Universidad de las Ciencias Informáticas

Facultad 3

“Desarrollo de un sistema para la gestión de la información de los recursos humanos del CEIGE”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informática

Autores: Rogelio Porta Trimiño

Rafael Macías Toledo

Tutores: Ing. Arnolis Salgueiro Arzuaga

MsC. Cealys Alvarez Trujillo

Lic. Lytyet Fernández Capestany

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 _____.

_____
Rafael Macias Toledo
Autor

_____
Rogelio Porta Trimiño
Autor

_____
Ing. Arnolis Salgueiro Arzuaga

_____
MsC. Cealys Alvarez Trujillo

_____
Lic. Lytyet Fernández Capestany

Ing. Arnolis Salgueiro Arzuaga: Ingeniero en Ciencias Informáticas. Graduado en la Universidad de las Ciencias Informáticas en el 2009. Actualmente se desempeña como jefe de línea de Capital humano del departamento Aduana-CCHH, CEIGE, Facultad 3.
Correo electrónico: asalgueiro@uci.cu

MsC. Cealys Alvarez Trujillo: Ingeniera en Ciencias Informáticas. Graduada de la Universidad de las Ciencias Informáticas en el 2007. Miembro de la Reserva del Comandante. Máster en gestión de proyectos informáticos y profesora Asistente. Asesora de Planificación del Director del CEIGE. Jefa del colectivo de Fundamentos de Administración y Gestión de Organizaciones de la F3 de la UCI.
Correo electrónico: calvarez@uci.cu

Lic. Lytyet Fernández Capestany: Licenciada en Contabilidad y Finanzas. Graduada de la Universidad Central Marta Abreu de las Villas en el 2008. Asesora económica del Director del CEIGE. Profesora del colectivo de Fundamentos de Administración y Gestión de Organizaciones en la facultad 3 de la UCI.
Correo electrónico: lytyet@uci.cu

Rafael

A mis padres por la educación, dedicación, confianza y amor que me han brindado toda la vida en especial a mi mamá por ser madre y amiga.

A mis abuelos por comprenderme y quererme como un hijo.

A mi hermanita por quererme y preocuparse tanto por mí, eres la mitad de mi ser.

A mi familia por estar presente cuando más la necesitaba, en especial a mi tío Nene.

A mis amigos en especial a Daniel que me ha dado su apoyo incondicional y me ha soportado durante toda la carrera.

A mis tutores Cealys, Lytyet y Arnolis por brindarme su ayuda en todo momento, por guiarme por el mejor camino, por sus consejos y sugerencias durante el desarrollo de este trabajo.

A la Revolución y a Fidel por haberme dado la posibilidad de estudiar, por engendrar tantos sueños y hacer los míos realidad.

A la UCI y todos sus profesores, que me permitió forjarme como profesional y como ser humano.

A todos los que de una forma u otra han aportado su granito de arena en mi formación personal y profesional.

Les entrego un mundo de gratitud.

Rogelio

Quiero agradecer primeramente a mi mamá, la cual ha sido mi motor impulsor durante toda mi vida y si hoy soy una mejor persona es gracias a ella, gracias por apoyarme y darme tu amor.

A mi tía Barby y a mi abuela Olga por ir a la par de mi mamá en mi educación y mi vida.

A mi familia de forma general por estar presente siempre de mí y apoyarme incondicionalmente.

A mis compañeros de aula por haberme soportado durante estos cinco años, de todos me llevo lo mejor.

A mis amigos Inoelkis, Maricet, Tallys, Anabel, Roider, Lianet, Yaily y Claudia, conocerlos fue lo mejor de estos cinco años, gracias por hacer que este tiempo en la universidad haya sido agradable.

A mi compañero de tesis por aguantar todas mis pesadeces y hacer que este trabajo fuera mucho más simple.

A mis tutores Cealys, Lytyet y Arnolis por guiarnos y ayudarnos en la realización de este trabajo, ya que sin su apoyo no hubiese sido posible.

A todos los profesores que de una forma u otra estuvieron ahí para brindarme sus conocimientos, especialmente a la profesora Yaniselis y el profesor Leonid.

A todos ustedes GRACIAS.

Rafael

A los seres que más quiero en el mundo:

A mis abuelos, por quererme tanto, estar siempre a mi lado y haberme regalado una niñez llena de amor y cariño.

A mis padres, corazones por los cuales vivo y ocupan todo mi cariño, que tanta veces han sacrificado sus sueños para que pueda cumplir los míos.

A mi hermanita por ser mi mayor tesoro y darme tanto cariño.

A todos ellos quienes justifican cada obra que realizo, cada tarea que emprendo y cada acción que concluyo, por ser luz y guías de sacrificio, dedicación, sufrimiento, consagración, sencillez y honestidad, de la cual siempre estaré orgulloso.

Rogelio

Quiero dedicar este trabajo especialmente a mi mamá, a mi tía Barby y a mi abuela Olga, por ser las tres personas más importantes de mi vida.

A mi papá, mi hermana Yeni y mi hermano Javier por ser muy especiales en mi mundo, los quiero con todo mi corazón.

A mi familia por creer en mí y nunca dudar en que podía lograrlo.

A mis tutores porque sin su guía la tesis no hubiese quedado de esta forma, espero que estén orgullosos del trabajo realizado por nosotros.

A todos ustedes les dedico esta tesis con mucho cariño.

RESUMEN

La Universidad de las Ciencias Informáticas está conformada por varios centros productivos, dentro de los cuales se incluye el Centro de Informatización de la Gestión de Entidades. En el mismo existen dificultades con la gestión de la información de sus recursos humanos, fundamentalmente para mantener la actualización, homologación y unificación necesaria en cada uno de los niveles organizativos, que contribuya a un mejor control y organización del personal relacionado con esta entidad y a la toma de decisiones.

El presente trabajo describe el análisis, diseño e implementación de un sistema para la gestión de la información de los recursos humanos en dicho centro, para lo cual se realizó un estudio de sistemas que satisfacen requisitos similares en otras entidades, concluyendo que era necesario construir una solución propia.

La nueva solución informática contribuye al cumplimiento de las políticas de migración hacia el software libre. La misma gestiona la información de forma correcta y más cercana a las necesidades reales del centro. La solución se validó con la aplicación de pruebas de calidad al software y la evidencia de aceptación por parte de los clientes.

Palabras claves: Recursos Humanos, Sistema de Gestión de Información.

ÍNDICE

INTRODUCCIÓN

La informática ha sido capaz de infiltrarse en todas las esferas de la vida y la gestión empresarial. La evolución de la ciencia, el desarrollo científico, técnico y tecnológico ha proporcionado sofisticados sistemas como los Sistemas de Gestión de Información (SGI). En medio de esa revolución informática probablemente todavía la mayoría de los ejecutivos y gerentes toman sus decisiones sin tener la información necesaria y ni siquiera reciben las informaciones internas de sus instituciones en la forma y el momento preciso, a pesar de contar en muchos casos con computadoras avanzadas (Encinosa, 2011). Los SGI permiten la gestión de la información de los recursos tanto internos como externos. Su finalidad es generar servicios y productos que respondan a las necesidades y sobrepasen las expectativas de los usuarios, posibilitando que el sistema trabaje eficiente y económicamente a la vez. El SGI aprovecha al máximo sus recursos de información en función de la mejora continua y de la toma de decisiones organizacional a todos los niveles jerárquicos, desde la cúspide estratégica hasta la base operativa (Ponjuán, 2000). Por lo cual es vital que una organización cuente con un Sistema de Información de RRHH que sea el soporte de cualquier toma de decisión referente a la gestión del personal y a la propia entidad. Una organización está conformada por varios elementos que juntos trabajan y se relacionan persiguiendo un mismo objetivo. Esta se encuentra formada por distintos subsistemas entre los cuales se encuentra el de recursos humanos (RRHH). La administración de los RRHH forma parte importante de la organización debido a que es el subsistema que se encarga de seleccionar, contratar, emplear, controlar y mantener a los empleados que formarán parte de la organización. También una de sus funciones implica controlar la relación que existe entre la organización y los empleados (Pescador, y otros, 2012). La gestión de los RRHH en Cuba, como parte del proceso de perfeccionamiento de la gestión empresarial, juega un papel esencial que se corresponde con las concepciones modernas de este campo, teniendo en cuenta que en temas de RRHH cada organización debería ajustar sus modelos a sus características propias (Cuesta, 2008).

Los RRHH son considerados el activo más significativo de cualquier organización, de ahí la importancia de que los datos asociados con estos sean lo más válido y reales posibles. Por ello el Centro de Informatización de la Gestión de Entidades (CEIGE) de la Universidad de las Ciencias Informáticas (UCI) se ha dado a la tarea de evaluar las potencialidades y necesidades reales de contar con un sistema informático que contribuya a gestionar la información de todos sus RRHH. Este centro con frecuencia se encuentra sometido a retos y presiones, a los cuales tienen que responder con alto grado de creatividad y

realismo. Esto se debe fundamentalmente a que para gestionar la información referente a las personas se utilizan varias bases de datos en formato excel, en función de las necesidades de las subdirecciones y áreas organizativas en general y los reportes que demandan los niveles superiores, lo que trae consigo que para los responsables de dominar y facilitar estas informaciones a quien las solicite sea muy difícil obtener la información de valor real. Se dificulta sobre todo porque deben pedir a cada una de las áreas que les retroalimenten y/o actualicen los datos, proceso en el que se cambian los formatos, se pierde información, se crean diferentes versiones en función de las necesidades del momento y luego se regresa a versiones anteriores provocando que no haya una actualización horizontal de cada elemento de manera sistemática y que los errores humanos introduzcan datos erróneos o modifiquen la información afectando su veracidad y con ello la imagen del control y la organización del centro.

Lograr resolver los problemas que presenta el centro en la gestión de RRHH es una necesidad inminente, para lo cual se requiere de una solución novedosa que permita la actualización, homologación y unificación de la información de los RRHH dentro del centro, facilitando la gestión de la misma con el desarrollo de una herramienta informática que la sustente.

Por lo cual se plantea como **problema a resolver** que: los procedimientos y herramientas utilizadas en el CEIGE para la gestión de la información de los RRHH del centro no permiten la actualización, homologación y unificación de la información necesaria en cada uno de los niveles donde se utiliza. El **objeto de estudio** se enmarca por tanto en los procesos y sistemas de gestión de información, delimitando como **campo de acción** la gestión de la información de los RRHH en el CEIGE, por lo cual se plantea como **idea a defender** que el desarrollo de un sistema para la gestión de la información de los recursos humanos en el centro CEIGE permitirá la actualización, homologación y unificación de la información necesaria en cada uno de los niveles.

Se persigue como **objetivo general:** Desarrollar un sistema para la gestión de la información de los RRHH para el CEIGE, de forma tal que permita la actualización, homologación y unificación de la información necesaria en cada uno de los niveles donde se utiliza.

Para dar cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos:**

- Realizar el marco teórico de la investigación para sustentar los conceptos y métodos de la propuesta.
- Analizar la gestión de los RRHH en Cuba, teniendo en cuenta el proceso de perfeccionamiento de la gestión empresarial.

- Analizar las soluciones informáticas disponibles para identificar los elementos (funcionales y no funcionales) que puedan reutilizarse.
- Desarrollar la solución para la gestión de la información de los RRHH del CEIGE.
- Validar la solución mediante pruebas de liberación internas y pruebas de aceptación con los clientes.

Para resolver el problema anteriormente planteado y dar cumplimiento a los objetivos trazados se proponen las siguientes **tareas de investigación**:

- Realización del estado del arte basado en los procesos y sistemas de gestión de información.
- Realización de un análisis orientado a seleccionar de las herramientas y lenguajes que propone el modelo de desarrollo del centro cuáles utilizar.
- Construcción del marco teórico referencial como resultado de la consulta de bibliografía nacional e internacional relacionada a la gestión de los RRHH.
- Realización del modelo conceptual.
- Realización de los diagramas de procesos del negocio teniendo en cuenta las reglas del negocio.
- Descripción de los procesos del negocio del sistema.
- Validación del modelado de negocio obtenido.
- Descripción de los requisitos obtenidos a partir de las técnicas utilizadas y partiendo de buenas prácticas y patrones del análisis.
- Definición de los prototipos de interfaz de usuario.
- Realización del diagrama de componentes.
- Realización de los diagramas de clases del diseño.
- Definición del modelo de datos.
- Descripción de las clases definidas en el diseño.
- Implementación de las interfaces de usuario para los procesos Selección e integración, Vacaciones y Evaluación.
- Implementación de las interfaces de usuario para el proceso Sanciones, Incidencia y Pre-nómina.
- Validación del sistema a través de pruebas de caja blanca.
- Validación del sistema a través de pruebas de liberación internas.
- Validación del sistema a través de pruebas de aceptación del cliente.

Los métodos teóricos utilizados para la realización de este trabajo son:

- Analítico-Sintético: este método es utilizado en el análisis de la bibliografía donde se resumen los elementos más importantes de los procesos y sistemas de gestión de información.
- Análisis Histórico-Lógico: este método es de suma importancia y es utilizado para realizar un profundo análisis de cómo han evolucionado los sistemas de gestión de información de los RRHH durante los años.

El método empírico utilizado para la realización de este trabajo es:

- Entrevista: mediante la aplicación de este método se llevaron a cabo entrevistas no formales con el cliente donde se obtuvo la mayor cantidad de información posible, además se pudo determinar las características de la propuesta de solución teniendo en cuenta las necesidades del cliente.

La estructura en capítulos será la siguiente:

CAPÍTULO 1: Fundamentación teórica: Describe el estado del arte de los sistemas que gestionan los RRHH, así como la fundamentación de las metodologías, lenguajes y herramientas a utilizar.

CAPÍTULO 2: Análisis y diseño: Detalla el análisis y el diseño del sistema, teniendo en cuenta la modelación de los procesos de negocio y la definición de requisitos de software según el modelo de desarrollo del CEIGE.

CAPÍTULO 3: Propuesta y validación de la solución: Describe la documentación de la implementación de la solución y se lleva a cabo la etapa de verificación y validación de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

En este capítulo se abordan un grupo de conceptos relacionados con la gestión de la información de los RRHH. Se realiza una investigación en base a sistemas que gestionen la información de los RRHH, además se analiza el modelo de desarrollo, las tecnologías y las herramientas a utilizar en el desarrollo del sistema.

1.2 Conceptos Generales

1.2.1 Sistema de gestión de información

Algunos autores como Davis y Olson (1985) conceptualizan los SGI como un sistema integrado y automatizado para proveer la información que sostenga las funciones de operatividad, gestión y toma de decisiones en una organización.

1.2.2 Sistema de información de Recursos Humanos

Un sistema de información de RRHH es un sistema empleado con el fin de recopilar, registrar, almacenar, analizar y recuperar datos relativos al personal de una organización (Heredero, y otros, 2006).

1.2.3 Vacaciones anuales pagadas

Las vacaciones anuales constituyen el periodo de interrupción de la prestación efectiva de servicios más largo que establece la legislación cubana a lo largo de cada año, a efectos que el trabajador pueda reponerse de la fatiga física y psíquica inherente al desarrollo de su actividad laboral. Es un periodo que concibe, también, para garantizar al trabajador un tiempo no sólo de reposo, sino de ocio, de liberación de cargas laborales y de posible incremento de la convivencia familiar, por lo que enlaza con la necesidad de promoción personal y protección de la familia inherente a cualquier sociedad que persiga el progreso social.

Todos los trabajadores tienen derecho al disfrute de un mes de vacaciones pagadas por cada once (11) meses de trabajo efectivo. El mes de vacaciones se considera de treinta (30) días naturales. El trabajador que por la índole de la actividad que desempeña u otras circunstancias, no labora once (11) meses, tiene derecho a vacaciones pagadas de duración proporcional al tiempo trabajado (Ley 49, 1984).

1.2.4 Evaluación del desempeño

La evaluación del desempeño es la actividad clave de la gestión de los RRHH consistente en un procedimiento que pretende valorar, de la forma más sistemática y objetiva posible, el rendimiento de los trabajadores en la organización. Se realiza sobre la base de: el trabajo desarrollado, los objetivos fijados,

las responsabilidades asumidas junto a las condiciones de trabajo y las características personales. Cualquiera que sea el método de evaluación que se asuma, el sistema (procedimiento) habrá de comprender esos elementos esenciales (Cuesta, 1999).

1.2.5 Pre-nómina

Documento que relaciona el tiempo correspondiente a ausencias, impuntualidades, vacaciones, licencias, subsidios, penalizaciones de trabajo, etcétera que incidan en deducciones del tiempo a devengar por cada trabajador (CEIGE, 2010).

1.3 Sistemas informáticos similares

1.3.1 OrangeHRM

- OrangeHRM está dirigido hacia empresas de pequeño y mediano tamaño para **gestión y administración del personal** de las mismas.
- Es un software de código libre.
- El programa se basa en una solución de Apache, PHP y MySql.
- El programa está organizado de forma modular, estos módulos son: Administración, Personal Information Manager (PIM) y Reportes.

Desde el módulo de Administración se podrá visualizar información de los otros módulos, generar reportes y visualizar a cada uno de los trabajadores de la organización. En el módulo PIM se podrá administrar el registro del personal de la empresa. Permite la introducción de las características: horas de trabajo, fecha de ingreso y datos personales. En el módulo de Reportes es donde se concentran todos los reportes requeridos y necesarios de los datos que se tienen registrados (Orange HRM, 2008).

Este sistema tiene como aspecto negativo la falta de documentación referente a su utilización, cuenta con una ayuda muy pequeña y confusa.

1.3.2 ASSETS NS

ASSETS NS es un sistema de gestión integral, una aplicación cliente-servidor programada en Visual Basic 6.0 y Microsoft SQL Server 2000. Es un sistema estándar y parametrizado que permite el control de los procesos de compras, ventas, producción, taller, inventario, finanzas, contabilidad, presupuesto, activos fijos, útiles y herramientas y RRHH.

