Universidad de las Ciencias Informáticas

Facultad 15

Título: Implementación y prueba del Subsistema Estructura y Composición del Sistema Cedrux.

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

Autores: Dennis Rosa Sánchez
Rolando Ramos Morales

Tutores: Ing. Nemury Silega Martínez
Ing. Ramón Alberto Hernández Fernández

JUNIO DE 2010

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Facultad 15 de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Rolando Ramos Morales **Dennis Rosa Sánchez**

Firma de los autores

_____ _____

Nemury Silega Martínez **Ramón Alberto Hernández Fernández**

Firma de los Tutores

Dennis Rosa Sánchez

Agradecimientos

A mis **padres** por la educación, dedicación, amor, confianza y por haberme dado la vida los quiero mucho. A mis **dos hermanos** y a mis amigos que me han dado su apoyo incondicional todo el tiempo. A mi **familia** por estar presente cuando más la necesitaba y eso siempre lo tengo presente. A la **Revolución** y a **Fidel** por haberme dado la posibilidad de estudiar, por engendrar tantos sueños y hacer los míos realidad. A la **UCI** y todos sus **profesores** que me permitió forjarme como profesional y como ser humano. A mis tutores Ramón y a Nemury que ha sido como un hermano.

Dedicatoria

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

Rolando Ramos Morales

Agradecimientos

A mis padres por darme todo su cariño, amor y dedicación, además de ofrecerme todo lo necesario para estudiar y llegar hasta aquí. A mis Tíos primos, abuelos y toda mi familia en general que siempre me ayudaron, aquí está el fruto de su ayuda. A mi novia querida (Yaimara) por estar siempre a mi lado todo el tiempo y atenderme en momentos difíciles, a toda su familia que me acogió muy bien. A Robert, Marlenis y su hijo Robertico que son como parte de mi familia y siempre me ayudaron en el tiempo de estudios aquí en la Capital. A todos los profesores que han formado parte del claustro que me ha impartido la educación que he recibido. A mis tutores en especial Nemury y Ramón.

Dedicatoria

El trabajo se lo dedico a mi hermanito que es lo más preciado para mí. A mis padres queridos y a mi Cosita…

RESUMEN

Es necesidad en Cuba realizar un sistema de planificación de recursos materiales ERP con las características de las entidades cubanas. Se necesita automatizar la estructura y la información de las entidades en Cuba para mejorar el control, la planificación y la toma de decisiones. El ERP cubano debe cumplir la responsabilidad de ser multientidad, por lo que se debe gestionar todas las características de las entidades para que todos sus módulos utilicen la información de la entidad en que esté funcionando. El ERP cubano está en desarrollo en la Universidad de las Ciencias Informática (UCI) donde comprende un subsistema nombrado Estructura y Composición encargado de la gestión de la estructura de las entidades y la composición de las mismas. El subsistema posee una funcionalidad para definir el concepto estructural de Cuba, pero la misma es configurable y genérico logrando con esto que se visualice cualquier definición estructural que se necesite. En la implementación del módulo se utilizó las tecnologías y lenguajes propuestos por la línea de Arquitectura del proyecto ERP Cuba, se utilizó la técnica de los metadatos para lograr que el subsistema fuera configurable, dinámico y flexible en cuanto a la información que se gestione.

Palabras clave: ERP, Estructura y Composición.

ÍNDICE

INTRODUCCIÓN

Los sistemas de planificación de recursos empresariales (ERP por sus siglas en inglés) permiten gestionar procesos de una empresa con el objetivo de integrar información a lo largo de las entidades y eliminar las barreras entre diferentes áreas o entidades fluyendo la información por toda la empresa, contribuyendo a la mejora de los procesos fundamentales.

Además, un buen ERP es capaz de gestionar y de llevar a cabo el control de diversas actividades como: control de inventario, facturación y administración de nómina. Con el uso óptimo de un sistema ERP se puede optimizar los procesos más importantes de una entidad, puede tener acceso a toda la información de forma segura, precisa y oportuna porque todos los datos de la entidad están integrados, igualmente tiene la posibilidad de compartir la información entre todos los componentes de la organización, además se elimina los datos y las operaciones innecesarias de reingeniería. Por estas razones un ERP adquiere mayor relevancia en la actualidad, donde las tecnologías de la información y las comunicaciones son protagonistas en todas las esferas de la vida.

En la implementación de un ERP generalmente se requiere de un largo período de desarrollo, además de integrar varios factores que conlleven al éxito de la puesta en marcha. Es importante que los usuarios estén convencidos de los beneficios que se obtendrán con el ERP, pues facilitará la implementación en la empresa.

Además de los problemas que enfrenta Cuba como país pobre para poder proporcionar recursos y servicios, tiene que resistir la gran presión que ejerce el gobierno de los Estados Unidos que no cesa en su propósito de hundirla y privarla del disfrute de las nuevas tecnologías que existen. También se está llevando a cabo una crisis mundial muy aguda de la que Cuba no queda excluida.

Por todos los elementos expuestos se hace necesaria la creación de un ERP en Cuba, para poder maximizar las ganancias, llevar a cabo una mejor planificación, reducir la corrupción, mejorar el control de todos los recursos, facilitar la auditoría, entre otros elementos.

En correspondencia con lo anterior se decidió desarrollar un sistema de planificación de recursos empresariales que se pueda aplicar a todas las empresas cubanas para el desarrollo y el perfeccionamiento de las tomas de decisiones. Donde se presentan una serie de dificultades como es la relacionada con la gestión de las estructuras organizativas.

Situación Problemática

En Cuba se realiza el trabajo de la gestión de información organizativa de la estructura y de la composición de cualquier entidad de forma manual, donde está demostrado que se cometen errores en las anotaciones, errores conceptuales y pérdida de información. Es imprescindible controlar la información organizativa de las entidades evitando así la corrupción. La forma de inspeccionar o auditar a las entidades por sus organizaciones superiores es difícil al igual que la obtención de información a todos los niveles por las personas autorizadas. Es muy trabajoso conocer toda la información de una determinada entidad o de los niveles estructurales en general lo que obstaculiza la toma de decisiones. En las empresas se puede modificar la plantilla por personas indebidas incluyendo cargos no autorizados y a veces no se lleva un histórico lo que proporciona el desvío de dinero y de cualquier tipo de material. El sistema Cedrux es un software multientidad por lo que sus subsistemas especializados no conocen en qué entidad están funcionando, necesitan información que contienen las entidades para poder realizar correctamente sus funciones. Cedrux no puede concederles permisos a cada uno de sus módulos por niveles estructurales a los que ellos pueden acceder.

Por todo lo antes expuesto se determinó el siguiente **problema científico**:

El deficiente proceso de gestión de estructuras actualmente en Cuba trae como resultado insuficiencias en la gestión empresarial.

Objeto de Estudio

Proceso de desarrollo de software.

Campo de Acción

Desarrollo del software para la gestión de estructuras en el Sistema ERP Cedrux.

Objetivo General

Implementar el subsistema Estructura y Composición en el Sistema ERP Cedrux.

Objetivos Específicos

1. Determinar el diseño teórico de la investigación.
2. Estudiar y comprobar los requerimientos del software.
3. Implementar el subsistema.

4. Probar el subsistema

Idea a desarrollar

Al implementar el subsistema Estructura y Composición permitirá mejorar el proceso de gestión de estructuras en las entidades cubanas.

Tareas investigativas

1. Recopilar bibliografía necesaria para el estudio del arte y familiarización con las herramientas a utilizar.
2. Estudiar el marco de trabajo utilizado para el desarrollo del sistema ERP Cedrux.
3. Estudiar e investigar sobre el negocio y los requisitos funcionales del subsistema.
4. Realizar comparaciones de los diferentes sistemas para la gestión de estructuras.
5. Preparar y realizar taller para el entendimiento de los requisitos con los especialistas en el tema.
6. Realizar taller con los especialistas para la validación de los requisitos funcionales descritos anteriormente.
7. Estudiar y documentar sobre los metadatos como propuesta de solución para lograr que el software sea dinámico y configurable.
8. Estudiar sobre los patrones arquitectónicos y de diseño.
9. Estudiar sobre el análisis y diseño realizado para la implementación del módulo Estructura y Composición.
10. Evaluar el análisis y diseño realizado.
11. Estudiar estándares de codificación y de diseño de interfaz.
12. Implementar el componente EAV para la configuración del módulo Estructura y Composición.
13. Implementar el componente Gestión de Estructura.
14. Realizar la integración de los componentes.
15. Realizar y documentar artefactos resultantes de la implementación del software.

16. Planificar y diseñar pruebas a realizar al software.
17. Realizar pruebas de Unidad al software.
18. Realizar pruebas funcionales al software y documentarlas, para verificar el cumplimiento de las funcionalidades.

Estrategia de Investigación

Con el objetivo de establecer las características estructurales y funcionales, así como las correlaciones y clasificaciones de la gestión de estructuras en Cuba, se empleó una estrategia descriptiva de investigación.

Métodos y Técnicas Investigativas a utilizar

Para desarrollar la investigación científica del objeto de estudio se hace necesario el uso de los métodos teóricos. Se empleará el Análisis Histórico lógico ya que se estudiará y se investigará la trayectoria real del funcionamiento y desarrollo de la gestión de estructuras y su composición en las entidades. También se va a recurrir al método de Modelación y dentro del mencionado el modelo Analógico para representar la relación y propiedades fundamentales que representen una reproducción simplificada de la realidad, con esto se establece una similitud entre el sistema a desarrollar y lo que ocurre en el entorno real.

El trabajo está estructurado de la siguiente forma:

Capítulo I Fundamentación Teórica: En este capítulo se realiza un estudio del arte de los sistemas similares. Se hace un estudio sobre los conceptos de desarrollo de software y las tecnologías y herramientas que se utilizaron para el desarrollo del módulo.

Capítulo II Descripción de la implementación del sistema: Se describirá la solución propuesta para el Subsistema Estructura y Composición del Sistema Cedrux y se detallará sobre la utilización de los metadatos como parte del desarrollo del sistema.

Capítulo III Validación de la solución propuesta: Se documentan las pruebas realizadas a la solución propuesta para demostrar la validez del software.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se realiza una investigación sobre los sistemas similares que existen en el mundo, se resaltan los principales conceptos sobre las tecnologías utilizadas como son los lenguajes, herramientas de programación y modelado. Además, se describen los marcos de trabajos que ayudan a realizar una aplicación más robusta, flexible y segura.

1.2 Los ERP en la actualidad

1.2.1 ¿Qué es un ERP?

Los sistemas de planificación de recursos empresariales son sistemas de información gerencial que integran y manejan muchos de los negocios asociados con las operaciones de producción y de los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

La Planificación de Recursos Empresariales es un término derivado de la Planificación de Recursos de Manufactura (MRPII) y seguido de la Planificación de Requerimientos de Material (MRP). Los sistemas ERP típicamente manejan la producción, logística, distribución, inventario, envíos, facturas y contabilidad de la compañía. Sin embargo, la Planificación de Recursos Empresariales o el software ERP pueden intervenir en el control de muchas actividades de negocios como ventas, entregas, pagos, producción, administración de inventarios, calidad de administración y la administración de recursos humanos. (Ing. Henry Cruz Mulet, 2009).

1.2.2 Tendencia actual de los ERP

Las tendencias comerciales actuales y futuras obligan a las empresas a ser cada vez más competitivas. Para ser competitiva es necesario que una compañía tenga optimizado e integrado sus flujos internos de información y sus relaciones comerciales externas, y así conseguir objetivos básicos como son las mejoras de la productividad, la calidad, el servicio al cliente y la reducción de costes. Las tecnologías de la información han permitido, en gran medida, la consecución de dichos objetivos. En esta área, podemos reseñar la aportación de los ERP y las ventajas del comercio electrónico o intercambio electrónico de información con asociados comerciales y clientes finales a través de proyectos basados en aplicaciones WEB y de la mensajería.

Los ERP están funcionando ampliamente en todo tipo de empresas modernas. Todos los departamentos funcionales que están involucrados en la operación o producción están integrados en un solo sistema.

La Universidad de las Ciencias Informáticas siguiendo una estrategia del país y la necesidad económica de un software de este calibre se encuentra desarrollando el producto Cedrux, un ERP enfocado en solucionar problemáticas propias del país (como el uso de y el trabajo con dualidad monetaria) pero pensado para que se convierta en una solución genérica aplicable a cualquier entorno nacional o internacional (Ing. Henry Cruz Mulet, 2009) e integrarse a sistemas externos como por ejemplo software de bancos, carnet de identidad, etc.

1.2.3 Sistemas ERP que existen en el mundo

Algunos de los ERP más usados en el mundo son: Compiere, Openbravo, SAP, K2B, Sentai, EXACT, BKMIS, OpenXpertya, VERSAT - Sarasola. Se mostrarán características de algunos:

SAP es el nombre de un ERP comercial y de una compañía alemana creada en 1972 que emplea a más de 27 mil personas en 50 países diferentes. Dicho ERP está formado por varios módulos dentro de los cuales se encuentra Contabilidad Financiera, Inversiones, Tesorería, Gestión Datos Generales de Logística, Gestión del Mantenimiento, Enterprise Controlling, Ventas y Distribución, Gestión de Proyectos, Gestión de Materiales, Calidad, Producción, Gestión del Personal, ayudando así a optimizar los procesos de negocios de la empresa. Uno de los principales problemas del ERP SAP con la implantación del software es la resistencia al cambio, además de la inversión económica en el paquete y la capacitación, la empresa debe invertir tiempo y esfuerzo en hacer el cambio en la forma de trabajar de sus integrantes. (www.sap.com, 2009).

Oracle E-Business Suite es un conjunto totalmente integrado y completo de aplicaciones comerciales para la empresa. Ya sea que usted implemente un solo módulo a la vez, múltiples módulos o la suite completa. Es desarrollado por la corporación Oracle, esta es considerada como una de las mayores compañías de software del mundo, donde sus productos van desde bases de datos hasta sistemas de gestión. Tiene su sede en la localidad Californiana de Redwood City de los Estados Unidos. Contiene diferentes módulos los cuales gestionan las principales funcionalidades de una empresa, dentro de estos se encuentran la Adquisición Avanzada, Contratos, Gestión del Rendimiento Corporativo, Inteligencia de

Negocio Diaria, Gestión de Datos de Cliente, Gestión de Relaciones con los Clientes, Finanzas, Gestión de Recursos Humanos, Centro de Interacción, Gestión del Aprendizaje, Logística y Mantenimiento. (www.oracle.com, 2009).

OpenBravo ERP es un revolucionario ERP que ofrece varias ventajas para el que lo utiliza, entre estas podemos encontrar:

- ✓ Es orientado a pequeñas y medianas empresas, frente a otros ERP orientados a grandes corporaciones.
- ✓ Se adapta a las necesidades del cliente.
- ✓ Posee un entorno web basado en JAVA, que le garantiza no necesitar la instalación de aplicación ninguna en los equipos clientes.
- ✓ Es software libre, para asegurar el acceso libre y sin costes al código del programa, no tendrá ataduras con el proveedor pudiendo cambiar de proveedor sin problemas.

Por el uso de este sistema no se incurre en costo alguno, al ser una solución de software libre no pagará licencias, por número de usuarios, por módulos o funcionalidades como con los ERP privativos, solo pagará por los servicios que necesiten en el momento que lo precise. (www.openbravo.com, 2009).

OpenXpertya es un completo software ERP/CRM, de código abierto (software libre), desarrollado usando tecnología multiplataforma (Java), y preparado para soportar varias bases de datos (Oracle, PostgreSQL, Firebird, Sybase), tiene capacidad multientidad, multiempresa, multicentro, multialmacén, multicaja, etc., haciendo posible la descentralización de una organización y siendo el tipo de aplicación ideal para una cadena de franquicias, una empresa de distribución, de producción o de servicios. (www.openxpertya.org, 2009).

El VERSAT - Sarasola es un Sistema integrado de gestión económica, diseñado para ser utilizado por el sector empresarial Cubano, que se adecua a las características de cada entidad, ya que es configurable por cada una de ellas en el momento de su instalación y tiene como objetivo fundamental ofrecerle a los usuarios la posibilidad de contar con un instrumento seguro, rápido, eficaz y de fácil manejo para la Planificación, Control y el Análisis de la Gestión Económica. Se considera el primer sistema de Contabilidad Cubano certificado, creado en la provincia Villa Clara, municipio de Santa Clara y desarrollado por el Ministerio del Azúcar. Es una aplicación escritorio, constituido por 12 módulos que

incluyen Configuración, Administración, Paquetes de Gestión Contabilidad General, Activos Fijos, Control y Pago, Finanzas (Caja y Banco), Presupuesto Maestro, Costos y Procesos, Inventarios, Contratación y Facturación, Nominas de Salarios. Las herramientas que se utilizan son: Delphi Versión 5.0 para su implementación, el Gestor Base de Datos SQL Server 7.0 y puede correr sobre el sistema operativo Windows 2000 o uno superior. (Lic. Miguel P. Cabrera González, 2004).

1.2.4 Valoración crítica.

Luego de realizar un estudio de algunos ERP que existen en el mundo, es correcto indicar que la solución de estos sistemas sería capaz de resolver algunos problemas de las empresas cubanas. Pero tienen grandes desventajas porque utilizan tecnologías que son privadas a Cuba debido al bloqueo impuesto por los Estados Unidos, además de que muchos son software propietarios y el país no esta en condiciones de pagar por la instalación de un sistema ERP ni por las licencias y la renovación de las mismas ya que son altamente costosas, en el caso de los que son software libre y código abierto, igual usan alguna tecnología como puede ser de base de datos o de desarrollo que son de propiedad de las empresas privadas y las mismas lo impiden por lo antes mencionado. Muchos dependen de la plataforma y del sistema operativo en el que pueden trabajar. Los sistemas descritos no poseen funcionalidades que puedan resolver el trabajo con la organización estructural de las entidades cubanas y no poseen un módulo configurable que resuelva el problema de la multientidad en Cuba después de ser instalado. Los ERP se ven como sistemas difíciles de adaptarse al flujo específico de los trabajadores y del proceso de negocios de algunas compañías, este punto se menciona como una de las causas principales de falla. Corresponde a Cuba desarrollar un sistema ERP que sea multi-plataforma, software libre, que pueda ser utilizado con pocos recursos y donde cuente con un módulo que se integre con los demás, brindando mediante servicios la información de las entidades.

