Cedrux
SISTEMA INTEGRAL DE GESTIÓN

Universidad de las Ciencias Informáticas

Facultad 3

"Diseño e implementación del componente Vacaciones del subsistema Capital Humano del Sistema Integral de Gestión Cedrux"

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

Autor: Carlos Antonio Ibarra Romero
Tutor: Ing. Rodolfo Rodríguez Molinet
Co-tutor: Ing. Yoandry Pardo Escobar

La Habana, Junio de 2012

DECLARACIÓN DE AUTORÍA

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2012.

_____

Carlos Antonio Ibarra Romero

Autor

_____ _____

Ing. Rodolfo Rodríguez Molinet **Ing. Yoandry Pardo Escobar**

Tutor **Co-Tutor**

DATOS DE CONTACTO

Síntesis de los tutores

Ing. Rodolfo Rodríguez Molinet

Ingeniero en Ciencias Informáticas. Graduado en la Universidad de las Ciencias Informáticas en el 2008. Ha impartido las asignaturas de pregrado de Matemática I y II y Algebra Lineal. Actualmente Profesor de Investigación de Operaciones, posee la categoría docente de Instructor, actualmente es Desarrollador de la Línea Capital Humano del Departamento Desarrollo de Productos, CEIGE, Facultad 3. Ha cursado y aprobado 8 cursos de postgrado.

Correo: rmolinet@uci.cu .

Ing. Yoandry Pardo Escobar

Ingeniero en Ciencias Informáticas. Graduado en la Universidad de las Ciencias Informáticas en el 2010. Actualmente es Desarrollador de la Línea Capital Humano del Departamento Desarrollo de Productos, CEIGE, Facultad 3. Ha cursado y aprobado 8 cursos de postgrado y participado en eventos como la Semana Tecnológica organizada por FORDES.

Correo: ypescobar@uci.cu .

Todo trabajo realizado tiene detrás un conjunto de personas sin los que no hubiese sido posible llegar al resultado final y a los que les debo mi mayor agradecimiento.

En primer lugar le quisiera agradecer a mis padres por todo el amor, preocupación, apoyo, y dedicación que me han brindado a lo largo de toda mi vida, por estar siempre presentes, en los buenos y malos momentos, y por confiar en mí.

A mi hermano Manuel Alejandro, por estar siempre pendiente de mí a pesar de la distancia.

A mi tío Olber, por todos sus consejos, preocupación y todos los buenos momentos que hemos compartido en la vida.

A mi familia por todo su cariño y apoyo.

A Claudia, por ser esa persona especial en mi vida, por estar siempre pendiente de mí, brindándome su amor, apoyo y confianza.

A mis tutores, Yoandry y Rodolfo, que estuvieron en la disposición de ayudarme en todo lo que me hizo falta.

A Donel, por darme la oportunidad de integrar el equipo de trabajo de Capital Humano de CEIGE.

A toda la familia de Capital Humano, en especial a William, Arnolis, Aliankis, Aylin, Aliuska, Dayli, Yunet y Yoel.

A mi tribunal y oponente, por sus críticas constructivas y su ayuda, haciendo este de este trabajo un éxito.

A esas personas especiales, que son Analina y Yarenis, por soportarme todo este tiempo, por los buenos y malos momentos compartidos durante estos 5 años.

A mis amigos, Felipe, Yosmany (Sufus), Manuel, Jose M., Lino A. (Cervantes), Rafael A., Dannier, Dayron (Boom), Pedro F. (Makano), Raul (Void), Ariel, Leandro (Kobold), Armando (Mercenario), y Andy, por estar siempre allí, compartiendo en las buenas y en las malas situaciones, por hacer de este piquete una comunidad.

A todas las personas que he conocido a lo largo de estos cinco años, que me han demostrado su amistad y han contribuido de una forma u otra a superarme como persona y como profesional.

A todos…muchas gracias.

DEDICATORIA

A "mi madre", por darme la oportunidad de vivir y confiar en mí en todo momento…
Ya soy el primer ingeniero de tus niños.
A "mi padre", por todo el apoyo brindado y por estar tan orgulloso de mi…
Para ti el fruto de todo mi esfuerzo realizado.
A "mi abuela" Vilma, por ser la mejor abuela del mundo, por todas las cosas buenas hechas por mi…
Gracias por existir, ya tu nieto se convirtió en ingeniero.
A "mi hermano" Manuel Alejandro, por permitirme ser un ejemplo para ti…
Espero ser siempre un orgullo para ti, nunca te defraudaré.
A mi familia de Santiago, mi tío Manolito, mi abuela Gloria, Mayté, Hugo, Trilce, Arturo, Damián y Ernesto…
Gracias por todo lo que me brindaron en la vida.

A toda mi familia por su cariño y confianza.
A todo el que este trabajo le pueda servir.

Para todos ustedes es este trabajo.

Resumen

En la actualidad, las instituciones y empresas cubanas han incrementado el uso de aplicaciones informáticas con el objetivo de elevar el nivel económico y tener un control eficiente sobre los recursos que manejan. Entre las aplicaciones informáticas puestas en práctica se encuentran los sistemas de gestión empresarial, los cuales son softwares cuyo objetivo es integrar y automatizar todos los procesos que se llevan a cabo en las empresas, conocidos como Enterprise Resource Planning (ERP[1]).

La Universidad de las Ciencias Informáticas (UCI) se encuentra inmersa en la realización del sistema integral de gestión: Cedrux, dándole cumplimiento a la voluntad política de la informatización paulatina del país, basada en los principios de independencia tecnológica y con funcionalidades generales de los procesos y particularidades de la economía cubana. El presente trabajo de diploma siguiendo este propósito, tiene como objetivo realizar el diseño e implementación del componente[2] Vacaciones que se integra al subsistema Capital Humano de Cedrux.

Durante la investigación se realiza el diseño del sistema basado en los requisitos previamente identificados, se definen las herramientas a emplear para implementar el componente, además de la realización de pruebas al diseño y a la implementación para garantizar la calidad del producto.

Como resultado de esta investigación se obtiene el componente Vacaciones que se integra a la solución informática Cedrux cumpliendo con todos los requisitos identificados en el estudio del proceso.

Palabras Claves

Capital Humano, Cedrux, Subsistema, Vacaciones.

[1] Enterprise Resource Planning (Planificación de recursos empresariales).

[2] Un componente es una parte no trivial, casi independiente, y reemplazable de un subsistema que llena claramente una funcionalidad dentro de un contexto en una arquitectura bien definida. (Krutchen, 2002)

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

Durante los últimos años, ha crecido sustancialmente el reconocimiento sobre la importancia del conocimiento en la gestión de las organizaciones. El capital humano constituye uno de los factores determinantes para la obtención de los resultados positivos en la producción y los servicios. Este valor se potencia cuando el conocimiento se coloca en función del logro de los objetivos de la organización. El capital humano depende en gran medida de la capacidad de las organizaciones para desarrollar y aprovechar el conocimiento. De este término parte el conocimiento, las habilidades, los valores y el potencial innovador de la organización, entre otros elementos. La gestión de dicho capital requiere de una atención muy especial, que supone la capacidad de los directivos de identificar, medir, desarrollar y renovar el activo intangible para el futuro éxito de la organización.

La gestión del capital humano abarca un grupo de procesos que interactúan entre sí, como son: la Organización del trabajo, Competencias laborales, Selección e integración, Administración de capital humano, Estimulación moral y material, Evaluación de desempeño, Capacitación y desarrollo, Seguridad y salud en el trabajo, entre otros. Específicamente el proceso de Vacaciones se encuentra dentro de la Administración de capital humano, cuenta con el submayor de vacaciones, el cual permite la gestión de los datos referentes a los días e importes acumulados por los trabajadores durante cada período de pago. Permite además realizar la planificación anual de las vacaciones. Así como llevar el control de las notificaciones, donde se recogen los datos relacionados con la solicitud y aprobación de las vacaciones a descansar por los trabajadores y mediante el cual se notifica la misma a Nóminas.

En las entidades cubanas, los procesos relacionados con las vacaciones de los trabajadores se realizan de manera manual o mediante varios sistemas informáticos que no suplen en su totalidad las necesidades reales de las empresas y no se integran con las nóminas de salario; tales como VERSAT-Sarasola, RODAS XXI, SISCONT5, desarrollados en Cuba, y otros de producción extranjera tales como OpenERP y ASSETS. Esta situación ocasiona demora en la entrega de la información, aumento del margen de error en cuanto a la planificación de los días reales de vacaciones, salarios y regímenes de pago[3] de los trabajadores. De esta forma el seguimiento y control de la información se torna confuso y complejo, lo que trae consigo el mal funcionamiento de las entidades.

[3] período y forma de pago del salario. (Gaceta Oficial, 2008)

Como parte del fortalecimiento de la gestión de las entidades y la informatización de la sociedad cubana, la dirección del país se planteó la necesidad de crear un sistema integral de gestión, que fuese capaz de informatizar los procesos de gestión de las entidades a nivel nacional, con el fin de erradicar los problemas existentes. Por lo que en julio de 2008 la Universidad de las Ciencias Informáticas, en conjunto con el Ministerio de Finanzas y Precios (MFP), el Ministerio de Economía y Planificación (MEP) y la participación de especialistas de otras entidades desarrolladoras de software como la Unidad de Compatibilización, Integración y Desarrollo de Software para la Defensa (UCID), asumen la responsabilidad de desarrollar el sistema integral de gestión Cedrux. Integrado por varios subsistemas: Contabilidad, Logística, Capital Humano, Costos y Procesos, Finanzas, Estructura y Composición y Configuración.

Por lo cual se plantea como **problema a resolver** que, la gestión manual de las Vacaciones para la Administración de capital humano dificulta generar la nómina mediante el subsistema de Capital Humano del Sistema Integral de Gestión Cedrux, centrando su **objeto de estudio** en los sistemas de Administración de capital humano y el **campo de acción** en los procesos de las Vacaciones.

Se persigue como **objetivo general** realizar el diseño e implementación del componente Vacaciones para su integración al subsistema de Capital Humano del Sistema Integral de Gestión Cedrux.

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

- Determinar el marco teórico de la investigación que permita identificar los principales logros y limitaciones en cuanto a la gestión de Vacaciones para la Administración del capital humano y su relación con la Estimulación moral y material.
- Diseñar el componente Vacaciones basado en los requisitos previamente identificados.
- Implementar el componente Vacaciones teniendo en cuenta el diseño realizado.
- Validar el componente obtenido mediante la realización de pruebas y su integración al subsistema Capital Humano.

Para obtener nuevos conocimientos sobre el objeto que se estudia se plantean un conjunto de **tareas de investigación**:

- Estudio de la gestión de Vacaciones para la Administración del capital humano y su relación con la Estimulación moral y material.
- Análisis de los sistemas existentes identificando sus características y deficiencias fundamentales.

- Justificación del uso de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
- Elaboración de los artefactos de la solución propuesta.
- Implementación de la solución apoyándose en los artefactos generados.
- Realización de pruebas a la solución.

Por esta razón se plantea como **idea a defender** que, si se obtiene el componente Vacaciones mediante su diseño e implementación entonces se logrará generar la nómina mediante el subsistema de Capital Humano del Sistema Integral de Gestión Cedrux.

Para el correcto desarrollo del presente trabajo de diploma se emplearon métodos científicos de corte teórico y empírico tratando de mantener siempre un equilibrio entre lo cualitativo y lo cuantitativo.

1. **Métodos teóricos:**

Análisis – Síntesis: Estos métodos se utilizaron para el estudio de los procesos asociados al problema, haciendo un análisis de la información existente, permitiendo la extracción de los elementos más importantes. (Hernández León, y otros, 2011)

Histórico – Lógico: Permite conocer y comprender el estado del arte de los procesos relacionados con las Vacaciones. (Hernández León, y otros, 2011)

Modelación: Se aplica en la generación de los artefactos necesarios para el diseño de los procesos de Vacaciones. (Hernández León, y otros, 2011)

2. **Método empírico:**

Experimento: Favorece el desarrollo de pruebas para la verificación de las funcionalidades implementadas, con el fin de detectar errores y comprobar su correcto funcionamiento. (Hernández León, y otros, 2011)

Observación: A través del cual se pudo percibir y planificar como quedaría concebido el sistema. (Hernández León, y otros, 2011)

Para realizar una descripción detallada, este documento muestra el resultado de la investigación de la siguiente forma:

Capítulo 1. Fundamentación teórica: En este capítulo se enuncian los principales conceptos relacionados con los procesos asociados a las Vacaciones, se realiza un estado del arte de los Sistemas de gestión del capital humano que se utilizan a nivel nacional e internacional, se referencia a la metodología de desarrollo y la Arquitectura propuesta por el proyecto ERP-Cuba.

Capítulo 2. Diseño e implementación: En este capítulo se elabora el diseño a partir de la Especificación de requisitos como base para la implementación del mismo. Se describen las clases que fueron definidas, se analiza la descripción del estándar de codificación utilizado, de manera que sirvan de documentación a los que den continuidad a la implementación del software. Se detalla la estrategia de integración entre componentes y subsistemas.

Capítulo 3. Validación de la solución propuesta: En este capítulo se valida la solución propuesta a través de la aplicación de métricas al diseño, y de la realización de pruebas, las cuales se encuentran divididos en dos partes, la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y las pruebas de caja negra que se realizaron a la interfaz del mismo. Se tramita los resultados de las pruebas realizadas al componente obtenido como resultado de la investigación. El análisis de estos resultados ratifica el cumplimiento del objetivo propuesto con la investigación.

Capítulo 1: Fundamentación teórica

1.1. Introducción

En el presente capítulo se abarcan un grupo de conceptos relacionadas con el negocio para su mejor entendimiento. Se realiza el estudio del estado del arte de los sistemas de gestión empresarial que desarrollan los procesos relacionados con las Vacaciones, dentro y fuera de Cuba. Además se hará referencia al modelo de desarrollo elaborado por la dirección del proyecto, las herramientas, los lenguajes de modelado y de programación a utilizar, para dar solución a este problema.

1.2. Conceptos generales

1.2.1. Capital humano

Durante los últimos años, ha crecido sustancialmente el reconocimiento sobre la importancia del conocimiento en la gestión de las organizaciones. El capital humano constituye actualmente uno de los factores determinantes para la obtención de valor agregado. Este valor se potencia cuando el conocimiento se coloca en función del logro de los objetivos de la organización.

El capital humano depende en gran medida de la capacidad de las organizaciones para desarrollar y aprovechar el conocimiento. Del capital humano parten el conocimiento, las habilidades, los valores y el potencial innovador de la organización, entre otros elementos. La gestión de dicho capital requiere de una atención muy especial, que supone la capacidad de los directivos de identificar, medir, desarrollar y renovar el activo intangible para el futuro éxito de la organización. (Hernández Silva, y otros, 2006).

1.2.2. Administración del capital humano

La administración del capital humano abarca todas las funciones y responsabilidades dirigidas a atraer, contratar, desarrollar, retener los recursos de la gente y maximizar el valor del capital humano. Una parte esencial en el proceso de selección de personal exitoso incluye puestos claramente definidos, instrumentos de evaluación objetivos y procesos de contratación estandarizados. (DGPLADES, 2010)

1.2.3. Vacaciones

Las vacaciones anuales constituyen el período de interrupción de la prestación efectiva de servicios más largo que establece nuestra legislación 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 período concebido 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 se 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. Nómina

Se entiende por nómina a la lista conformada por los salarios, retenciones y otros datos relacionados con los trabajadores de una entidad, que ofrecerá información contable y estadística para la empresa y el organismo encargado de regular las relaciones laborales. Permite realizar el pago de los salarios y justificar por medio de la firma del trabajador el haberlos recibido. (Castillo Ortiz, 2009)

1.3. Sistemas informáticos existentes vinculados al objeto de estudio.

1.3.1. Versat Sarasola

Es el primer sistema de contabilidad cubano certificado, en cuya evaluación participaron el Ministerio de finanzas y precios, consultorías internacionales y el organismo encargado de la seguridad informática. (EICMA, 2010) Es un sistema económico conformado por 12 módulos que permite llevar el control y registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades, pues es configurable por cada una de ellas en el momento de su instalación. Uno de sus objetivos es permitirle a los directivos, analizar, controlar y evaluar los resultados del negocio o actividad en tiempo real. (Romero, 2009). Está compuesto por los siguientes módulos:

- Configuración.
- Contabilidad general.
- Control de inventarios.
- Generador de reportes.
- Control de activos fijos.
- Costos y procesos.
- Finanzas, caja y banco.
- Contratación y facturación.
- Planificación económico-productiva.
- Análisis económico empresarial.
- Paquetes de gestión.
- Nóminas de salario. (Cabrera, y otros, 2009)

Observación: Versat-Sarasola es una de las soluciones informáticas más utilizadas actualmente en el país. En sus últimas versiones tiene en cuenta las particularidades de la economía nacional, comprende

la gestión de recursos humanos, da tratamiento a cuatro modalidades de pagos a trabajadores, entre la que se incluye por vacaciones. Pero tiene el inconveniente que está soportado sobre tecnología privativa.

