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Cuba.

PgAdmin III. (n.d.). Retrieved Enero 22, 2009, from http://guia-ubuntu.org/index.php?title=PgAdmin_III

php. (2001 - 2009). Retrieved Diciembre 19, 2008, from http://php.net/

PHP POR ENCARGO. (n.d.). Retrieved Dicimbre 21, 2008, from http://www.sitgeshost.com/cms-wordpress-joomla-php-seo-dominios-hopedaje-sitges/?page_id=7

Pruebas. (n.d.). Retrieved Marzo 5, 2009, from http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php

QUERSYSTEM. (2000 - 2008). Retrieved Febrero 6, 2009, from http://www.quersystem.com/index.php?option=com_content&view=article&id=20&Itemid=3

Rodas XXI. (2002- 2009). (CITMATEL) Retrieved Diciembre 22, 2008, from Sistema Integral Económico Administrativo: http://www.rodasxxi.cu/

Saavedra López, E. (2009, Marzo 9). Vistos desde otros horizontes. *Revista de Software Libre ATIX* , 78.

slideshare. (n.d.). Retrieved Marzo 12, 2009, from http://www.slideshare.net/thomasw/phpbootcamp-zend-framework?src=related_normal&rel=184203

Spket IDE. (2005 - 2008). Retrieved Febrero 6, 2009, from http://www.spket.com

Studio Creative Weblog. (n.d.). Retrieved Enero 14, 2009, from JSON para PHP: http://mcarrillo.wordpress.com/category/herramientas-de-programacion/

TechTarget. (n.d.). *Whatis.com*. (TechTarget) Retrieved Enero 23, 2009, from The leading IT Encyclopedia and learning center: http://whatis.techtarget.com/definition/0,,sid9_gci1103696,00.html

tictech consulting. (n.d.). Retrieved Diciembre 20, 2008, from Openbravo ERP: http://www.tictech.es/index.php/openbravo.

TortoiseSVN. (n.d.). Retrieved Enero 11, 2009, from Un cliente de Subversion para Windows: https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf

UDLAP UNIVERSIDAD DE LAS AMERICAS PUEBLA. (2002). Retrieved Enero 8, 2009, from Capítulo 3 MOVILIDAD EN LA NAVEGACIÓN Y ALMACENAMIENTO EN BASES DE DATOS: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/mendez_a_ky/capitulo3.pdf

XHTML 1.0: El Lenguaje de Etiquetado Hipertextual Extensible. (n.d.). Retrieved Enero 13, 2009, from http://www.sidar.org/recur/desdi/traduc/es/xhtml/xhtml11.htm#xhtml

Xtandard.com. (n.d.). Retrieved Enero 13, 2009, from ¿Qué es XHTML?: http://www.xtandard.com/2005/05/03/%C2%BFque-es-xhtml/

YeRu$h@. (n.d.). Retrieved Enero 28, 2009, from ExtJS 2.0: http://yerusha.wordpress.com/2008/01/10/extjs-20/

Zend Technologies. (2006 - 2009). Retrieved Marzo 23, 2009, from http://framework.zend.com/

Referencias Bibliográficas

1. ih Informática Hoy. *Evolución Histórica del Software ERP.* [Online] [Cited: Diciembre 20, 2008.] http://www.informatica-hoy.com.ar/software-erp/Evolucion-Historica-del-Software-ERP.php.

2. El ERP de SAP: R/3. [Online] [Cited: Diciembre 22, 2008.] http://www.tuobra.unam.mx/publicadas/040702105342-El.html.

3. Rodas XXI. *Sistema Integral Económico Administrativo.* [Online] CITMATEL, 2002- 2009. [Cited: Diciembre 22, 2008.] http://www.rodasxxi.cu/.

4. CONSOLTIC. *ERP OPENBRAVO Software libre para la gestión empresarial.* [Online] [Cited: Diciembre 22, 2008.] http://www.consoltic.com/productos/consoltic_erp_openbravo/.

5. **Molina Díaz, Yasmany.** *Trabajo de diploma para optar por el título de Ingeniería en Informática.* Caracas, Venezuela : s.n., 2007.

6. PHP POR ENCARGO. [Online] [Cited: Dicimbre 21, 2008.] http://www.sitgeshost.com/cms-wordpress-joomla-php-seo-dominios-hopedaje-sitges/?page_id=7.

7. **Eguíluz Pérez, Javier**. librosweb.es. *Introducción a AJAX.* [Online] [Cited: Diciembre 17, 2008.] http://www.librosweb.es/ajax/capitulo1.html.