El Módulo de Recursos Humanos (Versión 3.1 Access 97) está concebido para calcular las nóminas y controlar los recursos laborales de una entidad. Desde Recursos Humanos se pueden controlar íntegramente los recursos laborales: empleados, estructura organizativa de la entidad y plantilla. Siempre que se introducen altas, bajas y otros movimientos, se actualiza automáticamente el registro de

empleados y se generan los reportes correspondientes. Es posible modificar plantillas, introducir cambios en la estructura organizativa, crear nuevos cargos y realizar conversiones de plazas. Una vez calculadas y pagadas las nóminas, permite generar automáticamente los comprobantes de operaciones a la contabilidad (ASSETS, 2011).

Como aspecto negativo de este software se puede mencionar que está enfocado al trabajo con la nómina, no es multiplataforma y de código abierto, así como que no garantiza la independencia tecnológica debido a que está desarrollado en tecnologías privativas.

1.3.3 VERSAT SARASOLA

Versat Sarasola, es un sistema de gestión contable-financiero cubano, el programa tiene diferentes subsistemas, fue desarrollado en Delphi y como base de datos usa SQL.

Características del VERSAT SARASOLA:

- Herramienta para la planificación económica, el control y el análisis de gestión.
- Diseñado para su empleo en cualquier tipo de entidad empresarial y presupuestada.
- Se estructura en un grupo de subsistemas en los cuales se procesan y contabilizan los documentos primarios, donde se anotan los movimientos, los recursos materiales, laborales y financieros que se utilizan en una entidad.
- Se logra establecer un proceso de interacción usuario-sistema.
- Rapidez y fiabilidad, a partir de la configuración del proceso de contabilización de los documentos primarios y de las propias posibilidades de trabajo contenidas en cada subsistema (Sosa, y otros, 2008) .

Aspectos negativos:

- Es una aplicación de escritorio.
- Está desarrollado en Delphi que es una tecnología privativa.

1.3.4 Rodas XXI

Sistema multiempresa creado por CITMATEL para la automatización de la gestión empresarial. Contiene diferentes módulos que pueden usarse integrados o independientes: contabilidad, activos fijos, nóminas, inventarios, facturación, finanzas, RRHH y telecobranzas.

Sus módulos pueden ser empleados e integrados en su totalidad, y se puede formar cualquier subconjunto entre ellos, o cada uno de forma independiente. Además, trabaja con doble moneda.

El módulo de RRHH permite el control, gestión y planificación de los RRHH en general con la actualización de los datos del trabajador incluyendo su foto, así como exportar las incidencias y los datos

de los trabajadores para el módulo de Nóminas. En este módulo es donde se define la ficha del trabajador; entre los datos que se recogen se encuentran: foto, datos personales, datos del cónyuge, datos de los hijos, datos laborales, datos de las organizaciones a que pertenece y otros datos (Rodas XXI, 2002).

RODAS XXI ha sido diseñado para trabajar en los sistemas operativos de Microsoft: Windows XP 2000 ó 2003. No es un sistema multiplataforma.

1.3.5 Sistema Integral de Gestión Cedrux

El primer sistema integral de gestión de entidades presupuestadas Cedrux está compuesto por diversos módulos, dentro de los cueles se encuentra el subsistema de capital humano.

El subsistema de capital humano de Cedrux está basado en las condiciones de las entidades cubanas, su objetivo inmediato es proporcionar la información necesaria para la gestión más básica de información del personal y el pago a los trabajadores. Como objetivo final se espera realizar la gestión integral del capital humano basada en las competencias laborales.

Se liberó su versión 1.1 del módulo, con un total de 5 macro procesos, entre los que se encuentran:

- Organización del trabajo
- Selección e integración
- Administración de Capital Humano
- Estimulación moral y material
- Incluido el proceso de Configuración

Se encuentran en elaboración:

- Competencias laborales
- Evaluación del desempeño
- Seguridad y salud en el trabajo
- Sistema de cuadros
- Capacitación y desarrollo

Las proyecciones futuras del módulo hasta el momento son:

- Autocontrol
- Comunicación institucional

Resultados del estudio de los sistemas informáticos

Después del estudio de los sistemas anteriores se reflejan en la Tabla 1 las características que estos deben cumplir para ser utilizados. Las características deseadas aparecen por las filas y por las columnas los sistemas analizados.

Sistemas Informáticos / Características	OrangeHRM	ASSETS NS	Versat Sarasola	Sistema Integral de Gestión Cedrux	Rodas XXI
Software accesible.	X	X	✓	✓	✓
Utilización de software libre	✓	X	X	✓	X
Cumplimiento con el paradigma de independencia tecnológica del país.	✓	X	X	✓	X
Ejecución de todas las funcionalidades necesarias.	X	X	X	X	X
Aplicación Web	✓	✓	X	✓	X

Tabla 1: Resultados del estudio de los sistemas informáticos

✓ si cumple X. no cumple

Luego de la realización del estudio y el análisis de las características de los SGI ya existentes se puede concluir que ninguno cumple con todas las características deseadas, destacando que el Sistema Integral de Gestión Cedrux cumple con la mayoría de dichas características, excepto por ejecución de todas las funcionalidades deseadas por el centro. Por lo que sería factible un sistema que responda a las particularidades del centro y sea consecuente con el paradigma de independencia tecnológica por el cual apuesta el país, para lo cual sería imprescindible la utilización de tecnologías y herramientas libres para el desarrollo de la solución.

1.4 Modelo de desarrollo

El CEIGE cuenta con un modelo estandarizado (*Modelo de desarrollo de software para el CEIGE*), que establece las distintas disciplinas por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas, con la incorporación de los distintos subprocesos dictados por el *Nivel II de CMMI* (Capability Maturity Model Integration)[1].

1.4.1 Descripción general de la disciplina Modelado del negocio y artefactos a generar

Es la disciplina destinada a comprender los procesos de negocio de la organización y cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito.

Los artefactos que se generan en esta disciplina son:

- Modelo conceptual

[1] Certificación obtenida por el centro en julio de 2011 y reconocida por el SEI (Software Engineering Institute) como aval de la calidad de su proceso de desarrollo de software.

- Mapa de procesos de Negocio.
- Descripción de procesos de negocio (CEIGE, 2012)

1.4.2 Descripción general de la disciplina Requisitos y artefactos a generar

El esfuerzo principal en la disciplina de Requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de artefactos que describen todas las interacciones que tendrán los usuarios con el software y que responden a los requisitos funcionales del sistema. Se especifican los requisitos funcionales y no funcionales.

Los artefactos que se generan en esta disciplina son:

- Especificación de requisitos de software
- Descripción de requisitos (CEIGE, 2012)

1.4.3 Descripción general de la disciplina Análisis y diseño y artefactos a generar

En esta disciplina es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima disciplina. Los artefactos generados en esta etapa son más formales y específicos de una implementación. En caso de llevarse a cabo la reutilización de componentes de software ya desarrollados, durante esta disciplina se ajusta el modelado existente a los requisitos actuales.

Los artefactos que se generan en esta disciplina son:

- Diagrama de clases del diseño por componentes
- Diagrama de componentes
- Diseño de casos de prueba
- Modelo de datos

1.4.4 Descripción general de la disciplina Implementación y artefactos a generar

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes de software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes. En este paso se muestra toda la implementación del sistema.

1.4.5 Descripción general de la disciplina Pruebas internas y artefactos a generar

En esta disciplina se desarrollan las pruebas del grupo de calidad del centro, verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple (CEIGE, 2012).

- No conformidades

- Acta de liberación

1.5 Herramientas y tecnologías

1.5.1. Herramientas CASE

Las Herramientas CASE son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software.

En la actualidad existen una gran cantidad de herramientas CASE, entre las más utilizadas se encuentran Platinun Erwin, EasyCASE, Oracle Designer, System Architect y Visual Paradigm para UML. Para la realización de la propuesta de solución del trabajo se va a utilizar Visual Paradigm para UML pues este soporta los dos lenguajes de modelado que se utilizarán, los cuales son UML y BPMN (Zaragoza, 2008).

1.5.1.1. Visual Paradigm 8.0

Soporta el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (Pressman, 2002).

1.5.2. Lenguajes de modelación

1.5.2.1. 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. Es un lenguaje de propósito general para el modelado orientado a objetos. También es un lenguaje de modelado visual que permite una abstracción del sistema y sus componentes (Prf. Luza, 2008).

1.5.2.2. BPMN

BPMN (Business Process Modeling Notation) es un estándar de modelado de procesos de negocio, en donde se presentan gráficamente las diferentes etapas del proceso del mismo. Ha sido diseñado específicamente para coordinar la secuencia de procesos y los mensajes que fluyen entre estos. BPMN y las extensiones de UML ayudan a modelar la situación actual y deseada en los procesos de negocio. BPMN ha sido desarrollado para proveer a los usuarios de una notación de uso libre (MILESTONE Consulting, 2012).

1.5.3. Lenguajes de programación

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

1.5.3.1. PHP

PHP (Hypertext Processor) es un lenguaje script, para el desarrollo de páginas web dinámicas del lado del servidor, es de código abierto, el más popular y extendido en la web.

Entre las características que posee están:

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

1.5.3.2. 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 (Asleson, 2008).

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 reducen de manera significativa los tiempos de carga inicial.

DOM

El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento.

JSON

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

CSS

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

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.
- JavaScript no requiere un tiempo de compilación; ya que los scripts se pueden desarrollar en un período de tiempo relativamente corto.

- Es independiente de la plataforma hardware o sistema operativo, y funciona correctamente siempre y cuando exista un navegador con soporte JavaScript.

1.5.4. Framework

Para seleccionar el framework a utilizar se analizaron varias opciones, fundamentalmente symfony y zend. En aras de darle cumplimiento a la soberanía tecnológica y a la migración del software libre propuesta por el país y poder minimizar el tiempo de desarrollo del sistema se decidió utilizar el framework symfony. Debido a que posee una documentación de gran calidad así como una gran comunidad Open Source muy activa y profesional que lo respalda, integra algunas de las mejores ideas y herramientas para el desarrollo en PHP, una capa ORM como Propel o Doctrine haciendo que las tablas de la base de datos estén disponibles como objetos en el código. Basa su arquitectura en el Modelo Vista Controlador, además de éstas ventajas, el framework symfony se puede extender fácilmente con código propio, o bien con funciones, código de otros frameworks y librerías. De esta forma los desarrolladores ganan tiempo en el proceso de desarrollo de la aplicación y consiguen un código más fácil de mantener. Tiene una integración con las librerías javascript más populares (jQuery, Prototype, Scriptaculous, YUI, entre otras), las cuales incluyen una serie de funciones AJAX, así como su avanzado sistema de caché que puede integrarse con otros sistemas de caché existentes. Symfony genera código orientado a objetos para las funcionalidades más comunes del manejo de bases de datos, dando como resultado uno de los frameworks más estables y robustos (Blasco, 2009).

1.5.4.1. Symfony 1.2.8

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web basado en el patrón Modelo Vista Controlador. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Symfony fue diseñado para ajustarse a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.
- Su capa de abstracción, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza programación orientada a objetos.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.

- Una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo (Symfony.es, 2012).
- Usando como ORM[2] Doctrine:

Doctrine es un mapeador de objetos-relacional (ORM) escrito en PHP que proporciona una capa de persistencia para objetos PHP. Es una librería muy completa y muy configurable. Puede generar clases a partir de una base de datos existente y después el programador puede especificar relaciones y añadir funcionalidad extra a las clases autogeneradas.

1.5.4.2. ExtJS 3.4

ExtJS 3.4 es una biblioteca o conjunto de librerías de JavaScript para el desarrollo de aplicaciones web interactivas, usa tecnologías AJAX, DHTML y DOM. ExtJS permite realizar completas interfaces de usuario, fáciles de usar, muy parecidas a las conocidas aplicaciones de escritorio. Esto permite a los desarrolladores web concentrarse en la funcionalidad de las aplicaciones en vez de en las advertencias técnicas (BAIRESIT, 2010).

1.5.5. Ambiente de desarrollo integrado (IDE)

Un entorno de desarrollo integrado o Integrated Development Environment (IDE), en inglés, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Algunos ejemplos de IDE de desarrollo son Eclipse, NetBeans IDE, Geany y CodeRun (Maldonado, 2007). Para la realización de la propuesta de solución se utilizará el NetBeans debido a que es multiplataforma y open-source y tiene una mayor estabilidad sobre otros IDE.

1.5.5.1. Netbeans IDE 7.0.1

Es un reconocido entorno de desarrollo integrado, disponible para Windows, Mac, Linux y Solaris. El proyecto Netbeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++ (NetBeans, 2010).

1.5.6. Herramientas de base de datos

[2]ORM (Object Relational Mapping): es una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional utilizado en el desarrollo de nuestra aplicación.

Existen numerosas herramientas que se relacionan directamente con servidores de bases de datos. Estas aplicaciones de orden avanzado ofrecen la oportunidad de administrar completamente los servidores basados en lenguaje de datos. Algunos ejemplos de ellas son PGAcces, TOra, SQuirreL, eRWin y pgAdmin III la cual se estará utilizando.

1.5.6.1. PgAdmin 3

Es una potente herramienta de código abierto y multiplataforma para la administración de bases de datos. La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

1.5.7. Sistema Gestor de Base de Datos (SGBD)

Los sistemas de gestión de bases de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan también organizan los datos con un impacto mínimo en el código de los programas. Algunos ejemplos de SGDB son: MySQL, Microsoft SQL Server y postgreSQL, el cual se estará utilizando por las ventajas que brinda (Bertino, y otros, 1995).

1.5.7.1. PostgreSQL

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS), de código abierto. Este sistema aporta potencia y flexibilidad adicional, 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). Brinda un control de concurrencia multi-versión (MVCC por sus siglas en inglés) que permite trabajar con grandes volúmenes de datos (Bertino, y otros, 1995).

1.5.8. Servidor Web

Un servidor web es un programa informático que procesa una aplicación del lado del servidor realizando conexiones bidireccionales y unidireccionales y síncronas o asíncronas con el cliente generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente (Farfán, 2012).

1.5.8.1. Apache 2.2

Entre sus principales características se encuentran:

- Corre en una gran cantidad de Sistemas Operativos, que lo hace prácticamente universal.
- 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.

- Permite a los administradores elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor.

1.5.9. Navegadores

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

1.5.9.1. Mozilla Firefox 18.0.1

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. Además protege al usuario de la publicidad de ventanas emergentes no solicitadas.

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

1.5.10. Control de versiones

Una versión, revisión o edición de un producto, es el estado en el que se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Existen muchos sistemas que se utilizan para el control, algunos de estos son: Bazaar 2.5.1, Git 1.8.0, Monotone 1.0 y el que se estará utilizando RapidSVN.

1.5.10.1. RapidSVN

RapidSVN es un cliente de control de versiones multiplataforma que se distribuye bajo la Licencia Pública General de GNU y puede ser usado para administrar cualquier conjunto de ficheros.

Características:

- **Simple:** proporciona una interfaz fácil de usar para las características del cliente gráfico.
- **Eficiente:** simple para los principiantes pero lo suficientemente flexible como para aumentar la

productividad para los usuarios con experiencia.

1.6. Patrones de diseño

Los patrones de diseño brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Proporcionan una estructura conocida por todos los programadores, de manera que la forma de trabajar no resulte distinta entre los mismos. La utilización de los mismos, permite ahorrar grandes cantidades de tiempo en la construcción de software. El software construido es más fácil de comprender, mantener y extender, reduce los esfuerzos de desarrollo y mantenimiento, mejora la seguridad informática, eficiencia y consistencia del diseño. Proporciona considerables ahorros en la inversión y además mejoran (aumentan, elevan) la flexibilidad, modularidad y extensibilidad, factores internos e íntimamente relacionados con la calidad percibida por el usuario (Larman, 1999). Dentro de los patrones de diseño se encuentran los patrones Gof y GRASP.

Patrones Gof

Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns—Elements of Reusable Software” de Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides.

Estos patrones se clasifican según su propósito en: creacionales, estructurales y de comportamiento.

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es un acrónimo que significa (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns, Patrones Generales de Software para Asignación de Responsabilidades). El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

Existen nueve patrones GRASP los cuales son: Experto, Creador, Alta cohesión, Bajo acoplamiento, Controlador, Polimorfismo, Fabricación pura, Indirección y No hables con extraños.

1.7. Patrones de arquitectura

El patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC), fue diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales están dadas por el hecho de que, el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

- El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.
- La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa preferentemente con el Controlador, pero es posible que trate directamente con el Modelo a través de una referencia al propio Modelo.
- El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo, centra toda la interacción entre la Vista y el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

1.8. Conclusiones parciales del capítulo

En este capítulo fueron descritos conceptos y definiciones encargados de trasmitir las especificidades del negocio sobre el que se desarrollará el sistema. Fueron también analizadas las funcionalidades y características de otros sistemas de gestión ya existentes, llegando a la conclusión de que el sistema integral de gestión CEDRUX cumple con la mayoría de los requisitos necesarios para la gestión de los RRHH en el CEIGE, pero brinda funcionalidades incompletas a los requisitos deseados, por lo que se desarrollará un sistema propio para el centro tomando como base el sistema integral de gestión CEDRUX. También fueron descritos y se dieron argumentos de por qué se seleccionaron las tecnologías, metodologías y herramientas con las cuales se va a desarrollar el trabajo en cuestión, conformando la fundamentación teórica que sustenta el desarrollo del trabajo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO.

2.1. Introducción

En el presente capítulo se realiza un estudio preliminar del sistema, se especifican los patrones de arquitectura y del diseño, se obtienen los artefactos: diagramas de clases del diseño que permite una visión más clara de la implementación del sistema, los diagramas de componentes y despliegue con la descripción de cada uno de ellos, el modelo de datos donde se representan las tablas vinculadas a la propuesta de solución y se modelará una propuesta de solución a partir de los requisitos previamente definidos.

2.2. Diseño del sistema

El diseño es un modelo del sistema o producto que se va a construir, debe ser suficiente para que la implementación de dicho sistema se realice sin ambigüedades, encontrando la forma para que soporte todos los requisitos y restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis que proporciona una descripción detallada de los requisitos (Martínez, y otros, 2010).

