# Universidad de las Ciencias Informáticas

# Facultad 3


## Implementación del componente Control de Asistencia del subsistema Capital Humano del Sistema Integral de Gestión Cedrux

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

**Autor:** Yasmany Flores Loy

**Tutor:** MSc. Donel Vázquez Zambrano

**Co-Tutor:** Ing. José Antonio Linares Torres

Ciudad de la Habana, __ de Junio del 2011.

## *Declaración de autoría*

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Yasmany Flores Loy

MSc. Donel Vazquez Zamabrano

_____

_____

**Datos del Tutor:**

**MSc. Donel Vázquez Zambrano**

Profesor Instructor graduado de Ingeniería en Ciencias Informáticas con Título de Oro y de Máster en Gestión de Proyectos Informáticos en la Universidad de las Ciencias Informáticas (UCI). Ha cursado y aprobado diecisiete cursos de postgrado. Ha participado en cuatro eventos nacionales y uno internacional como ponente o autor. Tiene en su haber nueve publicaciones científicas.

dvz@uci.cu

**Datos del Co-tutor:**

**Ing. José Antonio Linares Torres**

**Profesión:** Ingeniero en ciencias informáticas

**Años de graduado: 2**

jalinares@uci.cu

# *Resumen*

La Universidad de las Ciencias Informáticas (UCI) es una universidad productiva, cuya misión es producir software y servicios informáticos a partir de la vinculación estudio – trabajo como modelo de formación. La necesidad de mejorar los procesos de Control de Asistencia está impulsando a la adopción de sistemas automatizados para la administración de dicho proceso. Debido a las deficiencias que contrae consigo para el personal que trabaja en determinada institución, el hecho de que el control de la asistencia se lleve a cabo de manera manual o por simple observación, se realiza la presente investigación, que consiste en la implementación de un componente de Control de Asistencia del Subsistema de Capital Humano correspondiente al Sistema Integral de Gestión de Entidades Cedrux, que permite controlar los elementos de asistencia, incidencias laborales y es capaz de emitir evaluaciones parciales de acuerdo a los registros de asistencia almacenados. Para cumplir con los objetivos trazados, el sistema brindará la posibilidad de definir las sesiones de producción de la institución y de establecer un horario o régimen de producción (teniendo en cuenta las sesiones creadas) para cada estudiante o trabajador del mismo. Así como de generar reportes en base a los registros de asistencia almacenados. De esta manera al finalizar dicho trabajo quedará implementado el componente de Control de Asistencia perteneciente al subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

**Palabras claves:** Capital Humano, CEDRUX, Régimen de trabajo, Nómina.

# *Tabla de contenido*

# Tabla de Contenido

# *Índice de figuras*

# *Índice de tablas*

# *Introducción*

En el actual siglo, cualquier actividad laboral donde esté presente la entrega de resultados y su evaluación, las personas vinculadas deben estar reguladas en cuanto a distribución de puestos de trabajo, horarios de entrada y salida, roles, evaluaciones y otras características propias de cada centro.

Llevar el Control de Asistencia y Presencia de los empleados de una empresa es una labor muy sencilla en algunos casos y muy compleja en otros.

Sencilla, si se cuenta con una persona que registre el ingreso y salida del personal, es todo lo que podría necesitarse. Pero con este método es claro que se presentarían gran número de dudas sobre la información que se manipula, si el encargado registra las horas y movimientos a su antojo y criterio; además de ser una labor puramente manual y sujeta a continuos errores.

Compleja, cuando se requiera llevar un control de los días de vacaciones tomados por un empleado particular, también tener conocimiento exacto de todos los movimientos (comisiones de servicio, descanso médico, capacitación, estímulos), o cuando la institución tiene en plantilla un gran número de empleados.

Originalmente en el mundo la tarea de identificar al trabajador y mantener el registro de hora de entrada y salida era manual y, en empresas grandes, llevada a cabo por un empleado específico para tal fin, algo que todavía en la actualidad se mantiene vigente en muchas instituciones, constituyendo esto una tarea agotadora y engorrosa la cual maneja un gran papeleo.

Contar con una solución para la administración y control de asistencia en las empresas es de vital importancia. Este tipo de herramientas agiliza, controla y facilita la elaboración de la nómina a través de una buena definición de las reglas y políticas de negocio, que deben de regir en el ámbito de la administración del personal.

El dato principal para llevar a cabo el control de asistencia, junto con el registro de entrada y salida, es el horario o régimen de trabajo que indica cuando se espera que la persona se encuentre trabajando; de modo que comparando el horario con los registros de entradas y salidas se llevan a cabo las tareas de control. La responsabilidad de este proceso suele recaer sobre la oficina de Capital Humanos de la empresa.

En el Sistema Integral de Gestión CEDRUX no existe actualmente una aplicación para realizar este control capaz de integrarse al componente de nómina, resulta muy difícil realizar esta tarea cuando se tiene que introducir los datos manualmente, esto trae consigo atraso, pues se depende de la entrega de esta información en tiempo, pudiendo ser transformada con fines lucrativos o ser víctima de múltiples errores. Esto trae como consecuencia que la nómina no contempla las afectaciones de los trabajadores y por esta causa se suele pagar a trabajadores que tuvieron ausencias en el período por estos días no trabajados. Por esta razón se plantea como **problema a resolver:** La gestión manual del Control de Asistencia para la Administración de Capital Humano dificulta generar la nómina mediante el subsistema de Capital Humano del Sistema Integral de Gestión Cedrux.

**Objetivo General**: Implementar el componente Control de Asistencia para integrarlo al proceso de generación de la nómina del subsistema de Capital Humano del Sistema Integral de Gestión CEDRUX.

**Objeto de estudio**: Implementación de Administración de Capital Humano.

**Campo de acción**: Implementación del Control de Asistencia.

**Idea a defender**: La implementación del componente Control de Asistencia, contribuirá a disminuir las dificultades en el proceso de generación de la nómina en el subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

**Objetivos específicos:**

- ✓ Determinar el marco teórico de la investigación para conocer las principales tendencias (y autores) en cuanto a la gestión del Control de Asistencia para la Administración del Capital Humano.
- ✓ Implementar los requerimientos especificados para los procesos de Control de Asistencia.
- ✓ Validar el componente obtenido.

**Posibles resultados:** Un componente capaz de realizar el Control de Asistencia integrado al subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

**Se ha estructurado la información expuesta en la investigación en tres capítulos.**

**Capítulo 1 Fundamentación teórica**, se realiza un estudio del estado del arte de las principales tendencias, técnicas, tecnologías, metodologías y software usados para la solución del problema que se enfrenta en el ámbito nacional e internacional, dándole una valoración a cada uno de estas soluciones.

**Capítulo 2 Propuesta de Solución**, se elaboró una crítica al diseño propuesto por el analista, dado la importancia del mismo para la implementación. Se describe uno de los algoritmos más complejo con el que cuenta la aplicación explicando además la estructura de datos usada en el mismo, se describen las clases que fueron definidas, que sirven de documentación a los que den continuidad a la implementación del software y como se integran las mismas.

**Capítulo 3 Validación de la Propuesta**, se valida la solución propuesta a través de la realización de pruebas, el mismo se encuentra dividido 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.

# *Capítulo 1: Fundamentación Teórica*

## Introducción.

En el presente capítulo se abordan los fundamentos teóricos generales del proceso de control de asistencia, que sirven de punto de partida a la comprensión de la propuesta; junto a ello se realiza un estudio del estado del arte de los distintos sistemas para controlar la asistencia y su utilidad en las diferentes esferas en las que se emplean. Además, se detallan las tendencias y tecnologías actuales, metodologías y herramientas para el desarrollo de la solución propuesta.

La finalidad de un sistema para control de asistencia es mantener el registro de la hora de entrada y salida del personal y la responsabilidad del mismo suele recaer sobre la oficina de Capital Humano de la empresa.

Los objetivos pueden ser alguno o varios de los siguientes:

- ✓ **Controlar la asistencia**: Quién vino y quién está ausente.
- ✓ **Controlar la puntualidad**: Quién llegó tarde y quién fue puntual.
- ✓ **Calcular las horas trabajadas**: y opcionalmente categorizarlas en normales, extras al 50%, extras al 100%, nocturnas.
- ✓ **Analizar las horas no trabajadas**: con la posibilidad de agrupar y totalizar por causas (vacaciones, enfermedad**).**
- ✓ **Mantener estadísticas y cuentas corrientes de ausentismo por personas**.

## 1.1 Sistemas de control de asistencia existentes.

Empresas de todos los tamaños requieren contar con herramientas de gestión para controlar y cuantificar los distintos aspectos de su operación. Una de las áreas que más ha tardado en automatizarse ha sido la del Control de Asistencia y/o Permanencia del personal, debido en gran medida por la aún amplia utilización de sistemas tradicionales de control basados en los antiguos relojes mecánicos que utilizan tarjetas de cartón, o únicamente con “partes de asistencia” que no

son otra cosa que hojas de papel con líneas numeradas en las que los empleados firman e indican (ellos mismos) la hora en la cual ingresan o salen de la empresa.

Los Sistemas de Control de Asistencia modernos se basan en Tecnologías de Identificación Automática con Códigos de Barras, Banda Magnética, Tarjetas de Proximidad por radio frecuencia (RFID) e incluso Sistemas Biométricos de Huella Digital. Todos ellos solo forman una parte de la solución debido a que el componente fundamental es el Software de Control de Asistencia, pues los datos capturados con los distintos modelos de lectores necesitan ser procesados para recién entonces llegar a convertirse en información (tardanzas, inasistencias, horas extras, etc.) (Villegas 2004).

### 1.1.1 Sistemas automatizados empleados en el mundo para control de asistencia.

#### 1.1.1.1 E-time Software para Control de Tiempo y Asistencia.

Calcula automáticamente las horas trabajadas de sus empleados, así como sus tiempos extras, considerando horas de comida, ausentismos, vacaciones y retardos.

**Características relevantes de operación:**

- ✓ Rotación automática de turnos en base a una programación.
- ✓ Cambios globales de turnos de trabajo, por emergencias.
- ✓ Exportación de reportes a múltiples formatos (Excel, Word, CSV, pdf, etc.).
- ✓ Archivo de prenómina con formato configurable.
- ✓ Archivo de registros de entradas y salidas (“chequeadas”) con formato configurable.
- ✓ Maneja hasta 10 clasificaciones del sistema (Ej., área, centro de costos, categorías, tipos de empleados) configurables por el usuario, las cuales se pueden agrupar hasta en cinco jerarquías.
- ✓ Programación de movimientos directamente desde la pantalla del Empleado.
- ✓ Manejo de fotografía digitalizada como parte de los datos de empleado.

- ✓ Módulo opcional integrado de diseño e impresión de credenciales, multiusuario y multiestación, usa la base de datos de E-time, y maneja impresoras de tarjetas de PVC o Poliéster Teslin. El gafete puede imprimirse directamente desde la pantalla de consulta de un empleado.
- ✓ Uso de motivos en autorizaciones de tiempo extra, bajas de empleados, incapacidades, permisos, suspensiones para reportes estadísticos.
- ✓ Seguridad a nivel módulo, acción (agregar, modificar y borrar), tiempo de inactividad y caducidad de contraseña.
- ✓ Módulo de comedor (consumo y platillos).
- ✓ Manejo de ligas entre supervisores y trabajadores.
- ✓ Almacena y administra las huellas digitales en equipos Bioscrypt a través del software. (Software 2002; Support 2002)

#### 1.1.1.2 SICON Software Avanzado de Control de Asistencia.

**Características** (Design-soft 2005; SOFT 2005)**:**

- ✓ **Acceso al sistema:** cada usuario ingresa al SICON mediante su contraseña, la cual le permite acceder únicamente a los módulos previamente autorizados por el administrador del sistema.
- ✓ **Consulta de marcadores y asistencias:** consulta de la asistencia de un empleado directamente en una sola pantalla, indicando la hora del mes actual o meses anteriores, desde aquí también se pueden realizar las justificaciones respectivas de un empleado.
- ✓ **Programación de vacaciones y saldo vacacional:** controla los días solicitados a cuenta de vacaciones para el año en curso o a futuro. Vacacional indicando número de días disponibles de tomar.
- ✓ **Gestión de permisos y Movimientos:** impresión de listados indicando múltiples criterios de selección. SICON permite llevar un control de los distintos movimientos y permisos del

empleado, permitiendo topes máximos para cada uno de ellos en el año indicando si son con o sin Goce.

- ✓ **Captura de datos:** lee los datos provenientes de cualquiera de los lectores que DESIGN SOFT S.A.C (empresa peruana formada en el año 2001) comercializa. Entre los dispositivos compatibles se encuentran: lectores de barras, de huellas digitales, 4030, DR519 y FP4500.
- ✓ **Sistema operativo:** Windows XP.

#### 1.1.1.3 Doce versión 2.

Es un programa de control de asistencia, que incorpora un registro de datos del personal de una empresa o institución. El control de asistencia se efectúa usando el teclado de una computadora en el cual el personal registra un código individual de la misma manera en que se podría hacer con un reloj tarjetero. Permite utilizar una cámara Web para que tome fotografías de la persona que marca su entrada y salida de manera que, posteriormente, a través de una revisión manual sea posible identificar algún fraude cometido por personas que marcan a cuenta de otras. (Paez 2008)

Los reportes que el programa puede generar son:

- ✓ Varias nóminas.
- ✓ Historial.
- ✓ Reporte de atrasos y faltas.
- ✓ Tiempo no trabajado.
- ✓ Control de marcas en el reloj.
- ✓ Selección manual.
- ✓ Asistencia de personal por día.
- ✓ Planilla mensual.
- ✓ Cuadro mensual exportado a Excel.

#### 1.1.1.4 SmartClk Control de Asistencia.

Es un programa diseñado para asistir en la gestión de control horario y de asistencia de personal que permite evitar errores de cálculo producidos en el control de asistencia por métodos no informatizados de forma fácil, eficaz y eficiente. Además de la incorporación de registros horarios de cualquier reloj electrónico en sus bases de datos ayudando al operador en la depuración de las marcas de asistencia y el control de los horarios mediante la incorporación de herramientas efectivas y el ingreso de registros de justificación de falta por distintos conceptos. Los mismos pueden crearse en el sistema especificando el tipo de concepto para la justificación de falta. Ejemplos de estos son licencias, faltas con aviso, etc. (A&M 2010)

**Características y funciones:**

- ✓ **Seguridad:** incorpora elementos de seguridad que permiten el registro de los operadores y todas las operaciones realizadas en el programa. Esta información se almacena en un registro de auditoría el cual puede analizarse posteriormente a los efectos de controlar la buena gestión de los datos. El ingreso a la aplicación es mediante usuario y contraseña.

- ✓ **Horarios:** permite el control de horarios fijos y rotativos fijos, en los cuales se debe especificar la hora de entrada, de salida y los días o el período que se debe controlar; horarios libres, en los cuales no se especifica más que una carga horaria y los días que debe cumplir esa carga horaria; y horarios presencia, donde el sistema controla que esas personas por lo menos cuente con una marca indicando su presencia. En todos los casos el sistema soporta horarios que pasen la media noche.

- ✓ **Informes:** posibilita emitir informes de inconsistencias, inasistencias, flash de asistencia, diferencias de horario (entradas tardes / salidas anticipadas), cálculo de horas trabajadas y planilla resumen. Los informes permiten gran flexibilidad para la selección, filtro de registros (filas y columnas), impresión y vista previa de impresión.

- ✓ **Carga de Datos Externos:** proporciona la carga de los datos iniciales del sistema a partir de archivos de datos en formato de texto externos. Soporta la carga de funcionarios, secciones y horarios. Adicionalmente puede leer los registros de asistencia de archivos de texto generados por cualquier reloj de control.

- ✓ **Soporte para equipos biométricos:** a partir de la versión 3.11 se incluye la lectura de marcas y la administración de las huellas para verificación biométrica para terminales Identix Finger Scan V20 directamente en la aplicación. Para otros equipos se puede integrar con las

aplicaciones Smart Lan para terminales F4/A4 y relojes de tarjetas magnéticas y de proximidad RTA600.

