Universidad de las Ciencias Informáticas.

Facultad 3

Título: Implementación del componente Gestión de incidencia del subsistema Capital Humano del Sistema Integral de Gestión Cedrux

---

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

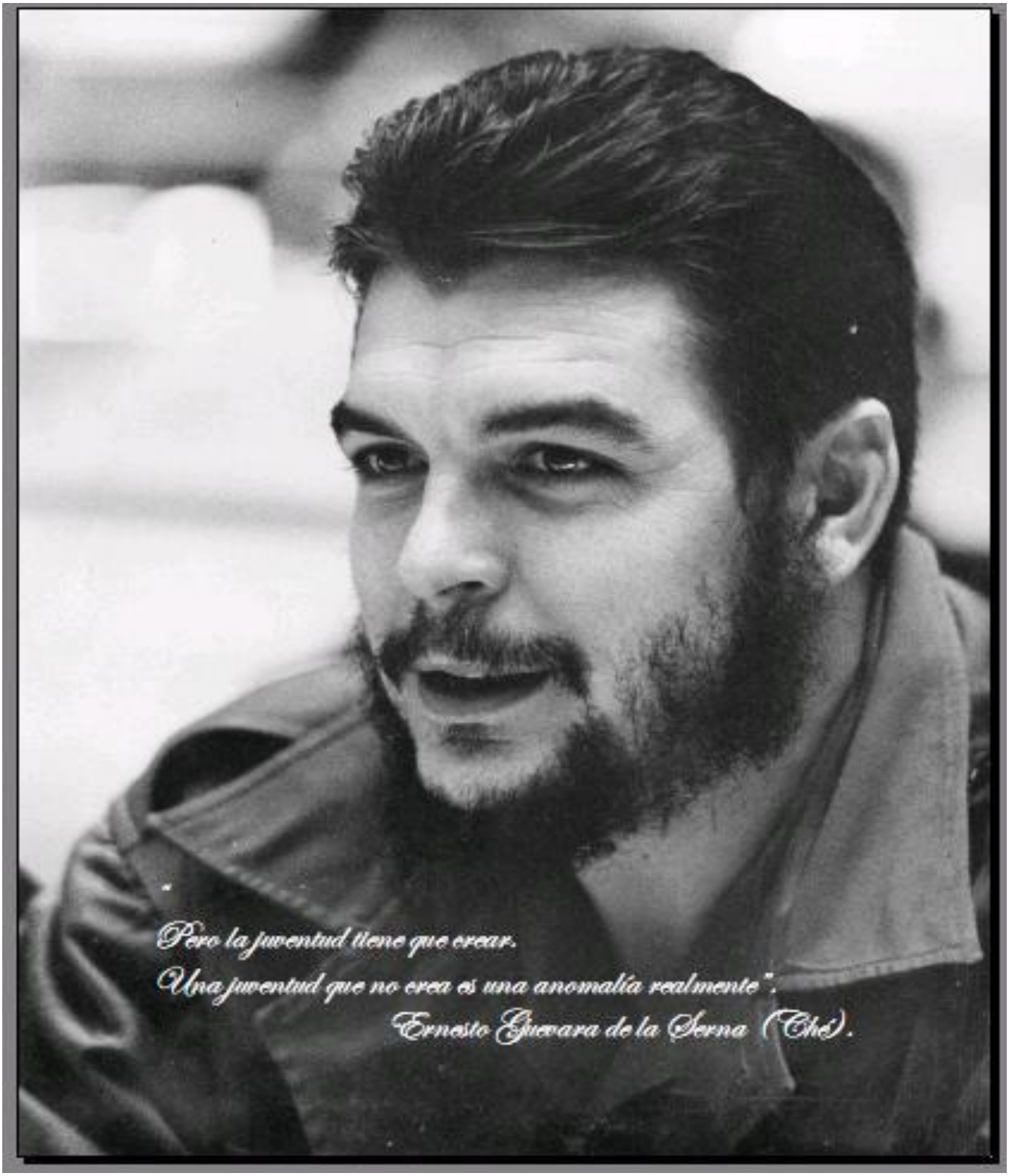
Autor: Grettel Marcos Martínez

Tutor: MSc.Donel Vázquez Zambrano

Cotutor: Ing. Arnolis Salgueiro Arzuaga

Ciudad de la Habana, Junio/2011

"Año del 52 Aniversario del Triunfo de la Revolución"


Pero la juventud tiene que crear.
Una juventud que no crea es una anomalía realmente".
Ernesto Guevara de la Serna (Che).

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ___ días del mes _____ del año 2011.

Grettel Marcos Martínez

_____

Autor

MSc.Donel Vázquez Zambrano

_____

Tutor

Ing. Arnolis Salgueiro Arzuaga

_____

Cotutor

AGRADECIMIENTOS

A mi mamá por ser la mejor del mundo, por apoyarme siempre en estos largos años de estudios, por haber sabido guiarme por el buen camino para llegar a ser una persona mejor en la vida, por brindarme su amor incondicional, por aguantarme todas las malcriadeces que aún así de grande no he podido superar y por depositar toda su confianza en mí.

A mi papá por ser mi ejemplo a seguir en la vida, ya que ha sido un padre muy luchador y que ha sabido llenar de amor y comprensión a su familia, por mantenerla siempre unida y por enseñarme muchas cosas en la vida, por estar siempre a mi lado y por complacerme en todo lo que he querido.

A mi hermano que ha sido lo más lindo que mis padres me han dado en la vida, por estar ahí cuando lo he necesitado, por enriquecer mi vida con su alegría, por ser parte de mí.

A mis abuelos, por recordarme que existen personas maravillosas en el mundo.

A mis tutores por ser los principales protagonistas de esta obra, ya que sin su ayuda no hubiera sido posible la realización de este trabajo.

A todos mis amigos que siempre me han apoyado y que me han permitido compartir momentos muy felices a su lado durante estos 5 años en la universidad.

Muchas gracias por hacer posible este sueño.

DEDICATORIA

Me gustaría dedicar esta Tesis a toda mi familia.

Para mis padres Idania e Iván, por su comprensión y ayuda en momentos malos y menos malos. Me han enseñado a encarar las adversidades sin perder nunca la dignidad ni desfallecer en el intento. Me han dado todo lo que soy como persona, mis valores, mis principios, mi perseverancia y mi empeño, todo ello con una gran dosis de amor y sin pedir nunca nada a cambio.

A mis amistades las cuales me ayudaron con su apoyo incondicional a ampliar mis conocimientos y estar más cerca de mis metas profesionales. Esto fue posible gracias a los intercambios y exposiciones de ideas con mis compañeros y amigos de estudios durante el proceso de desarrollo del trabajo de diploma.

MSc.Donel Vázquez Zambrano

Profesor Instructor graduado de Ingeniería en Ciencias Informáticas con Título de Oro y de 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.

Correo: dvz@uci.cu

Ing. Arnolis Salgueiro Arzuaga

Graduado de Ingeniería en Ciencias Informáticas con Título de Oro en la Universidad de las Ciencias Informáticas (UCI), Ciudad de La Habana, Cuba 2009.
Ha sido tutor de una tesis de pregrado.
Tiene en su haber 2 publicaciones. Ha cursado y aprobado 9 cursos de postgrado.

Correo: asalgueiro@uci.cu

Resumen

Para poder triunfar en el mercado, las empresas necesitan gestionar con mayor eficiencia los procesos que se llevan a cabo en ellas. Como parte del proceso de informatización de la economía cubana que se desarrolla actualmente en todo el país y dada la importancia de contribuir con la independencia tecnológica, se decide desarrollar un sistema que gestione los procesos de una empresa de manera integral, dirigido por la Universidad de las Ciencias Informáticas. En las entidades cubanas no existe una solución que englobe los procesos de gestión de incidencias, se utilizan actualmente algunas soluciones extranjeras que no cumplen con las necesidades de las empresas cubanas, lo que trae consigo que en gran parte de las entidades cubanas estos procesos se lleven a cabo de forma manual. Teniendo en cuenta las dificultades existentes en estos procesos empresariales donde el margen de error es mayor y la confiabilidad no es óptima, trae consigo la necesidad de implementar una solución que automatizara los procesos de gestión de incidencias de los trabajadores, donde los sistemas informáticos nacionales y extranjeros, utilizados actualmente no se adecuan a las particularidades propias de las entidades cubanas y en su mayoría la gestión de las incidencias está enfocada solamente al pago de las nóminas. El objetivo del trabajo es la implementación de un componente que gestione las incidencias, brinde en tiempo real la información necesaria para procesar las nóminas, sea implementado sobre tecnologías libres, adaptables a todas las entidades cubanas, respondiendo a las exigencias establecidas en la legislación laboral vigente.

Palabras Claves:

Capital Humano, Componente, Gestión, Incidencia, Procesos

Índice

INTRODUCCIÓN

Durante los primeros años de este siglo el país se ha propuesto informatizar la sociedad, lo que trajo consigo la creación de sistemas a la altura de las nuevas tecnologías. Las entidades empresariales no están ajenas a este proceso, debido a la existencia de muchos recursos que deben ser controlados de forma eficaz para un mejor funcionamiento y desarrollo de las mismas. El Capital Humano contribuye a que el desarrollo y la eficiencia de las entidades sean aún mayores. Su importancia en las empresas ha ido en aumento con el paso de los años, ya que estos requieren cada vez más de personal altamente calificado y motivado para adaptarse a los constantes cambios del entorno. Cambios que motivan la adopción de nuevas estrategias de desarrollo del potencial humano.

El trabajo de los gestores de Capital Humano es arduo por la cantidad de información que manejan, que pueden dar soporte al volumen de procesos que se realizan en estos departamentos, para la correcta adquisición, mantenimiento, desarrollo y gestión del personal. Además de los resultados derivados de su trabajo. Uno de estos procesos es la gestión de incidencias de los trabajadores en una empresa.

Es indispensable tener un sistema que recupere la información con mayor rapidez, fiabilidad y seguridad para gestionar las incidencias de los trabajadores, entendiéndose por incidencia, todo lo que le suma o le resta dinero al salario que estos devengan. En el país la gestión de las incidencias de un trabajador como proceso de Capital Humano son seguidas de forma manual. La información puede ser duplicada o perdida por el deterioro de los documentos haciéndose engorroso, lo que provoca que existan errores que afectan su funcionamiento. Un buen control de las mismas permite conocer el trabajador y el área de Contabilidad, la solicitud y aprobación de las vacaciones, el importe de los descuentos, así como los pagos a efectuar por conceptos de licencias o subsidios, sirviendo de base para la confección de las nóminas.

Debido a estas deficiencias existentes en el país se propone informatizar los procesos de gestión de incidencias, lo que trae consigo la creación de nuevos sistemas a la altura de las tecnologías y de la informatización de la sociedad.

Siguiendo una estrategia del país se ejecuta en la Universidad de las Ciencias Informáticas el programa ERP-Cuba cuyo objetivo fundamental es la obtención del producto Cedrux: sistema integral de gestión cubano que sustituye la amplia gama de aplicaciones informáticas usadas para actividades empresariales en el país. Este sistema contiene un subsistema de Capital Humano, el cual contará con un módulo para gestionar las incidencias.

De la problemática planteada surge el siguiente **problema a resolver**: La gestión manual de las Incidencias para la Administración de Capital Humano dificulta generar la nómina mediante el subsistema de Capital

Humano del Sistema Integral de Gestión Cedrux.

Teniendo en cuenta el problema antes planteado, queda enmarcado el **objeto de estudio:** Sistemas de Gestión del Capital Humano.

El **campo de acción** entonces quedaría delimitado por: Sistemas de Gestión de Incidencias.

El **objetivo general** de la presente investigación es**:** Implementar el componente Incidencias para su integración a la nómina del subsistema Capital Humano.

Con la intención de poder alcanzar el resultado deseado el objetivo general se desglosó en los siguientes **objetivos específicos:**

- ✓ Realizar la fundamentación teórica de la investigación mediante la elaboración del marco teórico para conocer el estado del arte del objeto de estudio de la presente investigación.
- ✓ Realizar la implementación de los requerimientos especificados para los procesos de gestión de incidencias.
- ✓ Validar la solución obtenida.

Con la finalidad de dar cumplimiento a los objetivos específicos se proponen las siguientes **tareas investigativas**:

- ✓ Estudio de los procesos de gestión de incidencias para la Administración del Capital Humano.
- ✓ Análisis de los sistemas existentes identificando sus características.
- ✓ Estudio de las herramientas y lenguajes propuestos para el desarrollo de la aplicación.
- ✓ Implementación de los requerimientos especificados para los procesos de gestión de incidencia.
- ✓ Realización de pruebas al resultado obtenido de caja blanca.
- ✓ Realización de pruebas al resultado obtenido de caja negra.

Hipótesis

Si se implementa el componente Incidencias, deben disminuir las dificultades para gestionar la nómina del Subsistema Capital Humano.

Los **posibles resultados** a obtener serían:

- ✓ Obtención del componente Gestión de incidencia integrado al subsistema Capital Humano del Sistema Integral de Gestión Cedrux que soporte los requisitos identificados en el estudio del proceso y permita el procesamiento de la nómina.
- ✓ Documentación generada en los procesos de análisis, diseño e implementación del componente Gestión de incidencia.

La tesis estará estructurada de la siguiente manera:

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA: En este capítulo se realiza el estudio de las metodologías, tecnologías y herramientas empleadas para el desarrollo de la solución propuesta.

CAPÍTULO 2: DESARROLLO DE LA SOLUCIÓN: Se describe la implementación de la solución realizando las integraciones entre el componente desarrollado y los existentes en el subsistema del Capital Humano.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN: Se abordan las pruebas realizadas al componente, específicamente las pruebas de caja blanca, además se hace una valoración de las mismas según los resultados obtenidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se realizará una investigación de los sistemas existentes vinculados a los procesos de gestión de las incidencias y la elaboración de las pre-nóminas, también se analizarán las metodologías, tecnologías, arquitectura y herramientas utilizadas para el desarrollo del componente propuesto.

En el país se utilizan aplicaciones informáticas para la gestión del Capital Humano. Algunos de estos sistemas sólo cuentan con la gestión del Capital Humano, otros brindan más servicios, como los ERP (Enterprise Resource Planning, Planificación de Recursos Empresariales) o se vinculan con otros subsistemas.

Este grupo de sistemas en su gran mayoría se utilizan para informatizar los procesos contables y financieros de las entidades. Muchas aplicaciones que se utilizan en empresas del mundo no se ajustan al sistema del país, por las características del modelo económico que existe en Cuba.

1.2 Sistemas informáticos estudiados

En el país las empresas utilizan para la gestión de los recursos humanos sistemas tanto de producción extranjera como desarrollado en Cuba. Algunos de estos sistemas sólo cuentan con la gestión del capital humano.

1.2.1 Sistemas Internacionales

1.2.1.1 Sistema Ontime

El Sistema Ontime es un sistema de control de asistencias que administra las incidencias de nómina de las empresas. Es decir, entradas, salidas, salidas a comer, faltas, retardos, incapacidades, vacaciones, permisos con y sin goce de sueldo. Todo esto con sus respectivos reportes para revisar en pantalla o impresos.

Además el Sistema Ontime cuenta con un módulo generador de credenciales con fotografía y código de barras, muy útil para crear identificaciones de empleados sin que la información salga de tu empresa. La identificación puede ser mediante código de barras, tarjeta de proximidad, Radiofrecuencia (RFID) y/o un sistema biométrico, que puede ser: huella digital, Palma de la mano (hand punch), reconocimiento de iris o reconocimiento de rostro.

- ✓ La comunicación con tu PC o servidor puede ser mediante TCP/IP, USB o RS232.
- ✓ En los casos de reconocimiento biométrico, la ventaja más atractiva es que elimina por completo chequeadas falsas y horas extras inexistentes. (1)

Valoración del sistema

Este sistema realiza la gestión de las incidencias, pero no permite la creación de la nómina, por lo que dicho proceso queda incompleto. Además resulta imprescindible incorporarle la dualidad monetaria.

1.2.1.2 ADP Expert

ADP Expert le da la oportunidad de compartir las herramientas de trabajo, el conocimiento y el capital humano de manera flexible y progresiva, en función de sus necesidades, su presupuesto y su organización, con dos modalidades de servicio que pueden incorporar servicios opcionales y porque la externalización es la solución escalable para alcanzar los objetivos de eficiencia y calidad de servicio de su departamento de recursos humanos (RRHH), ADP Expert le permite evolucionar el nivel de servicio.