2.2.1. Patrones de diseño utilizados

Para contribuir a que el sistema sea más flexible y robusto se utilizaron en la elaboración del diseño los siguientes patrones GRASP:

Creador: Las acciones definidas para cada módulo se encuentran en la clase action.class.php, Table y Form. Las acciones contienen toda la lógica de la aplicación, además se crean los objetos de las clases que representan las entidades y los objetos de otras clases que intervienen en la lógica del módulo.

Alta Cohesión: En cada clase Actions se definen las acciones para las plantillas, además estas colaboran con otras para realizar diferentes operaciones, se instancian objetos, se acceden a las properties, es decir en una Actions se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un software flexible ante los cambios.

Experto: Este es uno de los más utilizados, puesto que se estará utilizando doctrine como ORM el mismo realiza una capa de abstracción en el modelo, encapsula toda la lógica de los datos y son generadas las clases con todas las funcionalidades comunes de las entidades.

Además se utilizaron los siguientes patrones GoF:

Patrón estructural:

Decorator (Envoltorio): Añade funcionalidades a una clase dinámicamente. En cada archivo layout.php se define el código html común de cada vista, evitando que este sea repetido en cada página.

Patrones creacionales:

Singleton (Instancia Única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En la acción, se usan los métodos ->getRequest(),->getUser(), esto se debe a que , en la acción, el método ->getContext(), guarda una referencia a todos los objetos del núcleo de symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, solo varía la forma de llamarlos.

2.2.2. Patrones de Arquitectura

Los diagramas de clase realizados para el diseño, están basados en el patrón arquitectónico modelo vista controlador (MVC), el cual separa en tres capas diferentes el sistema: la interfaz, la lógica de negocio y los datos. Las páginas indexSuccess.*php* y las clases nombreVista.*js* representan la vista, las clases *action.class.php* el controlador y en el modelo se encuentran; dentro de model (Entidad.class.php, EntidadTable.class.php, BaseEntidad.class.php) y dentro de form (EntidadForm.class.php, BaseEntidadForm.class.php); logrando que cualquier cambio que se realice en la vista no afecte ni la lógica del negocio, ni el dominio y viceversa. A continuación se muestra la Figura 1 que refleja cómo están distribuidas dichas clases.

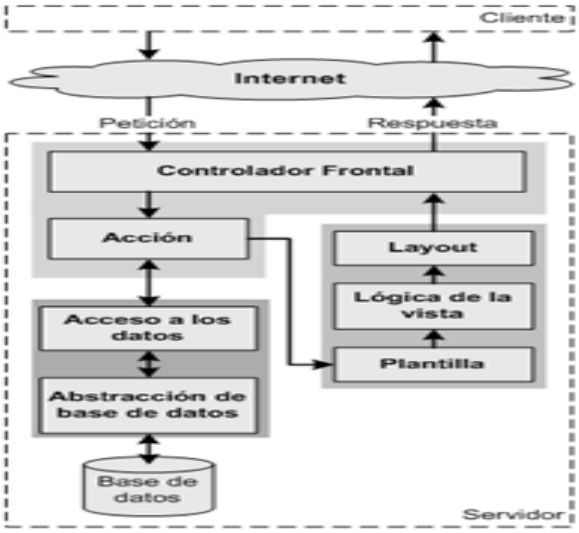

Figura 1: Flujo de trabajo en Symfony mediante MVC (jotadeveloper, 2009)

2.2.3. Diagrama de procesos de negocio

Un proceso de negocio es una colección de actividades que tomando una o varias clases de entradas crean una salida que tiene valor para un cliente. Los procesos de negocio representan el flujo de trabajo y

de información a través del negocio (Ruiz, 2011). En el diagrama de procesos de negocio se representan cada uno de los procesos que componen un sistema así como sus relaciones principales.

En la Figura 2 se muestran los procesos que componen el sistema de RRHH y las relaciones entre ellos.

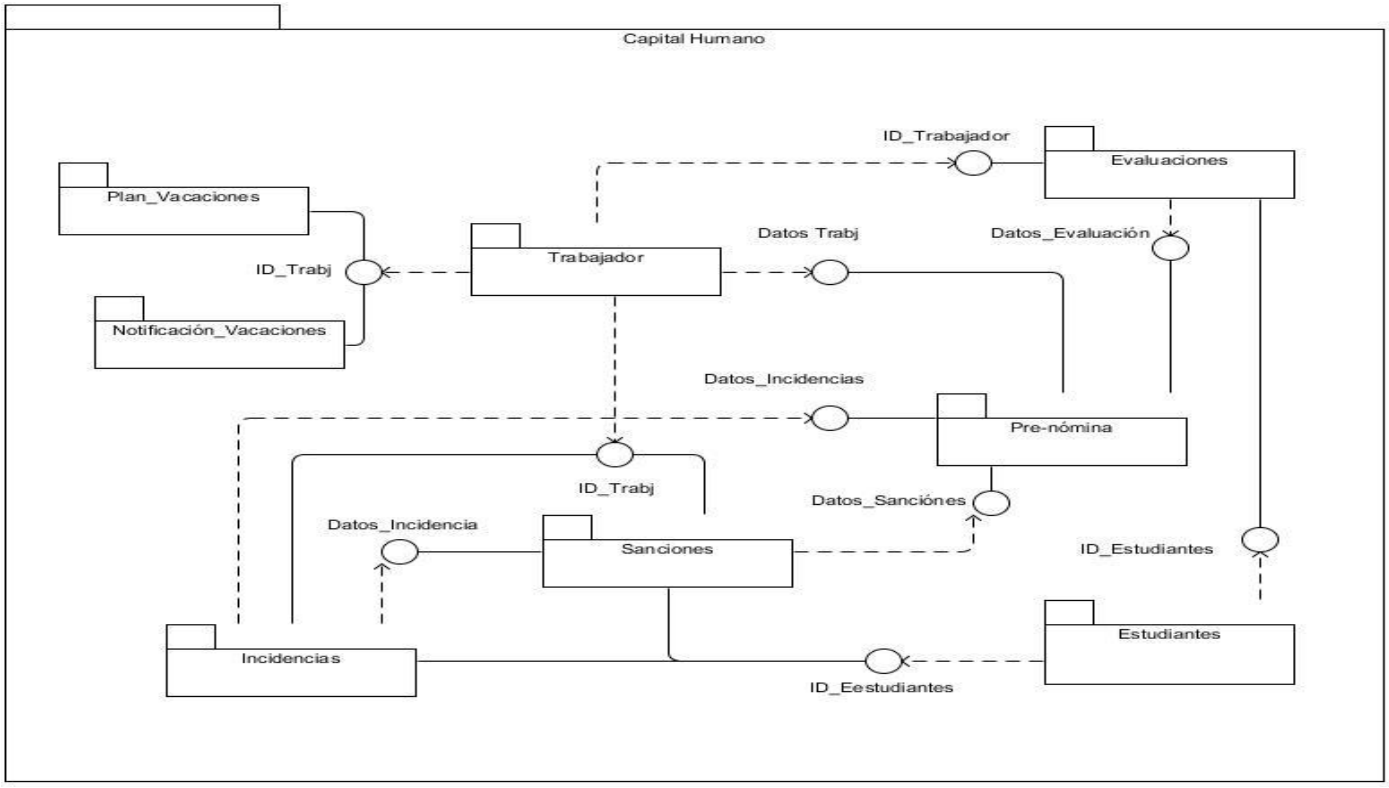

Figura 2: Diagrama de procesos de negocio

2.2.4. Diagrama de descripción del proceso de negocio Dar Evaluación

A continuación se muestra en la Figura 3 el diagrama de descripción del proceso de negocio Dar Evaluación en el cual el técnico general solicita la evaluación de las personas a cada uno de los departamentos, estos buscan la evaluación de cada uno de las personas de su departamento y se la envía al técnico y luego este lo actualiza en el excel de RRHH. Los demás diagramas de descripción de procesos de negocio se encuentran desde el Anexo 1 hasta el Anexo 7.

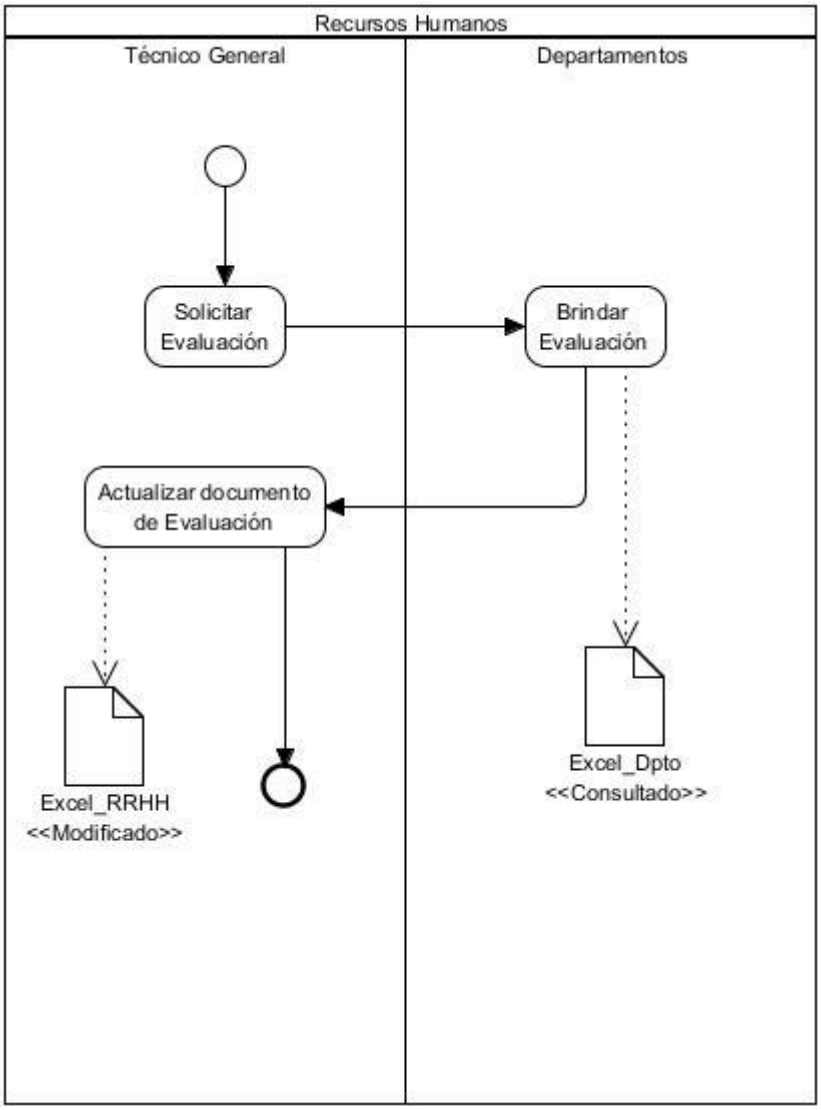

Figura 3: Diagrama de descripción del proceso de negocio Dar Evaluación

2.2.5. Descripción del proceso Dar Evaluación

La Tabla 2 describe el proceso Dar Evaluación.

Objetivo	Dar evaluación.
Evento(s) que lo genera(n)	NA
Pre condiciones	Halla al menos una persona para evaluar
Marco legal	• Resolución No. 21 del 2007 del MTSS "Evaluación del desempeño de los trabajadores" • Resolución No. 8 del 2005 del MTSS "Reglamento general sobre Relaciones

	laborales". Capítulo X • Decreto No. 281 del 2007, "Reglamento para la implantación y consolidación del Sistema de Dirección y Gestión empresarial estatal". • Normas Cubanas NC 3000-2007 "Sistema de gestión integrada de capital humano"
Clientes internos	Técnico general
Clientes externos	NA
Entradas	NA
Flujo de eventos	
Flujo básico Control del expediente laboral.	
1. El técnico general pide la evaluación de las personas a los departamentos. 2. Los departamentos buscan la evaluación de cada una de las personas y se la envían al técnico. 3. El técnico actualiza los datos en el excel de capital humano.	
Pos-condiciones	
1. NA.	
Salidas	
Evaluación	
Flujos paralelos	
NA	
Pos-condiciones	
NA	
Salidas	
NA	
Flujos alternos	
2.a NA	
1 NA	
Pos-condiciones	
NA	
Salidas	
NA	
1.	

Tabla 2: Descripción del proceso evaluación

2.2.6. Modelo conceptual

El modelo conceptual es una especificación del dominio del problema a través de la representación mediante objetos. Le permite a los desarrolladores simplificar de manera notable algunos aspectos de las fases de modelación iniciales, aprovechar el trabajo y las experiencias previas y acelerar por vías probadas el proceso total de elaboración de software (Delgado, y otros, 2002).

En la Figura 4 se muestran los conceptos entre los que se encuentra “persona” la cual es el RRHH con que cuenta el centro y esta puede ser un trabajador o un estudiante, la persona está ubicada en un proyecto, y la misma puede cometer una incidencia por la cual puede ser sancionada y por lo mismo tener una baja evaluación, si la persona es un trabajador tiene derecho a vacaciones y estará ubicado en una pre-nómina. En el Anexo 8 se muestra el modelo conceptual íntegramente.

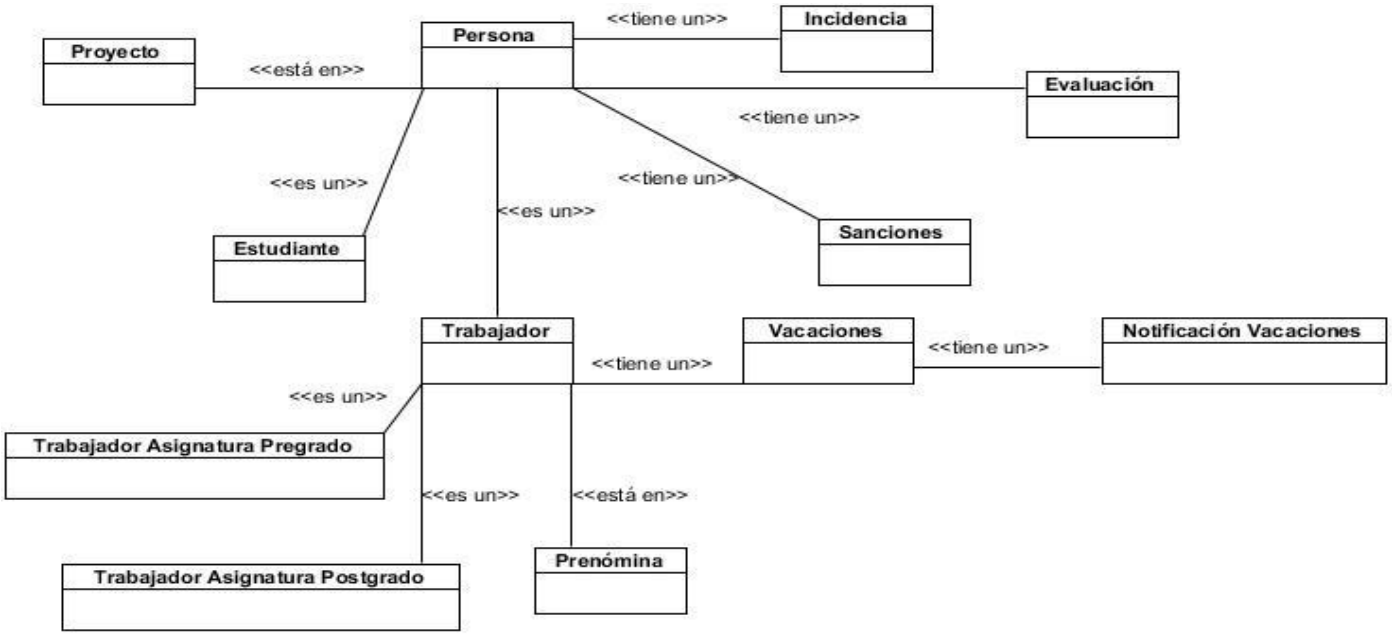

Figura 4: Modelo conceptual

2.2.7. Especificación de requisitos del sistema

El objetivo principal de la Especificación de Requisitos del Sistema (ERS) es servir como medio de comunicación entre clientes, usuarios, ingenieros de requisitos y desarrolladores. En la ERS deben recogerse tanto las necesidades de clientes y usuarios (necesidades del negocio) como los requisitos que debe cumplir el sistema a desarrollar para satisfacer dichas necesidades (requisitos del producto, también conocidos como requisitos de sistema o requisitos de software) (JuntAndalucia, 2009).

