Universidad de las Ciencias Informáticas

Facultad 15

Título: 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

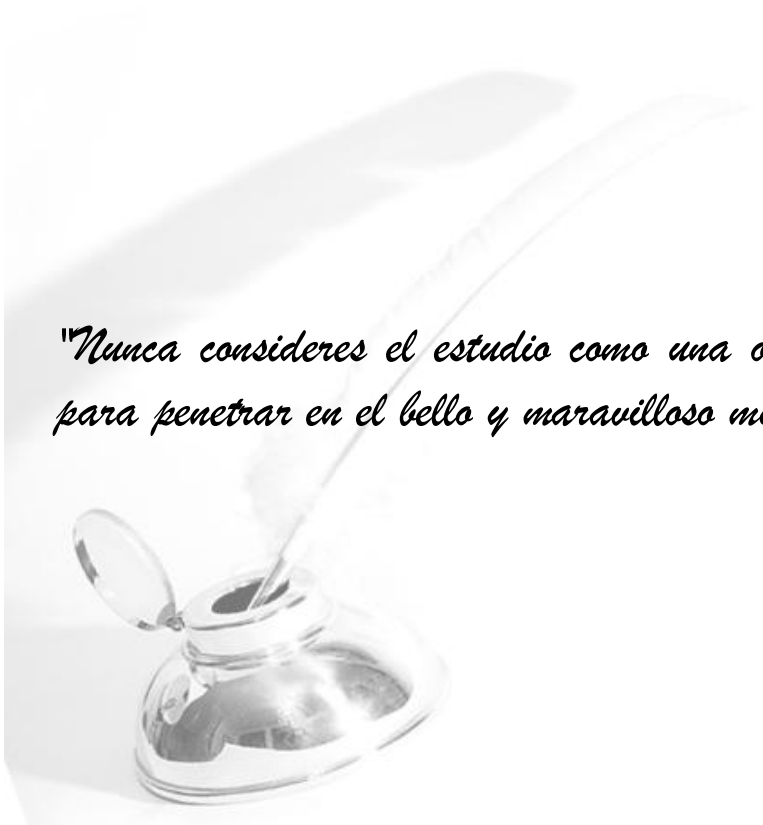
Autores: Mairelys Fernández González
Osley Zorrilla Rivera

Tutores: Ing. Giselle Almeida González
Ing. Joisel Pérez Pérez

Ciudad de la Habana. Junio de 2010. "Año 52 de la Revolución"

"Nunca consideres el estudio como una obligación sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein



Declaración de autoría:

Declaramos que somos los únicos autores del trabajo “Diseño e implementación del componente Ajuste al Costo del Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CERUX” y autorizamos a la Facultad 15 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____ _____

Mairelys Fernández González — Autor

Osley Zorrilla Rivera — Autor

_____ _____

Ing. Giselle Almeida González — Tutor

Ing. Joisel Pérez Pérez — Tutor

Resumen:

El país avanza con amplios pasos dentro del proceso de utilización ordenada y masiva de las tecnologías de la información y las comunicaciones, siendo este un factor decisivo para el desarrollo de las empresas, economía y de la sociedad cubana; proceso que busca satisfacer las necesidades de información y conocimiento de todas las personas y esferas de la sociedad. La necesidad de mejorar los procesos de la Contabilidad de Costos está impulsando a la adopción de sistemas de planificación de recursos empresariales, los cuales integran y manejan muchos de los negocios asociados con las operaciones de producción y planificación estratégica, en base a esta, se pueden determinar los capitales destinados a los materiales necesarios para llevar a cabo las actividades empresariales, sean estas de producción industrial o de servicios. Con el propósito de mejorar la eficiencia, confiabilidad y rapidez en la gestión de los procesos de la Contabilidad de Costos en cualquier entidad el presente trabajo comprende el Diseño y la Implementación del componente Ajuste al Costo del Subsistema Costos y Procesos correspondiente al Sistema Integral de Gestión de Entidades CEDRUX.

Palabras clave:

Centros de Costos, Elementos de Gastos, Cuentas de Gastos, Contabilidad, Costos, Ajuste al Costo.

Índice de contenidos:

Índice de tablas:

Introducción:

En la actualidad, la competencia económica mundial es un hecho del que ninguna empresa puede escapar. Las compañías de hoy en día necesitan anticiparse y responder de manera ágil y eficiente a las condiciones del mercado para poder ser competitivas. Lograrlo implica entre otros factores la utilización de sistemas informáticos que realicen las actividades de manera flexible, para optimizar e integrar el flujo interno de información y tomar las mejores decisiones sobre los presentes y futuros desafíos del negocio de manera que se gane en productividad, calidad y servicio al cliente incluyendo todos los ámbitos de la empresa, desde lo financiero hasta el recurso humano.

La Contabilidad de los Costos constituye un eslabón fundamental para la organización y planificación de los recursos empresariales basándose en el registro de los hechos económicos de cada entidad. La Gestión de los Costos se ha convertido en uno de sus pilares y por esta razón se utilizan mecanismos que facilitan su contabilización. Un proceso fuertemente implicado en esta actividad lo constituye el Ajuste de los Costos, que consiste en ajustar los costos reales de la producción al final del período partiendo de una estimación hecha al comienzo.

En Cuba se gestionan los costos empleando métodos manuales y automatizados. Los métodos manuales son lentos, engorrosos, traen consigo errores matemáticos y con ellos no existe un control eficiente de los cambios ni de la información manipulada por las entidades. Los automatizados están dados por la utilización de sistemas que durante años la industria de software ha venido desarrollando, que bien pueden ser nacionales o internacionales. De forma general solo algunos se ajustan parcial o completamente a las características de la economía nacional, la mayoría no realizan todas las operaciones relacionadas con la gestión de la contabilidad y el hecho de que determinadas aplicaciones sean propietarias o que estén desarrolladas sobre tecnologías y herramientas privativas trae consigo que las posibilidades de uso estén limitadas. La variedad de estas soluciones informáticas y el inconveniente de brindarles el mantenimiento que requieren, obstaculizan la gestión eficiente de la economía en las entidades repercutiendo negativamente en la actividad contable.

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, de manera que pueda ser utilizado 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.

La Universidad de las Ciencias Informáticas en conjunto con el Ministerio de Finanzas y Precios y la participación de especialistas de otras entidades desarrolladoras de software actualmente se encuentran vinculados en la realización de un Sistema Integral de Gestión de Entidades denominado CEDRUX con el objetivo de responder a las necesidades contables de las entidades cubanas de modo que modele y automatice la mayoría de sus procesos (Contabilidad General, Caja, Banco, Costos y Procesos, Cobros y Pagos, Inventario, Auditoría, Planificación, etc.). Su misión es facilitar la planificación de todos los recursos y la integración de la información de la empresa.

El sistema se implementa a través de subsistemas correctamente diseñados entre los cuales se establece una eficiente comunicación. El Subsistema Costos y Procesos, responsable de gestionar las operaciones asociadas a los costos, no ofrece la posibilidad de realizar los ajustes de los costos de los productos en estado terminado, lo que trae consigo que no se pueda conocer con exactitud los gastos y los costos en los que se incurren como resultado de la producción que se realiza a nivel empresarial.

El presente trabajo de diploma para dar solución a la situación descrita con anterioridad tiene el siguiente **problema a resolver**: ¿Cómo mejorar la gestión de la Contabilidad de Costos para lograr una mayor eficiencia, confiabilidad y rapidez en la realización de los procesos de ajuste de los costos en cualquier entidad?

Se define por lo tanto como **objeto de estudio:** Los procesos contables de ajuste a los costos en la Contabilidad y el **campo de acción:** Diseño e implementación del proceso Ajuste al Costo del Sistema Integral de Gestión de Entidades CEDRUX.

El **objetivo general** del presente trabajo de diploma es: Obtener el componente Ajuste al Costo mediante el diseño y la implementación, para lograr una mayor eficiencia, confiabilidad y rapidez en la realización de este proceso.

Objetivos específicos:

1. Caracterizar y evaluar la realización de operaciones de Ajuste al Costo en el Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX.
2. Fundamentar la investigación, mediante la elaboración del Marco Teórico.
3. Diseñar e implementar el componente Ajuste al Costo.
4. Validar el componente implementado.

Idea a defender: Si se obtiene el componente Ajuste al Costo mediante el diseño y la implementación, se logrará a una mayor eficiencia, confiabilidad y rapidez en la realización de los procesos de ajuste de los costos en cualquier entidad.

Tareas para cumplir los objetivos:

1. Estudio del estado del arte referente a soluciones informáticas asociadas al área de conocimientos de Ajuste al Costo de la Contabilidad General y de Costos.
2. Análisis de la Arquitectura de Sistema para conocer sus características fundamentales, frameworks, herramientas y tecnologías definidas para el desarrollo, así como la estrategia de integración entre los diferentes componentes.
3. Estudio de la descripción del proceso Ajuste al Costo del Subsistema Costos y Procesos.
   3.1 Estudio del requerimiento funcional: Configurar Destinos del proceso Ajuste al Costo.
   3.2 Estudio del requerimiento funcional: Ajustar Costos del proceso Ajuste al Costo.
   3.3 Estudio del requerimiento funcional: Ajustes Realizados del proceso Ajuste al Costo.
4. Diseño e Implementación del componente Ajuste al Costo del Subsistema de Costos y Procesos.
   4.1 Diseño e Implementación del requisito funcional Configurar Destinos.
   4.2 Diseño e Implementación del requisito funcional Ajustar Costos.
   4.3 Diseño e Implementación del requisito funcional Ajustes Realizados.
5. Validación del componente implementado a través de pruebas.

Para el correcto desarrollo del presente trabajo de Diploma se emplearon métodos científicos de corte teórico y empírico tratando de mantener siempre un equilibrio entre lo cualitativo y lo cuantitativo. Entre los métodos teóricos se utilizaron los de Análisis – Síntesis, para el estudio y análisis de la bibliografía, la formulación del problema y los objetivos, así como para la elaboración de las conclusiones y recomendaciones a fin de alcanzar los objetivos de la investigación. Además, se utilizó el método teórico Histórico – Lógico permitiendo hacer un estudio del comportamiento actual del desarrollo de sistemas de gestión empresarial. Se utilizó también el método Sistémico, empleado en el establecimiento de las jerarquías funcionales que deben mantener los métodos implementados para dar solución a los requerimientos funcionales del sistema automático a desarrollar, permitiendo un desglose de los elementos que contendrá el mismo.

El método empírico utilizado fue la Observación, a través del cual se pudo percibir y planificar como quedaría concebido el sistema.

Estructura del documento:

Capítulo 1: En este capítulo se expone el estado del arte referente a los procesos contables de ajuste de los costos en Cuba y el resto del mundo, se realiza la fundamentación teórica del tema. Se mencionan y describen algunos sistemas existentes. Se hace un estudio del Modelo de desarrollo propuesto y de la Arquitectura Base definida para el Sistema. Por último, se justifican las herramientas y tecnologías a utilizar para el diseño e implementación del componente.

Capítulo 2: En este capítulo se elabora el diseño a partir de la Especificación de Requisitos del proceso Ajuste al Costo como base para la implementación del mismo. Está orientado a la construcción de la solución en términos de componentes y diagramas de clases de diseño, según la Arquitectura Base definida, enfocada al componente y los requerimientos del sistema. Se plantean estándares de la implementación. Se definen las clases que van a contener las funcionalidades del componente, es precisada la estructura de datos a utilizar, son descritas las clases y los algoritmos implementados, se especifica la estrategia de integración y por último se describe en términos generales el funcionamiento del componente.

Capítulo 3: En este capítulo se valida la solución propuesta a través de la realización de pruebas, el mismo se encuentra dividido en dos partes: la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y las pruebas de caja negra que se realizaron a la interfaz del mismo. Por último, se aplican los instrumentos de evaluación de las métricas: Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC).

Posibles resultados:

Diseño e implementación del componente Ajuste al Costo para el adecuado funcionamiento del Subsistema Costos y del Sistema Integral de Gestión de Entidades.

Capítulo 1: Fundamentación teórica

1.1 Introducción

El presente capítulo estará centrado en la realización de un profundo análisis para obtener información actualizada de algunos sistemas contables que han sido implementados nacional e internacionalmente y a partir de ahí conocer qué aspectos deben ser tomados en cuenta para que el sistema esté al nivel que se exige hoy en día. Se valorarán las tendencias actuales en el contexto informático partiendo de que continuamente surgen aplicaciones novedosas con mejoras que se van imponiendo en la industria del software. Será explicado el Modelo de desarrollo elaborado por la dirección del proyecto a partir del cual estará orientada la solución. Se realizará un análisis de la Arquitectura definida para el Sistema y se especificarán las tecnologías y herramientas de desarrollo a utilizar durante la implementación para facilitar el cumplimento de los requisitos tanto técnicos como funcionales.

1.2 Marco conceptual:

El marco conceptual estará centrado en la definición del proceso de Ajuste al Costo; que consiste específicamente en ajustar los costos reales al final de cada período, ya sea por concepto de producción o servicios brindados partiendo de una estimación hecha al comienzo y donde se tiene en cuenta la existencia y los destinos de los productos en cada una de las entidades.

Existe una configuración que es única para cada ajuste y esta se registra por la relación Cuenta - Centro de costo. Se parte de un análisis de los costos reales de la producción, específicamente, mano de obra, materia prima y gastos indirectos.

Posteriormente se procede a realizar los cálculos iniciales básicos para efectuar el ajuste.

Donde se define:

- UF: Unidades Físicas (UFE+UFD)
- UFE: Unidades Físicas en existencias.
- UFD: Unidades Físicas en destinos.
- UEP: Unidades equivalentes en proceso.
- IM: Importe Monetario.
- CUR: Costo unitario real de los productos.
- IAA: Importe a ajustar.
- IFP: Importe final del proceso.

- CD: Coeficiente de distribución.
- A: Ajuste.
- SaldoSub: Saldo de la Subcuenta cuando no se ajusta a inventario.
- SaldoP: Saldo del proceso (Total de Importes Acumulados).
- Importe: Importe de la producción.
- PrecioP: Precio Promedio.