Ventajas económicas y organizativas

- ✓ El beneficio en costes: Cuenta con personal especializado sin incrementar la masa salarial.
- ✓ La disminución del riesgo: Los datos están a salvo y accesibles ante cualquier siniestro. La continuidad del servicio está asegurada en todas las circunstancias (cambios organizativos y bajas imprevistas).
- ✓ El aumento de la flexibilidad: El departamento de recursos humanos se adapta mejor a los cambios de plantilla.
- ✓ La optimización de los recursos internos: Ahorran tiempo y se dedican a tareas más estratégicas.
- ✓ La tranquilidad: Tiene a su lado el mejor partner del sector de los recursos humanos (RRHH).
- ✓ Su empresa se encarga de la gestión de su nómina y sus recursos humanos (RRHH) con una herramienta de la máxima eficacia mantenida por ADP.
- ✓ ADP se encarga de todos los procesos de nómina y sus recursos humanos (RRHH) de su empresa y le proporciona además un asesoramiento experto. (2) (3)

Valoración del sistema

Este sistema solo permite realizar la gestión de la nómina y las empresas cubanas también necesitan gestionar las incidencias e incorporar la dualidad monetaria.

1.2.1.3 ASSETS NS

ASSETS NS es un Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos.

El Módulo de Recursos Humanos está concebido para calcular las nóminas y controlar los recursos laborales de una entidad.

El sistema puede calcular y contabilizar nóminas de salario para cualquier tipo de pago (sueldo fijo, jornal, pago por rendimiento), incluyendo el pago de horas extras, interrupciones y condiciones laborales anormales. De igual forma, calcula y contabiliza las nóminas de vacaciones y subsidios y las nominillas de diferentes tipos (salario, vacaciones, subsidios, reintegros y estimulación). Realiza deducciones automáticas de cualquiera de las nóminas que se calculen. Todos los procesos automáticos se realizan siguiendo los criterios y restricciones establecidos por la legislación laboral vigente (determinación del fondo de tiempo, cálculo de las vacaciones y salario promedio para el subsidio).

Las opciones que brinda este módulo son las siguientes:

- ✓ Nominas
- ✓ Pagos de Vacaciones
- ✓ Pagos de Subsidios
- ✓ Pagos de Horas Extras
- ✓ Pagos por Condiciones Laborales Anormales
- ✓ Nominillas
- ✓ Nómina de Retenciones
- ✓ Reintegros de Salario, Subsidio y Vacaciones
- ✓ Submayor de Vacaciones
- ✓ Registro de Salario y Tiempo de Servicio
- ✓ Submayor de Retenciones
- ✓ Nómina de Divisa
- ✓ Prenómina
- ✓ Nómina de Divisa

Requerimientos Técnicos:

✓ Requerimientos mínimos: PC Pentium a 100 Mhz, 32 MB RAM, 100 MB de espacio disponible de disco duro, Windows 95 o superior

✓ Requerimientos recomendados: PC Pentium a 166 Mhz, 64 MB RAM, 100 MB de espacio disponible de disco duro, Windows 95 o superior

Ventajas y beneficios

- ✓ ASSETS NS es una aplicación cliente-servidor programada en Visual Basic 6.0 y Microsoft SQL Server 2000, utilizando adicionalmente Crystal Reports 7.0 para la generación de reportes de salidas
- ✓ Al estar en plataforma SQL, garantiza mayor seguridad y consistencia en los datos, se obliga que sea ilimitado el número de usuarios conectados y hace posible la utilización de servidores remotos
- ✓ Todos los procesos están implementados con inicio y final de transacciones, lo que garantiza la integridad de la base de datos ante fallos de corriente, cambios de voltaje o cualquier otra eventualidad que provoque una falla en la operación del sistema (4) (5)

Valoración del sistema

Este sistema tiene la desventaja de que solo permite la realización del proceso de pago de la nómina de los trabajadores y no facilita la gestión de las incidencias. Además encuentra desarollado sobre plataforma propietaria.

1.2.2 Sistemas Nacionales

1.2.2.1 Versat Sarasola

Es un producto cubano cuyo Sistema Integral de gestión es el primer sistema de contabilidad certificado, desarrollado para la gestión económica, este estuvo asesorado por el Ministerio de Finanzas y Precios, consultorías internacionales y el organismo encargado de la seguridad informática. Surge en 1998 con el objetivo de gestionar la contabilidad y finanzas de una empresa.

El Versat Sarasola, incluye varios módulos. Control de Activos Fijos e Inventarios, Finanzas y Planificación, son algunos de ellos. Otros, como Contabilidad General, Nómina de Salarios y Facturación, se incluyen hasta sumar diez. El resultado es un sistema integral, que se actualiza constantemente en función de nuevas demandas y viabiliza, sin dudas, la organización y el control en el área económica.

El módulo de recursos humanos permite el control, planificación y gestión de la actividad de recursos humanos aplicable en todas las entidades, como exportar las incidencias y los datos de los trabajadores para el módulo de Nóminas.

Se pueden realizar movimientos de nóminas, se generan automáticamente los contratos según la legislación vigente y la actualización automática de la plantilla. (6) (7) (8)

Características:

- ✓ Es una aplicación de escritorio.
- ✓ Implementado en Delphi.
- ✓ Trabaja sobre el sistema operativo Windows.
- ✓ Soporte para base de datos SQL Server 2000.

Valoración del sistema

Este sistema cumple con las necesidades de las empresas cubanas para la gestión de incidencias y la elaboración de la nómina, pero tiene el inconveniente de estar desarrollado sobre una plataforma propietaria lo cual no es factible para la economía del país.

1.3.2.2 Rodas XXI

La empresa CITMATEL ha desarrollado el Sistema Integral Económico Administrativo RODAS XXI que posibilita automatizar el funcionamiento de cualquier empresa. Es un sistema multiempresa que cuenta actualmente con seis módulos: Finanzas, Contabilidad, Activos Fijos, Nóminas, Inventario y Facturación. Estos módulos pueden emplearse integrados en su totalidad, formando cualquier subconjunto entre ellos, o cada uno de forma independientes. Rodas XXI es un sistema que gestiona los trabajadores de forma organizada sin embargo no cumple con requisitos establecidos en las Normas Cubanas (NC) para un Sistema de Gestión. (9)

Configuración mínima requerida para ejecutarse.

- ✓ Computadora IBM PC o compatible.
- ✓ Pentium III o superior.
- ✓ 256 MB de memoria RAM o más.
- ✓ Unidad de disco flexible, un disco duro, lector de CD y sistema operativo Windows 2000/XP.
- ✓ 20GB de espacio en disco duro.

RODAS XXI le ofrece además las siguientes posibilidades:

- ✓ Maneja un número ilimitado de empresas.
- ✓ Puede instalarse tanto en estaciones de trabajo como en redes locales.
- ✓ Sus módulos pueden ajustarse fácilmente a las características de cada usuario.
- ✓ Permite el intercambio automático de los comprobantes generados por cada módulo con el de Contabilidad.
- ✓ Trabaja con doble moneda.
- ✓ Crea reportes fácilmente.
- ✓ La información está protegida por claves.
- ✓ Lleva un registro de las operaciones relacionadas con el sistema, que permiten auditar el mismo.

Valoración del sistema

Este sistema facilita la gestión de las nóminas, pero no cumple con los requisitos establecidos en las normas

cubanas y no permite realizar la gestión de las incidencias.

1.3.2.3 Fastos.

El sistema de Recursos Humanos (Fastos), 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 capital humano tales como:

- ✓ 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: lo cual incorpora el control de claves de asistencias, tarjeta de asistencia e incidencias de cada empleado. Permite acoplar relojes (RTA 600) para actualizar la información de la tarjeta de asistencia de forma automática, 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.

Valoración del sistema

Este sistema permite la gestión de las incidencias mediante el control de asistencia, pero no facilita la creación de la nómina para el pago de los trabajadores, por lo que el proceso de las incidencias queda incompleto.

1.3 Análisis del estudio realizado

Aún cuando en el ámbito nacional e internacional se encuentran en explotación varias aplicaciones que gestionan las incidencias, se observó que las soluciones que brindan los software internacionales solo posibilitaban un trabajo incompleto, pues no cumplen con los requisitos necesarios. Algunos permiten trabajar con las nóminas, otros con las incidencias, pero ninguno integra estos dos procesos en un mismo paquete. Otra desventaja que presentan estos sistemas es que en su mayoría son soluciones que están desarrolladas sobre plataformas propietarias, lo que implica el pago de licencias, capacitación y soporte,

además es un inconveniente para el proceso de migración hacia software libre en el que está enmarcado el país. Por otro lado estas soluciones están desarrolladas para economías totalmente diferentes a la de Cuba, por lo que no se incluyen en su solución la dualidad monetaria, una característica totalmente nacional. Luego de no encontrar un sistema adaptable a las características y necesidades del cliente, así como a las tecnologías usadas en el desarrollo del proyecto ERP-Cuba, se concluye que es necesario crear un nuevo componente que gestione las incidencias, para su integración con el componente nómina del subsistema Capital Humano del sistema Cedrux. (10) (11)

1.4 Modelo de Desarrollo

Para el desarrollo de Cedrux se elaboró una metodología de software específica que responde a la dinámica del proyecto. Con la descripción de la misma los equipos de desarrollo poseen un modelo estandarizado, así como una definición clara y precisa de las responsabilidades y acciones a realizar en cada momento. Esta metodología o modelo de desarrollo está basado en las metodologías RUP y XP, tomando de estas las características más notables; de esta forma se obtiene como resultado el documento Modelo de desarrollo orientado a componentes del Proyecto ERP-Cuba. En dicho documento se definen las características y necesidades de un Proyecto ERP, el organigrama organizativo de las líneas de desarrollo, los diferentes roles involucrados con las responsabilidades de cada uno, las actividades a realizar durante todo el procesos de desarrollo del software con cada uno de los roles involucrados en dichas actividades y los artefactos generados por cada uno de ellos. Además de las métricas para la medición del avance del proyecto.

1.5 Tecnología de Desarrollo

1.5.1 Marco de trabajo

El desarrollo de la solución se realizará utilizando el marco de trabajo Sauxe, implementado por el Departamento de Tecnología del Centro de Informatización de la Gestión de Entidades (CEIGE).El primer elemento que debe regir una selección adecuada de alcance de un marco tecnológico radica en las restricciones de diseño que el mismo asume, basados en las tecnologías, la capacidad técnica del equipo de desarrollo, así como la infraestructura, tanto de la organización productora como del cliente de los productos que el mismo facilitará desarrollar. Sauxe cuenta con una arquitectura en capas como se muestra en la figura 1 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. (12)

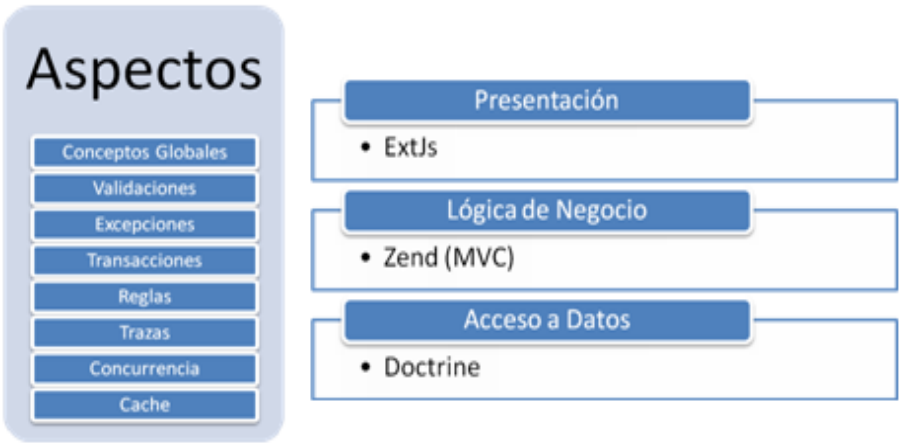


Figura1. Arquitectura de Sauxe.

1.5.1.1 Zend_Ext

Es un marco de trabajo de código abierto, que está diseñado para PHP 5 y tiene buenas capacidades de ampliación. Es elaborado a partir de zend framework de trabajo cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizadas por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor, se le incluyó el IoC para la comunicación entre los módulos o componentes, la integración con el ORM Doctrine framework para trabajo en la capa de abstracción a base de datos y el ExtJs para el desarrollo de las vistas.

1.5.1.2 Zend_Framework 1.7

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 (código abierto) 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 aúnque alguno es posible que no lo usemos nunca, hay otros que pueden que los usemos hasta la saciedad, por ejemplo el componente para la BD. Entre los componentes de vital importancia se encuentran: 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 tendremos que añadir en el fichero de configuración php.ini, el path hasta la carpeta library del framework con la instrucción include_path. (13)

Características:

✓ Trabaja con MVC (Model View Controller)

- ✓ Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo).
- ✓ El Marco de Zend también incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar base de datos, sin tener que escribir ninguna consulta SQL.
- ✓ Una solución para el acceso a base de datos que balancea el ORM con eficiencia y simplicidad.
- ✓ Completa documentación y tests de alta calidad.
- ✓ Soporte avanzado.
- ✓ Un buscador compatible con Lucene.
- ✓ Robustas clases para autenticación y filtrado de entrada.
- ✓ Clientes para servicios web, incluidos Google Data APIs y StrikeIron.
- ✓ Muchas otras clases útiles para hacerlo tan productivo como sea posible.(13, 14)

1.5.1.3 Doctrine 0.1

Doctrine es un potente y completo sistema mapeador relacional de objetos (ORM) para PHP con un capa de abstracción de bases de datos (DBAL) incorporado que permite exportar una base de datos a sus clases correspondientes y viceversa, o sea, a partir de las clases creadas y siguiendo las especificaciones de ORM, generar las tablas de la base de datos. Se encuentra en la parte superior de una poderosa Capa de Abstracción de Base de Datos (DBAL). Una de sus principales características es la opción de escribir las consultas de base de datos en un objeto con una propiedad orientada al dialecto SQL llamada Doctrine Query Language (DQL), inspirada en Hibernate HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (15)

1.5.1.4 Extjs 2.2

Es una librería Java Script código abierto de alto rendimiento para la creación y desarrollo de aplicaciones web dinámicas. Su potencia radica en la rica colección de componentes para el diseño de interfaces de usuarios del lado del cliente haciendo uso extensivo de Ajax. Permite la creación de aplicaciones complejas utilizando componentes predefinidos. Es extensible para la gran mayoría de los navegadores, evitando el tedioso problema de validar el código para cada uno de estos. Entre sus principales ventajas se encuentra el balance entre Cliente-Servidor, distribuyendo la carga de procesamiento en el último, este al tener menor carga, maneja los clientes de manera más eficiente. La comunicación asíncrona permite el intercambio de información con el servidor sin necesidad de pedirle una acción al usuario, dando la libertad de cargar la información sin que este lo note. (15)

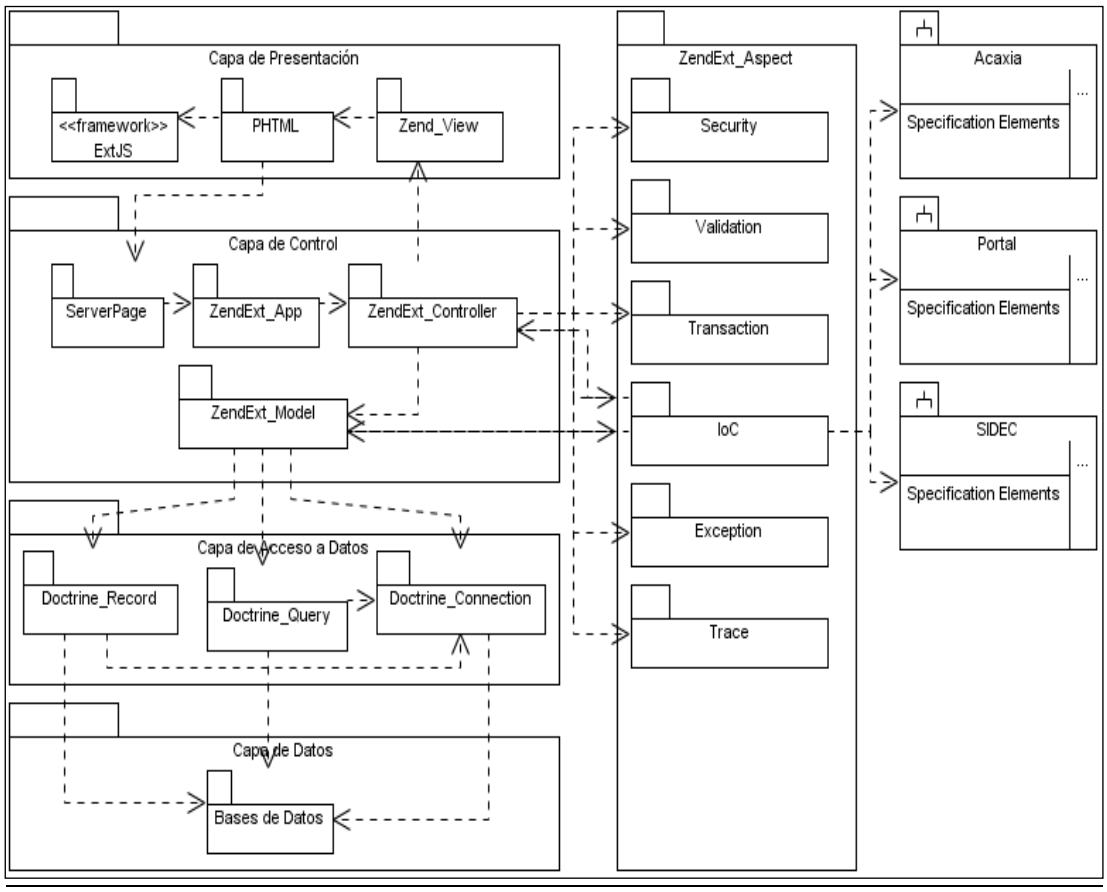


Figura 2: Gráfico estructural

1.5.2 Arquitectura

La arquitectura es basada en capas aunque en las capas superiores implementa un Modelo Vista Controlador (MVC).Está compuesto básicamente por cinco niveles o capas:

✓ **Capa de Presentación:** En esta capa se emplea las facilidades que brinda el Marco de Trabajo ExtJS para la construcción de interfaces amigables a la vista de los usuarios.

