Universidad de las Ciencias Informáticas

“Facultad 15”

Título: “Diseño e Implementación del Componente Nómina del subsistema Capital Humano del Sistema Integral de Gestión CEDRUX.”

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

Autor(es): Lázaro Pérez Bajuelo
Danier Estévez Laborí

Tutor(es): Ing. Rosendo Leonardo Hernández Claro
Co-tutor: Ing. Arnolis Salgueiro Arzuaga

Ciudad de La Habana, Junio 2010
“Año 52 Aniversario de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos 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 _____.

Autores:

_____ _____

Lázaro Pérez Bajuelo Danier Estévez Laborí

Tutor:

_____

Ing. Rosendo Leonardo Hernández Claro

DATOS DE CONTACTO

- Profesor Instructor, graduado en el 2008 de Ingeniero en Ciencias Informáticas de la Universidad de Ciencias Informáticas.
- Se desempeñó como Arquitecto de Información del proyecto de desarrollo de software del SIGEP (Sistema de Gestión Penitenciaria de la República Bolivariana de Venezuela) en el período de 2006 a 2008.
- Participó en misiones internacionalistas vinculadas al desarrollo del SIGEP en el año 2008.
- Actualmente ocupa el rol de Jefe de Factoría de la línea Capital Humano.

Correo electrónico: rlhernandez@uci.cu

AGRADECIMIENTOS

Quiero agradecer:

A mis padres y a mis hermanos por su apoyo incondicional, su amor, su comprensión, su dedicación hacia mi persona y su orgullo de mí.

A mis abuelos por todo el cariño y el amor que me han brindado.

A mi sobrinita Natalí por ser mi motivación para seguir hacía un futuro mejor.

A mis amigos y amigas, por su ayuda y su amistad incondicional.

A todos aquellos que han aportado un granito de arena en mi formación.

A mi tutor Rosendo por la ayuda y la dedicación brindada.

A todos los que siempre han confiado en mí y en lo que puedo hacer.

A todos aquellos con los que he vivido excelentes momentos.

Lázaro

Quiero agradecer:

A Fidel, por su capacidad única de transformar la vida ajena con la fuerza de sus convicciones y el poder de su sonrisa.

A mis padres: Adelaida Maricel Laborí Guilarte y Dalmey Estévez Santana, por haber seguido confiando en mí, aún cuando les defraudé muchas veces, aún cuando dije muchas cosas que no debí decir, y dije otras que debí callar, por estar siempre a mi lado y formar conmigo la perfecta familia imperfecta que me gustaría formar a mi alguna vez.

A mi hermanita: Dailin, por haber llegado a mi vida y convertirse en una parte esencial de ella, la vida es un camino de a dos, y nadie mejor que un hermano con quien recorrerlo.

A Lázaro, por ser a pesar de tantas cosas el hermano que me gustaría haber tenido, porque a su lado aprendí que en la vida hay pocas cosas recíprocas, incluyendo la amistad, pero que el lado que cuenta es el propio, lo demás es circunstancial y vago.

A mis amigos de siempre: Jorge, Yosley y José Liván, porque a pesar de todo, de discusiones estúpidas, de ofensas inoportunas, siempre supe que si no había nadie más, ellos iban a estar, gracias por eso.

A Arnoldo, sentimental, melodramático, agobiante y detallista, pero en el fondo una muy buena persona, por estar ahí cuantas veces le necesité sin que hiciera falta decirlo, por ser capaz de convertirse en poco tiempo en la persona en quien más confío.

Agradecimientos

A Daynelis, por demostrarme que una pizca de chabacanería en una mujer es agradable, si se complementa con una gran personalidad, por ser una muy buena amiga y una persona de primera.

A Alaincillo, un gran amigo, magnífica persona, un tipo fuera de serie, con él aprendí que un ápice de bobería es recomendable: si es poquito; espero aprender un poco de esto el tiempo que nos queda juntos.

A esas personas que sé que van a estar ahí si se les necesita, un grupo cerrado de amigos, de personas que admiro, quiero y respeto: Dina, Javier, Lio (Leosdenys), Pepe, Dayana, Daimi, Sahilis, Roy.

A Rosendo, un buen amigo, el tutor de esta tesis, la persona más "cool" de la UCI, aunque él difícilmente lo sepa.

A todas las personas que han sabido estar en estos años.

A esas otras personas que prefirieron no estar, que desearon mal o fueron prestas en juzgar, por su incapacidad quizás de comprender, que todo el mundo merece oportunidades, que las diferencias no nos hacen mejores, ni peores, sólo fragmentos de un mundo complejo, salvaje y diverso, pero estas personas son quienes al final nos imponen con su actitud retos, que nos hacen mejores o al menos más fuertes.

Danier

DEDICATORIA

A mi madre y a mi padre, por dedicar sus vidas a mí, a mi formación y por su sueño de que sea alguien importante en la vida.

A la memoria de mi abuela Alba Rosa, la persona que me quiso con la vida y que siempre esperó algo bueno de mí.

A mi abuela Noemis, la mujer que siempre ha estado a mi lado, cuidándome y guiándome por el buen camino.

Lázaro

Dedico esta tesis a las personas que con tanto esfuerzo han dedicado su vida a hacer más fácil y mejor la mía: a mis padres; a la persona que con orgullo me gustaría que se encontrara un día en el lugar que hoy yo me encuentro: a mi hermana; a esos que no dudaron en tender su mano cuando fue necesario: a mis amigos; a quien hizo posible la realización de los sueños de tanta gente, con el esfuerzo incansable de toda una vida: Fidel.

Danier

RESUMEN

En el área de Capital Humano de las entidades cubanas se realizan varios procesos entre los que se encuentra la confección y cálculo de la nómina, que permite realizar el pago de los trabajadores.

Actualmente se ha comprobado que el uso de las tecnologías en los procesos de negocios provoca un aumento de la productividad haciendo más manejables la información y los cálculos relacionados con ésta. En nuestro país no existe un sistema que permita la interrelación de la gestión de incidencias que se lleva a cabo en el departamento de Recursos Humanos y el procesamiento de la nómina de los trabajadores que se realiza en Contabilidad.

Con el propósito de contribuir al procesamiento de la nómina de los trabajadores en las entidades de nuestro país se propone desarrollar el componente Nómina que forma parte del Subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

PALABRAS CLAVES

Capital humano, nómina, procesamiento nómina, concepto de pago, patrones, componente.

Índice

INTRODUCCIÓN

El mundo actual evoluciona a una velocidad vertiginosa y el uso de las tecnologías de la información y las comunicaciones (TIC) se ha vuelto indispensable para su permanente desarrollo. Como parte de las tecnologías de la información y las comunicaciones *la informatización de los procesos* ha sido un motor propulsor para potenciar el cumplimiento de las funciones a todos los niveles, con un máximo de racionalidad y control en las diferentes áreas de la misma. La conveniencia de la utilización de las TIC está siendo aprovechada por el gobierno de Cuba, con el fin de crear una sociedad con un alto nivel cultural, permitiendo elevar la economía del país con la implantación de sistemas informáticos en las diferentes entidades existentes, muchos de los cuales han sido importados. Una de las principales metas es desarrollar la Industria del Software y convertirla en uno de los principales renglones económicos del país, no solamente por los beneficios en el desarrollo de sistemas para uso interno, sino también con el fin de insertarse en el mercado de software a nivel mundial por su perspectiva económica.

Como parte de las iniciativas que se están llevando a cabo para la sustitución de importaciones de sistemas, se tomó la decisión del desarrollo de una solución informática para la gestión integral de entidades que proveerá de una herramienta para el control y gestión de recursos financieros, materiales y de capital humano. Un sistema o ERP (por sus siglas en inglés de Planificación de Recursos Empresariales) que proporcionará mayor integración de la información y modernización de los procesos de negocio que le permitirá a las entidades empresariales y presupuestadas mayor eficiencia y productividad.

Este sistema informático denominado Cedrux está siendo desarrollado por estudiantes y profesionales de la Universidad de las Ciencias Informáticas (UCI), apoyados por los diferentes ministerios del país. En la UCI se tiene ya la experiencia de desarrollar sistemas de grandes dimensiones aunque no con la misma envergadura de un sistema de estas características que representa un reto para el equipo de desarrollo implicado en su creación por la complejidad de su negocio y la interrelación que existe entre sus procesos. El mismo dará la posibilidad de poder mantener un mejor control sobre las diferentes entidades que existen en el país, posibilitando que cumpla con todas las funcionalidades requeridas. Uno de los subsistemas más abarcadores que forman parte del sistema Cedrux es el de Capital Humano. Desde hace ya varios años el factor humano en las organizaciones ha ido adquiriendo una importancia vital en

las empresas e instituciones de todo tipo; ya que independientemente de la tecnología que pueda disponerse, el hombre decide en el éxito o no del cumplimiento de su misión por lo que una gestión eficiente de los recursos humanos es fundamental. La gestión del Capital Humano abarca un grupo de procesos, que constituyen los módulos del sistema, y que interactúan entre sí como son: Organización del trabajo, Selección e integración de personas, Evolución del desempeño, Capacitación y desarrollo, Seguridad y salud del trabajo, Autocontrol y Estimulación moral y material y, dentro de éste, los procesos de nómina donde se efectúa el cálculo para el pago a los trabajadores.

En la actualidad la industria cubana de software está optando por la independencia tecnológica, pero no cuenta con un sistema propio que permita la interrelación de la gestión de incidencias (afectaciones al pago de los trabajadores) que se lleva a cabo en el departamento de Recursos Humanos y el procesamiento de la nómina de los trabajadores que se realiza en Contabilidad. Esto trae consigo que al no estar integrados, la información se tenga duplicada, aumente el margen de contaminación de la información sobre todo por errores de captura, es decir la información al intercambiarse de un departamento a otro puede ser alterada por lo que se crea un ambiente propicio para efectuar pagos indebidos. Todo esto implica que la información obtenida no sea oportuna lo que imposibilita un control eficiente de la misma, dificultando de esta forma el cumplimiento de las fechas establecidas en el convenio colectivo de trabajo para el pago de los trabajadores. Es por esto que no se le puede dar un adecuado seguimiento al proceso del negocio y se hace engorroso el trabajo a las personas encargadas de la toma de decisiones dentro de las entidades afectando la elaboración de estrategias organizacionales.

Basándose en lo antes planteado se puede inferir como **problema a resolver** en este trabajo:
¿Cómo proporcionar una solución integrada para la gestión de los procesos de Nómina en las entidades nacionales, partiendo de los requisitos capturados del módulo Estimulación moral y material y obedeciendo la política de independencia tecnológica? A partir del problema planteado se define como **objeto de estudio:** Los procesos de Capital Humano, teniendo como **campo de acción:** Los procesos de nómina en las entidades cubanas.

El **objetivo general** trazado para darle solución al problema planteado es:

Realizar el diseño e implementación del componente Nómina del subsistema Capital Humano para realizar el cálculo de la nómina de los trabajadores en las entidades cubanas de acuerdo a los requisitos capturados del módulo Estimulación moral y material, obedeciendo la política de independencia tecnológica.

Con el propósito de darle cumplimiento al mismo se derivan los siguientes **objetivos específicos:**

- Realizar un estudio de las vías existentes para el procesamiento de la nómina de los trabajadores.
- Diseñar e Implementar la solución bajo los estándares definidos en el proyecto ERP-Cuba.
- Realizar pruebas a la solución para comprobar la calidad de la misma.

Idea a defender:

Informatizando el cálculo de la nómina de los trabajadores en las entidades cubanas, garantizando el correcto funcionamiento de los requisitos capturados del módulo Estimulación moral y material, se realizaría el cálculo de la nómina de los trabajadores en las entidades cubanas de manera integral y obedeciendo la política de independencia tecnológica.

De esta manera para darle cumplimiento a los objetivos específicos propuestos se determinan las siguientes **tareas de investigación: Realizar el estado del arte.**

- Analizar documentación del marco de trabajo del ERP.
- Analizar documentación de las herramientas a utilizar para el desarrollo del componente.
- Analizar los artefactos entregados por el equipo de Análisis.
- Realizar el diseño e implementación de clases persistentes.
- Diseñar el modelo datos teniendo en cuenta el análisis realizado.
- Realizar pruebas de caja blanca al componente.

Posibles resultados:

Se espera obtener Componente Nómina del subsistema Capital Humano del sistema Cedrux que gestione de manera integral los procesos de nómina en las entidades nacionales.

EL presente trabajo está estructurado en 3 capítulos donde la información de la investigación realizada se encuentra distribuida de la manera siguiente:

Capítulo #1: Se abordan varios temas entre los que se encuentran la fundamentación de algunos conceptos generales con el objetivo de un mejor entendimiento de los procesos relacionados con el procesamiento de la nómina, se realiza un estudio de algunas soluciones de software existentes relacionadas con los sistemas contables y de recursos humanos tanto a nivel nacional como internacional y por último se hace referencia a las metodologías, tecnologías y herramientas utilizadas para el desarrollo de la solución propuesta.

Capítulo #2: Se analizan los artefactos entregados por el equipo de Análisis. Se modela el diagrama de componentes, permitiendo con ello tener una visión más clara de la implementación del sistema, y de las integraciones entre los componentes que están relacionados con el sistema. Se realiza la descripción de la propuesta de solución, se abordan temas entre los que se encuentran los patrones de diseño utilizados, se modelan los diagramas de clases del diseño permitiendo dar paso a la implementación del sistema.

Capítulo #3: Se realiza una valoración del diseño propuesto y se abordan temas como las pruebas realizadas al software, en específico las pruebas de caja blanca, además se hace una valoración de las mismas según los resultados obtenidos.

CAPÍTULO #1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo serán abordados una serie de definiciones y conceptos que servirán de base para el entendimiento del ámbito en que se desarrollará este proyecto, los cuales tratarán de dar una idea clara del negocio sobre el que se va a desarrollar. Serán abordados además otros sistemas existentes de gestión, que brinden funcionalidades similares al que se expone en este trabajo. Finalmente serán descritas las tecnologías, metodologías y herramientas sobre las cuales será desarrollado el sistema.

1.2. Conceptos

En este epígrafe se abordan los conceptos generales del negocio relacionados con la solución propuesta.

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 Moral y Material

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)

Concepto de pago

El concepto de pago es la caracterización del pago en términos de su naturaleza y del tratamiento de los parámetros de tiempo, importe, acumulación de vacaciones, aplicación de impuestos y procedimientos de contabilización. De modo que deben definirse tantos conceptos de pago como combinaciones de estos componentes se identifiquen. (2)

Además un trabajador puede tener vario conceptos de pago por los cuales se le pagará al procesar la nómina.

Impuesto

En este concepto se recogen los impuestos y contribuciones que debe aportar la entidad al presupuesto del estado, la diferencia está en que existen dos tipos: los empresariales con base el total de lo pagado a los trabajadores, y los salariales que son un aporte del trabajador y tienen como base el salario devengado de este.