1.3 Metodología de desarrollo de software

“Una metodología es un sistema de prácticas, técnicas, procedimientos y normas utilizado por quienes trabajan en una disciplina.” (Guía de PMBOK, 2004)

El ciclo de vida de un proyecto es un conjunto de fases del proyecto que, generalmente son secuenciales, cuyos nombres y números son determinados por las necesidades de control de la organización u organizaciones involucradas en el proyecto. Un ciclo de vida puede ser documentado con una metodología. (Guía de PMBOK, 2004)

El ERP Cedrux por su magnitud, por sus características especificas y por su necesidad de desarrollo propone para el ciclo de vida del proyecto una metodología de desarrollo de software nombrada Modelo de desarrollo orientada a componentes del proyecto ERP Cuba. (Equipo de producción, 2009)

La misma presenta las siguientes cararcteristicas:

- ✓ **Centrado en la arquitectura**

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

- ✓ **Orientada a componente**

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

- ✓ **Iterativo e incremental**

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

- ✓ **Ágil y adaptable**

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados. (CENTRO DE SOLUCIONES DE GESTIÓN, 2009)

La metodología propuesta cuenta con cuatro fases (Concepción, Elaboración, Construcción y Cierre) y una serie de disciplinas y actividades por roles que se deben cumplir durante el paso por las distintas fases. En el caso del modelo de desarrollo Ágil y adaptable no es recomendable para proyectos de gran envergadura como los ERP porque el mismo no genera toda la documentación o evidencia, por lo que es necesario complementarlo con otros procesos. No es el objetivo del presente trabajo buscar una metodología para el desarrollo del software por lo tanto se utilizará la propuesta por la dirección del proyecto.

1.4 Estilos arquitectónicos

Cuando lo que interesa no es obtener una configuración concreta, sino extraer los patrones genéricos que definen a una familia de sistemas, se habla de estilos arquitectónicos. La definición de estilos tiene una gran importancia desde el punto de vista de la reutilización, y es un aspecto fundamental en la arquitectura del software. (Iribarne Martínez, 2003)

Actualmente, en la comunidad de arquitecturas de software existe una gran variedad de elementos arquitectónicos que simplifican las tareas de diseño en la construcción de una arquitectura. Estos elementos arquitectónicos se conocen con el nombre de estilos arquitectónicos. “Un estilo arquitectónico está compuesto por un conjunto de estilos de componente a nivel arquitectónico y por unas descripciones de patrones de interacción entre ellos” (Shaw, y otros, 1996) (Bass, y otros, 1998). Estos tipos de componente son utilizados para modelar las interacciones que tienen lugar en una infraestructura de componentes. (Iribarne Martínez, 2003)

1.4.1 Arquitectura basada en Componentes

Una arquitectura de componentes es un conjunto de componentes de software a nivel de aplicación, con sus relaciones estructurales y sus dependencias de comportamiento. Una arquitectura de componentes puede ser utilizada para una aplicación de software (a partir de componentes) o para un sistema de software como un conjunto de aplicaciones (consideradas estas como componentes y subcomponentes). (Iribarne Martínez, 2003).

“Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio”. (Szyperski, 1998)

Otra definición dada por Pressman “ Un componente es una parte reemplazable, casi independiente y no trivial de un sistema que cumple una función clara en el contexto de una arquitectura bien definida” (Pressman, 2002).

Los componentes de software son piezas de software auto contenidas, reusables, accesibles sólo a través de interfaces bien definidas. (Szyperski, 1998).

Una de las grandes ventajas de los componentes es la reusabilidad. Una reutilización efectiva depende no sólo de la identificación apropiada de los componentes, sino del modo en que dichos componentes son

combinados y organizados. Las arquitecturas de software basadas en componentes brindan el soporte para la integración de partes en sistemas mayores, facilitando la definición de una estructura de ensamblado adecuada. El empleo de esta técnica de desarrollo de software requiere por lo tanto de un cuidadoso modelado arquitectural y análisis, a los fines de asegurar reusabilidad y compatibilidad entre los componentes a interactuar. (Iribarne Martínez, 2003)

1.5 Patrón arquitectónico Modelo-Vista-Controlador

- ✓ Es un patrón arquitectónico de software que separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes. (Burbeck, 1992)
- ✓ Modelo. El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).
- ✓ Vista. Maneja la visualización de la información.
- ✓ Controlador. Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

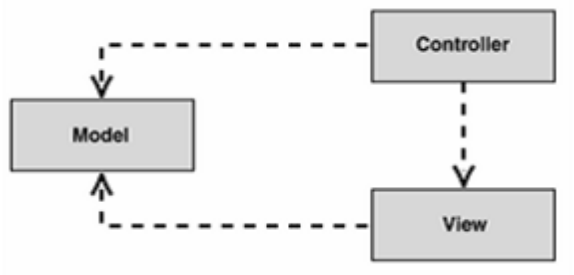


Figura 1.1 Modelo Vista Controlador.

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. En aplicaciones Web, por otra parte, la separación entre la vista (el browser) y el controlador (los componentes del lado del servidor que manejan los requerimientos de HTTP) está mucho más claramente definida. (Burbeck, 1992).

Entre las ventajas y desventajas del patrón señaladas en la documentación de Patterns & Practices de

Microsoft están las siguientes:

Ventajas:

- ✓ **Soporte de vistas múltiples.** 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 de 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. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Desventajas:

- ✓ **Complejidad.** El patrón introduce nuevos niveles de indirección y por lo tanto aumenta ligeramente la complejidad de la solución. También se profundiza la orientación a eventos del código de la interfaz de usuario, que puede llegar a ser difícil de depurar. En rigor, la configuración basada en eventos de dicha interfaz corresponde a un estilo particular (arquitectura basada en eventos) que aquí se examina por separado.
- ✓ **Costo de actualizaciones frecuentes.** Desacoplar el modelo de la vista no significa que los desarrolladores del modelo puedan ignorar la naturaleza de las vistas. (Microsoft, 2003).

1.6 Patrones de diseños

1.6.1 Controlador

Un Controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. (Visconti, y otros, 2004)

Problema:

¿Quién debería encargarse de atender un evento del sistema?

Un evento del sistema es un evento de alto nivel generado por un actor externo; es un evento de entrada externa. Se asocia a operaciones del sistema: las que emite en respuesta a los eventos del

sistema. Por ejemplo, cuando un cajero que usa un sistema de terminal en el punto de venta oprime el botón, "terminar venta", está generando un evento sistémico que indica que "la venta ha terminado". (Visconti, y otros, 2004).

Solución:

Asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones: el "sistema" global (controlador de fachada) la empresa u organización global (controlador de fachada) algo en el mundo real que es activo (por ejemplo, el papel de una persona) y que pueda participar en la tarea (controlador de tareas) un manejador artificial de todos los eventos del sistema de un caso de uso, generalmente denominados, "Manejador<NombreCasodeUso>" o (controlador de casos de uso). (Visconti, y otros, 2004)

1.6.2 Controlador Frontal

Es un patrón de diseño que se basa en usar un controlador como punto inicial para la gestión de las peticiones. El controlador gestiona estas peticiones, y realiza algunas funciones como: comprobación de restricciones de seguridad, manejo de errores, mapear y delegación de las peticiones a otros componentes de la aplicación que se encargarán de generar la vista adecuada para el usuario. La siguiente figura muestra un esquema de ello: (www.programacion.com, 2009)

Ventajas:

- ✓ Está centralizado en un único punto la gestión de las peticiones
- ✓ Aumenta la reusabilidad de código
- ✓ Mejora la gestión de la seguridad

Desventajas:

- ✓ La velocidad de respuesta disminuye al tener que ser procesadas las peticiones primero por el controlador.

1.6.3 Bajo Acoplamiento

Problema

¿Cómo dar soporte a una dependencia escasa y a un aumento de la reutilización?

Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. (Visconti, y otros, 2004)

- ✓ Acoplamiento bajo significa que una clase no depende de muchas clases.
- ✓ Acoplamiento alto significa que una clase recurre a muchas otras clases. Esto presenta los siguientes problemas:
  - Los cambios de las clases afines ocasionan cambios locales.
  - Difíciles de entender cuando están aisladas.
  - Difíciles de reutilizar puesto que dependen de otras clases.

Solución

Asignar una responsabilidad para mantener bajo acoplamiento.

1.6.4 Alta Cohesión

Problema:

¿Cómo mantener la complejidad dentro de límites manejables?

Cohesión:

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase.

Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. (Visconti, y otros, 2004)

Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo.

Esto presenta los siguientes problemas:

- ✓ Son difíciles de comprender.
- ✓ Difíciles de reutilizar.
- ✓ Difíciles de conservar.
- ✓ Las afectan constantemente los cambios.

Solución:

Asignar una responsabilidad de modo que la cohesión siga siendo alta.

1.6.5 Experto

Solución:

Asignar una responsabilidad al experto en la información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

Problema:

¿Cuál es el principio fundamental en virtud del cual se asignan las responsabilidades en el diseño orientado a objetos?

Un modelo de clase puede definir docenas y hasta cientos de clases de software, y una aplicación tal vez requiera el cumplimiento de cientos o miles de responsabilidades. Durante el diseño orientado a objetos, cuando se definen las interacciones entre los objetos, tomamos decisiones sobre la asignación de responsabilidades a las clases. Si se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se nos presenta la oportunidad de reutilizar los componentes en futuras aplicaciones. (Larman, 1999).

1.7 Herramientas tecnológicas

1.7.1 NetBeans 6.5

NetBeans IDE es un Entorno de Desarrollo Integrado (IDE) disponible para Windows, Mac, Linux y Solaris. Es un producto libre y gratuito sin restricciones de uso escrito completamente en Java usando la plataforma NetBeans. La plataforma NetBeans es una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Ambos productos son de código abierto y gratuito para uso tanto comercial como no comercial. El código fuente está disponible para su reutilización de acuerdo con la Licencia Común de Desarrollo y Distribución (CDDL) v1.0 y la GNU Licencia Pública General (GNU GPL) v2. NetBeans IDE contiene las herramientas para que los desarrolladores de software puedan crear aplicaciones desktop, Enterprise, web, y aplicaciones móviles, con el lenguaje Java, así como también C/C++, PHP, JavaScript, Groovy y Ruby. El IDE provee mejoras en el soporte de entornos generales (frameworks) web (Hibernate, Spring, JSF, JPA), el servidor GlassFish, y bases de datos. Adicionalmente tiene una interfaz de usuario más amigable y una Compilación automática al Guardar. (www.netbeans.org, 2008)

1.7.2 Subversion 1.4.5

Subversion (SVN) es un software libre bajo una licencia de tipo Apache/BSD que se utiliza en el control de versiones en la que los desarrolladores puedan trabajar en un mismo proyecto en forma ordenada. Tiene una arquitectura cliente-servidor con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo, esto permite que se tenga un historial o que se recuperen versiones anteriores de los archivos. (www.subversion.apache.org, 2010).

1.7.3 Apache servidor 2.0

Es un programa que te permitirá crear un servidor http en tu propio ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir que características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor.

Características principales:

- ✓ Multiplataforma
- ✓ Es un servidor de web conforme al protocolo HTTP/1.1
- ✓ Basado en hebras en la versión 2.0.
- ✓ Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- ✓ Se desarrolla de forma abierta
- ✓ Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que se destaca PHP, un lenguaje de programación del lado del servidor. (www.apache.com, 2009)

1.7.4 PostgresSQL 8.3

PostgreSQL es un poderoso sistema de base de datos relacional de código abierto. Cuenta con una arquitectura de sólida reputación de confiabilidad, integridad de datos y corrección. Funciona en todos los principales sistemas operativos, incluyendo Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), y Windows. Es totalmente compatible con ACID, tiene soporte completo para claves foráneas,

uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas). Se incluye la mayoría de SQL: 2008 tipos de datos, incluyendo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, y TIMESTAMP. También es compatible con el almacenamiento de objetos binarios, incluyendo imágenes, sonidos o vídeo. Tiene interfaces de programación nativas de C / C + +, Java, .NET, Perl, Python, Ruby, Tcl, ODBC, entre otros. Es compatible con conjuntos de caracteres internacionales, codificación de caracteres multibyte, Unicode, y es consciente de la configuración regional para la clasificación, caso-sensibilidad, y el formato. Es altamente escalable, tanto en la enorme cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede acomodar. (www.postgresql.org, 2009)

1.7.5 PgAdmin III 1.8

PgAdmin III es una aplicación gráfica para gestionar el sistema de bases de datos relacional PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets (bibliotecas multiplataforma y libres para el desarrollo de interfaces gráficas programadas en lenguaje C++), lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres. (www.guia-ubuntu.org, 2009).

1.7.6 EMS SQL Manager for PostgreSQL 4.1.0.1

Es una herramienta de alto rendimiento para la administración y desarrollo de bases de datos PostgreSQL. Funciona con cualquier versión de PostgreSQL, hasta la más reciente y soporta las últimas características incluyendo PostgreSQL enumerados, búsqueda de texto, XML, las matrices de los tipos de compuestos, y otros. SQL Manager for PostgreSQL ofrece un montón de herramientas potente como Visual Database Designer para facilitar la creación de bases de datos, Visual Query Builder para crear complicadas consultas y muchas características más útiles para la administración eficiente de PostgreSQL. SQL Manager for PostgreSQL tiene además una amigable interfaz gráfica de usuario. (www.sqlmanager.net, 2009).

1.7.7 Navegador web Firefox Mozilla versión 3.5

Es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS

X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre y tiene variedades de plugins para una mejor configuración y desarrollo de aplicaciones web.

Las características de Mozilla Firefox son las siguientes:

- ✓ Multiplataforma.
- ✓ Cuenta con una protección antimalware e integración con el antivirus.
- ✓ La navegación por pestañas.
- ✓ Bloqueador de ventanas emergentes.
- ✓ Múltiples Extensiones.
- ✓ Posee gestor de descargas.
- ✓ Utiliza el sistema SSL para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS. (www.mozilla-europe.org, 2009)

1.7.8 Visual Paradigm 6.4

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

Dentro de sus características. (www.visual-paradigm.com, 2009).

- ✓ Multiplataforma.
- ✓ Interoperabilidad.
- ✓ Modelamiento de los Requisitos.
- ✓ Colaboración de Equipo.
- ✓ Generación de Documentación.
- ✓ Editor de Detalles de Casos de Uso.
- ✓ Ingeniería de Código.
- ✓ Modelado de Procesos de Negocio.
- ✓ Integración con Entornos de Desarrollo.
- ✓ Modelamiento de Bases de Datos.

1.7.9 JMeter 2.3.4

JMeter es una herramienta de carga para llevar a cabo simulaciones sobre cualquier recurso de Software. Inicialmente diseñada para pruebas de estrés en aplicaciones web, hoy en día, su arquitectura ha evolucionado no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en Bases de Datos, programas en Perl, requisiciones FTP y prácticamente cualquier otro medio.

Además, posee la capacidad de realizar desde una solicitud sencilla hasta secuencias de requisiciones que permiten diagnosticar el comportamiento de una aplicación en condiciones de producción. En este sentido, simula todas las funcionalidades de un Navegador ("Browser"), o de cualquier otro cliente, siendo capaz de manipular resultados en determinada requisición y ser reutilizables para ser empleados en una nueva secuencia. JMeter es una herramienta código abierto muy completa, implementada en Java para su utilización es necesario instalar la maquina virtual de Java versión 1.3 en adelante y una computadora con 1GB de RAM o superior. (http://www.osmosislatina.com/jmeter/basico.htm, 2009).

1.8 Lenguajes

1.8.1 Hipertext Preprocesor 5.2.5

Es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor para la creación de páginas webs dinámicas, gratuito e independiente de plataforma, rápido, con una gran librería de funciones y compatible con todos los navegadores. Además, es un lenguaje de código abierto que permite la programación Orientada a Objeto. (www.php.net, 2010)

1.8.2 Java Script

Es un lenguaje de programación del lado del cliente que se puede utilizar para construir sitios Web y para hacerlos más interactivos. Aunque comparte muchas de las características y de las estructuras del lenguaje Java, fue desarrollado independientemente. El lenguaje JavaScript puede interactuar con el código HTML, permitiendo a los programadores web utilizar contenido dinámico. JavaScript es de código abierto, por lo que cualquier persona puede utilizarlo sin comprar una licencia. (www.developer.mozilla.org, 2009)

1.8.3 Lenguaje de Marcas de Hipertexto

HTML, siglas de HyperText Markup Language (Lenguaje de Marcas de Hipertexto) es el lenguaje con el que se definen las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir

el texto y otros elementos que compondrán una página web.

1.8.4 Lenguaje de marcas extensible

Son las siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). XML es una versión de SGML, diseñado especialmente para los documentos de la web. Permite que los diseñadores creen sus propias etiquetas, permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y organizaciones. (www.masadelante.com, 2009).

1.9 Framework

1.9.1 ZendFramework 1.3.7

Se trata de un marco de trabajo para desarrollo de aplicaciones y servicios Web con PHP, te brinda soluciones para construir sitios web modernos, robustos y seguros. Además, es código abierto y trabaja con PHP 5. Contiene implementación completa del patrón arquitectónico MVC. Este framework (marco de trabajo) está formado por una serie de métodos estáticos y componentes. Los componentes son varios y variados y aunque alguno es posible que no lo usemos nunca hay otras que puede que las usemos hasta la saciedad. (www.zend.com, 2009).

1.9.2 Doctrine Framework 1.2.1

Es un marco de trabajo que mapea objetos relacionales (ORM por sus siglas en inglés) para PHP con una potente capa de abstracción de bases de datos (dbal por sus siglas en inglés). Uno de sus principales características es la opción de escribir las consultas de base de datos Orientada a Objeto en un lenguaje SQL llamada Doctrine Query Language (DQL por sus siglas en inglés). Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (www.doctrine-project.org, 2009).

1.9.3 Extjs 2.2

Es un framework para construir Aplicaciones de Internet Enriquecidas (RIA por sus siglas en inglés). Es basado en librerías JavaScript de código abierto, ligera y de alto rendimiento, compatible con la mayoría de los navegadores que permiten crear páginas e interfaces webs dinámicas. El mismo incluye tecnologías como Ajax, DHTML, XML, XSLT, JSON, CSS y DOM. Tiene incluidos la mayoría de los

controles de los formularios Web incluyendo tablas para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. (www.extjs.es, 2009)

1.10 Patrón Inversión de control

Inversión de control (Inversion of Control, IoC por sus siglas en inglés) es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. El flujo habitual se da cuando es el código del usuario quien invoca a un procedimiento de una librería. La inversión de control sucede cuando es la librería la que invoca el código del usuario. Típicamente sucede cuando la librería es la que implementa las estructuras de alto nivel y es el código del usuario el que implementa las tareas de bajo nivel. (Java Injection Framework, 2009).

