Universidad de las Ciencias Informáticas

Facultad 15

Título: "Diseño e Implementación de los componentes: Pagos adicionales, Puesto de Trabajo y Movimiento de Fuerza de Trabajo del subsistema Capital Humano."

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

Autor(es): Carlos Miguel Pedroso Caraballo
Yosniel Ramos O'farrill

Tutor(es): Ing. Rosendo Leonardo Hernández Claro

Co-tutor: Ing. Arnolis Salgueiro Arzuaga

Ciudad de la Habana, Junio 2010.

"Año 52 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:

_____ _____

Carlos Miguel Pedroso Caraballo Yosniel Ramos O'farrill

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. Ha impartido las asignaturas Matemática I y II así como Idioma Extranjero I, II, III y IV como alumno ayudante. 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. Durante el curso 2008-2009 atendió dos tesis como tutor y oponente respectivamente. Durante el curso 2008-2009 ocupó el rol de arquitecto principal de la línea Capital Humano del proyecto ERP. Actualmente ocupa el rol de Jefe de Factoría de la línea Capital Humano.

Correo electrónico: rlhernandez@uci.cu

AGRADECIMIENTOS

Quiero agradecer a todos mis compañeros del proyecto que de alguna forma u otro influyeron en el desarrollo del mismo.

A mi tutor Rosendo.

A mi compañero de tesis Carlos Miguel y a su novia Leticia.

A mi esposa Yamilé.

Yosniel

Carlos Miguel

DEDICATORIA

Este trabajo está dedicado a la revolución y en especial a nuestro comandante en jefe que ha hecho posible que lleguemos hasta aquí.

A mi tía Marlén.

A mi esposa Yamilé.

A mis padres Roberto y María.

Yosniel

Carlos Miguel

RESUMEN

Los procesos de movimiento de fuerza de trabajo y los puestos de trabajo en las entidades empresariales y presupuestadas no se gestionan de la forma más eficiente porque los sistemas existentes implantados no cumplen con las exigencias de las empresas cubanas. La realización de estos procesos de forma manual ocasiona la pérdida de la documentación de los movimientos de nómina de los trabajadores, descontrol en la plantilla de trabajadores en las entidades y cúmulo de documentación, dificultando la claridad del trabajo. Con el objetivo de mejorar los procesos referentes a los trabajadores y los puestos de trabajo en las entidades del territorio nacional se propone desarrollar los componentes Movimiento de Fuerza de Trabajo, Pagos Adicionales y Puesto de Trabajo, que forman parte del Subsistema Capital Humano del Sistema Integral de Gestión Cedrux. De esta forma dentro del subsistema Capital Humano del proyecto ERP-Cuba, después de analizar los requisitos elaborados por los analistas se diseñó e implementó una solución que automatiza los procesos de movimiento de fuerza de trabajo y puestos de trabajo cumpliendo con los requisitos de las empresas cubanas.

PALABRAS CLAVE

Capital Humano, componente, pagos adicionales, proceso, puesto de trabajo, movimiento de fuerza de trabajo.

Tabla de Contenido

Introducción

El mundo actual evoluciona a una velocidad vertiginosa y el uso de las tecnologías de la información y las comunicaciones (TIC) ha sido indispensable para su permanente desarrollo. Las TIC se han desarrollado con el fin de mejorar la calidad de vida de las personas en los diferentes entornos y están presentes en la educación, la salud, la cultura, la economía y en el resto de las esferas que constituyen la sociedad.

Para poder triunfar en el mercado las empresas necesitan gestionar con mayor eficiencia los procesos que se llevan a cabo en ellas, dígase: la gestión de la cadena de aprovisionamiento y fabricación, proyectos, finanzas, gestión de Recursos Humanos, marketing, ventas y procesos de servicio. El área de Recursos Humanos (RH) es considerada dentro de las más importantes en una empresa, es en ella donde se tiene como propósito garantizar la eficiencia y la eficacia de la participación del personal en el éxito empresarial. Como parte de la administración de RH se tienen en cuenta por lo menos los siguientes elementos: reclutamiento del personal, la selección de personal, la evaluación del desempeño de personal, los sistemas de compensación, el plan de beneficio social, la higiene y seguridad en el trabajo, y la capacitación y desarrollo de personal. Formando parte de los procesos que se llevan a cabo dentro de la compleja y amplia área de RH se encuentran los movimientos de fuerza de trabajo y puestos de trabajo, que a su vez están compuestos por subprocesos como: formalización de la relación laboral, movimientos de nómina, definición de puestos de trabajo, régimen de trabajo y descanso y pagos adicionales. Estos procesos de gestión en gran parte de las entidades presupuestadas y empresariales del gobierno cubano se llevan a cabo de forma manual y semiautomática, archivando la información en formato duro (papel), esto provoca demora, pérdida y duplicación de información, que los reportes estadísticos no sean los más confiables, la cantidad de documentación que se maneja dificulta la operatividad del trabajo; de esta forma el desarrollo de estos procesos se torna engorroso y complejo, trayendo consigo el mal funcionamiento de las entidades en general, sin dejar de mencionar que los RH han alcanzado un gran desarrollo y se hace cada vez más compleja su administración. Existen en Cuba sistemas certificados que de una u otra forma alcanzan a gestionar parcialmente los procesos de movimiento de fuerza de trabajo y puestos de trabajo, los mismos están enfocados hacia objetivos muy específicos y ninguno de ellos logra una gestión integral que permita realizar un análisis detallado de la amplia gama de aspectos que caracterizan estos procesos en un entorno laboral. Como parte del amplio proceso de automatización de la

economía cubana que se viene desarrollando en todo el país y la importancia de contribuir con la independencia tecnológica, la importancia de identificar y explotar las reservas de productividad del trabajo para poder satisfacer las exigencias requeridas en la legislación laboral vigente requiere una mediación informática.

De lo anteriormente explicado resulta como **problema a resolver:** Los sistemas de capital humano existentes no proporcionan todas las funcionalidades necesarias para realizar los procesos de movimiento de fuerza de trabajo y puestos de trabajo cumpliendo con los requisitos de las empresas cubanas.

Teniendo en cuenta el problema planteado, queda enmarcado el **objeto de estudio** en los procesos de Capital Humano en las entidades cubanas, delimitando el **campo de acción** a los procesos de movimiento de fuerza de trabajo y puestos de trabajo en las entidades cubanas.

El **objetivo general** para darle solución al problema anteriormente expuesto es: Diseñar e implementar los procesos de movimiento de fuerza de trabajo y puestos de trabajo siguiendo la legislación nacional, para satisfacer las necesidades de las entidades cubanas.

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

- Realizar un estudio del estado del arte.
- Diseñar e implementar los componentes Trabajador, Puesto de trabajo y Pagos adicionales.
- Validar la solución.

Con la finalidad de dar cumplimiento al objetivo general se proponen realizar las siguientes **tareas de investigación:**

- Estudiar sistemas existentes, nacionales y extranjeros.
- Estudiar cómo se realizan los procesos de movimiento de fuerza de trabajo y puestos de trabajo en Cuba.
- Estudiar los requisitos funcionales definidos para los componentes movimiento de fuerza de trabajo, puestos de trabajo y pagos adicionales.
- Modelar la solución de software a través del diseño de clases.

- Implementar la solución bajo los estándares definidos por el proyecto ERP-Cuba.
- Realizar pruebas a la solución.

Idea a defender

Realizándose el diseño e implementación de los componentes Pagos adicionales, Puesto de trabajo y Trabajador, siguiendo las normas nacionales, se lograría una mejor gestión de los movimientos de fuerza de trabajo y puestos de trabajo en las entidades cubanas.

Los **posibles resultados** son: Un sistema que gestione los procesos de movimiento de fuerza de trabajo y puestos de trabajo, las bases arquitectónicas de integración a nivel de componentes y con otros subsistemas, así como la documentación del proceso investigativo vinculado a la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA: En este capítulo se abordan el estado del arte así como otros sistemas existentes, fundamentación de algunos conceptos generales con el objetivo de un mejor entendimiento de los procesos, el estudio de las metodologías, tecnologías y herramientas empleadas para el desarrollo de la solución propuesta.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN: Se analizan los artefactos entregados por los analistas, y a partir del resultado obtenido de dicho análisis se diseña la solución a implementar. Se modela el diagrama de componentes de la solución propuesta. Se identifican los patrones de diseño a utilizar, se modelan los diagramas de clases del diseño permitiendo dar paso a la implementación del sistema, se representa además el diagrama de datos para mostrar las tablas vinculadas al sistema y sus relaciones. Se implementa la solución y se realizan las integraciones entre los componentes que están relacionados con el sistema.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN. PRUEBAS: 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.

Introducción

En el presente capítulo se perfilarán los conceptos fundamentales que servirán de base para el desarrollo del trabajo. Además se realizará una investigación en base a sistemas existentes que están vinculados a los proceso de gestión de los trabajadores, así como se analizarán las metodologías, tecnologías y las herramientas utilizadas para el desarrollo del sistema propuesto.

1.1. Conceptos Generales

Recursos Humanos

“La ciencia que se dedica a la administración de empresas agrupa al conjunto de los empleados y colaboradores de una organización bajo el concepto de recursos humanos. Ese mismo nombre recibe el departamento o la persona que se encarga de seleccionar, contratar, formar y retener a los trabajadores de una empresa.” (1)

Capital Humano

El concepto de CH tiene una trascendencia a través de la historia dado su significado, hace muchos siglos atrás sería interpretado su nombre como referencia a los fondos que se utilizarían para la puesta en funcionamiento de una empresa.

“Aún así en la actualidad existen disímiles concepciones e interpretaciones ya sean de carácter económico o no de acuerdo al concepto de Capital Humano, de esta manera es muy común referirse al Capital Humano como las cualidades y características de las personas de una organización, o sea sus aspectos intangibles, por ejemplo, experiencia, capacidades, formación, educación, escolarización, conocimiento, salud, condiciones de vida y trabajo e información, y cuando se trata de los aspectos tangibles en la organización como cantidad, salario, contratación y demás, entonces se consideran recursos humanos”. (2)

“El Capital Humano es un conjunto de conocimientos, experiencias, habilidades, sentimientos, actitudes, motivaciones, valores y capacidad para hacer, portados por los trabajadores para crear más riquezas con eficiencia. Es, además, conciencia, ética, solidaridad, espíritu de sacrificio y heroísmo”. (3)

Trabajador

Un trabajador es una persona que con la edad legal suficiente y que voluntariamente preste sus servicios retribuidos por cuenta ajena y dentro del ámbito de organización y dirección de otra persona, física o jurídica, denominada empleador o empresario. (4)

La Ley Federal del Trabajo en su artículo 8 plantea que un trabajador es la persona física que presta a otra, física o moralmente, un trabajo personal subordinado. Para los efectos de esta disposición, se entiende por trabajo toda actividad humana, intelectual o material, independientemente del grado de preparación técnica requerido por cada profesión u oficio. (5)

Puestos de Trabajo

Como base conceptual para el desarrollo de este trabajo ya que se refiere al Puesto de Trabajo de una forma más concreta y específica se toma la definición de Puesto de Trabajo concretada en la emitida por la Oficina Nacional de Normalización, donde refiriéndose al cargo como equivalente al puesto lo define como: los que aparecen recogidos en los calificadores comunes, de rama o actividad y propios de organismos, pertenecientes a las diferentes categorías ocupacionales, así como en resoluciones. En ellos se definen la denominación, contenido de trabajo y los requisitos para ocuparlos y expresan el empleo u oficio que desempeñan, los trabajadores en la organización. (3)

Pagos Adicionales

En las relaciones contractuales que se establecen entre los empleadores y los trabajadores, mediante los contratos de trabajo y/o convenios colectivos que rigen en las empresas, se incorporan una serie de partidas económicas al margen del salario base y formando parte del salario total, que tienen varias denominaciones y cuyo objetivo es aumentar la remuneración a cambio de más producción, o mitigar lo negativo de trabajar en determinadas condiciones. En el ámbito de la economía cubana estas partidas económicas son conocidas como Pagos Adicionales y en el ámbito mundial como pluses salariales y complementos del salario, estas denominaciones incluyen todas aquellas retribuciones monetarias que sin formar parte del salario base o salario escala son atribuidas al trabajador a causa de determinadas condiciones que tienen que ver con su persona, puesto de trabajo o resultados de la entidad.

“El salario base es el pago por complejidad y responsabilidad aprobado para cada ocupación o cargo, según el grupo de la escala salarial vigente”. (3)

Los pagos adicionales varían de acuerdo al marco económico vigente, pero sin importar el mismo se destacan algunos como:

- Pago de nocturnidad: cuando se trabaja en horario nocturno.

- Pago de productividad o de beneficios: Se otorga generalmente en función de los resultados económicos de la entidad.
- Pago de transporte: cuando la empresa está lejos de la residencia de los trabajadores.

Los pagos adicionales del Puesto de Trabajo (complemento objetivo) son percepciones que el trabajador recibe por razón de las características del Puesto de Trabajo. Son: complementos por periodicidad, toxicidad, peligrosidad, suciedad, máquinas, turnos, trabajo nocturno o cualquier otro que deba percibir el trabajador por razón de las características del Puesto de Trabajo y de la forma de realizar su actividad profesional que comporte conceptuación distinta del trabajo corriente.

En la antes mencionada NORMA CUBANA 3000:2007 se definen los Pagos Adicionales como: “Pagos por trabajar en determinadas condiciones, cargos, actividades, ramas o sectores debido a su importancia económica, o del servicio que se presta, o como reconocimiento a trabajadores por sus aportes excepcionales y significativos al desarrollo económico y social del país y otros factores”. (3)

1.2. Sistemas estudiados

En nuestro país varias empresas para la gestión de RH utilizan diversos software tanto de producción extranjera como desarrollados en Cuba. Algunos de estos sistemas sólo cuentan con la gestión del CH, otros brindar más servicios como los sistemas de planificación de recursos empresariales (ERP) o se vinculan con otros subsistemas.

Ejemplos de esto son:

- ASSETS NS
- SAGE MAS 500
- KEWAN
- VERSAT SARASOLA
- RODAS XXI

ASSETS NS

Es utilizado por diferentes sectores en el país entre los que están, el Ministerio de la Educación Superior, el Consejo de Estado, la Aduana General de la República, el Ministerio de Auditoría y Control, el de Justicia y el de Finanzas y Precios. Posee un módulo en el cual se desarrollan los procesos relacionados con la gestión de los Recursos.

Este sistema aunque logra la gestión de los trabajadores no permite mantener un registro actualizado de todo el personal con el que cuenta la entidad en un momento determinado, no es un sistema multiplataforma, y el lenguaje que se utiliza para su desarrollo, en este

caso Visual Basic, no cumple con la soberanía que se requiere para las entidades cubanas.

KEWAN

El sistema dirige más sus esfuerzos a las gestiones contables y sus activos, presentando algunas deficiencias en cuanto al modelo cubano para el diseño e implementación de un Sistema de Gestión Integrada de los Recursos Humanos, no posibilita tener el control de los diferentes movimientos realizados sobre un trabajador, así como llevar un registro actualizado de los trabajadores que posee una entidad.

SAGE MÁS 500

En SAGE MAS 500 los RH incluyen tres módulos que son Recursos Humanos Abra, Nómina Abra y Control de Asistencia Abra, a través del primero se provee la gestión de los RH, y a su vez incluye la gestión de los empleados. Mantiene una información detallada de los empleados a través de un historial.

SAGE MAS 500 gestiona de forma más integral los trabajadores dentro de su módulo de RH sin embargo no es una propuesta viable para la economía del país debido a que se incurre en costosos gastos en conceptos de licenciamientos y mantenimientos de software y las tecnologías que se utilizan para el desarrollo y sustento del sistema son propietarios.

SAP

Entre las funcionalidades que SAP ERP ofrece esta la gestión del CH esta funcionalidad la proporciona específicamente SAP ERP Human Capital Management.

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 como para adquirirlo y sus licencias tienen un alto valor monetario. Además de que es un software extranjero y desarrollado para plataforma de software propietario. (6)

VERSAT SARASOLA

Dicho sistema aunque gestiona los RH, no lo hace de forma organizada debido a que no cuenta con un módulo específico que gestione los mismos, su objetivo principal va dirigido hacia la elaboración de la nómina y recoge de los trabajadores los datos necesarios para la misma. No disponiendo de una plataforma de intercambio de información controlado, flexible y ágil, y no es un sistema multiplataforma.

RODAS XXI

El RODAS XXI es un sistema que gestiona los trabajadores de forma organizada sin embargo no cumple con requisitos establecidos en las Normas Cubanas (NC) para un Sistema de Gestión Integrada de Capital Humano, además de no ser multiplataforma pues sólo es compatible con el sistema operativo Windows 2000/XP.

Análisis del estudio realizado

Teniendo en cuenta la investigación con relación a varios de los sistemas analizados se llega a la conclusión de la necesidad de crear un nuevo sistema que permita la gestión de los trabajadores y que cumpla con las necesidades de las entidades a nivel nacional, tomándose la decisión de desarrollar un sistema de gestión de fuerza laboral que permita integrarse al subsistema CH del Sistema Integral de Gestión Cedrux, que cubra las necesidades de la sociedad cubana y respalde los lineamientos nacionales de desarrollo de soluciones bajo el principio de independencia tecnológica.

1.3. Patrones

1.3.1. Patrones de asignación de responsabilidades

Un patrón es una descripción de un problema y su solución (pareja problema/solución), con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas.

Los Patrones Generales de Software para Asignar Responsabilidades (GRASP) describe los principios de la asignación de responsabilidades de un objeto, ejemplos de estos son:

- Experto
- Creador
- Alta cohesión
- Bajo acoplamiento
- Controlador

Cada uno de estos tiene sus particularidades y funciones que lo diferencia, como el creador que asigna la tarea de crear el objeto a la clase que tiene más información de lo que se quiere realizar. (7)

1.3.2. Patrones de Diseño

Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

El grupo de GoF clasificaron los patrones en tres grandes categorías basadas en su

propósito: creacionales, estructurales y de comportamiento.

- **Creacionales:** Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- **Comportamiento:** Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

1.4. Modelo de desarrollo de Software del proyecto ERP-Cuba

- **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 significado arquitectónico 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.

Descripción del ciclo de vida

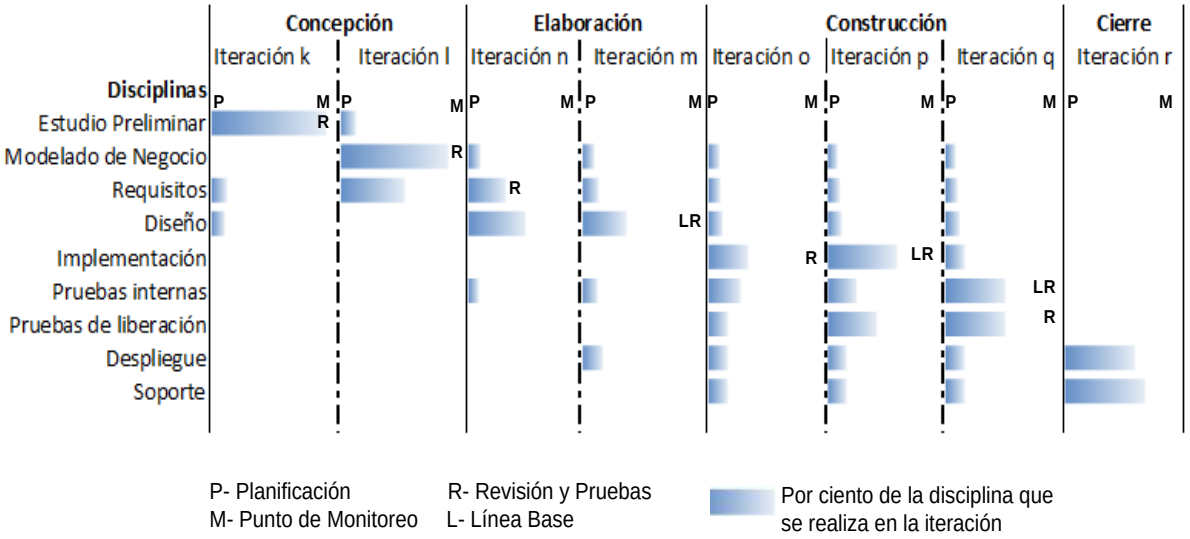

Figura 1. Descripción del ciclo de vida

La descripción de las fases y disciplinas se encuentra en el documento “Ciclo de Vida del Producto” que se encuentra en la documentación del proyecto ERP-Cuba.

Este epígrafe fue citado del documento “Ciclo de Vida del Producto” que se encuentra dentro de la documentación del proyecto.