Nómina

Una nómina se define como una lista conformada por el conjunto de trabajadores de una entidad determinada a los cuales se les va a remunerar por los servicios prestados. Es el instrumento que permite de una manera ordenada, realizar el pago de salarios a los trabajadores, así como proporcionar información contable y estadística, tanto para la empresa como para el organismo encargado de regular las relaciones laborales. Aunque en cada entidad puede variar la forma de calcular y contabilizar la nómina, existen ciertos pasos comunes a todas, como la preparación de la nómina con los nombres y las remuneraciones de los trabajadores. (3)

Salario

Salario: Parte del producto nacional que se distribuye a los trabajadores de forma individual, atendiendo a la cantidad y calidad del trabajo aportado, según las condiciones económicas de cada momento histórico. Comprende lo percibido por el trabajador, por rendimiento, unidad de tiempo, pagos adicionales, trabajo extraordinario, laborar en día de conmemoración nacional y feriados, y vacaciones anuales pagadas. (1)

Salario básico: Remuneración que comprende la tarifa de la escala salarial, más los pagos adicionales establecidos legalmente, y se utiliza en los casos previstos en la legislación. (1)

Salario escala: Cantidad de dinero, en moneda nacional de curso legal, que corresponde a cada grupo de la escala de complejidad. (1)

Submayores

Es un registro donde se analizan las subcuentas y las cuentas que lo requieran. En ellos se reflejan todos los movimientos que tienen las cuentas desde su apertura.

Retenciones

Son los adeudos contraídos por el trabajador que se descuentan periódicamente, según se determine mensual, quincenal o semanalmente, del salario devengado por el trabajador.

Comprobante de operaciones

En este se registran las operaciones contables que se realizan en una nómina. En un comprobante de operaciones están registradas las operaciones en forma de asientos es decir un comprobante contiene varios asientos.

Asientos

En esta entidad se registran todos los pases u operaciones realizados sobre una cuenta. Un asiento es el resumen de todas las operaciones o pases realizados sobre una cuenta.

Pase

En este se registran las operaciones realizadas sobre una cuenta determinada por el asiento al que pertenece. Se registra además otros detalles de la operación como importe base, moneda contable.

Documento de nómina

El documento de nómina (o nómina solamente) es el documento que se crea para procesar los pagos de los trabajadores. Sirve para definir el procesamiento que se le dará (tipo de nómina) a los conceptos de pago que se procesarán en el periodo de pago. Por cada trabajador en el Documento de nómina se tendrá un procesamiento de nómina diferente.

Período de pago

En esta entidad se definen los distintos periodos de pago de la empresa. Un periodo de pago es un lapso de tiempo puede ser una semana, 15 días o un mes.

Procesamiento de nómina

En este se realiza el procesamiento de cada trabajador por cada nómina. Aquí se registra el resultado de los cálculos de los conceptos de pagos, además de las retenciones y otros descuentos. En caso de que se

haya cometido algún error en el cálculo este se puede corregir mediante un ajuste de un tipo de los definidos en la entidad.

Tipo de nómina

En esta entidad se definen los tipos de nómina. El tipo de nómina determina el procesamiento que se le dará a las nóminas de cada tipo al ser calculadas.

Tipo de ajuste

En esta entidad se definen los tipos de ajustes. Los ajustes se utilizan para corregir los errores al reportarles los pagos a los trabajadores después de procesada la nómina.

Componente

Para que un elemento pueda ser considerado un componente existen algunas características claves que hay que tener en cuenta, entre ellas se encuentran:

- **Identificable**: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- **Auto contenido**: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- **Puede ser remplazado por otro componente**: Se puede remplazar por nuevas versiones u otro componente que lo remplace y mejore.
- **Bien Documentado**: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo.
- **Es genérico:** Sus servicios debe servir para varias aplicaciones.
- **Reutilizado dinámicamente:** Puede ser cargado en tiempo de ejecución en una aplicación.
- **Independiente de la plataforma:** Hardware, Software, S.O. (4)

Por lo que se define como componente de software todo aquel recurso desarrollado para un fin concreto y que puede formar, solo o junto con otros, un entorno funcional requerido por cualquier proceso predefinido.

1.3. Proceso de confección y cálculo de la nómina

Se realiza basado en el resumen de incidencia de cada trabajador en el periodo de pago a analizar. Inicialmente se crean el Mayor de trabajadores, los Submayores correspondientes (submayor de vacaciones y submayor de retenciones), el Fichero histórico devengado y el Registro de salario y tiempo de servicio. Luego se procede a realizar las nóminas correspondientes a Salarios, Subsidios, Vacaciones, entre otros, en el período que se establezca, con las informaciones obtenidas de la pre nómina que contienen las incidencias procesadas por el área de personal; las retenciones de los trabajadores utilizando las notificaciones bancarias y los movimientos de nómina que permiten actualizar el maestro de trabajadores.

Cuando se realiza la nómina se tiene en cuenta si el trabajador tiene alguna retención verificándolo con el submayor de retenciones, en caso de tener se le descuenta y se actualiza este submayor al terminar el procesamiento de la nómina; si la nómina es de vacaciones se verifica la duración de las vacaciones que tomará el trabajador para sacar proporcionalmente el importe al tiempo de acuerdo a lo que tiene acumulado en el submayor de vacaciones y esta cantidad es lo que se le pagará, actualizando este submayor al concluir el procesamiento; si la nómina es de subsidio se debe tomar del fichero histórico devengado el acumulado del tiempo trabajado en los meses anteriores para el cálculo de la misma. Al terminar se actualizan los submayores el vacaciones con la provisión de vacaciones acumuladas en ese período y el de retenciones con las respectivas deducciones descontadas al trabajador; además de los submayores se actualiza el registro de salario y tiempo de servicio y se obtiene la distribución de moneda para el pedido del efectivo al banco.

La información de las nóminas se enviará mediante el comprobante de operaciones al subsistema de Contabilidad con el propósito de realizar la contabilización de las mismas y se obtendrá además la distribución de moneda para la solicitud del efectivo al banco.

Finalmente se procede a hacer el cierre del periodo contable para lo cual se comprueba la actualización de los submayores realizando la conciliación con el importe de las cuentas del departamento de Contabilidad, se verifica que todas las nóminas estén confirmadas y todos los comprobantes de operaciones estén confirmados.

1.4. Sistemas existentes

En el mundo actual existen numerosos sistemas contables con el objetivo de dar una solución factible para la ejecución de los cálculos que se realizan durante el procesamiento de la nómina. En los siguientes epígrafes se realiza un análisis de algunos sistemas investigados sobre las funcionalidades que brindan, las plataformas en que están desarrollados, y quienes son los autores de los mismos.

1.4.1. Sistemas similares en el área nacional

RODAS XXI El Sistema Integral Económico Administrativo RODAS XXI desarrollado por la empresa CITMATEL posibilita automatizar el funcionamiento de cualquier empresa o unidad presupuestada.

RODAS XXI es un sistema multi-empresa que cuenta actualmente con seis módulos: Finanzas, Contabilidad, Activos Fijos, Nóminas, Inventario y Facturación. Estos módulos pueden emplearse integrados en su totalidad, formando cualquier subconjunto entre ellos, o cada uno de forma independiente. En el módulo de nómina para la obtención de las nóminas en cada uno de los períodos de año el sistema divide las operaciones en dos pasos fundamentales, el cálculo de nóminas y su emisión.

Como resultado del cálculo de cada nómina y su comprobante el sistema muestra cómo quedarían en caso de emitirse dicha nómina, en caso de detectar algún error u omisión en las incidencias agregadas a los trabajadores que les corresponda o en el comprobante calculado se pueden realizar las correcciones pertinentes calculándola nuevamente hasta que todo está correcto y una vez logrado esto puede procederse a emitirla lo que implica que en el sistema a nivel de módulo se registrará la nómina emitida en el período que se está trabajando así como su comprobante correspondiente. 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. También son generados de forma automática los modelos SNC 4-2-25 y SNC 4-1-25, el 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. Este módulo permite además visualizar información correspondiente a períodos anteriores, tan sólo con cambiar de período contable a periodos anteriores ya cerrados, aunque en dichos

periodos no podrá realizar ninguna operación. El sistema contable RODAS XXI aunque tiene funcionalidades muy útiles como lo es la de guardar por trabajador los pagos y retenciones que se le realizan y solo actualizarlas cada mes para la posterior elaboración de la nómina, no es una solución factible al problema expuesto, aun siendo un producto nacional. Esto se debe a que está desarrollado para plataforma de software propietario y no presenta una gestión de recursos humanos integrada a la nómina.

VERSAT SARASOLA

El sistema **VERSAT-Sarasola** está desarrollado por la Empresa de Tecnologías de la Información y el Conocimiento (TEICO) del Ministerio del Azúcar en una plataforma de Delfi con servidor SQL Server, está constituido por 10 módulos o subsistemas que incluyen configuración y seguridad, contabilidad general y de gastos, costos y procesos, finanzas y caja, activos fijos, planificación y presupuestos, control de inventarios, pago de salario (nómina), facturación y generador de reportes. Es un paquete para la gestión económica financiera se distingue por ser el primer sistema de contabilidad cubano certificado, según las nuevas normativas establecidas por los Ministerios de Finanzas y Precios y de la Informática y las Comunicaciones, para este tipo de Software. Logra establecer un proceso de interacción usuario-sistema y posee una gran rapidez y agilidad a partir de la configuración del proceso de contabilización de los documentos primarios y de las propias posibilidades de trabajo contenidas en cada subsistema. El subsistema Nómina de Salarios es lo suficientemente configurable a los diversos sistemas de pago, estimulaciones y demás regulaciones laborales que existen en el país. Posee además todo lo relacionado con el descuento, liquidación y submayor de retenciones, así como vacaciones acumuladas y SNC- 2-25. Se puede configurar el sistema definiendo de antemano todas las variantes de impuestos, bonificaciones, condiciones anormales, retenciones, descuentos, penalizaciones, etc. con las denominaciones y los parámetros numéricos que se requieran. Mediante los conceptos de pago puede definir todas las variantes de cálculos y procedimientos contables que se presenten en su entidad y designarlas con las denominaciones más naturales.

El sistema contable VERSAT Sarasola presenta funciones útiles como lo es la de poder configurar el sistema definiendo de antemano todas las variantes de impuestos, pagos adicionales y retenciones. Aun así no es una solución viable debido a que está desarrollado para software propietario y no presenta una

gestión de recursos humanos integrado a la nómina. Y al igual que otros de los sistemas analizados anteriormente se deben comprar las licencias del mismo, viéndose en la necesidad de comprar el sistema en su totalidad para que funcione correctamente, debido a que necesita de una integración con el resto de los subsistemas, siendo poco útil adquirir solamente algunos módulos, puesto que estos requerirían de una interrelación con algún otro quedando funcionalidades sin poder utilizar.

1.4.2. Sistemas similares en el área internacional

ASSETS NS Assets S.A. es una compañía de desarrollo de software registrada en la República de Panamá con capital italiano, sus operaciones fundamentalmente se realizan en el área de América Latina con representaciones en Italia, Panamá, Cuba y en República Dominicana donde se halla la Casa Matriz. ASSETS NS es un Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos. Como Sistema Integral todos sus módulos trabajan en estrecha relación, generando, automáticamente, al Módulo de Contabilidad los Comprobantes de Operaciones por cada una de las transacciones efectuadas, esto permite que se pueda trabajar bajo el principio de Contabilidad al Día. El Módulo de Recursos Humanos Versión 3.1 desarrollada en Access 97 está concebido para calcular las nóminas y controlar los recursos laborales de una entidad. Sin embargo, debido a las necesidades existentes en el país, y a que ASSETS NS es un software extranjero, lo cual significa gastar recursos monetarios en adquirirlo, este sistema no se puede utilizar como solución al problema que se expone. Además para su instalación se requiere que sea en el sistema operativo Windows utilizando el Access 97.

SAGE MAS 500 Sage Software fue quien desarrollo el sistema Sage MAS 500, es una empresa que radica en California EEUU. Este software ofrece servicios de software de automatización de los procesos de recursos humanos y la nómina, permiten automatizar las actividades y el cumplimiento de estas por los recursos humanos, además de permitir realizar un seguimiento de la asistencia, los empleados y de la nómina. Abra nómina es el módulo que permite realizar el proceso de la nómina y la presentación de informes, y brinda las siguientes funcionalidades:

- Simplificar la organización de procesamiento de la nómina.
- Gestión de requisitos de presentación de informes.

- Usar la completa funcionalidad de gestión tributaria que le ayudarán a permanecer compatible.
- Crear ingresos ilimitados y la deducción códigos.
- Crear automáticamente archivos ACH (Automated Clearing House) para procesar las transacciones de depósito directo.
- Asignar un número ilimitado de cuentas de depósito directo para sus empleados.
- Vista previa de impresión antes de los controles, controles de impresión de prueba, y simplificar aún más comprobar la impresión láser con MICR (Magnetic Ink Character Recognition) y opciones de la firma.
- Eliminar la duplicación de la entrada de datos.

Sin embargo, SAGE MAS 500 es un software extranjero no certificado, desarrollado para plataforma de software propietario. Para que funcione correctamente necesita de una integración con el resto de los subsistemas, es decir para que el funcionamiento sea el más óptimo posible debería comprarse el sistema completo, siendo poco útil adquirir solamente algunos módulos, puesto que estos necesitarían de una interrelación con algún otro quedando funcionalidades sin poder utilizar. Debido a lo anteriormente expuesto este sistema no se puede utilizar como una solución posible al problema.

SAP

SAP AG (Sistemas, Aplicaciones y Productos) es una empresa alemana y uno de los proveedores de software empresarial más importantes del mundo. Uno de sus principales productos es SAP ERP, este proporciona a sus usuarios la tecnología necesaria para contar con una amplia visión de todas las actividades de su negocio. De esta manera, sus decisiones serán más acertadas, debido a que reflejarán la situación de la empresa en el momento oportuno y no deberán de basarse en estimaciones o información obsoleta.

SAP ERP está basado en una plataforma abierta que proporcionara un completo control sobre la operativa y estrategia empresarial. Sin embargo SAP ERP, aún estando certificado para su posible utilización en Cuba, no es una solución factible al problema que se expone, debido a que el país no cuenta con los recursos financieros suficientes para adquirirlo y sus licencias tienen un alto valor monetario. Además de que es un software extranjero y desarrollado para plataforma de software propietario.

Seven2000

Seven2000 es un producto de SQL EVEN SEVEN C.A. que es una empresa radicada en Valencia, Estado Carabobo, Venezuela que brinda asesoría en las áreas de Desarrollo de Aplicaciones (Oracle Developer, Oracle Designer) así como en las áreas de Administración de Base de Datos y Entonación de Aplicaciones en distintas plataformas (Unix, Windows/NT, Linux, etc.). También tiene varios productos denominados Seven2000, que han sido instalados tanto en las empresas privadas como en las empresas públicas.