1.3.2. Sistcon 5

Sistema cubano creado por la empresa Tecnomática en el año 2007, el cual se aviene a las definiciones y conceptos del Ministerio de la Industria Básica (MINBAS) aunque por las acciones contables financieras que permite puede ser utilizado en otras entidades nacionales. Está formado por varios módulos: (Sistcont, 2008)

- ✓ Cobros y Pagos.
- ✓ Facturación.
- ✓ Activos Fijos Tangibles.
- ✓ Nóminas.
- ✓ Contabilidad de Costos.
- ✓ Inventarios.
- ✓ Efectivo en Caja y Bancos.

***Observación**:* Aplicación que al ser nacional está más a fin con las propias características del sistema cubano, permite la gestión económico-financiera y de los recursos humanos, sin embargo, el empleo de tecnología privativa es el principal inconveniente que presenta dicho sistema.

1.3.3. Rodas XXI

Sistema integral económico - administrativo creado por CITMATEL para la automatización de la gestión empresarial. Es un sistema multiempresa que cuenta actualmente con diferentes módulos: Recursos humanos, Finanzas, Contabilidad, Activos fijos, Nóminas, Inventario y facturación. Estos módulos pueden emplearse integrados en su totalidad, formando cualquier subconjunto entre ellos, o cada uno de forma independiente. (Rodas XXI, 2002)

Observación: El sistema contable Rodas XXI tiene funcionalidades muy útiles pero no presenta una gestión de recursos humanos integrada a la nómina, está desarrollado para plataforma de software propietario.

1.3.4. OpenERP

Es un sistema de gestión de empresas de licencia libre, que cubre las necesidades de las áreas de contabilidad, ventas, compras, almacén, inventario, proyectos, recursos humanos y tiendas virtuales.

Incorpora funcionalidades de gestión de documentos, conexiones con otras aplicaciones y permite trabajar remotamente mediante una interfaz web o aplicación de escritorio multiplataforma. Según su propia definición y orden de módulos básicos, a continuación se muestran algunos de ellos:

- ➢ Contabilidad.
- ➢ Estadísticas.
- ➢ Productos.
- ➢ Recursos humanos.

- Facturación.
- Gestión documental.
- Gestión de Informes.

***Observación**:* OpenERP se describe a sí mismo como el ERP de código abierto más destacado y sencillo que existe hasta el momento. Es una aplicación que gestiona de manera eficiente tanto los datos económicos como los recursos humanos en las empresas, sin embargo, a pesar que maneja las vacaciones de los trabajadores, no concibe el pago de salario por este concepto, por tanto no se adapta a las características reales del pago salarial en cuba.

1.3.5. ASSETS NS 2.0

Es un Sistema de gestión integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos fijos, Útiles y herramientas y Recursos humanos. (ASSETS, 2005)

El módulo Recursos humanos de ASSETS NS está concebido para calcular las nóminas y controlar los recursos laborales de una entidad.

Esta aplicación está desarrollada en Visual Basic 6.0 y Microsoft SQL Server 2000.

***Observación**:* Este sistema está bastante adaptado a las necesidades del país. Como Sistema integral todos sus módulos trabajan en estrecha relación, generando automáticamente al módulo de Contabilidad, los Comprobantes de operaciones por cada una de las transacciones efectuadas. Dentro del módulo de Capital humano gestiona el proceso de las vacaciones, pero tiene en contra que está desarrollado con tecnología privativa.

1.4. Valoración del estudio del estado del arte

Después de realizar un estudio a los sistemas informáticos antes mencionados donde se tuvieron en cuenta no sólo los aspectos fundamentales desde el punto de vista del software sino también del producto, se evidencia la inexistencia de un software que responda cabalmente a lo que en realidad necesita la economía cubana, presentan diferentes inconvenientes ya sea por las herramientas sobre las que están soportadas, por las licencias de dichos productos o bien porque no son capaces de cumplir con los requisitos establecidos a nivel nacional. Se denota la importancia de la implementación del componente Vacaciones para el Sistema de Gestión Integral Cedrux, para suplir la necesidad de este para gestionar sus funciones con la nómina de forma correcta y eficiente, sería entonces factible que el proyecto y el país en el proceso de lograr su soberanía tecnológica trabajaran sobre herramientas y tecnologías que no fueran propietarias, para desarrollar un sistema de este tipo con el fin de lograr un alto grado de eficiencia, confiabilidad y rapidez en la gestión de las empresas.

1.5. Modelo para el desarrollo del software

La propuesta de modelo de desarrollo que a continuación aparece fue elaborada por el equipo de producción en colaboración con las Líneas de desarrollo del proyecto ERP-Cuba de acuerdo con las necesidades presentadas por cada una de ellas y donde se tuvieron en cuenta los principales riesgos con los que se cuentan en el proyecto. (Fernández, 2010)

1.5.1. Características

- **Orientado a componentes**

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

- **Iterativo e incremental**

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

- **Centrado en la arquitectura**

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

- **Ágil y adaptable al cambio**

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

1.6. Tendencias y tecnologías actuales

La industria del software ha mostrado ser una de las áreas más dinámicas y con mayor crecimiento en los últimos años. La evolución hacia a un modelo más racional para los usuarios, con menos costes de licencia, donde se intensifique la prestación de servicios, que reduzca el tiempo de desarrollo e incremente la calidad, viene siendo lo más importante a la hora de desarrollar cualquier tipo de aplicación.

1.6.1. Enfoque Orientado a Objetos (OO)

El enfoque Orientado a Objetos actualmente se encuentra en una etapa de madurez como paradigma del desarrollo de sistemas de información. Ayuda a explotar el poder expresivo de todos los lenguajes de programación basados en objetos y los orientados a objetos, como: java y php 5, apoya la reutilización no sólo del software, sino de diseños completos, produce sistemas que están construidos en formas intermedias estables y por ello son más resistentes al cambio en especificaciones y tecnología, y brinda un mecanismo para formalizar el modelo de la realidad.

1.6.2. Aplicación Web

La creación de aplicaciones de este tipo ha tomado gran auge debido a las funcionalidades que ofrece; algunas de estas se presentarán a continuación:

- ✓ Compatibilidad Multiplataforma: Una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows y Linux.
- ✓ Menos requerimientos de hardware: Este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor donde reside la aplicación.
- ✓ Actualización: Una de las características fundamentales de las aplicaciones web es que siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones ni realizar tareas de instalación.
- ✓ Acceso inmediato y desde cualquier lugar: Las aplicaciones basadas en tecnología web no necesitan ser descargadas, instaladas o configuradas. Además, pueden ser accedidas desde cualquier computadora conectada a la red en donde se accede a la aplicación.
- ✓ Seguridad en los datos: Los datos se alojan en servidores con sistemas de almacenamiento altamente fiables y se ven libres de problemas que comúnmente sufren los ordenadores de usuarios comunes como virus y roturas de disco. (Alvarez, 2009)

1.6.3. Arquitectura Cliente/Servidor

La arquitectura cliente/servidor no es más que un sistema distribuido entre múltiples procesadores donde hay clientes que solicitan servicios y servidores que los proporcionan. La mayoría de las aplicaciones que se están desarrollando actualmente en la industria de software utilizan este tipo de arquitectura debido a las funcionalidades que brindan, entre ellas:

- ✓ Permite integrar y compartir información entre sistemas diferentes sin necesidad de que todos tengan que utilizar el mismo sistema operativo.
- ✓ Posibilita el mantenimiento y el desarrollo rápido de aplicaciones. (Alvarez, 2007)

La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones. Se puede asociar el uso de esta arquitectura con los principios de Arquitectura Orientada a Servicios (SOA[4]).

1.6.4. Software libre

El software libre; extendido prácticamente en el mundo; permite crear soluciones cada vez más sofisticadas y más personalizadas debido a la libertad de los usuarios para ejecutar, copiar, distribuir y mejorar el software, tiene entre sus ventajas fundamentales:

- ✓ Independencia tecnológica: El acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero.
- ✓ Libertad de uso y redistribución: Las licencias de software libre existentes permiten la instalación del software tantas veces y en tantas máquinas como el usuario desee. (GNU, 2009)

De ahí la repercusión que tiene en el ámbito informático hoy día, siendo una tendencia que va madurando cada vez más.

1.7. Lenguajes de desarrollo y de modelado

1.7.1. Lenguaje de modelado

Llamamos lenguaje de modelado de objetos a un conjunto estandarizado de símbolos y de modos de disponerlos para modelar (parte de) un diseño de software.

- ✓ **UML**

Es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Probablemente, una de las innovaciones conceptuales en el mundo tecnológico del desarrollo de software que más expectativas y entusiasmos haya generado en muchos años. Es un estándar en la industria del software, creado por Grady Booch, James Rumbaugh e Ivar Jacobson.

[4] Conjunto de servicios tanto de negocio como tecnológicos que interactuando entre ellos, proporcionan la lógica necesaria para construir aplicaciones de una manera rápida y cumpliendo siempre con los principios de la Orientación a Servicios. Además SOA proporciona una serie de guías y recomendaciones para conseguir los objetivos que se impone una organización a la hora de desarrollar aplicaciones. (Coello, 2002)

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos se crea una documentación que cualquier desarrollador con conocimientos de UML será capaz de entender. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro rama. (HtmlCastellano., 2009)

Se estará haciendo uso del UML 2.0

1.7.2. Lenguajes de programación

El término lenguaje de programación está dado por un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen la estructura, el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias.

1.7.2.1. Lenguaje del lado del servidor

Se clasifica así al lenguaje de programación en la tecnología cliente servidor que se ejecuta del lado del servidor y del cual los usuarios solo obtienen el beneficio del procesamiento de la información.

- **Lenguaje php**

Php es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. Es un potente lenguaje de secuencia de comandos diseñado específicamente para permitir a los programadores crear aplicaciones en la web con distintas prestaciones de forma rápida. No requiere definición de tipos de variables, no es un lenguaje de marcas. Su interpretación y ejecución se realiza en el servidor en el cual se encuentra almacenada la página y el cliente sólo recibe el resultado de la ejecución. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle, ODBC, Microsoft SQL Server, entre otras, lo cual permite la creación de Aplicaciones web muy robustas.

Php tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux, Windows y Mac OS X, y puede interactuar con los servidores de web más populares. (Rasmus Lerdorf, 2006)

Ventajas

- ✓ Es un lenguaje multiplataforma.

- ✓ Capacidad de expandir su potencial utilizando gran cantidad de módulos.
- ✓ Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- ✓ Es libre, lo que representa que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.
- ✓ Permite las técnicas de Programación Orientada a Objetos[5].
- ✓ Biblioteca nativa de funciones sumamente amplia e incluida.

Desventajas

- ✓ No posee una abstracción de base de datos estándar, sino bibliotecas especializadas.
- ✓ Por sus características promueve la creación de código desordenado y complejo de mantener.
- ✓ Todo el trabajo lo realiza el servidor, por tanto puede ser más ineficiente a medida que aumenten las solicitudes.
- ✓ La orientación a objetos es aún muy deficiente para aplicaciones grandes.

En este caso se estará haciendo uso de php 5.2.6.

1.7.2.2. Lenguajes del lado del cliente

Un lenguaje del lado cliente es totalmente independiente del servidor, y en el caso de una aplicación web esto permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y esto puede afectar la seguridad. A continuación se describen algunos de estos lenguajes.

- **HTML[6]**

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un

[5] La terminología Programación Orientada a Objetos (POO u OOP según siglas en inglés) representa un paradigma de programación donde se definen los programas en términos de "clases de objetos", tales objetos son entidades que combinan estado (datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto).

[6] Acrónimo inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web. HTML es una aplicación de Lenguajes de Marcas Generalizados (*Standard Generalized Markup Language* SGML por sus siglas en inglés) conforme al estándar internacional ISO 8879.

script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. (XPPS, 2009)

En un documento HTML se pueden ver tres partes bien diferenciadas:

- ✓ Una cabecera del tipo de documento, en ella se especifica el tipo de documento HTML.
- ✓ Una cabecera de documento, donde se especifica información del documento.
- ✓ El cuerpo del documento, donde se coloca toda la información que contiene la página HTML4.0.

- **XHTML**

XHTML es el acrónimo en inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto). Es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos." (sistemas, 2007)

XHTML reúne la capacidad de formato de HTML y se consolida con la formalidad del XML a la hora de estructurar documentos para la portación de datos. Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. Algunos de estos requisitos son:

1. Elementos correctamente anidados
1. Etiquetas en minúsculas
2. Elementos cerrados correctamente
3. Atributos[7] de valores entrecomillados.

- **CSS**

Es un lenguaje de hojas de estilos (Cascading Style Sheets) creado para controlar la presentación de documentos estructurados y escritos en HTML y XHTML, 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. (SOA, 2006)

- **XML**

[7] Contenedor de un tipo de datos asociados a un objeto, que hace los datos visibles desde fuera del objeto, y cuyo valor puede ser alterado por la ejecución de algún método.

Es el metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos. Por lo tanto el Lenguaje de Marcas Extensible (XML) no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas.

XML se desarrolló para proporcionar una flexibilidad y consistencia que no se podían alcanzar con HTML, no sólo es un lenguaje de marcado, sino también un metalenguaje cuya particularidad más importante es que no posee etiquetas prefijadas con anterioridad, permitiendo describir otros lenguajes de marcado y definir lenguajes de presentación propios en dependencia del contenido del documento. (W3C, 2011)

- **JavaScript**

Java Script es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM[8]. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape y Mozilla Firefox. (Vega, 2009)

1.8. Frameworks

Un framework o marco de trabajo, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

El desarrollo de la solución se realizará haciendo uso del marco de trabajo Sauxe, implementado por el Departamento de Tecnología del Centro de Informatización de la Gestión de Entidades (CEIGE).

[8] DOM: El Document Object Model (Modelo de Objetos de Documento), Es una plataforma que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos HTML y XML, proporcionando una interfaz estándar para que otro software manipule los documentos.

1.8.1. ExtJS

Posee una librería inmensa que permite configurar las interfaces Web de manera semejante a aplicaciones desktop, incluye la mayoría de los controles de los formularios Web basándose en Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería contiene componentes para el manejo de datos, lectura de XML, lectura de datos JSON[9] e implementaciones basadas en AJAX.
Se estará haciendo uso de ExtJS. 2.2.

1.8.2. Zend Framework

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con php. Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernos, robustos y seguros sitios web, está diseñado para php 5 y posee buenas capacidades de ampliación. Presenta entre otras, las siguientes características:

- ✓ Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos.
- ✓ Proporcionan los componentes que forma la infraestructura del patrón MVC.
- ✓ Proporciona una capa de acceso a base de datos, construida sobre PDO[10] pero ampliándola con diferentes características.
- ✓ Proporciona mecanismos de filtrado y validación de entradas de datos.
- ✓ Permite convertir estructuras de datos php a JSON y viceversa, para su utilización en aplicaciones AJAX.
- ✓ Proporciona capacidades de búsqueda sobre documentos y contenidos. (ZEND FRAMEWORK, 2010).

Se estará haciendo uso de Zend Framework 1.9.7.

1.8.3. Doctrine

Doctrine es un potente y completo sistema ORM[11] (en inglés Object Relational Mapper) para php 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO), debido a que doctrine

[9] "JavaScript Object Notation", es un formato ligero para el intercambio de datos.
[10] PDO: (en inglés: PHP Data Objects) es una interface de acceso a datos que permite la conexión a diferentes bases de datos utilizando tecnología orientada a objetos.
[11] ORM: Componente de software que permite trabajar con los datos persistidos como si fueran parte de una base de datos orientada a objetos.

utiliza el patrón Active Record[12] para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente. Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Funcionalidades:

1- Exporta una base de datos existente a sus clases correspondientes.
2- Convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (Doctrine, 2008).

