Universidad de las Ciencias Informáticas

Facultad 3

Título: Componente para la reevaluación de cuentas del Sistema Integral de Gestión Cedrux.

Trabajo de diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor: Yessika Martín Pérez.

Tutores: MSc. Osmar Leyet Fernández.
Ing. Tamara Rodríguez Sánchez.

La Habana.
2013.

"El mundo camina hacia la era de la informática, la electrónica, las comunicaciones y todos sus derivados. Todo parece indicar que estas ciencias se concientizarán en algo como una medida del desarrollo, quien las domine será un país de vanguardia.

Vamos a volcar nuestros esfuerzos en ese sentido con audacia revolucionaria."

Ernesto "Che" Guevara.

DECLARACIÓN DE AUTORÍA

Declaro por este medio que yo, Yessika Martín Pérez soy la autora del trabajo “Componente para la reevaluación de cuentas del Sistema Integral de Gestión Cedrux” y autorizo a la Universidad de las Ciencias Informáticas hacer uso del mismo en su beneficio, así como los derechos patrimoniales, con carácter exclusivo.

Para que así conste firmó el presente en La Habana a los ____ días del mes de _____ del año _____.

Autor:

_____

Yessika Martín Pérez.

Tutores:

_____ _____

MSc. Osmar Leyet Fernández Ing. Tamara Rodríguez Sánchez

DATOS DEL CONTACTO

 MSc. Osmar Leyet Fernández

- ✓ Miembro del equipo de implementación del sistema Saren, del proyecto Registros y Notarías.
- ✓ Analista Principal del sistema de Comercio Electrónico B2B para la empresa Cubalse de la república de Cuba.
- ✓ Arquitecto de Sistema Integral de Gestión Cedrux, del proyecto ERP Cubano.
- ✓ Jefe de la línea de desarrollo de Contabilidad y Finanzas del Sistema Cedrux, del proyecto ERP Cubano.

Institución: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Boyeros, La Habana, Cuba.

e-mail: oleyet@uci.cu

 Ing. Tamara Rodríguez Sánchez

- ✓ Analista principal de Cedrux, del programa ERP-Cuba.
- ✓ Jefa de proyecto Cedrux 1.1 del programa ERP-Cuba.

Institución: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Boyeros, La Habana, Cuba.

e-mail: trodriquez@uci.cu

AGRADECIMIENTOS

Agradecer a todas las personas que ayudaron a la realización de esta tesis:

A mi mamita linda, a mi mamita del alma. Por estar aquí hoy a pesar de todas las adversidades, por ser tan buena, comprensiva y paciente todos estos años conmigo, por ser mi amiga, mi confidente. Eres lo más lindo que tengo en mi vida y tus consejos y reflexiones siempre estarán conmigo. Es lo que más quiero en el mundo y sé que tú lo sabes.

A mi papá por sus regaños constructivos y por su apoyo incondicional, por todo su sacrificio y su amor, porque este, más que mi sueño, es también tu sueño hecho realidad. Tus consejos me han hecho crecer y tu ejemplo me ha hecho seguirte. Te adoro.

A Alena, por quererme como si fuese su hija, gracias por apoyarme y ser incondicional conmigo. Por toda tu preocupación, por todos los momentos que hemos vivido y compartido junticas, sabes que en mi corazón tienes un lugar muy preciado.

A Erisel, por todo lo que me ha dado, por tratarme también como una hija, por no tener diferencias, por siempre escucharme y ayudarme. Porque sé que aunque estés lejos estas muy orgulloso de mi.

A mi abuelita Estela y mi abuelito Mario por siempre estar cerquita cuidándome y apoyándome.

A mi hermanos que los adoro.

A mis tías y primas a las cuales les debo muchísimo, gracias por siempre estar al pendiente de mí y de lo que necesite.

A Crisvel, (Mi Cris), por ser tan especial conmigo y hacerme muy feliz desde que nos conocimos. Por dedicarme todo su tiempo, por su entrega y dedicación, por calar tan hondo en mi alma. Por enseñarme que ante las dificultades se crece. Por eso y más te amo mucho mucho.

Un agradecimiento súper especial a mis tutores, de corazón que fueron los mejores del mundo, miles de gracias por soportarme, por el apoyo y la ayuda que me han brindado; a Osmar porque a pesar de ser una persona que inspira mucho respeto, me ha brindado grandes de conocimientos de los cuales me he nutrido y buscaba siempre una solución a los problemas. A Tamara por ser tan dulce y buena, por siempre estar pendiente de mí, por enseñarme a pensar, por su preocupación y sus consejos.

A mi otra familia la de mi novio, por aceptarme como un miembro más, por todas las risas, consejos y todo el apoyo. Pedro miles de gracias por la ayuda en los preparativos para la defensa de la tesis.

A Mayte gracias por tu ayuda en los momentos difíciles y gracias por tu amistad.

A Miguel, Ernesto y Carlos por ser personas maravillosas que he tenido la dicha de conocer a lo largo de mi carrera, por ayudarme en los momentos en los que más los necesite

A todos mis profesores que han ido dejando una huella profunda en mí, para llegar a ser el profesional que siempre soñé ser. A Omarito, Sisley, Leo, William, Alain, Dailien miles de gracias.

A mis amigos de la universidad y de Taguasco por compartir muchos momentos buenos y malos conmigo, que nunca los olvidaré esté donde esté, siempre serán mis amigos.

RESUMEN

El país avanza con paso firme dentro del proceso de utilización de las Tecnologías de la Información y las Comunicaciones (TIC), siendo este un factor decisivo para el desarrollo de las empresas, la economía y la sociedad cubana. En la Universidad de las Ciencias Informáticas (UCI) se viene desarrollando una solución informática denominada Cedrux, con el propósito de informatizar los principales procesos de las entidades presupuestadas y empresariales. En el presente trabajo se expone un estudio de varios sistemas de gestión contable para comprobar de forma teórica como se ha comportado el proceso de reevaluación de cuentas en cada uno de ellos. La aplicación del estudio y propuesta principal de la investigación es el desarrollo de un componente que se encargue de la reevaluación de cuentas en Cedrux, para garantizar la integridad de los saldos contables. Su importancia radica en que facilita la toma de decisiones a los usuarios interesados en la situación económica y financiera de las empresas. Se identificaron los requisitos funcionales con los que debe cumplir el componente. Se desarrolló un modelo de diseño que fue implementado posteriormente. Por último se realizaron un conjunto de pruebas con el fin de validar la calidad del componente desarrollado.

PALABRAS CLAVES:

Cuenta, reevaluación de cuentas.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las exigencias y los avances de las TIC en la actualidad, juegan un papel importante en las estrategias del negocio para mejorar los procesos claves de las empresas. Pasaron de un rol operativo que solo resolvía cuestiones técnicas, a jugar un rol estratégico dentro de las empresas. El ambiente competitivo que vive hoy por hoy el sector empresarial, requiere de promover los procesos y actividades de negocio que generan las ventajas competitivas de las compañías ante sus más fuertes competidores. Por lo que se necesitan sistemas de información eficiente, precisa, oportuna y con mayor calidad.

En los últimos tiempos, la demanda de los sistemas de gestión se ha hecho mayor, lo que hace imprescindible la optimización e integración de los flujos internos de información y las relaciones comerciales externas. La implantación de un sistema de gestión, sirve de soporte para una administración eficiente. Permitiendo la disponibilidad de la información, la reducción de tiempos y costos de los procesos y mejora del servicio al cliente.

Con el fin de tomar mejores decisiones las empresas requieren de herramientas que les permitan controlar la información de manera resumida, íntegra y confiable; trayendo consigo el surgimiento de Sistemas de Planificación de Recursos Empresariales (ERP por sus siglas en inglés). Los ERP constituyen una solución robusta conformada por sistemas de gestión de información que integran e informatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos con el cual se espera tener integradas todas las áreas y departamentos dentro de la empresa (Correa, 2004).

La autonomía de un ERP está determinada por tres características fundamentales: son sistemas integrales, modulares y adaptables (Tahirí, y otros, 2012):

Integrales: porque permite controlar los diferentes procesos de la empresa entendiendo que todos los departamentos o áreas se encuentran interrelacionados entre sí por la información que comparten y que se genera. El resultado de un proceso es el punto de partida del otro.

Modulares: porque están formados por un número específico de módulos, independientes entre sí, pero que a la vez están comunicados, lo que permite una gran adaptabilidad a las empresas de acuerdo a su tamaño y disponibilidad de recursos. Los cuales se instalan de acuerdo a los requerimientos del cliente.

Adaptables: porque están creados para adaptarse a la peculiaridad de cada empresa, sin tener en cuenta las características de los procesos de negocio.

Debido a la gran utilidad e importancia de los ERP, el país no se encuentra ajeno al uso de los mismos. En la actualidad, no existe en Cuba un sistema informático integral de gestión que cumpla con la totalidad de los requerimientos de funcionalidad, interoperabilidad y seguridad que espera el gobierno cubano de una solución de este tipo. De manera que pueda ser utilizada como herramienta para potenciar el

cumplimiento de las funciones de las entidades a todos los niveles con un máximo de racionalidad y control de los recursos financieros, materiales y humanos. Como parte del fortalecimiento de la gestión y la informatización de la sociedad cubana, la dirección del país, ha planteado la necesidad de informatizar los principales procesos de las entidades presupuestadas y empresariales (del Toro, y otros, 2008). Emprender un proyecto para la construcción de un ERP propio es un reto. Dicho proyecto, se decidió llevar a cabo en el Centro para la Informatización de la Gestión de Entidades (CEIGE), enmarcado en la UCI y en conjunto con el Ministerio de Finanzas y Precios (MFP) y el Ministerio de Economía y Planificación (MEP). Como resultado se obtendrá el ERP cubano registrado como: el Sistema Integral de Gestión Cedrux.

El proceso de reevaluación de cuentas constituye una necesidad para el país pues surge como consecuencia directa del tratamiento de la multimoneda. En la actualidad Cedrux, no brinda la posibilidad de realizar este proceso ya que solo puede efectuarse manualmente a partir de los datos que brinda el sistema. La reevaluación de cuentas implica fórmulas matemáticas que relacionan el importe de la moneda original, con el factor de conversión y la tasa de cambio de la moneda en la que se va a reevaluar, que puede provocar errores contables. Esta operación tiene una estrecha relación con la cuenta en la que se va a reflejar el efecto de la reevaluación pues se debe tener en cuenta si ingresa ganancia por partida no monetarias o pérdida por partida no monetarias. Para que la información sea íntegra, debe estar completa y ser lo más fiable posible, de ahí que omitir la realización del proceso implicaría que:

- ✓ Los saldos contables estén desactualizados en dependencia de la variabilidad de las escalas de conversión y sean deficientes en términos de relevancia por lo que afectaría la toma de decisiones en una entidad determinada.
- ✓ No se tiene dominio real del capital, afectando la gestión empresarial y el análisis financiero en las entidades cubanas.

De la problemática antes explicada se identifica el siguiente **problema a resolver**: ¿Cómo realizar la reevaluación de cuentas de manera que se contribuya a la integridad de los saldos contables que maneja el sistema Cedrux?

Se define por lo tanto como **objeto de estudio:** los sistemas informáticos en la gestión de procesos contables.

Para dar solución al problema se plantea entonces como **objetivo general** de la investigación:

Desarrollar un componente para la reevaluación de cuentas que contribuya a la integridad de los saldos contables que maneja el sistema Cedrux.

Incidiendo en el **campo de acción:** componentes de software para el proceso de reevaluación de cuentas.

Al logro del objetivo general tributan los siguientes **objetivos específicos:**

- ✓ Realizar el marco teórico de la investigación relativo al proceso de reevaluación de cuenta.
- ✓ Desarrollar el análisis correspondiente para la identificación de las funcionalidades y características del componente.
- ✓ Diseñar la estructura de componentes, clases, datos aplicando los patrones de diseño y técnicas consideradas como necesarias en el estudio.
- ✓ Realizar la implementación del componente siguiendo las técnicas de programación estudiadas en la plataforma de desarrollo seleccionada.
- ✓ Validar el componente a partir de la ejecución de técnicas y métricas aplicables al desarrollo de la solución.

Para cumplir con los objetivos específicos se planificaron las siguientes **tareas de la investigación:**

Objetivo específico 1:

- ✓ Elaboración de un informe sobre la descripción del proceso reevaluación de cuentas, su implementación en otros sistemas ERP y su relación funcional con el resto de los procesos del sistema ERP; destacando su importancia.
- ✓ Elaboración de un informe sobre las buenas prácticas a seguir (Resumen) en el desarrollo de las fases de análisis, diseño, implementación y pruebas, tomando como guía para el desarrollo de estas fases el modelo de desarrollo propuesto por el centro CEIGE.

Objetivo específico 2:

- ✓ Identificación, descripción y validación de los requisitos del componente.
- ✓ Elaboración del mapa conceptual.

Objetivo específico 3:

- ✓ Elaboración del diseño de la base de dato.
- ✓ Elaboración de los diagramas de clases propuesto por el modelo de desarrollo.

Objetivo específico 4:

- ✓ Implementación del diseño propuesto.
- ✓ Presentación del estándar de codificación usado.
- ✓ Documentación del código.

Objetivo específico 5:

- ✓ Validación del diseño propuesto.

- ✓ Elaboración los casos de pruebas necesarios.
- ✓ Realización de las pruebas de caja blanca.
- ✓ Liberación del componente por el equipo de calidad del centro.

Planteándose la siguiente **idea a defender:**

Si se desarrolla un componente para la reevaluación de cuentas se contribuirá a la integridad de los saldos contables que maneja el sistema Cedrux.

Alcanzando como **posibles resultados:**

- ✓ El fortalecimiento del subsistema Configuración General que forma parte del Sistema Integral de Gestión Cedrux.
- ✓ Todos los artefactos correspondientes a las fases de análisis, diseño, implementación y pruebas, relativo al proceso de reevaluación de cuentas.
- ✓ La incorporación de dicho componente a la versión 1.1 de Cedrux a desplegarse en las entidades del país.

Con la intención de mantener lo cualitativo y lo cuantitativo equilibrado en el desarrollo de la tesis y para dar cumplimiento a las tareas de investigación propuestas se utilizarán los **métodos científicos.**

Métodos teóricos:

Histórico-Lógico: Se utilizó para comprobar de forma teórica, como se ha comportado el proceso de reevaluación de cuentas en la implementación de algunos sistemas ERP a lo largo de su proceso de desarrollo. Esto a partir del estudio de la evolución que han tenido los conceptos más importantes que se tratan en la investigación.

Analítico-Sintético: Después de hacer un análisis minucioso de la bibliografía y los referentes teóricos en cuestión, se hizo una extracción y síntesis de los principales elementos más significativos relacionados con el proceso de reevaluación de cuentas. Teniendo como finalidad la base teórica para el desarrollo y completamiento en la solución.

Métodos empíricos

Entrevistas: Se utilizó porque permitió obtener información del proceso de reevaluación de cuentas, intercambiando directamente con las personas involucradas para su desarrollo. Además que permite la recopilación de la información para la almacenamiento de requisitos del componente.

Estructura del documento

Con el propósito de organizar la información de la investigación realizada y garantizar un mayor entendimiento, el contenido se estructuró en 3 capítulos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se analizarán los temas que se necesitan investigar para mostrar el proceso de la reevaluación de cuentas, realizando un estudio del estado del arte. Se hará un análisis minucioso del modelo de desarrollo propuesto por CEIGE y se conceptualizarán elementos importantes para la captura y la validación de requisitos; así como los distintos patrones de diseño, las herramientas, lenguajes y tecnologías a utilizar para el diseño e implementación del componente.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

En este capítulo se describirá la propuesta de solución del componente a desarrollar y el proceso que será informatizado. También se presentarán los requisitos funcionales y no funcionales con los que debe cumplir el componente propuesto. Se hará referencia a las técnicas empleadas tanto en la captura como en la validación de los requisitos. Se mostrarán el modelo de datos del sistema y el diagrama de clase del diseño que será implementado. Además se especificará la utilización de un conjunto de patrones dentro del diseño del componente. Por último se realizará una breve descripción del marco de trabajo utilizado, así como los estándares a utilizar durante la implementación del componente, describiendo las clases y funcionalidades. En consecuencia, se obtendrá un conjunto de artefactos establecidos por la metodología de desarrollo propuesta en el capítulo 1.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En este capítulo se realizará el diagrama de despliegue para obtener una visión más clara sobre el componente. Se validará el diseño de la solución a través de métricas, se describirán las pruebas de caja negra y caja blanca realizadas al componente y los resultados obtenidos de cada una de ellas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción.

El presente capítulo estará centrado en la realización de un análisis profundo para obtener la mayor cantidad de información actualizada de algunos sistemas informatizados que se han implementado nacional e internacionalmente. Se conocerán los principales aspectos que deben ser tomados en cuenta para que el componente esté al nivel de las tendencias actuales de la industria del software. Además se explicará el modelo de desarrollo de software, por el cual se rige esta investigación a partir de la cual estará encaminada la solución. Teniendo en cuenta el marco de trabajo del Sistema Integral de Gestión Cedrux, se realizará un análisis de la arquitectura definida en el sistema y se seleccionarán las tecnologías, lenguajes de programación y herramientas a utilizar.