Seven2000 es un sistema integrado compuesto por módulos que pueden actuar separados o integrados. El módulo de Sistema de Finanzas es un sistema de manejo Financiero (Contabilidad, Cuentas por Pagar, Cuentas por Cobrar, Bancos, Activo Fijo, Control Presupuestario). Multi-Empresa/ Multi-Moneda/ Moneda alterna. El sistema de Comercialización es un sistema de manejo Comercial. Compras, Inventarios y Facturación. El Sistema de nómina y Recursos Humanos permite la total automatización de los procesos asociados al control de los pagos del personal y mantenimiento de historias de los trabajadores. Sin embargo Seven2000 no es una solución factible al problema expuesto debido a que es un software extranjero no certificado, desarrollado para plataforma de software propietario. Además no presenta una gestión de Recursos Humanos integrado a la nómina; uno de los objetivos que se desea lograr con la realización de este trabajo es que la nómina se realice como parte de la gestión de Recursos Humanos.

1.5. Modelo de desarrollo

Para el desarrollo del componente se siguieron los lineamientos arquitectónicos establecidos por la dirección del proyecto, que plantean el uso de un modelo estandarizado, y la definición clara y precisa de las responsabilidades de cada uno de los roles que se ven involucrados en el desarrollo de la solución.

El modelo de desarrollo de software propuesto se estableció con el objetivo de producir software de alta calidad, basado en la retroalimentación continua entre el cliente y el equipo de desarrollo; siendo factible adaptar las características de las metodologías ágiles y robustas en la construcción de un nuevo modelo debido a la necesidad de producción de un sistema ERP para el país. (37)

Este modelo de desarrollo se caracteriza por ser:

- **Centrado en la arquitectura**

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

- **Orientado a componentes**

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

- **Iterativo e incremental**

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

- **Ágil y adaptable al cambio**

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

1.6. Tecnologías

En el marco de trabajo del proyecto ERP-Cuba donde se está desarrollando el Sistema de Gestión Integral “Cedrux” se definieron las tecnologías y herramientas a usar para la construcción del software las cuales se les dan a conocer más adelante.

1.6.1. AJAX

AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML) es una tecnología que facilita la creación de aplicaciones interactivas en la Web que se ejecutan en el navegador de los usuarios y mantienen comunicación asíncrona con el servidor; posibilitando que se puedan efectuar cambios sobre una página sin necesidad de recargarla, aumentando de esta manera la interactividad, velocidad y usabilidad de la misma. (38)

AJAX está conformado por:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación. (40)
- XML, JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

Además:

- Provee un mecanismo para mezclar y hacer coincidir XML con XHTML.
- Las aplicaciones son más rápidas e interactivas, al estilo aplicaciones de escritorio.
- Reduce de manera significativa tener que cargar información continuamente del servidor, actualizando solamente porciones de la página.
- Cuando se utiliza AJAX adecuadamente en el desarrollo de una aplicación, se reduce de manera significativa los tiempos de carga inicial. (5)

A continuación se explican las características más importantes de algunas de las tecnologías que componen AJAX:

XML

Es el estándar de Extensible Markup Language (Lenguaje de Etiquetado Extensible), que se encuentra conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que son: extensibilidad, estructura y validación. Ofrece un formato para la descripción de datos estructurados, facilitando declaraciones de contenido más precisas y resultados de búsquedas más significativos en varias plataformas.XML permite compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. (6)

JSON

JSON, acrónimo de —JavaScript Object Notation-, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Es muy sencillo de usar, especialmente como alternativa a XML. Una de las ventajas de JSON sobre XML

como formato de intercambio de datos es que es mucho más sencillo escribir un analizador semántico de JSON. (7)

XHTML

XHTML es el acrónimo en inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto). Es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. (39)

Algunos de estos requisitos son:

- Elementos correctamente anidados.
- Etiquetas en minúsculas.
- Atributos de valores entrecomillados. (8)

CSS

Es un lenguaje de hojas de estilos en Cascada en sus siglas en ingles (Cascading Style Sheets) creado para controlar la presentación de documentos estructurados, aspectos como: el color, el tamaño, el tipo de letra, la separación entre párrafos y la tabulación con la que se muestran los elementos de una lista. El propósito del desarrollo de CSS es separar la estructura y el contenido de la presentación estética en un documento, esto permite un control mayor del documento y sus atributos, convirtiendo al HTML en un documento muy versátil y liviano. Entre los beneficios concretos de CSS se encuentran:

- Control de la presentación de muchos documentos desde una única hoja de estilo.
- Control más preciso de la presentación. (9)

JavaScript

Es un lenguaje basado en objetos y guiado por eventos, diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet. Los programas escritos con este lenguaje se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios, convirtiéndolo en un lenguaje interpretado. Su uso se basa fundamentalmente en la creación de efectos especiales en las páginas y la definición de interactividades con el usuario. (40)

Ventajas de JavaScript:

- Los programas escritos en este lenguaje no requieren de mucha memoria ni tiempo adicional de transmisión, por ser pequeños y compactos.
- JavaScript no requiere un tiempo de compilación; ya que los scripts se pueden desarrollar en un período de tiempo relativamente corto.
- Es independiente de la plataforma hardware o sistema operativo, y funciona correctamente siempre y cuando exista un navegador con soporte JavaScript. (10)

1.7. Lenguajes de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. Son considerados además herramientas que nos permiten crear programas y software. Estos facilitan las tareas de programación ya que poseen formas adecuadas para su entendimiento y resultan independientes de la computadora a utilizar.

1.7.1. PHP

PHP (Hypertext Processor) es un lenguaje script (no se compila para conseguir códigos máquina si no que existe un intérprete que lee el código y se encarga de ejecutar las instrucciones que contiene éste código), para el desarrollo de páginas web dinámicas del lado del servidor, Es un lenguaje Open Source (código abierto), es el más popular y extendido en la web. (28)

Entre las características que posee este lenguaje y que lo convierten en una potente herramienta están:

- Es un software de código abierto.
- Soporta muchas bases de datos entre las que se encuentran (MySQL y PostgreSQL).
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación. (13)

1.8. Framework

En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo, la cual extiende o utiliza las aplicaciones del dominio. En general, los frameworks son construidos en base a lenguajes orientados a objetos. Además, en la mayoría de los casos un framework implementará uno o más patrones de diseño de software que aseguren la escalabilidad del producto. (16)

1.8.1. ExtJS 2.0

La nueva versión ExtJS 2.0 contiene nuevos rasgos y características que los distinguen entre los que se encuentran el aumento considerable de la biblioteca. Existen además muchos rasgos adicionales y cambios arquitectónicos que hacen el funcionamiento con 2.0 más intuitivo y flexible que liberaciones anteriores. Se determinan algunos rasgos únicos de esta versión 2.0 entre los que se encuentran:

- Columna de nivel sumamente configurable que agrupa capacidades así como resumen.
- Vistas de árbol que apoya columnas adicionales para cada nodo de hoja.
- Etiquetas de desplazamiento. (29)

1.8.2. Zend Framework

El Zend Framework, es un framework con una arquitectura flexible con el que podremos construir sin demasiadas complicaciones potentes aplicaciones web. Además se trata de un framework para desarrollo de servicios Web con PHP, te brinda soluciones para construir sitios web modernos y seguros. Además es Open Source y trabaja con PHP 5. Entre sus principales características se encuentran:

- Trabaja con MVC (Model View Controller).
- El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar su base de datos, sin tener que escribir ninguna consulta SQL.
- Robustas clases para autenticación y filtrado de entrada. (12)

1.8.3. Zend_Ext Framework

Es un framework Open Source, que está diseñado para PHP 5 y buenas capacidades de ampliación. Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizada por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor, se le incluyó el IoC[1] para la comunicación en entre los módulos o componentes. Se le incorporó la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos y el ExtJs Framework para el desarrollo de las vistas.

1.8.4. Doctrine Framework

Es un potente y completo sistema ORM (Object Relational Mapper) para PHP 5.2+ que incorpora una(capa de abstracción a base de datos), en sus siglas en inglés (DBL). Posee la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto les brinda una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (13)

1.9. Herramientas

Las herramientas utilizadas para el desarrollo de la solución fueron definidas por la dirección del proyecto ERP-Cuba.

1.9.1. Visual Paradigm

Visual Paradigm para UML (Lenguaje de Modelado) es una herramienta que emplea UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se usó para el modelado del sistema propuesto debido a que entre sus utilidades permite construir aplicaciones con mejor calidad, además posibilita entre sus

[1] IoC (Inversión de Control): La inversión de control se hace necesaria para gestionar las dependencias entre subsistemas y framworks. En la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario.

funciones realizar diagramas de clases, código inverso, así como generar código desde diagramas y generar documentación.

UML

El Lenguaje Unificado de Modelado, en sus siglas en inglés (UML), es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.
UML un lenguaje de propósito general para el modelado orientado a objetos. UML es también un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes.

Objetivos del UML:
UML es un lenguaje de modelado que puede ser usado por todos los modeladores. Incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso. Debe ser un lenguaje universal, como cualquier lenguaje de propósito general. (36)

1.9.2. Ambiente de desarrollo integrado (IDE)

El ambiente de desarrollo (Development Environment) es algo imprescindible en la producción de software. Es donde se definen el conjunto de herramientas y tecnologías (frameworks), versiones a usar y su integración, que intervienen en un proceso de desarrollo de software. Un entorno de desarrollo integrado o Integrated Development Environment (IDE), en inglés, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic por ejemplo puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word. Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic y Object Pascal. Es posible

que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, que mediante pluggins se le puede añadir soporte de lenguajes adicionales. (14)

Zend Studio para Eclipse

El Zend Studio para Eclipse constituye la nueva generación del Zend Studio. Proporciona a las organizaciones que hagan uso del mismo ya que posee un entorno mucho más flexible y profesional para controlar todo el ciclo de vida de un desarrollo. Entre sus funcionalidades, destacaría las capacidades de refactorización del código fuente, funcionalidad que permite adecuar el comportamiento externo de una función clase sin cambiar el funcionamiento interno, que junto a las capacidades de generación de código le facilitaría el trabajo a los desarrolladores. Entre las nuevas características que incluye este IDE están:

- El acceso al ecosistema de plugins de Eclipse.
- Apoyar el desarrollo de múltiples idiomas.
- Mejora el Editor de PHP con el formato avanzado, para listas de tareas y problemas de vista.
- Mejora del soporte de JavaScript.
- Mejora de apoyo, incluyendo HTML, códigos plegables, arrastrar y soltar los componentes.
- Mejora de depuración y Perfiles con Path Mapping.
- Mejora de Zend Framework con el apoyo del Proyecto Framework, plantillas y código MVC. (15)

1.9.3. Herramientas de base de datos

Existen numerosas herramientas que se relacionan directamente con servidores de bases de datos. Estas aplicaciones de orden avanzado ofrecen la oportunidad de administrar completamente los servidores basados en lenguaje de datos.

EMS SQL Manager

EMS SQL Manager es una herramienta de alto rendimiento para la administración de bases de datos PostgreSQL y el desarrollo. Funciona con cualquier versión de PostgreSQL hasta la más reciente y soporta las últimas características incluyendo PostgreSQL enumerados, búsqueda de texto, XML y tipos de arreglos de tipos compuestos. SQL Manager para PostgreSQL ofrece muchas de las poderosas herramientas de base de satos, como bases de datos visuales Designer, Visual Query Builder para crear consultas de PostgreSQL complicado. (21)

Sistema Gestor de Base de Datos (SGBD)

Los sistemas de gestión de bases de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Proveen facilidades para la manipulación de grandes volúmenes de datos. Entre éstas:

- Simplifican la programación de equipos de consistencia.
- Manejando las políticas de respaldo adecuadas, garantizan que los cambios de la base serán siempre consistentes sin importar si hay errores correctamente, etc.
- Organizan los datos con un impacto mínimo en el código de los programas.
- Bajan drásticamente los tiempos de desarrollo y aumentan la calidad del sistema desarrollado si son bien explotados por los desarrolladores.
- Usualmente, proveen interfaces y lenguajes de consulta que simplifican la recuperación de los datos.

PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS), de código abierto. Ofrece incorporar los siguientes cuatro conceptos adicionales básicos de manera que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos, funciones. Otras características aportan potencia y flexibilidad adicional: Restricciones (Constraints), Disparadores (triggers), Reglas (rules), Integridad transaccional. Debido a eso se considera la base de datos de código abierto más avanzada hoy día disponible, soporta casi toda la sintaxis SQL y cuenta también con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, PHP, Java, Perl, Tcl y Python). (20)

1.9.4. Servidor Web

Un servidor web es un programa que está diseñado para transferir hipertextos, páginas web o páginas HTML (Hypertext Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. Además sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP.

Apache

Entre sus principales características se encuentran:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- Tiene una alta configurabilidad en la creación y gestión de logs.(26)

1.9.5. Navegadores

Un navegador o navegador web (del inglés, web browser) es un programa que permite visualizar la información que contiene una página web (ya esté alojada en un servidor dentro de la World Wide Web o en uno local). El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos. La funcionalidad básica de un navegador web es permitir la visualización de documentos de texto, posiblemente con recursos multimedia incrustados.

Mozilla Firefox

Mozilla Firefox es un navegador de software libre. Entre sus características se encuentran que presenta una forma rápida y eficiente de navegar por la web, que le permite abrir varias páginas en una misma ventana mediante el empleo de pestañas separadas. Mozilla Firefox además protege al usuario de la publicidad de ventanas emergentes no solicitadas. (22)

Firefox utiliza además a Firebug que es una extensión creada y diseñada especialmente para desarrolladores y programadores web. Es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, estructura DOM)[2], editar, monitorizar y depurar el código fuente, CSS, HTML y JavaScript de una página web de manera instantánea y online.

[2] DOM: Modelo de Objetos del Documento es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

1.9.6. Control de versiones

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas.

Subversion

Subversion es un sistema de control de versiones libre y de código fuente abierto, es decir, maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. Entre las características que tienen estos sistemas están:

- Entendimiento con los lenguajes de programación.
- Suministro de herramientas para la construcción de software.
- Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros. (23)

TortoiseSVN

Es un cliente de código abierto para el sistema de control de versiones Subversion. Maneja ficheros y directorios a lo largo del tiempo. Es fácil de usar, permite ver el estado de los archivos desde en el explorador de Windows y permite también crear gráficos de todas las revisiones asignadas. (18)

1.10. Conclusiones parciales del capítulo

En este capítulo fueron descritos conceptos y definiciones encargados de trasmitir las especificidades del negocio sobre el que se desarrollará el componente. Fueron también analizadas las funcionalidades y características de otros sistemas de gestión ya existentes, se dieron argumentos de por qué ninguno de estos sistemas cumple con los requisitos necesarios para la gestión de la nómina de los trabajadores en

Cuba de una manera eficiente. Fueron además descritas y se dieron argumentos de por qué fueron seleccionadas las tecnologías, metodologías y herramientas con las cuales se va a desarrollar el trabajo en cuestión, conformando la fundamentación teórica que sustenta el desarrollo del trabajo.

CAPÍTULO #2: DISEÑO E IMPLEMENTACIÓN

2.1. Introducción

Este capítulo recoge los resultados obtenidos en los modelos de negocio y análisis de la solución propuesta. Primeramente se realizará una valoración de los artefactos generados durante el análisis, se lleva a cabo una descripción de la propuesta de solución, se abordarán los patrones de diseño utilizados, se modelarán los diagramas de clases del diseño y el diagrama de componentes que permite una visión más clara de la implementación del sistema. Se explicarán además los modos de implementación y las integraciones entre los componentes que están relacionados con el sistema.

