Universidad de las Ciencias Informáticas

Proyecto ERP Cuba

Título: Implementación del Módulo Inventario Físico del Subsistema Inventario del Sistema de Gestión Integral Cedrux.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Alien Fernández Fuentes

Tutor: Lic. Arismayda Dorado Risco

Ciudad de la Habana. Junio de 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

Pensamiento

"Muchos me dirán aventurero, y lo soy, solo que de un tipo diferente y de los que ponen el pellejo para demostrar sus verdades."

ERNESTO CHE GUEVARA

Datos de contacto

Tutor: Lic. Arismayda Dorado Risco.

Categoría Científica: Licenciada.

Categoría Docente: Instructor.

Correo electrónico: dorado@uci.cu

Síntesis del Tutor:

Profesora con 4 años de experiencia. Licenciada en Ciencias de la Computación. Actualmente es Arquitecta de Sistema del Subsistema Inventario del Sistema de Gestión Integral Cedrux.

Agradecimientos

A mis padres por la confianza, el apoyo y el amor, por sus consejos, por ser nuestros principales guías y ayudarnos a llegar hasta aquí. ¡Nunca los defraudaré!

A mi novia por estar a mi lado todos estos años, dedicándome confianza y apoyo, día a día, ¡noche a noche!

A mi tutora por todo lo que ha hecho por mí en estos últimos meses.

A todas las personas que de una forma u otra han contribuido en mi desarrollo profesional, que cada día han estado ahí para enseñarme cosas nuevas de la vida y llegar a la meta que me he propuesto, esas personas que han estado en los buenos y mucho más los que han estado en las malas. Gracias por la paciencia de todos aquellos que han estado a mi lado cada segundo de preocupación y dedicación, muchas gracias. A mis amigos por los consejos, por la seguridad que siempre han tenido en mí.

A Fidel, a Raúl y a la Revolución.

A todos, muchas gracias.

Dedicatoria

A mis padres, a mi hermana, a mis abuelos y a mi novia: por todos los valores que han sabido inculcarme y por haber confiado en mí todos estos años.

A mi novia por ser una parte importante de mi vida.

Resumen

En el presente trabajo de diploma se expone la implementación del módulo de Inventario Físico del Subsistema Inventario del Sistema Integral de Gestión Cedrux. Actualmente la situación general de los procesos de gestión de las entidades presupuestadas y empresariales a escala nacional está afectada por la existencia de sistemas informáticos que no cumplen con las expectativas de las nuevas tecnologías y la misión de nuestro país en el desarrollo de sus empresas. Proteger los recursos de las empresas así como llegar a la exactitud y confiabilidad de la información económica y contable, ha llevado a la necesidad de mejorar los procesos de gestión de las áreas que las conforman, utilizando plataformas confiables y eficientes. Con del desarrollo de éste módulo se deberá lograr optimizar el proceso de gestión de inventario físico, disminuir los costos totales de operación, compartir información entre todos los componentes de la empresa y disminuir el margen de contaminación y repetición de la información. Una vez finalizado el módulo se ratifica la solución propuesta a partir de la utilización de métricas para validar el diseño y pruebas de caja blanca para verificar que el código funcione de manera correcta.

PALABRAS CLAVES

Control de inventario, inventario físico, métricas.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

El desarrollo de sistemas automatizados de gestión empresarial ha tomado un gran auge a nivel mundial, sobre todo a partir de la generalización de las nuevas tecnologías de la información y las comunicaciones.

En la última década la economía cubana ha experimentado una notable recuperación, en este impulso ha jugado un papel determinante el sector empresarial, al poner en práctica nuevas estrategias productivas y perfeccionar los métodos para el control de los medios materiales como la Gestión de Inventarios.

El inventario es una parte primordial de muchas empresas, el control del mismo es un aspecto crítico de la administración exitosa, pues mantener un inventario implica un alto costo tanto financiero como de recursos humanos, muchas empresas realizan este proceso a través de software desarrollados en varias partes del mundo, los cuales viabilizan el Control del Inventario.

Los software se especializan en determinados negocios. En la actualidad se ha hecho imprescindible el uso de los Planeadores de Recursos Empresariales (Enterprise Resource Planning, ERP), donde estos integran y manejan las operaciones de producción y los aspectos de distribución de una entidad comprometida en la producción de bienes o servicios, son parte del conjunto de sistemas de información gerencial que permiten tener el control de la empresa por sus directivos en tiempo real.

En el año 2008 la dirección del país propuso la creación de un software de gestión para automatizar los procesos que realizan las entidades, surgiendo en la Universidad de las Ciencias Informáticas el proyecto ERP Cuba, donde se desarrolla el producto Cedrux[1] como solución tecnológica a los problemas existentes en Cuba con la gestión de los procesos en sus empresas.

Implementar este producto es un proceso largo, costoso, complejo y que requiere de gran cantidad de desarrolladores. Para enfrentar esta situación el proyecto se estructuró por líneas con el objetivo de agilizar el desarrollo y de cubrir todas las áreas a informatizar de una empresa. Logística, nombre que se le dio a una de las líneas propuestas contiene varios Subsistemas, uno de ellos es Inventario, el cual se encarga del control de todos los medios materiales en almacenamiento (aperturas, movimientos, posteo,

[1] **Cedrux:** Vocablo formado por la unión de las palabras “Cedro” (fortaleza, resistencia) y “Linux” (tecnología libre).

operaciones contables, cierre y la obtención de la información necesaria), procesos que en algunos casos se realizan de forma manual, y en otros, no cumplen con todos los requisitos específicos de cada entidad.

Hay que destacar que la gestión de inventarios es un aspecto crítico de la administración de recursos de las empresas. Los inventarios surgen porque permiten reducir los costes en la actividad empresarial, y pueden servir tanto para regular los procesos de producción como para determinar el nivel de mercancías almacenadas. Es importante señalar que el control de los mismos posibilita la optimización de los tiempos, mantenimiento del nivel competitivo y protección contra aumentos de precios y escasez de materia prima (Wikidot.com, 2008).

Debido a la importancia que posee la gestión de inventario en las entidades enfocada en el producto que se desarrolla, el equipo de desarrollo debe entender el negocio al cual se va a enfrentar y conocer con detalles las funcionalidades que debe realizar el sistema a implementar.

Este trabajo se centra precisamente en implementar los requisitos obtenidos por el equipo de análisis para el Subsistema Inventario por lo que se puede identificar como **problema** a resolver:

¿Cómo obtener un producto funcional a partir de los requerimientos identificados para la gestión de los procesos de Inventario Físico del Subsistema Inventario para el país?

Luego de lo antes expuesto se definió el **objeto de estudio** en los procesos para el Control de Inventario y como **campo de acción** los procesos de Inventario Físico de los medios materiales en almacenamiento del país.

Para solucionar el problema planteado se propone como **objetivo:** implementar el módulo de Inventario Físico, perteneciente al subsistema de Inventario del Sistema de Gestión Integral Cedrux.

Para darle cumplimiento a este objetivo se plantearon los siguientes **objetivos específicos**:

- Analizar el proceso de Inventario Físico y los sistemas que existen actualmente para su gestión, así como las herramientas que se utilizarán para el desarrollo de la solución.
- Implementar un módulo para la gestión del Inventario Físico de los productos en almacenamiento.
- Validar la solución propuesta.

Las **tareas** que se realizarán para dar solución a los objetivos específicos planteados son:

- Análisis de los procesos de inventario físico.
- Análisis de los sistemas existentes para el control de inventarios.
- Análisis de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
- Análisis de los artefactos entregados por los analistas para los procesos de inventario físico.
- Implementación de las interfaces a partir del prototipo entregado por los analistas.
- Implementación de la capa de negocio que de respuesta a los requisitos propuestos por los analistas.
- Implementación de la capa de acceso a datos.
- Implementación de las validaciones, excepciones.
- Implementación de los servicios incluidos dentro de sus responsabilidades que se necesiten para la implementación de otros módulos.
- Realización de pruebas de unidad a los componentes obtenidos.
- Aplicación de métricas para validar los resultados del producto obtenido.

Para la realización de esta investigación se ha planteado la siguiente **idea a defender**:

Si se realiza la implementación del componente Inventario Físico, perteneciente al subsistema Inventario del Sistema de Gestión Integral Cedrux, se obtendrá un producto funcional para la gestión de dichos procesos.

Como posibles resultados se esperan:

Desarrollar un subsistema para controlar la gestión de los Inventarios Físicos de los medios materiales en almacenamiento.

Los resultados esperados con el sistema en la producción de software son:

- Mayor organización de los productos o equipos de una entidad.

- Elevar la confiabilidad en los sistemas ERP.
- Disminuir considerablemente el tiempo empleado en el proceso de gestión y organización.
- Optimizar los costos.
- Facilitar el trabajo en grupo de los especialistas que trabajan en el área, logrando un mayor control y organización.
- El valor social del sistema se expresa en la contribución a mejorar las condiciones de trabajo, desempeño y equidad de los especialistas del área, al mismo tiempo que le permitirá al país centralizar y controlar mejor la economía.

Los siguientes **métodos teóricos** sustentan la investigación:

Histórico-Lógico: Su empleo permitió el desarrollo evolutivo y coherente en el estudio de la metodología orientada a objetos, patrones de diseño, herramientas de desarrollo de software y sistemas ERP para el desarrollo de los artefactos que proponen los flujos estudiados.

Analítico-Sintético: Permitió integrar y descomponer el conocimiento, descubriendo las relaciones para la utilización de artefactos propuestos por constituir este método una unidad dialéctica, determinando los aspectos esenciales y el arribo a conclusiones prácticas y teóricas.

Modelación: Su utilización permitió crear abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados.

El **aporte práctico** esperado del trabajo de diploma es: La obtención del módulo inventario físico perteneciente al subsistema de Inventario.

El presente documento se estructura en tres capítulos que incluyen todo lo relacionado con el trabajo investigativo y con la implementación del módulo.

En el primer capítulo se expone el estado del arte, mostrando los sistemas vinculados al campo de acción. Al mismo tiempo se describe el objeto de estudio. También se detallan tendencias y tecnologías actuales utilizadas para el desarrollo del subsistema y el motivo de su utilización. Se realiza un análisis del modelo de desarrollo utilizado, así como de la herramienta utilizada para el desarrollo del módulo de inventario.

En el segundo capítulo se realizará una valoración crítica del diseño propuesto por el analista, las posibles implementaciones, componentes o módulos ya existentes y que puedan ser rehusados, estrategias de integración, se realizará la descripción de las principales clases que se implementaron.

En el tercer capítulo se realizará la validación de la solución propuesta, esto incluye el diseño, descripción teniendo los objetivos, alcance, tipo y detalles de los test, además se hará una descripción de los valores utilizados para los mismos, se evaluará su ejecución y los resultados obtenidos.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Es indudable que el ambiente competitivo en el que se vive, en el ámbito empresarial, actualmente requiere de promover los procesos y actividades de negocio que generan las ventajas competitivas de las compañías ante sus competidores más fuertes.

Hoy más que nunca las empresas requieren de herramientas que les proporcionen control y centralización de su información, esto con el fin de tomar las mejores decisiones para sus procesos y estrategias de negocios. Los ERP son una solución robusta para aquellas empresas que buscan la centralización de su información, estos sistemas automatizan los procesos para el Control de Inventario. Además de los sistemas mencionados anteriormente, también se han desarrollado software relacionados con el Control de Inventarios que no forman parte de paquetes integrados, cuyo análisis ha servido para tener en cuenta las ventajas y desventajas que presentan. A esto se debe sumar la política que implementa Cuba para migrar a software libre con el objetivo de lograr su soberanía tecnológica producto a restricciones impuestas por el bloqueo de Estados Unidos de América.

Debido a la diversidad de sistemas existentes, surge la necesidad de hacer un estudio tanto en nuestro país como en el mundo y las herramientas a utilizar, buscando obtener la información más actual de las mismas.

1.2 Conceptos generales

1.2.1 ¿Qué es un ERP?

Los sistemas de Planificación de Recursos Empresariales (Enterprise Resource Planning, ERP) son sistemas de gestión de información gerenciales que integran, automatizan y manejan muchas de las acciones asociadas con las operaciones de producción y distribución de una compañía dedicada a la producción de bienes o servicios.

Los ERP pueden intervenir en el control de muchas actividades de negocios, tales como ventas, compras, entregas, pagos, producción, contabilidad, administración de inventarios y de recursos humanos.

Funcionan en todo tipo de empresas. Para ello, integran todos los departamentos funcionales de la empresa, herramientas de mercadotecnia y administración estratégica en un solo sistema (Wailgum, 2008).

1.2.1.1 Características de los sistemas ERP.

Un ERP se distingue de otro software empresarial porque es un sistema integral, con modularidad y adaptable (Wailgum, 2008).

- Integrales: Todos los departamentos o áreas de una empresa se relacionan entre sí, permitiendo de esta forma controlar los distintos procesos de la entidad, donde la salida de un proceso puede ser la entrada de otro.
- Modulares: Se dividen en módulos, que son las áreas que se integran como un todo para el manejo óptimo de la información. Estos módulos se instalan según las necesidades del cliente.
- Adaptables: Son diseñados para ser configurables según lo que cada empresa necesite.
- Base de datos centralizada.
- Sus componentes interactúan entre sí consolidando todas las operaciones.
- En un sistema ERP los datos se ingresan sólo una vez y deben ser consistentes, completos y comunes.
- Las empresas que lo implanten suelen tener que modificar alguno de sus procesos para alinearlos con los del sistema ERP. Este proceso se conoce como Reingeniería de Procesos, aunque no siempre es necesario.

1.2.1.2 Importancia.

Aplicando un ERP en una empresa se brinda apoyo a los clientes del negocio, se ofrecen respuestas rápidas a los problemas y se optimiza el manejo de información que permite la toma oportuna de decisiones y disminución de los costos totales de operación. Esto se obtiene gracias a que es un sistema integrado. De esta forma se evita tener varios programas que controlen todos los procesos de una

empresa, situación en la cual es más difícil lograr que la información no se duplique, que no aumente el margen de contaminación en la información y que no se cree un escenario favorable para malversaciones. Con un sistema ERP la información no se manipula y se encuentra protegida.

1.2.2 ¿Qué es un Inventario?

El inventario consiste en verificar físicamente los medios con que cuenta la organización. Su finalidad es llevar a cabo un registro de la existencia, cantidad, características, condiciones de uso, valor de los medios y las personas responsables de su manejo.

Al hacer un inventario hay que tener el mayor cuidado posible, para evitar repeticiones, para que no se incluyan materiales o mercancías que no correspondan. Un inventario es además una relación de los activos circulantes que posee la entidad en un momento dado y que pueden estar destinados para ser vendidos o como insumos en el proceso productivo (Gonzalez Brito, et al., 2005)

La gestión de inventario tiene como objetivo:

- Conocer con exactitud la cantidad de medios materiales en almacenamiento de una entidad.
- Llevar el control del uso de los medios materiales, verificando que se mantenga la cantidad y calidad adecuadas a las necesidades de la organización.
- Tener el control estricto de las entradas y salidas de los medios materiales del almacén.
- Asignar responsabilidades al personal encargado del uso para garantizar su cuidado y correcta utilización.
- Vigilar el buen uso de los medios, para prevenir reparaciones o reacondicionamientos y así prolongar su utilización.
- Determinar que las existencias físicas inventariadas correspondan al registro en los libros.

1.3 Situación actual de los procesos

En la actualidad no existe en Cuba una sistema informático integral de gestión que cumpla con la totalidad de los requerimientos de funcionalidad, interoperabilidad y seguridad que espera el gobierno cubano de

una solución de este tipo, de manera que pueda ser utilizada como herramienta para potenciar el cumplimiento de las funciones de las entidades a todos los niveles con un máximo de racionalidad y control de los recursos financieros, materiales y humanos (del Toro Ríos, et al., 2009).

1.4 Sistemas automatizados existentes

1.4.1 Sistemas nacionales

1.4.1.1 Versat-Sarasola

Es un producto cubano. Es el primer Sistema Integral de gestión de contabilidad certificado, desarrollado para la gestión económica eficaz y fiable. Actualmente es utilizado en Cuba en alrededor de 200 entidades de varias provincias y en lo adelante será introducido en más de dos mil 500 unidades presupuestadas. Este sistema integrado cuenta con un conjunto de 12 módulos entre los que se encuentran:

- Configuración y seguridad.
- Contabilidad general y de gastos.
- Costos y procesos.
- Análisis económico empresarial.
- Control de activos fijos.
- Finanza y caja.
- Planificación y presupuestos.
- **Control de inventarios.**
- Pago de salario.
- Paquete de gestión.
- Contratación.
- Facturación.

(del Toro Ríos, et al., 2009)

En el módulo de **Control de Inventarios** se definen formatos del clasificador de productos para lograr una uniformidad en el registro y la agregación de información en los reportes de salida, se conceptualizan los movimientos para lograr una información amplia sobre los orígenes y destinos de los recursos. Permite el control de las existencias y movimientos en diferentes monedas. Muestra el cuadre diario de cada uno de los almacenes por las diferentes cuentas. Ofrece la posibilidad de duplicar documentos para agilizar los pases de los mismos y lograr que un mismo documento se convierta en otro con solo adicionar un mínimo de información, realiza un control de las existencias y movimientos por custodios y se emiten diferentes reportes e información de utilidad para la correcta administración de los recursos materiales.

Características:

- Es una aplicación de escritorio.
- Implementado en Delphi.
- Trabaja sobre el sistema operativo Windows.
- Soporte para base de datos SQL Server 2000.

1.4.1.2 SISCONT5

El sistema se aviene a las definiciones y conceptos del Ministerio de la Industria Básica 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.

Puede ser explotado en régimen monousuario y multiusuario. Se define para monoentidad y multientidad, en esta última existe el control de su acceso para las entidades en un mismo equipo de cómputo como servidor (del Toro Ríos, et al., 2009).

El módulo de Inventario maneja toda la información referida al Submayor de Inventarios de la entidad, garantizando el cuadre permanente con las respectivas cuentas de la Contabilidad General. El sistema esta preparado para controlar el saldo de cada material en dos monedas, a partir de los procedimientos vigentes en el país en cuanto a política monetaria.

Permite el tratamiento de las contabilizaciones de forma transaccional, por resúmenes diarios o de forma personalizada según se defina por parámetros. Incorpora tratamiento de lotes a los productos y dos tipos de valoración de las cuentas FIFO y Promedio Ponderado (Minbas.cu, 2007).

1.4.1.3 Rodas XXI

Sistema multiempresa y multiusuario creado por CITMATEL para la automatización de la gestión empresarial. Contiene diferentes módulos que pueden usarse integrados o independientes:

- Contabilidad.
- Efectivo caja y banco.
- Nóminas.
- Activos fijos tangibles.
- **Inventarios.**
- Cobros y pagos.
- Facturación.
- Finanzas.
- Tele-cobranza.

Además, cuenta con el módulo Administrador, que brinda mayor integralidad al sistema y garantiza facilidades adicionales durante su instalación y explotación (del Toro Ríos, et al., 2009).

El módulo de **Inventario** de Rodas XXI le permite tener un control detallado de los inventarios de su entidad, realizando en el mismo momento que se registra un movimiento, su contabilización. Se pueden realizar todo tipo de operaciones de entradas y salidas de los almacenes con facilidad en el momento que se desee, generando el documento asociado al movimiento de que se trate de forma automática previa configuración del sistema para ello. Es posible trabajar con varios almacenes y a su vez cada uno de ellos de forma independiente.