1.2 Consideraciones generales para el proceso de la reevaluación de cuentas.

El proceso de reevaluación de cuentas se ha convertido en una necesidad del país, puesto que surge como consecuencia directa de la existencia de monedas distintas a la contable durante el proceso de registro de las operaciones económicas. Esto se puede hacer en cuentas de cualquier tipo de moneda, dependiendo directamente del importe de la moneda original, con el factor de conversión y la tasa de cambio de la moneda en la que se va a reevaluar, ya que es una operación que permite actualizar el saldo de las cuentas. Con este proceso se permite tener un total control y dominio de la información a partir de la actualización de los saldos contables, lográndose un dominio real del capital trayendo consigo una buena gestión empresarial.

1.2.1 Sistemas de gestión contable que incluyen el proceso de reevaluación de cuentas.

El sistema económico cubano, tiene características especiales y en la actualidad muchas entidades presupuestadas y empresariales necesitan de sistemas de gestión contable que le permitan realizar el proceso de reevaluación de cuentas. El estudio del estado del arte se centró fundamentalmente en estudiar y comprender el tratamiento que recibe este tema en sistemas ya existentes, debido a las implicaciones que posee.

Para comprender el dominio del problema se analizaron varios sistemas de gestión contable dentro de los cuales se encuentran: SAP R/3, Microsoft Dynamics, Openbravo, VERSAT-Sarasola, Rodas XXI y Assets NS.

Sistemas de gestión contable extranjeros.

SAP R/3: Sistema alemán que permite controlar todos los procesos que se realizan en una empresa. Se define como software abierto, multiplataforma y diseñado para manejar las necesidades de información de la empresa (SAP, 2010). Controla todos los procesos de una empresa y la integración total de sus módulos ofrece real compatibilidad de sus funciones, mejorando la toma de decisiones. Tiene presente

que la mejora continua y los procesos de calidad, conjuntamente con el desarrollo de estrategias de satisfacción, proporcionan productos y servicios de software al nivel del entorno competitivo mundial.

En lo referido al tema de la investigación este sistema es capaz de diferenciar entre la moneda local y la moneda de transacción. Donde las empresas que lo utilizan puedan procesar sus operaciones, aun cuando la moneda local fuese distinta. Igualmente, brinda la posibilidad de convertir informes individuales como balances o cuenta de pérdidas y ganancias en su moneda local, aún en el caso de que continúen llevando su contabilidad en moneda nacional (SAP, 2010).

Sus inconvenientes principales son: ser un software privativo que funciona sobre el sistema operativo Windows, costoso y muy complejo. Presenta soporte para bases de datos Oracle y además es un software que no cuenta con la funcionalidad de multientidad (SAP, 2010). Por lo tanto no se ajusta a las condiciones y necesidades tecnológicas del país.

Microsoft Dynamics: Desde sus inicios se estableció como opción ideal para organizaciones de tamaño medio que buscan una solución completa de sus problemas. Es un producto de la compañía Microsoft que se integra para la gestión empresarial, muy fácil de usar y ofrece una adaptabilidad rápida a todo tipo y escala de negocio que permite una gran certeza en la toma de decisiones.

En lo referido al tema de la investigación, se tiene que la administración multimoneda permite manejar un número ilimitado de monedas, así como las tasas de cambio para cada una de ellas. Permite además: transacciones en la moneda original o en la moneda funcional con su equivalente reflejado en la pantalla, registrar tasas de cambio con la hora y la fecha permitiendo el uso de múltiples tasas de cambio por día y generar reportes financieros en varias monedas para compartirlos con sus asociados alrededor del mundo (Microsoft Corporation, 2012).

Sus inconvenientes principales son: no admitir dualidad monetaria y ser un software privativo que funciona sobre el sistema operativo Windows, que niega el derecho de poder modificarlo. Para su uso se tienen que pagar grandes costos por el tema de las licencias, por lo que no se ajusta a las necesidades tecnológicas del país.

Openbravo: Constituye la solución ERP líder mundial en software libre y entorno web. Está dirigido a cualquier tipo de empresa de pequeño y mediano tamaño, es fácil de utilizar. Ha sido desarrollado siguiendo el patrón arquitectónico Modelo-Vista-Controlador (MVC) que facilita la separación de las áreas de desarrollo. Sigue un licenciamiento de software libre asegurando el acceso público al código fuente y tiene la posibilidad de ser modificado y actualizado sin la implicación de grandes gastos (Openbravo, 2007).

En lo referido al tema de la investigación, el sistema admite definir rangos de conversión, siendo estos valores los que se utilizarán para el cálculo de los importes de los documentos en diferentes divisas, donde se precisa el período de tiempo en el que se validará la conversión. Contabiliza en base a la

moneda establecida en el esquema contable, utilizando los rangos de conversión se introduzcan en el sistema (Openbravo, 2007).

Sus inconvenientes principales son: no tener implementadas las operaciones de reevaluación de cuentas lo que implica que estos cálculos se realicen mediante el método manual y no admitir la dualidad monetaria.

Assets NS: Sistema estándar y parametrizado que permite el control y la integración de sus módulos y la estrecha relación de cada uno de ellos. Se adapta fácilmente a las exigencias y las características de cada entidad en particular. Este diseñado para Multi compañía, con una estructura organizativa a varios niveles (Assets, 2005).

En lo referido al tema de la investigación admite trabajar con los diferentes tipos de monedas que la empresa maneja para emitir los estados financieros con una tasa determinada. Permite tener al día la contabilidad de la empresa ya que genera asientos diarios de cada una de las transacciones de los diferentes módulos.

Su inconveniente principal, es ser una aplicación importada desarrollada para plataformas privativas por lo que no se ajusta a las necesidades tecnológicas del país, además de no realizar la reevaluación de cuentas.

Sistemas de gestión contable nacionales.

VERSAT-Sarasola: Primer sistema cubano de gestión contable que adoptó la postura de adaptarse a las características generales del país, tiene la posibilidad de ser modificado y actualizado sin la implicación de grandes gastos. Informatiza casi todas las actividades de la empresa y le brinda la oportunidad al usuario de almacenar toda la información en una base de datos centralizada lo que garantiza: consistencia, seguridad y calidad de sus datos (Cabrera González, y otros). Además de ser un instrumento de fácil manejo, eficaz y rápido.

En lo referido al tema de la investigación admite trabajar con los diferentes tipos de monedas que la empresa desee utilizar para revalorizar los estados financieros en una fecha y con una tasa determinada. Determinar además las ganancias o las pérdidas que se produce por la aplicación de las tasas de cambio, vigentes en el momento de las transacciones. También se procesan y contabilizan los documentos primarios.

Sus inconvenientes principales son: no poseer una centralización de estos aspectos, sino que se hace en dependencia de las necesidades que tenga cada módulo. Está implementado sobre tecnología privativa, compatible con el sistema operativo Windows. Es una aplicación de escritorio implementada en Delphi que es propietaria. Además es un software que no cuenta con la funcionalidad de multientidad.

Rodas XXI: Sistema multiempresa desarrollado por CITMATEL que brinda la posibilidad de informatizar el funcionamiento de cualquier empresa. Actualmente cuenta con ocho módulos (Rodas XXI, 2002): Finanzas, Contabilidad, Activos Fijos, Nóminas, Inventario, Facturación, Recursos Humanos y Telecobranzas; que pueden usarse integrados en su totalidad o de forma independiente. Crea reportes fácilmente y la información la protege mediante claves.

En lo referido al tema de la investigación se tiene que permite el trabajo con la dualidad monetaria que permite realizar operaciones tanto en moneda nacional como en moneda extranjera (Rodas XXI, 2002). Permite tener el control detallado de la ejecución del presupuesto ya que se pueden obtener datos actualizados de él a partir de los estados financieros que se obtienen a partir de informes generados.

Sus inconvenientes principales son: que está desarrollado para plataformas privativas, utiliza Microsoft SQL Server 2000 como servidor de base de datos y no permite la realización del proceso de reevaluación de cuentas, por lo que no cumple con las necesidades de independencia tecnológicas del país.

Valoración de los sistemas estudiados.

A partir del estudio de los sistemas de gestión contables se implantó un análisis comparativo de los principales parámetros que deben tener para satisfacer las necesidades del país y los requerimientos necesarios de Cedrux. Ver (Tabla 1).

Parámetros	**Sistemas de gestión contable**					
	Nacionales		**Extranjeros**			
	Rodas XXI	V.Sarasola	Assets	M.Dinamys	Openbravo	SAP ERP
Tipo de software	Privativo	Privativo	Privativo	Privativo	Libre	Privativo
Multiplataforma	No	No	No	Si	No	Si
Licencias	Si	Si	Si	Si	Si	Si
Realizar reevaluación de cuentas	Si	No	No	No	Si	Si
Multimoneda	No	No	Si	Si	Si	Si
Dualidad monetaria	Si	Si	Si	No	No	Si

Tabla 1: Comparación de los sistemas estudiados.

De los sistemas estudiados solo el 16,7% son de software libre, donde solamente el 33,3% de estos son multiplataforma. De ellos el 50% permiten la realización de la reevaluación de cuentas, de los cuales el 100% necesitan licencias costosas para su funcionamiento. El tratamiento de la multimoneda se ve reflejado en un 67,7% de los sistemas analizados y los términos de la dualidad monetaria lo manejan un 67,7%. En términos generales, estos sistemas de gestión contable presentan la limitante de no respaldar

los Lineamientos de la Política Económica y Social del VI Congreso del Partido Comunista de Cuba (PCC) para el desarrollo de soluciones bajo el principio de independencia tecnológica, además de que proporcione seguridad y fiabilidad en la gestión de la información de las entidades nacionales (PCC, Abril 2011).

1.2.2 Relación funcional del proceso de reevaluación de cuentas con el resto de los procesos del sistema Cedrux. Importancia.

El proceso de reevaluación de cuentas tiene una estrecha vinculación con los procesos del sistema Cedrux que generan contabilizaciones. Al realizarse el proceso de reevaluación de cuentas se verían afectados los módulos de:

Caja

Este módulo, incluye la mayoría de los procesos financieros de una caja. Comprende los fondos monetarios con los que se operan en la empresa, donde en el arqueo y el depósito no debe existir ninguna diferencia entre el fondo y el efectivo con el que se cuenta. Por tanto si se realiza en una o varias cuentas una reevaluación, se tendrán que actualizar cada una de las operaciones que incluye el módulo fundamentalmente la de los recibos en efectivo que es una operación que se realiza como consecuencia de un cobro.

Costo

Si en este módulo se realiza una o varias reevaluaciones de cuentas, se deberá tener presente en el traspaso que se realiza entre las cuentas de gastos indirectos de la producción a la de gastos directos de la producción, puesto que las cuentas se registran en un componente que permite registrar estas operaciones contables.

Banco

Este módulo genera asientos contables que deben ser actualizados cuando se realice una reevaluación de cuentas, pues cuando se realiza el proceso de las liquidaciones de los cobros y pagos anticipados, los saldos tendrán que estar equilibrados en todos los lugares que aparezcan. También en el momento de realizar una conciliación bancaria, se deben tener en cuenta los valores registrados de la entidad con los que el banco suministra.

Cobros y Pagos

Este módulo genera asientos contables que se pueden invocar por diferentes vías: al efectuarse un cobro o pago de un derecho o una obligación y al liquidar una operación anticipada; que deben ser actualizados cuando se realice una reevaluación de cuentas.

Inventario y Facturación

En estos módulos deben ser actualizadas las cuentas cuando se realice una reevaluación, producto de que en los procesos recepción, despacho, ajuste y facturación se manejan saldos contables que de no estar actualizados provocarían errores en la contabilidad.

1.3 Modelo de desarrollo.

CEIGE propone un modelo de desarrollo que detalla el ciclo de vida de sus proyectos con la incorporación de los distintos subprocesos dictados por el Nivel II de Modelo de Capacidad de Madurez del Software (CMMI, por sus siglas en inglés) (Entidades, 2012).

Al realizar las modelaciones de las interacciones y relaciones que acontecen entre las personas (roles), las actividades que estas desarrollan en cada una de las fases y los artefactos de salida que se crean o actualizan durante el proceso, se logra que los elementos de un proceso de desarrollo de software y sus relaciones deben responder a Quién debe hacer Qué, Cuándo y Cómo (Jacobson, y otros, 2010).

Características.

Este modelo de desarrollo brinda la posibilidad de que todos los proyectos de CEIGE cuenten con un modelo estandarizado que incluye las fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. Como principales características se tiene que es orientado a componentes debido a que se desarrolla teniendo en cuenta la estructura de agrupar por componentes, es adaptable al cambio y a las características específicas de cada proyecto; siendo un modelo iterativo incremental que además está centrado en la arquitectura. Tiene en cuenta la Línea de Productos de Software (LPS).

Descripción de las fases del ciclo de vida de los proyectos.

Figura 1: Ciclo de vida de proyectos.

La fase **Inicio o Estudio preliminar** tiene por finalidad:

- ✓ Asegurar la factibilidad del proyecto.
- ✓ Establecer un plan para la ejecución del proyecto.

En esta fase se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. Realizándose un estudio inicial de la

organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto (Entidades, 2012). Teniendo como hitos de fase el Plan de desarrollo de software y el Acta de inicio del proyecto firmada.

La fase **Desarrollo** tiene por finalidad:

- ✓ Obtener un sistema que satisfaga las necesidades de los clientes y los usuarios finales.

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación (Entidades, 2012). Teniendo como hito de fase: el Producto liberado por la empresa certificadora de calidad.

1.4 Ingeniería de Requisitos.

Pressman afirmó: "Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software" (Pressman, 2005). La meta de la Ingeniería de Requisitos (IR) es entregar una especificación de requisitos de software completa y correcta pues cometer errores a la hora de entender los requisitos con la información que se suministra es fatal, ya que muchas decisiones del diseño dependen de ellos. Teniendo como objetivo principal proporcionar un modelo de las necesidades del problema que se debe resolver de forma clara, no ambigua y coherente. El ciclo de vida de la IR está compuesto por: elicitación, análisis, especificación y validación.

Para lograr una mayor comprensión y entendimiento de lo que es la IR y lo que constituye es necesario ante todo comprender la definición de un requisito, cómo se definen y para qué sirven. El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en inglés) define un requisito como: (IEEE, 1993)

- ✓ Una condición o capacidad necesaria para un usuario para resolver un problema o alcanzar un objetivo.
- ✓ Una condición o capacidad que debe ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto.

Los requisitos se pueden clasificar en dos grandes grupos los requisitos funcionales y los no funcionales que a su vez se dividen en varias categorías: (Jacobson, y otros, 2000).

Requisitos funcionales: Condición o capacidad que el sistema debe cumplir. Definen las funciones que el sistema será capaz de realizar. Deben ser capaces de definir y describir los procesos y actividades que componen el sistema completo.

Requisitos no funcionales: Propiedades o cualidades que el producto debe tener. Son aquellos que no están relacionados directamente con los procesos que definen el sistema, se enfocan en las características adicionales, tales como seguridad, hardware, software, apariencia o interfaz externa, seguridad, usabilidad y soporte.

1.5 Técnicas para la captura y validación de los requisitos.

El proceso de la identificación de los requisitos es clave en el desarrollo de un software y la comunicación y satisfacción del cliente dependen de ellos. Para esto se hace necesario el empleo de técnicas que permitan establecer una buena captura de requisitos y de validación de los cuales tienen como finalidad fundamental demostrar que su definición es la que el usuario necesita. La **captura de requisitos** sirve de base para verificar si los objetivos que se establecieron se cumplieron. Dentro de sus técnicas se encuentran (Sánchez-Segura, y otros, 2010):

- ✓ Entrevistas.

Es una de las más usadas ya que se establece una conversación entre personas de ambas partes, son aplicadas a los especialistas funcionales. Se emplean para obtener información de personas o de grupos donde se realizará el sistema de software.

- ✓ Tormenta de ideas (brainstorming)

Reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. Consiste en la simple acumulación de ideas sin detenerse en el análisis del valor de ellas, utilizada para el entendimiento común de los requisitos capturados y determinación de posibles cambios o errores en dicha captura.

- ✓ Cuestionarios.

Consiste en hacer preguntas relacionadas con varios aspectos del sistema, debe ser preciso y dirigido a quienes deben conocerlo. Debe ser simple específico, no patrocinado y con precisión técnica.

La **validación de los requisitos** tiene como objetivo demostrar que su definición es la que el usuario final necesita. Dentro de sus técnicas se encuentran (Sánchez-Segura, y otros, 2010):

- ✓ La Revisión Técnica Formal (RTF)

Son reuniones con el personal técnico con el objetivo de validar la especificación de requisitos. Su aplicación a los documentos de práctica permitirá detectar deficiencias, ambigüedades, omisiones y

errores de formato y contenido. Los especialistas funcionales deben ser independientes del equipo que ha realizado la especificación de requisitos.

- ✓ Auditorías

Revisar la documentación con esta técnica, consiste en un chequeo de los resultados contra una (Listas de Chequeo) predefinida o definida a comienzos del proceso, es decir solo una muestra es revisada.

- ✓ Prototipo de interfaz

Al diseñar los prototipos de interfaz se hacen simulaciones del posible producto, que luego son utilizados por el usuario final, permitiendo conseguir una importante retroalimentación en cuanto a si el sistema diseñado con base a los requerimientos capturados le permite al usuario realizar su trabajo de manera eficiente y efectiva.