Se estará haciendo uso de Doctrine en su versión 1.2.1

1.8.4. Zend_Ext Framework

Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizada por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor, se le incluyó el IoC[13] para la comunicación en entre los módulos o componentes, la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos, el ExtJS Framework para el desarrollo de las vistas y un controlador de trazas para controlar las acciones del sistema (acción, excepciones, rendimiento, integración, y excepción de integración).

1.8.5. UCID Framework

Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJS Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación y el multilenguaje.

1.9. Tecnologías y herramientas de desarrollo

Para la realización de un proyecto de esta magnitud es necesario que cada uno de los equipos de desarrollo posea un modelo estandarizado de las tecnologías y herramientas a utilizar conjuntamente con sus versiones para la implementación de las capas de presentación, negocio y acceso a datos. De acuerdo con lo planteado anteriormente la dirección del proyecto determinó emplear:

[12] El patrón “Active Record es una extensión del patrón Domain Model (”Modelo de Dominio”), que se entiende como una clase o un grupo de clases que representan a “objetos” o responsabilidades particulares en la aplicación”. (Almada, 2009)

[13] Inversión de control (Inversion of Control, IoC por sus siglas en inglés). Es un método de programación en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones.

1.9.1. Tecnología

- **Tecnología AJAX**

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. (Doctrine, 2008)

1.9.2. Herramienta CASE

CASE son las siglas correspondientes a Computer Aided Software Engineering, que en su traducción al español significa Ingeniería de Software Asistida por Computadoras. Las herramientas CASE son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Están destinadas a:

1. Mejorar la productividad en el mantenimiento y desarrollo del software.
2. Automatizar el desarrollo del software, generación de código, pruebas de errores y gestión del proyecto.
3. Mejorar el tiempo y costo de desarrollo y mantenimiento de los sistemas informáticos.
4. Aumentar la calidad del software.
5. Facilitar el uso de las distintas metodologías propias de la ingeniería del software

- **Visual Paradigm**

Es una herramienta CASE de modelado que utiliza UML como lenguaje de modelado profesional y que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa. La herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como HTML, Pdf y permite control de versiones. Dentro de sus características principales se puede destacar su robustez, usabilidad y portabilidad. (Ecured, 2011).

Para el desarrollo de este trabajo se escogió como herramienta CASE de modelado al Visual Paradigm porque tiene un coste favorable y realiza de una forma íntegra los planes de construcción del software. Permite ingeniería inversa, del modelo físico se puede llegar al modelo lógico. Genera de forma automática códigos desde diagramas, además brinda la posibilidad de generar documentación.
Se estará haciendo uso de Visual Paradigm en su versión 3.4.

1.9.3. Herramienta de desarrollo colaborativo

- **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. Un sistema de control de versiones debe proporcionar un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos. Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

1.9.3.1. Subversion

También conocido como SVN, es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red y se distribuye bajo licencia libre.

- ✓ Mantiene versiones no sólo de archivos, sino también de directorios
- ✓ Mantienen versiones de los metadatos asociados a los directorios.
- ✓ Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- ✓ Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- ✓ Soporte tanto de ficheros de texto como de binarios.

- ✓ Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (UBUNTU, 2009)

1.9.3.2. RapidSVN

RapidSVN es un cliente gráfico de Subversión multiplataforma. Que se distribuye bajo la Licencia Pública General de GNU[14].

Características:

- **Simple** - proporciona una interfaz fácil de usar para las características de Subversion.
- **Eficiente** - simple para los principiantes pero lo suficientemente flexible como para aumentar la productividad para los usuarios de Subversion con experiencia.
- **Portable** - se ejecuta en cualquier plataforma en la que Subversion y wxWidgets puede ejecutar: Linux, Windows, Mac OS / X, Solaris, etc.
- **Rápido** - completamente escrito en C + +.
- **Multilingüe** - que ha sido traducido a muchos idiomas ya: alemán, francés, italiano, portugués, ruso, ucraniano, chino simplificado, japonés.
- **Soporte** completo para Unicode. (RapidSVN, 2011)

1.9.4. Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado o IDE (en inglés: Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación que permite de forma cómoda y ágil editar, compilar, ejecutar y depurar programas.

1.9.4.1. NetBeans

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 onRails, Groovy and Grails y C/C++. (Netbeans, 2011).

Se estará haciendo uso del NetBeans IDE 7.01

1.9.5. Servidor de Aplicaciones web

Es un programa que permite crear un servidor http en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Es además un programa que implementa el protocolo HTTP el cual se encarga

[14] Acrónimo recursivo que significa No es Unix.

de transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

- ***Servidor Web Apache***

Apache 2.0 es un servidor Web potente, flexible y disponible para distintas plataformas y entornos. Es altamente configurable y de diseño modular, posibilitando que los administradores de sitios Web puedan elegir los módulos que serán incluidos y ejecutados en el servidor.

Características de Apache:

Es una tecnología gratuita y de código abierto, lo que proporciona transparencia en todo el proceso de instalación.

Es prácticamente universal, por su disponibilidad en multitud de sistemas operativos.

Es altamente configurable en la creación y gestión de logs, de este modo es posible tener un mayor control sobre lo que sucede en el servidor. (Apache, 2010)

Este servidor Web tiene una fácil integración con varios lenguajes de programación como: Java, Perl y especialmente php. Dicha relación a dado lugar el desarrollo de aplicaciones como el APPSERV y XAMPP los cuales instalan el Apache y el php configurados para su uso.

Se estará haciendo uso del servidor web Apache 2.0 o superior.

1.9.6. Sistemas de gestión de base de datos

- ***Base de Datos (BD)***

Conjunto de datos persistentes, interrelacionados entre sí de forma lógica, que representa una situación del mundo real.

- ***Gestor de Base de Datos***

Un Sistema Gestor o Manejador de Bases de Datos (SGBD) es un conjunto de programas que permite a los usuarios crear y mantener una BD, por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones, asegurando su integridad, confidencialidad y seguridad. Pueden ser de propósito general o específico. (UCI, 2007-2008). Es difícil pensar en una aplicación tanto desktop como Web que no esté conectada a una base de datos. Es que precisamente los sistemas de base de datos brindan una serie de ventajas y funcionalidades a las aplicaciones. Existen diversas formas de manejar dichas bases de datos: con gestores como Oracle, SQL Server, MySql, entre otros.

1.9.6.1. PostgreSQL 8.3

PostgreSQL es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se

destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha herramienta es multiplataforma. Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos. (PostgreSQL, 2010)

Se estará haciendo uso del PostgreSQL en su versión 8.3.

1.9.6.2. PGAdmin III

"Es una aplicación gráfica usada para la gestión de PostgreSQL, siendo la más completa y popular con licencia Open Source[15]. PGAdmin está escrito en C++ y utiliza la librería gráfica multiplataforma wxWidgets[16], permitiendo que se pueda usar en sistemas operativos como: GNU/Linux[17], MacOS y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3, ejecutándose en cualquier plataforma. PGAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente su administración."

1.9.7. Navegador Web

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden. Permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web.

1.9.7.1. Mozilla Firefox

Mozilla Firefox es el nuevo e innovador navegador open source. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. Se trata de un práctico y ágil navegador, que está en

[15] Open Source: Código abierto, es el término con el que se conoce al software distribuido y desarrollado libremente.

[16] Son unas bibliotecas multiplataforma y libres, para el desarrollo de interfaces gráficas programadas en lenguaje C++. Están publicadas bajo una licencia LGPL, similar a la GPL con la excepción de que el código binario producido por el usuario a partir de ellas, puede ser propietario, permitiendo desarrollar aplicaciones empresariales sin coste.

[17] GNU/Linux: Es el término empleado para referirse al sistema operativo Unix-like que utiliza como base las herramientas de sistema de GNU y el núcleo Linux.

renovación constante. Tiene la capacidad de modificarlo totalmente a gusto del usuario y según las necesidades del mismo. Esto se consigue gracias a la multitud de "extensiones" que existen, y que cada día aparecen más, que permiten añadirle nuevas funciones de todo tipo. (Mozilla, 2011)
Se estará haciendo uso de Mozilla Firefox 3.0.

1.10. Conclusiones del capítulo

La gestión de las vacaciones pagadas es un proceso fundamental para la planificación y la organización de los recursos empresariales en las entidades cubanas; este capítulo fue esencial para valorar el estado actual de este proceso a partir de un estudio de sistemas tanto nacionales como internacionales, evidenciando la no existencia de un sistema informático capaz de ejecutar tales funcionalidades ni de cumplir con los requisitos establecidos a nivel nacional. Este capítulo fue básico para la definición de un conjunto de conceptos fundamentales asociados al dominio del problema. Se explicó el modelo de desarrollo a utilizar. Se analizaron las tendencias y tecnologías actuales para el desarrollo de aplicaciones informáticas y por último se realizó una presentación de las tecnologías y herramientas conjuntamente con sus versiones, propuestas por la dirección del proyecto para el desarrollo de la solución.

Capítulo 2: Diseño e implementación.

2.1. Introducción

En el presente capítulo se especifican los patrones de arquitectura y del diseño del sistema, se modela la solución a partir de los requisitos previamente definidos, obteniendo un grupo de artefactos que tienen un valor significativo en la etapa de implementación, tales como: los modelos de clases y datos, los diagramas de componentes y despliegue, con la descripción de cada uno de ellos. Se describen las clases y sus métodos principales, se describe los estándares de código utilizados en la implementación, así como la estrategia de integración y los servicios que interactúan con el componente.

2.2. Diseño de la solución:

2.2.1. Valoración crítica de la especificación de requisitos

Los artefactos de Especificación de requisitos del software generados como resultado del previo análisis efectuado al proceso Vacaciones, constituyen un elemento clave para el diseño y la posterior implementación. Facilitan el entendimiento de los procesos a implementar permitiendo una buena comprensión del problema y posibilitando la identificación de las clases y las funcionalidades a desarrollar. Se determinaron un total de 19 requisitos funcionales, identificados en 3 agrupaciones según el objetivo al que respondían.

Plan de Vacaciones:

- ✓ Adicionar plan de vacaciones.
- ✓ Modificar plan de vacaciones.
- ✓ Eliminar plan de vacaciones.
- ✓ Confirmar plan de vacaciones.
- ✓ Listar plan de vacaciones.
- ✓ Listar plan de vacaciones por área.
- ✓ Listar plan de vacaciones por año.

Submayor de vacaciones:

- ✓ Listar submayor de vacaciones.
- ✓ Listar submayor de vacaciones por área.
- ✓ Listar submayor de vacaciones por centro de costo.
- ✓ Detalles de un trabajador.

Notificación de vacaciones:

- ✓ Adicionar notificación de vacaciones.

- ✓ Modificar notificación de vacaciones.
- ✓ Eliminar notificación de vacaciones.
- ✓ Listar notificación de vacaciones.
- ✓ Listar notificación de vacaciones por área.
- ✓ Listar notificación de vacaciones por período de pago.
- ✓ Confirmar notificación de vacaciones.
- ✓ Importar notificación de vacaciones.

2.2.2. Patrones

Un patrón es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio, son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además facilitan la comunicación entre diseñadores, pues establecen un marco de referencia. (Larman, 1999)

2.2.2.1. Patrón de arquitectura

Un elemento clave en el desarrollo del proceso de software es el diseño de la arquitectura, sobre ella se apoyan todas las representaciones de la estructura general de la aplicación a desarrollar, la misma es la que le da forma al software para que soporte todos los requisitos y establece los fundamentos para que todos los involucrados en el equipo de desarrollo trabajen en una línea común que permita alcanzar los objetivos y necesidades del sistema.

La arquitectura proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por el programa. (Larman, 1999)

2.2.2.2. Modelo vista controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Su principal finalidad es mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos.

De ésta manera se desarrolla usando la librería ExtJS para la vista, Zend Framework para el controlador y el Doctrine para el modelo.

2.2.2.3. Patrones de diseño empleados:

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema del diseño.

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas de software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.
- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

2.2.2.4. Patrones GRASP

Los patrones "GRASP" se utilizan con el objetivo de asignar responsabilidades a las diferentes clases que se definen en el diseño. Dentro de este grupo de patrones se utilizaron: experto, creador, controlador, bajo acoplamiento y alta cohesión.

- ✓ Experto: Se pone en práctica con el uso de clases que poseen responsabilidades específicas a cumplir, de acuerdo con la información que manejan. El componente cuenta con clases controladoras, modelos y de entidad que poseen funciones concretas de acuerdo con la información que gestionan.
- ✓ Bajo acoplamiento: Consiste en tener las clases lo menos relacionadas entre sí, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás, potenciando la reutilización y disminuyendo la dependencia entre las clases.
- ✓ Alta cohesión: Este patrón determina, que la información almacenada en una clase debe ser coherente y estar relacionada con esta en mayor medida, enfocada en sus responsabilidades. Realizando un diseño donde las clases mantengan una alta cohesión, es posible ganar en claridad y facilidad a la hora de entender el diseño, además de simplificar el mantenimiento y soportar mayor capacidad de reutilización.
- ✓ Creador: El patrón de creación hizo posible que el diseño pudiera soportar bajo acoplamiento, encapsulación y reutilización, así como mayor claridad. Las clases controladoras son responsables de crear el objeto de las modelos y estas a su vez de las entidades.

La Arquitectura Base y el diseño flexible y escalable a través del correcto uso de patrones de diseño en la generación de los artefactos necesarios para el desarrollo, posibilitaron crear una entrada apropiada como punto de partida a las actividades de implementación, con la máxima de lograr una mayor calidad del producto y la satisfacción del cliente.

2.2.3. Diagramas de clases del diseño.

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

- Clases asociadas.
- Métodos
- Navegabilidad
- Dependencias

En la figura 1 se muestra el diagrama de clases del diseño del escenario Plan de vacaciones, donde se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a los requisitos funcionales relacionados con la gestión de los datos de la nómina hasta su procesamiento. Las clases *planvacaciones.js* y *planvacaciones.phtml* conforman la capa arquitectónica de presentación. La clase *PlanvacacionesController* solo maneja la comunicación entre la vista y el modelo, las clases *Model* son las encargadas de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento de los requisitos identificados; encargadas del acceso a los datos se encuentran *DatPlanvacaciones*, *DatPeriodosvac* y todas las clases Base de las cuales extienden las mencionadas.

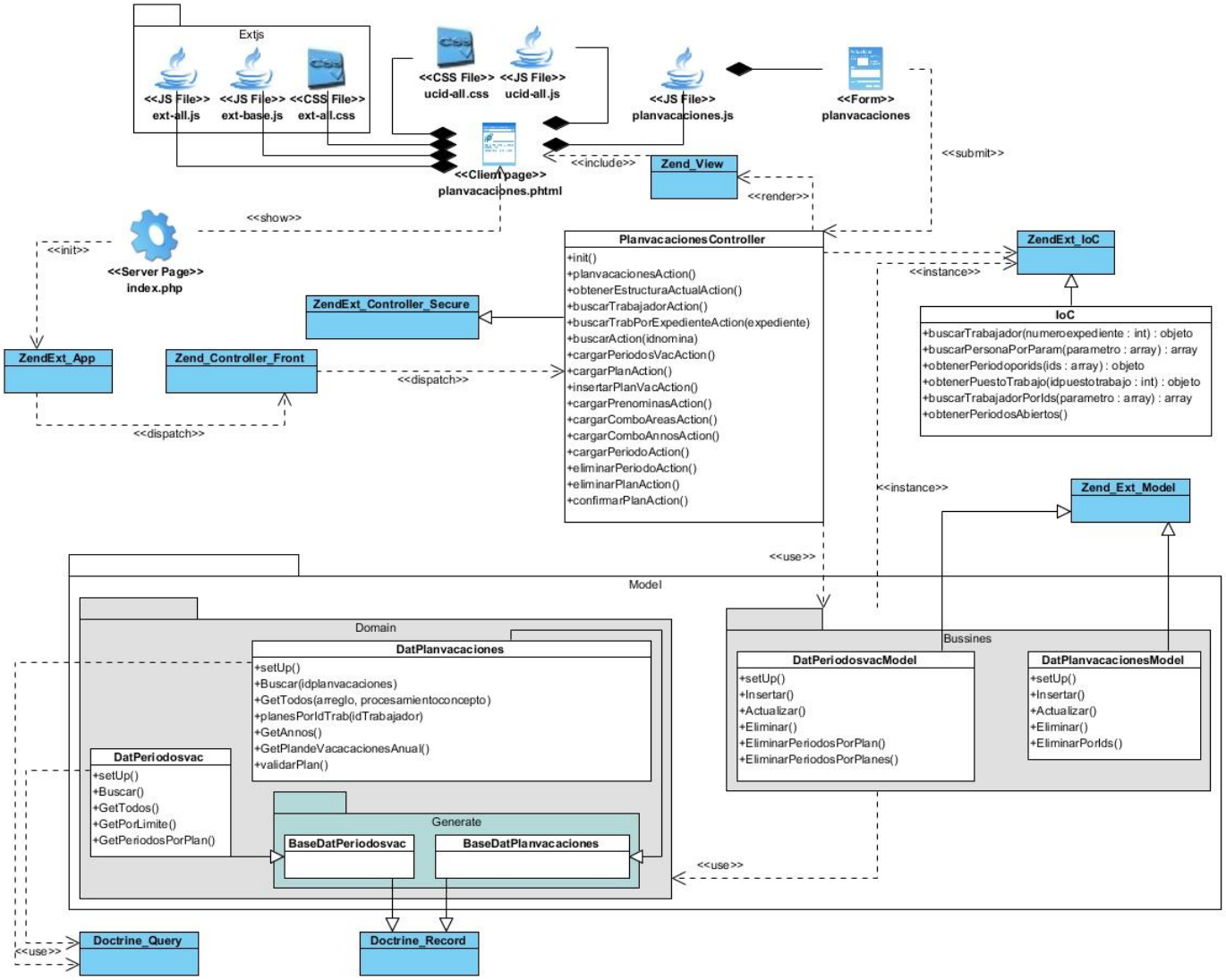

Fig.1 Diagrama de clase Plan de vacaciones.