✓ **Capa de Control o Negocio:** En esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC). De forma vertical al modelo descrito hasta este momento

✓ **Capa de Acceso a Dato:** En esta capa estará presente el ORM (Object Relational Mapping) Doctrine, como Marco de Trabajo de persistencia para la comunicación con el servidor de datos mediante el protocolo PDO.

✓ **Capa de Dato:** En esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de Ficheros de Configuración de la arquitectura tecnológica.

✓ **Capa de Servicio:** En esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí.

Esta gran estructura descrita, se comunica con unas series de servicios Web mediante protocolos SOAP, e IoC, que interactúan con el sistema proporcionándole un conjunto de funcionalidades tanto de seguridad como de negocio.(16)

1.5.3 Herramientas utilizadas

1.5.3.1 Herramientas Case

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Una innovación en la organización, un concepto avanzado en la evolución de tecnología con un potencial efecto profundo en la organización. Se puede ver al CASE como la unión de las herramientas automáticas de software y las metodologías de desarrollo de software formales. (17)

1.5.3.1 Sistema Gestor de Base de Datos (SGBD) PostgreSQL 8.3

Es un sistema de gestión de bases de datos libre basado en el proyecto Postgres, perteneciente a la Universidad de Berkeley. Es un sistema objeto-relacional, que incluye características como la herencia, valores no atómicos (atributos basados en vectores y conjuntos), funciones, disparadores, entre otras. Es altamente extensible, permitiendo el uso de operadores, funciones y tipos de datos definidos por el usuario. Soporta la integridad referencial garantizando la integridad de los datos en la base de datos. PostgreSQL permite realizar múltiples conexiones desde procesos clientes, existiendo un proceso maestro en el servidor que siempre se ejecuta y que está a la espera de nuevas conexiones clientes, de forma tal que cuando alguien se conecta, se inicia un nuevo proceso, asegurando que el cliente y la nueva conexión no necesitan del proceso postgres original. Una de sus principales características es la alta concurrencia. Esto permite que mientras se realizan cambios en una tabla, otros procesos accedan a la misma sin la necesidad de bloqueos, además de que cada usuario tiene visión de la última modificación. Presenta el inconveniente de que para bases de datos pequeñas su velocidad de respuesta no es muy eficiente en comparación con otras relativamente grandes. (18)

1.5.3.2 Navegador Mozilla Firefox 2.17

Es un navegador de software libre. Entre sus características se encuentran que presenta una forma rápida y eficiente de navegar por la web, que le permite abrir varias páginas en una misma ventana mediante el empleo de pestañas separadas. Contiene un Plugin Firebug que se utiliza para ver los errores del código.

Firefox es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre, publicado bajo una triple licencia GPL/LGPL/MPL. (19)

1.5.3.3 Servidor Web Apache 2.2.9

Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal. Es una tecnología gratuita de código fuente abierta. Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor. Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Tiene una alta configurabilidad en la creación y gestión de logs. (20)

1.5.3.4 Visual Paradigm 6.4

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. (21)

1.5.3.5 NetBeans 6.9

Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso. (22)

1.5.3.6 SVN (Subversión) 1.6.6

Es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversión es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo. (23)

1.6 Conclusiones

Con el estudio de sistemas que permiten el control de la gestión de las incidencias existentes en el mundo y

en Cuba, se llega a la conclusión de que los mismos han servido para agilizar los procesos de gestión de incidencias en una empresa, teniendo un mayor control y erradicando en gran medida los errores que se cometían cuando se hacía todo de forma manual. Estos sistemas tienen la limitante al abordar solamente partes del problema de la gestión de las incidencias y de la creación de la nómina en una entidad. Estas soluciones están desarrolladas para economías totalmente diferentes a la nuestra y no incluyen en sus soluciones la dualidad monetaria, que es una característica totalmente nacional. Otros fueron implementados con tecnologías propietarias interfiriendo en el proceso de migración hacia software libre que está llevando el país. Se determinó la necesidad de desarrollar un componente de gestión de incidencias que se integra al componente nómina del subsistema Capital Humano perteneciente al sistema Cedrux. En el desarrollo del componente se utilizarán las herramientas y tecnologías definidas por el marco de trabajo Sauxe.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

En este capítulo se realiza una valoración de la propuesta de diseño del sistema, exponiendo las principales ventajas y deficiencias del mismo. Se expone el uso de la arquitectura y las posibilidades proporcionadas por el framework y las librerías utilizadas en la programación de la aplicación. Para una mayor comprensión de las funcionalidades del componente implementado, se realizaron las descripciones de las clases así como las operaciones utilizadas.

2.2 Valoración crítica de los artefactos propuestos por los analistas

Los requisitos funcionales descritos por los analistas facilitaron el entendimiento del proceso, 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 las capas, lógica del negocio, presentación y datos.

2.3 Especificación de los requerimientos del software

Los Requerimientos funcionales: Son capacidades o condiciones que el sistema debe cumplir. Los requisitos funcionales descritos por los analistas y que posibilitaron la identificación de las clases y las funcionalidades a implementar son los siguientes:

Tabla 1: Descripción de los Requisitos funcionales

Agrupación	Requisito
R1-Gestionar tipo de incidencia	R1.1- Adicionar tipo de Incidencia.
R2-Gestión incidencias	R2.1-Modificar tipos de incidencias
	R2.2-Eliminar tipo de incidencia
	R2. 3-Listar tipo de incidencia
	R2.4-Imprimir tipos de incidencias
R3-Gestionar incidencias	R3.1-Adicionar incidencia
	R3.2-Modificar incidencia
	R3.3-Eliminar incidencia
	R3.4-Listar incidencia
	R3.5-Justificar incidencia
	R3.6-Imprimir incidencias

R4-Elaborar pre-nomina	R4.1-Adicionar prenómina
	R4.2-Modificar pre-nomina
	R4.3-Eliminar Pre-nomina
	R4.4-Listar pre-nomina
	R4.5-Confirmar pre-nomina
	R4.6-Imprimir pre-nomina
	R4. 7-Imprimir reporte de ausencias
	R4.8-Adicionar trabajador a una pre-nomina
	R4.9-Eliminar trabajador de una pre-nomina
	R4.10-Registrar incidencias al trabajador

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

Los requerimientos no funcionales: Son propiedades o cualidades que el producto debe tener, que lo hacen atractivo, usable, rápido y/o confiable.
Existen múltiples categorías para clasificar a los requerimientos no funcionales.

Seguridad

Autenticación y Autorización (Contraseña de acceso). Protección contra maniobras no autorizadas o que puedan afectar la integridad de los datos. La vigilancia al sistema así como la salva de la información, se realizará de forma centralizada por el administrador, además el sistema debe mostrar opción de advertencia antes de borrar cualquier elemento o información que pueda existir.

Software

Para el cliente:

- ✓ Navegador Mozilla Firefox 2.2 o superior.
- ✓ Sistema operativo Windows XP o Linux.

Para el servidor:

- ✓ Sistema operativo Linux en cualquiera de sus distribuciones.
- ✓ Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.

- ✓ Un servidor de base de datos PostgreSQL 8.3 o superior.

Hardware

Para el servidor:

- ✓ Requerimientos mínimos: Procesador Pentium IV a 2GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- ✓ Al menos 50Gb de espacio libre en disco duro.
- ✓ Tarjeta de red.

Para el cliente:

- ✓ Requerimientos mínimos: Procesador Pentium III a 1GHz con 256Mb de memoria RAM.
- ✓ Tarjeta de red.

Confiabilidad

El subsistema debe ser confiable y preciso en la información que le suministra al usuario para evitar cualquier tipo de error. Estará disponible todo el tiempo, permitiendo el trabajo a los usuarios y las acciones de mantenimiento. Este debe ser estable, fiable y la velocidad de respuesta debe ser rápida durante la utilización del mismo. La información almacenada debe ser confiable en cuanto a su veracidad e integridad desde su recopilación.

Portabilidad

El subsistema será multiplataforma (Linux-Windows) lo que permitirá ejecutarse sobre diferentes sistemas operativos sin importar sus versiones y sin necesidad de modificar su código fuente.

2.4 Descripción del componente

Componente Incidencias

Se encarga de registrar los tipos de incidencias y de asignar una incidencia a un trabajador específico. Con estos datos se elabora la prenómina la cual es la entrada al componente nómina. Este a su vez se encuentra integrado con otros componentes del subsistema Capital Humano.

✓ Componente Vacaciones: Se encarga de la planificación de las vacaciones pagadas, de emitir notificaciones de vacaciones con respecto a estos planes o fuera de ellos .Además brinda también la posibilidad de generar un reporte de submayor de vacaciones.

✓ Componente Subsidio (Seguridad Social): Se encarga de realizar un registro, control de los certificados médicos de los trabajadores y las notificaciones de subsidio, por las cuales se realizará el pago

de estos mediante el sistema de nómina.

✓ Componente Trabajador: Se encargara de gestionar los datos de los trabajadores, mediante la contratación y posterior actualización de estos datos cuando ocurre alguna modificación en ellos.

✓ Componente Concepto de Pago: Permite definir los conceptos de pago utilizados en la entidad dígase subsidio, vacaciones y estimulación.

✓ Componente Período de Pago: Permite definir el período de pago ya se la primera quincena de un mes, o un rango de fecha específico.

2.5 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes y bibliotecas cargadas dinámicamente.

Muestra como el sistema está dividido en componentes y las dependencias entre ellos. Provee una vista arquitectónica de alto nivel del sistema y ayuda a los desarrolladores a visualizar el camino de la implementación. (24) (25)

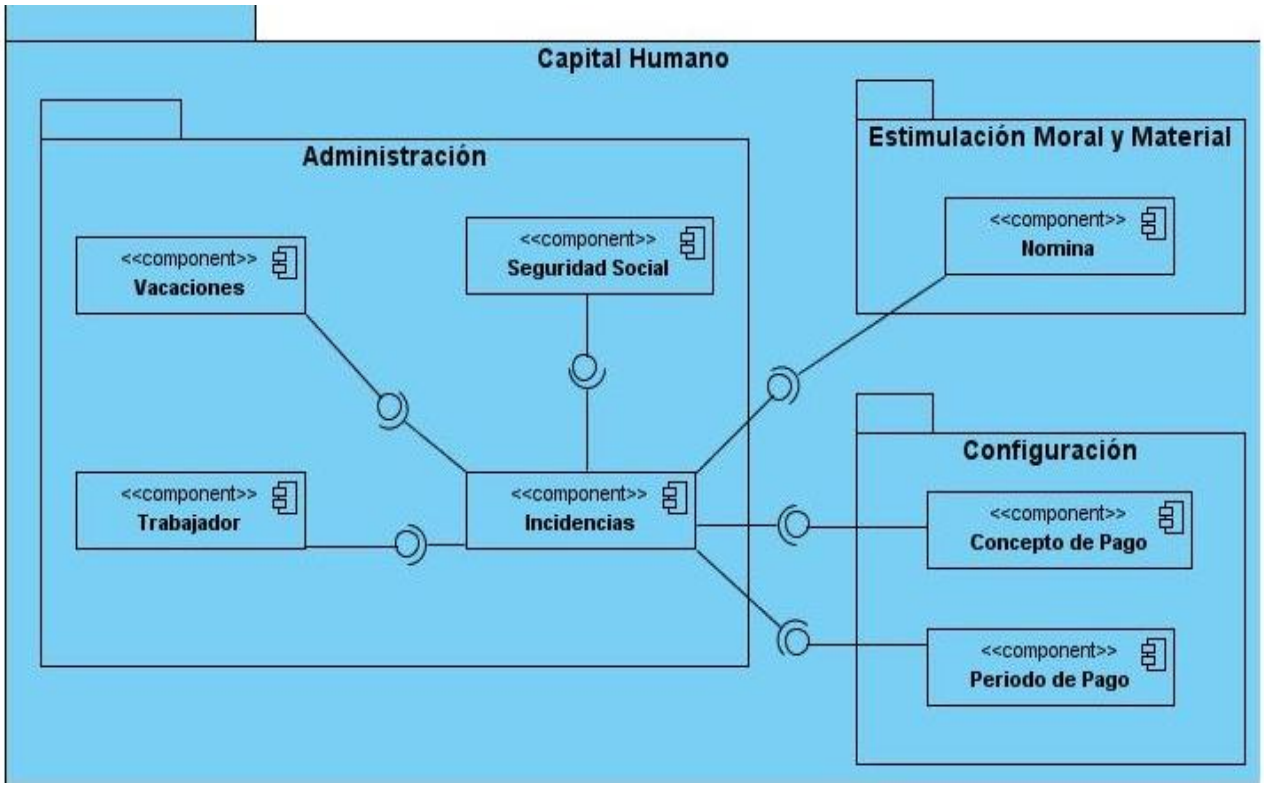


Figura 3: Diagrama de componentes.

La base para el desarrollo del componente gestión de incidencia, la constituye la arquitectura descrita en el capítulo 1. La misma provee componentes genéricos reutilizables que garantizan un conjunto de escenarios arquitectónicos. La figura 4 muestra la distribución de los componentes por niveles o capas, enfatizando la integración entre componentes.

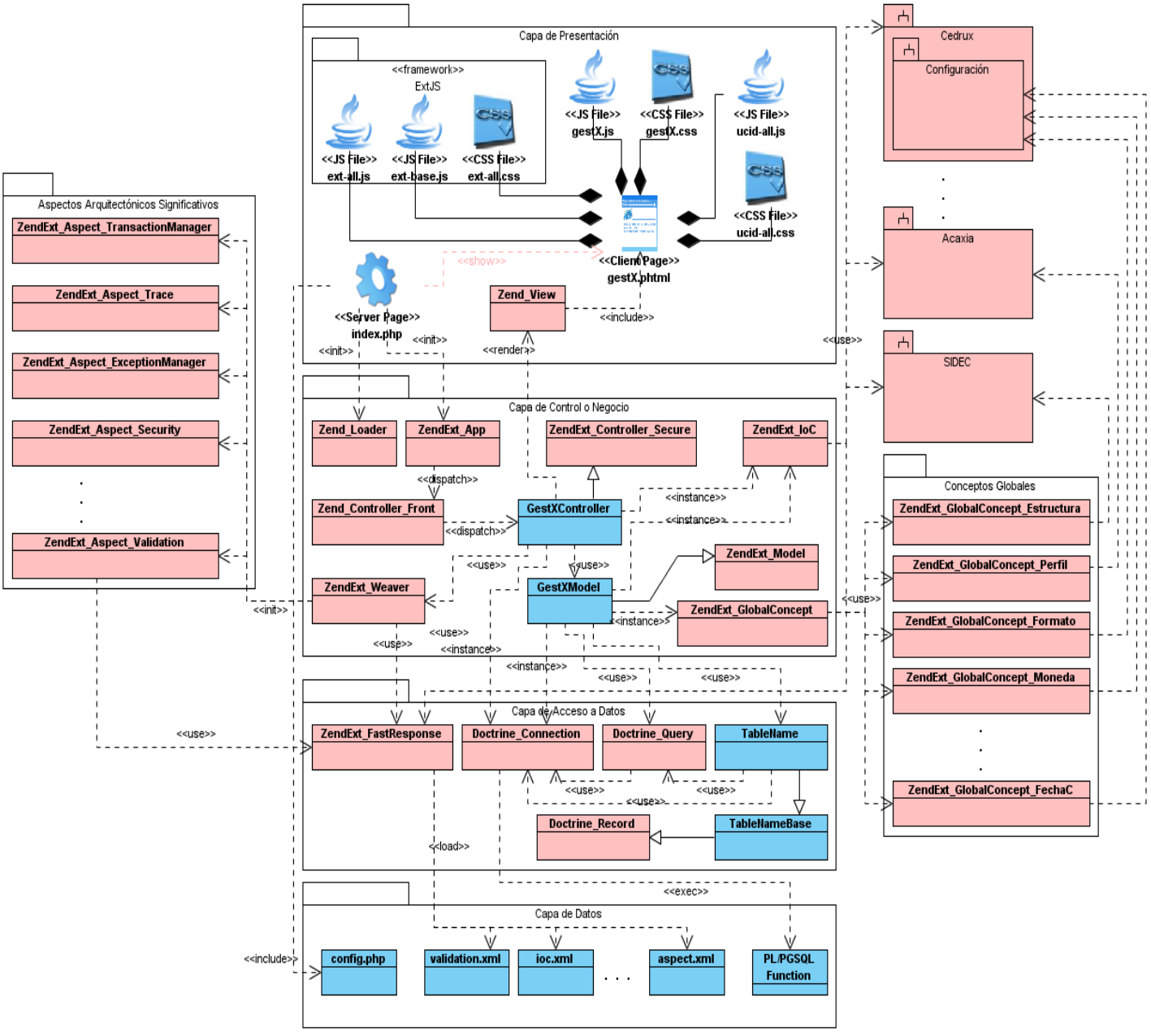


Figura 4: Taxonomía Arquitectónica

El uso del mecanismo de diseño permitió conformar la estructura arquitectónica del componente gestión de incidencia y la integración del mismo con el componente nómina del subsistema Capital Humano que está integrado al sistema Cedrux.