Para la obtención de los requisitos funcionales, se utilizaron las técnicas de captura de requisitos: entrevista con los clientes y análisis de soluciones existentes. Con la utilización de estas técnicas se obtuvieron 21 agrupaciones de requisitos para un total de 75 requisitos funcionales, a continuación se muestra la Tabla 3 con la selección de los mismos:

- **Requisitos Funcionales**

1. Gestionar datos del estudiante	Adicionar datos del estudiante

	Modificar datos del estudiante
	Eliminar datos del estudiante
2. Gestionar datos del trabajador	Adicionar datos del trabajador
	Modificar datos del trabajador
	Eliminar datos del trabajador
3. Gestionar datos educacionales del estudiante	Adicionar datos educacionales del estudiante
	Modificar datos educacionales del estudiante
	Eliminar datos educacionales del estudiante
4. Gestionar datos educacionales del trabajador	Adicionar datos educacionales del trabajador
	Modificar datos educacionales del trabajador
	Eliminar datos educacionales del trabajador
5. Gestionar datos personales del estudiante	Adicionar datos personales del estudiante
	Modificar datos personales del estudiante
	Eliminar datos personales del estudiante
6. Gestionar datos personales del trabajador	Adicionar datos personales del trabajador
	Modificar datos personales del trabajador
	Eliminar datos personales del trabajador
7. Gestionar evaluación de las personas	Adicionar evaluación de las personas
	Modificar evaluación de las personas
	Eliminar evaluación de las personas
	Listar evaluación de las personas
8. Gestionar localización del estudiante	Adicionar localización del estudiante
	Modificar localización del estudiante
	Eliminar localización del estudiante
9. Gestionar localización del trabajador	Adicionar localización del trabajador
	Modificar localización del trabajador
	Eliminar localización del trabajador
10. Gestionar otros datos del trabajador	Adicionar otros datos del trabajador
	Modificar otros datos del trabajador

	Eliminar otros datos del trabajador
11. Gestionar incidencias de las personas	Adicionar incidencias de las personas
	Modificar incidencias de las personas
	Eliminar incidencias de las personas
	Listar incidencias de las personas
12. Gestionar vinculación a organización	Adicionar vinculación a organización
	Modificar vinculación a organización
	Eliminar vinculación a organización
	Listar vinculación a organización
13. Gestionar plan de vacaciones	Adicionar plan de vacaciones
	Eliminar plan de vacaciones
	Listar plan de vacaciones
14. Gestionar pre-nómina	Adicionar pre-nómina
	Modificar pre-nómina
	Eliminar pre-nómina
	Listar pre-nómina
15. Gestionar vinculación a proyecto	Adicionar vinculación a proyecto
	Modificar vinculación a proyecto
	Eliminar vinculación a proyecto
	Listar vinculación a proyecto
16. Gestionar sanciones de las personas	Adicionar sanciones de las personas
	Modificar sanciones de las personas
	Eliminar sanciones de las personas
	Listar sanciones
17. Gestionar superación del trabajador	Adicionar superación del trabajador
	Modificar superación del trabajador
	Eliminar superación del trabajador
18. Gestionar trabajo metodológico del trabajador	Adicionar trabajo metodológico del trabajador
	Modificar trabajo metodológico del trabajador

	Eliminar trabajo metodológico del trabajador
19. Gestionar vinculación asignatura de postgrado	Adicionar vinculación asignatura de postgrado
	Modificar vinculación asignatura de postgrado
	Eliminar vinculación asignatura de postgrado
	Listar vinculación asignatura de postgrado
20. Gestionar vinculación asignatura de pregrado	Adicionar vinculación asignatura de pregrado
	Modificar vinculación asignatura de pregrado
	Eliminar vinculación asignatura de pregrado
	Listar vinculación asignatura de pregrado
21. Gestionar notificación de vacaciones	Adicionar notificación de vacaciones
	Modificar notificación de vacaciones
	Eliminar notificación de vacaciones
	Listar notificación de vacaciones
22. Asignar superación al trabajador	
23. Listar estudiante	
24. Listar trabajador	
25. Llenar información de la pre-nómina	

Tabla 3: Requisitos funcionales

- **Descripción del requisito funcional Gestionar evaluación**

Las Tablas 4, 5, 6 y 7 describen al requisito funcional Gestionar evaluación.

- **Ejemplo textual de Adicionar evaluación**

Precondiciones	Se ha registrado al menos un trabajador en el sistema.
Flujo de eventos	
Flujo básico Realizar evaluación	
1. Se introducen los datos de la evaluación: Tipo de evaluación: según la persona es el tipo de evaluación Resultado: resultado que tiene en la evaluación Fecha de inicio: fecha en que inicia la evaluación Fecha de fin: fecha en que termina la evaluación	
2. El sistema valida (ver validación 1) los datos introducidos.	
3. Si los datos son correctos el sistema los registra.	
4. El sistema confirma el registro de los datos.	
5. Concluye el requisito.	
Pos-condiciones	

1. Se registró en el sistema una nueva evaluación.		
Flujos alternativos		
Flujo alternativo Información errónea		
1. El sistema señala los datos erróneos y permite corregirlos.		
2. El usuario corrige los datos.		
3. Volver al paso 2 del flujo básico.		
Pos-condiciones		
1. N/A		
Validaciones		
1. Se validan los datos según lo establecido en el Modelo Conceptual RRHH		
Relaciones	**Requisitos Incluidos**	NA
	Extensiones	N/A
Conceptos	**Evaluación**	Visibles en la interfaz: Tipo de evaluación Resultado Fecha de inicio Fecha de fin
Requisitos especiales	NA	
Asuntos pendientes	NA	

Tabla 4: Descripción textual del requisito Adicionar Evaluación

- **Ejemplo textual de Modificar evaluación**

Precondiciones	Se ha registrado al menos una evaluación en el sistema.
Flujo de eventos	
Flujo básico Modificar evaluación	
1. Se modifican los datos de la evaluación: Tipo de evaluación Resultado Fecha de inicio Fecha de fin	
2. El sistema valida (ver validación 1) los datos introducidos.	
3. Si los datos son correctos el sistema los registra	
4. El sistema confirma el registro de los datos.	
5. Concluye el requisito.	
Pos-condiciones	
1. Se modificó en el sistema una nueva evaluación.	
Flujos alternativos	
Flujo alternativo Información errónea	
1. El sistema señala los datos erróneos y permite corregirlos.	
2. El usuario corrige los datos.	
3. Volver al paso 2 del flujo básico.	

Pos-condiciones		
1. N/A		
Validaciones		
1. Se validan los datos según lo establecido en el Modelo Conceptual RRHH		
Relaciones	**Requisitos Incluidos**	Buscar trabajador.
	Extensiones	N/A
Conceptos	**Proceso evaluativo**	Visibles en la interfaz: Tipo de evaluación Resultado Fecha de inicio Fecha de fin
Requisitos especiales	NA	
Asuntos pendientes	NA	

Tabla 5: Descripción textual del requisito Modificar evaluación

- **Ejemplo textual de Eliminar evaluación**

Precondiciones	Se ha registrado al menos una evaluación en el sistema
Flujo de eventos	
Flujo básico Eliminar Evaluación	
1. El trabajador autorizado selecciona la evaluación a eliminar.	
2. El sistema verifica (ver validación 1) que se pueda eliminar la evaluación.	
3. Se solicita confirmación para eliminar la evaluación.	
4. Si el usuario confirma se elimina la evaluación.	
5. Concluye el requisito.	
Pos-condiciones	
1. Se eliminó la evaluación.	
Flujos alternativos	
Flujo alternativo 4.a No se puede eliminar la planilla.	
1. El sistema notifica por qué no puede eliminarse la evaluación.	
Pos-condiciones	
1. N/A	
Flujo alternativo *.a El usuario cancela la acción	
1. Concluye el requisito.	
Pos-condiciones	
1. No se elimina la planilla.	
Validaciones	
1. Se valida que no se haya aprobado la evaluación.	
Relaciones	**Requisitos Incluidos** N/A
	Extensiones N/A

Conceptos	**Proceso evaluativo**	Visibles en la interfaz: Tipo de evaluación Resultado Fecha de inicio Fecha de fin
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 6: Descripción textual del requisito Eliminar evaluación

- **Ejemplo textual de Listar evaluación**

Precondiciones	Se ha registrado al menos una evaluación en el sistema	
Flujo de eventos		
Flujo básico		
1. El sistema muestra un listado de las evaluaciones registradas Se muestra en un listado: Tipo de evaluación Resultado Fecha de inicio Fecha de fin		
2. Concluye el requisito.		
Pos-condiciones		
1. N/A		
Flujos alternativos		
Flujo alternative		
1. N/A		
Pos-condiciones		
1. N/A		
Validaciones		
1. N/A		
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Indicadores**	Visibles en la interfaz: Tipo de evaluación Resultado Fecha de inicio Fecha de fin
Requisitos especiales	N/A	
Asuntos	N/A	

pendientes

Tabla 7: Descripción textual del requisito Listar evaluación

- **Requisitos no funcionales**

La Tabla 8 muestra los requisitos no funcionales.

Clasificación	Requisitos no funcionales	Descripción
1. Funcionalidad	**Seguridad**	▪ Todo uso de las funcionalidades del sistema requiere la autenticación de los usuarios. ▪ El sistema concederá acceso a cada usuario autenticado sólo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema. ▪ El sistema manejará mecanismos de encriptación para las contraseñas de los usuarios.
	Madurez	▪ El sistema no permitirá la entrada de datos incorrectos. ▪ El sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario. ▪ Ninguna información que se haya ingresado en el sistema y se haya asociado a alguna operación será eliminada físicamente de la base de datos.
	Tolerancia ante fallo	▪ Ante el fallo de una funcionalidad del sistema, el resto de las funcionalidades que

		no dependen de esta deberán seguir funcionando. ▪ El sistema permite detectar fallos internos y notificar al usuario de la ocurrencia de estos.
2. Usabilidad	**Comprensibilidad**	▪ Todos los mensajes de error del sistema deberán incluir una descripción textual del error. ▪ Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.
	Atracción	▪ El sistema diferenciará los mensajes de información de los mensajes de error y de advertencia valiéndose de distintos íconos para cada tipo. ▪ El sistema presentará los términos capitalizados, es decir, la primera palabra tendrá su primera letra en mayúsculas. ▪ La tipografía y colores serán estándares en toda la aplicación.
3. Eficiencia	**Rendimiento**	▪ El sistema no excederá los 3 segundos de respuesta al efectuar acciones de cargar un registro (esta cifra no incluye

		los retardos por concepto de tráfico de red). ▪ El sistema no excede los 2 s para efectuar acciones de salva de información (esta cifra no incluye los retardos por concepto de tráfico de red).
4. Mantenibilidad	**Flexibilidad**	▪ El sistema permitirá agregar nuevas funcionalidades o modificar alguna existente sin romper la estructura y consistencia de los componentes.
5. Contrastabilidad	**Adaptabilidad**	▪ El sistema podrá ser visualizado en todos los navegadores.
	Instalabilidad	▪ El sistema podrá ser instalado en el ambiente especificado en los requisitos tecnológicos para servidores.
	Coexistencia	▪ El sistema interactuará con herramientas ofimáticas y visualizador de ficheros PDF para la presentación de los reportes que genere.

Tabla 8: Requisitos no funcionales

2.2.8. Diagrama de componentes

Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema dividida en componentes y el comportamiento del servicio que estos proporcionan y utilizan a través de las interfaces.

En la Figura 5 se muestra el diagrama de componentes que representa los componentes existentes en el sistema y las relaciones entre ellos, con el objetivo de obtener información de los mismos.

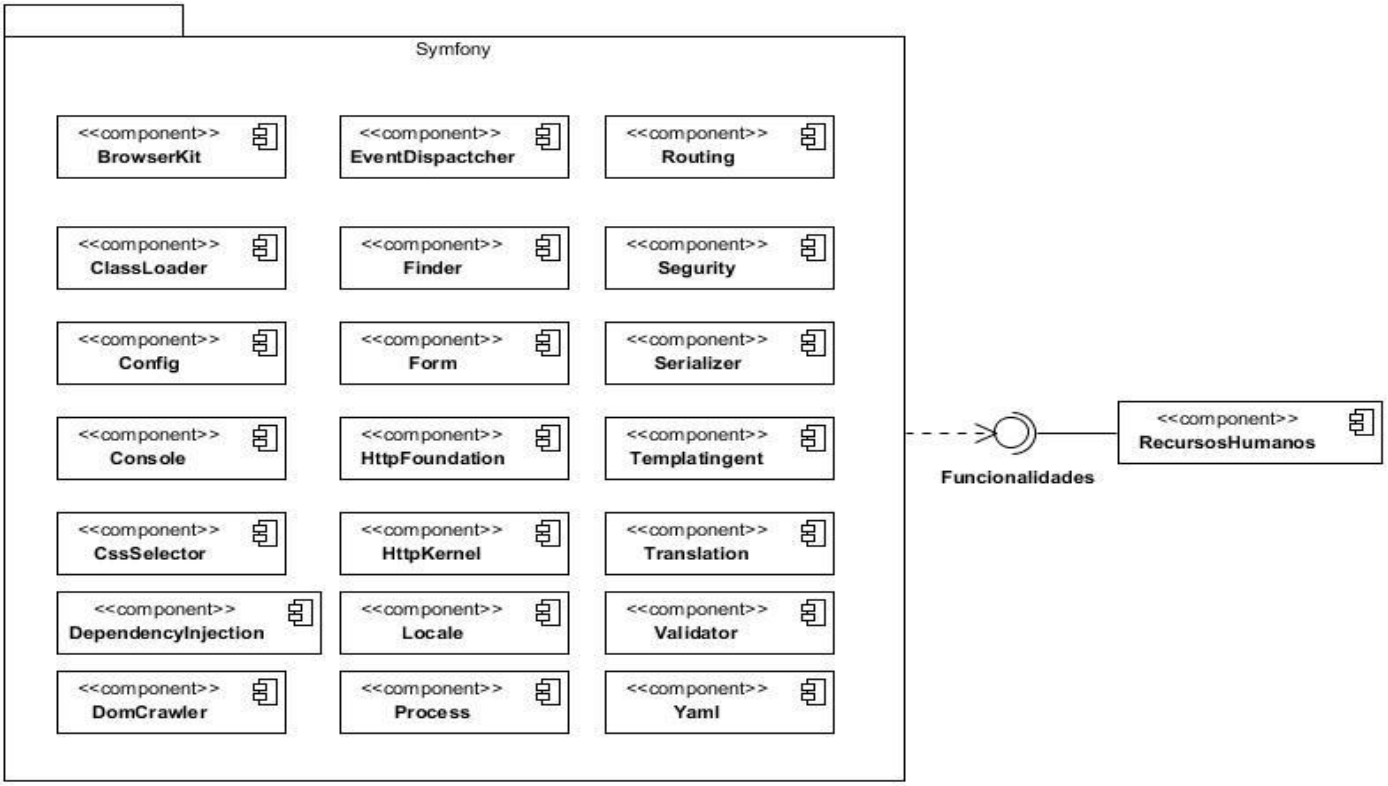

Figura 5: Diagrama de componentes

2.2.9. Diagramas de clases del diseño por componente

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y las interfaces en una aplicación. Es el diagrama principal de análisis y diseño para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones. La Figura 6 muestra el diagrama de clases del diseño para Evaluación, el resto de los diagramas aparecen desde el Anexo 9 hasta el Anexo 14.

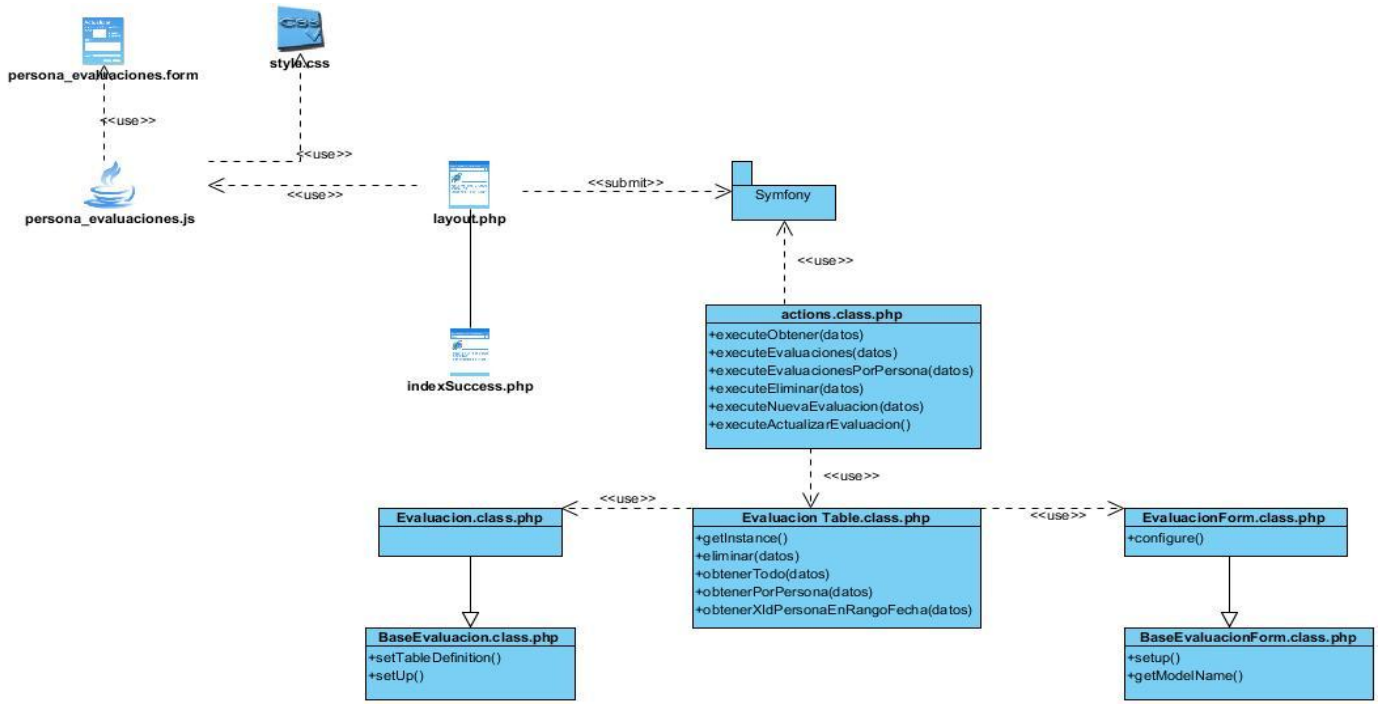

Figura 6: Diagrama de clase del diseño por componente

2.2.10. Modelo de datos

Un modelo de datos es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos, a la cual se denomina esquema. Según el nivel de abstracción, el modelo de datos que permite su descripción será un modelo externo, global o interno, cada uno de los cuales ofrece distintos elementos de descripción.

El modelo de datos tiene como objetivo, identificar y representar las entidades de importancia para el funcionamiento del negocio, sus atributos y la forma en que estas entidades se comunican entre sí (relaciones). Este modelo se desarrolla para facilitar el diseño de la base de datos y mostrar los datos que contendrá el sistema.

El modelo de datos propuesto en la solución cuenta con un total de 63 tablas, de ellas 5 son las encargadas de seguridad, 36 son nomencladores encargados de gestionar conceptos específicos del negocio ya predefinidos, ejemplo: la tabla “nom_Rol_Comisión” gestiona los tipos de roles definidos en las comisiones disciplinarias.