En este módulo se hacen los cierres diarios cada vez que se realizan movimientos. En las opciones de informes es posible ver el cuadre diario y una variedad de informes de utilidad como el submayor de inventario, documentos emitidos tanto de entrada como de salida, la existencia y los movimientos de un producto o un grupo de productos, el importe por proveedores así como la existencia por cuentas, resumida o detallada.

Al igual que el resto de los módulos de Rodas XXI, el módulo de Inventario está pensado para garantizar la mayor seguridad y trazabilidad de las operaciones que se realizan en el mismo (RodasXXI.cu, 2007).

1.4.1.4 Valoración crítica.

Una vez analizados los sistemas implantados en Cuba, se concluye que no resultan soluciones factibles para las entidades cubanas debido a que fueron desarrollados sobre plataformas de software propietario, lo que implica incrementos de gastos en licencias de uso y mantenimiento del software. Además, las soluciones nacionales constituyen aplicaciones de escritorio lo que trae como desventaja que el usuario deba instalar la aplicación en cada estación de trabajo. Son productos que se caracterizan por abordar solamente partes del problema de la gestión de la empresa o la unidad presupuestada, no soportan mecanismos estándares de integración con otras aplicaciones donde casi ninguno bajo conceptos de informática multicapa y distribuida en la red.

1.4.2 Sistemas internacionales

1.4.2.1 Openbravo

Openbravo ERP ha sido específicamente diseñado para ayudar a las empresas a mejorar su rendimiento. La cobertura funcional del producto incluye todas las áreas típicas de un sistema de gestión integrado, destacando la solución de contabilidad.

Además, esta aplicación se integra de manera natural con otras áreas como la gestión de relaciones con clientes o CRM[2], inteligencia de negocio o BI (Business Intelligence) y terminales punto de venta o POS (Point of Sale).

Openbravo ERP utiliza tecnologías modernas, pero sólidas y suficientemente probadas, para cumplir los requerimientos estrictos de rendimiento y escalabilidad de cualquier entorno empresarial:

- Java y Javascript
- SQL y PL/SQL
- XML
- HTML

Los procesos de gestión de almacenes que incorpora Openbravo ERP permiten que las existencias en su organización estén siempre al día y correctamente valoradas. La posibilidad de definir la estructura de almacenes de su organización hasta el mínimo nivel (ubicación) facilita que los stocks estén siempre perfectamente localizados. Adicionalmente, las capacidades para gestionar los lotes de mercancías y la posibilidad de utilizar números de serie aseguran el cumplimiento de los requisitos de trazabilidad impuestos en la mayoría de industrias (Openbravo.com, 2009).

La Gestión de almacenes incluye:

- Almacenes y ubicaciones (multi-almacén).
- Stock por producto en doble unidad (por ejemplo, en kilogramos y cajas).
- Atributos del producto en almacén personalizables (color, talla, descripción de calidad, etc.).

[2] **CRM**: Customer Relationship Management

- Lote y número de serie.
- Impresión de etiquetas. Códigos de barras (EAN, UPC, UCC, Code, otras.).
- Gestión de bultos en almacén.
- Control de reposición.
- Trazabilidad configurable por producto.
- Movimiento entre almacenes.
- Gestión automática de salidas de stock (vaciado según existencias, con reglas de prioridad por caducidad, ubicación, etc.).
- **Inventario físico. Planificación de inventarios. Inventario continuado.**
- Informes de movimientos, seguimiento, stocks, entradas/salidas, caducidades, inventario, ubicaciones, etc. Informes personalizables.
- Integrado con Openbravo POS.
- Sincronización y control del stock en la misma tienda.

1.4.2.2 Condor

Sistema automatizado de alta complejidad y seguridad que abarca todos los aspectos del proceso contable de una entidad, tales como la dualidad de moneda y el pago por resultados. Está formado por varios módulos:

- Activos fijos.
- Contabilidad general.
- Nóminas.
- **Control de inventarios.**
- Condexce.
- Recursos humanos.

Este brinda mayor autonomía al cliente para efectuar cambios de estructura sin necesidad de la intervención de especialistas, quedando registrados de forma que puedan ser auditables. Incluye la contabilidad multimonedas (del Toro Ríos, et al., 2009).

Entre sus módulos se encuentra el de Gestión de Inventarios, el cual permite entre otras funcionalidades:

- Control de los esquemas de depósitos y almacenes.
- Realizar inventarios rotativos.
- Serializar los productos.
- Generar transacciones de costeo.

1.4.2.3 SAP

Software desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM. La corporación se ha desarrollado hasta convertirse en la quinta más grande compañía mundial de software. Cuenta con el módulo Controlling (CO), que permite el control de los gastos generales, costes de producto, cuenta de resultados y centros de beneficio.

El subsistema **Inventario** de SAP permite reducir los costes de almacenamiento, transporte, cumplimiento de pedidos y manipulación de materiales y, a la vez, mejorar el servicio al cliente. Puede mejorar significativamente la rotación de inventario, optimizar el flujo de mercancías y acortar las rutas en su almacén o centro de distribución. Entre los beneficios adicionales de la gestión de inventarios se encuentran la mejora del flujo de caja, la visibilidad y la toma de decisiones.

Para la gestión de almacenes, puede realizar un seguimiento de la cantidad y el valor de todos sus materiales, realizar un inventario físico y optimizar los recursos del almacén. Los empleados pueden planificar, introducir y documentar movimientos de almacén interno gestionando las entradas y las salidas de mercancías, el almacenamiento, la recogida y el embalaje, los traslados físicos, etc.

Entre sus características principales se encuentran:

- Implementado en .NET y WebSphere.

- SAP también ofrece una nueva plataforma tecnológica denominada SAP NetWeaver, esta plataforma tecnológica convierte a SAP en un programa Web-enabled, lo que significa que estaría totalmente preparado para trabajar con él mediante la web.
- Trabaja sobre el sistema operativo Windows.
- Soporte para bases de datos Oracle .

(SAP España, 2007)

1.4.2.4 Valoración crítica

Desde el punto de vista de la solución estos sistemas serían capaces de resolver el problema del módulo

Inventario Físico debido a su alto nivel de configuración y servicios que proveen, pero tienen la desventaja de que algunos de ellos utilizan tecnologías que no son accesibles a Cuba debido a las restricciones impuestas por Estados Unidos. Algunos sistemas como OpenBravo están basado en la plataforma J2EE cuya máquina virtual es propiedad de SUN, empresa norteamericana, aunque ha comenzado a liberar el código sigue estando bajo las leyes de su gobierno; además J2EE requiere un consumo de memoria elevado en comparación con PHP/Apache. Como otra desventaja aparece que el diseño de estos ERP ha sido para empresas capitalistas que tienen un modelo de gestión y de procesos muy diferente a las empresas o unidades presupuestadas cubanas donde la economía es centralizada y operan otros mecanismos. Por último, el resto de los softwares propietarios no constituyen una opción viable pues representan gastos muy elevados al país por conceptos de licencias y mantenimiento.

1.5 Modelo de desarrollo de software basado en componentes

La metodología es un proceso de software detallado y completo; define con precisión los artefactos, roles y actividades involucradas, junto con prácticas y técnicas recomendadas es por ello que la dirección del proyecto definió su propia metodología basada en un modelo de desarrollo.

Para un proyecto de esta magnitud es necesario que cada uno de los equipo de desarrollo posean un modelo estandarizado, así como una definición clara y precisa de las responsabilidades de cada uno de los roles que se ven involucrados en el desarrollo de la solución. (Centro de Soluciones de Gestión, 2008)

A continuación se muestra la tabla donde aparecen las responsabilidades definidas para cada uno de los roles en el proyecto.

Roles	Responsabilidades
Jefe de Línea de Desarrollo	Responsable de garantizar los cronogramas y compromisos de la línea Supervisar el proceso de desarrollo Organiza y controla el trabajo de los miembros de su línea Controla los indicadores de eficiencia
Planificador	Mantener actualizado el cronograma Mantener actualizada la plantilla de Capital Humano. Planificar y controlar las tareas de los miembros del equipo, según las prioridades Controlar los horarios de trabajo y distribución de máquinas. Llevar las actas de las reuniones y talleres. Controlar los planes de trabajo Individuales
Arquitecto de Sistema	Que se cumplan las políticas y estándares definidos en la Arquitectura. Las decisiones de integración en el proyecto y la Arquitectura del Sistema. Modera el Taller de Diseño.
Arquitecto de Datos	Construye y actualiza el Modelo de Datos, además responde por el manejo y recuperación de la información del mismo
Analista Principal	Dirigir y organizar el trabajo del grupo de analistas de la Línea. Elaborar el Mapa de Procesos de la Línea según los estándares. Participar en la definición y construcción de la Arquitectura de Negocio del ERP.

Especialista de Calidad	Revisar, controlar las normas y estándares que establece el grupo de aseguramiento de la calidad incluyendo el proceso de desarrollo. Guiar al grupo de auditoría y revisiones Coordinar el proceso de diseño de casos de prueba. Coordinar las pruebas de aceptación o liberación.
Especialista Funcional	Participar en las sesiones de trabajo para identificar, describir y validar los procesos de negocio y los requisitos de software Validar, desde el punto de vista funcional, los procesos de negocio y requisitos de software Elaborar Casos de Prueba según los estándares establecidos para ello
Analista	Participar en las sesiones de trabajo para identificar, describir y validar los procesos de negocio y los requisitos de software Elaborar la Descripción de Procesos de Negocio, Especificación de Requisitos y Casos de Prueba según los estándares establecidos para ello Participar en el Taller de Diseño
Desarrollador	Diseña y Construye los componentes de software de la línea

Tabla 1 Responsabilidades por roles definidas en el modelo de desarrollo del proyecto ERP Cuba.

1.6 Herramientas y tecnologías utilizadas

1.6.1 Herramientas de desarrollo de software

Actualmente se considera a las Herramientas de Desarrollo de Software (HDS) como herramientas basadas en computadoras que asisten el proceso de ciclo de vida de software, consolidadas en la literatura en la forma de ingeniería de software asistida por computadora (CASE, por sus siglas en inglés).

Permiten automatizar acciones bien definidas, reduciendo también la carga cognitiva del ingeniero, quien requiere libertad para concentrarse en los aspectos creativos del proceso. Este soporte se traduce en mejoras a la calidad y la productividad en el diseño y desarrollo. Las HDS automatizan metodologías de software y desarrollo de sistemas y se vinculan con los diferentes conceptos involucrados en el desarrollo.

El soporte que brindan al proceso de desarrollo proporciona importantes ventajas para el equipo de trabajo. Estas mejoras se sintetizan en:

- Apoyar a las metodologías y métodos, integrando actividades y propiciando visión de continuidad entre fases metodológicas.
- Mejorar la comunicación entre los actores involucrados, facilitándoles compartir su trabajo y desempeñarlo de forma dinámica e iterativa.
- Establecer métodos efectivos para almacenar y utilizar los datos, lo que permite organizar y correlacionar componentes, para acceder a estos a través de un repositorio.
- Agregar eficiencia al mantenimiento, ya que los programas son construidos sobre las mismas estructuras y estándares, facilitando la adherencia a la disciplina de diseño y facilitan también la conversión automática de programas a versiones más recientes de lenguajes de programación.
- Automatizar porciones del análisis y diseño tediosos y propensos a error, con influencia sobre la generación de código, las pruebas y el control. Resalta la consideración de que los beneficios potenciales sólo pueden ser alcanzados si las HDS son utilizadas de forma correcta.

1.6.1.1 Herramientas CASE

Visual Paradigm for UML 6.1

Visual Paradigm para UML[3] es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El

[3] **UML:** (Unified Modeling Language) Lenguaje Unificado de Modelado

software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Visual Paradigm es una herramienta CASE concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas entre otras opciones. Constituye una herramienta software libre de probada utilidad para el analista. Dentro de sus características se aprecia que soporta BPMN y UML versión 2.1. Muestra también:

- Diagramas de Procesos de Negocio.
- Modelado colaborativo con CVS y Subversión.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras.

(freedownloadmanager.org, 2008)

Lenguajes para el modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos o parte de este.

Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo.

UML (Lenguaje Unificado de Modelado)

UML (Unified Modeling Lenguaje) o Lenguaje Unificado de Modelado es un lenguaje basado en una notación gráfica la cual permite: especificar, construir, visualizar y documentar los objetos de un sistema

programado. UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones, ha sido ampliamente aceptado debido al prestigio de sus creadores. Hay que tener en cuenta que el estándar UML no es un proceso, no es una metodología de desarrollo, sino una notación un lenguaje (Gonzalez Brito, y otros, 2005).

De forma general las principales características son:

- Lenguaje unificado para la modelación de sistemas
- Tecnología orientada a objetos
- El cliente participa en todas las etapas del proyecto
- Corrección de errores viables en todas las etapas
- Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor
- Facilita a los integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, estos integrantes siendo los analistas, diseñadores, especialistas de área y desde luego los programadores.

1.6.1.2 Base de Datos

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.

Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc.

Las características de un Sistema Gestor de Base de Datos SGBD son: Abstracción de la información, Independencia, Redundancia mínima, Consistencia, Seguridad, Integridad, Respaldo y recuperación, Control de la concurrencia.

Un SGBD debe permitir:

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

(Cavsi.com, 2007)

PostgreSQL 8.3

PostgreSQL es un sistema de gestión de Bases de Datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES de la universidad de Berkeley. Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo, publicado bajo la licencia BSD.

A continuación se enumeran las principales características de este gestor de bases de datos:

- Implementación del estándar SQL92/SQL99.
- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta querys complejos, incluyendo subselects, integridad referencial (Foreign Keys), triggers, vistas (Views), integridad transaccional (ACID), control de versionado concurrente (MVCC)

(Netpecos.org, 2008)

Herramientas de Base de Datos

pgAdmin III

pgAdminIII es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y

Windows. Es capaz de gestionar versiones a partir de PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL entre ellas: Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres. Incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP[4] o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL[5] para mayor seguridad (Guia-ubuntu.org, 2007).

SQLmanager 2007

SQL Manager 2007 para PostgreSQL es una poderosa herramienta gráfica para administración del servidor de PostgreSQL y el desarrollo. Su interfaz gráfica es fácil de usar y un montón de características hacen el trabajo con PostgreSQL tan fácil como puede ser (Sqlmanager.net, 2007).

Principales características:

- Soporte de datos UTF8.
- Excelentes herramientas visuales y texto de consulta para la construcción.
- Nuevo estado de la técnica interfaz gráfica de usuario.
- Rápida gestión de bases de datos y la navegación.
- Avanzadas herramientas de manipulación de datos.
- Soporte completo de PostgreSQL hasta la versión 8.3.
- Mejora de la base de datos para facilitar su explorador gestión de todos los objetos de PostgreSQL.
- Eficaz de gestión de la seguridad.
- Potente base de datos de diseño visual.

[4] **TCP/IP:** (Transmission Control Protocol/Internet Protocol) Protocolo de control de transmisión/Protocolo de Internet

[5] **SSL:** (Secure Sockets Layer) Protocolo de Capa de Conexión Segura

- Conexión de puerto local a través de la transmisión a través de un túnel SSH[6].
- Acceso a servidor PostgreSQL a través de protocolo HTTP[7].

1.6.1.3 IDE de desarrollo

Un IDE[8] es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante pluggins se le puede añadir soporte de lenguajes adicionales.

Zend Studio for Eclipse 6.0

Zend Studio para Eclipse posee las capacidades para crear poderosas herramientas web. Permite a los desarrolladores crear estrategias de integración más eficientes. Al proporcionar potentes capacidades de acción con el lenguaje PHP, mejoras de soporte con JavaScript y profunda integración a Zend Framework, el desarrollo de aplicaciones se realiza en un tiempo récord. Incluye un poderoso editor de código con soporte para todos los formatos web. Posee un fuerte manejo de la arquitectura Cliente/Servidor, excelente depuración, elaboración de perfiles y un código de cobertura. Zend estudio tiene todas las

[6] **SSH:** (Secure Shell) Intérprete de órdenes seguro

[7] **HTTP:** (HyperText Transfer Protocol) Protocolo de Transferencia de Hipertexto

[8] **IDE:** (Integrated Development Environment) Entorno de desarrollo integrado

herramientas que un desarrollador necesita para asegurarse de que el código es correcto y empezar a diagnosticar problemas con antelación. Tiene características de depuración avanzadas, incluyendo: condiciones límites, visualización de errores, variables y buffer de salida. Asegura la protección máxima de ubicaciones de proyectos o en Internet con depuradores remotos (Zend.com, 2007).

1.6.2 Tecnologías Web

1.6.2.1 Lenguajes de programación del lado del cliente

HTML

HTML[9] es un lenguaje de programación muy sencillo que se utiliza para crear los textos y las páginas web. Es un lenguaje que se basa en las marcas para crear los hipertextos. Permite que se creen enlaces entre distintas partes del mismo documento o entre distintas fuentes de información a través de hiperenlaces o hipervínculos, e incluso insertar otros elementos como imágenes y sonidos. (Lanzillotta, 2004)

No es mas que un conjunto de etiquetas o comandos, complementados en la mayoría de los casos por extensiones que permiten dar formato a un archivo, con el objetivo básico de crear un documento que pueda ser visualizado en ambiente Internet en forma de Página Web y que esta, además, pueda, por medio de dichas etiquetas, tener la estructura o forma deseada por quien la diseñó. Las etiquetas HTML tienen, por lo general, una etiqueta de apertura y una de cierre; aunque existen algunas excepciones en las que solo basta con colocar la de apertura. (Cova, 2004)

JavaScript

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web, puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos.

Entre los diferentes servicios que se encuentran realizados con JavaScript en Internet se encuentran: Correo, Chat, Buscadores de Información.

[9] **HTML:** (Hyper Text Markup Language) Lenguaje de Marcas de Hipertexto

También podemos encontrar o crear códigos para insertarlos en las páginas como: Reloj, Contadores de visitas, Fechas, Calculadoras, Validadores de formularios, Detectores de navegadores e idiomas (Maestrosdelweb.com, 2007).

Ajax

AJAX, acrónimo de Asynchronous JavaScript And XML[10] (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

XHTML (o HTML) y hojas de estilos en cascada (CSS[11]) para el diseño que acompaña a la información.

DOM[12] accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y Script, para mostrar e interactuar dinámicamente con la información presentada.

El objeto XMLHttpRequest[13] para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano y JSON.

AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente (Webexperto.com, 2006).

[10] **XML: (**Extensible Markup Language) Lenguaje de Etiquetado Extensible

[11] **CSS:** (Cascading Style Sheets) Hojas de Estilo en Cascada

[12] **DOM:** (Document Object Model) Modelo en Objetos para la representación de Documentos

[13] **XMLHttpRequest**: Interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB.

Json

JSON [14] es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

El formato JSON es el más adecuado para la respuesta del servidor cuando la acción Ajax debe devolver una estructura de datos a la página que realizó la llamada de forma que se pueda procesar con JavaScript. Este mecanismo es útil por ejemplo cuando una sola petición Ajax debe actualizar varios elementos en la página.

JSON se ha convertido en un estándar en el desarrollo de aplicaciones web. Los servicios web proponen la utilización de JSON en vez de XML para permitir la integración de servicios en el navegador del usuario en vez de en el servidor. El formato JSON es seguramente la mejor opción para el intercambio de información entre el servidor y las funciones JavaScript (Json.org, 2007).

CSS

CSS es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación, permite crear páginas web de una manera más exacta, gracias a las CSS el desarrollador es mucho más dueño de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas.

[14] **JSON:** (JavaScript Object Notation) Notación de Objetos de JavaScript

Entre los beneficios concretos de CSS encontramos:

- control de la presentación de muchos documentos desde una única hoja de estilo.
- control más preciso de la presentación.
- aplicación de diferentes presentaciones a diferentes tipos de medios (pantalla, impresión, etc.).
- numerosas técnicas avanzadas y sofisticadas.