**1.1.1.5 Exactus Pro 2010.**

Es un efectivo sistema de control de asistencia basado en un ordenador que registra la marcación del personal a través de un código de barra, una tarjeta magnética u otro medio de lectura. El programa brinda una capacidad de manejo ilimitado tanto de turnos y/o horarios por empleado, como de empleados por estación, adaptación inmediata a sistemas multiusuario y emisión de múltiples reportes de asistencia/ausencias en línea, sin necesidad de procesos previos. (ArchivosPC 2006; Pro 2006)

**Características y funciones:**

- ✓ Ofrece diversos informes: tiempo trabajado, ausencias, tardanzas, sobre tiempo y todos aquellos reportes que se deseen adicionar. El sistema registra las marcaciones a través de la lectura de un código de barra (o lector magnético) impreso en tarjetas de marcación personalizadas llamadas Exactus Pro Cards.

- ✓ Es compatible con relojes biométricos U300, TFT 500, F7, F4, entre otros.

- ✓ Puede mantener un turno definido para cada día del mes por cada empleado. Mantiene los datos de todos los empleados que trabajan en su empresa y la historia de todas las marcaciones registradas por si se requiriese alguna comprobación posterior.

- ✓ Capacidad de migrar datos de tiempo a cualquier programa de planilla gracias a la flexibilidad de su interfaz. Permite revisar e importar las marcaciones desde un Exactus Pro 2010 a otra terminal dentro de una red o remotamente desde otra sucursal a través de módem.

- ✓ Controles de acceso que brindan la seguridad necesaria para salvaguardar la información. Ningún empleado podrá marcar una vez vencido su contrato de trabajo e igualmente no podrá hacerlo antes de su fecha de ingreso.

- ✓ Permite administrar el tiempo que su personal toma como Coffee Break (pausa para el café) así como los constantes permisos para asuntos personales.

- ✓ Permite ver en tiempo real todos los informes del sistema.

- ✓ Posee generador de reportes, calculadora, calendario, verificador de archivos.
- ✓ Mensajería Interna.
- ✓ Sistemas operativos soportados: Windows 2000, Windows XP y Windows 7.

**1.1.1.6 Aplicación Web del Control Horario - Universidad de Zaragoza.**

Esta aplicación de control del horario se emplea en la actualidad en la Universidad de Zaragoza, España, para lograr un adecuado cumplimiento de la normativa en materia de calendario laboral, jornada y horario de los trabajadores de este centro. Se utiliza el término fichaje como sinónimo de marcaje, ambos términos definen una incidencia de una persona en una fecha y hora determinada, realizada desde un lector o desde la propia aplicación. El personal de administración y servicios, en general, puede realizar fichajes mediante un ordenador personal, desde su puesto de trabajo, ver sus fichajes y visualizar su cuadro mensual de bloques de trabajo una vez revisado por el responsable de su unidad. Los responsables de las unidades se encargarán de la supervisión del cumplimiento de la jornada y horario del personal a su cargo, así como del control de las diferentes causas de inasistencia al trabajo. (Zaragoza 2001)

Existen dos tipos fundamentales de incidencias:

1. Incidencias dentro de la jornada laboral: el fichaje de las entradas y salidas y demás incidencias es responsabilidad de cada persona y pueden ser realizadas desde el lector o desde la Web (los errores u olvidos deben ser corregidos por los responsables de unidad).
2. Ausencias planificadas: estas incidencias pueden corresponder, en algunos casos, a días completos y es el responsable de unidad quien debe introducir este tipo de incidencias en la aplicación.

Para el desarrollo de esta aplicación se emplearon tecnologías basadas en la Web, con una interfaz y navegación homogénea con el resto de las aplicaciones que viene desarrollando el Servicio de Informática, se deberá contar con un navegador Web (Microsoft Internet Explorer o Netscape Navigator) para poder utilizarla. Los desarrolladores recomiendan la instalación de la última versión disponible del navegador seleccionado.

Entre las principales funcionalidades que brinda la aplicación se encuentran las tres siguientes:

1. **Fichar ahora:** esta opción permite fichar desde cada ordenador seleccionando el código correspondiente a la incidencia en un menú desplegable. La aplicación informática identifica la máquina desde la que se realiza la operación de marcaje.

2. **Ver fichajes:** desde esta opción se consultan los fichajes realizados por cada usuario entre dos fechas determinadas. La modificación de los fichajes es función de los responsables de la unidad. Se podrán ver los fichajes realizados entre las fechas seleccionadas indicando: el día de la semana, la fecha, la hora, la duración de la jornada de ese día si es una ausencia planificada, el tipo de marcaje mediante un icono (lector o Web), la procedencia (una etiqueta que identifica el lector o la dirección IP (*Internet Protocol*) del ordenador desde el que se efectuó el fichaje), quién realizó ese fichaje mediante un icono (el propio usuario o el administrador) y el código y la incidencia asociada.

3. **Cuadros mensuales**: desde esta opción se consultan los horarios de entradas, salidas e incidencias en un mes determinado para cada usuario. No estará disponible si existe algún error (por esa razón no podrá verse dentro de la jornada de trabajo, hasta que se cierre el bloque de salida aunque todo esté correcto). Se indica: día de la semana, fecha, hora de entrada y salida, el número de horas de presencia real (Hp), el número de horas de incidencias (Hi), el número de horas de ausencia por motivos particulares (Haap), el número de horas teóricas justificadas (Ht=Hp+Hi), los códigos de incidencia usados en el bloque (Ci), las horas trabajadas el sábado (Hs), las horas extraordinarias (He), las horas en exceso de jornada irregular (Hx) y las horas de recuperación de ausencias particulares (Hrap).

### 1.1.2 Análisis de los sistemas empleados en el mundo.

Los software SICON, E-time, SmartClk y Exactus Pro 2006 brindan muchas funcionalidades relacionadas con el control de la asistencia, pero requieren leer los datos provenientes de terminales colectoras de datos, usando tarjetas con código de barras, proximidad, banda magnética, dispositivos biométricos u otros similares. Debido al hecho de que todos presentan licencia privativa, imposibilita reutilizarlos para dar solución al problema de la investigación empleando uno de estos sistemas.

Por otra parte Doce versión 2 no necesita contar con lectores de datos para la captura de los mismos, pues en este sistema el control de asistencia se efectúa usando el teclado de una computadora en la cual el personal registra un código individual de la misma manera en que se

podría hacer con un reloj tarjetero. Se considera que de implantarse este sistema pondría de manifiesto la limitación de que cada estudiante o trabajador tendría que marcar su entrada sólo desde una computadora. Esto haría imposible poder determinar cuántas horas el usuario trabaja en el laboratorio en un período determinado en su computadora por lo que tampoco se podría calcular el uso real de la tecnología. Este software al igual que la Aplicación Web del Control Horario de la Universidad de Zaragoza, posee funcionalidades que se adaptan a las necesidades de la presente investigación; pero se imposibilita la adecuación a los requerimientos de la situación problemática porque ambos poseen licencia privativa.

### 1.1.3 Sistemas automatizados utilizados en Cuba para control de asistencia.

#### 1.1.3.1 Sistema de Recursos Humanos (Fastos).

El sistema de Recursos Humanos (Fastos) desarrollado por DESOFT, perteneciente a la empresa nacional de software del Ministerio de Informática y las Comunicaciones, está formado por los módulos Configuración, Personal, Capacitación y Cuadros, permite controlar las informaciones fundamentales de los empleados de una entidad, también realizar varios procesos y operaciones que son inherentes al área de recursos humanos, tales como (Desoft 2008):

- ✓ Registro de los empleados: se guardan los datos de los empleados, así como informaciones referentes a los reportes de vacaciones, certificados médicos, licencias, resolución.
- ✓ Control de la plantilla: permite establecer la estructura organizativa de las plazas de la entidad.
- ✓ Control de asistencia: incorpora el control de claves de asistencias, turnos de trabajos, horarios, tarjetas de asistencia e incidencias de cada empleado. Además, posibilita acoplar relojes (RTA 600) para actualizar la información de la tarjeta de asistencia de forma automática.
- ✓ Informes y modelos: proporcionan la obtención de un total de 56 informes, por ejemplo cierre del período, análisis de fondo de tiempo, estadísticos, entre otros y 11 modelos entre los que se encuentran: el acta de entrega del expediente laboral, contratos de trabajo, guía de documentos archivados, convenio de trabajadores, etc.

- ✓ Control de la capacitación respecto a: acciones de capacitación, estudios realizados, cursos, eventos, experiencia docente, publicaciones, conocimientos, idiomas extranjeros, plan de desarrollo, informes y otros aspectos.
- ✓ Control de la información de los cuadros: se establece el registro de los cuadros, dirigentes y reserva, referente a evaluaciones, inspecciones, sanciones, necesidades de capacitación, entre otros.

**El sistema tiene las siguientes características generales:**

- ✓ Aplicación cliente servidor, con base de datos de SQL 2000 Server.
- ✓ Ayuda incorporada.
- ✓ Protección contra copias ilegales.
- ✓ Control y registro de acceso al sistema.
- ✓ Facilidades para la exportación de información.

**Requerimientos para la explotación del producto:**

- ✓ Las PC (Personal Computer) deben tener instaladas el sistema operativo Windows 98 o superior, 256MB RAM como mínimo.
- ✓ Sistema de redes estable y confiable pues este permitirá tener acceso a la información con mayor rapidez y seguridad.
- ✓ Gestor de Base de Datos SQL 2000.

#### 1.1.3.2 “GREHU: Un sistema integral para gestionar los Recursos Humanos”.

Este software desarrollado en el Instituto Superior Politécnico “José Antonio Echeverría”, constituye el resultado de un estudio realizado en la temática de Gestión de los Recursos Humanos, obteniéndose una herramienta de software que permite registrar, procesar y gestionar de forma integrada las principales funciones que se desarrollan en la dirección de Recursos Humanos de una entidad laboral entre ellas la del control de la asistencia de los trabajadores. El software desarrollado permite realizar de forma automatizada e integrada las principales funciones que se

realizan en la Dirección de Recursos tales como: el inventario de personal, el control de las sanciones y amonestaciones, la selección y contratación, la evaluación del desempeño, el trabajo con los dirigentes y cuadros y el procesamiento de las nóminas. El sistema informatizado, conjuntamente con los procedimientos metodológicos propuestos se está explotando desde hace algunos años en las Direcciones de Recursos Humanos y de Personal de varias instalaciones de Cubanacán S.A y de la Cadena Hotelera de Gran Caribe, obteniéndose resultados satisfactorios, según opiniones de los funcionarios de estas instalaciones. Con su utilización, se controla, almacena y obtiene una gran cantidad de información sobre el Capital Humano en particular, pueden realizar múltiples inferencias, para la toma de decisiones oportunas con datos actualizados. (Rodríguez 1998)

Este sistema, con sus innumerables salidas tanto estadísticas como gráficas, hace posible que en mayor o menor medida, los directivos relacionados con la gestión de personal, puedan conocer y prever las posibles promociones, necesidades de formación y capacitación, los reclutamientos futuros, el comportamiento de la disciplina laboral, el desempeño del personal, así como el desarrollo y un control efectivo de la asistencia a la empresa. El sistema fue programado en FoxPro para WINDOWS versión 2.6.

### 1.1.4 Análisis de los sistemas utilizados en Cuba.

El Sistema de Recursos Humanos Fastos (Desoft 2008), permite controlar las informaciones fundamentales de los empleados de una entidad, también realizar varios procesos y operaciones que son inherentes al área de recursos humanos. Pero necesita tener instalado el sistema operativo Windows 98 o superior, lo cual no favorece la migración a software libre. GREHU (Rodríguez 1998) es un sistema integral para gestionar los Recursos Humanos, brinda la posibilidad de realizar de forma automatizada e integrada las principales funciones que se llevan a cabo en la Dirección de Recursos tales como: el inventario de personal, el control de las sanciones y amonestaciones, la selección y contratación, la evaluación del desempeño, el trabajo con los dirigentes y cuadros y el procesamiento de las nóminas. Se considera que la principal deficiencia de este sistema estriba en que no captura de forma automática las inasistencias e impuntualidades, por lo que en los centros donde está implantado este sistema se debe realizar por medio de un libro de firmas. Además, ambos sistemas presentan la limitación de que sólo pueden operar sobre sistema operativo Windows.

## 1.2. Tendencia y tecnología actual para el desarrollo de la aplicación Web.

Teniendo en cuenta las necesidades anteriormente planteadas y las características del entorno donde se aplicará la solución propuesta, se realizó un estudio de las tendencias y tecnologías actuales posibles a emplear, puntualizadas a continuación.

### 1.2.1. Lenguajes del lado del cliente.

#### 1.2.1.1 Tecnología AJAX

AJAX acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (JavaScript asíncrono y XML), es una técnica de desarrollo Web para crear aplicaciones interactivas o **RIA** (Rich Internet Applications). 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. De esta forma es posible 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. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. (Team 2006)

AJAX está conformado por:

- ✓ XHTML y CSS, para crear una presentación basada en estándares.
- ✓ DOM, para la interacción y manipulación dinámica de la presentación.
- ✓ XML, JSON, para el intercambio y la manipulación de información.
- ✓ XMLHttpRequest, para el intercambio asíncrono de información.
- ✓ JavaScript, para unir todas las demás tecnologías.
- ✓ Provee un mecanismo para mezclar y hacer coincidir XML con XHTML.
- ✓ Las aplicaciones son más rápidas e interactivas, al estilo aplicaciones de escritorio.

- ✓ Reduce de manera significativa tener que cargar información continuamente del servidor, actualizando solamente porciones de la página.

Cuando se utiliza AJAX adecuadamente en el desarrollo de una aplicación, se reduce de manera significativa los tiempos de carga inicial.

#### 1.2.1.2 CSS.

*Cascade Style Sheet* u Hoja de Estilos es un lenguaje formal que define cómo se va a mostrar un documento escrito en HTML o XML. Establece la separación definitiva de la lógica (estructura) y el físico (presentación) del documento. El *World Wide Web Consortium* (W3C) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores. (CSS 2006)

#### 1.2.1.3 XHTML.

XHTML es el acrónimo en inglés de Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto). Es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de remplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. Está encaminado al uso de un etiquetado correcto, por lo que exige una serie de requisitos básicos a cumplir en cuanto al código. (XHTML 2010)

Algunos de estos requisitos son:

- ✓ Elementos correctamente anidados.
- ✓ Etiquetas en minúsculas.
- ✓ Elementos cerrados correctamente.
- ✓ Atributos de valores entrecomillados.

#### 1.2.1.4 JavaScript.

Es un lenguaje orientado a objetos, interpretado, que se ejecuta del lado del cliente. Utilizado principalmente en páginas Web y con sintaxis semejante a Java o C, sigue la programación basada en prototipos. Además usa DOM (Document Object Model) para acceder y modificar el contenido,

estructura y estilo de los documentos HTML y XML. Dentro de las principales tecnologías para interactuar con DOM que usa Javascript se encuentran AJAX y DHTML. (Pérez 2008)

#### 1.2.1.5 JSON.

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. (JSON 2010)

JSON está constituido por dos estructuras:

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

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

### 1.2.2. Lenguajes del lado del servidor.

#### 1.2.2.1 PHP.

PHP (*Hypertext Preprocessor*) es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Contiene infinidades de bibliotecas con funciones orientadas al manejo de bases de datos, imágenes, películas flash, directorio, entre muchas otras. Soporta además la programación orientada a objetos y es una alternativa potente para la creación de portales Web muy dinámicos. PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo GNU&Linux, muchas variantes Unix (HP-UX, Solaris y Open BSD), Microsoft Windows, Mac OS X y RISC OS. Soporta la mayoría de servidores Web de hoy en día, incluyendo Apache, *Microsoft Internet Information Server* (Microsoft

IIS), Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. Quizás la característica más destacable de PHP es su soporte para una gran cantidad de bases de datos. Destacan entre ellas MySQL, PostgreSQL, Oracle (OCI7 y OCI8), Sybase, ODBC, FrontBase, Unix dbm, Adabas D, entre otras.

### 1.2.3. Gestores de Bases de Datos.

#### 1.2.3.1 PostgreSQL.

Gestor de base de datos distribuido bajo la licencia BSD, relacional y multiplataforma. Sostenida por una activa comunidad de desarrolladores denominada PostgreSQL Global *Development Group*. Presenta una amplia gama de características como *triggers*, vistas, integridad transaccional, variedad de tipos de datos (texto de largo ilimitado, *arrays*, direcciones IP y MAC, figuras geométricas) y con posibilidad de que los usuarios puedan crear los suyos propios. Lenguajes como C, C++, Java, Perl, Python, Ruby, PHP, entre otros contienen librerías que permiten usar este gestor. (Group 2010)

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.

