Universidad de las Ciencias Informáticas

Facultad 3

“Diseño e implementación del componente Retenciones del subsistema Capital Humano del Sistema Integral de Gestión Cedrux”

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

Autor: Luis Carlos Boza Cabrera

Tutor: Ing. Tania Teresa Loureiro Valladares

Co-Tutor: Ing. William González Obregón

La Habana, junio de 2012

Un hombre que no arriesga nada por sus ideas, o no valen nada sus ideas, o no vale nada el hombre.

Platón.

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Luis Calos Boza Cabrera

_____
Firma del Tutor

Ing. William González Obregón

_____
Firma del Tutor

Ing. Tania Teresa Loureiro Valladares

_____
Firma del Tutor

Datos de contacto

Ing. Tania Teresa Loureiro Valladares. Ingeniera en Ciencias Informáticas, graduada de la Universidad de las Ciencias Informáticas en el 2009. Actualmente se desempeña como Jefa del proyecto Caja de la línea Finanzas del departamento Desarrollo de productos del centro CEIGE en la UCI. Correo electrónico: ttloureiro@uci.cu.

Ing. William González Obregón. Ingeniero en Ciencias Informáticas, graduado con título de oro en la Universidad de las Ciencias Informáticas en el 2010. Se desempeña como Arquitecto de la Línea Capital Humano del Departamento Desarrollo de Productos del centro CEIGE. Correo electrónico: wobregon@uci.cu.

Agradecimientos

A mi querida Madre Margot, que la quiero con la vida, que es mi mayor tesoro, que es la inspiración suprema para alcanzar mis éxitos, a quien debo toda mi vida y a quien siempre estaré eternamente agradecido por darme todo el amor, el apoyo y la felicidad de tenerla siempre conmigo.

A mi hermana Rosita, gracias mi hermana por estar siempre ahí para mí.

A mi papá por sus consejos a tiempo. Por su ayuda y apoyo . Por creer en mí siempre y darme la confianza para crecer en la vida.

A mi novia por su comprensión y amor durante todos estos años, por entenderme y ayudarme a alcanzar todos mis sueños.

A Teresa por apoyarme y por haber sido paciente en todo momento.

A mi tutor William, le agradezco por su ayuda durante toda la tesis y su apoyo incondicional. Y a mi tutora Tania por enseñarme que siempre se pueden hacer mejor las cosas, a mis tutores por todo el apoyo del mundo y porque sin ustedes esta tesis no hubiera sido posible.

A julio el pinareño por ser un amigo en todo momento.. A mis amigos por estar siempre presente en todo momento y brindarme su apoyo y compañía.

A todas las personas del proyecto por su ayuda.

A todos muchísimas gracias

Dedicatoria

A toda mi familia y especialmente a mi madre, mi padre y mi hermana

A mi novia por su cariño y paciencia

A todos los amigos y compañeros con los que he compartido estos 5 años

Resumen

La Universidad de las Ciencias Informáticas está desarrollando el Sistema Integral de Gestión Cedrux. Este está compuesto por varios subsistemas entre los que se encuentra Capital humano, el que contiene el macroproceso de Estimulación moral y material el cual abarca el proceso de administración de la nómina. Una de las responsabilidades de este es garantizar el cálculo del pago a los trabajadores, siendo imprescindible para ello el proceso de gestión de las retenciones, estas son las deudas bancarias o empresariales por las que se le realizan descuentos al salario del trabajador cada cierto plazo acordado por ambas partes. El presente trabajo describe la solución informática ofrecida para la gestión de las retenciones mediante Cedrux. Para lograr esto se realizó una investigación sobre las características de los principales software que gestionan las retenciones, identificando las ventajas y desventajas de la solución que ofrecen, concluyendo que era necesario el desarrollo de una solución propia.

El resultado consiste en el diseño y la implementación de un componente software correctamente validado mediante pruebas de caja negra y caja blanca que cumple con los requisitos previamente identificados. La solución constituye una alternativa válida para realizar el proceso en cualquier empresa cubana, que permite manejar el submayor de retenciones que actualiza los débitos del trabajador y los pagos realizados, permitiendo al cálculo de la nómina tener los datos necesarios para realizar los descuentos al calcular los pagos de estos trabajadores. Teniendo de esta forma un producto netamente cubano y realizado con tecnología libre.

Palabras clave:

Capital humano, nómina, retenciones, submayor de retenciones

Índice de figuras

Índice de figuras

Índice de tablas.

Introducción

Las tecnologías de la información han alcanzado un avance significativo en los últimos tiempos, llegándose a hablar de una nueva revolución tecnológica en este aspecto para la humanidad. Los mayores aportes de dichas tecnologías se ven reflejados en el actuar diario del hombre al hacerse más simple el acceso a grandes cuantías de información. La realización ágil y fiable de procesos sobre cualquier tipo de dato y el enorme volumen para el almacenamiento de estos, además de incrementar día a día la necesidad de hacer más amena la visualización de los contenidos orientados al usuario.

Las empresas no se han quedado exentas de este avance. Para llevar su funcionamiento, hace ya varios años se vienen creando estrategias para la gestión de la información y el conocimiento, respondiendo a nuevos tipos de demandas provenientes de las modernas directrices gerenciales de las organizaciones. Dichas estrategias se han visto implementadas en sistemas de gestión. Abarcando los procesos fundamentales en las organizaciones.

Los sistemas de planificación de recursos empresariales (ERP por sus siglas en inglés) se diseñan con el propósito de gestionar procesos de una empresa u organización y ofrecer soluciones prácticas para problemas reales. Además de integrar la información en las entidades, suprimiendo los obstáculos en el acceso a la totalidad de los datos, contribuyendo así a la mejora de sus procesos fundamentales y proporcionando la modernización de los procesos de negocio, lo cual conlleva a mayor eficiencia y productividad.

Una de las áreas esenciales de cualquier ERP es la que permite gestionar las actividades que se realizan en el departamento de Capital Humano (CH) de una empresa, en dicho departamento se ejecutan diariamente procesos muy complejos como la Estimulación Moral y Material**,** que abarca varios procesos como las retenciones, el procesamiento de los diferentes tipos de nóminas de pago a los trabajadores, el pago por resultados, la estimulación moral y la estimulación material. Entre dichos procesos uno de los más importantes es la Gestión de las Retenciones, son las deudas bancarias o empresariales por las que se le realizan descuentos al salario del trabajador cada cierto plazo acordado por ambas partes y además sus submayores, definidos como un resumen de las retenciones de un trabajador determinado (1), con estos procesos se registrará gran cantidad de información perteneciente a los trabajadores existentes en la entidad.

Introducción

En Cuba, con el proceso de informatización que se viene realizando hace ya varios años, se ha logrado contar con un conjunto de empresas que utilizan software para gestionar sus procesos básicos, entre ellos los procesos de gestión de Retenciones para el Capital Humano de estas entidades. Sin embargo, esta práctica no se ha extendido completamente, y en las entidades donde se aplica, los reportes finales no ofrecen un tratamiento homogéneo al sistema de gestión de la información, teniendo esto como consecuencia desviaciones en los datos contenidos debido a la gestión manual de las retenciones, pues se trabaja con grandes volúmenes de información lo cual dificulta la entrega a tiempo de los reportes. Se evidencia además como limitación los altos costos implícitos en la adquisición de licencias para su uso. Por lo cual se pretende disponer de un Sistema cubano para la gestión de las retenciones, que esté capacitado para normalizar los procesos vinculados a la información de forma fidedigna, y ajustado a las peculiaridades de las entidades nacionales.

La Universidad de las Ciencias Informáticas (UCI) se encuentra trabajando en el desarrollo de un sistema de gestión nombrado Sistema Integral de Gestión Cedrux, este cuenta con un subsistema de Capital humano, que incluye el macro proceso Estimulación moral y material. Como parte de este macro proceso se tiene implementado el proceso de administración de la nómina, donde se determina el salario a pagar a los trabajadores durante su procesamiento. Para esto se hace necesario verificar si cada uno de los trabajadores cuenta con alguna retención y en caso positivo confirmar si dispone del saldo necesario para descontársela. En este momento se debe comprobar además cuánto es lo establecido para descontar en dependencia del período de pago al que pertenece la nómina, es decir, el valor de una quincena o el mensual. Una vez contabilizada la nómina se deben registrar los valores de cada una de esas deducciones para cada retención de cada trabajador y mantener de esta manera un submayor actualizado. En la actualidad la gestión de las retenciones no se encuentra automatizada, esto trae consigo que cuando se genere una nómina, se tenga que entrar manualmente por cada trabajador sus retenciones. Una vez realizado el descuento se tendría que mantener en formato duro el conjunto de operaciones que forman parte del submayor de retenciones, lo que puede provocar pérdida de la información, violación de la misma, así como una gran dificultad en el manejo de los totales pagados hasta el momento.

Por todo lo anteriormente expuesto se presenta como **problema a resolver:** La gestión manual de las retenciones afecta la integridad de su información para generar la nómina del subsistema Capital Humano del Sistema Integral de Gestión Cedrux. Por lo que se plantea el **objeto de estudio:** Los sistemas de

gestión del Capital Humano, enmarcado en el **campo de acción**: Los sistemas de gestión de retenciones. Para la resolución de dicho problema se plantea el siguiente **objetivo general:** Realizar el diseño y la implementación del componente Retenciones, de manera que se garantice la integridad de su información para la generación de las nóminas en el subsistema Capital Humano del Sistema Integral de Gestión Cedrux, que se desglosa en los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación que permita identificar los principales logros y limitaciones en cuanto a la gestión de las retenciones.
- Diseñar el componente Retenciones basado en los requisitos previamente identificados.
- Validar el diseño propuesto mediante la utilización de métricas.
- Implementar el componente Retenciones teniendo en cuenta el diseño realizado.
- Validar el componente obtenido mediante la realización de pruebas de caja blanca y caja negra.

Para darle solución a dichos objetivos se diseñan las siguientes **tareas a cumplir**:

- Actualización de los principales conceptos asociados a la gestión de las retenciones como parte del proceso de Estimulación moral y material.
- Análisis de los sistemas existentes para la gestión de las retenciones identificando sus características y deficiencias fundamentales.
- Caracterización de las tecnologías, lenguajes y herramientas propuestas para el desarrollo del componente.
- Valoración del diseño propuesto teniendo en cuenta los requisitos existentes.
- Diseño del componente Retenciones basado en los requisitos previamente identificados.
- Validación del diseño mediante la aplicación de métricas.
- Implementación del componente Retenciones teniendo en cuenta el diseño realizado.
- Realización de pruebas de caja blanca a las principales funcionalidades implementadas.
- Validación del componente obtenido mediante la realización de pruebas de caja negra.

Se presenta como **idea a defender** del presente trabajo de diploma: Si se realiza el diseño e implementación del componente Retenciones entonces se logrará la integridad de su información para la generación de las nóminas en el subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

Para dar cumplimiento a las distintas tareas antes señaladas, se llevarán a la práctica los siguientes **métodos de investigación:**

Métodos teóricos:

- **Analítico – sintético:** Este método posibilita el procesamiento de toda la información enfocada hacia la investigación, permitiendo organizar y simplificar el análisis, la teoría, los documentos, la bibliografía; extrayendo los elementos más importantes relacionados con la Gestión del Capital Humano.
- **Histórico – lógico**: Se utilizará para identificar los antecedentes y tendencias actuales en la gestión del Capital Humano en los ERP.

El presente documento está estructurado de la siguiente forma:

Capítulo 1: En el capítulo se describe el estado del arte de los sistemas que gestionan el capital humano, así como los elementos fundamentales acerca de las herramientas a utilizar para el diseño e implementación del componente Retenciones de Cedrux del que es objeto este trabajo de diploma.

Capítulo 2: En el capítulo se describen los principales elementos que componen el diseño y la implementación del componente Retenciones. Contando con los artefactos generados en estos procesos para el logro de este propósito.

Capítulo 3: En este capítulo se plasma la validación de la solución propuesta mediante las pruebas de caja blanca, caja negra y la validación del diseño, así como los resultados obtenidos en cada una de ellas luego de su aplicación. Estas pruebas son ejecutadas con el objetivo de avalar el cumplimiento de las exigencias del cliente y la calidad del sistema.

.

Capítulo 1: Fundamentación teórica.

1.1 Introducción

En el capítulo se describe el estado del arte de los sistemas que gestionan el capital humano, así como los elementos fundamentales acerca de las herramientas a utilizar para el diseño e implementación del módulo de Retenciones de Cedrux del que es objeto este trabajo de diploma.

1.2 Diseño teórico. Conceptos fundamentales

Los conceptos que se presentan a continuación constituyen los pertenecientes al dominio del problema y resultan imprescindibles para conseguir un mejor entendimiento del mismo.

Capital humano

Conjunto de conocimientos, experiencias, habilidades, sentimientos, actitudes, motivaciones, valores y capacidad para hacer, portados por los trabajadores para crear más riquezas con eficiencia (1).

Estimulación material

Sistema de acciones que interactúan y se integran con la estimulación moral, para motivar a los trabajadores en el logro de la eficiencia y eficacia y en la consecución de los objetivos estratégicos de la organización. El pago con arreglo al trabajo, por cantidad y calidad, es el elemento principal de la estimulación material (1).

Estimulación moral

Sistema de acciones que se realizan para propiciar el desarrollo de la moral socialista en el trabajo y el sentido de pertenencia; reconocer y promover el aporte laboral de los trabajadores en la consecución de los objetivos estratégicos y la elevación de la cultura de la organización, así como la satisfacción individual y colectiva de los trabajadores (1).

Retención

Parte o totalidad retenida de un sueldo, salario u otro haber (2).

1.3 Gestión de retenciones

Las deducciones o retenciones son descuentos que se realizan en las nóminas de acuerdo con las condiciones definidas en los expedientes de retenciones (3).

Dependiendo del caso que se esté tratando existen diversos tipos de retenciones como son las retenciones deducidas del Salario; retenciones deducidas del Subsidio; retenciones deducidas de las Vacaciones; retenciones deducidas del Estipendio y Neto.

Informes sobre retenciones

Para manejar dichas retenciones se utilizan un conjunto de informes en correspondencia con el proceso que se quiera realizar con la retención. Dichos informes pueden ser:

Submayor de retenciones: Se imprime para un mes determinado y muestra, para cada tipo de retención, el saldo inicial, la cuota, lo descontado en el mes, el ajuste y el saldo final. El informe puede obtenerse General, para una Dirección o un Centro de Costo. Se totalizan los datos para cada tipo de retención (3).

Nómina de retenciones: Muestra el empleado, el valor de la retención, el nombre y la dirección del Beneficiario. Puede imprimirse General o Para una Dirección, para todas las retenciones o solo para un tipo de ellas. Se utiliza generalmente en las pensiones alimenticias, en la cual un beneficiario puede cobrarla (3).