(Desarrolloweb.com, 2008)

1.6.2.2 Lenguaje de programación del lado del servidor

Un lenguaje de programación es 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 su estructura y 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. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

Se les clasifica como lenguajes del lado del servidor a los lenguajes de programación en la tecnología cliente servidor que se ejecutan del lado del servidor y de los que los cuales los usuarios solo obtienen el beneficio del procesamiento de la información.

PHP

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar. Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante.

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, y ODBC[15], por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

No es un lenguaje de marcas y la meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. PHP también soporta el uso de otros servicios que usen protocolos como IMAP[16], SNMP[17], POP3[18], HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos.

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas: Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader), analizar código XML. Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.

(Desarrolloweb, 2009)

1.6.2.3 Arquitectura

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Esta se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más

[15] **ODBC:** (Open Database Connectivity) Estándar de acceso a Bases de datos.

[16] **IMAP:** (Internet Message Access Protocol) Protocolo de acceso a mensajes almacenados en un servidor.

[17] **SNMP:** (Simple Network Management Protocol) Protocolo de la capa de aplicación que facilita el intercambio de información de administración.

[18] **POP3:** (Post Office Protocol) Protocolo 3 de Correo, está diseñado para recibir correos, no para enviarlo.

recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

Patrón Modelo-Vista-Controlador (MVC)

MVC es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de los conceptos para que el desarrollo este estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. MVC sugiere la separación del software en 3 estratos.

Modelo: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en un contexto del sistema proveen de información al usuario o a la aplicación misma.

Vista: Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación web la "Vista" es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

El **Modelo** es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema).
- Llevar un registro de las vistas y controladores del sistema.
- Si se encuentra ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El **Controlador** es responsable de:

- Recibir los eventos de entrada.

- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las **Vistas** son responsables de:

- Recibir datos del modelo y lo muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

En la siguiente figura se muestra la estructura del patrón Modelo-Vista-Controlador.

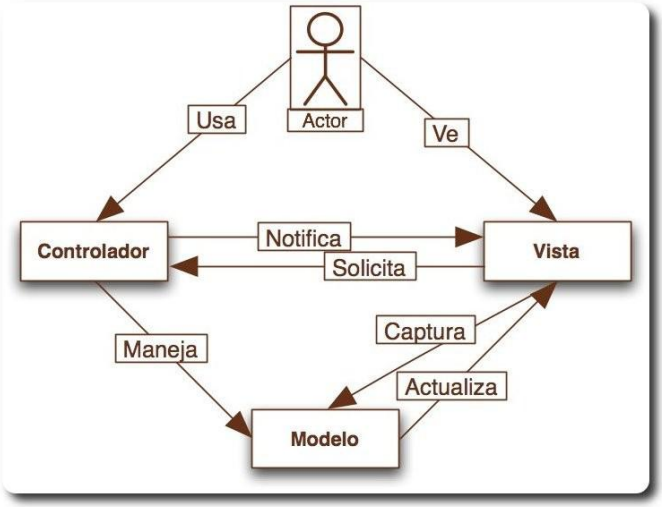

Figura 1 Estructura Modelo-Vista-Controlador

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

Ventajas:

Soporte de múltiples vistas: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos

datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes.

Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Desventaja:

Costo de actualizaciones frecuentes: Si el modelo experimenta cambios frecuentes, por ejemplo, podría desbordar las vistas con una lluvia de requerimientos de actualización.

Arquitectura Cliente-Servidor

Esta arquitectura consiste básicamente en un programa cliente que realiza peticiones a otro programa -servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

Ventajas

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema.

Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P[19]).

- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

Desventajas

- La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.
- El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.
- El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste.

[19] **P2P: (**Peer to Peer) Red de pares, son redes que funcionan sin clientes ni servidores fijos.

- El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

Servidor Web Apache

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

Entre sus principales características se encuentran:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si queremos ver que es lo que estamos instalando como servidor, lo podemos saber, sin ningún secreto, sin ninguna puerta trasera).
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un módulo para realizar una función determinada.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs[20]. Este permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor.

(Ciberaula.com, 2008)

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

1.6.2.4 Frameworks

Un frameworks es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.

Un framework permite separar en capas la aplicación:

- La lógica de presentación que administra las interacciones entre el usuario y el software.
- La lógica de dominio o de negocio, que manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.
- La lógica de datos que permite el acceso a un agente de almacenamiento persistente u otros.

Zend Frameworks

Es un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es código abierto y trabaja con PHP 5.

Presenta entre otras las siguientes características:

- Simplifica la gestión de archivos de configuración.
- Proporciona los componentes que forma la infraestructura del patrón Modelo-Vista-Controlador.
- Proporciona una capa de acceso a base de datos, construida sobre PDO[21] pero ampliándola con diferentes características.

[21] **PDO:** (PHP Data Objects) Capa de abstracción de acceso a datos para PHP

- Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo)
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron

(Echarte, 2007)

En la siguiente figura se muestra la estructura de Zend Framework.

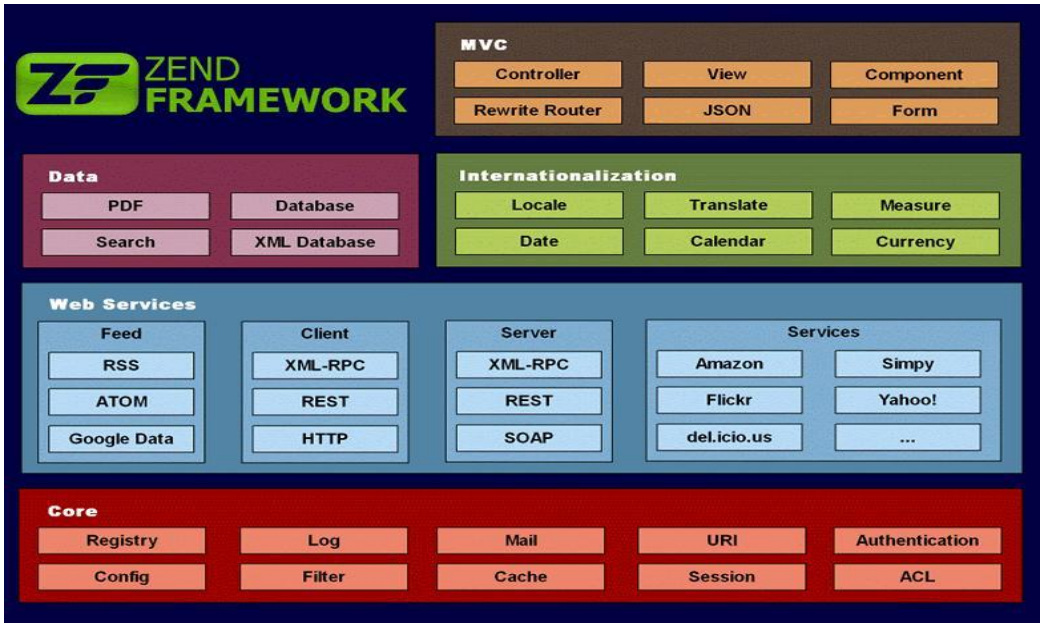

Figura 2 Estructura de Zend Framework

Zend_Ext Framework.

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

Doctrine Framework

Es un potente y completo sistema ORM (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 orientado a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos (Doctrine-Project.org, 2008).

En la siguiente figura se muestra la estructura de Doctrine ORM.

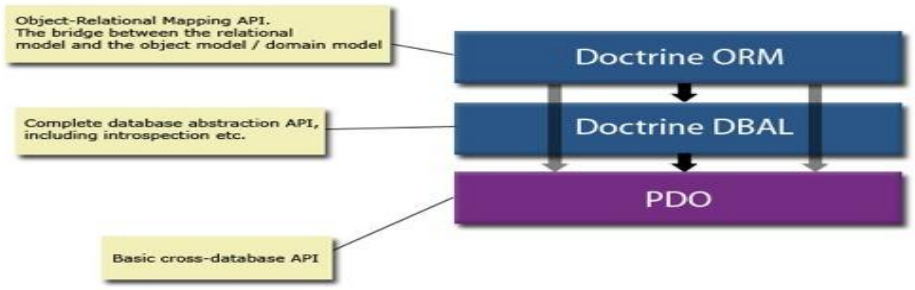

Figura 3 Estructura de Doctrine ORM

ExtJs Framework

Es una librería construida con JavaScript que proporciona una interfaz a las famosas librerías de Yahoo, jQuery, y Prototype + Scriptaculous, su potencia radica en la rica colección de componentes para el diseño de GUI's del lado del cliente haciendo uso extensivo de Ajax.

ExtJS es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, ExtJS no se preocupará de lo que pase en el servidor. Hay docenas de widgets a escoger en ExtJS, incluyendo composiciones automáticas de páginas, pestañas, menús, barras de herramientas, diálogos, vistas en árbol. Proporciona un selector de nodos DOM extremadamente poderoso llamado DomQuery (puede usarse como una librería independiente, pero en el contexto de ExtJS se usará para seleccionar elementos para poder interactuar con ellos a través de la interfaz

Element, contiene mucho de los métodos y propiedades de DOM que se necesitará proporcionando una interfaz conveniente, unificada y multinavegador).

Entre los componentes que esta librería ofrece encontramos cuadros de diálogo, menús, tablas editables, layouts, paneles, pestañas y todo lo necesario para construir atractivos desarrollos al estilo de Web 2.0.

Ventajas:

- La orientación a objetos intensa hará modular todos los scripts.
- El diseño está completamente separado de la funcionalidad.
- Funciones comunes como validación, comboxes editables, ventanas arrastrables (con minimizar y maximizar), grillas editables, son muy fáciles de implementar.
- Buena y amplia documentación, así como también su comunidad.

Desventajas:

- Crear un sistema serio con esta herramienta requiere un previo uso prolongado, ya que se dificulta el manejo de los nuevos objetos en su extensa y bien documentada API. El tiempo de aprendizaje puede llegar a compararse con aprender a programar en un lenguaje nuevo.
- Al estar todo el sitio en JS, no podrá ser accesible para los buscadores, limitando su uso a sistemas y no sitios web.
- Si se desea algún objeto y no está, se debe asumir la compleja tarea de crear un nuevo objeto (sólo apto para programadores JS avanzados).

(Wordpress.com, 2007)

UCID Framework

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

1.6.2.5 Navegador

Un navegador, navegador red o navegador web (del inglés, *web browser*) es un programa que permite visualizar la información que contiene una página web (ya esté esta alojada en un servidor dentro de la World Wide Web o en uno local).

El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

Mozilla Firefox 3.0

Es el nuevo e innovador navegador open source acerca del cual todo el mundo habla. Firefox ha sido creado por el proyecto Mozilla, un esfuerzo open source sin ánimo de lucro que incluye a miles de voluntarios alrededor del mundo. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. El apoyo organizativo del proyecto Mozilla es proporcionado por Mozilla Foundation (en los Estados Unidos de América), Mozilla Europe y Mozilla Japón.

Por nombrar algunas de las posibilidades que ofrece Firefox y que no ofrece IE[22], están:

- Es Software libre.
- En Firefox no existen la cantidad de bugs que posee el catastrófico IE, inmediatamente se encuentra un bug en el producto es notificado al Proyecto Mozilla para que sea reparado el problema.
- Navegación por tabs: Esta es una de las principales características que tiene Firefox.
- También existen excelentes extensiones de fácil instalación, estos mejoran la usabilidad y el aspecto del navegador, cosa que no se logra en IE el cual siempre permanece con los mismos colores.

(Firefox, 2009)

[22] **IE:** Internet Explorer

1.6.3 Control de Versiones

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:

- Mecanismo de almacenaje de los elementos que deba gestionar (ej. archivos de texto, imágenes, documentación...).
- Posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente pudiendo volver o extraer un estado anterior del producto).

Subversion 1.4.5

Subversión es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido. CVS[23] considerado su antecesor es uno de los controladores de versiones más utilizados en proyectos de software libre, sin embargo, a pesar de su amplio uso, el mismo diseño de CVS resultó ineficiente para diversos grupos de usuarios, y ante estas inconformidades se dio inicio al proyecto que hoy es conocido como Subversión, el mismo que ha empezado a socavar el dominio de CVS.

Ventajas

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O (1)) y no lineal (O (n)) como en CVS.

[23] **CVS:** Concurrent Versions System

- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP[24], PAM[25]).

Existen varias interfaces a Subversión, ya sea programas individuales como interfaces que lo integran en entornos de desarrollo.

- TortoiseSVN. Provee integración con el explorador de Windows. Es la interfaz más popular en este sistema operativo.
- Subclipse. "Plugin" que integra Subversión al entorno Eclipse.
- Subversive. "Plugin" alternativo para Eclipse.
- ViewVC. Interfaz web, que también trabaja delante de CVS.

Para Mac, pueden emplearse los interfaces SvnX, RapidSVN y Zigversion.

(Osmosislatina.com, 2009)

1.7 CONCLUSIONES PARCIALES

En el presente capítulo se ha tratado de forma general y resumida los diferentes sistemas que abarcan la gestión de inventario físico, determinando las ventajas, desventajas y factibilidad de su utilización en las entidades cubanas; además se realizó un estudio de las herramientas, tecnologías y la metodología propuesta por la dirección de arquitectura del proyecto, permitiendo sentar las bases para el desarrollo.

[24] **LDAP:** (Lightweight Directory Access Protocol) Protocolo Ligero de Acceso a Directorios

[25] **PAM:** Pluggable Authentication Modules

Capítulo 2: Descripción de la Solución Propuesta

2.1 Introducción

En este capítulo se realiza una profunda valoración de los artefactos propuesto por el analista de sistema exponiendo las principales ventajas y desventajas que ofreció la obtención de los mismos y se delimitan varios puntos importantes en el desarrollo del módulo como:

- Se hace un análisis de posibles implementaciones, componentes o módulos ya existentes.
- Se describen las principales clases utilizadas en la implementación del mismo.
- Se describen los estándares de codificación para una mejor legibilidad del código.

2.2 Flujo de procesos

2.2.1 Valoración crítica de los artefactos propuestos por los analistas

La obtención de la especificación de requisitos propuesto por los analistas, resultó de gran importancia, pues permitió una mejor comprensión de los aspectos relacionados con los requisitos funcionales y componentes reutilizables. Se pretende mostrar cómo está aprovechada la arquitectura y las posibilidades que proporcionan los marcos de trabajo utilizados en la programación de la aplicación, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. Se muestra el diagrama de componentes y el diagrama de despliegue propuesto por el equipo de arquitectura.

La principal fuente de información que brinda el analista es la especificación de requisitos donde estos no son más según la IEEE[26] Standard Glossary of Software Engineering Terminology:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Una representación documentada de una condición o capacidad como en 1 o 2.

[26] **IEEE:** (Institute of Electrical and Electronics Engineers) Instituto de Ingenieros Eléctricos y Electrónicos

El propósito de la especificación de requerimientos es reunir en un documento escrito los requisitos de todo el sistema de software o parte de él. Esto con la finalidad de plasmar qué es el sistema y cuál es su alcance.

Los requisitos se pueden clasificar en funcionales y no funcionales.

Requerimientos funcionales: Son capacidades o condiciones que el sistema debe cumplir.

Requerimientos no funcionales: Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Existen múltiples categorías para clasificar a los requerimientos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

- Requerimientos de Software
- Requerimientos de Hardware
- Restricciones en el diseño y la implementación
- Requerimientos de apariencia o interfaz externa
- Requerimientos de Usabilidad
- Requerimientos de Soporte
- Requerimientos Legales
- Requerimientos de confiabilidad
- Requerimientos de interfaz Interna
- Requerimientos de Seguridad

Los requisitos funcionales descritos por los analistas y que posibilitaron la identificación de las clases y las funcionalidades a implementar son:

Módulo Inventario

Componente Inventario:

- RF 1: Gestionar hojas de inventario físico.
  - RF 1.1 Adicionar hoja de inventario físico.
  - RF 1.2 Modificar hojas de inventario físico.
  - RF 1.3 Eliminar hojas de inventario físico.
  - RF 1.4 Buscar hojas de inventario físico.
  - RF 1.5 Búsqueda avanzada de hojas de inventario físico.
- RF 2 Confirmar hojas de inventario físico.
- RF 3 Aprobar hojas de inventario físico.
- RF 4 Cancelar estado a hojas de inventario físico.
- RF 5 Registrar conteo final.
- RF 6 Actualiza conteo final sin diferencia.
- RF 7 Gestionar productos de las hojas de inventario físico.
  - RF 7.1 Registrar Productos de la hoja de inventario físico.
  - RF 7.2 Buscar productos de inventario físico.
  - RF 7.3 Eliminar productos de la hoja de inventario físico.
- RF 8 Ayuda inventario físico.
- RF 9 Imprimir modelo de hojas de inventario físico.
- RF 10 Imprimir listado de hojas de inventario físico.

Componente planes de conteo:

- RF 11 Gestionar planes de conteo físico.
  - RF 11.1 Crear plan de conteo físico.
  - RF 11.2 Modificar plan de conteo físico.
  - RF 11.3 Modificar todos los meses del plan de conteo físico.
  - RF 11.4 Eliminar plan de conteo físico.

- RF 11.5 Buscar un plan de conteo físico.

- RF 12 Confirmar plan de conteo físico.
- RF 13 Aprobar plan de conteo físico.
- RF 14 Rechazar plan de conteo físico.
- RF 15 Cancelar estado de un plan de conteo físico.
- RF 16 Ayuda de planes de conteo físico.
- RF 17 Imprimir modelo de planes de conteo físico.
- RF 18 Imprimir listado de planes de conteo físico.

2.2.2 Objetos de automatización

Para realizar el proceso inventario físico de medios materiales en almacenamiento, anualmente se elabora un “Plan de Conteo” y cada vez que se realicen estos conteos se confecciona el modelo “Hoja de Inventario” que puede ser parcial mensual, parcial o general, con la creación de estas se controlan los medios y se detectan las diferencias entre las existencias reales y las existencias en los registros. Estas actividades se harán de forma automatizada permitiendo conocer la cantidad de veces que ha sido contado un medio material en un período de tiempo.

2.2.3 Propuesta del sistema

Al iniciar la sesión el sistema verificará los privilegios que posee el usuario logueado, permitiendo acceder solo a las entidades y Subsistemas que tenga acceso, así como a las funcionalidades y las distintas acciones definidas según el rol que desempeña. Para poder llevar a cabo los procesos de Gestión de los planes de conteo y las hojas de inventario físico presentes en el Subsistema de Inventario se muestran dos funcionalidades:

- Gestionar planes de conteo.
- Gestionar hojas de inventario.

Si se selecciona la primera funcionalidad el sistema carga los planes de conteo definidos para el depósito actual donde se puede realizar las siguientes acciones:

- Permite crear un plan de conteo si no existe todavía ninguno para ese depósito en ese año.
- Modificar un plan seleccionado.

  Donde se pueden realizar varias acciones como:

  - Modificar todos los meses donde se seleccionan los grupos de productos a repetir en los demás meses que no se encuentren estado preparado ni aprobado.
  - Resetear grupos donde se eliminan todos los grupos de productos asociados a un mes.
  - Confirmar un mes de un plan de conteo donde el estado pasa a ser preparado.
  - Aprobar un mes de un plan de conteo donde el estado pasa a ser aprobado.
  - Cancelar el estado de un mes de un plan de conteo donde el estado pasa a ser en elaboración.
  - Imprimir el modelo del plan mostrando una matriz con todos los meses del año y los grupos asociados a cada mes.
  - Ayuda que se encuentra en todas las interfaces para mejora la integración entre el usuario y la aplicación.
- Eliminar un plan de conteo para esto se verifica que el plan no tenga meses en estado de preparado ni aprobados.
- Confirmar el plan de conteo donde el estado pasa a ser preparado.
- Aprobar el plan de conteo donde el estado pasa a ser aprobado.
- Cancelar el estado de un plan de conteo donde el estado pasa a ser en elaboración.
- Se puede imprimir tanto el listado de los planes que posee el depósito como el modelo.
- Ayuda que se encuentra en todas las interfaces para mejora la integración entre el usuario y la aplicación.