8. **Cocchi Guerrero, Hanz.** h@nz...el Geek. *Introducción a JSON en JavaScript y .Net.* [Online] Agosto 11, 2007. [Cited: Diciembre 17, 2008.] http://hanzcocchi.net/introduccion-a-json-en-javascript-y-net/.

9. **Eguíluz Pérez, Javier**. Introducción a XHTML. [Online] [Cited: Diciembre 16, 2008.] http://www.librosweb.es/xhtml.

10. **Eguíluz Pérez, Javier.** librosweb.es. *Introducción a CSS.* [Online] [Cited: Diciembre 16, 2008.] http://www.librosweb.es/css/.

11. HTML.net. *Tutoriales sobre HTML y CSS - Construye tu propio sitio web.* [Online] [Cited: Diciembre 17, 2008.] http://es.html.net/tutorials/css/lesson1.asp.

12. Manuales,Tutoriales y Herramientas. *Curso de JavaScript .* [Online] [Cited: Diciembre 22, 2008.] http://max-alva.webs.com/javascript.htm.

13. HERRAMIENTAS CASE. [Online] [Cited: Enero 19, 2009.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

14. Free Download Manager. *Paradigma visual para UML.* [Online] [Cited: Enero 11, 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/.

15. PgAdmin III. [Online] [Cited: Enero 22, 2009.] http://guia-ubuntu.org/index.php?title=PgAdmin_III.

16. **TechTarget.** Whatis.com. *The leading IT Encyclopedia and learning center.* [Online] TechTarget. [Cited: Enero 23, 2009.] http://whatis.techtarget.com/definition/0,,sid9_gci1103696,00.html.

17. YeRu$h@. *ExtJS 2.0.* [Online] [Cited: Enero 28, 2009.] http://yerusha.wordpress.com/2008/01/10/extjs-20/.

18. *Vistos desde otros horizontes.* **Saavedra López, Esteban.** Marzo 9, 2009, Revista de Sotware Libre ATIX, p. 78.

19. **Celis, Ismael.** ESTADOBETA desarrollo web con estándares. *Active Record.* [Online] 2005. [Cited: Febrero 12, 2009.] http://www.estadobeta.com/2006/05/02/active-record/.

20. **Leopoldo Magaña, Carlos.** Zend Framework, una introducción. [Online] 2005-2008. [Cited: Enero 29, 2009.] http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/.

21. **Booch, Grady.** *EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE.* [ed.] Addison-Wesley Professional. 2005.

22. **JohannA.** Modelo N-Capas. [Online] [Cited: Febrero 17, 2009.] http://n-capas.blogspot.com/2008/11/modelo-n-capas_13.html.

23. **Medinilla, Nelson.** Proyecto Práctico de Construcción de un Sistema Software. *Resumen.* [Online] [Cited: Enero 7, 2009.] http://is.ls.fi.upm.es/udis/docencia/proyecto/docs/Confuso_Modelo_Vista_Controlador.doc..

24. UDLAP UNIVERSIDAD DE LAS AMERICAS PUEBLA. *Capítulo 3 MOVILIDAD EN LA NAVEGACIÓN Y ALMACENAMIENTO EN BASES DE DATOS.* [Online] 2002. [Cited: Enero 8, 2009.] http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/mendez_a_ky/capitulo3.pdf.

25. Burbujas en .NET. *IoC o el poder de ceder el control.* [Online] [Cited: Febrero 20, 2009.] http://geeks.ms/blogs/etomas/archive/2008/10/28/ioc-o-el-poder-de-ceder-el-control.aspx.

26. **Coello Costa, Helkyn R.** informatizate. [Online] Noviembre 27, 2002. [Cited: Febrero 20, 2009.] http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html.

27. **D'Onofrio, Diego Lucio.** Probando software . [Online] [Cited: Marzo 2, 2009.] http://www.elguille.info/Clipper/probando.htm.

28. Etapa de pruebas . [Online] Septiembre 16, 2007. [Cited: Febrero 22, 2009.] http://fabian-quintosemestre.blogspot.com/2007/09/etapa-de-pruebas.html.

29. Pruebas. [Online] [Cited: Marzo 5, 2009.] http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php.

30. **Pérez Hernández, Yorji.** *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Cuba : s.n., 2008.

31. **Pressman, Roger**. 1998. Ingeniería de Software. Un enfoque práctico.