Retenciones pendientes: Constituye un listado de las retenciones que no pudieron deducirse porque los empleados no tenían capacidad de pago por concepto de salario. Este informe debe visualizarse una vez calculada la nómina para detectar si algún empleado cobró por salario menos que el valor de la cuota de la retención y aplicársela por otro concepto (3).

Expedientes de retenciones: Describen las deducciones aplicadas a los empleados. Aportando datos fijos que se utilizan en el cálculo de las nóminas: designan el empleado y la cuantía del descuento a aplicar (3).

1.4 Sistemas existentes para la gestión de las retenciones.

Un sistema de gestión es una herramienta informática probada para la gestión y mejora continua de la organización. Se crean para contribuir al logro de los objetivos de la organización implementando un conjunto de estrategias, que abarcan desde la optimización de procesos, hasta el enfoque centrado en la gestión y el pensamiento disciplinado (4). Se desarrollan para gestionar los riesgos sociales y financieros, perfeccionar la efectividad operativa, disminuir costos, incrementar la satisfacción de

clientes y otros interesados, proteger la marca y el prestigio, conseguir mejoras continuas, fomentar la innovación, suprimir impedimentos que puedan afectar al comercio y brindar claridad al mercado.

A continuación se abordan significativos elementos sobre los principales sistemas de gestión existentes a nivel internacional y en Cuba y específicamente, cómo ellos trabajan el proceso de gestión de retenciones.

1.4.1 Sistemas de gestión internacionales.

Los sistemas de gestión empresarial se han desplegado en el mercado del software con una fuerte presencia, entre los más representativos se encuentra **OpenbravoERP**, creado bajo licencia libre, completamente funcional, integrado y basado en tecnología web, contiene además funcionalidades CRM (*Customer Relationship Management* o Sistema de Gestión de la Relación con Clientes) básicas y un módulo de Inteligencia de Negocio. Openbravo permite gestionar dos tipos de Retenciones: Retenciones en Factura y Retenciones en el Pago. El segundo tipo, que se contabiliza en el momento del pago de la factura (5), se ajusta en mayor medida a la necesidad de realizar el descuento de las retenciones durante el procesamiento de la nómina. Sin embargo, trabaja el valor del descuento en por ciento. Este software no es el más adecuado debido a que no concuerda en gran medida con los procesos de gestión de las retenciones en Cuba. Otro ERP importante en el mundo es **Sage ERP X3,** constituye una eficaz solución multi-empresa, posibilita el trabajo a través de gran variedad de idiomas, divisas, empresas, ubicaciones y legislaciones. Integra toda la información y los procesos de negocio de la empresa en un único sistema de software y base de datos. Gestiona las operaciones de Finanzas, Ventas, Inventarios, CRM, Compras y Producción, de forma global y racionaliza todos los procesos de la compañía (6). Dicho sistema presenta igual desventaja que el ERP anterior, además de constituir un software propietario. **Solmicro** constituye otro referente mundial entre los ERP, se encuentra especialmente diseñado para instalaciones distribuidas y todo tipo de entornos de trabajo (Internet, VPN, ASP), sin dependencia de software de conexión. Construido orientado a componentes, en un modelo de arquitectura de 3 capas y .NET, constituyendo el único en el mercado con estas funcionalidades (7). El caso de Solmicro es semejante a los anteriores, no siendo recomendable de esta forma su utilización en el país.

Elementos sobre la gestión de retenciones en sistemas ERP

Después de realizar una investigación acerca de la gestión de las retenciones a nivel mundial en los sistemas ERP, se encontraron algunos ejemplos significativos que se presentan a continuación:

SAP: Sistema de gestión escalable, se ajusta a todos los tamaños y tipos de organizaciones empresariales. Soporta servidores ilimitados y se ejecuta en diversas bases de datos. Dicho sistema explica en gran medida cómo se gestionan las retenciones en su negocio, esto se realiza mediante la integración del componente de Cálculo de nómina, Gestión de personal, Gestión de tiempos, Incentivo y Finanzas, calculando las remuneraciones bruta y neta; esto incluye los devengos y retenciones individuales utilizando diferentes claves de concepto de nómina. En este sistema existe una cuenta denominada I.R.P.F (impuesto directo que grava el total de los rendimientos de una persona obligada tributariamente), que para ella se realiza de forma particular la gestión de las retenciones. El proceso de datos para el cálculo de las retenciones e ingresos a cuenta del I.R.P.F. está definido en el esquema principal de cálculo de la nómina. Dicho subesquema maneja las funciones y reglas de cálculo correspondientes y luego de realizar una serie de pasos es que se calculan los importes de retención. Registra los gastos totales para poder determinar las utilidades brutas en operaciones como componente del valor agregado bruto. Así como tener registrado también el comportamiento de los gastos de salarios y sus costos asociados. Determina el volumen del valor agregado bruto para poder determinar la productividad por trabajador y también el salario medio por trabajador. Sin embargo, este sistema presenta como inconveniente fundamental su condición de software privativo que contrasta con la política de migración a software libre de Cuba, derivando así un costo por concepto de licencia para las entidades. (8)

EL software **EPICOR** gestiona las retenciones en organizaciones medianas en industrias tales como los servicios, comercio minorista, fabricación, distribución, etc. Además provee en profundidad CRM, Inteligencia de Negocio y SCM (Administración de la Cadena de Suministros). En él se incluyen la visualización de la nómina por parte del empleado, y dentro de dicha nómina en específico los importes por concepto de retenciones (9). Este software tampoco es adecuado para la implantación en Cuba pues no está acorde a la política de independencia tecnológica llevada a cabo en el país.

1.4.2 Sistemas de gestión nacionales.

SISCONT5: Sistema cubano desarrollado por la empresa Tecnomática en 2007, el cual se ajusta a las definiciones y conceptos del Ministerio de la Industria Básica (MINBAS), además por las acciones

contables financieras con que cuenta puede ser utilizado en otras empresas nacionales. Se compone por varios módulos: Efectivo en Caja y Bancos, Inventarios, Cobros y Pagos, Facturación, Activos Fijos Tangibles, Nóminas y Contabilidad de Costos. Brinda la posibilidad de efectuar los descuentos por los diferentes conceptos de retenciones a los trabajadores en las Nóminas que se procesen, obteniendo listados como constancia de las aplicaciones o no en cada período de pago y sus saldos. Incluye además el Salario Devengado Bruto, que no es más que el importe total de conceptos salariales devengados por Nóminas oficiales, sin considerar los descuentos por Retenciones, Contribución Especial a la Seguridad Social, etc. (10).

El sistema cuenta con una interfaz para la captación de los Conceptos de Retenciones por Pagar. Para la apertura, si no tiene el trabajador asociado en el Maestro de Retenciones, se pueden modificar todos los atributos, en caso contrario permite la Descripción, Prioridad y Nombre para Impresión. El documento Maestro de Retenciones mencionado anteriormente es de vital importancia para el sistema, debido a que permite captar en él los datos generales y específicos de cada trabajador, utilizados tanto por el Subsistema de Nóminas como por el Subsistema de Cobros y Pagos. Permite además definir las Retenciones por Trabajador para su pago a terceros, analizadas por claves, que se van a descontar en las Nóminas que procedan en el corte que se precise y tratadas para su pago por Cobros y Pagos si así se configura. Dicho documento se mantiene actualizado a través de las opciones de Alta, Modificar o Baja. En el caso de las Retenciones con Plazos se permite modificar el Corte, la Deuda y el Importe del Plazo y para las Retenciones sin Plazos el Corte y el Importe del plazo. Sin embargo, el empleo de tecnología privativa es el principal inconveniente que presenta dicho sistema contable.

Versat-Sarasola: Es una de las soluciones informáticas con mayor uso actualmente en el país, desarrollado por la entidad cubana TEICO (empresa del Azcuba encargada de la Informática y las Comunicaciones), que gestiona la mayor parte de las actividades de planificación, control y análisis económico de cualquier tipo de entidad, debido a su condición de ser configurable. Es una aplicación de entorno de escritorio, concebido sobre una plataforma de trabajo Cliente-Servidor lo que permite su instalación en red, dicha instalación establece dos variantes fundamentales: la variante típica que incluye la totalidad de los subsistemas del Versat-Sarasola y la variante personalizada en la que solo se instalarán los subsistemas seleccionados por el usuario.

Permite el control y registro contable individual de todos los hechos económicos creados en las estructuras internas de las empresas, y posibilita la obtención de los estados financieros y análisis económicos y financieros en estos niveles. Presenta el subsistema de contabilidad general como rector del sistema que maneja el resto de los subsistemas. Versat-Sarasola facilita el trabajo con las retenciones, un factor importante dentro de las nóminas, estas traen una descripción de las retenciones del trabajador, establece si las mismas van a ser notificadas al banco, la cuenta que se asocia a la retención y en caso de que esta requiera ser notificada al banco, se debe especificar la cuenta donde se va a liquidar la retención (11).

Entre otros aspectos de interés en la gestión de las retenciones, dicho sistema también maneja las bajas de dichas retenciones, las cuales pueden ser: traslados, bajas, esas últimas pueden ser por renuncias, fallecimiento y jubilación, etc. Cuenta además con un submayor de retenciones, en el cual no se pueden modificar ni capturar, este se encuentra definido por tipo de operaciones como son: reintegro de vacaciones, reintegro de salarios, ajuste del salario acumulado etc. Versat-Sarasola es una de las soluciones informáticas con mayor uso actualmente en Cuba. Una de las facilidades más significativas con que cuenta este software es que tiene en cuenta la dualidad de monedas y la multientidad. Presenta el inconveniente de que debe ser utilizado en Windows, que es una aplicación de escritorio y utiliza como sistema gestor de base de datos SQL Server 2000, por lo que no cumple con el principio de independencia tecnológica que promueve el país y no presenta mecanismos estándares de integración con otras aplicaciones de su tipo.

RODAS XXI: Es un Sistema Integral Económico Administrativo cubano desarrollado por CITMATEL (Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados). Concebido como un sistema multiempresa y presenta un conjunto de módulos entre los que se encuentran: Contabilidad, Activos Fijos, Nóminas, Inventario, Facturación y Recursos Humanos. Dichos módulos se encuentran integrados entre sí logrando un intercambio eficiente y coherente de la información. Permite la gestión, planificación y control de los Recursos Humanos utilizando la actualización de los datos del personal incluyendo su foto, así como el registro de incidencias y los datos ineludibles para el módulo de Nóminas.

El sistema es capaz de guardar por trabajador los pagos y retenciones fijas que se realizan a cada uno de ellos, por lo que para la confección de las nóminas cada mes solo es necesario actualizar las

incidencias que correspondan y todo el trabajo posterior de cálculo es realizado de forma automática. El submayor de vacaciones, el de retenciones y el de decreto ley 91 son generados automáticamente por el sistema, al igual que los salarios devengados y las retenciones por trabajador (12). También son generados de forma automática varios modelos, además del reporte de bajas, el resumen de otros pagos y de salarios devengados. Siempre que se desee se pueden ver los reportes correspondientes a cada una de las nóminas emitidas, la nómina en sí, su comprobante, los sobres para pago, el desglose de efectivo, las retenciones realizadas. El RODAS XXI es un sistema de gestión del personal de forma organizada, sin embargo, como principal desventaja, no es multiplataforma pues solo es compatible con el sistema operativo Windows 2000/XP.

ASSETS NS: Introducido en Cuba en 1997, es utilizado por diferentes sectores en el país entre los que están, el Ministerio de la Educación Superior, el Consejo de Estado, la Aduana General de la República, el Ministerio de Justicia y el de Finanzas y Precios. Contiene un módulo para desarrollar las actividades relacionadas con la gestión de los Recursos Humanos donde se pueden controlar plenamente los recursos laborales: empleados, estructura organizativa de la entidad y plantilla, además posibilita la realización de funciones como la introducción de altas, bajas y otros movimientos de empleados en una organización determinada.

En asset, los Expedientes de Retenciones se comportan mientras están activos, aportando datos fijos que se utilizan en el cálculo de las nóminas: designan el empleado y la cuantía del descuento a aplicar. Desde el módulo Expedientes de Retenciones no es posible borrar o eliminar un expediente, sin embargo, un expediente puede ser desactivado en cualquier momento. El sistema cuenta con una Ficha Informes que muestra para cada expediente y tipo de retención, el saldo inicial, la cuota, lo descontado en el mes, el ajuste en caso de que exista y el saldo final. Al empleado se le permite definir en cuál de los pagos se efectuará la retención de la cuota definida, estas no son excluyentes, por lo que pueden realizarse retenciones en ambos pagos (13). Dicho sistema aunque cumple con la gestión de los trabajadores no permite mantener un registro actualizado de la totalidad del personal con el que se cuenta en la entidad en un momento determinado, además no constituye un sistema multiplataforma, y el lenguaje utilizado para su desarrollo Visual Basic por lo que su código se encuentra cerrado e invisible, por lo que no cumple con la soberanía tecnológica que se desea para las entidades cubanas.

1.4.3 Valoración de los sistemas de gestión de retenciones.

Después de la realización de una investigación sobre los sistemas de gestión anteriormente mencionados, se llega a la conclusión de que presentan disímiles inconvenientes para la gestión de las retenciones y las licencias de dichos productos. Por lo que sería entonces más económico y factible para el país crear un sistema de gestión que responda a sus individualidades. Igualmente que sea consecuente con el proceso de conseguir la soberanía tecnológica, haciéndose necesario el desarrollo de dicho sistema sobre herramientas y tecnologías libres basadas en la web, con el fin de lograr un alto grado de eficiencia, confiabilidad y rapidez en la gestión de las empresas. A continuación se muestra una tabla con algunas de sus características principales (ver tabla 1).

Sistemas	Cumplimiento con el paradigma de independencia tecnológica del país	Ejecución de todas las funcionalidades necesarias.
OpenbravoERP	Sí	Sí
Sage ERP X3	No	No
Solmicro	No	No
SAP	No	No
EPICOR	No	No
SISCONT5	No	No
Versat-Sarasola	No	No
RODAS XXI	No	No
ASSETS NS	No	No

Tabla 1 Características generales de los sistemas.

1.5 Modelo de desarrollo del software

Para el desarrollo del componente Retenciones y con el objetivo de lograr su exitosa culminación, se sigue una probada visión de la aplicación de los procesos de desarrollo de software dictadas por el Centro de Informatización de la Gestión de Entidades (CEIGE), en específico por su Departamento de Desarrollo de Productos, en su modelo de desarrollo. Este modelo describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones. Logrado por la combinación de los modelos Orientado a componentes e Iterativo e incremental (14), el mismo presenta las siguientes características:

- Centrado en la arquitectura
- Iterativo e Incremental
- Orientado a componentes
- Ágil y adaptable al cambio

1.6 Herramientas y tecnologías seleccionadas para la solución.

En el Departamento de Tecnología del centro CEIGE se realizó un estudio para determinar las herramientas y tecnologías que serían más propicias para el desarrollo del Sistema Integral de Gestión Cedrux (14). La integración de estas consiste en un marco de trabajo denominado Sauxe. Seguidamente se abordan los elementos principales de dichas herramientas y tecnologías:

1.6.1 Lenguaje de modelado.

En el proceso de desarrollo de proyectos informáticos convertir las necesidades de los clientes en funcionalidades del sistema constituye una de las principales actividades a llevar a cabo, pero en este proceso surgen diversos problemas, dados principalmente por las malas interpretaciones debido a la gran diversidad en las notaciones para el modelado, las metodologías, y otras definiciones que pueden crear confusiones entre los desarrolladores y los usuarios finales del sistema. Dando como resultado productos solamente comprendidos por sus creadores.