2.2. Análisis de los artefactos entregados por los analistas

El punto de partida para diseñar la solución técnica es realizar un análisis de los artefactos entregados por los analistas, éstos son el Modelo Conceptual y la Descripción de los Requisitos Funcionales (48 requisitos agrupados en 12 procesos). Luego de un análisis de la Descripción de los Requisitos se comprobó que los mismos estaban correctamente redactados, eran claros, posibles de probar y cubrían todas las funcionalidades necesarias en el componente y el enfoque era correcto para un buen funcionamiento y un rendimiento adecuado de la solución, además no se detectaron ocasiones para las malas interpretaciones o falta de información para los desarrolladores ni ambigüedades en dicha especificación. Al no detectarse errores en dichos artefactos no se propusieron cambios. Una vez realizado el análisis de los artefactos entregados se concluyó que la solución propuesta es correcta, clara, completa y consistente, de ahí se procede a modelar el diseño.

2.3. Diseño

Es el proceso de aplicar distintas técnicas y principios con el propósito de definir un sistema con suficiente detalle como para permitir su implementación. Durante el diseño modelamos el sistema y su arquitectura para que soporte los requisitos funcionales y no funcionales. En el modelo de diseño se muestra los diferentes componentes y las relaciones entre ellos. Se especifican las interfaces, clases y sus relaciones agrupadas por subsistemas y paquetes de diseño, así como un resumen de las responsabilidades de cada una de ellas.

2.3.1. Diagrama de componentes

Un diagrama de componentes es modelado por el arquitecto del sistema y representa gráficamente cómo un sistema de software es dividido en componentes mostrando las dependencias existentes entre estos. El diagrama contiene componentes, interfaces, relaciones entre ellos y puede contener paquetes utilizados para agrupar elementos del modelo; mostrando las dependencias lógicas entre componentes software.

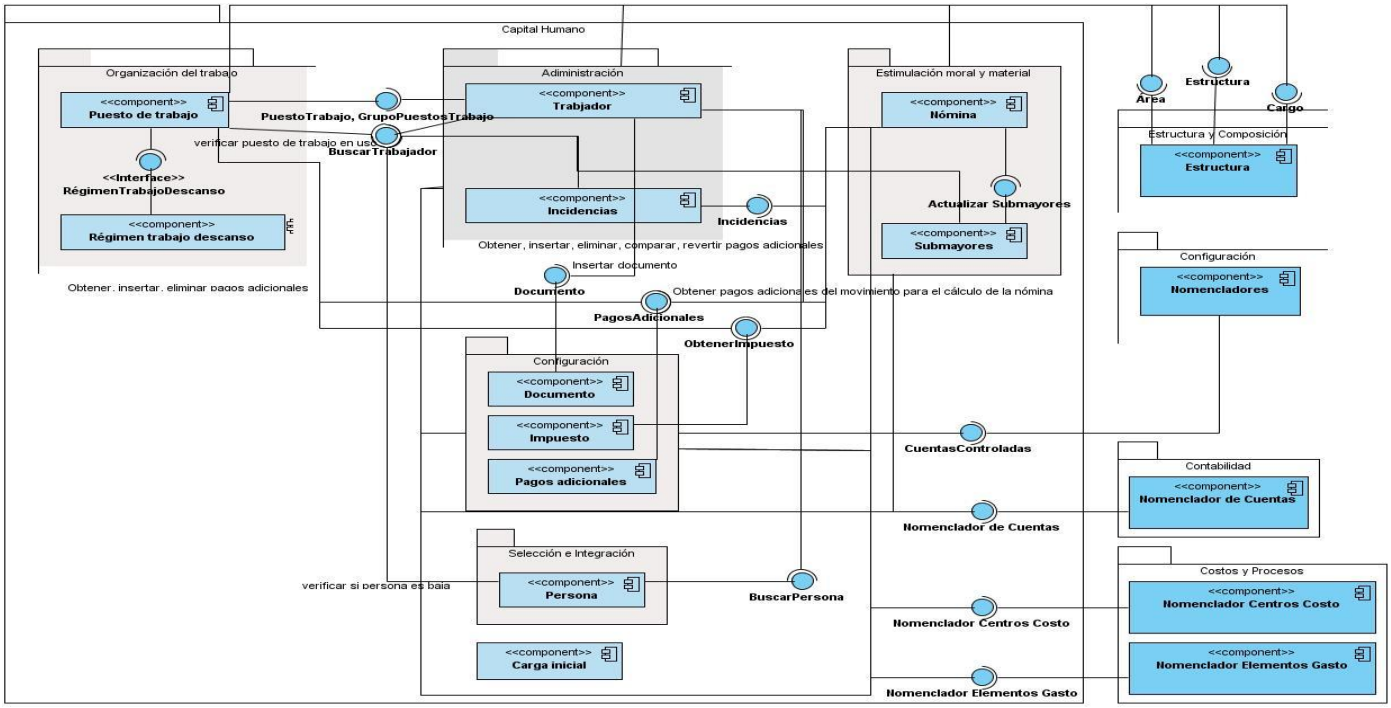

Figura 2.1 Diagrama de componentes

En el diagrama de componentes anterior se puede observar las dependencias que posee el componente Nómina con el resto de los representados, las cuales se describen brevemente a continuación.

- El componente Nómina tiene relación de dependencia con el componente Persona utilizando su interfaz que contiene los datos personales de los trabajadores.
- Se relaciona con el componente Pago adicional utilizando los datos necesarios para realizar el cálculo de los pagos adicionales de cada trabajador.
- Se relaciona con el componente Puesto de trabajo para obtener los pagos adicionales asociados a un puesto, el área, cargo y el salario escala.

- Se relaciona con el componente Trabajador mediante la interfaz movimiento de nómina para obtener el listado actualizado de los trabajadores en la entidad y con la interfaz trabajador para obtener los datos del personal.
- Se relaciona con el componente Incidencia proporcionándole los tipos de nómina para que este a su vez pueda definir los tipos de incidencias y los períodos de pagos con el objetivo que se realice el registro de las incidencias de los trabajadores, este registro utilizado posteriormente para el cálculo de la nómina.
- Se relaciona con el componente Submayores utilizando el registro de retenciones para realizar el descuento de las retenciones durante el procesamiento de la nómina; si la nómina es de vacaciones utiliza la interfaz provisión de vacaciones para realizar el cálculo de las vacaciones de los trabajadores. Luego de realizado el procesamiento actualiza el acumulado de las vacaciones y las retenciones descontadas por cada trabajador.
- Se relaciona con el componente Nomenclador de cuentas del subsistema Contabilidad mediante la interfaz cuentas para realizar el comprobante de operaciones, que luego de terminado se envía a este subsistema.
- Se relaciona con el componente Nomenclador elemento de gasto del subsistema Costos y procesos para contabilizar los conceptos de pagos y la realización del comprobante de operaciones.

2.3.2. Diagramas de clases del diseño

Un diagrama de clase del diseño es un diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos, operaciones y las relaciones existentes entre ellos. Contienen la siguiente información:

- Clases asociadas.
- Métodos
- Navegabilidad
- Dependencias

A continuación se muestran los diagramas de clases del diseño correspondientes al componente Nómina del subsistema Capital Humano del sistema Cedrux.

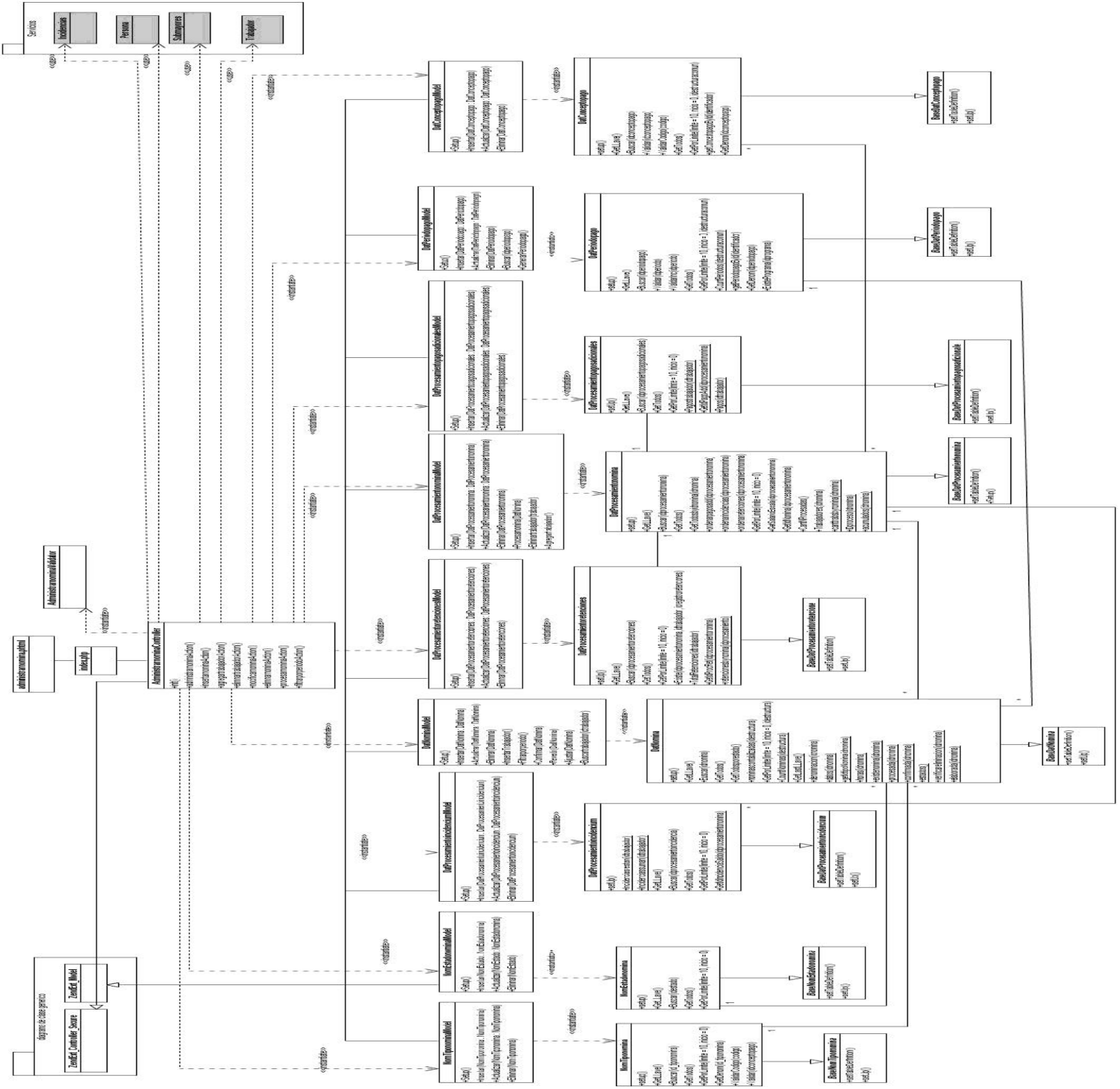

Figura 2.2. Diagrama de clase Administrar nómina.

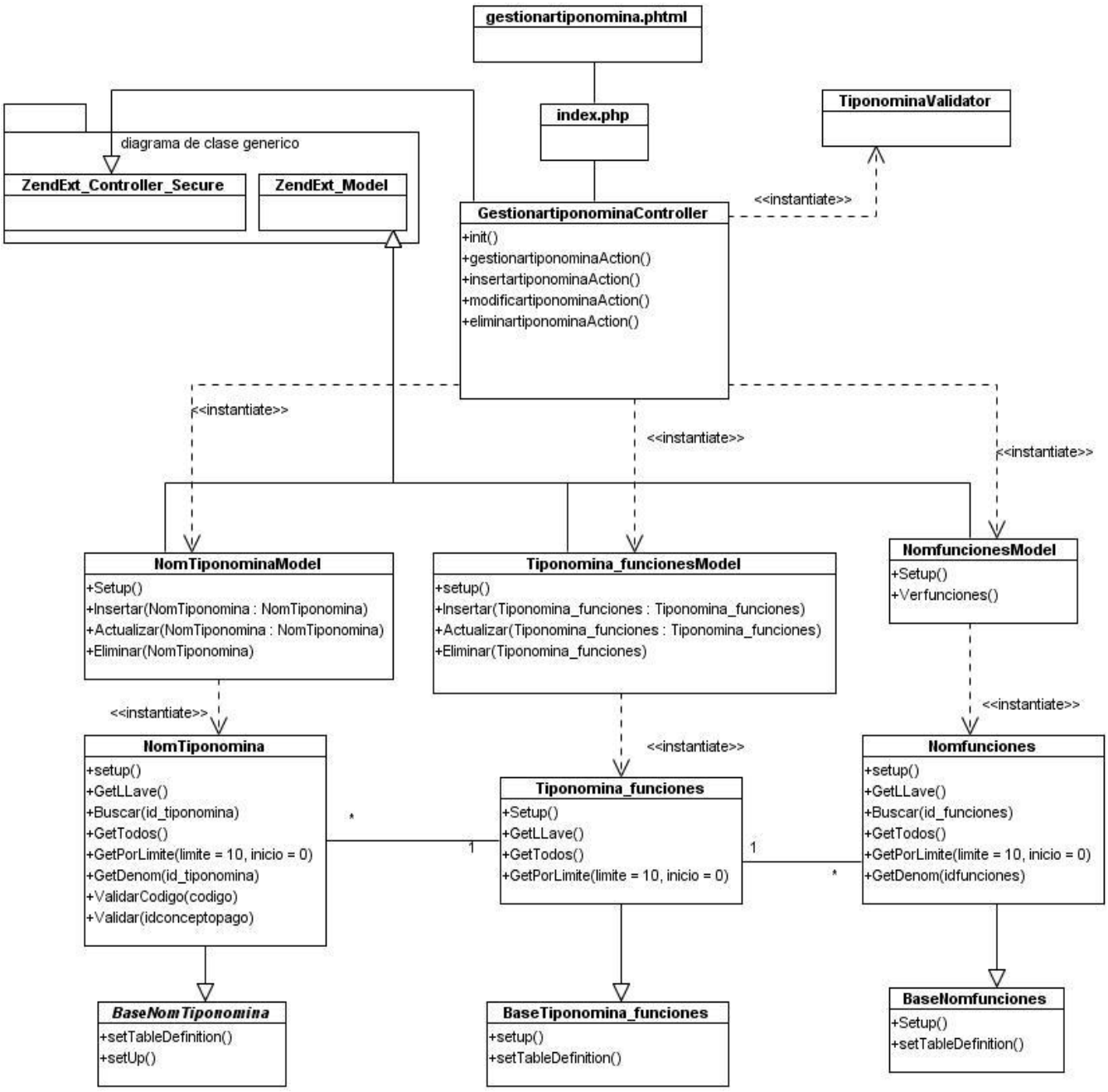

Figura 2.3 Diagrama de clase Gestionar tipo de nómina.