Las 22 tablas restantes se encargan de gestionar los datos que son necesarios tener registrados, por ejemplo, la tabla “Estudiante” almacena los principales datos relacionados con los estudiantes como son el grupo al que pertenece y el año en que se encuentra. Otro ejemplo es la tabla “Evaluación” encargada de

guardar la fecha en que se realiza la evaluación y qué tipo de evaluación se va a realizar en dependencia de la persona que vaya a ser evaluada. En “Sanción” se registra el tipo de violación cometida y si se apela o no. La Figura 7 representa el modelo de datos del sistema.

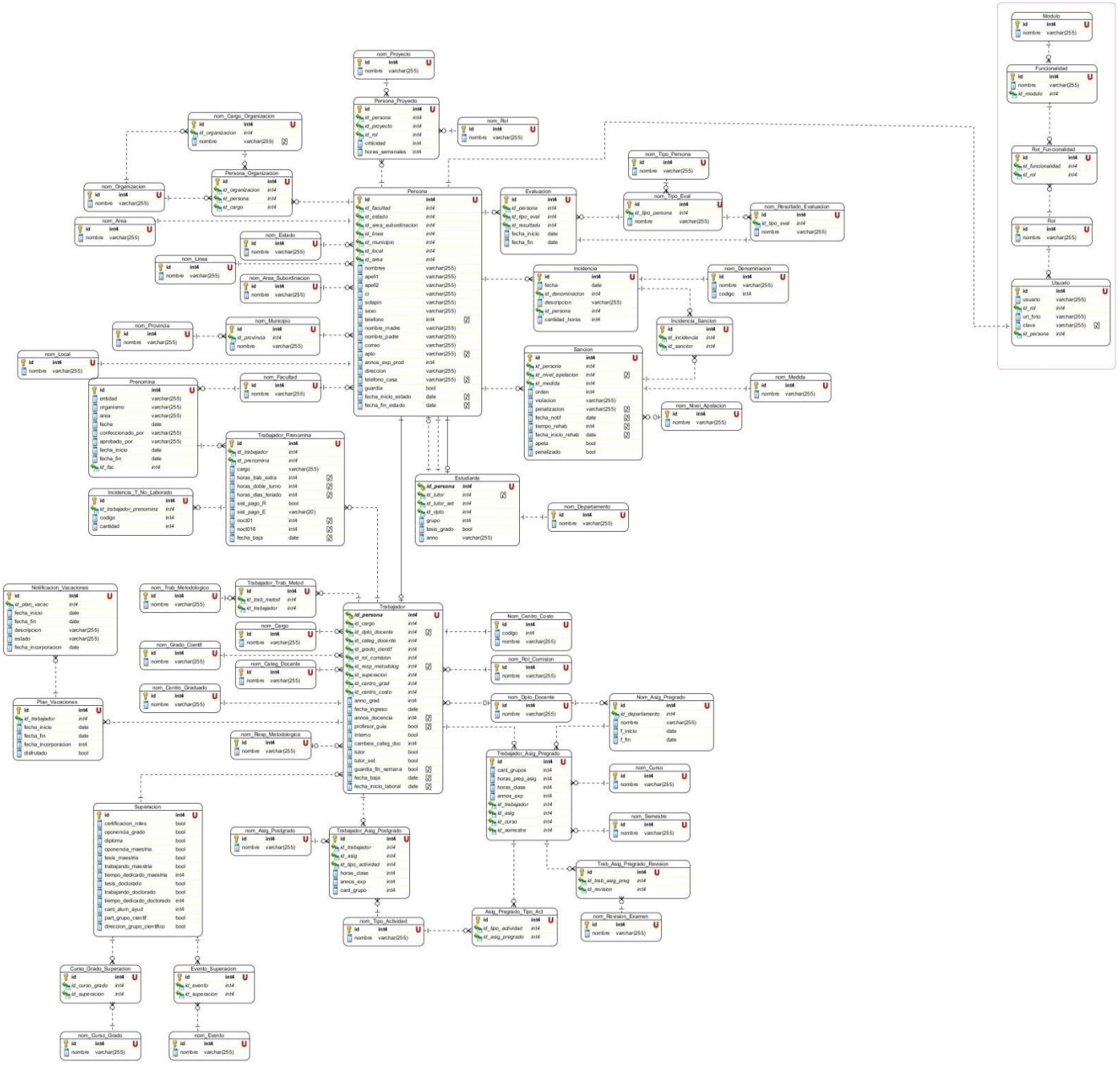

Figura 7: Modelo de datos

2.3. Conclusiones parciales del capítulo

En el presente capítulo se diseñó el mapa de proceso, los diagramas de procesos de negocio y la descripción de los mismos. Se realizó la modelación del negocio para identificar los procesos que intervienen en el mismo, para luego definir y describir los requisitos. Se realizó el modelo conceptual con el fin de identificar las entidades del negocio. Además se logró modelar el diseño a partir de la representación gráfica de los diagramas de clase del diseño por componente lo que permitió que se pudiera describir cómo quedará la estructura del sistema, también se realizó el modelo de datos para percibir cómo persisten los datos en la base de datos.

CAPÍTULO 3: PROPUESTA Y VALIDACIÓN DE LA SOLUCIÓN.

3.1. Introducción

En el presente capítulo se presenta la propuesta desarrollada, además se realiza la validación del diseño a través de métricas con el objetivo de verificar el cumplimiento de los atributos de calidad, también se describen las pruebas efectuadas al software con el objetivo de detectar y corregir el máximo de errores en el sistema.

3.2. Estructura del sistema

Dentro de la carpeta raíz del sistema se encuentran definidas las carpetas apps y web, ver Figura 8, que contienen la lógica de negocio y las vistas del sistema.

Figura 8: Contenido de las carpetas apps y web

La Figura 9 muestra la estructura de la carpeta apps:

Figura 9: Estructura de la carpeta apps

Backend y frontend: incluye las carpetas de configuración, la carpeta template, las librerías con las cuales se trabaja y la carpeta model donde se encuentran los módulos del sistema.

La Figura 10 muestra la estructura de la carpeta web:

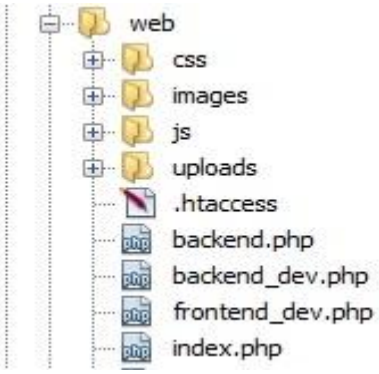

Figura 10: Estructura de la carpeta web

css: En esta se encuentran las plantillas y estilos para el diseño del sistema.

js: Es la carpeta donde se incluyen las clases *javascript* y los ficheros con la extensión *.js* donde se encuentra el código correspondiente a la capa de presentación.

3.2.1. Servicios utilizados

Un servicio es cualquier objeto PHP que realiza algún tipo de tarea global. Cada servicio se puede utilizar en cualquier parte de la aplicación. La ventaja de pensar en servicios es que te obliga a separar cada funcionalidad de tu aplicación en una serie de servicios. Como cada servicio se limita a una tarea específica, se puede acceder fácilmente a cada servicio y usar su funcionalidad siempre que se necesite (Potencier, y otros, 2010).

3.2.2. Servicios que consume

Los servicios consumidos por la aplicación son:

- LDAP (Figura 11): Se obtienen determinados datos de estudiantes y trabajadores a través del usuario.

```
$dn_ldap = 'cn=ad search, ou=Systems, ou=UCI Domain Impersonals, dc=uci, dc=cu';
$clave_ldap = 'uF2SODWAHiW0eJboFFQEAvVzJ';
$server_ldap = '10.0.0.3';

$ds = ldap_connect($server_ldap);
ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3);
ldap_set_option($ds, LDAP_OPT_REFERRALS, 0);
$resultado = ldap_bind($ds, $dn_ldap, $clave_ldap);

$valor = $t->Nombres . ' ' . $t->Apellidos;
$dn = "DC=uci,DC=cu";
$filter = "(|(displayname=$valor*)(sn=$valor*))";

$sr = ldap_search($ds, $dn, $filter);
$result = ldap_get_entries($ds, $sr);
```

Figura 11: Código del servicio LDAP

- ASSET (Figura 12): Se obtienen los datos de los trabajadores a partir del carnet de identidad y el solapín.

```
$servicio = new SoapClient('http://assets.uci.cu/servicios/v4/AssetsWS.wsdl');
if ($criterio == 'ci') {
    $t = $servicio->ObtenerPersonaDadoCIdentidad($valor);
} else {
    $t = $servicio->ObtenerPersonaDadoIdExpediente($valor);
}
```

Figura 12: Código del servicio ASSET

3.2.3. Diagrama de despliegue

El diagrama de despliegue muestra la configuración en funcionamiento del sistema incluyendo el software y hardware, estos representan a los nodos y sus relaciones. A continuación se muestran los posibles escenarios con los requisitos de software.

Servidores

Servidor de Aplicaciones Web

- Sistema Operativo: Ubuntu Server
- Servidor Web: Apache 2.2
- Librerías Adicionales: PHP 5

Servidor de Base de Datos

- Sistema Operativo: Ubuntu Server
- Sistema Gestor de Base de Datos: PostgreSQL 9.1

Servidor de Clientes Ligeros

- Sistema Operativo: Nova Server
- Navegador Web: Mozilla Firefox 2.17 ó superior
- Herramientas Ofimáticas
- Visualizador de ficheros pdf
- Herramienta de administración de clientes ligeros

Clientes

PC Cliente con disco duro

- Sistema Operativo: Linux o Windows
- Navegador Web: Mozilla Firefox 2.17 ó superior
- Herramientas Ofimáticas
- Visualizador de ficheros pdf

PC Cliente sin disco duro

- Todo se instala en el servidor de clientes ligeros

Las Figuras 13 y 14 muestran los diagramas de despliegue tanto para los clientes con disco o sin disco:

Figura 13: Diagrama de despliegue de escenario para PC cliente con disco

Figura 14: Diagrama de despliegue de escenario para PC cliente sin disco

3.2.4. Funcionalidades implementadas

En la Figura 15 se muestra la interfaz Gestionar Trabajador que ofrece las opciones de Adicionar, Modificar y Eliminar (Trabajador, Incidencia, Sanciones, Evaluaciones, Plan de Vacaciones, Notificación de Vacaciones, Asignaturas de Pregrado y Postgrado), Vincular a Organización y Proyecto, Seleccionar los Cursos y Eventos en los que participa y realizar Búsquedas.

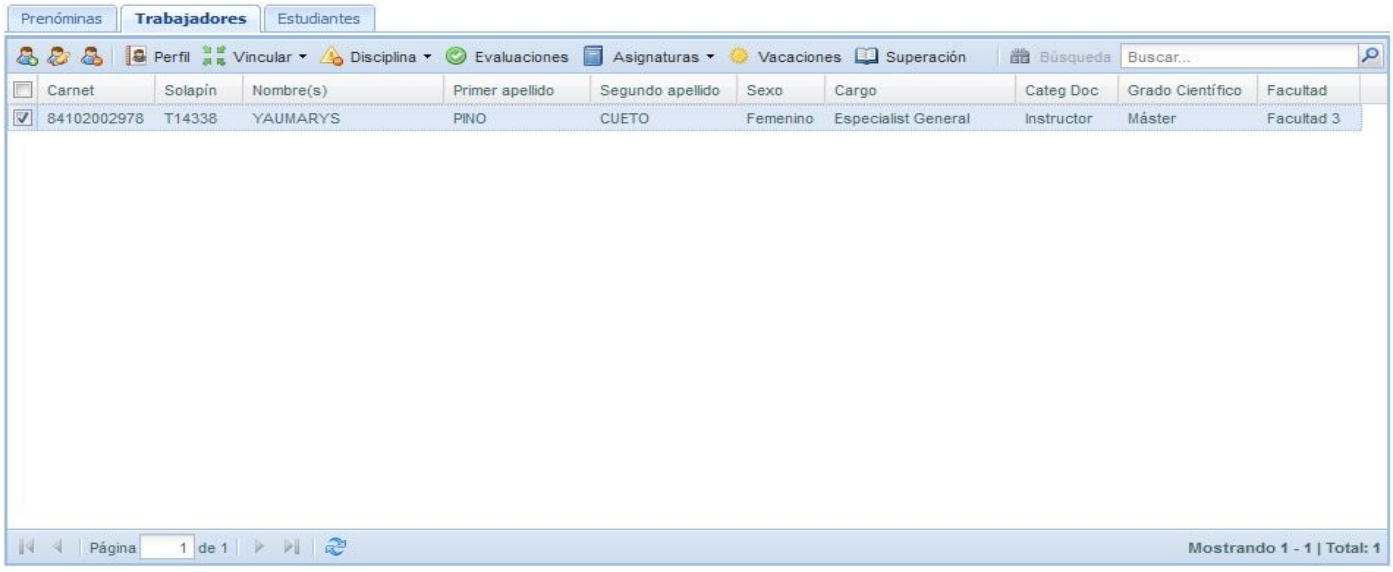

Figura 15: Interfaz Gestionar Trabajador

En la Figura 16 se muestra la interfaz Búsqueda avanzada de trabajadores que ofrece la opción de buscar a partir de varios atributos.

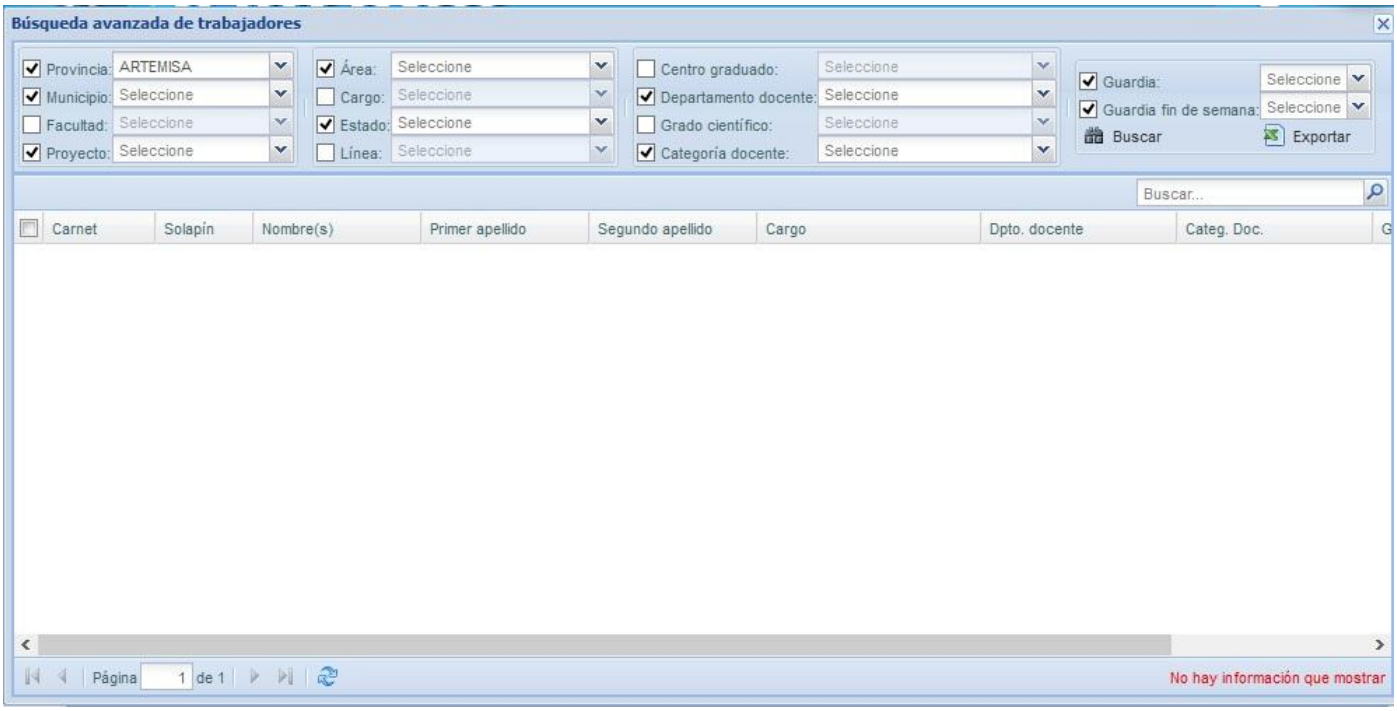

Figura 16: Búsqueda avanzada de trabajadores

En la Figura 17 se muestra la interfaz Gestionar Pre-nómina la cual incluye las opciones de Adicionar, Modificar y Eliminar, Exportar (Excel y PDF), ver Trabajadores que conforman la pre-nómina, así como registrar las horas de (Trabajo extra, Doble Turno, Día Feriado, Nocturnidad) y Buscar.