1.5. Tecnologías

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

1.5.1. 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 permiten crear software. Estos facilitan las tareas de programación ya que poseen formas adecuadas para su entendimiento y resultan independientes de la computadora a utilizar.

PHP

Es un lenguaje script procesado en el lado del servidor que se muestra como código embebido dentro de una página HTML. Su modo de operaciones es el siguiente:

- El Navegador realiza una petición al servidor (se escribe la URL).
- Después el servidor ejecuta el código PHP solicitado y retorna el código HTML

generado al Navegador.

- Por último el Navegador muestra la respuesta del servidor.

Este tipo de iteración permite algunas operaciones complejas como conexiones a bases de datos o ejecución de complejos programas. PHP además de soportar un número masivo de bases de datos, incluyendo INFORMIX, ORACLE, Sybase, Solid y PostgreSQL, etc. También nos ofrece una gran variedad de funciones que nos permiten desarrollar múltiples funcionalidades que van desde enviar un e-mail, subir un archivo (upload), crear una imagen en tiempo de ejecución, interactuar con diversos protocolos de comunicación, interactuar con documentos XML, autenticación, creación dinámica de documentos PDF, entre muchas otras cosas.

Las principales características de PHP son: su rapidez; su facilidad de aprendizaje; su soporte multiplataforma tanto de diversos Sistemas Operativos, como servidores HTTP y de bases de datos; y el hecho de que se distribuye de forma gratuita bajo una licencia abierta. (8)

1.5.2. Tecnología 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. AJAX está conformado por:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- 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. (9)

A continuación se explican las características más importantes de algunas de las

tecnologías que componen AJAX:

XML

Es el estándar de Lenguaje de Etiquetado Extensible (Extensible Markup Language), 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. (10)

JSON

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

JSON está constituido por dos estructuras:

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

Estas son estructuras universales y virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras. (11)

XHTML

El Lenguaje Extensible de Marcado de Hipertexto, en inglés Extensible Hypertext Markup Language (XHTML) 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. (12)

CSS

Es un Lenguaje de Hojas de Estilos en Cascada, en inglésCascading 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. (13)

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. Ventajas de JavaScript:

- Los programas escritos en este lenguaje no requieren de mucha memoria ni tiempo adicional de transmisión, por ser pequeños y compactos.
- 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. (14)

1.5.3. 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. También podemos definirlo en el contexto de la

programación como un set de funciones o código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicaciones (creación de objetos, conexión a base de datos, limpieza del código, etc.). 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. (15)

ExtJS 2.0

Es una librería JavaScript de código abierto, ligera y de alto rendimiento, compatible con la mayoría de los navegadores que nos permiten crear páginas e interfaces web dinámicas. El mismo incluye tecnologías como Ajax, DHTML, XML, XSLT, JSON, CSS y DOM. Tiene incluidos la mayoría de los controles de los formularios Web incluyendo tablas para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. (16)

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. (17)

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.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada (18)

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 para la comunicación entre los módulos o componentes. Se le incorporó la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos y el ExtJs Framework para el desarrollo de las vistas.

Doctrine Framework

Es un potente y completo sistema ORM (ObjectRelationalMapper) 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. (19)

1.6. Herramientas

1.6.1. Visual Paradigm

Visual Paradigm es una herramienta que emplea Lenguaje Unificado de Modelado (UML) que 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 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 y modelado visual que permite una abstracción del sistema y sus componentes.

Objetivos del UML:

- 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. (20)

1.6.2. Ambiente de Desarrollo Integrado

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 en inglés Integrated Development Environment (IDE), 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. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. (21)

Zend Studio para Eclipse

El Zend Studio para Eclipse proporciona un entorno 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 características que incluye este IDE están:

- El acceso al paquete de plugins de Eclipse.
- Apoyar el desarrollo de múltiples idiomas.
- Editor de PHP con el formato avanzado, para listas de tareas y problemas de vista.
- Soporte de JavaScript.
- Apoyo, incluyendo HTML, códigos plegables, arrastrar y soltar los componentes.
- Depuración y perfiles con Path Mapping.
- Mejora de Zend Framework con el apoyo del Proyecto Framework, plantillas y código MVC. (22)

1.6.3. Servidor Web

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el

protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web básico cuenta con un esquema de funcionamiento muy simple, basado en ejecutar infinitamente el siguiente bucle:

- Espera peticiones en el puerto TCP indicado.
- Recibe una petición.
- Busca el recurso.
- Envía el recurso utilizando la misma conexión por la que recibió petición.
- Vuelve al segundo punto.

Un servidor web que siga el esquema anterior cumplirá todos los requisitos básicos de los servidores HTTP, aunque sólo podrá servir ficheros estáticos.

A partir del anterior esquema se han diseñado y desarrollado todos los servidores de HTTP que existen, variando sólo el tipo de peticiones (páginas estáticas, CGIs, Servlets, etc.) que pueden atender, en función de que sean o no sean multi-proceso o multi-hilados, etc. A continuación se detallan algunas de las características básicas de los servidores web. (23)

Apache

Corre en una multitud de Sistemas Operativos y es una tecnología gratuita de código fuente abierta. 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. (24)

1.6.4. Sistema Gestor de Base de 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). (25)

1.6.5. Herramientas de base 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 PosgreSQL ofrece muchas de las poderosas herramientas de base de satos, como bases de datos visuales Designer, Visual QueryBuilder para crear consultas de PosgreSQL complicadas. (26)

1.6.6. Navegadores

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. (27) Contiene un Plugin firebug que se utiliza para ver los errores del código.

1.6.7. Control de versiones

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. (28)

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. (28)

Conclusiones parciales del capítulo

En la elaboración de este capítulo se abordaron varios conceptos generales que servirán de guía para la realización de este trabajo, se realizó una descripción de cómo se gestionan los procesos referentes a la gestión de los trabajadores así como un estudio y análisis sobre algunos sistemas ya existentes. Se determinaron además las tecnologías, lenguajes, herramientas y metodología a utilizar para el desarrollo de la solución propuesta.

Capítulo 2: Diseño e Implementación.

Introducción

En este capítulo se abordarán los temas referentes al diseño e implementación, se modelarán los artefactos que permitirán dar continuación a la implementación del sistema como los diagramas de clases del diseño, a los cuales se les aplican patrones de diseño para hacer el trabajo más eficiente y el modelo de datos representando las tablas de la base de datos vinculadas a la propuesta de solución. En el mismo se muestra el modelo de implementación, donde se representan las clases de diseño en términos de componentes, se muestra además el diagrama de componentes.

2.1. Análisis de los artefactos de entrada

Las especificaciones de los requisitos fueron elaboradas por los analistas de la línea CH. Para el desarrollo de este trabajo se realizó la revisión y estudio de las funcionalidades de los componentes antes de realizar el diseño de las mismas. En esta actividad se comprobó que los requisitos estaban correctamente redactados, eran claros, posibles de probar y cubrían todas las funcionalidades necesarias en los componentes. A partir del resultado obtenido del análisis de los artefactos de entrada se dio comienzo al diseño e implementación de la solución.

Los componentes desarrollados engloban las funcionalidades definidas mediante ochenta requisitos funcionales, divididos en diez del componente Pagos Adicionales, trece de Régimen de trabajo, treinta y nueve de Trabajador y dieciocho de puesto de Trabajo. Estos requisitos guiaron la implementación de los componentes, asegurando que estos contaran con todas las funcionalidades requeridas.

2.2. Modelo de Diseño

En el diseño se muestran las vistas más significativas para la arquitectura, compuestas por subsistemas de diseño encargados de definir y brindar las interfaces. En el modelo de diseño se muestran 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.

2.2.1. Diagrama de componentes

A continuación se presenta el diagrama de componentes modelado por el Arquitecto de sistema, donde se representan la interrelación entre cada uno de los componentes del subsistema CH así como con otros subsistemas del sistema Cedrux. Se resaltan en otro color los componentes a los que se refiere el documento en cuestión.

El componente pagos adicionales presenta relaciones con puesto de trabajo, movimiento de fuerza de trabajo y nómina porque los puestos y los trabajadores tienen pagos adicionales asignados para el pago de su salario que son calculados en el componte de nómina. El componente movimiento de fuerza de trabajo presenta relaciones con persona para obtener los datos personales, submayores para tener un control de las vacaciones del trabajador, incidencia para gestionar las indisciplinas y llegadas tardes del trabajador, régimen de trabajo expone el tiempo de trabajo para cuando se calcule la nómina. Puesto de trabajo presenta relaciones con otras líneas como estructura y composición para obtener la entidad, el cargo y las áreas del puesto de trabajo, movimiento de fuerza de trabajo actualiza el estado del puesto (ocupado o vacante).

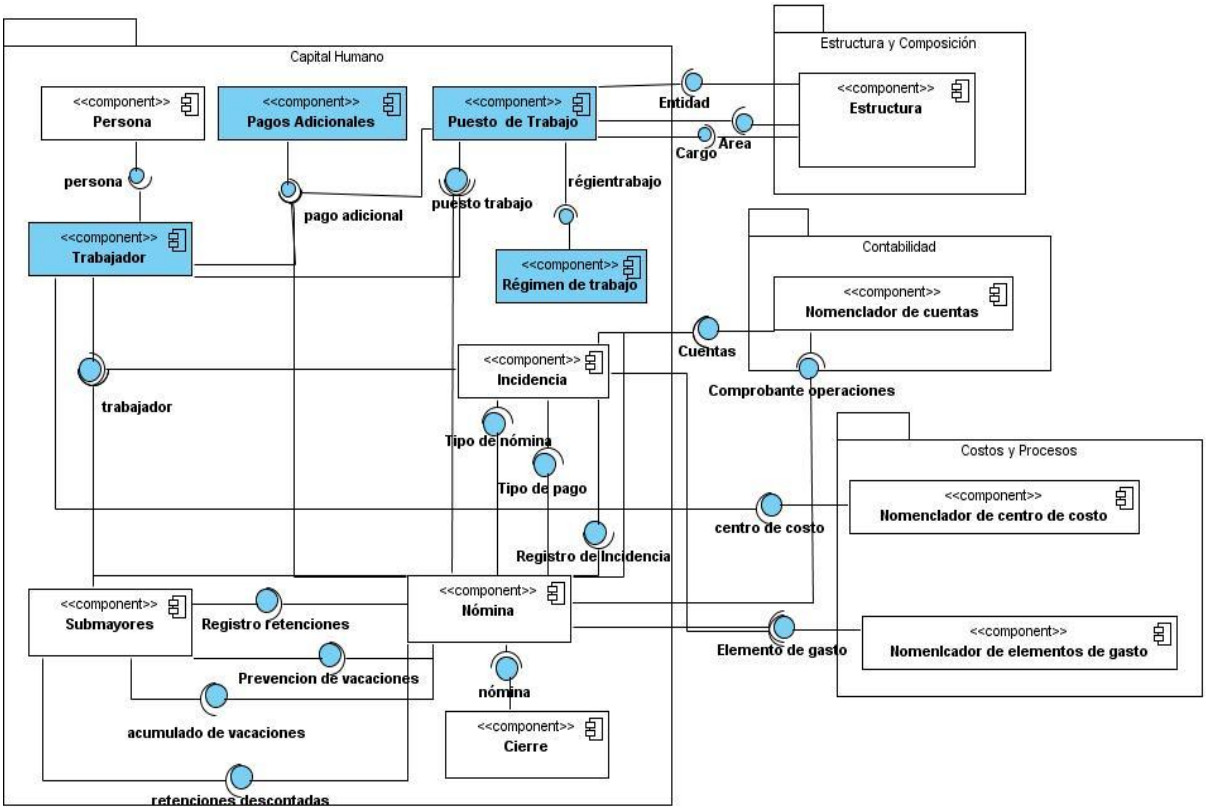

Figura 2: Diagrama de componentes

2.2.2. Diagramas de clases

A continuación se muestran los diagramas de clases del diseño referentes a los componentes Pagos adicionales, Puesto de Trabajo y Movimiento de Fuerza de Trabajo del subsistema CH del sistema Cedrux.

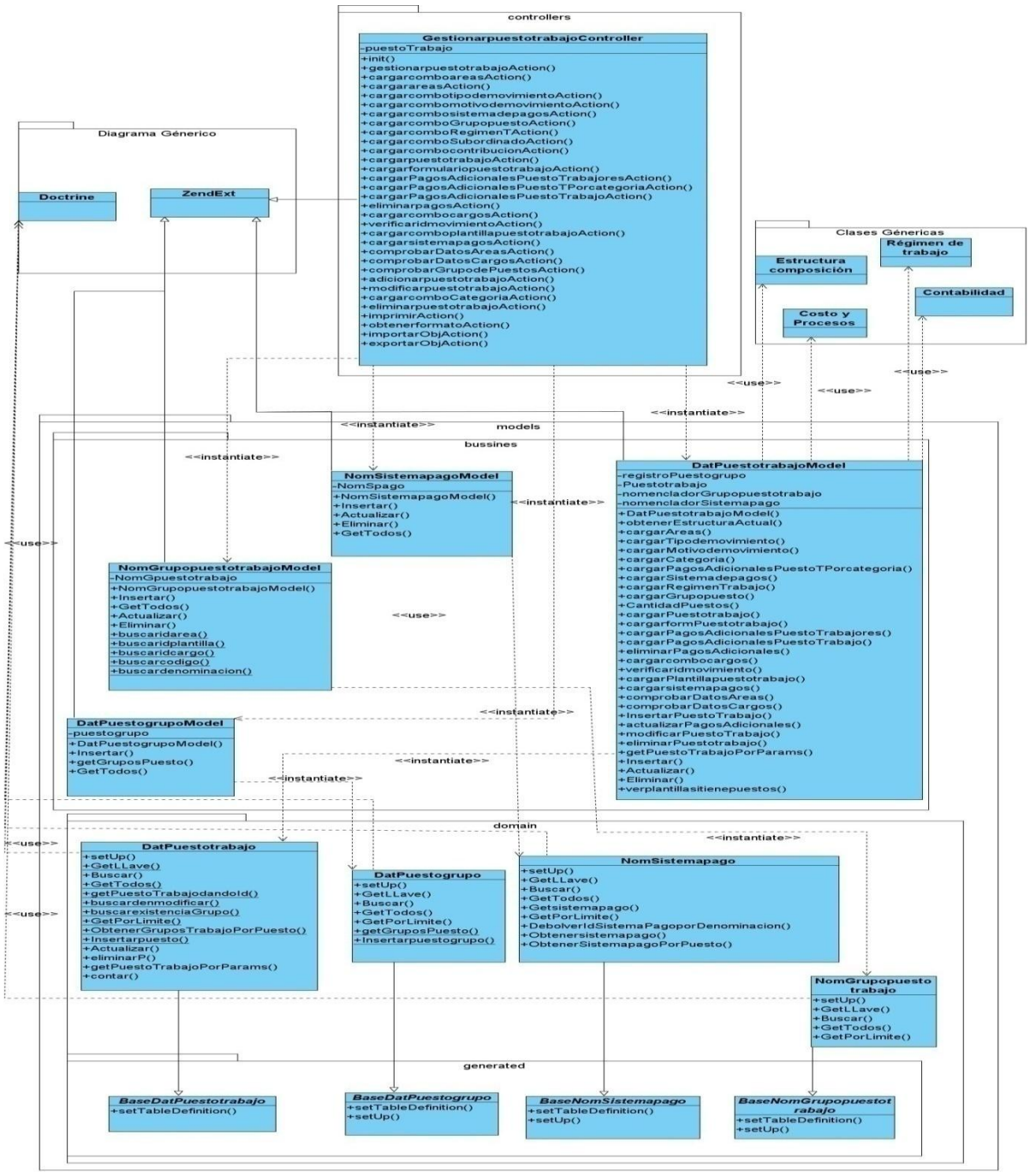

Figura 3: Diagrama de clases Puesto de Trabajo ver descripción

En la figura se muestra el diagrama de clases del componente puesto de Trabajo que se encarga gestionar todos los datos de un puesto de trabajo y consta de diecinueve clases que están estructuradas de la siguiente forma: una clase controladora que captura la información de la vista y la envía a las cuatro clases del modelo que realizan toda la lógica del componente, cuatro clases del dominio que realizan todas las consultas a la base de datos, cuatro clases bases que permiten la conexión con las tablas correspondientes, dos clases que heredan de los frameworks y las clases con que se relaciona con otros componentes.

El diagrama de clases del componente Pagos adicionales que maneja todos los referente a los pagos consta de veintitrés clases que están estructuradas de la siguiente forma: una clase controladora, seis clases del modelo, cinco clases del dominio, cinco clases bases, dos clases que heredan de los frameworks y las clases con que se relaciona con otros componentes como se muestra en la figura a continuación.

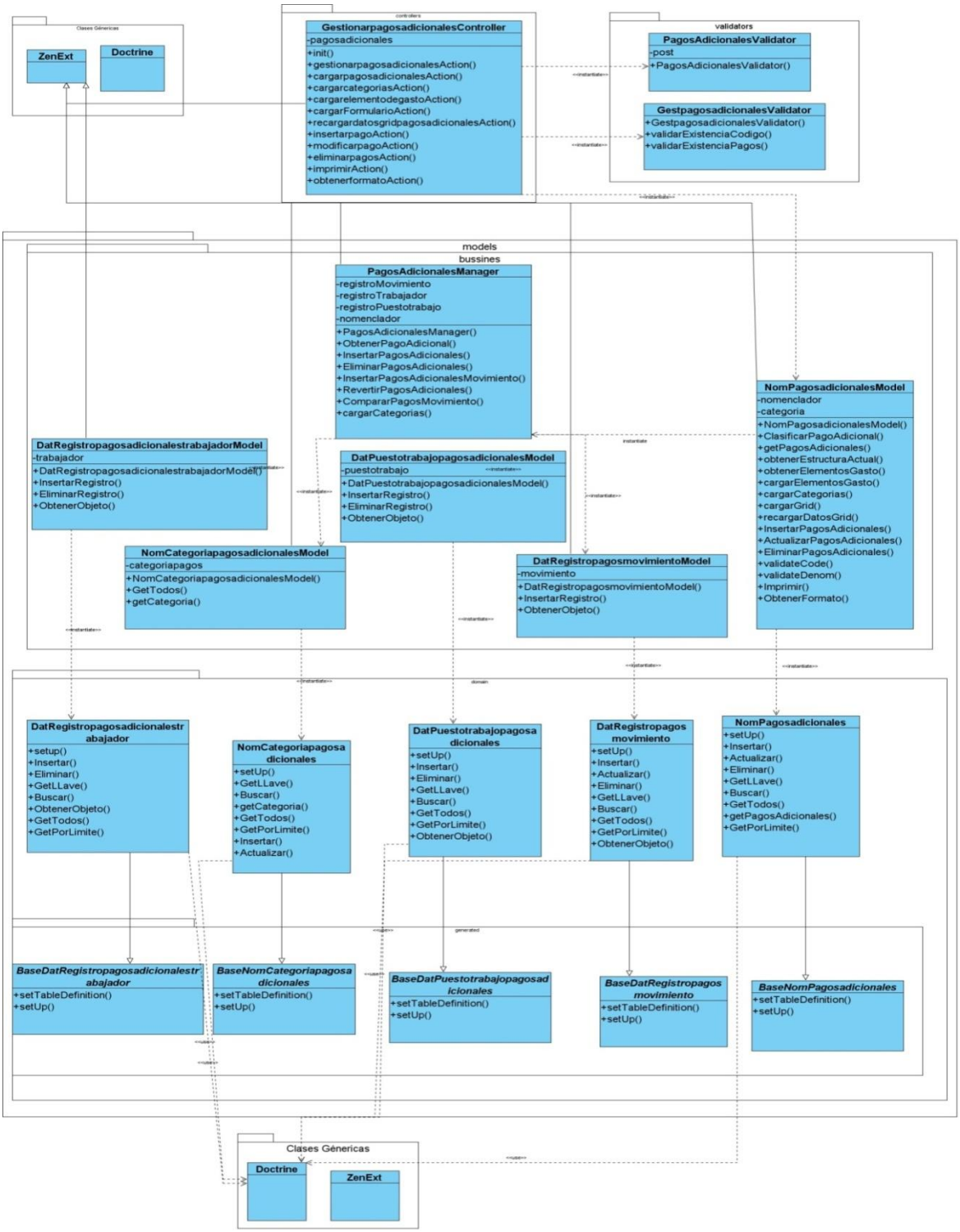

Figura 4: Diagrama de Clase Pagos Adicionales [ver descripción]

El diagrama de clases del componente Trabajador que controla todo lo referente a los trabajadores consta de cincuenta y cinco clases que contiene la siguiente estructura: cinco clases controladoras, trece clases del modelo, cinco clases del dominio, doce clases bases, dos clases que heredan de los frameworks y las clases con que se relaciona con otros componentes como se muestra en la figura a continuación.