**PgAdmin.**

La aplicación se puede utilizar en Linux, FreeBSD, Solaris, Mac OSX y Windows para administrar PostgreSQL 7.3 y anteriores que se ejecutan en cualquier plataforma, así como las versiones comerciales y derivados de PostgreSQL como Postgres Plus Advanced Server y base de datos Greenplum.

Características incluidas:

- ✓ Entradas SQL aleatorias.
- ✓ Pantallas de información y 'Ayudas' para bases de datos, tablas, índices, secuencias, vistas, programas de arranque, funciones y lenguajes.
- ✓ Preguntas y respuestas para configurar Usuarios, Grupos y Privilegios.
- ✓ Control de revisión con mejora de la generación de script.
- ✓ Configuración de las tablas de Microsoft MSysConf.
- ✓ `Ayudas´ para importar y exportar datos.
- ✓ `Ayuda´ para migrar Bases de datos.
- ✓ Informes predefinidos en bases de datos, tablas, índices, secuencias, lenguajes y vistas.

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir simples consultas SQL para el desarrollo de bases de datos complejas. La interfaz gráfica compatible con todas las características de PostgreSQL y facilita la administración. La aplicación también incluye un editor de resaltado de sintaxis SQL, un editor de código del lado del servidor, una sentencia de SQL / lote / shell de trabajo agente de programación, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor se puede hacer a través de TCP / IP o Unix Domain Sockets (en plataformas * nix), y puede ser encriptado SSL para la seguridad. No se requieren drivers adicionales para comunicarse con el servidor de base de datos. (PgAdmin 2008)

Es desarrollado por una comunidad de expertos de PostgreSQL en todo el mundo y está disponible en más de una docena de idiomas. Es un software libre publicado bajo la licencia de PostgreSQL

### 1.2.4 Herramientas para el desarrollo del software.

#### 1.2.4.1 NetBeans IDE.

Primer IDE que brinda soporte completo para Java EE 6 y *Sun Glass Fish Enterprise Server* v3 y ofrece PHP mejorado, soporte para JavaFX y C/C ++. Además ofrece soporte completo para Java ™ Platform Enterprise Edition 6 (Java EE 6) y *Sun Glass Fish™ Enterprise Server* v3, así como otras funcionalidades innovadoras. Junto con el soporte para Java EE 6 y *Glass Fish* v3, el NetBeans IDE 6.8 ofrece otras nuevas características y mejoras que incluyen:

Soporte PHP Ampliado: expande el soporte de los lenguajes dinámicos con apoyo para PHP 5.3 y el esquema de Symfony acelera el desarrollo de aplicaciones Web PHP.

Una integración más ajustada con Project Kenai: Project Kenai, un entorno de colaboración para acoger proyectos Open Source, ofrece ahora soporte completo para JIRA así como mensajería instantánea mejorada y una integración de seguimiento de issues.

Mejora de C / C + + *Profiling*: perfila y sintoniza aplicaciones C / C + + con el nuevo indicador *Microstate Accounting*, supervisor de uso I/O.

JavaFX ™: código de finalización mejorado, sugerencias y navegación para JavaFX en el editor NetBeans. (Infosertec 2009)

**1.2.4.2 Firebug.**

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

Firebug no es un simple inspector como DOM Inspector, además edita y permite guardar los cambios. Su atractiva e intuitiva interfaz, con pestañas específicas para el análisis de cada tipo de elemento (consola, HTML, CSS, Script, DOM y red), permite al usuario un manejo fácil y rápido. Firebug está encapsulado en forma de plug-in o complemento de Mozilla, es *Open Source* y de distribución gratuita. (Evolved 2010)

**1.2.4.3 Subversion TortoiseSVN.**

Subversion (SVN) es un software libre desarrollado para el control de versiones. Es un sistema centralizado para compartir información que permite realizar modificaciones atómicas y gestionar archivos, directorios y sus cambios a través del tiempo, lo que facilita las tareas administrativas. Su capacidad para que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. TortoiseSVN es el cliente gratuito para el sistema de control de versiones Subversion, con código abierto y software libre bajo la licencia GNU GPL. Está disponible en 28 idiomas diferentes y puede ser usado sin un entorno de desarrollo.

#### 1.2.4.4 Visual Paradigm.

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

### 1.2.5 Metodologías.

Se define como metodología al conjunto de estrategias, procedimientos, métodos o actividades intencionadas, organizadas, secuenciadas e integradas, que permitan el logro de aprendizajes significativos y de calidad. A su vez, una metodología para el desarrollo del proceso de software es el conjunto de actividades necesarias para transformar los requisitos de los usuarios en un sistema software. En el vertiginoso mundo de la informática, por lo riesgoso y difícil de controlar que es un proceso de software, el uso de metodologías que apoyen el mismo, es un tema muy discutido. Una correcta elección evitaría entre otros problemas, pérdida de tiempo, capital, insatisfacción en clientes y desarrolladores o peor aún, el colapso mismo del producto. En realidad no existe una metodología obligatoria para un proyecto, sino que las características de cada proyecto, las de un equipo de desarrollo, recursos disponibles y tiempo para su elaboración exigen la flexibilidad del proceso, adaptándose el mismo al entorno y teniendo como objetivo alcanzar la máxima calidad en lo que se produce. Hoy en día, dentro de las más conocidas encontramos: *Rational Unified Process* (RUP), *eXtreme Programming* (XP) y *Feature Driven Development* (FDD); en su traducción al español, Proceso Unificado de Desarrollo, Programación Extrema y Desarrollo Guiado por Funcionalidad, respectivamente. (Molpereces 2003; Mendoza Sanchez 2004; Salesiana 2008)

#### 1.2.5.1 El Modelo de desarrollo.

El modelo usado para el desarrollo de los proyectos del CEIGE (Centro de Informatización de la Gestión de Entidades) fue uno propio desarrollado por el centro, el cual fue el modelo de desarrollo orientado a Componentes combinado con el Iterativo y el Incremental. Este es un modelo de desarrollo orientado a las necesidades y artefactos generados durante el proceso de desarrollo del programa ERP-Cuba. Es una combinación de diferentes metodologías de las cuales se ha tomado lo que sería más conveniente. (Elianys Hurtado Sola 2008)

**Desarrollo iterativo e incremental**

Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento.

**Desarrollo basado en componentes**

Nos lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo.

### 1.2.6 Librerías y Marcos de trabajo.

El desarrollo de la solución se realizará utilizando el marco de trabajo Sauxe, desarrollado por el Departamento de Tecnología del Centro de Informatización de la Gestión de Entidades (CEIGE), el cual contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo.

Sauxe cuenta con una arquitectura en capas que a su vez presenta en su capa superior un MVC. Contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Siguiendo el paradigma de independencia tecnológica por el cual apuesta el país, reutiliza las tecnologías libres mostradas en la Figura 1.

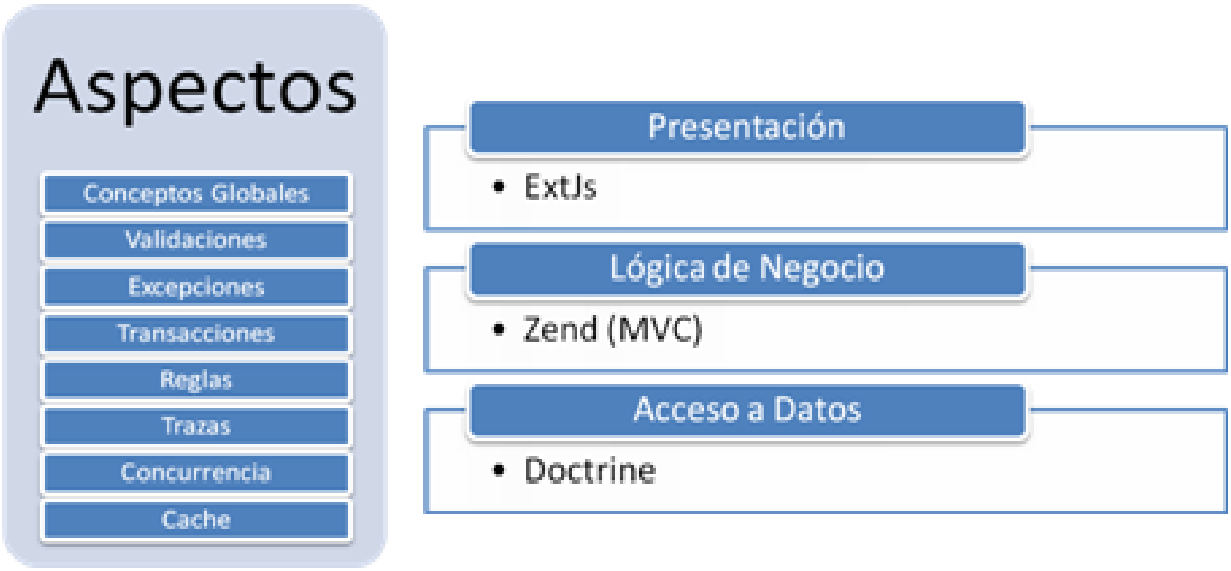


*Figura 1: Arquitectura de Sauxe.*

.

**Modelo Vista Controlador (MVC)** es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. (Baryolo 2010)

**1.2.6.1 ExtJS.**

ExtJS permite que con pocas líneas de código sea posible realizar interfaces amigables para los usuarios. Es una librería muy avanzada para el desarrollo rápido de aplicaciones con una apariencia totalmente novedosa y una arquitectura flexible. ExtJS es una librería Javascript que permite construir aplicaciones complejas en Internet. (Frederick 2008.)

Esta librería incluye:

- ✓ Componentes UI del alto performance y personalizables.
- ✓ Modelo de componentes extensibles.
- ✓ Un API fácil de usar.
- ✓ Licencias Open Source y comerciales.

#### 1.2.6.2 Zend Framework.

Se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP, brinda soluciones para construir sitios Web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. A diferencia de CakePHP que trabaja con PHp 4 y PHP 5. Este framework está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Los componentes son varios y variados y aunque alguno es posible que no se use nunca, hay otros que puede que se usen hasta la saciedad, por ejemplo el componente para la BD. Entre los componentes que se encuentran tienen vital importancia: Zend_Config para temas de configuración de aplicaciones Web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed entre otros. La instalación es sencilla, tan solo se tendrá que añadir en el fichero de configuración php.ini, el path hasta la carpeta library del framework con la instrucción include_path. (Esser 2009.)

#### 1.2.6.3 Doctrine.

Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado. Se está empezando a ver su potencial, pero de la documentación se puede decir que tiene todas las características necesarias para ser funcional en casi cualquier proyecto. Entre muchas otras cosas brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Por otro lado, como la librería es bastante grande ésta tiene un método para ser 'compilada' al pasar a producción. (Aquino 2009)

Cuando se trabaja con Doctrine, se necesita informar a su motor interno de cuál es el modelo de la aplicación, para ello se puede hacer ingeniería inversa de la base de datos existente, o si se empieza la aplicación desde 0, crear el modelo en la sintaxis específica que nos propone Doctrine y luego generar toda la base de datos. Para crear el modelo, doctrine brinda dos alternativas, hacer una clase por tabla e indicarle mediante PHP el tipo de datos que almacenaremos en él.

Ventajas que nos facilitan enormemente tareas comunes y de mantenimiento:

- ✓ **Reutilización:** La principal ventaja que aporta un ORM es la reutilización permitiendo llamar a los métodos de un objeto de datos desde distintas partes de la aplicación e incluso desde diferentes aplicaciones.

- ✓ **Encapsulación:** La capa ORM encapsula la lógica de los datos pudiendo hacer cambios que afectan a toda la aplicación únicamente modificando una función.
- ✓ **Portabilidad:** Utilizar una capa de abstracción nos permite cambiar en mitad de un proyecto de una base de datos MySQL a una Oracle sin ningún tipo de complicación. Esto es debido a que no se utiliza una sintaxis MySQL, Oracle o SQLite para acceder al modelo, sino una sintaxis propia del ORM utilizado que es capaz de traducir a diferentes tipos de bases de datos.
- ✓ **Seguridad:** Los ORM suelen implementar mecanismos de seguridad que protegen nuestra aplicación de los ataques más comunes como SQL Injection.
- ✓ **Mantenimiento del código:** Gracias al correcto ordenamiento de la capa de datos, modificar y mantener el código es una tarea sencilla. (Aquino 2009; Smith Apr 2009)

## Conclusiones del Capítulo 1.

Con el estudio de algunos sistemas que permiten el control de asistencia existentes en el mundo y en Cuba se llega a la conclusión de que los mismos han demostrado el gran avance tecnológico que existe en la actualidad en cuanto a este tema, capaces de erradicar en gran medida los errores que se cometían cuando se hacía todo de forma manual, aunque con la limitante de que por muy ventajoso y moderno que sea un producto no siempre se ajusta a los requerimientos de una institución determinada. Por tanto se decidió elaborar un software multiplataforma, que brinde facilidades de uso a los usuarios, que reúna características que permitan su implantación en todas las entidades empresariales y unidades presupuestadas del país, que lleve un control del proceso de asistencia y que sea capaz de integrarse al componente de nómina, siendo este un producto capaz de cumplir con todos los requisitos de un determinado cliente.

# *Capítulo 2: Propuesta de solución.*

## Introducción.

En el siguiente capítulo se abordan los elementos inmersos en el desarrollo del subsistema. Se realiza una valoración de la propuesta del diseño del sistema exponiendo las principales ventajas y deficiencias del mismo. Se expone la explotación de la arquitectura y las posibilidades proporcionadas por el framework y las librerías utilizadas en la programación de la aplicación, con el objetivo de brindar mayor comprensión de las funcionalidades de los componentes implementados y por último se realizan las descripciones de clases y operaciones utilizadas.

## 2.1 Propuesta del sistema.

Para llevar a cabo la propuesta del sistema, es factible implementar una aplicación Web, basada en el uso del estilo arquitectónico Modelo – Vista – Controlador, usando el lenguaje de programación PHP y como soporte para los datos el Sistema Gestor de Bases de Datos PostgreSQL. Se propone utilizar además, para facilitar el trabajo, el Zend Framework, desarrollado en PHP bajo el estilo arquitectónico MVC y muy utilizado en la actualidad. como lenguaje de programación del lado del servidor PHP, para una mejor interacción entre el usuario y la aplicación se utilizaron las potencialidades que brinda ExtJS, como gestor de base de datos PostgreSQL aprovechando todas las potencialidades que este brinda, la programación por el lado del cliente XHTML, Java Script, CCS. Con auxilio del NetBeans para el desarrollo del módulo y el Visual Paradigm para el modelado de los diagramas de componente y despliegue. Dicha aplicación, automatiza los procesos que integran la gestión del Control de Asistencia para la administración del capital humano en el sector empresarial cubano y la documentación asociada a su desarrollo sirve de base para futuras implementaciones, mantenimientos o mejoras al software desarrollado con este mismo propósito.

## 2.2 Valoración y descripción de los requisitos.

Los requisitos funcionales descritos por los analistas del sistema facilitaron 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. Tomándose como base la

descripción detallada de los requisitos funcionales, se puede establecer una estrategia de trabajo que permita la implementación de la capa lógica, de presentación y de datos de la aplicación.

Los requisitos funcionales descritos por los analistas y que posibilitaron la identificación de las clases y las funcionalidades a implementar son los agrupados en la Tabla 1.

*Tabla 1: Requisitos funcionales*

| **Agrupación** | **Requisitos** |
|---|---|
| R1-Gestionar Control de Asistencia | R1.1-Ver control de asistencia |
| | R1.2-Justificar incidencia |
| | R1.3-Imprimir control de asistencia |
| | R1.4-Listar Asistencia |
| | R1.5-Exportar incidencias |
| | R1.6-Filtrar por Fecha |
| | R1.7-Filtrar por Área |
| R2-Gestionar Presencia | R2.1-Ver Presencia |
| | R2.2-Imprimir Presencia |
| | R2.3-Listar Presencia |
| | R2.4- Filtrar por Fecha |
| | R2.5- Filtrar por Área |
| R3-Registrar Marcaje | R3.1-Listar Marcajes |
| | R3.2-Registrar Trabajador |
| R4-Gestionar Dispositivo | R4.1-Adicionar Dispositivo |
| | R4.2-Eliminar Dispositivo |
| | R4.3-Modificar Dispositivo |
| | R4.4-Listar Dispositivo |
| R5-Gestionar Configuración | R5.1-Adicionar Nocturnidad-Franja Horaria |
| | R5.2-Adicionar Horas Extras |
| | R5.3-Adicionar Configuración de Dispositivo |
| | R5.4-Guardar Configuración |

Para hacer posible el cumplimiento de los requerimientos antes descritos de manera eficiente, así como para lograr una mejor aceptación por parte de los clientes, se identificaron algunas capacidades y características (requisitos no funcionales) que debe tener el sistema.