Cálculos:

1. CUR = SaldoP/ (UF + UEP).
2. IFP = UEP * CUR.
3. IAA = SaldoP – IFP.
4. CD (Cuatro combinaciones válidas).
5. Importe = PrecioP * UF.
6. A = Cantidad * CD – Importe.

El Coeficiente de distribución se calcula a partir de cuatro combinaciones posibles en dependencia del Modo de ajuste y de la Base de distribución:

1. No se ajusta a inventario y se hace por Unidades Físicas.

CD= IA + SaldoSub / UF

2. Se ajusta a inventario y se hace por Unidades Físicas.

CD= IA / UF

3. No se ajusta a inventario y se hace por Importe Monetario.

CD= IA + SaldoSub / IM

4. Se ajusta a inventario y se hace por Importe Monetario.

CD= IA / IM

Cualquiera de estas combinaciones es única para cada período. Siempre que se realiza un ajuste se genera un Comprobante de operaciones, y un Documento de inventario en caso de hacer el ajuste por inventario.

1.2.1 Conceptos básicos asociados al dominio del problema:

Centro de Costo: Es la subdivisión mínima en el proceso de registro contable de los gastos en la entidad con el objetivo de medir los recursos utilizados y los resultados económicos obtenidos. (1)

Elemento del Gasto: Es un concepto económico asociado al gasto que permite la cuantificación de los recursos materiales, laborales y monetarios en los cuales se expresan los gastos de trabajo vivo y pretérito para un período en el conjunto de la actividad empresarial. (2)

Las **Cuentas de Gastos**: Son aquellas que recogen los gastos que se producen como consecuencia de la actividad empresarial y que originan una disminución del neto patrimonial. (3)

Los **Gastos Indirectos** de producción: Comprenden los importes de los gastos que se incurren en las actividades asociadas a la producción, no identificables con un producto o servicio determinado. Incluyen los gastos de las actividades de mantenimiento, reparaciones corrientes y explotación de equipos, dirección de la producción, control de calidad, depreciación de Activos Fijos Tangibles de producción y servicios auxiliares a ésta, entre otros. (4)

Los **Gastos Directos** de producción: Comprenden los importes de los gastos en los que se incurren en las actividades asociadas a la producción, identificados con un producto o servicio determinado, ya sea en la materia prima o mano de obra. (5)

1.3 Sistemas contables existentes vinculados al campo de acción:

A continuación, serán descritas y caracterizadas algunas aplicaciones informáticas que se encargan de la Gestión de Recursos Empresariales tanto en el ámbito nacional como internacional y que de alguna manera se relacionan con la Contabilidad de Costos, se valorará en cada caso la realización del proceso Ajuste al Costo.

1.3.1 Openbravo

Software desarrollado por la Universidad de Navarra en España en el año 2005, destinado principalmente para las pequeñas y medianas empresas. Las grandes áreas que integra actualmente este sistema de gestión son:

- Gestión de almacenes: almacenes y ubicaciones, unidades de almacén, lotes, números de serie, bultos, etiquetas, entradas, salidas, movimientos entre almacenes, inventarios y valoración de existencias y transportes.
- Gestión de proyectos: proyectos, fases, presupuestos, gastos y las compras asociadas.
- Gestión de servicios: recursos, servicios y los gastos.
- Gestión comercial y gestión de las relaciones con clientes (CRM): pedidos de venta, tarifas, albaranes, facturación y CRM.

➢ Gestión económico-financiera: plan de cuentas, cuentas contables, impuestos, contabilidad general, cuentas a pagar, cuentas a cobrar, contabilidad bancaria, balance, cuenta de resultados, y activos fijos.

Se caracteriza por ser una aplicación completamente web que ha sido desarrollada siguiendo el modelo MVC (en inglés Model-View-Controler) que facilita el desacoplamiento de las áreas de desarrollo. Implementada en el lenguaje Java, presenta soporte para bases de datos PostgreSQL y Oracle, se ejecuta sobre Apache y Tomcat. La estructura de datos de la aplicación está basada originalmente en una versión antigua de Compiere[1]. Una de las principales ventajas con las que cuenta este ERP es que sigue un licenciamiento de software libre asegurando el acceso público al código fuente y la posibilidad de modificar dicho código libremente, adaptándolo a las necesidades de cualquier empresa. (6)

Observación: Este sistema es uno de los más usados debido a que es libre; pero a pesar de esta facilidad y que tiene un control eficiente de los productos y los servicios a partir de la gestión económico-financiera que comprende las cuentas y los gastos, no permite ajustar los costos reales de la producción.

1.3.2 OpenERP

Software desarrollado en Argentina, soporta múltiples monedas y múltiples entidades, implementa las funcionalidades:

➢ Gestión contable y financiera.
➢ Gestión de compras.
➢ Gestión de ventas.
➢ Gestión de inventario.

El módulo contable que posee brinda las funcionalidades relacionadas con la contabilidad general, analítica y presupuestaria, permite además llevar los libros contables en forma rigurosa. Puede, entre otras tareas definir centros de costos, gestionando de una manera eficiente los datos económicos.

Se caracteriza por ser un sistema con componentes separados en un esquema Cliente-Servidor. Es

[1] Compiere es una aplicación para negocios de código abierto, ERP y CRM destinada para las empresas de pequeño y mediano tamaño.

multiplataforma; funciona sobre Linux y Windows. Implementado en el lenguaje Python, dispone de interfaces XML-RPC[2], y SOAP[3], presenta como soporte para bases de datos PostgreSQL, liberado bajo la GPL[4]. Puede ser utilizado como un programa independiente o integrado. (7)

Observación: OpenERP se describe así mismo como el ERP de código abierto más destacado y sencillo que existe hasta el momento. Es una de las aplicaciones que gestiona de manera eficiente los datos económicos basándose en los registros contables de las empresas según los centros de costos, sin embargo, el proceso Ajuste al Costo comprende una gama de funcionalidades que no están comprendidas dentro de sus componentes.

1.3.3 SAP

Software desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM. Puede configurar cada uno de los procesos según las necesidades de las empresas con el fin de obtener una ventaja competitiva en su sector. SAP ERP está diseñado para incrementar la eficiencia de la planificación y la gestión de procesos a lo largo de las empresas. Ofrece dos conjuntos de funcionalidades: uno centrado en el soporte a las operaciones y otro centrado en la generación de valor.

Respecto al primero, permite gestionar operaciones logísticas, satisfacer requisitos de calidad y cumplir con la normativa y estándares de su industria. También cuenta con las herramientas para el desarrollo y la introducción de nuevos productos con cobertura para el ciclo de vida completo del producto. El segundo conjunto de funcionalidades está diseñado para incrementar la eficiencia en los procesos de producción. Permite mejorar la totalidad de las operaciones logísticas relacionadas con la creación y mejora de sus productos y servicios.

SAP da soporte a procesos de negocios globales en las áreas de Finanzas, Desarrollo de productos, Ventas y Servicio. Entre sus áreas funcionales podemos encontrar:

---

[2] XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

[3] SOAP es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

[4] GPL: Licencia Pública General de GNU.

- Contabilidad general.
- Presupuesto.
- Bancos.
- Informes financieros.

Se caracteriza por ser un sistema totalmente preparado para trabajar con él mediante la web debido la nueva plataforma tecnológica denominada SAP NetWeaver[5]. Implementado en .NET. Funciona sobre el sistema operativo Windows y presenta soporte para bases de datos Oracle. (8)

Observación: SAP, como aplicación informática da la posibilidad de ejecutar todos los procesos del negocio haciéndolos más ágiles; incluida la administración, gestión de las relaciones con los clientes, operaciones, seguimiento, análisis y mejora del sistema de control de costes y de la contabilidad analítica. Inconveniente fundamental: Software privativo.

1.3.4 SISCONT5

Sistema cubano creado por la empresa Tecnomática en el año 2007, el cual se aviene a las definiciones y conceptos del Ministerio de la Industria Básica (MINBAS) aunque por las acciones contables financieras que permite puede ser utilizado en otras entidades nacionales. Está formado por varios módulos:

- Efectivo en Caja y Bancos.
- Inventarios.
- Cobros y Pagos.
- Facturación.
- Activos Fijos Tangibles.
- Nóminas.
- Contabilidad de Costos.

Este sistema registra todos los gastos atendiendo al objeto de costo que afecta, incluye la distribución y el cálculo de costos unitarios y define las bases de distribución de los gastos indirectos de un objeto de costo. SISCONT5 puede ser explotado en régimen mono usuario y multiusuario,

[5] SAP NetWeaver es una plataforma conformada por varias herramientas enfocadas a optimizar y sincronizar los recursos informáticos con los requisitos particulares a las estrategias y aplicaciones.

mono entidad y multientidad; para esta última existe el control de su acceso para las entidades en un mismo equipo de cómputo como servidor.
Presenta como soporte para bases de datos SQL Server 2000. Trabaja sobre el sistema operativo Windows. Hecho en la herramienta de desarrollo de software basada en conocimiento GeneXus [6]. (9)

Observación: Esta aplicación permite el cálculo de los costos unitario reales y del coeficiente de distribución de los productos, funcionalidades que tributan al ajuste de los costos, sin embargo, el empleo de tecnología privativa es el principal inconveniente que presenta dicho sistema contable.

1.3.5 Versat-Sarasola

Software desarrollado por la entidad cubana TEICO de Villa Clara, constituye un sistema que automatiza prácticamente todas las actividades de Planificación, Control y Análisis Económico de cualquier tipo de Entidad, ya que es configurable. Presenta dos variantes para su instalación: la variante típica que incluye todos los subsistemas del VERSAT-Sarasola y la variante personalizada en la que solo estarán los subsistemas seleccionados por el usuario.
Permite llevar el control y el registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades y obtener los Estados Financieros y Análisis Económicos y Financieros en estos niveles. Aparece el Subsistema de Contabilidad General como rector del sistema que maneja el resto de los subsistemas:

- Costos y Procesos
- Cobros y pagos
- Activos Fijos
- Finanzas
- Banco

Permite llevar un registro contable de gastos e incluye actividades para realizar el ajuste a los costos como son:

- El cálculo de los costos de producción y el ajuste correspondiente de forma automatizada.
- Traspasar los gastos indirectos de forma automatizada utilizando para ello diferentes bases de

[6] GeneXus es una herramienta de desarrollo de software, orientada principalmente a aplicaciones de clase empresarial para la web y plataforma Windows.

distribución.

- Determinar los gastos fijos y variables de acuerdo a la actividad que se trate.
- Trabajar con costos predeterminados.

Para el ajuste a los costos de producción se diseñan hojas de costos configurables por el usuario de acuerdo a la actividad económica que realiza.

Se caracteriza por ser una aplicación de escritorio. Concebido sobre una plataforma de trabajo Cliente-Servidor lo que permite además su instalación en red por las posibilidades que esta tecnología facilita para el trabajo en un entorno multiusuario. Implementado en Delphi. Trabaja sobre el sistema operativo Windows y presenta como soporte para bases de datos SQL Server 2000. (10)

Observación: Versat-Sarasola es unas de las soluciones informáticas más utilizadas actualmente en el país. En sus últimas versiones tiene en cuenta la dualidad de monedas y la multientidad; define detalladamente todos y cada uno de los procesos que comprende el Ajuste al Costo en las entidades eficientemente de forma automática, sin embargo, los grupos presupuestados y elementos de gastos están comprendidos para sólo 3 niveles imposibilitando un desglose más detallado, está además soportado sobre tecnología privativa.

1.4 Valoración del estado del arte:

Después de realizar un estudio a los sistemas contables antes mencionados donde se tuvieron en cuenta no sólo los aspectos fundamentales desde el punto de vista del software sino también del producto, se evidencia la no existencia de un software que responda cabalmente a lo que en realidad necesita la economía cubana, presentan diferentes inconvenientes ya sea en la de gestión de los costos, por las herramientas sobre las que están soportadas, por las licencias de dichos productos o bien porque no son capaces de cumplir con los requisitos establecidos a nivel nacional, sería entonces factible que el proyecto y el país en el proceso de lograr su soberanía tecnológica trabajaran sobre herramientas y tecnologías que no fueran propietarias, siendo una aplicación web y el trabajo con PHP lo más indicado para desarrollar un sistema de este tipo con el fin de lograr un alto grado de eficiencia, confiabilidad y rapidez en la gestión contable de las empresas.

1.5 Tendencias y tecnologías actuales:

La industria del software ha mostrado ser una de las áreas más dinámicas y con mayor crecimiento en los últimos años. La evolución hacia a un modelo más racional para los usuarios, con menos costes de licencia, donde se intensifique la prestación de servicios, que reduzca el tiempo de

desarrollo e incremente la calidad, viene siendo lo más importante a la hora de desarrollar cualquier tipo de aplicación. A continuación se ofrecerá una valoración sobre algunas de las tendencias y tecnologías que marcan un nivel alto en el mundo del software y que bien contribuye a lo dicho anteriormente:

1. **Aplicación Web:**

La creación de aplicaciones de este tipo ha tomado gran auge debido a las funcionalidades que ofrece; algunas de estas se presentarán a continuación:

➢ Compatibilidad Multiplataforma: Una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows y Linux.

➢ Menos requerimientos de hardware: Este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor donde reside la aplicación.

➢ Actualización*:* Una de las características fundamentales de las aplicaciones web es que siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones ni realizar tareas de instalación.

➢ Acceso inmediato y desde cualquier lugar: Las aplicaciones basadas en tecnología web no necesitan ser descargadas, instaladas o configuradas. Además, pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.

➢ Seguridad en los datos: Los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco. (11)

Cuando se desarrolla una aplicación web existe una tendencia al uso de Arquitectura en capas y del patrón de diseño: Modelo-Vista-Controlador (MVC).

2. **Arquitectura Cliente/Servidor:**

La arquitectura cliente/servidor no es más que un sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan. La mayoría de las

aplicaciones que se están desarrollando actualmente en la industria de software utilizan este tipo de arquitectura debido a las funcionalidades que brindan, entre ellas:

- Permite integrar y compartir información entre sistemas diferentes sin necesidad de que todos tengan que utilizar el mismo sistema operativo.
- Posibilita el mantenimiento y el desarrollo rápido de aplicaciones.(12)

La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
Se puede asociar el uso de esta arquitectura con los principios de Arquitectura Orientada a Servicios (SOA[7]).