Anexos

Anexo 1. Diagrama Despliegue – Clientes Ligeros

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación, que describe la distribución, entrega e instalación de las partes que configuran el sistema físico. Muestra las relaciones físicas de los nodos que participan en la ejecución del sistema describiendo la arquitectura física del sistema en términos de: procesadores, dispositivos y componentes de software.

A continuación se muestra el diagrama de despliegue realizado de acuerdo a las necesidades del sistema.

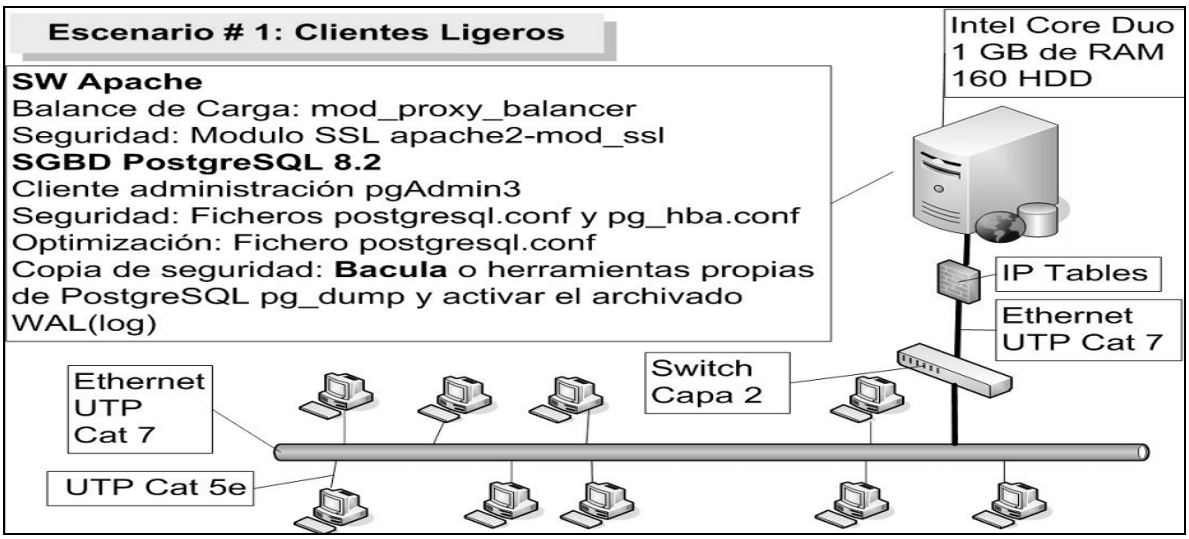

Figura 29 Despliegue - Clientes Ligeros.

Anexo 2. Extensión del Despliegue - Clientes Ligeros

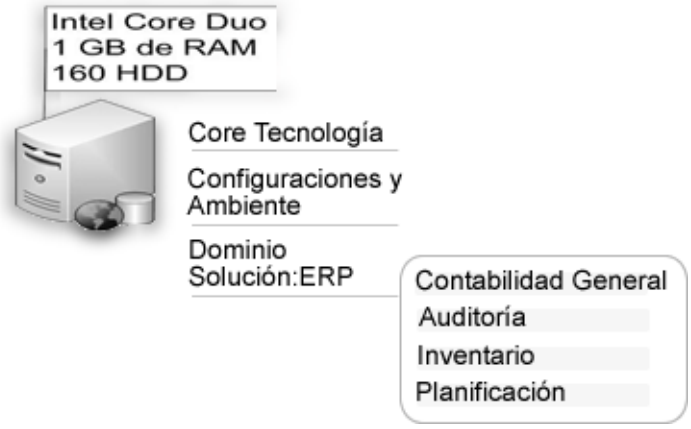

Figura 30 Extensión del Despliegue – Clientes Ligeros.

Anexo 3. Diagrama de Componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los mismos. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación. Es otra forma de representar una vista estática del sistema, que representa la organización y dependencia entre los componentes físicos que se necesitan para ejecutar la aplicación, sean estos componentes de código fuente, librerías, binarios o ejecutables.