1.11 Conclusiones parciales

En el capítulo se mostró los conceptos necesarios e importantes para comprender el software y decidir los lenguajes y la tecnología a utilizar para desarrollar el módulo de Estructura y Composición. Se utilizará los lenguajes de programación PHP, JavaScript, lenguaje extensible XML para la configuración y los servicios mediante el IoC, las bases de datos en Postgres SQL y la ayuda de los frameworks. Se trabajará sobre un estilo arquitectónico basado en componentes, compuesta por patrones arquitectónicos (MVC) y de diseños. La herramienta para desarrollar; el Netbeans 6.5, para modelar el Visual Paradigm, para bases de datos pgAdmin III y el EMS PostgreSQL por otra parte el navegador web Mozilla con su plugin firebug para la visualización de las interfaces.

CAPÍTULO II: DESCRIPCIÓN DE LA IMPLEMENTACIÓN DEL SISTEMA

2.1 Introducción

En este capítulo se realiza una validación al análisis y diseño realizado, se describe la implementación del sistema desarrollada a partir del diseño evaluado. Se hace énfasis en los metadatos, utilizados como solución del componente para lograr una mayor dinámica en el mismo.

2.2 Requisitos del Software

El ERP cubano además de gestionar los procesos de las entidades debe brindar la posibilidad de ser multientidad. El subsistema Estructura y Composición se encarga de brindarle esa característica al definir la estructura funcional de la entidad que se estarán gestionando, también especifica a quien se subordina dichas entidades y la estructura de las mismas. El módulo tiene tres responsabilidades fundamentales:

- ✓ Brindar la posibilidad al usuario que defina la estructura organizativa en la cual se va a ubicar su entidad y la estructura dentro de dicha entidad, así como permitir que se pueda especificar los cargos por la cuales van a estar compuestas las diferentes áreas dentro de las unidades.
- ✓ Brindarle servicios al resto de los módulos presentes en el Sistema Cedrux donde la mayoría de estos necesitan algún servicio de Estructura y Composición.
- ✓ Ser configurable para que los usuarios no estén obligados a usar los mismos conceptos estructurales.

Para cumplir las responsabilidades mencionadas se propusieron los siguientes requisitos a implementar.

Tabla 2.1 Listado de los requisitos funcionales.

Definición de niveles estructurales	
Definir Niveles Estructurales	Adicionar nivel estructural
	Modificar nivel estructural
	Eliminar nivel estructural
	Adicionar relación
	Eliminar relación
Gestionar Campos	Adicionar campo

	Modificar campo
	Eliminar campo
Gestión de estructuras	
Gestionar Nivel 1	Adicionar Nivel 1
	Modificar Nivel 1
	Eliminar Nivel 1
Gestionar Agrupación	Adicionar Agrupación
	Modificar Agrupación
	Eliminar Agrupación
Gestionar Entidad	Adicionar Entidad
	Modificar Entidad
	Eliminar Entidad
Gestionar Unidad	Adicionar Unidad
	Modificar Unidad
	Eliminar Unidad
Gestionar Relaciones de Estructuras	Adicionar estructura a la agrupación lógica
	Eliminar estructura de la agrupación lógica
Gestión de Estructuras Internas	
Gestionar Área	Adicionar Área
	Modificar Área
	Eliminar Área
Gestionar Plantilla Civil	Adicionar Cargo civil al área
	Modificar Cargo civil al área
	Eliminar Cargo civil al área
Gestionar Plantilla Militar	Adicionar Cargo militar al área
	Modificar Cargo militar al área
	Eliminar Cargo militar al área
Gestionar Medios	Adicionar Medio
	Modificar Medio
	Eliminar Medio
Gestión de los Nomencladores	
Gestionar Órgano	Adicionar Órgano
	Modificar Órgano
	Eliminar Órgano
Gestionar Subcategoría	Adicionar Subcategoría
	Modificar Subcategoría
	Eliminar Subcategoría
Gestionar Nivel Jerárquico	Adicionar Nivel jerárquico
	Modificar Nivel jerárquico
	Eliminar Nivel jerárquico

Gestionar Tipo cifra	Adicionar Tipo cifra
	Modificar Tipo cifra
	Eliminar Tipo cifra
Gestionar Cargo civil	Adicionar Cargo civil
	Modificar Cargo civil
	Eliminar Cargo civil
Gestionar Calificador de cargos	Adicionar Calificador
	Modificar Calificador
	Eliminar Calificador
Gestionar Cargo militar	Adicionar Cargo militar
	Modificar Cargo militar
	Eliminar Cargo militar
Gestionar Responsabilidad	Adicionar Responsabilidades
	Modificar Responsabilidades
	Eliminar Responsabilidades
Gestionar Categoría ocupacional	Adicionar Categoría Ocupacional
	Modificar Categoría Ocupacional
	Eliminar Categoría Ocupacional
Gestionar Tipo de escala salarial	Adicionar Tipo de escala salarial
	Modificar Tipo de escala salarial
	Eliminar Tipo de escala salarial
Gestionar Grupo de complejidad	Adicionar Grupo de Complejidad
	Modificar Grupo de Complejidad
	Eliminar Grupo de Complejidad
Gestionar Escala Salarial	Adicionar Escala Salarial
	Modificar Escala Salarial
	Eliminar Escala Salarial
Gestionar Medios	Adicionar Medios
	Modificar Medios
	Eliminar Medios
Gestionar Nivel de utilización del cargo	Adicionar Nivel de utilización
	Modificar Nivel de utilización
	Eliminar Nivel de utilización
Gestionar Preparación militar	Adicionar Preparación militar
	Modificar Preparación militar
	Eliminar Preparación militar
Gestionar Agrupación Lógica	Adicionar Agrupación lógica
	Modificar Agrupación lógica
	Eliminar Agrupación lógica
Gestionar Grado militar	Adicionar Grado militar
	Modificar Grado milita
	Eliminar Grado militar

Gestionar Tipo de calificador	Adicionar Tipo de calificador
	Modificar Tipo de calificador
	Eliminar Tipo de calificador
Gestionar Tipo de plantilla	Adicionar Tipo de plantilla
	Modificar Tipo de plantilla
	Eliminar Tipo de plantilla
Reportes	
Cantidad de Entidades y Unidades por Agrupaciones de un Nivel 1	
Detalles del Calificador de Cargos	
Resumen de Plantilla de Cargos	
Relación de Entidades por Agrupación	
Relación de la Localización de las Entidades por Nivel Estructural	
Relación de la Localización de las Unidades por Entidad	
Relación de Niveles 1 según su Clasificación	
Relación de Registros de las Entidades por Agrupación	
Resumen de Cargos por Áreas y Categoría Ocupacional	
Resumen de Cargos por Grupos de Complejidad y Categoría Ocupacional	
Resumen de Entidades por Categoría	
Resumen de Entidades por Agrupación según su Clasificación	

2.3 Validación del análisis y diseño

Del diseño propuesto se pudo extraer las clases fundamentales que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, logrando de esta manera adentrarse en las especificidades de los requisitos no funcionales, las restricciones de los lenguajes de programación, las características de la programación en capas, tecnologías de desarrollo y las ventajas que da la multiplataforma. La obtención del diseño, resultó de gran importancia, pues posibilitó crear una entrada apropiada y un punto de partida para las actividades de implementación.

El diseño propuesto fue creado con los siguientes patrones, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, permitiendo llevar a cabo la implementación del subsistema bajo patrones como el MVC (Modelo Vista Controlador) y los GRASP.

Entre los patrones usados en la solución se encuentran:

- ✓ **MVC**: El patrón MVC realiza un diseño que desacopla la vista del modelo, con la finalidad de mejorar la reusabilidad, permitiendo esto que las modificaciones realizadas en las vistas influyan

en menor medida en la lógica de negocio o de datos, el cual dicho patrón se ve evidenciado en el diagrama de clases del diseño definir estructura. Anexo 8.

- ✓ **Creador:** El patrón Creador se encarga de identificar la clase responsable de la creación de nuevos objetos. Presente en todos los diagramas de clase porque las clases controladoras crean diferentes objetos del modelo según los datos que necesiten. Anexo 7, Anexo 9, Anexo 8.
- ✓ **Alta cohesión:** El patrón Alta Cohesión indica que la información que almacena una clase debe ser coherente, de manera que todos sus métodos tengan un comportamiento bien definido. Es utilizado para la construcción del subsistema debido a que cada clase implementa las funcionalidades que le son correspondidas. Anexo 7.
- ✓ **Bajo acoplamiento:** El patrón Bajo Acoplamiento su objetivo en el subsistema es tratar de mantener las clases lo menos ligadas entre sí, de tal forma que en caso de producirse una modificación en alguna de ellas se tenga la mínima repercusión posible en el resto, potenciando la reutilización y disminuyendo sus dependencias. Anexo 7.
- ✓ **Experto:** El patrón Experto es el encargado de asignar una responsabilidad al experto en información. En el Anexo 6 se evidencia porque la clase controladora delega la responsabilidad a las clases del modelo de realizar la acción, por ejemplo al insertar, las clases del modelo son las encargadas de realizar esta función porque conocen toda la información.
- ✓ **Controlador:** El patrón Controlador se encarga de gestionar los eventos generados en capas anteriores, se muestra en todos los diagramas de clases (Anexo 6, Anexo 7, Anexo 8, Anexo 9) donde las clases controladoras a partir de dichos eventos toman las decisiones apropiadas, invocando funcionalidades contenidas en capas más profundas como el acceso a datos. Su función es de mediador o intermediario, del controlador del negocio asociado que en este caso son las clases del modelo.

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

2.4 Vistas de la Arquitectura de software

El proyecto ERP Cuba propone las vistas arquitectónicas (Figura 2.1):

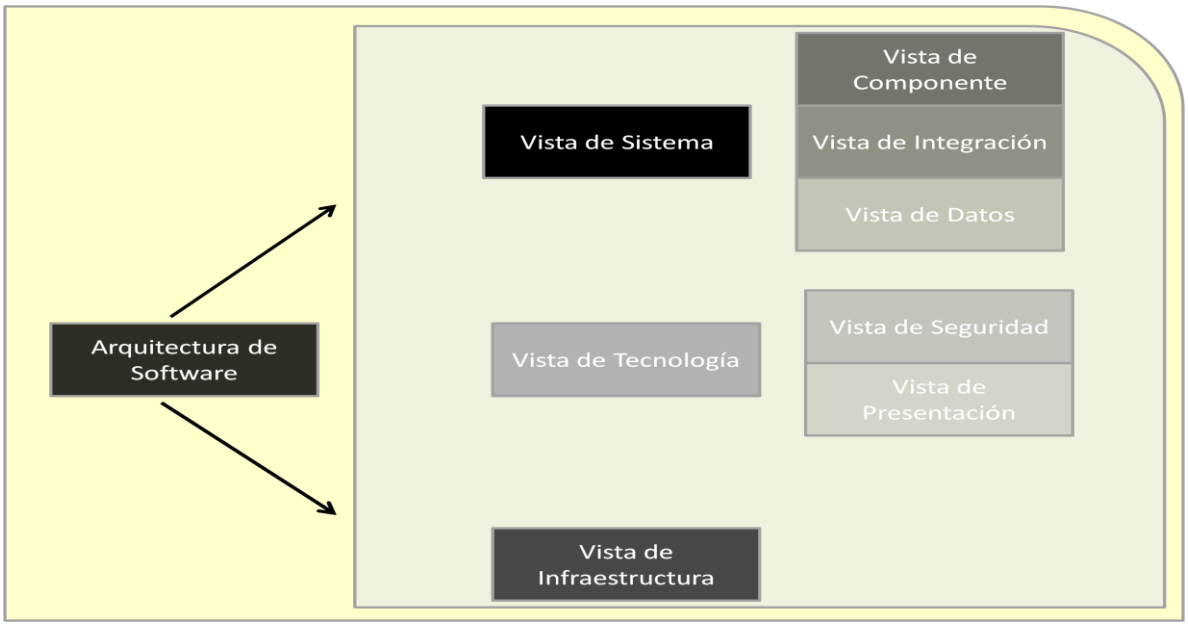


Figura 2.1 Vista de la arquitectura de software del proyecto ERP Cuba.

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

- ✓ Vista de Datos: Encargada de todas las definiciones a nivel de datos, de la integración de los distintos modelos, de los patrones, estándares y definiciones a este nivel.
- ✓ Vista de Integración: Encargada de los procesos de integración interna (entre componentes de un mismo proyecto) y externa (entre proyectos distintos), establece las definiciones, estándares, protocolos de comunicación y reglas de intercambio de información.
- ✓ Vista de Componentes: Encargada de las definiciones de los tipos de componentes posibles a definir en el proyecto, de la especificación de sus características, así como de la composición estructural interna de cada uno de estos componentes (vista vertical de la arquitectura). (Leyet Fernández, 2010)

Vista Tecnológica

Es la base del software, propicia los elementos necesarios para crear el producto, esta a su vez se

subdivide en dos vistas:

- ✓ Vista de seguridad: Chequea e implementa todos los aspectos relacionados con el acceso a la aplicación, la modificación, lectura o eliminación de la información, etc.
- ✓ Vista de presentación: Encargada de cómo luce el software, cuáles son los colores que lleva la aplicación, cómo son los botones, los vínculos y todos los elementos de alta significancia desde el punto de vista de la presentación. (Leyet Fernández, 2010)

Vista de Infraestructura

Es la encargada de la definición de la plataforma tecnológica a utilizar en la elaboración del producto, así como de la definición y disponibilidad de los distintos servicios telemáticos necesarios en la confección del producto, así como del diseño de los distintos escenarios de despliegue posibles. (Leyet Fernández, 2010)

Vista Arquitectura de Sistema

En el ámbito del proyecto ERP, la arquitectura de sistema es la vista encargada de: Proponer las partes del software: componentes, conectores, las restricciones y las configuraciones de estas partes, se subdivide en tres vistas fundamentales.

Por las características que presenta el dominio de los negocios a incidir con el desarrollo de la aplicación, y las tendencias y experiencias del desarrollo de otros sistemas ERP, se decide adoptar para el desarrollo horizontal del sistema el estilo arquitectónico orientado a componentes.

Por consiguiente, todas las funcionalidades levantadas y modeladas en las fases de negocio y requerimientos quedan expresadas o contenidas en al menos un componente, y las distintas interacciones entre estos componentes originan funcionalmente la existencia de subsistemas (Leyet Fernández, 2010).

De esta forma, el sistema quedaría constituido por un conjunto de componentes que responden a un conjunto de funcionalidades, un grupo de interacciones entre estos componentes respondiendo a las distintas integraciones y dependencias originadas en el negocio, estos componentes están agrupados en una unidad mayor denominada subsistemas (responden a las áreas de procesos más generales identificadas en el negocio). (Leyet Fernández, 2010).

Una representación gráfica del desarrollo de la vista de sistema quedaría expresada de la siguiente forma:

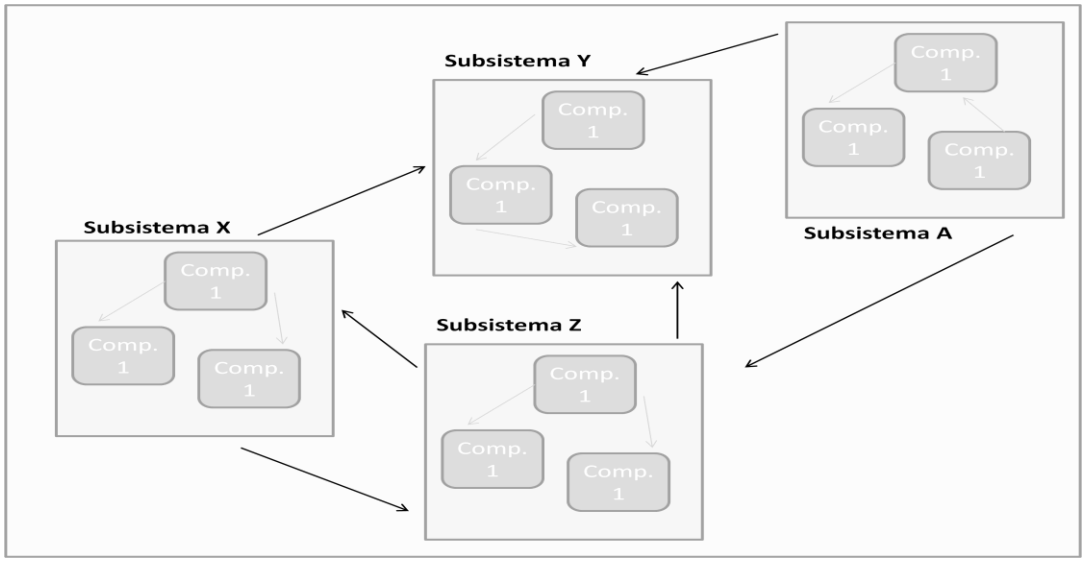


Figura 2.2 Representación gráfica de la vista de sistema.

Estructura y Composición es un subsistema del producto Cedrux dentro del cual se utiliza el estilo arquitectónico Modelo-Vista-Controlador (MVC), por lo que la estructura de carpeta del subsistema es de la siguiente forma (figura 2.3) ya que el MVC separa las clases por las funciones que realizan de modo tal que sea posible manejar dinámicamente la forma en que se procesan solicitudes y se gestiona la manera en que se muestran resultados al usuario final. Para dar cumplimiento a lo establecido por el proyecto se utilizó el framework de JavaScript ExtJS, con el mismo se construir Aplicaciones de Internet Enriquecidas por sus librerías y la tecnología AJAX que utiliza. El Zend-Framework otro marco de trabajo utilizado para el manejo de las clases controladoras y de negocios. Es muy utilizado en aplicaciones de PHP por la gran cantidad de funcionalidades que ofrece y por traer implementado el patrón MVC. En el caso del trabajo con los datos se utilizó un marco de trabajo para PHP llamado Doctrine que es muy potente para mapear las diferentes bases de datos como por ejemplo PostgresSQL, gestor de base de datos utilizado por el proyecto ERP Cuba.

Figura 2.3 Estructura de carpetas del subsistema.

La arquitectura del sistema de Estructura y Composición está basada en la gestión de las estructuras, para esto se desarrolló un componente dinámico basado en los metadatos (EAV) para establecer cualquier estructura dada (Figura 2.4).