3. **Software libre:**

El software libre; extendido prácticamente en el mundo; permite crear soluciones cada vez más sofisticadas y más personalizadas debido a la libertad de los usuarios para ejecutar, copiar, distribuir y mejorar el software, tiene entre sus ventajas fundamentales:

- Independencia tecnológica: El acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero.
- Libertad de uso y redistribución: Las licencias de software libre existentes permiten la instalación del software tantas veces y en tantas máquinas como el usuario desee. (14)

De ahí la repercusión que tiene en el ámbito informático hoy día, siendo una tendencia que va madurando cada vez más.

1.6 Modelo de desarrollo:

La propuesta de modelo de desarrollo que a continuación aparece fue elaborada por el equipo de producción en colaboración con las Líneas de desarrollo del proyecto ERP-Cuba de acuerdo con las

[7] SOA: Conjunto de servicios tanto de negocio como tecnológicos que interactuando entre ellos proporcionan la lógica necesaria para construir aplicaciones de una manera rápida y cumpliendo siempre con los principios de la Orientación a Servicios. SOA proporciona una serie de guías y recomendaciones para conseguir los objetivos que se impone una organización a la hora de desarrollar aplicaciones. (13)

necesidades presentadas por cada una de ellas y donde se tuvieron en cuenta los principales riesgos con los que se cuentan en el proyecto. (15)

1.6.1 Características:

- **Centrado en la arquitectura:**

La arquitectura determina la línea base y los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades en la producción y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

- **Orientado a componentes:**

Las iteraciones son orientadas según la significación arquitectónica de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

- **Iterativo e incremental:**

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

- **Ágil y adaptable al cambio:**

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

1.7 Arquitectura:

Uno de los elementos clave en todo proceso de desarrollo de software es el diseño de la Arquitectura. Básicamente sobre ella se sustentan todas las representaciones de la estructura general de la aplicación a desarrollar. En la medida que sea concebida la arquitectura basada en los principios de cohesión, utilidad y flexibilidad de los componentes se obtendrá un mejor acabado del producto.

"La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución". (16)

En otras palabras la arquitectura de una aplicación es la vista conceptual de la estructura de esta y donde se establecen los fundamentos básicos para que analistas, diseñadores y programadores trabajen en una línea común y así alcanzar los objetivos del sistema de acuerdo con las necesidades del cliente. La correcta definición de los estilos arquitectónicos a utilizar, patrones y mecanismos de diseño es la esencia de lo dicho anteriormente.

1.7.1 Características de la Arquitectura Base:

Para el desarrollo del Sistema CEDRUX se decidió adoptar la propuesta de Arquitectura Base definida por la línea de Arquitectura del proyecto ERP-Cuba conformada por las diferentes vistas y estilos arquitectónicos que serán especificados a continuación:

Vista de Sistema:
Propone las partes del software: componentes, conectores, las restricciones y las configuraciones de estas partes, se subdivide en tres vistas fundamentales:

- Vista de Componentes: Encargada de las definiciones de los tipos de componentes posibles a definir en el proyecto, de la especificación de sus características, así como de la composición estructural interna de cada uno de estos componentes.
- Vista de Integración: Encargada de los procesos de integración interna y externa, establece las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información.
- Vista de Datos: Encargada de todas las definiciones a nivel de datos, de la integración de los distintos modelos, de los patrones, estándares y definiciones a este nivel.

Vista Tecnológica:
Es la base del software, propicia los elementos necesarios para crear el producto, esta a su vez se subdivide en dos vistas:

- Vista de seguridad: Chequea e implementa todos los aspectos relacionados con el acceso a la aplicación, la modificación, lectura o eliminación de la información, etc.
- Vista de presentación: Encargada de cómo luce el software, cuáles son los colores que lleva la aplicación, cómo son los botones, los vínculos y todos los elementos significativos desde el punto de vista de la presentación.

Vista de Infraestructura:
Es la encargada de determinar la plataforma tecnológica a utilizar en la elaboración del producto, la definición y disponibilidad de los distintos servicios telemáticos necesarios en la confección del mismo, así como del diseño de los distintos escenarios de despliegue posibles. (17)

Estilos arquitectónicos:

- Arquitectura Basada en Componentes:

Uno de los enfoques en los que actualmente se trabaja es la arquitectura basada en componentes que tiene como objetivo hacer un uso correcto de software reutilizable, para la construcción de aplicaciones mediante el ensamblaje de partes ya existentes.

"Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y de requisitos, que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio" (18)

El equipo de arquitectura del proyecto ERP-Cuba dentro de la vista del sistema antes presentada incluye la vista de componente como una de sus subdivisiones; en ese caso propone que todas las funcionalidades levantadas y modeladas en las fases de negocio y requerimientos deben ser expresadas o contenidas en al menos un componente y que las distintas interacciones entre ellos originen funcionalmente la existencia de subsistemas en consecuencia a las dependencias definidas. Ejemplificado en el modelo de componentes del Subsistema Costos y Procesos que será presentado posteriormente en el epígrafe 2.3.2.

- Modelo-Vista-Controlador (MVC):

El patrón arquitectónico MVC es utilizado para el desarrollo de aplicaciones Web con el fin de separar en tres componentes distintos la interfaz de usuario, la lógica de negocio y los datos persistentes, potenciando la flexibilidad y la adaptabilidad a futuros cambios. Por su parte:
El Modelo: Es la representación de la información que maneja la aplicación. Son los datos puros que puestos en un contexto del sistema son mostrados al usuario por medio del Controlador, proveen de información al usuario o a la aplicación misma.
La Vista: Constituye la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En una aplicación web la "Vista " es la página HTML con contenido dinámico sobre el cual

el usuario puede realizar operaciones.
El Controlador: Se encarga 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. (19)

Para el desarrollo del Sistema Integral de Gestión de Entidades CEDRUX se decidió trabajar con este patrón, evidenciándose en los framework definidos para cada una de las capas como parte del MVC de cada componente dentro de la aplicación, es decir: para la Vista: Extjs-Framework, el cual es muy utilizado en el desarrollo de aplicaciones Web con tecnología AJAX, para el Controlador: Zend-Framework quien emplea específicamente el estilo Modelo- Vista- Controlador como base de su funcionamiento y para agilizar el acceso a datos en el Modelo se utilizó Doctrine, un potente y completo sistema ORM (Mapeo Objeto Relacional).

De forma general, según lo descrito con anterioridad, en la arquitectura del Proyecto ERP-Cuba los estilos arquitectónicos que se emplean no se pueden ver de forma independiente sino como un estilo híbrido que comprende numerosas ventajas:

- Desarrollos paralelos: en cada capa.
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo: es más sencillo cambiar un componente que modificar íntegramente una aplicación.
- Mayor flexibilidad: se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad.
- Alta escalabilidad: La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. (20)

1.8 Patrones:

En términos generales, un patrón es un cúmulo de información que proporciona respuesta a un conjunto de problemas similares en un contexto dado. Los patrones hacen la producción de software más resistente al cambio, establecen parejas problema-solución, ayudan a especificar interfaces, facilitan la reutilización del código y permiten una fácil comprensión debido a la documentación estándar que presentan.

Patrones de diseño:

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Patrones **GRASP**: Patrones de Software para la Asignación General de Responsabilidad. En este caso las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento:
Conocer: Conocer los datos privados encapsulados, conocer los objetos relacionados, conocer las cosas que puede derivar o calcular.
Hacer: Hacer algo él mismo, como crear un objeto o hacer un cálculo, iniciar una acción en otros objetos, controlar y coordinar actividades en otros objetos.

GRASP destaca 5 patrones principales: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador.

Experto: Este patrón es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para ejecutar la tarea. Refuerza el encapsulamiento y esto redunda en bajo acoplamiento.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la instanciación o creación de nuevas clases u objetos. La clase podrá crear la nueva instancia si y sólo si tiene en cuenta al menos uno de los siguientes criterios:

- Tiene la información necesaria.
- Usa directamente las instancias creadas del objeto.
- Almacena o maneja varias instancias de la clase.
- Contiene o agrega la clase.

La visibilidad entre la clase creada y la clase creadora es una de las facilidades que se deriva del uso de patrón y además conduce a un bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización así como mayor claridad.

Alta cohesión: Indica que la información que almacena una clase debe ser coherente, de manera que todos sus métodos tengan un comportamiento bien definido.

- Las Clases se pueden reutilizar con mayor facilidad y flexibilidad.

- Una Clase con baja cohesión hace muchas cosas no relacionadas.
- Una Clase con alta cohesión hace lo que uno podría esperar que hiciera.

Si el sistema fallara por alguna razón es mucho más fácil encontrar responsabilidades si las Clases del sistema son cohesivas.

Bajo acoplamiento: Patrón evaluativo que asigna responsabilidades de modo que se mantenga un engranaje pobre entre las clases y objetos, reduce el impacto de los cambios y aumenta de reutilización.

Controlador: Asignar la responsabilidad a una clase de manejar mensajes correspondientes a eventos en un sistema. Encargada de recibir los datos del usuario y enviarlos a las distintas clases según el método llamado a modo de intermediario entre una determinada interfaz y el algoritmo que la implementa. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización de código y a la vez tener un mayor control. (21)

Patrones **GOF**:

Fachada: Patrón estructural que permite proveer una interfaz unificada y sencilla como intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas.

Mediador: Patrón de comportamiento que coordina las relaciones entre sus asociados. Permite la interacción de varios objetos, sin generar acoples fuertes en esas relaciones.

Cadena de Responsabilidad: La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición. (22)

1.9 Lenguajes de modelado y desarrollo:

1.9.1 Lenguaje de modelado:

Se denomina lenguaje de modelado de objetos al conjunto estandarizado de símbolos y las distintas combinaciones de la disposición para modelar un diseño de software.

- **UML:**

Lenguaje que permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Se estará haciendo uso del mismo para todo lo que

comprende el diseño del componente.

1.9.2 Lenguajes de programación:

El término lenguaje de programación está dado por un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen la estructura, el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

1.9.2.1 Lenguaje del lado del servidor:

Se clasifica así al lenguaje de programación en la tecnología cliente servidor que se ejecuta del lado del servidor y del cual los usuarios solo obtienen el beneficio del procesamiento de la información.

- **Lenguaje PHP:**

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. No requiere definición de tipos de variables, no es un lenguaje de marcas. Su interpretación y ejecución se realizan en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle, Microsoft SQL Server, entre otras, lo cual permite la creación de aplicaciones web muy robustas.
PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux y Windows.

Ventajas:

1. Es un lenguaje multiplataforma.
2. Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.
3. Capacidad de expandir su potencial utilizando una enorme cantidad de módulos.
4. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las

funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

5. Es libre, lo que representa que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.
6. Permite las técnicas de Programación Orientada a Objetos[8].
7. Biblioteca nativa de funciones sumamente amplia e incluida.
8. No requiere definición de tipos de variables.
9. Tiene manejo de excepciones.

Desventajas:

1. No posee una abstracción de base de datos estándar, sino bibliotecas especializadas.
2. Por sus características promueve la creación de código desordenado y complejo de mantener.
3. Todo el trabajo lo realiza el servidor, por tanto, puede ser más ineficiente a medida que aumenten las solicitudes.
4. La orientación a objetos es aún muy deficiente para aplicaciones grandes. (23)

A partir de la decisión de la dirección del proyecto y del equipo de arquitectura, lo más factible para la implementación es la utilización del lenguaje PHP debido a las facilidades ya presentadas, la experiencia alcanzada sobre el mismo en el desarrollo de aplicaciones web y la posibilidad de utilizarlo libremente y comercializar los productos realizados con él.

En este caso se estará haciendo uso de PHP en su versión 4.0 o superior, con los siguientes módulos o extensiones: pdo, pdo_pgsql, pgsql, soap, gd2, mbstring, mysql, mysqli, pdo_sqlite, sqlite, xsl.

1.9.2.2 Lenguajes del lado del cliente:

Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

- **HTML:**

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para

[8] La terminología Programación Orientada a Objetos (según siglas en inglés POO u OOP) representa un paradigma de programación donde se definen los programas en términos de "clases de objetos", tales objetos son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto).

describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. (24)

- **XML:**

Es el metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos. No es un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. (25)

- **JavaScript:**

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM[9]. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape y Mozilla Firefox. (26)

1.10 Frameworks:

Un framework, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro

[9] DOM: El Modelo de Objetos de Documento (en inglés Document Object Model), Es una plataforma que proporciona un conjunto estándar de objetos, que permite crear documentos HTML y XML, navegar por su estructura, modificar, añadir y borrar tanto elementos como contenidos. No se ajusta a un lenguaje de programación específico, lo que facilita el diseño de páginas web activas, de manera tal que proporciona una interfaz estándar para que otro software manipule los documentos.

proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.10.1 Ext:

Librería construida empleando JavaScript con el fin de desarrollar aplicaciones web interactivas usando tecnologías como AJAX y DOM. Ext es muy potente ya que contiene rica colección de componentes para el diseño de Interfaces Gráficas de Usuario (GUI's) del lado del cliente haciendo uso extensivo de AJAX. Dispone de un conjunto de componentes gráficos como:

1. Cuadros y áreas de texto.
2. Campos para fechas.
3. Campos numéricos.
4. Selectores estáticos y dinámicos.
5. Botones.
6. Editor HTML.
7. Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
8. Árbol de datos.
9. Pestañas.
10. Barra de herramientas.
11. Menús al estilo de Windows. (27)

Se estará haciendo uso de Ext en su versión 2.2.

1.10.2 Zend:

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP.

Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernos, robustos y seguros sitios web, está diseñado para php 5 y posee buenas capacidades de ampliación. Presenta entre otras, las siguientes características:

1. Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos.
2. Proporcionan los componentes que forma la infraestructura del patrón MVC.
3. Proporciona una capa de acceso a base de datos, construida sobre PDO[10] pero ampliándola con diferentes características.
4. Proporciona mecanismos de filtrado y validación de entradas de datos.
5. Permite convertir estructuras de datos PHP a JSON[11] y viceversa, para su utilización en aplicaciones AJAX (especificado en el epígrafe 1.9.1).
6. Proporciona capacidades de búsqueda sobre documentos y contenidos. (30)

Se estará haciendo uso de Zend Framework en su versión 1.9.7.