**Apariencia o interfaz externa:**

- El sistema debe tener una interfaz fácil de usar y amigable para que pueda ser utilizada sin mucho entrenamiento por el usuario.
- Empleo de imágenes identificadas con el negocio donde se implantará el sistema y colores agradables a la vista, siendo estos claros**.**

**Usabilidad:**

- El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora.

**Rendimiento:**

- Los tiempos de respuesta y velocidad de procesamiento de la información serán rápidos, no mayores de 5 segundos para las actualizaciones y 20 para las recuperaciones.

**Portabilidad:**

- El sistema debe ser multiplataforma, importante el correcto funcionamiento en Linux y Windows.

**Seguridad:**

- Autenticación (Contraseña de acceso).
- Autorización (Atribución a los usuarios respecto a sus funciones de trabajo).
- Implementación de auditoría (Registrar la confirmación de cada operación efectuada por el usuario que afecte los registros contables).
- La atención al sistema incluyendo, el mantenimiento de las bases de datos así como la salva de la información se realizarán de forma centralizada por el administrador.

**Legales:**

- ✓ El sistema está avalado por los tres documentos rectores emitidos en el país para la certificación y validación de los sistemas contables:
- ✓ La Resolución Conjunta de los ministerios de Finanzas y Precios de fecha 8.04.04.
- ✓ La Resolución 340 del Ministerio de Finanzas y Precios de fecha 8.12.04.
- ✓ La Resolución No. 12 del Ministerio de la Informática y las Comunicaciones de fecha 24.01.05.

**Software:**

- **Para el cliente:**
  - ✓ Navegador Mozilla Firefox.
  - ✓ Sistema operativo Linux, Windows 98 o superior.
- **Para el servidor:**
  - ✓ Sistema operativo Windows Advanced Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
  - ✓ Un servidor Apache 2.0 o superior con módulo PHP 5.2.6 disponible.
  - ✓ Un servidor de base de datos PostgreSQL 8.4 o superior.
  - ✓ Tarjeta de red.

**Restricciones para el diseño e implementación:**

- ✓ Emplear como servidores Web y de bases de datos Apache y PostgreSQL respectivamente.
- ✓ Utilizar como lenguaje del lado del servidor al PHP 5.2.6 y del lado del cliente el JavaScript.

## 2.3 Patrones de diseño empleados

Un patrón define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones.

Los patrones de diseño pretenden:

- ✓ Proporcionar catálogos de elementos reusables en el diseño de sistemas 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.

**Patrones Grasp (Patrones generales de software para asignar responsabilidades)**

Los patrones Grasp describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

**Patrones Gof (Gang of Four)**

Los patrones Gof son patrones de diseño publicados en el libro Design Patterns: Elements of Reusable Object-Oriented Software por Gamma, Helm, Jonson y Vlissides conocidos mundialmente por Gang of Four o Pandilla de los cuatro. En este libro se encuentran recopilados un total de 23 patrones clasificado en patrones creacionales, estructurales y de comportamiento.

**Los mencionados a continuación son los patrones utilizados en la presente investigación:**

### 2.3.1 GRASP

**Experto:** se puso en práctica en todos los componentes, con el uso de clases que poseen responsabilidades específicas a cumplir de acuerdo con la información que manejan, los componentes cuentan con clases controladoras, modelos y entidades que poseen funciones concretas de acuerdo con la información que gestionan. Además, se modeló una clase entidad por cada tabla de la base de datos, posibilitando el trabajo específico y directo con el experto en la información.

**Creador:** Es útil contar con el principio general para la asignación de responsabilidades de creación porque permite que el diseño pueda soportar bajo acoplamiento, mayor claridad, encapsulación y reutilización. En el módulo se evidencia este patrón en cada componente, las

clases controladoras son responsables de crear el objeto de las modelos, y estas a su vez de las entidades.

**Controlador**: Se utilizan cuando la aplicación es muy extensa, de esta forma, en el módulo en vez de tener un solo controlador y saturarlo, se tienen clases Controllers, que son controladores más pequeños especializados en las funcionalidades de cada componente.

**Bajo acoplamiento:** Los componentes en el módulo fueron diseñados bajo este principio, porque solo establecen las relaciones necesarias entre ellos. La base de datos fue definida de modo que entre las tablas existiera dependencia mínima, haciéndolas más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

**Alta cohesión:** En el módulo existe afinidad entre cada clase y los métodos que implementan, estas poseen responsabilidades vinculadas acordes a la información que controlan; y colaboran con otros objetos para compartir el esfuerzo si la tarea es grande, facilitando su mantenimiento y reutilización.

### 2.3.2 GOF

**Fachada:** Su aplicación posibilita promover un débil acoplamiento entre los diferentes componentes que conforman el módulo y su integración con otros subsistemas. Se usa una única clase Service por componente que proporciona los servicios necesarios publicados en el IOC donde se implementan todas las funcionalidades que el componente es capaz de brindar.

**Cadena de responsabilidad**: La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando más de un objeto la posibilidad de manejar la petición. Este es utilizado en la mayoría de los diagramas de clases siendo aplicado en el tratamiento de excepciones. Un ejemplo de su uso es cuando se produce un error al insertar en la base de datos, el cual es captado por las capas superiores, reenviando la excepción hasta la capa de aplicación donde traduce al lenguaje del usuario.

## 2.5 Diagrama de Componentes

En esta investigación se adopta como definición de componente la de Clemens Szyperski: Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que puede ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (Szyperski 1998).

En la Figura 3 se muestra el modelo de componentes correspondiente al módulo de Control de Asistencia. Cada paquete en los diagramas representa una división lógica del sistema.

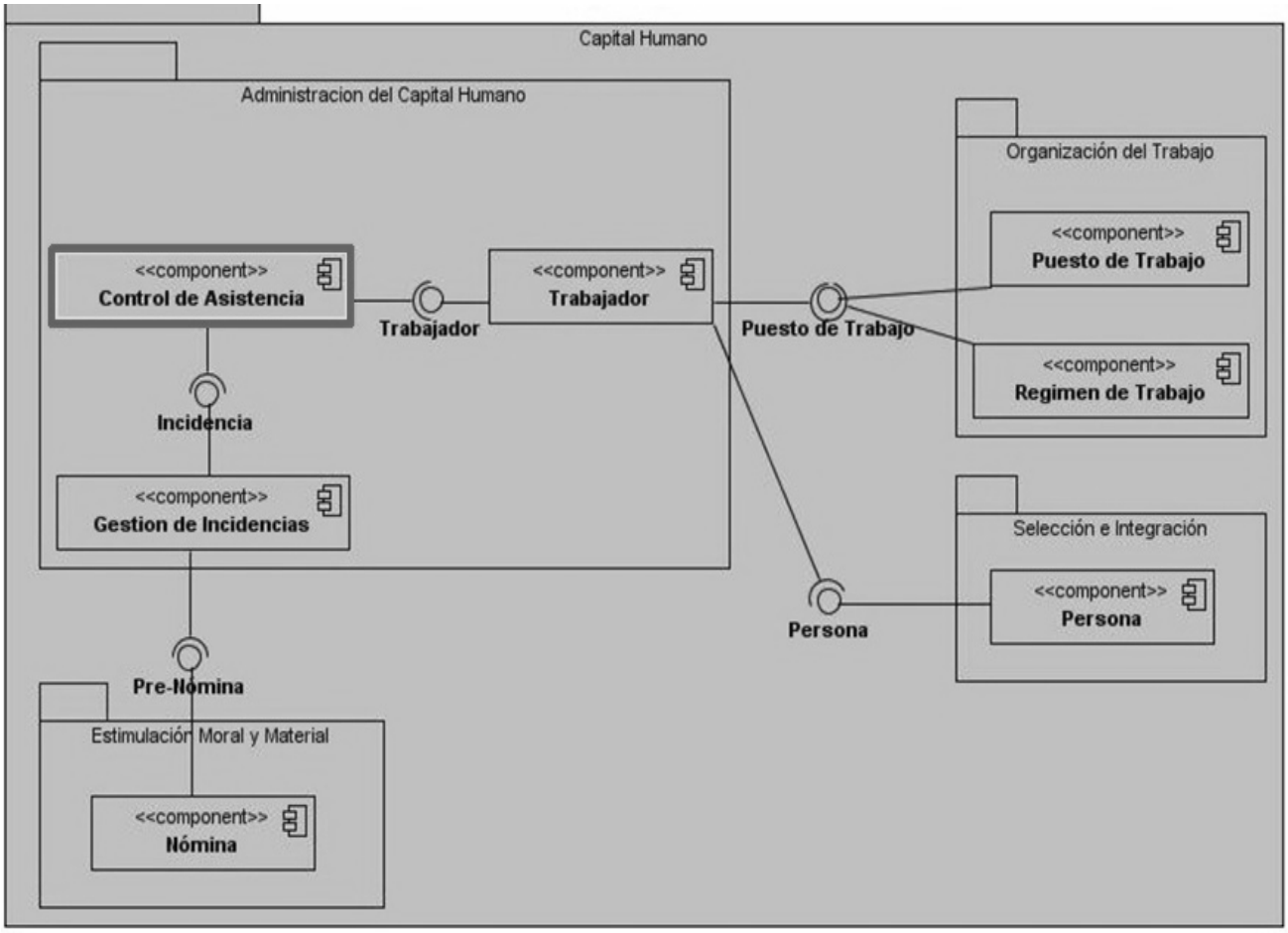


*Figura 2: Modelo de componentes*

### 2.6 Interacción entre componentes

En la Figura 4 se muestra un fragmento del diagrama de interacción de componentes. Viéndose la secuencia a seguir por cada clase para lograr relacionarse con las demás.

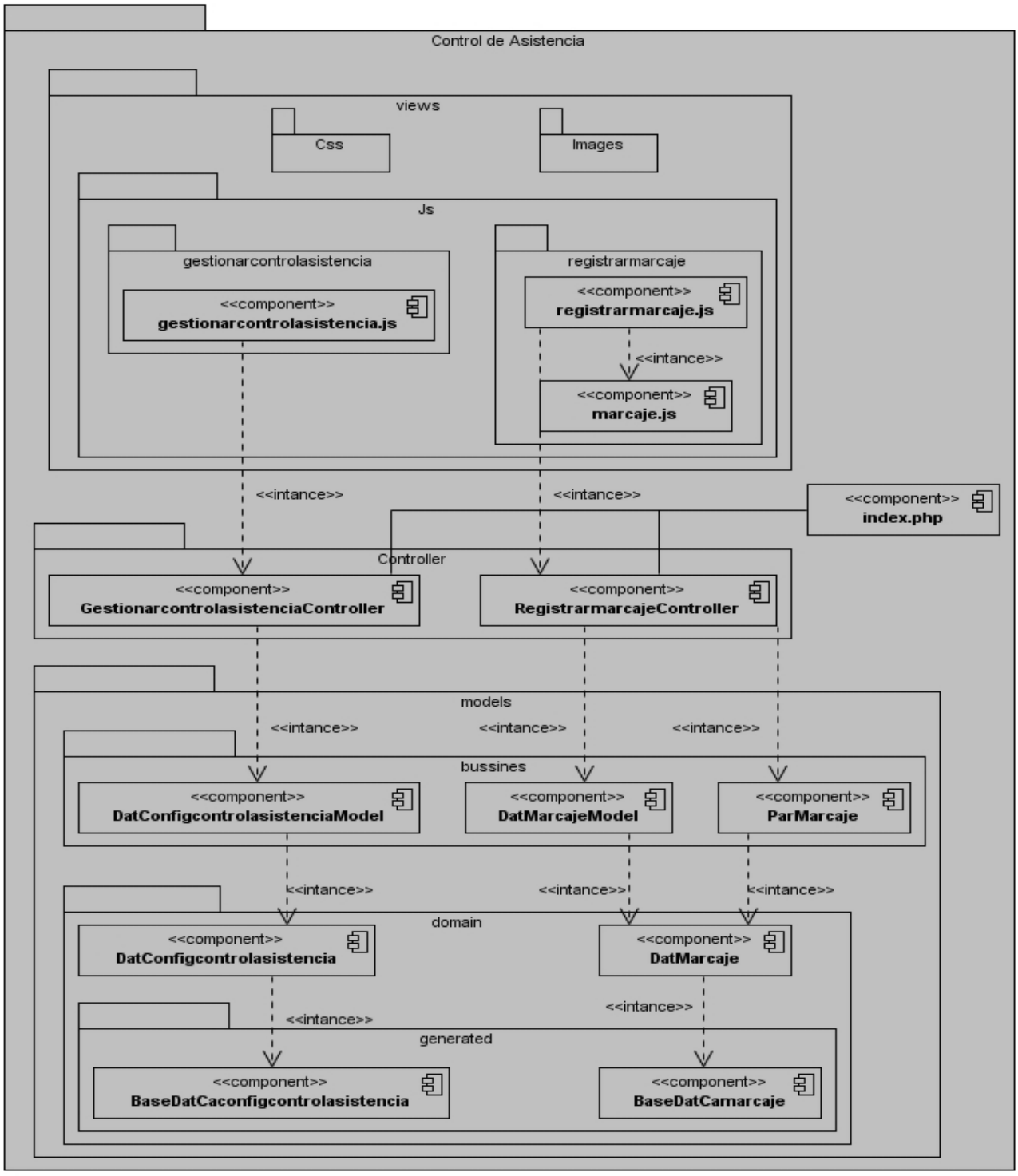


*Figura 3: Fragmento del Diagrama de Interacción de componentes.*

La aplicación está definida por capas, en la capa de funcionamiento se utiliza el patrón modelo – vista-controlador (MVC) que consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatro nodos de integración, el que se encuentra entre la vista y el controlador, el que está entre el controlador y el modelo, el que vincula el modelo con el ORM (object relational mapper) doctrine y el que se encuentra entre el doctrine y la base de datos. Todo el código dentro de un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes módulos y componentes se realiza mediante llamadas a la inversión de control. Cada componente tiene su registro de los datos de los módulos en un fichero XML que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IOC (inversión de control) y registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. El IOC especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. La base de datos es accedida de forma directa mediante las clases entidades y los componentes rehusados son integrados mediante interfaces sencillas, garantizando así una total integración de las capas en el sistema. Para acceder a funcionalidades o servicios brindados por otros subsistemas, se hace a través de una variable global definida en el framework llamada ‘’Integrate’’, el nombre del subsistema, el componente y el nombre del método que accede al IOC el cual se encuentra en una carpeta que está en el común del sistema, a la cual tienen acceso todos los módulos del sistema.

La arquitectura en 3 capas: presentación (view), negocio (controller) y acceso a datos (models), consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, a través de los diferentes elementos que la componen. Consta de 4 nodos de integración: vista-controlador, controlador-modelo, modelo-framework Doctrine y Doctrine-base de datos.

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.

### 2.7 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 framework ZendExt_Controller_Secure.

  Ejemplo: GestionarcontrolasistenciaController

- ✓ **Clases de los 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 súper clase del framework Zend_Ext llamada ZendExt_Model.

    Ejemplo: NomDispositivoModel, DatConfigcontrolasistenciaModel.

  - ***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: NomDispositivo, DatConfigcontrolasistencia.

- ***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: BaseNomDispositivo, BaseDatCamarcaje.

✓ **Validators: Clases validadoras.**

El nombre de la clase validadoras se especifica con nombres compuestos con el distintivo que terminan en Validator. Ejemplo: DatoscontablesValidator.

✓ **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: ControlAsistenciaService.

### 2.7.1 Algunos servicios que ofrece el componente Control de Asistencia:

**ObtenerHoras($idsesion)→** Este servicio devuelve las horas de una sesión de trabajo de acuerdo al parámetro idsesion.

**ObtenerIdJornadaDenom($denom) →** Este servicio devuelve el id de la jornada de acuerdo al parámetro denominación (denom).

**ObtenerRegimenDandoDenom($denomregimen) →** Este servicio devuelve el régimen de trabajo de acuerdo a un denominación del régimen que se pasa como parámetro.