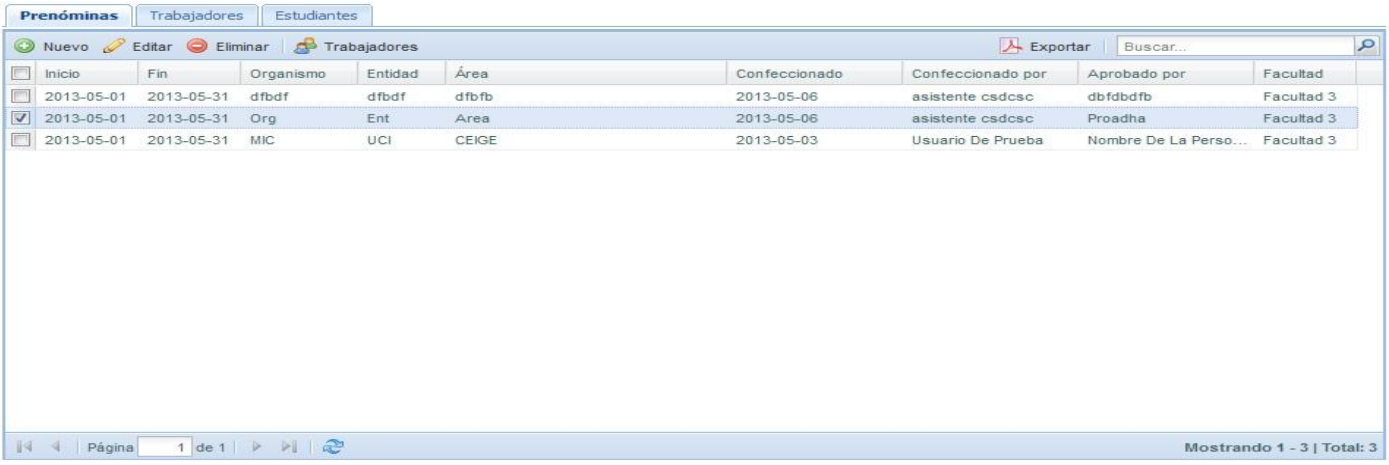

Figura 17: Interfaz Gestionar Pre-nómina

3.3. Validación del modelo de diseño propuesto

Con el objetivo de comprobar cuán bien están definidas las clases, se emplean las métricas Relaciones entre clases y Tamaño operacional de clase, diseñadas para evaluar los siguientes atributos de calidad:

- **Responsabilidad:** consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.
- **Complejidad de implementación**: consiste en el grado de dificultad que se implementa un diseño de clases determinado.
- **Reutilización**: consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- **Acoplamiento**: consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- **Complejidad del mantenimiento**: consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software.
- **Cantidad de pruebas**: consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases) diseñado (Echemendia, y otros, 2012).

3.3.1. Métrica Tamaño Operacional de Clase

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase.

A continuación se muestran una serie de tablas encaminadas a un mejor entendimiento de la utilización de esta métrica.

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

Tabla 9: Métrica Tamaño Operacional de Clase

Atributo	Categoría	Criterio

Responsabilidad	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Complejidad de Implementación	Baja	<= Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<=Promedio

Tabla 10: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

Se le aplicó la métrica a un total de 327 clases. Para determinar la categoría en que se encuentran cada uno de los atributos, se calcula el promedio de la columna cantidad de procedimientos (el cual fue de 4,504587156) y este es el que se utiliza en la Tabla 10 en la columna criterio.

Las gráficas que se muestran a continuación (Figuras 18, 19, 20 y 21) reflejan los resultados obtenidos para cada uno de los atributos medidos.

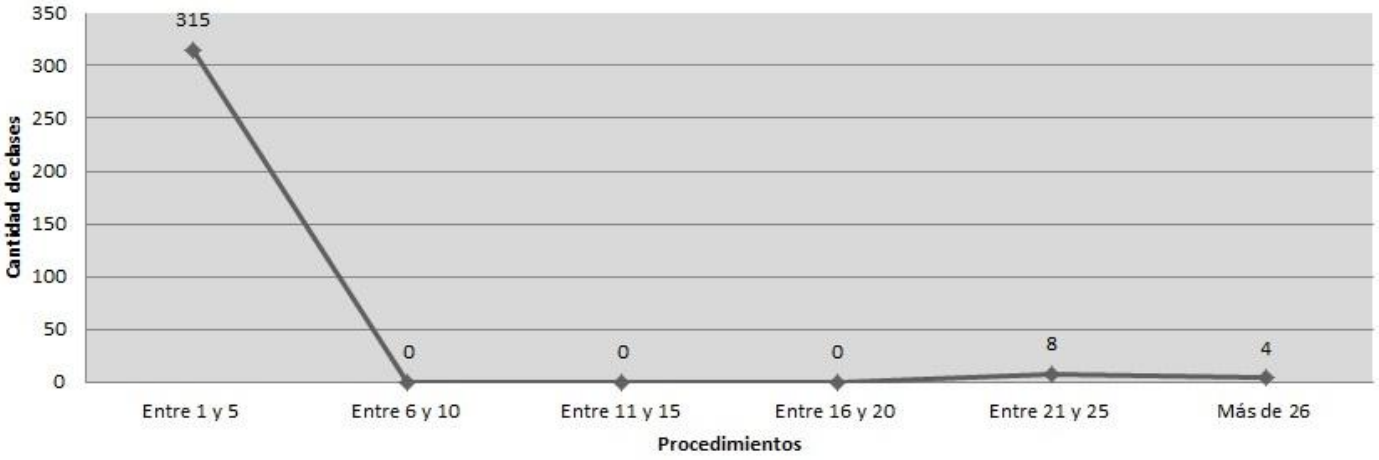

Figura 18: Resultados obtenidos de la aplicación de la métrica TOC en el instrumento agrupados en los intervalos definidos

Figura 19: Representación en % de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

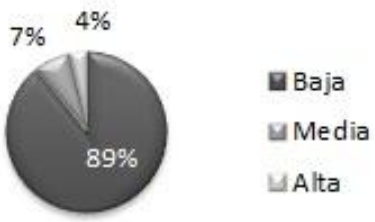

Figura 20: Representación en % de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de implementación

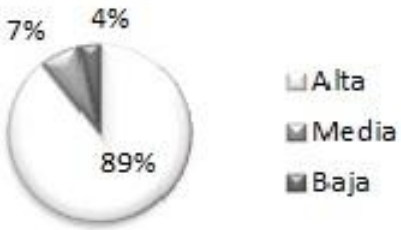

Figura 21: Representación en % de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos para los atributos de la métrica TOC en la evaluación del instrumento, se puede observar que la mayoría de las clases que conforman el sistema para los atributos responsabilidad y complejidad están dentro de la categoría baja para un 89% del total, mientras que el atributo reutilización cuenta con igual por ciento en la categoría alta mostrando así que el sistema cuenta con una elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

3.3.2. Métrica Relaciones entre Clases

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otras.

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

Tabla 11: Relaciones entre Clases

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

Tabla 12: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica RC

Se le aplicó la métrica a un total de 327 clases. Para determinar la categoría en que se encuentran cada uno de los atributos, se calcula el promedio de la columna cantidad de relaciones de uso (el cual fue de 2,220183486) y este es el que se utiliza en la tabla 13 en la columna criterio.

Las gráficas que se muestran a continuación (Figura 22 hasta Figura 26) reflejan los resultados obtenidos para cada atributo medido.

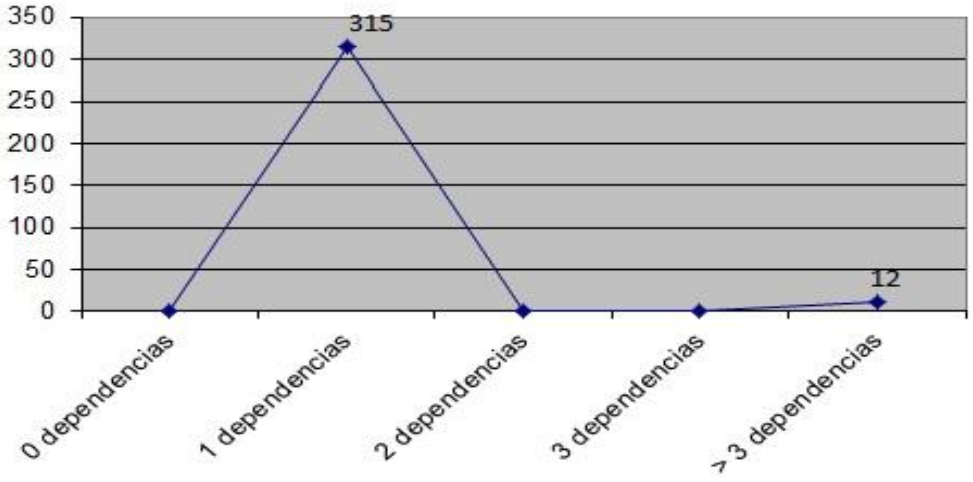

Figura 22: Resultados obtenidos de la aplicación de la métrica RC en el instrumento agrupados en los intervalos definidos

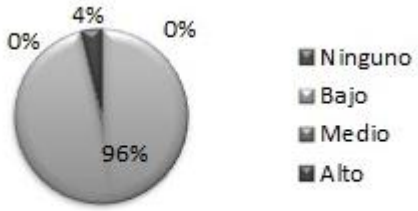

Figura 23: Representación en % de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

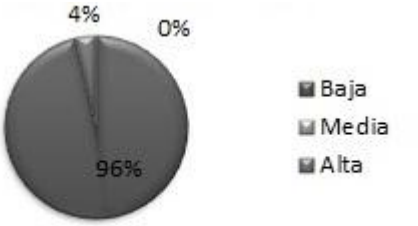

Figura 24: Representación en % de los resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento

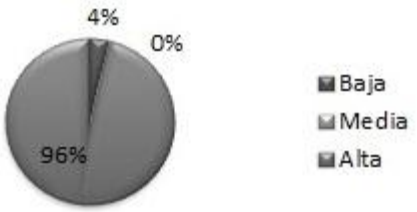

Figura 25: Representación en % de los resultados de la evaluación de la métrica RC en el atributo Reutilización

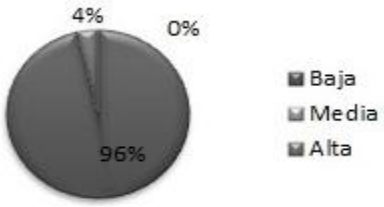

Figura 26: Representación en % de los resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que las clases del diseño poseen un bajo acoplamiento, bajo mantenimiento y baja cantidad de pruebas, ya que para estos atributos la categoría bajo sumó un 96% del total, mostrando igual por ciento en la categoría alta del atributo reutilización. Lo que demuestra que no es necesario un elevado esfuerzo en el momento de realizar cambios, rectificaciones y pruebas al software.

3.4. Pruebas de Software

3.4.1. Pruebas de Caja Blanca o Estructurales

La prueba de Caja Blanca, es un método de diseño que usa la estructura de control descrita como parte del diseño al nivel de componentes para derivar los casos de prueba. Con los métodos de prueba de Caja Blanca se podrán derivar casos de prueba que:

1. Garanticen que todas las rutas independientes dentro del módulo se han ejercitado por lo menos una vez.
2. Ejerciten los lados verdaderos y falsos de todas las decisiones lógicas.
3. Ejecuten todos los bucles en sus límites y dentro de sus límites operacionales.
4. Ejerciten estructuras de datos internos para asegurar su validez (Pressman, 2002).

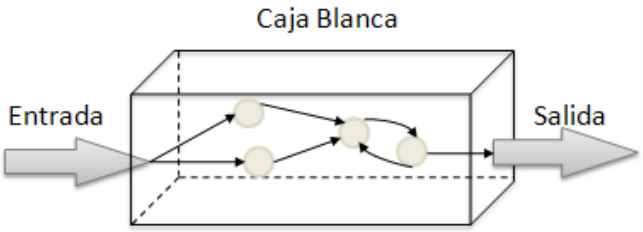

Figura 27: Representación de técnica de pruebas de Caja Blanca

A continuación se citan algunas de las técnicas de prueba de Caja Blanca:

- Condición.
- Flujo de Datos.
- Bucles.
- Camino Básico.

Dentro de la prueba de Caja Blanca, la técnica que se utilizó fue Camino Básico. Para aplicar esta técnica se debe introducir la notación para la representación del flujo de control, este puede representarse por un grafo de flujo en el cual:

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

Para construir el grafo se debe tener en cuenta la notación para las instrucciones (Figura 28).

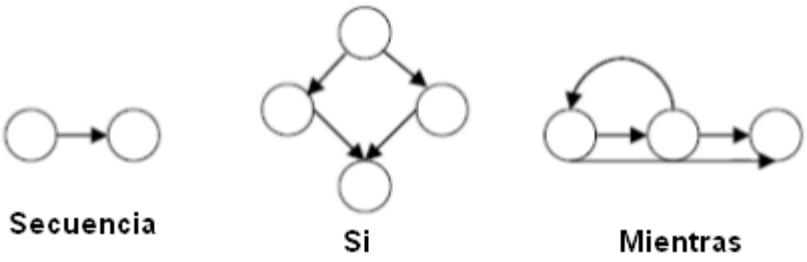

Figura 28: Notación de grafos de flujo para las instrucciones: Secuenciales, Si, Mientras

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

Componentes del grafo de flujo:

- **Nodo:** son los círculos representados en el grafo, el cual contiene una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
- **Aristas:** son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aún cuando el nodo no representa la sentencia de un procedimiento.
- **Regiones:** son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalentes a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la prueba del Camino Básico es preciso calcular la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se muestra en la Figura 29 el código del método *executeActualizarDisfrutado (sfWebRequest $request)* encargado de actualizar si el trabajador disfruta o no las vacaciones.

```
public function executeActualizarDisfrutado(sfWebRequest $request) {
  $id = $request->getParameter('id');//1
  $disfrutado = $request->getParameter('disfrutado');//1
  $obj = Doctrine::getTable('PlanVacaciones')->find($id);//1
  $formPlan = new PlanVacacionesForm($obj);//1
  $id_trabajador = $formPlan->getObject()->getIdTrabajador();//1
  $fecha_inicio = $formPlan->getObject()->getFechaInicio();//1
  $fecha_fin = $formPlan->getObject()->getFechaFin();//1
  $fecha_incorp = $formPlan->getObject()->getFechaIncorporacion();//1
  $disp = $this->calcularDisponibilidadVacaciones($fecha_inicio, $fecha_fin, $id_trabajador);
  $pidio = $disp['dias_a_pedir'];//1
  $trabajados = $disp['dias_trabajados'];//1
  $formT = new TrabajadorForm(Doctrine::getTable('Trabajador')->find($id_trabajador));//1
  $dias = $formT->getObject()->getDiasPorDisfrutar();//1
  if ($disfrutado == 'false') {//2
      $cantidad = $dias + (int) ($pidio - $trabajados);//3
  } else //4
      $cantidad = $dias - (int) ($pidio - $trabajados);//4
  }//4
  $formT->getObject()//5
          ->setDiasPorDisfrutar($cantidad)//5
          ->setFechaInicioLaboral($fecha_incorp)//5
          ->save();//5
  $formPlan->getObject()//5
          ->setDisfrutado($disfrutado)//5
          ->save();//5
```

```
    $planes_no_disf = Doctrine::getTable('PlanVacaciones')
    ->findBy('disfrutado', false, Doctrine::HYDRATE_ARRAY);//5
    foreach ($planes_no_disf as $plan => $p) {//6
        $fi = $p['fecha_inicio'];//7
        $ff = $p['fecha_fin'];//7
        $idt = $p['id_trabajador'];//7
        $disp = $this->calcularDisponibilidadVacaciones($fi, $ff, $idt);//7
        if (!$disp['disponible']) {//8
            $formPlan = new PlanVacacionesForm(Doctrine::getTable('PlanVacaciones')//9
            ->find($p['id']));//9
            $formPlan->getObject()->delete();//9
        }//9
    }//10
    $this->renderText('OK');//11
    return sfView::NONE;//11
}//12
```

Figura 29: Método executeActualizarDisfrutado (sfWebRequest $request)

Para el cálculo de la complejidad ciclomática es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la Figura 30:

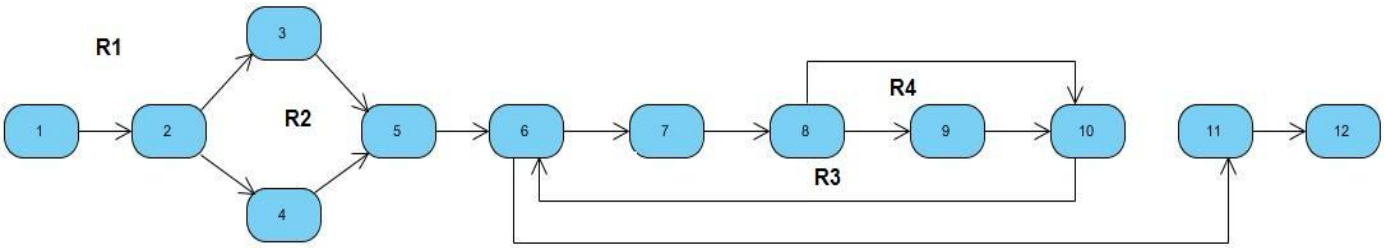

Figura 30: Grafo de flujo asociado al método

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres vías o fórmulas, se debe utilizar el mismo grafo en cada caso:

Fórmula 1 .V (G) = (A – N) + 2

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

Resultado.

V (G) = (14 - 12) + 2

V (G) = 4

Fórmula 2 .V (G) = P + 1

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

Resultado.

V (G) = 3 + 1

V (G) = 4

Fórmula 3 .V (G) = R

- Siendo “R” la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado.

V (G) = 4

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir 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.

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Camino básico # 1: 1, 2, 3, 5, 6, 7, 8, 10, 6, 11, 12.

Camino básico # 2: 1, 2, 4, 5, 6, 7, 8, 10, 6, 11, 12.

Camino básico # 3: 1, 2, 3, 5, 6, 7, 8, 10, 6, 11, 12.

Camino básico # 4: 1, 2, 4, 5, 6, 7, 8, 9, 10, 6, 11, 12.

Es necesario aclarar que existen otros caminos en el grafo que no son considerados independientes ya que son combinaciones de los anteriormente especificados.