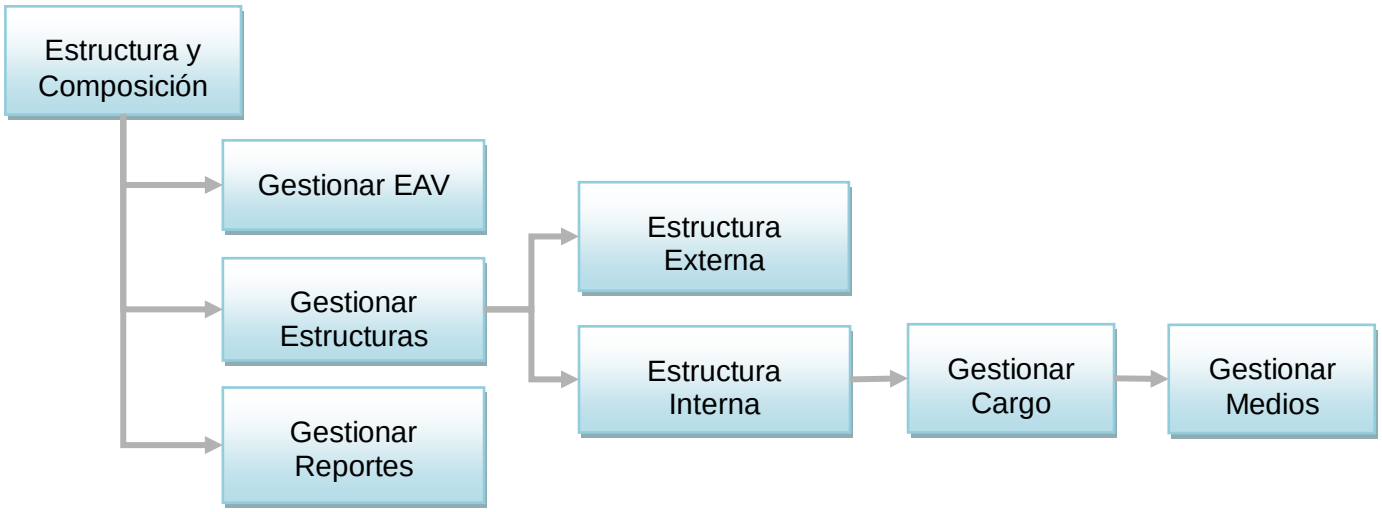


Figura 2.4 Gráfico de la arquitectura del sistema.

Las funcionalidades de Estructuras y Composición en su relación intercambian información. (Figura 2.5).

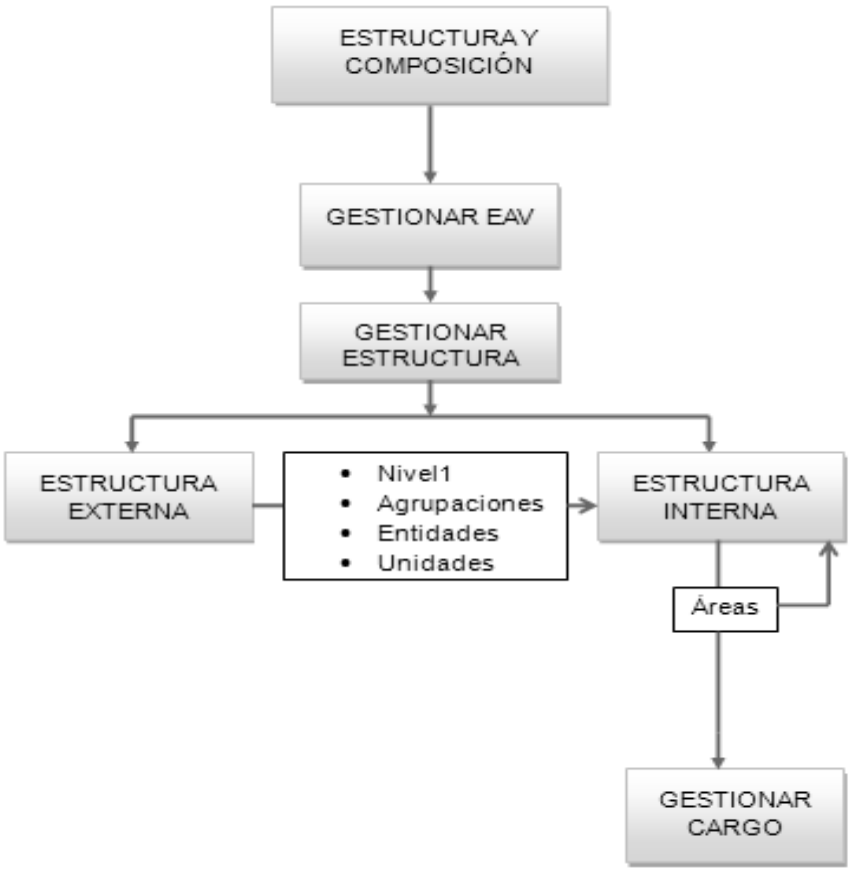


Figura 2.5 Intercambio de información entre las funcionalidades.

2.5 Estándar de código utilizado en la implementación

2.5.1 Nomenclatura de las clases

El nombre de las clases se escribe la primera letra con mayúsculas y las demás con minúsculas en caso de ser un nombre compuesto, el segundo nombre es seguido del primero y tiene la misma estructura. Ejemplo: Reporte, ReporteEstructura.

2.5.2 Nomenclatura de las clases según el tipo

- ✓ Las clases Controladoras que se encuentran dentro de la carpeta “controller”:

Las clases controladoras después de escribir el nombre según su nomenclatura se le añade la palabra “Controller”. Ejemplo: *EstructuraController*.

- ✓ Las clases del modelo que se encuentran dentro de la carpeta “bussines”

Las clases del modelo después de escribir el nombre según su nomenclatura se le añade la palabra “Model”. Ejemplo: *EstructuraModel*.

- ✓ Las clases Dominios que se encuentran dentro de la carpeta “domain”.

Las clases dominios son generadas por el marco de trabajo Doctrine y se llaman igual que la tabla de la base de datos. Ejemplo: *DatEstructura.*

- ✓ Las clases bases que se encuentran dentro de la carpeta “generate”

Las clases bases son generadas por el marco de trabajo Doctrine y delante del nombre de la clase se le añade “Base”. Ejemplo: *BaseDatEstructura*.

2.5.3 Nomenclatura de los métodos o funciones

Los nombres de los métodos de una clase comienzan con minúsculas, en caso de que sea compuesto seguido del primer nombre comienza el segundo con la primera letra en mayúscula. Deben describir el propósito del mismo. Ejemplos: insertar(), insertarEstructura().

Si es una función de una clase controladora después del nombre se le agrega la palabra “Action”. Ejemplo: insertarEstructuraAction().

2.5.4 Nomenclatura de las variables

El nombre a emplear para los atributos se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto seguido de la primara palabra se escribe la segunda con inicial mayúscula. Además, en caso de ser un objeto se comienza con: “_” y después se escribe el nombre. Ejemplo: *intAtributo , objMoneda, _estructura.*

2.6 Descripción de la implementación del subsistema Estructura y Composición

El subsistema Estructura y composición tiene como responsabilidad más importante ser configurable para que los usuarios no estén obligados a usar los mismos conceptos estructurales, por lo que el equipo de desarrollo se vio obligado a implementar un metadatos.

2.6.1 Definición de Metadatos

El término metadatos no tiene una definición única. Según la definición más difundida de los metadatos son datos estructurados sobre los datos. Debido a que muchas veces no se tiene en cuenta la diferencia

entre datos e informaciones, también hay muchas declaraciones como informaciones sobre datos, datos sobre informaciones e informaciones sobre informaciones. (Martínez Usero, 2006).

Otra clase de definiciones trata de precisar el término como descripciones estructuradas y opcionales que están disponibles de forma pública para ayudar a localizar objetos o datos estructurados y codificados que describen características de instancias conteniendo informaciones para ayudar a identificar, descubrir, valorar y administrar las instancias descritas. Esta clase de definiciones hace mayor hincapié en los metadatos en relación con la recuperación de información, y surgió de la crítica de que las declaraciones más simples son tan difusas y generales que dificultarán la tarea de acordarse de estándares, pero estas definiciones no son muy comunes. (Martínez Usero, 2006)

Los metadatos pueden describir colecciones de objetos y también los procesos en los que están involucrados, describiendo cada uno de los eventos, sus componentes y cada una de las restricciones que se les aplican. Los metadatos definen las relaciones entre los objetos, como las tuplas en una base de datos o clases en orientación a objetos, generando estructuras. (Martínez Usero, 2006).

2.6.2 ¿Por qué se utilizan los metadatos?

El subsistema Estructura y Composición como un módulo novedoso para un ERP tiene como objetivo gestionar la información de todos los niveles estructurales de Cuba. La organización de los niveles estructurales en Cuba tiene diferentes características por lo que el subsistema debe ser flexible, dinámico y configurable para cualquier concepto organizativo. En consecuencia, tiene que brindar la posibilidad de que el usuario defina los conceptos que se ajusten a las características de las empresas, no solo deben ser las entidades cubanas. Para resolver las propiedades mencionada se hace uso de los metadatos porque los mismos permiten generar distintos puntos de vista conceptuales para sus usuarios o sistemas, y liberan a estos últimos de tener conocimientos avanzados sobre la existencia o características del objeto que describen.

2.6.3 Descripción de la implementación del metadatos

En la implementación del metadato se utilizó el modelo de metadatos Entidad Atributo Valor (EAV) o Objeto Atributo Valor, también es nombrado matriz dispersa o esquema abierto. En la implementación del patrón Entidad Atributo Valor se utiliza una relación de cinco tablas (Figura 2.6) para guardar los conceptos estructurales que el usuario pretenda definir. Para comprender mejor como esto puede definir cualquier valor: se tiene la tabla (“nom_nomencladoreavestruc”) que guarda los nombres de los niveles

estructurales, en el caso de la tabla (“nom_filaestruc”) guarda los identificadores de los valores que van a tener los niveles, en la tabla (“nom_campoestruc”) se guarda el identificador, nombre y tipo de los campos que tiene los niveles estructurales (“nom_nomencladoreavestruc”) y la tabla (“nom_valorestruc”) que se crea por la relación mucho a mucho que existe entre (“nom_campoestruc”) y (“nom_filaestruc”) tiene el valor en específico con el identificador de (“nom_filaestruc”) y (“nom_campoestruc”). De esta forma, se logra que se defina un nivel estructural cualquiera con sus atributos, para establecer las relaciones entre los niveles estructurales se le añade una nueva tabla nombrada (“nom_aristaeav”), la que tiene dos identificadores (el origen y el destino) que son el identificador de la tabla (“nom_nomencladoreavestruc”) de esta forma se puede definir cual nivel contiene a otros.

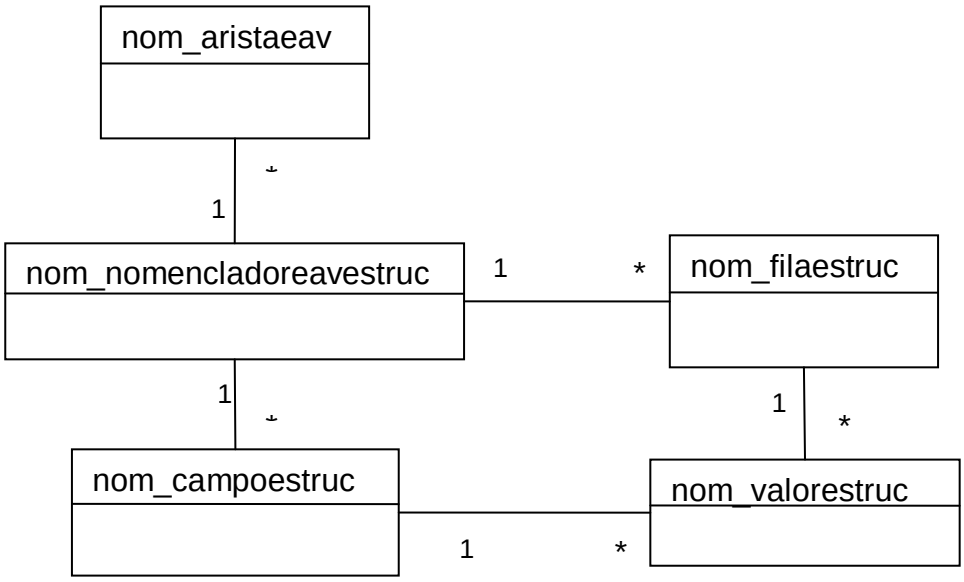


Figura 2.6 Relación de las tablas para el componente EAV.

Después de escoger la funcionalidad de Definir Nivel Estructural a continuación se muestra la interfaz que gestiona los niveles estructurales (Anexo 1) donde en el botón Adicionar adiciona los niveles estructurales acción que es controlada por la clase “EavController” con el método “insertarTablasAction”, utilizando un objeto la clase modelo “TablaModel” para el trabajo con los datos.

En el caso de modificar el nivel estructural se listan los niveles en un grid al seleccionar y presionar el botón modificar se cargan los valores del mismo y al modificarlos se gestionan los datos por la clase “EavController” en la acción “modificarTablaAction” y trabajando con el mismo objeto del modelo que la

función descrita anteriormente.

De la misma manera funciona el botón eliminar que al seleccionar el nivel estructural los elimina con la función “eliminarTablaAction” de la clase controladora “EavController” e instanciando al modelo “TablaModel”.

En este componente hay una funcionalidad importante que se llama establecer las relaciones entre los niveles estructurales que se logra con darle doble clic a un nivel estructural para activar el grid de Relaciones (Anexo 3) arrastrar con el Mouse hacia el grid o pinchar en el botón adicionar que se encuentra en el grid de Relaciones. La clase controladora encargada de está funcionalidad es “EavController” con la acción insertarConexionesAction y un objeto de “TablaModel”, de igual forma sucede el eliminar las conexiones listadas en el grid Relaciones, al seleccionar una y presionar el botón eliminar la misma se eliminan con el método “eliminarConexiocesAction”.

Otra propiedad que se realiza en el mismo es gestionar los campos de un nivel estructura (Anexo 4), donde se listan en un grid los campos que contiene un nivel estructural. Se utiliza la misma metodología que la descripción anterior lo que en esta ocasión la controladora “EavController” utiliza los métodos de “adicionarCamposAction”, “modificarCamposAction” y “eliminarCamposAction” instanciando al modelo “CampoModel”.

2.6.4 Descripción de la implementación de la funcionalidad Gestionar Nomencladores

Al escoger la funcionalidad de Gestionar nomencladores se muestra una interfaz (Anexo 2) que lista los nomencladores, al seleccionar uno se muestran sus datos en un grid. Con los botones Adicionar, Modificar, y Eliminar se puede gestionar los datos de los nomencladores controlados por la clase controladora “NomencladorController” instanciando los diferentes modelos de cada nomenclador. Aparece una funcionalidad para buscar por la denominación los datos de los nomencladores no sensibles a las mayúsculas y minúsculas. Para la implementación del diseño de las vistas se utilizó la técnica de programación orientada a objeto en Java Script aplicando con esto el patrón experto donde la clase gestionarnomenclador.js le asigna la responsabilidad a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Con esto se logra que se adicione, se modifique o se elimine un nomenclador sin que sufra cambios en el diseño de las vista solo con cambiar la clase responsable del nomenclador se soluciona el problema. Anexo 11.

2.6.5 Descripción de la implementación de la funcionalidad Gestionar Estructura

La funcionalidad de Gestionar estructuras tiene como precondición haber configurado los niveles estructurales en la funcionalidad de Definir Nivel Estructural y de adicionar datos a los nomencladores. Dentro de la funcionalidad se muestra un interfaz (Anexo 5) donde muestra en forma de árbol los niveles estructurales insertados en la aplicación además de su composición. Para gestionar los datos de algún nivel se puede realizar de dos formas; con clic derecho o con utilizar los botones Adicionar, Modificar y Eliminar las acciones son controladas por la clase controlador "EstructuraController" en los métodos "insertaValoresAction", "modificaValoresAction", "eliminaValoresAction", "buscarHijosAction", que instancia de la clase modelo "FilaModel", "EstructuraModel" y "EsturcturaopModel". Desplegando el árbol se puede ir gestionando todos los niveles, además de que cada elemento que se selecciona muestra los niveles contenidos en un grid como facilidad de búsqueda. Dentro de la composición de los niveles se pueden gestionar también los cargos y medios. Los cargos se gestionan de la misma forma que lo anterior; clic derecho o por los botones Adicionar, Modificar y Eliminar presentes en el panel y controladas por la clase "EstrcuraController" y con las acciones "adicionarCargoCivilAction", "modificarCargoMilitarAction", "eliminarCagoCivilAction", "eliminarCagoMilitarAction", "buscarDatosCargoAction" instanciando las clases del modelo "CargoModel", "CargocivilModel", "CargomtarModel".

2.6.6 Descripción de clases y operaciones del módulo Estructura y Composición

2.6.6.1 Clases Modelo

Tabla 2.2 Generalización de las clases del Modelo.

Nombre NombreClaseModel	
Tipo de clase: Model	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
NombreClaseModel	Constructor de la clase que instancia a las entidades.
insertar($parametros)	Encargada de los datos en la base de datos.
modificar($parametros)	Encargada de modificar los datos de un objeto.
eliminar($parametro)	Encargada de eliminar los datos de un objeto.
mostrar($limit,$start)	Encargada de mostrar los datos de un objeto de forma lineal.

2.6.6.2 Clases Controlador

Tabla 2.3 Nombre de la clase: Controlador EavController.

Nombre: EavController	
Tipo de clase: *Controladora*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertarcamposAction	Encargada de insertar los campos a un nivel estructural.
insertarTablasAction	Encargada de insertar un nivel estructural.
eliminartablaAction	Encargada de eliminar un nivel estructural.
eliminarcamposAction	Encargada de eliminar los campos de un nivel estructural.
mostrarcamposAction	Encargada de mostrar los campos de un nivel estructural.
mostrartablasAction	Encargada de mostrar los niveles estructurales.
modificarcamposAction	Encargada de actualizar los campos de los niveles estructurales.
modificarTablasAction	Encargada de actualizar los niveles estructurales.
buscarconexionesAction	Encargada de buscar las relaciones de un nivel estructural con otros niveles estructurales.
insertarconexionesAction	Encargada de insertar a un nivel estructural las relaciones otros niveles estructurales.
eliminarconexionesAction	Encargada de eliminar la relación de un nivel estructural con otros niveles estructurales.

Tabla 2.4 Nombre de la clase: Controlador EstructuraController.