1.6 Marco de trabajo.

En la actualidad los desarrollos de entornos web están creciendo considerablemente y se hace necesario contar con un software capaz de administrar, organizar y manejar ciertos procesos en la etapa de desarrollo. También conocido como Framework, puede tener soporte de programas, bibliotecas, un lenguaje interpretado, entre otros programas para desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Cedrux es un sistema de gestión empresarial que está implementado sobre la plataforma tecnológica **Sauxe** (Gómez Baryolo, y otros, 2011). Creado por el departamento de Tecnología del centro CEIGE. Contiene un conjunto de componentes reutilizables que logran una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo de software. **Sauxe 1.5**, como marco de trabajo tiene definidos frameworks para cada una de las capas como parte del MVC de cada componente dentro de la aplicación. Analizándolo se tiene que para la Vista se usa el ExtJs, para el Controlador: Zend-Framework, como base de su funcionamiento y para acelerar el acceso a datos en el modelo se utilizó Doctrine. A continuación se explicará brevemente cada uno de ellos (Gómez Baryolo, y otros, 2011):

ExtJs 2.2:

Es una librería construida con JavaScript, que proporciona una interfaz amigable a los usuarios. Siendo una de las más avanzada para el desarrollo de aplicaciones web, con una apariencia totalmente novedosa y una arquitectura flexible. Su potencia radica en la colección de componentes para el diseño de GUI's[1] del lado del cliente haciendo uso extensivo de Ajax. Permite crear aplicaciones complejas usando componentes predefinidos como son: cuadros y áreas de texto, campos para fechas, botones,

[1] **GUI:** Interfaz gráfica de usuario, conocida también como GUI (del inglés graphical user interface).

pestañas, barra de herramientas y muchas más (Gómez Baryolo, y otros, 2011). Posee una amplia documentación y tiene el diseño separado de la funcionalidad y la orientación a objetos.

Zend Framework 1.4:

Es un framework de alta calidad que se usa para el desarrollo de aplicaciones web y servicios web con PHP, que proporciona soluciones para construir sitios web modernos, robustos y seguros (ZEND FRAMEWORK, 2010). Es de código abierto y trabaja con PHP 5, simplifica la gestión de archivos de configuración y proporciona los componentes que forman la base del patrón MVC. Ampliando con diferentes características: facilita una capa de acceso a bases de datos, proporciona la capacidad de búsquedas en documentos y contenidos y además proporciona mecanismo de filtrado y validación para la entrada de datos.

Doctrine 1.2:

Doctrine es un potente y completo sistema ORM[2] para PHP 5.2.3 y posterior. Uno de sus puntos fuertes es su lenguaje DQL (Doctrine Query Language) inspirado en el HQL[3] de Hibernate, lenguaje para consultas a la base de datos (Doctrine, 2008). Un ORM es una técnica de programación que permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, es decir, las tablas de la base de datos se convierten en clases y los registros en objetos, que se pueden manejar con facilidad. Utilizar un ORM tiene una serie de ventajas que facilitan tareas comunes y de mantenimiento como: reutilización de código, encapsulación de lógica de datos, portabilidad ya que es capaz de traducirse a diferentes tipos de bases de datos, seguridad contra inyecciones de código y mantenimiento de código, gracias a la correcta ordenación de la capa de datos. (Sañudo Clemente, y otros, 2010).

1.7 Arquitectura.

La arquitectura proporciona una visión global del sistema a construir. Para el desarrollo de esta investigación se utilizará el marco de trabajo Sauxe, definido para el desarrollo de Cedrux, que se basa en el patrón arquitectónico MVC. Este estilo es usado fundamentalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas, facilita la programación en diferentes capas de forma paralela e independiente y sugiere la separación del software en 3 capas (Sánchez-Segura, y otros, 2010). Para más detalles, ver (Figura 2).

Modelo: Encapsula los datos y las funcionalidades. Es independiente de cualquier representación de salida y/o comportamiento de entrada. Es el encargado de llevar un registro de las vistas y controladores del sistema. Permite gestionar la información y acceder a la capa de almacenamiento de datos.

[2]ORM**:** Mapeador de relación de objetos (Object Relation Mapper, ORM por sus siglas en inglés)

[3]**HQL:** Lenguajes de consultas para Hibernate (Hibernate Query Language, HQL por sus siglas inglés). Es un ORM para el lenguaje JAVA

Vista: Intercambia la información con el usuario. Cada vista tiene incorporado un componente controlador donde pueden existir múltiples vistas del modelo. Son las encargadas de recibir los datos del modelo y mostrarlos al usuario Pueden dar el servicio de actualización, para que sea invocado por el controlador o por el modelo.

Controlador: Recibe las entradas, traducidas a solicitudes de servicio para el modelo. Es el encargado de responder a las solicitudes del usuario desde la interfaz, manejando los diferentes eventos a través de las funcionalidades necesarias y la información perteneciente al modelo.

Figura 2: Estructura del patrón MVC.

1.8 Lenguaje de modelado.

Cedrux, se está emprendiendo en condiciones muy favorables. Por lo que todo el proceso de su diseño y desarrollo tiene que hacerse de manera sencilla para que permitan a los usuarios y a todas las personas involucradas comprender toda su estructura y contenido que permiten graficar un sistema o parte de él.

Lenguaje Unificado de Modelado UML 2.0

Para modelar el sistema se hará uso del Lenguaje Unificado de Modelado (UML por sus siglas en inglés), puesto que es un estándar de modelo de sistemas para representar gráficamente los modelos y posibilitar que los diseños gráficos realizados se pudieran compartir fácilmente entre los involucrados en el proceso de desarrollo (Sommerville, 2002). Incluye aspectos conceptuales tales como las funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Larman, 2004).

1.9 Patrones de diseño y arquitectura.

Una vez definida la arquitectura, los patrones constituyen una parte indispensable en el desarrollo del software, estos hacen la producción más resistente al cambio, establecen parejas problema-solución y ayudan a especificar interfaces. Sirven para la búsqueda de soluciones exitosas a problemas comunes en

el desarrollo de software y otros ámbitos referentes al diseño de interfaces. Son indispensables para las distintas etapas del desarrollo desde el análisis hasta el diseño y desde la arquitectura hasta la implementación. (Jacobson, y otros, 2000).

Patrones GRASP

Patrones de Software para la Asignación General de Responsabilidades. En este caso las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento (Larman, 2004). Dentro de este grupo de patrones se destacan: Experto, Creador, Controlador, Bajo acoplamiento y Alta cohesión a los cuales se analizarán a continuación:

Experto**:** Ofrece una analogía con el mundo real. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe incurrir sobre la clase que conoce toda la información necesaria para ejecutar la tarea.

Creador**:** Guía la asignación de responsabilidades que tienen relación con la creación de objetos. Ayuda a identificar quién debe ser el responsable de la instancia o de la creación de nuevas clases u objetos.

Controlador**:** Se encarga de asignar la responsabilidad a una clase en el momento de manejar mensajes correspondientes a eventos en un sistema a clases específicas, facilitando la centralización de actividades.

Bajo acoplamiento**:** Se debe recordar durante las decisiones del diseño, es la meta principal que es preciso tener siempre presente. De modo que se conserve un mecanismo poco dependiente entre las clases y objetos, provocando la reducción del impacto de los cambios y que se incremente la reutilización.

Alta cohesión: Importante en todas las decisiones del diseño, es la meta principal que debe buscarse en todo momento. Indica que la información que recopila una clase debe ser coherente, de modo que todos sus métodos tengan un comportamiento bien definido.

Patrones GOF

Los patrones de diseño pueden tener propósito creacional, estructural o de comportamiento (García, 2005):

Patrones creacionales: Se encargan de las formas de crear instancias en los objetos. Su objetivo es: abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. Ejemplos de patrones de creación: Abstract Factory, Builder, Prototype y Singleton.

Patrones estructurales: Están relacionados y se aprecia cómo las clases y los objetos se combinan para formar nuevas estructuras más complejas y proporcionar nuevas funcionalidades. Ejemplos de patrones estructurales: Adapter, Bridge, Proxy, Decorator y Facade.

Patrones de comportamiento: Están relacionados con algoritmos y asignación de responsabilidades a los objetos. Describen no solamente patrones de objetos o clases, sino también patrones de comunicación entre ellos. Ejemplos de patrones de comportamiento: Chain of Responsibility, Mediator, Observer y Visitor.

1.11 Tecnologías y Herramientas de desarrollo.

Para el desarrollo del componente se utilizarán un conjunto de tecnologías y herramientas definidas por las políticas del proyecto, ellas son:

Tecnología Ajax

Técnica de desarrollo web para crear aplicaciones interactivas. Se ejecuta en el navegador del usuario manteniendo una comunicación directa con el servidor. De esta forma, es posible realizar cambios sobre la página sin necesidad de recargarla, aumentando la interactividad, velocidad y usabilidad (Garrett, 2005). Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

Herramienta CASE

Ingeniería de Software Asistida por Ordenador (CASE, por sus siglas en inglés) son un conjunto de herramientas que proporcionan ayuda automatizada facilitando el desarrollo del software. A través del apoyo en la realización de los diseños, la compilación automática, documentación o detección de errores reduciendo el esfuerzo, el costo y el tiempo (Scribd, 2010).

- ✓ **Visual Paradigm for UML 6.4.**

Usa un lenguaje estándar para todo el equipo de desarrollo, facilitando la comunicación. Permite la representación del ciclo de vida completo del desarrollo del software. Emplea UML, posibilita la representación de todo tipo de diagrama (Visual Paradigm, 2007). Es multiplataforma favoreciendo los principios de independencia tecnológica que se fomentan en el país.

Herramientas de programación

IDE: Un entorno de desarrollo integrado (IDE, por sus siglas en inglés), es un programa informático compuesto por un conjunto de herramientas de programación que puede dedicarse en exclusiva a emplear un sólo lenguaje de programación o varios (Ecured, 2010).

- ✓ **NetBeans 7.2**

NetBeans IDE es un producto libre y gratuito, sin restricciones de uso. Es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris, de código abierto, escrito completamente en Java (Netbeans 7.2, 2010). Es una plataforma de aplicaciones que permite a los desarrolladores crear

rápidamente aplicaciones, utilizando un número importante de módulos para extenderlo a otros lenguajes como PHP (Zend y Symfony), JavaScript, Ajax, Groovy y Grails, y C / C + +.

Servidor de Base de Datos

- ✓ **PostgreSQL 8.3.**

Es un sistema de gestión de bases de datos, objeto-relacional, destacado por ejecutar complejas consultas sobre vistas, subconsultas y joins de gran tamaño. PostgreSQL como Sistema Gestor de Base de Datos garantiza la integridad de la información porque es escalable, software libre y multiplataforma (Martínez, 2009). Tiene amplio soporte por una comunidad mundial. Presenta características orientadas a objetos, herencia entre tablas, se destaca por ser robusto y cumplir con estándares SQL.

Servidor WEB

- ✓ **Apache 2.0 o superior**

Es servidor web, altamente configurable de diseño modular, gratuito y multiplataforma. Es un software robusto y estable que proporciona la confianza de todos aquellos que lo utilizan (Ciberaula, 2010). Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor y la creación de ficheros de logs[4] a la medida de las necesidades del administrador, de este modo se puede tener un mayor control sobre lo que sucede el servidor.

Navegador web

- ✓ **Mozilla Firefox 3.6.24 o superior**

Es libre y de código abierto. Usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva y marcadores dinámicos. Además se pueden añadir funciones a través de complementos desarrollados por terceros (Mozilla Firefox, 2009). Es multiplataforma, permite la integración con el antivirus, realiza la navegación por pestañas y presenta compatibilidad para múltiples extensiones.

1.12 Lenguaje programación

Es un idioma artificial que permite a las computadoras interpretar las órdenes que se le dan, así como controlar su comportamiento. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Wilson, 1996). Los lenguajes de programación para el desarrollo de aplicaciones web, están divididos en dos grandes grupos: los lenguajes del lado del cliente y los lenguajes del lado del servidor.

Lenguajes del lado del cliente.

- ✓ **HTML**

[4]Logs**:** Registro de errores.

El Lenguaje de Marcas de Hipertexto (HTML por sus siglas en inglés), predomina en la construcción de páginas web y es muy usado para describir la estructura y el contenido en forma de texto (HTML, 2005). Este es un formato universal, flexible, rico y compacto Lo constituyen un conjunto de etiquetas y comandos que permiten dar formato a un archivo con la finalidad de crear un documento que pueda ser visualizado en forma de página web. Permite utilizar estilos en formato CSS en las páginas para una mayor facilidad en su modificación y los contenidos son fáciles de actualizar.

- ✓ **XML 1.0**

El Lenguaje de Etiquetado Extensible (XML, por sus siglas en inglés) es estricto y juega un papel fundamental en el intercambio de una gran variedad de datos pero a su vez es bastante sencillo (Jean Paoli, 2010). Es utilizado para estructurar, almacenar e intercambiar información; permite la lectura de datos a través de diferentes aplicaciones. A diferencia del HTML su función principal es describir datos y no mostrarlos.

- ✓ **Notación de Objetos de JavaScript (JSON)**

JSON (acrónimo de JavaScript Notación de Objetos por sus siglas en inglés) es un formato de intercambio de datos. Está constituido por una colección de pares de nombre/valor. En varios lenguajes esto se conoce como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. JSON se ha convertido en un estándar en el desarrollo de aplicaciones web. En los servicios web es empleado en vez de XML para permitir la integración de servicios en el navegador del usuario en vez de en el servidor (Colectivo de Json, 2011).

- ✓ **JavaScript 1.6**

Lenguaje interpretado que puede usarse sin necesidad de adquirir una licencia que permite a los desarrolladores añadir y crear interactividad en el desarrollo y diseño de sitios web (Territorio PC, 2010). Por otra parte permite validar datos y no requiere compilación. No requiere de compilación y los navegadores son los encargados de interpretar estos códigos. Es independiente de la plataforma.

Lenguaje del lado del servidor.

- ✓ **Lenguaje PHP 5.2 o superior**

Es dinámico, usado normalmente para la creación de páginas web. Es de código abierto y resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos, permite la conexión a algunas de ellas como son: Postgres, MySQL, Oracle, Microsoft SQL Server (The PHP Group, 2001). Dentro de sus ventajas está ser: rápido, seguro, multiplataforma y permite las técnicas de la POO. Contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de rendimiento.

1.11 Pruebas de software.

Son esencialmente un conjunto de actividades dentro del desarrollo del software, que involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados. El único instrumento adecuado para determinar el estado de la calidad de un producto de software, es el proceso de pruebas. En este se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requisitos (EcuRed: Enciclopedia cubana, 2009).

Todo el proceso de pruebas, sus objetivos, los métodos y las técnicas usadas se describen en el plan de prueba. Este proceso asegura la calidad del sistema con el fin principal de presentar la información sobre la calidad del producto a las personas responsables de esta.

Existen dos tipos de pruebas: las técnicas y las funcionales (EcuRed: Enciclopedia cubana, 2009). Las pruebas técnicas son responsabilidad de los ingenieros de software que han desarrollado el producto. Así como las pruebas funcionales donde el responsable es el técnico de pruebas, que dispone de los conocimientos y aptitudes necesarios para esta tarea tan importante y específica.

1.11.1 Métodos de prueba.

Pruebas de caja blanca

Se conocen también como Prueba de Caja Transparente o de Cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento. Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación (EcuRed: Enciclopedia cubana, 2009)

- ✓ Prueba del Camino Básico: permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos.
- ✓ Prueba de Condición: ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- ✓ Prueba de Flujo de Datos: se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ Prueba de Bucles: se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

Pruebas de caja negra

Se conocen también como Prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se

espera de un módulo, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa (S. Pressman, 2005).

En esencia permite encontrar:

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

Conclusiones del capítulo.

Luego de valorar y analizar varios sistemas de gestión contables tanto nacionales como extranjeros; así como varias herramientas y metodologías utilizadas para el desarrollo del software se puede arribar a las siguientes conclusiones:

- ✓ El análisis de los sistemas de gestión contable estudiados, fundamenta la necesidad de desarrollar un nuevo componente que se adapte a los requerimientos del sistema Cedrux y necesidades del país.
- ✓ El desarrollo del componente, teniendo en cuenta las características de Cedrux, debe estar basado en el modelo de desarrollo propuesto, con el marco de trabajo Sauxe y con las tecnologías, lenguajes y herramientas definidas.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN.

2.1 Introducción.

En este capítulo se describe la propuesta de solución del componente a desarrollar y el proceso que será informatizado. Se presentan los requisitos funcionales y no funcionales con los que debe cumplir el componente. Se hace referencia a las técnicas empleadas en la captura de los requisitos, así como los patrones utilizados en el diseño de la aplicación. Se obtendrá el modelo de datos, describiendo cada una de las tablas. Se realizará el diagrama de clases del diseño, donde se describe la función de cada una de las clases presentes y además se hará una breve descripción del marco de trabajo a utilizar, así como los estándares que se utilizarán en la implementación del componente.

Objeto de informatización.

Se define como objeto de informatización, el proceso que se realiza en una entidad económica para obtener los saldos contables actualizados. El presente trabajo de diploma, pretende desarrollar un componente capaz de informatizar el proceso relacionado con la reevaluación de cuentas en Cedrux. Este proporcionará información financiera actualizada para la toma de decisiones y tiene lugar cuando el usuario haya realizado el cálculo.