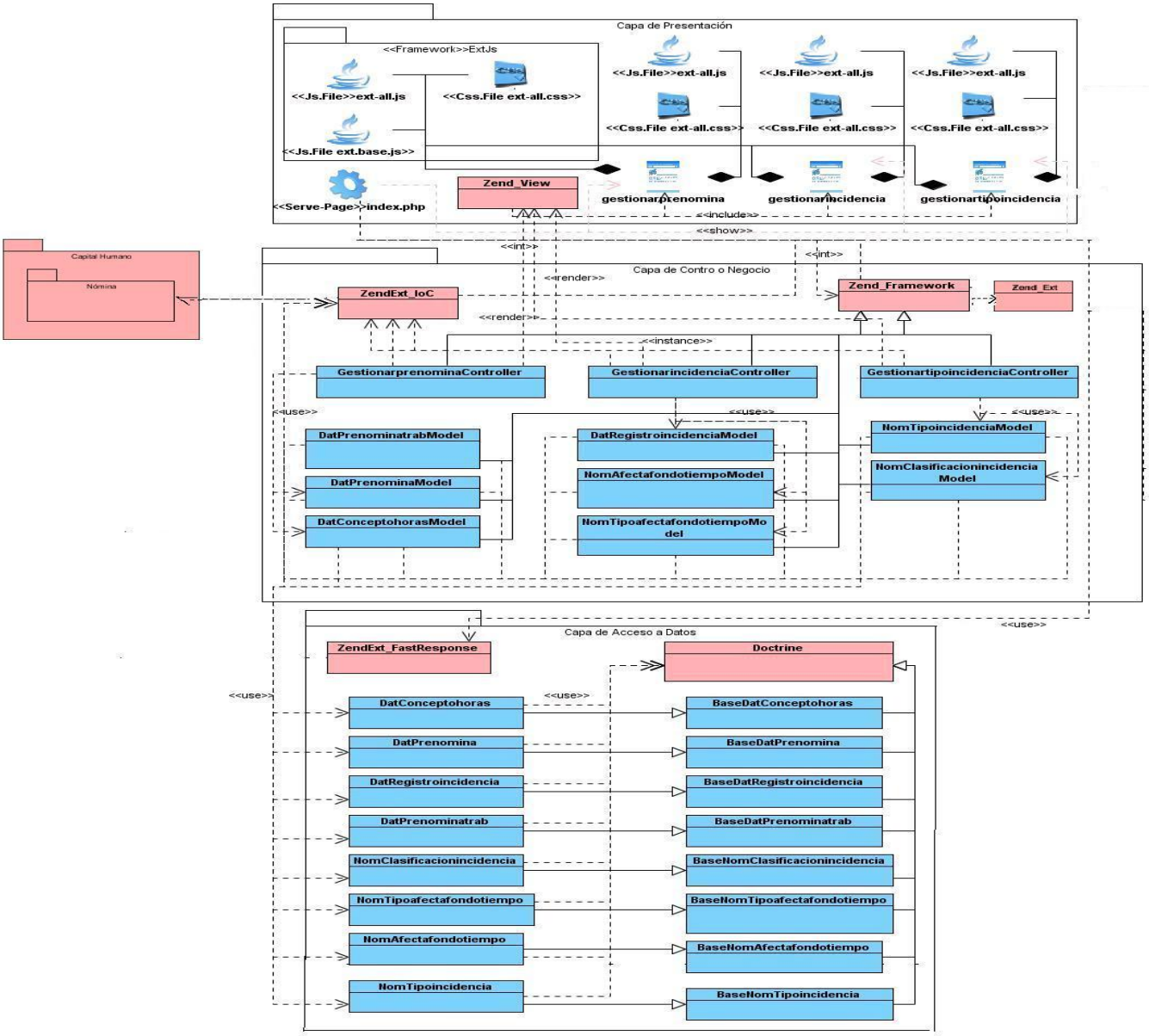


Figura 5: Diagrama de integración del componente Incidencia

2.6 Integración entre componentes

En cada uno de los componentes del sistema el flujo de datos que va desde la vista hacia el modelo y viceversa, responde completamente a una estrategia de integración vertical a partir de cada uno de los elementos arquitectónicos definidos.

✓ 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 la base de datos.

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

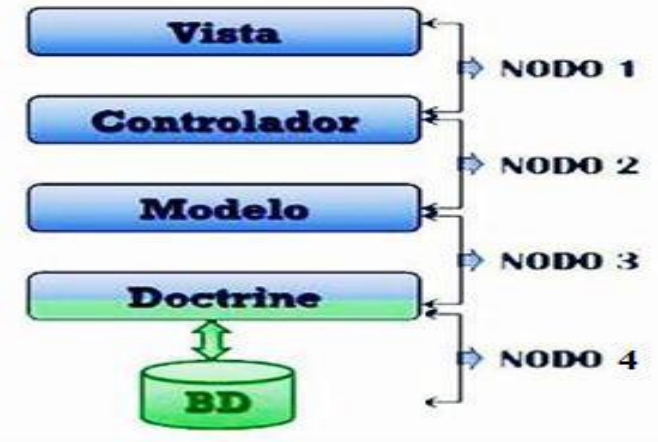


Figura 6: Nodos involucrados en la integración.

La comunicación dentro de un mismo componente se ejecuta de forma directa, sin embargo, la que se establece entre los diferentes componentes y subsistemas de la aplicación va más allá de un simple llamado a un servicio. Esta se basa en el funcionamiento del componente llamado: inversión de control (IoC). El IoC es donde se publican los servicios que brindan cada uno de los componentes del subsistema en cuanto a nombre de clases, funciones y tipo de resultado.

Tabla 2: Descripción de los servicios brindados

Componente	Servicios que brinda	Componentes que lo utilizan	Descripción
Incidencias	obtenerPrenominasParams($params)	Prenómina, Nómina.	Devuelve todas las prenómina que cumplan los requisitos especificados en el objeto params. En caso de params

			estar vacío devuelve todas las prenóminas.
	insertarRegistroIncidencia($incidencia)	Control de asistencia.	Inserta un registro de incidencia, permitiendo que se actualicen los registros desde otros componentes.
	buscarTrabPrenomina($params)	Prenómina, Nómina.	Devuelve todos los trabajadores que pertenezcan a una prenómina en específico.
	insertarIncidencia($incidencia)	Seguridad Social	Permite insertar un tipo de incidencia desde otros componentes.
	buscarIncidencia($iparams)	Prenómina, Nómina, Seguridad social, Control de asistencia.	Devuelve los tipos de incidencias registrados y que cumplan con el criterio de búsqueda establecido en params.
	actualizarIncidencia($idofdx)	Seguridad Social	Permite actualizar una incidencia desde otros componentes.
	ObtenerTipoafectafondotiempo()	Seguridad Social	Devuelve las diferentes formas en que una incidencia puede afectar el fondo de tiempo laboral del trabajador.
	ObtenerIncidenciasSubsidio($params, $limite, $inicio)	Seguridad Social	Devuelve las incidencias que sean de tipo subsidio.

Tabla 3: Descripción de los servicios usados

Componente	**Servicios que uso**	**Componentes que lo utilizan**	**Descripción**
Trabajador	BuscarTrabajador($params)	Incidencia Prenómina	Obtener trabajador/trabajadores dado un set de parámetros
Persona	BuscarPersonaPorParam($params)	Incidencia Prenómina	Obtener una persona dado un parámetro
Periodo de pago	PeriodosDePago()	Prenómina	Devuelve todos los

			periodos de pago

2.7 Modelo de datos

El modelo de datos permite identificar algunos detalles de implantación para el manejo del hardware de almacenamiento, proporciona referencias de cómo se almacenan los datos en la computadora, el formato de los registros, la estructura de los ficheros (ordenados, desordenados) y los métodos de acceso utilizados (índices). Los conceptos de este modelo están dirigidos fundamentalmente al personal informático, no a los usuarios finales. (26)

El modelo de datos propuesto en la solución cuenta con un total de 12 tablas. De ellas 4 son de datos y 8 son nomencladores. Para su construcción se tuvo en cuenta la reducción a la mínima expresión de los campos nulos y la persistencia de campos resúmenes para agilizar recuperaciones frecuentes de algunos datos que son complejos de calcular.

A continuación se muestra la descripción de las tablas del modelo de datos:

- ✓ Dat_registroincidencia: Es la encargada de guardar los registros de incidencia de cada trabajador, la misma se relaciona con las tablas, dat_trabajador, nom_tipoincidencia y dat_prenomina, es la tabla de mayor importancia del componente ya que es el centro del mismo.
- ✓ Dat_prenomina: En ella persisten los datos referentes a las prenóminas, la misma se relaciona con las tablas nom_periodopago, dat_registroincidencia y tiene una relación de muchos a muchos con la tabla dat_trabajador, formándose la tabla intermedia dat_prenominatrab.
- ✓ Dat_prenominatrab: Surge de una relación de muchos a muchos entre las tablas dat_prenomina y dat_trabajador, en ella se guardan los datos referentes a la asociación de un trabajador a una prenómina.
- ✓ Dat_nómina: En esta tabla persiste la información referente a las nóminas que se generan, se relaciona con la tabla dat_prenomina.
- ✓ Nom_tipoincidencia: Es la encargada de guardar los datos referentes a los tipos de incidencias, se relaciona con las tablas dat_registroincidencia, nom_tipovacaciones, nomtipoafectafondotiempo, nom_clasificacionincidencia y nom_tiposubsidio.
- ✓ Nom_tipoafectafondotiempo: Es un nomenclador estático que guarda las formas en que una incidencia puede afectar el fondo de tiempo de los trabajadores, la misma se relaciona con la tabla nom_tipoincidencia.
- ✓ Nom_clasificacionincidencia: Es un nomenclador estático que guarda las clasificaciones de una incidencia, la misma se relaciona con la tabla nom_tipoincidencia.
- ✓ Nom_tiposubsidio: En esta tabla persisten los tipos de subsidios registrados, la misma se relaciona

con la tabla nom_tipoincidencia.

✓ Nom_tipovacaciones: Es un nomenclador estático que guarda los tipos de vacaciones, la misma se relaciona con la tabla nom_tipoincidencia.

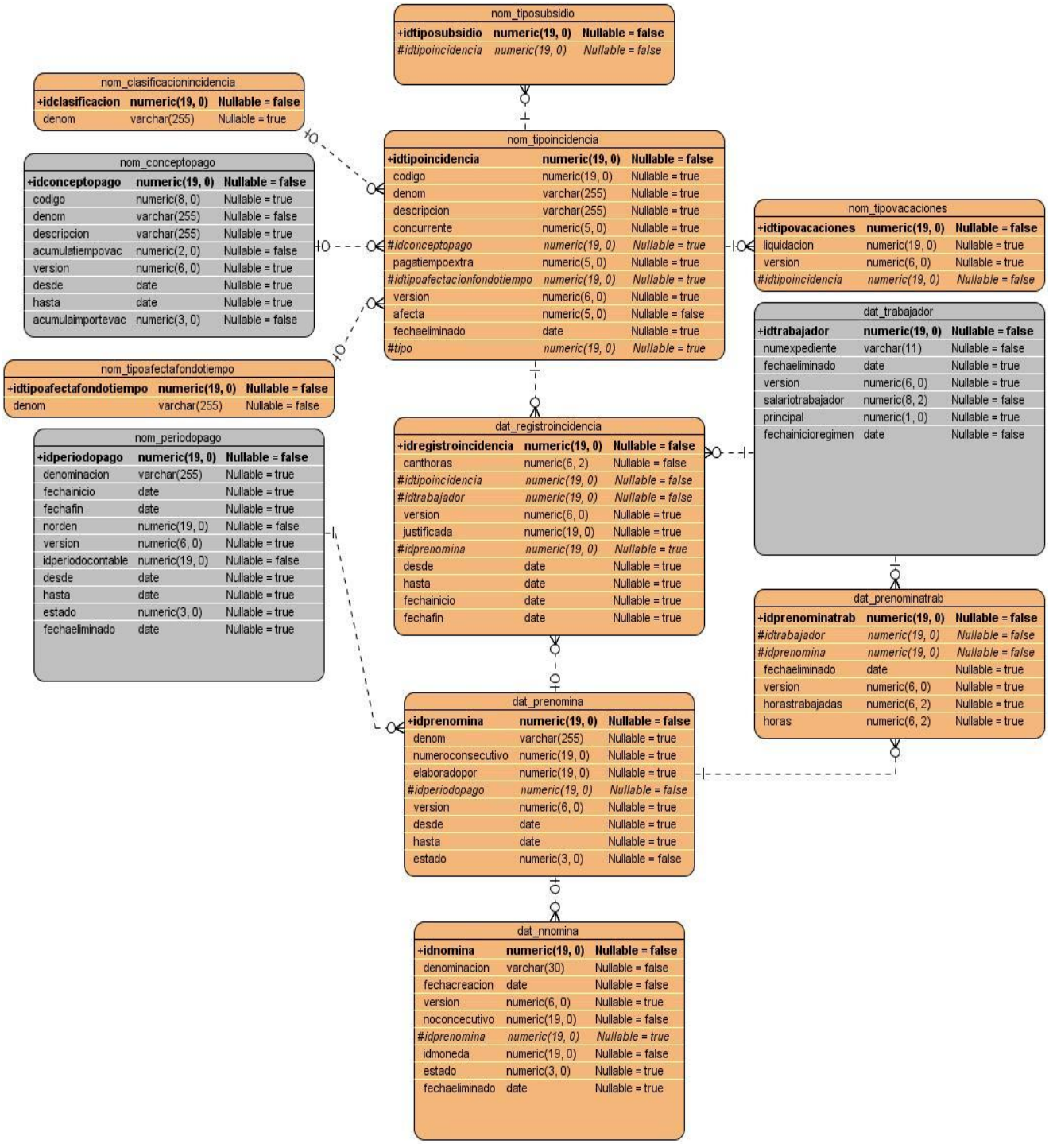


Figura 7: Modelo de Datos del componente Incidencias

2.8 Descripción de algoritmos no triviales a implementar

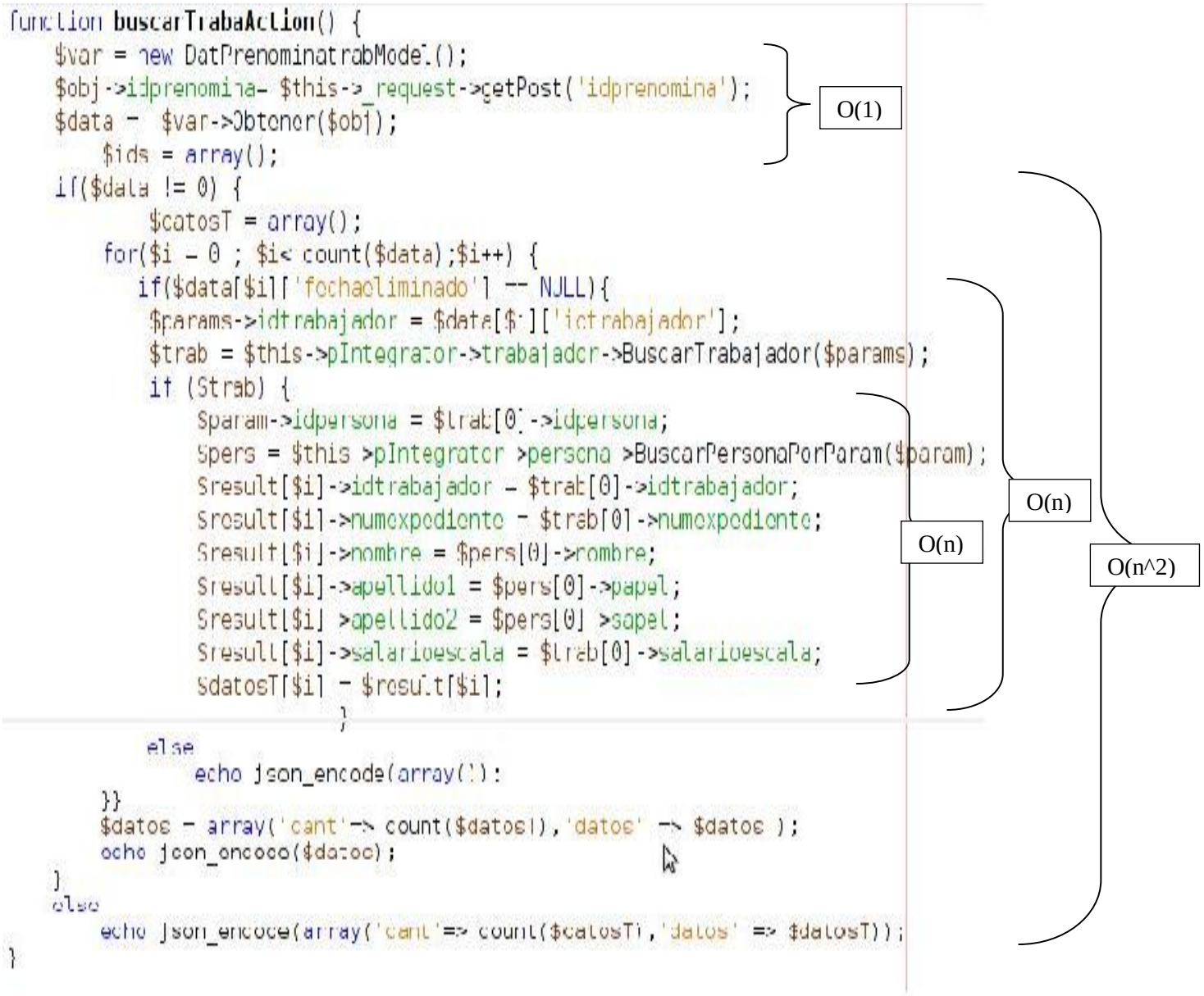


Figura 8: Algoritmo no Trivial a analisar.