Nombre: EstructuraController	
Tipo de clase: *Controladora*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
insertavaloresAction	Encargada de insertar los valores a un nivel estructural.
insertarcargocivilAction	Encargada de insertar un cargo civil.
modificarcargocivilAction	Encargada de actualizar los valores de un cargo civil.
eliminarvaloresAction	Encargada de eliminar los valores de un nivel estructural.
insertarcargomilitarAction	Encargada de insertar los cargos militares.
mostrardatoscargoAction	Encargada de mostrar los datos de los cargos.
modificarvaloresAction	Encargada de actualizar los valores de los niveles estructurales.

modificarcargomilitarAction	Encargada de actualizar los cargos militares.
buscarhijosAction	Encargada de buscar los hijos de un nivel estructural.
buscarhijosagrupAction	Encargada de buscar los hijos de una agrupación.
buscarcomposicionAction	Encargada de buscar las áreas que compone una entidad.
eliminarcargoAction	Encargada de eliminar los cargos.
eliminarestructuraagrupacionAction	Encargada de eliminar una estructura en una agrupación.
insertarestructuraagrupacionAction	Encargada de insertar una estructura en una agrupación.
mostraragrupacionAction	Encargada de mostrar las agrupaciones.
mostrarestructuraagrupacionAction	Encargada de mostrar las estructuras que compone una agrupación dada.
mostrardatostecnicaAction	Encargada de mostrar los medios existentes.
insertartecnicaAction	Encargada de insertar un medio.
eliminartecnicaAction	Encargada de eliminar un medio.
modificartecnicaAction	Encargada de actualizar un medio dado.
hijosdpaAction	Encargada de mostrar los hijos dado un lugar en la división política administrativa.
hijoespecialidadAction	Encargada de mostrar los hijos dado una especialidad dada.

Tabla 2.5 Nombre de la clase: Controlador NomencladorController.

Nombre: NomencladorController	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
mostrarcalificadorAction	Encargada de mostrar los calificadores.
insertarcalificadorAction	Encargada de insertar un calificador.
modificarcalificadorAction	Encargada de actualizar un calificador.
eliminarcalificadorAction	Encargada de eliminar un calificador.
mostrarprefijoAction	Encargada de mostrar los prefijos existentes.
insertarprefijoAction	Encargada de insertar un prefijo.
modificarprefijoAction	Encargada de actualizar los prefijos existentes.
eliminarprefijoAction	Encargada de eliminar los prefijos existentes.
mostrartipocalificadorAction	Encargada de mostrar los tipos de calificadores existentes.
insertartipocalificadorAction	Encargada de insertar un tipo de calificador.
modificartipocalificadorAction	Encargada de actualizar un tipo de calificador.
eliminartipocalificadorAction	Encargada de eliminar un tipo de calificador.

mostrargradomilitarAction	Encargada de mostrar los grados militares existentes.
insertargradomilitarAction	Encargada de insertar un grado militar.
modificargradomilitarAction	Encargada de actualizar un grado militar.
eliminargradomilitarAction	Encargada de eliminar un grado militar.
mostrarespAction	Encargada de mostrar una especialidad.
insertarorganosAction	Encargada de insertar un órgano.
mostrarorganosAction	Encargada de mostrar los órganos existentes.
modificarorganosAction	Encargada de actualizar un órgano.
eliminarorganosAction	Encargada de eliminar un órgano.
mostrarnvelestrAction	Encargada de mostrar los niveles estructurales existentes.
insertarnvelestrAction	Encargada de insertar un nivel estructural.
modificarnvelestrAction	Encargada de actualizar un nivel estructural.
eliminarnvelestrAction	Encargada de eliminar un nivel estructural.
insertarsbcategoriaAction	Encargada de insertar una sub-categoría.
modificarsbcategoriaAction	Encargada de actualizar una sub-categoría.
eliminarsbcategoriaAction	Encargada de eliminar una sub-categoría.
insertartipocifraAction	Encargada de insertar un tipo de cifra.
modificartipocifraAction	Encargada de actualizar un tipo de cifra.
eliminartipocifraAction	Encargada de eliminar un tipo de cifra.
mostrarcatgriacvilAction	Encargada de mostrar las categorías civiles existentes.
insertarcatgriacvilAction	Encargada de insertar una categoría civil.
modificarcatgriacvilAction	Encargada de actualizar una categoría civil.
eliminarcatgriacvilAction	Encargada de eliminar una categoría civil.
mostrarcatgriaocpnalAction	Encargada de mostrar una categoría ocupacional.
insertarcatgriaocpnalAction	Encargada de insertar una categoría ocupacional.
modificarcatgriaocpnalAction	Encargada de actualizar una categoría ocupacional.
eliminarcatgriaocpnalAction	Encargada de eliminar una categoría ocupacional.
mostrarntecnicaAction	Encargada de mostrar los medios técnicos existentes.
insertarntecnicaAction	Encargada de insertar un medio técnico.
modificarntecnicaAction	Encargada de actualizar un medio técnico.
eliminarntecnicaAction	Encargada de eliminar un medio técnico.
mostrarcargomtarAction	Encargada de mostrar los cargos militares.
insertarcargomtarAction	Encargada de insertar un cargo militar.
modificarcargomtarAction	Encargada de actualizar un cargo militar.
eliminarcargomtarAction	Encargada de eliminar un cargo militar.
mostrarcargocivilAction	Encargada de mostrar los cargos civiles existentes.
insertarcargocivilAction	Encargada de insertar un cargo civil.
modificarcargocivilAction	Encargada de actualizar un cargo civil.
eliminarcargocivilAction	Encargada de eliminar un cargo civil.
mostrarnagrupacionAction	Encargada de mostrar una agrupación.
insertarnagrupacionAction	Encargada de insertar una agrupación.
modificarnagrupacionAction	Encargada de actualizar una agrupación.
eliminarnagrupacionAction	Encargada de eliminar una agrupación.

2.7 Integración con los módulos del producto Cedrux

Una de las particularidades que debe cumplir el Sistema ERP Cedrux es multientidad por lo que debe existir un módulo que gestiones todas las características de las entidades cubanas así como la estructura organizativa del país. Por lo mencionado anteriormente se crea el subsistema Estructura y Composición con la responsabilidad de brindarles al resto de los subsistemas información que estos necesiten de las entidades. Una de las precondiciones de Cedrux al instalarse en una entidad es configurar el subsistema Estructura y Composición porque para que un subsistema gestione procesos de una entidad primero se debió crear la entidad en el modulo Estructura y Composición. Todo lo planteado se evidencia en la matriz de integración de los componentes donde se muestran los contratos que consume y brinda Estructura y Composición. Anexo 10

2.8 Conclusiones parciales

Con la realización de este capítulo se arribó a la conclusión de que la obtención del diseño propuesto resultó ser muy importante; pues permitió adquirir una comprensión de todo lo relacionado con las especificidades de los requisitos no funcionales, las restricciones de los lenguajes de programación, las características de la programación en capas, tecnologías de desarrollo y las ventajas que da la multiplataforma.

Se realiza un análisis de las vistas de la arquitectura de software propuesta por el proyecto ERP Cuba y de la arquitectura del sistema del subsistema Estructura y Composición, logrando de esta manera entender el funcionamiento del módulo para realizar una implementación robusta. Se ofrecen estándares de codificación, por los que se rigen los programadores del proyecto, logrando de esta forma hacer más legible el código fuente, así como lograr un mejor entendimiento del mismo, queda de parte de quienes la extiendan tratar de mantener un equilibrio, permitiendo una codificación homogénea y evitar las redundancias.

Igualmente se proporcionó una breve descripción de la implementación del subsistema Estructura y Composición donde en la cual se describió la responsabilidad más importante del módulo (ser configurable) para que los usuarios no estén obligados a usar los mismos conceptos estructurales. Por lo que también se abordó sobre el metadatos implementado y sus características. Otra responsabilidad del subsistema que se detalló en el presente capítulo fue brindar servicios al resto de los módulos presentes en el Sistema Cedrux donde la mayoría de estos necesitan algún servicio de Estructura y Composición.

CAPÍTULO III: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

En el presente capítulo definen las pruebas que se le realizan a los software como son pruebas de caja negra y pruebas de caja blanca. Se describen las pruebas de caja blanca realizadas; como son las pruebas del método del caminos básico y las desarrolladas con un software automatizado nombrado JMeter. Se aplicaron técnicas de medición o métricas para conocer como se afectan los valores de reutilización, acoplamiento, responsabilidad de las clases y complejidad de la implementación.

3.2 Prueba de software

Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón, se debe definir en el proceso de la ingeniería del software una plantilla para las pruebas del software: un conjunto de pasos en los que podamos situar los métodos específicos de diseño de casos de prueba. (Pressman, 2002).

El objetivo de la prueba de software es descubrir errores. Para conseguir este objetivo, se planifica y se ejecutan una serie de pasos; pruebas de unidad, de integración, de validación y del sistema. Las pruebas de unidad y de integración se centran en la verificación funcional de cada módulo y en la incorporación de los módulos en una estructura de programa. La prueba de validación demuestra el seguimiento de los requisitos del software y la prueba del sistema valida el software una vez que se ha incorporado en un sistema superior (Pressman, 2002).

3.3 Prueba de Unidad

Es la prueba enfocada a los elementos probables más pequeño del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. (Pressman, 2002).

3.4 Métodos de prueba de caja blanca

3.4.1 Prueba de caja blanca

La prueba de la **caja blanca** es un método de diseño de casos de prueba que usa la estructura de

control del diseño procedimental para obtener los casos de prueba. (Pressman, 2002). Según Pressman mediante las pruebas de caja blanca el ingeniero de software puede obtener casos de prueba que:

- ✓ Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

3.4.1.1 Prueba de condición

El método de la prueba de condiciones se centra en la prueba de cada una de las condiciones del programa. Las estrategias de prueba de condiciones tienen, generalmente, dos ventajas. La primera, que la cobertura de la prueba de una condición es sencilla. La segunda, que la cobertura de la prueba de las condiciones de un programa da una orientación para generar pruebas adicionales del programa. (Pressman, 2002).

3.4.1.2 Prueba de bucles

La prueba de bucles es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles. Se pueden definir cuatro clases diferentes de bucles (Beizer, 1990): bucles simples, bucles concatenados, bucles anidados y bucles no estructurados.

3.4.1.3 Prueba del camino básico

La prueba del camino básico es una técnica de prueba de caja blanca que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, 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 ejecuta cada sentencia al menos una vez. (Pressman, 2002).

3.5 Aplicación de la prueba de caja blanca

Para realizar pruebas de caja blanca se utilizó un software nombrado JMeter que es extensible y ofrece la posibilidad de que los propios usuarios desarrollen en Java un "Controller" a su medida, cumplimiento de Interfaz Java y el Jar correspondiente al desarrollo en el directorio "lib" de JMeter, con el mismo se midió el rendimiento que es uno de los indicadores más importantes en el desempeño de una aplicación. El JMeter permite realizar pruebas distribuidas en distintos ordenadores que actuarán como clientes, se pueden realizar pruebas de estrés y se puede obtener muchas otras estadísticas del software como por ejemplo la mediana, el muestreo, por ciento de errores, entre otras.

3.5.1 Caso de prueba para probarlo en el JMeter

Descripción: Se accede a la aplicación Cedrux por el portal hasta abrir el módulo de Estructura y Composición y probar las funcionalidades de Definir Nivel estructural, Gestionar estructuras, y Gestionar nomencladores.

Condición de ejecución:

- ✓ Deben estar corriendo los servidores de web y base de datos.
- ✓ Características de los servidores debe ser 1Gb de RAM.
- ✓ Características de la red 100 Mbps.

Resultado de la prueba.

Tabla 3.1 Tabla de resultado de las pruebas de rendimiento.

ID del escenario	Escenarios de la sección	Descripción	Resultado esperado	Resultado de la prueba
EC 1: Definir Nivel estructural	EC 1.1 Adicionar Nivel estructural	En esta página se adicionan los niveles estructurales que se definan.	Respondido a una velocidad menor de 25 seg	18,8 seg
	EC 1.2 Modificar Nivel estructural	En esta página se modifican los niveles estructurales que se definan.	Respondido a una velocidad menor de 25 seg	18,4 seg
	EC 1.3 Eliminar Nivel estructural	En esta página se eliminan los niveles estructurales que se definan.	Respondido a una velocidad menor de 25 seg	23,3 seg
EC 2 Gestionar Estructuras	EC 2.1 Adicionar estructura	En esta página se adicionan las estructuras a los niveles estructurales definidos.	Respondido a una velocidad menor de 25 seg	15,5 seg

	EC 2.2 Modificar estructura	En esta página se modifican las estructuras a los niveles estructurales definidos.	Respondido a una velocidad menor de 25 seg	21,1 seg
	EC 2.3 Eliminar estructura	En esta página se eliminan las estructuras a los niveles estructurales definidos.	Respondido a una velocidad menor de 25 seg	20,0 seg
EC 3 Gestionar Nomencladores	EC 3.1 Adicionar nomenclador	En esta página se adicionan los nomencladores a utilizar por las estructuras definidas.	Respondido a una velocidad menor de 25 seg	18,5 seg
	EC 3.2 Modificar nomenclador	En esta página se modifican los nomencladores a utilizar por las estructuras definidas.	Respondido a una velocidad menor de 25 seg	12,2 seg
	EC 3.3 Eliminar nomenclador	En esta página se eliminan los nomencladores a utilizar por las estructuras definidas.	Respondido a una velocidad menor de 25 seg	16,6 seg

Según las pruebas realizadas se ha demostrado que la aplicación con los requerimientos que necesita está en perfectas condiciones para resolver el objetivo con el cual fue desarrollada. Las pruebas de rendimiento realizadas con el JMeter ayudaron a conocer la velocidad y la eficiencia de la aplicación. Se conoce que la aplicación responde correctamente en menos de 25 segundos en todas las peticiones que realiza al servidor. Este es un indicador muy importante para el software porque es lo que hoy en la actualidad más le importa al usuario. Los usuarios necesitan un software eficiente y además rápido. El trabajo con este software también validó que no existiera errores en el código de la aplicación ya que brinda el dato de las peticiones realizada que respondieron con errores.

3.5.2 Aplicación del método el Camino Básico

Otra forma de realizar las pruebas de caja blanca es utilizando los métodos mencionados en el **epígrafe 3.4**. Aplicando la prueba del camino básico, a continuación se enumera las sentencias de código del procedimiento a probar (Figura 3.1).

Luego de haber construido el grafo (Figura 3.2) se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas siguientes:

Fórmulas para calcular complejidad ciclomática:

1. V (G) = (A – N) + 2.

2. V (G) = P + 1.

3. V (G) = R.

```
function hijosdpaAction()
{
 $idpadre       =   $this->_request->getPost('node');//1
 $obj  = new ZendExt_Nomencladores_ADT();//1
 if($idpadre == 'iddpa')//2
 {
    $arreglo = $obj->getForest("nom_dpa"); //3
    for($i=0 ;$i<count($arreglo);$i++)//4
    {
     $leaf    = ($arreglo[$i]->_values['idtipodpa']==3) ? 0:1;//5
     $hijo[] = array('id'=>$arreglo[$i]->_id,'text'=>$arreglo[$i]->_values['denominacion'],'leaf'=>$leaf);//5
    }
    echo json_encode($hijo);//6
 }
else
 {
    $mostrar = $obj->getTree("nom_dpa",$idpadre,1); //7
    $arreglo =$mostrar->Childrens();//7
    if (count($arreglo)==0)//8
    {
        echo json_encode(array());//9
    }
    else
    {
        for($i=0 ;$i<count($arreglo);$i++)//10
        {
         $leaf    = ($arreglo[$i]->_values['idtipodpa']==3) ? 1:0;//11
         $hijo[] = array('id'=>$arreglo[$i]->_id,'text'=>$arreglo[$i]->_values['denominacion'],'leaf'=>$leaf);//11
            }
         echo json_encode($hijo);//12
        }//13
    }//14

}
```

Figura 3.1 Código de la función hijosdpaAction().

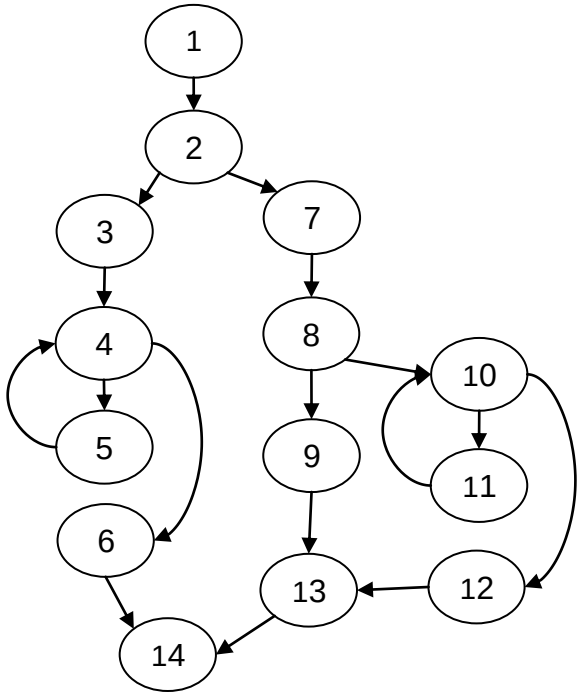


Figura 3.2 Grafo de flujo asociado al algoritmo hijosdpaAction().

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

Calculando mediante la fórmula 1:

V (G) = (17 - 14) + 2

V (G) = 5.

Calculando mediante la fórmula 2:

V (G) = 4+ 1

V (G) = 5.

Calculando mediante la fórmula 3:

V (G) = 5.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 5, lo que significa que existen cinco 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:

Tabla 3.2 Caminos básicos.

Camino básico #1	*1 - 2 - 3 - 4 - 5 - 4 - 6 - 14*
Camino básico #2	*1 - 2 - 3 - 4 -6 - 14*
Camino básico #3	*1 - 2 - 7 - 8 -9 - 13 - 14*
Camino básico #4	*1 - 2 - 7 - 8 -10 - 11 - 10 - 12 - 13 - 14*
Camino básico #5	*1 - 2 - 7 - 8 -10 - 12 - 13 - 14*

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

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

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

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

Entrada: Se muestran los parámetros que entran al procedimiento

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

- ✓ **Caso de prueba para el camino #1**

Descripción: se debe darle un valor a la variable idpadre.

Condición de ejecución: cuando la variable idpadre toma valor de iddpa, además el arreglo que contiene los datos del DPA es distinto de vacío.

Entrada: iddpa.

Resultado: se espera que devuelva un arreglo con los valores del DPA ajustado para mostrarlo en un árbol.

- ✓ **Caso de prueba para el camino #2**

Condición de ejecución: cuando la variable idpadre toma valor de iddpa, además el arreglo que contiene los datos de los DPA que son raíces está vacío.

Resultado: se espera que devuelva un arreglo vacío.

Después de realizar los casos de prueba se verifica que el algoritmo esté cumpliendo con las condiciones necesarias que se plantean.