2.2 Propuesta de solución.

Debido a las dificultades existentes para reevaluar las cuentas en Cedrux, se toma la decisión de desarrollar un componente que permita solucionar tales inconvenientes. En la propuesta de la solución se centralizan un conjunto de operaciones que permiten mejorar el trabajo del sistema contable en general. La reevaluación de cuentas, es una operación que surge como consecuencia directa de la existencia de varias monedas distintas a la contable durante el proceso de registro de las operaciones económicas. Cuenta con una secuencia definida de acciones para su realización. Este componente será capaz de permitir el proceso de reevaluación de cuentas. Para ello el usuario deberá autenticarse en el sistema y en dependencia de su rol se le permitirá el acceso a las áreas y funcionalidades. La reevaluación de cuentas permitirá conocer los saldos de las cuentas, ya que brinda la posibilidad de calcular la reevaluación en una o varias cuentas, dependiendo directamente de la tasa de cambio, la cuenta para reflejar el efecto y la moneda a reevaluar; también tendrá otros atributos como son la denominación y la fecha. Por lo que el sistema también permitirá imprimir, modificar, consultar y eliminar una reevaluación de cuentas, así como realizar búsquedas de ellas para listarlas.

2.3 Modelado del sistema.

Primeramente se hace un levantamiento de requisitos, tanto funcionales como no funcionales, generándose el artefacto Descripción de Requisitos del Software para el componente de reevaluación de

cuentas, que constituye un elemento clave para el diseño y la posterior implementación. Se caracteriza fundamentalmente por presentar los requisitos de forma completa, definiéndose todas las responsabilidades del componente. Presenta una adecuada organización y documentación donde los términos, las tablas y los prototipos están correctamente descritos y referenciados. Cada requisito que comprende tiene una única interpretación evitando la ambigüedad en las definiciones y funcionalidades.

Según lo referido anteriormente, la descripción de requisitos puede ser tomada como artefacto de entrada al diseño para facilitar la comprensión del desarrollo del componente. A continuación se presentan los requisitos funcionales y no funcionales del proceso de reevaluación de cuentas.

2.3.1 Requisitos funcionales y no funcionales.

Para realizar el proceso de captura de los requisitos, se hizo necesaria la utilización de algunas técnicas que sirvieron para verificar la veracidad de los objetivos propuestos para el desarrollo de este componente. Dentro de las técnicas utilizadas se encuentran: las **entrevistas**, aplicada a los funcionales y clientes para los cuales va a ser desarrollado el componente y las **tormentas de ideas** que posibilitaron los debates donde cada cliente explicó de forma clara sus necesidades.

Listado de los requisitos funcionales

RF-1 Calcular reevaluación de cuentas.
RF-2 Consultar reevaluación de cuentas.
RF-3 Listar cuentas reevaluadas.
RF-4 Imprimir reevaluación de cuentas.
RF-5 Buscar reevaluación de cuentas.
RF-6 Modificar cuenta reevaluada.

Para el desarrollo de la solución final del componente se tuvo en cuenta agregar los siguientes requisitos funcionales:

RF-7 Realizar búsqueda avanzada de cuenta reevaluada.
RF-8 Eliminar cuentas reevaluadas.

Aunque no fueron solicitados por el cliente, se consideraron importantes ya que el requisito de Búsqueda avanzada permitirá buscar de forma más detallada una cuenta reevaluada, a partir de los atributos: denominación, fecha, moneda a reevaluar y tasa de cambio. Por otra parte el requisito de Eliminar cuentas reevaluadas, ejecuta la eliminación de cuentas reevaluadas que no han generado asientos contables.

A continuación se muestra la descripción textual del requisito Calcular reevaluación de cuentas, con su prototipo de interfaz de usuario funcional, ver (Tabla 2) conjuntamente con (Figura 3). Además como ejemplo ilustrativo se encuentra en ¡Error! No se encuentra el origen de la referencia.**1,** la descripción del

requisito Búsqueda avanzada de cuenta reevaluada. Las descripciones de los requisitos funcionales restantes se encuentran en el expediente de proyecto del CEIGE.

Precondiciones	El usuario debe haberse autenticado en el sistema. Deben haberse registrado las cuentas en la Contabilidad General.
Flujo de eventos	
Flujo básico	
1	Seleccionar opción Calcular de la interfaz Principal.
2	Se seleccionan la(s) cuenta(s) que deben cambiarse. Identificación selectiva y/o masiva.
3	Seleccionar la moneda a la cual se realizará la reevaluación. Ejemplo: CUC, USD.
4	A partir de esta moneda se cargan los valores de la Tasa de Cambio que se va a utilizar como base para la reevaluación. Ejemplo: 24.000, 25.000.
5	Seleccionar la cuenta que se va a utilizar para reflejar el efecto de la reevaluación. Ejemplo: Ganancia por partida no monetarias y Pérdida por partidas no monetarias.
6	El sistema valida (ver validación 1) que se hayan seleccionado cada uno de los datos pedidos.
7	Si los datos seleccionados son correctos el sistema valida (ver validación 2) que se puede realizar la reevaluación.
8	El sistema procede a reevaluar presionando la opción Aceptar.
9	Se indica el comienzo de la reevaluación.
10	El sistema procede a registrar la reevaluación.
11	El sistema confirma la reevaluación.
12	Concluye el requisito.
Pos-condiciones	
1	Se registra una nueva reevaluación de cuenta en el sistema.
Flujos alternativos	
Flujo alternativo (2.a, 3.a, 4.a, 5.a) Información incompleta.	
1	El sistema notifica a través de un mensaje de que se deben seleccionar todos los elementos.
2	Volver al paso 2, 3, 4, 5 del Flujo Básico.
Pos-condiciones	
1	N/A

Flujo alternativo 8.a El subsistema de Contabilidad no ha cerrado su período de apertura.		
1	El sistema notifica que no se pueden hacer reevaluación mientras que contabilidad este en su período de apertura.	
2	Volver al paso 1 del Flujo Básico.	
Pos-condiciones		
1	N/A	
Flujo alternativo *.a El usuario cancela la acción		
1	Concluye el requisito.	
Pos-condiciones		
1	No se registran los datos.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual CIG-ERP-N-MUL-i2201.	
2	Si el subsistema de contabilidad no ha cerrado su apertura no se puede reevaluar.	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Cuenta reevaluada**	Visibles en la interfaz: Cuentas a reevaluar Cuentas para reflejar el efecto Denominación Tasa de Cambio Moneda Utilizados internamente: Fecha en la que se crea la reevaluación
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 2: Descripción del requisito funcional Calcular reevaluación de cuentas.

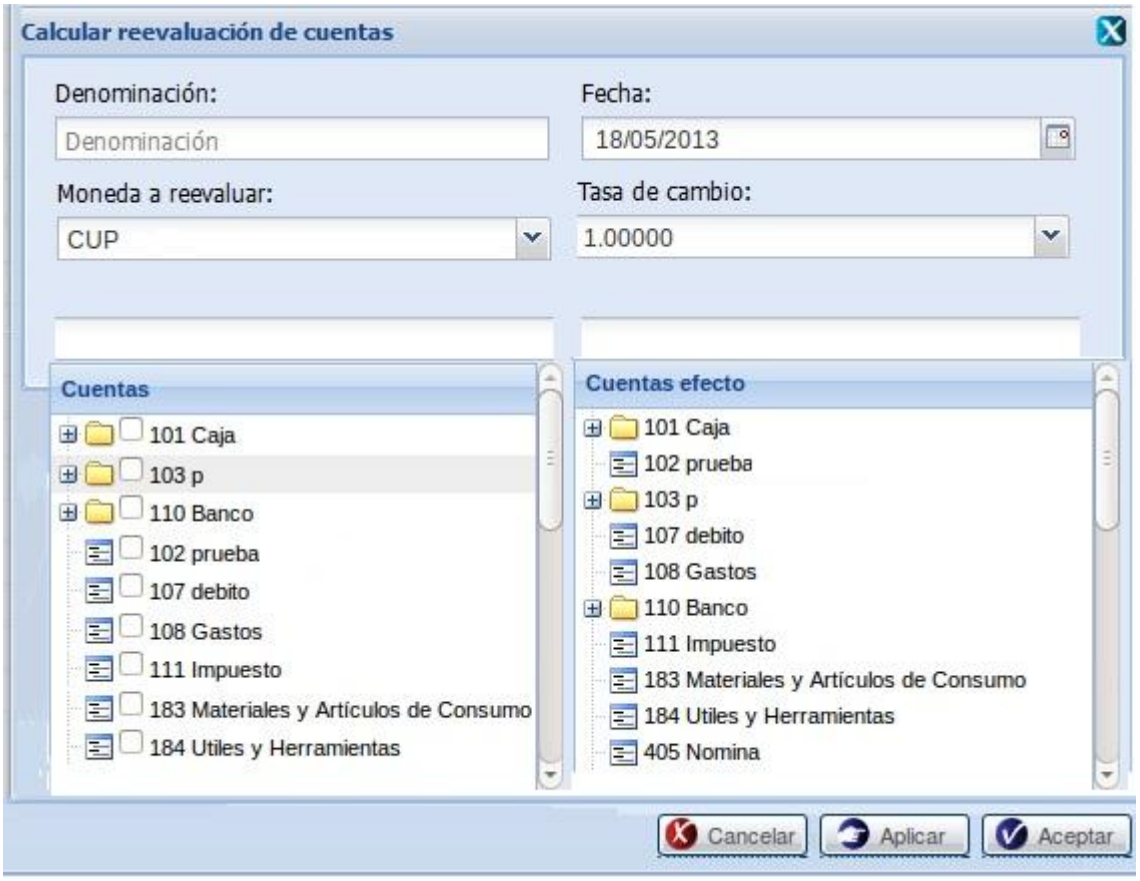

Figura 3: Prototipo elemental de interfaz gráfica de usuario Calcular reevaluación de cuentas.

Listado de los requisitos no funcionales

- ✓ **Funcionalidad**

El sistema debe poseer la capacidad para proporcionar efectos o resultados correctos con un grado de exactitud necesario y la interacción equitativa con uno o más sistemas específicos.

RNF-1: Las respuestas del sistema ante búsquedas corresponderán en un 100% a lo solicitado en los criterios.

RNF-2: El sistema será capaz de leer datos provenientes del sistema: *Estructura y Composición*.

RNF-3: El sistema producirá datos para los sistemas *Finanzas* e *Inventario, Contabilidad, Costos y Procesos*.

- ✓ **Seguridad**

El sistema debe poseer la capacidad para proteger la información y los datos; donde se garantizará el acceso de personas o sistemas y los que no tengan una previa autorización no podrán leer ni modificar.

RNF-4: El sistema concederá el acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.

RNF-5: El sistema registrará las trazas de operaciones realizadas por cada usuario en todo momento.

- ✓ **Confiabilidad**

El sistema debe poseer la capacidad de proteger la información ante el acceso no autorizado y la divulgación indebida.

RNF-6: La información manejada por el sistema debe estar protegida ante el acceso no autorizado y la divulgación indebida.

RNF-7: El sistema no permitirá la entrada de datos incorrectos.

RNF-8: El sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario.

- ✓ **Usabilidad**

El sistema debe poseer la capacidad para permitirle al usuario entender si el software es idóneo, operarlo y controlarlo para las tareas y condiciones de uso particulares.

RNF-9: Todos los mensajes de error del sistema deberán incluir una descripción textual del error.

RNF-10: Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función.

RNF-11: El sistema expondrá el menú general en todo momento para que pueda ser utilizado por el usuario en cualquier momento.

- ✓ **Mantenibilidad**

El sistema debe poseer la capacidad para permitir la aplicación de una modificación especificada.

RNF-12: La modificación interna del componente del sistema no afectará a las funcionalidades o componentes que dependan de esta.

- ✓ **Portabilidad**

El sistema debe poseer la capacidad del producto de software de ser adaptado a los ambientes especificados y de coexistir con otro software independiente en un ambiente común y compartir los recursos comunes.

RNF-13: El sistema podrá ser visualizado en todos los navegadores que estén soportados por el framework javascript ext en su versión 2.2.

RNF-14: El sistema podrá ser instalado en el ambiente especificado en los requisitos tecnológicos para servidores.

- ✓ **Hardware**

Para el cliente:

RNF-15: Requerimientos mínimos: procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.

RNF-16: Tarjeta de red para establecer conexión.

Para el servidor:

RNF-17: Requerimientos mínimos: procesador Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.

RNF-18: Tarjeta de red para establecer la conexión.

RNF-19: Al menos 40Gb de espacio libre en disco duro.

- ✓ **Software**

Para el cliente:

RNF-20: sistema operativo con interfaz gráfica y soporte para red.

RNF-21: las interfaces deben ser compatibles con Mozilla Firefox 3.0 o superior.

Para el servidor:

RNF-22: Utilizará un servidor web Apache 2.0 o superior.

RNF-23: Utilizará un gestor de base de datos PostgreSQL 8.3.

RNF-24: Se requiere de un navegador (Internet Explorer versión 6 o Firefox versión 2.0.0.15).

RNF-25: Se garantizará versiones de Windows 2000 o superior, así como Linux y sus correspondientes distribuciones.

2.3.2 Aplicación de las técnicas de validación de requisitos.

Las técnicas que se utilizaron para la validación de los requisitos identificados fueron las siguientes: la **Revisión Técnica Formal,** una vez terminada la descripción de cada uno de los requisitos se realizaron revisiones en cada una de estas descripciones. Ante los errores detectados se volvieron a analizar para una nueva revisión. Con esta técnica se pudo validar que la interpretación de cada una de las descripciones no fuera ambigua, ni tuviese omisiones o errores y además que cada uno de los requisitos cumplía con lo que necesitaba el cliente. Los **prototipos de interfaz** es otra de las técnicas utilizadas, ya que se hicieron simulaciones del posible producto. Cada uno de los prototipos le permitió a los especialistas tener una idea de cada una de las interfaces gráficas del componente. Estos se realizaron de forma no funcional con la herramienta Visual Paradimg for UML para lograr una mayor aprobación por parte del cliente.

2.4 Modelo Conceptual.

El Modelo Conceptual, ver (Figura 4), explica los principales conceptos en el dominio del problema. Puede mostrar conceptos, asociaciones entre conceptos y atributos de conceptos (Sommerville, 2002).

La finalidad de esta actividad es la de estipular una especificación del dominio del problema y los requisitos desde la perspectiva de la clasificación por objetos y desde el punto de vista de entender los términos empleados en el dominio.

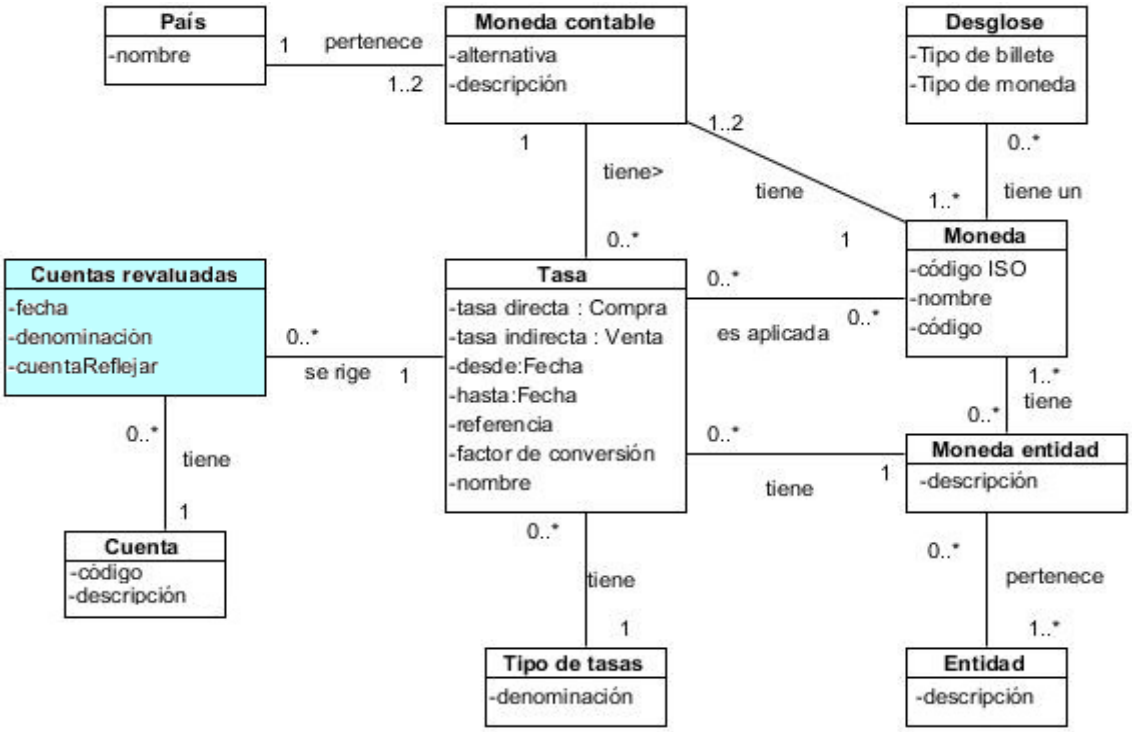

Figura 4: Modelo conceptual del proceso reevaluación de cuentas.

2.5 Diseño de la solución en términos de componentes.

El componente de reevaluación de cuentas perteneciente a Cedrux, es el encargado de informatizar todo el proceso de cálculo y de actualización de los saldos contables que se aplican a una determinada entidad que lo utilice. Un modelo de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan (msdn, 2012).