2.8.1 Análisis de complejidad del mismo

Los elementos que intervienen en la eficiencia de un algoritmo tienen una naturaleza variada y están determinados fundamentalmente por el propio algoritmo, los datos de entrada y el medio de cómputo sobre el cual se ejecuta.

Órdenes de Complejidad: Se dice que O (f(n)) define un "orden de complejidad", de manera que se tiene:

O (1) orden constante

O (log n) orden logarítmico
O (n) orden lineal
O (n^2) orden cuadrático
O (c^n) orden exponencial

Reglas para determinar la complejidad temporal de un algoritmo.
Las reglas que a continuación presentamos son la base para realizar el cálculo del tiempo necesario para la ejecución de un algoritmo en base a las instrucciones, es decir, de las operaciones elementales que lo componen en el modelo de maquina presentado anteriormente.

Regla 1: El tiempo requerido para el acceso a un valor es un valor constante, así como el tiempo para realizar operaciones aritméticas elementales, como la adición, substracción, multiplicación, división y para las comparaciones es constante. El tiempo para almacenar el resultado en memoria también es constante, con un valor de O (1).

Regla 2: El tiempo requerido para la ejecución de un algoritmo de secuencia lineal es la sumatoria de los tiempos de cada instrucción.

Regla 3: El tiempo requerido para la ejecución de una instrucción condicional ***if*** C **then** I1 **else** I2, es la suma del tiempo necesario para evaluar la expresión más el tiempo máximo de ejecución de los bloques asociados a las ramas ***then*** y ***else***.

T = T(C) + Máximo (T (I1), T (I2))

Regla 3a: Se aplica la regla de la suma, de modo que se calcula el tiempo de ejecución tomando el máximo de los tiempos de ejecución de cada una de las partes, (sentencias individuales) en que puede dividirse.

Regla 4: El tiempo requerido para la ejecución de una instrucción repetitiva **while (C) do I**, es la suma del tiempo necesario para la evaluación de la expresión ***C*** + el número de iteraciones multiplicado por el tiempo para la instrucción sumado con el tiempo necesario para la evaluación de la expresión por la cantidad de iteraciones.

T = T(C) + iteraciones*T(C) + iteraciones*T (I)
Agrupando términos semejantes obtenemos: T = T(C) + (iteraciones)*(T(C)+T (I)).

Regla 4a: El tiempo requerido para la ejecución de una instrucción repetitiva **for** (I1, C, I2) I3 es equivalente al tiempo empleado en ejecutar la secuencia de instrucciones siguientes:

I1;

While(C) **do**
{
I3;
I2;
}

Así mismo el tiempo requerido para la ejecución repetitiva **do** I **while** (C) es equivalente al tiempo empleado en ejecutar el bloque de instrucciones siguiente:

I;
While(C) **do**
I;

En general de haber otras estructuras repetitivas se han de convertir estas en equivalentes

Regla 5: Bloque de sentencias. Se aplica la regla de la suma, de forma que se calcula el tiempo de ejecución tomando el máximo de los tiempos de ejecución de cada una de las partes.

Regla 6: Llamadas a funciones. Si una determinada función P tiene un tiempo de O (f(n)), con n a la medida del tamaño de los argumentos, cualquier función que llame a P tiene en la llamada una cota de O (f(n)). Las asignaciones con diversas llamadas a funciones deben de sumar las cotas del tiempo de ejecución de cada llamada.

En el algoritmo antes presentado el orden de complejidad es cuadrática: O (n^2) que comparada con los demás órdenes indica la eficiencia del mismo, partiendo de que los algoritmos que presentan complejidad mayor a O (n^2) se consideran intratables o desprovistos de solución.

2.9 Implementación de los componentes

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

reutilización.

En el componente a desarrollar, en la capa de acceso a datos se implementan la menor cantidad de consultas posibles, construyendo el filtro en dependencia de cada uno de los parámetros recibidos al invocar el método, teniendo en cuenta además que el tiempo de respuesta del servidor es mucho menor cada vez que se ejecuta una misma consulta. Por tanto el rendimiento de la aplicación es considerablemente más efectivo de esta forma que en lugar de implementar una consulta por cada posible parámetro de entrada. En la vista, al cargar las interfaces principales las peticiones al servidor se realizan una vez que se obtiene respuesta de la anterior, en lugar de realizarlas todas de una vez, lo que puede provocar sobrecarga en el servidor y generar errores y/o demora en las respuestas del controlador. Todos los mensajes se declaran en el Json, de esta forma del controlador se envían números o booleanos por cada petición realizada desde la vista. Este modo de implementación se realiza para que las aplicaciones sean más ligeras y rápidas.

2.10 Normas y Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Las normas y estándares de codificación en el marco del ERP permiten una mejor integración entre las líneas de producción, estableciendo un modelo de desarrollo que conlleva a lograr un código más legible y reutilizable, de tal forma que se pueda facilitar su mantenimiento a lo largo del tiempo. Entre las normas y estándares de codificación utilizados en la implementación de este producto se encuentran:

2.10.1 Estándares de Nomenclatura

En el estándar de nomenclatura se define como serán nombradas las clases atendiendo entre otras cosas al tipo de clases que sean. Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Con sólo leerlo se reconoce el propósito de la misma Las clases controladoras después del nombre llevan la palabra:"Controller" y las clases de los modelos que están dentro de Busines llevan la palabra "Model", las de los dominios reciben el nombre de las tablas de la base de datos y las que se encuentran dentro de los dominios bases comienzan con la palabra "Base" seguido del nombre de la tabla en la Base de Datos.

- ✓ Clases controladoras.

Ejemplo: Gestionarincidencia.controller

- ✓ Clases de los modelos

Ejemplo: NomTipoincidenciaModel

- ✓ Domain (Dominio)

Ejemplo: NomTipoincidencia

- ✓ Generated (Dominio bases)

Ejemplo: BaseNomTipoincidencia

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: obtenerTipoIncidencia

En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:"Action"
Ejemplo: gestionartipoincidenciaAction

2.10.2 Normas de comentario

Es una necesidad comentar todo lo que se haga dentro del desarrollo, es decir, establecer las pautas que conlleven a lograr un código más legible y reutilizable, de manera que se pueda facilitar su mantenimiento a lo largo del tiempo. Los comentarios deben ser lo bastante claros y precisos de forma tal que se entiende el propósito de lo que se ha desarrollado. Se pueden encontrar comentarios de código donde deje explicado lo más preciso posible la funcionalidad del mismo, como se muestra en la figura 9.

```
 /**
*FiltrarAction
*Obtiene un listado con los controles de asistencia entre un per?odo dado
*@return retorna arreglo con los controles de asistencia
*/
  function FiltrarAction(){
        $fecha1=$this->_request->getPost("fecha");
        $fecha2=$this->_request->getPost("fecha2");
        $objG = GestionarPresenciaModel::GetInstancia();
    $arrData=$objG->filtrar($fecha1,$fecha2);
        echo  '({"datos":' . json_encode($arrData) . '})';
  } //fin de la funci?n FiltrarAction
```

Figura 9: Ejemplo del uso de los comentarios

2.10.3 Estilo del Código

En la implementación cuando se vaya a escribir una sentencia en php la forma de utilizar las etiquetas del mismo es la siguiente:

<? Php
//código
¿>

La política de sangría a utilizar en la implementación es por tabulaciones. Las clases se comienzan a declarar pegado al margen izquierdo, después de poner el nombre de la clase se pone un espacio y se abre llave en la misma línea. Se establecieron pautas para utilización de sangrías dentro de las declaraciones dentro de métodos, bloques, al igual que para la apertura y cierre de llaves, el espaciado en blanco en las declaraciones, los controles, las Expresiones y Arreglos. Todo este estándar fue tomado del documento Normas y estándares de codificación de la línea de arquitectura del ERP el cual establece estos y otros conjuntos de normas por las cuales se rigió el desarrollo logrando una mejor organización y entendimiento de lo realizado.

2.10.4 Tratamiento de Errores.

El tratamiento de errores es de suma importancia para garantizar un correcto funcionamiento del sistema. Este último es el encargado de capturar todas las excepciones que son lanzadas y se facilita un tratamiento evitando que colapse. De esta forma se le da solución a las excepciones dentro del sistema según las peticiones del usuario, capturando así los tipos de errores lanzados y mostrándole al usuario mensajes de confirmación para saber con certeza lo que está incorrecto y como debe darle solución.
En el sistema existen varias funciones que necesitan validaciones previas para poder ejecutarse sin problemas, para dichas validaciones se utilizan ficheros de tipo XML, entre ellos el validator que contiene las llamadas a las precondiciones y poscondiciones a cumplirse. Por otro lado está el fichero exception que contiene los mensajes que se muestran según la excepción lanzada .El otro fichero XML es el ZendExt_Exception que captura excepciones previas definidas lo mismo para excepciones de la capa de lógica de negocio y la de acceso a datos. De esta manera se realizan las operaciones deseadas y se rectifican los errores.

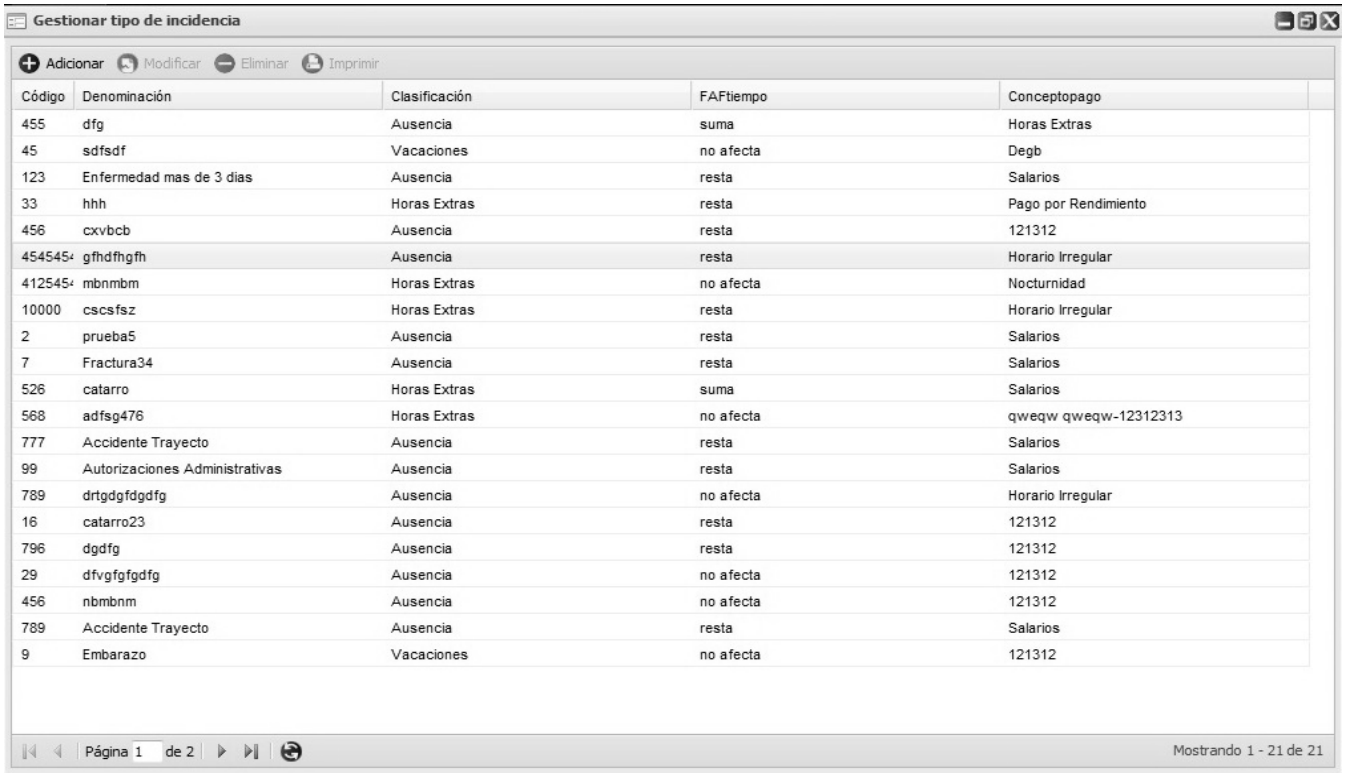
2.11 Prototipo de interfaz

De manera general el flujo de trabajo del componente gestión de incidencias comienza con el registro de los

tipos de incidencias en la empresa. Luego se le registran las incidencias a los trabajadores que hayan incurrido en las mismas y por último se pasa a la elaboracián de la prenómina, que es la entrada principal al componente de nómina. Este proceso se explica más detalladamente en la descripción de cada una de las interfaces que forman el componente.

Gestionar tipo de Incidencia

Esta interfaz es la encargada de gestionar las posibles definiciones de los tipos de incidencias que afectan al salario del trabajador, ya sea para sumarle o restarle tiempo al fondo de tiempo laboral. Cada tipo de incidencia tiene asociado un concepto de pago.



Código	Denominación	Clasificación	FAFtiempo	Conceptopago
455	dfg	Ausencia	suma	Horas Extras
45	sdfsdf	Vacaciones	no afecta	Degb
123	Enfermedad mas de 3 dias	Ausencia	resta	Salarios
33	hhh	Horas Extras	resta	Pago por Rendimiento
456	cxvbcb	Ausencia	resta	121312
454545	gfhdfhgfh	Ausencia	resta	Horario Irregular
412545	mbnmbm	Horas Extras	no afecta	Nocturnidad
10000	cscsfsz	Horas Extras	resta	Horario Irregular
2	prueba5	Ausencia	resta	Salarios
7	Fractura34	Ausencia	resta	Salarios
526	catarro	Horas Extras	suma	Salarios
568	adfsg476	Horas Extras	no afecta	qweqw qweqw-12312313
777	Accidente Trayecto	Ausencia	resta	Salarios
99	Autorizaciones Administrativas	Ausencia	resta	Salarios
789	drtgdgfdgdfg	Ausencia	no afecta	Horario Irregular
16	catarro23	Ausencia	resta	121312
796	dgdfg	Ausencia	resta	121312
29	dfvgfgfgdfg	Ausencia	no afecta	121312
456	nbmbnm	Ausencia	no afecta	121312
789	Accidente Trayecto	Ausencia	resta	Salarios
9	Embarazo	Vacaciones	no afecta	121312

Figura 10: Prototipo de interfaz Gestionar tipo de Incidencia.

Gestionar incidencia

Esta interfaz es la encargada de gestionar todas las incidencias que tiene un trabajador, la misma permite adicionar incidencias a un trabajador o justificar las mismas.