3.6 Prueba de caja negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software (Pressman, 2002).

La prueba de caja negra intenta encontrar errores de las siguientes categorías.

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

- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y de terminación.

3.6.1 Métodos de prueba de caja negra

3.6.1.1 Pruebas basados en grafo

El primer paso en la prueba de caja negra es entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. (Beizer, 1995). Para llevar a cabo estos pasos, el ingeniero del software empieza creando un grafo de objetos importantes y sus relaciones y se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores (Pressman, 2002).

3.6.1.2 Partición equivalente

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores (por ejemplo, proceso incorrecto de todos los datos de carácter) que, de otro modo, requieran la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. (Pressman, 2002). Dicho de otra manera, la prueba del software empieza creando un grafo de objetos importantes y sus relaciones, y después diseñando una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir 10s errores

Si un conjunto de objetos puede unirse por medio de relaciones simétricas, transitivas y reflexivas, entonces existe una clase de equivalencia. (Beizer, 1995).

El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases. (Pressman, 2002).

3.6.1.3 Análisis del valor límite

Por razones que no están del todo claras, los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límite (AVL) como técnica de prueba. (Pressman, 2002).

El análisis de valores límite es una técnica de diseño de casos de prueba que complementa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el AVL lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el AVL obtiene casos de prueba también para el campo de salida. (Meyers, 1979).

3.7 Validación de la implementación

El desarrollo de software es algo muy complejo y la medida de su calidad real no es automatizable. Por esta razón la aplicación de métricas de calidad para evaluar el diseño orientado a objeto posee gran importancia. A continuación, presentaremos un estudio que brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software, siendo esto la principal razón de la concepción de las métricas (Pressman, 2002).

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 se tiene al implementar un diseño de clases determinado.
- ✓ *Reutilización:* Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- ✓ *Acoplamiento:* Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, 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. (Herrera, y otros, 2007)

3.7.1 Tamaño operacional de clase

Esta métrica está relacionada con el número de métodos asignados a una clase. Teniendo en cuenta los siguientes atributos:

Tabla 3.3 Tamaño operacional de clase (TOC).

Atributo que afecta	Modo en que lo afecta
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 3.4 Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Complejidad de implementación	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Reutilización	Baja	> 2* Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	< =Promedio.

Tabla 3.5 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

No	Clases	Procedimientos	Responsabilidad	Complejidad	Reutilización
1	CampoModel	11	Baja	Baja	Alta
2	DatagrupacionestModel	6	Baja	Baja	Alta
3	DatcargocivilModel	7	Baja	Baja	Alta
4	DatcargoModel	16	Baja	Baja	Alta
5	DatcargomtarModel	7	Baja	Baja	Alta
6	DominioModel	18	Media	Media	Media
7	EstructuraModel	38	Alta	Alta	Baja
8	EstructuraopModel	22	Media	Media	Media
9	FilaModel	8	Baja	Baja	Alta
10	NomagrupacionesModel	10	Baja	Baja	Alta
11	NomcalificadorModel	10	Baja	Baja	Alta
12	NomcargocivilModel	10	Baja	Baja	Alta
13	NomcargomilitarModel	10	Baja	Baja	Alta
14	NomcategcivilModel	10	Baja	Baja	Alta
15	NomcategoriaocupacionalModel	10	Baja	Baja	Alta
16	NomclasificacioncargoModel	9	Baja	Baja	Alta
17	NomescalasalarialModel	10	Baja	Baja	Alta
18	NomgradomilitarModel	10	Baja	Baja	Alta
19	NomgrupocompleModel	11	Baja	Baja	Alta
20	NommoduloModel	8	Baja	Baja	Alta
21	NomnivelestrModel	11	Baja	Baja	Alta
22	NomorganoModel	16	Baja	Baja	Alta
23	NomprepmilitarModel	10	Baja	Baja	Alta
24	NomsalarioModel	11	Baja	Baja	Alta
25	NomtipocalificadorModel	9	Baja	Baja	Alta

26	TablaModel	17	Media	Media	Media
27	ValordefectoModel	8	Baja	Baja	Alta
28	ValorModel	5	Baja	Baja	Alta
29	EavController	24	Media	Media	Media
30	EstructuraController	61	Alta	Alta	Baja
31	NomencladorController	93	Alta	Alta	Baja

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

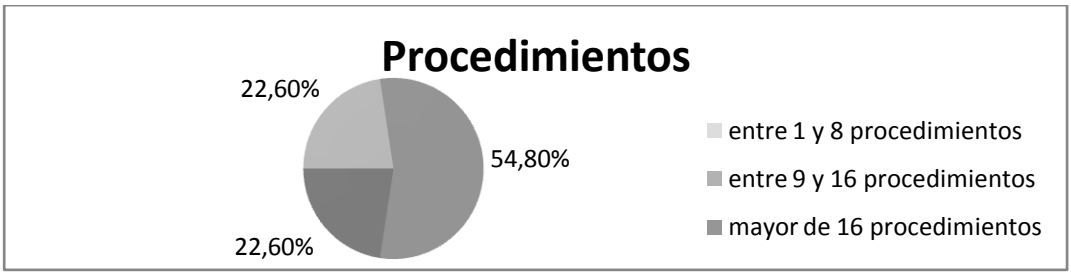


Figura 3.3 Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

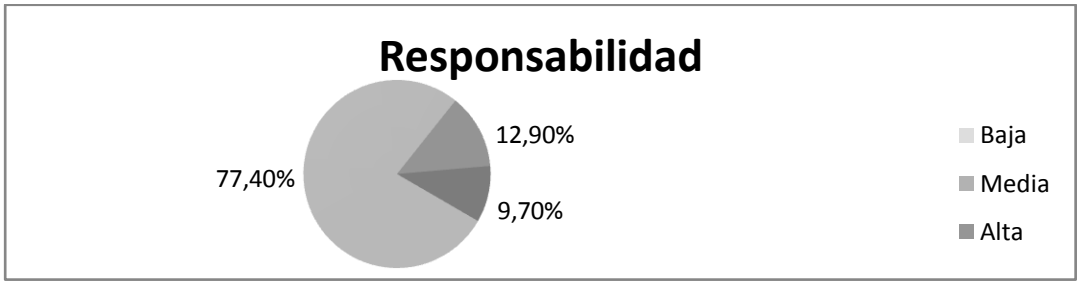


Figura 3.4 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

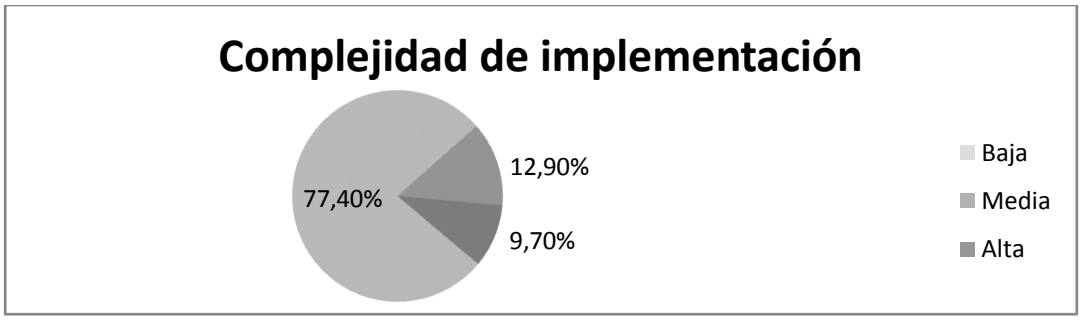


Figura 3.5 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

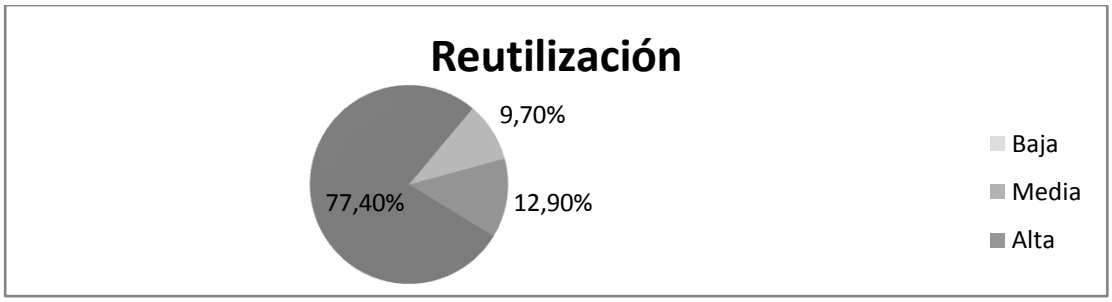


Figura 3.6 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC en los diferentes atributos estudiados, se puede concluir que el diseño del módulo Estructura y Composición tiene una buena calidad pudiéndose observar que el 77.4% de las clases posee menos cantidad de procedimientos que la mitad registrada en las mediciones. Igualmente el 77.4% de las clases posee evaluaciones positivas en los atributos Responsabilidad, Complejidad de Implementación y Reutilización.

3.7.2 Relaciones entre clases

Esta métrica esta dada por el número de relaciones de uso de una clase con otras. Teniendo en cuenta los siguientes atributos:

Tabla 3.6 Relaciones entre clases (RC).

Atributo que afecta	Modo en que lo afecta
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.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.

Tabla 3.7 Rango de valores para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

Atributo	Categoría	Criterio
Acoplamiento	Bajo	1
	Medio	2
	Alto	>2
Complejidad del mantenimiento	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Cantidad de pruebas unidad	Baja	< =Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Reutilización	Baja	> 2* Promedio.
	Media	Entre Promedio y 2* Promedio.
	Alta	< =Promedio.

Tabla 3.8 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).

No	Clases	Cantidad de relaciones de uso	Acoplamiento	Complejidad del Mantenimiento	Reutilización	Cantidad de pruebas
1	CampoModel	1	Bajo	Baja	Alta	Baja
2	DatagrupacionestModel	1	Bajo	Baja	Alta	Baja
3	DatcargocivilModel	1	Bajo	Baja	Alta	Baja
4	DatcargoModel	1	Bajo	Baja	Alta	Baja
5	DatcargomtarModel	1	Bajo	Baja	Alta	Baja
6	DominioModel	1	Bajo	Baja	Alta	Baja
7	EstructuraModel	1	Bajo	Baja	Alta	Baja
8	EstructuraopModel	1	Bajo	Baja	Alta	Baja
9	FilaModel	1	Bajo	Baja	Alta	Baja
10	NomagrupacionesModel	1	Bajo	Baja	Alta	Baja
11	NomcalificadorModel	1	Bajo	Baja	Alta	Baja
12	NomcargocivilModel	1	Bajo	Baja	Alta	Baja
13	NomcargomilitarModel	1	Bajo	Baja	Alta	Baja
14	NomcategcivilModel	1	Bajo	Baja	Alta	Baja
15	NomcategoriaocupacionalModel	1	Bajo	Baja	Alta	Baja
16	NomclasificacioncargoModel	1	Bajo	Baja	Alta	Baja
17	NomescalasalarialModel	1	Bajo	Baja	Alta	Baja
18	NomgradomilitarModel	1	Bajo	Baja	Alta	Baja
19	NomgrupocompleModel	1	Bajo	Baja	Alta	Baja
20	NommoduloModel	1	Bajo	Baja	Alta	Baja
21	NomnivelestrModel	1	Bajo	Baja	Alta	Baja
22	NomorganoModel	1	Bajo	Baja	Alta	Baja
23	NomprepmilitarModel	1	Bajo	Baja	Alta	Baja
24	NomsalarioModel	1	Bajo	Baja	Alta	Baja
25	NomtipocalificadorModel	1	Bajo	Baja	Alta	Baja
26	TablaModel	1	Bajo	Baja	Alta	Baja

27	ValordefectoModel	1	Bajo	Baja	Alta	Baja
28	ValorModel	1	Bajo	Baja	Alta	Baja
29	EavController	5	Alto	Alta	Baja	Alta
30	EstructuraController	13	Alto	Alta	Baja	Alta
31	NomencladorController	20	Alto	Alta	Baja	Alta

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

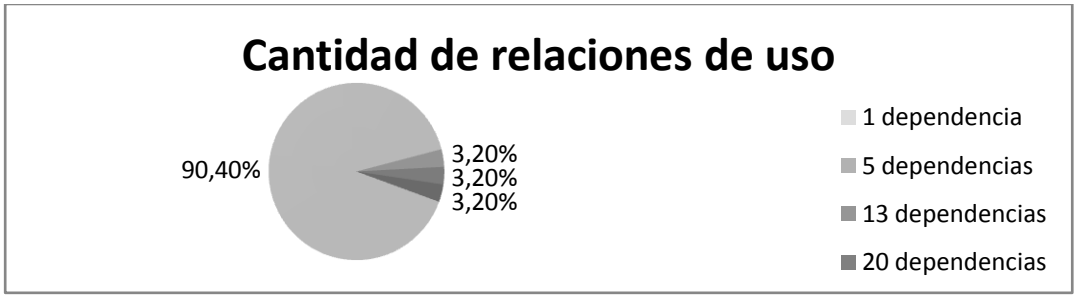


Figura 3.7 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

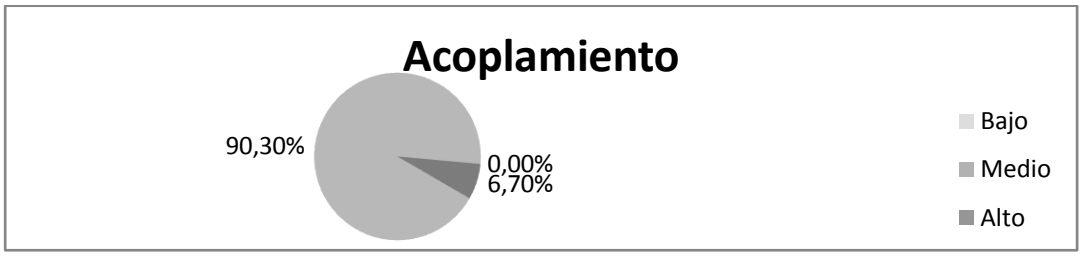


Figura 3.8 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

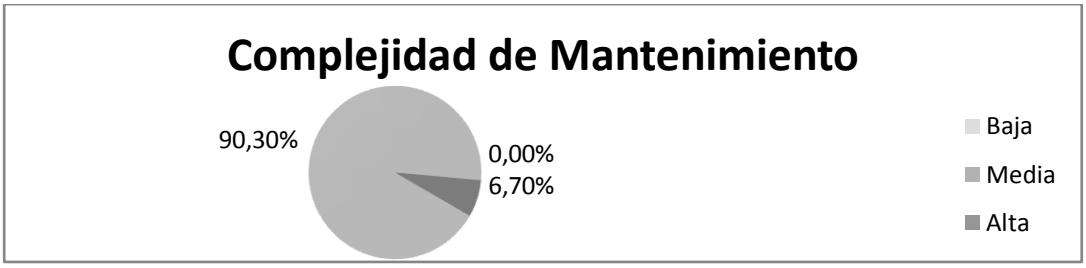


Figura 3.9 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

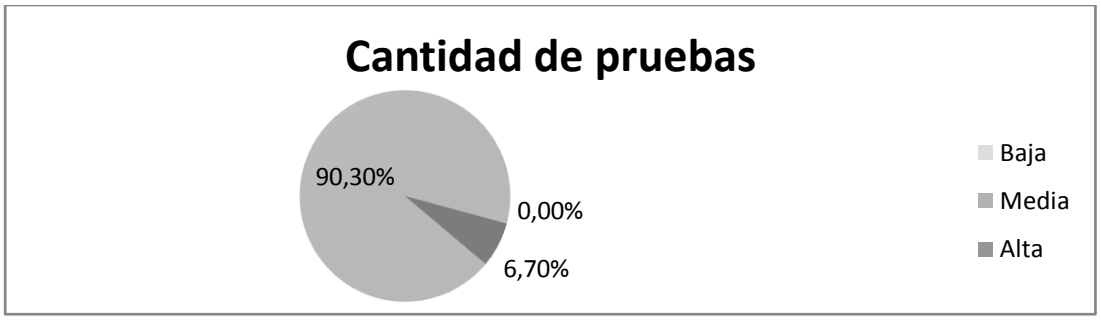


Figura 3.10 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

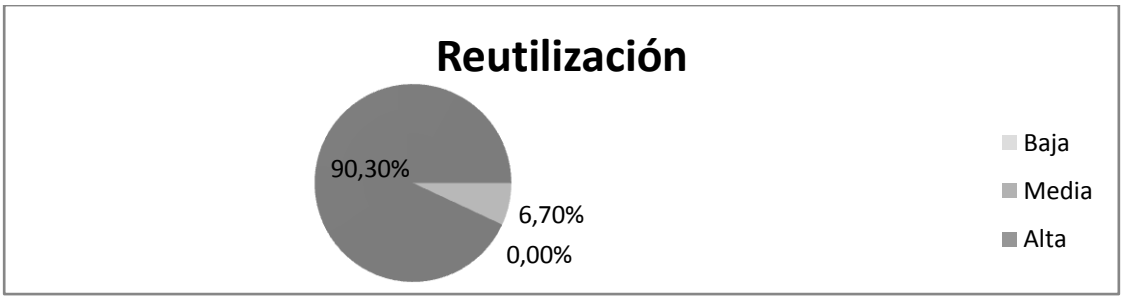


Figura 3.11 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del subsistema Estructura y Composición tiene una buena calidad pudiéndose observar que el 90.3% de las clases posee una simple dependencia de otras clases. Igualmente el 90.3% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento y Reutilización se comportan satisfactoriamente en el mismo valor de por ciento (90.3%) en las clases. Excepto en el caso de los resultados de la Cantidad de Pruebas fueros regulares en la mayor parte, solamente el 6.7 % se le realizaron pruebas de unidad alta.

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

Tabla 3.9 Umbrales.

Criterio	Evaluación
<=0	Mal
0< a >= 0.5	Regular
0.5 < a >= 1	Bien

Tabla 3.10 Resumen de los resultados.

Atributos	TOC	RC	Total
Complejidad de implementación	1	-	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad del mantenimiento	-	1	1
Cantidad de pruebas	-	0.5	0.5
Responsabilidad	1	-	1
Total			**1**

3.8 Conclusiones parciales

En el presente capítulo se analizaron diferentes aspectos como el significado de las pruebas de software, sus objetivos y alcance. Se realizó también una breve descripción de la prueba de unidad y de los diferentes métodos de pruebas de caja blanca y prueba de caja negra, haciendo énfasis en la prueba del camino básico y el cálculo de la complejidad ciclomática.