Estas relaciones, ver (Figura 5), fueron definidas para dar solución arquitectónica al componente entre los cuales se encuentran: Comprobante, Nomenclador de cuenta, Cierre, Moneda, Tasa y el componente a desarrollar Reevaluación de cuentas.

A continuación, una explicación más puntualizada de la funcionalidad de cada uno de estos componentes a partir de los servicios que brindan y reciben:

Comprobante: Es el encargado de registrar los asientos contables a través de pases que se le realizan a las cuentas, ya sea desde el subsistema Contabilidad o desde el resto de los subsistemas que generan contabilizaciones.

Nomenclador de cuenta: Es el componente encargado de definir las cuentas que se utilizarán en la entidad. Está definido a partir de un contenido económico y tiene una estructura arbórea.

Cierre: Es el componente encargado de realizar el cierre de período y de ejercicio contable de la entidad. Define validaciones que permiten verificar los procesos contables acaecidos en la entidad en un período determinado.

Moneda: Es el componente encargado de definir el nomenclador de cuentas de la entidad.

Tasa: Es el componente encargado de definir la tasa de cambio de la moneda a partir de un factor de conversión, una tasa directa o indirecta.

Reevaluación de cuentas: Es el componente a partir del cual se realizará el cálculo pertinente para llevar a cabo el proceso de reevaluación de cuentas y todos los demás procesos que incluye, para tener un total control y dominio de los saldos contables.

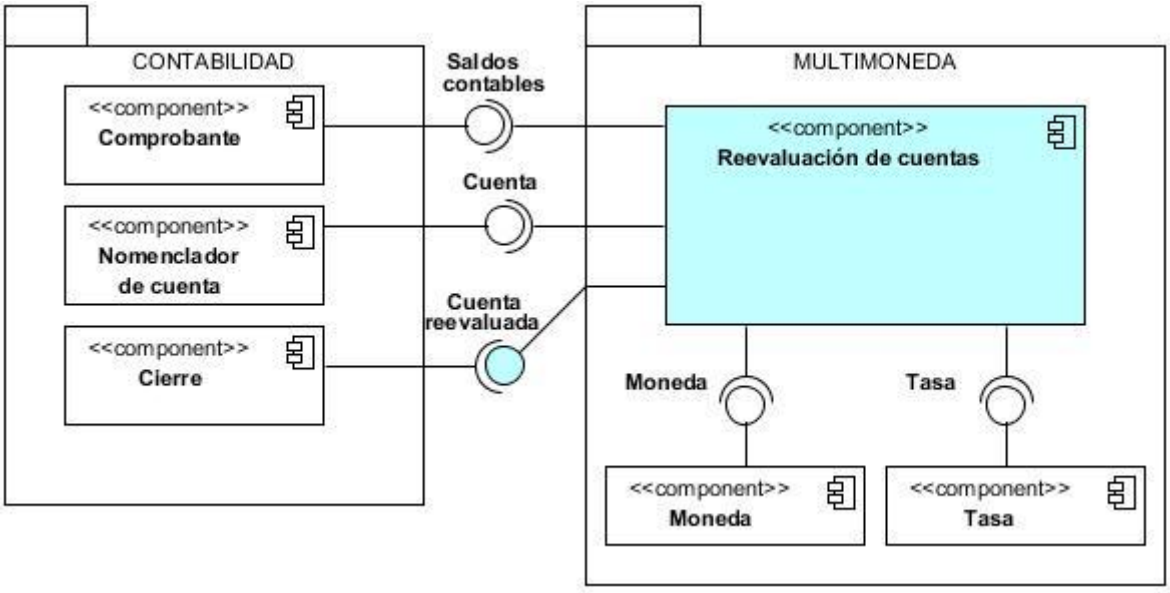

Figura 5: Modelo de componentes del proceso Reevaluación de cuentas.

2.6 Modelo de datos del sistema.

Un modelo de datos es un lenguaje utilizado para la descripción de una base de datos. Por lo general permite describir las estructuras de datos de la base, su tipo, descripción y la forma en que se relacionan, restricciones de integridad y las operaciones de manipulación de los datos (Modelo de Datos, 2012).

El modelo de datos propuesto en la solución está estrechamente relacionado con los mod_datosmaestros y con los mod_contabilidad, ver (Figura 6), en la que aparece su representación. Se creó una nueva tabla denominada mod_datosmaestros.dat_cuentasreevaluadas, teniendo en cuenta el componente a implementar y la persistencia entre los conceptos relacionados. Para el requisito Calcular reevaluación de cuentas y todos los que dependen de él, se crea la tabla mod_datosmaestros.dat_reevaluacion, donde se almacenan los datos correspondientes a cada reevaluación de cuentas. Esta tabla es asociada a la de mod_datosmaestros.dat_cuentasreevaluadas, encargada de almacenar los datos correspondientes a las

cuentas reevaluadas. Se relaciona directamente con las tablas mod_datosmaestros.dat_reevaluacion, mod_datosmaestros.dat_tasa, mod_datosmaestros.dat_nommoneda_entidad y mod_contabilidad.nom_cuenta.

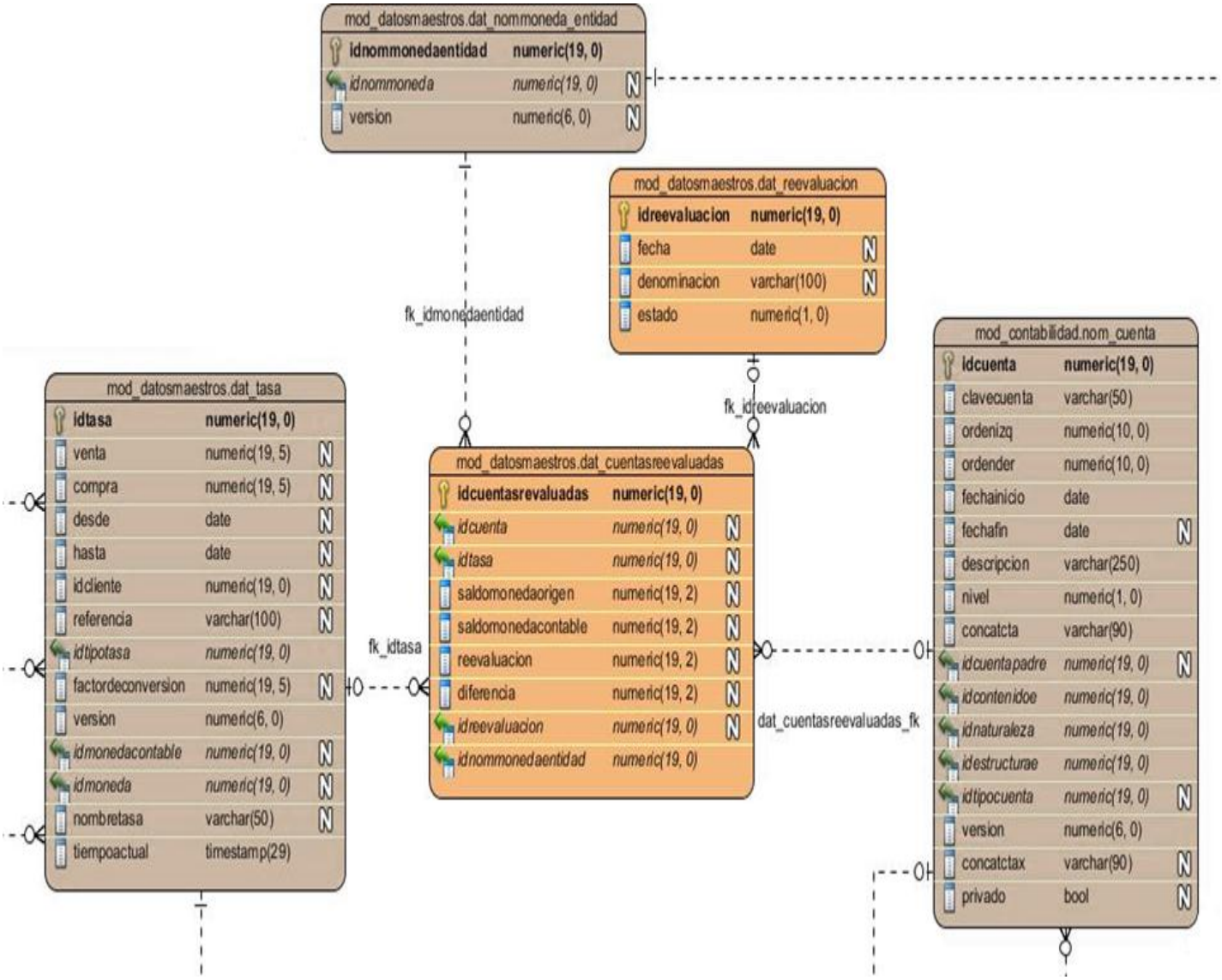

Figura 6: Modelo de datos del sistema.

Descripción de las tablas generadas.

A continuación se muestran las descripciones de las tablas generadas por el modelo de datos del sistema.

Nombre: “mod_datosmaestros.dat_reevaluacion”		
Descripción: Almacena los datos correspondientes a cada reevaluación de cuentas.		
Atributo	**Tipo**	**Descripción**
idreevaluacion	numeric	Identificador de la tabla.

fecha	date	Fecha de creación de la reevaluación.
denominacion	varchar	Nombre de la reevaluación de cuentas.
estado	numeric	Es reevaluado.

Tabla 3: Diccionario de datos tabla mod_datosmaestros.dat_reevaluacion.

Nombre: "mod_datosmaestros.dat_cuentasreevaluadas"		
Descripción: Almacena los datos correspondientes a las cuentas reevaluadas.		
Atributo	**Tipo**	**Descripción**
idcuentasreevaluadas	numeric	Identificador de la tabla.
idcuenta	numeric	Identificador de la tabla "mod_contabilidad.nom_cuenta" (Llave foránea).
idtasa	numeric	Identificador de la tabla "mod_datosmaestros.nom_tasa" (Llave foránea).
saldomonedaorigen	numeric	Moneda definida para realizar diferentes transacciones.
saldomonedacontable	numeric	Moneda utilizada para el registro contable de las operaciones, según la legislación vigente.
reevaluacion	numeric	Registra los datos de cada reevaluación que se efectúe.
diferencia	numeric	Registra la diferencia entre el saldo actual en función de la tasa vigente.
idreevaluacion	numeric	Identificador de la tabla "mod_datosmaestros.nom_reevaluacion" (Llave foránea).
idnommonedaentidad	numeric	Identificador de la tabla "idnommonedaentidad" (Llave foránea).

Tabla 4: Diccionario de datos tabla nom_datosmaestros.dat_cuentasreevaluadas.

2.7 Diseño de clases.

Los diagramas de clases según la clasificación UML, son diagramas de estructura estática donde la representación de los requisitos se lleva a cabo a través de las clases del sistema y sus interrelaciones (Universidad Central de Chile, 2011). Su principal utilidad radica en mostrar a través de sus atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación (PHP, en este caso).

A continuación se muestra el diagrama de clases del diseño del proceso que será informatizado con el desarrollo del componente, ver (Figura 7) y su respectiva descripción ver (Tabla 5).

Diagrama de clases del diseño del proceso Reevaluación de cuentas.

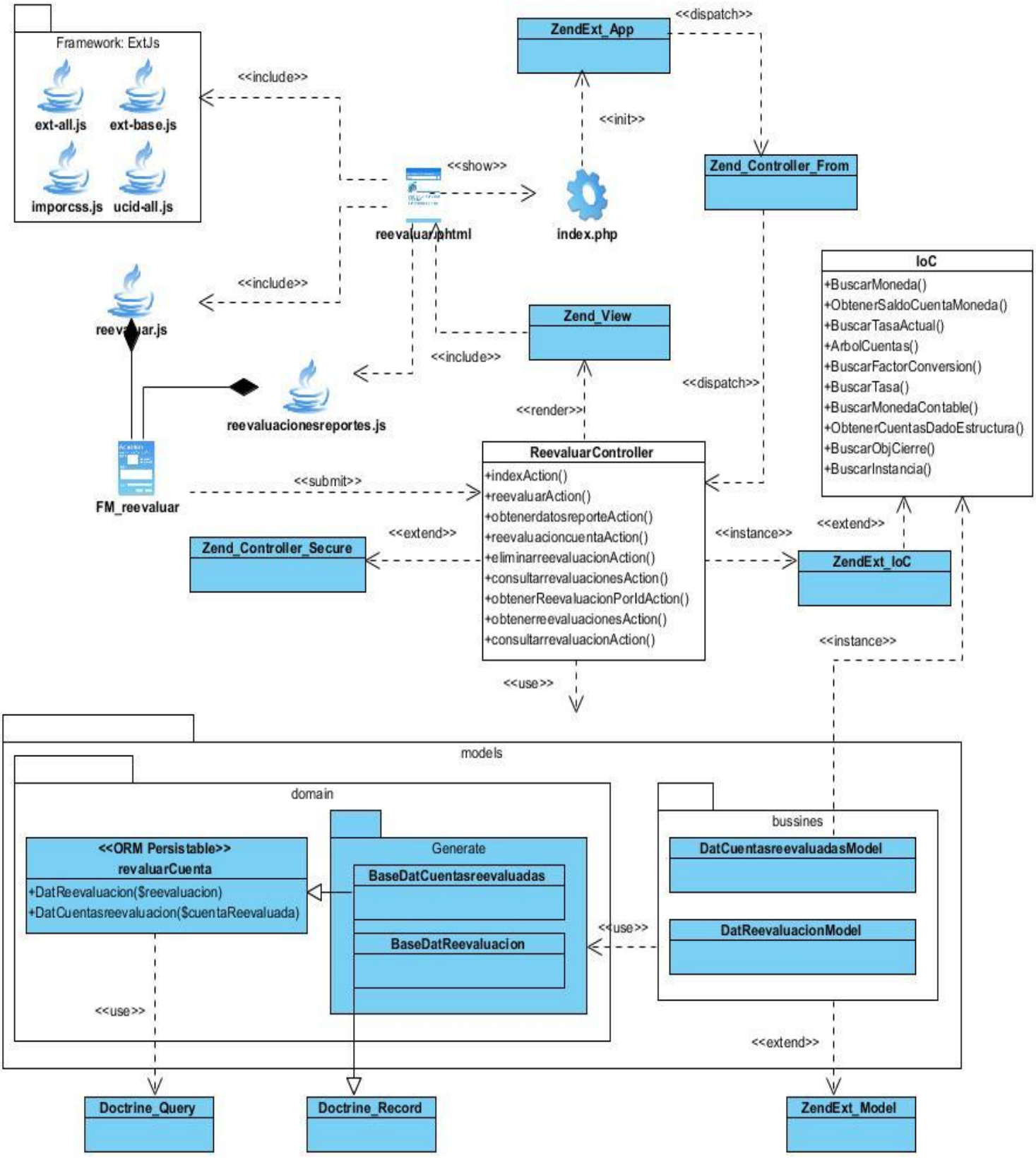

Figura 7: Diagrama de clases de diseño del proceso Reevaluación de cuentas.

Descripción del diseño de clases del proceso Reevaluación de cuentas.

Clases	Descripción
reevaluar.phtml	Encargada de visualizar, a través de los js que debe incluir, la información necesaria del proceso reevaluación de cuentas.
reevaluar.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería Extjs los componentes necesarios a través de los cuales se puedan realizar las operaciones que genera el cálculo de la reevaluación de cuentas. Responsable de enviar y recibir los datos de la controladora utilizando tecnología Ajax.
reevaluacionesreportes.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería Extjs los reportes de la funcionalidad imprimir. Responsable de enviar y recibir los datos de la controladora utilizando tecnología Ajax.
Fm_reevaluar	Encargado de recoger los datos entrados por el usuario en las funcionalidades de calcular y buscar.
Index.php	Encargado de redireccionar el sistema, comienza con el levantamiento del sistema y va al componente Portal.
ReevaluarController	Clase controladora responsable de calcular la reevaluación de cuentas, donde a partir de ella se deben ejecutar las diferentes funcionalidades, según las peticiones del usuario.
IoC	Permite fomentar la reutilización de componentes que tienen dependencias de servicios o componentes.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.

Tabla 5: Descripción del diseño del proceso Reevaluación de cuentas.

2.8 Patrones de diseño y arquitectura empleados en la solución.

Cedrux está implementado bajo el marco de trabajo de Sauxe. La arquitectura y el diseño a través del correcto uso de patrones de diseño en la generación de los artefactos necesarios para el desarrollo, facilitaron crear una entrada apropiada como punto de partida a las actividades de implementación, con el

fin de lograr una mayor calidad del producto y la satisfacción del cliente. La selección de los patrones de diseño y arquitectura estuvo basada en el marco de trabajo y se emplearon como:

Patrones GRASP

Experto: Este patrón se evidencia en la clase que cuenta con la información necesaria para cumplir las funcionalidades que deben realizar a partir de la información manejada dentro del componente, por ejemplo las clases del modelo. Es evidenciado en la clase DatRevaluacionModel, la cual es la encargada de insertar, actualizar, buscar, eliminar, obtener los datos de las reevaluaciones para generar los reportes y así obtener toda la lógica para cada una de estas funcionalidades.

Creador: Este patrón es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

Controlador: Este patrón se evidencia en la clase controladora: ReevaluarController ya que tendrá a cargo la responsabilidad de manejar todos los eventos dentro del componente.