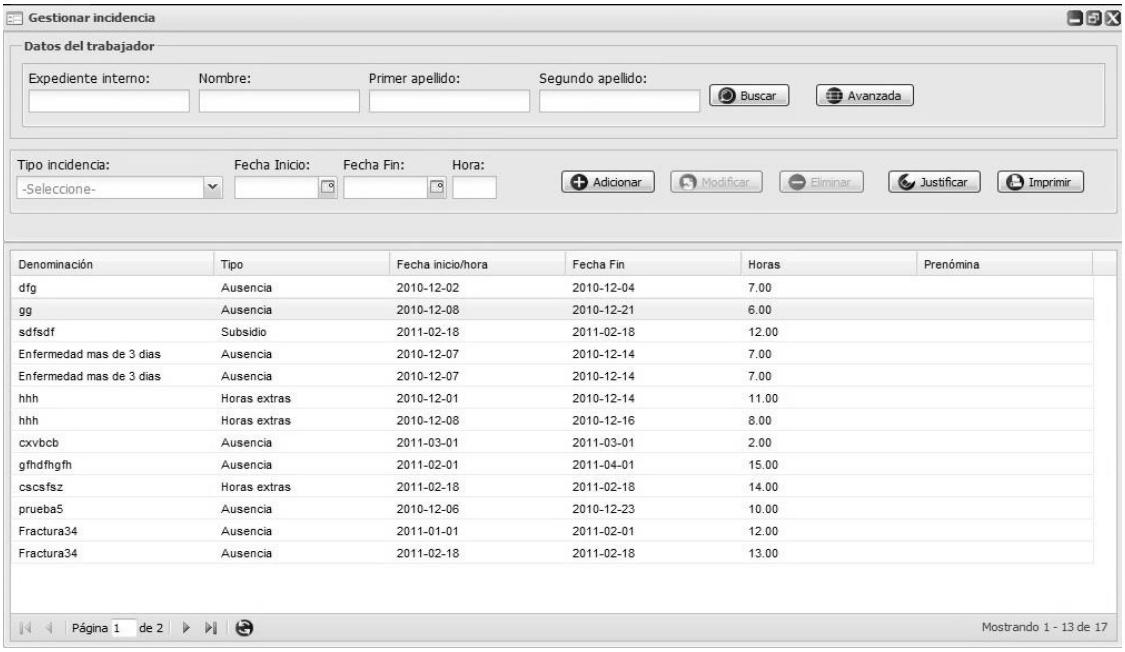


Figura 11: Prototipo de interfaz Gestionar Incidencia.

Gestionar prenómina

Esta interfaz es la encargada de gestionar los datos necesarios para elaborar y asignar trabajadores a una prenómina.

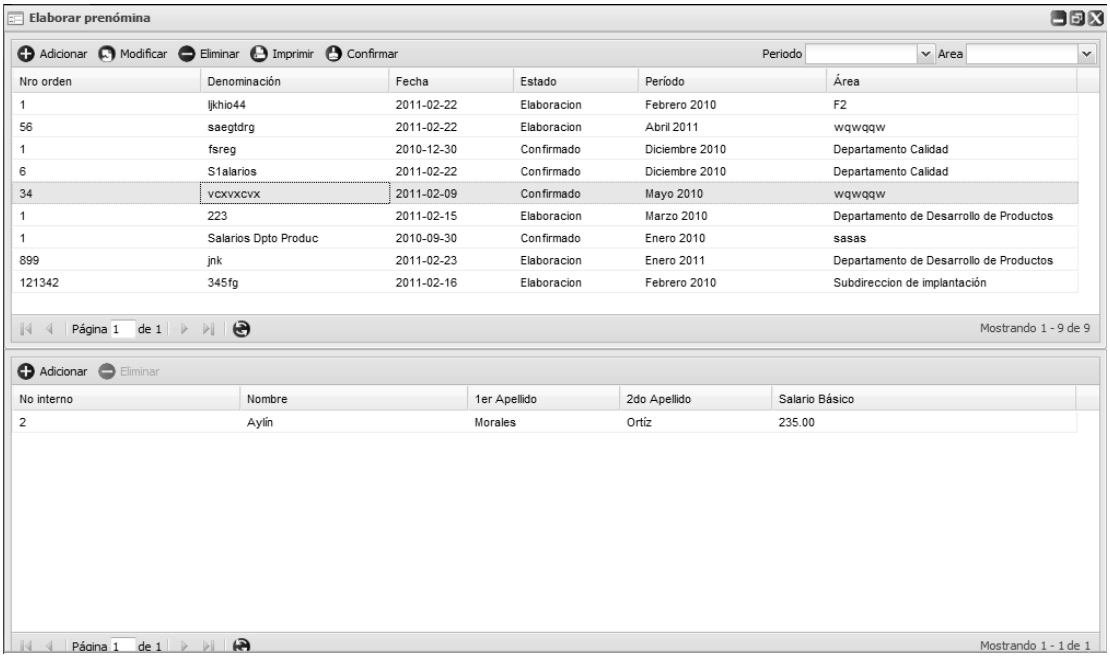


Figura 12: Prototipo de interfaz Elaborar prenómina

ANEXO 1: Se muestran el resto de las interfaces relacionadas con Gestionar Incidencias, Gestionar tipo de Incidencia y Elaborar prenómina.

2.12 Descripción de las principales clases a utilizar

En este tópico se describen las principales clases del componente gestión de incidencias así como sus métodos más significativos, los cuales contribuirán al cumplimiento satisfactorio de los requisitos propuestos por los analistas.

ANEXO 2: Descripción de las principales clases a implementar del componente gestión de incidencias.

2.13 Diagrama de despliegue

Los diagramas de despliegues describen la arquitectura física del sistema durante la ejecución en términos de procesadores, dispositivos y componentes de software.

Describen la topología del sistema: la estructura de los elementos de hardware y el software que ejecuta cada uno de ellos.

Un diagrama de despliegue muestra la configuración de nodos que participan en la ejecución y los componentes que residen en ellos. (27) (28)

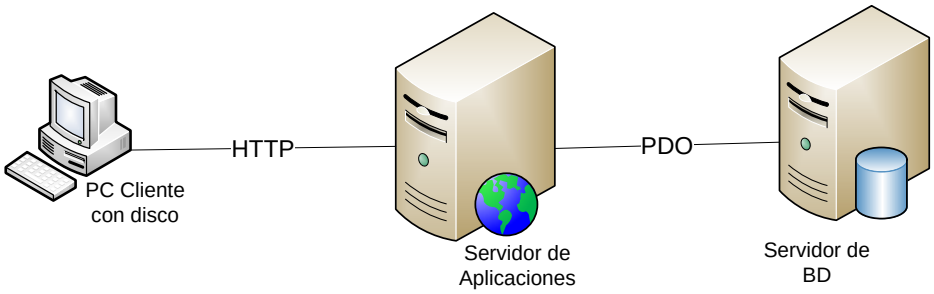


Figura 13: Diagrama de Red para PC Cliente con Disco Duro.

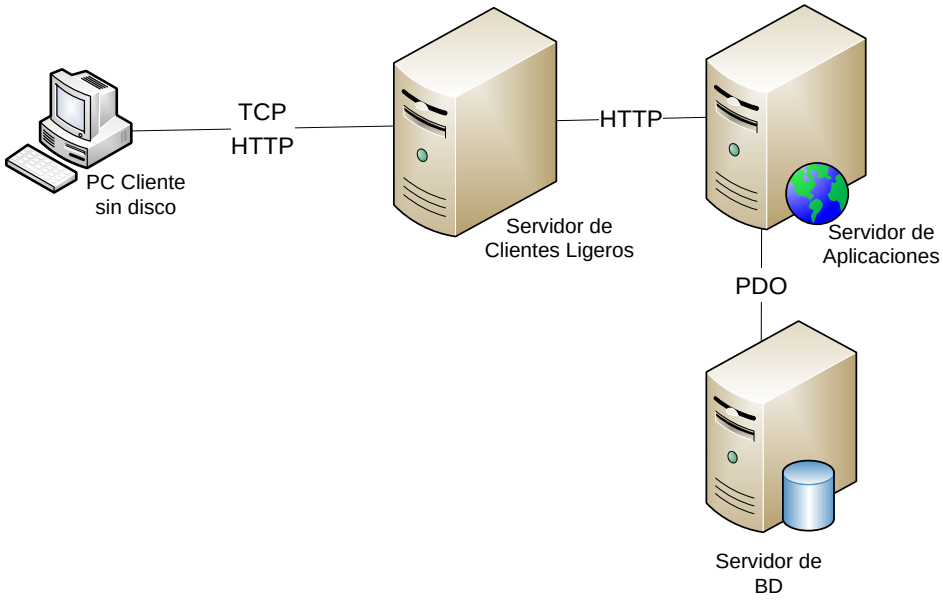


Figura 14: Diagrama de Red para PC Cliente sin Disco Duro.

2.14 Conclusiones parciales

En este capítulo se realizó un resumen de los principales escenarios del sistema y se describió la interacción resultante entre ellos. Se explicaron los componentes utilizados para el desarrollo de la aplicación, lográndose un mejor entendimiento de la solución propuesta al mostrar la integración entre los mismos. La descripción de las clases permitió tener una visión de la implementación realizada. Se analizó la complejidad algorítmica de las funciones más significativas y se mostraron los posibles escenarios en los cuales puede ser desplegado el sistema. De forma general se lograron implementar los requerimientos definidos por los analistas para los procesos de gestión de incidencias, pertenecientes al subsistema Capital Humano del Sistema Integral de Gestión Cedrux. Una vez concluido el desarrollo de la solución puede darse paso a los flujos de prueba de la misma.

.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

3.1 Introducción

El desarrollo del software implica una serie de actividades de producción, en las cuales 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. Debido a la imposibilidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software se acompaña de una actividad que garantice la calidad.

Para dar solución a estos problemas surge un nuevo proceso dentro del ciclo de desarrollo del software, el proceso de pruebas. Estas son actividades en las cuales un sistema o componente es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados y registrados, además se realiza una evaluación del sistema o componente. Las pruebas y validación de los resultados se ejecutan en cada una de las etapas de desarrollo. Las pruebas se realizan una y otra vez hasta que sean erradicados todos los errores que presenta el sistema y se pueda comprobar la eficiencia de las operaciones utilizadas para satisfacer las necesidades del cliente. (29).

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

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 de vida del software. 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 iníciales del sistema. (30)

Dentro de las actividades para obtener un software con la madurez necesaria están:

- ✓ Revisiones*:* consiste en que cada integrante del equipo de desarrollo revisa el producto que va generando.
- ✓ Inspecciones*: e*s el trabajo de revisar cada producto por parte de colegas.
- ✓ Validaciones*:* es el cliente quien revisa el producto para decir si cumple con sus necesidades.

Esta definición implica que se considera una prueba exitosa, si se demuestran deficiencias en el software.

Las fallas pueden ser en el código o en el modelado, en dependencia del tipo de pruebas que se le apliquen al software.

3.3 Objetivos de las Pruebas

El objetivo de las pruebas de un programa es detectar el 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 aplicar 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.

3.4 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.

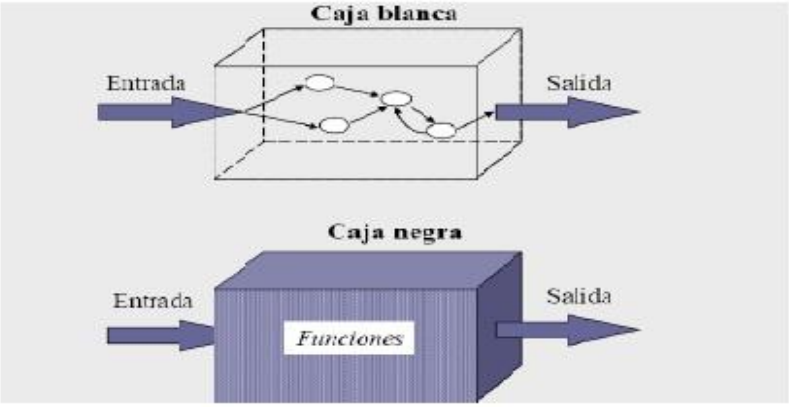


Figura 15: Representación de pruebas de Caja Blanca y Caja Negra

La Figura 15 representa gráficamente la filosofía de las pruebas de caja blanca y caja negra, como se puede observar para las pruebas de caja blanca se necesita conocer los detalles procedimentales del código.

Mientras que para las de caja negra únicamente se necesita saber el objetivo o funcionalidad que el código debe proporcionar.

3.4.1 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.

En resumen, mediante la prueba de la caja blanca se puede obtener casos de prueba que:

- ✓ Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- ✓ Ejerciten todas las decisiones lógicas en las vertientes verdaderas y falsas.
- ✓ Ejecuten todos los bucles en sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

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 de 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 de caminos de ejecución. Además también garantiza que durante la realización de 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.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones.

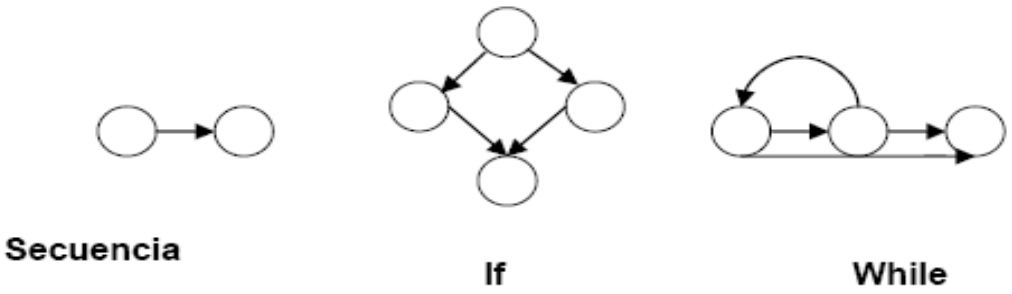


Figura 16: Notación de grafos de flujo para las instrucciones: Secuenciales, If, While

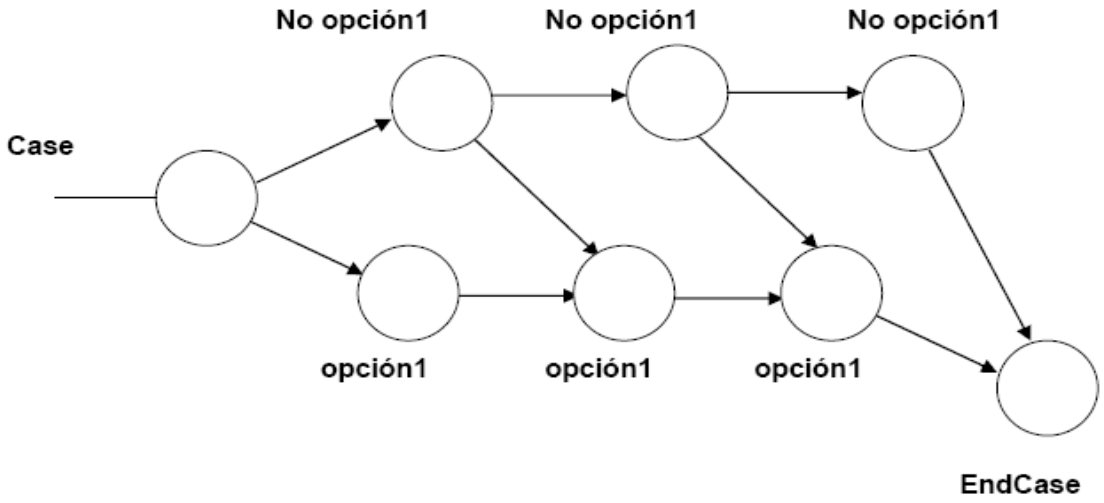


Figura 17: Notación de grafos de flujo para la instrucción Case

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, aún cuando el nodo no representa la sentencia de un procedimiento.

✓ **Regiones:** son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones 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 la complejidad ciclomática del algoritmo o fragmento de código a analizar, pues esta proporciona una medición cuantitativa de la complejidad lógica de un programa. A continuación se enumeran las sentencias de código del procedimiento realizado al método Insertar ($NomTipoincidencia), el cual tiene como función insertar una incidencia.

```
public function Insertar($NomTipoincidencia)
   { if($this->validateCode($NomTipoincidencia) ) 1
   {
      $NomTipoincidencia->idestructuracomun = $this->global->Estructura->idestructura; 2
            return $this->tipoincidencia->Insertar($NomTipoincidencia); 3
   }
      else
          return -1; 4
   }
```

Figura 18: Código a analizar Insertar ($Nomtipoincidencia).

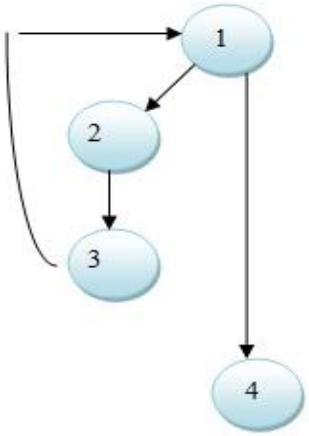


Figura 19: Grafo de flujo asociado al algoritmo Insertar ($NomTipoincidencia)

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

El valor calculado como complejidad ciclomática define el número de caminos independientes, del conjunto básico de un programa y define un límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecute cada sentencia al menos una vez. Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son, la cantidad total de aristas del grafo y la cantidad total de nodos, que se emplean en la siguiente fórmula:

V (G) = (Aristas – Nodos) + 2
V (G) = (4-4)+2
V (G) = 2
Se puede usar también:
V (G) = Nodos Predicados + 1
V (G) = 1+1
V (G) = 2
*Los nodos predicados son los nodos de los cuales parten dos o más aristas.
V (G) = Numero de Regiones
V (G) = 2
Para cada formula “V (G)” representa el valor del cálculo.