1.10.3 Doctrine:

Doctrine es un potente y completo sistema ORM[12] (en inglés Object Relational Mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto.
Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Funcionalidades:

1- Exporta una base de datos existente a sus clases correspondientes.
2- Convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (32)

Se estará haciendo uso de Doctrine en su versión 1.2.1.

1.11 Tecnologías y Herramientas de desarrollo:

Para la realización de un proyecto de esta magnitud es necesario que cada uno de los equipos de desarrollo posea un modelo estandarizado de las tecnologías y herramientas a utilizar conjuntamente

[10] PDO: (en inglés: PHP Data Objects) es una interface de acceso a datos que permite la conexión a diferentes bases de datos utilizando tecnología orientada a objetos. (28)

[11] JSON: es un formato ligero para el intercambio de datos. (29)

[12] ORM: Componente de software que permite trabajar con los datos persistidos como si fueran parte de una base de datos orientada a objetos. (31)

con sus versiones para la implementación de las capas de presentación, negocio y acceso a datos.

De acuerdo con lo planteado anteriormente la dirección del proyecto determinó emplear:

1.11.1 Tecnologías:

- **Tecnología AJAX:**

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.
Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.
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. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. (33)

1.11.2 Herramienta CASE:

- **Visual Paradigm:**

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Entre sus principales características podemos encontrar:

1. Soporta aplicaciones Web.
2. Es un producto de calidad.
3. Fácil de instalar y actualizar.
4. Compatibilidad entre ediciones. (34)

Se estará haciendo uso de Visual Paradigm en su versión 6.4.

1.11.3 Herramienta de desarrollo colaborativo:

- **Control de versiones:**

Una versión, revisión o edición de un producto, es el estado en el se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios

que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos. Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

Subversion:

También conocido como SVN, es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red y se distribuye bajo licencia libre.

1. Mantiene versiones no sólo de archivos, sino también de directorios
2. Mantienen versiones de los metadatos asociados a los directorios.
3. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
4. Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
5. Soporte tanto de ficheros de texto como de binarios.
6. Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (35)

Se estará haciendo uso de un cliente Subversion: TortoiseSVN-1.4.5.

1.11.4 Entorno integrado de desarrollo:

Un entorno de desarrollo integrado o IDE (en inglés: Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación que permite de

forma cómoda y ágil editar, compilar, ejecutar y depurar programas.

- **Zend Studio para Eclipse:**

Es un IDE (Entorno integrado de desarrollo) para el lenguaje de PHP escrito en Java destinado a desarrolladores profesionales, propietario, compatible con las plataformas Linux, MAC y Windows. Incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Zend Studio permite agilizar el desarrollo web y permite simplificar proyectos complejos. Posibilita un excelente completamiento de código, coloreado en la sintaxis del código, administración avanzada de proyectos, múltiples lenguajes, incorpora el Framework de Zend, integración con subversión, integración avanzada con FTP, soporte para Web Services, PHP4 y PHP5, entre otras características están:

1. No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
2. Detección de errores de sintaxis en tiempo real.
1. Manual de PHP integrado.
2. Ofrece soporte básico para otros lenguajes web, como HTML, JavaScript y XML.
3. Acceso al paquete de plug-ins de Eclipse.
4. Mecanismo de actualización automática. (36)

Se estará haciendo uso de Zend Studio para Eclipse en su versión 3.3.

1.11.5 Servidor de Aplicaciones web:

Es un programa que permite crear un servidor http en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Es además un programa que implementa el protocolo HTTP el cual se encarga de transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

- **Servidor web Apache:**

Es una tecnología gratuita de código fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable es decir se pueden elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. (37)

Se estará haciendo uso del Servidor web Apache en su versión 2.0 ó superior.

1.11.6 Sistema Gestor de Base de Batos:

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

- **PostgreSQL:**

PostgreSQL es un servidor de base de datos relacional[13], libre. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que es multiplataforma. Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas procedimientos almacenados en el servidor, transacciones, almacenamiento de objetos de gran tamaño y además tiene ciertas características orientadas a objetos. (39)

Se estará haciendo uso de PostgreSQL en su versión 8.3 ó superior e inferior a 8.4

1.11.7 Navegador:

Software que permite al usuario recuperar y visualizar documentos de hipertexto desde servidores web a través de Internet.

- **Mozilla Firefox:**

Mozilla Firefox es el nuevo e innovador navegador open source. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. Es Software libre. Se trata de un práctico y ágil navegador, que está en renovación constante. Tiene la capacidad de modificarlo totalmente a gusto del usuario y según las necesidades del mismo. Algunas características de Mozilla Firefox son:

1. Navegación por tabs, esta es una de las principales características que tiene Firefox, es posible

[13] Base de datos relacional: Modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente basados en el empleo de relaciones entre las tablas. (38)

navegar por pestañas.

2. Es Software libre.

3. Trabaja de forma excelente en computadoras sin hardware muy potente, el programa está diseñado para realizar un bajo consumo de recursos.

4. Soporta Windows (en cualquiera de sus versiones), Linux (Desde la versión 2.2 del Kernel) y Mac (Desde Mac OS X 10.1.x a la 10.2.x y nuevas versiones). (40)

Se estará haciendo uso de Mozilla Firefox en su versión 2.0.1 ó superior.

1.12 Conclusiones del capítulo:

La Gestión de los Costos, específicamente el Ajuste al Costo es un proceso fundamental para la planificación y la organización de los recursos empresariales en las entidades cubanas; este capítulo fue esencial para valorar el estado actual del mismo a partir de un análisis de sistemas contables tanto nacionales como internacionales, evidenciando la no existencia de un sistema informático capaz de ejecutar tales funcionalidades ni de cumplir con los requisitos establecidos a nivel nacional.

Este capítulo fue básico para la definición de un conjunto de conceptos fundamentales asociados al dominio del problema. Se explicó el modelo de desarrollo a utilizar. Se analizaron las tendencias y tecnologías actuales para el desarrollo de aplicaciones informáticas y por último se realizó una presentación de las tecnologías y herramientas conjuntamente con sus versiones, propuestas por la dirección del proyecto para el desarrollo de la solución.

Capítulo 2: Diseño e Implementación

2.1 Introducción:

Transformar los requisitos funcionales en el diseño del futuro componente para una posterior implementación a partir de la arquitectura previamente definida por la dirección del proyecto, será el punto de partida del presente capítulo. En consecuencia, se obtendrán un conjunto de artefactos que serán de gran valor para las posteriores etapas del desarrollo como lo son: el modelo de componentes, el diseño de clases que comprende diagramas y descripción de las mismas y el modelo de datos. Igualmente se especificará la utilización de un conjunto de patrones dentro del diseño del componente.

El capítulo comprenderá además elementos fundamentales de la implementación del componente, en ese caso se presentan: la organización del marco de trabajo, la definición de los estándares de codificación y la estructura de datos a utilizar, la descripción de clases y algoritmos implementados, la representación de la estrategia de integración y por último la descripción general del funcionamiento del componente.

2.2 Diseño de la solución:

2.2.1 Valoración crítica de la Especificación de Requisitos:

El artefacto Especificación de Requisitos del Software para el Subsistema Costos y Procesos como resultado del previo análisis efectuado al proceso Ajuste al Costo, constituye un elemento clave para el diseño y la posterior implementación.

Básicamente se caracteriza por presentar los requisitos de forma completa, estando definidas todas las responsabilidades del sistema respecto a los datos de entrada, válidos o no y respecto a los datos de salida. Presenta una adecuada organización y documentación; 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, están clasificados por la importancia arquitectónica y desde el punto de vista del cliente. Viabiliza las modificaciones, la comprobación y la trazabilidad (origen- implementación) de los requisitos especificados.

Según lo descrito con anterioridad, la Especificación de Requisitos presentada por los analistas puede ser tomada como artefacto de entrada al Diseño para facilitar la comprensión del equipo de desarrollo en general.

A continuación se presentan los requisitos funcionales del proceso Ajuste al Costo:

Requisitos funcionales correspondientes al paquete Configurar Destinos:

1. Seleccionar concepto de inventario para destinos.
2. Guardar la configuración.
3. Cancelar la configuración.

Requisitos funcionales correspondientes al paquete Ajustar Costos:

1. Definir Producto.
2. Calcular Producción Equivalente.
3. Calcular Costo Unitario Real.
4. Calcular Ajuste al Costo.
   - 4.1 Ejecutar Ajuste (Se ajusta a inventario y se hace por unidades físicas).
     - 4.1.1 Mostrar Comprobante de operaciones.
     - 4.1.2 Mostrar Documento de inventario.
   - 4.2 Ejecutar Ajuste (No se ajusta a inventario y se hace por unidades físicas)
   - 4.3 Ejecutar Ajuste (Se ajusta a Inventario y se hace por importe monetario).
     - 4.3.1 Mostrar Comprobante de operaciones.
     - 4.3.2 Mostrar Documento de inventario.
   - 4.4 Ejecutar Ajuste (No se ajusta inventario y se hace por importe monetario).
5. Mostrar datos del Ajuste.
6. Mostrar datos a ajustar en Existencias.
7. Mostrar datos a ajustar en Destinos.

Requisitos funcionales correspondientes al paquete Ajustes Realizados:

1. Mostrar Ajustes realizados.
2. Cancelar Ajustes realizados.

2.2.2 Diseño de la solución en términos de componentes:

El Subsistema Costos y Procesos perteneciente al Sistema Integral de Gestión de Entidades CEDRUX es el encargado de automatizar toda la gestión de los costos en la entidad que lo utilice, permitiendo medir la suma de gastos de toda naturaleza expresada monetariamente, que se aplica a una producción o servicio determinado de la entidad que lo utilice.

Se presentará el modelo de componentes propuesto para el Subsistema Costos y Procesos, donde se relacionan a través de interfaces de comunicación un conjunto de componentes definidos para dar

solución arquitectónica al sistema entre los cuales se encuentran: Nomencladores y Configuración, Cierre-Traspasos, Recuperación de Información y el componente a desarrollar Ajuste al Costo.

Modelo de componentes para el Subsistema Costos y Procesos:

Un modelo de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan.

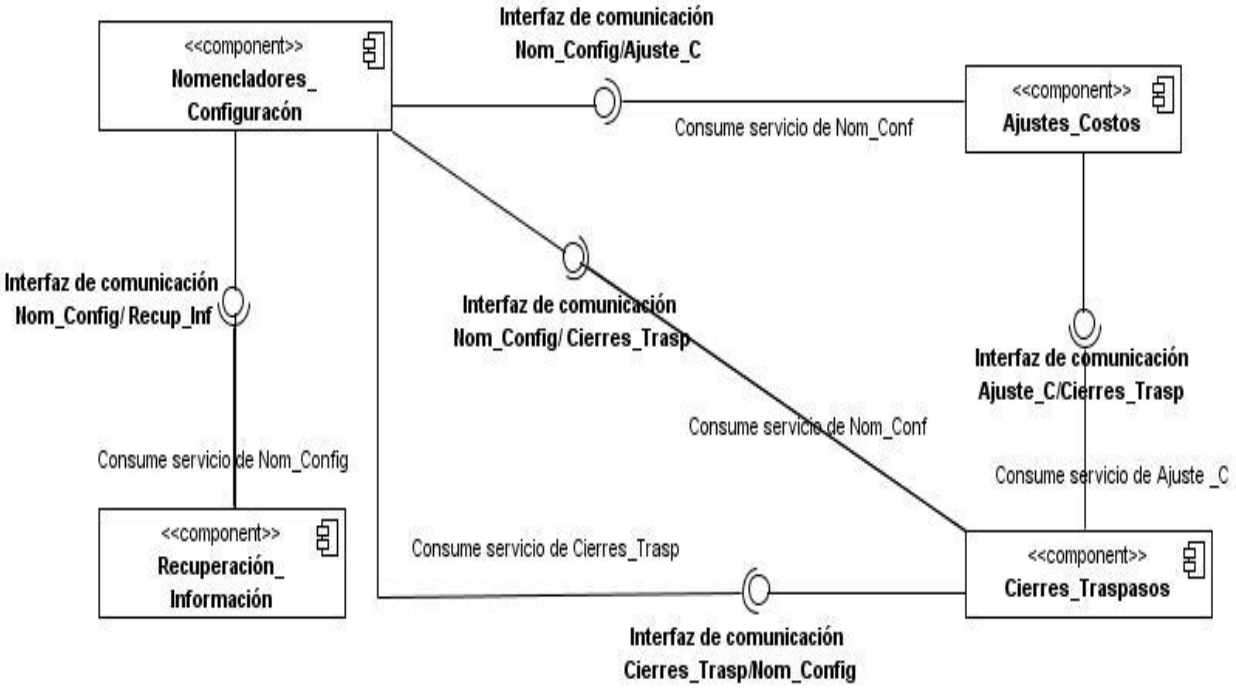


Figura 1 Modelo de componentes del Subsistema Costos y Procesos.

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

Nomencladores y Configuración:

Tiene a cargo la gestión de los Elementos de Gastos, Centros de Costo y las Cuentas de Gastos; ya sean patrimoniales o presupuestadas. Permite las asociaciones entre Centro–Cuenta–Elemento de Gasto y Centro–Área de Responsabilidad. Brinda un conjunto de servicios a los restantes componentes del Subsistema.

Cierres y Traspasos:

Componente donde se gestionan los traspasos de los saldos de las cuentas de gastos indirectos

hacia las cuentas que gestionen los gastos directos. De la misma manera que consume servicios de Nomencladores y Configuración, le ofrece los centros y elementos que tiene operaciones contables realizadas.

Recuperación de Información:

Es el componente encargado de la generación de los reportes estadísticos de la gestión de los costos a través de la recopilación de la información que brinda Nomencladores y Configuración. Brinda el anexo al costo de venta, hace reportes de operaciones, de asociaciones y de los Submayores de Procesos, de Gastos por Subelementos y de Gastos por Partida.

Ajuste al Costo:

Componte que debe ser diseñado e implementado. El ajuste de los costos predeterminados es la esencia del mismo y para ello debe tener en cuenta una previa Configuración de los Destinos de los productos. Es necesaria su integración con la gestión de los inventarios.