Además se pueden realizar búsquedas por diferentes criterios como estado y año.

Si se selecciona la segunda funcionalidad el sistema carga las hojas de inventario definidas para el depósito actual donde se pueden realizar varias operaciones.

Se permite crear un documento de inventario físico que avala la búsqueda de los documentos por diferentes criterios y el registro de productos. Los criterios de búsqueda son los siguientes: por año de creación, estado, número del documento, tipo de inventario, así como por el creador o el encargado de aprobar el mismo y por rango de fecha. Para la realización de alguna acción sobre el documento hoja de inventario el sistema verifica primeramente que no esté siendo usado por otro usuario, de ser así el sistema dará la opción de ver el documento en forma de reporte, de lo contrario según los privilegios que posea podrá realizar diferentes acciones según el estado del documento:

- Gestionar parcialmente los productos del inventario (adicionar, eliminar, buscar).
- Registrar el conteo final donde se especifica un conteo para cada producto.
- Permite confirmar un documento, este debe tener productos y debe existir un conteo registrado para cada uno, cambiando el estado del documento a preparado.
- Permite aprobar un documento cambiando el estado del documento a aprobado.
- Permite cancelar un documento cambiando el estado del documento a elaboración.
- Mostrar el documento en forma de reporte sin poderse realizar ningún cambio.

Si no existe ningún documento el sistema permite crear nuevos documentos donde una vez elaborados se puede modificar, eliminar y mostrar el documento en formato de reporte para su visualización y futura impresión en caso de que el usuario lo desee. Recordar siempre que según el rol será la visibilidad o no de las acciones.

2.3 Análisis de reutilización de componentes, código o módulos

A continuación se muestran los componentes que son reutilizados para la implementación del módulo Inventario Físico.

Componente documento

Dentro de los componentes que se utilizan en la implementación del módulo de Inventario se encuentra el *Componente documento* que tiene como función la de gestionar todos los documentos del Subsistema Inventario. Este se realizó debido a que la mayoría de los módulos existentes en dicho subsistema realizaban las mismas acciones pero con un comportamiento diferente, siendo este el padre de los documentos de inventario. Cada módulo tiene su clase documento que hereda de la clase documento modelo general (DocumentoModel), donde se encuentran todas las generalidades de los módulos (**Inventario**, Ajuste, Despacho, Recepción, etc.). Se acceden a estos mediante llamadas por el Integrador (IoC) y un patrón proxy que realiza la función de interfaz entre el integrador y los documentos modelos de cada módulo.

Componente producto

Dentro de los componentes que se utilizan en la implementación del módulo de Inventario se encuentra el *Componente producto* que es el encargado de la gestión de los productos dentro del almacén, ya sea los que están nomenclados, como los nuevos que se incluyen. Este tiene un funcionamiento parecido al Componente documento en lo que respecta a estructura y el acceso al mismo, una clase Producto Modelo general donde heredan de él varias clases de diferentes módulos, se acceden a estos mediante llamadas por el Integrador (IoC) y un patrón proxy que realiza la función de interfaz entre el integrador y los documentos modelos de cada módulo. Este brinda todos los servicios necesarios para operar sobre los productos, que a través de los movimientos se relacionan con los documentos. Los productos pueden ser equipos, no equipos o munición, también tienen categorías según la calidad del mismo, además se registra la nacionalidad, el tipo de control, el número de pieza y otros atributos que son importantes para los almacenes.

Componente movimiento

Dentro de los componentes que se utilizan en la implementación del módulo de Inventario se encuentra el *Componente movimiento* que tiene una función muy importante, pues es el encargado de la unión entre los documentos y los productos. Dicho componente facilita todas las operaciones que son necesarias para operar con los movimientos de ese documento. Un movimiento es una copia que se le hace al producto para no afectar los valores reales del mismo y una vez contabilizado el documento serán afectados los

valores del producto con los del movimiento. Dentro del componente de movimiento se encuentra también el movimiento de los productos de inventario, para el caso del módulo Inventario.

2.4 Integración entre componentes

2.4.1 Estrategias de integración

Nuestra aplicación esta definida por 3 capas: capa de presentación (view), negocio (controller) y acceso a datos (models), esta arquitectura posibilita un trabajo seguro, rápido y eficiente. La integración vertical o llamada arquitectura en 3 capas consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatro nodos de integración, el que encontramos entre la vista y el controlador, el que está entre el controlador y el modelo, el que vincula el modelo con el framework doctrine y el que se encuentra entre el doctrine y la base de datos. Todo el código dentro de un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes módulos y componentes se realiza mediantes llamadas a la inversión de control. El IoC especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

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

En la siguiente figura se muestra la colaboración entre clases definidas por arquitectura.

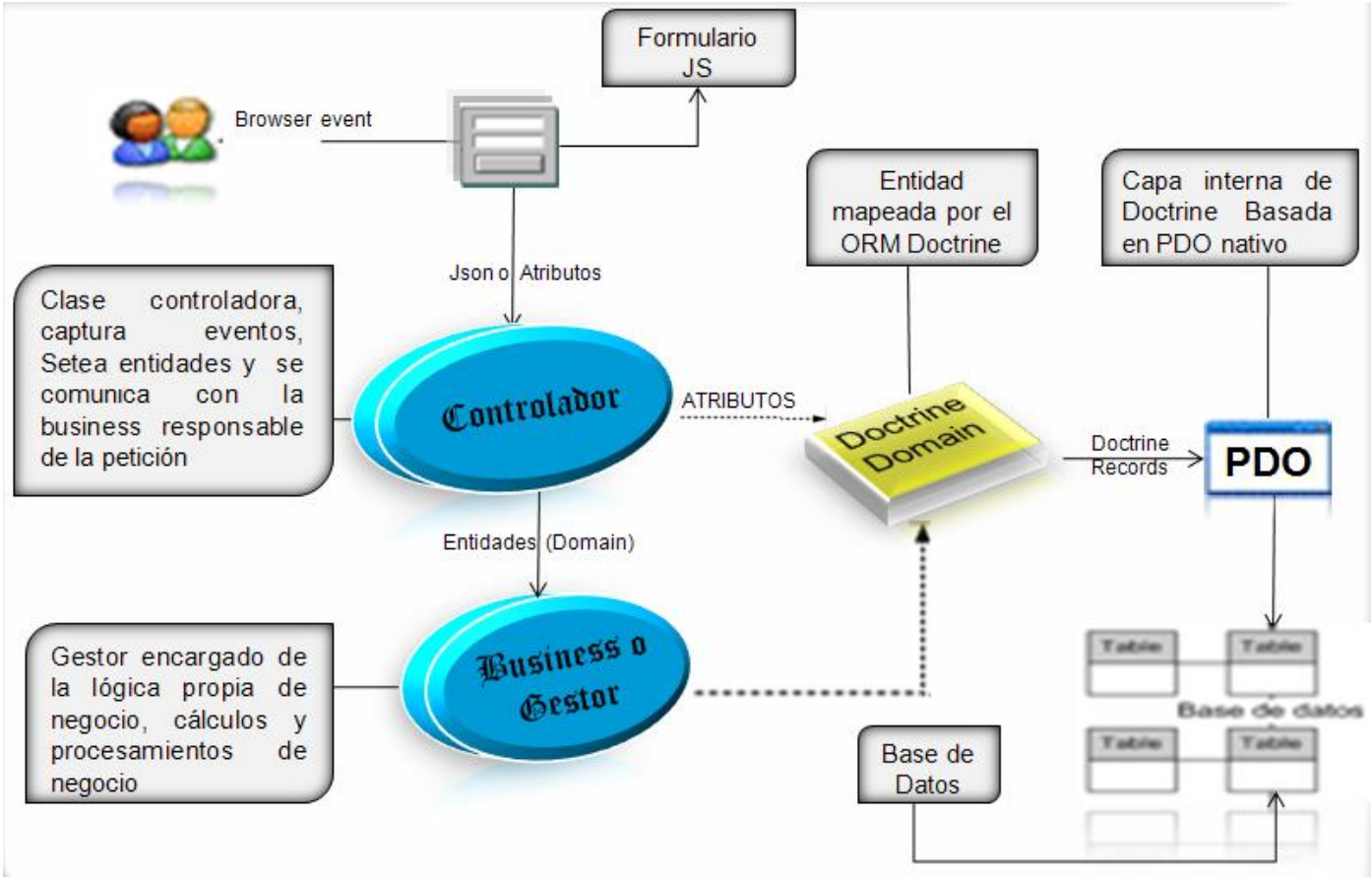

Figura 4 Colaboración entre clases definidas por Arquitectura

2.4.2 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, librerías compartidas, módulos, ejecutables, o paquetes.

En la siguiente figura se muestra el diagrama de componente del Módulo de Inventario Físico.

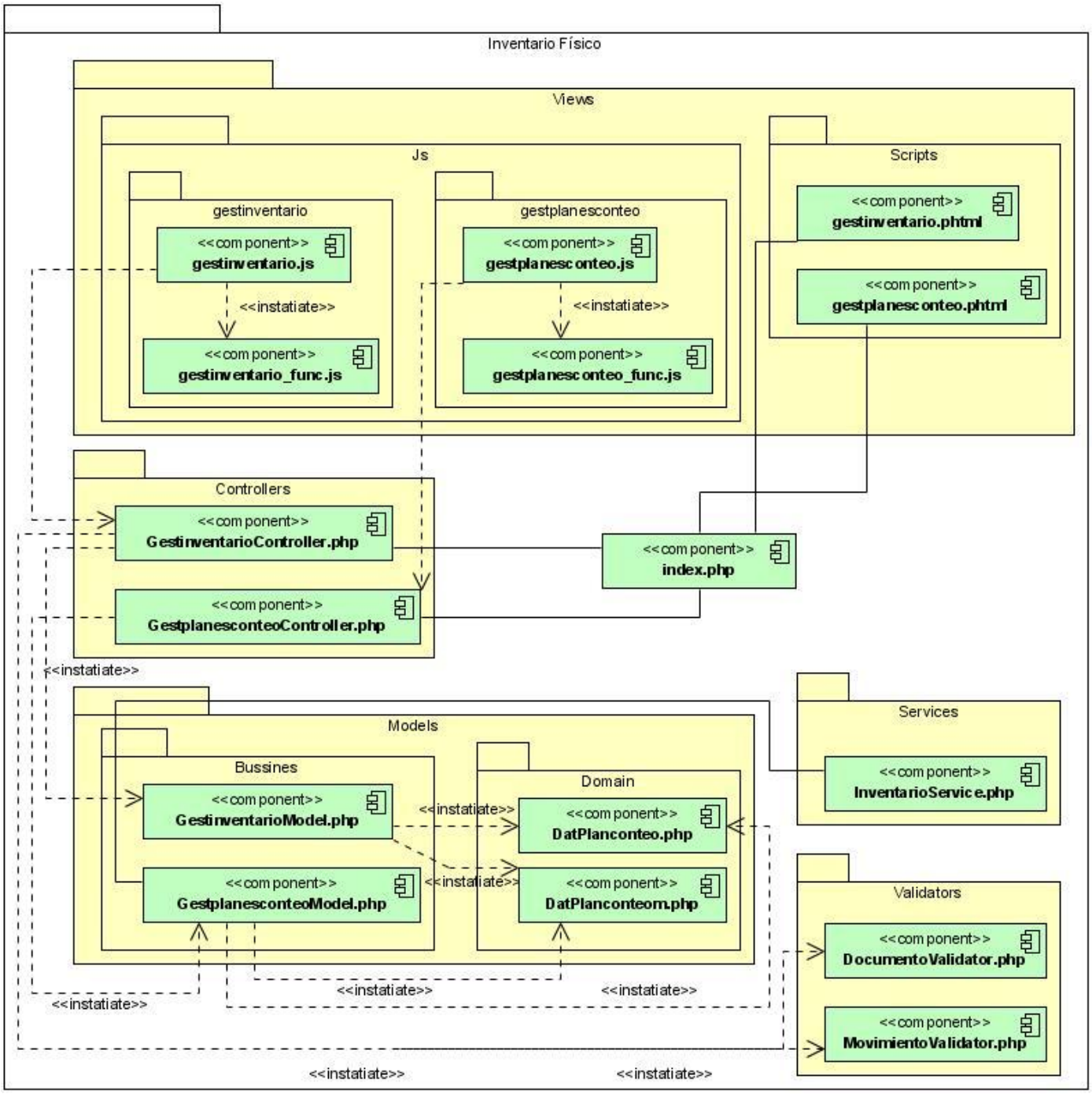

Figura 5 Diagrama de componentes del Módulo de Inventario Físico.

2.4.3 Diagrama de integración de componentes

En la siguiente figura se muestra el diagrama de integración de componentes del módulo inventario físico.

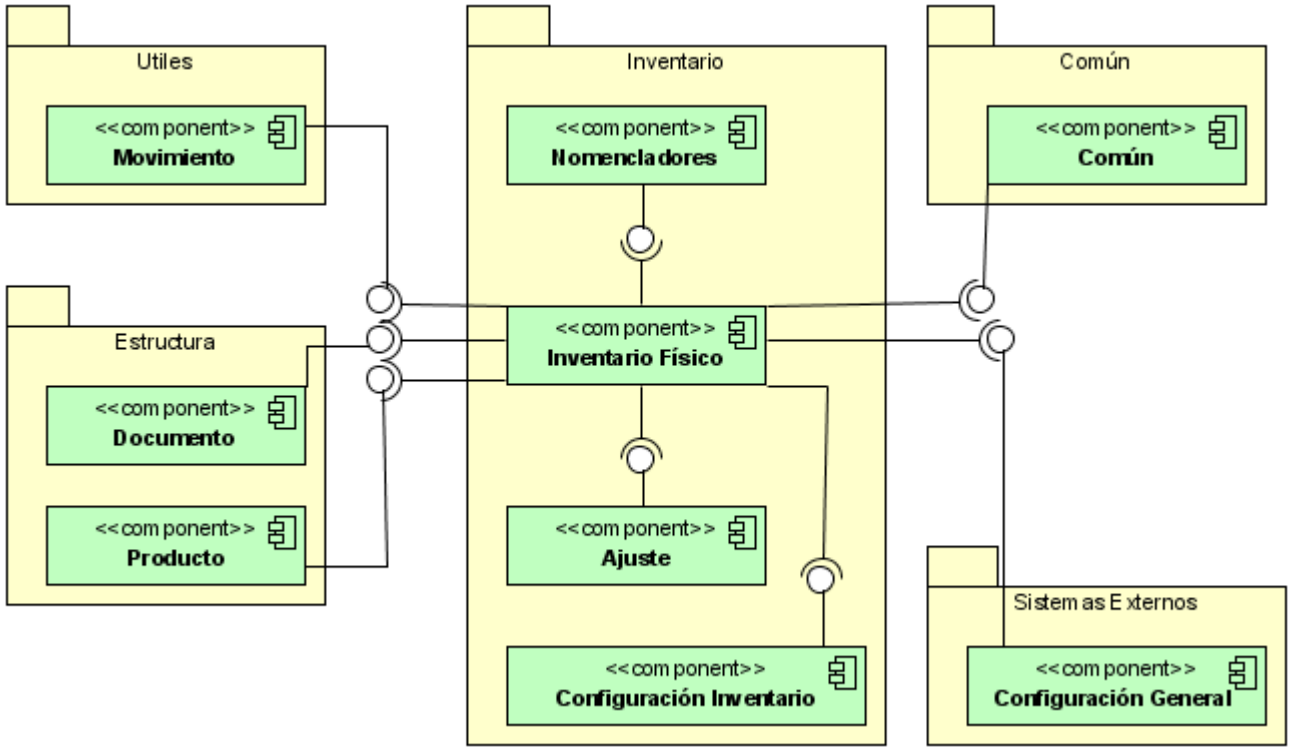

Figura 6 Diagrama de integración de componentes del Módulo Inventario Físico.

2.5 Estándares de codificación

2.5.1 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. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: Gestionarinventario

2.5.2 Nomenclatura según el tipo de clases

Clases controladoras

Las clases controladoras después del nombre llevan la palabra: “Controller”.

Ejemplo: GestionarinventarioController

Clases de los modelos

Business

Las clases que se encuentran dentro de Business después del nombre llevan la palabra: “Model”.

Ejemplo: GestionarinventarioModel

Domain

Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos.

Ejemplo: DatInventario

Generated

Las clases que se encuentran dentro de Generated el nombre comienza con el prefijo “Base” y seguido el nombre del dominio correspondiente.

Ejemplo: BaseDatInventario

2.5.3 Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing, que con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: generarHojaInventario

En caso de ser una acción de la clase controladora se especifica el nombre de la acción todo en minúscula y seguida el sufijo”Action”.

Ejemplo: generarhojainventarioAction

2.5.4 Nomenclatura de las constantes

El nombre a emplear para las constantes se escribe con todas las letras en mayúscula.

Ejemplo: INVENTARIO.

2.5.5 Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación *CamelCasing,* en la que se comienza con un prefijo según el tipo de datos.

Ejemplo: $arrMoneda

Prefijos para los tipos de datos

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

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
float	flt
Boolean	Boo

Tabla 2 Prefijos a utilizar en la creación de variables.

2.5.6 Normas de comentariado

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda aumentar su mantenibilidad a lo largo del tiempo.

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 las clases

Antes de la declaración de una clase se escribe una breve descripción donde se explique el propósito de la misma. Que se escribe de la siguiente forma:

Ejemplo:

```
/**
 * GestinventarioModel
 * Modelo de Gestión de Inventario Físico
 *
 * @package Inventario
 * @subpackage Inventario
 * @author Alien Fernández Fuentes
 * @copyright ERP Cuba
 * @version 1.0
 */
```

Figura 7 Nomenclatura de comentarios en la clase.

En las funciones

Antes de la declaración de la función se escribe una breve descripción donde se explique el propósito de la misma. Que se escribe de la siguiente forma:

Ejemplo:

```
/**
 * @author Alien Fernández Fuentes.
 * @name generarhojainventario
 * Acción que permite generar una Hoja de Inventario.
 * @return void
 * @throws ZendExt_Exception - Excepción declarada en el xml de excepciones.
 * @version 1.0
 */
```

Figura 8 Nomenclatura de comentarios al inicio de una función.

Función complicada

En caso de ser una función complicada se comentaría dentro de la misma para lograr una mejor comprensión del código.

Ejemplo:

```
function confirmardocinventario($iddocumento,$idusuario,$version){
  $flag = 0;
  //Obtengo la cantidad de productos de la hoja de inventario
  $existencia = $this->pIntegrator->movimientos->Cantidad_PI($iddocumento);
  if ($existencia > 0){
   //Obtengo los datos de los productos de la hoja de inventario
    $arrDat = $this->pIntegrator->movimientos->ObtenerDatos_PI($iddocumento);
      foreach ($arrDat as $index => $value){
       //verifico que el conteo no sea vacio ni -1
        if ($value->conteof == '' || $value->conteof == -1){
          //Activo la bandera como que hay productos que no tienen conteo
          $flag = 1;
          break;
        }
      }
    }else
       $flag = -1;
   //Verifico que todos los productos tengan conteo registrado
    if ($flag == 0){
      //Registro el usuario que confirma la hoja de inventario
      $this->pIntegrator->documentos->Setalias($idusuario);
      //Confirmo la hoja de inventario
      $this->pIntegrator->documentos->ConfirmarDocInventario($iddocumento,$version);
    }
    return $flag;
}
```

Figura 9 Nomenclatura de comentarios en una función complicada.