Para la representación visual del interior de dichos productos informáticos se utilizan modelos conceptuales. Un modelo es una representación de (parte de) un sistema, detallado en un lenguaje bien

definido, es decir, un lenguaje con sintaxis y semántica precisa, y que puede ser interpretado y manipulado por un ordenador (15).

Para la creación de los modelos mencionados anteriormente, se emplean lenguajes de modelado. Dichos lenguajes brindan los elementos básicos para la descripción de cualquier sistema. Estos han tomado gran importancia en la industria del software, debido a que el modelado es una parte central de todas las actividades que rigen la implementación de buen software.

Con la aplicación de un correcto modelado se logra:

- La visualización de lo que se quiere que sea un sistema.
- La especificación de la estructura interna o comportamiento de un sistema.
- Proporcionar plantillas para guiar la elaboración de un sistema.
- Documentar las decisiones tomadas.

UML constituye uno de los lenguajes de modelado más utilizados y estandarizados en el mundo (16), es por eso que se utilizará para el modelado de las clases, paquetes y subsistemas en la solución del componente Retenciones del subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

Lenguaje unificado de modelado (UML por su acrónimo en inglés)

Constituye un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. Concebido como un lenguaje de modelado de propósito general, por lo que puede emplearse para la especificación de la mayoría de los sistemas orientados a objetos o solo para componentes, e igualmente para modelado de aplicaciones de disímiles dominios de aplicación (telecomunicaciones, comercio, sanidad, etc.) y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET o CORBA), proporcionando una gran flexibilidad y expresividad a la hora de modelar sistemas (15).

Características principales:

- Lenguaje distribuido y adecuado a las necesidades de conectividad actual y futura.
- Reemplaza diversas notaciones empleadas con otros lenguajes.
- Permite el modelado de complicadas estructuras.

- Las estructuras fundamentales que soportan poseen su cimiento en tecnologías orientadas a objetos, tales como objetos, clase, componentes y nodos.
- Emplea operaciones abstractas para guiar las variaciones futuras, agregando variables si se necesita.
- Brinda soporte para conocer el comportamiento del sistema: casos de uso, diagramas de secuencia y de colaboraciones, con el objetivo de evaluar el estado de las máquinas.

UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- Acrecentado rigor en la especificación.
- Posibilita la verificación y validación del modelo creado.
- Brinda la posibilidad de automatización de determinados procesos.
- Permite la generación de código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos).

1.6.2 Herramienta de modelado.

El desarrollo del software ha generado la construcción de herramientas para que los analistas y diseñadores de software realicen los procesos de modelado del sistema de forma eficiente, con mayor calidad y fiabilidad.

Las herramientas CASE (*Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) son un ejemplo de ello, constituyen un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un *software* (17).

Actualmente existen una gran cantidad de herramientas CASE. Entre las más utilizadas se encuentran *Easy CASE, Oracle Designer, System Architect* y *Visual Paradigm for* UML. Para la realización de la solución de este trabajo de diploma se empleará el *Visual Paradigm for* UML.

Visual Paradigm 6.4

Herramienta profesional que brinda soporte al modelado visual mediante la notación UML. Soporta la totalidad del ciclo vital del desarrollo de software: análisis y diseño orientados a objetos, construcción,

pruebas y despliegue. Herramienta con funcionalidades muy completas y con abundantes facilidades, su empleo contribuye a la construcción rápida de aplicaciones de calidad, óptimas y con una minoría en cuanto a costo. Permite además modelar todo tipo de diagramas de clases, código inverso, generar el código desde los diagramas, posee interoperabilidad con otras aplicaciones e integración con distintos Ambientes de Desarrollo Integrado (IDE), suministra abundantes tutoriales y crea de manera simple toda la documentación, incluyendo formatos tales como PDF y HTML (18).

Algunas de las ventajas que posee esta herramienta son las siguientes:

- Creado para múltiples plataformas.
- Ofrece la oportunidad de intercambiar información mediante la importación y exportación de ficheros con aplicaciones.
- Brinda un apoyo adicional en cuanto a generación de artefactos automáticamente.
- Se puede generar código a partir de los diagramas, así como obtener los diagramas a partir del código, denominado generación de código e ingeniería inversa.
- Posee las funcionalidades para documentar todo el trabajo sin necesidad de utilizar herramientas externas.

Después de modelado el sistema ya todo estaría listo para su implementación, es ahí donde vienen a jugar su papel la selección de uno o varios lenguajes de programación adecuados en correspondencia con la aplicación que se quiera construir.

1.6.3 Lenguajes de programación.

Un lenguaje de programación está definido como un lenguaje para describir el conjunto de acciones consecutivas que un equipo debe ejecutar (19). Por lo tanto, constituye un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Está compuesto por un grupo de reglas gramaticales, un grupo de símbolos utilizables, un grupo de términos con sentido único y una regla principal que resume las demás.

Para la realización de la solución en cuestión se utilizará **PHP** y **JavaScript**.

ExtJS 2.2

Conjunto de librerías Java Script que admite el desarrollo de aplicaciones RIA (*Rich Internet Applications*) basadas en un navegador. Ofrece al desarrollador un gran conjunto de *widgets* (componentes como por ejemplo celdas, ventanas de diálogo) completamente integrados y un API (*Application Programming Interface*) para lograr interfaces web más dinámicas e interactivas al usuario (20).

Para la creación de las interfaces de usuario, así como para el manejo de eventos en cada una de las páginas, ExtJS emplea el lenguaje JavaScript en conjunto con HTML.

Dichas librerías incluyen:

- Modelo de componentes extensibles.
- APIs (*Application Programming Interface*) fáciles de usar.
- Licencias de códigos abiertos y comerciales.

Se utilizará la versión 2.2 debido a que posee una alta capacidad de soporte de navegadores, manteniendo igual apariencia independientemente del navegador y la plataforma del cliente. Se integra con aplicaciones desarrolladas con .Net o PHP.

PHP 5.2.6

Acrónimo de *Hipertext Preprocesor*, constituye un lenguaje de programación script del lado del servidor gratuito e independiente de la plataforma sobre la que se trabaje, eficiente, cuenta con una gran librería de funciones y mucha documentación (21).

Entre las características que definen a este lenguaje y que lo convierten en una poderosa herramienta se encuentran:

- Lenguaje licenciado bajo código abierto.
- Soporta diversos gestores de bases de datos entre los que se encuentran (MySQL y PostgreSQL).
- Se integra con diferentes bibliotecas externas, permite la generación de documentos en formato PDF (documentos de *Acrobat Reader*) y el análisis de código XML.

- Brinda una solución sencilla y global para las paginaciones dinámicas Web de fácil programación.

Doctrine 1.2.1

Doctrine es un potente y completo sistema ORM (*Object Relational Mapear*) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus características fundamentales es la capacidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto ofrece una alternativa poderosa a los diseñadores de SQL debido a que aporta un máximo de flexibilidad sin requerir la duplicación redundante de código. Además permite la exportación de una base de datos previamente creada a sus clases correspondientes y viceversa. En resumen consiste en una serie seleccionada de librerías PHP, enfocadas principalmente en la prestación de servicios de persistencia y funcionalidad (22).

Todo sistema de gestión necesita de una base de datos para guardar la información que en él se maneja, para ello es imprescindible contar con herramientas que soporten y faciliten estos procesos.

1.6.4 Herramientas de base de datos.

Gestor de base de datos

En un ERP se hace imprescindible poseer una base de datos para el soporte de la persistencia de los datos. Debido a esto se hace necesario el uso de una herramienta que garantice la gestión de dicha base de datos. Para estas funciones se utilizará un Sistema Gestor de Base de Datos.

Un Sistema Gestor de Bases de Datos (SGBD) se define como la colección de programas cuyo principal objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Estos posibilitan la definición de los datos en distintos niveles de abstracción y manipularlos, avalando la seguridad e integridad de los mismos (23).

Dentro de los numerosos SGBD que se utilizan en la actualidad para la gestión de bases de datos se seleccionó el PostgreSQL.

PostgreSQL 8.3

Constituye en la actualidad el SGBD de código abierto más desarrollado del mundo y opera bajo licencia BSD (*Berkeley Software Distribution*). Está compuesto por: herencia, tipos de datos, funciones,

restricciones, disparadores, reglas e integridad transaccional. Sin embargo su concepción no es meramente orientada a objetos. La versión seleccionada incluye gran cantidad de mejoras en las herramientas de administración, funcionalidades, comandos, consultas y programación de BD simplificándoles el trabajo a los usuarios (24).

Algunas de las características con las que consta son:

- Sistema multiplataforma.
- La durabilidad asegura que un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.
- El aislamiento consiste en que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generarán ningún tipo de error.
- Documentación basta, muy bien organizada, pública y libre.
- Plenamente adaptable a las necesidades del cliente.
- Soporte nativo para los lenguajes más populares del medio: PHP, C, C++, Perl, Python, etc.

PgAdmin III

Considerada como la herramienta más completa y popular de código abierto para la administración de bases de datos en PostgreSQL. Cuenta con una interfaz gráfica que soporta todas las particularidades de este a partir de su versión 7.3, ejecutándose en cualquier plataforma (25). Facilita la administración y está diseñado para solventar las necesidades de los usuarios, teniendo en cuenta la realización de consultas SQL desde la más sencilla, hasta el desarrollo de bases de datos de alta complejidad.

1.6.5 Navegador web.

La aplicación a desarrollar como propuesta de solución al problema planteado es basada en tecnología web, por lo que se hace necesaria la utilización de un navegador para la interacción con su contenido.

Un navegador o navegador web es un programa que permite visualizar la información de una página web. Para ello interpreta el código con el que se implementa la página web y lo presenta en pantalla, permitiendo al usuario interactuar con su contenido y navegar hacia otros sitios mediante enlaces o hipervínculos.

Mozilla Firefox 2.17

Se empleará como navegador predeterminado Mozilla Firefox debido a que es un navegador de software libre. Posee como características principales que brinda una forma rápida y eficiente de navegar por la web, permite abrir varias páginas en una misma ventana mediante el empleo de pestañas separadas, entre otros aspectos que ofrecen facilidad de navegación y visualización para los usuarios (26).

1.6.6 Control de versiones

El empleo de un correcto control de las versiones en el desarrollo de un sistema es de vital importancia, debido a que en la mayoría de los casos la construcción de los sistemas se realizan por diferentes desarrolladores, por lo cual, es necesario contar con una herramienta que guarde todos los cambios hechos a los ficheros y directorios.

Se denomina control de versiones a los métodos y herramientas disponibles para controlar los aspectos relacionados con los cambios en el tiempo de un archivo.

Subversion (SVN 1.6.6)

Constituye un sistema de control de versiones libre y de código fuente abierto. El componente fundamental de Subversion es su repositorio, el cual es un almacén central de datos. El almacenamiento de la información se realiza en forma de árbol de archivos. Soporta cualquier número de clientes conectados al repositorio leyendo o escribiendo en esos archivos **(27)**.

Entre las características que poseen dichos sistemas se encuentran:

- Entendimiento con los lenguajes de programación.
- Ofrece una serie de herramientas para la construcción de software.
- Puede ser utilizado para administrar cualquier conjunto de ficheros.

1.7 Conclusiones del capítulo

Después de realizado un análisis bibliográfico han quedado expuestos los principales elementos conceptuales que centran la investigación en cuestión. Además se ha fundamentado la selección de las herramientas y metodologías a utilizar al ser presentadas sus fundamentales características. Se realizó

un análisis de sistemas que presentan características similares al que se desea realizar, teniendo en cuenta sus funcionalidades, explicando los inconvenientes que presentan y por lo que no pueden ser considerados como propuestas de soluciones. Después todo lo antes expresado se consideran sentadas las bases para el desarrollo del trabajo. Cumpliéndose de esta forma con los objetivos que se pretendían cumplir con el desarrollo del presente capítulo.

Capítulo 2: Diseño e implementación.

2.1 Introducción

En el capítulo se describen los principales elementos que componen el diseño y la implementación del componente Retenciones. Contando con los artefactos generados en estos procesos para el logro de este propósito.

2.2 Análisis de los artefactos entregados por los analistas

El punto de partida para diseñar la solución técnica constituye el análisis de los artefactos entregados por los analistas, los cuales son el Modelo conceptual y la Descripción de los requisitos funcionales. Después de analizar la Descripción de los requisitos se evidenció que los mismos se encontraban correctamente redactados, de forma clara, posibles de probar y además cubrían todas las funcionalidades necesarias para el componente. Como consecuencia de no haberse detectado errores en dichos artefactos no se propusieron cambios. Al analizar todo lo mencionado anteriormente se concluye que la propuesta es correcta, clara, completa y consistente, de ahí se procede a modelar el diseño.

2.3 Diseño

En el proceso de diseñar un sistema informático se aplican una serie de técnicas y principios con el propósito de concebir y documentar el diseño de un producto de software integral comprendiendo todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos, que corresponda con los requisitos identificados previamente. Se describe además la línea base de la arquitectura. Todo esto con el objetivo de constituir una abstracción de la implementación del sistema (28). Para lograr este objetivo se emplean una serie de patrones arquitectónicos y de diseño que se describirán posteriormente.

2.3.1 Patrones arquitectónicos

Los patrones arquitectónicos delimitan la interacción de los objetos dentro o entre niveles arquitectónicos. Estos proporcionan la adaptabilidad a requerimientos cambiantes, el rendimiento del sistema, su modularidad y acoplamiento. Solucionando las llamadas entre objetos, decisiones y criterios arquitectónicos, mediante el empaquetado de funcionalidades (28).

Los patrones arquitectónicos “expresan el esquema de organización estructural fundamental para sistemas de software, proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y pautas para la organización de las relaciones entre ellos” (29).

Modelo Vista Controlador (MVC).

El patrón MVC divide una aplicación web interactiva en tres módulos en la cual, generalmente, en el modelo están comprendidas las funcionalidades básicas y los datos del sistema, la vista es la encargada de visualizar la información, y la función básica de los controladores es gestionar las contribuciones ingresadas por los usuarios (30).

Seguidamente se abordan con mayor profundidad las responsabilidades de cada uno de ellos.

Modelo: responsable del acceso a la capa de almacenamiento de los datos, posibilita la definición de las reglas de negocio, es decir las funcionalidades del sistema, el registro de las Vistas y Controladores del sistema, además permite notificar a las Vistas los cambios que en los datos produce un agente.