El resultado del cálculo de la complejidad ciclomática del código fue 2, lo que significa que existen dos

posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 4: Caminos independientes

Número	Camino básico
1	1-2-3-1
2	1-4

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

✓ Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o se entre algún dato erróneo.
✓ Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada, para ver el funcionamiento del procedimiento.
✓ Entrada: Se muestran los parámetros que entran al procedimiento.
✓ Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Tabla 5: Posible representación de casos de prueba para pruebas estructurales

Número de Camino	Descripción	Entrada	Resultados Esperados
1	El parámetro NomTipoincidencia no está vacío, contiene los datos de la incidencia que se conforma cuando se llenan todos los datos correspondientes al insertar un tipo de incidencia.	código=890 denom= trabajando en otro centro descripción= comisión de servicio idclasificación= ausencia idtipoafectafondotiempo= no afecta idconceptopago= Horario Irregular	Muestra un mensaje informando al usuario “Se ha adicionado la incidencia satisfactoriamente”

2	En el parámetro NomTipoincidencia que contiene los datos de la incidencia que se conforma cuando se llenan todos los datos correspondientes al insertar un tipo de incidencia, este contiene un código que ya existe.	código=890 denom= Accidente Trayecto descripción= Accidente producido en traslado hacia y desde el centro de trabajo idclasificación= ausencias idtipoafectafondotiempo=resta idconceptopago= 90000024	Muestra un mensaje informando al usuario ”Ya existe un tipo de incidencia con ese número de código”

Resultados de las pruebas de caja blanca

Para la realización de las pruebas de caja blanca al sistema fueron analizados un total 45 funcionalidades, cada clase tiene de 6 a 9 funcionalidades, lo que conllevó a obtener resultados positivos en el 80% las funciones, en el otro 20% los resultados no fueron los deseados, los servicios no cumplían con su funcionalidad y no devolvían los datos necesarios usados en otras funciones.

Luego de ser corregidos los errores encontrados en las pruebas, se pudo comprobar que el flujo de trabajo de las funciones estaba correcto, ya que cumplía con las condiciones necesarias que se habían planteado.

3.4.2 Pruebas de Caja Negra

Estas pruebas son llevadas a cabo sobre la interfaz del software, actuando sobre ella como una caja negra, proporcionando entradas y estudiando las salidas para ver si son o no las esperadas. Conociendo la función para la que fue diseñado, se hacen pruebas que demuestren que cada función es operativa y al mismo tiempo se buscan errores en cada una. Las llamadas pruebas de caja negra se basan en la especificación del programa o componente a ser probado, para elaborar los casos de prueba. También son conocidas como pruebas de comportamiento o pruebas inducidas por los datos.

- ✓ Permite obtener un conjunto de condiciones de entrada que ejecutan todos los requisitos funcionales de un programa.
- ✓ Las pruebas de caja negra no son alternativas a las técnicas de prueba de caja blanca. Es un enfoque complementario.

Las pruebas de caja negra intentan hallar errores tales como:

- ✓ Funciones incorrectas o ausentes.
- ✓ Error de interfaz.
- ✓ Errores en estructuras de datos o en accesos a base de datos.
- ✓ Errores de rendimiento.

Algunas técnicas utilizadas en la prueba 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.

A continuación se aplica la prueba de partición de equivalencia, como parte de la realización de la prueba de caja negra sobre la interfaz que responde al requisito funcional Gestionar tipo de Incidencia. La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Estas variables se definen en dos tipos, las válidas, que representan entradas válidas al programa y las no válidas, que representan valores de entrada erróneos, aúnque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

ANEXO 3: Caso de prueba de caja negra para validar el requisito funcional Gestionar tipo de Incidencia.

Resultados de las pruebas de caja negra

Para la realización de las pruebas de caja negra al sistema fueron analizados 21 requisitos. Se hizo una descripción de cada escenario de prueba. Se realizaron un total de tres iteraciones para poder alcanzar los

resultados satisfactorios desde el punto de vista interno y funcional del componente, atendiendo al correcto comportamiento del mismo ante diferentes situaciones (entradas válidas y no válidas).

✓ En la primera iteración se encontraron un total de 32 no conformidades (NC)

Elaborar prenómina 10 NC

Registro de incidencias 11 NC

Tipo de incidencia 11 NC

✓ En la segunda iteración se encontraron un total de 15 no conformidades (NC)

Elaborar prenómina 6 NC

Registro de incidencias 4 NC

Tipo de incidencia 5

✓ En la tercera iteración no se encontraron no conformidades (NC)

Elaborar prenómina 0 NC

Registro de incidencias 0 NC

Tipo de incidencia 0 NC

La siguiente gráfica es una representación de las no conformidades detectadas durante la ejecución de las pruebas de caja negra. En las mismas se encontraron mayormente errores de faltas de ortografías en las interfaces, de validaciones, los mensajes de información estaba mal elaborados y faltaban los iconos de los botones.

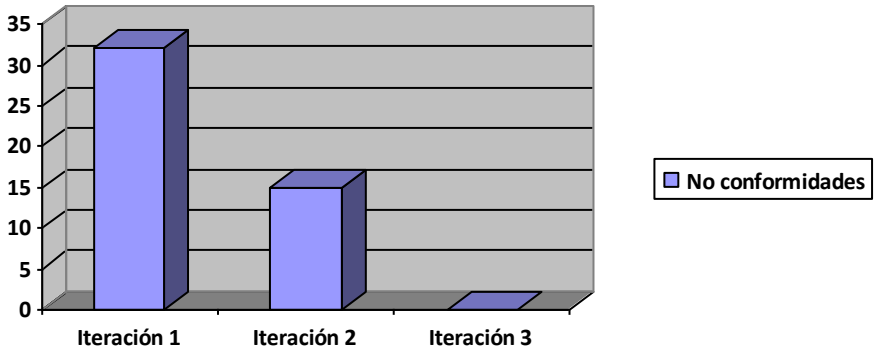


Figura 20: Resultados de las Iteraciones

Luego de ser corregidos los errores encontrados en las pruebas, se pudo comprobar que el flujo de trabajo de las interfaces estaba correcto, ya que cumplía con las condiciones necesarias que se habían planteado.

3.5 Conclusiones

Las pruebas realizadas al software arrojaron resultados de gran importancia, pues ayudaron a identificar algunos errores que habían pasado desapercibidos durante la etapa de implementación, además de validar el código fuente y servir como respaldo funcional al mismo. Se hicieron pruebas de caja blanca que permitieron conocer la complejidad ciclomática de cada funcionalidad. Mediante las pruebas de caja negra se probaron las interfaces que se utilizarán para la gestión de incidencias en una empresa. Estas pruebas se llevaron a cabo en tres iteraciones, corrigiendo los errores encontrados en cada una de ellas y asegurando una aplicación totalmente funcional que cumplía con los requisitos descritos.

CONCLUSIONES

A partir del análisis de los sistemas existentes vinculados a la gestión de incidencias, se demostró la necesidad de desarrollar un sistema capaz de llevar a cabo esta actividad. Para esto se realizó un estudio de las herramientas y tecnologías necesarias para implementar el componente que soluciona la problemática planteada, haciendo uso del software libre. Se realizaron pruebas al sistema que permitieron validar el código fuente y las interfaces pertenecientes al componente para la gestión de las incidencias de los trabajadores como parte de los procesos de gestión de Capital Humano.

En sentido general se obtuvieron resultados satisfactorios que dieron cumplimiento a los objetivos propuestos:

- ✓ Mediante el estudio de los procesos de negocio y los requisitos funcionales se realizó la descripción de un sistema que gestiona las incidencias de los trabajadores.
- ✓ Las pruebas estructurales y funcionales realizadas a la aplicación, validaron el correcto funcionamiento de la misma.
- ✓ Se logró implementar un sistema que permite gestionar las incidencias para su integración a la nómina del subsistema Capital Humano, del sistema Cedrux.

:RECOMENDACIONES

- ✓ Proseguir con la investigación e identificar nuevos requisitos para actualizar el sistema desarrollado.
- ✓ Que el componente realizado puedas ser utilizado en un futuro como base para la implementación de otros componentes relacionados con la gestión de incidencias.

REFERENCIAS BIBLIOGRAFICAS

1. **BUCOMSA.** One Time. [En línea] [Citado el: 12 de junio de 2011.] http://www.bucomsa.com.mx/html/ontime.htm.

2. ADP Expert. [En línea] [Citado el: 12 de junio de 2011.] http://www.adp-spain.com/adpexpert/index.htm.

3. ADP Expert. [En línea] [Citado el: 12 de junio de 2011.] http://www.es-adp.com/hasta-500/.

4. assets. [En línea] [Citado el: 13 de junio de 2011.] http://www.assets.co.cu/assets.asp.

5. assets ventajas. [En línea] [Citado el: 12 de junio de 2011.] http://www.assets.co.cu/ventajas.asp.

6. **ECU RED.** Versat Sarasola. [En línea] http://www.ecured.cu/index.php/Versat_Sarasol..

7. **ChaoVeitía, Marelys.** Sistema Económico Integrado VERSAT- Sarasola. [En línea] Casa Consultora DISAIC. http://www.disaic.cu/modules.php?name=Products&file=index&pag=prod&id=28. .

8. Versat_Sarasola. [En línea] http://www.ecured.cu/index.php/Versat_Sarasola.

9. RodasXX. [En línea] CITMATEL, 2002. [Citado el: 25 de marzo de 2011.] http://www.rodasxxi.cu/rodasxxi.php..

10. Humanos, Ha sido elaborada por el NC/CTN 110 Sistema de Gestión Integrada de los Recursos. *NC-3000.* 2007.

11. —. *3002.* 2007.

12.Gomez, O. TRABAJO ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE.

13. Pedro Boda, K.N., Javier Martinez, Benjamín Gonzales, Z*end Framework Manual en Español.* 2009.

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

15. Cross-Browser, Rich. Ext JS. [En línea] Internet Application Framework. http://www.sencha.com/products/extjs/..

16. *Taxonomía estructural de Sauxe.*

17. *Hieramientas Case*. (s.f.). Obtenido de http://www.inei.gob.pe/biblioineipub/bancopub/Inf/Lib5103/Libro.pdf

18. PostgreSQL. [En línea] . http://www.guia-ubuntu.org/index.php?title=PostgreSQL...

19. López, Alejandro Cadavid. Mozilla Firefox, el navegador web del momento. [En línea] http://www.maestrosdelweb.com/editorial/firefox/..

20. Una Introducción a APACHE. [En línea] http://linux.ciberaula.com/articulo/linux_apache_intro/..

21. Visual Paradigm para UML. [En línea] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p/.

22. Oracle Corporation. 2011. Bienvenido a NetBeans. [En línea] http://netbeans.org/index_es.html..

23. subversion. [En línea] [Citado el: 14 de junio de 2011.] http://www.osmosislatina.com/subversion/basico.htm.

24. Diagrama de componente. [En línea] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

25. Alva, Eduardo Rivera. [En línea] http://es.scribd.com/doc/7884665/Arquitectura-de-Software-II-Diagrama-de-Componentes-y-Despliegue.

26. modelo de datos. [En línea] http://definicion.de/modelo-de-datos/.

27. Diagrama de despliegue. [En línea] http://es.scribd.com/doc/19808824/diagramas-de-despliegue-2222.

28. diagrama de despliegue. [En línea] http://www.desarrolloweb.com.

29. Proceso de Prueba. [En línea] . http://www.buenastareas.com/ensayos/Proceso-De-Pruebas-De-Software/4887.html.

30. Pruebas de Software. [En línea] http://www.buenastareas.com/ensayos/Pruebas-De-Software/47445.html .

GLOSARIO DE TÉRMINOS

Capital Humano: Cuantificación y valoración de los recursos humanos. Valor de las habilidades, capacidades, experiencias y conocimientos de las personas que integran una organización.

Gestión: Es el conjunto de diligencias que se realizan para desarrollar un proceso o para lograr un producto determinado. Es también la dirección o administración de una empresa o de un negocio.

Entidad: Empresa, unidad presupuestada u otro tipo de organización similar con una gestión económica, financiera, organizativa, técnica, productiva, comercial, laboral y contractual, con autonomía controlada, en cumplimiento de lo establecido por el Gobierno.

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

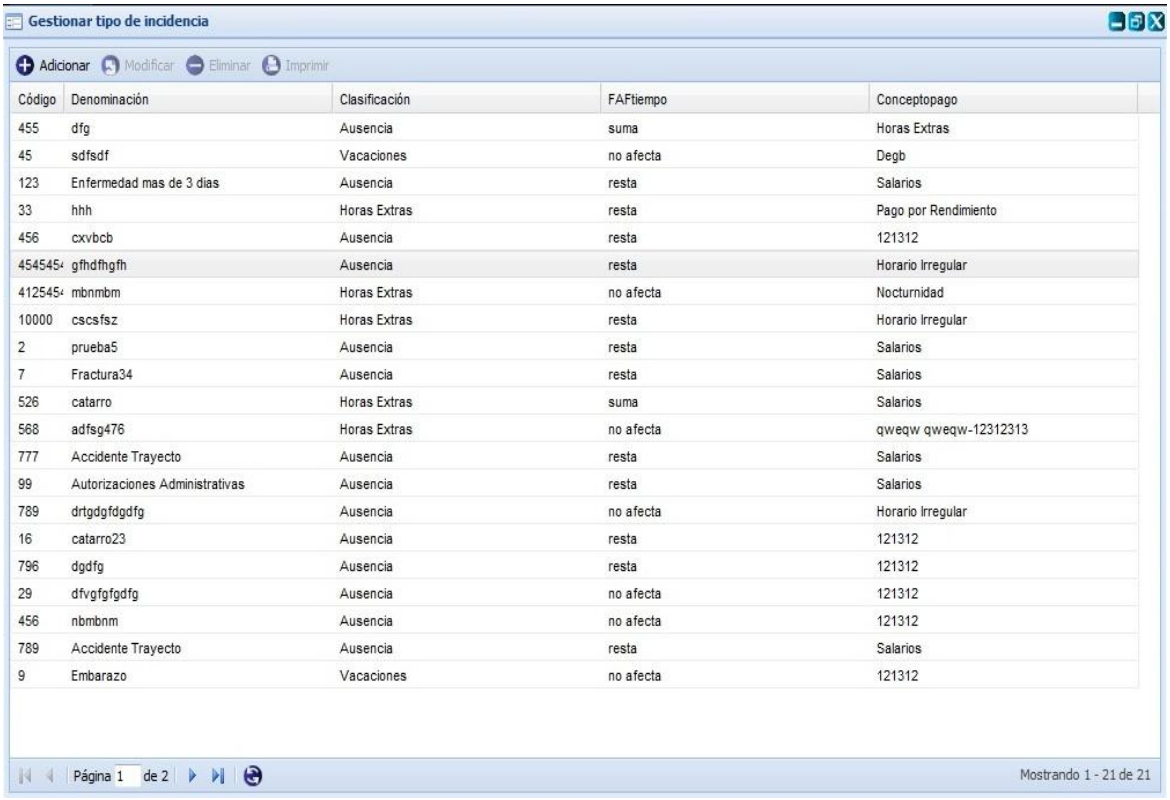
Proceso: Según lo establecido en el apartado 3.4.1 de la NC ISO 9000:2005, el proceso es un conjunto de actividades mutuamente relacionadas o que interactúan, las cuales transforman elementos de entrada en resultados.

Salario: Según lo establecido en la Norma Cubana 3000 del 2007 se considera salario la parte del producto nacional que se distribuye a los trabajadores de forma individual, atendiendo a la cantidad y calidad del trabajo aportado, según las condiciones económicas de cada momento histórico. Comprende lo percibido por el trabajador, por rendimiento, unidad de tiempo, pagos adicionales, trabajo extraordinario, laborar en día de conmemoración nacional y feriados y vacaciones anuales pagadas.