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 (Tablas 13, 14, 15 y 16). Para realizarlos es necesario cumplir con las siguientes exigencias:

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

Camino básico # 1: 1, 2, 3, 5, 6, 7, 8, 10, 6, 11, 12.	
Descripción	Se actualiza el estado del plan de vacaciones y la cant_dias por disfrutar así como validar si el próximo plan de vacaciones es válido o no según la cant_dias disponibles del trabajador, en caso de no serlo se elimina.
Condición de ejecución	Debe estar creado al menos un plan de vacaciones. Se debe seleccionar el plan al que se le quiere actualizar su estado.
Entrada	$id = $request->getParameter('id'); $disfrutado = $request->getParameter('disfrutado'); $obj = Doctrine::getTable('PlanVacaciones')->find($id); $formPlan = new PlanVacacionesForm($obj); $id_trabajador=$formPlan->getObject()->getIdTrabajador(); $fecha_inicio = $formPlan->getObject()->getFechaInicio(); $fecha_fin = $formPlan->getObject()->getFechaFin(); $fecha_incorp = $formPlan->getObject()->getFechaIncorporacion(); $disp = $this->calcularDisponibilidadVacaciones($fecha_inicio, $fecha_fin, $id_trabajador); $pidio = $disp['dias_a_pedir']; $trabajados = $disp['dias_trabajados']; $formT = new TrabajadorForm(Doctrine::getTable('Trabajador')->find($id_trabajador)); $dias = $formT->getObject()->getDiasPorDisfrutar();
Resultados esperados	Se debe actualizar el estado del plan de vacaciones y la cant_dias por disfrutar.
Resultados	Se actualizó el estado del plan de vacaciones y se disminuyó la cant_dias por disfrutar.
Salida	Rende-rizar texto.

Tabla 13: Caso de prueba para el camino básico 1

Camino básico # 2: 1, 2, 4, 5, 6, 7, 8, 10, 6, 11, 12.

Descripción	Se actualiza el estado del plan de vacaciones y la cant_dias por disfrutar así como validar si el próximo plan de vacaciones es válido o no según la cant_dias disponibles del trabajador, en caso de no serlo se elimina.
Condición de ejecución	Debe estar creado al menos un plan de vacaciones. Se debe seleccionar el plan al que se le quiere actualizar su estado.
Entrada	$id = $request->getParameter('id'); $disfrutado = $request->getParameter('disfrutado'); $obj = Doctrine::getTable('PlanVacaciones')->find($id); $formPlan = new PlanVacacionesForm($obj); $id_trabajador=$formPlan->getObject()->getIdTrabajador(); $fecha_inicio = $formPlan->getObject()->getFechaInicio(); $fecha_fin = $formPlan->getObject()->getFechaFin(); $fecha_incorp = $formPlan->getObject()- >getFechaIncorporacion(); $disp = $this- >calcularDisponibilidadVacaciones($fecha_inicio, $fecha_fin, $id_trabajador); $pidio = $disp['dias_a_pedir']; $trabajados = $disp['dias_trabajados']; $formT = new TrabajadorForm(Doctrine::getTable('Trabajador')->find($id_trabajador)); $dias = $formT->getObject()->getDiasPorDisfrutar();
Resultados esperados	Se debe actualizar el estado del plan de vacaciones y la cant_dias por disfrutar.
Resultados	Se actualizó el estado del plan de vacaciones y se aumentó la cant_dias por disfrutar.
Salida	Rende-rizar texto.

Tabla 14: Caso de prueba para el camino básico 2

Camino básico # 3: 1, 2, 3, 5, 6, 7, 8, 10, 6, 11, 12.	
Descripción	Se actualiza el estado del plan de vacaciones y la cant_dias por disfrutar así como validar si el próximo plan de vacaciones es válido o no según la cant_dias disponibles del trabajador, en caso de no serlo se elimina.
Condición de ejecución	Debe estar creado al menos un plan de vacaciones. Se debe seleccionar el plan al que se le quiere actualizar su estado.
Entrada	$id = $request->getParameter('id'); $disfrutado = $request->getParameter('disfrutado'); $obj = Doctrine::getTable('PlanVacaciones')->find($id); $formPlan = new PlanVacacionesForm($obj); $id_trabajador=$formPlan->getObject()->getIdTrabajador(); $fecha_inicio = $formPlan->getObject()->getFechaInicio(); $fecha_fin = $formPlan->getObject()->getFechaFin(); $fecha_incorp = $formPlan->getObject()- >getFechaIncorporacion(); $disp = $this- >calcularDisponibilidadVacaciones($fecha_inicio, $fecha_fin, $id_trabajador); $pidio = $disp['dias_a_pedir']; $trabajados = $disp['dias_trabajados']; $formT = new TrabajadorForm(Doctrine::getTable('Trabajador')->find($id_trabajador)); $dias = $formT->getObject()->getDiasPorDisfrutar();
Resultados esperados	Se debe actualizar el estado del plan de vacaciones y la cant_dias por disfrutar.
Resultados	Se actualizó el estado del plan de vacaciones y se aumentó la cant_dias por disfrutar.

Salida	Rende-rizar texto.

Tabla 15: Caso de prueba para el camino básico 3

Camino básico # 4: 1, 2, 4, 5, 6, 7, 8, 9, 10, 6, 11, 12.	
Descripción	Se actualiza el estado del plan de vacaciones y la cant_dias por disfrutar así como validar si el próximo plan de vacaciones es válido o no según la cant_dias disponibles del trabajador, en caso de no serlo se elimina.
Condición de ejecución	Debe estar creado al menos un plan de vacaciones. Se debe seleccionar el plan al que se le quiere actualizar su estado.
Entrada	$id = $request->getParameter('id'); $disfrutado = $request->getParameter('disfrutado'); $obj = Doctrine::getTable('PlanVacaciones')->find($id); $formPlan = new PlanVacacionesForm($obj); $id_trabajador=$formPlan->getObject()->getIdTrabajador(); $fecha_inicio = $formPlan->getObject()->getFechaInicio(); $fecha_fin = $formPlan->getObject()->getFechaFin(); $fecha_incorp = $formPlan->getObject()- >getFechaIncorporacion(); $disp = $this- >calcularDisponibilidadVacaciones($fecha_inicio, $fecha_fin, $id_trabajador); $pidio = $disp['dias_a_pedir']; $trabajados = $disp['dias_trabajados']; $formT = new TrabajadorForm(Doctrine::getTable('Trabajador')->find($id_trabajador)); $dias = $formT->getObject()->getDiasPorDisfrutar();
Resultados esperados	Se debe actualizar el estado del plan de vacaciones y la cant_dias por disfrutar.

Resultados	Se actualizó el estado del plan de vacaciones, se disminuyó la cant_dias por disfrutar y se eliminó el siguiente plan porque no tiene acumulado los días solicitados.
Salida	Rende-rizar texto.

Tabla 16: Caso de prueba para el camino básico 4

Resultado de las pruebas de Caja Blanca

Con la realización de las pruebas de Caja Blanca a diferentes métodos significativos de la aplicación, fue posible apreciar como estos respondían satisfactoriamente a los resultados que se esperaban por cada caso de prueba. Se puede observar que los códigos implementados cumplen con lo que se necesita para lograr una buena implementación de los diferentes requisitos que se definieron en el negocio.

3.4.2. Pruebas de Caja Negra

Las pruebas de Caja Negra, también denominadas, pruebas de comportamiento, se concentran en los requisitos funcionales del software. Permiten derivar conjuntos de condiciones de entrada para ejercitar por completo todos los requisitos funcionales de un programa, por lo que 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 una salida correcta.

El diseño de estas pruebas tiene el propósito de detectar:

- Funciones incorrectas o faltantes.
- Errores de interfaz.
- Errores en estructuras de datos o en acceso a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y término (Pressman, 2002).

Figura 31. Representación de técnica de prueba de Caja Negra

Existen diferentes técnicas de prueba de Caja Negra descritas en (Pressman, 2002) para validar la funcionalidad del sistema sin entrar a analizar su ejecución interna:

- Métodos gráficos de prueba.
- Partición de equivalencia.
- Análisis de valores límites.
- Prueba de tabla ortogonal.

De estas técnicas, la seleccionada fue partición de equivalencia la cual divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba.

La partición de equivalencia define casos de pruebas que descubran ciertas clases de errores, reduciendo así el número total de casos de pruebas.

- Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.
- Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:
    1. Si una condición de entrada especifica un rango, se define una clase de equivalencia válida y dos no válidas.
    2. Si una condición de entrada requiere un valor específico, se define una clase de equivalencia válida y dos no válidas.
    3. Si una condición de entrada especifica un miembro de un conjunto, se define una clase de equivalencia válida y dos no válidas.
    4. Si una condición de entrada es booleana, se define una clase de equivalencia válida y otra no válida (Pressman, 2002).

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Se realizaron 75 diseños de casos de prueba para validar el sistema atendiendo a la técnica partición de equivalencia. A continuación se especifica en las Tablas 17, 18 y 19 el caso de prueba para el requisito “Adicionar incidencia”.

Descripción general

Se debe seleccionar a un trabajador, luego en el menú Disciplina escoger Incidencia y por último agregar nueva Incidencia.

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.

- Debe haber al menos un trabajador en el sistema o debe haber al menos un estudiante en el sistema.
- Se debe presionar el botón Nueva de la barra de opciones de Incidencia.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar incidencia	El sistema debe permitir Adicionar incidencias.	EP 1.1: Adicionar incidencia introduciendo datos válidos.	– Se introducen los datos de la incidencia correctamente. – Se presiona el botón **Guardar**. – Se muestra la incidencia adicionada.
		EP 1.2: Adicionar incidencia introduciendo datos inválidos.	– Se introducen los datos de la incidencia. – No permite introducir datos inválidos. – Mantiene la ventana abierta hasta que se de en el botón guardar.
		EP 1.3: Adicionar incidencias dejando campos vacíos.	– Se introducen algunos datos de la incidencia dejando campos en blanco. – Se presiona el botón **Guardar**. – Automáticamente se muestra una línea roja en el campo vacío indicando que es obligatorio. – Al pasarle el mouse muestra el mensaje, ”Este campo es requerido”. – Se mantiene la ventana

		abierta hasta introducir todos los datos o cerrarla.
	EP 1.4: Cerrar.	– Se introducen o no los datos de la incidencia. – Se presiona el botón **Cerrar**.

Tabla 17: Posibles escenarios al Adicionar una Incidencia

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Inválido	Inválido
1	Fecha	Campo de selección	NA	NA	Vacío	NA	NA
2	Denominación	Campo de selección (no editable)	NA	NA	Vacío.	NA	NA
3	Horas	Campo de texto.	NA	NA	Vacío.	NA	NA
4	Descripción	Campo de texto.	Letras, caracteres especiales y números.	NA	NA	NA	NA

Tabla 18: Descripción de las variables del Adicionar Incidencia

Id del escenario	Escenario	Fecha	Denominación	Horas	Descripción	Respuesta del sistema
EP 1.1	Adicionar incidencia introduciendo datos válidos	V(26-04-2013)	V(Denom A)	V(4)	V(abc@abc123)	El sistema adiciona la incidencia satisfactoriamente. El sistema cierra la interfaz.

EP 1.2	Adicionar incidencia introduciendo datos inválidos.	NA	V(Denom A)	V(4)	V(abc@abc123)	
		V(26-04-2013)	NA	V(4)	V(abc@abc123)	
		V(26-04-2013)	V(Denom A)	I(ah4)	V(abc@abc123)	El sistema no permite la inserción de letras o caracteres especiales. El sistema mantiene la interfaz abierta.
		V(26-04-2013)	V(Denom A)	V(4)	NA	
EP 1.3	Adicionar incidencias dejando campos vacíos.	I(Vacío)	V(Denom A)	V(4)	V(abc@abc123)	El sistema marca con una línea roja los campos vacíos. Al pasarle por encima con el mouse muestra un mensaje de error. El sistema mantiene la interfaz abierta.
		V(26-04-2013)	I(Vacío)	V(4)	V(abc@abc123)	El sistema marca con una línea roja los campos vacíos. Al pasarle por encima con el mouse muestra un mensaje de error. El sistema mantiene la

						interfaz abierta.
		V(26-04-2013)	V(Denom A)	I(Vacío)	V(abc@abc123)	El sistema marca con una línea roja los campos vacíos. Al pasarle por encima con el mouse muestra un mensaje de error. El sistema mantiene la interfaz abierta.
		V(26-04-2013)	V(Denom A)	V(4)	I(Vacío)	El sistema marca con una línea roja los campos vacíos. Al pasarle por encima con el mouse muestra un mensaje de error. El sistema mantiene la interfaz abierta.
EP 1.5	Cerrar.					El sistema cierra la interfaz sin realizar ninguna operación.

Tabla 19: Respuestas del sistema para los posibles escenarios al Adicionar una Incidencia

Resultado de las pruebas de Caja Negra

Las pruebas de Caja Negra a la solución fueron desarrolladas por el Grupo de Aseguramiento de la Calidad del CEIGE. Estas se realizaron en tres iteraciones, donde finalmente se comprobó en la tercera iteración que el sistema estaba libre de no conformidades culminando así la fase de pruebas internas. En el Anexo 16 se puede observar el acta de liberación emitida.

La Tabla 20 muestra los resultados obtenidos en cada una de las iteraciones realizadas, desglosando las no conformidades detectadas en significativas (S) y no significativas (NS).

Iteración	No conformidades (S)	No conformidades (NS)	Total
1	3	18	21
2	3	8	11
3	0	0	0

Tabla 20: Resultados de las iteraciones de las pruebas de caja negra realizadas por calidad

3.5. Resultados de la encuesta

Se realizó una encuesta con el objetivo de caracterizar la gestión de la información de los RRHH del CEIGE a partir de la evaluación de los indicadores: actualización, homologación y unificación. La muestra seleccionada para la encuesta mediante Muestreo no probabilístico, tipo por conveniencia fue de 21 gestores de información de RRHH del centro, lo que representa un 70 % de la población. Luego de haber tabulado la información, arrojó como resultado que el 61,9 % de los encuestados coinciden en que la información referente a sus RRHH no se encuentra actualizada siempre y no coincide con la de los niveles superiores, mientras que un 52,38 % dice que la información no se encuentra en una misma base de datos. De esta forma se reafirma que los procedimientos utilizados en el centro para la gestión de la información de los RRHH no permiten la actualización, homologación y unificación de la información de forma correcta. Para un mayor entendimiento de estos resultados dirigirse al Anexo 15.

3.6. Conclusiones parciales

Se mostraron los posibles escenarios en los cuales puede ser desplegado el sistema a través del diagrama de despliegue, así como los requerimientos de software necesarios para un correcto funcionamiento de este. Se realizó la implementación del SGI de RRHH que permitió contar con la solución. Se realizó la validación del diseño a través de las métricas seleccionadas, las cuales mostraron que el diseño cumple satisfactoriamente con los atributos medidos. Se realizaron las pruebas de caja blanca y caja negra para verificar tanto el código como las funcionalidades del sistema, arrojando que el sistema cumple con las necesidades del cliente.

CONCLUSIONES

Con la realización y culminación del presente trabajo, se logró obtener un SGI para los RRHH del centro CEIGE, el cual es consecuente con el paradigma de independencia tecnológica por el cual apuesta el país. Este sistema contribuye a un mejor manejo de la información de los recursos humanos del centro, ya que es capaz de recopilar la mayor cantidad de información en un mismo sistema y ayuda a la toma de decisiones.

Para lograr dicho resultado se realizó un estudio que permitió conocer detalles de cómo se maneja la gestión de información de los recursos humanos en las empresas, a partir de diferentes sistemas tanto nacionales como internacionales, en el análisis se concluyó que es más factible crear una solución propia que responda a las particularidades del centro, puesto que el mismo se encuentra incluido dentro de un centro educacional. Se detallaron las tecnologías y herramientas seleccionadas para la implementación del sistema y se efectuó el modelado del sistema aplicando patrones, que contribuyeron a un ahorro considerable de tiempo en la construcción del software.

Se realizó la validación del diseño aplicando métricas las cuales arrojaron valores positivos. Se desarrolló el SGI de RRHH a partir del diseño validado y se realizaron pruebas al mismo que validaron su calidad, cumpliendo así los objetivos de la investigación.

RECOMENDACIONES

Se recomienda:

- Sistematizar el uso de la solución en el CEIGE e incorporarle posibles mejoras que surjan.
- Desplegar el sistema en otras áreas de la Universidad.
- Vincular el trabajo realizado a un proyecto que se encargue de brindar soporte y mantenimiento.

REFERENCIAS BIBLIOGRÁFICAS

Potencier, Fabien y Weaver, Ryan . 2010. *Symfony 2.0, el libro oficial.* EEUU : s.n., 2010.

2000, Seven. seven2k. [En línea] Amplus Sistemas C.A. [Citado el: 15 de 2 de 2012.] http://www.seven2k.com/index.html.

Alexander, Christopher. 1979. *The Timeless Way of Building*. 1979.

Ana Celia Botta Parapar, Guadalupe Martínez Suárez. *Rediseño del Subsistema de Relaciones Financieras del sistema de Perfeccionamiento Empresarial.*

Arqhys. [En línea] [Citado el: 1 de 4 de 2013.] http://www.arqhys.com/general/diagrama-de-clases.html.

Asleson, Ryan. 2008. *Foundations of Ajax.* s.l. : Apress, 2008. ISBN: 1-59059-582-3.

ASSETS. 2011. assets. [En línea] 2011. [Citado el: 7 de diciembre de 2012.] http://www.assets.co.cu/humanos.asp.

autores, Colectivo. *Manual del Usuario y de Explotación Versat Sarasola.* VillaClara, Cuba : s.n.

BAIRESIT. 2010. BAIRESIT. [En línea] 2010. [Citado el: 21 de 03 de 2013.] http://www.bairesit.com/2010/12/lanzamiento-de-nueva-version-de-sencha-ext-js/.

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

Baryolo, Oiner Gómez, y otros. 2008. *Plantilla Registro de la Propiedad intelectual(Sauxe).* Habana : s.n., 2008.

Bertino, E. A. y Martino, L. A. 1995. *Sistemas de bases de datos orientadas a objetos.* s.l. : Díaz de Santo , 1995.

Blasco, Albert. 2009. symfony-beneficios-e-informacion. [En línea] 15 de 05 de 2009. [Citado el: 21 de febrero de 2013.] http://www.thaira.net/blog/desarrolloweb/symfony-beneficios-e-informacion/.

blogs.ua.es. [En línea] [Citado el: 3 de abril de 2013.] http://blogs.ua.es/mu171m3d14/2011/04/17/modelo-de-datos-1%C2%AA-parte/.

Borbón, Yoandry Morejón, Baryolo, Oiner Gómez and García, Darien. ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. [Online] [Cited: 11 23, 2012.] http://www.buenastareas.com/ensayos/Arquitectura-Tecnol%C3%B3gica-Para-El-Desarrollo-De/2210642.html..