2.6 Tratamiento de errores

Para garantizar el correcto funcionamiento del sistema se debe tener en cuenta un tratamiento de errores, el sistema captura todas aquellas excepciones que son lanzadas y se le facilita un tratamiento para que no colapse.

De esta manera se da solución a las problemáticas de los lanzamientos de excepciones que se dan en el sistema según las peticiones del usuario, capturando los tipos de errores lanzados y mostrando de manera visible al usuario mensajes de confirmación, esto le dará una idea de qué es lo que está incorrecto y como solucionarlo.

Mediante la interfaz Web se impedirá que el usuario asuma un papel activo en la introducción de la información, para esto se contará con cuadros de opción, menú de selección, lo cual facilitará la entrada de datos. La información que requiera ser adicionada por el usuario se validará mediante funciones o expresiones regulares que garanticen que sea válida y que el cuadro de texto no esté vacío si es obligatorio llenarlo. Si hay un error en la información le saldrá al usuario un mensaje en pantalla indicándole el error, al oprimir **Aceptar** el mensaje desaparece y el usuario podrá seguir introduciendo los datos en el formulario. También se validan las opciones correspondientes a la extracción o modificación de datos del servidor de base datos. Si se desea eliminar algún elemento de la BD se pregunta al usuario si está seguro de realizar dicha acción, al igual que cuando desee modificar alguna información, antes de actualizarla se pregunta si desea realizarla o no.

En el sistema existen funciones que necesitan validaciones previas para que se ejecuten sin ningún problema, para estas validaciones se usan varios ficheros XML entre ellos el *validator*, el cual contiene las llamadas a las precondiciones y poscondiciones que deben cumplirse, igualmente esta el fichero *exception* el cual contiene los mensajes que se muestran según la excepción lanzada, otro de los ficheros XML es el *ManageException* el cual captura excepciones previas que se definieron ya sean para excepciones del Doctrine o del Framework. Así se logra realizar las operaciones deseadas y que se rectifique al cometer un error.

2.7 Descripción de las principales clases a utilizar

2.7.1 Clases Controladoras

Las clases de control son objeto que se encargan de dirigir el flujo del control de la aplicación, representan coordinación, secuencia, transacciones y control de los objetos, se usan para encapsular el control de un caso de uso en concreto, coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso, por lo que definen el flujo de control y las transacciones dentro de un caso de uso delegando el trabajo a otros objetos.

En la siguiente tabla se muestra la clase controladora GestinventarioController que tiene como función capturar los eventos relacionado con la gestión de las hojas de inventario y la comunicación con la clase del negocio GestinventarioModel.

Nombre: GestinventarioController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
init()	Constructor de la clase.
gestinventarioAction()	Validar todos los datos necesarios para gestionar documentos de inventario.
cargarproductosinventarioAction ()	Cargar los productos que tiene una hoja de inventario.
cargarhojainventarioAction ()	Carga las hojas de inventario.

cargararbolproductosAction ()	Carga el árbol de productos ordenados por genérico.
cargarproductosgenericosAction()	Carga los productos de un nivel del árbol de genéricos.
generarhojainventarioAction()	Crear una hoja de inventario.
modificarhojainventarioAction()	Modificar una hoja de inventario.
registrarconteofAction()	Registrar el conteo final a los productos.
eliminarhojainventarioAction()	Eliminar una hoja de inventario.
eliminarprodinventarioAction()	Eliminar un producto de una hoja de inventario.
cancelarestadoAction()	Cancelar el estado a una hoja de inventario.
confirmarinventarioAction()	Confirmar una hoja de inventario.
aprobardocumentoAction()	Aprobar una hoja de inventario.
noaprobardocumentoAction()	Rechazar una hoja de inventario.
adicionarprodinventarioAction()	Adicionar productos a una hoja de inventario.
cargargriddocsAction()	Carga las hojas de inventario listas para la vista de impresión.

Tabla 3 Clase controladora “GestinventarioController”.

En la siguiente tabla se muestra la clase controladora GestplanesconteoController que tiene como función capturar los eventos relacionado con la gestión de las planes de conteo y la comunicación con la clase del negocio GestplanesconteoModel.

Nombre: GestplanesconteoController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
init()	Constructor de la clase.
gestplanesconteoAction()	Validar todos los datos necesarios para gestionar planes de conteo.
cargarplanesconteoAction()	Cargar planes de conteo
cargartodosmesesmodAction()	Carga los meses para modificar.
modificartodosmesesAction()	Los porcientos de los grupos seleccionados de un mes los modifica para todos los demás.
cargargruposproductoAction()	Carga los grupos de productos del nomenclador.
crearplanconteoAction()	Crea un plan de conteo.
registrarporcientoAction()	Registrar porciento a un grupo.
cancelarestadoplanconteoAction()	Cancelar estado de un plan de conteo.

aprobarplanconteoAction()	Aprobar un plan de conteo.
noaprobarplanconteoAction()	Rechazar un plan de conteo.
confirmarplanconteoAction()	Confirmar un plan de conteo.
confirmarmesplanconteoAction()	Confirmar un mes de un plan de conteo.
cancelarmesplanconteoAction()	Cancelar el estado de un mes de un plan de conteo.
aprobarmesplanconteoAction()	Aprobar un mes de un plan de conteo.
resetmesplanconteoAction()	Eliminar todos los grupos de un mes.
eliminarplanconteoAction()	Eliminar un plan de conteo.
cargarplanAction()	Carga un plan de conteo para la vista de impresión.

Tabla 4 Clase controladora “GestplanesconteoController”.

2.7.2 Clases modelos de la aplicación

Estas clases guardan poca o ninguna información del estado por si misma, pero asisten en la ejecución de tareas complejas.

En la siguiente tabla se muestra la clase modelo de la aplicación GestinventarioModel que tiene como función la lógica propia del negocio gestión de hojas de inventario, además de cálculos y procesamiento del negocio en cuestión.

Nombre: GestinventarioModel	
Tipo de clase: Modelo de la aplicación	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
GestinventarioModel ()	Constructor de la clase.
cargararbolproductos($idestructura,$nivel,$idprod)	Carga el árbol de productos ordenados por genérico.
cargarproductosgenericos($idprod,$nivel)	Carga los productos de un nivel del árbol de genéricos.
cargarhojainventario($datos)	Cargar las hojas de inventario que posee un depósito.
cargarproductoinventario($iddoc, $codigo, $den, $nropieza,$limit,$start,$idestructura)	Cargar los productos que tiene una hoja de inventario.
crearArrProdAleatorios($arrProdPorContar,$arrPorciento)	Crear un arreglo de productos aleatorios.
cargarproductoinventario($iddocumento,$codigo,$denominacion,$nropieza,$limit,$start,$idestructuracomun)	Cargar los productos de una hoja de inventario.

generarhojainventario($idest,$usuario,$tipo,$mes,$anno, $observaciones)	Crear una hoja de inventario.
modificarhojainventario($iddocumento,$idtipo,$idtipoant, $observaciones,$version)	Modificar una hoja de inventario.
registrarconteof($iddocumento,$idproducto,$conteo)	Registrar el conteo final a los productos.
registrarsindiferencias($iddocumento)	Actualiza los valores del conteo final con las existencias del producto.
eliminarhojainventario($iddoc,$version,$usuario)	Eliminar una hoja de inventario.
eliminarprodinventario($idprodinventario,$idmovimiento)	Elimina un producto de una hoja de inventario.
cancelarestadodoc($iddocumento,$version)	Cancelar el estado de una hoja de inventario.
confirmardocinventario($iddocumento,$usuario,$version)	Confirmar una hoja de inventario.
aprobardocumentoinventario($iddocumento,$idusuario, $idestructuracomun,$version)	Aprobar una hoja de inventario.
noaprobardocumento($iddoc,$usuario,$version)	Rechazar una hoja de inventario.
adicionarprodinventario($arrProd,$iddocumento,$tipoinv)	Adicionar productos a una hoja de inventario.
cargargriddocs($idestructuracomun,$iddocumento)	Carga las hojas de inventario listas para la vista de impresión.

Tabla 5 Clase modelo de la aplicación “GestplanesconteoModel”.

En la siguiente tabla se muestra la clase modelo de la aplicación GestinventarioModel que tiene como función la lógica propia del negocio gestión de hojas de inventario, además de cálculos y procesamiento del negocio en cuestión.

Nombre: GestplanesconteoModel	
Tipo de clase: Modelo de la aplicación	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
GestplanesconteoModel ()	Constructor de la clase.
cargarplanesconteo($datos)	Cargar planes de conteo.
cargartodosmesesmod($idplan,$estado)	Carga los meses para modificar.
cargargruposproducto($idmes,$idplan,$descripcion,$observaciones,$idestructuraop)	Cargar los grupos de productos del nomenclador.
crearplanconteo($idestructura,$idusuario)	Crear un plan de conteo.
registrarporciento($mes,$idgrupo,$pc,$idplan)	Registrar porcientos a un mes de un plan de conteo.
modificartodosmeses($idplan,$datGrupos)	Modificar todos los porcientos de los grupos de un mes de un plan de conteo.
cancelarestadoplanconteo($idplanconteo)	Cancelar estado de un plan de conteo.

confirmarplanconteo($idplan,$idusuario)	Confirmar un plan de conteo.
confirmarmesplanconteo($idplan,$idmes)	Confirmar un mes de un plan de conteo.
cancelarmesplanconteo($idplan,$idmes)	Cancelar el estado de un plan de conteo.
aprobarmesplanconteo($idplan,$idmes)	Aprobar un mes del plan de conteo.
aprobarplanconteo($idplanconteo,$idusuario)	Aprobar un plan de conteo.
resetmesplanconteo($idplan,$idmes)	Eliminar los porcientos registrados de un mes.
noaprobarplanconteo($idplanconteo)	Rechazar un plan de conteo.
eliminarplanconteo($idplanconteo)	Elimina un plan de conteo.
cargarplan($idplan)	Carga un plan de conteo para la vista de impresión.

Tabla 6 Clase modelo de la aplicación “GestplanesconteoModel”.

2.7.3 Clases modelo del dominio

Estas clases modelan información que poseen una larga vida y que a menudo son persistentes. La fuente principal de obtención son las clases entidades del negocio y el glosario de términos que se ha ido elaborando. Se encargan de modelar la información del sistema y el comportamiento asociado a una información.

En la siguiente tabla se muestra la clase modelo del dominio DatPlanconteo que tiene como función el acceso a la base de datos para la gestión de los planes de conteo anuales.

Nombre: DatPlanconteo	
Tipo de clase: Modelo del dominio	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
setUp()	Constructor de la clase.
ObtenerPlanDadoIdplan($idplan)	Obtener un plan de conteo.
ObtenerIdPlanDadoAnno($anno,$idestructura)	Obtener el id de un plan de conteo.
verificarExistePlanAnual($anno,$idestru,$mes)	Verificar que exista un plan de conteo.
eliminarPlan($idplan)	Eliminar plan de conteo.

Tabla 7 Clase modelo del dominio “DatPlanconteo”.

En la siguiente tabla se muestra la clase modelo del dominio DatPlanconteo que tiene como función el acceso a la base de datos para la gestión de los planes de conteo mensuales.

Nombre: DatPlanconteom	
Tipo de clase: Modelo del dominio	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	Descripción:
setUp()	Constructor de la clase.
ObtenerPlanMensualDadoIddocumento($id)	Obtener plan mensual.
ActualizaPorciento($idmes,$grupo,$plan,$pc)	Actualizar los porcientos.
ActualizaDocrefDadoMes($mes,$iddocument,$plan)	Actualizar la referencia del documento.
ObtenerPlanesConteoDadoMes($mes,$idplan)	Obtener plan de conteo dado mes.
CantidadMesesPorEstado($idplan,$estado)	Cantidad de meses por estado.
ObtenerMesesSinDocInventario($idplan,$mes)	Obtener meses sin Documento de inventario asociado.
ProductosporContar($anno,$idestructura)	Productos que no han sido contados por un plan de conteo.
BuscarPorcientos($mes,$idgrupo)	Buscar porcientos por el que va a ser contado un grupo.

Tabla 8 Clase modelo del dominio “DatPlanconteom”.

En la siguiente tabla se muestra la clase modelo del dominio DatInventario que tiene como función el acceso a la base de datos para la gestión de las hojas de inventario.

Nombre: DatInventario
Tipo de clase: Modelo del dominio

Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
setUp()	Constructor de la clase.
ObtenerTupla($iddocumento)	Obtener la fila del inventario.
verificarExisteInventarioGeneral($idest,$anno)	Verificar que exista un inventario general.
aprobadoPor()	Buscar todos los usuario que hallan aprobado un inventario.

Tabla 9 Clase modelo del dominio “DatInventario”.

2.8 Diagrama de despliegue

Este tipo de diagrama se utiliza para modelar el hardware utilizado en la implementación del sistema y las relaciones entre sus componentes. Los elementos usados por este tipo de diagrama son nodos (representados como un prisma), componentes (representados como una caja rectangular con dos protuberancias del lado izquierdo) y asociaciones.

Además, los diagramas de despliegue muestran la configuración en funcionamiento del sistema incluyendo su software y su hardware. Para cada componente de un diagrama es necesario que se deba documentar las características técnicas requeridas, el tráfico de red y el tiempo de respuesta. (elcodigok.com, 2006).

En la siguiente figura se muestra el diagrama de despliegue propuesto por el equipo de arquitectura.

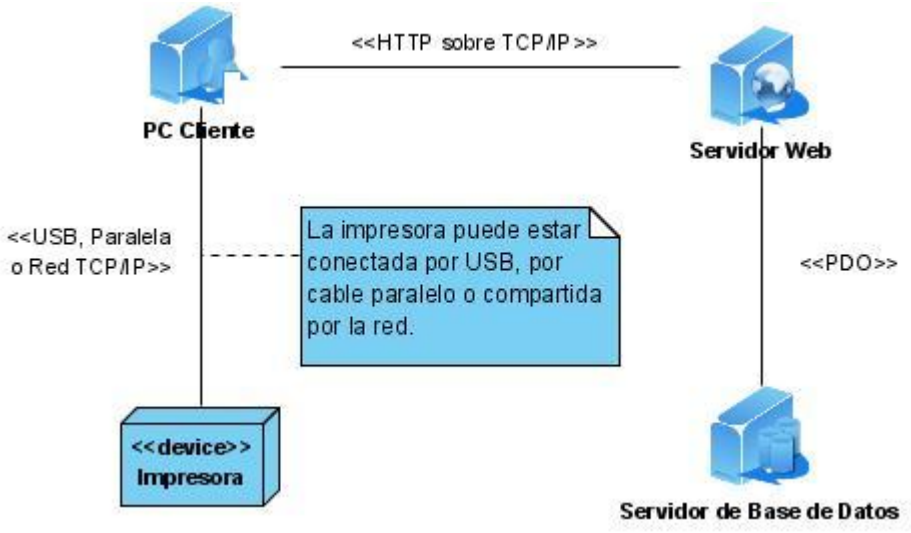

Figura 10 Diagrama de despliegue propuesto por el equipo de arquitectura.

2.9 Conclusiones Parciales

En este capítulo se especificó la propuesta del módulo, la descripción de las principales clases para facilitar la comprensión de la implementación realizada. Se explicaron los componentes que fueron reutilizados para el desarrollo de la aplicación.

Se comentan los diagramas de despliegue e integración de componentes. En el diagrama de despliegue se determinaron las relaciones existentes entre el sistema y el hardware donde se implantará la aplicación. Con la realización del diagrama de integración de componentes se conocieron las relaciones de dependencias entre los componentes y las interfaces que estos soportan. Además se conoció todo lo referente a la integración de los componentes mediante el IoC y la arquitectura en capas utilizada.

Por último se analizaron los estilos de código debido a que hacen más legible el programa fuente, con el objetivo de ayudar la comunicación entre desarrolladores.

Capítulo 3 Validación de la Solución Propuesta

3.1 Introducción

El desarrollo de un software es algo complejo y son innumerables las posibilidades de errores, por lo que este ha de ir acompañado de alguna actividad que garantice la calidad; las pruebas son un elemento crítico para la garantía de calidad del software. Es por eso que se utilizan técnicas como las pruebas unitarias que no son más que una forma de verificar el correcto funcionamiento del código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa ya que requiere un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80% siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.

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

En este capítulo se realiza la validación a la solución propuesta en el capítulo anterior, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas para dar respuesta a los distintos requisitos planteados por el cliente. Además se utilizan métricas para la validación del modelo de diseño como son:

- Tamaño operacional de clase (TOC).
- Relaciones entre clases (RC).

3.2 Pruebas de Software

3.2.1 Objetivos

El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Esto implica:

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

3.2.2 Pruebas de unidad

Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las pruebas de integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

Fomentan el cambio: facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Simplifica la integración: permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente.

Documenta el código: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.

Separación de la interfaz y la implementación: Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro.

Los errores están más acotados y son más fáciles de localizar: dado que tenemos pruebas unitarias que pueden desenmascararlos.

3.3 Pruebas de caja blanca

Las pruebas de caja blanca se realizan sobre las funciones internas de un módulo en concreto, están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos, pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos, comprobación de bucles.

En la siguiente tabla se representa las pruebas de Caja Blanca.

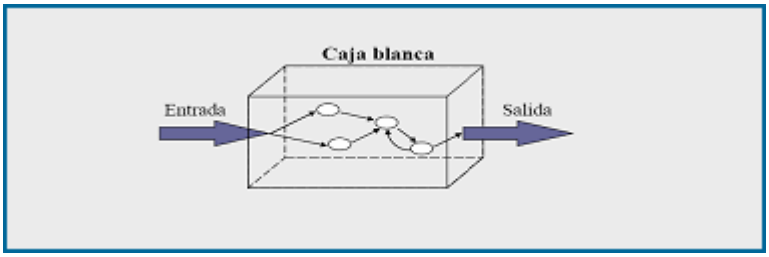

Figura 11 Representación de pruebas de Caja Blanca.

Tipos de prueba de caja blanca:

Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de Flujo de Datos: Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo

de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

La técnica del camino básico permite obtener una medida de la complejidad lógica del código de cada método, programa o módulo dado. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes que existen en la codificación por los cuales puede circular el flujo de control. Es además una de las más eficientes en cuanto a cobertura de código, pues logra que se ejecuten todos los bucles en sus límites operacionales (Márquez Alpízar, 2008).

Los pasos que se siguen para aplicar esta técnica son:

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

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

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un grafo de flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.
- Todo segmento de código de cualquier programa se puede traducir a un grafo de flujo.
- Se calcula la complejidad ciclomática del grafo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones.

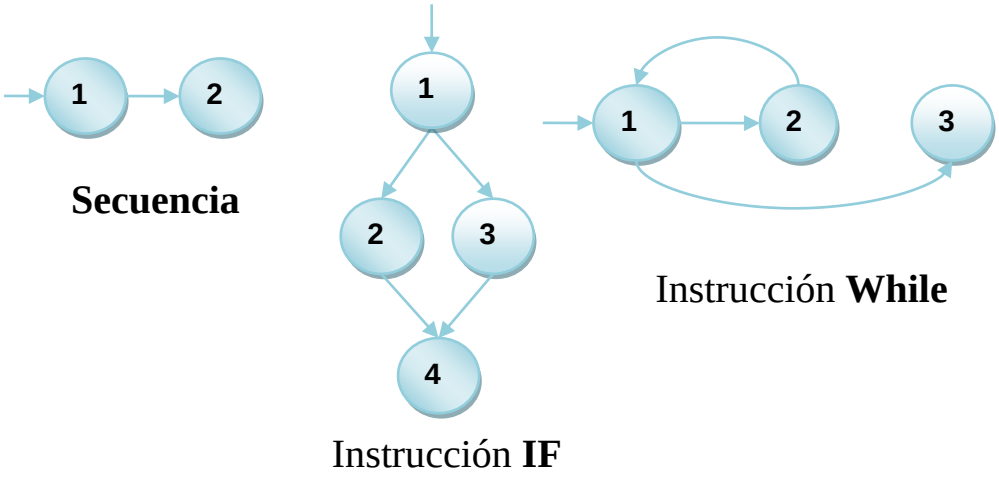

Figura 12 Notación de grafos de flujo para las instrucciones: Secuenciales, If y While.