### 2.7.2 Algunos servicios que utiliza el componente Control de Asistencia:

**BuscarTrabajadorPorIds($ids) →** Este servicio devuelve un arreglo con todos los trabajadores que tengan el id igual a alguno de la lista de Ids que se le pasa por parámetro.

**BuscarPersonaPorParam($params) →**Este servicio devuelve un arreglo con todos los datos de una persona de acuerdo al parámetro pasado $params correspondiente a esta persona.

**ObtenerPuestoTrabajo($idpuesto)** → Este servicio devuelve un arreglo con todos los datos de un puesto de trabajo de acuerdo al parámetro pasado $idpuesto correspondiente a este puesto de trabajo.

**CargarMarcajes($fecha)** → Este servicio devuelve un arreglo con todos los marcajes realizados en un día determinado de acuerdo al parámetro pasado $fecha que corresponde a este día específico.

### 2.7.3 Descripción de clases por componente y tipo:

A continuación se describen las clases y sus operaciones más importantes por componentes agrupándolas de acuerdo con la clasificación siguiente:

**Clases Controladoras** Las clases controladoras son responsables de la lógica de negocio, gestionando todo el flujo de datos y operaciones entre la vista y demás clases del negocio.

**Clases Modelo** Actúan como intermediarias entre las clases controladoras y las clases dominio. Complementan las funcionalidades de las clases controladoras y realizan las funciones de insertar, modificar y eliminar datos.

**Clases Dominio** Las clases dominio modelan los objetos del sistema y comportamiento asociado; frecuentemente representan conceptos y datos persistentes de larga duración. Contienen métodos para la gestión consultas y solicitud de información de la base de datos**.**

**Componente Control de Asistencia**

- **Clases Controladoras**

*Tabla 2: Descripción de la Clase GestionarconfiguracionController.*

| **Nombre:** GestionarconfiguracionController | |
|---|---|
| **Tipo de clase:** Controladora | |
| **Para cada responsabilidad** | |
| **Nombre.** | **Descripción.** |
| init() | Constructor de la clase. |
| gestionarconfiguracionAction() | Responsabilidad encargada de validar todos los datos necesarios y redireccionar a la |

| | |
|---|---|
| | interfaz. |
| GuardarConfigAction() | Responsabilidad encargada guardar todos los datos insertados. |
| CargarComboAusenciaAction() | Funcionalidad que se encarga de cargar los datos para llenar el combo box que hace referencia a las Ausencias. |
| CargarHorasExtrasAction() | Funcionalidad que se encarga de cargar los datos para llenar el combo box que hace referencia a las Horas Extras. |
| CargarConfiguracionAction() | Responsabilidad encargada de cargar el formulario. |

- **Clases Modelos**

*Tabla 3: Descripción de la clase DatCaconfigtipodispositivoModel.*

| **Nombre**: DatCaconfigtipodispositivoModel | |
|---|---|
| **Tipo de clase:** modelo | |
| **Para cada responsabilidad** | |
| **Nombre.** | **Descripción.** |
| setUp() | Constructor de la clase. |
| Insertar(DatCaconfigtipodispositivo $DatCaconfigtipodispositivo) | Responsabilidad encargada de insertar un dispositivo. |
| Actualizar(DatCaconfigtipodispositivo $DatCaconfigtipodispositivo) | Responsabilidad encargada de actualizar dispositivo. |
| Eliminar($DatCaconfigtipodispositivo) | Responsabilidad encargada de eliminar un dispositivo. |

## 2.8 Interfaces.

### 2.8.1 Interfaz Registrar Marcaje.

La interfaz Registrar Marcaje se muestra en la Figura 6 y en la Figura 27 (Ver Anexo 1) se puede ver como se registra un trabajador en dicha interfaz.

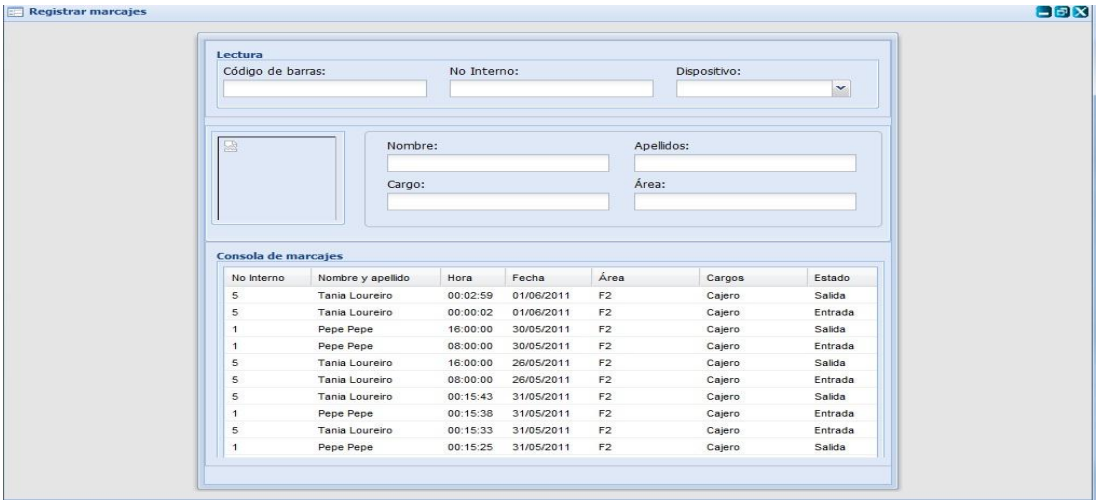


*Figura 4: Interfaz Registrar Marcajes.*

### 2.8.2 Interfaz Gestionar Control de Asistencia.

En esta interfaz se muestra los registros de entrada y salida de los trabajadores en el día, como se muestra en la Figura 7. Se puede filtrar la búsqueda de estos marcajes por área o por fecha. Dicha interfaz tiene funcionalidades como son los botones: Ver, Exportar Inc., Justificar e Imprimir. En las Figura 28 y 29, se muestran las funcionalidades Ver y Exportar Incidencias respectivamente. (Ver Anexo 2 y 3)

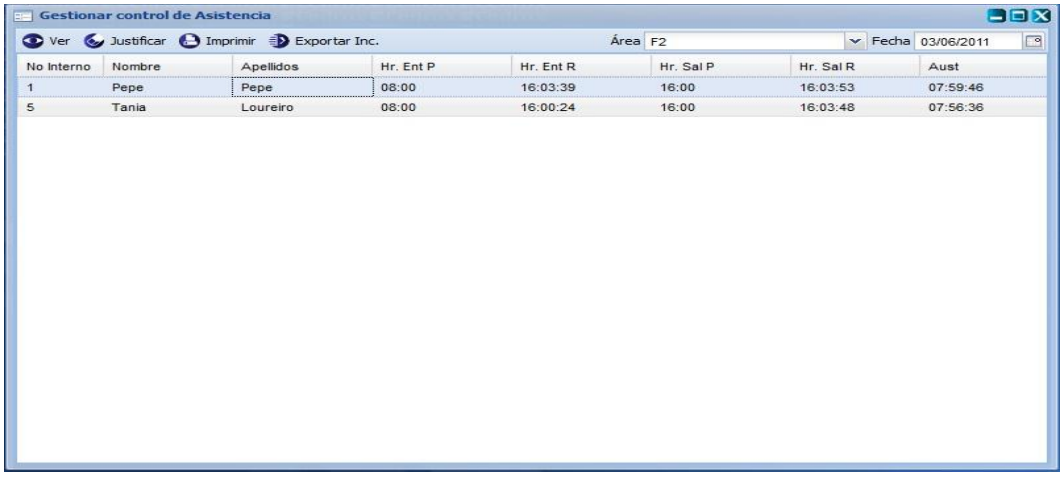

*Figura 5: Interfaz Gestionar Control de Asistencia.*

### 2.8.3 Interfaz Gestionar Configuración de Control de Asistencia.

En esta interfaz se establece la configuración por la cual se regirá todo el sistema de control de asistencia, en la Figura 8 se muestra.

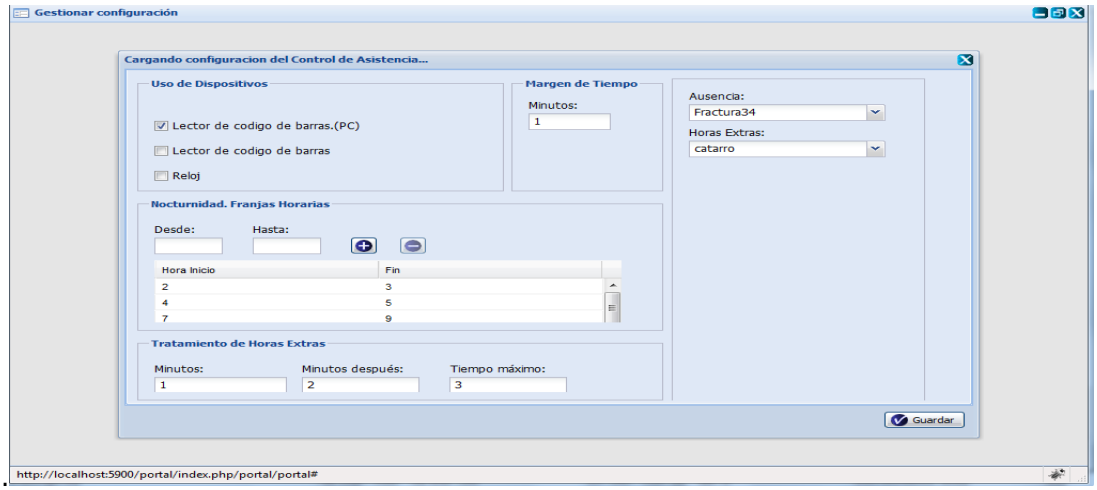


*Figura 6: Interfaz Gestionar Configuración de Control de Asistencia.*

### 2.8.4 Interfaz Gestionar Dispositivo.

Esta interfaz se muestra en la Figura 9 y se encarga de gestionar los dispositivos. Presenta las funcionalidades de Adicionar, Modificar y Eliminar; mostrándose respectivamente en las Figuras 30, 31 y 32. (Ver Anexos 4, 5 y 6)

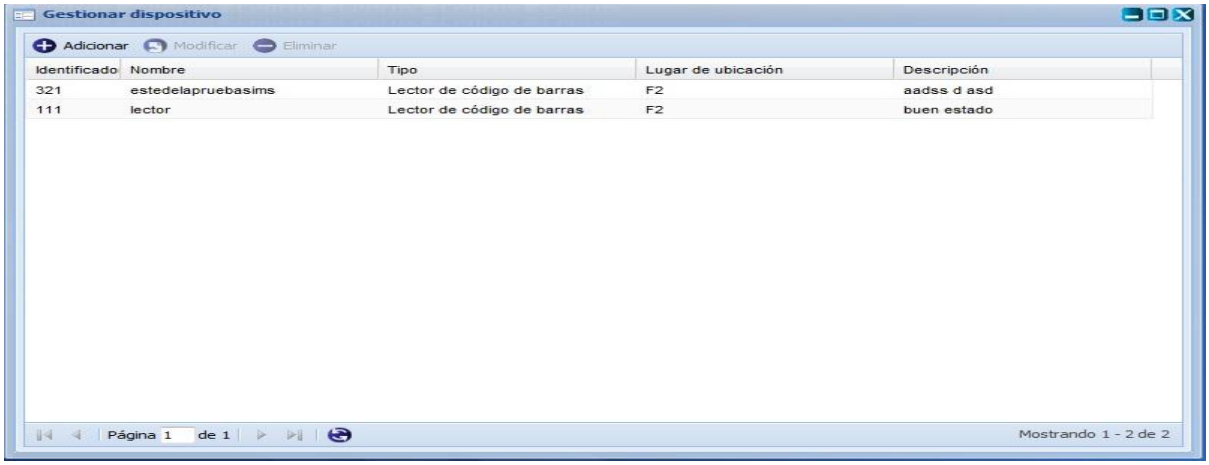

*Figura 7: Interfaz Gestionar Dispositivo.*

### 2.8.5 Interfaz Gestionar Presencia.

En esta interfaz que se muestra en la Figura 10, se gestiona la presencia de los trabajadores que han realizado marcaje en la institución. Se puede filtrar la búsqueda de los mismos por área o por fecha, mostrándose en la Figura 33 (Ver Anexo 7). Dicha interfaz tiene funcionalidades como son los botones: Ver e Imprimir. En la Figura 34, se muestra la funcionalidad Ver (Ver Anexo 8).

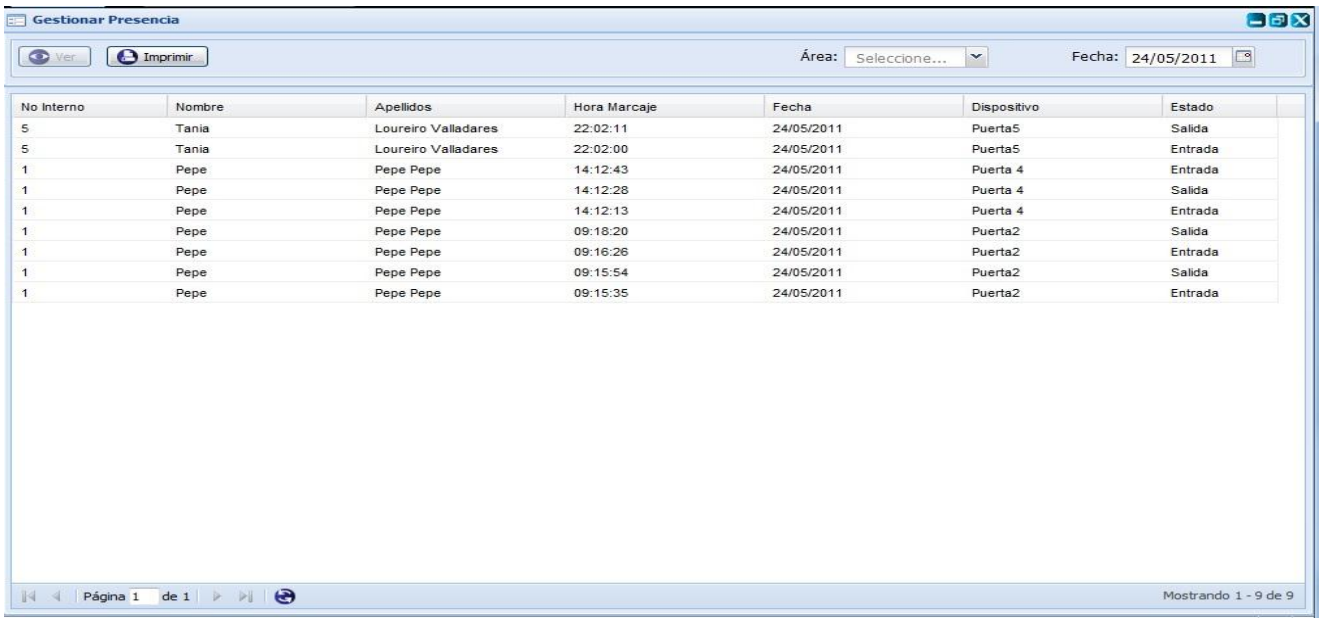


*Figura 8: Interfaz Gestionar Presencia.*

## 2.9 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. El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. A continuación se explican los recursos presentes en cada nodo del diagrama, mostrado en la Figura 19:

- ✓ **Nodo PC Cliente**: a través del cual los clientes se conectarán al sistema y podrán operar en el mismo. Esta PC deberá contar como navegador con Mozilla Firefox 3.0 o superior que le permitirán al cliente tener acceso a la aplicación.

- ✓ **Nodo Servidor Web:** puede tener instalado como Sistema Operativo Windows XP o superior, GNU/Linux o Mac OS. Además debe tener instalado el Apache Server 2.2 con el módulo ModPython o WSGI instalados.

- ✓ **Nodo Servidor Base de Datos:** este servidor deberá presentar como Sistema Gestor de Base de Datos PostgreSQL 8.3 o superior.