A continuación se muestran los componentes creados en la solución propuesta, a través del diagrama de componentes:

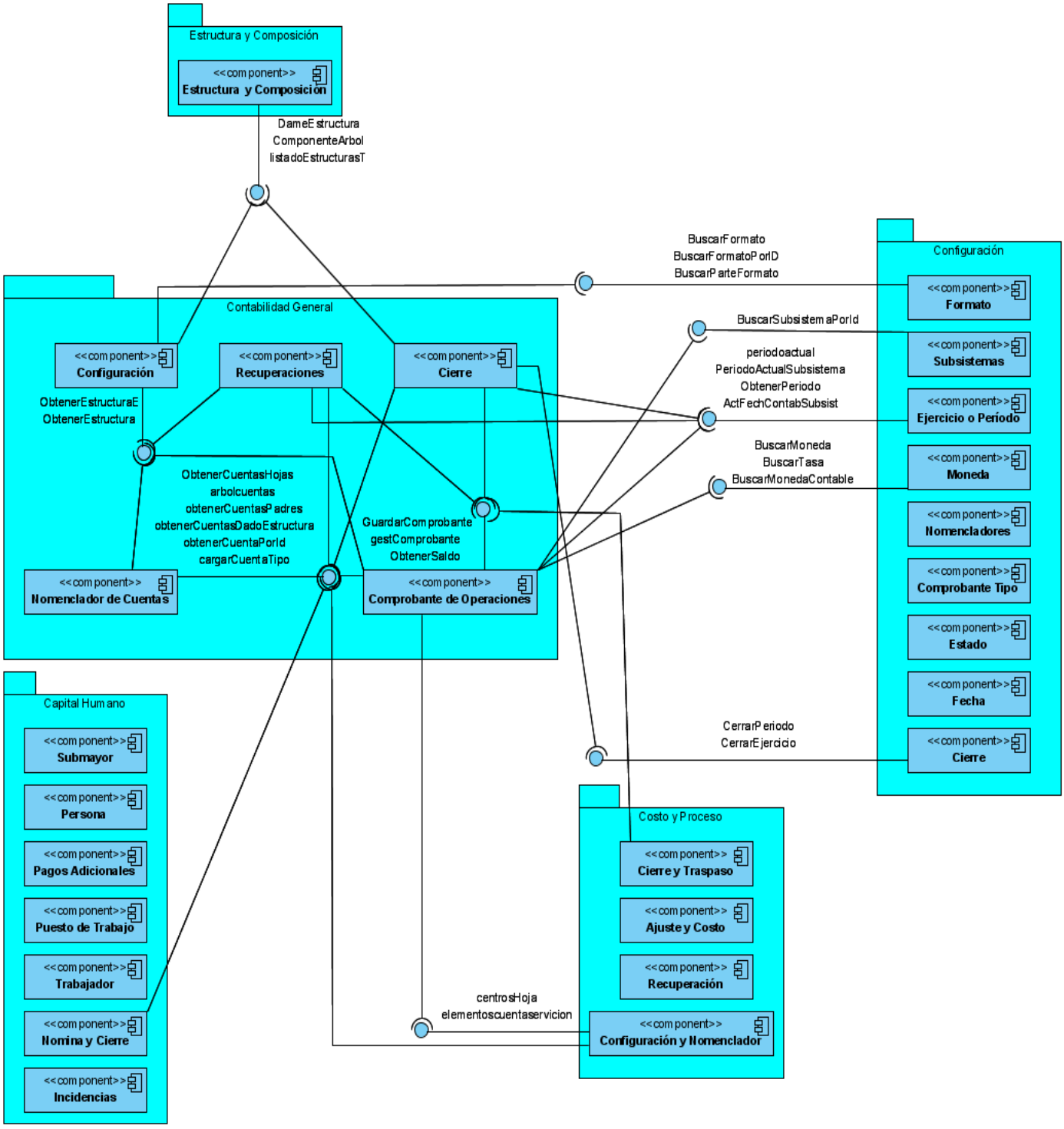

Figura 31 Diagrama de Componentes

Anexo 4. Descripción de cada una de las clases controladoras pertenecientes a cada uno de los procesos de cada componente:

Clases del componente: Configuración.

Nombre: EstructuraeconomicaController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
estructuraeconomicaAction()	Responsabilidad encargada de validar todos los datos necesarios para gestionar estructuras económicas y redireccionar a la interfaz.
cargarcriterioAction()	Responsabilidad encargada de cargar todos los criterios económicos.
cargararbolAction()	Responsabilidad encargada de cargar el árbol de estructura económica y mostrárselo a los usuarios para su uso.
cargarformatoAction()	Responsabilidad encargada de cargar los formatos para las cuentas de la estructura económica.
cargarestructuraAction()	Responsabilidad encargada de cargar la estructura física a la que se va a pertenecer la estructura económica.
adicionarAction()	Responsabilidad encargada de adicionar una estructura económica.
eliminarAction()	Responsabilidad encargada de eliminar una estructura económica.
ValidaFormatoAction()	Responsabilidad encargada de validar el formato de la estructura que se va a adicionar, para este ser válido debe ser igual al formato de la estructura padre o tener contenido al dicho formato.