Al aplicar estas pruebas de unidad a un fragmento de código de la implementación, del cual se obtuvo una complejidad ciclomática de 5 unidades, dando lugar a 5 caminos básicos, comprobándose que eran los únicos por los que siempre transitaba el flujo, se diseñaron 2 casos de pruebas de los cuales se obtuvo los resultados esperados. Además, se ejecutó casos de pruebas de rendimiento en una aplicación llamada JMeter donde se adquirieron resultados satisfactorios.

Por último, se elaboraron instrumentos inspirados en las métricas para la calidad del diseño como Tamaño Operacional de Clase (TOC) y las Relaciones entre Clases (RC), evaluándose varios factores clave como la reutilización, complejidad de implementación, entre otros, de los cuales se obtuvo para las métricas TOC y RC una calidad aceptable.

Los resultados de las pruebas realizadas al software y la aplicación de las métricas en ambos casos descritos anteriormente dan lugar a que la implementación realizada presenta una calidad admisible, siendo positiva la validación de la solución propuesta.

CONCLUSIONES GENERALES

Al concluir el presente trabajo, se confirma la necesidad y la importancia de lograr el cumplimiento del objetivo general de implementar el subsistema Estructura y Composición en el Sistema ERP Cedrux. Ayudando de dicha manera a resolver de modo eficiente los problemas referentes al proceso de gestión de estructuras de cualquier entidad en Cuba. Y de manera general se puede concluir lo siguiente:

- ✓ Se realizó el diseño teórico donde se logró comprender la necesidad de desarrollar el subsistema Estructura y Composición, pues los sistemas de planificación de recursos empresariales que existen no resuelven el problema de la multientidad, y otros elementos que caracterizan a las entidades cubanas. Además, la determinación del marco teórico permitió estudiar y analizar el uso de las tecnologías y herramientas propuestas por la línea de Arquitectura lo cual posibilitó entender las causas de su elección, constituyendo estándares y políticas acordes a las necesidades del país de migrar a Software Libre.
- ✓ Se hizo un estudio y se comprobaron los requerimientos del software permitiendo demostrar que los requisitos funcionales fueran los que el cliente necesitaba. Asimismo, se comprendió y entendió la concepción de los mismos para la realización del subsistema Estructura y Composición.
- ✓ Se cumplió el objetivo propuesto de realizar la implementación del módulo Estructura y Composición, basado en los requisitos funcionales, permitiendo obtener como resultado la implementación de un subsistema confiable y eficiente.
- ✓ Se probó el subsistema donde se aplicaron las pruebas unitarias del camino básico y se utilizó un software llamado JMeter para las pruebas de caja blanca donde se proyectó resultados satisfactorios. También se realizó una validación de la implementación mediante la utilización de instrumentos de medición que se inspiraron en métricas para la calidad con el objetivo de comprobar la vialidad de la solución. Los resultados arrojados permitieron concluir que la implementación presentaba valores positivos en los indicadores de calidad evaluados. Esto apoya la afirmación de que la implementación desarrollada se puede considerar como aceptable, y el código puede ser reutilizado en futuras implementaciones.

RECOMENDACIONES

Con la realización del presente trabajo se recomienda:

- ✓ Continuar incorporando funcionalidades con el objetivo de lograr una mayor integración del módulo con el resto de los subsistemas del Sistema ERP Cedrux.
- ✓ Extender el uso del módulo Estructura y Composición a otras entidades del país como parte del Sistema ERP Cedrux.

BIBLIOGRAFÍA

Bass, L., Clements, P. and Kasman, R. 1998. *Software Architecture in Practice.* 1998.

Beizer, B. 1995. *Black-Bos Testing.* 1995.

—. 1990. *Software Testing Techniques.* 1990.

Burbeck, Steve. 1992. Application programming in Smalltalk-80: How to use Model-View-Controller (MVC). [Online] 1992. http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html.

CENTRO DE SOLUCIONES DE GESTIÓN. 2009. Ciclo de vida del Proyecto. 2009.

Equipo de producción. 2009. Modelo de Desarrollo orientado a componentes del proyecto ERP -CUBA. 2009.

Guía de PMBOK. 2004. Guía de los Fundamentos de la Dirección de Proyectos (Tercera edición) . 2004.

Herrera, Yzquierdo, et al. 2007. *El modelo de diseño del sistema HyperWeb. Módulos de Tratamiento Farmacológico y Configuración.* Ciudad de la Habana: Universidad de las Ciencias Informáticas : s.n., 2007.

http://www.osmosislatina.com/jmeter/basico.htm. 2009. [Online] 2009. http://www.osmosislatina.com/jmeter/basico.htm.

Ing. Henry Cruz Mulet, Ing. Nemury Silega Martinez, Ing. Donel Vázquez Zambrano. 2009. *Modelo de desarrollo orientado a componentes.* 2009.

Iribarne Martínez, Luis F. 2003. *Un Modelo de Mediación para el Desarrollo de Software basado en Componentes COTS.* 2003.

Java Injection Framework. 2009. Java Injection Framework. [Online] 2009. http://deveeel.files.wordpress.com/2008/09/google_guice.ppt.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* Mexico : PRENTICE HALL, 1999.

Leyet Fernández, Osmar. 2010. *DOCUMENTO DE DESCRIPCIÓN DE LA ARQUITECTURA DE SOFTWARE.* La Habana, Cuba : s.n., 2010.

Lic. Miguel P. Cabrera González, Msc. Guillermo Obregón Rodríguez, Msc. Margarita Cárdenas Negrin, Lic. Luis Mario Carralero Silva. 2004. *verSat sarasola.* 2004.

Martínez Usero, José Ángel. 2006. *El uso de metadatos para mejorar la interoperabilidad.* Madrid : s.n., 2006.

Meyers, G. 1979. *The Art of Software Testing.* 1979.

Microsoft, Patterns & Practices de. 2003. [Online] 2003. http://msdn.microsoft.com/practices/type/Patterns/Enterprise/DesMVC/.

Pressman. 2002. Ingieneria de software un enfoque práctico. Quinta edición. 2002.

Shaw, D. and Galan, M. 1996. *Software Architecture: Perspectives.* 1996.

Szyperski, C. 1998. Component Software. Beyond Object-Oriented Programming. 1998.

Visconti, Marcello and Astudillo, Hernán. 2004. Fundamento de Ingienería de Software. 2004.

www.apache.com. 2009. [Online] 2009. http://www.apache.com/.

www.developer.mozilla.org. 2009. [Online] 2009. https://developer.mozilla.org/en/About_JavaScript.

www.doctrine-project.org. 2009. Doctrine. [Online] 2009. http://www.doctrine-project.org.

www.extjs.es. 2009. Extjs. [Online] 2009. http://extjs.es.

www.guia-ubuntu.org. 2009. [Online] 2009. http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

www.masadelante.com. 2009. [Online] 2009. http://www.masadelante.com/faqs/xml.

www.mozilla-europe.org. 2009. [Online] 2009. http://www.mozilla-europe.org/es/firefox/features/#personalization.

www.netbeans.org. 2008. [Online] 2008. http://www.netbeans.org/.

www.openbravo.com. 2009. [Online] 2009.

www.openxpertya.org. 2009. [Online] 2009.

www.oracle.com. 2009. Sun oracle. [Online] 2009. http://www.oracle.com.

www.php.net. 2010. [Online] 2010. http://www.php.net.

www.postgresql.org. 2009. [Online] 2009. http://www.postgresql.org.

www.programacion.com. 2009. [Online] 2009. http://www.programacion.com/java/articulo/corej2ee_patterns/.

www.sap.com. 2009. [Online] 2009. http://www.sap.com/mexico/solutions/business-suite/erp/sistemas-erp.epx.

www.sqlmanager.net. 2009. [Online] 2009. http://sqlmanager.net/products/postgresql/manager.

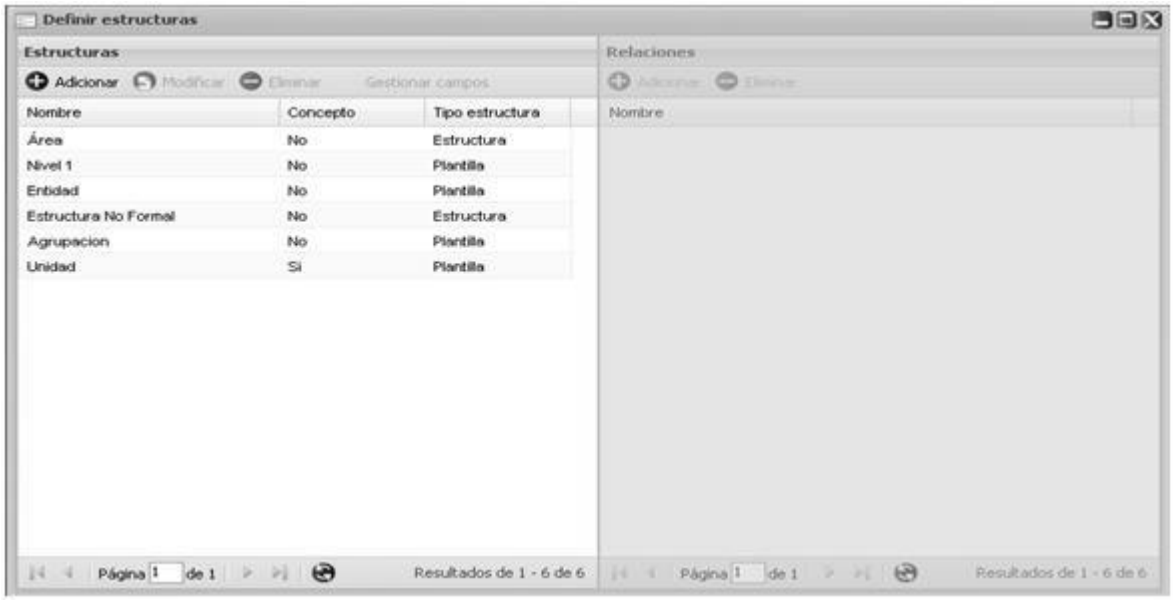
www.subversion.apache.org. 2010. [Online] 2010. http://subversion.apache.org/.

www.visual-paradigm.com. 2009. [Online] 2009. http://www.visual-paradigm.com/.

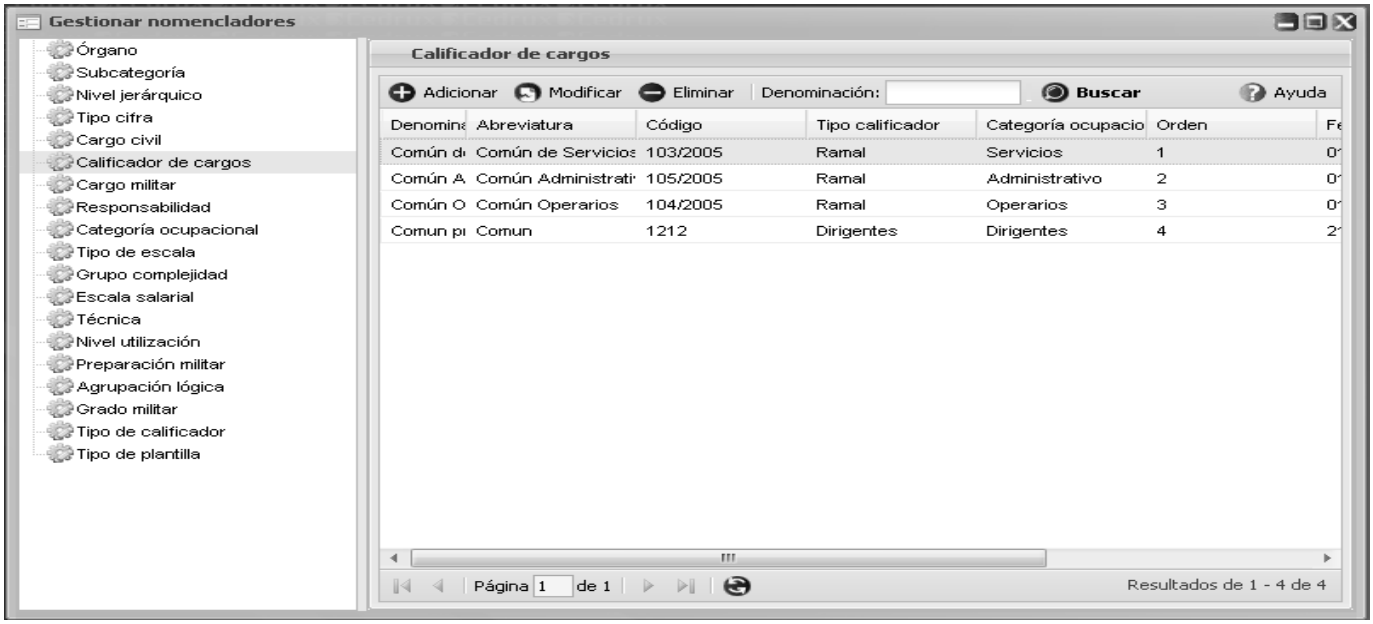
www.zend.com. 2009. www.zend.com. [Online] 2009. http://www.zend.com/en/community/php.

ANEXOS

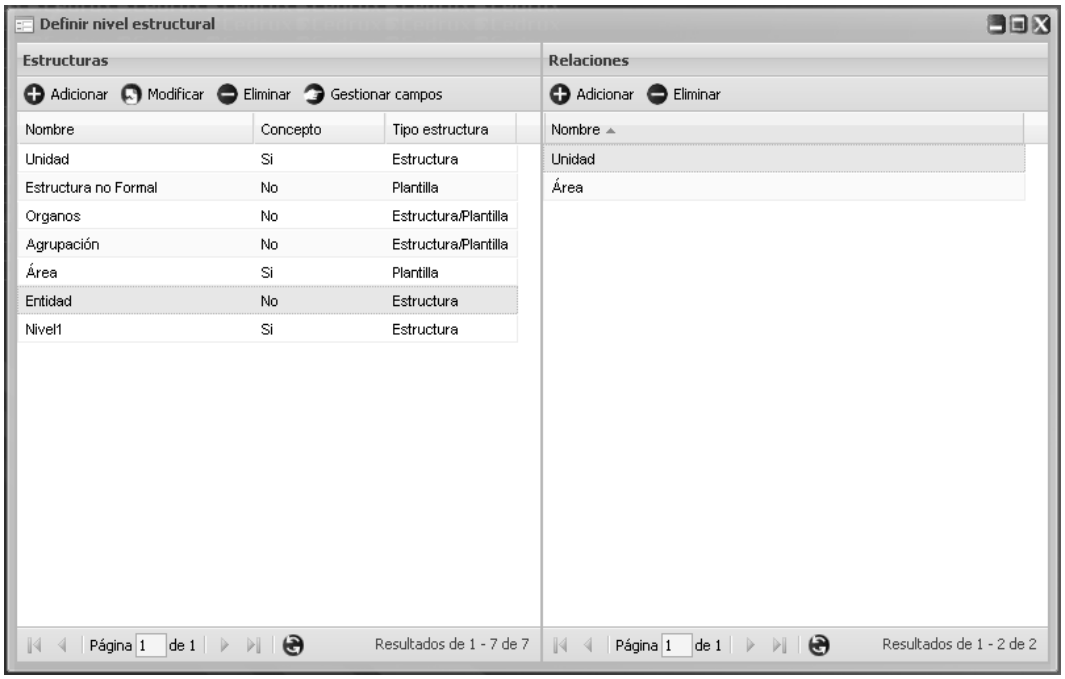
Anexo 1 Interfaz de la funcionalidad Definir nivel estructural.



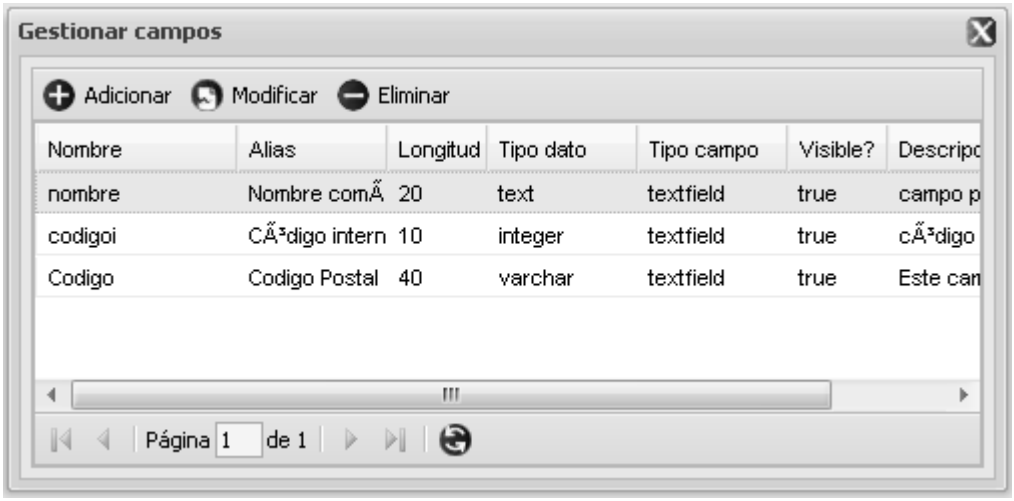
Anexo 2 Interfaz de la funcionalidad Gestionar nomencladores.



Anexo 3 Interfaz de la funcionalidad Definir nivel estructural en la funcionalidad de insertar las Relaciones.