Debe consumir los servicios que brinda Nomencladores y Configuración, específicamente: las Cuentas de Procesos, los Centros de Costos y sus cuentas asociadas, a partir de las cuales se ajustan los costos iniciales basados en los gastos reales de la producción al final de cada período.

2.2.3 Diseño de clases:

2.2.3.1 Diagramas de Clases del Diseño:

Los diagramas de clases según la clasificación UML son diagramas de estructura estática donde la representación de los requerimientos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema.

Específicamente los diagramas de clases de diseño son muy útiles porque muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación (PHP en este caso).

- **Diagrama de Clase del Diseño del proceso Configurar Destinos:**

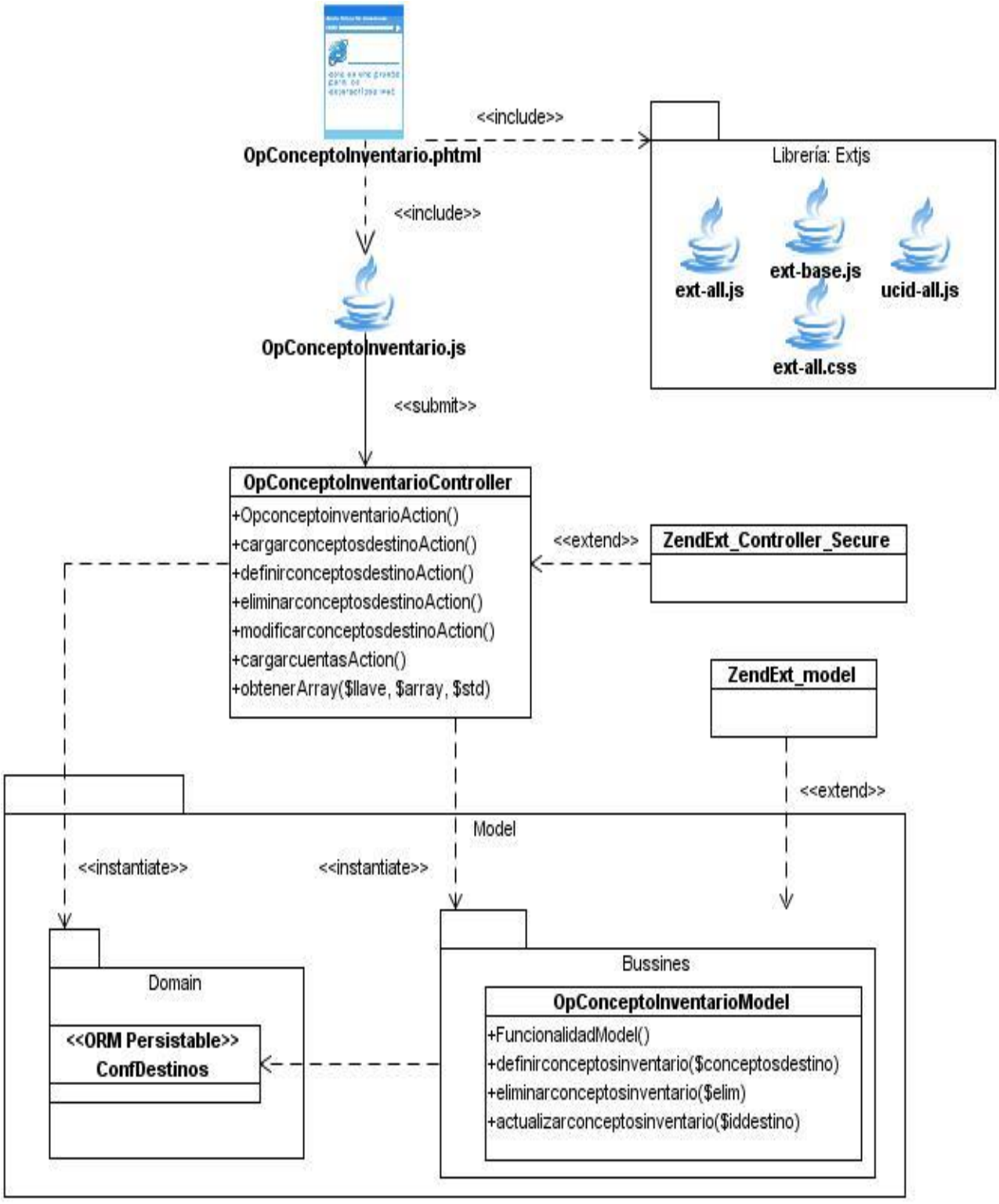


Figura 2 Diagrama de Clases de Diseño del proceso Configurar Destinos.

Descripción del diseño de clases del proceso Configurar Destinos:

Tabla 1 Descripción del diseño de clases del proceso Configurar Destinos.

Clases	Descripción
Librería Extjs	Contiene los componentes generados a través de la librería JavaScript Extjs.
OpConceptoInventario.phtml	Página encargada de visualizar, a través de los js que debe incluir, la información relacionada con la configuración de los destinos, como una forma de salida de los productos.
OpConceptoInventario.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería Extjs los componentes que manejen la información en la vista necesaria para seleccionar los destinos de los productos. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
OpConceptoInventarioController	Clase controladora responsable de configurar los destinos donde se deben ejecutar las diferentes funcionalidades, según las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.

2.2.4 Modelo de datos:

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. (41)

El modelo de datos propuesto en la solución cuenta con un total de 12 tablas. Para su construcción se tuvo en cuenta la reducción a la mínima expresión de los campos nulos y la persistencia de

campos resúmenes para agilizar recuperaciones frecuentes de algunos datos que son complejos de calcular.

- Teniendo en cuenta el requisito funcional Configurar Destinos se crea la tabla conf_destinos, en la misma se guardan los datos de los destinos configurados como forma de salida de los productos a partir de las operaciones definidas previamente.

- Según el requisito funcional Ajustar Costos se especificaron las siguientes tablas:

En primer lugar, la tabla conf_ajustecosto donde se guardan los datos generales de la configuración del ajuste, la misma está relacionada con las tablas conf_ajusteproducto y dat_saldoantes donde se registran los productos para los cuales se va a efectuar el ajuste y el saldo inicial de la cuenta sobre la cual se efectuará la operación respectivamente.

En segundo lugar, la tabla conf_calcularajuste donde se almacena la información correspondiente al cálculo del ajuste, dicha tabla se relaciona con nom_basedistribucion, conf_calcajusteinv y conf_calcajustenoinv para conocer de manera que se corresponda la Base de distribución de los productos y el Modo de ajuste en dependencia de la selección realizada.

Una vez seleccionada la opción ajustar costos se inserta en las tablas dat_ajustedestinoantes donde se guarda la información de los productos ajustados en destinos, dat_ajustexistenciantes para los productos ajustados en existencia y en la tabla dat_saldodespues donde es recogida la información referente al saldo de la cuenta después de efectuado el ajuste.

Las tablas conf_ajustecosto y conf_calcularajuste se relacionan a través de una tabla intermedia denominada conf_ajustecalcajuste donde se guarda el histórico del ajuste realizado atendiendo a la configuración, cálculo y datos generales del mismo.

- Para mostrar los ajustes efectuados en un período, como parte del requisito funcional Ajustes Realizados, se consulta la tabla conf_ajustecalcajuste y a partir de ella son referenciadas otras tablas que de igual manera contienen la información que se debe visualizar.

El modelo de datos es presentado a continuación:

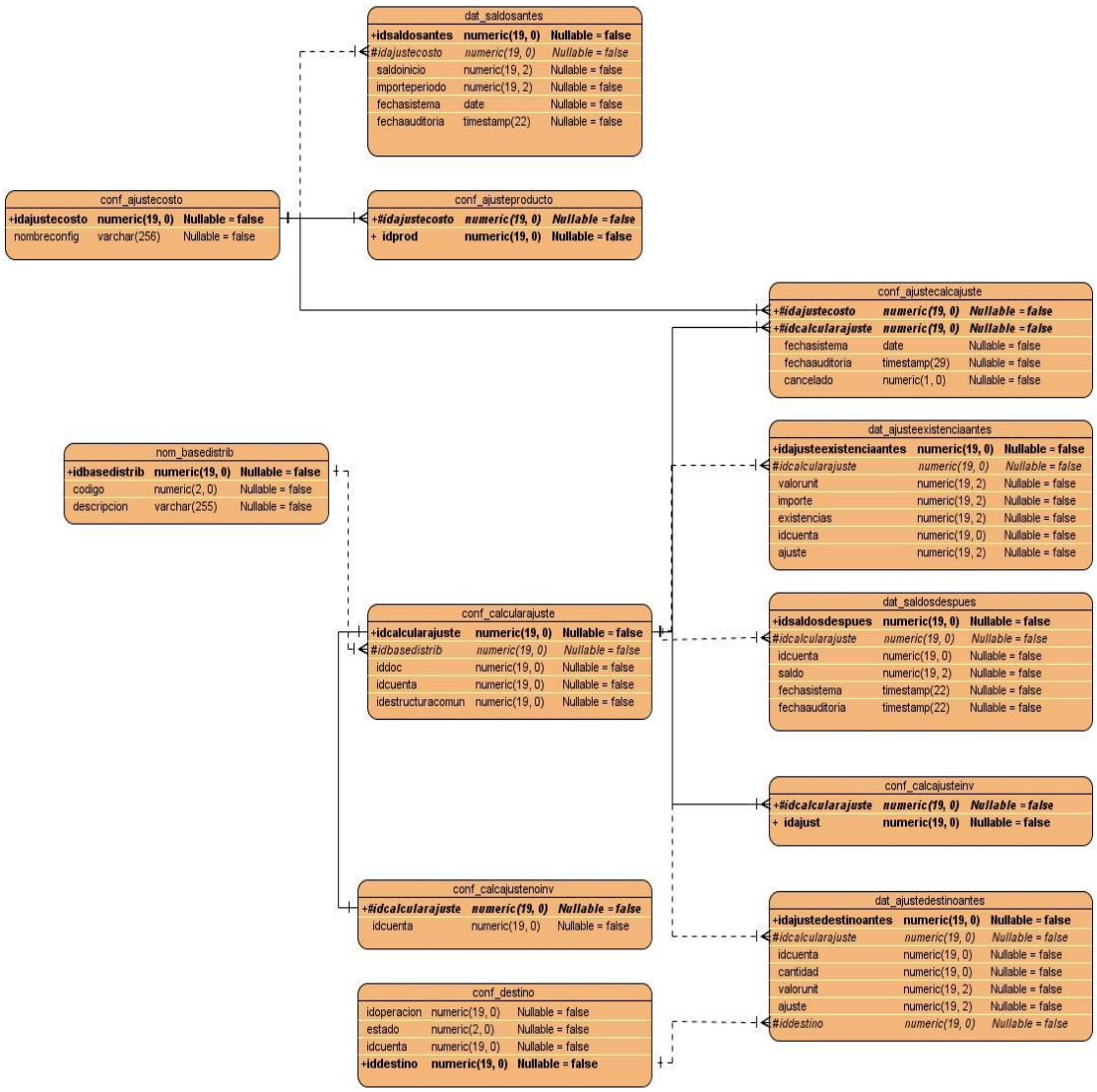


Figura 3 Modelo de datos del Componente Ajuste al Costo.

2.2.5 Patrones de diseño empleados:

El diseño fue elaborado siguiendo patrones basados en la experiencia, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. En este caso se emplearon los patrones GRASP (en ingles General Responsability Assignment Software Patterns), los que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones GRASP que se utilizaron son los siguientes:

Experto: Dicho patrón es evidenciado en la definición de las clases de acuerdo a las funcionalidades

que deben realizar a partir de la información manejada dentro del componente, como por ejemplo las clases controladoras y las del modelo. Específicamente: la clase OpconceptoInventarioModel, en la configuración de los destinos de los productos, será la responsable de efectuar las operaciones que conciernen a las funciones: definir, eliminar y actualizar los conceptos de inventario, asumiendo toda la lógica para cada una de ellas. Sobre este mismo principio se realiza el diseño de las restantes 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.

Alta cohesión: Este patrón fue utilizado en el diseño del componente de manera general; donde se agruparon las clases en dependencia de los requerimientos (Configurar Destinos, Ajustar Costos y Ajustes Realizados) a los que se les debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Bajo acoplamiento: En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

Controlador: Las clases controladoras definidas: OpAjustarCostosController, OpAjustesRealizadosController, OpConceptoInventarioController son un ejemplo de la aplicación de este patrón, las mismas tendrán a cargo la responsabilidad de manejar los eventos dentro del componente.

Durante el diseño del componente se emplearon patrones GOF, específicamente:

Fachada: La aplicación de este patrón en el componente Ajuste al Costo se evidencia en la interfaz de servicios simple que se proporciona para establecer la comunicación con otros componentes dentro y fuera del Subsistema.

Mediador: La comunicación en la base de datos se puede tornar compleja debido a las dependencias marcadas entre las tablas que la componen, esto resulta engorroso a la hora de acceder a un determinado valor. La solución a este inconveniente viene dada por la utilización de este patrón; específicamente; creando una nueva tabla entre todas las tablas unidas mediante una relación de

muchos a muchos. La tabla mediadora posee una relación de uno a muchos con las vinculadas a ella. De esta forma, el comportamiento distribuido entre las clases queda adaptado a las circunstancias y necesidades del diseño.
Dicha operación se concreta en el modelo de datos antes presentado entre las tablas asociadas a Conf_AjusteCosto y Cof_CalcularAjuste donde la tabla mediadora resultó ser Conf_AjusteCalcAjuste.

Cadena de Responsabilidad: Está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo. De esta manera se distribuyen las responsabilidades entre las diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.

La Arquitectura Base y el diseño flexible y escalable a través del correcto uso de patrones de diseño en la generación de los artefactos necesarios para el desarrollo, posibilitaron crear una entrada apropiada como punto de partida a las actividades de implementación, con la máxima de lograr una mayor calidad del producto y la satisfacción del cliente.

2.3 Implementación del componente:

2.3.1 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, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa.
Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito. Partiendo de lo dicho anteriormente, se definen 3 partes principales dentro de un estándar de programación:

1. Nomenclatura de las clases:

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que 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 sólo leerlo se reconoce el propósito de la misma.
Ejemplo: ContabilizarAjusteCosto. En este caso el nombre de clase esta compuesto por 3 palabras iniciadas cada una con letra mayúscula.