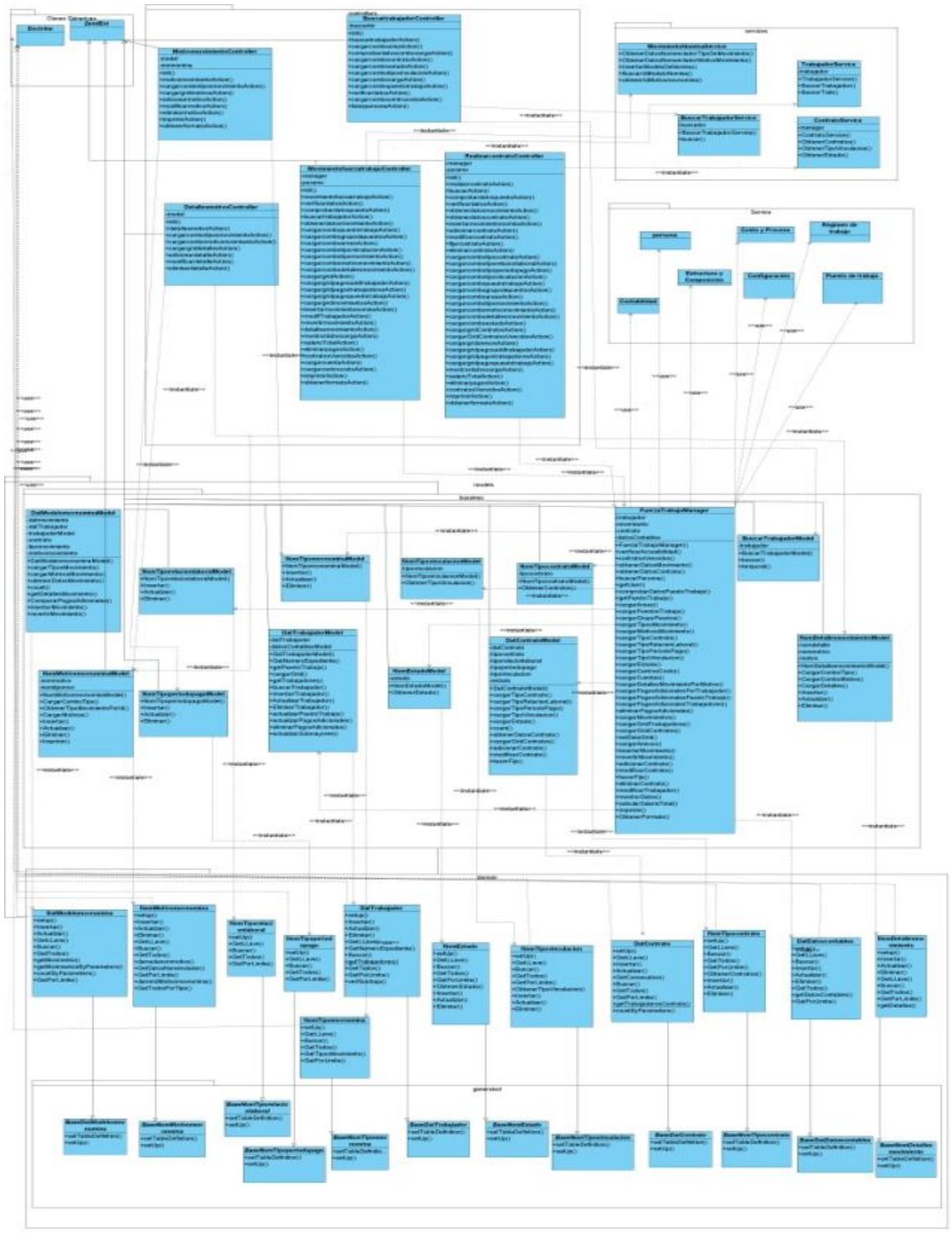

Figura 5: Diagrama de Clases Trabajador ver descripción

El diagrama de clases del componente Régimen de Trabajo que gestiona los datos de los régimen de trabajo consta de dieciséis clases que contiene la siguiente estructura: una clase controladora, cuatro clases del modelo, cuatro clases del dominio, cuatro clases bases, dos clases que heredan de los frameworks y la clase con que se relaciona con otros componentes como se muestra en la figura a continuación.

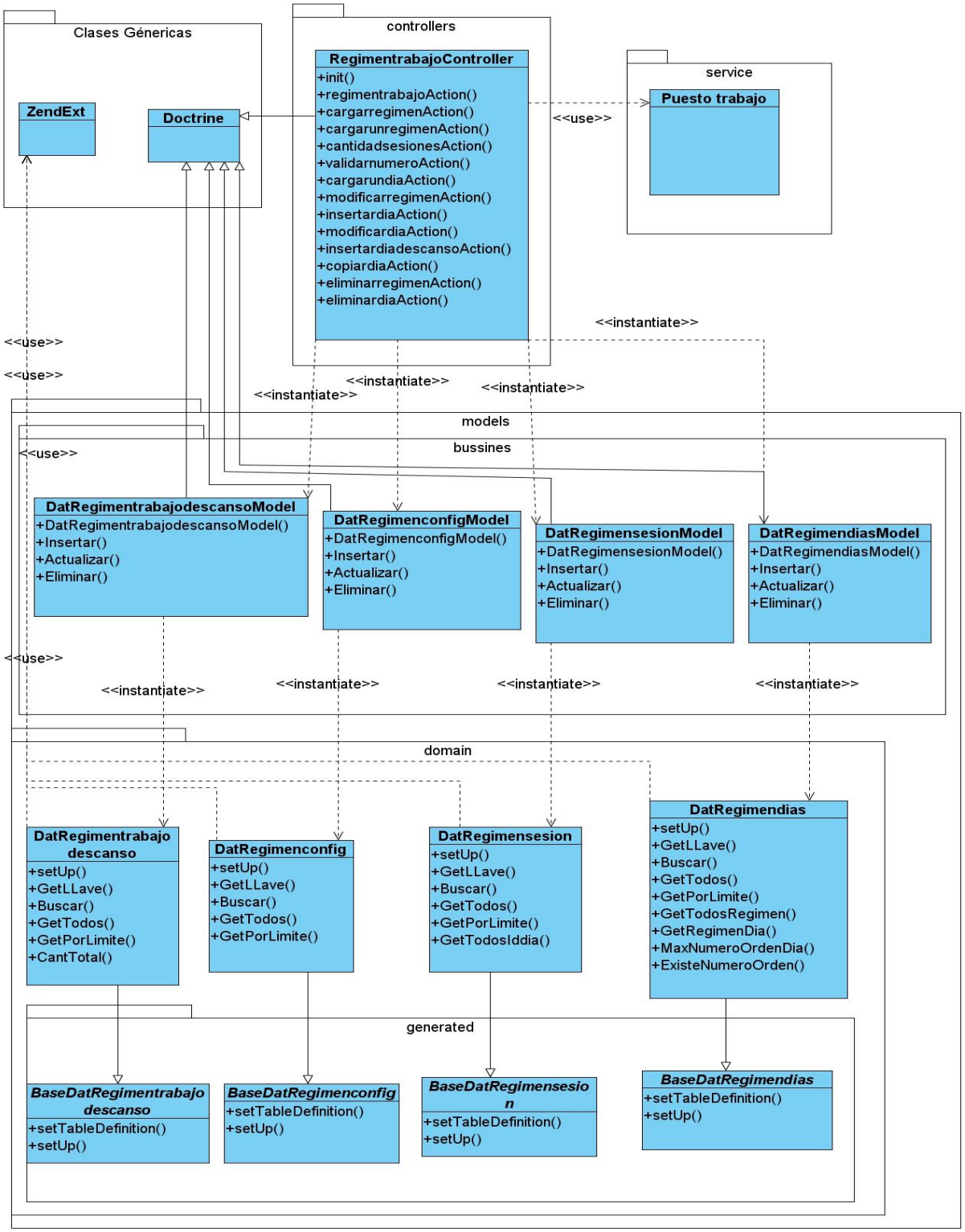

Figura 6: Diagrama de Clases Régimen de Trabajo.

2.2.3. Patrones de diseño

El uso de patrones da la posibilidad de organizar las clases en estructuras comunes y bien probadas; modificando el sistema para mejorar su flexibilidad y extensibilidad. En otras palabras, incrementando la facilidad para adaptar el software a los cambios de especificación e incrementando su posible reutilización. De alguna forma, los mismos beneficios producidos por la programación orientada a objetos pero mejorando aún más la calidad del diseño, y por lo tanto el software en sí. Emplear patrones de diseño ayuda a tener un lenguaje común con otros programadores. Para que los componentes desarrollados cuenten con estas características se utilizaron una serie de patrones que se explican a continuación.

Patrones Grasp (Patrones Generales de Asignación de Responsabilidades)

Los patrones Grasp describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Booch y Rumbaugh definen la responsabilidad como "un contrato u obligación de un tipo o clase".

De los patrones de asignación de responsabilidades se utilizaron Bajo Acoplamiento y Alta Cohesión con el fin de que contribuyan a que el sistema sea más robusto y flexible.

Bajo Acoplamiento

Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades. El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

Consiste en tener las clases lo menos ligadas entre sí, permitiendo que al producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Mantener el bajo acoplamiento entre clases más que un patrón es una buena práctica del diseño dado que al realizar cambios no se afectan otros componentes, es más fácil de entender de manera aislada y de esta forma se pueden reutilizar mejor las clases. Este patrón se aplica en todas las clases definidas en cada uno los componentes desarrollados.

Alta Cohesión

Este patrón dice que la información que almacena una clase debe ser coherente y estar en la mayor medida de lo posible relacionada con la clase.

Realizando un diseño donde las clases mantengan una alta cohesión se mejora la claridad y facilidad con que se entiende el diseño, se simplifica el mantenimiento y las mejoras de

funcionalidad, a menudo se genera un bajo acoplamiento, soportando mayor capacidad de reutilización.

Los patrones Bajo Acoplamiento y Alta Cohesión generalmente trabajan muy unidos. Un buen diseño requiere la correcta utilización de ambos patrones. En cada una de las clases del diseño de modelado en la solución se manifiestan ambos patrones, cada una de ellas implementando las responsabilidades específicas que le fueron asignadas, y delegando las demás.

Patrón Experto

El patrón fue usado con el objetivo de darle a las clases las responsabilidades necesarias siempre que contaran con la información para cumplirlas. Logrando así un mejor comportamiento entre las clases y haciendo que éstas fueran más cohesivas, fáciles de comprender y mantener, permitiendo mayores facilidades de soporte.

Se evidencia en las clases modelos las cuales tienen la responsabilidad de gestionar un modelo en específico y cuentan con todos los permisos para ello.

Patrón Creador

Permite a un cliente crear un objeto especificando tipo y contenido, ocultándose el resto de los detalles. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Este soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de“ clase sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Así se brinda soporte a una alta cohesión.

Este patrón permite tener una política general para la creación de objetos. Se usa en las clases controladoras para crear instancias de las clases del modelo, principalmente las del negocio, así como en las clases del negocio para instanciar las clases del dominio.

Patrón Controlador

Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a clases que representen: un sistema global o la fachada, algo en el mundo real que pueda ejecutar el papel (personas), o un manejador artificial que pueda ejecutar los eventos. El patrón se pone de manifiesto en todas las clases controladoras que se implementaron.

Patrones GoF (Gang of Four o Pandilla de los Cuatro)

Los patrones de diseño proponen una forma de reutilizar la experiencia de los desarrolladores, para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Se clasifican de acuerdo a su propósito en: creacionales, estructurales y de comportamiento.

Los patrones creacionales se encargan de la creación de los objetos ayudando a que el sistema sea independiente de la creación, composición y representación de los mismos.

Mientras que los patrones estructurales son los encargados de cómo las clases y objetos están compuestos para formar estructuras más grandes. Los patrones estructurales usan la herencia para componer interfaces u objetos en tiempo de ejecución. Los patrones de comportamiento plantean algoritmos y la asignación de responsabilidades entre objetos. Estos patrones no sólo describen clases y objetos sino también describen la comunicación entre ellos. Se seleccionaron los siguientes patrones Gof en la modelación del diseño.

Patrones de comportamiento

Cadena de responsabilidad

La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición. Aplicar la cadena de responsabilidad trae beneficios como la reducción de acoplamiento, mayor flexibilidad en la asignación de responsabilidades a los objetos con el inconveniente de que como la recepción no está garantizada pueden existir excepciones que nunca se manejen si la cadena está mal configurada. Este patrón es utilizado en la mayoría de los diagramas de clases siendo aplicado en el tratamiento de excepciones. Un ejemplo de esto es cuando se insertar un puesto de trabajo y el código ya se encuentra en la base de datos, el Doctrine genera una excepción la cual es capturada por la clase DatPuestotrabajoModel, modificada y envidada al controlador GestionarpuestotrabajoController y la hace llegar a la vista donde se elabora el mensaje a mostrar al usuario.

2.3. Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros. (29)

2.3.1. Estándares de codificación.

A continuación se muestra el estilo de código utilizado en la implementación de los

componentes antes mencionados, el cual sigue los estándares definidos por el marco de trabajo.

El nombre de las clases se escribe la primera letra con mayúsculas y las demás con minúsculas. En caso de ser un nombre compuesto es seguido del primer nombre y tiene la misma estructura que el primer nombre. Ejemplo: Reporte, ReporteTrabajador.

Nomenclatura de las clases según el tipo.

- Las Controladoras se encuentran dentro de la carpeta (paquete) “controller”: Las clases controladoras después de escribir el nombre según su nomenclatura se le añade “Controller”. Ejemplo: TrabajadorController.
- Las clases 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: TrabajadorModel.
- 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: DatTrabajador.
- 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: BaseDatTrabajador.

Nomenclatura de los métodos o funciones.

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

Nomenclatura de las variables

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

2.3.2. 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 29 tablas que representan de manera general el negocio de los componentes Trabajador, Pagos adicionales y Puesto de trabajo del subsistema CH del sistema Cedrux.

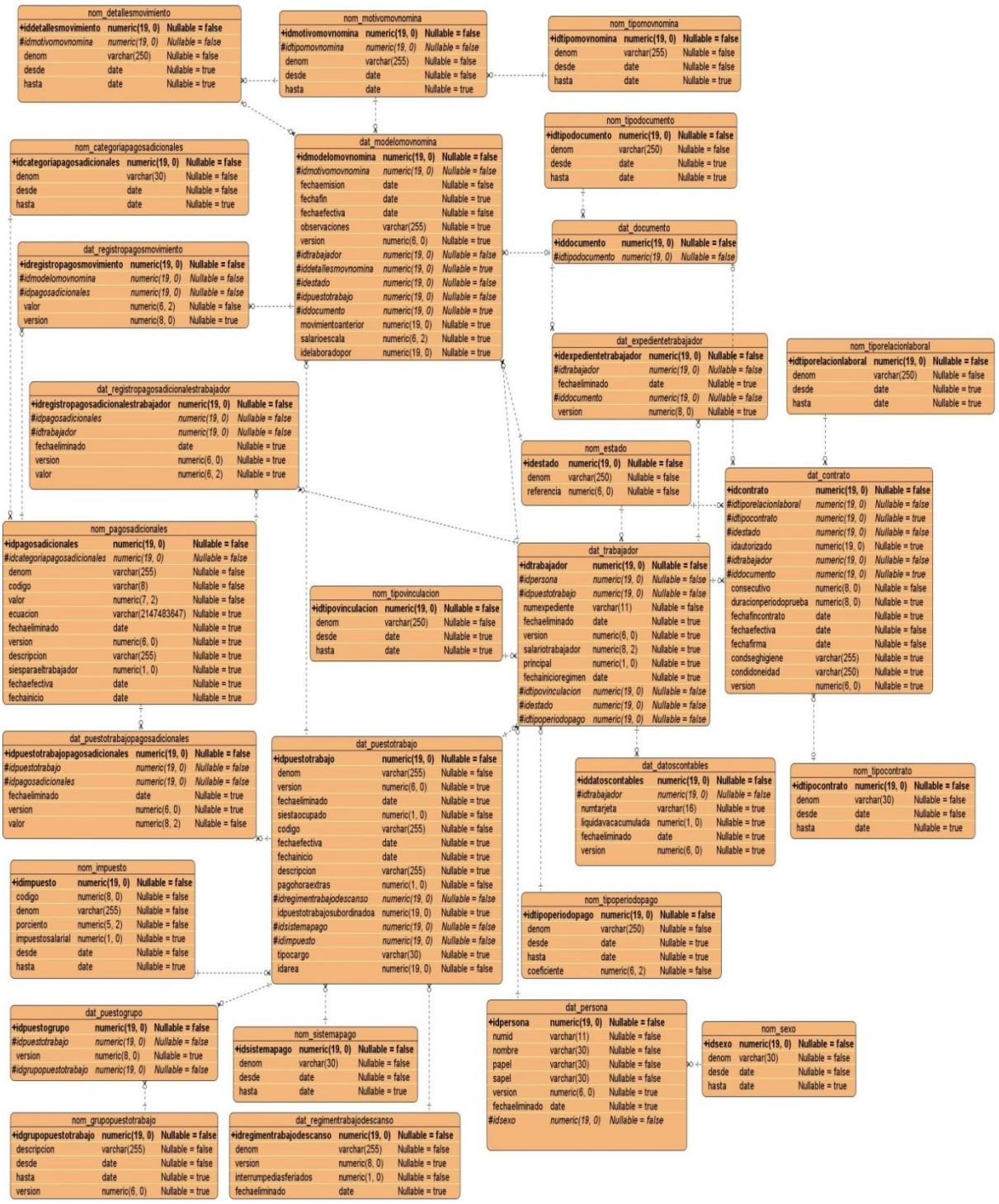

Figura 7: Modelos de datos

2.3.3. Implementación de los componentes

La implementación de aplicaciones por componentes se basa en la reutilización de código elaborado con anterioridad, este código en su momento fue probado, y su funcionalidad fue comprobada. Mediante el uso de varios componentes simples se pueden construir proyectos bastante complejos los cuales si se realizaran desde el principio tomarían demasiado tiempo, de este modo lo que se tiene que hacer es revisar proyectos anteriores, luego de esto se puede pasar a juntar las piezas que se van a reutilizar y las demás piezas que se deben obligatoriamente especificar para cada proyecto, ya que no todos los proyectos van a tener la misma oportunidad de reutilización.

En la capa de acceso a datos se implementó la menor cantidad de consultas posibles, generalmente una para recuperar datos y otra para contar la cantidad de resultados encontrados, parametrizando las consultas, permitiendo 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. En la vista al cargar las interfaces principales las peticiones al servidor se realizan una vez que se obtiene respuesta de la anterior, 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. Este modo de implementación se realiza para que las aplicaciones sean más ligeras, rápidas y bien estructuradas.

2.3.4. Descripción de las funcionalidades

A continuación se presenta el diagrama de componentes implementados donde se representan la interrelación entre cada uno de los componentes para una mayor comprensión durante la descripción de las funcionalidades.

Figura 8: Diagrama de componentes.

Un puesto de trabajo es un lugar físico en unidad básica de una entidad donde el trabajador realiza su función durante un tiempo determinado. Entre las funcionalidades que brinda el componente implementado es el gestionar puestos de trabajo agrupa varios requisitos como son adicionar un nuevo puesto de trabajo, modificar o eliminar uno existente e imprimir. Otros requisitos que apoyan el trabajo con los puestos de trabajo son listar puestos por grupo de puesto de trabajo, agrupar puestos de trabajo dentro de un grupo y desagrupar. Para adicionar un nuevo puesto de trabajo el sistema muestra una interfaz (Figura 8), en la cual se debe introducir el código del puesto de trabajo, la denominación, la descripción, se debe marcar si recibe pagos de horas extras, seleccionar el área de trabajo y el cargo al que pertenece, el grupo de puestos de trabajo en el cual agrupará este dato no es obligatorio, el puesto al cual se subordina, los pagos adicionales que recibirá el trabajador, el sistema de pago en caso que aporte un porciento de sus salario a la seguridad social, y por último el régimen de trabajo al cual se acogerá el puesto de trabajo.