Tabla 42 Descripción de la clase controladora EstructuraeconomicaController.

Nombre: ClasificadorcontenidoeconomicoController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
clasificadorcontenidoeconomicoAction ()	Responsabilidad encargada de validar todos los datos necesarios para gestionar los contenidos económicos.
cargargrupoAction()	Responsabilidad que se encarga de mostrarle al usuario los grupos contables ya existentes.
cargarnaturalezasAction()	Responsabilidad encargada de cargar la naturaleza a asignar en los contenidos económicos.
cargarformatosAction()	Responsabilidad encargada de cargar los formatos a asignar en los contenidos económicos.
cargarcontenidoseAction()	Responsabilidad que se encarga de mostrarle al usuario los contenidos económicos ya existentes.
insertarcontenidoeAction()	Responsabilidad que se encarga de adicionar un contenido económico
modificarcontenidoeAction()	Responsabilidad que se encarga de modificar y controlar el versionado de los contenidos económicos.
eliminarcontenidoeAction()	Responsabilidad que se encarga de eliminar los contenidos económicos.
importarcontAction()	Responsabilidad que se encarga de importar los contenidos económicos desde un fichero XML.
exportarAction()	Responsabilidad que se encarga de exportar los contenidos económicos a un fichero XML.

Tabla 43 Descripción de la clase controladora ClasificadorcontenidoeconomicoController.

Nombre: ClasificadorcuentasconfController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
clasificadorcuentasconfAction ()	Responsabilidad encargada de validar todos los datos necesarios para gestionar las cuentas para redondeo.
cargardatosgridgastoAction () cargardatosgridingresoAction() cargardatosgridresultAction()	Responsabilidades que se encargan de mostrar las cuenta configuradas para redondeo por gasto, ingreso y estado de resultado.
cargarcombogastoAction() cargarcomboingresoAction() cargarcamboresultadoAction()	Responsabilidades que se encargan de mostrar al usuario las cuentas de gasto, ingreso y estado de resultado para ser asignadas para redondeo.
insertarCtaAction()	Responsabilidad que se encarga de adicionar las cuentas para redondeo.
EliminarCtaAction()	Responsabilidad que se encarga de eliminar las cuentas para redondeo.

Tabla 44 Descripción de la clase controladora ClasificadorcuentasconfController.

Nombre: EstadosfinancieroController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
estadosfinancieroAction ()	Responsabilidad encargada de validar todos los datos necesarios para configurar los estados financieros y redireccionar a la interfaz.
cargarcuentasAction()	Responsabilidad que se encarga de cargar las cuentas para la configuración de los estados financieros.
adicionarestadofAction ()	Responsabilidad que se encarga de adicionar la configuración para los estados financieros.
cargarEstadosFAction ()	Responsabilidad que se encarga de mostrar los estados financieros ya configurados.

Tabla 45 Descripción de la clase controladora EstadosfinancieroController.

Nombre: ClasificadorgrupoController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
clasificadorgrupoAction()	Responsabilidad encargada de validar todos los datos necesarios para gestionar los grupos contables.
cargarformatosAction()	Responsabilidad que se encarga de obtener todos los formatos para los grupos contables.
cargargruposAction()	Responsabilidad encargada de obtener los grupos contables e importar los grupos desde un fichero.
adicionargrupoAction()	Responsabilidad encargada de adicionar los grupos contables.
modificargrupoAction()	Responsabilidad encargada de modificar los grupos contables, actualizando el control de versiones.
eliminagrupoAction()	Responsabilidad que se encarga de eliminar un grupo contable.
exportarAction()	Responsabilidad que se encarga de exportar los grupos contables para un fichero XML.

Tabla 46 Descripción de la clase controladora ClasificadorgrupoController.

Clases del componente: Recuperaciones.