1.1 Nomenclatura según el tipo de clases:

Clases controladoras: Las clases controladoras después del nombre llevan la palabra:"Controller". Ejemplo: OpAjustarCostosController
Clases de los modelos:
Business (Negocio): Las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model". Ejemplo: ContabilizarAjusteCostoModel.

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

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: BaseConfDestino.

2. Nomenclatura de las funcionalidades y atributos:

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

Ejemplo de método: calcularAjuste. El nombre de método esta compuesto por 2 palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.
Las principales funcionalidades de las clases controladoras se les pone el nombre y seguida la palabra:"Action" Ejemplo: calcularAjusteAction ().

Ejemplo de atributo: unidadesFisicas. El nombre del atributo esta compuesto por 2 palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

3. Nomenclatura de los comentarios:

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de

lo que se está desarrollando. En caso de ser una función complicada se debe comentar para lograr una mejor comprensión del código.

2.3.2 Estructura de datos a utilizar:

Se define como estructura de datos en programación, a la forma de organizar un conjunto de datos elementales con el fin de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema. Existen diversas estructuras que en sí poseen ventajas y desventajas según la simplicidad y eficiencia para la realización de cada operación. Sin embargo a la hora de elegir la estructura de datos más conveniente es preciso evaluar factores básicos como la frecuencia y el orden en que se realiza cada operación sobre los datos.

Durante la realización de una sintaxis de código en PHP; lenguaje definido para la implementación de los diferentes componentes de la aplicación debido a las facilidades que brinda; aparecen variables que tienen información similar y que se procesan de forma semejante, concretamente, se está hablando de los arrays como estructura de datos.
Un array o como también se le conoce: arreglo, es un conjunto de variables (elementos) agrupadas bajo un único nombre en distintas casillas.

En dicho lenguaje de programación existen dos tipos de arreglos: los de índices numéricos y los de índices asociativos, ambos poseen una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, para contar el número de elementos que comprende y movernos por dentro de él hacia delante o atrás. A partir de lo antes descrito y por decisión del equipo de arquitectura esta vendría siendo la estructura de datos a utilizar de forma general para el desarrollo en todos los subsistemas.

2.3.3 Descripción de las clases y funcionalidades del componente.

Clases Controladoras:

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.

Tabla 2 Descripción de la clase OpConceptoInventarioController.

Nombre: OpConceptoInventarioController
Tipo de clase: Controladora

Atributo	**Tipo**
idestructura	int
Para cada responsabilidad:	
Nombre:	**Descripción**:
opConceptoInventarioAction()	La función muestra la vista que permite configurar los destinos.
cargarConceptosDestinoAction()	Permite cargar todos los conceptos de destino teniendo en cuenta una estructura.
definirConceptosDestinoAction()	La función permite definir en la base datos los destinos seleccionados según los identificadores de cuenta y operación, el estado y la estructura.
eliminarConceptosDestinoAction()	La función debe eliminar en la base datos los conceptos de inventarios seleccionados.
modificarConceptosDestinoAction()	La función permite modificar los datos de un concepto de inventario en la base de datos.
cargarCuentasAction()	Se obtienen de Contabilidad dado una estructura todas las cuentas hijas a partir de un nodo específico.
encabezadoAction()	Muestra la información referente al Ejercicio y Período contable para el cual se efectuará el ajuste, así como la Fecha del sistema.

Clases Auxiliares:

Son las clases que guardan poca o ninguna información del estado por sí misma, pero asisten en la ejecución de tareas complejas.

Tabla 3 Descripción de la clase ContabilizarAjusteCosto.

Nombre: ContabilizarAjusteCosto
Tipo de clase: Auxiliar

Atributo	Tipo
paseObjeto	Array
paseCta	Array
Para cada responsabilidad:	
Nombre:	**Descripción:**
Contabilizar($idcalcularajuste,$contabilizar)	Permite conformar un comprobante de los datos del ajuste.
conformarPases($parámetro)	Organiza las cuentas por origen y destino y dentro de ellas por débito y crédito, para conformar los pases y los registros anexos al pase del comprobante.

Clases Model:

Estas clases manejan la información persistente que poseen una larga vida, conceptos y sucesos que ocurren en el mundo real.

Tabla 4 Descripción de la clase OpConceptoInventarioModel.

Nombre: OpConceptoInventarioModel	
Tipo de clase: Model	
Para cada responsabilidad:	
Nombre:	**Descripción:**
definirConceptosInventario($conceptosdestino)	Permite definir un nuevo concepto de inventario.
eliminarConceptosInventario($elim)	Permite eliminar un determinado concepto de inventario.
actualizarConceptosInventario($iddestino)	Permite modificar un determinado concepto de inventario.

Domain:

Tabla 5 Descripción de la clase ConfDestino.

Nombre: ConfDestino	
Tipo de clase: Domain	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
devolverdestinos($idestructura)	Devuelve los datos de los destinos dado una estructura determinada.
obtenerOperacionPorId($idoperacion, $estado, $idestructura)	Permite obtener los datos de una configuración de destino existente dado el id del mismo y la estructura a la que pertenece.

2.3.4 Estrategia de integración:

En cada uno de los componentes del sistema el flujo de datos que va desde la vista hacia el modelo y viceversa, responde completamente a una estrategia de integración vertical concebida sobre 4 nodos y a partir de cada uno de los elementos arquitectónicos definidos.
El primer nodo se sitúa entre la vista y el controlador, el segundo está entre el controlador y el modelo, el tercero vincula el modelo con el framework doctrine y el último se encuentra entre la base de datos y el doctrine.

Vista – Controlador: Los datos recogidos en un formulario son enviados al Controlador haciendo uso del protocolo de comunicación HTTP a través del método “post” para ser procesados y los resultados son enviados por el controlador a la vista en un JSON a través del método “echo”.

Controlador – Modelo: El Controlador toma los datos recibidos desde la vista, instancia una determinada clase del modelo y llama a uno de sus métodos, pasándole como parámetros los datos recibidos.

Modelo – Doctrine: El Modelo utiliza llamadas a métodos de Doctrine que le permitan crear, modificar, eliminar o actualizar los datos almacenados en las tuplas de la base de datos.

Doctrine – Base de Datos: Doctrine ejecuta las consultas a la Base de Datos utilizando programación orientada a objetos.

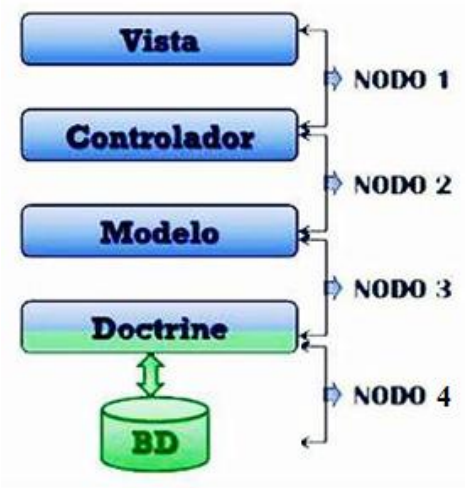


Figura 4 Nodos involucrados en la integración.

La comunicación dentro de un mismo componente se ejecuta de forma directa, sin embargo, la que se establece entre los diferentes componentes y subsistemas de la aplicación va mas allá de un simple llamado a un servicio, esta se basa en el empleo de un registro de datos de los subsistemas contenidos en un fichero xml mapeado por el framework para el funcionamiento del componente llamado: inversión de control (IoC). El IoC registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema, especifica respuestas deseadas a sucesos o solicitudes de datos concretas, orden necesario y el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio.

Servicios que recibe el componente como parte de la integración:

- **obtenercentrosp:** Servicio que brinda el componente Nomencladores y Configuración del Subsistema Costos y Procesos que le permite al componente Ajuste al Costo obtener los Centros de costos patrimoniales asociados a una Cuenta determinada.

- **obtenerelemnom:** Servicio que brinda el componente Nomencladores y Configuración del Subsistema Costos y Procesos que devuelve los elementos específicos de un nomenclador.

- **obtenercuentaspadrespatr:** Servicio que brinda el componente Nomencladores y Configuración del Subsistema Costos y Procesos que permite conocer las cuentas padres patrimoniales dado una estructura determinada.

- **ObtnerIdOperacionDadoIdEstructura:** Servicio que brinda el Subsistema de Logística que permite conocer el identificador de una operación perteneciente a una estructura determinada.

- **cantProductosMov:** Servicio que brinda el Subsistema de Logística que permite conocer la cantidad los productos que fueron movidos a destinos.

- **cantProductosAlmacen:** Servicio que brinda el Subsistema de Logística que permite conocer el total de productos que se encuentra en el almacén dado una estructura determinada.

- **ObtenerCuentasPorId:** Servicio que brinda el Subsistema de Contabilidad que permite obtener las cuentas dado los identificadores de cada una.

- **ObtenerCuentaDadoEstructura:** Servicio que brinda el Subsistema de Contabilidad que permite obtener las cuentas que pertenecen a una determinada estructura.

- **ObtenerSaldoCuenta:** Servicio que brinda el Subsistema de Contabilidad que permite conocer el saldo de una cuenta perteneciente a una estructura y a un período determinado.

- **ObtenerEjercicio:** Servicio que brinda Configuración que permite conocer la información del ejercicio actual.

- **ObtenerPeriodo:** Servicio que brinda Configuración que permite conocer la información de un período contable dado el id del período, el del ejercicio y la fecha de inicio.

- **ObtConcOperSumRest:** Servicio que brinda Configuración que permite obtener los conceptos de operación.

Servicio que brinda el componente como parte de la integración:

- **Obtenerultimaejecucion:** Servicio que brinda el Subsistema Costos y Procesos y que permite conocer el último ajuste calculado o ejecutado.

2.4 Conclusiones del capítulo:

La Arquitectura Base definida y los artefactos generados como resultado de la especificación de los requisitos correspondientes al proceso Ajuste al Costo, fueron la base para la realización del presente capítulo a partir de los cuales fue posible llevar a cabo el diseño y la implementación del componente, con el fin de responder a las funcionalidades clave en la gestión de los costos.

Capítulo 3: Validación de la solución propuesta

3.1 Introducción:

La calidad de un producto de software; conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia; se ha convertido en un elemento estratégico de las grandes organizaciones debido a su fuerte impacto en la competitividad de las empresas.

Durante el proceso de desarrollo de software las posibilidades de errores son múltiples, estas pueden aparecer desde la misma especificación de requisitos donde se define lo que el sistema debe hacer. Como elementos críticos aparecen entonces la prueba y la validación de los resultados, estas en lugar de efectuarse una vez desarrollado el software, se llevan a cabo en cada una de las etapas de desarrollo para detectar a tiempo las imperfecciones e irregularidades y proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.

A continuación se evalúa el grado con que se le dio cumplimiento a las necesidades del cliente o usuario, para ello se definen inicialmente los tipos y métodos de prueba, y posteriormente se aplica a un requisito y procedimiento específico. Se valida además el diseño propuesto en el capítulo anterior para el desarrollo del componente Ajuste al Costo a través de las métricas de calidad: Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC) teniendo en cuenta las clases y operaciones definidas.

3.2 Pruebas de Software:

Al conjunto de técnicas experimentales para la búsqueda de fallos en los programas, que determinan en cierto grado la calidad de un producto, se les denomina pruebas de software.

La competencia mundial en el ámbito informático exige productos de calidad, de esta forma se muestra la necesidad de contar con pruebas de este tipo desde las primeras etapas de concepción de un proyecto.

"La prueba no puede asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software." (42)

- **Niveles de Prueba:**

A la hora de evaluar dinámicamente un sistema se debe comenzar por los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas se aplican en distintos niveles de trabajo, dentro de estos se distinguen:

- **Pruebas de Unidad:**

Prueba individual a las unidades separadas de un sistema de software.

- **Pruebas de Integración:**

Los componentes individuales son combinados con otros componentes para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente.

- **Pruebas del Sistema:**

Son usualmente conducidas para asegurar que todos los módulos trabajan como sistema sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio.

- **Pruebas de Aceptación:**

Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

- **Métodos de Prueba:**

Enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba [14] para detectar diferentes tipos de errores. (43)
Dos enfoques alternativos:

Caja Negra: Se comprueban las funcionalidades sin tener en cuenta la estructura interna.

Caja Blanca: Se comprueban los componentes internos.

3.2.1 Objetivo:

Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. El aspecto fundamental que rige esta etapa de pruebas es determinar cómo y en qué sentido el componente cumple con las expectativas del cliente, a partir de los requisitos establecidos y las restricciones impuestas. En ese ámbito se trazan un conjunto de objetivos dentro de los que se sitúan:

- Verificar la implementación del componente.
- Verificar la integración adecuada del componente.

[14] Casos de prueba: especifican una forma de probar el componente, incluye: la entrada, las condiciones bajo las cuales ha de probarse y los resultados esperados.

- Verificar que todos los requisitos se han implementado correctamente.
- Identificar los errores y asegurar que estos sean corregidos de la mejor manera.

Con la idea de diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (44)

3.3 Prueba de Software que será aplicada al componente:

3.3.1 Descripción general de las pruebas para el Nivel de unidad:

Se le denomina Prueba de Unidad al proceso de separar y probar el correcto funcionamiento de las partes individuales de un programa para asegurar que cada uno de estos se ejecuta correctamente por separado.

Ventajas de usar este tipo de prueba:

1. Los errores son más fáciles de localizar.
2. Los errores están más acotados.
3. Se fomenta el cambio.

“Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido.” (45)

El equipo de desarrollo a partir de los conocimientos adquiridos durante la implementación del componente se encargará de aplicar las pruebas para este nivel mediante los métodos antes presentados y las técnicas correspondientes que serán especificadas más adelante.

3.3.1.1 Descripción y aplicación de la Prueba de Caja Blanca o Estructural:

- **Descripción:**

La Prueba de Caja Blanca también se conoce 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:

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 todos los bucles en sus límites operacionales.

- **Aplicación:**

Según descripción de la prueba de caja blanca presentada con anterioridad y a partir de la necesidad de crear un producto de alta calidad es preciso valorar qué tan certera ha sido la implementación del componente Ajuste al Costo y para ello es necesario aplicar una de las técnicas que esta comprende, en este caso la del camino básico.