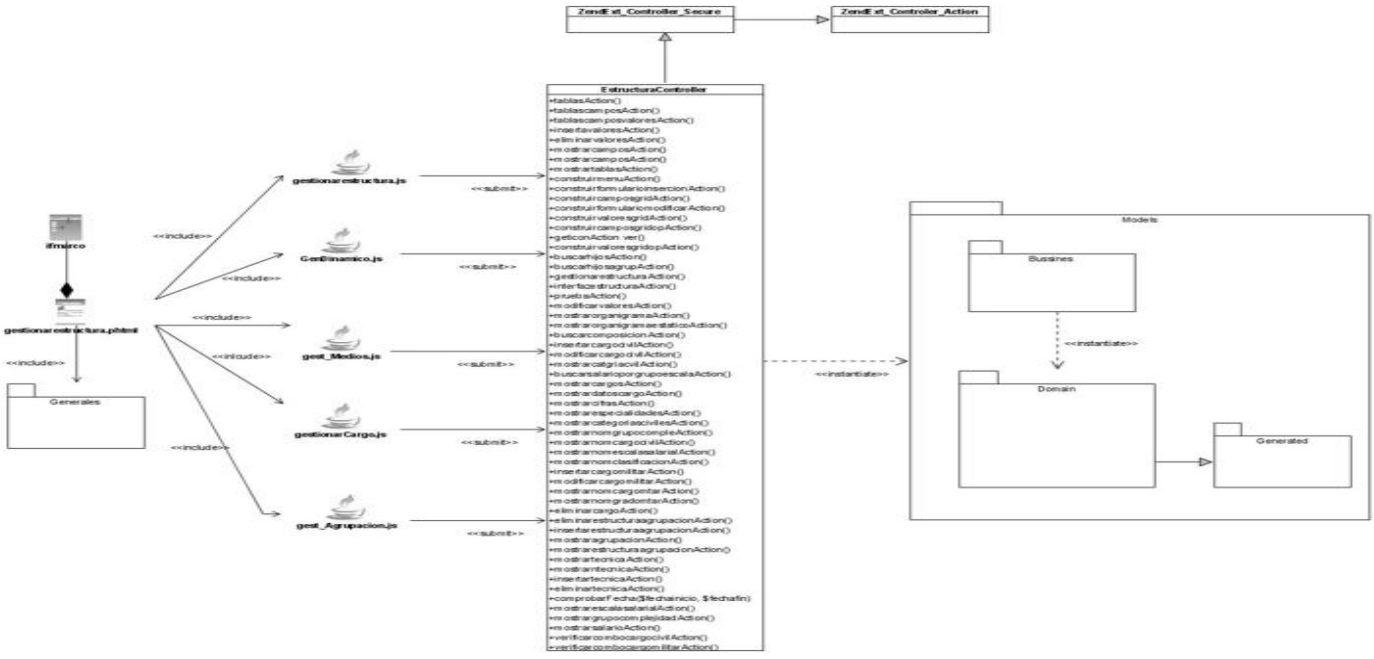


Anexo 4 Interfaz para la gestión de los campos de un nivel estructural seleccionado.

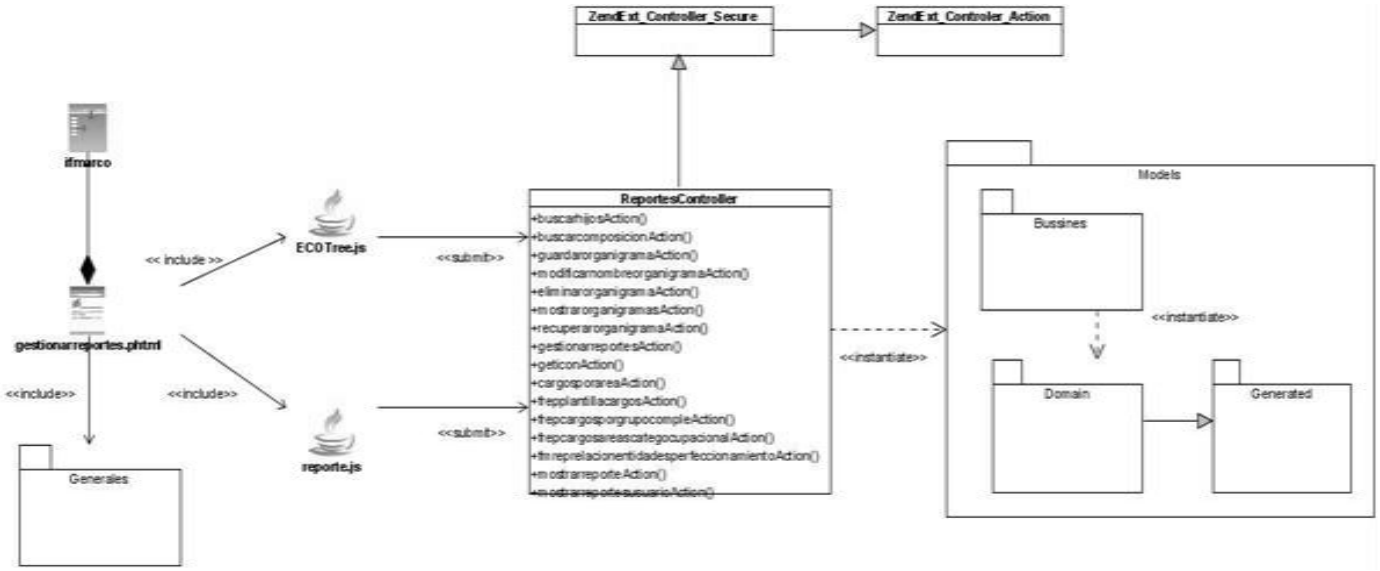


Anexo 5 Interfaz de la funcionalidad Gestionar estructuras.

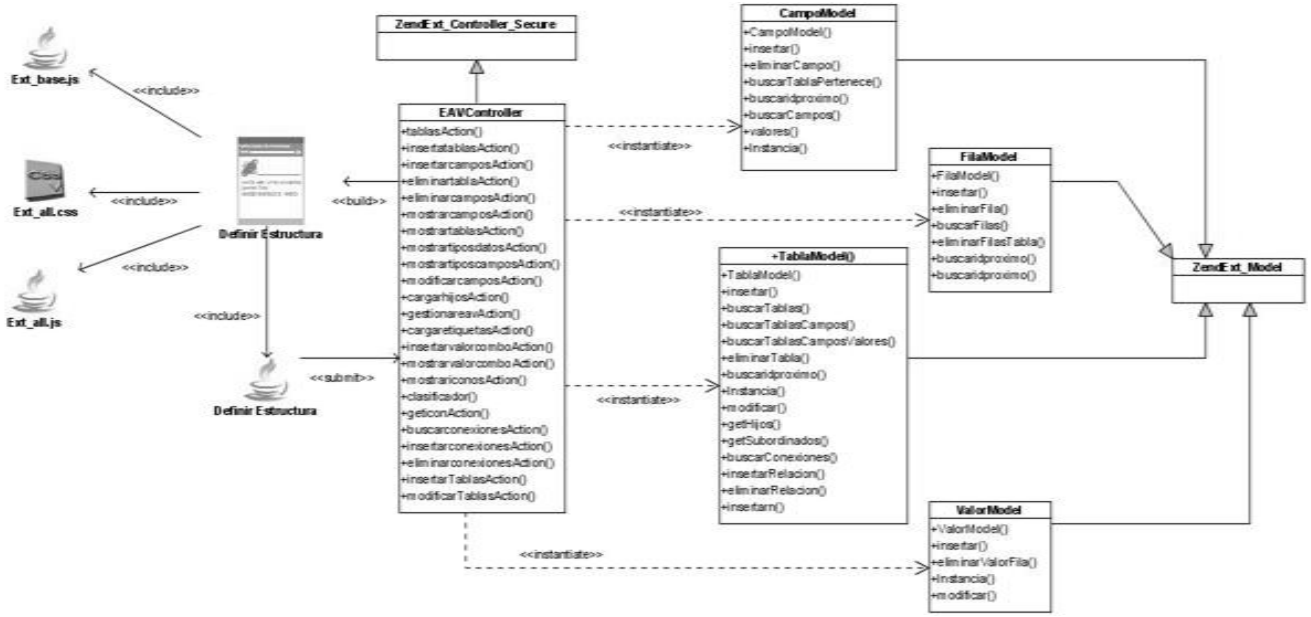
Anexo 6 Diagrama de clases Gestionar Estructuras



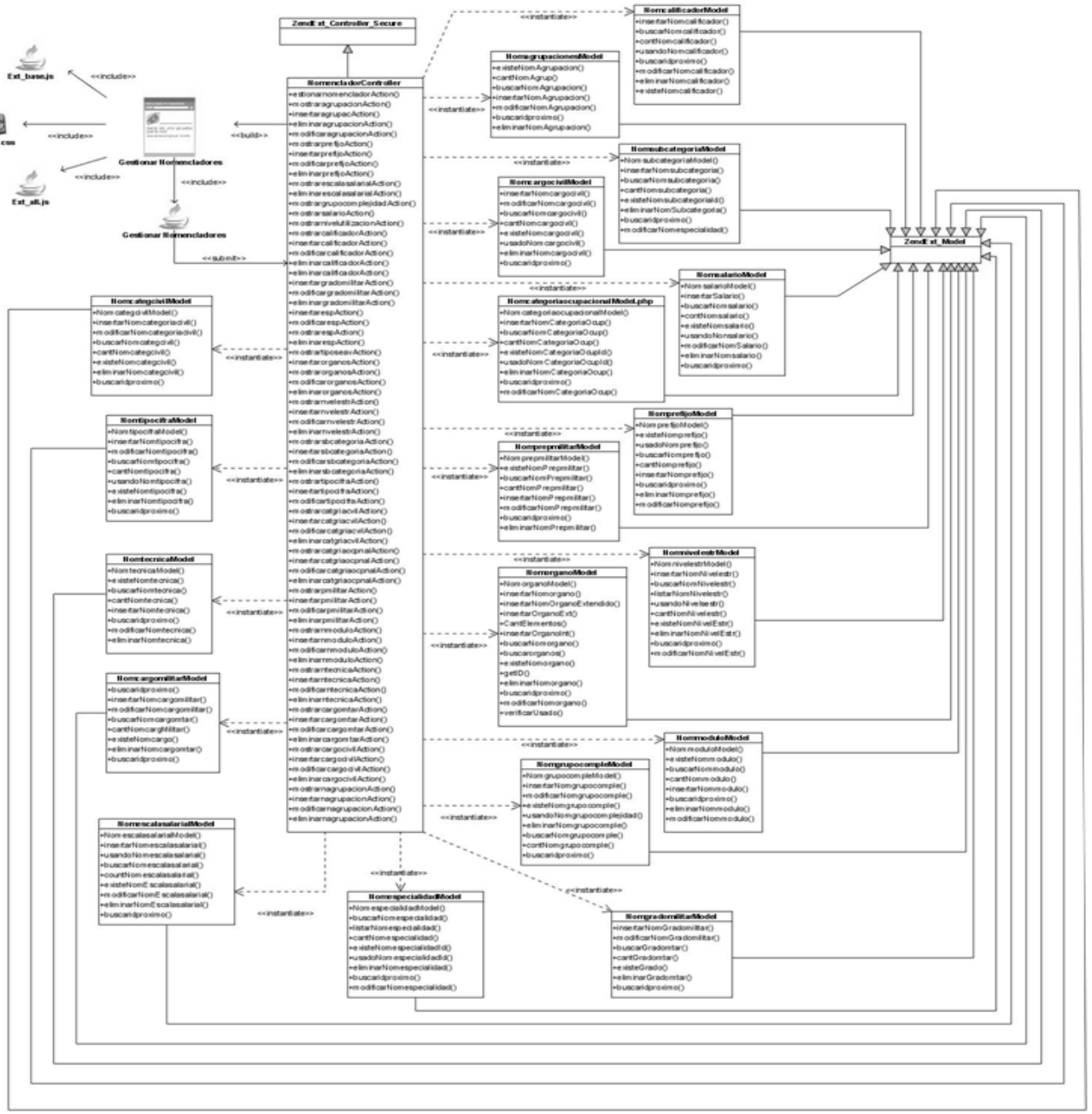
Anexo 7 Diagrama de clases Gestionar Reportes.



Anexo 8 Diagrama de clases Definir Estructura.



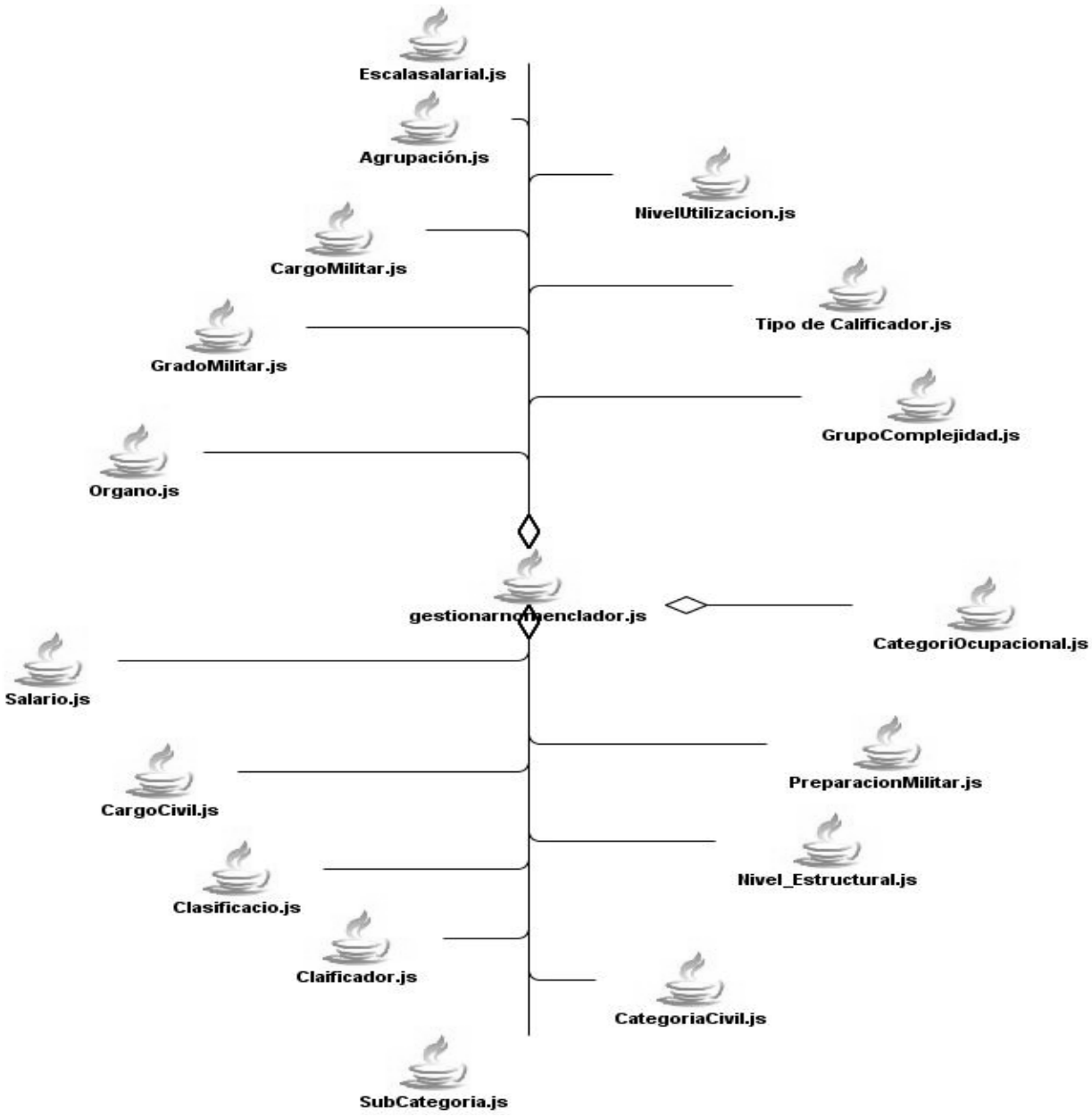
Anexo 9 Diagrama de clases Gestionar Nomencladores.



Anexo 10 Matriz de integración del subsistema Estructura y Composición.

		Brindan	
Consumen	**Componentes**	**Estructura y Composición**	**Nomencladores**
	Estructura y Composición		DPA Especialidad
	Capital Humano	DameEstructura, DameEstructuraInternas, ListadoEstructurasInternas DameAreaPorId, DameDatosCargo, DameCargoPorId, getCargobyId	
	Configuración	ListadoEstructurasT, DameEstructura, MostrarCamposEstructura	
	Costos y procesos	DameEstructurasInternas, DameHijosInterna, DameEstructura	
	Contabilidad	DameEstructura DameEstructurasinChecked, DameHijosInternaSeguridadSinCheked, DameEstructurasInternasSeguridadSinCheked, listadoEstructurasT	
	Finanzas	DameEstructrura, DameEstructurasInternas, BuscarCargosPorTipos, DameHijosEstructura, ListarEstructurasExternasDadoArrayId, DameAreasPorId, DameCargoPorId, buscarCargoPorNombreArea, DameDatosCargo, ListadoEstructurasT, MostrarCamposEstructura, getPadreSuperior	
	Logística	DameAreaPorId, DameEstructurasIternas, DameHijosInternas, DameEstructuras, BuscarGradoMilitar, BuscarPadrePorSubordinacion, BuscarHijoPorSubordinacion	
	Planificación	DameEstructura, MostrarCamposEstructura, DameHijosEstructura	
	Seguridad	DatosDominioDadoID, MostrarCamposEstructuraSeguridad, EstructurasInternasDadoIDSeguridad, CargoDadoIDSeguridad, BuscarDomio, DameEstructurasinChecked, DameHijosInternaSeguridadSinCheked, BuscarCargosPorTiposSeguridad, ListarEstructurasDadoArrayId, DameEstructurasInternasSeguridad, DameEstructuraSeguridad, DameHijosInternaSeguridad, BuscarCargosPorTiposSeguridad, EstructurasInternasDadoIDSeguridad, ListarEstructurasExternasDadoArrayId, ObtenerEstructurasPorIdOrgano, BuscarDomioDadoNombre, verificarNombreDominio, InsertarDominio, DatosDominioDadoID, IdEstructurasDominio, DominiosPorEstructura, ExtraerEstructura, ModificarDominio, EliminarDominio, BuscarIdDominioSeguridad, ObtenerEstructurasNoFormales	

Anexo 11 Diagrama de clases que representa la relación que existe entres las clases interfaces de la funcionalidad Gestionar Nomenclador.



GLOSARIO DE TÉRMINOS

- ✓ **EAV:** Por sus siglas en inglés Entity-attribute-value. Es un modelo de datos que se utiliza en los casos en que el número de atributos (propiedades, parámetros) que pueden ser utilizados para describir una cosa (una entidad u objeto) que es potencialmente muy amplio.
- ✓ **IDE:** Un entorno de desarrollo integrado o IDE (acrónimo en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación.
- ✓ **GRASP:** Patrones generales de software para asignación de responsabilidades por sus siglas en inglés General Responsibility Assignment Software Patterns.
- ✓ **Framework:** Significado en español marco de trabajo.
- ✓ **Grid**: Es un componente del Framework ExtJS para mostrar en forma de tabla los datos.
- ✓ **Panel**: Es un componente del Framework ExtJS contenedor de otros componentes del mismo Framework.
- ✓ **Cedrux**: Nombre del producto que se realiza en el proyecto del ERP Cuba.
- ✓ **RIA**: Sus siglas significan Aplicaciones de Internet Enriquecidas. Las aplicaciones RIA son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales. Normalmente, en las aplicaciones Web, hay una recarga continua de páginas cada vez que el usuario pulsa sobre un enlace. De esta forma, se produce un tráfico muy alto entre el cliente y el servidor, llegado muchas veces, a recargar la misma página con un mínimo cambio.