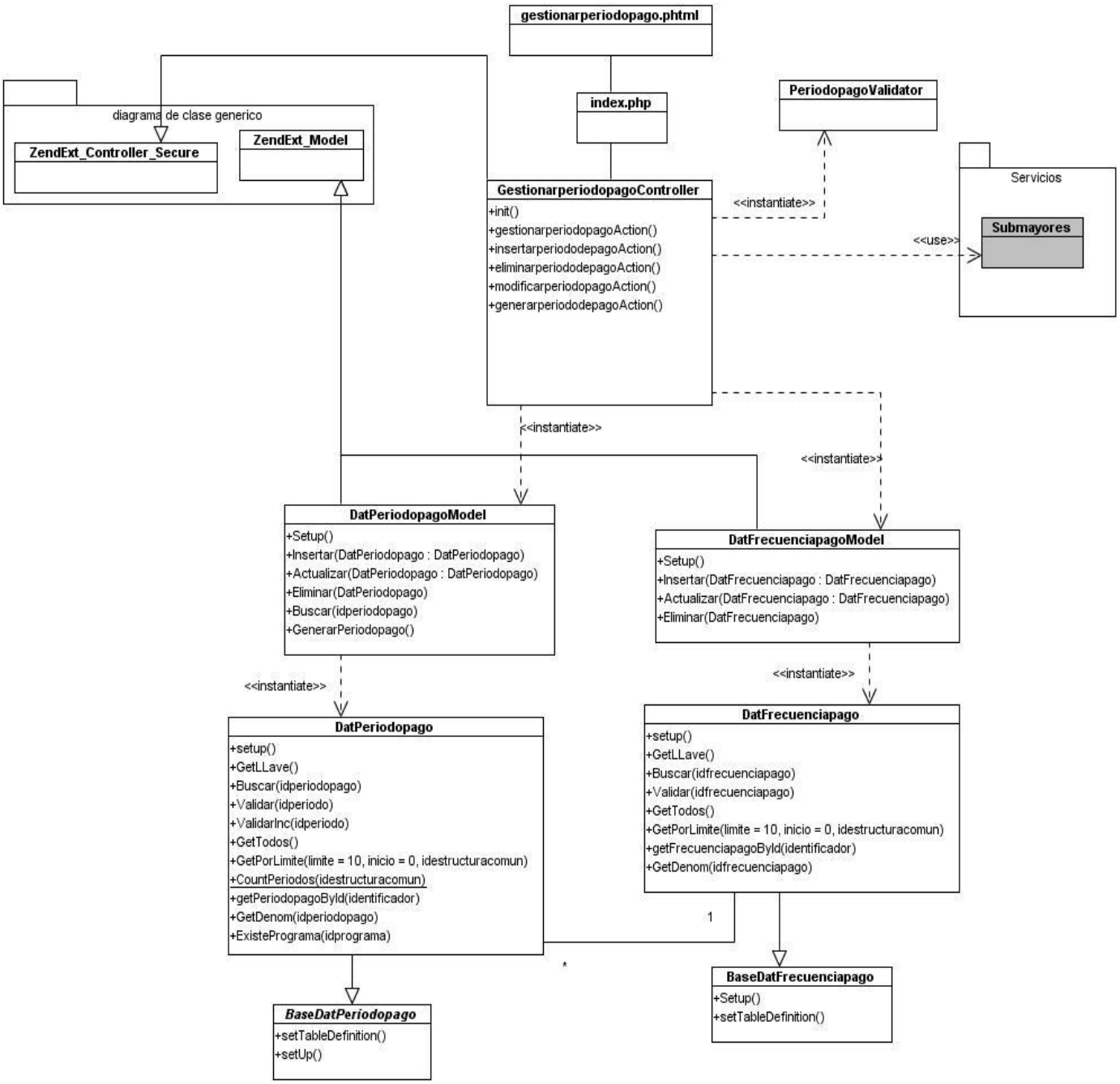

Figura 2.4 Diagrama de clase Gestionar período de pago.

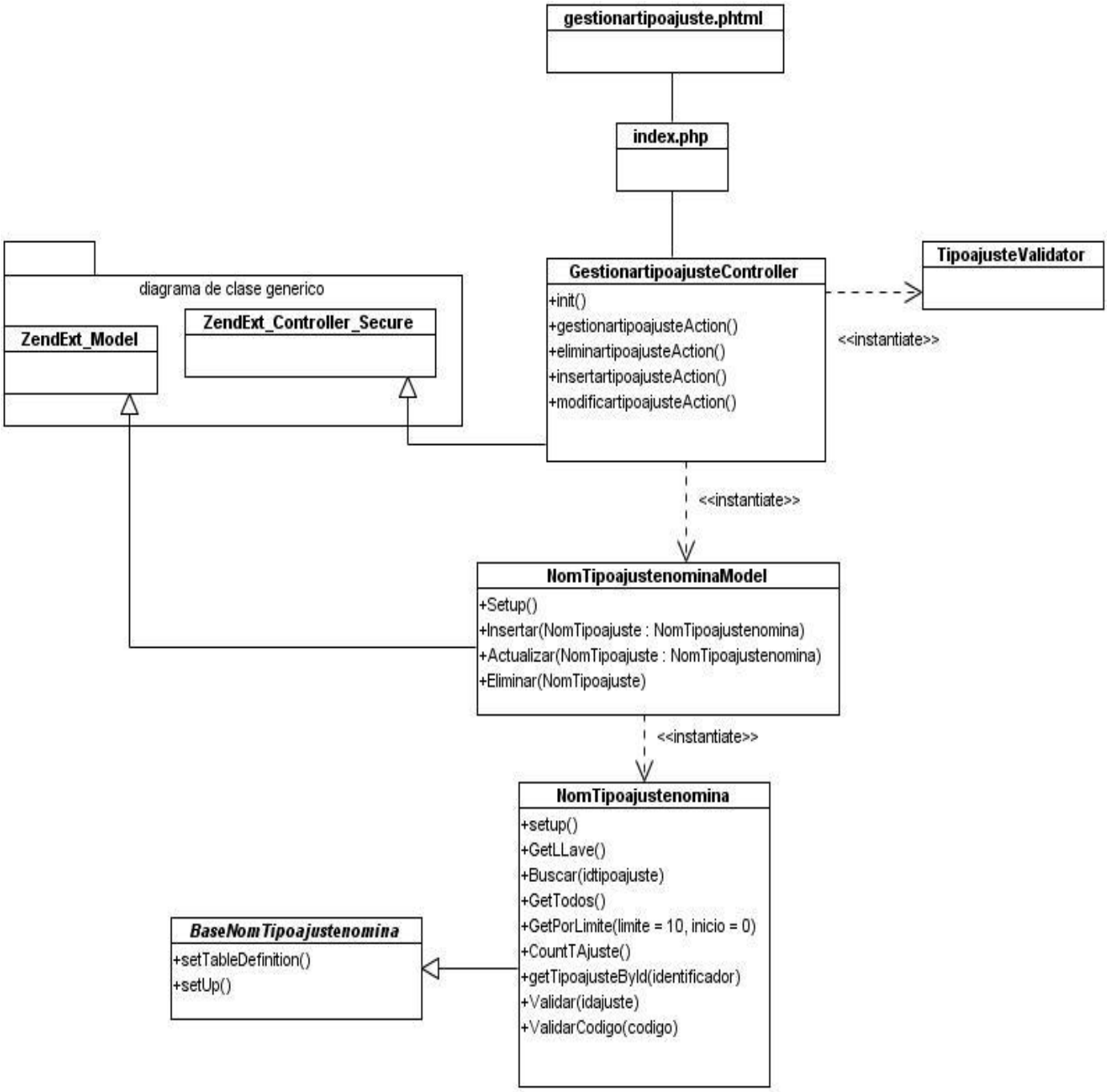

Figura 2.5 Diagrama de clase Gestionar tipos de ajustes.

2.3.3. Patrones de diseño

Para realizar un diseño más eficiente se utilizaron un conjunto de patrones, que al ser experiencias de diseñadores expertos en orientación a objetos permiten dar solución a problemas a través de la codificación del conocimiento y principios existentes, facilitando notablemente el trabajo posterior. Cada patrón describe un problema que ocurre una y otra vez, y luego describe el núcleo de la solución a ese problema. Los patrones que se utilizados fueron: el patrón arquitectónico Modelo – Vista – Controlador (MVC), los patrones de diseño, GRASP para la asignación de responsabilidades y Patrones de Comportamiento.

Patrón Arquitectónico Modelo-Vista-Controlador (MVC)

Por decisión de la dirección del proyecto ERP-Cuba fue empleado en el desarrollo de la solución el patrón de arquitectura de software Modelo-Vista-Controlador, que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Modelo-Vista-Controlador separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes:

Modelo: El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar, sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación.

Controlador: El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador.

Vista: Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo

modelo. Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación.

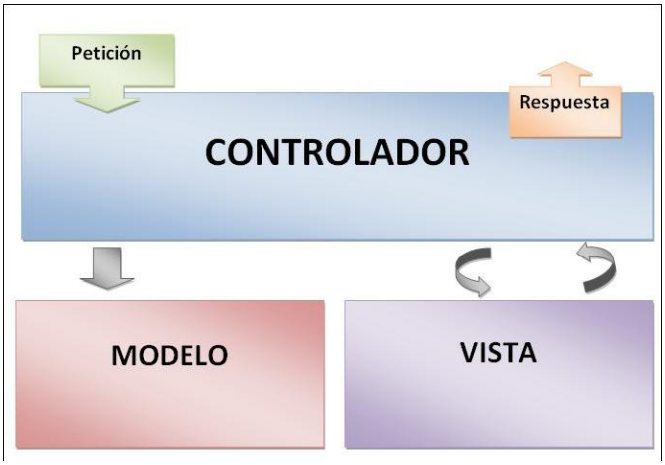

Figura 2.7 Funcionamiento del patrón MVC

Desarrollar una aplicación siguiendo este patrón de diseño tiene muchas ventajas:

- La aplicación esta implementada modularmente.
- Sus vistas muestran información actualizada siempre.
- El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
- Si se desea hacer una modificación al modelo del dominio, como aumentar métodos o datos contenidos, solo debe modificarse el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.
- Las modificaciones a las vistas no afectan en absoluto a los otros módulos de la aplicación.
- MVC está demostrando ser un patrón de diseño bien elaborado pues las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.

El Patrón Arquitectónico Modelo-Vista-Controlador fue utilizado en la solución de la siguiente forma: En las clases del modelo representadas en los diagramas de clases se representa la lógica de negocio. La vista

transforma el modelo en una página Web que permite al usuario interactuar con ella. El controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. Ejemplo:

Un usuario realiza una petición a la clase controladora GestionarnominaController.php la cual procesa la información y solicita los datos en la clase modelo, ejemplo DatNominaModel.php 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 mediante una url en la vista gestionarnomina.js donde se le da la respuesta al usuario.

Patrones de comportamiento

Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal. Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal. Debido a sus funciones se destacan para la solución los siguientes patrones de comportamiento:

Cadena de responsabilidad: Permite la separación entre objetos, pasando la tarea de un objeto al siguiente dentro de una cadena hasta que la tarea es reconocida.

El patrón cadena de responsabilidad se reflejó mediante el tratamiento de errores puesto que a través del lanzamiento de excepciones de las clases y las capturas de las mismas por el aspecto gestor global de excepciones (managerexception) junto al identificador de las mismas (exception), se manifiesta que el manejador específico que satisfará la petición, no se conoce —a priori y se quiere trasladar dicha petición a un objeto de entre unos cuantos, sin especificar explícitamente el receptor final. Ejemplo:
Cuando se le va a adicionar un trabajador a una nómina y el mismo ya existe, la clase DatNominaModel lanza la excepción la cual es capturada por la clase controladora GestionarNominaController y esta a su vez se la muestra al usuario a través de la interfaz GestionarNomina.

Estado: Este patrón permite a un objeto modificar su comportamiento cuando su estado interno cambia. El objeto aparecerá para cambiar su clase. En operaciones grandes, de varias declaraciones

condicionales que dependen sobre estado del objeto. Este estado suele ser representado por uno o más enunciados constantes. A menudo, varias operaciones de esta misma contendrán una estructura condicional. El patrón Estado pone cada rama de la condicional en una clase separada. Esto le permite tratar el estado del objeto como un objeto en su derecho propio, que puede variar independientemente de otros objetos.

Este es aplicado en las validaciones realizadas utilizando la clase GestionarnominaValidator, donde se llama un grupo de funciones según sea la validación que se quiere realizar teniéndose en tiempo de ejecución la construcción de un nuevo conjunto de operaciones.

Patrones estructurales

Fachada: Simplifica los accesos a un conjunto de objetos relacionados, funciona como intermediario entre uno y otro grupo objetos, se utiliza para comunicar estos dos grupos.

Tipo: Estructura, a nivel de objetos.
Propósito: Simplificar el acceso a un conjunto de clases o interfaces. Proporcionar una interfaz unificada de alto nivel que representa a todo un subsistema facilitando su uso. La fachada satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.
Motivación: Reducir la dependencia entre clases Fachada ofrece un punto de acceso al resto de clases, si éstas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente. No oculta las clases sino que ofrece una forma más sencilla de acceder a ellas, en los casos en que se requiere se puede acceder directamente a ellas. En el desarrollo de esta solución la utilización del patrón estructural Fachada se ve en que cada componente tiene un paquete denominado servicios donde se encuentran las clases que en este caso serían NominaService.php y PeriodoPagoService.php. Estas clases contienen los servicios que brindan estos componentes a otros a través del IoC interno a los componentes internos (del mismo subsistema) y a través del IoC externo a otros componentes de otros subsistemas. La Fachada sería entonces el IoC tanto interno como externo, ya que funciona como intermediario para la comunicación entre componentes.

2.4. Implementación

La implementación se empieza con el resultado del diseño y se implementa la solución en término de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables, y similares.

La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo trabajo de implementación durante la fase de elaboración, para crear la línea base ejecutable de la arquitectura, y durante la fase de transición para tratar defectos tardíos.

Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no necesariamente es la que demanda mayor trabajo ni la más complicada. La complejidad y la duración de esta etapa está íntimamente relacionada al o a los lenguajes de programación utilizados, así como al diseño previamente realizado.

2.4.1. Estándares de codificación

Un estándar de codificación son reglas que se siguen para la escritura del código fuente. De tal manera que otros programadores se les facilite entender tu código (como identificar las variables, las funciones o métodos, etc.). Aseguran que todos los programadores trabajen de forma coordinada y mantengan un código de calidad. Si se aplica de forma continuada un estándar de codificación bien definido caben muchas posibilidades de que un proyecto de software se convierta en un sistema de software fácil de comprender y de mantener.

Los estándares de codificación en el marco del proyecto Cedrux van a permitir una mejor integración entre las líneas de producción y se establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo. El módulo Nómina se rige por estos estándares, pues desde el punto de vista arquitectónico estandarizan el código a implementar.

Los estándares que a continuación se especifican fueron definidos por la dirección del proyecto ERP-Cuba.

Nomenclatura de las clases

El nombre de las clases se escriben la primera letra con mayúsculas y las demás con minúsculas en caso de ser un nombre compuesto se empleará notación PascalCasing, es seguido del primer nombre y tiene la misma estructura que el primer nombre. Ejemplo: Reporte, ReporteNómina.