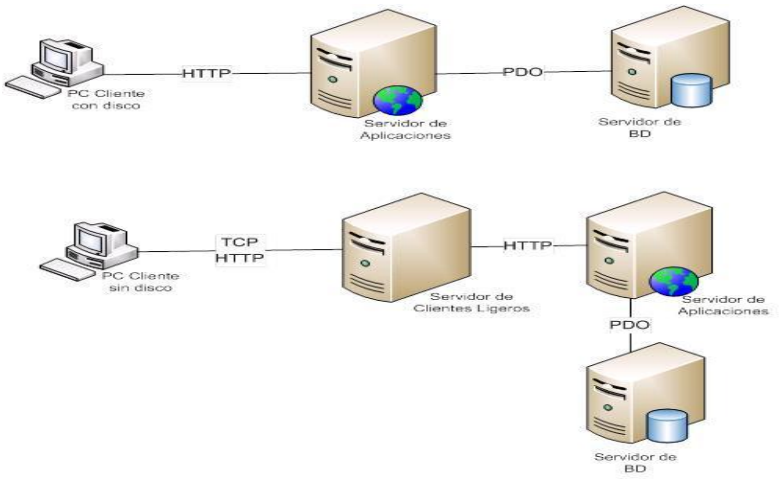


*Figura 9: Diagrama de Despliegue.*

## Conclusiones del capítulo.

En este capítulo fueron expuestos los artefactos generados durante la implementación de la solución propuesta, lo que proporcionó una temprana idea de la complejidad de la solución. Se mencionan los componentes que fueron usados para el desarrollo de la aplicación, teniendo en cuenta la arquitectura del sistema, para un mejor entendimiento de la solución propuesta al mostrarse la integración entre los mismos. La descripción de las principales clases permitió tener una visión de la implementación realizada. De forma general se logró implementar los requerimientos definidos por los analistas para los procesos de Control de Asistencia. Una vez concluido el diseño de la solución puede darse paso a los flujos de prueba de la misma.

# *Capítulo 3: Validación de la solución propuesta.*

## Introducción.

El desarrollo del software implica una serie de actividades de producción en las que las posibilidades de que aparezcan fallos humanos son muy grandes. Los errores pueden presentarse debido a especificaciones erróneas e imperfectas de los requisitos, uso indebido de las estructuras de datos, errores al integrar módulos, entre otras causas. Dado a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software ha de ir acompañado de una actividad que garantice la calidad.

Para dar solución a todos estos problemas surge un nuevo proceso dentro del ciclo de desarrollo del software: el proceso de pruebas. Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. Las pruebas y validación de los resultados no se realizan una vez acabado el software, sino que deben ejecutarse en cada una de las etapas de desarrollo. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar., de forma que se pueda comprobar la eficiencia de las operaciones utilizadas para satisfacer las necesidades del cliente.

En este capítulo se abordarán temas como las pruebas realizadas al software, en específico las pruebas de caja blanca y caja negra, además se hace una valoración de las mismas según los resultados obtenidos.

## 3.1 Validación del modelo de diseño propuesto.

El desarrollo de software es algo muy complejo y la medida de su calidad real no es automatizable. Por esta razón la aplicación de métricas de calidad para evaluar el diseño orientado a objeto posee gran importancia. A continuación se presenta un estudio que brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software. Siendo esto la principal razón de la concepción de las métricas inspiradas en lo propuesto por Pressman (PRESSMAN, 1998).

Atributos de calidad que se abarcan:

1. **Responsabilidad**. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

2. **Complejidad del diseño.** Consiste en la complejidad que posee una estructura de diseño de clases.

3. **Complejidad de implementación**. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

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

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

6. **Complejidad del mantenimiento**. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

7. **Cantidad de pruebas**. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.

8. **Nivel de Cohesión**. Consiste en el grado de especialización de las clases concebidas para modelar un dominio o concepto específico.

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

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

**Resultados del instrumento de evaluación de la métrica Tamaño operacional de clase (TOC)** (Ver instrumentos y tabla de resultados en Anexo 9).

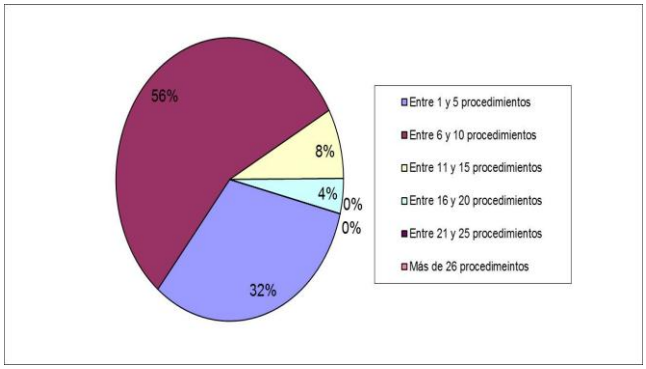


*Figura 10: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.*

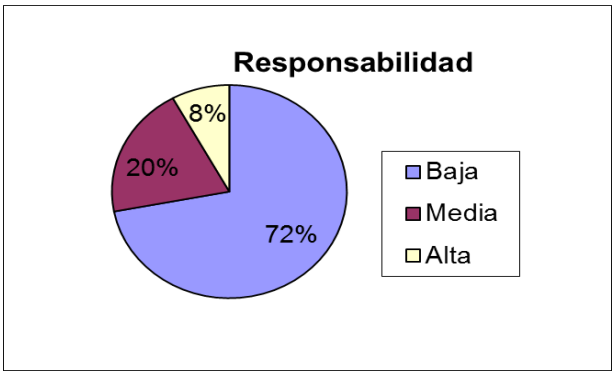


*Figura 11: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.*

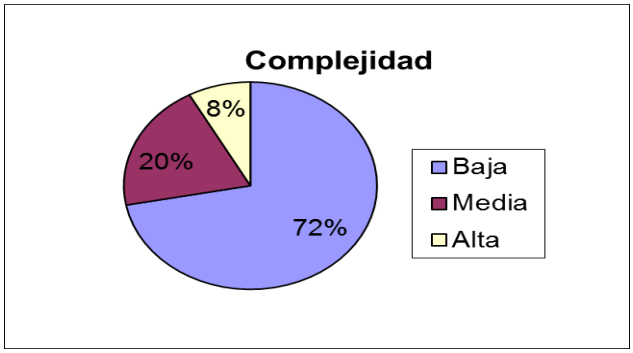


*Figura 12: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.*

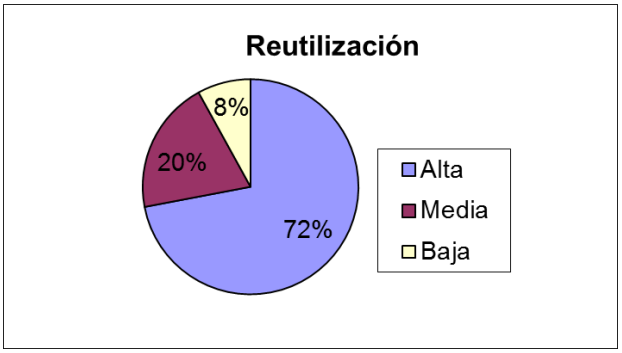


*Figura 13: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.*

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del componente Control de Asistencia tienen una calidad buena pudiéndose observar que el 88% de las clases posee menos cantidad de operaciones que la media registrada en las mediciones. Además el 72% de las clases posee evaluaciones positivas por encima de la media en los atributos (Responsabilidad, Complejidad de Implementación y Reutilización).

**Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC).**

(Ver instrumentos y tabla de resultados en Anexo 10).

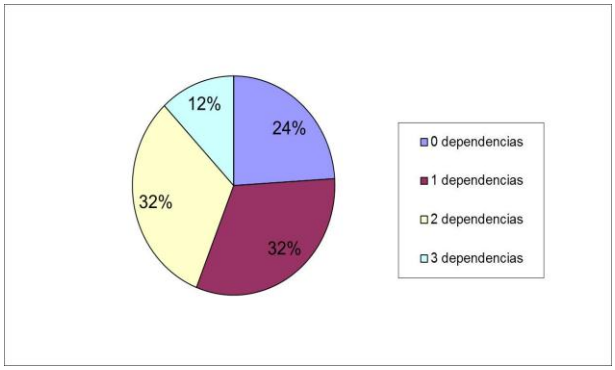


*Figura 14: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.*

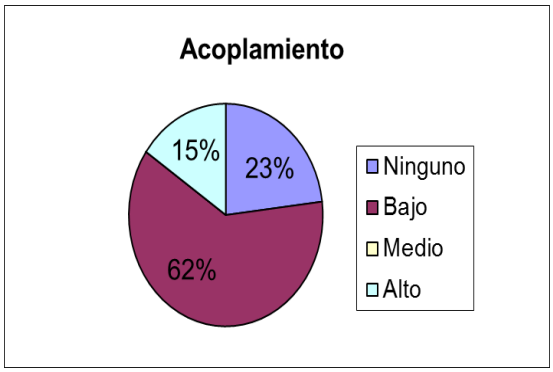


*Figura 15: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.*

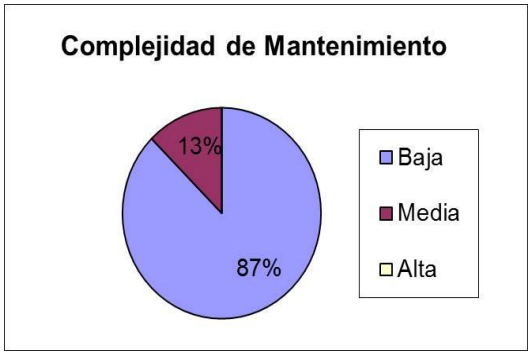


*Figura 16: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.*

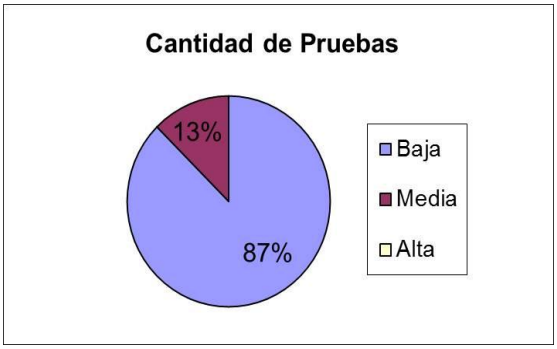


*Figura 17: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.*

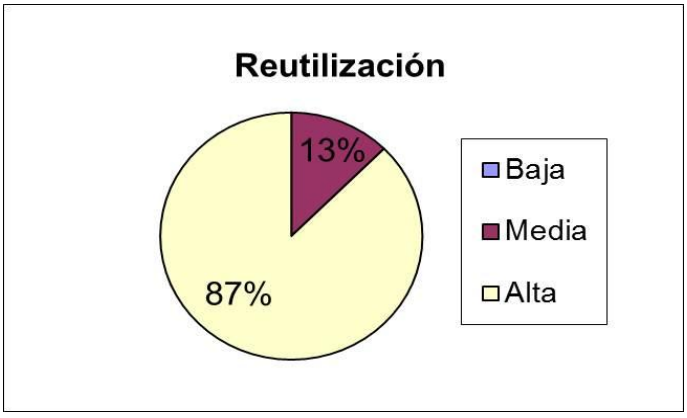


*Figura 18: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.*

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del componente Control de Asistencia tienen una calidad buena pudiéndose observar que el 88% de las clases posee menos de 3 dependencias de otras clases. Además el 23% de las clases no poseen acoplamiento con otras y el 77% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 87% de las clases.

### 3.2 Pruebas de software.

Las pruebas de software son el conjunto de técnicas que permiten determinar la calidad de un producto. Estas se integran dentro de las diferentes fases del Ciclo del Vida del software dentro de la Ingeniería. Así se ejecuta un programa y mediante técnicas experimentales se trata de descubrir que errores presenta. La calidad de un sistema informático es algo subjetivo que depende del contexto y del objeto que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar medidas o pruebas que permitan comprobar el grado de cumplimiento con respecto a las especificaciones iniciales del sistema.

#### 3.2.1 Objetivos de las pruebas.

El objetivo de las pruebas de un programa es el de detectar todo posible mal funcionamiento, un error puede ser costoso de reparar mientras más se avanza en las etapas del ciclo de vida del software. Dentro de los objetivos fundamentales que se persiguen al aplicarles pruebas a un software se encuentran los siguientes:

- ✓ Brindar un mayor nivel de confiabilidad en los productos que se van generando.
- ✓ Detectar fallas o errores.
- ✓ Aumentar la calidad del producto final.

Si las pruebas que se llevan a cabo tienen éxito se descubrirán errores en el software**,** dándole mayor fiabilidad. Es importante tener en cuenta una frase de Pressman:

*"La prueba no puede asegurar la ausencia de defectos; solo puede demostrar que existen defectos en el software"* (Pressman, 2002).

### 3.3 Modelos de pruebas.

#### 3.3.2 Tipos de pruebas.

Existen 2 enfoques principales de prueba:

- ✓ El enfoque estructural o de caja blanca: que se basa en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.
- ✓ El enfoque funcional o de caja negra: que realiza pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

#### 3.3.3 Pruebas de Caja Blanca o Estructurales.

A este tipo de técnicas se le conoce también como Técnicas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa examinando su lógica interna sin considerar los aspectos de rendimiento. El objetivo de la técnica es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa.

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

- ✓ Prueba de Condición.
- ✓ Prueba de Flujo de Datos.
- ✓ Prueba de Bucles.
- ✓ Prueba del Camino Básico.

Dentro de la prueba de caja blanca, la técnica que se utilizó fue la de prueba del camino básico. Esta prueba permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, además también garantiza que durante la prueba en los casos de prueba

obtenidos a través del camino básico se ejecute cada sentencia del programa por lo menos una vez.

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

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

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

Componentes del grafo de flujo:

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

Para realizar la técnica de prueba de caja blanca, específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumera las sentencias de código del procedimiento realizado sobre el método registrarAction () el cual se encarga de registrar un marcaje.

```
66  // funcion registrar
67  function registrarAction(){
68    $objDaM = new DatMarcajeModel();1
69    $params->numeroexpediente= $this->_request->getPost('numInterno');2
70    $trab = $this->pIntegrator->trabajador->BuscarTrabajador($params);3
71    if($trab){4
72    $obj->idtrabajador= $trab[0]->idtrabajador;5
73    $obj->fechamarcaje= date("d-m-Y");6
74    $obj->horamarcaje= date("H:i:s");7
75    $obj->iddispositivo= $this->_request->getPost('iddispositivo');8
76    $marcajes = $objDaM->buscarMarcaje($obj);9
77    if(!$marcajes){ 10
78      $obj->entrada = 1; 11
79    } 12
80    else{ 13
81        if($marcajes[0][entrada] == 1)14
82            $obj->entrada = 2;15
83        else 16
84            $obj->entrada = 1; 17
85    }18
86    echo $objDaM->registrarMarcaje($obj); 19
87    } 20
88
89  }//fin de la función registrar 21
```

*Figura 19: Sentencias 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:

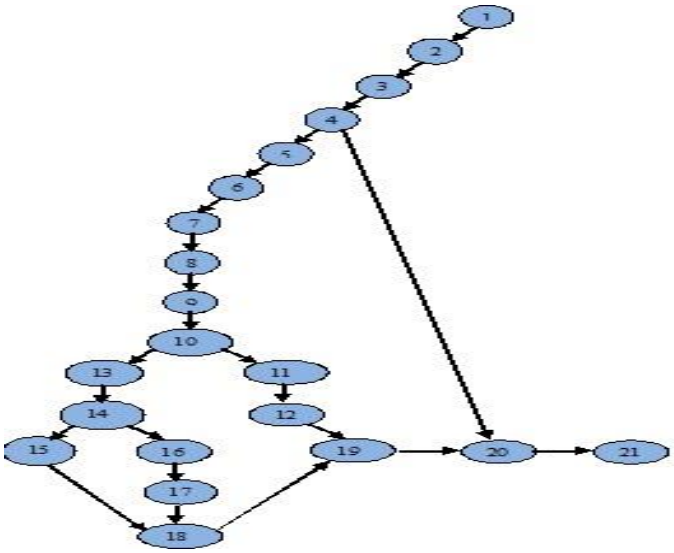


*Figura 20: Grafo de flujo asociado al código.*

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:

V (G) = (A – N) + 2

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

V (G) = (23 -21) + 2 V (G) = 4. 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. 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.

Camino básico # 1: 1-2-3-4-5-6-7-8-9-10-11-12-19-20-21

Camino básico # 2: 1-2-3-4-5-6-7-8-9-10-13-14-15-18-19-20-21

Camino básico # 3: 1-2-3-4-5-6-7-8-9-10-13-14-16-17-18-19-20-21

Camino básico # 4: 1-2-3-4-20-21

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**.

Descripción: A partir de un determinado identificador (nointerno) se busca al trabajador que su nointerno corresponda con este identificador, y de este trabajador se capturan los datos necesarios para realizar un marcaje, especificando siempre si el marcaje es de entrada o salida.