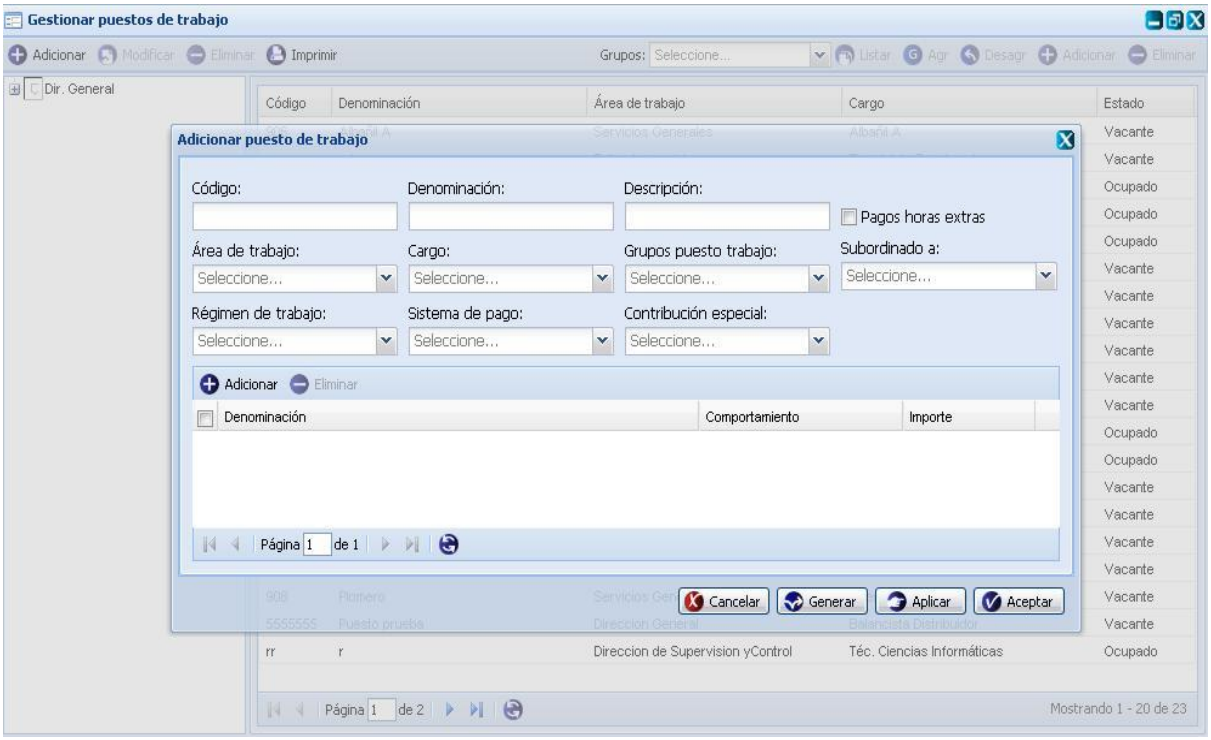

Figura 9. Interfaz para adicionar un puesto de trabajo.

Un régimen de trabajo y descanso define los días en que debe laborar el trabajador que se acoja al mismo. El régimen también define de cada día de trabajo las sesiones y la hora de inicio y fin de cada sesión.Desde la interfaz de gestionar régimen de trabajo y descanso se puede acceder a las funcionalidades adicionar un nuevo régimen de trabajo y descanso, modificar un régimen ya existente o eliminarlo. Esta interfaz se muestra en la figura 10.

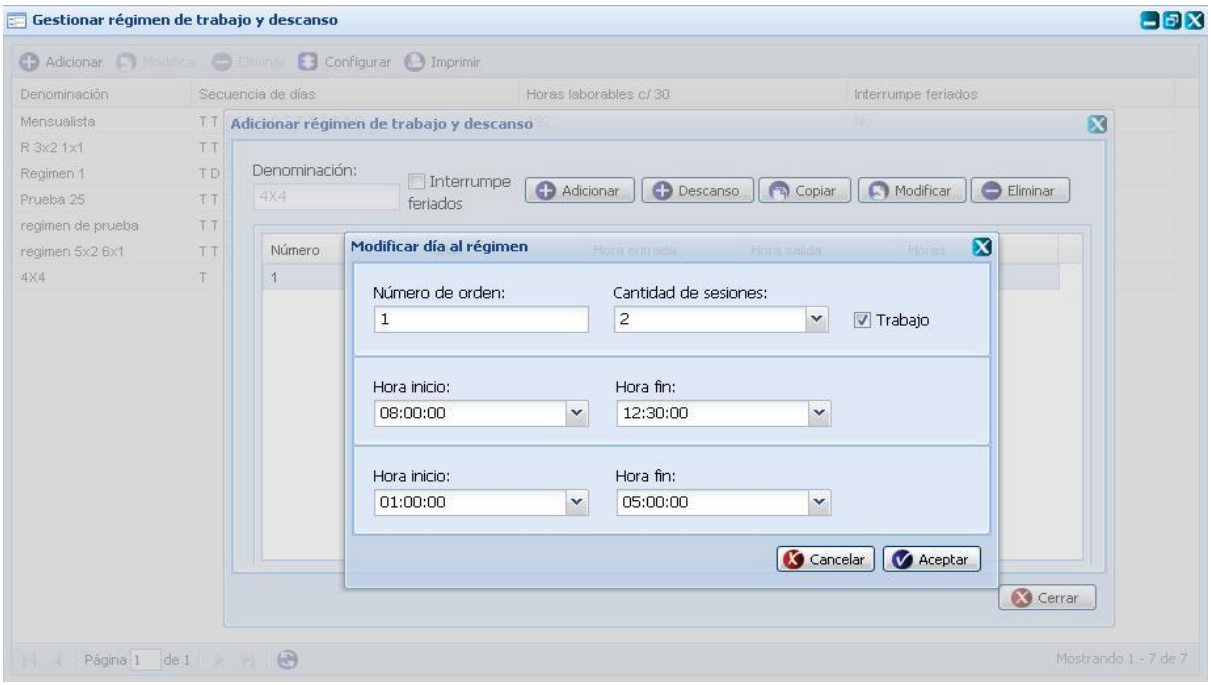

Figura 10. Interfaz gestionar régimen de trabajo y descanso.

Para adicionar el nuevo régimen de trabajo y descanso se debe introducir la denominación, si es interrumpido por los días feriados y por último la secuencia de días del régimen. La secuencia de días se le ha llamado a la sucesión de días que tendrá el régimen. Utiliza la secuencia de días para calcular los días que se deben trabajar para cada régimen. Para definir esta secuencia de días se utilizan las funcionalidades adicionar días, adicionar día de descanso, copiar un día dado, modificar o eliminar un día. Adicionar un día de descanso es una funcionalidad que adiciona de forma rápida sin mostrar interfaz un día de descanso al final de la secuencia de días. Copiar día es una funcionalidad que permite copiar un día seleccionado agregándolo al final de la secuencia. Modificar día de la secuencia permite modificar los datos de un día mostrando una interfaz similar a la del adicionar. Eliminar día de la secuencia permite eliminar de la secuencia de días el día seleccionado.

El componente trabajador permite formalizar la relación laboral de los trabajadores así como y notificar a las personas implicadas los diferentes tipos de movimiento de los trabajadores que sufran modificaciones en su estructura salarial, cargo o área de trabajo, constituyendo el documento que respalda las anotaciones para mantener actualizados los datos para el pago de las Nóminas. Se obtiene como resultado la actualización del registro de trabajadores y los movimientos de nómina, permitiendo esto realizar los análisis de Movimiento de Fuerza de trabajo. Para el uso de las funcionalidades del componente se deben haber definido los pagos adicionales, los motivos de movimiento de nómina y los detalles de los motivos de movimientos de nómina. Estos son nomencladores en los que se registran las diferentes causas por las que a un trabajador se le puede realizar un movimiento de nómina y los detalles de la misma.

La formalización de la relación laboral es el proceso mediante el cual se hace legal la relación del trabajador con la entidad, ya sea por contrato o por designación entre otros. Este proceso es la fuente de entrada de los trabajadores a la entidad. Desde la interfaz de Formalizar relación laboral el componente posibilita adicionar un nuevo documento de formalización (contrato), modificar un documento, eliminarlo (siempre que no se haya confirmado), confirmar el documento, ver los contratos vencidos, hacer fijo un contrato determinado o a prueba, prorrogar un contrato determinado o a prueba, ver los anexos que se le han hecho a una contrato determinado. Esta interfaz se muestra en la figura 11.

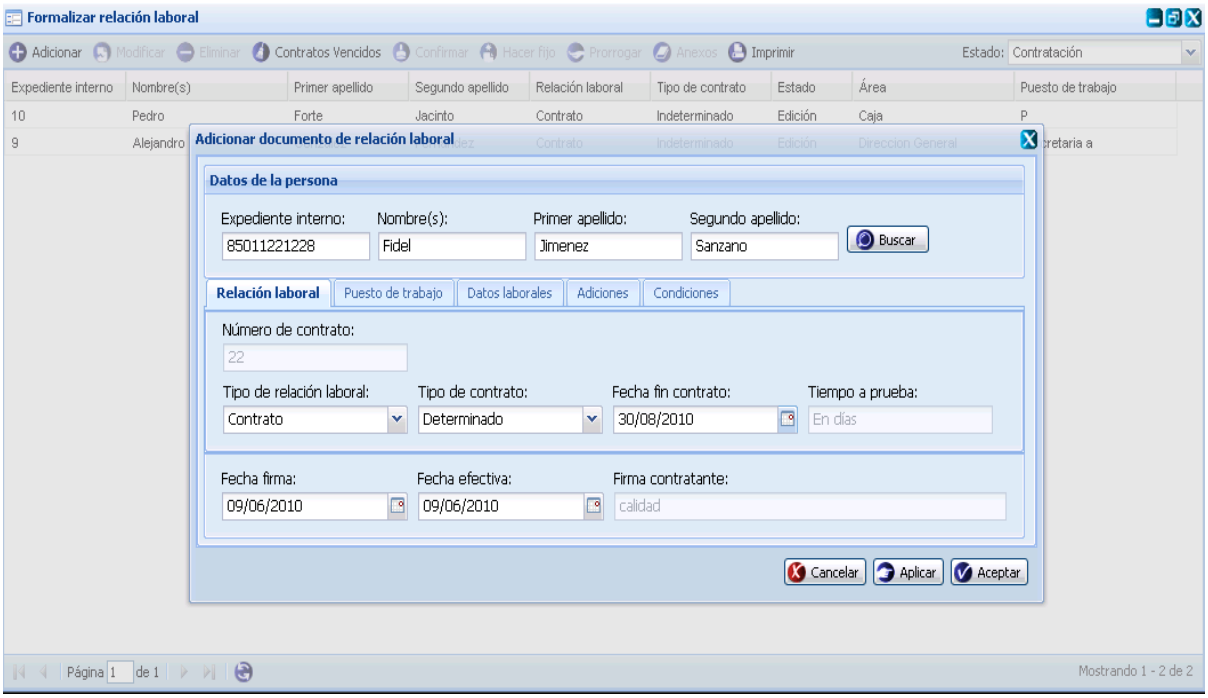

Figura 11. Interfaz Formalización Laboral.

Al acceder a la funcionalidad de movimiento de fuerza de trabajo se muestra una interfaz donde se listan los trabajadores de la entidad ordenados por el número de expediente interno y se muestra además el nombre, primer apellido, segundo apellido, área, cargo puesto de trabajo, tipo del movimiento de nómina por el cual se le paga, y si está contabilizado o no.

Figura 12. Interfaz movimiento de fuerza de trabajo.

Esta interfaz permite realizar una búsqueda de trabajadores, modificar sus datos, dar de baja a un trabajador dado, ver los detalles del movimiento de nómina por el cual se le paga al trabajador, actualizar los datos del trabajador en el movimiento de nómina, revertir un movimiento de nómina, listar todos los movimientos de nómina que se le han realizado a un trabajador, imprimir el listado de trabajadores, y filtrar trabajadores por área. Brinda una búsqueda Avanzada por varios criterios como son el número de expediente interno, nombre, apellidos, los centros de costos, número de tarjetas, Áreas, Cargos, Tipo de Vinculación. Esta interfaz se muestra en la figura 13.

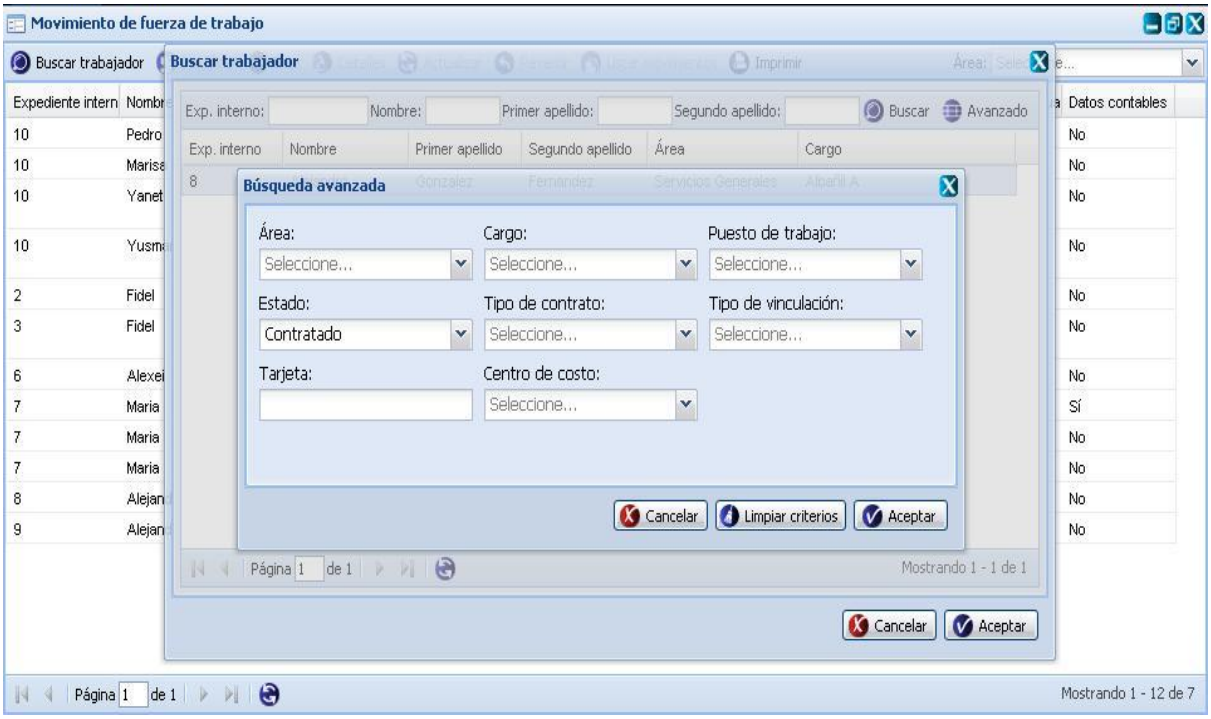

Figura 13. Interfaz Buscar Trabajador.

Además en la pestaña de datos laborales se permite modificar el puesto de trabajo, para esto se selecciona el área y después el puesto de trabajo. Se muestra información relacionada con el puesto de trabajo como es el cargo, categoría ocupacional, grupo escala, sistema de pago, contribución especial en caso de que tenga, salario escala, salario total, régimen de trabajo descanso, jornada laboral, horario de trabajo. Otros datos que se pueden modificar son el tipo de vinculación del trabajador, si este es su puesto de trabajo principal y la fecha de inicio del régimen laboral para este trabajador.

En los datos contables se recogen los datos necesarios para realizar la contabilización de los gastos de los trabajadores, se selecciona el centro de costo al que pertenece, y la cuenta del trabajador. Según la forma de pago del trabajador se deberá llenar los siguientes datos: si el trabajador tiene tarjeta entonces se introduce su número de tarjeta, en caso de que el

trabajador cobre por caja se recoge el centro de pago al que pertenece. El componente permite realizar el movimiento de nómina correspondiente a las modificaciones realizadas. En cualquier momento se puede mostrar el movimiento de nómina por el cual se le está pagando al trabajador y actualizar los cambios realizados y que por algún motivo no se actualizaron en el momento. Si se realizan cambios en los datos del trabajador y no se emite un movimiento de nómina con estos, los cambios no serán visibles en el procesamiento de la nómina. El componente permite listar todos los movimientos de nómina, revertir un movimiento de nómina, restaurando todos los datos del movimiento anterior y facilitando así la corrección de errores que hayan sido cometidos al emitir el movimiento.

En el componente PA se definen los pagos adicionales que se utilizarán en la entidad. Este nomenclador permite adicionar un nuevo pago adicional, modificar o eliminar uno ya existente, listar e imprimir los pagos adicionales.

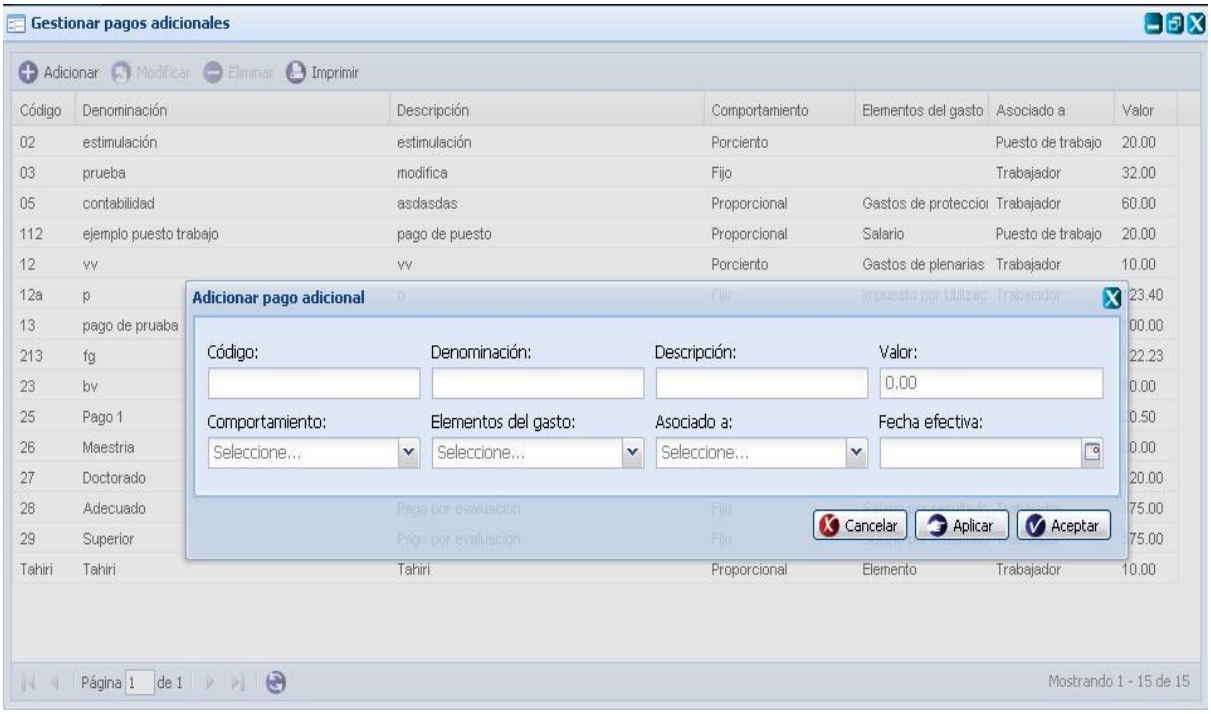

Figura 14. Interfaz gestionar pagos adicionales.

Al adicionar un nuevo pago adicional en la interfaz se debe introducir el código del pago que debe ser un número único, denominación que es el nombre o abreviatura que recibe el pago, descripción que es una explicación más detallada, el comportamiento que es la forma en que se comporta el pago a la hora de hallar el importe, se debe seleccionar el elemento de gasto del pago, se introduce el valor del pago que puede ser el importe del pago o el porciento. En dependencia del comportamiento, se debe seleccionar si estará asociado directamente al trabajador o al puesto de trabajo, y por último la fecha efectiva.

2.3.5. Descripción de Algoritmo no trivial.

El algoritmo insertarMovimiento() correspondiente a la clase DatModelomovnominaModel permite insertar un movimiento de nómina. Se valida que no exista ningún movimiento pendiente. Se verifica que se hayan efectuado cambios salariales al trabajador o al puesto de trabajo, de lo contrario se lanza una excepción que no se puede realizar el movimiento de nómina. Se valida que no exista el número de movimiento que se pretende adicionar. Se obtiene el último movimiento del trabajador y se asocia como movimiento anterior al que se debe insertar. Se inserta un nuevo documento en el componente Configuración debido a que cada movimiento de nómina es un documento. Si la fecha efectiva del movimiento es posterior a la fecha actual se adiciona como movimiento en pendiente de ejecutarse, de lo contrario se adiciona como movimiento Activo. Se actualiza el movimiento anterior cambiando el estado de Activo a Inactivo. Se insertan los pagos adicionales del trabajador y del puesto de trabajo al registro de pagos adicionales del movimiento de nómina. Si se cambia el puesto de trabajo ocupado por el trabajador se actualiza en el componente Puesto de Trabajo, cambiando la disponibilidad del puesto anterior (como vacante) y del puesto actual (como ocupado). Finalmente se notifica al usuario que el movimiento de nómina se ha adicionado satisfactoriamente.