Nomenclatura de las clases según el tipo

Las Controladoras que se encuentran dentro de la carpeta “controller”: Las clases controladoras después de escribir el nombre según su nomenclatura se le añade “Controller”.
Ejemplo: GestionarnominaController.

Las Modelos que se encuentran dentro de la carpeta “bussines”. Las clases modelos después de escribir el nombre según su nomenclatura se le añade “Model”. Ejemplo: DatNominaModel.

Las clases Dominios que se encuentran dentro de la carpeta “domain”. Las clases dominios son generadas por el marco de trabajo Doctrine y se llaman igual que la tabla de la base de datos. Ejemplo: DatNomina.

Las clase bases que se encuentran dentro de la carpeta “generated”. Las clases bases son generadas por el marco de trabajo Doctrine y delante del nombre de la clase se le añade “Base”.
Ejemplo: BaseDatNomina.

Nomenclatura de los métodos o funciones

El nombre de los métodos de una clase comienzan con minúsculas, en caso de que sea compuesto se empleará notación CamelCasing, seguido del primer nombre comienza el segundo con la primera letra en mayúscula. Deben describir el propósito del mismo. Ejemplos: insertar(), insertarNomina(). Si es una función de una clase controladora después del nombre se le agrega la palabra “Action”. Ejemplo: insertarNominaAction().

Nomenclatura de las variables

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

Nomenclatura de los atributos

El nombre a emplear para los atributos se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleara notación CamelCasig. Además en caso de ser un objeto se comienza con:"_" y después se escribe el nombre.

Ejemplo: intnomina = nomina

objNomina =_nomina

2.4.2. Diseño de la Base de Datos

Uno de los pasos cruciales en la construcción de una aplicación que maneje una base de datos, es sin duda el diseño de la base de datos. En la base de datos que se utiliza en los componentes que se están desarrollando como propuesta de solución, para lograr el acceso eficiente a la información con redundancia mínima, se toman varias consideraciones, entre las que se encuentran: la velocidad y facilidad de acceso a la información para extraerla.

Modelo de datos

El modelo de datos representa los datos en el más bajo nivel y permite identificar algunos detalles de implantación para el manejo del hardware de almacenamiento, proporciona referencias de cómo se almacenan los datos en la computadora, el formato de los registros, la estructura de los ficheros (ordenados, desordenados) y los métodos de acceso utilizados (índices). Los conceptos de este modelo están dirigidos fundamentalmente al personal informático, no a los usuarios finales.

El modelo de datos propuesto es una representación de las tablas existentes en la base de datos así como las relaciones entre ellas. El mismo cuenta con 30 tablas que representan de manera general el negocio del componente Nómina. A continuación se muestra el modelo de datos:

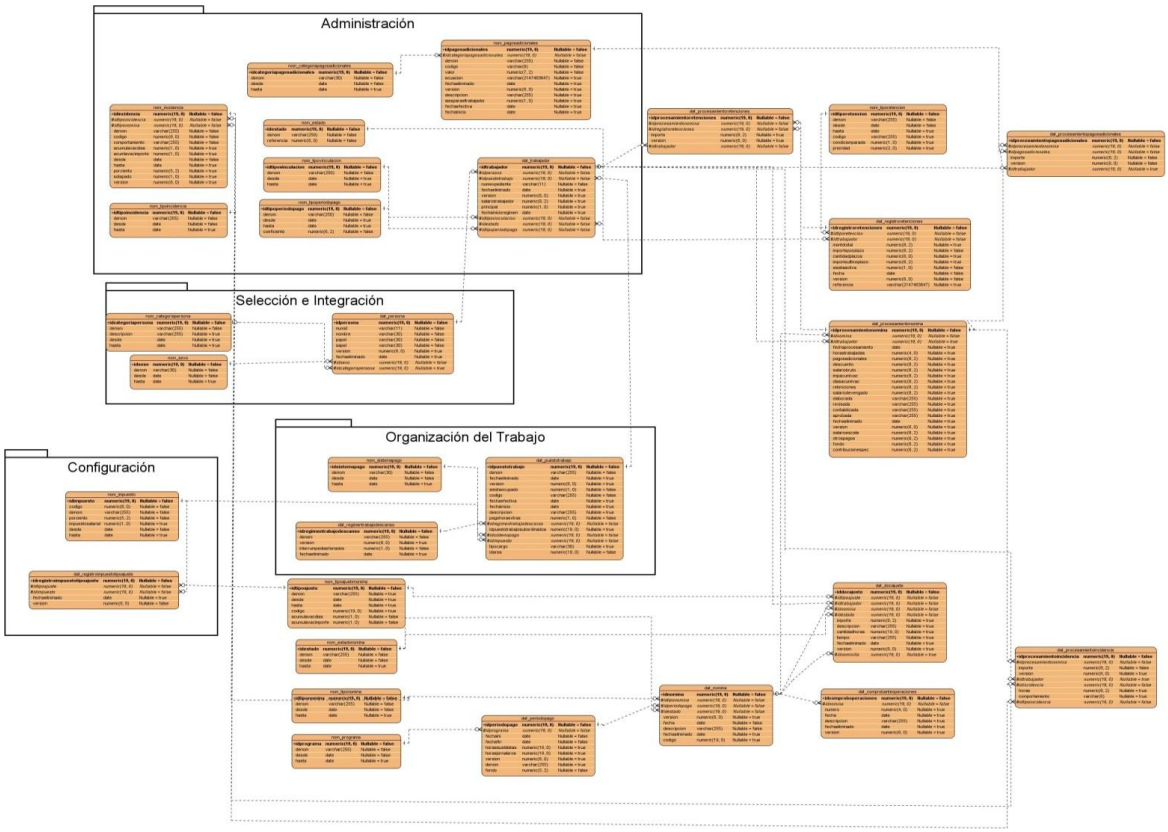

Figura 2.8 Modelo de datos

2.4.3. Soluciones específicas

Se asignan responsabilidades muy específicas a cada método con el objetivo de reutilizarlo cada vez que sea necesario.

Modo de implementación en la capa de datos:

Con el objetivo de lograr una mayor optimización de las consultas a la base de datos así como una mayor rapidez en el acceso a los datos se implementaron las consultas de la siguiente forma:

En el acceso a datos se implementan la menor cantidad de consultas posibles, generalmente una para recuperar datos y otra para contar la cantidad de resultados encontrados, parametrizando las consultas, es decir, que la misma consulta reciba muchos parámetros y se construya el filtro en dependencia de cada uno de los parámetros recibidos al invocar el método, teniendo en cuenta además que el tiempo de

respuesta del servidor es mucho menor cada vez que se ejecuta una misma consulta, por tanto el rendimiento de la aplicación es considerablemente más efectivo de esta forma en lugar de implementar una consulta por cada posible parámetro de entrada.

Modo de implementación en las vistas:

Con el objetivo de evitar sobrecarga en el servidor y consumo de tiempo durante la ejecución de las vistas, se llevó a cabo la implementación de las mismas de la siguiente forma:

Al cargar la interfaz principal las peticiones al servidor se realizan una vez que se obtiene respuesta de la anterior, es decir se realiza una petición y cuando esta se ejecute se realiza la siguiente petición en lugar de realizarlas todas de una vez, lo que puede sobrecargar el servidor y generar errores y/o demora en las respuestas del controlador. Todos los mensajes se declaran en el json, de esta forma del controlador se envían números o booleanos por cada petición realizada desde la vista, contribuyendo así a la internacionalización de la solución.

2.4.4. Descripción de funcionalidades

En este epígrafe se realizará una breve descripción de las funcionalidades implementadas.

Gestionar Nómina

En este componente se confecciona las nóminas correspondientes a Salarios, Subsidios, Vacaciones así como las nóminas adicionales que desee elaborar la entidad en forma automatizada, en el período que se establezca, con las incidencias, procesada por el área de personal y de las retenciones de los trabajadores.

Las nóminas van pasando por un proceso y a lo largo del mismo van tomando diferentes estados. Estos estados son Elaborado en un inicio cuando se crean y se le agregan los trabajadores, Procesado cuando se realizan los cálculos de los salarios, Confirmado cuando ya están listas y revisadas y Contabilizado cuando se ha enviado el comprobante de operaciones a Contabilidad.

La Figura 2.17 muestra la vista que se muestra cuando accede a la funcionalidad Gestionar nómina.

Figura 2.9 Gestionar nóminas

La vista principal mostrará las nóminas existentes y sus principales atributos: Fecha, Descripción, Tipo y Estado. Brinda las opciones de Adicionar, Procesar, Modificar, Eliminar, Detalles y Confirmar.

Crear una nómina

Al seleccionar la opción Adicionar se muestra la siguiente pantalla:

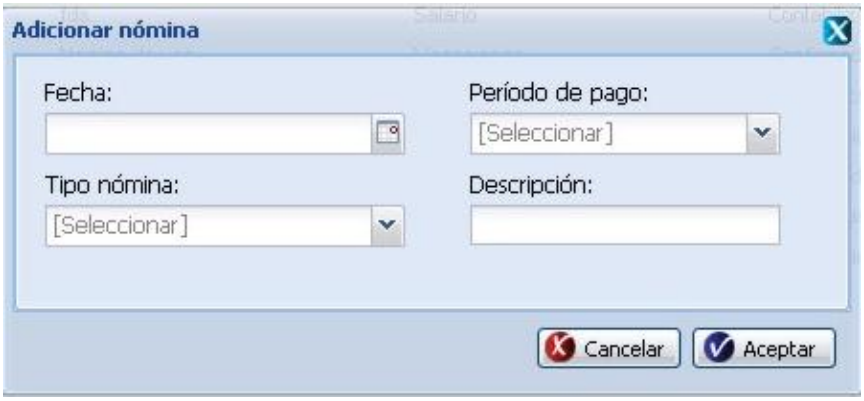

Figura 2.10 Crear nómina

Aquí se introducen los datos necesarios para crear una nómina. Finalmente se añaden a la lista la nómina que se creó haciendo un clic en el botón Aplicar o Aceptar. Seguidamente se mostrará la pantalla Buscar trabajador para añadir los trabajadores a la nómina.

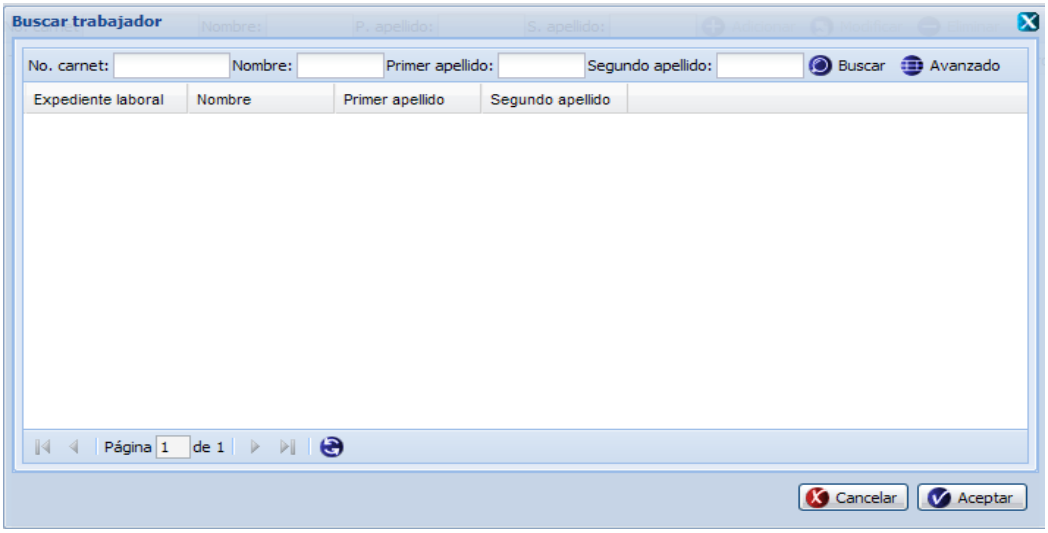

Figura 2.11 Trabajadores de la nómina

La diferencia en esta funcionalidad es que se puede seleccionar la cantidad de trabajadores que se desean agregar a la nómina utilizando la opción Avanzado y una vez seleccionado el lote de trabajadores y aceptado la búsqueda se mostrará un mensaje de información, que mostrará la cantidad de trabajadores que se han añadido. Los trabajadores se pueden añadir de dos formas, uno por uno o en lotes, o se puede también combinar ambos métodos, por ejemplo, utilizando adiciones en lotes que respondan a ciertas caracterizaciones y luego agregar uno a uno ciertos casos individuales. Esto puede hacerse varias veces hasta que se tenga a todos los trabajadores de la nómina seleccionada.

Una vez añadido los trabajadores en la nómina la misma aparece en la lista de nóminas existentes con el estado de Elaborada.

Modificar una nómina

Si se quiere modificar una nómina de las creadas se selecciona la misma y posteriormente se va a la opción Modificar que activará una pantalla similar a la de Adicionar pero con todos los campos del elemento cargados. Se modifican los campos que se quieren cambiar y se presiona Aplicar o Aceptar para confirmar la operación. Se mostrará un mensaje de información.

Eliminar una nómina

Para eliminar una nómina se selecciona la dicha nómina y posteriormente se va a la opción Eliminar. Cuando se haya terminado de realizar la operación se mostrará un mensaje de confirmación. Si se selecciona Aceptar se mostrará un mensaje de información.

2.4.5. Integración entre componentes

En los componentes se aplica la integración vertical que consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatros nodos de integración, el que encontramos entre la vista y el controlador, el que está entre el controlador y el modelo, el que vincula el modelo con el framework doctrine y el que se encuentra entre el doctrine y la base de datos. Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes módulos y componentes se realiza mediantes llamadas a la inversión de control (Inversion of Control en inglés, IoC). El IoC especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Cada componente tiene su registro de los datos de los módulos en un fichero xml con el nombre IoC que en este se publican los métodos o servicios que se brindan a los demás componentes del subsistema (servicios internos) y/o a otros subsistemas (servicios externos), que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IoC y registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. La base de datos es accedida de forma directa mediante controladoras y los componentes rehusados son integrados mediantes interfaces sencillas. Cada componente tiene una clase Service que se encarga de publicar los métodos o servicios que se le puede brindar a otras líneas o componentes. Para acceder a los servicios externos se llama al objeto integrator, el nombre del subsistema y el nombre del método publicado en el IoC externo (que se encuentra en el paquete comun del sistema). Para acceder a los servicios internos se llama al objeto pIntegrator, el nombre del componente y el nombre del método publicado en el IoC interno (que se encuentra en el paquete comun del propio subsistema). Ejemplo:

$area = $this->integrator->metadatos->DameAreasPorId ($datospuestotrabajo->idarea);

$datos = $this->pIntegrator->trabajador->ObtenerDatosTrabajador ($doc->idtrabajador);

Para una visión gráfica de la integración de componentes consulte el diagrama de componentes.

2.4.6. Publicación de servicios

El modelo de publicación de servicios que se siguió está basado en la misma filosofía del modo de implementación en la capa de datos, es decir, publicar la menor cantidad de servicios posible, implementando cada servicio (siempre que sea posible) de manera que reciba muchos o ningún parámetro, publicando en las clases Services los mismos métodos que se utilizan en las clases model o bussines para recuperar datos en cada uno de los componentes con el objetivo de una mayor reutilización de código.