Controlador: encargado de recibir los eventos de entrada, dígase un clic, cambio en algún campo de texto, entre otros. Comprende un conjunto de elementos o reglas para la gestión de eventos. Estas operaciones pueden suponer peticiones al Modelo o a las Vistas.

Vista: responsable de mostrar al usuario los datos recibidos desde el Modelo, poseen además un registro de su Controlador asociado. Posibilita brindar el servicio de "Actualización ()", para que este sea invocado por el controlador o por el modelo en el caso de ser un modelo activo.

El Patrón Arquitectónico Modelo-Vista-Controlador (MVC) se empleó en la solución de la forma siguiente: Las clases del modelo incluidas en los diagramas de clases representan la lógica del negocio. La vista convierte el modelo en una página Web que permite la interacción con los usuarios del sistema. El controlador es el responsable de procesar las interacciones del usuario y efectúa los cambios adecuados en el modelo o en la vista. Por ejemplo: Un usuario realiza una petición a la clase controladora RegistroretencionController es la que procesa la información y solicita los datos en la clase modelo NomTiporetencionesModel, que se encarga de realizar las operaciones pertinentes en la base de datos y enviar los datos pedidos a la clase controladora donde son capturados en la vista registroretencion.js donde se le da la respuesta al usuario.

Por todo lo anteriormente expuesto, además por posibilitar un mejor acoplamiento entre las capas y debido a la modularidad y el diseño independiente que este proporciona, permite a los desarrolladores y diseñadores hacer cambios en alguna parte de la aplicación sin afectar a los demás.

2.3.2 Patrones de diseño

Un patrón de diseño constituye una solución probada a un problema de diseño común documentado en un formato estándar. Los patrones de diseño soportan el proceso de diseño, proporcionando:

- Una forma abreviada y un lenguaje común para referirse a problemas comunes del par problema de diseño / solución.
- Un medio para construir un repositorio de conocimientos de la profesión y fomentar la reutilización de sus mejores prácticas (31).

Patrones GRASP (Patrones generales de software para asignar responsabilidades)

Los patrones GRASP, acrónimo en inglés de *General Responsibility Assignment Software Patterns*, describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Estos comprenden un total de nueve, los cuales son: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador, Polimorfismo, Fabricación pura, Indirección y No hables con extraños. De ellos se emplearon para la realización del diseño del componente.los siguientes patrones: Creador, Controlador, Experto, Bajo acoplamiento y Alta cohesión con el propósito de contribuir a una mayor robustez y flexibilidad del sistema (31).

Bajo acoplamiento

El bajo acoplamiento es un principio a tener en cuenta durante las decisiones de diseño. Constituye un patrón evaluativo para que el diseñador juzgue sus decisiones de diseño. Se pone de manifiesto cuando se evidencian pocas dependencias entre las clases. Soporta de esta manera el diseño de clases más independientes, minimizando el impacto de los cambios, y convirtiéndolas en más reutilizables. No puede considerarse independiente de patrones como Experto o Alta cohesión, sino que más bien se debe incluir como parte de los principios del diseño que intervienen en la decisión de asignar responsabilidades. El uso de este patrón se pone de manifiesto en la solución al estar cada clase acoplada a las clases estrictamente necesarias, por ejemplo: La clase RegistroretencionController realiza una integración con el componente Trabajador y no conoce la existencia de la clase BuscarTrabajadorModel del componente

Trabajador, que es la que contiene los datos que la misma pide mediante el uso del IoC para obtener los datos o colección de datos que se necesiten en el componente Retenciones, logrando una escasa dependencia entre las clases y un aumento en la reutilización.

Alta cohesión

La alta cohesión se pone de manifiesto porque se asignaron responsabilidades a las clases de manera que todos sus métodos tuvieran un comportamiento bien definido, por ejemplo: la responsabilidad de la claseRegistroretencionController es la de acceder a los valores y métodos de las clases NomTiporetenciones y NomTiporetencionesModel.

Experto

Se pone de manifiesto con el uso de clases que poseen responsabilidades específicas a cumplir con relación a la información que manipulan, es decir, cuando una clase contiene toda la información necesaria para realizar la labor que tiene encomendada. En el caso del componente Retenciones este cuenta con clases controladoras, modelos y entidades que tienen funciones concretas de acuerdo con los datos que gestionan. Además, se modeló una clase entidad para cada tabla de la base de datos, facilitando el trabajo específico y directo con el experto en la información. Ejemplo de su uso son las clases, NomTiporetencionesModel y la clase DatRegistroretencionesModel. Estas clases efectuarán los procedimientos para: InsertarTipoRetenciones, Actualizartiporetencion y Eliminartiporetencion cada elemento.

Creador

Este patrón maneja la asignación de responsabilidades relacionadas con la creación de objetos. El principal objetivo de este patrón es encontrar un creador que se debe conectar con el objeto producido en un evento cualquiera. Con su aplicación se ofrece soporte para un bajo acoplamiento lo que supone un menor grado de dependencia con respecto al mantenimiento y al mismo tiempo mayor reutilización. Este patrón se evidencia en las clases del paquete Domain, las que crean los objetos de tipo Doctrine_Query, posibilitando el acceso a la información almacenada a nivel de datos.

Controlador

Asigna la responsabilidad a una clase de manejar mensajes correspondientes a eventos en un sistema. Dicha clase es la encargada de recibir los datos del usuario y enviarlos a las distintas clases. La aplicación

del patrón conlleva a separar la lógica de negocios de la capa de presentación. Esto facilita la centralización de actividades (validaciones, seguridad, entre otras cosas). Si se aplican estos principios, el controlador no realiza las actividades mencionadas sino que las delega en otras clases con las que mantiene un modelo de alta cohesión. El patrón se ejemplifica a través de las clases RegistroretencionController, GestionartiporetencionController, las que tendrán la responsabilidad de manejar los eventos dentro del componente.

2.3.3 Diagramas de clases del diseño.

Un diagrama de clase del diseño es uno de los diagramas estáticos definidos para UML, estos describen la estructura de un sistema mostrando sus clases, métodos y las relaciones existentes entre ellos. Además entre su contenido se encuentran:

- Clases asociadas
- Métodos
- Navegabilidad
- Dependencias

A continuación se muestran los diagramas de clases del diseño pertenecientes al componente Retenciones:

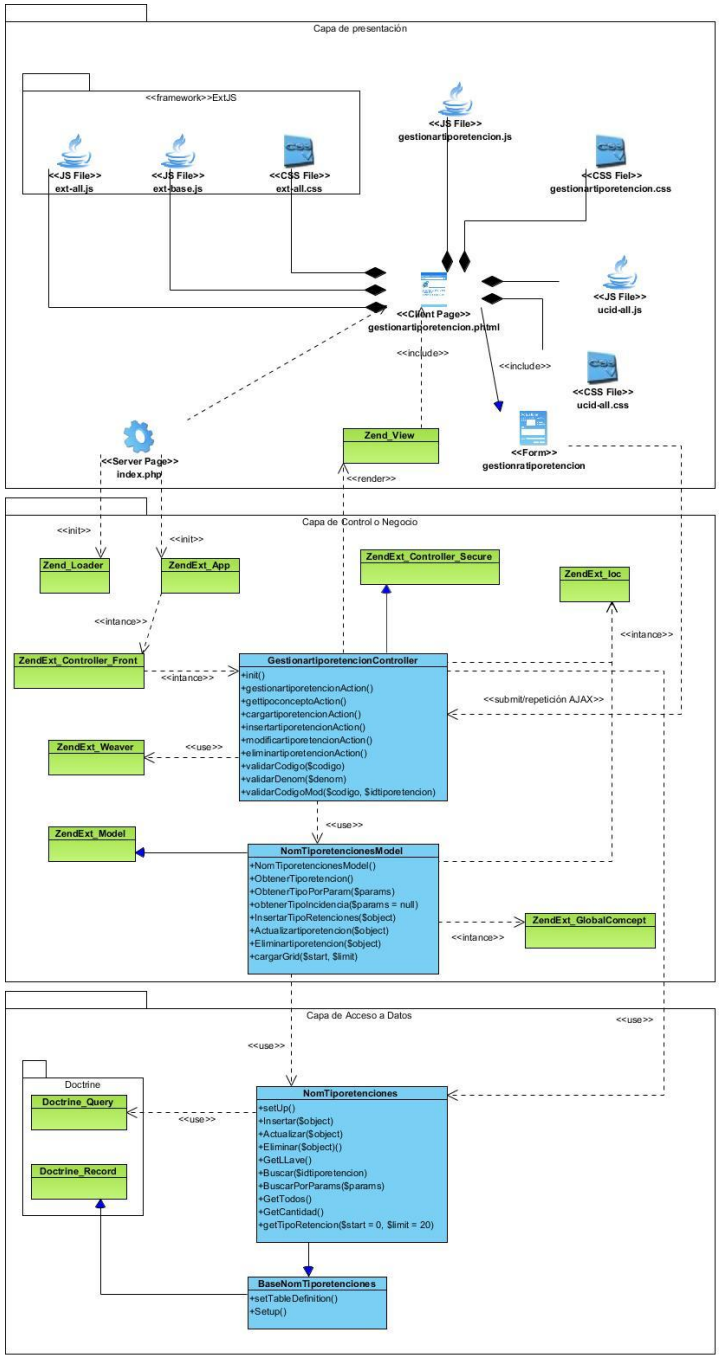

Figura 1 Diagrama de clase del diseño Gestionar Tipo Retención.

En la figura 1 se muestra el diagrama de clases del diseño gestionar tipo de retenciones, en la cual se le da solución a los requisitos funcionales adicionar, modificar, eliminar e imprimir, se representan las

principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a estos requerimientos, donde las clases gestionartiporetencion.js y gestionartiporetencion.phtml conforman la capa arquitectónica de presentación. La clase GestionartiporetencionController solo maneja la comunicación entre la vista y la *model*, mientras que la clase NomTiporetencionesModel es la encargada de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento de los requisitos identificados, y las NomTiporetenciones y BaseNomTiporetenciones son las encargadas del acceso a los datos de las tablas.

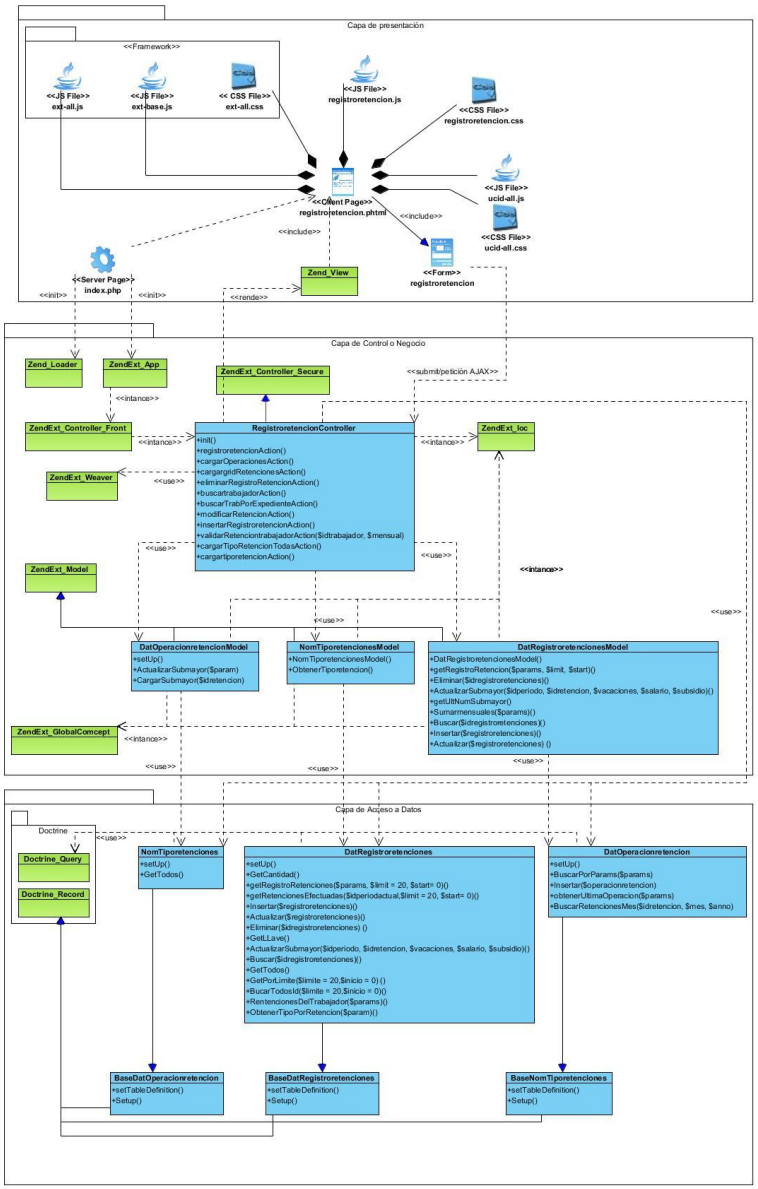

Figura 2 Diagrama de clase del diseño Registro de Retenciones.

En la figura 2 se muestra el diagrama de clases del diseño registro de retenciones, en el cual se le da solución a los requisitos funcionales adicionar, modificar, eliminar, imprimir y listar registro de retenciones, se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a estos requerimientos, donde las clases registroretencion.js y registroretencion.phtml conforman la capa arquitectónica de presentación. La clase RegistroretencionController es la encargada de la comunicación entre la vista y la *model*, las clases DatRegistroretencionesModel, DatOperacionretencionModel y

NomTiporetencionesModel son las encargadas de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento de los requisitos identificados y las clases NomTiporetenciones, DatRegistroretenciones, DatOperacionretencion, BaseDatOperacionretencion, BaseDatRegistroretenciones y BaseNomTiporetenciones son las encargadas del acceso a los datos de las tablas.