Análisis de complejidad del algoritmo no trivial:

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.

Teniendo en cuenta la complejidad ciclomática se podrá conocer qué tan complejo es el algoritmo presentado. Se parte de un análisis del código o el diseño del mismo y se enumeran cada una de las instrucciones las cuales representan un camino que se pudiera seguir al ejecutar el algoritmo, a continuación se enumeran dichas instrucciones y a partir de las fórmulas pertinentes se calcula la complejidad ciclomática.

```
public function insertarMovimiento($movimiento, $params) {
    $params->idestado = 10;
    $movimientoanterior = $this->datmovimiento->getMovimentosByParameters($params, $limit, $start);
    if(is_array($movimientoanterior)) throw new ZendExt_Exception('CHT079');
    $trabajador = $this->datTrabajador->Buscar($params->idtrabajador);
    $params->idestado = 4;
    $movimientoanterior = $this->datmovimiento->getMovimentosByParameters($params, $limit, $start);
    if($movimiento->tipomovimiento == 1 || $movimiento->tipomovimiento == 5) $changes = true;
    else $changes = $this->CompararPagosAdicionales($trabajador, $movimientoanterior[0]);
    if(!$changes) throw new ZendExt_Exception('CHT033');
    else {
        $aux = $this->datmovimiento->getMovimento($movimiento->numeroconsecutivo);
        if(count($aux) != 0) throw new ZendExt_Exception('CHT042');
        else {
            $filter->idpuestotrabajo = $trabajador->idpuestotrabajo;
            $puestotrabajo = $this->integrator->capitalhumano->ObtenerPuestoTrabajo($filter);
            $t = $this->datTrabajador->getTrabajadores($filter);
            if(($movimiento->tipomovimiento == 1 && $puestotrabajo[0]->ocupado == 1)
                || count($t) > 2) throw new ZendExt_Exception('CHT043');
            $fechaactual = date('d/m/Y');
            $salarioescala = $trabajador->salariotrabajador;
            $elaboradopor = $this->global->Perfil->idusuario;
            $this->datmovimiento->idmotivomovnomina = $movimiento->idmotivomovnomina;
            $this->datmovimiento->fechaemision = $movimiento->fechaemision;
            $this->datmovimiento->fechaefectiva = $movimiento->fechaefectiva;
            $this->datmovimiento->fechafin = $movimiento->fechafinreubicacion;
            $this->datmovimiento->observaciones = $movimiento->observaciones;
            $this->datmovimiento->version = 0;
            $this->datmovimiento->idtrabajador = $movimiento->idtrabajador;
            $this->datmovimiento->iddetallesmovnomina = $movimiento->iddetallesmovnomina;
            if($this->datmovimiento->fechaefectiva <= $fechaactual) $this->datmovimiento->idestado = 4;
            else $this->datmovimiento->idestado = 10;
            $this->datmovimiento->idpuestotrabajo = $filter->idpuestotrabajo;
            $this->datmovimiento->iddocumento = $movimiento->iddocumento;

            if($movimientoanterior[0]['idmodelomovnomina'])
                $this->datmovimiento->movimientoanterior = $movimientoanterior[0]['idmodelomovnomina'];
            else $this->datmovimiento->movimientoanterior = 0;
            $this->datmovimiento->salarioescala = $salarioescala;
            $this->datmovimiento->idelaboradopor = $elaboradopor;
            $idmovimiento = $this->datmovimiento->Insertar($this->datmovimiento);
            if(!$idmovimiento) throw new ZendExt_Exception('CHT040');
            if($idmovimiento && $movimientoanterior[0]['idmodelomovnomina'] != 0) {
                $anterior->idmodelomovnomina = $movimientoanterior[0]['idmodelomovnomina'];
                if($this->datmovimiento->fechaefectiva <= $fechaactual) $anterior->idestado = 5;
                $anterior->version = $movimientoanterior[0]['version']++;
                $success1 = $this->datmovimiento->Actualizar($anterior);
                if(!$success1) throw new ZendExt_Exception('CHT041');
            }
            $objeto->idmodelomovnomina = $idmovimiento;
            $objeto->idtrabajador = $movimiento->idtrabajador;
            $objeto->idpuestotrabajo = $filter->idpuestotrabajo;
            $success3 = $this->pIntegrator->pagos_adicionales
                        ->InsertarPagosAdicionalesMovimiento ($objeto);
            return $objeto;
        }
    }
}
```

Figura 15. Representación del algoritmo no trivial (insertarMovimiento ()).

Después de este paso, es necesario representar el grafo de flujo asociado al código antes presentado:

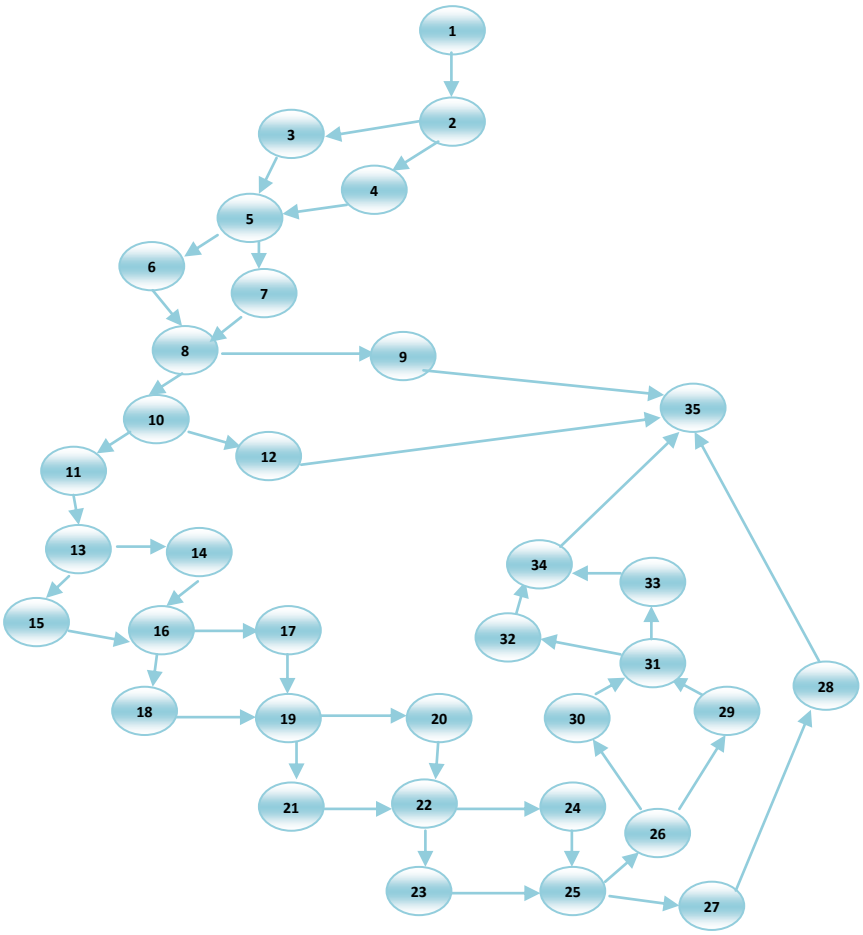

Figura 16. Grafo de flujo asociado al algoritmo no trivial (insertarMovimiento ()).

Luego de construir el grafo de flujo asociado al algoritmo no trivial insertarMovimiento () se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A – N) + 2

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

V (G) = (45– 35) + 2

V (G) = 12

2. V (G) = P + 1

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

V (G) = 11 +1

V (G) =12

3. V (G) = R

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

V (G) = 12

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática igual a 12 lo que significa que a la hora de ejecutar los caminos lógicos del mismo debe ser a lo sumo doce caminos lógicos para recorrerlo.

2.3.6. Integración entre componentes

En el sistema las relaciones entre los componentes se realizan a través el IoC cada componente tiene al menos una clase Service que se encarga de publicar los métodos o servicios que se le pueden brindar a otras líneas o componentes. Para permitir la comunicación en la carpeta común de cada subsistema hay un XML con el nombre IoC donde se publican los métodos o servicios que se brindan a las líneas. Para acceder a los servicios de otras líneas se llama al objeto integrator, el nombre del subsistema, el componente y el nombre del método publicado en el IoC que está en el común del sistema. Para acceder al servicio en el mismo subsistema se llama al objeto pIntegrator, el componente y el nombre del método publicado en el IoC en la carpeta común del subsistema. Ejemplo:

```
$datos = $this->pIntegrator->persona->BuscarPersonaPorParam ($params);
$centros = $this->integrator->costosprocesos->centrosCosto ($_SESSION ['idestructura']);
```

Publicación de servicios

El modelo de publicación de servicios consiste en utilizar la menor cantidad de métodos e implementar los mismos cumpliendo con la condición de que reciban uno o muchos parámetros. Tratar de publicar los mismos métodos utilizados para recuperar los datos en cada uno de los componentes poniendo en práctica la reutilización del código.

Tabla 2: Servicios brindados por componentes

Componentes	Servicios que brinda	Componentes que lo utilizan	Código	Descripción
Trabajador	GuardarTrabajador		CHTrabajador01	Persistir Trabajadores en la base de datos
	BuscarTrab	Nómina Puesto de trabajo	CHTrabajador02	Servicio para levantar el winIFrame buscar trabajador
	BuscarTrabajador	Incidencia Trabajador Nómina	CHTrabajador03	Obtener trabajador/trabajadores dado un set de parámetros
	FiltrarTrabajad	Persona	CHTrabajador04	Obtener persona ó

	or	trabajador		trabajador/personas ó trabajadores dado un set de parámetros
	ObtenerContratos	Puesto trabajo Nómina	CHTrabajador05	Obtener todos los contratos o uno por parámetros
	ObtenerTipoVinculacion	Trabajador Nómina	CHTrabajador06	Obtener todos los tipos de vinculación o uno por parámetros
	ObtenerEstado	Nómina	CHTrabajador07	Obtener los estados por concepto o por parámetros
Puesto de Trabajo	ModificarEstadoPuestoTrab	trabajador	CHPuestoTrabajo01	Modifica el estado de un puesto de trabajo al ser liberado u ocupado por un trabajador
	ObtenerPuestoTrabajo	Nómina trabajador	CHPuestoTrabajo02	Obtener puesto/puestos de trabajo dado un set de parámetros
	VerificarDisponibilidadPuesto	Trabajador	CHPuestoTrabajo03	Verificar si el puesto de trabajo está aún disponible
	ObtenerGruposPuestos	Trabajador	CHPuestoTrabajo04	Obtener todos los grupos de puestos de trabajo de la entidad que no estén eliminados
Pagos Adicionales	ObtenerPagoAdicional	Puesto trabajo Trabajador Nómina	CHPagos01	Obtiene los pagos adicionales dado un set de parámetros
	InsertarPagos Adicionales	Trabajador Puesto trabajo	CHPagos02	Insertar pagos adicionales del trabajador o del puesto de trabajo
	InsertarPagos AdicionalesMovimiento	trabajador	CHPagos03	Insertar pagos adicionales del movimiento de nómina. Se recuperan los pagos del trabajador por el idtrabajador recibido, los pagos del puesto de trabajo por el idpuestotrabajo recibido y se insertan en la tabla intermedia entre el nomenclador de pagos adicionales y movimiento de nómina.
	EliminarPagos Adicionales	Trabajador Puesto trabajo	CHPagos04	Eliminar pagos adicionales del trabajador o del puesto de trabajo
	RevertirPagos Adicionales	Trabajador Puesto trabajo	CHPagos05	Revertir los pagos adicionales de un movimiento. Se eliminan los pagos adicionales del trabajador en la tabla intermedia entre el nomenclador de pagos adicionales y el trabajador, se clasifican los pagos adicionales del movimiento de nómina para desechar los que sean asociados al puesto de trabajo, se insertan los asociados al trabajador en la tabla intermedia.

	CompararPagosMovimiento	Trabajador	CHPagos06	Compara los pagos del movimiento de nómina con los del trabajador y el puesto de trabajo para al insertar un movimiento de nómina verificar que se hayan efectuado cambios con respecto al movimiento anterior.
	ObtenerCategorias	Trabajador Puesto trabajo	CHPagos07	Obtiene las categorías de los pagos adicionales

2.3.7. Diagrama de despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento.

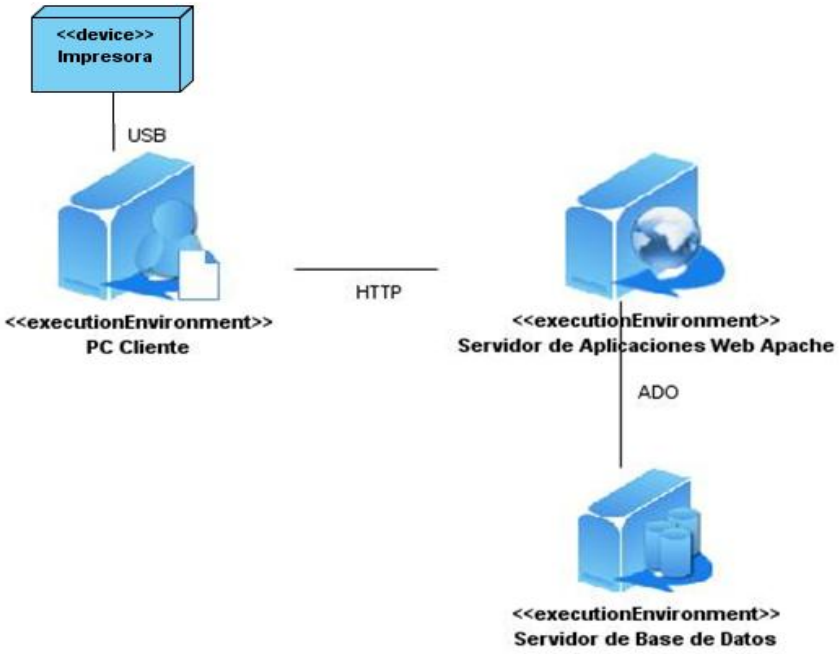

Figura 17: Diagrama de Despliegue

Conclusiones parciales del capítulo

En el capítulo se realizó el diseño de los componentes Pagos Adicionales, Puesto de Trabajo y Movimiento de Fuerza de Trabajo utilizando patrones de diseño y de asignación de responsabilidades. Este diseño permitió la implementación de los componentes de manera cómoda y garantizando que se diera cumplimiento a los objetivos.

Capítulo 3: Validación de la solución. Pruebas.

Introducción

En este capítulo se abordarán temas como las métricas para validar la solución propuesta, las pruebas realizadas al software, en específico las pruebas de caja blanca y caja negra, así como la demostración de la eficiencia de los algoritmos, además se hace una valoración de las mismas según los resultados obtenidos.

3.1. Métricas para validar diseño.

Para medir la calidad del diseño se utilizaron métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto teniendo en cuenta que este estudio brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software.

Atributos de calidad que se abarcan:

- Responsabilidad. Es la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad de implementación. Es el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización. Es el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento. Es 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. Es el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas. Es el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.

Las métricas concebidas como instrumento para evaluar la calidad del diseño de los componentes Trabajador, Pagos Adicionales y Puesto de Trabajo del subsistema CH 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).

3.1.1. Tamaño operacional de clase

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

El instrumento que se utiliza para efectuar la validación de la métrica TOC está dado en la tabla 3.

Tabla 3. Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación, Reutilización) relacionados con la métrica TOC.

	Categoría	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. Y 2* Prom.
	Alta	> 2* Prom.
Complejidad de Implementación	Baja	< =Prom.
	Media	Entre Prom. Y 2* Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. Y 2* Prom.
	Alta	<= Prom.

En la tabla 4 se ilustran las clases del sistema aplicándole la métrica seleccionada.

Tabla 4.Resultados de la aplicación de la métrica TOC

No	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	GestionarpagosadicionalesController	14	Media	Media	Media
2	DatPuestotrabajopagosadicionalesModel	4	Baja	Baja	Alta
3	DatRegistropagosadicionalestrabajadorModel	4	Baja	Baja	Alta
4	DatRegistropagosmovimientoModel	3	Baja	Baja	Alta
5	NomCategoriapagosadicionalesModel	3	Baja	Baja	Alta
6	NomPagosadicionalesModel	16	Alta	Alta	Baja
7	PagosAdicionalesManager	8	Media	Media	Media
8	BaseDatPuestotrabajopagosadicionales	2	Baja	Baja	Alta
9	BaseDatRegistropagosadicionalestrabajador	2	Baja	Baja	Alta
10	BaseDatRegistropagosmovimiento	2	Baja	Baja	Alta
11	BaseNomCategoriapagosadicionales	2	Baja	Baja	Alta
12	BaseNomPagosadicionales	2	Baja	Baja	Alta
13	DatPuestotrabajopagosadicionales	8	Media	Media	Media
14	DatRegistropagosadicionalestrabajador	8	Media	Media	Media
15	DatRegistropagosmovimiento	9	Media	Media	Media
16	NomCategoriapagosadicionales	8	Media	Media	Media
17	NomPagosadicionales	9	Media	Media	Media
18	PagosAdicionalesService	9	Media	Media	Media
19	GestpagosadicionalesValidator	3	Baja	Baja	Alta
20	BuscartrabajadorController	12	Media	Media	Media

21	DetallesmotivoController	8	Media	Media	Media
22	MotivomovimientoController	9	Media	Media	Media
23	MovimientofuerzatrabajoController	30	Alta	Alta	Baja
24	RealizarcontratoController	35	Alta	Alta	Baja
25	BuscarTrabajadorModel	3	Baja	Baja	Alta
26	DatContratoModel	12	Media	Media	Media
27	DatDatoscontablesModel	4	Baja	Baja	Alta
28	DatModelomovnominaModel	12	Media	Media	Media
29	DatTrabajadorModel	13	Media	Media	Media
30	FuerzaTrabajoManager	42	Alta	Alta	Baja
31	NomDetallesmovimientoModel	7	Baja	Baja	Alta
32	NomEstadoModel	2	Baja	Baja	Alta
33	NomMotivomovnominaModel	8	Media	Media	Media
34	NomTipocontratoModel	2	Baja	Baja	Alta
35	NomTipomovnominaModel	4	Baja	Baja	Alta
36	NomTipoperiodopagoModel	4	Baja	Baja	Alta
37	NomTiporelacionlaboralModel	4	Baja	Baja	Alta
38	NomTipovinculacionModel	2	Baja	Baja	Alta
39	BaseDatContrato	2	Baja	Baja	Alta
40	BaseDatDatoscontables	2	Baja	Baja	Alta
41	BaseDatModelomovnomina	2	Baja	Baja	Alta
42	BaseDatTrabajador	2	Baja	Baja	Alta
43	BaseNomDetallesmovimiento	2	Baja	Baja	Alta
44	BaseNomEstado	2	Baja	Baja	Alta
45	BaseNomMotivomovnomina	2	Baja	Baja	Alta
46	BaseNomTipocontrato	2	Baja	Baja	Alta
47	BaseNomTipomovnomina	2	Baja	Baja	Alta
48	BaseNomTipoperiodopago	2	Baja	Baja	Alta
49	BaseNomTiporelacionlaboral	2	Baja	Baja	Alta
50	BaseNomTipovinculacion	2	Baja	Baja	Alta
51	DatContrato	10	Media	Media	Media
52	DatDatoscontables	9	Media	Media	Media
53	DatModelomovnomina	10	Media	Media	Media
54	DatTrabajador	11	Media	Media	Media
55	NomDetallesmovimiento	9	Media	Media	Media
56	NomEstado	9	Media	Media	Media
57	NomMotivomovnomina	13	Media	Media	Media
58	NomTipocontrato	9	Media	Media	Media
59	NomTipomovnomina	6	Baja	Baja	Alta
60	NomTipoperiodopago	5	Baja	Baja	Alta
61	NomTiporelacionlaboral	5	Baja	Baja	Alta
62	NomTipovinculacion	9	Media	Media	Media
63	BuscarTrabajadorService	2	Baja	Baja	Alta
64	ContratoService	4	Baja	Baja	Alta
65	MovimientoNominaService	5	Baja	Baja	Alta
66	TrabajadorService	3	Baja	Baja	Alta