La solución cuenta con dos clases para la publicación de los servicios que se deben de brindar a otros componentes con los que se tiene una interrelación y que se encuentran tanto dentro como fuera del mismo subsistema. Las mismas son NominaService y PeriodoPagoService.

A continuación se muestran los servicios que brinda el componente:

Tabla 2.2 Servicios que brinda Nómina

Componente	Servicios que brinda	Componentes que lo utilizan	Código	Descripción
Nómina	ObtenerTipoNominas	Incidencias	CHNomina02	
	ObtenerPeriodoPago		CHNomina04	
	HorasPeriodoPago		CHNomina05	Obtener horas dado un período de pago
	ExistePrograma	Submayores	CHNomina03	
	ObtenerPeriodoPago		CHNomina04	Obtener todos los períodos de pago
	DenominacionPorId		CHNomina06	Obtener denominación de nómina dado id
	DenominacionIdNominas		CHNomina07	
	ConfirmacionNominas	Cierre	CHNomina01	

2.5. Conclusiones parciales del capítulo

A modo de conclusión se puede decir que en éste capítulo se abarcaron los aspectos más significativos del diseño e implementación del componente Nómina del sistema Cedrux, se expusieron los patrones de diseño utilizados, también se elaboraron artefactos como los diagramas de clases de diseño, que permitieron desarrollar de manera más eficiente la implementación del componente y el diagrama de componentes que permitió una visión más clara de la implementación del sistema, y de las integraciones entre los componentes que están relacionados con el mismo. Se hizo una explicación de los modos de implementación tanto de las vistas como de acceso a datos y las integraciones entre los componentes que están relacionados con el sistema.

CAPÍTULO #3: VALIDACIÓN Y PRUEBAS

3.1. Introducción

En el presente capítulo se hace una valoración crítica del diseño propuesto abordándose temas como las métricas para evaluar la calidad del diseño de software y se valida mediante pruebas el software realizado, haciendo uso específico de las pruebas de caja blanca y caja negra. Se aborda además el modelo de pruebas en el cual se especifican los casos de prueba que componen el sistema.

3.2. Evaluación del modelo de diseño propuesto

A partir del diseño obtenido para la solución fue necesario llevar a cabo una evaluación del mismo al componente Nómina del subsistema Capital Humano integrado al sistema Cedrux. Para la evaluación de la calidad del diseño se hizo un estudio de la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto; en el mismo se abarcan atributos de calidad 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. Puede influir indirecta, pero fuertemente en los costos y la planificación del proyecto.
- Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño del componente Nómina del subsistema Capital Humano integrado al sistema Cedrux y su relación con los atributos de calidad definidos son las siguientes:

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

Las métricas escogidas son bastante eficientes ya que dan una medida de la calidad del diseño del componente y su utilización es sencilla y fácil.

3.2.1. Tamaño Operacional de Clase (TOC)

El tamaño operacional de clase (TOC), está dado por el número de métodos asignados una clase.

Tabla 3.1 Tamaño operacional de clase (TOC)

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

Resultados de la evaluación de la métrica

El instrumento y la tabla de resultados, se muestran en el Anexo 1.

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, teniendo en cuenta que el 80% del las clases pertenecientes al componente Nómina posee menos cantidad de operaciones que el doble del promedio registrado en las mediciones, y que el 89% de las clases de dicho componente poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización), podemos concluir que el diseño es aceptable.

3.2.2. Relaciones entre Clases (RC)

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

Tabla 3.2 Relaciones entre clases (RC)

Atributo que afecta	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
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.

A continuación se presentan los resultados obtenidos a partir de la evaluación de los instrumentos aplicados a las métricas descritas anteriormente así como una valoración de los mismos.

Resultados de la evaluación de la métrica

El instrumento y la tabla de resultados, se muestran en el Anexo 2.

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del módulo Nómina tiene una calidad aceptable para realizar la implementación, teniendo en cuenta que el 85% de las clases incluidas en el módulo poseen menos de 3 dependencias de otras clases. Además de que el 80% no tiene o es bajo el acoplamiento entre clases. Por último los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización poseen niveles aceptables en un 83,3 % de las clases.

3.3. Pruebas de software

Las pruebas del software son un elemento crítico para la garantía de la calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. Las pruebas son realizadas con el objetivo de detectar errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista. Los casos de prueba especifican una forma de probar el sistema, incluyendo la entrada o resultado con el que se ha de probar y las condiciones bajo las que ha de probarse.

Con el propósito de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente. Entre sus objetivos se encuentran:

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.
- Reducir costos de mantenimiento.
- Obtener información concreta acerca de fallas, que pueda usarse como apoyo en la mejora de procesos, y en la de los desarrolladores.

Si se producen modificaciones en el programa, habrá que probar de nuevo todas las partes afectadas por las modificaciones. Por lo dicho anteriormente, se desarrolló este epígrafe, con el objetivo de especificar las pruebas que serán hechas al software y describir todo lo encontrado en las mismas.

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.

Las pruebas de caja negra están definidas en el plan de aseguramiento de la calidad del proyecto ERP-Cuba como pruebas a aplicar por este departamento. Algunos métodos empleados en las pruebas de caja negra son:

Partición equivalente

Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

Análisis de valores límite

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el análisis de valores límite lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse solamente en las condiciones de entrada, el análisis de valores límite obtiene casos de prueba también para el campo de salida.

Condiciones de ejecución:

- El usuario debe estar autenticado en el sistema.
- Se debe seleccionar menú Inicio.
- Se debe seleccionar Sub-sistemas.
- Se debe seleccionar Capital Humano.
- Se debe seleccionar Remuneración - Nómina.
- Se debe seleccionar Administrar nómina.
- Se debe seleccionar la opción Administrar nómina.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario

1. Crear nómina.	Se crea una nómina.	EP 1.1: Crear nómina introduciendo los datos correctamente.	– Se debe presionar el botón **Adicionar**. – Se debe introducir los datos de la nómina en el formulario. – Se debe presionar el botón **Aceptar**.
		EP 1.2: Crear nómina dejando campos requeridos en blanco.	– Se debe presionar el botón **Adicionar**. – Se debe introducir los datos de la nómina en el formulario dejando al menos un campo requerido en blanco. – Se debe presionar el botón **Aceptar**.
		EP 1.3: Crear nómina introduciendo errores en los datos.	– Se debe presionar el botón **Adicionar**. – Se debe introducir los datos de la nómina en el formulario introduciendo algunos datos inválidos. – Se debe presionar el botón **Aceptar**.

		EP 1.4: Cancelar operación.	– Se debe presionar el botón **Adicionar**. – Se debe introducir los datos de la nómina o no en el formulario. – Se debe presionar el botón **Cancelar**.

Tabla 3.3 Descripción del caso de prueba para el requisito Crear nómina.

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, comprobación de bucles.
Los tipos de pruebas de caja blanca son:

Prueba de Condición

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

Prueba de Flujo de Datos

Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles

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

Prueba del Camino Básico

Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

De las pruebas anteriormente mencionadas al sistema desarrollado se aplicó la prueba de caja blanca, específicamente la técnica de prueba del camino básico, ya que lo que se quiere evaluar es la complejidad lógica del diseño para obtener una medición cuantitativa de la lógica de los segmentos de código. 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 también 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.

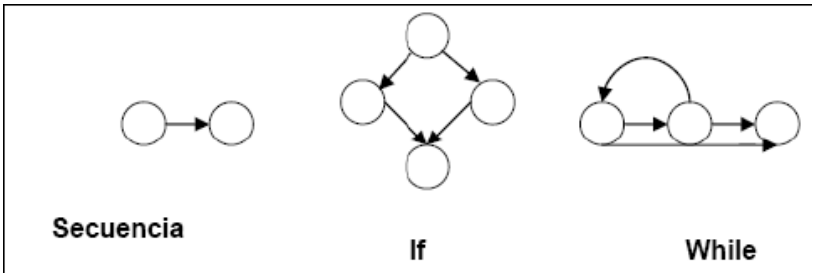

Figura 3.1 Notación de grafos de flujo para las instrucciones: Secuenciales, If, While

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 enumera las sentencias de código del procedimiento realizado sobre el método gestionarretenciones ($retencions,$trabajador,$datpnomina).

```
private function gestionarretenciones($retenciones,$trabajador,DatProcesamientonomina $datpnomina) {
    if (count ( $retenciones ) > 0) {//1
        foreach ( $retenciones as $retencion ) {//2
            $noprocesada = DatProcesamientoretencione::Existe($datpnomina->idprocesamientonomina,
                                                              $datpnomina->idtrabajador,
                                                              $retencion->idregistroretenciones);//3

            if ($datpnomina->salariobruto > $retencion->importe && $noprocesada) {//4
                $datretenciones = new DatProcesamientoretencione ( );//5
                $datretenciones->idprocesamientonomina = $datpnomina->idprocesamientonomina;//5
                $datretenciones->idregistroretenciones = $retencion ->idregistroretenciones;//5
                $datretenciones->importe = $retencion->importe;//5
                $datretenciones->idtrabajador = $trabajador ['idtrabajador'];//5
                $datretenciones->idcuenta = $retencion->idcuenta;//5
                $datretenciones->save();//5

                $cantidad = $datpnomina->salariobruto;//5
                $descuento = $cantidad - $retencion->importe;//5
                $datpnomina->salariobruto = $descuento;//5
                $datpnomina->descuento = $datpnomina->descuento + $descuento;//5
                $datpnomina->retenciones = $datpnomina->retenciones + $retencion->importe;//5
                $datpnomina->save();//5
            }//6
        }//7
    }//8
}//9
```

Figura 3.2 Representación del algoritmo gestionarretenciones ($retenciones, $trabajador, $datpnomina).

A continuación se representa el Grafo de flujo asociado al algoritmo anteriormente descrito:

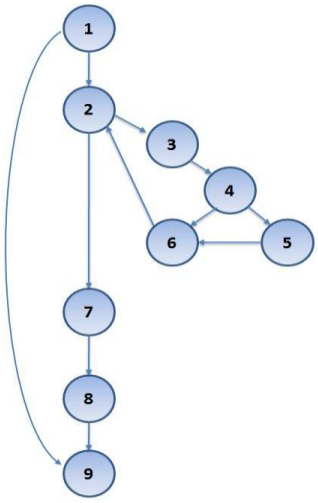

Figura 3.3 Grafo de flujo asociado al algoritmo gestionarretenciones ($retenciones, $trabajador, $datpnomina).

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) = (11 – 9) + 2

V (G) = 4

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

Se puede usar también:

V (G) = P + 1

V (G) = 3 + 1

V (G) = 4

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

V (G) = R

V (G) = 4

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo. El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 4, lo que significa que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 3.4 Caminos básicos del flujo

Número	Camino básico
1	1 – 2 – 3 – 4 – 5 - 6 – 2 – 7– 8 – 9
2	1 – 2 – 3 – 4 – 6 – 2 – 7– 8 – 9
3	1 – 2 – 7 – 8 – 9
4	1 – 9

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

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

Caso de prueba para el camino básico 1

Camino 1: [1 – 2 – 3 – 4 – 5 - 6 – 2 – 7– 8 – 9]

Descripción: Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro $retenciones no está vacío, contiene datos el registro de retención de un trabajador.

El arreglo $trabajador no está vacío, contiene los datos de un trabajador.
El arreglo $datpnomina no está vacío, contiene los datos del procesamiento de la nómina.

Entrada:
$retenciones =array (_idregistroretenciones' =>455555, _importe' =>3, _prioridad' =>1, _idcuenta' =>34344).
$trabajador = array (_ idprocesamientonomina ' => 500000034, _ idtrabajador ' => 500000049,
_ salariodevengado ' => 1).
$datpnomina = array (_ idprocesamientonomina ' => 500008, _ idnomina ' => 545454 ,
_ idtrabajador '=> 534, _ fondo' =>28, _idcuenta '=>4545, _idcentrocosto'=>4545).

Resultados esperados: Se espera que se le realice la gestión de las retenciones del trabajador.

Caso de prueba para el camino básico 2

Camino 2: [1 – 2 – 3 – 4 – 6 – 2 – 7– 8 – 9]

Descripción: Los datos de entrada cumplirán con los siguientes requisitos:
El parámetro $retenciones no está vacío, contiene datos el registro de retención de un trabajador.
El arreglo $trabajador no está vacío, contiene los datos de un trabajador.
El arreglo $datpnomina no está vacío, contiene los datos del procesamiento de la nómina.

Entrada:
$retenciones =array (_idregistroretenciones'=>455555, _importe'=>3, _prioridad'=>1, _idcuenta' =>34344).
$trabajador = array (_ idprocesamientonomina '=> 500000034, _ idtrabajador '=> 500000049,
_ salariodevengado '=> 1).
$datpnomina = array (_ idprocesamientonomina '=> 500008, _ idnomina '=> 545454,
_ idtrabajador '=> 534, _ fondo' =>28, _idcuenta '=>4545, _idcentrocosto'=>4545).

Resultados esperados: Se espera que no se le realice la gestión de las retenciones del trabajador, ya que a pesar de contiene datos el registro de retención de dicho trabajador la nómina no ha sido procesada o el salario bruto del trabajador es menor que las retenciones que este posee.

Caso de prueba para el camino básico 4:

Camino 4: [1 – 9]

Descripción: Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro $retenciones está vacío, no contiene datos el registro de retención de un trabajador.

Entrada:

$retenciones = array ().

Resultados esperados: Se espera que no se le realice la gestión de las retenciones del trabajador, ya que no posee retenciones.

Se aplicaron los métodos de caja negra y caja blanca para validar tanto la interfaz como el correcto funcionamiento interno del software. Estas pruebas no se ejecutaron de forma aislada sino paralelo al desarrollo del software abarcando requerimientos, funciones y lógica interna del programa. Combinar ambos enfoques permitió lograr mayor fiabilidad en el proceso de pruebas al diseñar los casos de prueba usando los dos tipos de técnicas a la vez.

3.4. Conclusiones parciales del capítulo

En el capítulo se hace una valoración crítica del diseño propuesto mediante el uso de las métricas (TOC y RC) obteniéndose resultados aceptables. Se aborda acerca de las pruebas de software, haciéndose énfasis en las pruebas de menor escala tales como pruebas de caja blanca y caja negra y las particularidades de cada una de ellas. Se hace una se validación del software realizado utilizándose la prueba del camino básico, en la cual se obtuvieron resultados favorables, demostrándose con ello la calidad y eficiencia del componente Nómina del subsistema Capital Humano del sistema Cedrux.

CONCLUSIONES

Durante el desarrollo del presente trabajo se llevaron a cabo una serie de fases que permitieron dar cumplimiento al objetivo planteado y que abarcaron todas las tareas investigativas propuestas.