Alta cohesión: Sauxe presenta entre sus principales características la organización del trabajo en cuanto a estructura y responsabilidades bien definidas, esto permite que se trabaje con las clases con una alta cohesión. Un ejemplo de esto es la clase (class ReevaluarController extends ZendExt_Controller_Secure), la cual delega funciones específicas a otras clases, encargándose solo de definir las acciones a realizar.

Patrones GOF

Proxy: En la implementación de la solución se tuvo en cuenta este patrón **Proxy** el cual proporciona un punto de acceso común para un conjunto de peticiones diversas de varios subsistemas.

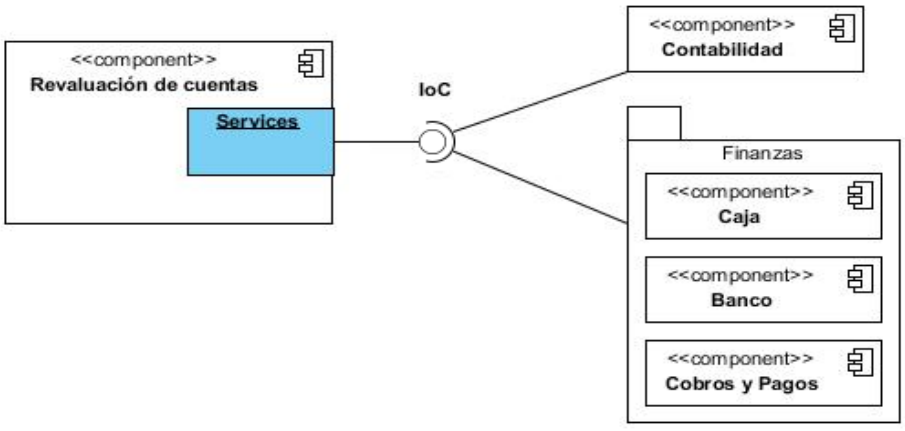

Figura 8: Aplicación del patrón Proxy.

Singleton: En la implementación de la solución se tuvo en cuenta este patrón porque ExtJs hace uso de la tecnología Ajax y la tecnología Ajax crea una sola instancia de cada clase creada en ExtJs.

2.9 Implementación del componente

Estructura del Marco de trabajo Sauxe.

A continuación se presenta la estructura del marco de trabajo Sauxe, mostrando cómo se organizará la implementación del componente a desarrollar, de manera que facilite la organización y claridad durante el desarrollo, además se explica cómo va a ser garantizada la seguridad del componente reevaluación de cuentas.

Contenido dentro de la carpeta apps:

En la carpeta denominada apps se almacenan los controladores y el modelo de cada una de las funcionalidades a desarrollar.

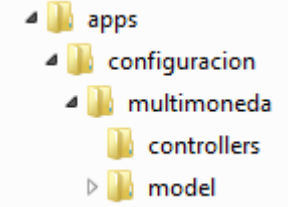

Figura 9: Contenido de la carpeta de la aplicación app.

- ✓ **comun:** contiene la carpeta recursos y dentro de esta una denominada xml. Esta última incluye a los ficheros: ioc, validator, exception que serán explicados a continuación.
- ✓ **ioc:** es donde se publican los servicios que brinda cada uno de los componentes en cuanto a nombre de clases, funciones y tipo de resultado.
- ✓ **validator:** chequea las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice.
- ✓ **exception:** se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción en el servidor.

El componente **reevaluación de cuentas** va a contener un conjunto de carpetas que serán especificadas a continuación:

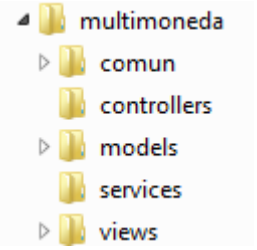

Figura 10: Contenido del componente reevaluación de cuentas dentro de la carpeta app.

En la carpeta **controllers** se encontrarán las clases controladoras encargadas de gestionar las funcionalidades del componente.

La carpeta **models** se estructura de la siguiente forma:

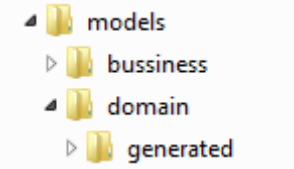

Figura 11: Contenido de la carpeta models.

Esta carpeta contiene dos carpetas las cuales a su vez agrupan clases y otras carpetas, que serán explicadas a continuación:

- ✓ **bussines:** contiene las clases necesarias para acceder a los datos que persisten en la base de datos.
- ✓ **domain:** contiene las clases generadas por el ORM Doctrine Generator a partir de cada una de las tablas existentes en la base de datos.

Cada una de estas clases mencionadas anteriormente heredará de una clase generada igualmente por el Doctrine Generator las cuales se ubican en otra carpeta dentro de esta llamado generated.

La carpeta **services** contiene todas las clases y funcionalidades de los servicios que va a brindar el sistema. Para solicitar un servicio que está en otro dominio, se accede a través de la carpeta services, esta analizará la solicitud e irá a la clase que tiene dicha funcionalidad y la devolverá a services, que a su vez se la entregará al dominio solicitante.

La carpeta **views** contiene las carpetas idioma y scripts, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente.

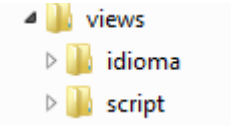

Figura 12: Contenido de la carpeta views.

Contenido dentro de la carpeta web:

Al mismo nivel de la carpeta app mencionada inicialmente debe existir además una carpeta que contenga las vistas de los subsistemas y componentes. Dicha carpeta se denomina web.

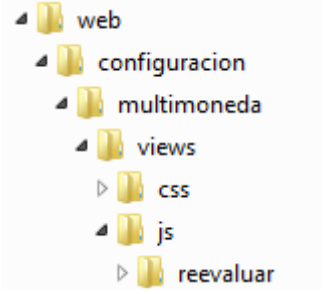

Figura 13: Carpeta de diseño correspondiente al componente reevaluación de cuentas.

Index: este fichero contiene la dirección del archivo de configuración y además inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento. Su código permanece igual para todos los componentes.

Figura 14: Contenido del componente reevaluación de cuentas dentro de la carpeta web.

La carpeta **views** contendrá los css y los js que se explican a continuación.

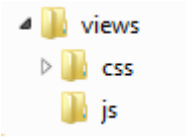

Figura 15: Contenido dentro de la carpeta views.

- ✓ **css:** incluye las clases necesarias para estructurar gráficamente el componente, separando de esta forma el estilo del contenido.
- ✓ **js:** comprenderá las clases Java Script necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios. Está compuesta por carpetas con los nombres de las funcionalidades del componente.

Seguridad del sistema.

La seguridad del sistema va a ser garantizada a través del Sistema Integral de Seguridad (ACAXIA), que tiene como objetivo principal garantizar la administración de la seguridad en sistemas de gestión. ACAXIA brinda sus servicios a todos los sistemas que se suscriban a él. Para ello se gestionan las conexiones a la base de datos, las funcionalidades asociadas y las acciones que realizan cada una de ellas. Una vez que se registra esta información se procede a la creación de roles a los cuales se les dan los permisos dentro de cada sistema. Luego se crean los usuarios con el perfil seleccionado y se le asignan uno o muchos roles en una o muchas entidades respectivamente.

2.10 Estándares de código.

Un estándar de código se basa en la estructura y apariencia física de un programa, con el fin de facilitar la lectura, comprensión, mantenimiento del código y reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa (Revisiones de código y estándares de codificación, 2013). Este no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino también tiene que ver con el orden y la legibilidad del código, aspecto decisivo a la hora de darle mantenimiento y mejorar las funcionalidades de un software. Teniendo en cuenta lo expresado anteriormente, se definen tres partes principales dentro de un estándar de programación:

- ✓ Convención de nomenclatura: es la forma de nombrar las variables, funciones, métodos.
- ✓ Convenciones de legibilidad de código: es la forma de organizar el código.
- ✓ Convenciones de documentación: es la manera de establecer los comentarios, la ayuda.

Nomenclatura de las clases.

Los nombres de las clases comienzan con la primera letra en minúscula y el resto en minúscula también, cuando sea un nombre compuesto se empleará notación **PascalCasing**, la cual define que los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula y con solo leerlo se reconoce el propósito de la misma. Ejemplo: ReevaluarController. En este caso el nombre de clase está compuesto por dos palabras iniciadas cada una con letra mayúscula, después de la primera.

- ✓ Nomenclatura según el tipo de clase.

Clases controladoras: Las clases controladoras después del nombre llevan la palabra:"Controller".

Ejemplo: ReevaluarController.

Clases de los modelos:

Business (Negocio): Las clases que se encuentran dentro de Business posteriormente del nombre llevan la palabra: "Model".

Ejemplo: DatReevaluacionModel.

Domain (Dominio): Las clases que se encuentran dentro de Domain el nombre que asumen es el de la tabla en la base de datos.

Ejemplo: DatReevaluacion.

Generated (Dominio base): Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: "base" y seguido el nombre de la tabla en la base de datos.

Ejemplo: BaseDatReevaluacion.

Nomenclatura de las funcionalidades y atributos:

El nombre a emplear para las funcionalidades y los atributos se escribe con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará también la notación **CamelCasing** que es similar a la antes mencionada **PascalCasing** con la excepción de la primera letra.

Ejemplo de un método: obtenerDatosReporteAction(). El nombre de método está compuesto por cuatro palabras, la primera en minúsculas y las restantes iniciadas con letra mayúscula.

Las principales funcionalidades de las clases controladoras se les escribe el nombre y seguida la palabra:"Action"

Ejemplo de método: insertarreevaluacionAction().

Ejemplo de atributo: idcuentasreevaluadas. El nombre del atributo está compuesto por tres palabras todas con letras minúscula.

Nomenclatura de los comentarios:

Los comentarios deben ser lo suficientemente claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. En la realización de un software se deben realizar comentarios en las funciones complejas para lograr una mejor comprensión del código y todo lo que se haga dentro del desarrollo. Con el objetivo de lograr un código más legible y reutilizable y así se pueda aumentar su mantenibilidad a lo largo del tiempo.

2.11 Descripción de las clases y funcionalidades del componente.

Clase Controladora

Las clases de controladoras coordinan las actividades de los objetos que implementan las funcionalidades, definen el flujo de control y las transacciones entre los objetos.

Nombre: reevaluarCuentaController	
Tipo de clase: controladora	
Para cada responsabilidad:	
Nombre:	**Descripción**:
reevaluarAction()	Permite levantar la ventana principal.
obtenerreevaluacionesAction()	Permite cargar todas las cuentas reevaluadas que posee el sistema.
obtenerdatosreporteAction()	Permite cargar las cuentas con sus respectivos datos para mostrar el reporte cuando se quiera imprimir.
reevaluacioncuentaAction()	Permite realizar el cálculo para reevaluar las cuentas.
eliminarreevaluacionAction()	Permite eliminar las reevaluaciones de cuentas existentes en el sistema y que no sean necesarias.
consultarrevaluacionesAction()	Permite consultar todas las reevaluaciones de cuentas.
obtenerReevaluacionPorIdAction()	Permite obtener todos los datos que tiene la reevaluación de cuentas con el identificador seleccionado.
obtenerreevaluacionesAction()	Permite obtener todas las reevaluaciones de cuentas cuando se seleccione un criterio de búsqueda.

Tabla 6: Descripción de la clase controladora: ReevaluarController.

Clase Modelo

Nombre: DatCuentasreevaluadasModel

Tipo de clase: modelo	
Para cada responsabilidad:	
Nombre:	**Descripción**:
Insertar($DatCuentasreevaluadas)	Permite insertar una reevaluación de cuentas en la base de datos del sistema.
Actualizar($DatCuentasreevaluadas)	Permite cargar todas las cuentas reevaluadas que posee el sistema.
EliminarDatCuentasreevaluadas($Dat Cuentasreevaluadas)	Permite eliminar todas las cuentas reevaluadas que posee el sistema y que no se vayan a utilizar.
BuscarCuentaId($idcuenta)	Permite buscar las cuentas por su identificador.
BuscarCuentasPorIdReeval	Permite buscar las cuentas que han sido reevaluadas por su identificador.

Tabla 7: Descripción de la clase modelo: DatCuentasreevaluadasModel.

Conclusiones del capítulo.

Al finalizar este capítulo vale destacar que:

- ✓ La definición y descripción de los requisitos del componente permitió cumplir con todas las funcionalidades requeridas para el proceso de reevaluación de cuentas.
- ✓ La utilización de las técnicas definidas para la validación de requisitos, demostró que estos presentan las condiciones requeridas y están en correspondencia con las necesidades del cliente
- ✓ La utilización de los patrones de diseño y arquitectura durante el desarrollo del componente, proporcionaron una mayor calidad del producto.
- ✓ El diseño del modelo de datos del componente permitió conocer las relaciones existentes entre las diferentes tablas de la base de datos.
- ✓ La descripción del diagrama de clases de diseño permitió conocer la estructura y las relaciones entre las clases que se manejan en el componente.
- ✓ La utilización de los estándares de implementación a utilizar, facilitaron el entendimiento del código por los programadores y el futuro mantenimiento del componente.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

3.1 Introducción

El presente capítulo mostrará el diagrama de despliegue para tener una visión más clara sobre el componente. Se realizará la validación del diseño a través de métricas y la validación de la implementación, donde en esta se aplicarán pruebas de caja negra y caja blanca, las cuales permitirán probar las funcionalidades del componente y los resultados obtenidos de estas.

3.2 Diagrama de despliegue

El diagrama de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. Esto muestra la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se conectan en esos nodos. Permite el mapeo de procesos, asegurando la distribución del comportamiento a través de aquellos nodos que son representados.

En el siguiente diagrama de despliegue ver (Figura 16) se representa la distribución física de un pequeño escenario del sistema en términos de cómo se distribuirán la funcionalidades entre los nodos, cada nodo representa un recurso de cómputo, siendo estos procesadores o dispositivos hardware que se necesitarán para el despliegue del sistema.

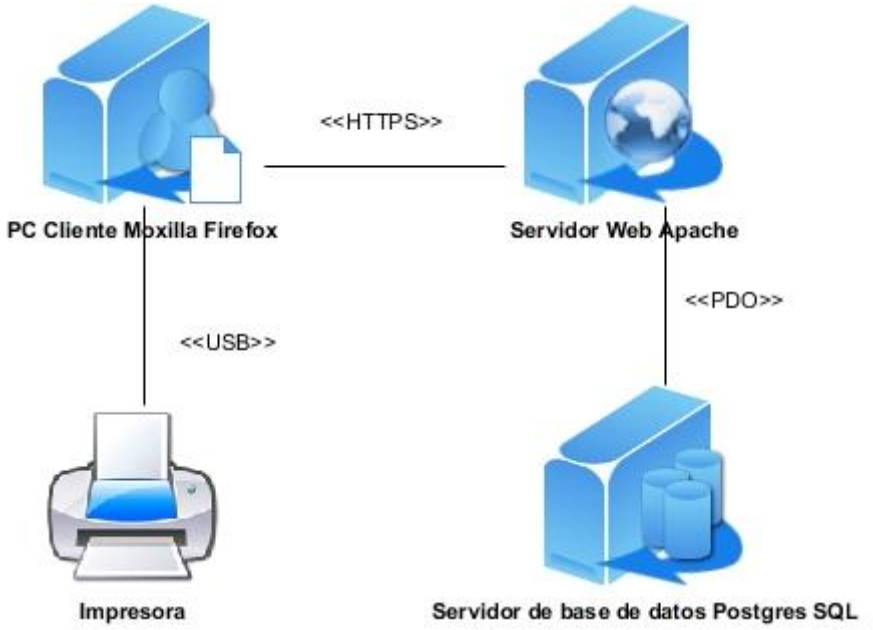

Figura 16: Diagrama de Despliegue.

PC Cliente: computadora en la que la aplicación se ejecutará a través de un navegador, debe usar el Mozilla Firefox.

Servidor Web: es donde radica toda la lógica de negocio de la aplicación. Servidor Web Apache 2.0 o superior, utilizando para ellos el lenguaje de programación del lado del servidor PHP 5.

Servidor de Base de datos: servidor de Datos PostgreSQL 8.3, donde se encuentra la base de datos que se emplea en el sistema, puede estar instalado en la misma computadora donde se encuentra el servidor Web.

Impresora: utilizada para imprimir las reevaluaciones de cuentas y los comprobantes de contabilidad en caso de ser necesario.

3.3 Validación del diseño propuesto.

Con el propósito de lograr una validación del diseño del componente se realizó un estudio minucioso de las principales técnicas utilizadas para dar solución al problema inicial, se hizo uso de las métricas como medidor de la legibilidad, reutilización, limpieza y buenas prácticas de diseño. Una métrica es un instrumento que cuantifica un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto (Pressman, 2005).

Las métricas básicas estudiadas abarcan atributos de calidad que permiten medir la calidad del diseño propuesto. Dentro de estos se encuentran (Pressman, 2005):

- ✓ Responsabilidad**:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- ✓ Complejidad de implementación: consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- ✓ Reutilización: consiste en el grado de reutilización 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, está muy ligada a la característica de Reutilización.
- ✓ 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 costos y la planificación del proyecto.
- ✓ Cantidad de pruebas: consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

Se decide hacer uso de las series de métricas propuestas por Chidamber y Kemerer (CK), así como las que proponen Lorenz y Kidd. Las métricas concebidas para evaluar la calidad del diseño se seleccionaron teniendo en cuenta las características propias del componente. Estas son:

Carencia de cohesión en los métodos (CCM) serie de métricas de CK.