2.3.4 Modelo de datos

El modelo de datos es una representación de las tablas existentes en la base de datos así como las relaciones entre ellas. Para el caso del componente Retenciones este contiene 6 tablas donde se muestra de manera general su negocio. La tabla nom_tiporetenciones gestiona los principales tipos de retenciones definiendo si se trata de una retención empresarial o no mediante el campo “empresarial”, lo cual define un tratamiento específico de la retención a la que se asocie. En el caso de la tabla dat_registroretenciones comprende los principales datos de los registros de retenciones que se le asignen a un trabajador, con datos de gran importancia como el saldo a descontar por la nómina y el estado “activo” o no de la misma. La tabla dat_operacionretencion almacena los registros de los descuentos realizados a la retención, constituye el submayor de cada una de esta. Las mencionadas anteriormente constituyen las principales tablas del componente Retenciones, a los que se le añaden las tablas dat_trabajador, que contiene los datos del trabajador, dat_nprocretenciones encargada de registrar temporalmente los valores de cada descuento de la retención durante el procesamiento de la nómina, hasta que esta es contabilizada y se actualiza el submayor de retenciones y finalmente la tabla nom_periodopago es la encargada de registrar el período de cada una de las operaciones de descuento en el submayor de cada retención.(Ver figura 3)

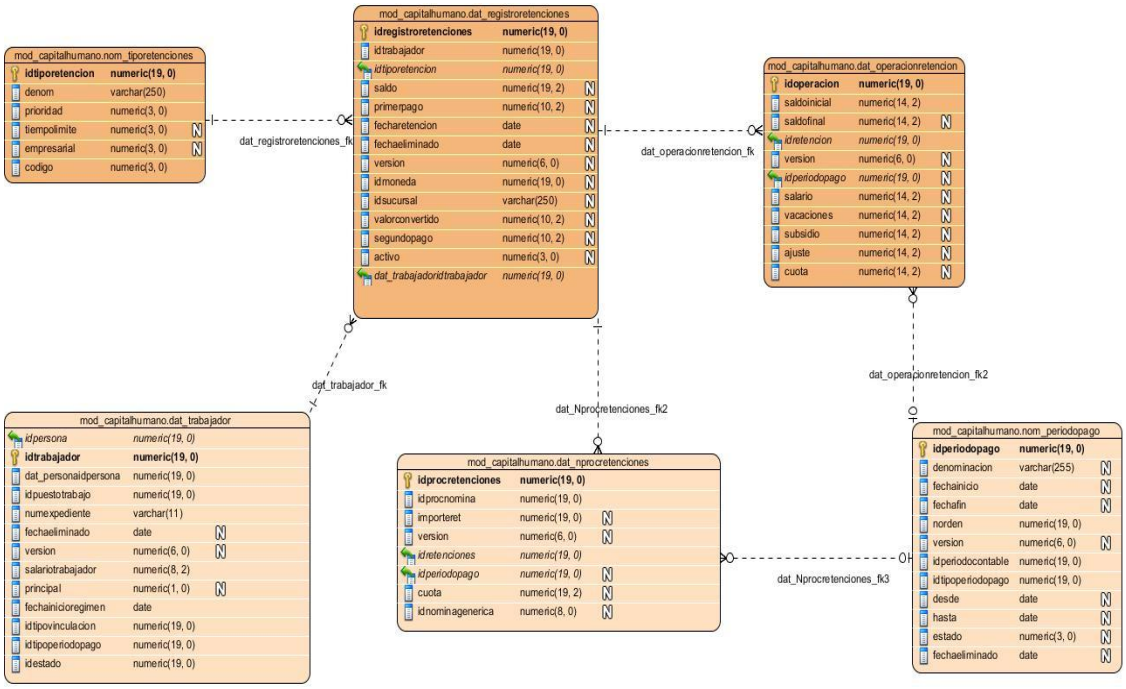

Figura 3 Modelo de datos.

2.4 Implementación

Con la obtención de los artefactos generados en el Diseño, se puede dar paso a la implementación del componente Retenciones como parte del subsistema de Capital humano perteneciente a Cedrux.

2.4.1 Estructura del marco de trabajo.

El marco de trabajo Sauxe fue el utilizado para el desarrollo del componente Retenciones y su unificación con el subsistema de Capital humano como se explicó en el capítulo anterior, a continuación se abordarán sus características fundamentales. Dentro de la carpeta raíz de desarrollo del proyecto se encuentran definidas las carpetas **apps** y **web** que contienen, respectivamente, la lógica de negocio y vistas de los componentes por módulos referentes a cada subsistema que se implementan en Cedrux. En cada una de ellas se localiza una carpeta por subsistema, específicamente dentro del subsistema **Capital Humano** se ubica **Estimulación Moral y Material** con sus componentes (ver figura 4).

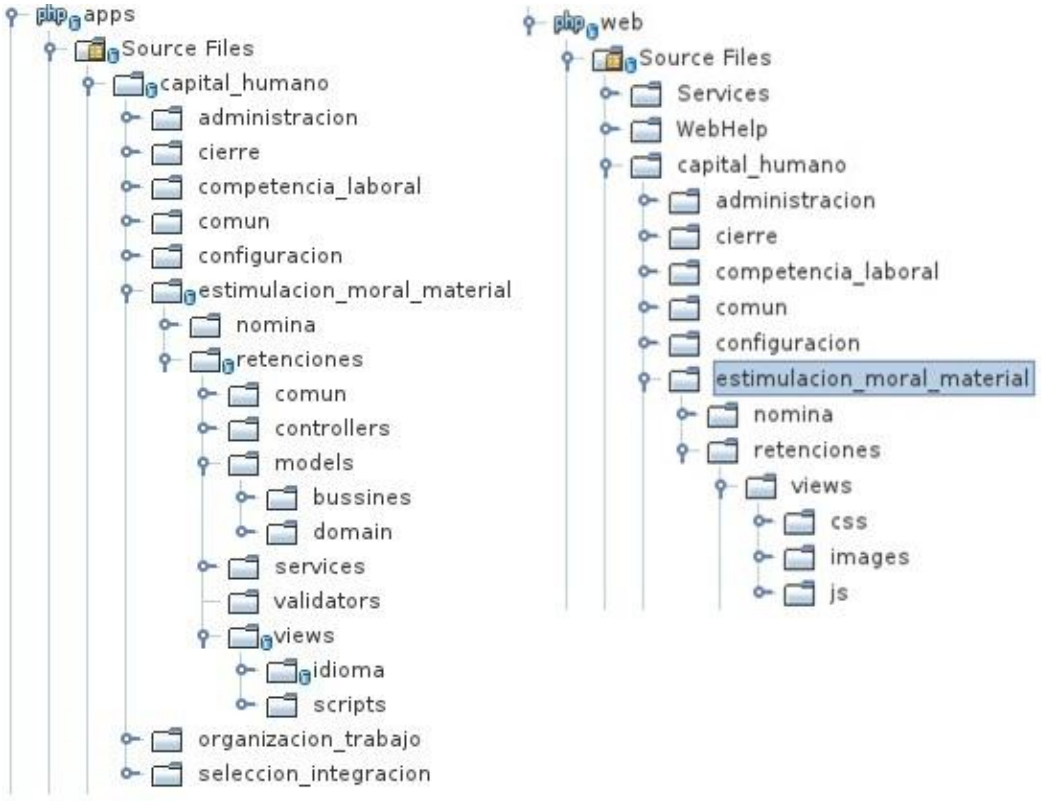

Figura 4 Estructura del marco de trabajo: Capital humano.

Contenido en el ***apps***, al mismo nivel de los subsistemas, se sitúa el paquete ***común*** que incluye los recursos asociados a la aplicación y los ficheros de tipo xml, donde los componentes especifican sus configuraciones propias y el código de dichos archivos está estructurado por módulos.

El componente Retenciones está estructurado como se muestra a continuación (ver figura 5):

Figura 5 Estructura del marco de trabajo: Retenciones.

Controllers: En esta carpeta se almacenarán las clases controladoras encargadas de la gestión de las funcionalidades del componente y el flujo de información entre las vistas y las modelos. Su responsabilidad principal es comunicar las interfaces de usuario con la lógica de negocio de la aplicación.

Models: Contiene la carpeta ***bussines***, en la cual se programa toda la lógica de negocio del componente en cuestión, y almacena además la carpeta ***domain***, donde se encuentran las clases que gestionan el acceso a los datos, o la interacción con la base de datos (ver figura 6).

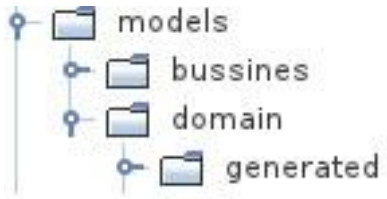

Figura 6 Estructura del marco de trabajo: Models.

Generated: Estas clases heredan de ficheros base definidos como clases php, a estas clases se le agrega la palabra base.

Services: Este paquete contiene clases que servirán de puente entre las funcionalidades que brinda el componente y las clases externas que las solicite, permitiendo la integración con otros y controlando el flujo de datos salientes del componente.

Validators: Esta carpeta contendrá las clases de tipo php que van a realizar acciones de validación en el componente, como las precondiciones que se deben cumplir antes de que un determinado método sea ejecutado.

Views: En esta carpeta se recopilan los ficheros que van a unir la capa de presentación con la clase controladora, estos ficheros se agrupan en 2 carpetas. (Ver figura 7)

Figura 7 Estructura del marco de trabajo: Views del apps

Idioma: Contiene ficheros de tipo json que recopilan etiquetas para la gestión de los mensajes vinculados a la presentación.

Scripts: En este directorio se incluyen todas las vistas, para ello se crea una carpeta para cada clase controladora y dentro se incluye la vista o script, archivos de extensión phtml donde se especifica el título de la página que se gestiona y se carga el archivo js que mostrará la presentación.

Views: En esta carpeta se recopilan los ficheros que van a gestionar la capa de presentación, estos ficheros se agrupan en 2 carpetas. (Ver figura 8)

Figura 8 Estructura del marco de trabajo: Views de la web

Css: En este directorio se encuentran las plantillas y estilos para el diseño del componente.

Js: Es la carpeta donde se incluyen las clases javascript, al igual que en la carpeta de scripts por cada clase controladora existe una carpeta que tendrá incluido el fichero js. El fichero js no es más que un documento con la extensión .js donde se escribirá el código correspondiente a la capa de presentación, las interfaces de usuario.

2.4.2 Diagrama de componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. A continuación se presenta el diagrama de componentes de Retenciones, en el que se puede observar sus relaciones con los demás componentes del subsistema Capital humano como por ejemplo con: Persona, Trabajador y Nómina (Ver figura 9).

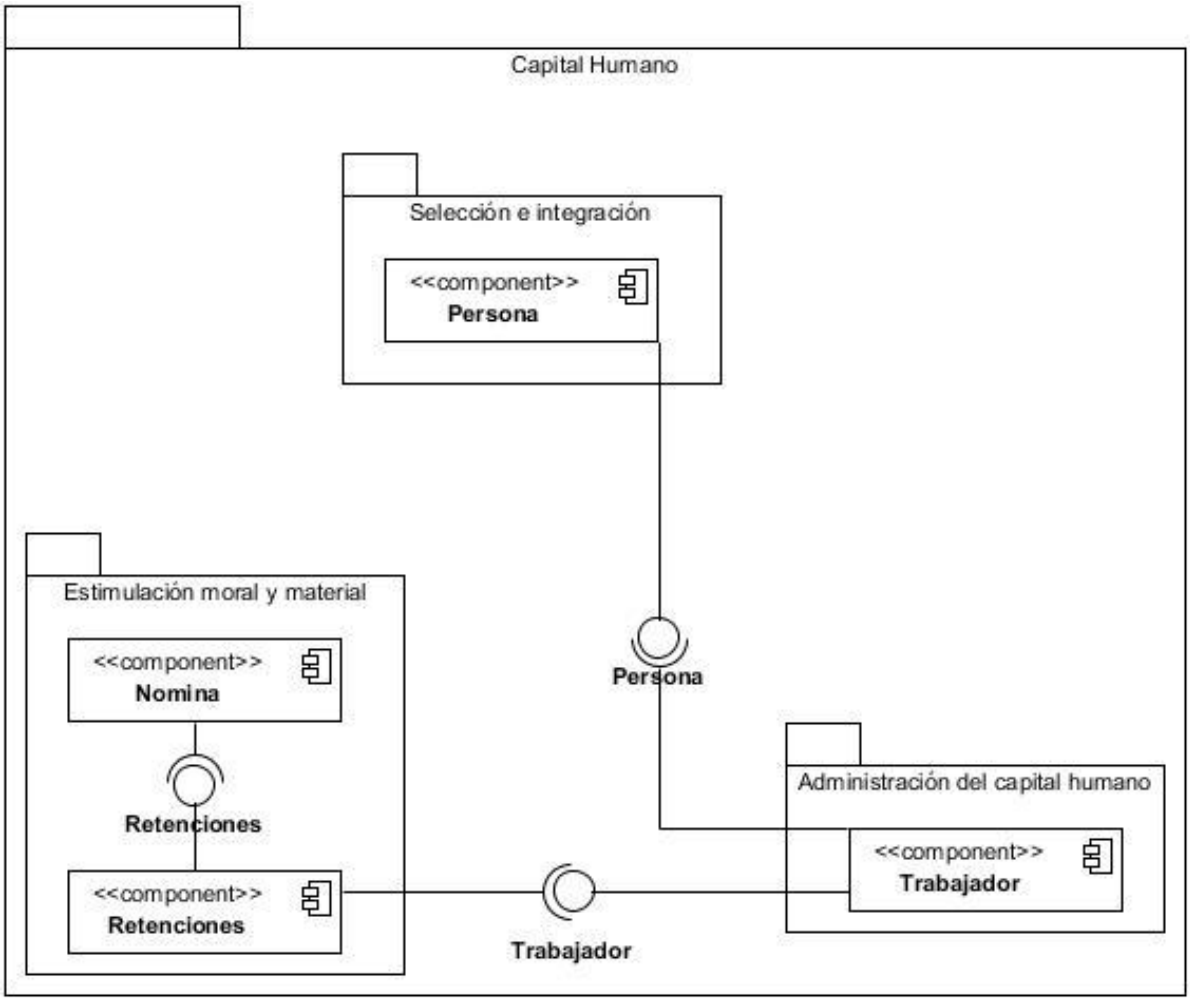

Figura 9 Diagrama de componentes.

2.4.3 Integración entre componentes.

La arquitectura en 3 capas: presentación (view), negocio (controller) y acceso a datos (models), consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, a través de los diferentes elementos que la componen. Consta de 4 nodos de integración: vista-controlador, controlador-modelo, modelo-framework Doctrine y Doctrine-base de datos. La comunicación entre las capas dentro de un mismo componente se realiza mediante llamadas a métodos o eventos de forma directa.

Entre diferentes módulos y componentes, la integración se basa en el patrón Inversión de Control (IoC) y se realiza a través de un componente incluido en el framework Zend_Ext, que permite operar sobre

distintos esquemas en la base de datos realizando las transacciones adecuadas. En dicho componente se define el fichero ioc.xml que contiene la ubicación de los componentes y los servicios que ofrecen las clases Services correspondientes. De esta manera, la puerta de entrada de cada componente es el paquete de las clases de servicios que buscan en los modelos o entidades las funcionalidades requeridas.

Inversión de control (*Inversion of Control* en inglés, IoC) es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones. IoC se hace necesario para gestionar las dependencias entre subsistemas y el framework (14).

Con la integración se persigue obtener una forma eficiente y flexible de combinar recursos internos o externos de los subsistemas usando:

IoC Interno: para la integración entre componentes de un subsistema.

IoC Externo: para la integración entre subsistemas.

Algunos servicios que ofrece el módulo Retenciones son:

ActualizarSubmayor: Servicio del componente Retenciones para el módulo Nómina del subsistema Capital humano que permite actualizar el submayor de cada una de las retenciones de los trabajadores.