- A partir del análisis de los procesos de nómina y sistemas contables vinculados a la actividad de gestión de Capital Humano se demostró la necesidad e importancia de desarrollar un sistema informático para realizar la gestión de los procesos de nómina de forma eficiente en las entidades cubanas.
- En tal sentido, se realizó un estudio de las herramientas y tecnologías que posibilitó entender y avalar la selección de técnicas utilizadas. Se efectuó el diseño e implementación del módulo Nómina bajo los estándares definidos en el proyecto ERP-Cuba, probado y validado mediante métricas y pruebas de software.
- De acuerdo con el objetivo general propuesto, se obtuvo el componente Nómina como producto configurable y totalmente funcional que cumple con los requerimientos descritos, que es capaz de gestionar de manera integral los procesos de nómina que se llevan a cabo en las entidades cubanas y se adapta a las necesidades del proceso de informatización en el país. El componente posibilitará realizar de forma eficiente y ágil el cálculo para el pago a los trabajadores en las entidades nacionales, integrándose con otros subsistemas necesarios para su correcto funcionamiento.
- La investigación realizada contribuye al mejoramiento de los procesos contables y financieros de las entidades cubanas y constituye un aporte decisivo a la informatización y economía del país.

RECOMENDACIONES

Luego de haber cumplimentado los objetivos propuestos mediante la realización del trabajo, se recomienda:

- Identificar nuevos requisitos para actualizar el sistema desarrollado y continuar realizando pruebas para garantizar una mayor operatividad y aceptación del cliente.
- Desplegar el sistema en entidades del territorio nacional.

BIBLIOGRAFÍA

1. *Ayuda del Versat Sarasola; Version 2.0 actualizacion 5.8. .*
2. *Cuba, Organismo Nacional de Normalización de la República de. Norma Cubana 3000. .* Ciudad de La Habana : s.n., 2007.
3. *Precios, Ministerio de Finanzas. Resolución 13. .* 2007.
4. **Rojas, Ariza y García, Maribel y Molina y Carlos, J.** INTRODUCCIÓN Y PRINCIPIOS BÁSICOS DEL DESARROLLO DE SOFTWARE BASADO EN COMPONENTES. [En línea] 2007. http://pegasus.javeriana.edu.co/~jcpymes/Docs/DSBC.pdf...
5. Guía Breve de Tecnologías XML. [En línea] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML. .
6. Manuales, Tutoriales y Herramientas. . [En línea] http://max-alva.webs.com/javascript.htm...
7. Geek OPEN TI. [En línea] http://geeknet.com.mx/principal.php?op=detalle1...
8. ¿Qué es un framework? [En línea] http://jordisan.net/blog/2006/que-es-un-framework/...
9. Guía del Programador de PostgreSQL. [En línea] https://forja.rediris.es/docman/view.php/312/461/Postgres-Programmer.pdf...
10. ¿Qué es Mozilla Firefox? [En línea] http://firefoxperu.blogspot.com/2005/12/qu-es-mozilla-firefox.html. ..
11. **Larman, Craig.** UML y Patrones. [En línea] http://bibliodoc.uci.cu...
12. Zend Framework, una introducción. . [En línea] 2009. http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/...
13. What is Doctrine? [En línea] http://www.doctrine-project.org/...
14. Universidad Catolica Boliviana San Pablo. . [En línea] 04 de Abril de 2009. http://www.ucbcba.edu.bo...
15. **Gracia, Joaquín.** Patrones de diseño. [En línea]. [En línea] http://www.ingenierosoftware.com...
16. JSON, acrónimo de —JavaScript Object Notation||. [En línea] http://felipengineer.wordpress.com/2009/01/26/json-acronimo-de-javascript-object-notation/…
17. **Rodas Hinostroza, Raúl.** . Características de PHP. [En línea] 2009. http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP...
18. **Sussman, Collins y y otros.** . Control de versiones con Subversion. [En línea] 2008. http://svnbook.red-bean.com/nightly/es/svn-book.pdf...
19. Departamento de Ingeniería de Sistemas. . [En línea] 03 de Abril de 2009. http://sophia.javeriana.edu.co...
20. El ataque de los frameworks. . [En línea] 2009. http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php...
21. **Markiewicz, y otros.** El Desarrollo del Framework Orientado al. [En línea] http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html...
22. El Mundo Informático. . [En línea] 05 de Abril de 2009. http://jorgesaavedra.wordpress.com...
23. El proyecto IDEE. . [En línea] http://www.ineter.gob.ni/sig/docs/Intro%20IDE%20-%20IDEE.pdf...
24. EMS SQL Manager para PostgreSQL. . [En línea] http://sqlmanager.net/en/products/postgresql/manager...
25. Glosario. . [En línea] http://www.pixelmio.net/glosario.php...
26. Guía Breve de CSS. [En línea] [En línea] http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo...
27. Introducción a Apache. . [En línea] 2008. http://linux.ciberaula.com/articulo/linux_apache_intro/...
28. Introducción, definición y evolución de PHP. . [En línea] http://php.ciberaula.com/articulo/introduccion_php/...
29. New Features. . [En línea] 2008. http://extjs.com/blog/2007/09/28/ext-20-alpha-release/...

30. Facultad de Ciencias Exactas, Físico-Químicas y Naturales . [En línea] 04 de Abril de 2009. http://dc.exa.unrc.edu.ar...
31. UPNA(Universidad publica de Navarra). . [En línea] 07 de Junio de 2009. http://www.ayc.unavarra.es...
32. msdn . Microsoft Corporation. [En línea] 05 de Junio de 2009. http://msdn.microsoft.com/es-es/...
33. Mitecnologico. [En línea] 08 de Junio de 2009. http://www.mitecnologico.com. ..
34. **Pressman, Roger.** *Ingeniería de software. Un enfoque práctico.* s.l. : McGraw.Hill/Interamericana de España., 2002.
35. **Rojas y Barrios, J. E.** Métodos de prueba de caja negra. [En línea] 2008. http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html]..
36. Object Management Group - UML. [En línea] 22 de Febrero de 2010. http://www.uml.org/..
37. *Equipo de producción. Modelo de Desarrollo orientado a componentes del proyecto ERP -CUBA. .* 2009.
38. **Asleson, Ryan & Schutta y Nathaniel T.** *. Foundations of Ajax. Primera Edición. Apress.* s.l. : 296 p. 1-59059-582-3, 2008. 978-1590595824.
39. **Crane, y otros.** *Ajax in Action. Greenwich, Manning Publications Co.,.* 2008.
40. **Heilmann, Christian.** *Beginning JavaScript with DOM Scripting and Ajax.* 2008.

ANEXOS

Anexo 1 - Instrumento de medición de la métrica Tamaño operacional de clase (TOC).

Tabla 1 Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

	Categoría	**Criterio**
Responsabilidad	Baja	< =Prom.(4.5)
	Media	Entre Prom. Y 2* Prom.
	Alta	> 2* Prom.
Complejidad 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 2 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

No	**Componente**	**Clase**	**Cantidad de Procedimientos**	**Responsabilidad**	**Complejidad**	**Reutilización**
1	Nómina	BaseDatNomina	2	Baja	Baja	Alta
2	Nómina	BaseDatProcesamiento nomina	2	Baja	Baja	Alta
3	Nómina	BaseDatRegistroimpuestotipoajuste	2	Baja	Baja	Alta

4	Nómina	BaseNomTipoajustenomina	2	Baja	Baja	Alta
5	Nómina	BaseDatComprobanteoperaciones	2	Baja	Baja	Alta
6	Nómina	BaseDatPeriodopago	2	Baja	Baja	Alta
7	Nómina	BaseDatProcesamientopagosadicionale	2	Baja	Baja	Alta
8	Nómina	BaseNomEstadonomina	2	Baja	Baja	Alta
9	Nómina	BaseNomTiponomina	2	Baja	Baja	Alta
10	Nómina	BaseDatDocajuste	2	Baja	Baja	Alta
11	Nómina	BaseDatProcesamientoincidencium	2	Baja	Baja	Alta
12	Nómina	BaseDatProcesamientoretencione	2	Baja	Baja	Alta
13	Nómina	BaseNomPrograma	2	Baja	Baja	Alta
14	Nómina	DatDocajusteModel	4	Baja	Baja	Alta
15	Nómina	DatProcesamientoincidenciumModel	4	Baja	Baja	Alta
16	Nómina	DatRegistroimpuestotipoajusteModel	4	Baja	Baja	Alta
17	Nómina	NomEstadonominaModel	4	Baja	Baja	Alta
18	Nómina	ComprobanteNomina	6	Media	Media	Media
19	Nómina	DatNominaModel	4	Baja	Baja	Alta
20	Nómina	DatProcesamientonominaModel	4	Baja	Baja	Alta
21	Nómina	INomina	4	Baja	Baja	Alta
22	Nómina	NomTipoajustenominaM	4	Baja	Baja	Alta

		odel				
23	Nómina	DatComprobanteoperacionesModel	4	Baja	Baja	Alta
24	Nómina	DatPeriodopagoModel	5	Media	Media	Media
25	Nómina	DatProcesamientopagosadicionaleModel	4	Baja	Baja	Alta
26	Nómina	IPeriodoPago	2	Baja	Baja	Alta
27	Nómina	NomTiponominaModel	4	Baja	Baja	Alta
28	Nómina	DatDocajuste	12	Alta	Alta	Baja
29	Nómina	DatProcesamientoincidencium	9	Media	Media	Media
30	Nómina	DatProcesamientoretencione	9	Media	Media	Media
31	Nómina	NomPrograma	10	Alta	Alta	Baja
32	Nómina	DatNomina	22	Alta	Alta	Baja
33	Nómina	DatProcesamientonomina	20	Alta	Alta	Baja
34	Nómina	DatRegistroimpuestotipoajuste	5	Media	Media	Media
35	Nómina	NomTipoajustenomina	10	Alta	Alta	Baja
36	Nómina	DatComprobanteoperaciones	6	Media	Media	Media
37	Nómina	DatPeriodopago	14	Alta	Alta	Baja
38	Nómina	DatProcesamientopagosadicionale	8	Media	Media	Media
39	Nómina	NomEstadonomina	5	Media	Media	Media
40	Nómina	NomTiponomina	7	Media	Media	Media

Anexo 2- Instrumento de medición de la métrica Relaciones entre clases (RC).

Tabla 3 Rango de valores de para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

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

Tabla 4 Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas).

No	Componente	Clase	Cantidad de Relaciones	Acoplamiento	Complejidad	Reutilización	Cantidad de Pruebas

1	Nómina	BaseDatNomina	0	Ninguna	Baja	Alta	Baja
2	Nómina	BaseDatProcesamientonomina	0	Ninguna	Baja	Alta	Baja
3	Nómina	BaseDatRegistroimpuestotipoajuste	0	Ninguna	Baja	Alta	Baja
4	Nómina	BaseNomTipoajustenomina	0	Ninguna	Baja	Alta	Baja
5	Nómina	BaseDatComprobanteoperaciones	0	Ninguna	Baja	Alta	Baja
6	Nómina	BaseDatPeriodopago	0	Ninguna	Baja	Alta	Baja
7	Nómina	BaseDatProcesamientopagosadicionale	0	Ninguna	Baja	Alta	Baja
8	Nómina	BaseNomEstadonomina	0	Ninguna	Baja	Alta	Baja
9	Nómina	BaseNomTiponomina	0	Ninguna	Baja	Alta	Baja
10	Nómina	BaseDatDocajuste	0	Ninguna	Baja	Alta	Baja
11	Nómina	BaseDatProcesamientoincidenciu m	0	Ninguna	Baja	Alta	Baja
12	Nómina	BaseDatProcesamientoretencion e	0	Ninguna	Baja	Alta	Baja
13	Nómina	BaseNomPrograma	0	Ninguna	Baja	Alta	Baja
14	Nómina	DatDocajusteModel	1	Baja	Media	Media	Media
15	Nómina	DatProcesamientoincidenciumMo del	1	Baja	Media	Media	Media
16	Nómina	DatRegistroimpuestotipoajusteM odel	1	Baja	Media	Media	Media
17	Nómina	NomEstadonominaModel	1	Baja	Media	Media	Media
18	Nómina	ComprobanteNomina	8	Alta	Alta	Baja	Alta
19	Nómina	DatNominaModel	1	Baja	Media	Media	Media
20	Nómina	DatProcesamientonominaModel	1	Baja	Media	Media	Media
21	Nómina	INomina	2	Media	Alta	Baja	Alta

22	Nómina	NomTipoajustenominaModel	1	Baja	Media	Media	Media
23	Nómina	DatComprobanteoperacionesModel	1	Baja	Media	Media	Media
24	Nómina	DatPeriodopagoModel	1	Baja	Media	Media	Media
25	Nómina	DatProcesamientopagosadicionaleModel	1	Baja	Media	Media	Media
26	Nómina	IPeriodoPago	1	Baja	Media	Media	Media
27	Nómina	NomTiponominaModel	1	Baja	Media	Media	Media
28	Nómina	DatDocajuste	3	Alta	Alta	Baja	Alta
29	Nómina	DatProcesamientoincidencium	2	Media	Alta	Baja	Alta
30	Nómina	DatProcesamientoretencione	2	Media	Alta	Baja	Alta
31	Nómina	NomPrograma	0	Ninguna	Baja	Alta	Baja
32	Nómina	DatNomina	4	Alta	Alta	Baja	Alta
33	Nómina	DatProcesamientonomina	3	Alta	Alta	Baja	Alta
34	Nómina	DatRegistroimpuestotipoajuste	0	Ninguna	Baja	Alta	Baja
35	Nómina	NomTipoajustenomina	1	Baja	Media	Media	Media
36	Nómina	DatComprobanteoperaciones	0	Ninguna	Baja	Alta	Baja
37	Nómina	DatPeriodopago	1	Baja	Media	Media	Media
38	Nómina	DatProcesamientopagosadicionale	2	Media	Alta	Baja	Alta
39	Nómina	NomEstadonomina	1	Baja	Media	Media	Media
40	Nómina	NomTiponomina	1	Baja	Media	Media	Media

GLOSARIO DE TÉRMINOS

Componente: Es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Configuración: Es una adaptación a condiciones definidas por el entorno, que le es hecha a una aplicación.

Devengado: Derecho ganado que todavía no ha sido cobrado.

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

Funcionalidad: Representa la forma en que un dispositivo funciona; es decir, los mecanismo o secuencia de eventos que hacen que el objeto realice cierta función.

Módulo: Es un componente autocontrolado de un sistema, el cual posee una interfaz bien definida hacia otros componentes. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará una de dichas tareas (o quizás varias en algún caso).

Nomenclador: Es una especie de catálogo que tiene la nomenclatura, o vocabulario de una rama.

Proceso: Conjunto de actividades enlazadas entre sí que, partiendo de una o más entradas las transforman en salidas.

Pagos Adicionales: Son formas de pagos adicionales que se expresan en términos de por cientos, de importes fijos o de importes proporcionales.

Subsistema: Cada uno de los componentes principales de un sistema que este dividido en componentes. Cada subsistema abarca aspectos del sistema que comparten alguna propiedad común.

Subsidio: Se refiere a los ingresos que recibe el trabajador en sustitución del salario, cuando se enferma o accidenta.

Sistema: Es un conjunto de procesos o elementos interrelacionados con un medio para formar una totalidad encauzada hacia un objetivo común.