Condición de ejecución: Debe existir al menos un trabajador registrado en el sistema, y que el número interno (nointerno) sea igual al número interno entrado.

Entrada: $params->numeroexpediente = $this->_request->getPost ('nointerno').

Resultados esperados: Se espera que el marcaje sea registrado satisfactoriamente.

Se pudo registrar el marcaje satisfactoriamente, siendo este marcaje el primero de este trabajador.

- ✓ **Caso de prueba para el camino básico # 2.**

Descripción: La misma descripción que el caso de prueba para el camino básico # 1.

Condición de ejecución: Debe existir al menos un trabajador registrado en el sistema, y que el número interno (nointerno) sea igual al número interno entrado.

Entrada: $params->numeroexpediente = $this->_request->getPost ('nointerno').

Resultados esperados: Se espera que el marcaje sea registrado satisfactoriamente.

Se pudo registrar satisfactoriamente el marcaje.

- ✓ **Caso de prueba para el camino básico # 3.**

Descripción: La misma descripción que el caso de prueba para el camino básico # 1.

Condición de ejecución: Debe existir al menos un trabajador registrado en el sistema, y que el número interno (nointerno) sea igual al número interno entrado.

Entrada: $params->numeroexpediente = $this->_request->getPost ('nointerno').

Resultados esperados: Se espera que el marcaje sea registrado satisfactoriamente.

Se pudo registrar satisfactoriamente el marcaje.

- ✓ **Caso de prueba para el camino básico # 4.**

Descripción: La misma descripción que el caso de prueba para el camino básico # 1.

Condición de ejecución: Debe existir al menos un trabajador registrado en el sistema, y que el número interno (nointerno) sea igual al número interno entrado.

Entrada: $params->numeroexpediente = $this->_request->getPost ('nointerno').

Resultados esperados: Se espera que el marcaje sea registrado satisfactoriamente.

No se pudo registrar satisfactoriamente el marcaje. Producto a que no existe ningún trabajador en el sistema o que la entrada del nointerno vino sin valor.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función está correcto ya que cumple con las condiciones necesarias que se habían planteado.

### 3.3.4 Aplicación de pruebas de caja negra.

Los casos de prueba para caja negra o también conocidas como Prueba de Caja Opaca, se basan en las diferentes entradas que puede recibir el software y sus correspondientes valores de salida, están centrados en realizar pruebas del software a través de la funcionalidad.

Los casos de prueba demuestran que:

- ✓ Las funciones del software son operativas.

- ✓ Las entradas se aceptan de la forma adecuada produciendo el resultado correcto.
- ✓ La integridad de la información externa (por ejemplo archivos de datos) se mantiene.

Permite encontrar:

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

Algunas técnicas utilizadas en las pruebas de caja negra son:

- ✓ **Partición de equivalencia:** Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.
- ✓ **Análisis de valores límite:** La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica lleva a elegir los casos de prueba que ejerciten los valores límite.
- ✓ **Grafos de causa-efecto:** Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Se utilizará en particular la técnica de partición de equivalencia en la prueba de caja negra que se aplicará. Para la aplicación de este tipo de pruebas se diseñaron un conjunto de escenarios de pruebas con el objetivo de evaluar el cumplimiento de los requisitos del sistema. A continuación se muestran los escenarios empleados.

**Aplicación.**

Para verificar que la aplicación se comporta según los requerimientos establecidos por el cliente, se diseñan casos de pruebas usando el método de caja negra. A continuación se especifica el caso de prueba para el requisito “Adicionar dispositivo”, el cual se ejecuta accionando el botón “Adicionar”, se abrirá una venta en la cual se introducirán los valores pertenecientes al dispositivo a insertar. Los valores serian: el nombre del dispositivo, selección del tipo de dispositivo, código, selección del área en la que se ubicará el dispositivo, y una descripción. (Ver Anexo 11)

**Conclusiones del capítulo.**

Una de las últimas fases del ciclo de vida antes de entregar un programa para su explotación, es la fase de pruebas. Esta fase del desarrollo de un software es la que mayor cantidad de tiempo y de esfuerzo requiere, se estima que la mitad del esfuerzo de desarrollo de un programa tanto en tiempo como en gastos se invierte en esta. Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre, siendo este el valor que le añade. Mientras más errores se encuentren al software en esta fase mejor.

En el capítulo 3 se efectuaron las pruebas de software necesarios para lograr la calidad requerida de la aplicación propuesta. 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 lograron aplicar las métricas: Relaciones entre Clases (RC) y Tamaño Operacional de la Clase (TOC) con el propósito de validar el diseño, las cuales devolvieron valores satisfactorios para cada uno de los indicadores correspondientes. El componente Control de Asistencia después de realizadas estas pruebas, desde el punto de vista funcional cumple con los requerimientos capturados y especificados en las primeras etapas de desarrollo.

# *Conclusiones*

Concluido el presente trabajo de diploma se dan por cumplidos los objetivos planteados en sus inicios pues se ha logrado una realización eficiente de los procesos involucrados, obteniéndose la implementación de un componente en el que se aplican los resultados de la investigación llevada a cabo. Se analizaron sistemas informáticos tanto nacionales como internacionales vinculados al proceso de Control de Asistencia; capturando de estos las ventajas y desventajas, y reutilizando las ideas necesarias de estos para la realización del componente propuesto. Se logró realizar la valoración del diseño propuesto obteniendo como resultado la implementación del componente Control de Asistencia correspondiente al Subsistema Capital Humano del Sistema Integral de Gestión de Entidades CEDRUX. Se logró validar la calidad del componente a través de pruebas de software efectuadas, las cuales devolvieron los favorables resultados esperados, permitiendo cumplir las funcionalidades previstas.

Este componente propuesto, tiene una gran importancia ya que trae consigo la posibilidad de integrarse a un componente Nómina, permitiendo la generación de reportes, y eliminando de una vez y por todas, la realización manual de estos procesos, proporcionando mayor rapidez, confiabilidad y seguridad. Presenta una interfaz amigable, sencilla y de fácil comprensión para los usuarios. Es una aplicación multiplataforma, capaz de adaptarse a diversos sistemas y marcos de trabajo.

# *Recomendaciones*

Se recomienda continuar con las siguientes acciones a favor del componente:

- ✓ Optimizar los algoritmos de los métodos que manejen grandes volúmenes de datos.
- ✓ Continuar realizando pruebas de calidad al componente.
- ✓ Profundizar en temas referentes al Control de Asistencia para detectar posibles debilidades en el componente y agregar mejoras al mismo.
- ✓ Realizar el despliegue del componente propuesto como parte del subsistema de Capital Humano del sistema Cedrux.

# Bibliografía

(FUNDACITE), F. p. e. D. d. l. C. y. l. T. d. e. M. (2007). "Sistema de Control de Asistencia." from http://sistemas.fsl.fundacite-merida.gob.ve/softwaremap/trove_list.php?form_cat=237&discrim=237.

A&M, S. (2010). "SecureTech A&M. SmartClk." from http://www.securetech.com.uy/documentos/securetech/SmartClk%20V316.pdf.

Aquino, A. y. L., Yasser (2009). "Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux."

ArchivosPC. (2006). "ArchivosPC. Exactus Pro." from http://exactus-pro-2006.archivospc.com/.

Assets. (2004). "Assets. Módulo de Recursos Humanos." from http://assets.co.cu/humanos.asp.

Baryolo, O. G. (2010). ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. La habana, Universidad de las Ciencias Informáticas.

CSS, V. d. (2006). Versiones de CSS. Aprendiendo CSS. **2011**.

Design-soft. (2005). "Design-soft. Sicon." from http://www.design-soft.com.pe/Sicon.htm.

Desoft, C. d. (2008). "Desoft. Fastos." from http://www.desoft.cu/Productos1/FastosPagus/tabid/442/Default.aspx?PageContentID=23.

Elianys Hurtado Sola, M. T. C. y. Y. M. B. (2008). Ayuda al usuario del caso de estudio en el nuevo marco de trabajo del ERP. La Habana, Universidad de las Ciencias Informáticas. **1.0**.

Esser, S. (2009.). "Secure Programming with the Zend-Framework."

Evolved, W. D. (2010). "Firebug. Web Development Evolved." from http://getfirebug.com.

Frederick, S., Ramsay, Colin y Blades, Steve 'Cutter'. (2008.). "Learning Ext JS."

Group, T. P. G. D. (2010). "PostgreSQL 8.4.4 Documentation." from http://www.postgresql.org/docs/8.4/static/intro-whatis.html.

Infosertec, W. (2009). "Sun Microsystems presenta NetBeans IDE 6.8." Retrieved 17 de diciembre, 2009, from http://www.infosertec.com.

JSON. (2010). "JSON." from http://www.json.org/json-es.html.

Linux., Á. t. d. (2009).

Mendoza Sanchez, M. A. (2004). Metodologias De Desarrollo De Software.

Molpereces, A. (2003). "Procesos de desarrollo: RUP, XP, FDD." from http://www.willydev.net/descargas/Articulos/General/cualxpfddrup.PDF.

Moreno, N. C. (2010). Módulo de control de acceso de UCI-Lab. Universidad de las Ciencias Informáticas.

NS, A. (2004). "Archivo de ayuda. ASSETS NS Sistema de Gestión Integral."

O.Gómez , Y. C., M.Tenrero (2010). "Libro de Ayuda de la Arquitectura Tecnológica del ERP-Cuba, en la versión 2.0 del Marco de Trabajo."

Paez, S. (2008). " Sistemas Paez Doce." from http://www.sistemaspaez.com/doce2/.

Pérez, I. N. (2008). "Curso de JavaScript." from http://www.elcodigo.net/tutoriales/javascript/javascript1.html.

PgAdmin, I. (2008). "PgAdmin III." from http://www.guia-buntu.org.

PostgreSQL, E. e. d. d. d. (2008). "Manual del usuario de PostgreSQL."

Pro, E. (2006). "Exactus Pro." from http://www.exactuspro.com/.

Rodríguez, R. M. (1998). "GREHU: Un sistema integral para gestionar los recursos humanos." Retrieved 8 de diciembre, 2010, from http://www.gestec.disaic.cu.
Salesiana, E. (2008) "Glosario Salesiano." Educación Salesiana.
Santiago, I. S. C. B. M. (noviembre 2009). "Programación Web II.".
Sitio, B. from http://www.sage.com.ar/pages/informacion_asistencia.html.
Smith, L. (Apr 2009). "Introduction to the Doctrine Object Relational Mapper."
SOFT, D. (2005). "DESIGN SOFT Control de asistencia." from http://occonsultores.com/software/nomina.html.
Software, S. R. d. (2002). "SPN. Red de Software." 2002, from http://www.elid.com/products/tms_etime.html.
Support, E. (2002). "Computer Design Company - Engineering Support. eTime Manual." Retrieved 26 de septiembre, 2002, from http://www.engineeringsupport.eu/timesheets/e-time_timesheet_manual.pdf.
Team, A. (2006). "Uso-de-la-tecnologia-ajax-en-el-desarrollo-Web." from http://www.abartiateam.com/desarrollo-Web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-Web.
UCI, M. (2010). Sistema de Control de Acceso a los laboratorios del modulo del MININT en la UCI. Universidad de las Ciencias Informáticas (UCI).
Villegas, S. A. (2004). " Proceso de registro y control de personal." El prisma.
XHTML, I. a. (2010). "Introducción a XHTML." from http://www.librosWeb.es/xhtml/.
Zaragoza, U. d. (2001). "Aplicación Web del Control Horario." from http://wzar.unizar.es/gerencia/pdf/control_horario.pdf.
Szyperski, C. (1998). "Component Software. Beyond Object-Oriented Programming."
PRESSMAN, R. S. 1998. Ingeniería de software. Un enfoque práctico. 1998.

# *Anexos*

**Anexo 1.** Registro de un Trabajador en la Interfaz Registrar Marcaje.

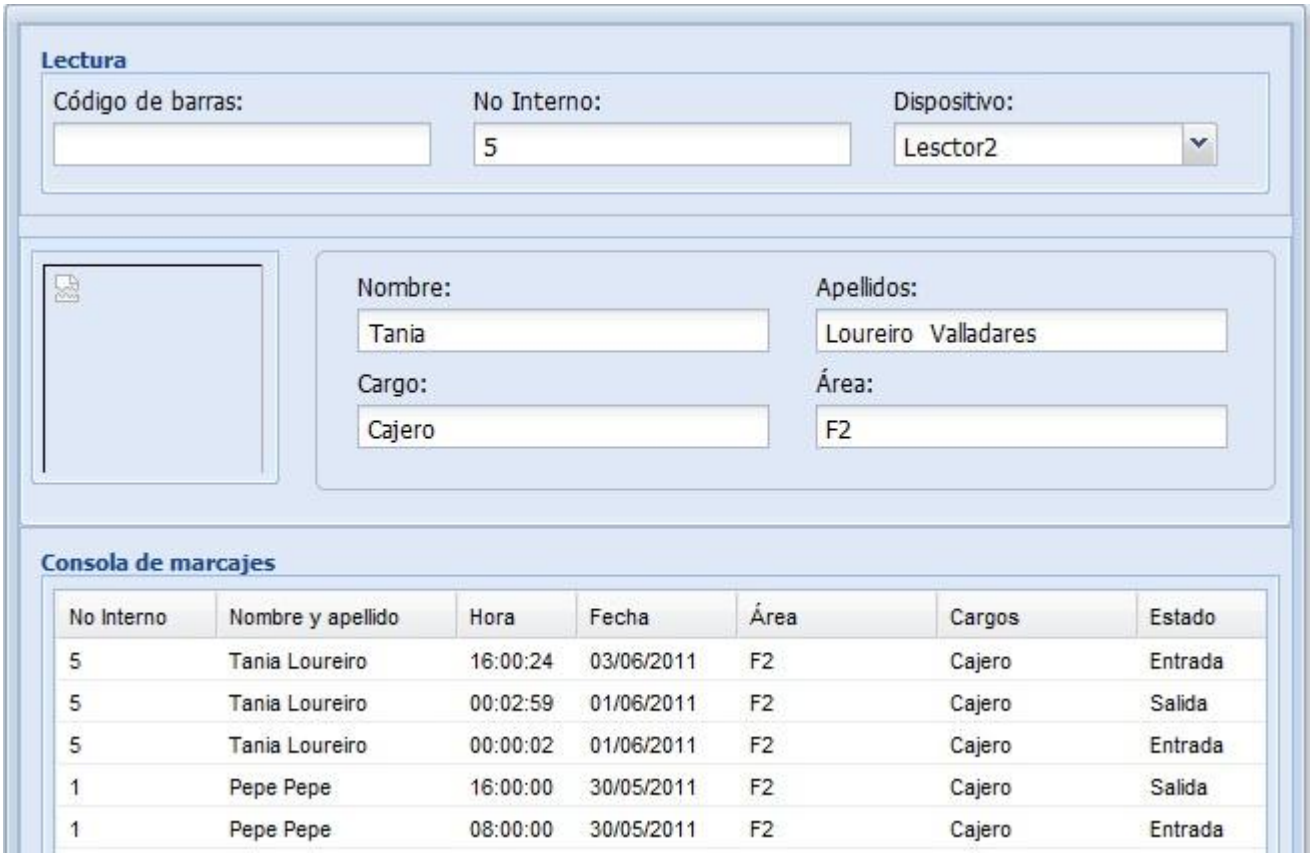


*Figura 21: Registro de un Trabajador en la Interfaz Registrar Marcaje.*

**Anexo 2.** Ventana de la Funcionalidad Ver en la Interfaz Gestionar Control de Asistencia.


*Figura 22: Ventana de la Funcionalidad Ver en la Interfaz Gestionar Control de Asistencia.*

**Anexo 3.** Ventana de la Funcionalidad Exportar Incidencias en la Interfaz Gestionar Control de Asistencia.


*Figura 23: Ventana de la Funcionalidad Exportar Incidencias en la Interfaz Gestionar Control de Asistencia.*

**Anexo 4.** Ventana de la Funcionalidad Adicionar en la Interfaz Gestionar Dispositivo.


*Figura 24: Ventana de la Funcionalidad Adicionar en la Interfaz Gestionar Dispositivo.*

**Anexo 5.** Ventana de la Funcionalidad Modificar en la Interfaz Gestionar Dispositivo.


*Figura 25: Ventana de la Funcionalidad Modificar en la Interfaz Gestionar Dispositivo.*