RentencionesDelTrabajador: Servicio del componente Retenciones para el módulo Nómina del subsistema Capital humano que permite obtener las retenciones activas que tiene el trabajador.

2.4.4 Diagrama de despliegue.

Un diagrama de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). Estarán formados por instancias de los componentes software que representan manifestaciones del código en tiempo de ejecución (los componentes que solo sean utilizados en tiempo de compilación deben mostrarse en el diagrama de componentes).

Un diagrama de despliegue es un grafo de nodos unidos por conexiones de comunicación. Un nodo puede contener instancias de componentes software, objetos, procesos (caso particular de un objeto). El

despliegue del componente Retenciones se rige por la especificación dada para el subsistema Capital humano (ver figura 10 y 11). Se detallan los protocolos usados en los enlaces entre los distintos componentes hardware, sin embargo, no se indican los puertos ya que pueden variar en dependencia la configuración de la red de la entidad donde se implante. A continuación se muestran los posibles escenarios con los requerimientos de software.

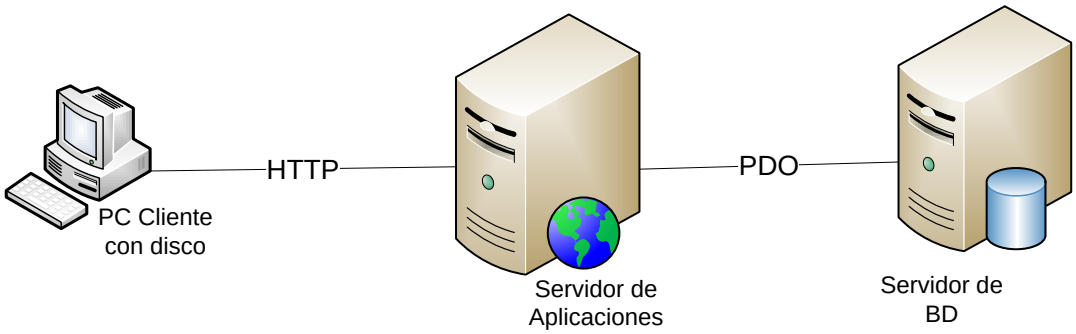

Figura 10 Diagrama de despliegue de escenario para PC cliente con disco.

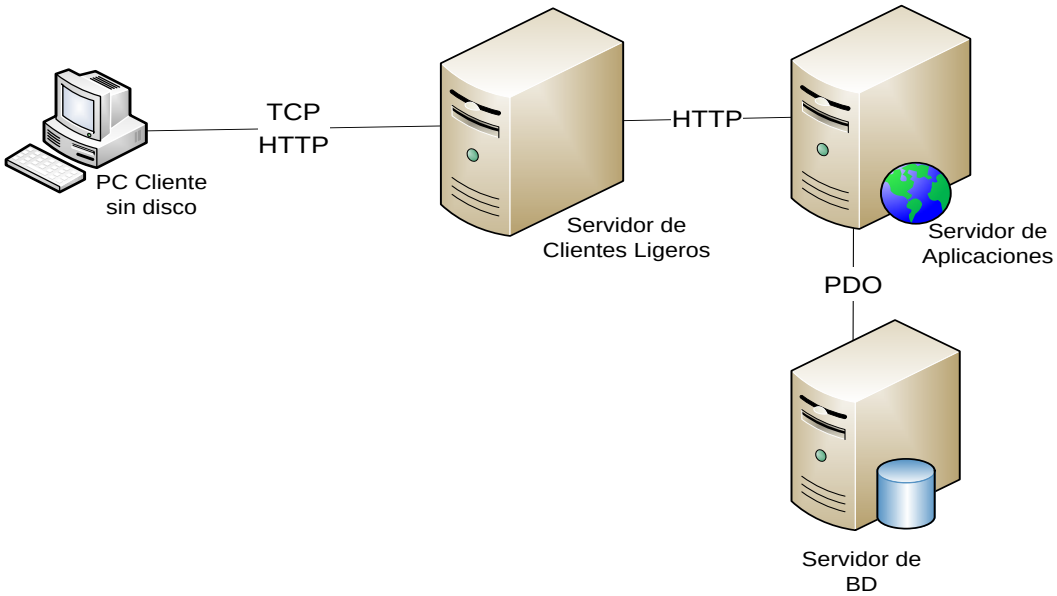

Figura 11 Diagrama de despliegue de escenario para PC cliente sin disco.

Requerimientos de software de cada uno de los componentes hardware que forman parte de los diagramas anteriores:

Servidores

Servidor de Aplicaciones Web

- Sistema Operativo: Ubuntu Server
- Servidor Web: Apache 2.0
- Librerías Adicionales: PHP 5

Servidor de Base de Datos

- Sistema Operativo: Ubuntu Server
- Sistema Gestor de Base de Datos: PostgreSQL 8.3.8

Servidor de Clientes Ligeros

- Sistema Operativo: Nova Server
- Navegador Web: Mozilla Firefox 2.2 o superior
- Herramientas Ofimáticas
- Visualizador de ficheros pdf
- Herramienta de administración de clientes ligeros

Clientes

PC Cliente con disco duro

- Sistema Operativo: Linux o Windows
- Navegador Web: Mozilla Firefox v2.2 ó superior
- Herramientas Ofimáticas
- Visualizador de ficheros pdf

PC Cliente sin disco duro

- Todo se instala en el servidor de clientes ligeros.

2.4.5 Interfaces

Como resultado de la implementación del componente Retenciones se muestran a continuación algunas de sus interfaces:

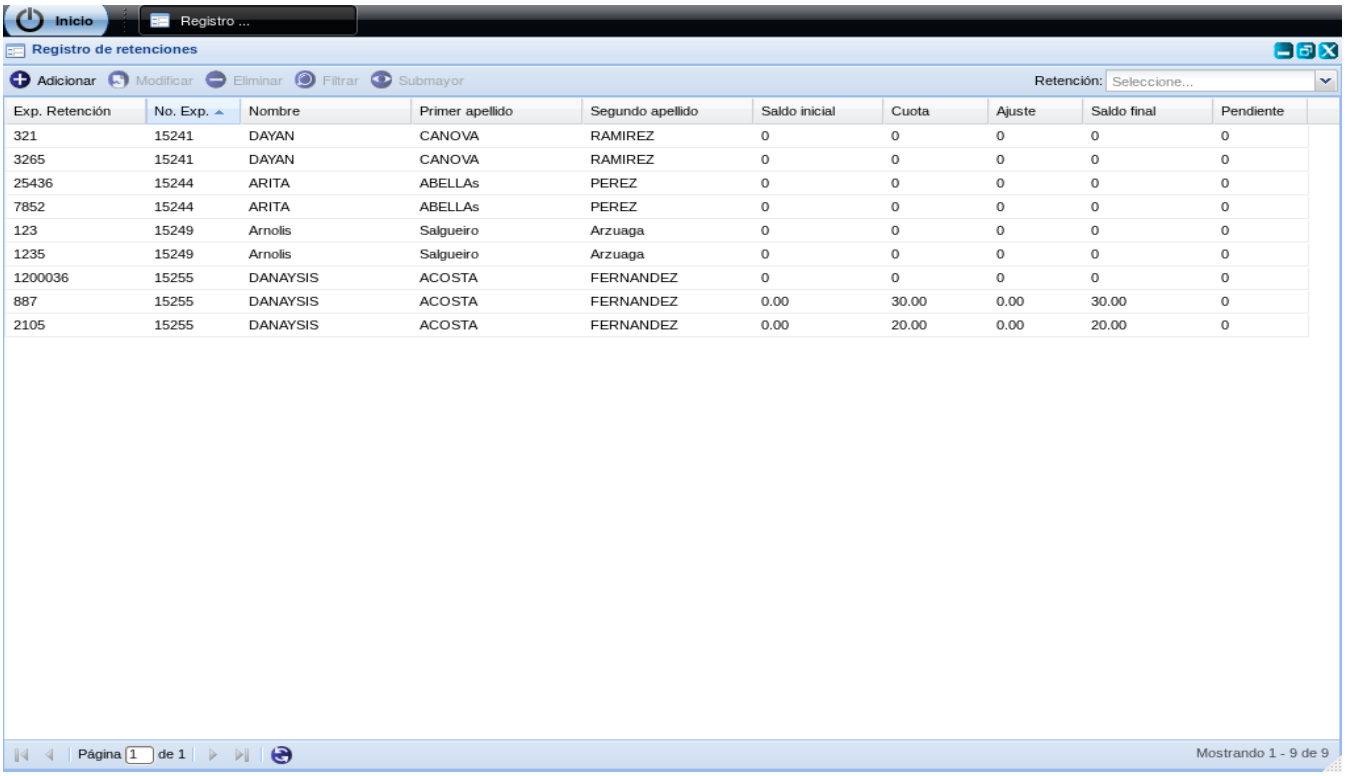

Figura 12 Interfaz del gestionar registro de retención.

En la figura 12 se muestra la interfaz principal del escenario gestionar registro de retención, que ofrece las opciones de Adicionar, Modificar, Eliminar y consultar Detalles del submayor de una retención. También permite Filtrar por trabajador y por tipo de retención. Para todas las retenciones que se listan se muestran sus principales campos como: Nombre y apellidos del trabajador a la que se encuentra asignada, su Saldo inicial, Saldo final, Cuota y Pendiente del último registro de pago insertado en su submayor. Además se muestran otros datos como el número de expediente de la retención y el del expediente del trabajador.

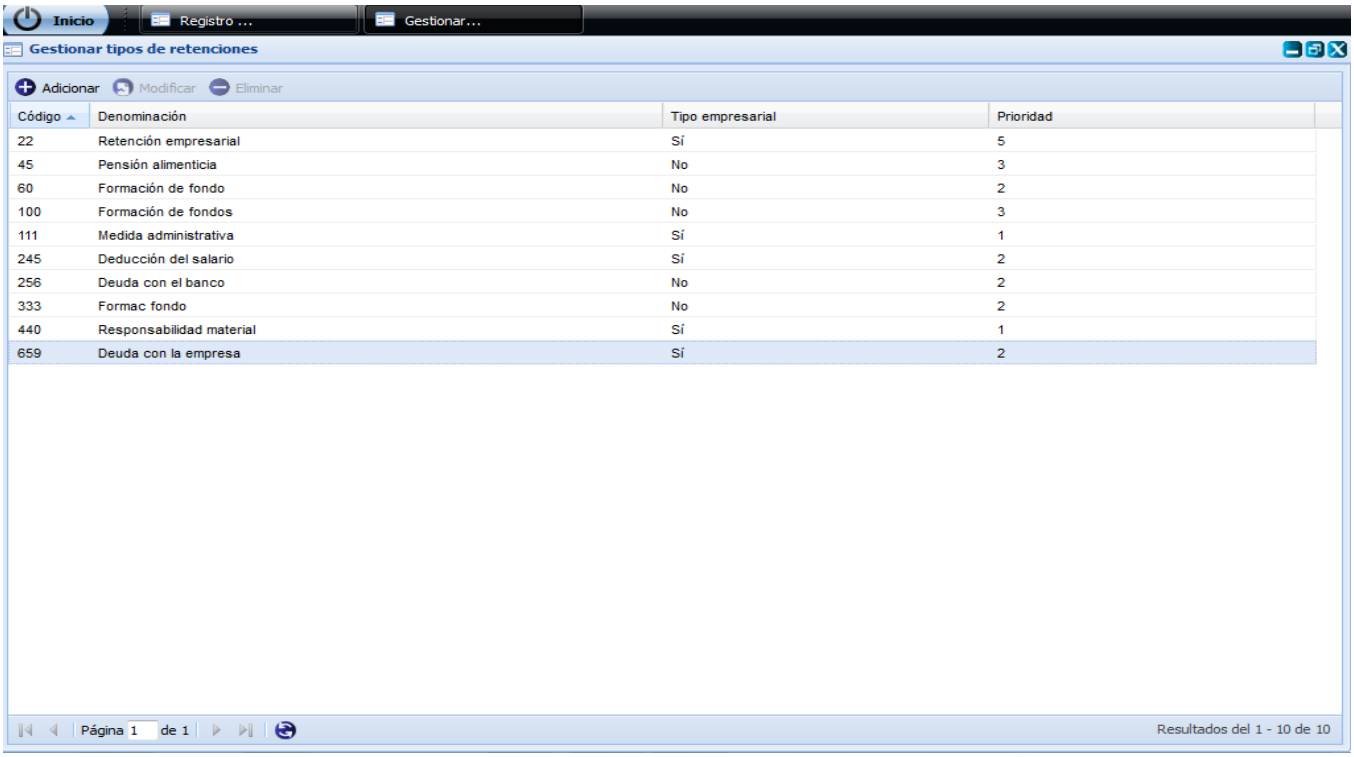

Figura 13 Interfaz del gestionar tipo de retención

En la figura 13 se muestra la interfaz principal del escenario gestionar tipo de retención, que ofrece las opciones de Adicionar, Modificar y Eliminar tipo de retención. Para cada tipo de retención que se lista se muestra sus principales campos como: Código, Denominación, si es de Tipo empresarial o no y su prioridad.

2.5 Conclusiones del capítulo.

Con el desarrollo de este capítulo se obtuvo la solución del módulo Retenciones del subsistema Capital humano del Sistema Integral de Gestión de Entidades Cedrux, con el objetivo de contribuir al mejoramiento de los procesos en las entidades cubanas, brindando de este modo un impulso decisivo a la informatización de la sociedad. La descripción del diseño de la solución arquitectónica facilitó la exitosa implementación e integración con el resto de los módulos del propio subsistema.

Capítulo 3: Validación de la solución

3.1 Introducción

En este capítulo se plasma la validación de la solución propuesta mediante las pruebas de caja blanca, caja negra y la validación del diseño, así como los resultados obtenidos en cada una de ellas luego de su aplicación. Estas pruebas son ejecutadas con el objetivo de avalar el cumplimiento de las exigencias del cliente y la calidad del sistema.

3.2 Evaluación del modelo de diseño propuesto.

Son varios los puntos de vista relacionados con la calidad del software. Desde metodologías hasta las distintas normas de calidad, que pueden estar orientados tanto a los procesos de desarrollo como a los productos de software. No es objetivo de este trabajo abundar sobre los temas de calidad, pero sí desarrollar una evaluación del diseño obtenido en la solución propuesta de diseño de software al componente Retenciones del subsistema Capital Humano integrado al sistema Cedrux.

Para la evaluación del diseño propuesto se hizo un estudio de la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objetos referenciadas por Pressman, en el mismo se abarcan atributos que permiten medir la calidad del diseño propuesto.

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

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

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

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño del componente Retenciones y su relación con los atributos de calidad definidos son las siguientes:

- Tamaño Operacional de Clase (TOC).
- Relaciones entre Clases (RC).