Para ello es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática. Se debe comenzar por un análisis del código, posteriormente son enumeradas cada una de las instrucciones, se construye el grafo de flujo asociado y según de las fórmulas pertinentes se calcula dicha complejidad:

1- Se realizó un análisis de uno de los procedimientos contenidos en la clase OpAjustarCostosController, específicamente: calcularCD, dicho algoritmo determina los coeficientes de distribución de los productos según la Base de distribución (Importe Monetario/ Unidades Físicas) y el Modo de ajuste (se ajusta a inventario /no se ajusta a inventario).
2- Se enumeraron las sentencias de código para luego construir el grafo de flujo asociado representado a través de nodos, aristas y regiones.
3- Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

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

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 8, de manera que existen ocho posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Camino básico #1: 1 – 2 – 3 – 4 – 6 – 22.

Camino básico #2: 1 – 2 – 3 – 5 – 6 – 22.

Camino básico #3: 1 – 2 – 7 – 8 – 9 –11– 22.

Camino básico #4: 1 – 2 – 7 – 8 – 10 –11– 22.

Camino básico #5: 1 – 2 – 7 –12 – 13 –14–16–22.

Camino básico #6: 1 – 2 – 7 –12 – 13 –15–16–22.

Camino básico #7: 1 – 2 – 7 –12 – 17 –18–19–21–22.

Camino básico #8: 1 – 2 – 7 –12 – 17 –18–10–21–22.

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo. 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.

De forma general:

La Base de distribución tiene sólo dos variantes: Unidades Físicas o Importe Monetario, y el ajuste se puede ejecutar por inventario o no.

$base = 90000 significa que la Base de distribución es por: Unidades Físicas.

$base = 90001 significa que la Base de distribución es por: Importe Monetario.

$ajuste = 1 significa que el ajuste es por inventario.

$ajuste = 0 significa que el ajuste no es por inventario.

A continuación, serán descritos los casos de prueba para los 4 primeros caminos básicos determinados:

1. Caso de prueba para el camino básico # 1.

Descripción: Se le debe calcular el Coeficiente de distribución a los productos implicados en el ajuste, para ello es necesario que se determine la Base de distribución y el Modo de ajuste correctamente.

Condición de ejecución: Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: las Unidades Físicas y el Importe Ajustar deben estar dados en valores numéricos. Los campos correspondientes a la Base de distribución y el Modo de ajuste no pueden estar vacíos.

Entrada:

$ajuste = 1
$base = 90000
Unidades Físicas = 36200
Importe Ajustar = 167500074563

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el cálculo del Coeficiente de Distribución:
CD = Importe a Ajustar / Unidades Físicas, de cómo resultado 4627073.882

El cálculo fue ejecutado correctamente.

2. Caso de prueba para el camino básico # 2.

Descripción: Se le debe calcular el Coeficiente de distribución a los productos implicados en el ajuste, para ello es necesario que se determine la Base de distribución y el Modo de ajuste correctamente.

Condición de ejecución: Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: las Unidades Físicas y el Importe a Ajustar deben estar dados en valores numéricos. Los campos correspondientes a la Base de distribución y el Modo de ajuste no pueden estar vacíos.

Entrada:

$ajuste = 1

$base = 90000
Unidades Físicas = 0
Importe a Ajustar = 167500074563

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el cálculo del Coeficiente de Distribución:
CD = Importe a Ajustar / Unidades físicas, de cómo resultado 0

El cálculo fue ejecutado correctamente.

3. Caso de prueba para el camino básico # 3.

Descripción: Se le debe calcular el Coeficiente de distribución a los productos implicados en el ajuste, para ello es necesario que se determine la Base de distribución y el Modo de ajuste correctamente.

Condición de ejecución: Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: el Importe Total y el Importe a Ajustar deben estar dados en valores numéricos. Los campos correspondientes a la Base de distribución y el Modo de ajuste no pueden estar vacíos.

Entrada:
$ajuste = 1
$base = 90001
Importe Total = 2413200
Importe a Ajustar = 167500074563

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el cálculo del Coeficiente de Distribución:
CD = Importe a Ajustar / Importe Total, de cómo resultado 69409.943

El cálculo fue ejecutado correctamente.

4. Caso de prueba para el camino básico # 4.

Descripción: Se le debe calcular el Coeficiente de distribución a los productos implicados en el ajuste, para ello es necesario que se determine la Base de distribución y el Modo de ajuste correctamente.

Condición de ejecución: Para ejecutar el algoritmo es necesario que los datos de entrada cumplan con los siguientes requisitos: el Importe Total y el Importe a Ajustar deben estar dados en valores numéricos. Los campos correspondientes a la Base de distribución y el Modo de ajuste no pueden estar vacíos.

Entrada:

$ajuste = 1

$base = 90001

Importe Total = 2413200

Importe a Ajustar = 0

Resultados esperados: Teniendo en cuenta los datos pasados por parámetro se espera que el cálculo del Coeficiente de Distribución:
CD = Importe a Ajustar / Importe Total, de cómo resultado 0.

El cálculo fue ejecutado correctamente.

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

3.3.1.2 Descripción y aplicación de la Prueba de Caja Negra o Funcional:

- **Descripción:**

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos.
Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, 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.
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.

Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

- **Aplicación:**

A continuación se aplica la prueba de partición de equivalencia como parte de la realización de la prueba de caja negra sobre la interfaz que responde al requisito funcional Ajustar Costos.

La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Luego de aplicados los métodos de prueba por el equipo de desarrollo al procedimiento calcularCD y al requisito funcional Ajustar Costos antes presentados, es válido señalar que los resultados obtenidos hasta el momento han sido satisfactorios desde el punto de vista interno y funcional del componente, atendiendo al correcto comportamiento del mismo ante diferentes situaciones (entradas válidas y no válidas).

3.4 Validación del modelo de diseño propuesto:

La aplicación de métricas al diseño y la implementación de un producto de software constituyen un elemento fundamental a la hora de evaluar la calidad del mismo.
“Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes” (46)

Seguidamente se hará una evaluación del diseño propuesto para el componente Ajuste al Costo del

Subsistema Costos y Procesos donde se tienen en cuenta tanto atributos como métricas de calidad. Atributos de calidad que se abarcan:

- **Responsabilidad:** Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
- **Complejidad del mantenimiento**: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
- **Complejidad de implementación:** Grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
- **Acoplamiento:** Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- **Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

A continuación se presentarán las métricas necesarias para evaluar la calidad del diseño propuesto:

- **Tamaño operacional de clase (siglas: TOC):** Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.
- **Relaciones entre clases (siglas: RC):** Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Resultados del instrumento de evaluación de la métrica Tamaño operacional de clase (TOC).
Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos:

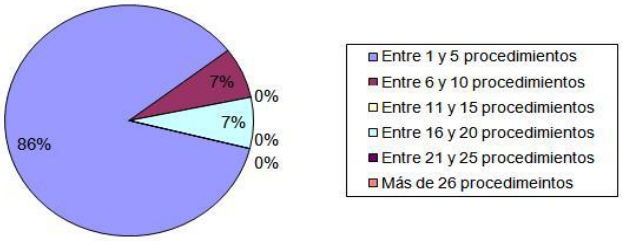


Figura 5 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el componente Ajuste al Costo está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (86%) posee menos cantidad de operaciones que la media registrada en las mediciones.

Los atributos de calidad se encuentran en un nivel satisfactorio en el 72% de las clases; de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

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

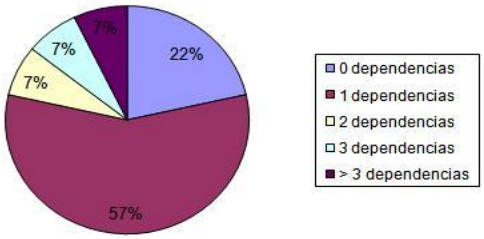


Figura 6 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el componente Ajuste al Costo está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (86%) poseen menos de 3 dependencias respecto a otras.

Los atributos de calidad se encuentran en un nivel satisfactorio; en el 79% de las clases el grado de dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 79% de las clases.

3.5 Demostración de la eficiencia de los algoritmos:

Los elementos que intervienen en la eficiencia de un algoritmo tienen una naturaleza variada, y están determinados fundamentalmente por el propio algoritmo, los datos de entrada y el medio de cómputo sobre el cual se ejecuta. El proceso de análisis se refiere específicamente a medir el uso eficiente de los recursos en función de dos parámetros:

1- **Tiempo**: el tiempo que tarda en ejecutarse (complejidad temporal).

2- **Espacio**: la memoria que el algoritmo utiliza (complejidad espacial).

Cotas de complejidad:

Las cotas permiten asegurar, en el caso de la Cota máxima que el problema se puede resolver en ese tiempo como máximo; en el caso de la Cota mínima que el problema no se puede resolver con ningún algoritmo que tenga un tiempo de ejecución menor que el de la función que lo acota.

Órdenes de Complejidad:

Se dice que O (f(n)) define un "orden de complejidad", de manera que se tiene:

O (1) orden constante

O (log n) orden logarítmico

O (n) orden lineal

O (n^2) orden cuadrático

O (c^n) orden exponencial

Sobre la complejidad temporal:

Reglas para determinar la complejidad temporal de un algoritmo:

1-El tiempo requerido para: acceder a un valor, realizar operaciones aritméticas y almacenar el resultado en memoria es constante, con un valor de O (1).

2-El tiempo requerido para la ejecución de un algoritmo de secuencia lineal es la sumatoria de los tiempos de cada instrucción.

3-El tiempo requerido para la ejecución de una instrucción condicional **if** C **then** I1 **else** I2 es:

T = T(C) + Máximo (T (I1), T (I2)).

Se aplica la regla de la suma, de modo que se calcula el tiempo de ejecución tomando el máximo de los tiempos de ejecución de cada una de las partes, (sentencias individuales) en que puede dividirse.

4-La complejidad de un ciclo está dada por el total de veces que se evalúa la condición por la complejidad de evaluar la condición más el producto del total de veces que se ejecuta el ciclo por la

complejidad del cuerpo del ciclo.
5-Bloque de sentencias: Se aplica la regla de la suma.
6-Llamadas a funciones: La sentencia de código pasa a tener la misma complejidad que la función invocada.
Según lo antes expuesto se calculó la cota máxima del algoritmo contabilizar, con el fin del demostrar cuán eficiente es el mismo a partir del tiempo máximo que tarda en ejecutarse:

Al aplicar la regla de la suma, teniendo en cuenta el número de sentencias y la complejidad de las mismas, se obtuvo como resultado O (n).

Sobre la complejidad espacial:

La eficiencia en memoria de un algoritmo indica la cantidad de espacio requerido para ejecutar el algoritmo; es decir, el espacio en memoria que ocupan todas las variables propias al algoritmo. Para calcular la memoria estática se suma la memoria que ocupan las variables declaradas en el algoritmo.

En este caso se aplica de igual manera la regla de la suma, lo que da como resultado:

En el algoritmo antes referenciado la utilización de los recursos se comporta de manera satisfactoria según los parámetros Espacio y Tiempo, en ambos casos el orden de complejidad es lineal: O (n) que comparada con los demás órdenes indica la eficiencia del mismo, partiendo de que los algoritmos que presentan complejidad mayor a O (n^2) se consideran intratables o desprovistos de solución.

3.6 Conclusiones del capítulo:

La calidad del componente desarrollado fue el elemento clave del capítulo que recién concluye. En ese sentido se efectuaron pruebas de software en el nivel de unidad mediante casos de pruebas, para los cuales se tuvieron en cuenta las entradas, las salidas, los resultados esperados y el tratamiento de errores en caso de anomalías; se aplicaron además las métricas: Relaciones entre Clases y Tamaño Operacional de la Clase para validar y evaluar el diseño, las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes.

El componente Ajuste al Costo desde el punto de vista funcional cumple con los requerimientos capturados y especificados en las primeras etapas de desarrollo a partir de las expectativas del cliente.

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, para esto:

- Se analizaron ventajas y deficiencias de sistemas informáticos tanto nacionales como internacionales vinculados a la gestión de los costos; evidenciándose de esta manera la no existencia de una solución informática capaz de ejecutar las funcionalidades referentes al ajuste de los costos ni de cumplir con los requisitos establecidos a nivel nacional.
- Se realizó el diseño y la implementación del componente Ajuste al Costo correspondiente al Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX, con el objetivo de erradicar los problemas de los sistemas existentes y fusionar sus mejores prácticas
- Se evaluó la viabilidad del componente a través de pruebas de software efectuadas para el nivel de unidad, las cuales arrojaron resultados favorables posibilitando dar cumplimiento a las funcionalidades previstas para el mismo.

Esta propuesta exhibe valor técnico, donde se destaca la incorporación de principios por los que se mide la factibilidad de un diseño de software, ejemplo: la utilización de patrones que posibilita la reutilización, garantizando la sostenibilidad y mantenimiento del sistema.

La solución propuesta es novedosa, su importancia radica en la realización de los procesos referentes al ajuste de los costos de la producción dentro de un único componente, permitiendo el ahorro de tiempo y recursos, así como el control eficiente de los cambios y la información manipulada.

Recomendaciones:

Al concluir el presente trabajo de diploma, considerando cumplidos los objetivos trazados en el mismo, se recomienda:

- Continuar realizando pruebas de calidad al componente.
- Realizar el despliegue de la aplicación en varias entidades para comprobar si cumple desde el punto de vista tecnológico con las expectativas del cliente.
- Optimizar los algoritmos para un mejor funcionamiento interno del componente.
- Profundizar en temas referentes al ajuste de los costos para detectar posibles debilidades en el componente y agregar mejoras al mismo.

Bibliografía referenciada:

1- **2009.** Businesscol.com. [En línea] 2009. [Citado el: 13 de 01 de 2010.] http://www.businesscol.com/productos/glosarios/economico/glossary.php?word=CENTRO%20DE%20COSTOS.

2- **Genes, Beatriz Elena.** Contabilidad de Costos y Presupuesto. . [En línea] [Citado el: 13 de 01 de 2010.] http://www.mailxmail.com/curso-acumulacion-costos/elementos-gasto.

3- La gran enciclopedia de economía. [En línea] [Citado el: 13 de 01 de 2010.] http://www.economia48.com/spa/d/cuentas-de-gastos/cuentas-de-gastos.htm.