En la figura 2 se muestra el diagrama de clases del diseño del escenario Submayor de vacaciones, donde se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a los requisitos funcionales relacionados con la gestión de los datos de la nómina hasta su procesamiento. Las clases *submayorvacaciones.js* y *submayorvacaciones.phtml* conforman la capa arquitectónica de presentación. La clase *SubmayorvacacionesController* solo maneja la comunicación entre la vista y el modelo, la clase *DatSubmayorvacModel* es la encargada de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento de los requisitos identificados; encargadas del acceso a los datos se encuentra *DatSubmayorvac,* así como la clase *BaseDatSubmayorvac* de la cual extiende.

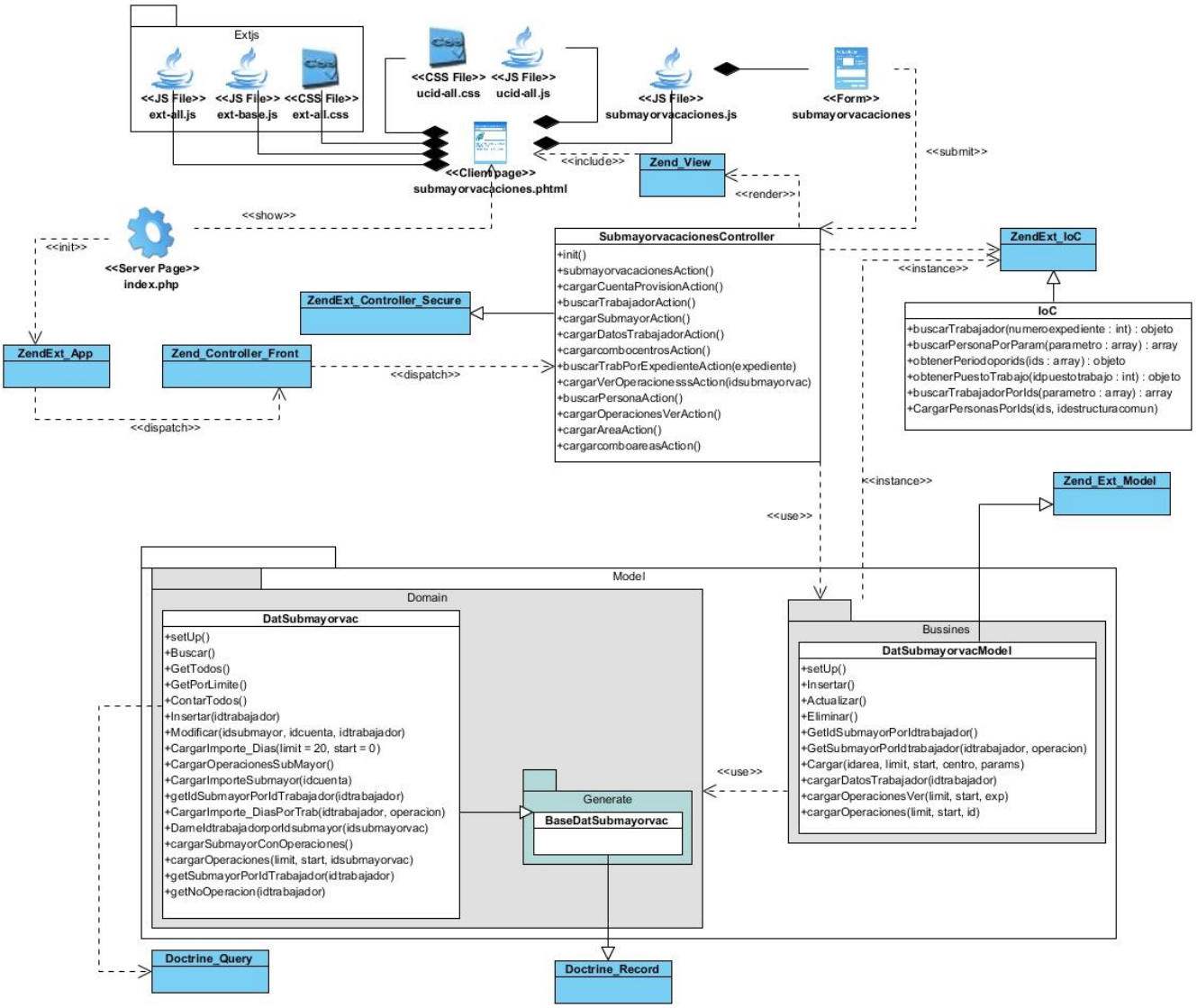

Fig. 2 Diagrama de clase Submayor de vacaciones.

En la figura 3 se muestra el diagrama de clases del diseño del escenario Notificar vacaciones, donde se representan las principales clases, operaciones y relaciones que se necesitan para darle cumplimiento a estos requerimientos, donde la capa arquitectónica de presentación está formada por las clases *notificarvacaciones.js* y *notificarvacaciones.phtml*. La comunicación entre la vista y el modelo es realizada por la clase *NotificarvacacionesController*, la clase *DatNotificacionvacModel* es la encargada de la lógica del negocio, implementando funcionalidades que garantizan el cumplimiento de los requisitos identificados, encargadas del acceso a los datos se encuentra *DatNotificacionvac,* así como la clase *BaseDatNotificacionvac* de la cual extiende.

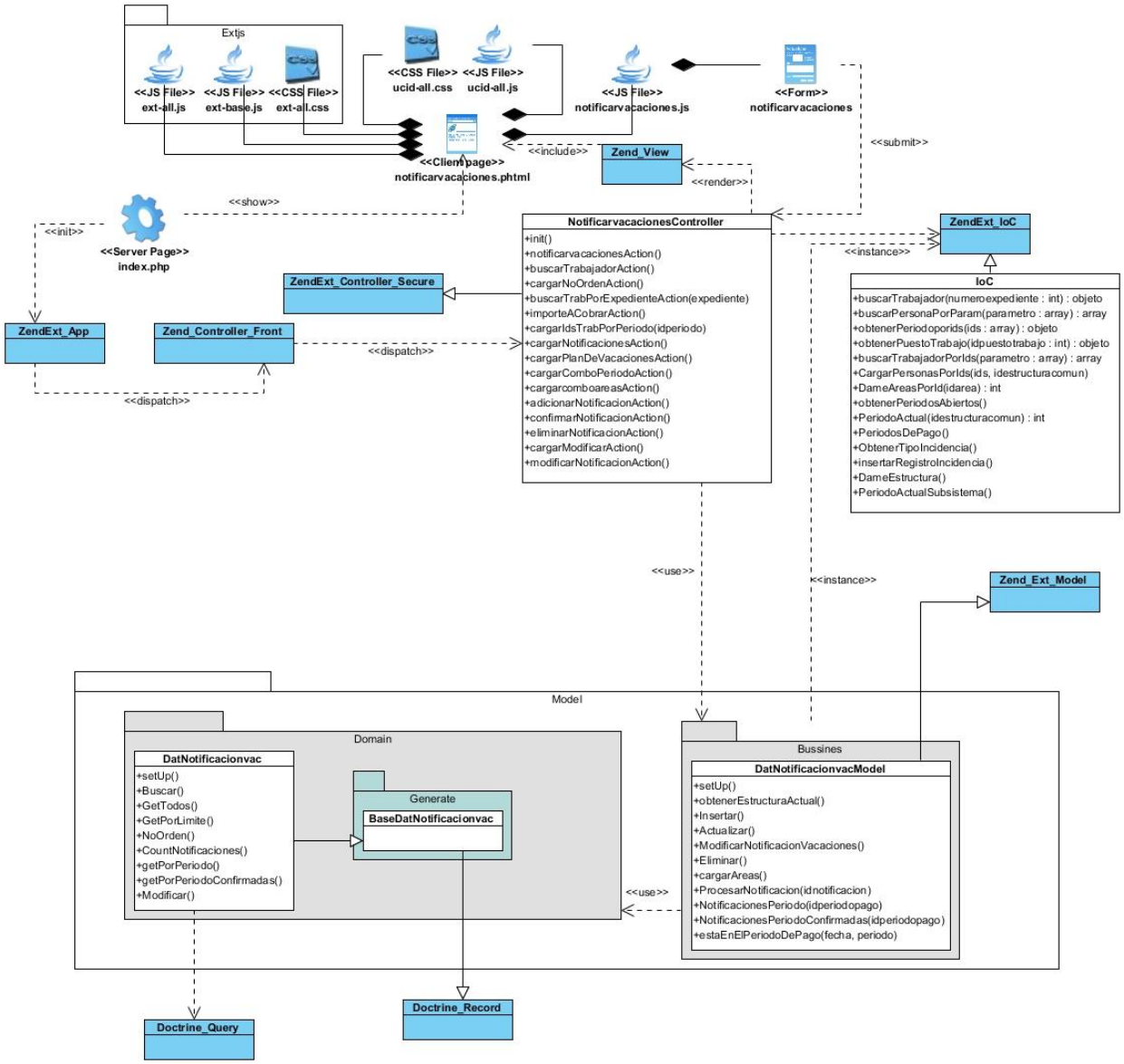

Fig. 3 Diagrama de clase Notificar vacaciones.

2.2.4. Modelo de datos

Una vez obtenido el diseño de todas las clases se definió el Modelo de datos para representar todas las clases persistentes necesarias para acceder a la Base de datos.

El objetivo de construir un modelo de datos es identificar y representar las tablas (entidades) de importancia para el funcionamiento del negocio, sus propiedades (atributos), y la forma en que estas tablas 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 9 tablas, de ellas 5 son específicas del componente vacaciones, ejemplo: la tabla “dat_notificacionvac” almacena los datos relacionados con las notificaciones de vacaciones, tales como el identificador del trabajador, las fechas de inicio, fin y reincorporación de las vacaciones, los días e importe a cobrar y el estado en que se encuentra. El estado de la notificación determina para la confección de la nómina de vacaciones. La tabla “dat_submayorvac” almacena los identificadores de los trabajadores que se encuentren en el submayor de vacaciones. La tabla “dat_operacionessubvac” es la encargada de registrar los datos de las operaciones efectuadas por cada submayor de vacaciones. En la tabla “dat_planvacaciones” es donde persisten los planes de vacaciones de los trabajadores, donde de estos se registra el estado en que se encuentra y el año de elaboración. La tabla “dat_periodosvac” almacena los períodos de las vacaciones registrados en el plan, está conformada por la cantidad de días, fechas de salida y entrada.

De las tablas que se muestran en el modelo de datos, hay 4 que no pertenecen directamente al componente, pero son imprescindibles para llevar a cabo los procesos que gestiona las vacaciones, por ejemplo, a partir de los datos que persisten en la tabla “dat_nnomina” se registra la nómina a la que pertenecen las operaciones del submayor. Otro ejemplo lo constituye la tabla “nom_periodopago” donde se almacenan los períodos en el cual se enmarcan las operaciones del submayor y los períodos de vacaciones. La tabla “dat_trabajador” donde se registran los datos de los trabajadores y por último la tabla “dat_persona” donde se registran los datos necesarios para conformar un trabajador (nombre(s), apellidos, etc).

A continuación se muestra el modelo de datos obtenido:

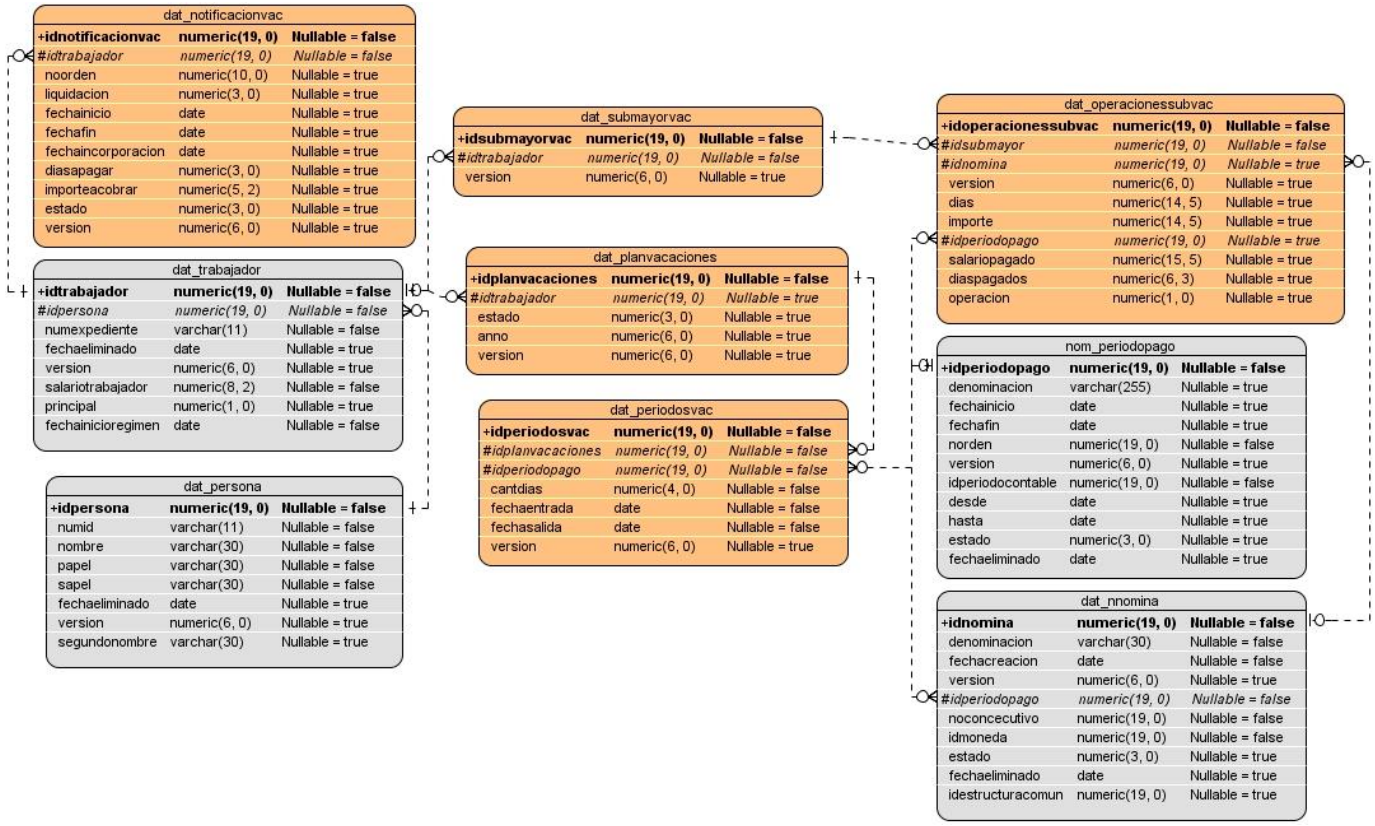

Fig. 4 Modelo de datos del componente Vacaciones.

2.3. Implementación del componente:

2.3.1. Estándares de código

Los estándares de codificación en el marco del proyecto Cedrux van a permitir una mejor integración entre las líneas de producción y se establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que nos permita un mejor entendimiento por parte del equipo de desarrollo y de soporte, el componente se rige por estos estándares, pues son una guía para el desarrollo y desde el punto de vista arquitectónico estandarizan el código a implementar. (Cañete Pupo, 2010)

Dentro de los estándares utilizados se encuentran:

2.3.1.1. PascalCasing

El estándar PascalCasing establece que los nombres de clases están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula.