67	DatoscontablesValidator	2	Baja	Baja	Alta
68	EmitirmovaltaValidator	3	Baja	Baja	Alta
69	EmitirmovimientobajaValidator	2	Baja	Baja	Alta
70	EmitirmovimientoreubicacionValidator	2	Baja	Baja	Alta
71	GestionarpuestotrabajoController	30	Alta	Alta	Baja
72	DatPuestogrupoModel	7	Baja	Baja	Alta
73	DatPuestotrabajoModel	33	Alta	Alta	Baja
74	NomGrupopuestotrabajoModel	6	Baja	Baja	Alta
75	NomSistemapagoModel	3	Baja	Baja	Alta
76	BaseDatPuestogrupo	2	Baja	Baja	Alta
77	BaseDatPuestotrabajo	2	Baja	Baja	Alta
78	BaseNomGrupopuestotrabajo	2	Baja	Baja	Alta
79	BaseNomSistemapago	2	Baja	Baja	Alta
80	DatPuestogrupo	9	Media	Media	Media
81	DatPuestotrabajo	12	Media	Media	Media
82	NomGrupopuestotrabajo	8	Media	Media	Media
83	NomSistemapago	6	Baja	Baja	Alta
84	PuestoTrabajoService	7	Baja	Baja	Alta
85	PuestotrabajoValidator	9	Media	Media	Media
86	RegimentrabajoController	20	Alta	Alta	Baja
87	DatRegimenconfigModel	4	Baja	Baja	Alta
88	DatRegimendiasModel	4	Baja	Baja	Alta
89	DatRegimensesionModel	4	Baja	Baja	Alta
90	DatRegimentrabajodescansoModel	6	Baja	Baja	Alta
91	BaseDatRegimenconfig	2	Baja	Baja	Alta
92	BaseDatRegimendias	2	Baja	Baja	Alta
93	BaseDatRegimensesion	2	Baja	Baja	Alta
94	BaseDatRegimentrabajodescanso	2	Baja	Baja	Alta
95	DatRegimenconfig	5	Baja	Baja	Alta
96	DatRegimendias	9	Media	Media	Media
97	DatRegimensesion	6	Baja	Baja	Alta
98	DatRegimentrabajodescanso	6	Baja	Baja	Alta
99	RegimenTrabajoService	2	Baja	Baja	Alta

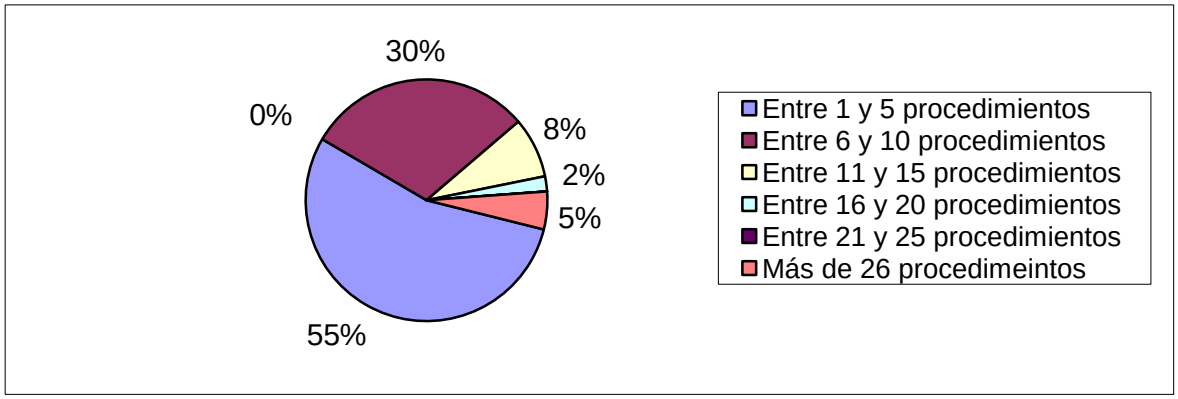

Figura 18. Representación de los resultados obtenidos en el instrumento agrupados en los

intervalos definidos

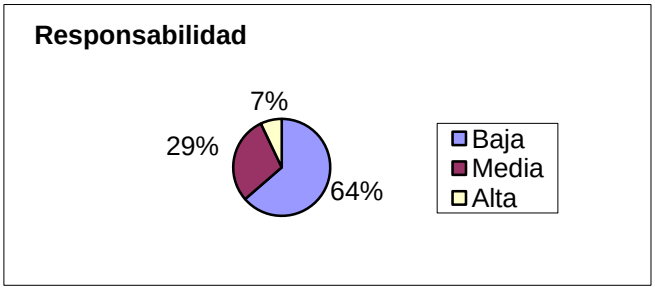

Figura 19. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

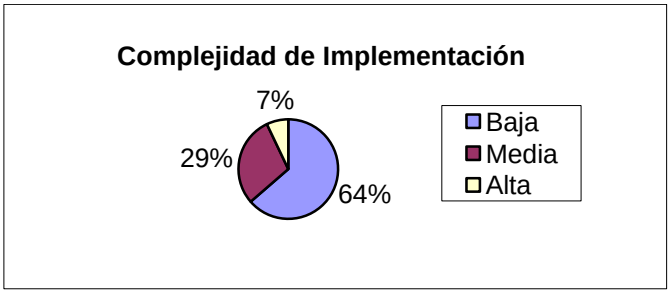

Figura 20. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación

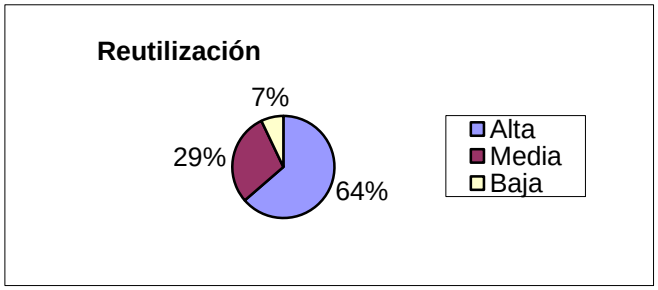

Figura 22. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, teniendo en cuenta que el 85% de las clases cuenta con un rango de funcionalidades entre 1 y 10, y que el 64% 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.

3.1.2. Relaciones entre Clases

Esta métrica está dada por la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. Se le aplica a las mismas clases que le fue aplicada

la métrica TC. Los aspectos de calidad que se miden son: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas.

El instrumento que se utiliza para efectuar la validación de la métrica RC está dado en la tabla 5.

Tabla 5. 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
	Categoría	Criterio
Complejidad de Mantenimiento.	Baja	<= Prom.
	Media	Entre Prom. Y 2*Prom.
	Alta	> 2*Prom.
	Categoría	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. Y 2*Prom.
	Alta	<= Prom.
	Categoría	Criterio
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. Y 2*Prom.
	Alta	> 2*Prom.

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.

Tabla 6. Cantidad de relaciones de uso entre las clases (RU).

No	Clase	RU	Acoplamiento	Complejidad de Mantenimiento.	Reutilización	Cantidad de Pruebas
1	GestionarpagosadicionalesController	1	Bajo	Baja	Alta	Baja
2	DatPuestotrabajopagosadicionalesModel	1	Bajo	Baja	Alta	Baja
3	DatRegistropagosadicionalestrabajadorModel	4	Alto	Alta	Baja	Alta
4	DatRegistropagosmovimientoModel	1	Bajo	Baja	Alta	Baja
5	NomCategoriapagosadicionalesModel	1	Bajo	Baja	Alta	Baja
6	NomPagosadicionalesModel	4	Alto	Alta	Baja	Alta
7	PagosAdicionalesManager	4	Alto	Alta	Baja	Alta
8	BaseDatPuestotrabajopagosadicionales	0	Ninguno	Baja	Alta	Baja
9	BaseDatRegistropagosadicionalestrabajador	0	Ninguno	Baja	Alta	Baja

10	BaseDatRegistropagosmovimiento	0	Ninguno	Baja	Alta	Baja
11	BaseNomCategoriapagosadicionales	0	Ninguno	Baja	Alta	Baja
12	BaseNomPagosadicionales	0	Ninguno	Baja	Alta	Baja
13	DatPuestotrabajopagosadicionales	1	Bajo	Baja	Alta	Baja
14	DatRegistropagosadicionalestrabajador	1	Bajo	Baja	Alta	Baja
15	DatRegistropagosmovimiento	1	Bajo	Baja	Alta	Baja
16	NomCategoriapagosadicionales	1	Bajo	Baja	Alta	Baja
17	NomPagosadicionales	1	Bajo	Baja	Alta	Baja
18	PagosAdicionalesService	1	Bajo	Baja	Alta	Baja
19	GestpagosadicionalesValidator	1	Bajo	Baja	Alta	Baja
20	BuscartrabajadorController	4	Alto	Alta	Baja	Alta
21	DetallesmotivoController	1	Bajo	Baja	Alta	Baja
22	MotivomovimientoController	2	Medio	Media	Media	Media
23	MovimientofuerzatrabajoController	1	Bajo	Baja	Alta	Baja
24	RealizarcontratoController	1	Bajo	Baja	Alta	Baja
25	BuscarTrabajadorModel	3	Alto	Alta	Baja	Alta
26	DatContratoModel	6	Alto	Alta	Baja	Alta
27	DatDatoscontablesModel	1	Bajo	Baja	Alta	Baja
28	DatModelomovnominaModel	6	Alto	Alta	Baja	Alta
29	DatTrabajadorModel	2	Medio	Media	Media	Media
30	FuerzaTrabajoManager	4	Alto	Alta	Baja	Alta
31	NomDetallesmovimientoModel	3	Alto	Alta	Baja	Alta
32	NomEstadoModel	1	Bajo	Baja	Alta	Baja
33	NomMotivomovnominaModel	2	Medio	Media	Media	Media
34	NomTipocontratoModel	1	Bajo	Baja	Alta	Baja
35	NomTipomovnominaModel	0	Ninguno	Baja	Alta	Baja
36	NomTipoperiodopagoModel	0	Ninguno	Baja	Alta	Baja
37	NomTiporelacionlaboralModel	0	Ninguno	Baja	Alta	Baja
38	NomTipovinculacionModel	1	Bajo	Baja	Alta	Baja
39	BaseDatContrato	0	Ninguno	Baja	Alta	Baja
40	BaseDatDatoscontables	0	Ninguno	Baja	Alta	Baja
41	BaseDatModelomovnomina	0	Ninguno	Baja	Alta	Baja
42	BaseDatTrabajador	0	Ninguno	Baja	Alta	Baja
43	BaseNomDetallesmovimiento	0	Ninguno	Baja	Alta	Baja
44	BaseNomEstado	0	Ninguno	Baja	Alta	Baja
45	BaseNomMotivomovnomina	0	Ninguno	Baja	Alta	Baja
46	BaseNomTipocontrato	0	Ninguno	Baja	Alta	Baja
47	BaseNomTipomovnomina	0	Ninguno	Baja	Alta	Baja
48	BaseNomTipoperiodopago	0	Ninguno	Baja	Alta	Baja
49	BaseNomTiporelacionlaboral	0	Ninguno	Baja	Alta	Baja
50	BaseNomTipovinculacion	0	Ninguno	Baja	Alta	Baja
51	DatContrato	1	Bajo	Baja	Alta	Baja
52	DatDatoscontables	1	Bajo	Baja	Alta	Baja
53	DatModelomovnomina	1	Bajo	Baja	Alta	Baja

54	DatTrabajador	1	Bajo	Baja	Alta	Baja
55	NomDetallesmovimiento	1	Bajo	Baja	Alta	Baja
56	NomEstado	1	Bajo	Baja	Alta	Baja
57	NomMotivomovnomina	1	Bajo	Baja	Alta	Baja
58	NomTipocontrato	1	Bajo	Baja	Alta	Baja
59	NomTipomovnomina	1	Bajo	Baja	Alta	Baja
60	NomTipoperiodopago	1	Bajo	Baja	Alta	Baja
61	NomTiporelacionlaboral	1	Bajo	Baja	Alta	Baja
62	NomTipovinculacion	1	Bajo	Baja	Alta	Baja
63	BuscarTrabajadorService	1	Bajo	Baja	Alta	Baja
64	ContratoService	1	Bajo	Baja	Alta	Baja
65	MovimientoNominaService	2	Medio	Media	Media	Media
66	TrabajadorService	1	Bajo	Baja	Alta	Baja
67	DatoscontablesValidator	0	Ninguno	Baja	Alta	Baja
68	EmitirmovaltaValidator	1	Bajo	Baja	Alta	Baja
69	EmitirmovimientobajaValidator	1	Bajo	Baja	Alta	Baja
70	EmitirmovimientoreubicacionValidator	1	Bajo	Baja	Alta	Baja
71	GestionarpuestotrabajoController	3	Alto	Alta	Baja	Alta
72	DatPuestogrupoModel	2	Medio	Media	Media	Media
73	DatPuestotrabajoModel	4	Alto	Alta	Baja	Alta
74	NomGrupopuestotrabajoModel	1	Bajo	Baja	Alta	Baja
75	NomSistemapagoModel	1	Bajo	Baja	Alta	Baja
76	BaseDatPuestogrupo	0	Ninguno	Baja	Alta	Baja
77	BaseDatPuestotrabajo	0	Ninguno	Baja	Alta	Baja
78	BaseNomGrupopuestotrabajo	0	Ninguno	Baja	Alta	Baja
79	BaseNomSistemapago	0	Ninguno	Baja	Alta	Baja
80	DatPuestogrupo	1	Bajo	Baja	Alta	Baja
81	DatPuestotrabajo	1	Bajo	Baja	Alta	Baja
82	NomGrupopuestotrabajo	1	Bajo	Baja	Alta	Baja
83	NomSistemapago	1	Bajo	Baja	Alta	Baja
84	PuestoTrabajoService	4	Alto	Alta	Baja	Alta
85	PuestotrabajoValidator	4	Alto	Alta	Baja	Alta
86	RegimentrabajoController	5	Alto	Alta	Baja	Alta
87	DatRegimenconfigModel	0	Ninguno	Baja	Alta	Baja
88	DatRegimendiasModel	0	Ninguno	Baja	Alta	Baja
89	DatRegimensesionModel	0	Ninguno	Baja	Alta	Baja
90	DatRegimentrabajodescansoModel	0	Ninguno	Baja	Alta	Baja
91	BaseDatRegimenconfig	0	Ninguno	Baja	Alta	Baja
92	BaseDatRegimendias	0	Ninguno	Baja	Alta	Baja
93	BaseDatRegimensesion	0	Ninguno	Baja	Alta	Baja
94	BaseDatRegimentrabajodescanso	0	Ninguno	Baja	Alta	Baja
95	DatRegimenconfig	1	Bajo	Baja	Alta	Baja
96	DatRegimendias	1	Bajo	Baja	Alta	Baja
97	DatRegimensesion	1	Bajo	Baja	Alta	Baja
98	DatRegimentrabajodescanso	1	Bajo	Baja	Alta	Baja

99	RegimenTrabajoService	3	Alto	Alta	Baja	Alta

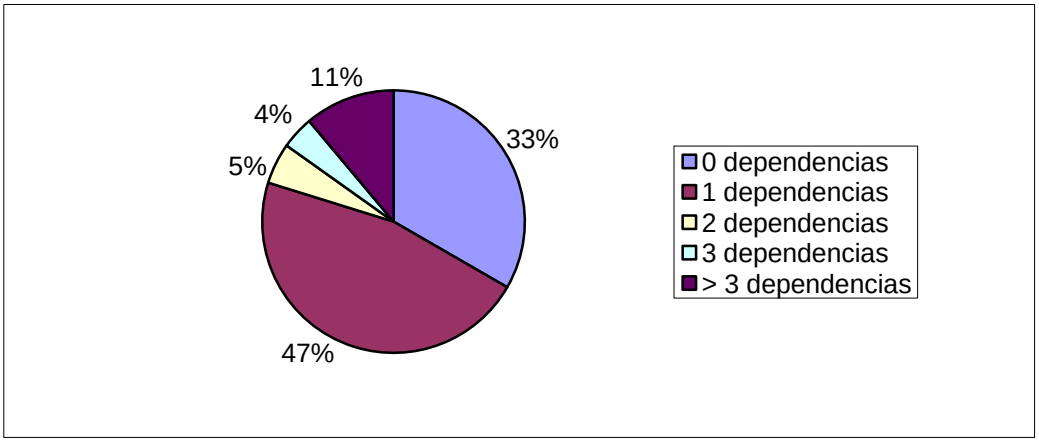

Figura 22. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

Los demás parámetros de calidad que mide esta métrica dependen del valor promedio de las dependencias de uso de todas las clases, en este caso ese promedio es de 1.18.

Para medir el acoplamiento según los resultados de esta métrica, algunos especialistas plantean los siguientes valores.

Tabla 7. Acoplamiento

Categoría	Relaciones de uso	Cantidad de Clases
Ninguno	0	33
Bajo	1	46
Medio	2	5
Alto	>2	15

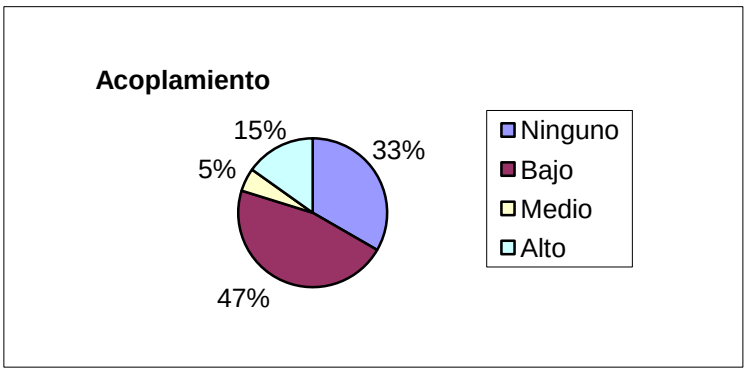

Figura 23. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

Tabla 8. Cantidad de Pruebas y Complejidad de Mantenimiento

Categoría	Criterio	Cantidad de Clases
Baja	<=Prom.	79
Media	>Prom. Y <=2*Prom.	5
Alta	>2*Prom.	15

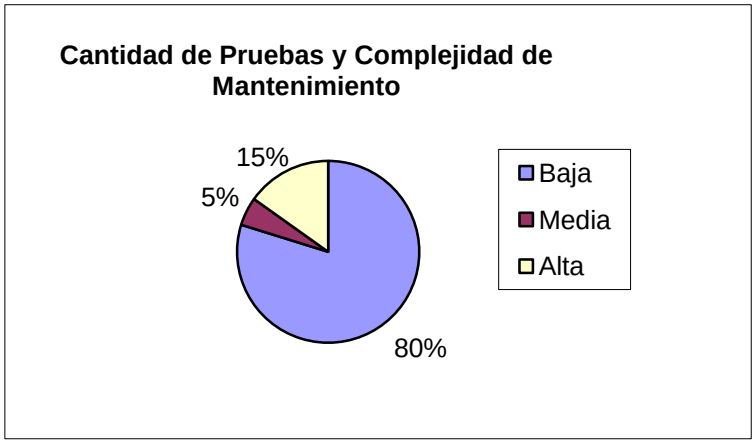

Figura 24. Representación de la incidencia de los resultados de la evaluación de la métrica RC en los atributos Cantidad de Pruebas y Complejidad de Mantenimiento.

Tabla 9. Reutilización

Categoría	Criterio	Cantidad de Clases
Baja	>2*Prom.	15
Media	>Prom. Y <=2*Prom.	5
Alta	<=Prom.	79

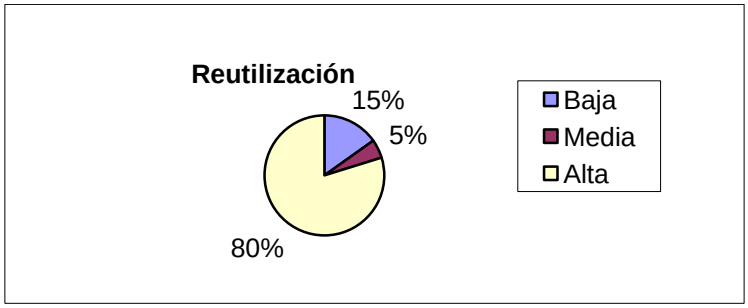

Figura 25. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, teniendo en cuenta que el 80% de las clases cuenta con un rango de funcionalidades entre 0 y 1, y que el 80% de las clases de dicho componente poseen evaluaciones positivas en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas), podemos concluir que el diseño es aceptable.