**Anexo 6.** Mensaje de Confirmación de la Funcionalidad Eliminar en la Interfaz Gestionar Dispositivo.

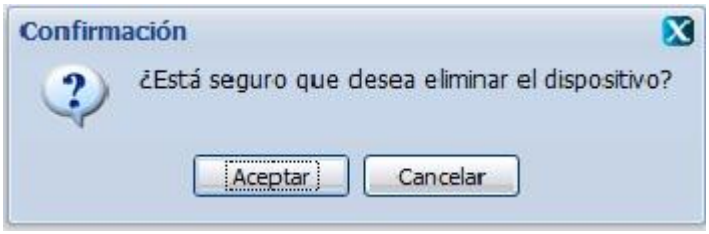


*Figura 26: Mensaje de Confirmación de la Funcionalidad Eliminar en la Interfaz Gestionar Dispositivo.*

**Anexo 7.** Filtro por Área y Fecha en la Interfaz Gestionar Presencia.


*Figura 27: Filtro por Área y Fecha en la Interfaz Gestionar Presencia.*

**Anexo 8.** Ventana de la Funcionalidad Ver en la Interfaz Gestionar Presencia**.**

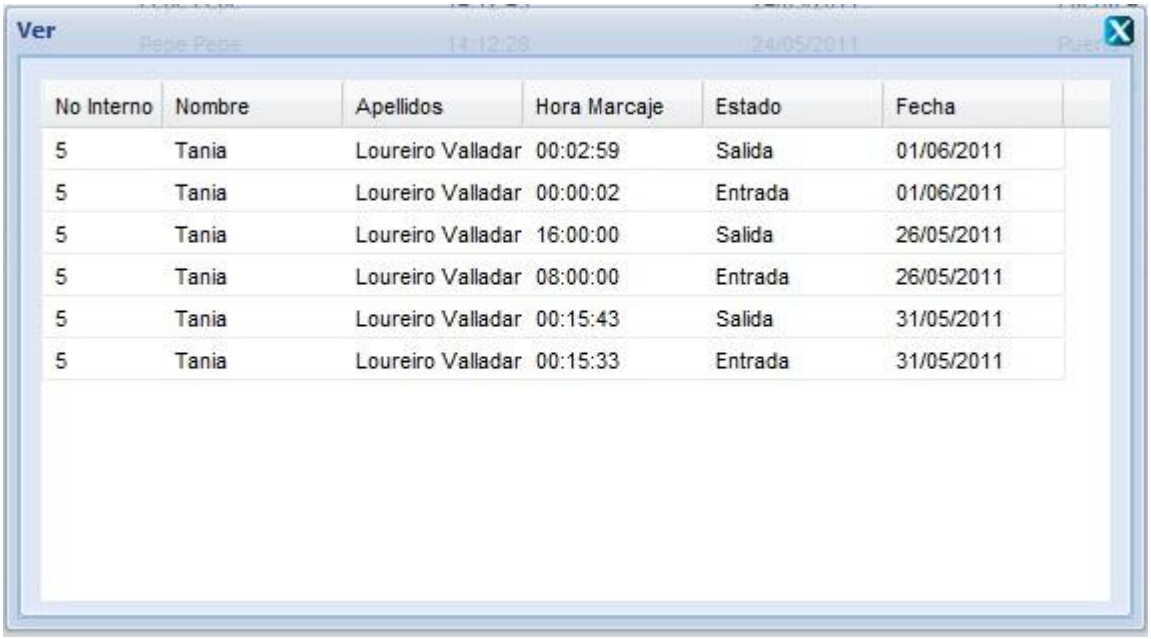


*Figura 28: Ventana de la Funcionalidad Ver en la Interfaz Gestionar Presencia.*

**Anexo 9.** Instrumento de medición de la métrica Tamaño operacional de clase (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. |

*Figura 29: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.*

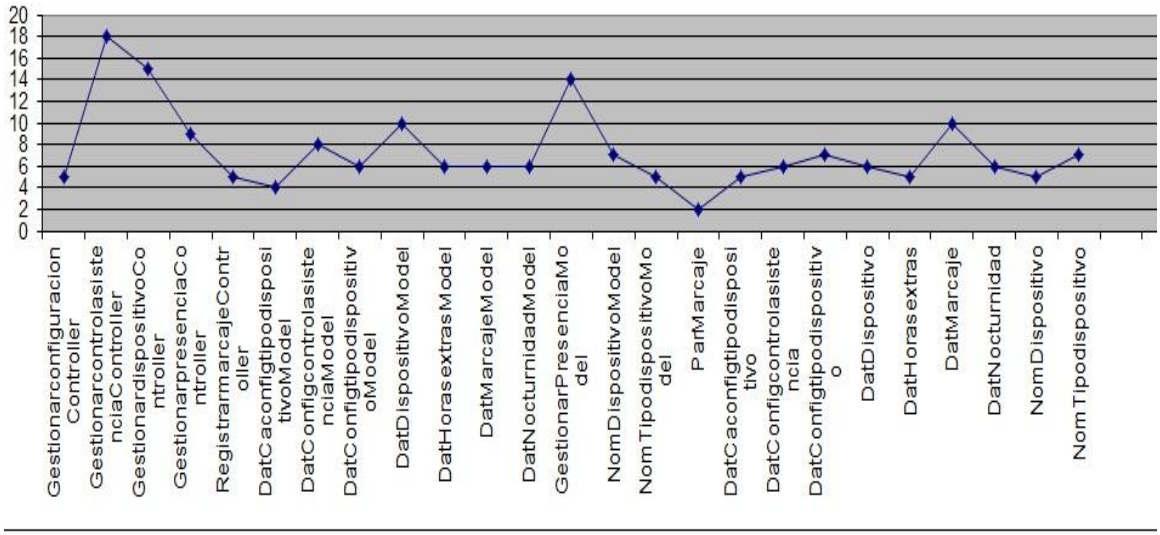


*Figura 30: Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).*

*Tabla 4: Categorización de las clases según sus dependencias.*

| Criterio | Categoría | Cantidad de clases | Promedio |
|---|---|---|---|
| 0 dependencias | Muy Bueno | 6 | 24 |
| 1 dependencias | Bueno | 8 | 32 |
| 2 dependencias | Regular | 8 | 32 |
| 3 dependencias | Malo | 3 | 12 |
| 3 dependencias | Muy Malo | 0 | 0 |
| Total | | 25 | 100 |

**Anexo 10.** Instrumento de medición de la métrica Relaciones entre clases (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. |

*Figura 31: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.*

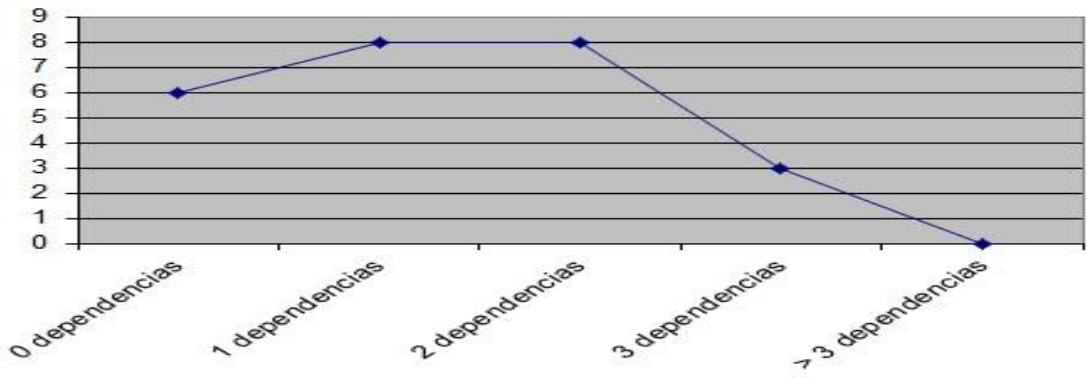


*Figura 32: Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los Valores.*

**Anexo 11.** Caso de Prueba Adicionar Dispositivo.

**Condiciones de ejecución.**

- ✓ Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- ✓ Se debe seleccionar el subsistema Capital humano/ Administración de capital humano/ Control de asistencia/Gestionar Dispositivo.

*Tabla 5: Requisitos a Probar.*

| **Nombre del requisito** | **Descripción general** | **Escenarios de pruebas** | **Flujo del escenario** |
|---|---|---|---|
| 1: Adicionar dispositivo | El sistema debe permitir adicionar dispositivos. | EP 1.1: Adicionar dispositivo introduciendo datos válidos. | – Se introducen los datos del dispositivo correctamente.<br>– Se presiona el botón **Aceptar**.<br>– Se muestra un mensaje de información.<br>– Se presiona el botón |

| | | | |
|---|---|---|---|
| | | | **Aceptar**. |
| | | EP 1.2: Adicionar dispositivo introduciendo datos válidos presionando el botón **Aplicar**. | – Se introducen los datos del dispositivo correctamente.<br>– Se presiona el botón **Aplicar**.<br>– Se muestra un mensaje de información.<br>– Se presiona el botón **Aceptar**. |
| | | EP 1.3: Adicionar dispositivo introduciendo datos inválidos. | – Se introducen los datos inválidos del dispositivo.<br>– Se presiona el botón **Aceptar**.<br>– Se muestra un mensaje informando del error. |
| | | EP: 1.4 Adicionar dispositivo introduciendo un código existente. | – Se introducen los datos del dispositivo con un código ya existente.<br>– Se presiona el botón **Aceptar**.<br>– Se muestra un mensaje informando del error. |
| | | EP 1.5: Adicionar dispositivo dejando campos vacíos. | – Se introducen los datos del dispositivo dejando algún campo en blanco.<br>– Se presiona el botón **Aceptar**.<br>– Se muestra un mensaje informando del error. |

| | | | |
|---|---|---|---|
| | | EP 1.6: Cancelar. | – Se introducen o no los datos del dispositivo.<br>– Se presiona el botón **Cancelar**. |

*Tabla 6: Descripción de variables.*

| **No** | **Nombre de campo** | **Tipo** | **Válido** | **Inválido** | **Inválido** | **Inválido** | **Inválido** |
|---|---|---|---|---|---|---|---|
| 1 | Identificador | Campo de texto | Números | Letras, caracteres especiales. | Vacío. | NA | NA |
| 2 | Denominación | Campo de texto | Letras | Números, caracteres especiales. | Vacío. | NA | NA |
| 3 | Descripción | Campo de texto. | Letras. | Números, caracteres especiales. | Vacío. | NA | NA |
| 4 | Tipo de dispositivo | Campo de texto no editable. | Letras. | Números, caracteres especiales. | Vacío | NA | NA |
| 5 | Área | Campo de texto no | Letras y números | Caracteres especiales. | Vacío | NA | NA |

*Tabla 7: Juego de datos.*

| **Id del escenario** | **Escenario** | **Identificador** | **Denominación** | **Descripción** | **Tipo de dispositivo.** | **Área** | **Respuesta del sistema** | **Resultado de la** |
|---|---|---|---|---|---|---|---|---|

| | | | | | | | | **prueba** |
|---|---|---|---|---|---|---|---|---|
| EP 1.1 | Adicionar dispositivo introduciendo datos válidos. | V(321) | V(Código de barra(PC)) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | El sistema adiciona el dispositivo y muestra el mensaje de información: | NA |
| EP 1.2 | Adicionar dispositivo introduciendo datos válidos presionando el | V(321) | V(Código de barra(PC)) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | El sistema adiciona el dispositivo y muestra el mensaje de información: “Se ha adicionado el | NA |
| EP 1.3 | Adicionar dispositivo introduciendo datos inválidos. | I(asad) | V(Código de barra(PC)) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | El sistema no permite la inserción de letras en este campo.<br><br>El sistema adiciona como código la | NA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | V(123) | I(2135) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | El sistema no permite la inserción de números en este campo.<br>El sistema adiciona como código la | |
| | | V(123) | V(Código de barra(PC) | I(212) | V(Código de barra(PC) | V(Facultad 3) | El sistema no permite la inserción de números en este campo. | |
| | | V(123) | V(Código de barra(PC) | V(Código de barra) | I(212) | V(Facultad 3) | El sistema no permite la inserción de números en este campo. | |
| | | V(123) | V(Código de barra(PC) | V(Código de barra) | V(Código de barra(PC) | I(123) | El sistema no permite la inserción de números en este campo. | |
| | | I(@#$$) | V(Código de barra(PC) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | El sistema no permite la inserción de | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | V(123) | I(@#$) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | caracteres especiales en este campo.<br>El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio.”.<br>El sistema mantiene la interfaz abierta. | |
| | | V(123) | V(Código de barra(PC) | I(@#$) | V(Código de barra(PC) | V(Facultad 3) | | |
| | | V(123) | V(Código de barra(PC) | V(Código de barra) | I(@#$) | V(Facultad 3) | | |
| | | V(123) | V(Código de barra(PC) | V(Código de barra) | V(Código de barra(PC) | I(@#$) | | |
| EP 1.4 | Adicionar dispositivo introducie | V(321) | V(Código de barra(PC)) | V(Código de barra) | V(Código de barra(PC) | V(Facultad 3) | El sistema muestra el mensaje “Ya existe un pago | |
| EP 1.5 | Adicionar dispositivo dejando campos vacíos. | I(vacío) | V(Código de barra(PC) | V(Código de barra) | V(Código de barra(PC) | V(facultad 3) | El sistema subraya el campo en rojo mostrando el mensaje: “Este | NA |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | V(212) | I(vacío) | V(Código de barra) | V(Código de barra(PC) | V(facultad 3) | campo es obligatorio.”.<br>El sistema mantiene la interfaz abierta. | |
| | | V(212) | V(Código de barra(PC) | I(vacío) | V(Código de barra(PC) | V(facultad 3) | | |
| | | V(212) | V(Código de barra(PC) | V(Código de barra) | I(vacío) | V(facultad 3) | | |
| | | V(212) | V(Código de barra(PC) | V(Código de barra) | V(Código de barra(PC) | I(vacío) | | |
| EP 1.6 | Cancelar. | NA | NA | NA | NA | NA | El sistema cierra la | NA |

# *Glosario de términos*

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

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

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

**Base de datos:** Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

**Programación del lado del cliente:** Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

**Programación del lado del servidor:** La Programación del lado del servidor es una tecnología que consiste en el procesamiento de una petición de un usuario mediante la interpretación de un script en el servidor Web para generar páginas HTML dinámicamente como respuesta.

**Programación orientada a objetos:** Consiste en ordenar datos en conjuntos modulares de elementos de información del mundo real (denominado un dominio). Estos elementos de datos se llaman objetos. Estos datos se agrupan de acuerdo a las características principales del mundo real de estos elementos (tamaño, color, etc.).

**Métricas:** Mediciones para el software que se pueden aplicar al proceso de desarrollo con el intento de mejorarlo sobre una base continua. Se utilizan para la estimación, el control de la calidad, la evaluación de productividad y el control de proyectos. Ayuda a evaluar la calidad de los resultados de trabajos técnicos y en la toma de decisiones tácticas a medida que el proyecto evoluciona.

**Cliente:** Persona, organización o grupo de personas que encarga la construcción de un sistema, ya sea empezando desde cero, o mediante un refinamiento de versiones sucesivas.

**CSS:** Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. XML: siglas en inglés de Extensible Markup Language («lenguaje de marcas), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, DOM: El Document Object Model (una traducción al español no literal, pero apropiada, podría ser Modelo en Objetos para la representación de Documentos), abreviado DOM, es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos HTML y XML, que es para lo que se diseñó principalmente. En efecto, el DOM es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como ECMAScript (Javascript).

**JSON:** Acrónimo de "JavaScript Object Notation", es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

**Linux:** Es el núcleo o kernel del sistema operativo libre denominado GNU/Linux (coloquial pero erróneamente llamado Linux). Lanzado bajo la licencia pública general (GPL - General Public License) de GNU y desarrollado gracias a contribuciones provenientes de todo el mundo, Linux es uno de los mejores ejemplos de software libre cuyos desarrolladores

originales siguieron la filosofía de ese movimiento. Linux fue creado por Linus Torvalds en 1991.

**Python:** Es un lenguaje de programación interpretado creado por Guido van Rossum en el año 1991. Python se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

**Apache:** El servidor HTTP Apache es un servidor Web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Tomcat: Es un servidor Web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache. Tomcat puede funcionar como servidor Web por sí mismo.

**Mac:** En redes de computadoras la dirección MAC (Media Access Control address o dirección de control de acceso al medio) es un identificador de 48 bits (6 bytes) que corresponde de forma única a una tarjeta o interfaz de red.