La nomenclatura de las clases del componente vacaciones anuales pagadas se realizó sobre la base de este estándar, usando palabras compuestas sugerentes acordes al propósito de la misma.

Nomenclatura de clases según su tipo:

- **Controllers**: clases controladoras del negocio.

El nombre de la clase controladora debe estar estructurado por el nombre propio de la misma en mayúsculas seguido por la palabra Controller y heredar siempre de la súper clase del frameworkZendExt_Controller_Secure. Ejemplo: PlanvacacionesController.

Clases modelos

- **Business**: Clases modelo del negocio.

Las clases modelo tendrán por identificador el nombre de la tabla en la que trabajan seguido por la palabra Model y heredarán de la clase del framework Zend_Ext llamada ZendExt_Model, pueden incluir los prefijos Dat o Nom para diferenciar entre sus usos. Ejemplo: DatPlanvacacionesModel.

- **Domain**: clases entidades del dominio.

Los archivos situados en el domain tienen el mismo nombre de las tablas que representan, definiéndolas como clases php, pueden incluir los prefijos Dat o Nom para diferenciar entre sus usos. Ejemplo: DatPlanvacaciones.

- ***Generated***: Clases bases del dominio

Las clases que se encuentran dentro de Generated comienza su nombre con la palabra: “Base”, seguido del nombre de la tabla en la Base de Datos. Ejemplo: BaseDatPlanvacaciones.

- ***Services***: Clases que ofrecen los servicios de los componentes.

Estas clases, de acuerdo con las operaciones que realizan y prestaciones que brindan, se definen con calificativos sugerentes, agregándoles la terminación Service. Ejemplo: PlanvacacionesService.

2.3.1.2. CamelCasing

El estándar CamelCasing es parecido al PascalCasing con la particularidad de que la letra inicial del identificador no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones.

Nomenclatura de las funciones

El identificativo a emplear para las funciones o métodos se escribe con la primera palabra en minúscula utilizando la notación CamelCasing y nombres que deduzcan su propósito. Ejemplo: insertar.

Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: insertarPlanVacAction.

2.3.2. Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. En la figura 5 se muestra el diagrama de componentes que representa las relaciones de las Vacaciones anuales pagadas con otros componentes

del subsistema capital humano y del resto de los subsistemas del Sistema Integral de Gestión Cedrux y los servicios que utiliza para obtener información de estos.

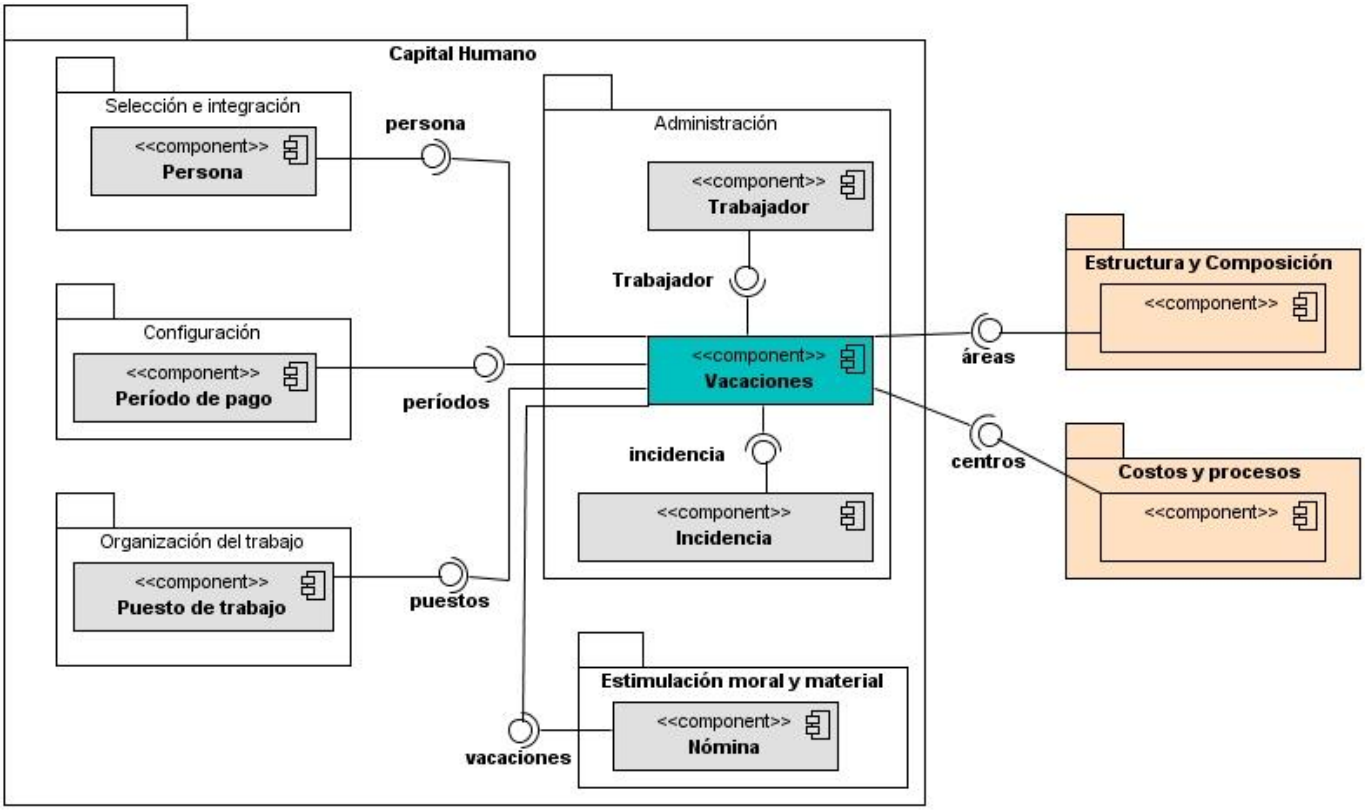

Fig. 5 Diagrama de componentes.

2.3.3. Estrategia de integración

La integración entre los componentes está definida en una arquitectura de 3 capas: presentación (view), negocio (controller) y acceso a datos (models), consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, a través de los diferentes elementos que la componen. Consta de 4 nodos de integración:

Vista - Controlador: Los datos recogidos en un formulario son enviados al Controlador haciendo uso del protocolo de comunicación HTTP a través del método “post” para ser procesados y los resultados son enviados por el controlador a la vista en un JSON a través del método “echo”.

Controlador - Modelo: El Controlador toma los datos recibidos desde la vista, instancia una determinada clase del modelo y llama a uno de sus métodos, pasándole como parámetros los datos recibidos.

Modelo - Doctrine: El Modelo utiliza llamadas a métodos de Doctrine que le permitan crear, modificar, eliminar o actualizar los datos almacenados en las tuplas de la base de datos.

Doctrine - Base de Datos: Doctrine ejecuta las consultas a la Base de Datos utilizando programación orientada a objetos.

La comunicación entre las capas dentro de un mismo componente se realiza mediante llamadas a métodos o eventos de forma directa.

Entre diferentes módulos y componentes, la integración se basa en el patrón Inversión de Control (IoC) y se realiza a través de un componente incluido en el framework Zend_Ext, que permite operar sobre distintos esquemas en la base de datos realizando las transacciones adecuadas. En dicho componente se define el fichero ioc.xml que contiene la ubicación de cada uno de los componentes y los servicios que ofrecen las clases services correspondientes. De esta manera, la puerta de entrada de cada componente es el paquete de las clases de servicios que buscan en los modelos o entidades las funcionalidades requeridas.

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

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

IoC Externo: para la integración entre subsistemas.

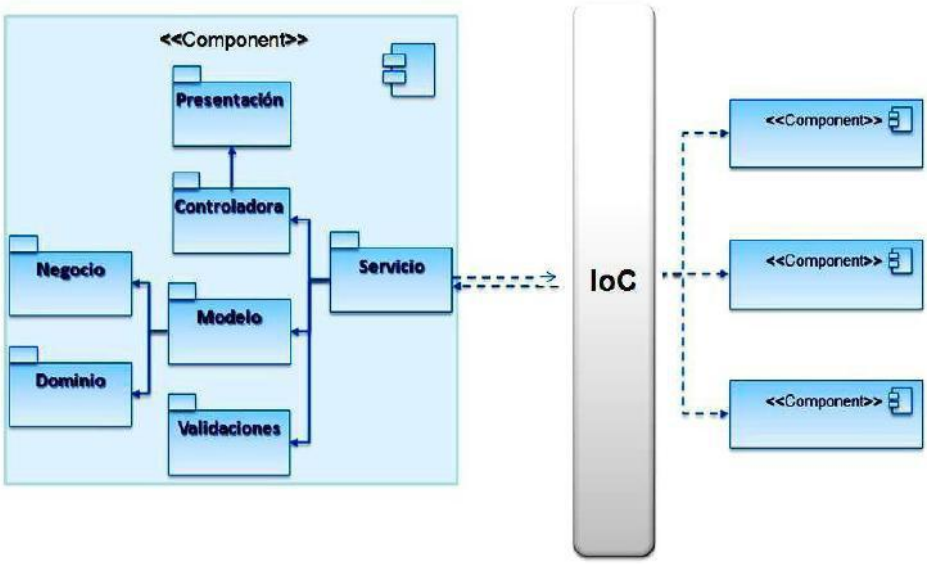

Fig. 6 Integración entre componentes.

- **Servicios que utiliza el componente como parte de la integración:**

BuscarTrabajador: Servicio que brinda el componente Trabajador del subsistema Capital Humano que le permite al componente Vacaciones obtener los datos de un trabajador mediante parámetros.

BuscarTrabajadorPorIds: Servicio que brinda el componente Trabajador del subsistema Capital Humano que le permite al componente Vacaciones obtener los datos de los trabajadores mediante los identificadores enviados por parámetros.

BuscarPersonaPorParam: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente Vacaciones obtener la persona que coincida con los parámetros determinados.

CargarPersonasPorIds: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente Vacaciones obtener las personas que coincidan con los parámetros determinados.

ObtenerPeriodoporids: Servicio que brinda el componente Período de Pago del subsistema Capital Humano que le permite al componente Vacaciones obtener los períodos de pagos que coincidan con los identificadores enviados por parámetros.

PeriodosDePago: Servicio que brinda el componente Período de Pago del subsistema Capital Humano que le permite al componente Vacaciones cargar todos los períodos de pagos que se encuentran abiertos.

PeriodoActual: Servicio que brinda el componente Período de Pago del subsistema Capital Humano que le permite al componente Vacaciones obtener los datos del período de pago actual a partir del identificador de la estructura común.

ObtenerPuestoTrabajo: Servicio que brinda el componente Puesto de Trabajo del subsistema Capital Humano que le permite al componente Vacaciones obtener los datos de los puestos de trabajos mediante parámetros.

insertarRegistroIncidencia: Servicio que brinda el componente Incidencia del subsistema Capital Humano que le permite al componente Vacaciones guardar un nuevo registro de incidencia.

DameAreasPorId: Servicio que brinda el subsistema Estructura y Composición que le permite al componente Vacaciones obtenerlos datos del área mediante el identificador.

DameEstructura: Servicio que brinda el subsistema Estructura y Composición que le permite al componente Vacaciones obtener los datos de la estructura que se está utilizando.

DameEstructurasInternas: Servicio que brinda el subsistema Estructura y Composición que le permite al componente Vacaciones obtener los datos de las áreas a partir del identificador de la estructura común.

centrosCosto: Servicio que brinda el componente Costos y procesos que le permite al componente Vacaciones obtener los datos de los centros de costos mediante el identificador de la estructura común enviada por parámetro.

PeriodoActualSubsistema: Servicio que brinda el subsistema Configuración que le permite al componente Vacaciones obtener el período contable actual del subsistema.

ObtenerPeriodo: Servicio que brinda el subsistema Configuración que le permite al componente Vacaciones obtener el período contable dado el identificador.

- **Servicios que brinda el componente como parte de la integración:**

insertarOperaciones: Servicio que brinda el componente Vacaciones que permite insertar los datos de las operaciones de la nómina de salario o vacaciones.

modificarOperacion: Servicio que brinda el componente Vacaciones que permite modificar los datos de la operación de la nómina de salario o vacaciones.

modificarnotificacionvacaciones: Servicio que brinda el componente Vacaciones que permite modificar la notificación de las vacaciones anuales acumuladas.

eliminarPlanPorIdTrab: Servicio que brinda el componente Vacaciones que permite eliminar los planes de vacaciones para un trabajador determinado.

AcumuladoDiasImporte: Servicio que brinda el componente Vacaciones que permite obtener el acumulado de días e importe para una operación de un trabajador determinado.

InsertarSubmayorVacaciones: Servicio que brinda el componente Vacaciones que permite insertar datos al submayor de vacaciones a un trabajador determinado.

NotificacionesPeriodo: Servicio que brinda el componente Vacaciones que permite obtener las notificaciones de vacaciones en el período determinado.

NotificacionesPeriodoConfirmadas: Servicio que brinda el componente Vacaciones que permite obtener las notificaciones de vacaciones confirmadas en el período determinado.

2.3.4. Diagrama de despliegue

El diagrama de despliegue es utilizado para modelar el hardware empleado en las implementaciones de sistemas y las relaciones entre sus componentes. Constituye un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. El despliegue del componente Vacaciones se rige por la especificación dada para el subsistema Capital humano. A continuación se muestran los posibles escenarios con los requerimientos de software.

Servidores

Servidor de Aplicaciones Web

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

Servidor de Base de Datos

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

Servidor de Clientes Ligeros

- Sistema Operativo: Nova Server
- Navegador Web: Mozilla Firefox 2.17 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 v2.2 ó superior
- Herramientas Ofimáticas
- Visualizador de ficheros pdf

PC Cliente sin disco duro

- Todo se instala en el servidor de clientes ligeros

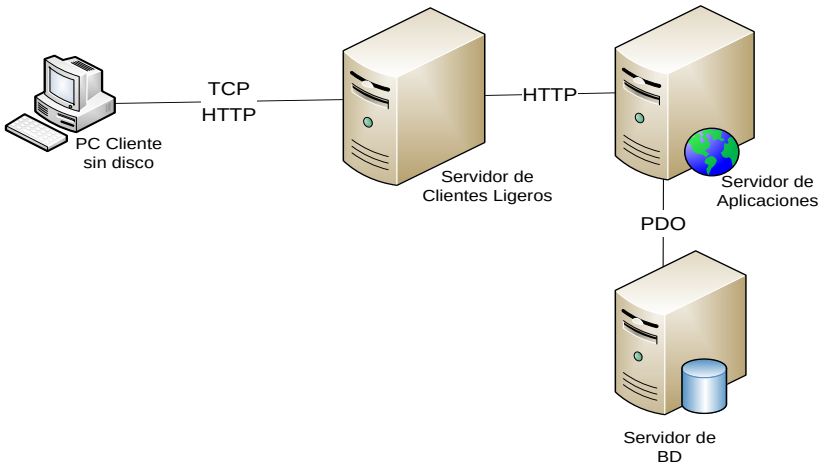

Fig. 7 Diagrama de despliegue de escenario para PC cliente sin disco.

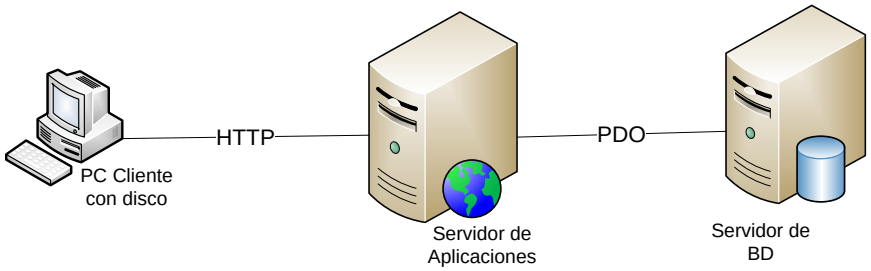

Fig. 8 Diagrama de despliegue de escenario para PC cliente con disco.

2.3.5. Interfaces

En la figura 9 se muestra la interfaz Plan de vacaciones, la cual ofrece las opciones de Adicionar, Modificar, Eliminar y Confirmar un plan de vacaciones, además permite filtrar por año y área y listar los planes de vacaciones del año.