3.2. Pruebas de software.

La importancia de los costos que están asociados a los errores promovió a la definición y aplicación de pruebas detalladas y bien planificadas a los componentes Puesto de Trabajo, Pagos Adicionales y Trabajador del subsistema CH integrado al Sistema Cedrux. Las

mismas fueron un elemento fundamental para determinar la calidad de la solución propuesta, representando una revisión final de las especificaciones, del diseño y de la codificación.

Las pruebas, por su gran importancia se llevan a cabo durante todo el ciclo de vida del producto, su mayor punto de desarrollo se encuentra en la etapa de implementación.

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

- Caja Negra.
- Caja Blanca.

3.2.1. 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. Estas pruebas permiten encontrar: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos ó en accesos a las Bases de Datos externas, errores de rendimiento, errores de inicialización y terminación.

Como resultado de las pruebas de caja negra se realizaron un total de cincuenta y cinco casos de pruebas en los cuales se detectaron quince no conformidades en la primera iteración, a las cuales se les dio cumplimiento en las próximas 72 hrs. Para la segunda iteración se detectaron cuatro no conformidades dándole solución en 24 hrs. En la tercera iteración no se detectaron no conformidades y fueron liberados los componentes para entregar al departamento de Calidad del proyecto ERP-Cuba. Luego de varias iteraciones se resolvieron un total de treinta y una no conformidades y se liberaron los componentes Movimiento de Fuerza de Trabajo, Pagos Adicionales, Puesto de Trabajo y Régimen de Trabajo y Descanso los cuales salieron para piloto a seis entidades del país. Actualmente se encuentra en el departamento de Calidad de la Universidad.

3.2.2. Pruebas de caja blanca.

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

Dentro de la prueba de caja blanca, la técnica que se utilizó fue la de prueba del camino básico. Esta prueba permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la

definición de un conjunto básico de caminos de ejecución, además 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.

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

Para realizar la técnica de prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método InsertarPagosAdicionalesMovimiento ($object) el cual se encarga de insertar pagos adicionales para los trabajadores y para los puestos trabajo en un movimiento de nómina.

```
public function InsertarPagosAdicionalesMovimiento($object){
    $result = true;
    if(!$object->idmodelomovnomina || !$object->idtrabajador || !$object->idpuestotrabajo) return 0;
    $trabajador = $this->registroTrabajador->ObtenerObjeto($object);
    $puestotrabajo = $this->registroPuestotrabajo->ObtenerObjeto($object);
    if($trabajador != 0){
        $result = $this->registroMovimiento->InsertarRegistro($trabajador, $object->idmodelomovnomina);
    }
    if($puestotrabajo != 0){
        $result = $this->registroMovimiento->InsertarRegistro($puestotrabajo, $object->idmodelomovnomina);
    }
    return $result;
}
```

Figura 26. Representación del algoritmo método InsertarPagosAdicionalesMovimiento ($object)

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

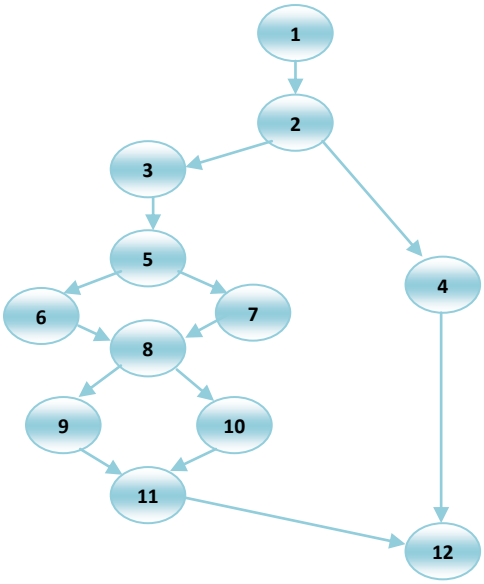

Figura 27. Grafo de flujo asociado al algoritmo InsertarPagosAdicionalesMovimiento ($movimiento, $params)

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

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) = (14 – 12) + 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 10. Caminos básicos del flujo

Número	Camino básico
1	1 – 2 – 3– 5 – 6 –8 – 9 – 11 – 12
2	1 – 2 – 3 – 5 – 6 – 8 – 10 – 11 – 12
3	1 – 2 – 3 – 5 – 7 – 8 – 9 – 11 – 12
4	1 – 2 – 3 – 5 – 7 – 8 – 10 – 11 – 12
5	1 – 2 – 4 – 12

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 –5 – 6 –8 – 9 – 11 – 12]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

- El parámetro object no está vacío, contiene los datos del movimiento de nómina que se conforman cuando se llenan todos los datos correspondientes al insertar un pago adicional en un movimiento de nómina.

Entrada:

$object->idmodelomovnomina = 9000303; $object->idtrabajador = 900000117; $object-> idpuestotrabajo = 900000136;

Resultados esperados del camino básico 1:

Se espera que se le agregue un pago adicional a un trabajar mediante un movimiento de nómina.

Caso de prueba para el camino básico 2

Camino 2: [1 – 2 – 3 – 5 – 6 –8 – 9 – 11 – 12]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

- El parámetro object no está vacío, contiene los datos del movimiento de nómina que

se conforman cuando se llenan todos los datos correspondientes al insertar un pago adicional en un movimiento de nómina, pero con un idpuestotrabajo no válido.

Entrada:

$object->idmodelomovnomina = 9000303; $object->idtrabajador = 900000117; $object-> idpuestotrabajo = 90000013;

Resultados esperados del camino básico 2:

Se espera que se le agregue un pago adicional a un trabajar mediante un movimiento de nómina pero no realiza un registro en los puesto de trabajo del movimiento.

Caso de prueba para el camino básico 3

Camino 3: [1 – 2 – 3 – 5 – 7 –8 – 10 – 11 – 12]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

- El parámetro object no está vacío, contiene los datos del movimiento de nómina que se conforman cuando se llenan todos los datos correspondientes al insertar un pago adicional en un movimiento de nómina, pero con un idtrabajador no válido.

Entrada:

$object->idmodelomovnomina = 9000303; $object->idtrabajador = 90000011; $object-> idpuestotrabajo = 900000136;

Resultados esperados del camino básico 3:

Se espera que se le agregue un pago adicional a un trabajar mediante un movimiento de nómina pero no realiza un registro al trabajador del movimiento.

Caso de prueba para el camino básico 4

Camino 4: [1 – 2 – 3 – 5 – 7 –8 – 10 – 11 – 12]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

- El parámetro object no está vacío, contiene los datos del movimiento de nómina que se conforman cuando se llenan todos los datos correspondientes al insertar un pago adicional en un movimiento de nómina, pero con un idtrabajador y un idpuestotrabajo no válido.

Entrada:

$object->idmodelomovnomina = 9000303; $object->idtrabajador = 90000011; $object-> idpuestotrabajo = 90000013;

Resultados esperados del camino básico 4:

Se espera que se le agregue un pago adicional a un trabajar mediante un movimiento de

nómina pero no realiza un registro al trabajador ni al puesto de trabajo del movimiento.

Caso de prueba para el camino básico 5

Camino 5: [1 – 2 – 4 – 12]

Descripción

Los datos de entrada cumplirán con los siguientes requisitos:

- El parámetro $object que contiene los datos del movimiento de nómina que se conforman cuando se llenan todos los datos correspondientes al insertar un pago adicional en un movimiento de nómina tenga alguno de sus parámetros vacíos o nulos.

Entrada:

$object->idmodelomovnomina = null; $object->idtrabajador = 900000117; $object-> idpuestotrabajo = 900000136;

Resultados esperados del camino básico 5:

Se espera que no agregue un pago adicional a un trabajar mediante un movimiento de nómina.

Resultados esperados de la prueba:

Luego de haberse aplicado pruebas de caja banca, en especifico las pruebas del camino básico, seleccionando diferentes caminos a través del Cálculo de la complejidad ciclomática a partir de un segmento de código, se ha arribado a la conclusión de que los resultados obtenidos fueron aceptables ya que se pudo comprobar que el flujo de trabajo de la función esta correcto ya que cumple con las condiciones necesarias que se habían planteado.

Resultados de las pruebas de caja blanca:

Para la validación de la implementación se le realizaron pruebas a un total de 566 operaciones de 77 clases en total, donde se construyeron un total de 7823 caminos.

Se detectaron errores en el retorno de los datos con: formatos incorrectos y duplicación, los cuales se solucionaron a medida que se fue implementado.

Conclusiones parciales del capítulo

En el capítulo se validó el diseño a través de las métricas demostrándose que posee bajo acoplamiento. Las pruebas de caja blanca, en específico la prueba del camino básico en las que se seleccionan diferentes caminos a través del cálculo de la complejidad ciclomática, a partir de un segmento de código de tamaño medio dieron los resultados esperados. En general se obtuvieron resultados favorables porque se pudo probar que los procesos implementados realizan las funcionalidades requeridas y cumplen con la calidad y eficiencia.

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.

Por lo tanto se concluye que:

- El estudio realizado a temas referentes a la gestión de los trabajadores y los puestos de trabajo como parte de los procesos de gestión de CH posibilita una mayor comprensión de requisitos necesarios en sistemas de este tipo.
- Se logró obtener un modelo diseño a partir de la descripción de los requisitos y se modeló la solución propuesta con las herramientas definidas por el marco de trabajo del proyecto ERP-Cuba.
- Los componentes desarrollados cumplen todos los requisitos analizados y se pueden integrar a otros componentes y módulos del proyecto.
- La validación del diseño a través de métricas de calidad, así como la realización de pruebas de caja blanca demuestra que los componentes desarrollados poseen un bajo acoplamiento entre sus clases y son fáciles de reutilizar.

Recomendaciones

Proseguir con la investigación e identificar nuevos requisitos para actualizar el sistema desarrollado.

Implementar la posibilidad de gestionar los grupos de puesto de trabajo en conjunto con estructura y composición.

Desplegar el sistema a otras entidades del territorio nacional.

Referencia Bibliográfica

1. definicion.de. *definicion.de.* [En línea] 2008. http://definicion.de/recursos-humanos/.

2. **Lazcano Herrera, MSc. Carlos y Font Graupera, Dra. Elena.** El prisma. [En línea] [Citado el: 18 de febrero de 2010.] http://www.elprisma.com/apuntes/administracion_de_empresas/capitalhumano/.

3. *Norma Cubana 3000 - SISTEMA DE GESTIÓN INTEGRADA DE CAPITAL HUMANO—VOCABULARIO.* Cuba : Oficina Nacional de Normalización, 2007.

4. Univesidad Nacional de Educación a Distancia. [En línea] [Citado el: 2 de febrero de 2010.] http://www.uned.es/deahe/alumnos/nuevo2000/apuntes/ade42204_derecho-leccion10.doc.

5. Concepto de Trabajador, Patrón, Intermediario, Empresa y Establecimiento. [En línea] [Citado el: 22 de febrero de 2010.] http://www.mitecnologico.com/Main/ConceptoDeTrabajadorPatronIntermediarioEmpresaYEstablecimiento. .

6. **Galuppo, Pablo.** SAP-Business Management Software Solutions Aplications and Services. *mySA ERP.* [En línea] 1 de Marzo de 2004. [Citado el: 22 de febrero de 2010.] http://www.sap.com/argentina/solutions/business-suite/erp/pdf/mySAP_ERP.pdf.

7. Ciberaula. [En línea] [Citado el: 22 de febrero de 2010.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee/.

8. **Rodas Hinostroza, Raul.** Características de PHP. [En línea] 22 de febrero de 2007. [Citado el: 22 de febrero de 2010.] http://www.linuxcentro.net/linux/staticpages/index.php?page=CaracteristicasPHP.

9. Geek OPEN TI. [En línea] [Citado el: 22 de febrero de 2010.] http://geeknet.com.mx/principal.php?op=detalle1.

10. Guía Breve de Tecnologías XML. [En línea] [Citado el: 22 de febrero de 2010.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML.

11. JSON. [En línea] [Citado el: 22 de febrero de 2010.] http://www.json.org/json-es.html.

12. Introdicción a XHTML. [En línea] [Citado el: 22 de febrero de 2010.] http://www.librosweb.es/xhtml/.

13. Guía Breve de CSS. [En línea] [Citado el: 22 de febrero de 2010.] http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo.

14. Manuales, Tutoriales y Herramientas. [En línea] [Citado el: 22 de febrero de 2010.] http://max-alva.webs.com/javascript.htm.

15. ¿Qué es un framework? [En línea] 29 de septiembre de 2006. [Citado el: 22 de febrero de 2010.] http://jordisan.net/blog/2006/que-es-un-framework/.

16. Ext español Comunidad de Desarrollo. [En línea] [Citado el: 23 de febrero de 2010.] www.extjs.es.

17. Ext 2.0 Alpha Release. [En línea] [Citado el: 22 de febrero de 2010.] http://extjs.com/blog/2007/09/28/ext-20-alpha-release/.

18. Zend Framework, una introducción. [En línea] 2007. [Citado el: 22 de febrero de 2010.] http://www.carlosleopoldo.com/post/zend-framework-una-introduccion/.

19. PHP Object Relational Mapper. [En línea] [Citado el: 22 de febrero de 2010.] http://www.doctrine-project.org/.

20. Object Management Group - UML. [En línea] [Citado el: 22 de febrero de 2010.] http://www.uml.org/.

21. El proyecto IDEE. [En línea] 21 de junio de 2007. [Citado el: 22 de febrero de 2010.] http://www.ineter.gob.ni/sig/docs/Intro%20IDE%20-%20IDEE.pdf.

22. El ataque de los Frameworks. [En línea] [Citado el: 22 de febrero de 2010.] http://www.webtaller.com/maletin/articulos/el-ataque-de-los-frameworks.php.

23. Conceptos básicos del servidor web. [En línea] [Citado el: 23 de febrero de 2010.] http://www.cibernetia.com/manuales/instalacion_servidor_web/1_conceptos_basicos.php.

24. Introducción a Apache. [En línea] [Citado el: 22 de febrero de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

25. **Lockhart, Thomas.** Forja. *Guía del Programador de PostgreSQL.* [En línea] 29 de octubre de 2001. [Citado el: 22 de febrero de 2010.] https://forja.rediris.es/docman/view.php/312/461/Postgres-Programmer.pdf.

26. EMS SQL Manager para PostgreSQL. [En línea] [Citado el: 22 de febrero de 2010.] http://sqlmanager.net/en/products/postgresql/manager.

27. ¿Qué es Mozilla Firefox? [En línea] [Citado el: 22 de febrero de 2010.] http://firefoxperu.blogspot.com/2005/12/qu-es-mozilla-firefox.html.

28. **Collins-Sussman, Ben, Fitzpatrick, Brian W. y Pilato, C. Michael.** Control de versiones con Subversion. [En línea] 2008. [Citado el: 22 de febrero de 2010.] http://svnbook.red-bean.com/nightly/es/svn-book.pdf.

29. umed.net. *umed.net.* [En línea] 2009. http://www.eumed.net/libros/2009c/584/RUP%20Diseno%20e%20implementacion%20del%20sistema.htm.

30. [En línea] 2009. http://www.google.com.cu/url?sa=t&source=web&ct=res&cd=6&ved=0CBMQFjAF&url=http%3A%2F%2Fvirtual.usalesiana.edu.bo%2Fweb%2Fpractica%2Farchiv%2Fdespliegue.doc&rct=j&q=diagrama+de+despliegue+&ei=9zrHS46eCILy9QT94_mPCw&usg=AFQjCNGu7s__JDdYCEwEmbkYnxa2XI2Q2Q..

31. **Pressman, Roger S.** *Ingeniería de Software. Un enfoque práctico.* 1998.

32. **Cabrera González, Lic. Miguel P., y otros.** Portal de Fórum de Ciencia y Técnica de Villa Clara. *XV FORUM DE CIENCIA Y TECNICA, SISTEMA ECONÓMICO INTEGRADO.* [En línea] [Citado el: 22 de febrero de 2010.] http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc.

33. BPMN. [En línea] [Citado el: 22 de febrero de 2010.] http://pana10.files.wordpress.com/2007/12/bpmn1.ppt.

34. ASSETS, Sistema de Gestión Integral. [En línea] [Citado el: 22 de febrero de 2010.] http://assets.co.cu/assets.asp..

35. **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides.** *Design Patterns.* 1995.

36. **Rubio, Ivonne.** *Seminario de valor competivo.* 2007.

37. **Lidia Fuentes, José M. Troya y Antonio Vallecillo.** *Desarrollo de Software Basado en Componentes.*

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

39. **Navathe, Elmasri y.** *Fundamentos de Sistemas de Bases de Datos.* 2002.

40. **Hector Garcia-Molina, Jeff Ullman, Jennifer Widom.** *Database systems: the complete book.* 2008.

41. Sitio oficial de Visual Paradigm. [En línea] http://www.visual-paradigm.com/.

42. Sitio oficial de Assets. [En línea] http://assets.co.cu/index.asp.

43. Sitio oficial de Rodas XXI. [En línea] http://www.rodasxxi.cu/index.php.

44. Sitio oficial de Sage Mas 500. [En línea] http://www.sagemas.com/products/sagemas500.

45. Ayuda del Versat Sarasola; Version 2.0 actualizacion 5.8.

46. Sitio oficial de PHP. [En línea] http://www.php.net/.

47. Sitio oficial de Extjs. [En línea] http://extjs.com/.

48. **Vallecillo, Lidia Fuentes y Antonio.** Una Introducción a los Perfiles UML. [En línea] http://www.lcc.uma.es/~av/Publicaciones/04/UMLProfiles-Novatica04.pdf.

49. Using BPMN to Model a BPEL Process. [En línea] http://www.bpmn.org/Documents/Mapping%20BPMN%20to%20BPEL%20Example.pdf .

50. **Cuba, Organismo Nacional de Normalización de la República de.** Norma Cubana 3000 . Ciudad de la Habana : s.n., 2007.

51. **Cuba, Asamblea Nacional del Poder Popular de la República de.** Ley 73. 1994.

52. —. Ley No. 49, Código de Trabajo. 1985.

53. **Cuba, Banco Nacional de.** Instrucción de Procedimiento No. 257 . 1968.

54. **Precios, Ministerio de Finanzas y.** Resolución 13 . 2007.

55. **Precios, Ministerio de Finanzas Y.** Resolución No. 105. 2008.

56. —. Resolución No. 116. 2002.

57. —. Resolución No. 198. 2006.

58. **Social, Ministerio de Trabajo y Seguridad.** Resolución 27 . 2006.

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

60. **Pressman, Roger S.** *Ingenieria de Software. Un enfoque practico* . 2005.

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

62. Zend Studio. [En línea] http://www.zend.com/products/studio/.

Anexos

Anexo 1
Descripción de las clases del diseño

Nombre: Gestionar Puesto Trabajo	
Tipo de clase: Controller	
Nombre :	gestionarpuestotrabajoAction()
Descripción :	Constructor de la clase.
Nombre :	cargarcomboareasAction()
Descripción :	Carga las aéreas de una estructura
Nombre :	cargarcombotipodemovimientoAction()
Descripción :	Carga los tipos de los movimientos de nómina
Nombre :	cargarcombomotivodemovimientoAction()
Descripción :	Carga los motivos de los movimientos de nómina
Nombre :	cargarcombosistemadepagosAction()
Descripción :	Carga los Sistemas de Pagos
Nombre :	cargarcomboGrupopuestoAction()
Descripción :	Carga los grupos de puesto de trabajo
Nombre :	cargarcomboRegimenTAction()
Descripción :	Carga el régimen de trabajo
Nombre :	cargarcomboSubordinadoAction()
Descripción :	Carga los puesto de trabajos a los que se puede subordinar el puesto a crear
Nombre :	cargarcombocontribucionAction()
Descripción :	Carga la contribuciones especiales de un puesto de trabajo
Nombre :	cargarpuestotrabajoAction()
Descripción :	Carga todos los puesto de trabajo de una entidad
Nombre :	cargarformulariopuestotrabajoAction()
Descripción :	Carga todos los datos de un puesto de trabajo
Nombre :	cargarPagosAdicionalesPuestoTrabajoresAction()
Descripción :	Carga todos los pagos adicionales de puesto de trabajo
Nombre :	cargarPagosAdicionalesPuestoTPorcategoriaAction()
Descripción :	Carga los pagos adicionales por categoría
Nombre :	cargarPagosAdicionalesPuestoTrabajoAction()
Descripción :	Carga los pagos adicionales de un puesto de trabajo
Nombre :	eliminarpagosAction()
Descripción :	Elimina los pagos adicionales de un puestos de trabajo
Nombre :	cargarcombocargosAction()
Descripción :	Carga los cargos de un área
Nombre :	verificaridmovimientoAction()
Descripción :	Verificar que no existan más id de movimiento
Nombre :	cargarcomboplantillapuestotrabajoAction()
Descripción :	Carga todos los grupos de puesto de trabajo de un área
Nombre :	comprobarDatosAreasAction()
Descripción :	Verifica que exista cargos en una área
Nombre :	comprobarGrupodePuestosAction()
Descripción :	Verifica que exista cargos en una área
Nombre :	adicionarpuestotrabajoAction()
Descripción :	Recoge los datos el puesto para adicionarlo
Nombre :	modificarpuestotrabajoAction()
Descripción :	Recoge los datos el puesto para modificar
Nombre :	cargarcomboCategoriaAction()
Descripción :	Cargas los pagos adicionales por categorías
Nombre :	eliminarpuestotrabajoAction()
Descripción :	Elimina un puesto de trabajo

Nombre :DatPuestotrabajoModel	
Tipo de clase: Model (Bussines)	
Nombre :	DatPuestotrabajoModel
Descripción :	Constructor de la clase.