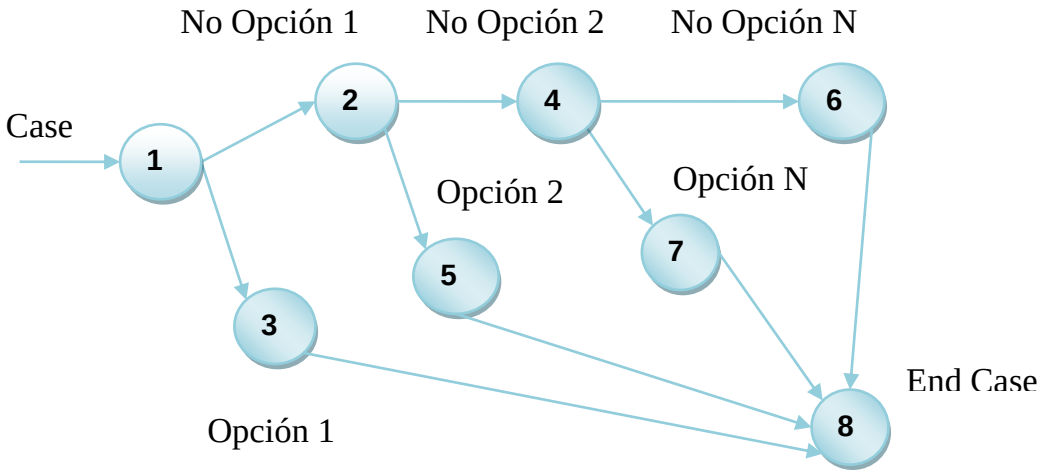

Figura 13 Notación de grafos de flujo para la instrucción Case.

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, su comprensión y brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo:

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

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

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

Para realizar la prueba de caja blanca específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método adicionarprodinventario ($arrProd,$iddocumento,$tipoinv) el cual se encarga de adicionar productos a una hoja de inventario.

```
function adicionarprodinventario($arrProductos,$iddocumento,$tipoinv){
  $flag = 0; (1)
  $integrator = ZendExt_IoC_Inter::getInstance(); (1)
  if (($tipoinv == 3 || $tipoinv == 2) && is_array($arrProductos)){ (2)
    foreach($arrProductos as $index => $value){ (3)
      $cantidad = $value->cantidaddisponible ;  (4)
      $existencia = $value->existencia ;        (4)
      $idproducto = $value->idproducto;         (4)
      $arrProducto = $integrator->productos->obtenerprodID($idproducto); (4)
      $pprom = $arrProducto->preciopromedio;  (4)
      $integrator->movimientos->Insertar_PI($iddocumento,$existencia,$idproducto,0,$pprom); (4)
    } (5)
    $flag = true; (5)
   }else
     $flag = -1;  (6)
  return $flag;   (7)
}
```

Figura 14 Representación del algoritmo adicionarprodinventario($arrProd,$iddocumento,$tipoinv).

En la siguiente figura se muestra el grafo de flujo asociado al código anterior.

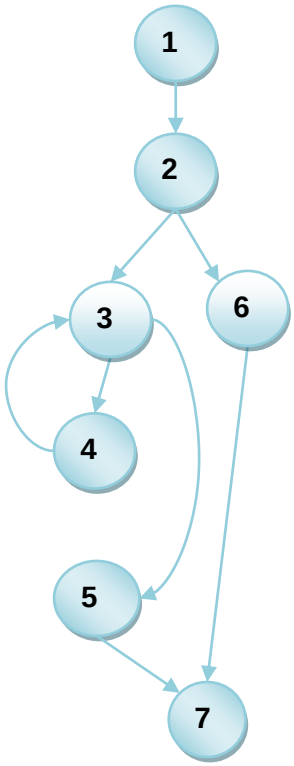

Figura 15 Grafo de flujo asociado al algoritmo adicionarprodinventario($arrProd,$iddocumento,$tipoinv)

Cálculo de la complejidad ciclomática a partir de un segmento de código.

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo, cantidad total de nodos para la siguiente fórmula:

V (G) = (A – N) + 2

V (G) = (8 – 7) + 2

V (G) = 3

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

Se puede usar también:

V (G) = P + 1

V (G) = 2 + 1

V (G) = 3

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

V (G) = R

V (G) = 3

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

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 3, lo que significa que existen tres posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

En la siguiente tabla se muestran los caminos básicos.

Número	Camino básico
1	1 – 2 – 3 – 5 – 7
2	1 – 2 – 3 – 4 – 3 – 5 – 7
4	1 – 2 – 6 – 7

Tabla 10 Caminos básicos del flujo.

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

Para realizarlos es necesario cumplir con las siguientes exigencias:

- Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- Entrada: Se muestran los parámetros que entran al procedimiento.
- Resultados Esperados: Se expone el resultado que se espera devuelva el procedimiento.

Caso de prueba para el camino básico 1:

Camino 1: [1 – 2 – 3 – 5 – 7]

Descripción:

- Los datos de entrada cumplirán con los siguientes requisitos:
  El parámetro arrProductos será un arreglo vacio y el tipoinv tendrá valor 2 o 3, el valor de iddocumento será constante para todas las pruebas del camino básico.

Condición de ejecución:

- El arrProductos será un arreglo vacio.

- El tipo de inventario será 2,
- El iddocumento tendrá valor 90000000008.

Entrada:

- $arrProductos = array ().
- $tipoinv = 2.
- $iddocumento = 90000000008.

Resultados esperados:

Se espera que no se inserte ya que el arreglo que debe contener los productos esta vacio.

Caso de prueba para el camino básico 2

Camino 2: [1 – 2 – 3 – 4 – 3 – 5 – 7]

Descripción:

- Los datos de entrada cumplirán con los siguientes requisitos:
  El parámetro arrProductos será un arreglo con un producto, el tipoinv tendrá valor 2 o 3 y el valor de iddocumento será constante para todas las pruebas del camino básico.

Condición de ejecución:

- El arrProductos será un arreglo con un producto.
- El tipo de inventario será 2.
- El iddocumento tendrá valor 90000000008.

Entrada:

- $arrProductos = array (1 producto).
- $tipoinv = 2.
- $iddocumento = 90000000008.

Resultados esperados:

Se espera que se lance un mensaje con el siguiente texto: “Los productos fueron adicionados al documento de inventario”.

Caso de prueba para el camino básico 3

Camino 3: [1 – 2 – 6 – 7]

Descripción:

- Los datos de entrada cumplirán con los siguientes requisitos:
  El parámetro arrProductos será un arreglo con un producto, el tipoinv tendrá valor 4 y el valor de iddocumento será constante para todas las pruebas del camino básico.

Condición de ejecución:

- El arrProductos será un arreglo con un producto.
- El tipo de inventario será 4.
- El iddocumento tendrá valor 90000000008.

Entrada:

- $arrProductos = array (1 producto).
- $tipoinv = 4.
- $iddocumento = 90000000008.

Resultados esperados:

Se espera que se lance un mensaje con el siguiente texto: “No se pudieron adicionar los productos al documento de inventario”.

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

3.4 Validación del modelo de diseño utilizando métricas

Un aspecto importante a tener en cuenta en la evaluación de la calidad del diseño ha sido la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto referenciadas por Pressman, teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software.

Atributos de calidad que se abarcan:

Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad del diseño. Consiste en la complejidad que posee una estructura de diseño de clases.

Complejidad de implementación. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización. Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.

Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, esta muy ligada a la característica de Reutilización.

Complejidad del mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.

Nivel de Cohesión. Consiste en el grado de especialización de las clases concebidas para modelar un dominio o concepto específico.

Abstracción del diseño. Consiste en la capacidad de modelar lo más cercano posible a la realidad un concepto o dominio determinado.

3.4.1 Tamaño operacional de clase (TOC)

Está dado por el número de métodos asignados una clase

Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 11 Tamaño operacional de clase (TOC).

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

Ver instrumentos y tabla de resultados en (Anexo 8 Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

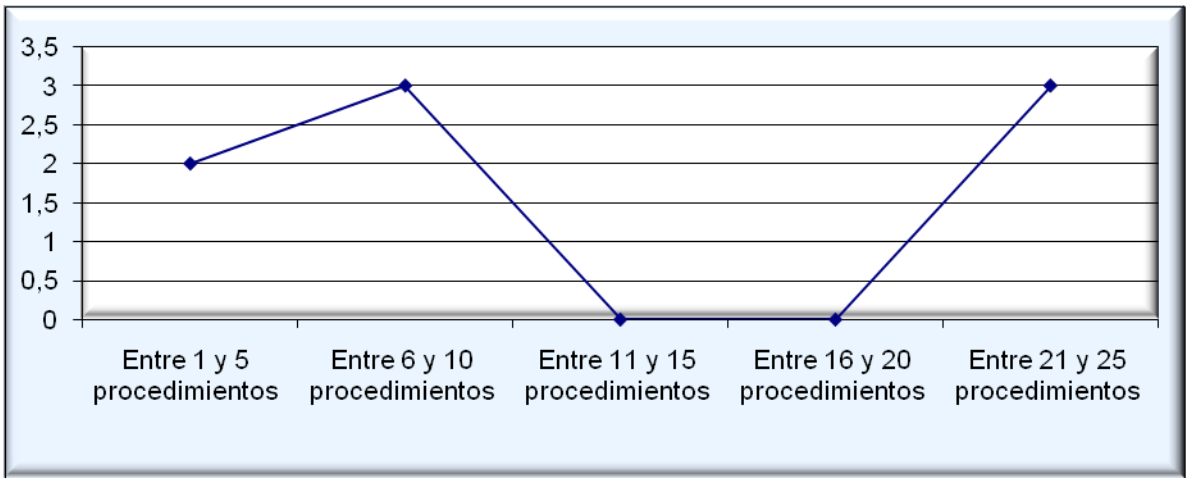

Figura 16 Resultados obtenidos en el instrumento agrupados en los intervalos definidos.

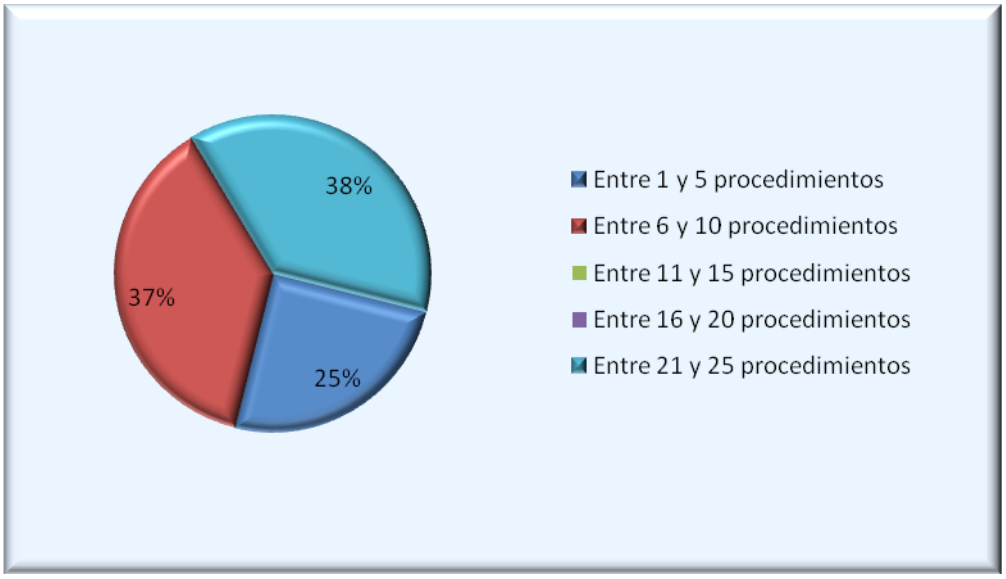

Figura 17 Por ciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

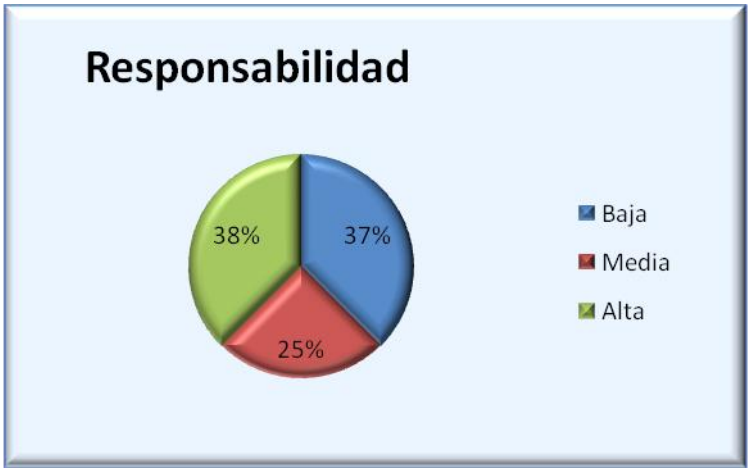

Figura 18 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

Figura 19 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

Figura 20 Incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica **TOC**, se puede concluir que el diseño del subsistema Inventario tiene una calidad aceptable teniendo en cuenta que el 62% de las clases incluidas en estos subsistemas posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Solo el 62% de las clases poseen evaluaciones positivas en los demás atributos de calidad.

3.4.2 Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otras.

Acoplamiento	Un aumento del RC implica un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 12 Relaciones entre clases (RC).

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

Ver instrumentos y tabla de resultados en (Anexo 9 Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

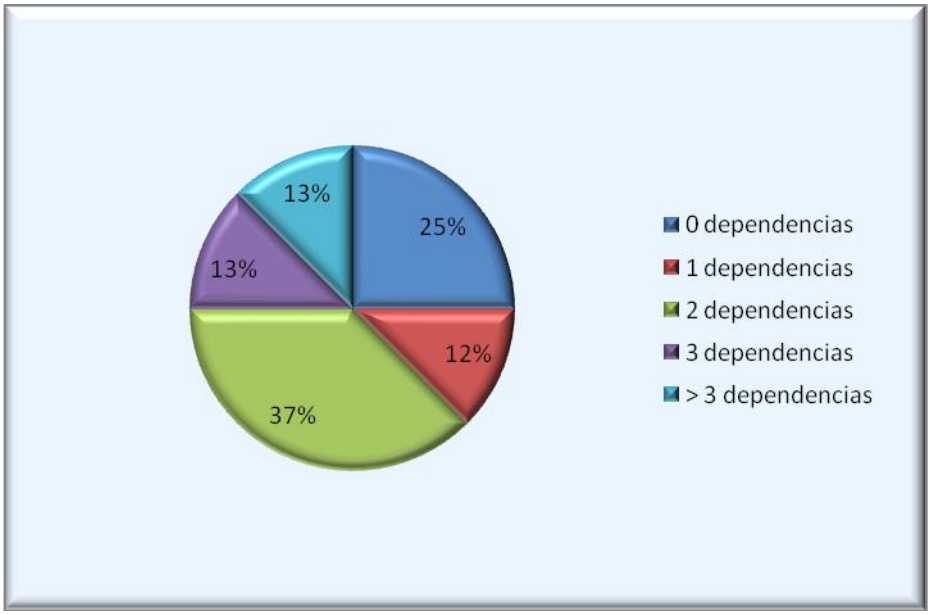

Figura 21 Por ciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

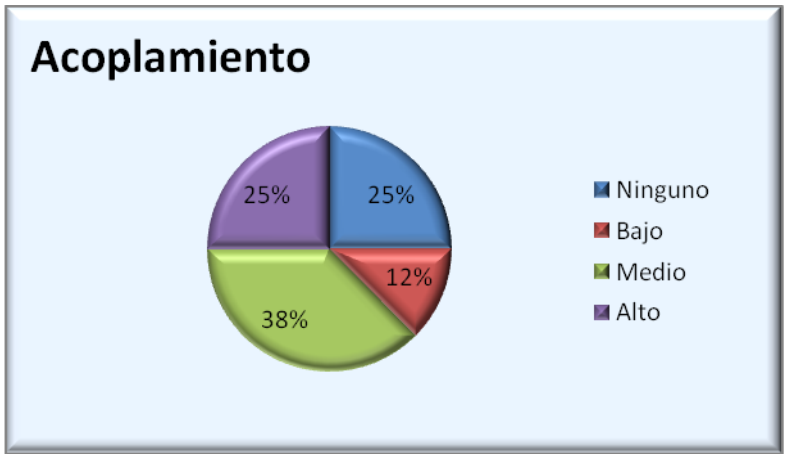

Figura 22 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

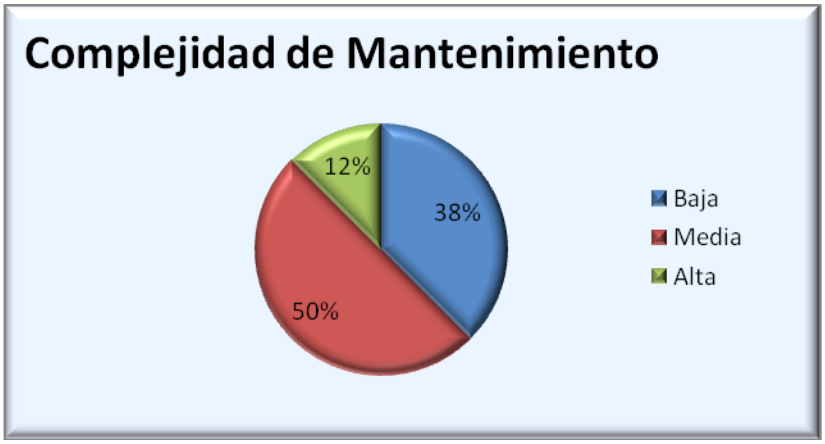

Figura 23 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

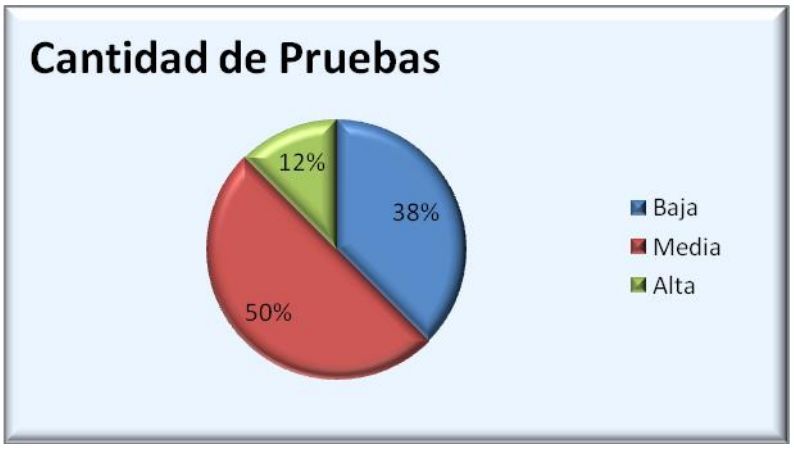

Figura 24 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

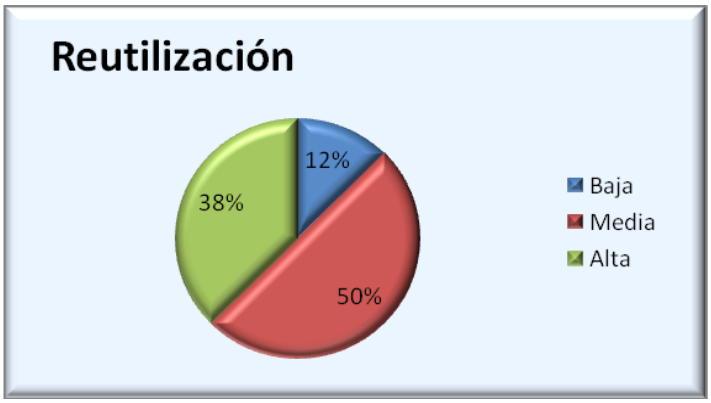

Figura 25 Incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del subsistema Inventario tiene una calidad aceptable teniendo en cuenta que el 74% de las clases incluidas en el subsistema poseen menos de 3 dependencias de otras clases. Además el 75% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 88% de las clases.