Cada método dentro de una clase, C, accede a uno o varios atributos (también llamados variables de instancia), CCM es el número de métodos que accede a uno o varios de los mismos atributos. Si no

existen métodos que accedan a los mismos atributos, entonces CCM = 0. En general, los valores altos para CCM implican que la clase debe diseñarse mejor descomponiendo en dos o más clases distintas.

- ✓ **Resultados obtenidos de la aplicación de la métrica al componente**

Después de aplicar la métrica CCM a una de las clases más importantes dentro del componente (ReevaluarController) y haber analizado los resultados obtenidos, se puede concluir que la clase tomada como prueba para la métrica, presenta un nivel de CCM de 4, este resultado representa un nivel medio para los pasos o medidas que proponen algunos autores en el campo de la métrica y el diseño.

Tamaño de clase (TC) propuesta por Lorenz y Kidd.

El tamaño general de una clase puede medirse determinando las siguientes medidas:

- ✓ El total de operaciones (operaciones tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- ✓ El número de atributos (atributos tanto heredados como privados de la instancia), encapsulados por la clase.

Grandes valores de TC representa gran responsabilidad de la clase. Esto implica la reducción de la reutilización de la clase y complica la implementación y las pruebas. De forma general, operaciones y atributos deben ser ponderados al determinar el tamaño de la clase. Para valores pequeños de TC para una clase existe mayor posibilidad de que la clase pueda ser reutilizada (Pressman, 2005).

En la siguiente tabla se muestran los parámetros de calidad para valores grandes de TC, ver (Tabla 8), propuestos por (Lorenz, y otros, 1994). Así como las medidas para las principales clases de la solución, ver (Tabla 9).

Atributos de calidad que afecta	Valores Grandes de TC
Reutilización	Reduce la reutilización de la clase.
Implementación	Grado de dificultad y complicación que tiene la implementación.
Complejidad de las pruebas	Hace compleja las pruebas del sistema.
Responsabilidad	La responsabilidad asignada a una clase debe ser bastante.

Tabla 8: Parámetros de calidad para valores grandes de TC.

Medidas para las principales clases de la solución.

No	Clases	Atributos	Operaciones	Tamaño
1	Reevaluar Controller	31	13	Grande

2	DatCuentasreevaluadasModel	6	6	Pequeño
3	DatReevaluacionModel	6	9	Pequeño
4	DatCuentasreevaluadas	5	6	Pequeño
5	DatReevaluacion	4	5	Pequeño

Tabla 99: Clases a las que se les aplicó la métrica TC.

- ✓ **Resultados obtenidos de la aplicación de la métrica al componente**

Se les aplicó la métrica de TC a un total de 5 clases para un total de 53 atributos y un promedio de atributos de 10,6 y un total de 39 operaciones y un promedio de operaciones de 7,8. De las clases analizadas se tienen un total de 4 de tamaño pequeño, 0 de tamaño medio y 1 grandes.

Umbral	TC	Cantidad de clases
<= 20	Pequeño	4
> 20 y <= 30	Medio	0
> 30	Grande	1

Tabla 100: Clases por tamaño.

Con la representación de los resultados obtenidos en un gráfico de por ciento obtenemos que del total de clases analizadas existe un 80% de clases de tamaño pequeño, un 0% de tamaño medio y un 20% de tamaño grande.

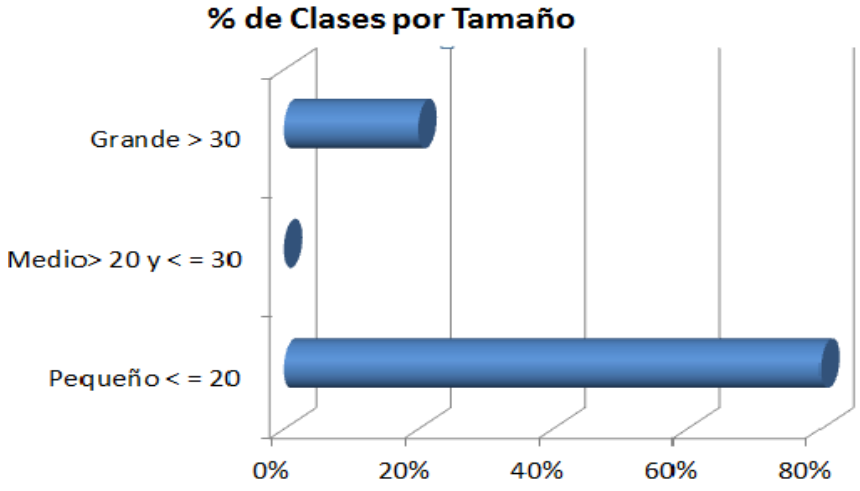

Figura 17: Por ciento de clases por tamaño.

Después de mostrados los datos en las tablas anteriores se puede arribar a la conclusión de que la mayoría de las clases se clasifican en pequeñas, ninguna mediana, y una grande, lo que implica un resultado positivo según los parámetros de calidad propuestos para esta métrica.

3.4 Pruebas de software aplicadas al componente

Las pruebas de software no garantizan que un sistema esté libre de errores, sino que se detecten la mayor cantidad de defectos posibles para su debida corrección. Estas son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa u ordenador, probando el comportamiento del componente (Ecured, 2009).

Aplicación de las Pruebas de caja blanca al componente.

Se realizan con conocimiento de la estructura interna del programa y su objetivo principal es probar que todos los caminos del código están correctos. Son usadas por lo general en pruebas unitarias y son realizadas por los programadores (Ramírez, 2012).

La prueba de caja blanca empleada, fue la Prueba del Camino Básico, la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Ecured, 2010).

Es necesario calcular antes la complejidad ciclomática, del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método obtenerreevaluacionesAction(), ver (Figura 18), y el grafo de flujo asociado al mismo, ver (Figura 19).

```
public function obtenerreevaluacionesAction() {
    $inicio = $this->_request->getPost('start');  // 1
    $limite = $this->_request->getPost('limit');  // 1
    $busqueda = $this->_request->getPost('busqueda');  // 1
    $denom = json_decode($this->_request->getPost('criterioDenom'));  // 1
    $fecha = json_decode($this->_request->getPost('criterioFecha'));  // 1
    $origen = json_decode($this->_request->getPost('cuentasOrigen')); // 1
    $efecto = json_decode($this->_request->getPost('cuentasEfecto')); // 1
    $idtasa = json_decode($this->_request->getPost('idtasa')); // 1
    $idnommonedaentidad = json_decode($this->_request->getPost('idnommonedaentidad')); // 1
    $reevaluacionModel = new DatReevaluacionModel(); // 1
    if (($busqueda == 1 || $busqueda == 2) && (strlen($denom) > 0)) { // 2
        if ($busqueda == 1) { // 3
        $reevaluaciones = $reevaluacionModel->BuscarReevaluaciones($limite,
        $inicio, $denom, $fecha = null); // 4
    } // 5
    else if ($busqueda == 2) { // 6
        $reevaluaciones = $reevaluacionModel->BuscarAvanzadaReevaluaciones
        ($limite, $inicio, $denom, $fecha, $idtasa, $idnommonedaentidad,
            $origen = null, $efecto = null); // 7
        } // 8
    }  // 9
    else {
        $reevaluaciones = $reevaluacionModel->ObtenerReevaluaciones($limite, $inicio); // 10
    }
```

```
    echo json_encode(array('results' => $reevaluaciones,
    'total' => count($reevaluaciones))); // 11
} 12
```

Figura 18: Código del método obtenerreevaluacionesAction().

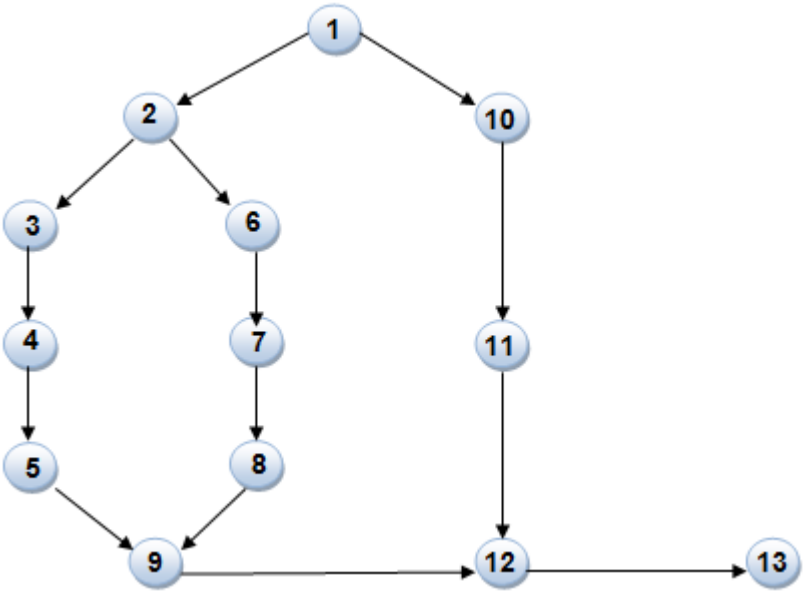

Figura 19: Grafo de flujo asociado al método reevaluarcuentaAction().

Fórmulas para calcular la complejidad ciclomática:

1. **V (G) = (A - N) + 2**

Donde **"A"** es la cantidad de aristas y **"N"** la cantidad de nodos.

V (G) = (13 - 12) + 2

V (G) = 3

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

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

V (G) = 2 + 1

V (G) = 3

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

Donde **"R"** representa la cantidad de regiones en el grafo.

V (G) = 3

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 3, lo que significa que existen 3 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, ver (Tabla 12).

Número	Caminos básicos
1	1-10-11-12
2	1-2-3-4-5-9-11-12
3	1-2-6-7-8-9-8-9-11-12

Tabla 111: Caminos básicos del flujo.

Posteriormente de haber determinado los caminos básicos se procede a ejecutar los casos de pruebas para cada uno de estos. Para definir los casos de prueba es necesario tener en cuenta:

- ✓ **Descripción:** se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.
- ✓ **Condición de ejecución:** se especifica cada parámetro para que cumpla una condición deseada y así ver el funcionamiento del procedimiento.
- ✓ **Entrada:** se muestran los parámetros que serán la entrada al procedimiento.
- ✓ **Resultados Esperados:** se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

Caso de prueba para el camino básico # 1.

Descripción**:** todas las reevaluaciones de cuentas que han sido insertadas en el sistema, después de haber realizado el cálculo correspondiente.
Condición de ejecución: que exista al menos una reevaluación de cuentas insertada en el sistema.
Entrada:

- ✓ Reevaluación_1, representa la denominación de la reevaluación de cuentas.
- ✓ (01/05/2013), representa la fecha en la que se realizó la reevaluación de cuentas.

Resultados esperados: se espera que muestre un listado de reevaluaciones de cuentas insertadas en el sistema y que coinciden con el criterio de búsqueda.
Resultados obtenidos: satisfactorio.

Caso de prueba para el camino básico # 2.

Descripción: todas las reevaluaciones de cuentas que han sido insertadas en el sistema, y que coinciden con el criterio de búsqueda seleccionado.
Condición de ejecución: que exista al menos una reevaluación de cuentas insertada en el sistema.
Entrada:

- ✓ Reevaluación_1, representa la denominación de la reevaluación de cuentas.
- ✓ (01/05/2013), representa la fecha en la que se realizó la reevaluación de cuentas.

Resultados esperados: se espera que muestre un listado de reevaluaciones de cuentas insertadas en el

sistema y que coinciden con el criterio de búsqueda.
Resultados obtenidos: satisfactorio.

Caso de prueba para el camino básico # 3.

Descripción: todas las reevaluaciones de cuentas que han sido insertadas en el sistema, y que coinciden con el criterio de búsqueda seleccionado.
Condición de ejecución: que exista al menos una reevaluación de cuentas insertada en el sistema.
Entrada:

- ✓ Reevaluación_1, representa la denominación de la reevaluación de cuentas.
- ✓ (01/05/2013), representa la fecha en la que se realizó la reevaluación de cuentas.
- ✓ CUC, representa la moneda en la cual se realizó el proceso de la reevaluación de cuentas.
- ✓ 23.000, representa la tasa contable de la moneda que se seleccionó.

Resultados esperados: se espera que muestre un listado de reevaluaciones de cuentas insertadas en el sistema y que coinciden con el criterio de búsqueda.
Resultados obtenidos: satisfactorio.

Aplicación de las Pruebas de caja negra

Para realizarlas no es necesario conocer los detalles internos del programa y su objetivo fundamental, es probar que el software está acorde a los requisitos (Ramírez, 2012).
Para la ejecución de este tipo de pruebas se realizaron un conjunto de casos de prueba que tenían como objetivo principal determinar si los requisitos estaban parcial o completamente satisfactorios. A continuación se representa el diseño de caso de prueba para el requisito funcional Calcular reevaluación de cuentas como se muestra en la tabla 13:

Caso de prueba para el requisito Calcular reevaluación de cuentas.

Condiciones de ejecución

- ✓ Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- ✓ Se debe seleccionar el subsistema Configuración/Multimoneda/Reevaluar.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

1: Calcular reevaluación de cuentas.	El sistema debe permitir calcular una reevaluación de cuentas con los atributos: Denominación, Fecha, Tasa de cambio, Moneda a reevaluar, Cuentas a reevaluar y Cuentas para reflejar el efecto.	EP 1.1: Calcular reevaluación de cuentas introduciendo datos válidos.	– Se introducen los datos para realizar el cálculo de reevaluación de cuentas correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.2: Calcular reevaluación de cuentas introduciendo datos válidos presionando el botón **Aplicar**.	– Se introducen los para realizar el cálculo de reevaluación de cuentas correctamente. – Se presiona el botón **Aplicar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.3: Calcular reevaluación de cuentas introduciendo datos inválidos	– Se introducen los datos para realizar el cálculo de reevaluación de cuentas. – Se presiona el botón Aceptar. – Se muestra un mensaje informando del error.
		EP 1.4: Calcular reevaluación de cuentas dejando campos vacíos.	– Se introducen los datos para realizar el cálculo de reevaluación de cuentas dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje

			informando del error.
		EP 1.5: Cancelar.	– Se introducen o no los datos para realizar el cálculo de reevaluación de cuentas. – Se presiona el botón **Cancelar**.

Tabla 122: Caso de Prueba para el requisito Calcular reevaluación de cuentas.

Para más información sobre el caso de prueba para el requisito Calcular reevaluación de cuentas, consultar el expediente de proyecto de CEIGE, en el cual se encuentran todos los Diseños de Casos de Prueba realizados para cada requisito funcional del componente.

3.6 Resultados de las Pruebas.

El componente fue sometido a dos iteraciones de pruebas de caja negra. En la primera iteración se detectaron 32 no conformidades, de las cuales 25 fueron de aplicación y 8 de documentación. Dentro de las no conformidades referentes a las de aplicación se encuentran 14 de validación, 9 de interfaz y 2 de funcionalidad. En cuanto a las de documentación se encontraron 3 no conformidades de redacción, 1 de ortografía y 4 de correspondencia. Todas estas no conformidades fueron resueltas seguidamente de haberlas detectado, lo que permitió que el componente pasara a una segunda iteración en la que no se detectaron no conformidades, obteniendo resultados satisfactorios.

La aplicación desarrollada fue presentada ante el cliente, el cual luego de haberla probado estuvo satisfecho con el producto entregado. Prueba de esto lo constituye el acta de aceptación emitida por este, en la cual consta que el componente realizado cumple con las condiciones de calidad necesarias y satisface las necesidades plasmadas por el cliente, además contribuye a mejorar la integridad de los saldos contables en el Sistema Integral de Gestión Cedrux. Para más información acerca del acta de aceptación emitida, ver **(Anexo 3).**

Conclusiones del capítulo.

En este capítulo se realizó la validación del componente, lo que permitió obtener importantes resultados como:

- ✓ La validación del diseño, a través de métricas arrojaron resultados satisfactorios sobre el diseño realizado, demostrando que este era simple y los atributos de calidad alcanzaban niveles favorables.
- ✓ La realización de pruebas de caja blanca y caja negra al componente arrojaron resultados satisfactorios, demostrándose con ello la calidad del componente realizado.

CONCLUSIONES GENERALES.

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, resaltando que:

- ✓ El análisis de los sistemas de gestión contable estudiados, fundamenta la necesidad de desarrollar el componente para la reevaluación de cuentas para el sistema Cedrux.
- ✓ El análisis del componente de reevaluación de cuentas, permitió recoger todas las necesidades de los clientes, garantizando una comprensión exacta sobre las funcionalidades a informatizar.
- ✓ El diseño de la estructura de componentes, clases y datos aplicando los patrones de diseño garantizó una comprensión exacta sobre las funcionalidades a informatizar.
- ✓ La implementación del componente utilizando las herramientas, lenguajes y tecnologías definidas por el marco de trabajo Sauxe permitió la informatización del proceso.
- ✓ Las diferentes pruebas que se realizaron al componente, arrojaron resultados satisfactorios demostrando la calidad del componente desarrollado, puesto que permite lograr la integridad de los saldos contables.