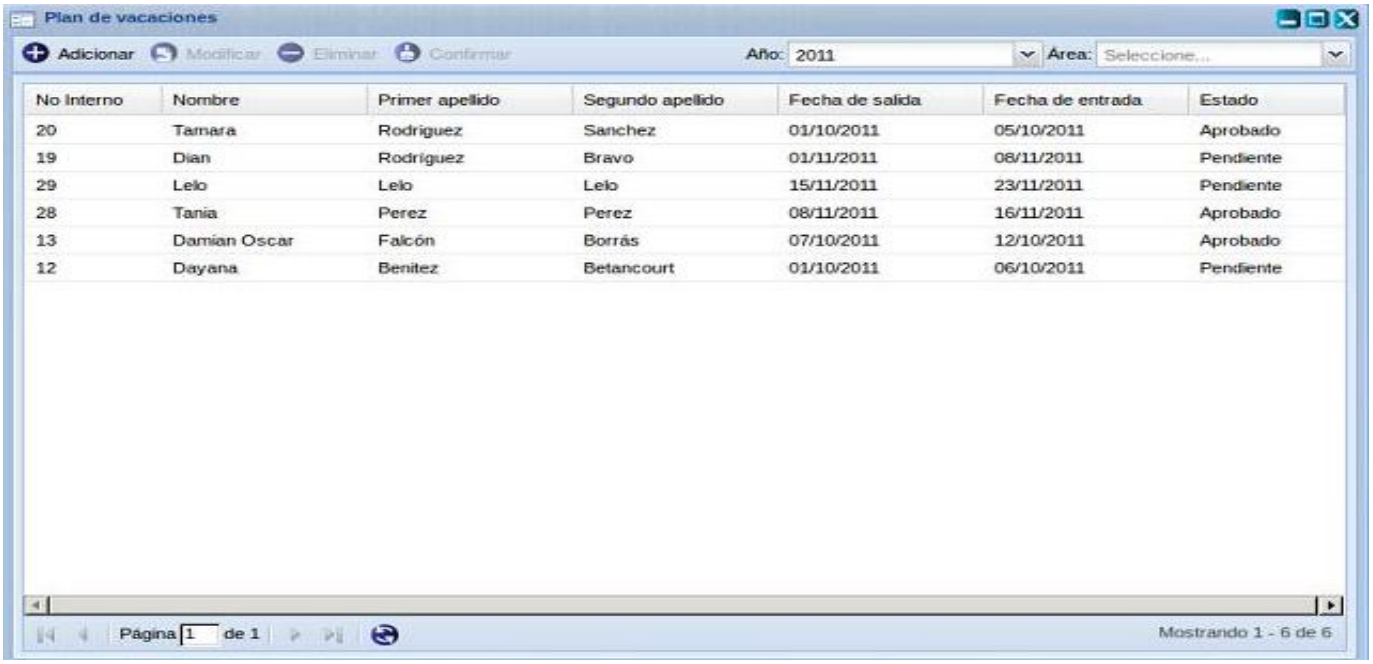

Fig. 9 Interfaz Plan de vacaciones.

En la figura 10 se muestra la interfaz Submayor de vacaciones, la cual ofrece la opción de Buscar trabajador, mediante los campos expediente, nombre, primer apellido y segundo apellido, además permite filtrar por centro de costo y área, listar los submayores de vacaciones y mostrar los detalles de un trabajador.

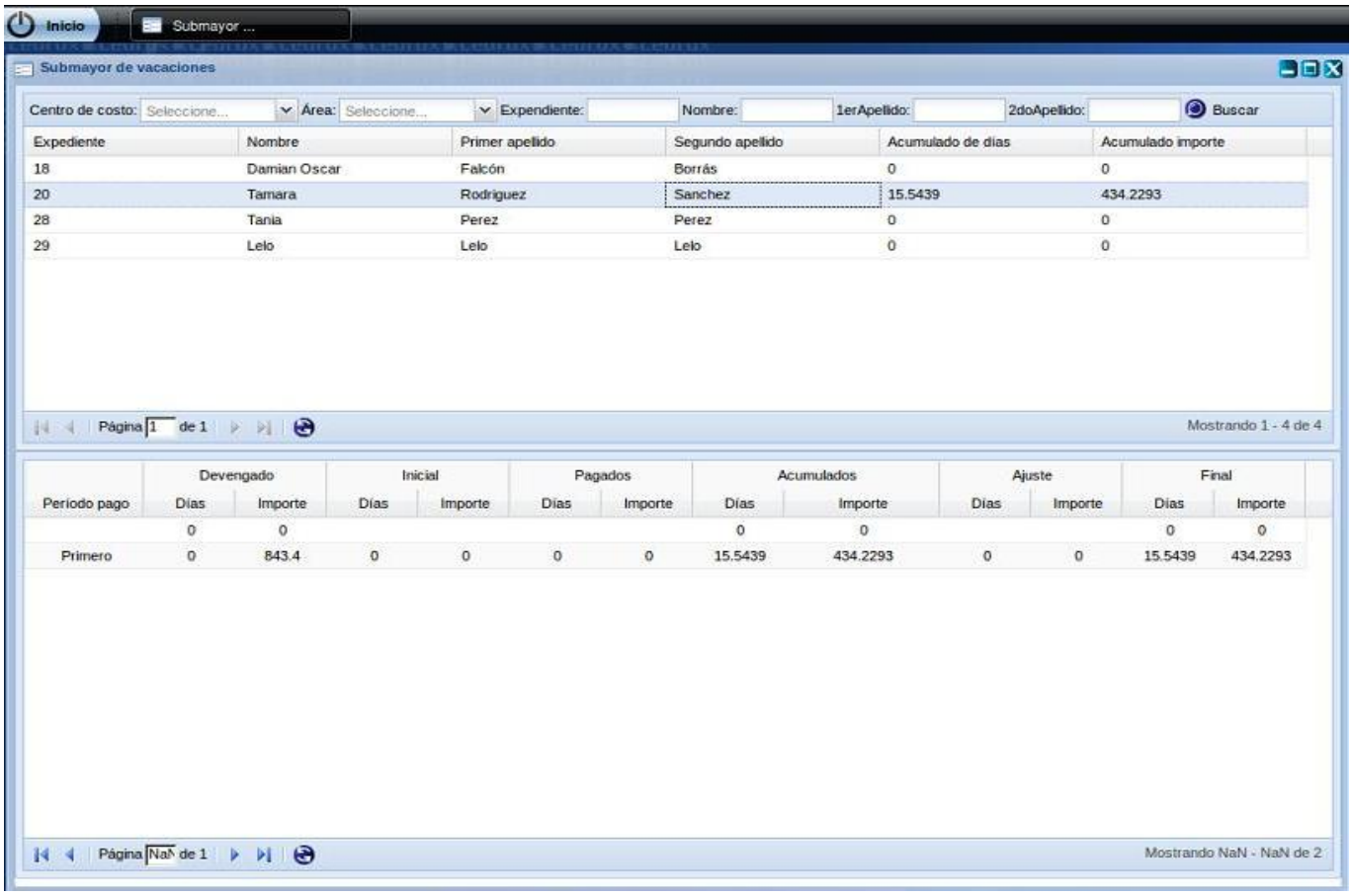

Fig. 10 Interfaz Submayor de vacaciones.

En la figura 11 se muestra la interfaz Notificar vacaciones, la cual ofrece las opciones de Adicionar, Modificar, Eliminar, Importar y Confirmar una notificación de vacaciones, además permite filtrar por período de pago y área y listar las notificaciones.

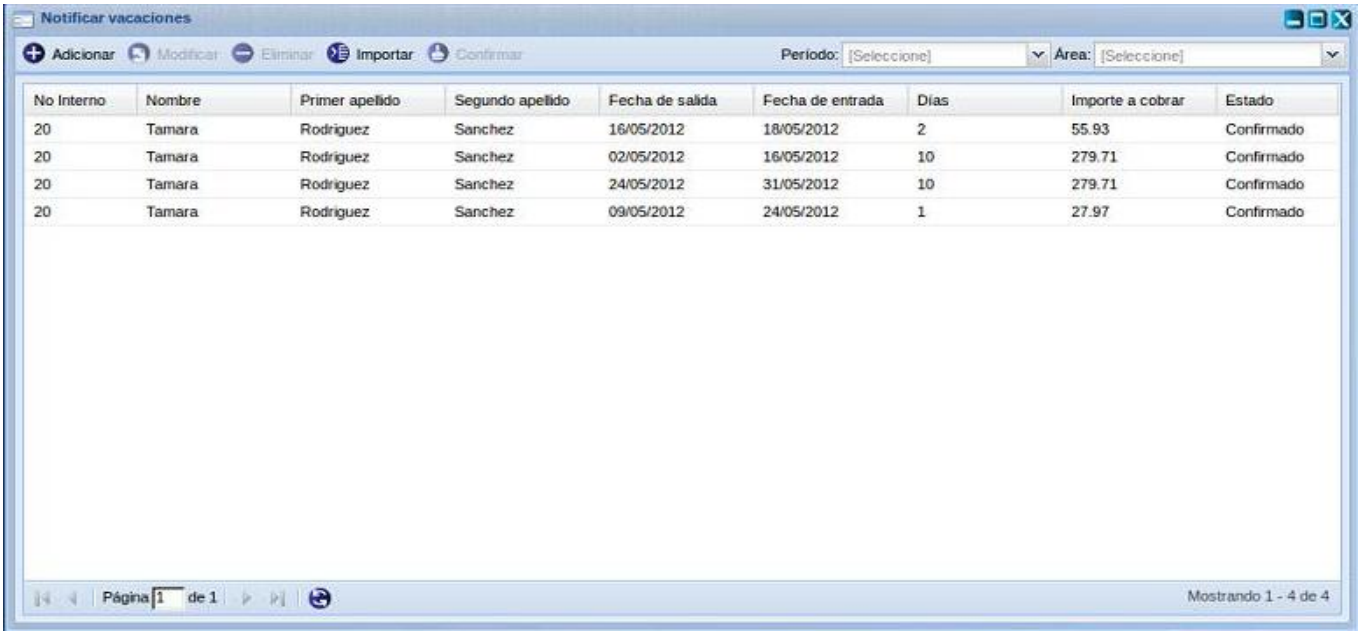

Fig. 11 Interfaz Notificar vacaciones.

2.4. Conclusiones del capítulo

En el capítulo se realizó la modelación del diseño representándose gráficamente los diagramas de clases del diseño para describir cómo quedará la estructura del sistema, además se realizó el modelo de datos físico de la base de datos para percibir cómo persisten los datos en la base de datos. Se realizó la implementación del componente vacaciones anuales pagadas para darle cumplimiento a los objetivos del trabajo. Al concluir este capítulo la propuesta de solución del trabajo está lista para ser validada, esto se realizará en el próximo capítulo.

Capítulo 3: Validación de la solución propuesta.

3.1. Introducción

El desarrollo del software ha de ir acompañado de alguna actividad que garantice la calidad; la prueba es un elemento crítico para la garantía de calidad del software. La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo.

En este capítulo se realizará la validación de la solución propuesta. Se validará el diseño propuesto mediante la aplicación de las métricas Tamaño Operacional de Clase (TOC) y Relación entre Clases (RC), que proporcionan una medida de la complejidad y calidad del software. Se aplican pruebas de caja blanca y caja negra, con el objetivo de verificar la funcionalidad y estructura del componente desarrollado.

3.2. Validación del diseño propuesto

La realización de métricas son claves en la ingeniería del software orientada a objetos. Estas permiten tener una visión más clara y profunda y proporcionan un mecanismo para la evaluación de la calidad del software.

Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen medidas de la cohesión, acoplamiento y complejidad del subsistema. Estas tres medidas pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes. (Pressman, 1998)

- **Atributos de calidad que se abarcan:**

Responsabilidad: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.

Complejidad del mantenimiento: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.

Complejidad de implementación: Grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.

Acoplamiento: Dependencia o interconexión de una clase o estructura de clase respecto a otras. **Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

➢ A continuación se presentarán las métricas necesarias para evaluar la calidad del diseño propuesto:

Tamaño operacional de clase: Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes mencionado.

- **Aplicación de la métrica TOC.**

Atributos que determinan la métrica TOC:

	Categoria	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Fig. 12 Métrica Tamaño Operacional de Clase (TOC).

Representación de la cantidad de procedimientos en cada clase del componente Vacaciones

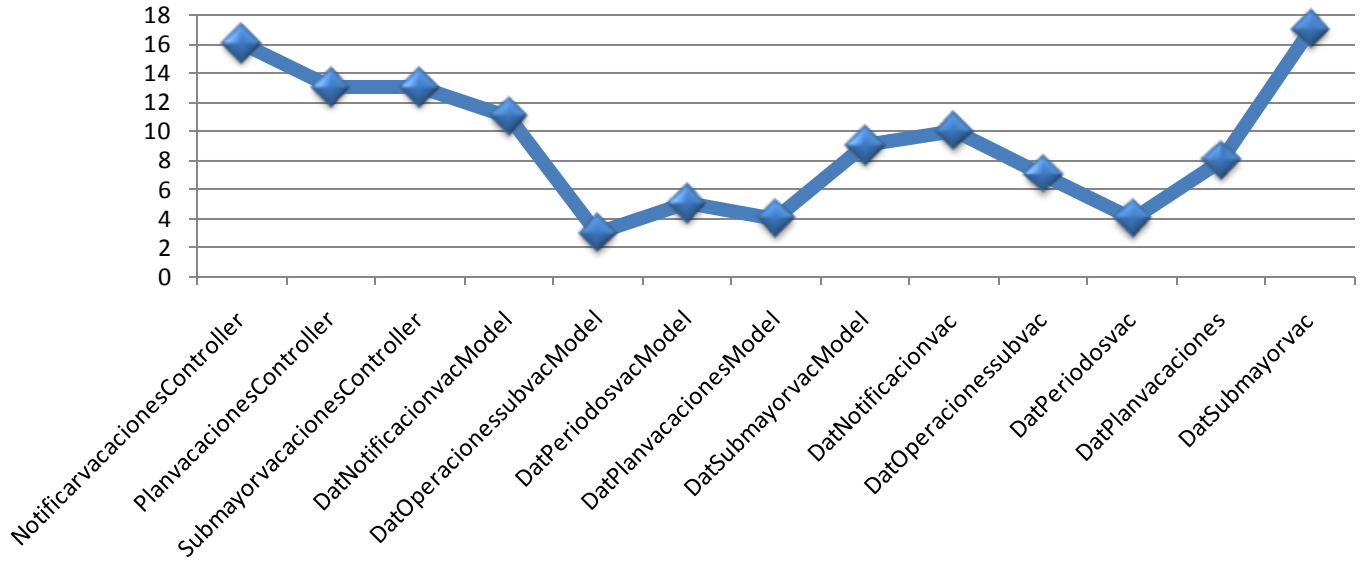

Fig. 13 Cantidad de procedimientos por clases del componente Vacaciones.

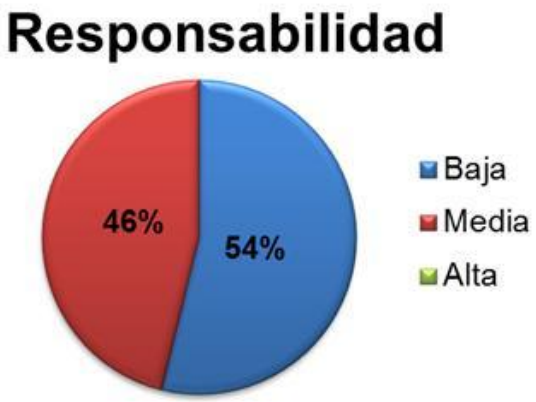

Fig. 14 Resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

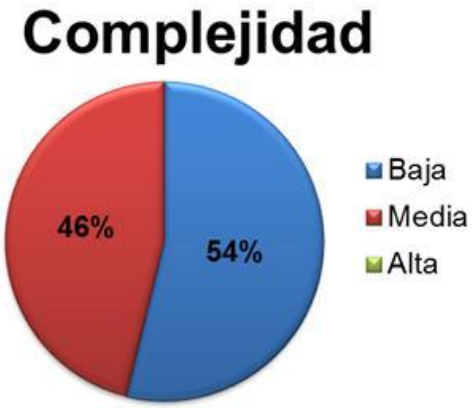

Fig. 15 Resultados de la evaluación de la métrica TOC en el atributo Complejidad.