A manera de resumen se han tabulado los resultados obtenidos en la siguiente tabla.

Atributos	**TOC (en %)**	**RC (en %)**
Responsabilidad	62	-
Complejidad de implementación	62	-
Reutilización	62	88
Acoplamiento	-	75
Complejidad del mantenimiento	-	88
Cantidad de pruebas	-	88

Tabla 13 Resultados generales obtenidos por las métricas.

3.5 Conclusiones parciales

En el presente capítulo se realizó un análisis de las pruebas de unidad, en específico las pruebas de camino básico. Se diseñaron casos de pruebas específicos para los principales algoritmos, comprobándose que el flujo de trabajo de los mismos estuvo correcto ya que cumplieron con las condiciones necesarias que se habían planteado.

Finalmente se elaboraron instrumentos inspirados en métricas para calidad del diseño como tamaño operacional de clase (TOC) y relaciones entre clases (RC) donde mediante su empleo permitió afirmar que el diseño realizado se puede valorar de aceptable debido a que el valor de los atributos de calidad, responsabilidad, complejidad de implementación, reutilización, acoplamiento, complejidad del mantenimiento y cantidad de pruebas, estuvo por encima del 62% en todos los casos.

Conclusiones Generales

A manera de conclusión se plantea que:

- Se analizaron soluciones existentes descubriendo deficiencias en los mismos.
- Se mostraron los aspectos más significativos de la solución propuesta como el análisis de reutilización de componentes, la descripción de las principales clases a utilizar, la descripción de la implementación y los estándares de codificación utilizados evidenciado la obtención de un producto funcional a partir de los requisitos propuestos por los analistas.
- Se realizó la validación de la solución propuesta mediante el diseño y aplicación de las pruebas de caja blanca para validar la calidad de la solución propuesta arrojando resultados positivos. También se hizo una evaluación de la implementación mediantes métricas donde arrojaron valores aceptables en los atributos de reutilización, facilidad de mantenimiento, complejidad del diseño, complejidad de implementación, cohesión, acoplamiento y cantidad de pruebas.

Por todo lo antes mencionado se evidencia el cumplimiento de los objetivos propuestos en el presente trabajo de diploma, lo cual conlleva a un cumplimiento del objetivo general.

Recomendaciones

Las recomendaciones propuestas para la continuidad del presente trabajo son:

- Arreglar las no conformidades detectadas en las pruebas pilotos con el objetivo de refinar la solución.
- Ampliar las funcionalidades del módulo con los nuevos requerimientos que surjan por necesidades del cliente.
- Consultar este documento como material de estudio, guía y apoyo para el posterior desarrollo del Módulo de Inventario Físico.

Referencias Bibliográficas

Adpime. 2008. ERP - Sistemas de Gestión PYME. [En línea] 2008. [Citado el: 12 de Abril de 2009.] http://www.adpime.com/ERP/Es_ERP_intro.htm.

Cavsi.com. 2007. *Que es un sistema gestor de bases de datos.* [En línea] 2007. [Citado el: 11 de Abril de 2009.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/.

Centro de Soluciones de Gestión. 2008. *Modelo de desarrollo orientado a componentes.* [Documento] La Habana : Universidad de las Ciencias Informáticas, 2008.

Ciberaula.com. 2008. *Introducción a Apache.* [En línea] 2008. [Citado el: 16 de Mayo de 2009.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

Cova, Rubén Darío. 2004. *Html.* [En línea] 2004. [Citado el: 27 de Mayo de 2009.] http://www.geocities.com/covag/defhtml.html.

Cssblog.es. 2008. *Frameworks.* [En línea] 2008. [Citado el: 13 de Abril de 2009.] http://www.cssblog.es/guias/Framework.pdf.

del Toro Ríos, José Carlos y González Brito, Henry Raúl. 2009. *Documento Visión. Proyecto ERP-Cuba.* [Documento] La Habana : Universidad de las Ciencias Informáticas, 24 de 04 de 2009.

Desarrolloweb.com. 2009. *Qué es PHP.* [En línea] 1 de Mayo de 2009. [Citado el: 13 de Abril de 2009.] http://www.desarrolloweb.com/articulos/392.php.

Desarrolloweb.com. 2008. *CSS.* [En línea] Septiembre de 2008. [Citado el: 13 de Abril de 2009.] http://www.desarrolloweb.com/articulos/26.php.

Desarrolloweb.com. 2007. *que-es-html.* [En línea] 2007. [Citado el: 10 de Abril de 2009.] http://www.desarrolloweb.com/articulos/que-es-html.html.

Doctrine-Project.org. 2008. [En línea] 2008. [Citado el: 15 de Abril de 2009.] http://www.doctrine-project.org.

Echarte, Patxi. 2007. *Frameworks de Zend para el desarrollo de aplicaciones PHP.* [En línea] 3 de Julio de 2007. [Citado el: 12 de Mayo de 2009.] http://www.eslomas.com/index.php/archives/2007/07/03/framework-de-zend-para-el-desarrollo-de-aplicaciones-php/.

elcodigok.com. 2006. *Diagrama de despliegue.* [En línea] 2006. [Citado el: 23 de Mayo de 2009.] http://www.elcodigok.com.ar/category/uml/page/2/.

Firefox, Mozilla. 2009. [En línea] 2009. [Citado el: 15 de Abril de 2009.] http://www.getfirefox.es/firefox-features.

freedownloadmanager.org. 2008. *Visual paradigm.* [En línea] 2008. [Citado el: 21 de Abril de 2009.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p .

Gonzalez Brito, Henry Raúl y Lezcano Lozada, Yuniesky. 2005. *Sistema de Inventario Participativo de la UCI.* [Documento] Ciudad de la Habana, Cuba : UCI, 2005.

González Rodríguez, Victoria. 2003. *El impacto de un ERP en la empresa.* 2003.

González, H. 2006. *ERP cubano, un paso estratégico para la consolidación del Software Libre en Cuba.* [Documento] 2006. Vol. I.

Gracia, Joaquin. 2005. *Patrones de diseño, diseño de software orientado a objetos.* 2005.

Guia-ubuntu.org. 2007. *PgAdmin_III.* [En línea] 2007. [Citado el: 13 de Abril de 2009.] http://guia-ubuntu.org/index.php?title=PgAdmin_III.

Imapax.com. 2009. *Que es subversion svn.* [En línea] 2009. [Citado el: 13 de Abril de 2009.] http://www.imapax.com/soluciones/joomla/que-es-subversion-svn.html.

Json.org. 2007. *Sitio oficial de JSON.* [En línea] 2007. [Citado el: 12 de Abril de 2009.] http://www.json.org/json-es.html.

Larman, Graig. 2004. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Pentrice Hall, 2004.

Maestrosdelweb.com. 2007. *Que es javascript.* [En línea] 3 de Julio de 2007. [Citado el: 12 de Abril de 2009.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

maestrosdelweb.com. 2003. IntroCSS. [En línea] 8 de Noviembre de 2003. [Citado el: 13 de Abril de 2009.] http://www.maestrosdelweb.com/editorial/introcss/.

Márquez Alpízar, Yaimí y Valdés Echavarría, Yenni. 2008. *Procedimiento general de pruebas de Caja Blanca aplicando la técnica del Camino Básico.* [Documento] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

Minbas.cu. 2007. *Inventario SISCONT5.* [En línea] 2007. [Citado el: 12 de Abril de 2009.]

http://siscont.tm.minbas.cu/Docs/Documentos/Manuales/Inventario%20SISCONT5.pdf .

Mitecnologico.com. 2007. *HerramientasCase.* [En línea] 2007. [Citado el: 12 de Abril de 2009.]

http://www.mitecnologico.com/Main/HerramientasCase.

Netpecos.org. 2008. *PostgreSql.* [En línea] 2008. [Citado el: 15 de Abril de 2009.]

http://www.netpecos.org/docs/mysql_postgres/x15.html.

Openbravo.com. 2009. *Características OpenBravo.* [En línea] 2009. [Citado el: 15 de Abril de 2009.]

http://www.openbravo.com/es/product/erp/features/.

Osmosislatina.com. 2009. *Subversion.* [En línea] 3 de Enero de 2009. [Citado el: 13 de Abril de 2009.]

http://www.osmosislatina.com/subversion/basico.htm.

Pressman, Roger. 2001. *Ingeneria de Software, un enfoque practico.* 2001.

Rivas, Lornel A, y otros. 2007. Herramienta de Desarrollo de Software. *Hacia la Construcción de una Ontología.* [En línea] 2007. [Citado el: 18 de Abril de 2009.]

http://www.lisi.usb.ve/publicaciones/05%20herramientas/herramientas_25.pdf.

RodasXXI.cu. 2007. Inventario. [En línea] 2007. [Citado el: 12 de Abril de 2009.]

http://www.rodasxxi.cu/inventario.php.

SAP España. 2007. *SAP: número uno en software ERP.* [En línea] 2007. [Citado el: 17 de Mayo de 2009.]

http://www.sap.com/spain/solutions/business-suite/erp/index.epx.

Sics.cu. 2007. *CONDOR.* [En línea] 2007. [Citado el: 12 de Abril de 2009.]

http://www.sics.cu/productos.aspx.

Sqlmanager.net. 2007. *Sqlmanager para PostgreSQL.* [En línea] 2007. [Citado el: 19 de Abril de 2009.]

http://sqlmanager.net/products/postgresql/manager.

Techtear.com. 2008. *Zend Studio for Eclipse-desarrollo profesional en php.* [En línea] 22 de Enero de 2008. [Citado el: 13 de Abril de 2009.] http://www.techtear.com/2008/01/22/zend-studio-for-eclipse-desarrollo-profesional-en-php/.

Torres Saquipova, Dina Yaksilik y Hernández., Carlos de la Rosa. 2008. *Análisis y Diseño de los módulos Inventario y Administración del proyecto ERP Cubano.* [Documento] Ciudad de La Habana : UCI, 2008.

Wailgum, Thomas. 2008. ABC: An introduction to ERP. [En línea] 2008. [Citado el: 10 de Abril de 2009.]

http://www.cio.com/research/erp/edit/erpbasics.html.

Webexperto.com. 2006. *AJAX.* [En línea] 7 de Julio de 2006. [Citado el: 12 de Abril de 2009.] http://www.webexperto.com/articulos/articulo.php?cod=223.

Wikidot.com. 2008. *Gestión de Inventario.* [En línea] 2008. [Citado el: 12 de Mayo de 2009.]

http://mask.wikidot.com/gestion-del-inventario.

Wordpress.com. 2007. *Libreria ExtJs.* [En línea] 06 de 08 de 2007. [Citado el: 12 de Abril de 2009.]

http://vargasti.wordpress.com/2007/08/06/libreria-extjs/.

Yzquierdo Herrera, Raykenler y Lazo Ochoa, René. 2007. *El modelo de diseño del sistema HyperWeb. Módulos de Tratamiento Farmacológico y Configuración.* [Documento] Ciudad de La Habana : Universidad de las Ciencias Informáticas, 2007.

Zend.com. 2007. *Zend Studio for Eclipse.* [En línea] 2007. [Citado el: 21 de mayo de 2009.]

http://www.zend.com/products/studio/.

Glosario de Términos

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

Componente: El componente es la unidad de construcción elemental del diseño físico. Las características de un componente son:

- Se define según como interactúa con otros.
- Encapsula sus funciones y sus datos.
- Es reutilizable a través de las aplicaciones.
- Puede verse como una caja negra.
- Puede contener otros componentes.

Framework: Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

Implementación: Proceso por el cual se escribe (en un lenguaje de programación), se prueba, se depura y se mantiene el código fuente de un programa informático.

Logística: Conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa o de un servicio.

Métricas: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Módulo: A efectos prácticos, hablar de *programas* es lo mismo que hablar de *productos de software* o hablar de *módulos de software*. Cada *módulo* es una parte del sistema, que se instala y funciona por separado, entrelazándose con otros módulos con los que intercambia información.

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

Plataforma: Entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones.

Software: Se refiere a los programas y datos almacenados en un ordenador.

- Los programas dan instrucciones para realizar tareas al hardware o sirven de conexión con otro software.

Los datos solamente existen para su uso eventual por un programa.

Anexos

Anexo 1 Interfaz principal del Sistema Integral de Gestión de Entidades.

Se muestra en el árbol desplegado el acceso al módulo de gestión de inventario físico.

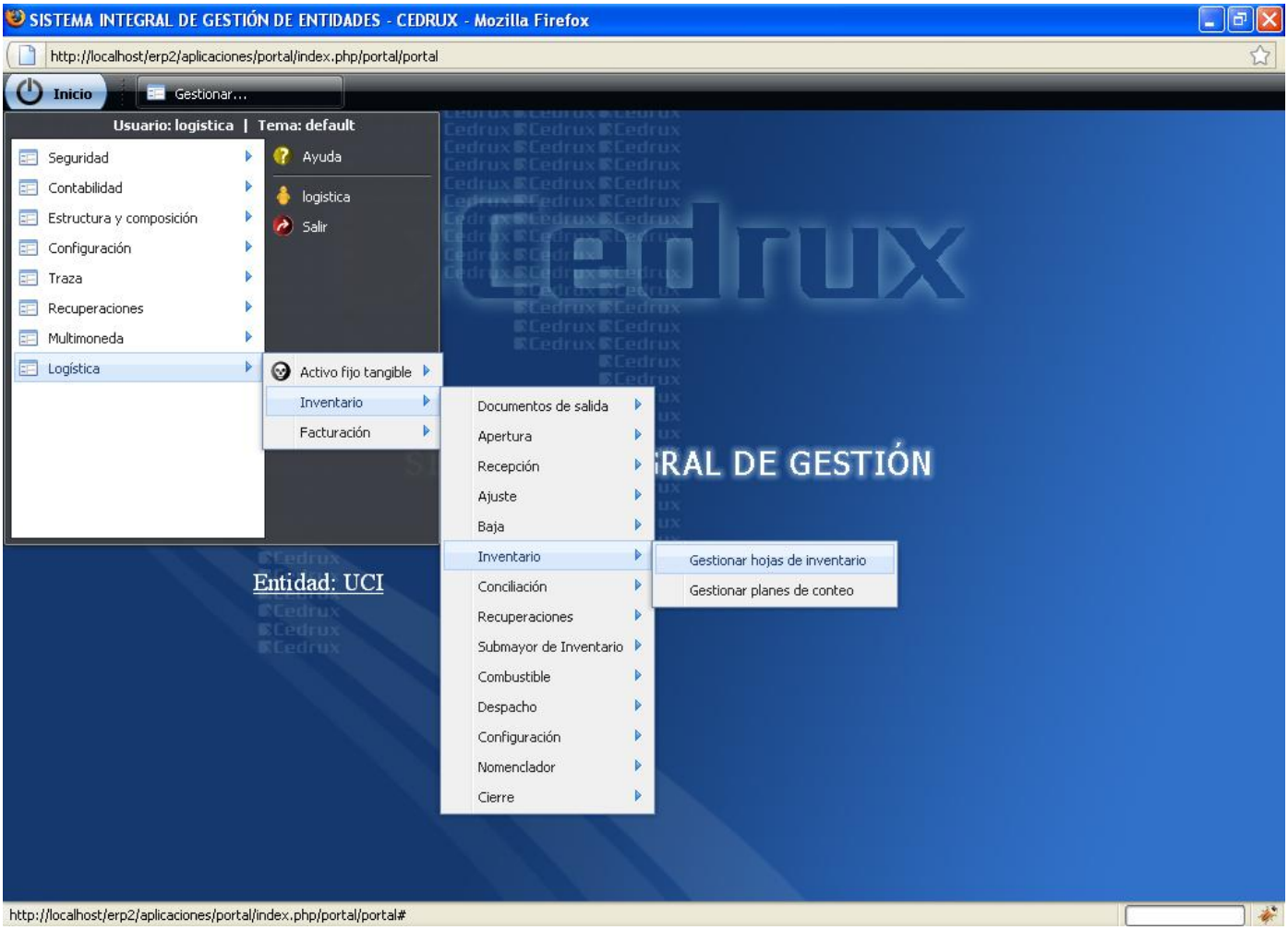

Figura 26 Interfaz para el acceso al módulo de inventario físico.

Anexo 2 Interfaz principal para la gestión de inventario físico.

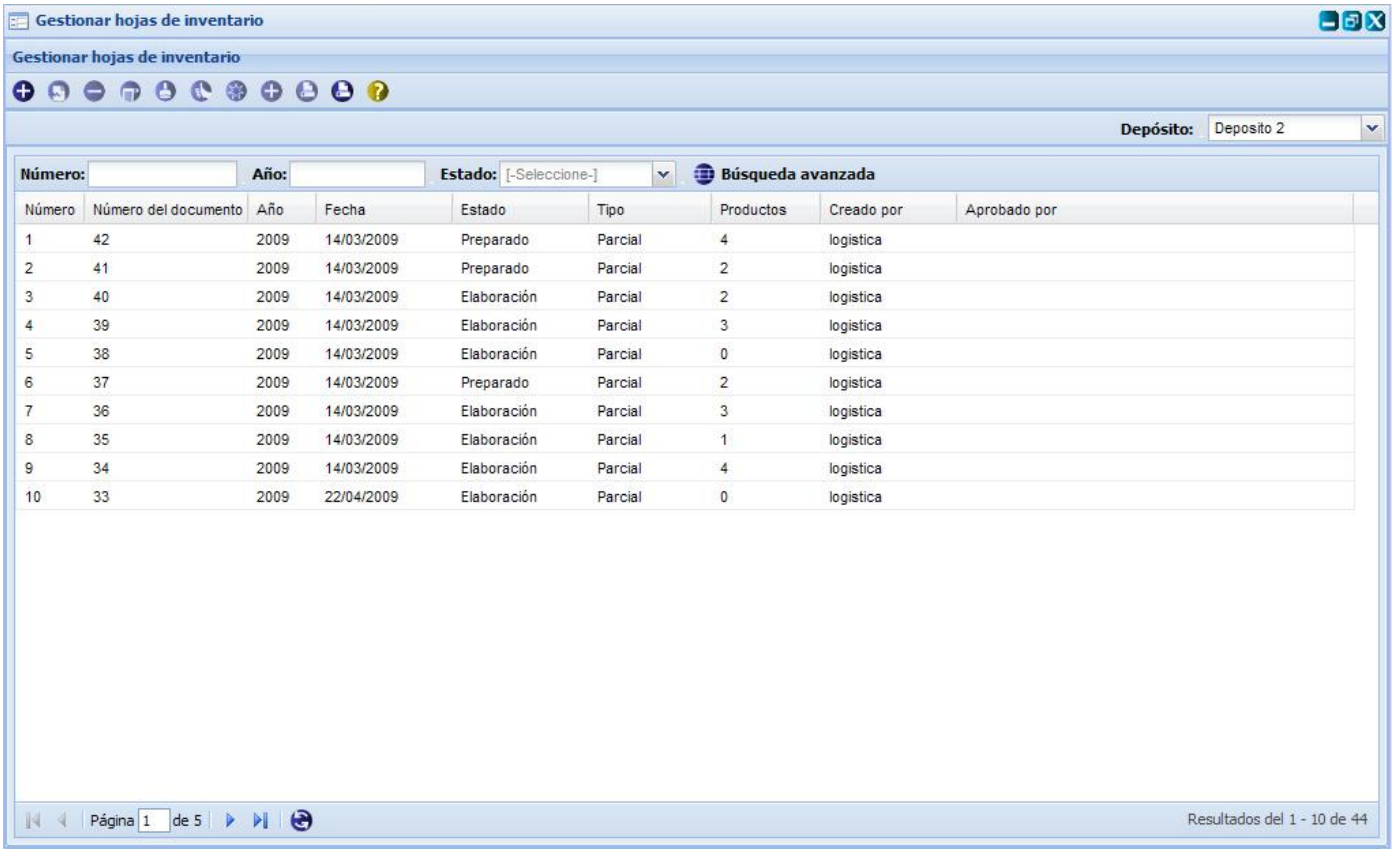

Figura 27 Interfaz principal para la gestión de inventario físico.