CAMILO ESTEBAN MAHECHA BADILLO, RAUL ORTIZ. 2009. www.calidadindustriaalimentaria.wordpress.com/.

www.calidadindustriaalimentaria.wordpress.com/2009/07/14/mapa-de-procesos/. [Online] julio 14, 2009. [Cited: marzo 6, 2013.]

2011. Catálogo de software. [En línea] DIGITAL WARE S.A., 29 de Diciembre de 2011. http://www.catalogodesoftware.com/producto-seven-erp-234.

Catillo Ortiz, Isabel Natali, y otros. 2009. *Nómina, su análisis y aplicación*. Mexico, D.F : s.n., 2009.

CEIGE. 2012. *CEIGE-Modelo de Desarrollo de Software.* La Habana : s.n., 2012.

—. 2012. *CEIGE-Modelo de Desarrollo de Software.* La Habana : s.n., 2012.

—. 2010. *Procesos Capital Humano*. 2010.

Chávez, Joel Manrrique. 2006. Lenguaje de Programación PHP. *www.monografias.com.* [Online] 11 4, 2006. [Cited: Noviembre 27, 2012.] http://www.monografias.com/trabajos38/programacion-php/programacion-php.shtml.

CITMAEL. *Sistema Integral Administrativo Rodas XXI y Servicios Postventa*. [En línea] [Citado el: 8 de diciembre de 2011.] http://www.citmatel.cu/geren1_1.php.

CITMATEL. 2002-2012. RodasXXI. [En línea] CITMATEL, 2002-2012. [Citado el: 15 de 1 de 2012.] http://www.rodasxxi.cu/.

Cuesta, Armando Santos. 1999. *Tecnología de la Gestión de Recursos Humanos*. La Habana : Editorial Academia, 1999.
Cuesta, Armando. 2008. *Tecnología de Gestión de Recursos Humanos y del conocimiento*. La Habana, Cuba : s.n., 2008. ISBN 9789588308661.
Definiciones.de. [En línea] [Citado el: 1 de 4 de 2013.] http://definicion.de/modelo-de-datos/.
Delgado, Marta y Machado, Neily. 2002. *Modelo conceptual de un proyecto de software utilizando el rasonamiento basado en casos*. La Habana, Cuba : s.n., 2002. ISSN 1815-5936.
Dra. Silvia Gil Fundora, Dr. Wilfredo Francisco Martín. 2011. *Implementación del Sistema de Dirección y Gestión Empresarial: Sistema de Seguridad y Salud en el Trabajo*. 2011.
2011. ebookbrowse.com. [En línea] 29 de Marzo de 2011. [Citado el: 5 de Marzo de 2012.] http://ebookbrowse.com/tema-1-normalizaci%C3%B3n-bibliograf%C3%ADa-libro-de-bd-de-rosa-maria-pdf-d91217792.
Echemendia, Yarenis y Benítes, Analina. 2012. *Diseño e implementación del componente Nómina del subsistema Capital humano del Sistema Integral de Gestión CEDRUX*. La Habana, Cuba : s.n., 2012.
EICMA, Dirección. 2010. EICMA. *Versat Sarasola, Un efectivo sistema cubano de contabilidad.* [En línea] 23 de septiembre de 2010. [Citado el: 9 de diciembre de 2011.] http://www.eimagr.cu/index.php/component/content/article/60-portada/101-versat-sarasola-un-sistema-para-trabajar.
2012. elwebmaster. [En línea] 7 de diciembre de 2012. http://www.elwebmaster.com/articulos/%C2%BFque-es-el-dom-modelo-de-objetos-del-documento.
Encinosa, Lázaro J. Blanco. 2011. *La informática en la dirección de empresas*. La Habana, Cuba : Felix Varela, 2011. ISBN 978-959-07-1629-4.
2008. es.scribd.com. [En línea] 11 de Noviembre de 2008. [Citado el: 24 de Febrero de 2012.] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.
Española, Real Academia. *Diccionario de la Lengua Española*. [En línea] [Citado el: 6 de diciembre de 2011.] http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=nomina.
Facultad de Informática – UPM. fermat.usach.cl. [En línea] [Citado el: 17 de Febrero de 2012.] fermat.usach.cl/~msanchez/comprimido/OBJETOS.pdf.
Farfán, Cesar. 2012. Cliente web y Servidor web. [En línea] 11 de 05 de 2012. [Citado el: 16 de 01 de 2013.] http://www.slideshare.net/olea_saavedra/cliente-web-y-servidor-web-12896858.
Flanagan, David. 2006. *Java Script. The definitive guide*. Estados Unidos de America : O´Reilly, 2006. 978-0-596-10199-2.
Foundation, The Apache Software. 2008. *www.apache.org*. [Online] 2008. [Cited: Noviembre 27, 2012.] http://projects.apache.org/indexes/quick.html.
2012. free download manager. [Online] diciembre 7, 2012. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.
G. Martínez, Larissa . 2004. *Administración de recursos humanos*. 2004. pp. 8-9.
Genix, Ing. Alina de la Concepción Isasi. 2010. *Estudio sobre la información que se gestiona de los recursos humanos, en los ERP*. Habana : DESOFT, 2010.
Geofisica. [En línea] [Citado el: 5 de abril de 2013.] http://mmc.geofisica.unam.mx/LuCAS/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html.
Gómez Baryolo, Oiner, Morejón Borbón, Yoandry y Garcia, Darien. 2011. ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. [En línea] 22 de mayo de 2011. [Citado el: 10

de diciembre de 2011.] http://www.buenastareas.com/ensayos/Arquitectura-Tecnol%C3%B3gica-Para-El-Desarrollo-De/2210642.html.
Heredero, Carmen de Pablos, y otros. 2006. *Dirección y gestión de los sistemas de información en la empresa: Una visión integradora*. Madrid : ESIC EDITORIAL, 2006. ISBN: 84-7356-445-6.
Hernandis, José Alberto. 2009. *http://www.versioncero.com*. [Online] 2009. [Cited: 11 23, 2013.] http://www.versioncero.com/noticia/210/visualparadigm-.
Hidelvys Cantero Cora, Elisa Leyva Cardeñosa , Ricardo Rojas Casas , Tania Ballester Marsal. 2012. PROCEDIMIENTO PARA EL DIAGNÓSTICO DE SEGURIDAD Y SALUD DEL TRABAJO (SST). *http://www.eumed.net/cursecon/ecolat/cu/2012/cccm.html*. [Online] 2012.
http://elvex.ugr.es. *http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf*. [Online] [Cited: 3 6, 2013.]
Ibarra, Antonio Aliaga. 2008. *www.iessanvicente.com*. [Online] Enero 21, 2008. [Cited: Noviembre 27, 2012.] http://www.iessanvicente.com/colaboraciones/postgreSQL.pdf.
Isasi, Alina de la Concepción, Roblejo, Yenlis González and Guibert, Pedro Valdés. 2010. Estudio sobre la información que se gestiona de los recursos humanos, en los ERP. 2010.
jotadeveloper. 2009. jotadeveloper blog. [En línea] 03 de 07 de 2009. [Citado el: 12 de 01 de 2013.] http://blog.jotadeveloper.es/tag/mvc/.
JuntAndalucia. 2009. Marco de desarrollo de la junta de Andalucía. [En línea] 2009. [Citado el: 3 de abril de 2013.] http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/407.
2012. kioskea. [En línea] 7 de diciembre de 2012. http://es.kioskea.net/contents/langages/langages.php3.
Laguna, Miguel A. www.infor.uva.es. *www.infor.uva.es/~mlaguna/is1/apuntes/2-requisitos.pdf*. [Online] [Cited: marzo 10, 2013.]
Larman, Craig. 1999. *UML y Patrones. Introducción al analisis y diseño orientado a objetos*. s.l. : PRENTICE HAL, 1999. ISBN: 970-17-0261-1.
Ley 49. 1984. Ley 49, Trabajo. *Capítulo 3,SECCIÓN OCTAVA*. s.l. : Ley del trabajo. Vacaciones anuales pagadas., 1984.
librosweb. [En línea] [Citado el: 26 de 1 de 2013.] http://www.librosweb.es/symfony_1_1/capitulo_1/symfony_en_pocas_palabras.html.
Lic.Antonio Toledo Carnicero, Lic. Pablo Perez Perez. 2004. *GESTIÖN DE SISTEMAS DE INFORMACIÖN*. Universidad de Deusto : Facultad de Ingeniería, 2004.
López, Alejandro Cadavid. Mozilla Firefox, el navegador web del momento. [En línea] [Citado el: 10 de diciembre de 2011.] http://www.maestrosdelweb.com/editorial/firefox/.
—. 2009. Mozilla Firefox, el navegador web del momento. [Online] 2009. [Cited: Noviembre 27, 2012.] http://www.maestrosdelweb.com/editorial/firefox/.
Mahecha, Camilo Steban. 2012. Scribd. [En línea] 03 de 10 de 2012. [Citado el: 3 de abril de 2013.] http://es.scribd.com/doc/84774712/MAPAS-DE-PROCESOS.
Maldonado, Daniel. 2007. El CoDiGo K. [En línea] 25 de 1 de 2007. [Citado el: 10 de 12 de 2012.] http://elcodigok.blogspot.com/2007/09/que-son-los-ide-de-programacin.html.
Martínez, Alejandro y Martínez, Raúl. 2010. www.dsi.uclm.es. [En línea] 2010. [Citado el: 1 de abril de 2013.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.
Martínez, Alejandro y Martínez, Raúl. Guía a Rational Unified Process. *Universidad de Castilla-La Mancha*. [En línea] [Citado el: 28 de febrero de 2012.] http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf.
MILESTONE Consulting. 2012. MILESTONE Consulting. [En línea] 2012. [Citado el: 7 de diciembre de 2012.] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.
Morales, Aylín, Caballero, Juan Alberto and Abrante, Yunet Suárez. 2011. *COMPONENTE DE SOFTWARE PARA GESTIONAR COMPETENCIAS LABORALES MEDIANTE CEDRUX*. 2011.

Msdn. [En línea] [Citado el: 1 de 4 de 2013.] http://msdn.microsoft.com/es-es/library/dd409390.aspx.
NetBeans, Sitio Oficial. 2010. Netbeans. [En línea] 2010. [Citado el: 18 de 1 de 2013.] http://netbeans.org/community/releases/69/index_es.html.
Ochoa, Prof. Sergio. 2005. *Diseño y Arquitectura*. 2005.
Orange HRM. 2008. AplicacionesEmpresariales.com. [En línea] 24 de septiembre de 2008. [Citado el: 30 de 05 de 2013.] http://www.aplicacionesempresariales.com/orange-hrm-control-de-personal.html.
Padilla, Lillian. 2007. PLANEACION DE LOS RECURSOS DE LA EMPRESA: ERP. *www.tec.url.edu.gt*. [Online] 2007. [Cited: Noviembre 13, 2013.] www.tec.url.edu.gt/boletin/URL_02_IND03.pdf.
Partner, Sage North America Top ERP. Blytheco. [En línea] [Citado el: 5 de 5 de 2012.] http://www.blytheco.com/mas500/price-list.asp.
Pérez, Mario Raúl. 2011. [Online] 2011. [Cited: 11 23, 2012.] https://sites.google.com/site/zendframeworkextjsdoctrine/guias/doctrine..
Pescador, Alison y Pinacho, Mayra. 2012. Scribd. [En línea] 7 de diciembre de 2012. [Citado el: 11 de 02 de 2013.] http://es.scribd.com/doc/20509903/Importancia-de-Los-Recursos-Humanos-en-La-Empresa.
plusformacion. [En línea] [Citado el: 26 de 1 de 2013.] http://www.plusformacion.com/recursos-pgadmin-kws50766.
Ponce, Ma. Eugenia López. *Definición y Comprensión del Proceso*.
Ponjuán, G. 2000. *Aplicaciones de Gestión de información en las organizaciones. El profecional de la informción y su dominio de las técnicas y herramientas de la Gestión*. La Habana,Cuba : s.n., 2000.
Pressman, Roger. 2002. *Ingeniería del Software. Un enfoque práctico*. Interamericana de España. : McGraw-Hil, 2002.
Pressman, Roger S. 2002. *Ingeniería de Software, un enfoque práctico. Quinta edición*. s.l. : McGraw-Hill Companies, 2002. ISBN: 8448132149.
Prf. Luza, César. 2008. *Técnicas de modelamiento. Sesión 2: Introducción a UML*. Lima, Preú : s.n., 2008.
Rasmus Lerdorf, Kevin Tatroe & Peter MacIntyre. 2006. *Programming PHP*. Estados Unidos de América : O´Relly, 2006. 978-0-596-00681-5.
Recursos, Sistema de Gestión Integrada de los. 2007. *Norma Cubana*. La Habana : s.n., 2007. 3000.
2003. RESOLUCION No. 19/03. 2003.
revistatelematica.cujae.edu.cu. *Patrón Modelo-Vista-Controlador*. [En línea] [Citado el: 18 de febrero de 2013.] http://revistatelematica.cujae.edu.cu.
Rodas XXI. 2002. RODAS XXI. *Sistema Integral Económico Administrativo*. [En línea] 2002. [Citado el: 7 de diciembre de 2012.] http://www.rodasxxi.cu/rrhh.php.
Ruiz, Francisco. 2011. Alarcos. [En línea] 2011. [Citado el: 3 de abril de 2013.] alarcos.inf-cr.uclm.es/doc/psgc/doc/psgc0809_parte4b_pn.pdf.
sage. Sage ERP Mas. [En línea] http://www.sagemas.com/Products/Sage-ERP-MAS-500/Human-Resources-Management/Abra-Payroll.
Sage Software, Inc. 2009. SageMas. [En línea] 12 de 2009. [Citado el: 24 de 2 de 2012.] http://www.dsdinc.com/dsd/pdf/MAS_500_StarShip_Parcel_spec.pdf. 56.
SAP. [En línea] [Citado el: 12 de 3 de 2012.] http://www.sap.com.
Schmidt, Douglas. 2000. *PATTERN-ORIENTED SOFTWARE ARCHITECTURE VOLUME 2: Patterns for Concurrent and Networked Objects*. 2000.
Schmidt, Douglas, y otros. 2000. *PATTERN-ORIENTED SOFTWARE ARCHITECTURE VOLUME 2: Patterns for Concurrent and Networked Objects*. 2000.
Sistemas de información de Recursos Humanos: ¿Cómo obtener, ordenar, analizar, evaluar y distribuirla información útil a la gestión de los Recursos Humanos de su empresa? 2012. s.l. : OPCION Consultores, 2012.

Slideshare. [En línea] [Citado el: 3 de abril de 2013.] http://www.slideshare.net/julietas/diagramas-de-proceso.
Sons, John Wiley &. 2007. *Javascript® bible, sixth edition*. New York : Inc. New York, 2007. 9780470069165.
Sosa, Maricel y Cobo, Pedro. 2008. www.disaic.cu. *DISAIC CASA CONSULTORA*. [En línea] 2008. [Citado el: 7 de diciembre de 2012.] http://www.disaic.cu/modules.php?name=content&pa=showpage&pid=818.
Soto, Lauro. 2010. *Nómina*. [En línea] 2010. [Citado el: 9 de diciembre de 2011.] http://www.mitecnologico.com/Main/Nomina.
Souza, Ana Carolina Carvalcho de Paula. 2011. EspWeb. [Online] 2011. [Cited: 11 23, 2012.] http://www.espweb.uem.br/wp/wp-content/uploads/2011/09/anacarolina.pdf.
SOUZA, ANA CAROLINA CARVALHO DE PAULA. 2011. EspWeb. [En línea] 4 de Abril de 2011. [Citado el: 21 de Mayo de 2012.] http://www.espweb.uem.br/wp/wp-content/uploads/2011/09/anacarolina.pdf.
Sparxsystems. [En línea] [Citado el: 3 de abril de 2013.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html.
Symfony.es. 2012. symfony.es. [En línea] 2012. [Citado el: 26 de 1 de 2013.] http://www.ecured.cu/index.php/Symfony.
Tedeschi, Nicolás. 2006. ¿Qué es un Patrón de Diseño? [Online] 2006. [Cited: Noviembre 27, 2012.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.
The Apache Software Foundation. *Apache Software Foundation Index: Project Listing*. [En línea] [Citado el: 10 de Marzo de 2012.] http://projects.apache.org/indexes/quick.html.
UBUNTU, GUÍA DOCUMENTADA PARA. 2009. *Subversion*. [En línea] 9 de enero de 2009. [Citado el: 7 de diciembre de 2011.] http://www.guia-ubuntu.org/index.php?title=Subversion.
Ubuntu, Guía. 2009. GUÍA DOCUMENTADA PARA UBUNTU. *Subversion*. [Online] 2009. [Cited: 11 23, 2012.] http://www.guia-ubuntu.org/index.php?title=Subversion.
Valdés, Damián Pérez. 2007. http://www.maestrosdelweb.com. *¿Qué es Javascript?* [Online] 2007. http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.
Vega Miniet, Ing. Yanet, y otros. 2009. *Ciclo de vida del proyecto*. 2009.
2012. Visual Paradigm. *Visual Paradigm for UML - UML tool for software application development*. [En línea] 13 de marzo de 2012. http://www.visual-paradigm.com/product/vpuml/.
web.ontuts.com. [En línea] [Citado el: 18 de 2 de 2013.] http://web.ntuts.com.
2013. www.cdlibre.org. [En línea] 25 de 1 de 2013. http://www.cdlibre.org/consultar/catalogo/Programacion_Sistemas-de-control-de-versiones.html.
www.practicasdesoftware.com.ar. [En línea] [Citado el: 1 de abril de 2013.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.
www.scriptcase.net. [En línea] [Citado el: 18 de 2 de 2013.] http://www.scriptcase.net.
Zaragoza, Jessica. 2008. Scribd. [En línea] 23 de 05 de 2008. [Citado el: 7 de diciembre de 2012.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.