Nombre: MayorController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
mayorAction ()	Responsabilidad encargada de validar todos los datos necesarios para realizar el reporte mayor.
cargarctapadreAction ()	Responsabilidad que se encarga de cargar las cuentas para la confección del reporte.
cargarmonedasAction ()	Responsabilidad que se encarga de cargar las monedas contables para la confección del reporte.
cargarperiodosAction ()	Responsabilidad que se encarga de cargar los períodos contables para la confección del reporte.
cargararbolestructuraeAction ()	Responsabilidad que se encarga de cargar las estructuras económicas para la confección del reporte según el nivel jerárquico.
calcularMayorAction()	Responsabilidad que se encarga de generar el reporte según la selección de los datos del usuario.
abrirComprobanteAction()	Responsabilidad que se encarga de mostrar el comprobante en la cual está afectada una cuenta dada.

Tabla 47 Descripción de la clase controladora MayorController.

Nombre: SubmayorgastosController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
submayorgastosAction ()	Responsabilidad encargada de validar todos los datos necesarios para realizar el reporte submayor de gasto.
cargarctaAction ()	Responsabilidad que se encarga de cargar las cuentas para la confección del reporte.
cargarmonedasAction ()	Responsabilidad que se encarga de cargar las monedas contables para la confección del reporte.
cargarperiodosAction ()	Responsabilidad que se encarga de cargar los períodos contables para la confección del reporte.
cargararbolestructuraeAction ()	Responsabilidad que se encarga de cargar las estructuras económicas para la confección del reporte según el nivel jerárquico.
cargargrupopresupAction()	Responsabilidad que se encarga de mostrar los grupos presupuestarios para la confección del reporte.
cargargastosAction()	Responsabilidad que se encarga de mostrar los elemento del gasto para la confección del reporte.
calcularsubmayorAction ()	Responsabilidad que se encarga de generar el reporte según la selección de los datos del usuario.
abrirComprobanteAction()	Responsabilidad que se encarga de mostrar el comprobante en la cual esta afectada una cuenta dada.

Tabla 48 Descripción de la clase controladora SubmayorgastosController.

Nombre: SubmayoranalisisController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
submayoranalisisAction ()	Responsabilidad encargada de validar todos los datos necesarios para realizar el reporte submayor de análisis.
cargarctapadreAction ()	Responsabilidad que se encarga de cargar las cuentas para la confección del reporte.
cargarmonedasAction ()	Responsabilidad que se encarga de cargar las monedas contables para la confección del reporte.
cargarperiodosAction ()	Responsabilidad que se encarga de cargar los periodos contables para la confección del reporte.
cargararbolestructuraeAction ()	Responsabilidad que se encarga de cargar las estructuras económicas para la confección del reporte según el nivel jerárquico.
calcularMayorAction()	Responsabilidad que se encarga de generar el reporte según la selección de los datos del usuario.
abrirComprobanteAction()	Responsabilidad que se encarga de mostrar el comprobante en la cual esta afectada la cuenta dada.

Tabla 49 Descripción de la clase controladora SubmayorgastosController.

Clases del componente: Nomenclador de Cuentas.

Nombre: ClasificadorcuentasController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
clasificadorcuentasAction ()	Responsabilidad que se encarga de validar los datos necesarios para la gestión de cuentas contables.
tieneOperaciones($idcuenta)	Responsabilidad que se encarga de buscar si una cuenta tiene asociada operaciones para poder ser eliminada, aperturada o sacada de vigencia.
obtenernaturalezaAction()	Responsabilidad que se encarga de mostrar las naturalezas existentes para ser asignadas a las cuentas.
obtenerconteconomicoAction()	Responsabilidad que se encarga de mostrar los contenidos económicos para ser asignado a las cuentas.
cargarparteformatoAction()	Responsabilidad que se encarga de obtener las partes del formato asociado a la entidad para la validación de las cuentas.
pudeteneraperturaAction()	Responsabilidad que se encarga de validar si una cuenta puede ser aperturada.
cargarnomsapertAction()	Responsabilidad que se encarga de obtener las configuraciones de aperturas existentes parta la entidad.
cargararbolnomapertAction()	Responsabilidad que se encarga de cargar un nomenclador arbólico para realizar las aperturas.
cargardetailsnomapertAction()	Responsabilidad que se encarga un nomenclador para realizar las aperturas.
insertarcuentaAction() eliminarcuentaAction() modificarCuentaAction()	Responsabilidades que se encargan del proceso de gestión de las cuentas contables, actualizando el historial de cuentas contables.
crearaperturatreeAction()	Responsabilidad que se encarga de gestionar la apertura por un

	nomenclador arbólico.
crearaperturaAction()	Responsabilidad que se encarga de gestionar la apertura por un nomenclador.
obtenercuentasAction()	Responsabilidad que se encarga de mostrar todas las cuentas existentes para la entidad.