Anexo 3 Interfaz adicionar hoja de inventario.

Figura 28 Interfaz Adicionar hoja de inventario.

Anexo 4 Interfaz aprobar hoja de inventario.

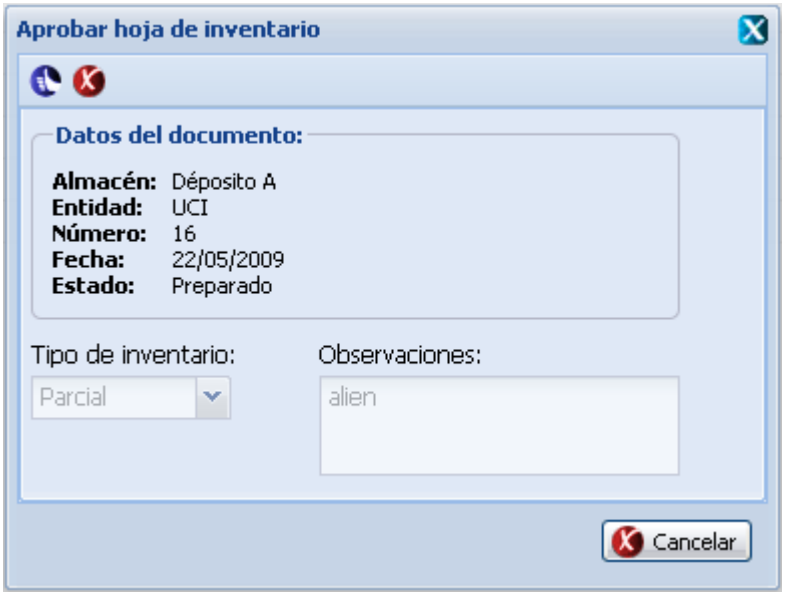

Figura 29 Interfaz Aprobar hoja de inventario.

Anexo 5 Interfaz para la gestión de productos.

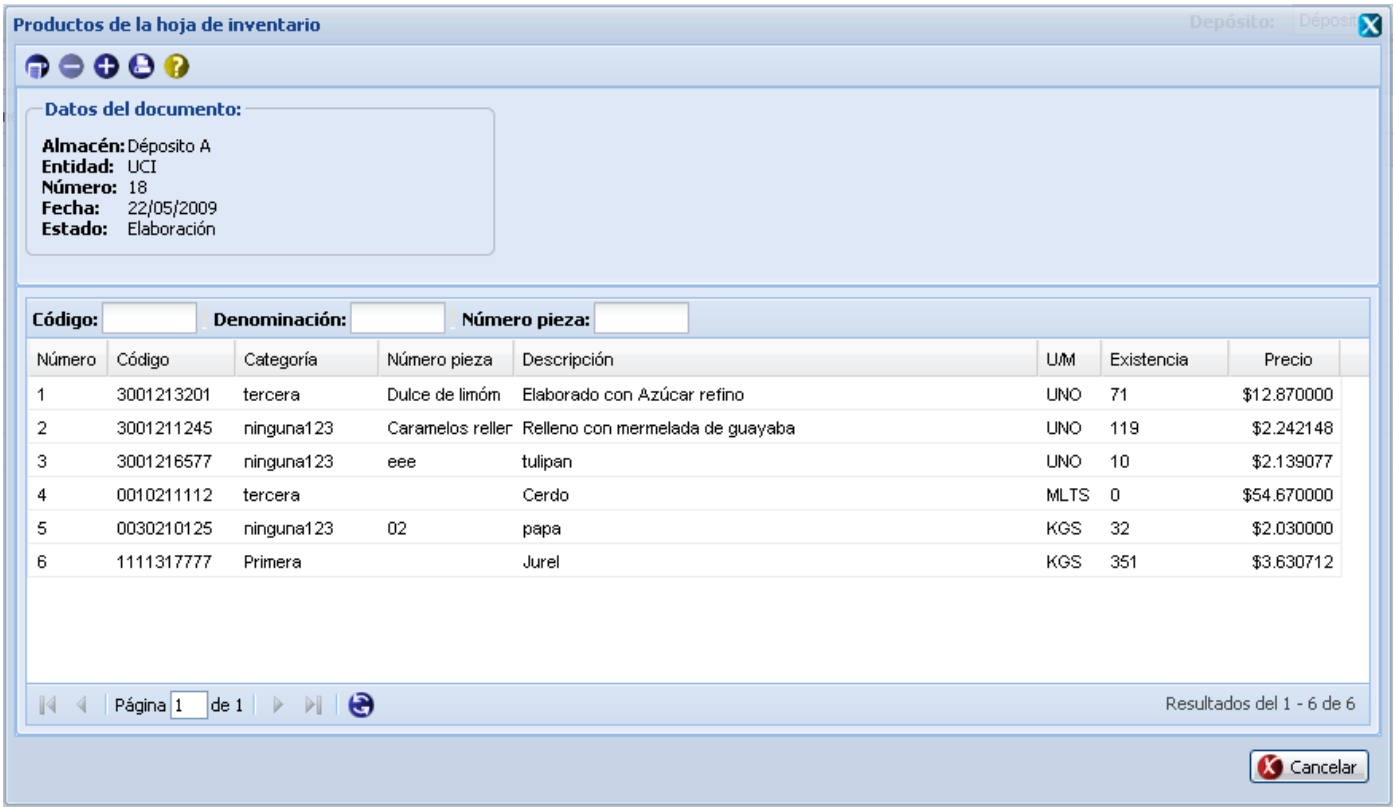

Figura 30 Interfaz Productos de la hoja de inventario.

Anexo 6 Interfaz registrar el conteo final.

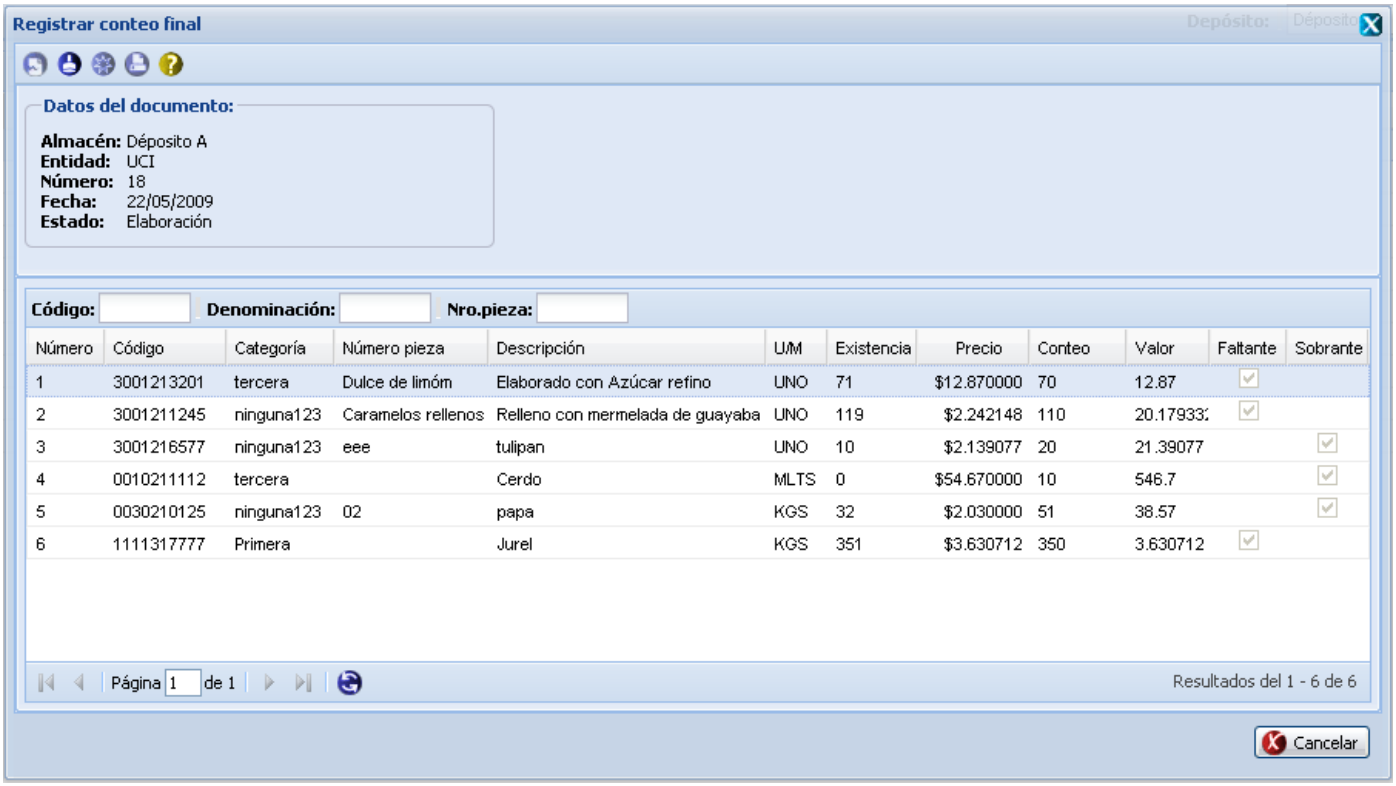

Figura 31 Interfaz para registrar el conteo final de los productos.

Anexo 7 Interfaz de productos ordenados por genérico.

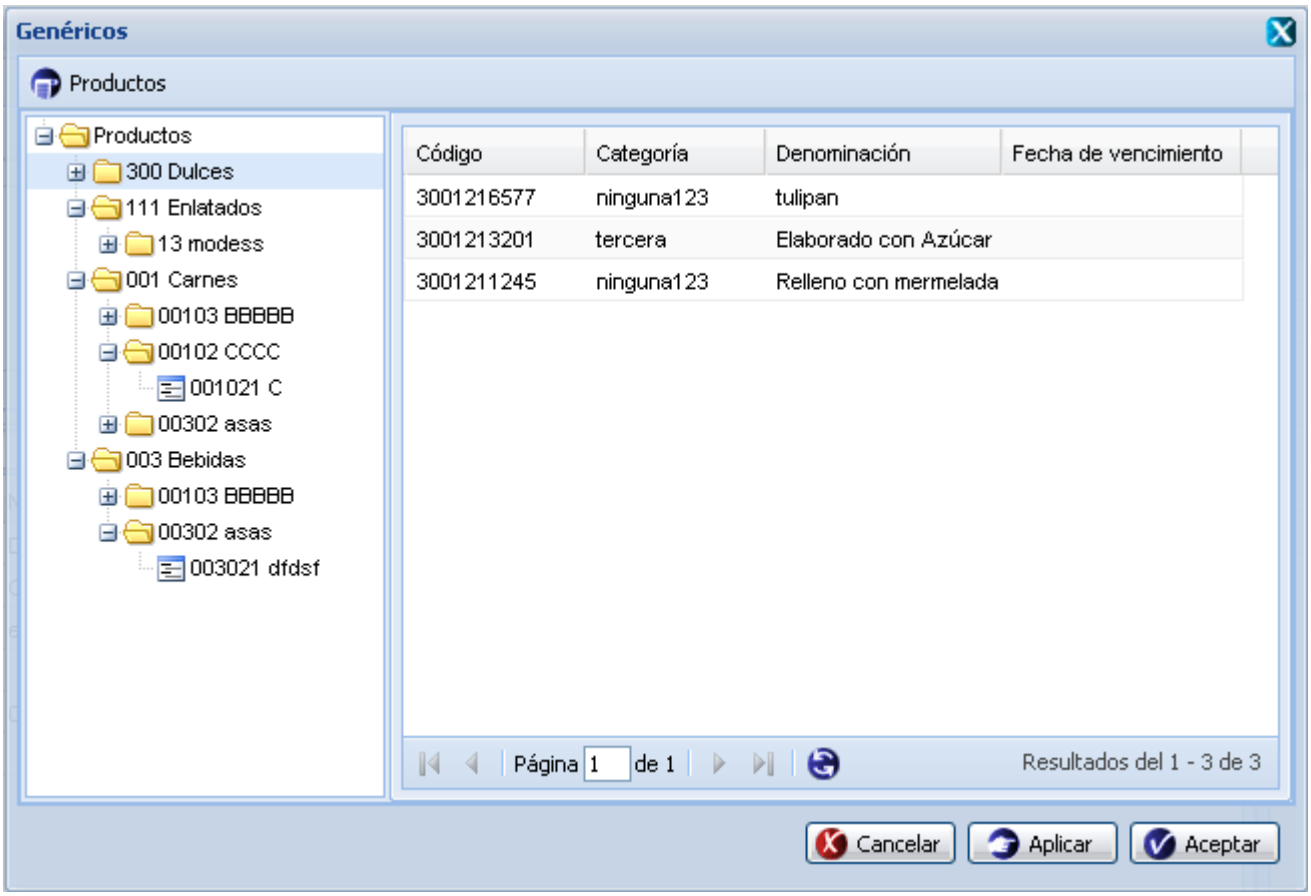

Figura 32 Interfaz de productos ordenados por génericos.

Anexo 8 Instrumento de medición de la métrica Tamaño operacional de clase (TOC).

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

Tabla 14 Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
GestinventarioModel	21	Alta	Alta	Baja
GestplanesconteoMode	22	Alta	Alta	Baja

I				
DatPlanconteo	9	Media	Media	Media
DatPlanconteom	25	Alta	Alta	Baja
BaseDatPlanconteo	2	Baja	Baja	Alta
BaseDatPlanconteom	2	Baja	Baja	Alta
GestproductoModel	7	Baja	Baja	Alta
GestmesesModel	9	Media	Media	Media

Tabla 15 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

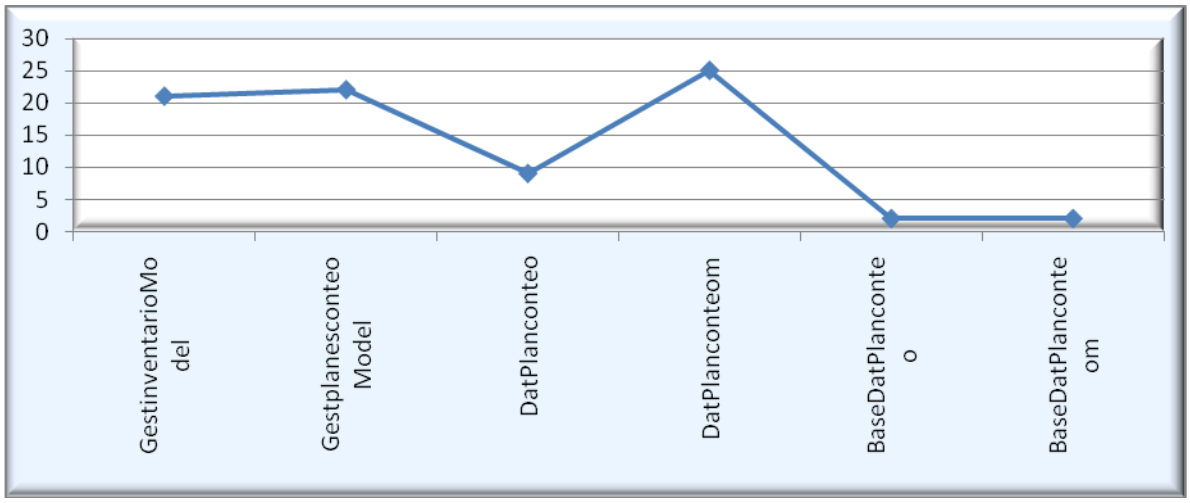

Figura 33 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

Anexo 9 Instrumento de medición de la métrica Relaciones entre clases (RC)

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

Tabla 16 Rango de valores de para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

Clase	Relaciones de Uso	Responsabilidad	Complejidad	Reutilización	Cantidad de Pruebas
GestinventarioModel	3	Alto	Media	Media	Media
GestplanesconteoModel	4	Alto	Alta	Baja	Alta
DatPlanconteo	2	Medio	Media	Media	Media
DatPlanconteom	2	Medio	Media	Media	Media
BaseDatPlanconteo	0	Ninguno	Baja	Alta	Baja
BaseDatPlanconteom	0	Ninguno	Baja	Alta	Baja
GestproductoModel	1	Bajo	Baja	Alta	Baja
GestmesesModel	2	Medio	Media	Media	Media

Tabla 17 Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas)

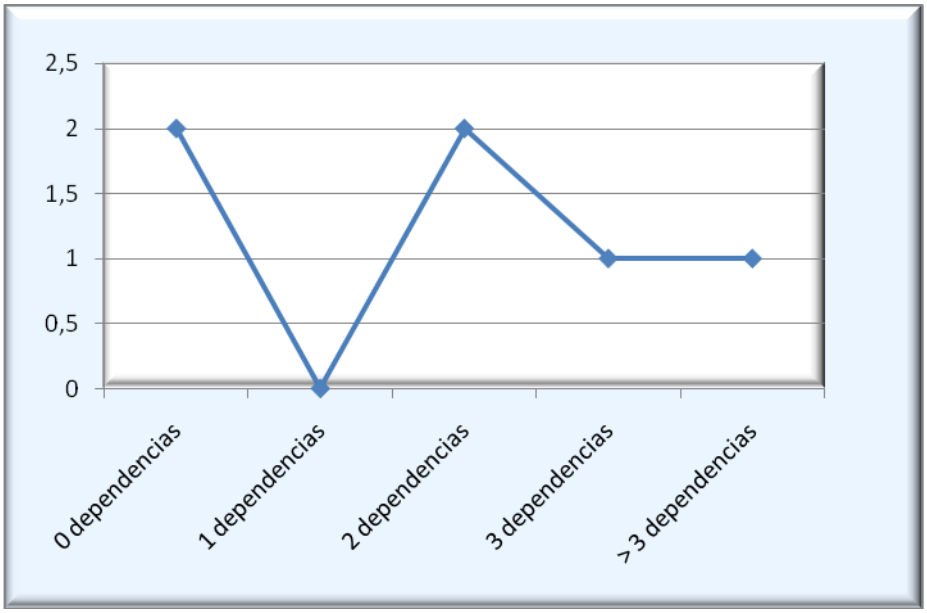

Figura 34 Resultados de la evaluación de la métrica RC agrupados por la tendencia de los valores.