Nombre :	obtenerEstructuraActual
Descripción :	Devuelve el id de la entidad
Nombre :	cargarTipodemovimiento()
Descripción :	Devuelve los tipo de movimientos
Nombre :	cargarMotivodemovimiento($idtipomov)
Descripción :	Devuelve los motivos de movimiento pasándole el id de este
Nombre :	cargarCategoria()
Descripción :	Devuelve las categorías de los pagos adicionales
Nombre :	cargarPagosAdicionalesPuestoTPorcategoria($params)
Descripción :	Devuelve los pagos adicionales por si categoría
Nombre :	cargarSistemadepagos()
Descripción :	Devuelve los sistemas de pagos
Nombre :	cargarRegimenTrabajo()
Descripción :	Devuelve los régimen de trabajo de los puesto de trabajos
Nombre :	cargarGrupopuesto()
Descripción :	Devuelve los grupo de puesto de trabajos
Nombre :	CantidadPuestos()
Descripción :	Devuelve la cantidad de puesto de trabajos de un área
Nombre :	cargarPuestotrabajo($params,$limite,$inicio)
Descripción :	Devuelve los puesto de trabajo pasándole un inicio y un limite
Nombre :	cargarformPuestotrabajo($params)
Descripción :	Devuelve los puesto de trabajo dado los parámetros
Nombre :	cargarPagosAdicionalesPuestoTrabajores()
Descripción :	Devuelve los pagos adicionales de un puesto de trabajo
Nombre :	cargarPagosAdicionalesPuestoTrabajo($idpuestotrabajo)
Descripción :	Devuelve los pagos adicional asignados a un puesto de trabajo
Nombre :	eliminarPagosAdicionales($arraypagos, $idpuestotrabajo)
Descripción :	Elimina los pagos adicionales
Nombre :	cargarcombocargos($tipo)
Descripción :	Devuelve los cargos por un área
Nombre :	cargarPlantillapuestotrabajo()
Descripción :	Devuelve los grupos de puestos de trabajos
Nombre :	cargarsistemapagos()
Descripción :	Devuelve los sistemas de pago de los puestos de trabajos
Nombre :	comprobarDatosAreas()
Descripción :	Verifica que exista aéreas en esa entidad
Nombre :	comprobarDatosCargos($tipoarea)
Descripción :	Verifica que existan cargos en el área
Nombre :	InsertarPuestoTrabajo($movimiento,$pagosadicionales)
Descripción :	Inserta un puesto de trabajo
Nombre :	actualizarPagosAdicionales($pagosadicionales,$idpuestotrabajo)
Descripción :	Modifica los pagos adicionales de un puesto de trabajo
Nombre :	modificarPuestoTrabajo($movimiento,$pagosadicionales)
Descripción :	Modifica el puesto de trabajo
Nombre :	eliminarPuestotrabajo($idpuestotrabajo)
Descripción :	Elimina el puesto de trabajo
Nombre :	getPuestoTrabajoPorParams($params,$limite,$inicio)
Descripción :	Devuelve los puesto de trabajo pasándole un inicio y un limite
Nombre :	getPuestoTrabajoPorParamsSinLimite($params)
Descripción :	Devuelve los puesto de trabajo

Nombre : NomGrupopuestotrabajoModel	
Tipo de clase:	
Nombre :	NomGrupopuestotrabajoModel()
Descripción :	Constructor de la clase.
Nombre :	Insertar(NomGrupopuestotrabajo $NomGrupopuestotrabajo)
Descripción :	Inserta un grupo de puesto de trabajo
Nombre :	Actualizar(NomGrupopuestotrabajo $NomGrupopuestotrabajo)
Descripción :	Modifica los grupo de puesto de trabajos
Nombre :	Eliminar($NomGrupopuestotrabajo)
Descripción :	Elimina los grupo de puesto de trabajos

Nombre :	buscaridarea($idgrupopuestotrabajo)
Descripción :	Devuelve el id del área de un grupo de puesto de trabajo
Nombre :	buscaridplantilla()
Descripción :	Devuelve el id de un grupo de puesto de trabajo
Nombre :	buscaridcargo($idgrupopuestotrabajo)
Descripción :	Devuelve el cargo de un grupo de puesto de trabajo
Nombre :	buscarcodigo($cod, $entidad)
Descripción :	Devuelve el código de un grupo de puesto de trabajos
Nombre :	buscardenominacion($idgrupopuestotrabajo, $denom, $entidad)
Descripción :	Devuelve la denominación de un grupo de puesto de trabajos

Nombre : NomSistemapago	
Tipo de clase:	
Nombre :	GetLLave()
Descripción :	Busca en la base de datos el último identificador y retorna este número más 1.
Nombre :	Buscar($idsistemapago)
Descripción :	Devuelve todos los sistemas de pagos
Nombre :	Getsistemapago($idsistemapago)
Descripción :	Devuelve un sistema de pago pasándole el id
Nombre :	GetPorLimite($limite = 10, $inicio = 0)
Descripción :	Devuelve los sistemas de pago pasándole un inicio y un limite
Nombre :	DebolverIdSistemaPagoporDenominacion($denom)
Descripción :	Devuelve un sistema de pago por la denominación de este
Nombre :	Obtenersistemapago($denom)
Descripción :	Devuelve un sistema de pago por la denominación de este
Nombre :	ObtenerSistemapagoPorPuesto($idsistemapago)
Descripción :	Devuelve un sistema de pago por la id de este

Nombre : DatPuestotrabajo	
Tipo de clase:	
Nombre :	GetLLave()
Descripción :	Busca en la base de datos el último identificador y retorna este número más 1.
Nombre :	Buscar($idpuestotrabajo)
Descripción :	Devuelve todo los datos del puesto de trabajo
Nombre :	GetTodos()
Descripción :	Devuelve todos los puesto trabajo
Nombre :	getPuestoTrabajodandoId($idpuestotrabajo)
Descripción :	Devuelve un puesto de trabajo por un id de puesto de trabajo
Nombre :	buscardenmodificar($den, $idpuesto)
Descripción :	Devuelve un grupo de puesto trabajo donde el id y el de puesto coincidan
Nombre :	buscarexistenciaGrupo($idplantilla, $idpuesto)
Descripción :	Verifica que el grupo de puesto de trabajo exista
Nombre :	GetPorLimite($idestructuracomun,$limite = 20, $inicio = 0)
Descripción :	Devuelve los puestos de trabajos pasándole un inicio y un límite
Nombre :	ObtenerGruposTrabajoPorPuesto($idpuestotrabajo)
Descripción :	Devuelve los grupo de puesto de trabajos de un puesto
Nombre :	Insertarpuesto($movimiento)
Descripción :	Inserta un puesto de trabajo
Nombre :	Actualizar($Puestotrabajo)
Descripción :	Modifica un puesto de trabajos
Nombre :	eliminarP($idpuestotrabajo)
Descripción :	Elimina un puesto de trabajo
Nombre :	getPuestoTrabajoPorParams($params, $limit = 20, $start = 0)
Descripción :	Devuelve los puesto de trabajos desde un inicio y un límite
Nombre :	getPuestoTrabajoPorParamsSinLimite($params)
Descripción :	Devuelve los puesto de trabajos cumpliendo con algunas de las condiciones
Nombre :	contar($params)
Descripción :	Devuelve la cantidad de puestos de trabajos por alguna condición

Anexo 2

Nombre :NomPagosadicionalesModel

Tipo de clase:	
Nombre :	ClasificarPagoAdicional($idpagosadicionales)
Descripción :	
Nombre :	getPagosAdicionales ($params, $limit = 20, $start = 0)
Descripción :	Devuelve los pagos adicionales que cumple con un inicio a un límite
Nombre :	obtenerEstructuraActual()
Descripción :	Devuelve el id de la entidad
Nombre :	obtenerElementosGasto()
Descripción :	Devuelve los elementos gastos
Nombre :	cargarElementosGasto()
Descripción :	Devuelve los elementos gastos
Nombre :	cargarCategorias()
Descripción :	Devuelve las categorías de los pagos adicionales
Nombre :	cargarGrid($limit, $start)
Descripción :	Devuelve todos los pagos adicionales desde un inicio hasta un límite
Nombre :	recargarDatosGrid($params)
Descripción :	Devuelve todos los pagos adicionales que cumple con los parámetros
Nombre :	InsertarPagosAdicionales($params)
Descripción :	Inserta un pago adicional
Nombre :	ActualizarPagosAdicionales($params)
Descripción :	Modifica un pago adicional
Nombre :	EliminarPagosAdicionales($params)
Descripción :	Elimina un pago adicional
Nombre :	validateCode($params)
Descripción :	Verifica que el código del pago adicional no exista en la base de datos
Nombre :	validateDenom($params)
Descripción :	Verifica que el denominación del pago adicional no exista en la base de datos

Nombre : PagosAdicionalesModel	
Tipo de clase:	
Nombre :	PagosAdicionalesModel()
Descripción :	Constructor de la clase
Nombre :	ObtenerPagoAdicional($params, $limit, $start)
Descripción :	Devuelve los pagos adicionales
Nombre :	InsertarPagosAdicionales($array)
Descripción :	Inserta un pago adicional
Nombre :	EliminarPagosAdicionales($array)
Descripción :	Elimina un pago adicional
Nombre :	InsertarPagosAdicionalesMovimiento($object)
Descripción :	Inserta un pago adicional a un movimiento
Nombre :	RevertirPagosAdicionales($object)
Descripción :	Deshace algún procedimiento realizado con el pago adicional
Nombre :	CompararPagosMovimiento($object)
Descripción :	Verifica que los pagos adicionales

Anexo 3

Nombre :BuscarTrabajadorModel	
Tipo de clase:	
Nombre :	BuscarTrabajadorModel()
Descripción :	Constructor de la clase
Nombre :	buscar($params, $limit = 20, $start = 0)
Descripción :	Devuelve los trabajadores que cumple con los parámetros desde un inicio a un limite
Nombre :	temporal($trabj, $puesto, $personas = null)
Descripción :	Devuelve los trabajadores que cumple con los parámetros

Nombre : DatContratoModel	
Tipo de clase:	
Nombre :	DatContratoModel()
Descripción :	Constructor de la clases
Nombre :	cargarTipoContrato($params)
Descripción :	Devuelve los tipo de contratos que cumpla con los parámetros

Nombre :	cargarTipoRelacionLaboral()
Descripción :	Devuelve los tipo de relación laboral
Nombre :	cargarTipoPeriodoPago()
Descripción :	Devuelve los tipo de período de pago
Nombre :	cargarTipoVinculacion($params)
Descripción :	Devuelve los tipo de vinculación que cumpla con los parámetros
Nombre :	cargarEstado($params)
Descripción :	Devuelve los estados que cumpla con los parámetros
Nombre :	obtenerDatosContrato()
Descripción :	
Nombre :	adicionarContrato($params)
Descripción :	
Nombre :	modificarContrato($contrato)
Descripción :	
Nombre :	hacerFijo($params)
Descripción :	

Nombre :DatTrabajadorModel	
Tipo de clase:	
Nombre :	DatTrabajadorModel()
Descripción :	Constructor de la cases
Nombre :	GetNumeroExpediente()
Descripción :	Devuelve un número de expediente
Nombre :	getPuestoTrabajo($idpuestotrabajo)
Descripción :	Devuelve los datos de un puesto de trabajo por el id de un puesto de trabajo
Nombre :	cargarGrid($params, $limit, $start)
Descripción :	Devuelve todos los trabajadores de unaentidad
Nombre :	getTrabajadores($params)
Descripción :	Devuelve los trabajadores que cumpla con los parámetros
Nombre :	buscarTrabajador($params, $limit, $start)
Descripción :	Devuelve los trabajadores que cumpla con los parámetros y una cierta cantidad
Nombre :	InsertarTrabajador($trabajador, $pagosadicionales)
Descripción :	Inserta un trabajador
Nombre :	ActualizarTrabajador($trabajador, $datoscontablesObject = null, $pagosadicionales = array())
Descripción :	Modifica un trabajador
Nombre :	EliminarTrabajador($trabajador)
Descripción :	Da baja a un trabajador
Nombre :	actualizarPuestoTrabajo($params)
Descripción :	Modifica el puesto de trabajo poniendo en estado de ocupado
Nombre :	actualizarPagosAdicionales($pagosadicionales, $idtrabajador)
Descripción :	Modifica los pagos adicionales de un trabajador
Nombre :	eliminarPagosAdicionales($arraypagos, $idtrabajador)
Descripción :	Elimina los pagos adicionales de un trabajador
Nombre :	actualizarSubmayores($params)
Descripción :	Actualiza los submayores del trabajador

Nombre :FuerzaTrabajoManager	
Tipo de clase: Model (domain)	
Nombre :	FuerzaTrabajoManager()
Descripción :	Constructor de la clases
Nombre :	contratosVencidos()
Descripción :	Verifica si hay trabajadores con contratos vencidos
Nombre :	obtenerDatosMovimiento()
Descripción :	Devuelve los datos del movimiento
Nombre :	obtenerDatosContrato($params)
Descripción :	Devuelve los datos del contrato que cumple con los parámetros
Nombre :	buscarPersona($params)
Descripción :	Devuelve las personas que cumple con los parámetros
Nombre :	comprobarDatosPuestoTrabajo($params)
Descripción :	Verifica los datos del puesto de trabajo

Nombre :	getPuestoTrabajo($parameters)
Descripción :	Devuelve un trabajador que cumpla con los parámetros
Nombre :	cargarAreas()
Descripción :	Devuelve las áreas
Nombre :	cargarPuestosTrabajo($params)
Descripción :	Devuelve los puesto de trabajo
Nombre :	cargarGrupoPuestos()
Descripción :	Devuelve los grupos de puesto de trabajo
Nombre :	cargarTiposMovimiento($tipomovimientonomina)
Descripción :	Devuelve datos de un tipo de movimientos
Nombre :	cargarMotivosMovimiento($tipomovimientonomina)
Descripción :	Devuelve los motivo de movimientos de tipo de movimiento
Nombre :	cargarTipoContrato($params = null)
Descripción :	Devuelve el tipo contrato
Nombre :	cargarTipoRelacionLaboral()
Descripción :	Devuelve el tipo relación laboral
Nombre :	cargarTipoPeriodoPago()
Descripción :	Devuelve el tipo de período de pago
Nombre :	cargarTipoVinculacion($params = null)
Descripción :	Devuelve las vinculación del trabajador
Nombre :	cargarEstado($params = null)
Descripción :	Devuelve todos los estados
Nombre :	cargarCentrosCosto()
Descripción :	Devuelve los centros de costo que se le puede asignar a un trabajador
Nombre :	cargarCuentas()
Descripción :	Cargas las cuentas que se le puede asignar al trabajador
Nombre :	cargarDetallesMovimientoPorMotivo($idmotivo)
Descripción :	Devuelve los detalles de un movimiento pasándonosle un motivo
Nombre :	cargarPagosAdicionalesPorTrabajador($idtrabajador, $limit, $start)
Descripción :	Devuelve todos los pagos adicionales que se le puede asignar un trabajador
Nombre :	cargarPagosAdicionalesPuestoTrabajo($idpuestotrabajo, $limit, $start)
Descripción :	Devuelve todos los pagos adicionales del puesto de trabajo asignado al trabajador
Nombre :	cargarPagosAdicionalesTrabajadores($idtrabajador, $limit, $start)
Descripción :	Devuelve los pagos adicionales de un trabajador
Nombre :	eliminarPagosAdicionales($arraypagos, $idtrabajador)
Descripción :	Elimina los pagos adicionales a un trabajador
Nombre :	cargarMovimientos($params, $boolean, $limit, $start)
Descripción :	Devuelve los movimientos de nómina que cumpla con los parámetros
Nombre :	cargarGridTrabajadores($params, $limit, $start)
Descripción :	Devuelve todos los trabajadores desde un inicio a un limite
Nombre :	cargarGridContratos($params, $boolean, $limit, $start)
Descripción :	Devuelve los contratos desde un inicio a un limite
Nombre :	insertarMovimiento($movimiento, $params)
Descripción :	Inserta un movimiento de nómina
Nombre :	revertirMovimiento($parameters)
Descripción :	Revierte un movimiento dejando actualizando todo a como estaba antes.
Nombre :	adicionarContrato($trabajador, $contrato, $pagosadicionales)
Descripción :	Inserta un contrato
Nombre :	modificarContrato($trabajador, $contrato, $pagosadicionales)
Descripción :	Modifica un contrato
Nombre :	hacerFijo($params)
Descripción :	Convierte a un trabajador en fijo en una puesto de trabajo
Nombre :	eliminarContrato($params)
Descripción :	Elimina un contrato
Nombre :	modificarTrabajador($params, $datoscontables, $pagosadicionales)
Descripción :	Modifica un trabajador
Nombre :	mostrarDatos($params)
Descripción :	Devuelve el trabajador
Nombre :	calcularSalarioTotal($params)
Descripción :	Devuelve el salario total de un trabajador

Glosario de términos

CH: Capital Humano.

Grupo de puestos de trabajo: es una agrupación lógica que el usuario puede hacer con el objetivo de que le sea más fácil el trabajo con la aplicación. Un puesto puede estar en ninguno, uno o varios grupos.

HTML: Hyper Text Markup Language en sus siglas en inglés (HTML), es el Lenguaje de Marcas de Hipertexto predominante para la construcción de páginas web. Usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

NC: Norma Cubana.

Nóminas: Según la Resolución No. 13-2007 del Ministerio De Finanzas y Precios de nuestro país el objetivo de la nómina es relacionar a todos los trabajadores de la entidad que perciban salarios y que les correspondan haberes por concepto de: sueldos, jornales, primas, vinculación, vacaciones, licencias y subsidios, obteniéndose la conformidad del cobro efectuado mediante la firma en este documento, siempre y cuando no se ejecute por Tarjetas Magnéticas.

Pago adicional: Según lo establecido en la Norma Cubana 3000 del 2007 son los pagos por trabajar en determinadas condiciones, cargos, actividades, ramas o sectores debido a su importancia económica, o del servicio que se presta, o como reconocimiento a trabajadores por sus aportes excepcionales y significativos al desarrollo económico y social del país y otros factores extracalificatorios.

Pago adicional de comportamiento asignado: es un tipo de pago que su importe se convierte en el salario bruto de quien lo recibe.

Pago adicional de comportamiento fijo: es un tipo de pago que su importe no varía.

Pago adicional de comportamiento por tarifa: es un tipo de pago que su importe depende de las horas trabajadas bajo el beneficio del pago.

Pago adicional de comportamiento porcentual: es un tipo de pago que su importe depende del importe de otros pagos adicionales y/o del salario escala y/o salario básico.

Pago adicional de comportamiento proporcional: es un tipo de pago que su importe varía de acuerdo a la proporcionalidad entre el tiempo a trabajar y el tiempo real trabajado.

Plantilla: Documento de ejemplo que sustenta un formato y que describe los lineamientos para la elaboración de documentos similares.

Proceso: Según lo establecido en el apartado 3.4.1 de la NC ISO 9000:2005, el proceso es

un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados.

Puestos de trabajo: Según lo establecido en la Norma Cubana 3000 del 2007 es una zona o lugar en que se ejecuta la actividad laboral por un trabajador o grupo de trabajadores equipados con instrumentos y medios de trabajo necesarios para su realización.

RH: Recursos Humanos.

Salario: Según lo establecido en la Norma Cubana 3000 del 2007 se considera salario la 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.

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.

Trabajador: es aquella persona que ocupa un puesto de trabajo en la entidad y desempeña una labor determinada por la cual recibe haberes o salarios.