Tabla 50 Descripción de la clase controladora ClasificadorcuentasController.

Nombre: cargartipocuentaController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
asignartipocuentaAction ()	Responsabilidad encargada de validar todos los datos necesarios para configurar los estados financieros y redireccionar a la interfaz
cargarcuentassintipoAction ()	Responsabilidad que se encarga de cargar las cuentas para la asignación de tipos,
cargartipocuentaAction ()	Responsabilidad que se encarga de cargar los tipos de cuentas.
modificarcuentaAction ()	Responsabilidad que se encarga de modificar los tipos a las cuentas.

Tabla 51 Descripción de la clase controladora cargartipocuentaController.

Clases del componente: Comprobante de Operaciones.

Nombre: ComprobanteController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
comprobanteAction ()	Responsabilidad encargada de validar todos los datos necesarios para gestionar un comprobante de operaciones y redireccionar a la interfaz.
cargarctaAction()	Responsabilidad que se encarga de mostrar las cuentas para la gestión del comprobante de operaciones.
crearcomprobanteparteAction()	Responsabilidad principal para la gestión del comprobante de operaciones.
addCO() addAsiento() addPase() addAnexo() ModificarPase() ModificarAnexo() Eliminapase() Eliminaanexo() Duplicar()	Responsabilidades que se encargan de la gestión del comprobante de operaciones
cargargridmovAction()	Responsabilidad que se encarga de mostrar los movimientos contables del comprobante en elaboración.
cargargridanexoAction()	Responsabilidad que se encarga de mostrar el registro de anexo al pase de un pase en elaboración.
cargarmonedaAction()	Responsabilidad que se encarga de mostrar las monedas contables para la gestión del comprobante.
cargartasaAction()	Responsabilidad que se encarga de mostrar las tasa de una

	moneda dada para la gestión del comprobante de operaciones.
validaCO ($id)	Responsabilidad que se encarga de validar un comprobante de operaciones es decir: la suma de los saldos del debe es igual al del haber.
centrocostoAction() objetogastoAction()	Responsabilidad que se encarga de mostrar los centro de costo y elementos del gasto asociados a la cuenta de gasto para la gestión del registro de anexo al pase.

Tabla 52 Descripción de la clase controladora ComprobanteController.

Nombre: RegistrocomprobantesController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
registrocomprobantesAction()	Responsabilidad encargada de validar todos los datos necesarios para la utilización del registro de comprobantes y redireccionar a la interfaz.
cargargAction()	Responsabilidad que se encarga de mostrar los comprobantes ya existentes y realizar las operaciones de búsqueda de los mismos.
estadoAction() numerocompAction()	Responsabilidades que se encargas de mostrar los estados y los números de los comprobantes existentes para agilizar la búsqueda de los mismos.
cambiarnumeroAction()	Responsabilidad que se encarga del cambio del número del comprobante.
eliminarAction()	Responsabilidad que se encarga de la eliminación de los comprobantes.
cambiarestadoterminadoAction() cambiarestadoasentadoAction()	Responsabilidades que se encargan en el cambio de estado de los comprobantes. Estados: Terminado y Asentado.
duplicarinvertirAction()	Responsabilidad que se encarga de invertir o duplicar un comprobante de operaciones.
imprimirAction()	Responsabilidad que se encarga de imprimir un comprobante de operaciones en el formato de impresión.

Tabla 53 Descripción de la clase controladora RegistrocomprobantesController.

Clases del componente: Cierre.

Nombre: CierrecontableController.	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
cierrecontableAction ()	Responsabilidad encargada de validar todos los datos necesarios para realizar un cierre de ejercicio o período contable y redireccionar a la interfaz.
cerrarperiodoAction ()	Responsabilidad que se encarga de gestionar el cierre del período contable.
cerrarejercicioAction ()	Responsabilidad que se encarga de gestionar el cierre del ejercicio contable.

Tabla 54 Descripción de la clase controladora CierrecontableController.

Glosario de Términos

[A]

Algoritmo: Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

Apertura: Iniciación de las operaciones contables después de realizada determinadas actividades al cierre de un período o ejercicio económico.

Atributo: Contenedor de un tipo de datos asociados a un objeto, que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.

Abstracción: Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción.

[C]

Ciclo contable: Serie de actividades contables que tienen lugar desde el comienzo hasta el final de un período contable. En cada período se repite la misma secuencia de procedimientos contables que comienzan con el registro de los hechos económicos que tienen lugar durante el período y concluye con la elaboración de los estados financieros correspondientes.