3.2.1 Tamaño operacional de clase (TOC)

El tamaño operacional de clase, está dado por el número de métodos asignados a una clase.

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 2 Métrica Tamaño Operacional de Clase (TOC).

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

Atributo	Categoría	Criterio
Responsabilidad	Baja	<= Prom.(Promedio)
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.

Complejidad de implementación	Baja	<= Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2* Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	<= Prom.

Tabla 3 Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

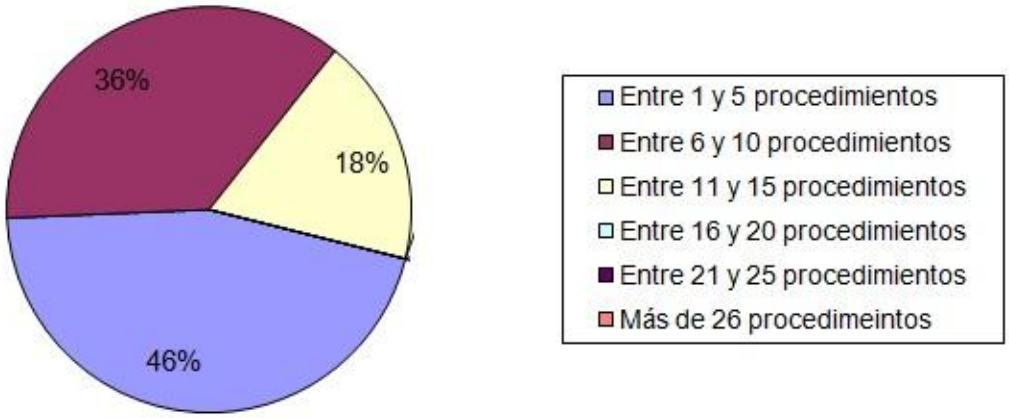

Figura 14 Resultados obtenidos de la aplicación de la métrica TOC en los instrumentos agrupados en los intervalos definidos.

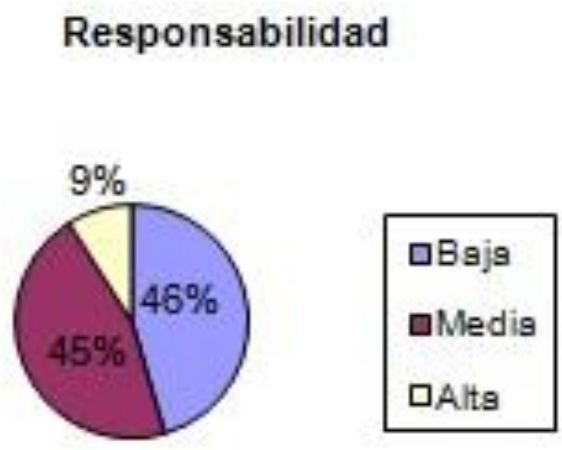

Figura 15 Resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

Figura 16 Resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación.

Figura 17 Resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Haciendo un análisis de los resultados obtenidos para los atributos de la métrica TOC en la evaluación del instrumento, se puede observar que la mayoría de las clases que conforman el sistema para los atributos responsabilidad y complejidad están dentro de la categoría Media y Baja para un 91% del total, mientras que el atributo Reutilización cuenta con igual por ciento en las categorías Alta y Media mostrando así que el componente cuenta con una elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

3.2.2 Relaciones entre clases (RC)

Las relaciones entre las clases, está dado por el número de relaciones de uso de una clase con otras.

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

Tabla 4 Métrica Relaciones entre Clases

Resultados del instrumento de evaluación de la métrica Relaciones entre clases

Atributo	Categoría	Criterio
Acoplamiento	Ninguna	0
	Baja	1
	Media	2
	Alta	>2
Complejidad de mantenimiento	Baja	<= Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Cantidad de pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2* Prom.
	Media	Entre Prom. y 2* Prom.

Tabla 5 Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica.

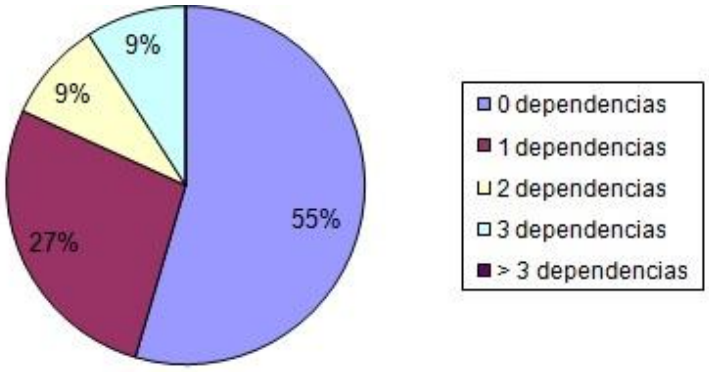

Figura 18 Resultados obtenidos de la aplicación de la métrica RC en los instrumentos agrupados en los intervalos definidos.

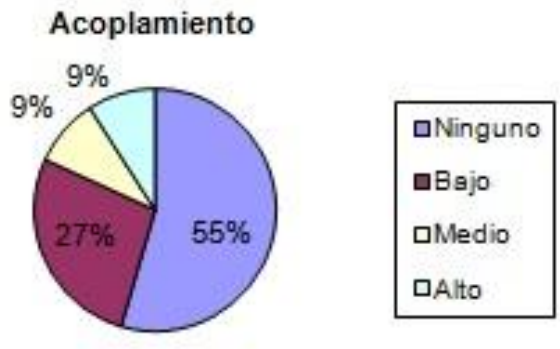

Figura 19 Resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

Figura 20 Resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento.

Figura 21 Resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

Figura 22 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 componente Retenciones tienen una calidad buena pudiéndose observar que el 100% de las clases posee menos de 3 dependencias de otras clases. Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que el diseño propuesto para el componente Nómina se encuentra dentro de los niveles requeridos.

3.3 Pruebas de software

La importancia de los costos que están asociados a los errores promovió a la definición y aplicación de pruebas detalladas y bien planificadas al componente Retenciones del subsistema Capital Humano integrado al sistema Cedrux. Las mismas fueron un elemento fundamental para determinar la calidad de la solución propuesta, representando una revisión final de las especificaciones, del diseño y de la codificación. Las pruebas, por su gran importancia se llevan a cabo durante todo el ciclo de vida del producto, su mayor punto de desarrollo se encuentra en la etapa de implementación.

Dentro de las pruebas se destacan principalmente dos tipos de pruebas:

- Caja Negra.
- Caja Blanca.

3.3.1 Pruebas de caja negra

Las pruebas de caja negra son las 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. (Ver figura 23)

Estas pruebas permiten encontrar:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación (32).

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

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

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

Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones (32).

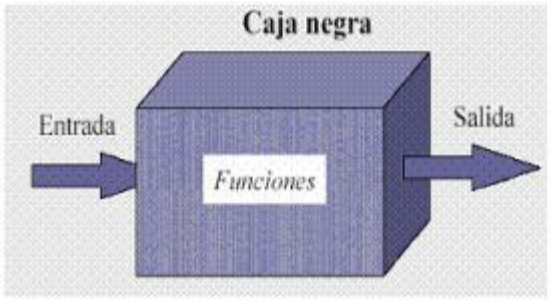

Figura 23 Representación de pruebas de caja negra

Para verificar que la aplicación se comporta según los requerimientos establecidos por el cliente, se diseñan once casos de pruebas usando el método de caja negra, con la técnica de partición de equivalencia. A continuación se especifica el caso de prueba para el requisito Adicionar tipo de retención.

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema Capital humano/Estimulación moral y material/Configuración/Gestionar tipos de retenciones.

Requisitos a probar

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar tipo de retención.	El sistema debe permitir adicionar tipo de retención.	EP 1.1: Adicionar tipo de retención introduciendo datos válidos y presionando el botón **Aceptar**.	– Se presiona el botón **Adicionar**. – Se introducen los datos del tipo de retención correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
	El sistema debe permitir adicionar uno o varios Tipos de retenciones.	EP 1.2: Adicionar tipo de retención introduciendo datos válidos presionando el botón **Aplicar**.	– Se presiona el botón **Adicionar**. – Se introducen los datos del tipo de retención correctamente. – Se presiona el botón

			Aplicar. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**. – Se limpian los campos de la ventana Adicionar Tipo de nómina y esta permanece abierta.
	El sistema debe obviar los caracteres no permitidos en el sistema introduciendo solo los permitidos.	EP 1.3: Adicionar tipo de retención introduciendo datos inválidos.	– Se introducen los datos inválidos del tipo de retención. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
	El sistema no debe permitir adicionar el Tipo de retención.	EP: 1.4 Adicionar tipo de retención introduciendo un código existente.	– Se presiona el botón **Adicionar**. – Se introducen los datos del tipo de retención con un código ya existente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
	– El sistema debe señalar el o los campos vacíos en	EP 1.5: Adicionar tipo de retención dejando campos vacíos.	– Se presiona el botón **Adicionar**. – Se introducen los datos

	rojo y una vez que se sitúe el cursor sobre el o los campos en cuestión se debe visualizar el siguiente globo de información "Este campo es obligatorio.".		dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
	El sistema debe cancelar la adición del Tipo de nómina.	EP 1.6: Cancelar.	– Se presiona el botón **Adicionar**. – Se introducen o no los datos del tipo de retención. – Se presiona el botón **Cancelar**.

Tabla 6 Descripción del caso de prueba para el requisito Adicionar tipo de retención.

Descripción de variable

No	Nombre de campo	Tipo	Válido	Inválido	Requerido
1	Código	Campo de texto.	Números	Caracteres especiales y letras.	Sí
2	Denominación	Campo de texto	Letras, números, comilla simple y espacios hasta 50 caracteres.	Caracteres extraños.	Sí

3	Prioridad	Campo de selección (No editable).	NA	NA	Sí
4	Empresarial	Campo de selección (No editable).	NA	NA	No
5	Tiempo Límite	Campo de selección (No	NA	Vacío	Sí

Tabla 7 Descripción de variables del caso de prueba para el requisito Adicionar tipo de retención.

Juegos de datos a probar

Id del escenario	Escenario	Código	Denominación	Prioridad	Empresarial	Tiempo Límite	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar tipo de retención introduciendo datos válidos.	V(122)	V(Retención 1)	V(1)	NA	V(Si)	El sistema adiciona el tipo de retención y muestra el mensaje de información: “Se ha adicionado el tipo de retención satisfactoriamente.”. El sistema cierra la interfaz.	NA
EP 1.2	Adicionar tipo de retención introducie	V(122)	V(Retención 1)	V(1)	NA	V(Si)	El sistema adiciona el tipo de retenciones y muestra el mensaje de información: “Se	

	ndo datos válidos presionando el botón **Aplicar**.						ha adicionado el tipo de retención satisfactoriamente.”. El sistema limpia los campos del formulario y mantiene la interfaz abierta.	NA
EP 1.3	Adicionar tipo de retención introduciendo datos inválidos.	I(A22%^$#)	V(Retención 1)	V(1)	NA	V(No)	El sistema no permite la inserción de caracteres inválidos en este campo. El sistema mantiene la interfaz abierta.	
		V(122)	I(Retención 1^$#)	V(1)	NA	V(No)		
		V(122)	V(Retención 1)	I(1%^$#)	NA	V(No)		NA

EP 1.4	Adicionar tipo de retención introduciendo un código existente.	V(122)	V(Retención 1)	V(1)	NA	V(Si)	El sistema muestra el mensaje “Ya existe un tipo de retención con ese código.” El sistema mantiene la interfaz abierta.	NA
EP 1.5	Adicionar tipo de retención dejando campos vacíos.	I(Vacío)	V(Retención 2)	V(1)	NA	V(No)	El sistema muestra el mensaje “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.	NA
		V(122	I(Vacío)	V(4)	NA	V(Si)		
		V(122)	V Retención 2)	I(Vacío)	NA	V(Si)		
		V(122)	V Retención 2)	V(4)	NA	I(Vacío)		
		I(Vacío)	I(Vacío)	I(Vacío)	NA	I(Vacío)		
EP 1.6	Cancelar.	NA	NA	NA	NA	NA	El sistema cierra la interfaz sin realizar ninguna operación.	NA

Tabla 8 Datos de prueba del caso de prueba para el requisito Adicionar tipo de retención.

Las pruebas de caja negra aplicadas al componente fueron desarrolladas primeramente por el departamento de calidad del centro CEIGE. Estas se realizaron en tres iteraciones, donde finalmente se comprobó en la tercera iteración que el componente estaba libre de no conformidades culminando así la fase de pruebas internas. El próximo paso para la liberación se está llevando a cabo por CALISOFT (Centro de calidad para soluciones informáticas), encontrándose actualmente en la segunda iteración, mostrando hasta el momento los resultados que se presentan en la Tabla 9. Todos los casos constituían no conformidades no significativas.

Agrupación de requisitos	**No Conformidades**	
	Iteración 1	**Iteración 2**
Gestionar tipo de retención	1	2
Registro de retención	3	0
TOTAL	**4**	**2**

Tabla 9 Resultado de las iteraciones de las pruebas de caja negra.

3.3.2 Pruebas de caja blanca

Las pruebas de la caja blanca se realizan sobre las funciones internas de un módulo en concreto, están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos, pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos y comprobación de bucles. (Ver figura 24)

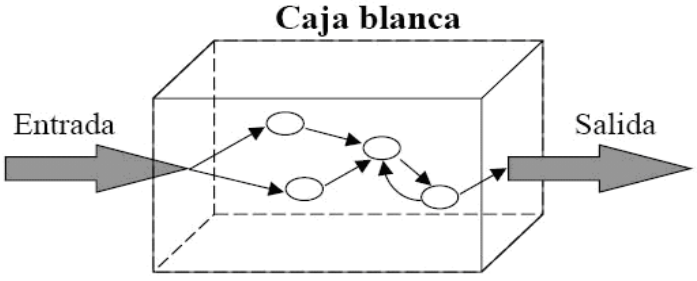

Figura 24 Representación de pruebas de Caja blanca.

De las pruebas anteriormente mencionadas al sistema desarrollado se le aplicaron pruebas de caja negra, realizadas por el departamento de calidad del proyecto y las pruebas de caja blanca por los desarrolladores. En el marco de las pruebas de caja blanca, la técnica que se utilizó fue la de prueba del camino básico. Esta prueba 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, además se garantiza que durante la prueba, en los casos de prueba obtenidos a través del camino básico se ejecute cada sentencia del programa por lo menos una vez.