Fig. 16 Resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Cuando existe un TOC alto se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el componente Vacaciones está entre los límites aceptables de calidad, teniendo en cuenta que en el **100 %** de las clases que conforman el sistema están dentro de las categorías de baja y media, lo que demuestra la elevada reutilización, baja complejidad de implementación y responsabilidad en el diseño propuesto.

Relaciones entre clases: Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Cantidad de pruebas y Reutilización, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

- **Aplicación de la métrica RC.**

Atributos que determinan la métrica RC:

	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

	Categoria	Criterio
Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoria	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

	Categoria	Criterio
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Fig. 17 Atributos de la métrica RC.

Representación de la cantidad de relaciones de uso en cada clase del componente Vacaciones

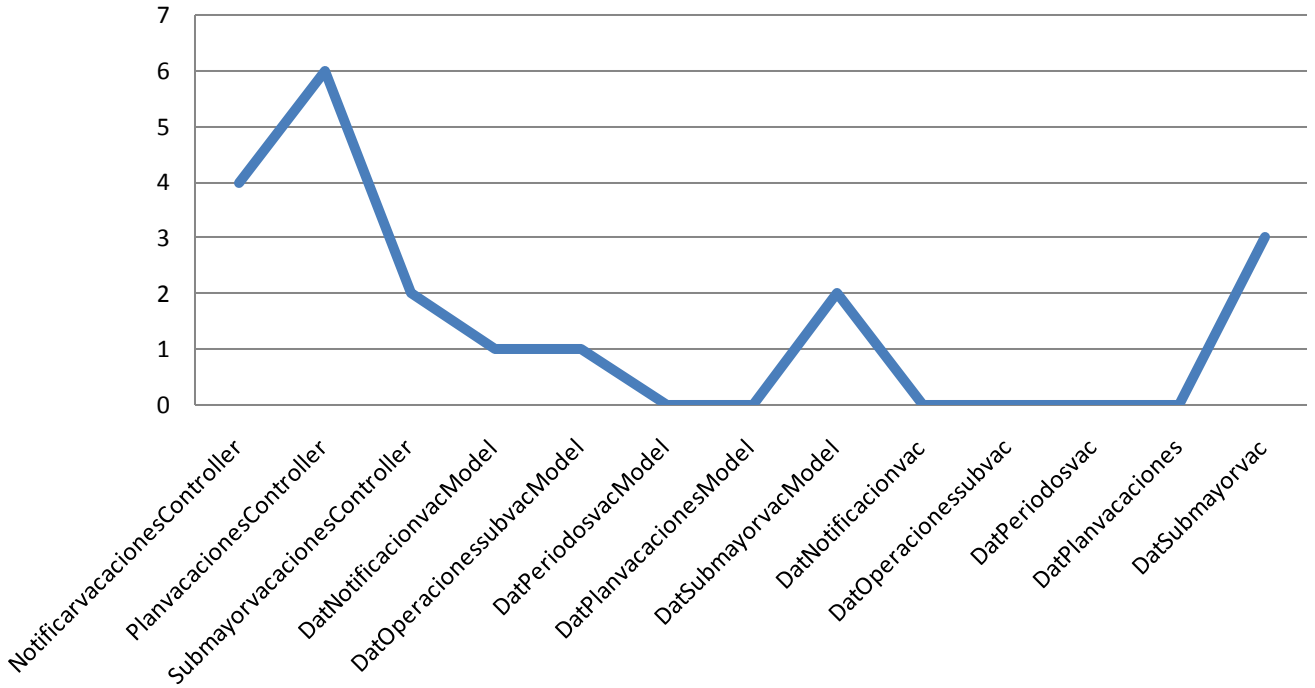

Fig. 18 Cantidad de relaciones de uso en cada clase del componente Vacaciones

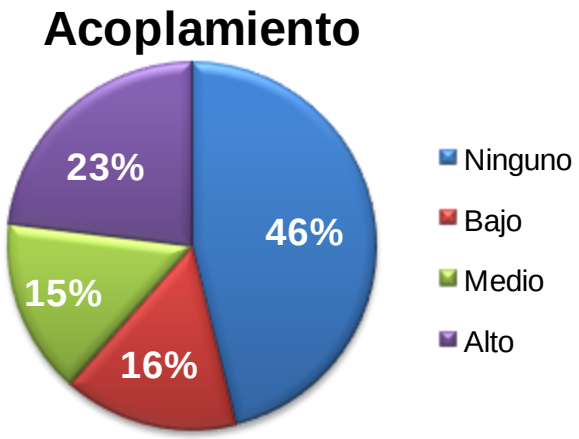

Fig. 19 Resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

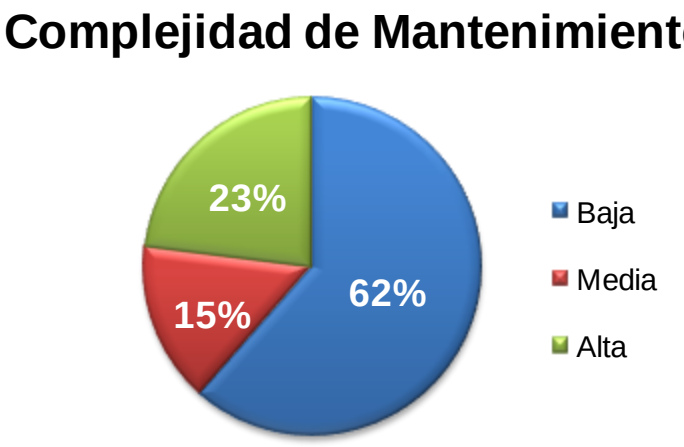

Fig. 20 Resultados de la evaluación de la métrica RC en el atributo Complejidad de mantenimiento.

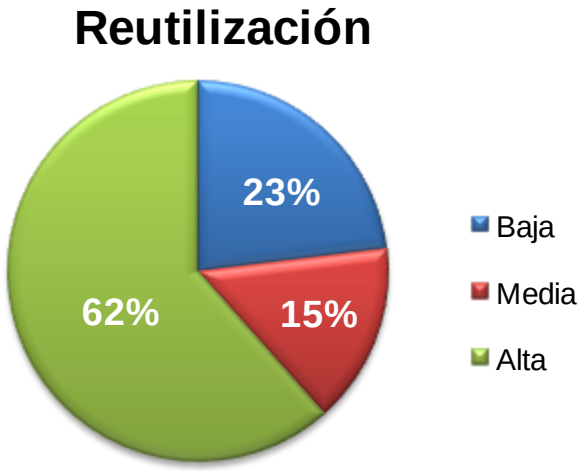

Fig. 21 Resultados de la evaluación de la métrica RC en el atributo Reutilización.

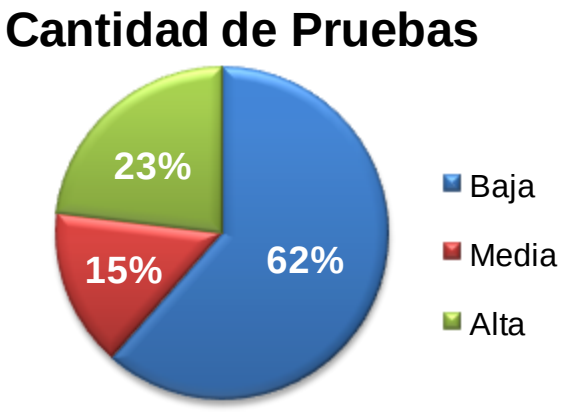

Fig. 22 Resultados de la evaluación de la métrica RC en el atributo Cantidad de pruebas.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el componente Vacaciones está entre los límites aceptables de calidad, teniendo en cuenta que el **77%** de las clases que conforman el sistema están dentro de las categorías de baja y media, lo que demuestra la elevada reutilización, bajo acoplamiento, complejidad y cantidad de pruebas en el diseño propuesto.

Las métricas de software aplicadas posibilitaron estimar la calidad de los atributos internos del producto, demostrando una aceptable calidad de diseño. La solución propuesta contribuirá a la disminución de inconvenientes durante la implementación del componente, garantizando la reutilización y agilidad en el proceso de desarrollo de software.

3.3. Pruebas de software

Las pruebas del software son un elemento crítico para la garantía de la calidad del software. Las pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto.

Para llevar a cabo el proceso de pruebas al componente se definen estrategias de pruebas con el propósito de garantizar la calidad del software.

- **Niveles de prueba**

A la hora de evaluar dinámicamente un sistema se debe comenzar por los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas se aplican en distintos niveles de trabajo, dentro de estos se distinguen:

Pruebas de unidad: Prueba individual a las unidades separadas de un sistema de software.

Pruebas de integración: Los componentes individuales son combinados con otros componentes para asegurar que la comunicación, enlaces y los datos compartidos ocurran apropiadamente.

Pruebas del sistema: Son usualmente conducidas para asegurar que todos los módulos trabajan como sistema sin error. Es similar a la prueba de integración pero con un alcance mucho más amplio.

Pruebas de aceptación: Son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

- **Métodos de prueba**

Caja Negra:

Se comprueban las funcionalidades sin tener en cuenta la estructura interna.

Caja Blanca:

Se comprueban los componentes internos.

3.3.1. Objetivo

Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. El aspecto fundamental que rige esta etapa de pruebas es determinar cómo y en qué sentido el componente cumple con las expectativas del cliente, a partir de los requisitos establecidos y las restricciones impuestas. En ese ámbito se trazan un conjunto de objetivos dentro de los que se sitúan:

- ✓ Verificar la implementación del componente.
- ✓ Verificar la integración adecuada del componente.
- ✓ Verificar que todos los requisitos se han implementado correctamente.
- ✓ Identificar los errores y asegurar que estos sean corregidos de la mejor manera.

3.3.2. Alcance

Las pruebas de software deben abarcar diversas ramas; desde la funcionalidad de los primeros prototipos, la estabilidad, cobertura y rendimiento de la arquitectura, hasta el producto final. Cuando se ejecuta una prueba, se tienen en cuenta qué tanto los resultados obtenidos se asemejan a los resultados esperados. Si la salida no es la esperada, indica la ocurrencia de un error y en ese caso es preciso corregirlo, esto implica todo un proceso de depuración de errores a través de los casos de prueba. Se debe tener en cuenta el costo de tiempo y esfuerzo que traen aparejado los errores, generalmente se presentan en situaciones complejas y en ocasiones las soluciones no las puede determinar el propio desarrollador.

3.4. Pruebas de software aplicadas al componente

3.4.1. Prueba de Caja Blanca o Estructural:

- **Descripción:**

La prueba de Caja Blanca también se conoce como prueba de Caja Transparente o de Cristal. En ese sentido el criterio de selección de casos de prueba buscará cierta cobertura no solo para la determinación de caminos independientes, sino también de valores para las condiciones de bucles dentro y fuera de sus límites operacionales basados en el contenido de los módulos.

Esta prueba consiste específicamente en como diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento.

Dentro de la prueba de caja blanca se incluyen las Técnicas de pruebas que serán descritas a continuación:

Prueba del camino básico: Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. Los casos de prueba obtenidos garantizan que durante la prueba se ejecute al menos una vez cada sentencia del programa.

Prueba de condición: Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.

Prueba de flujo de datos: Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.

Prueba de bucles: Método de prueba que se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución todos los bucles en sus límites operacionales.

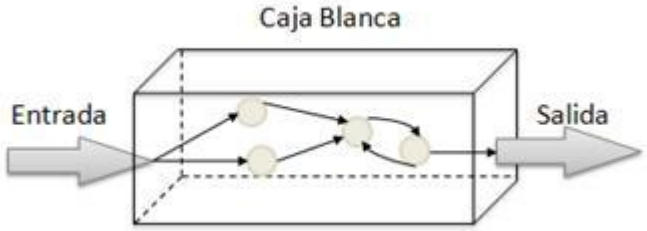

Fig. 23 Modelación de la prueba de Caja blanca.

- **Aplicación:**

Según la descripción de la prueba de caja blanca presentada con anterioridad y a partir de la necesidad de crear un producto de alta calidad es preciso valorar qué tan certera ha sido la implementación del componente Vacaciones y para ello es necesario aplicar una de las técnicas que esta comprende, en este caso la del camino básico. Para ello es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática. Se debe comenzar por un análisis del código, posteriormente son enumeradas cada una de las instrucciones, se construye el grafo de flujo asociado y según las fórmulas pertinentes se calcula dicha complejidad:

A continuación se analizan y enumeran las sentencias de código de uno de los procedimientos contenidos en la clase NotificarvacacionesController, específicamente: modificarNotificacion, este algoritmo se encarga de modificarla notificación de vacaciones de un trabajador, a partir del identificador de la misma.

```
function modificarNotificacionAction() {
    $idnotif = $this->_request->getPost('idnotificacion');
    $idestructuracomun = $this->global->Estructura->idestructura; //1
    if ($idnotif){  //2
        $NotificacionvacM = new DatNotificacionvac();
        $notif = $NotificacionvacM->Buscar($idnotif);
        $notifM = new DatNotificacionvacModel();
        $notif->idtrabajador = $this->_request->getPost('idtrabajador');
        $notif->diasapagar = $this->_request->getPost('diaspagar');
        $notif->fechainicio = $this->_request->getPost('Fsalida');
        $notif->fechafin = $this->_request->getPost('Fentrada');
        $notif->fechaincorporacion = $this->_request->getPost('Feincorporacion');
        $notif->importeacobrar = $this->_request->getPost('importecobrar');
        $liq = $this->_request->getPost('liquidacion');  //3
        if ($liq){  //4
            $tipo = $this->_request->getPost('tipoperiododepago');
            if ($tipo == 'Parcial')  //5
                $notif->liquidacion = 1;  //6
            else
                $notif->liquidacion = 2;  //7
        }
        else
            $notif->liquidacion = 0;  //8
        $notif->noorden = $this->_request->getPost('norden');
        $notif->estado = 1;
        $notifM->Insertar($notif);
        $this->showMessage(1, '');  //9
    }
    else
        $this->showMessage(3, '');  //10
}  //11
```

Fig. 24 Sentencia de código.

Después de este paso, es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, en ese caso:

Nodo: Círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, un nodo en sí puede representar un proceso, una secuencia de procesos o una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: Saetas a través de las cuales se unen los Nodos y constituyen el flujo de control del procedimiento.

Regiones: Las regiones son las áreas delimitadas por las aristas y nodos.

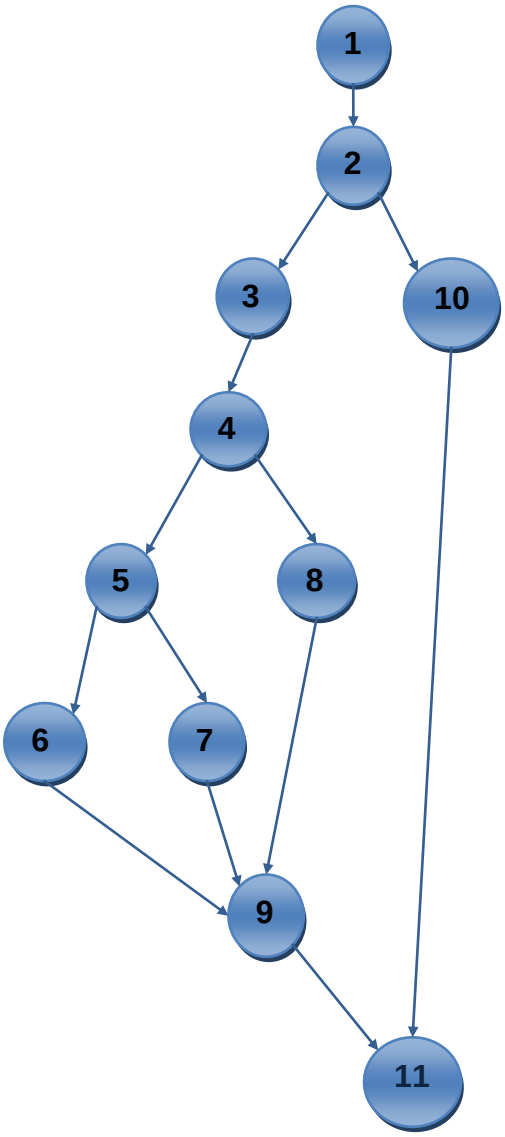

Fig. 25 Grafo de flujo asociado al código.

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 de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A – N) + 2

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

V (G) = (13 - 11) + 2

V (G) = 4.

2. V (G) = P + 1

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

V (G) = 3 + 1

V (G) = 4.

3. V (G) = R

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

V (G) = 4.

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 4, de manera que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el número mínimo 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.