4- **2010.** CubaIndustria. [En línea] 2010. [Citado el: 13 de 01 de 2010.] http://www.cubaindustria.cu/contadoronline/contabilidad/uso%20y%20contenido/731%20Gastos%20Indirectos%20de%20Producci%C3%B3n.htm.

5- **Martell, Maritza Díaz. 2009.** Fundamentos de los costos. [En línea] 2009. [Citado el: 13 de 01 de 2010.] http://www.mailxmail.com/curso/empresa/fundamentoscostos/capitulo5.htm.

6- **2007.** OpenBravo. [En línea] 2007. [Citado el: 10 de 01 de 2010.] http://www.openbravo.com/es/product/erp/features.

7- **2007.** OpenERP. [En línea] 2007. [Citado el: 10 de 01 de 2010.] http://www.openerpsite.com/?page_id=35.

8- **2010.** Portal de ayuda del SAP. [En línea] 2010. [Citado el: 10 de 01 de 2010.] http://help.sap.com/saphelp_40b/helpdata/es/eb/13771c43c411d1896f0000e8322d00/content.htm.

9- **2009.** SISCONT. Software Contable Financiero. [En línea] 2009. [Citado el: 14 de 01 de 2010.] http://www.siscont.com/DESCRIPTIVO%20SISCONT.pdf.

10- **Miguel P. Cabrera González, Guillermo Obregón Rodríguez, Margarita Cárdenas Negrin, Luis Mario Carralero Silva.** XV FORUM DE CIENCIA Y TECNICA. SISTEMA ECONOMICO INTEGRADO .

11- **Sara Alvarez. 2006.** Desarrollo Web. [En línea] 2006. [Citado el: 15 de 02 de 2010.] http://www.desarrolloweb.com/articulos/2477.php

12- **2007.** desarrolloweb.com. [En línea] 06 de 2007. [Citado el: 20 de 02 de 2010.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html

13- **2006.** SOA. [En línea] 03 de 2006. [Citado el: 20 de 02 de 2010.] http://arquitecturaorientadaaservicios.blogspot.com/2006/03/pero-qu-es-realmente-soa.html.

14- **2009.** GNU Operating System. [En línea] 2009. [Citado el: 16 de 02 de 2010.] http://www.gnu.org/philosophy/free-sw.html.

15- **Yanet Vega, L. S. 2009.** Definición del ciclo de vida del proyecto. Habana : s.n., 2009.

16- **2010.** Desarrollo web. [En línea] 2010. [Citado el: 02 de 15 de 2010.] http://www.desarrolloweb.com/articulos/392.php.

17- **Fernández, Osmar Leyet. 2010.** Documento Línea Base de proyecto-CEDRUX-1.0. 2010.

18- **Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2007.** Desarrollo de Software Basado en Componentes. Málaga, Spain : s.n., 2007.

19- **2009.** debug_mode=on. [En línea] 2009. [Citado el: 10 de 03 de 2010.] http://es.debugmodeon.com/articulo/el-patron-mvc.

20- **Ing. Joisel Pérez Pérez, Ing.Enrique Chaviano Gómez, Ing. Donel Vázquez Zambrano. 2009.** Diseño de solución informática para la gestión y control de los costos en las entidades. Ciudad de La Habana : s.n., 2009.

21- El mundo informatico. [En línea] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

22- **Prieto, Félix. 2009.** Patrones de diseño. 2009.

23- **2010.** Hypertext Preprocessor. [En línea] 2010. [Citado el: 01 de 15 de 2010.] http://php.net/index.php.

24- **2002.** Lenguaje HTML. [En línea] 2002. [Citado el: 26 de 02 de 2010.] http://www.desarrolloweb.com/articulos/711.php

25- **2003.** Extensible Markup Language (XML). [En línea] 2003. http://www.w3.org/XML/.

26- **2001.** territoriopc. [En línea] 2001. [Citado el: 16 de 02 de 2010.] http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.

27- **2010.** Extjs. [En línea] 2010. [Citado el: 01 de 25 de 2010.] http://www.extjs.com.

28- **2010.** Los datos de objetos de PHP. [En línea] 2010. [Citado el: 05 de 03 de 2010.] http://php.net/manual/es/book.pdo.php.

29- Introducing JSON. [En línea] http://www.json.org/.

30- **2010.** ZEND FRAMEWORK. [En línea] 2010. [Citado el: 25 de 01 de 2010.] http://framework.zend.com/manual/en/.

31- 2007. Metodologiasdesistemas. [En línea] 2007. [Citado el: 22 de 01 de 2010.] http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.

32- 2008. Doctrine. [En línea] 2008. [Citado el: 26 de 01 de 2010.] http://www.doctrine-project.org..

33- 2006. AbartiaTeam. [En línea] 2006. [Citado el: 26 de 01 de 2010.] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

34- 2005. visual-paradigm. [En línea] 2005. [Citado el: 01 de 30 de 2010.] http://www.visual-paradigm.com.

35- 2010 . Subversion. [En línea] 2010 . [Citado el: 29 de 01 de 2010.] http://subversion.apache.org.

36- 2010. Zend Studio - The Professional PHP IDE – Zend. [En línea] 2010. [Citado el: 01 de 22 de 2010.] http://www.zend.com/products/studio.

37- 2009. Documentación del Servidor HTTP Apache 2.0. [En línea] 2009. http://httpd.apache.org/docs/2.0/es/.

38- 2010. DISEÑO DE BASES DE DATOS RELACIONALES. [En línea] 2010. http://usuarios.multimania.es/cursosgbd/UD4.htm.

39- 2010. PostgreSQL: The world's most advanced open source database Postgresql. [En línea] 2010. [Citado el: 02 de 02 de 2010.] http://www.postgresql.org.

40- 2009. mozilla europe. [En línea] 2009. http://www.mozilla-europe.org/es/firefox/.

41- 2008. Definicion.de. [En línea] 2008. http://definicion.de/modelo-de-datos/.

42- Pressman, Roger. 1998. Ingeniería de Software. Un enfoque práctico. 1998

43- Ramírez, Jaime. Unidad de Programación. Métodos de Prueba del Software.

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

45- 2010. Pruebas de Software. [En línea] 2010. [Citado el: 6 de 04 de 2010.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

46- Aylienn Aquino Leiva, Yasser Linares Domínguez. 2009. Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux. Cuba : s.n., 2009.

Bibliografía consultada:

Rolando Alfredo Hernández, Sayda Coello González. *El paradigma cuantitativo de la investigación científica.*

Mañas, J. A. 1994. Pruebas de programas. Recuperado. [En línea] 1994. [Citado el: 7 de 04 de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s22.

Pélaez, Juan. 2001. [En línea] Octubre de 2001. [Citado el: 15 de 02 de 2010.] http://geeks.ms/blogs/jkpelaez/archive/2010/04/18/arquitectura-basada-en-componentes.aspx.

David Mestre. 2008. [En línea] 2008. [Citado el: 17 de 12 de 2009.] http://davidmaestre.com/2008/01/modulos-de-sap-r3.html..

Balcázar, José L. 2010. Apuntes sobre el cálculo de la eficiencia de los algoritmos. [En línea] 2010. [Citado el: 10 de 04 de 2010.] http://webdiis.unizar.es/asignaturas/TAP/material/eficiencia.pdf.

2010.], [En línea] . [Citado el: 4 de Febrero de. 1998. Firefox web browser | Faster, more secure, & customizable. [En línea] 1998. [Citado el: 4 de 02 de 2010.] http://www.mozilla.com/en-US/firefox/firefox.html.

2010. Pruebas de Software. [En línea] 2010. [Citado el: 6 de 04 de 2010.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

2001. José Enrique González Cornejo. *Arquitectura en Capas ~ DNA Un camino hacia los procesos distribuidos.* [En línea] 2001. [Citado el: 15 de 02 de 2010.] http://www.arquitectura_tres_capas.com.

2009. Businesscol.com. [En línea] 2009. [Citado el: 13 de 01 de 2010.] http://www.businesscol.com/productos/glosarios/economico/glossary.php?word=CENTRO%20DE%20COSTOS.

Genes, Beatriz Elena. Contabilidad de Costos y Presupuesto. . [En línea] [Citado el: 13 de 01 de 2010.] http://www.mailxmail.com/curso-acumulacion-costos/elementos-gasto.

2007. OpenBravo. [En línea] 2007. [Citado el: 10 de 01 de 2010.] http://www.openbravo.com/es/product/erp/features.

2007. OpenERP. [En línea] 2007. [Citado el: 10 de 01 de 2010.] http://www.openerpsite.com/?page_id=35.

2010. Portal de ayuda del SAP. [En línea] 2010. [Citado el: 10 de 01 de 2010.] http://help.sap.com/saphelp_40b/helpdata/es/eb/13771c43c411d1896f0000e8322d00/content.htm.

2009. SISCONT. Software Contable Financiero. [En línea] 2009. [Citado el: 14 de 01 de 2010.] http://www.siscont.com/DESCRIPTIVO%20SISCONT.pdf.

Miguel P. Cabrera González, Guillermo Obregón Rodríguez, Margarita Cárdenas Negrin, Luis Mario Carralero Silva. XV FORUM DE CIENCIA Y TECNICA. SISTEMA ECONOMICO INTEGRADO

Sara Alvarez. 2006. Desarrollo Web. [En línea] 2006. [Citado el: 15 de 02 de 2010.] http://www.desarrolloweb.com/articulos/2477.php

2007. desarrolloweb.com. [En línea] 06 de 2007. [Citado el: 20 de 02 de 2010.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html

2006. SOA. [En línea] 03 de 2006. [Citado el: 20 de 02 de 2010.] http://arquitecturaorientadaaservicios.blogspot.com/2006/03/pero-qu-es-realmente-soa.html.

2009. GNU Operating System. [En línea] 2009. [Citado el: 16 de 02 de 2010.] http://www.gnu.org/philosophy/free-sw.html.

Yanet Vega, L. S. 2009. Definición del ciclo de vida del proyecto. Habana : s.n., 2009.

2010. Desarrollo web. [En línea] 2010. [Citado el: 02 de 15 de 2010.] http://www.desarrolloweb.com/articulos/392.php.

Fernández, Osmar Leyet. 2010. Documento Línea Base de proyecto-CEDRUX-1.0. 2010.

Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2007. Desarrollo de Software Basado en Componentes. Málaga, Spain : s.n., 2007.

2009. debug_mode=on. [En línea] 2009. [Citado el: 10 de 03 de 2010.] http://es.debugmodeon.com/articulo/el-patron-mvc.

Ing. Joisel Pérez Pérez, Ing.Enrique Chaviano Gómez, Ing. Donel Vázquez Zambrano. 2009. Diseño de solución informática para la gestión y control de los costos en las entidades. Ciudad de La Habana : s.n., 2009.

El mundo informatico. [En línea] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

Prieto, Félix. 2009. Patrones de diseño. 2009.

2010. Hypertext Preprocessor. [En línea] 2010. [Citado el: 01 de 15 de 2010.] http://php.net/index.php.

2002. Lenguaje HTML. [En línea] 2002. [Citado el: 26 de 02 de 2010.] http://www.desarrolloweb.com/articulos/711.php

2003. Extensible Markup Language (XML). [En línea] 2003. http://www.w3.org/XML/.

2001. territoriopc. [En línea] 2001. [Citado el: 16 de 02 de 2010.] http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.

2010. Extjs. [En línea] 2010. [Citado el: 01 de 25 de 2010.] http://www.extjs.com.

2010. Los datos de objetos de PHP. [En línea] 2010. [Citado el: 05 de 03 de 2010.] http://php.net/manual/es/book.pdo.php.

Introducing JSON. [En línea] http://www.json.org/.

2010. ZEND FRAMEWORK. [En línea] 2010. [Citado el: 25 de 01 de 2010.] http://framework.zend.com/manual/en/.

2007. Metodologiasdesistemas. [En línea] 2007. [Citado el: 22 de 01 de 2010.] http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.

2008. Doctrine. [En línea] 2008. [Citado el: 26 de 01 de 2010.] http://www.doctrine-project.org..

2006. AbartiaTeam. [En línea] 2006. [Citado el: 26 de 01 de 2010.] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

2005. visual-paradigm. [En línea] 2005. [Citado el: 01 de 30 de 2010.] http://www.visual-paradigm.com.

2010 . Subversion. [En línea] 2010 . [Citado el: 29 de 01 de 2010.] http://subversion.apache.org.

2010. Zend Studio - The Professional PHP IDE – Zend. [En línea] 2010. [Citado el: 01 de 22 de 2010.] http://www.zend.com/products/studio.

2009. Documentación del Servidor HTTP Apache 2.0. [En línea] 2009. http://httpd.apache.org/docs/2.0/es/.

2010. DISEÑO DE BASES DE DATOS RELACIONALES. [En línea] 2010. http://usuarios.multimania.es/cursosgbd/UD4.htm.

2010. PostgreSQL: The world's most advanced open source database Postgresql. [En línea] 2010. [Citado el: 02 de 02 de 2010.] http://www.postgresql.org.

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

2010. Pruebas de Software. [En línea] 2010. [Citado el: 6 de 04 de 2010.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

Aylienn Aquino Leiva, Yasser Linares Domínguez. 2009. Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux. Cuba : s.n., 2009

Anexos:

Anexo 1: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Tabla 6 Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Anexo 2: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

Tabla 7 Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
	Categoría	Criterio
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
	Categoría	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
	Categoría	Criterio
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Glosario de términos:

1. Algoritmo: Cualquier procedimiento computacional bien definido mediante un conjunto de reglas que da solución a instancias de un problema (entrada) produciendo un valor o conjunto de valores (salida).

2. Aplicación web: En la ingeniería software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, etc.) en la que se confía la ejecución al navegador.

3. Código abierto (en inglés Open Source): Término con el que se conoce al software distribuido y desarrollado libremente.

4. Framework: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

5. Lenguaje de marcado: Es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

6. Navegador: Software que permite al usuario recuperar y visualizar documentos de hipertexto desde servidores web a través de Internet.

7. Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente

8. Sistema propietario: Cualquier programa informático en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o cuyo código fuente no está disponible o el acceso a éste se encuentra restringido.