RECOMENDACIONES

Una vez finalizado el presente trabajo de diploma se recomienda:

- ✓ Implementar la funcionalidad que permita generar las contabilizaciones de las reevaluaciones al subsistema de Contabilidad General.
- ✓ Implementar una mejora visual en el componente, para que se puedan realizar reportes de las cuentas reevaluadas con más detalles.
- ✓ Extender las pruebas realizadas al componente haciendo uso de herramientas que automaticen las mismas.

REFERENCIAS BIBLIOGRÁFICAS

Assets. 2005. Assets. Sistema de Gestión Integral. [En línea] 2005. [Citado el: 24 de Enero de 2013.] www.assets.co.cu/assets.asp.

Cabrera González, Miguel P. , y otros. 2007. *SISTEMA ECONOMICO INTEGRADO. XV FORUM DE CIENCIA Y TECNICA.* 2007.

Ciberaula. 2010. Ciberaula. [En línea] 2010. [Citado el: 26 de Noviembre de 2012.] linux.ciberaula.com/articulo/linux_apache_intro.

Colectivo de Json. 2011. JSON. [En línea] 2011. [Citado el: 2012 de Dieciembre de 20.] http://www.json.org/json-es.html.

Correa, Patricio Ramírez. 2004. *Rol y contribución de los sistemas de planificación de los recursos de la empresa (ERP).* 2004.

del Toro, José Carlos y González, Henry Raúl. 2008. *Documento Visión Proyecto ERP-Cuba.* Habana.Cuba : UCI, 2008.

Doctrine. 2008. Doctrine. [En línea] 2008. [Citado el: 5 de Diciembre de 2012.] www.doctrine-project.org.

Ecured. 2010. EcuRed. [En línea] 15 de Noviembre de 2010. [Citado el: 16 de Diciembre de 2012.] www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

EcuRed: Enciclopedia cubana. 2009. Proceso de Desarrollo y Gestión de Proyectos de Software. [En línea] 2009. [Citado el: 10 de Enero de 2013.] http://www.ecured.cu/index.php/Pruebas_de_software.

Entidades, Centro de Informatización de Gestión de. 2012. *Modelo de Desarrollo de Software.* Boyeros, Ciudad de la Habana, Cuba : s.n., 2012.

ERP Openbravo. 2007. ERP Openbravo. [En línea] 2007. [Citado el: 5 de Diciembre de 2012.] www.openbravo.com/es.

Fernández González, Mairelys y Zorrilla Rivera, Osley. 2010. Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX. *Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* Ciudad de la Habana : UCI, 2010.

Fowler, Martin. 2003. *Patterns of enterprise application architecture.* s.l. : Addison-Wesley, 2003. ISBN 978-0-321-12742-6.

Garrett, Jesse James. 2005. *Ajax: A New Approach to Web Applications.* 2005.

Gómez Baryolo, Oiner, Morejón Borbón, Yoandry y García Tejo, Darien. 2012. *Arquitectura tecnológica para el desarrollo de software.* 2012.

González, Miguel P. Cabrera. *SISTEMA ECONOMICO INTEGRADO.* Habana : XV FORUM DE CIENCIA Y TECNICA.

HTML. 2005. Guía para escribir documentos HTML. [En línea] 2005. [Citado el: 7 de Diciembre de 2012.] www.uv.es/jac/guia.

IEEE. 1993. *Standards Collection: Sofware Engineering.* 1993.

Jacobson, Ivar, Booch, Grady y Rumbaugh, Jim . 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid,España : s.n., 2000.

Json, Colectivo de. 2011. JSON. [En línea] JSON, 2011. [Citado el: 30 de Noviembre de 2011.] http://www.json.org/json-es.html.

Lago, Ramiro. 2007. *Patrones de diseño.* 2007.

Larman, Graig. 2004. *Una introducción al análisis y diseño orientado a objetos y al proceso unificado. UML y Patrones.* 2004.

Lorenz, M y Kidd, J. 1994. *Object-Oriented Software Metric.* 1994.

Martínez, Rafael. 2009. PostgreSQL-es. Procedimientos almacenados y PL/pgSQL. [En línea] 2009. [Citado el: 20 de Noviembre de 2012.] www.postgresql.org.es/node/297.

Microsoft Corporation. 2012. Microsoft Corporation.Web Microsoft Dynamics. [En línea] 2012. [Citado el: 27 de Noviembre de 2012.] http://www.microsoft.com/es-es/dynamics/erp.aspx.

Modelo de Datos. 2012. CHRONOTECH, el futuro esta en nuestras manos. [En línea] 2012. [Citado el: 2013 de Febrero de 25.] https://sites.google.com/site/jalexiscv/modelosdedatos.

Mozilla Firefox. 2009. Mozilla Firefox. GetFirefox. [En línea] 2009. [Citado el: 22 de Noviembre de 2012.] www.getfirefox.es/firefox-features.

msdn. 2012. [En línea] Microsoft, 2012. [Citado el: 2013 de Marzo de 20.] http://msdn.microsoft.com/es-es/library/dd409390.aspx.

Netbeans 7.2. 2010. Netbeans 7.2. Redes Zone ADSL ZONE. [En línea] 2010. [Citado el: 16 de Diciembre de 2012.] www.redeszone.net/2012/07/27/netbeans-7-2-nueva-version-de-este-popular-ide.

Openbravo. 2007. Openbravo.Home Openbravo. [En línea] 2007. [Citado el: 4 de Diciembre de 2012.] www.openbravo.com/es/product/erp/features.

Osmar, Leyet Fernández y Tahirí, Rivero Alvárez. 2012. *Gestión del conocimiento en el equipo del análisis del proyecto ERP-CUBA.* Boyeros, Ciudad de la Habana, Cuba : s.n., 2012.

Paoli, Jean. 2010. *Extensible Markup Language (XML) 1.0.* 2010.

PCC. Abril 2011. *VI CONGRESO DEL PARTIDO COMUNISTA DE CUBA.Lineamientos de la Política Económica y Social.* Cuba : s.n., Abril 2011.

Pressman, Roger S. 2005. *Ingeniería de Software, Un enfoque práctico.* s.l. : McGraw-Hill Companies, 2005. ISBN: 8448132149.

Ramírez, Jaime. 2012. Métodos de Prueba del Software. Unidad de Programación. [En línea] 2012. [Citado el: 10 de Enero de 2013.] http://lml.ls.fi.upm.es.

Revisiones de código y estándares de codificación. 2013. msdn. [En línea] 2013. [Citado el: 28 de Marzo de 2013.] http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx.

Rodas XXI. 2002. CITMATEL. Rodas XXI.Sistema Integral Económico Administrativo. [En línea] 2002. [Citado el: 24 de Noviembre de 2012.] www.rodasxxi.cu/contabilidad.php.

Sánchez-Segura, Maribel y Moreno, Ana M. 2010. *Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el momento de Arquitectónico.* Madrid, España : s.n., 2010.

SAP. 2010. SAP. Enterprise Resource Planning (ERP). [En línea] 2010. [Citado el: 4 de Diciembre de 2012.] http://www.gbm.net/soluciones/enterprise_resource_planning.php.

Scribd. 2010. Scribd. [En línea] 2010. [Citado el: 20 de Noviembre de 2012.] www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

Sommerville, Ian. 2002. *Ingeniería de Software. Sexta edición.* Addison-Wesley : s.n., 2002. ISBN: 970-26-0206-8.

2007. Sparx Systems. [En línea] Sparx Systems Pty Ltd, 2007. [Citado el: 5 de mayo de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.

Territorio PC. 2010. Territorio PC. [En línea] 2010. [Citado el: 1 de Diciembre de 2012.] territoriopc.com/category/javascript.

The FreeBSD Foundation. 2012. The FreeBSD Project. [En línea] 2012. [Citado el: 16 de Enero de 2012.] http://www.freebsd.org/doc/es/articles/explaining-bsd/article.html.

The PHP Group. 2001. PHP. Hypertext Preprocessor. [En línea] 2001. [Citado el: 7 de Diciembre de 2012.] www.php.net.

Unidad de Compatibilización Integración y Desarrollo de Soluciones Informáticas para la defensa. 2009. *Proceso de Desarrollo y Gestión de Proyectos de Software.* Ciudad de la Habana : s.n., 2009.

Unidad de Compatibilización Integración y Desarrollo. 2009. *Proceso de Desarrollo y Gestión de Proyectos de Software.* Ciudad de la Habana : s.n., 2009.

Universidad Central de Chile. 2011. Universidad de Chile. Departamentos de ciencias de la computacion.Sitio Web DCC. [En línea] 2011. [Citado el: 6 de Marzo de 2013.] www.dcc.uchile.cl/~psalinas/uml/modelo.html.

Visual Paradigm. 2007. Visual Paradigm. Free Download Manager.org. [En línea] 2007. [Citado el: 25 de Noviembre de 2012.] www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p.

Wilson, Leslie. 1996. *Comparative Programming Languages. 2da Edición.* 1996.

ZEND FRAMEWORK. 2010. ZEND FRAMEWORK. . [En línea] 2010. [Citado el: 5 de Diciembre de 2012.] framework.zend.com/manual/en.

ANEXOS

Anexo 1: Descripción de Requisitos Funcionales

Descripción de Requisitos Funcionales: Búsqueda avanzada de cuenta reevaluada.

Precondiciones	El usuario debe haberse autenticado en el sistema. Deben haberse registrado al menos una reevaluación de cuentas en el sistema.
Flujo de eventos	
Flujo básico	
1	Se presiona la opción Búsqueda Avanzada.
2	Se insertan los criterios de búsqueda avanzada: Denominación. Fecha. Moneda a reevaluar. Tasa de cambio.
3	El sistema muestra un listado de la(s) cuentas reevaluadas que cumplen con los criterios de búsqueda especificados. Mostrando los datos que poseen la(s) reevaluaciones encontradas en el sistema Denominación. Fecha.
4	Concluye el requisito.
Pos-condiciones	
1	N/A
2	N/A
Flujos alternativos	
Flujo alternativo 2.a No existen datos que cumplan con los criterios especificados	
1	El sistema notifica que no existen datos que cumplan con los criterios especificados.
2	Volver al paso 1 del Flujo Básico.
Flujo alternativo 2.b Existen campos vacíos	
1 El sistema notifica que no se ha insertado ningún criterio de búsqueda.	
2 Volver al paso 1 del Flujo Básico.	
Pos-condiciones	
1	N/A

Validaciones		
6	N/A	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Cuenta reevaluada**	Denominación. Fecha. Moneda a reevaluar. Tasa de cambio.
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Prototipo elemental de interfaz gráfica de usuario

Anexo 2: Diseño de caso de prueba.

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema Configuración/Multimoneda/Reevaluar.
- Se debe seleccionar la cuenta reevaluada que se desea Modificar.
- Se debe presionar el botón **Modificar** de la barra de opciones.

Requisitos a probar.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Modificar reevaluación de cuentas.	El sistema debe permitir modificar una reevaluación de cuentas con los atributos Denominación, Fecha, Tasa de cambio, Moneda a reevaluar, Cuentas a reevaluar y Cuentas para reflejar el efecto.	EP 1.1: Modificar reevaluación de cuentas introduciendo datos válidos.	– Se introducen correctamente los datos de la reevaluación de cuentas que se deseen modificar. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
	El sistema no debe permitir modificar la reevaluación de cuentas.	EP 1.2: Modificar la reevaluación de cuentas introduciendo datos inválidos.	– Se introducen los datos inválidos de la reevaluación de cuentas. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP: 1.3 Modificar la reevaluación de cuentas introduciendo datos existentes.	– Se introducen los datos de la reevaluación de cuentas con datos ya existentes. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP 1.4: Modificar la reevaluación de cuentas dejando campos vacíos.	– Se introducen los datos dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
	Se cierra la interfaz sin realizar ninguna acción.	EP 1.5: Cancelar.	– Se introducen o no los datos de la reevaluación de cuentas. – Se presiona el botón **Cancelar**.

Descripción de la variable.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Inválido	Inválido
1	Denominación	Campo de texto.	Letras, números_, -.	Números, caracteres especiales.	Vacío.	NA	NA
2	Fecha	Campo de selección (No editable).	Números.	Letras, caracteres especiales.	Vacío.	NA	NA
3	Tasa de cambio	Campo de selección (No editable).	Letras.	Números, caracteres especiales.	Vacío.	NA	NA
4	Moneda	Campo de selección (No editable).	Letras.	Números, caracteres especiales.	Vacío.	NA	NA
5	Cuentas a reevaluar	Campo de selección (No editable).	Letras, números.	Caracteres especiales.	Vacío.	NA	NA
6	Cuentas de efecto	Campo de selección (No editable).	Letras, números.	Caracteres especiales.	Vacío.	NA	NA

Juegos de datos a probar.

Id del escenario	Escenario	Denominación	Fecha	Tasa de cambio	Moneda	Cuentas a reevaluar	Cuentas de efecto	Respuesta del sistema	Resultado de la prueba
EP 1.1	Modificar reevaluación de cuentas introduciendo datos válidos.	V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema modifica la reevaluación de cuentas y muestra el mensaje de información: “La reevaluación de cuentas se ha modificado satisfactoriamente.”. El sistema cierra la interfaz.	NA

EP 1.2	Modificar reevaluación de cuentas introduciendo datos inválidos.	I (%$"@)	V(01/05/2013)	V(Tasa de moneda Contable 9	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema mantiene la interfaz abierta.	NA
		V(Reevaluación_1)	I(%$"@)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema mantiene la interfaz abierta.	
		V(Reevaluación_1)	V(01/05/2013)	I(%$·"@)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema mantiene la interfaz abierta.	
		V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	I(%$·"@)	V(107 Debito)	V(901 Cierre)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema mantiene la interfaz abierta.	

		V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	I(%$·”@)	V(901 Cierre)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema mantiene la interfaz abierta.	NA
		V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	I(%$·” @)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema mantiene la interfaz abierta.	NA
EP 1.3	Modificar reevaluación de cuentas con datos ya existentes.	V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema muestra el mensaje “Verifique que haya modificado algún dato.”. El sistema mantiene la interfaz abierta.	

EP 1.4	Modificar reevaluación de cuentas dejando campos vacíos.	(Vacío)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) vacíos”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	NA
		V(Reevaluación_1)	(Vacío)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) vacíos”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	NA

V(Reevaluación_1)	V(01/05/2013)	(Vacío)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) vacíos”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	NA
V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	(Vacío)	V(107 Debito)	V(901 Cierre)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) vacíos”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	NA

		V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	(Vacío)	V(901 Cierre)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) vacíos”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	NA
		V(Reevaluación_1)	V(01/05/2013)	V(Tasa de moneda Contable)	V(CUC)	V(107 Debito)	V(901 Cierre)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) vacíos”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	
EP 1.5	Cancelar.	NA	NA	NA	NA	NA	NA	El sistema cierra la	NA

UCI Universidad de las Ciencias Informáticas

CENTRO DE INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

Producto: Cedrux.

Componente: Reevaluación de cuentas.

Firma de los involucrados en el proceso:

- **Por la parte del Cliente (INTERAUDIT):** Odalys Domínguez Ferrer
- **Especialista del proyecto(Cedrux):** Osmar Joyet Fernández

Observaciones del proceso:

Teniendo en cuenta que las No Conformidades han sido debidamente resueltas y validada la eficacia de la corrección por parte del cliente, se ha tomado el acuerdo de Aceptar con fecha 16 de Mayo de 2013, de la aplicación "**CEDRUX**" en su versión **1.1** el componente:

- ✓ Reevaluación de cuentas

Para que conste la Aceptación de los resultados de las pruebas y por tanto la Aceptación de los entregables especificados, dando fe del acuerdo, se extiende la presente **Acta** en dos (2) ejemplares, rubricados por los principales **Representantes de las Partes**.

Entrega *[Parte Suministradora]*	**Recibe** *[Representante Parte Cliente]*
Nombre y Apellidos: Osmar Joyet Fdez	**Nombre y Apellidos:** Odalys Domínguez Ferrer
Cargo: Subdirector de Producción	**Cargo:** Consultora Económica
Firma:	**Firma:**

GLOSARIO DE TÉRMINOS

Capital: La mayoría de las empresas adoptan un concepto financiero del capital al preparar sus estados financieros. Bajo esta concepción del capital, que se traduce en la consideración del dinero invertido o del poder adquisitivo invertido, capital es sinónimo de activos netos o patrimonio neto de la empresa.

Cedrux: Denominación otorgada al ERP Cubano desarrollado por la UCI, su nombre se origina de la combinación de las primeras cuatro letras de Cedro y las dos últimas letras de Linux. El Cedro por ser una planta de tronco duro, persistente, robusto y Linux por las nuevas concepciones de la migración a software libre en el territorio nacional para lograr la soberanía e independencia tecnológica.

Cuenta: Es el instrumento que permite identificar, clasificar y registrar un elemento o hecho económico realizado por una empresa.

Dualidad monetaria: Consecuencia de la existencia de dos monedas, que comparten legalmente las mismas funciones del dinero en la economía nacional.

Factor de conversión: Factor que se utiliza para ajustar el importe de las monedas cuando la tasa de cambio está expresa en relaciones diferentes a la unidad.

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

Multimoneda: No es más que la utilización de varias monedas en las transacciones económicas, la convertibilidad de varias monedas con respecto a la moneda contable y todas las particularidades que implica el tema de la dualidad monetaria.

Tasa de cambio: Es la proporción utilizada para el intercambio de dos tipos de monedas diferentes.