Nro	Camino básico
1	**1 – 2 – 3 – 4 – 5 – 6 – 9– 11**
2	**1 – 2 – 3 – 4 – 5 – 7– 9– 11**
3	**1 – 2 – 3 – 4 – 8 – 9– 11**
4	**1 – 2 – 10 – 11**

Tabla 1 Caminos básicos.

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo. Para definir los casos de prueba es necesario tener en cuenta:

Descripción: Se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.

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

Entrada: Se muestran los parámetros que serán la entrada al procedimiento.

Resultados esperados: Se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

Caso de prueba para el camino básico # 1 (1 – 2 – 3 – 4 – 5 – 6 – 9 – 11)	
Descripción	Se modifica los datos de una notificación dado el identificador determinado.
Condición de ejecución	Debe existir la notificación registrada en el sistema, Se debe introducir el campo requerido.
Entrada	$idnotif = idnotificacion;
Resultados esperados	Se debe realizar la modificación de la notificación de vacaciones.
Resultados	Se modifica la notificación satisfactoriamente.

Tabla 2 Caso de prueba para el camino básico # 1.

Caso de prueba para el camino básico # 2 (1 – 2 – 3 – 4 – 5 – 7 – 9 – 11)	
Descripción	Se modifica los datos de una notificación dado el identificador determinado.
Condición de ejecución	Debe existir la notificación registrada en el sistema, Se debe introducir el campo requerido.
Entrada	$idnotif = idnotificacion;
Resultados esperados	Se debe realizar la modificación de la notificación de vacaciones.
Resultados	Se modifica la notificación satisfactoriamente.

Tabla 3 Caso de prueba para el camino básico # 2.

Caso de prueba para el camino básico # 3 (1 – 2 – 3 – 4 – 8 – 9 – 11)	
Descripción	Se modifica los datos de una notificación dado el identificador determinado.
Condición de ejecución	Debe existir la notificación registrada en el sistema, Se debe introducir el campo requerido.
Entrada	$idnotif = idnotificacion;
Resultados esperados	Se debe realizar la modificación de la notificación de vacaciones.
Resultados	Se modifica la notificación satisfactoriamente.

Tabla 4 Caso de prueba para el camino básico # 3.

Caso de prueba para el camino básico # 4 (1 – 2 – 10 – 11)	
Descripción	Se modifica los datos de una notificación dado el identificador determinado.
Condición de ejecución	Debe existir la notificación registrada en el sistema, Se debe introducir el campo requerido.
Entrada	$idnotif = idnotificacion;
Resultados esperados	Se debe realizar la modificación de la notificación de vacaciones.
Resultados	No se pudo modificar la notificación porque no hay ninguna notificación con el identificador introducido.

Tabla 5 Caso de prueba para el camino básico # 4.

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. Por lo que 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. Prueba de Caja Negra o Funcional:

- **Descripción:**

A este tipo de prueba también se le conoce como prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa.

En esencia permite encontrar:

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las bases de datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

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

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

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

Grafos de causa-efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

De estas técnicas, la seleccionada fue partición de equivalencia la cual permite examinar los valores válidos e inválidos de las entradas existentes en el software.

Para la aplicación de esta técnica se realizan diseño de casos de prueba los cuales se basan en una evaluación de las clases de equivalencia para una condición de entrada.

- Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.
- Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.
- Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:
    1. Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
    2. Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
    3. Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
    4. Si una entrada es booleana, hay 2 clases: si o no.

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

Fig. 26 Modelación de la prueba de Caja negra.

- **Aplicación:**

A continuación se especifica el caso de prueba para el requisito “Modificar notificación”.

Descripción general

Se debe seleccionar el subsistema Capital humano/ Administración de capital humano/ Vacaciones pagadas/Notificar vacaciones

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 una notificación de vacaciones registrada en el sistema.
- Se debe presionar el botón Modificar de la barra de opciones.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Modificar notificación de vacaciones	El sistema debe permitir modificar una notificación de vacaciones.	EP 1.1: Modificar notificación de vacaciones datos válidos.	– Se introducen los datos de la notificación de vacaciones correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.2: Modificar notificación de vacaciones introduciendo datos válidos presionando el botón **Aplicar**.	– Se introducen los datos de la notificación de vacaciones correctamente. – Se presiona el botón **Aplicar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**.
		EP 1.3: Modificar notificación de vacaciones introduciendo datos inválidos.	– Se introducen los datos inválidos de la notificación de vacaciones.

			– Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP 1.4: Modificar notificación de vacaciones dejando campos vacíos.	– Se introducen los datos dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP 1.5: Cancelar.	– Se introducen o no los datos de la notificación de vacaciones Se presiona el botón **Cancelar**.

Tabla 6 Descripción del caso de prueba para el requisito Modificar notificación.

No	Nombre de campo	Tipo	Válido	Inválido	Inválido
1	Exp. Interno	Campo de texto	Números	Letras y caracteres especiales	Vacío
2	Nombre	Campo de texto (campo de selección no editable)	Letras	Caracteres especiales y números	Vacío.
3	Primer Apellido	Campo de texto (campo de selección no editable)	Letras	Caracteres especiales y números	Vacío.
4	Segundo Apellido	Campo de texto (campo de selección no editable)	Letras	Caracteres especiales y números	Vacío.

5	F.entrada	Campo de texto (Editable).	Números.	Caracteres especiales.	Letras
6	F.salida	Campo de texto (Editable).	Números	Caracteres especiales.	Letras
7	No de orden	Campo de texto	Números	Caracteres especiales y letras	Vacío
8	Liquidación	Campo de selección	Si/No	NA	
9	Días Acumulados	Campo de texto	Números	Caracteres especiales y letras	Vacío
10	Importe Acumulado	Campo de texto	Números	Caracteres especiales y letras	Vacío
11	Fecha de incorporación	Campo de texto (Editable).	Números.	Caracteres especiales.	Letras
12	Días a pagar	Campo de texto	Números	Caracteres especiales y letras	Vacío
13	Importe a cobrar	Campo de texto	Números	Caracteres especiales y letras	Vacío

Tabla 7 Descripción de las variables del caso de prueba para el requisito Modificar notificación.

Id	Escenario	N. interno	F. entrada	F. salida	No de orden	Liqui-dación	Días acumu-lados	Imp. Acumu-lado	F. Incorpo-racion	Días a pagar	Imp. cobrar	Respuesta del sistema
EP 1.1	Modificar notificación de vacaciones introduciendo datos válidos.	V(121)	V(121)	V(212)	V(212)	NA	V(212)	V(212)	V(1/2/03)	V(212)	V(212)	El sistema adiciona notificación de vacaciones y muestra el mensaje de información: “Se ha adicionado satisfactoriamente.”. El sistema cierra la interfaz.
EP 1.2	Modificar notificación de vacaciones introduciendo datos válidos presionando el botón **Aplicar**.	V(121)	V(121)	V(212)	V(212)	NA	V(212)	V(212)	V(1/2/03)	V(212)	V(212)	El sistema adiciona notificación de vacaciones y muestra el mensaje de información: “Se ha adicionado satisfactoriamente.”. El sistema mantiene la interfaz abierta.

EP 1.3	Modificar notificación de vacaciones introduciendo datos inválidos.	I(letras)	V(212)	V(212)	V(212)	NA	V(212)	V(212)	V(1/2/03)	V(212)	V(212)	El sistema no permite la inserción de letras en este campo. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio”. El sistema mantiene la interfaz abierta.
		I(%$·”@)	V(121)	V(212)	V(212)	NA	V(212)	V(212)	V(1/2/03)	V(212)	V(212)	El sistema no permite la inserción de caracteres especiales en este campo. El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”. El sistema mantiene la interfaz abierta.

EP 1.4	Modificar notificación de vacaciones dejando campos vacíos.	I(vacio)	V(121)	V(212)	V(212)	NA	V(212)	V(212)	V(1/2/03)	V(212)	V(212)	El sistema subraya el campo en rojo mostrando el mensaje: "Este campo es obligatorio.". El sistema mantiene la interfaz abierta.
EP 1.5	Cancelar.	NA	NA	NA	NA	NA	NA	NA	NA	NA		El sistema cierra la interfaz sin realizar ninguna operación.

Tabla 8 Datos de prueba del caso de prueba para el requisito Modificar notificación.

Las pruebas de caja negra a la solución fueron desarrolladas por el Grupo de Aseguramiento de la Calidad de CEIGE. Estas se realizaron en 3 iteraciones de prueba, donde finalmente se comprobó en la tercera iteración que el componente estaba libre de no conformidades, para así lograr su liberación. Actualmente el componente Vacaciones se encuentra en el proceso de liberación en Calisoft, Centro nacional de calidad de software. En el cual se han realizado 2 iteraciones de prueba. La tabla 20 muestra los resultados de las iteraciones realizadas.

Agrupación de requisitos	No conformidades	
	Iteración 1	Iteración 2
Submayor de vacaciones	1	2
Plan de vacaciones	2	2
Notificación de vacaciones	6	1
Total	**9**	**5**

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

3.5. Conclusiones del capítulo

La calidad del componente desarrollado fue el elemento clave del capítulo que recién concluye. En ese sentido se efectuaron pruebas de software en el nivel de unidad mediante casos de pruebas, para los cuales se tuvieron en cuenta las entradas, las salidas, los resultados esperados y el tratamiento de errores en caso de anomalías; se aplicaron además las métricas: Relaciones entre Clases y Tamaño Operacional de la Clase para validar y evaluar el diseño, las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes.

El componente Vacaciones desde el punto de vista funcional cumple con los requerimientos capturados y especificados en las primeras etapas de desarrollo a partir de las expectativas del cliente.

Conclusiones generales

Una vez terminado el trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

- Se analizaron ventajas y desventajas de sistemas informáticos tanto nacionales como internacionales vinculados a la gestión de las vacaciones; evidenciándose de esta manera la no existencia de una solución informática capaz de ejecutar las funcionalidades referentes a las vacaciones anuales del trabajador, ni de cumplir con los requisitos establecidos a nivel nacional.
- Se obtuvo el diseño y la implementación del componente Vacaciones correspondiente al Subsistema Capital Humano del Sistema Integral de Gestión de Entidades Cedrux, con el objetivo de erradicar los problemas de los sistemas existentes.
- Se evaluó la viabilidad del componente a través de pruebas de software efectuadas para el nivel de unidad, las cuales arrojaron resultados favorables posibilitando dar cumplimiento a las funcionalidades previstas para el mismo.

Recomendaciones

Después de realizadas las conclusiones de este trabajo se recomienda:

- ✓ Continuar el estudio del tema con el objetivo de incluir nuevas funcionalidades en versiones posteriores del sistema.
- ✓ Realizar el despliegue del componente propuesto como parte del subsistema de Capital Humano del Sistema Integral de Gestión Cedrux.

Bibliografía

Almada, Federico. 2009. [En línea] 1 de enero de 2009. http://www.techtear.com/2008/01/22/zend-studio-for-eclipse-desarrollo-profesional-en-php.

Alvarez, Sara. 2007. Desarrollo Web. [En línea] 2007. http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html..

Alvarez, Sara. 2009. Desarrollo Web. [En línea] 2009. http://www.desarrolloweb.com/articulos/2477.php.

Apache, The Apache Sotfware Foundation. 2010. The Apache Sotfware Foundation. [En línea] 2010. http://apache.org.

ASSETS. 2005. ASSETS. *Sistema de gestión integral.* [En línea] 2005. http://www.assets.co.cu/assets.asp.

Cabrera, Miguel P., y otros. 2009. Sistema económico integrado VerSat Sarasola. [En línea] 2009. http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc.

Cañete Pupo, Yanisleydi. 2010. *Libro de Ayuda del Marco de Trabajo Sauxe, En su versión 2.0.* Habana : s.n., 2010.

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

Coello, Costa Helkyn R. informatizate. 2002. [En línea] noviembre de 2002. http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.html.

DGPLADES. 2010. Dirección general de planeación y desarrollo de salud. [En línea] 2010. http://www.dgplades.salud.gob.mx/descargas/dhg/ADMON_CH.pdf.

Doctrine. 2008. Doctrine. [En línea] 2008. http://www.doctrine-project.org.

Ecured. 2011. Ecured. [En línea] 7 de diciembre de 2011. http://www.ecured.cu/index.php/Visual_Paradigm.

EICMA. 2010. [En línea] 2010. http://www.eimagr.cu/index.php/component/content/article/60-portada/101-versat-sarasola-un-sistema-para-trabajar.

Fernández, Osmar Leyet. 2010. *Documento Línea Base de proyecto-CEDRUX-1.0.* 2010.

Gaceta Oficial, de Cuba. 2008. Gaceta Oficial de Cuba. [En línea] 2008. http://www.gacetaoficial.cu/html/codigodetrabajo.html.

GNU. 2009. GNU Operating System. [En línea] 2009. [Citado el: 22 de enero de 2012.] http://www.gnu.org/philosophy/free-sw.html.

Hernández León, Rolando Alfredo y Coello González, Sayda. 2011. *EL PROCESO DE INVESTIGACIÓN CIENTÍFICA.* La Habana : Universitaria, 2011. ISBN 978-959-16-1307-3.

Hernández Silva, Frank y Martí Lahera, Yohannis. 2006. Acimed. *Revista cubana de los profesionales de la información y de la comunicación de la salud.* [En línea] 18 de enero de 2006. [Citado el: 12 de enero de 2012.] http://bvs.sld.cu/revistas/aci/vol14_1_06/aci03106.htm.

HtmlCastellano. 2009. HtmlCastellano. [En línea] 2009. [Citado el: 10 de febrero de 2012.] http://www.programacion.com/html.

Krutchen, P. Kroll, P. 2002. *"Rational Unified Process".Segunda Edición.* 2002.

Larman, Craig. 1999. *UML y Patrones, Introducción al análisis y diseño orientado a objetos.* México : Prentice Hall, 1999.

Ley 49, Trabajo. 1984. Ley del trabajo. *Vacaciones anuales pagadas.* 1984. Vol. Capítulo 3, SECCIÓN OCTAVA.

Mozilla Firefox. 2011. [En línea] 2011. http://www.getfirefox.es/firefox-features.

Netbeans. 2011. Netbeans. [En línea] 2011. [Citado el: 13 de Diciembre de 2011.] http://netbeans.org/community/releases/69/index_es.html.

PostgreSQL. 2010. PostgreSQL. [En línea] 2010. [Citado el: 16 de abril de 2012.] http://www.postgresql.org/docs/8.3/static/intro-whatis.html.

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

RapidSVN. 2011. RapidSVN. [En línea] 2011. [Citado el: 16 de mayo de 2012.] http://www.rapidsvn.org/.

Rasmus Lerdorf, Kevin Tatroe & Peter MacIntyre. 2006. *Programming PHP.* 2006. 978-0-596-00681-5.

Rodas XXI. 2002. Rodas XXI. [En línea] 20 de diciembre de 2002. [Citado el: 19 de febrero de 2012.] http://www.rodasxxi.cu/rodasxxi.php.

Romero, Lorely Moya. 2009. Funcionalidades y principales opciones del módulo de planificación del software integrado Versat Sarasola. [En línea] 2009. [Citado el: 10 de enero de 2012.] http://www.eumed.net/cursecon/ecolat/cu/2009/lmr2.html.

Sistcont. 2008. Sistcont. *Software Contable Financiero.* [En línea] 2008. [Citado el: 13 de enero de 2012.] http://www.siscont.com/DESCRIPTIVO%20SISCONT.pdf.

sistemas, Metodología de. 2007. Metodologias de sistemas. [En línea] 2007. [Citado el: 1 de febrero de 2012.] http://metodologiasdesistemas.blogspot.com/2007/10/que-es-un-orm-object-relational-mapping.html.

SOA. 2006. SOA. [En línea] 2006. [Citado el: 4 de febrero de 2012.] http://arquitecturaorientadaaservicios.blogspot.com/2006/03/pero-qu-es-realmente-soa.html.

UBUNTU, GUÍA DOCUMENTADA PARA. 2009. Subversion. [En línea] 9 de enero de 2009. [Citado el: 18 de diciembre de 2011.] http://www.guia-ubuntu.org/index.php?title=Subversion..

Vega, Yanet. 2009. *Definición del ciclo de vida del proyecto.* La Habana : s.n., 2009.

W3C. 2011. The World Wide Web Consortium. [En línea] 2011. [Citado el: 12 de mayo de 2012.] http://www.w3.org/XML/.

XPPS. 2009. El lenguaje HTML. [En línea] 2009. [Citado el: 7 de diciembre de 2011.] http://www.xpps.net/contenido_xppsnet/areatec/HMTL.pdf.