Cierre del ejercicio económico: Acciones y anotaciones contables realizadas al final del ejercicio económico destinadas a cerrar los libros de contabilidad al efectuar los últimos asientos de la misma.

Componente: Un componente es una parte no trivial, casi independiente, y reemplazable de un subsistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. Un componente se conforma y provee la realización física por medio de un conjunto de interfaces.

Contabilidad: Es la acción de registrar, clasificar y resumir, en términos monetarios, las operaciones económicas que acontecen en una entidad y por medio de ella se interpretan los resultados obtenidos, representando un medio efectivo para la dirección de la entidad.

Clase: Definiciones de las propiedades y comportamiento de un conjunto de objetos concretos. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Componentes de un objeto: Atributos, identidad, relaciones y métodos.

[E]

Estados financieros: Resumen de la información al cierre del período o ejercicio económico que refleja los elementos primordiales que sintetizan la situación económica de una entidad. Estos constituyen la fuente de información para la propia entidad y para el nivel superior que la atiende financieramente.

Evento: Un suceso en el sistema. El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.

Estado interno: Es una propiedad invisible de los objetos, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto.

Encapsulamiento: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.

[F]

Framework: 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.

[G]

GPL: La licencia pública general de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

[H]

HTTPS: URL creada por Netscape Communications Corporation para designar documentos que llegan desde un servidor WWW seguro. Esta seguridad es dada por el protocolo SSL (Secure Sockets Layer) basado en la tecnología de encryptación y autentificación desarrollada por la RSA Data Security Inc.

HTML: Acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. HTML es una aplicación de SGML conforme al estándar internacional ISO 8879.

Herencia: las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que reimplementar su comportamiento. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple; esta característica no está soportada por algunos lenguajes (como Java).

[L]

LGPL: La Licencia Pública General Menor (inglés: Lesser General Public License o LGPL) es una modificación de la licencia GPL. La LGPL reconoce que muchos desarrolladores de software no utilizarán el código fuente que se distribuya bajo la licencia GPL, debido a su principal desventaja que determina que todos los derivados tendrán que seguir los dictámenes de esa licencia. La LGPL permite que los desarrolladores utilicen programas bajo la GPL o LGPL sin estar obligados a someter el programa final bajo dichas licencias.

[M]

Moneda contable: Moneda definida en el nomenclador de monedas que se utiliza para el registro contable de las operaciones, según la legislación vigente.

Moneda original: Moneda definida en el nomenclador de monedas en el que se realizan las diferentes transacciones.

Método: Algoritmo asociado a un objeto, cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

Mensaje: Una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.

[N]

Nomenclador: Catálogo de reglas estipuladas a nivel central o de entidad.

[S]

Sistema de software: Es un programa o aplicación de software que permite a los usuarios el control o realización de varias tareas y que hacen que el trabajo sea más cómodo, rápido y eficiente.

Smalltalk: Es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

Subsistema: Son las partes que forman un sistema. Cada sistema está compuesto de subsistemas, los cuales a su vez son parte de otros subsistemas; cada subsistema es delineado por sus límites.

SSL: Siglas de Secure Socket Layer. Es un protocolo desarrollado por Netscape Communications Corporation para dar seguridad a la transmisión de datos en transacciones comerciales en Internet. Utilizando la criptografía de llave pública, SSL provee autentificación del servidor, encriptar de datos, e integridad de los datos en las comunicaciones cliente/servidor.

Shell de Windows: Programa que determina el aspecto del escritorio Es el contenedor dentro de la que toda la interfaz de usuario se presenta, incluyendo la barra de tareas, el escritorio, el explorador de Windows, así como muchos de los cuadros de diálogo y controles de interfaz.

[O]

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad. Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema.

[P]

Período económico: Período de tiempo inferior a un año por el cual se confeccionan las informaciones financieras. Cuando el período es de un año este se denomina convencionalmente ejercicio económico.

Proceso: Conjunto de actividades que guían los esfuerzos de las personas implicadas para el cumplimiento de un objetivo.

Principio de ocultación: Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que específica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica. Algunos lenguajes proporcionan medios más estáticos (en "tiempo de compilación") de polimorfismo, tales como las plantillas y la sobrecarga de operadores de C++.

[R]

Representación de un objeto: Un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

[T]

Tasa de cambio: Es el precio en moneda doméstica de una unidad de moneda extranjera o convertible.

[U]

UML: Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.