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

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

Para construir el grafo se debe tener en cuenta la notación para las instrucciones. (Ver figura 25)

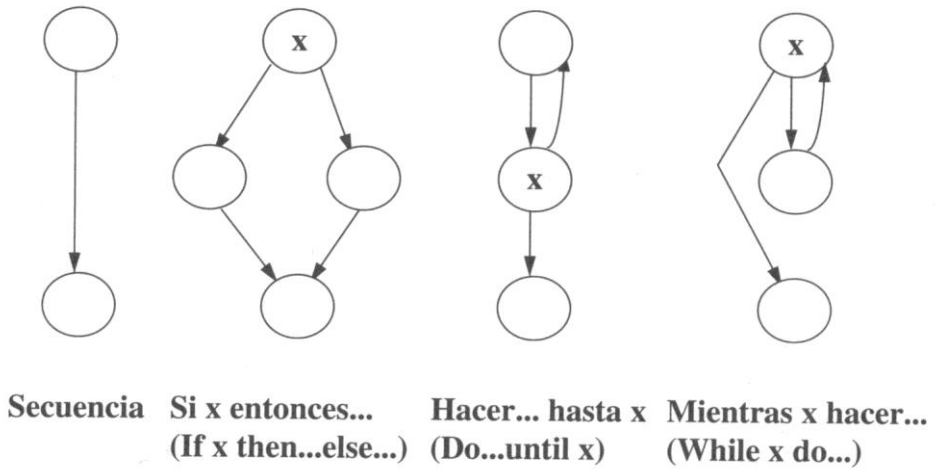

Figura 25 Notación de grafos de flujo.

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

Componentes del grafo de flujo:

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

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

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

Para realizar la técnica de prueba de caja blanca, específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el ActualizarSubmayor (). (Ver figura 26)

```
public function ActualizarSubmayor($param) {
   $idperiodo = $param->idperiodopago;//1
   $idretencion = $param->idretencion;//1
   $vacaciones = $param->vacaciones;//1
   $salario = $param->salario;//1
   $subsidio = $param->subsidio;//1
   $cuota = $param->cuota;//1
   $p->idretencion =$idretencion;//1
   $verRenteciones = new DatOperacionretencion();//1
   $operaciones = $verRenteciones->BuscarPorParams($p);//1
   if(count($operaciones)==0)//2
   {
      $param->saldoInicial = 0;//3
      $param->saldoFinal = $param->saldoInicial + $param->vacaciones+$param->salario+$param->subsidio;//3
      $param->ajuste =0;//3
    return $res = $verRenteciones->Insertar($param);//4
   }//5
   else//6
   {
      $ultimaoperacion = $verRenteciones->obtenerUltimaOperacion($p);//7
       $param->saldoInicial= $ultimaoperacion [0]['saldofinal'];//7
        $param->saldoFinal = $param->saldoInicial + $param->vacaciones+$param->salario+$param->subsidio;//7
        $param->ajuste =0;//7
      return $res =  $verRenteciones->Insertar($param);//8
   }//9
  }//10
```

Figura 26 Representación del método ActualizarSubmayor ().

Consecutivamente se construye el grafo de flujo asociado al código anterior. (Ver figura 27)

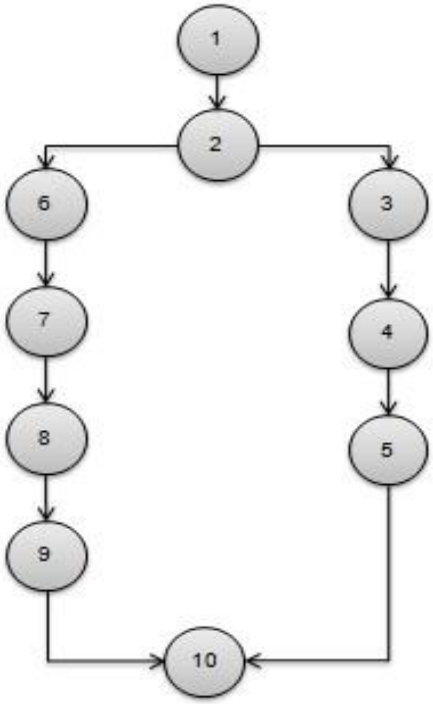

Figura 27 Grafo de flujo asociado al algoritmo ActualizarSubmayor ().

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

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

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

V (G) = (A – N) + 2

V (G) = (10 – 10) + 2

V (G) = 2

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

Se puede usar también:

V (G) = P + 1

V (G) = 1+ 1

V (G) = 2

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

V (G) = R

V (G) = 2

Siendo R la cantidad total de regiones, para cada fórmula V (G) representa el valor del cálculo.

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 2, de manera que existen dos posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

Camino básico #1:

1 – 2 – 3 – 4 – 5 – 10

Camino básico #2:

1 – 2 – 6– 7 – 8 – 9 – 10

A continuación se ejecutan los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. En este caso se diseñaron dos casos de prueba (ver tabla 5y 6). Para su correcta ejecución se deben tener en cuenta lo siguientes parámetros:

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

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

Entrada: Se muestran los parámetros que entran al procedimiento.

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

Descripción	El dato de entrada es un stdClass
Condición de ejecución	El arreglo $operaciones no tiene operaciones.
Entrada	$idperiodo =900033; $idretencion = 90000000084; $vacaciones =0; $salario =78; $subsidio = 0; $cuota =0;
Resultados Esperados	Se actualiza el submayor de retenciones con una operación. Se inserta una nueva operación con $saldoInicial == 0 y el $saldoFinal es igual a la suma de $vacaciones, $saldoInicial, $salario, $subsudio y $cuota.

Tabla 10 Resultado del primer camino básico.

Descripción	El dato de entrada es un stdClass
Condición de ejecución	El arreglo $operaciones tiene operaciones.
Entrada	$idperiodo =900033; $idretencion = 90000000084; $vacaciones =0; $salario =78; $subsidio = 0;$cuota =0;
Resultados Esperados	Se actualiza el submayor de retenciones, con otra operación. Se inserta un nueva operación con $saldoInicial igual al $saldoFinal de la última operación, $saldoFinal es igual a la suma de $vacaciones, $saldoInicial, $salario, $subsudio y $cuota.

Tabla 11 Resultado del segundo camino básico.

3.4 Conclusiones del capítulo.

En el capítulo se hace una valoración crítica del diseño propuesto mediante el uso de las métricas (TOC y RC) las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes. Se aborda acerca de las pruebas de software, haciéndose énfasis en las pruebas de caja blanca, para efectuar las revisiones al código y caja negra, mostrando cómo respondían adecuadamente a los requisitos funcionales, garantizando la satisfacción plena de las necesidades reales de los usuarios y demandas del cliente, demostrándose con ello la calidad y eficiencia del componente Retenciones del subsistema Capital Humano del sistema Cedrux.

Conclusiones generales

En la realización de este trabajo se logró darle cumplimiento a los objetivos específicos a través del desarrollo claro y definido de las tareas propuestas, desarrollándose una solución para la gestión de las retenciones mediante el subsistema Capital humano del Sistema Integral de Gestión Cedrux. Lo anterior se ve demostrado a través de:

- La realización de un estudio del estado del arte donde se analizaron algunas soluciones existentes en Cuba y el mundo para el procesamiento de las retenciones, adquiriendo nuevas ideas para la confección de un producto que permitiera una mayor gama de ventajas y funcionalidades.
- Se describieron las características principales de las herramientas, tecnologías y los lenguajes de modelado utilizados logrando la fundamentación teórica para el desarrollo del trabajo.
- Se realizó la modelación del diseño teniendo en cuenta un conjunto de patrones y llevando a cabo su validación mediante métricas, lo que permitió el desarrollo de una solución robusta.
- El componente obtenido mediante la debida implementación contribuye a garantizar que el proceso de gestión de las retenciones se desarrolle de forma eficiente, garantizando la integridad de la información para la generación de la nómina mediante Cedrux, demostrado con los resultados satisfactorios obtenidos en las pruebas de caja blanca y caja negra realizadas.

.

Recomendaciones

Se recomienda:

- Continuar realizando pruebas de calidad al componente.
- Profundizar en temas referentes a la gestión de las retenciones para detectar posibles debilidades en el componente y agregar mejoras al mismo.
- Realizar el despliegue del componente propuesto como parte del subsistema de Capital Humano del sistema Cedrux.

Bibliografía

1. **Comités Técnicos de Normalización.** *Sistema de Gestión Integrada de Capital Humano.* La Habana. Cuba : s.n., 2007.

2. Real Academia Española. *Real Academia Española.* [En línea] [Citado el: 8 de diciembre de 2011.] http://www.rea.es/rae.html.

3. **Dominicana, DMarco.** *Manual del Sistema Assets Premium Versión 3.0.* Santo Domingo, República Dominicana : s.n., 2012.

4. **Maldonado López, Elvert y Aliagra Benavides, Adriel Alejandro.** *Desarrollo de reportes para el sistema de gestión de ciencia, Tecnología e innovación SIGCTI.* Granma : Facultad Regional Granma de la Universidad de las Ciencias, 2010.

5. Sitio oficial OpenBravo. *OpenBravo.* [En línea] Openbravo, S.L.U., 2010. [Citado el: 8 de diciembre de 2011.] http://www.openbravo.com/es/.

6. Sitio oficial Sage. [En línea] Sage, 2008. [Citado el: 5 de diciembre de 2011.] http://www.sageerpx3.com.

7. Solmicro Organización y Software. *SOLUCIONES INFORMATICAS.* [En línea] 2007-2010. [Citado el: 5 de diciembre de 2011.] http://www.solmicro.com/.

8. Sitio oficial SAP. *SAP.* [En línea] [Citado el: 26 de mayo de 2012.] http://www.sage.es/.

9. Sitio oficial del EPICOR. *EPICOR.* [En línea] [Citado el: 6 de diciembre de 2011.] http://www.epicor.com/lac/Pages/default.aspx.

10. Sitio oficial Siscont. *Siscont.* [En línea] [Citado el: 8 de diciembre de 2011.] http://siscont.tm.minbas.cu/Paginas/Default.aspx.

11. **TEICO.** *Manual de usuario y de explotación Versat Sarasola, versión 2.0.* Santa Clara,Villa Clara, Cuba : s.n., 2005.

12. Sitio oficial RODAS XXI. *Sistema Integral Economico Administrativo.* [En línea] [Citado el: 5 de diciembre de 2011.] http://www.rodasxxi.cu/rodasxxi.php.

13. Sitio oficial ASSETS. *Sistema de Gestión ASSETS.* [En línea] [Citado el: 9 de diciembre de 2011.] http://assets.co.cu/assets.asp.

14. **Gómez Baryolo, Ing. Oiner.** *Solución Informática de Autorización en entornos multientidad y multisistema.* La Habana : s.n., Julio - 2010.

15. **James Rumbaugh, Ivar Jacobson, Grady Boch.** *Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison-Wesley, 2000.

16. Lenguaje de modelado. [En línea] 27 de marzo de 2009. [Citado el: 1 de Febrero de 2012.] http://revistaing.uniandes.edu.co/pdf/A2%2029.pdf.

17. HERRAMIENTAS-CASE. [En línea] [Citado el: 1 de diciembre de 2011.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE..

18. Sitio oficial Visual Paradigm. *Visual Paradigm.* [En línea] [Citado el: 4 de diciembre de 2011.] http://www.visual-paradigm.com.

19. Lenguajes de Programacion. [En línea] [Citado el: 1 de Febrero de 2012.] http://es.scribd.com/doc/49636428/Lenguajes-de-Programacion.

20. Sitio oficial ExtJS. *ExtJS.* [En línea] [Citado el: 5 de diciembre de 2011.] http://www.sencha.com/.

21. Introducción, definición y evolución de PHP. [En línea] [Citado el: 2 de diciembre de 2011.] http://www.php.net/.

22. Sitio oficial Doctrine. [En línea] [Citado el: 1 de diciembre de 2011.] http://www.doctrine-project.org/.

23. Sistema Gestor de base de datos. [En línea] [Citado el: 1 de diciembre de 2011.] http://www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php.

24. EMS SQL Manager para PostgreSQL. [En línea] [Citado el: 1 de diciembre de 2011.] http://sqlmanager.net/en/products/postgresql/manager..

25. Grupo de usuarios PostgresSQL. [En línea] [Citado el: 1 de diciembre de 2011.] http://www.arpug.com.ar/trac/wiki/PgAdmin..

26. Sitio oficial del Mozilla Firefox. *Mozilla Firefox.* [En línea] [Citado el: 2 de diciembre de 2011.] http://www.mozilla.org/es-ES/firefox/fx/.

27. **Collins Sussman, Fitzpatrick, Pilato, W.B.** Control de versiones con Subversion. [En línea] [Citado el: 1 de diciembre de 2011.] http://svnbook.red-bean.com/nightly/es/svn-book.pdf..

28. **Larman, Craig.** *UML y patrones, Introducción al análisis y diseño orientado a objeto.* 1999.

29. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* 2011.

30. **Burbeck, Steve.** [En línea] http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html..

31. **BUSCHMANN, F., y otros.** *Pattern – Oriented Software Architecture. A System of Patterns.* Inglaterra : John Wiley & Sons, 1996.

32. **Pressman, Roger.** *Ingeniería de software. Un enfoque práctico.* 2002.

33. **Inda Gonzaléz, Ana Mahé.** *Consideraciones sobre la informatización de la gestión empresarial en Cuba y el mundo.* Centro de Estudios de Técnicas de Dirección (CETED), Facultad de Contabilidad y Finanzas de la Universidad de la Habana. La Habana, Cuba : s.n.

34. **Aguinaga, Moncho.** *Experiencias en la implantación de un ERP libre.* Malaga, España : s.n., 21 Octubre, 2008.

35. **Corporation, IBM.** Ayuda de Rational Unified Process. [En línea] 7.0.1, 1987-2006.

36. **Díaz, Lic. Alain, Prieto Olivera, Lic. Mabel y Delgado, Ing. Ana Maria.** *Sistema Automatizado para la Gestión Estratégica de la Capacitación.* 2010.

37. **Hernández Cisneros, E. J.** *Tesis de Licenciatura. Ingeniería en Ciencias Computacionales.* Pueblas, México : Universidad de las Américas, 2001.

38. **Jacobson Ivar, Booch Grady, James Rumbaugh.** *El Proceso Unificado de Desarrollo de software.* 2000.

39. *Los sistemas ERP.* **Kramer, Enrique.** 34, Cladea, Bogotá, Colombia : Revista Latinoamericana de Administración, 2005.

40. **Reynoso, Carlos y Kiccillof, Nicolás.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

41. **Vega Miniet, Ing. Yanet, y otros.** *Ciclo de vida del proyecto.* 2009.

42. **Recursos, Sistema de Gestión Integrada de los.** *Norma Cubana.* La Habana : s.n., 2007. 3000.

43. **Baryolo, Oiner Gómez.** *SOLUCIÓN INFORMÁTICA DE AUTORIZACIÓN EN ENTORNOS MULTIENTIDAD Y MULTISISTEMA.* La Habana : s.n., 2010.

44. **Genix, Ing. Alina de la Concepción Isasi.** *Estudio sobre la información que se gestiona de los recursos humanos, en los ERP.* Habana : DESOFT, 2010.

45. **autores, Colectivo de.** *Manual Assets Premium Versión 3.0.* Santo Domingo, República Dominicana : s.n., 2005.