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

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

ANEXOS 1: Prototipos de la interfaz



Código	Denominación	Clasificación	FAFtiempo	Conceptopago
455	dfg	Ausencia	suma	Horas Extras
45	sdfsdf	Vacaciones	no afecta	Degb
123	Enfermedad mas de 3 dias	Ausencia	resta	Salarios
33	hhh	Horas Extras	resta	Pago por Rendimiento
456	cxvbcb	Ausencia	resta	121312
454545	gfhdfhgfh	Ausencia	resta	Horario Irregular
412545	mbnmbm	Horas Extras	no afecta	Nocturnidad
10000	cscsfsz	Horas Extras	resta	Horario Irregular
2	prueba5	Ausencia	resta	Salarios
7	Fractura34	Ausencia	resta	Salarios
526	catarro	Horas Extras	suma	Salarios
568	adfsg476	Horas Extras	no afecta	qweqw qweqw-12312313
777	Accidente Trayecto	Ausencia	resta	Salarios
99	Autorizaciones Administrativas	Ausencia	resta	Salarios
789	drtgdgfdgdfg	Ausencia	no afecta	Horario Irregular
16	catarro23	Ausencia	resta	121312
796	dgdfg	Ausencia	resta	121312
29	dfvgfgfgdfg	Ausencia	no afecta	121312
456	nbmbnm	Ausencia	no afecta	121312
789	Accidente Trayecto	Ausencia	resta	Salarios
9	Embarazo	Vacaciones	no afecta	121312

Figura 21: Prototipo de interfaz Gestionar Tipo de Incidencia

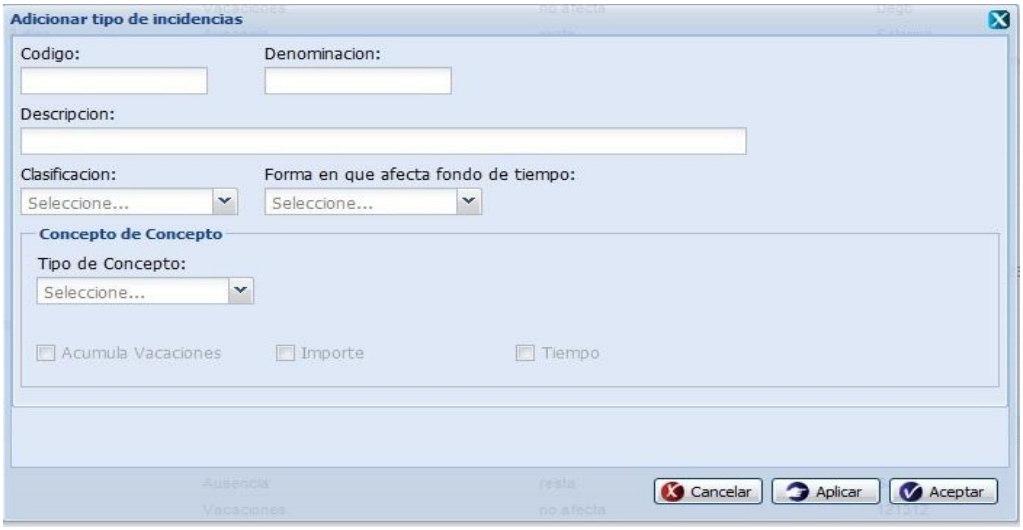


Figura 22: Prototipo de interfaz Adicionar Tipo de Incidencia

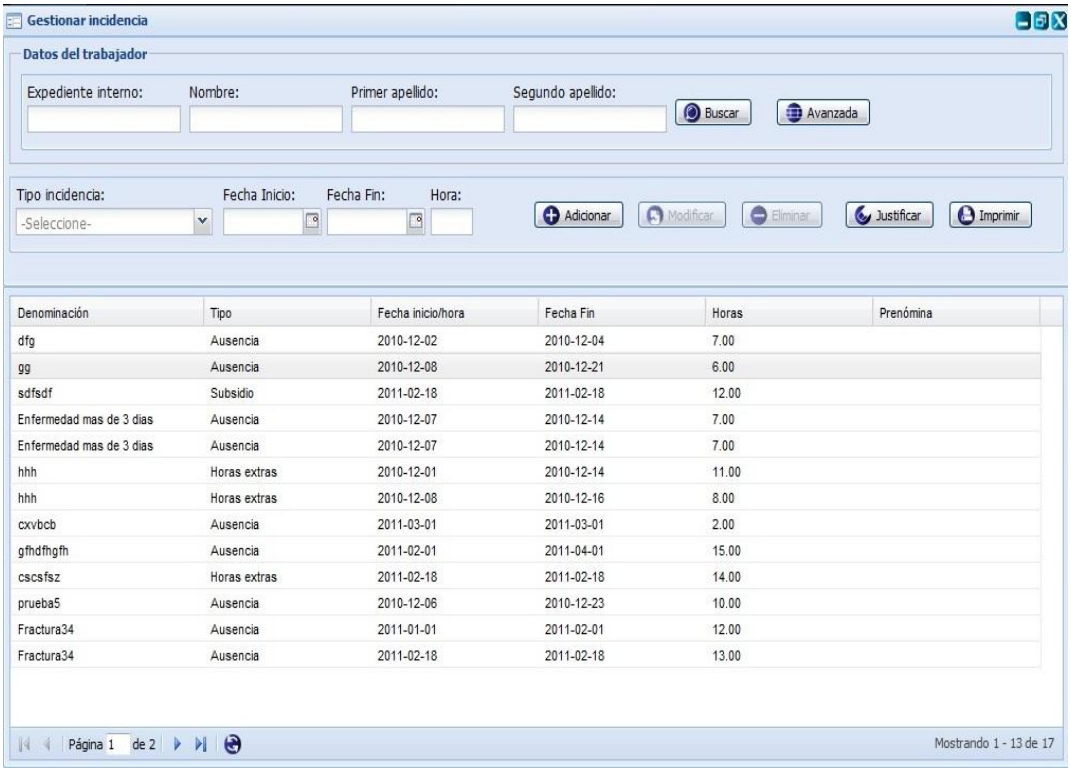


Figura 23: Prototipo de interfaz Gestionar Incidencia.

Figura 24: Prototipo de interfaz Justificar Incidencia.

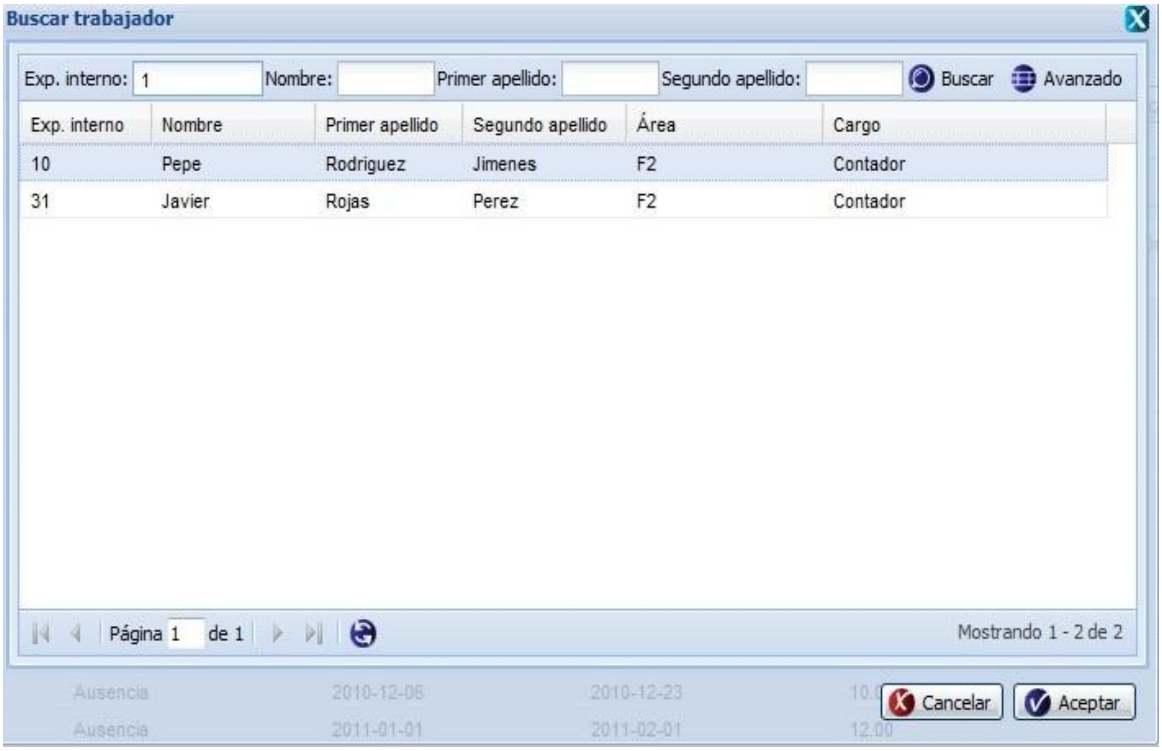


Figura 25: Prototipo de interfaz Buscar Trabajador de la interfaz Gestionar Incidencia.

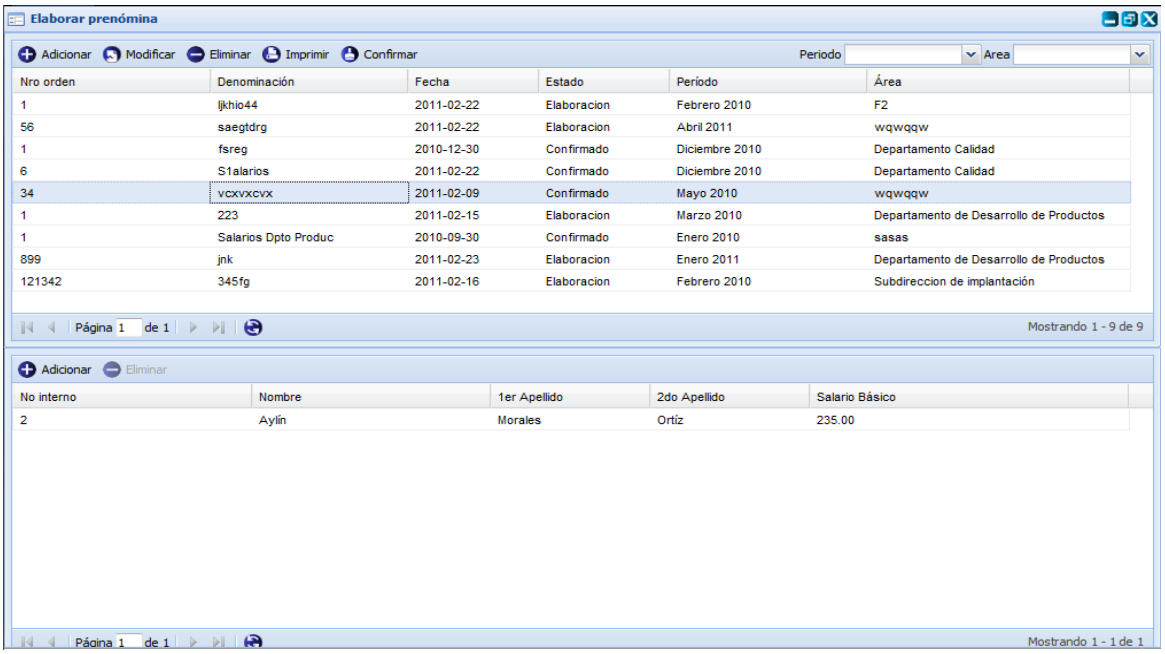


Figura 26: Prototipo de interfaz Elaborar Pre-nomina

Figura 27: Prototipo de interfaz Adicionar Pre-nomina

Figura 28: Prototipo de interfaz Modificar Pre-nomina

ANEXOS 2: Descripción de las clases y sus funcionalidades

Tabla 5: Descripción de la clase Gestionar Incidencia Controller

Nombre: gestionarincidencia.controller	
Tipo de clase: Controladora	
Para cada responsabilidad	
Nombre.	**Descripción.**
init()	Constructor de la clase.
gestionarincidenciaAction()	Encargada de validar todos los datos necesarios y redireccionar a la interfaz.
buscarAction() buscartrabajadorAction()	Encargada de buscar determinados datos de un trabajador.
obtenerTipoIncidenciaAction(){	Encargada de obtener los tipos de incidencias y mostrarla al usuario en los combox para su posterior uso.
adicionarIncidenciaAction() obtenerRegistroIncidenciaAction() modificarIncidenciaAction() eliminarincidenciaAction()	Encargadas de gestionar las incidencias.
cargarFormularioincidenciaAction() cargarFormularioTrabAction()	Encargada gestionar todos los datos de un trabajador determinado y de cargar todos los datos de ese trabajador referente a una determinada incidencia y mostrarlos al usuario para su posterior gestión.

Tabla 6: Descripción de la clase Tipo Incidencia Controller

Nombre: gestionartipoincidencia.controller	
Tipo de clase: controladora	
Para cada responsabilidad	
Nombre.	**Descripción.**
Int()	Constructor de la clase.
gestionartipoincidenciaAction()	Se encarga de la gestión de las incidencias y

	de mostrársela al usuario.
getclasificacionincidenciaAction()	Obtiene todas las clasificaciones de las incidencias.
getFormaAfectaFTAction()	Obtiene todas las formas en que se afecta el fondo del tiempo.
gettipoconceptoAction()	Obtiene todo los tipos de concepto de pagos.
obtenerTipoIncidenciaAction()	Responsabilidad de obtener los tipos de incidencias.
cargarFormularioAction()	Encargada de cargar el formulario para modificar un tipo de incidencia.
adicionarincidenciaAction() eliminarincidenciaAction() modificarincidenciaAction()	Responsabilidad de gestionar los datos de los tipos de incidencias.

Tabla 7: Descripción de la clase Gestionar Prenómina Controller

Nombre: gestionarprenomina.controller	
Tipo de clase: controladora	
Para cada responsabilidad	
Nombre.	**Descripción.**
Int()	Constructor de la clase.
obtenerPrenominaAction()	Encargada de obtener una prenómina.
buscarTrabaAction()	Encargada de buscar un trabajador determinado.
adicionarPrenominaAction() modificarPrenominaAction() eliminarPrenominaAction()	Responsabilidad de gestionar los datos de las prenómina.
cargarFormularioAction()	Se encarga de cargar el formulario para trabajar con una prenómina.
adicionarPrenominaTrabAction() eliminarPrenominaTrabAction() modificarPrenominaTrabAction()	Responsabilidad de gestionar los datos de las prenómina de un determinado trabajador.

Tabla 8: Descripción de la clase Tipo de Incidencia Model

Nombre: NomTipoincidenciaModel	
Tipo de clase: modelo	
Para cada responsabilidad	
Nombre.	**Descripción.**
DatRegistroincidenciaModel()	Constructor de la clase.
Insertar($DatRegistroincidencia) Actualizar($DatRegistroincidencia) Eliminar($obj) obtenerRegistroIncidencia($obj=null)	Encargada de gestionar las incidencias.
obtenerTipoIncidencia($params=null)	Responsabilidad de obtener las incidencias.

Tabla 9: Descripción de la clase DatPrenomina Model

Nombre: DatPrenominaModel	
Tipo de clase: modelo	
Para cada responsabilidad	
Nombre.	**Descripción.**
DatPrenominaModel()	Constructor de la clase.
Actualizar(DatPrenomina $DatPrenomina) Modificar($DatPrenomina)	Encargada de gestionar las prenóminas.
adicionarPrenomina($obj) obtenerPrenominaTodo() eliminarPrenomina($obj)	Encargada de gestionar las prenóminas.
obtenerPrenominaPorPeriodo($idperiodopago)	Encargada obtener una prenómina en un periodo especifico.

Tabla 10: Descripción de la clase DatRegistroincidencia Model

Nombre: DatRegistroincidenciaModel
Tipo de clase: **modelo**
Para cada responsabilidad

Nombre.	Descripción.
DatRegistroincidenciaModel()	Constructor de la clase.
Insertar($DatRegistroincidencia) Actualizar($DatRegistroincidencia) Eliminar($obj) obtenerRegistroIncidencia($obj=null)	Encargada de gestionar las incidencias.
obtenerTipoIncidencia($params=null)	Responsabilidad de obtener los tipos de incidencias.

Tabla 11: Descripción de la clase Tipo de Incidencia Domain

Nombre: NomTipoincidencia	
Tipo de clase: acceso a datos(domain)	
Para cada responsabilidad	
Nombre.	**Descripción.**
Buscar($idtipoincidencia)	Encargada de buscar en la base de datos los tipo de incidencias.
Insertar($temp) Actualizar($temp) Eliminar($idtipoconcepto)	Encargada de gestionar los tipos de incidencias en la base de datos.
obtenerTipoIncidencia($params, $start = 0, $limit = 20)	Devuelve los tipos de incidencias que cumplan con los requisitos especificados en el parámetro params.
obtenerTipoIncidenciaVacaciones($params , $limit = 20, $start = 0)	Devuelve los tipos de incidencias que sean del tipo vacaciones.
obtenerTipoIncidenciaSubsidio($params, $limit = 20, $start = 0)	Devuelve los tipos de incidencias que sean del tipo subsidio.

Tabla 12: Descripción de la clase DatPrenomina Domain

Nombre: DatPrenomina	
Tipo de clase: acceso a datos(domain)	
Para cada responsabilidad	
Nombre.	**Descripción.**

Buscar($idprenomina)	Encargada de buscar en la base de datos las prenómina.
obtenerPrenomina($paramet , $limite = 20, $start = 0) adicionarPrenomina($obj) eliminarPrenomina($idprenomina) Modificar($temp)	Encargada de gestionar las prenóminas en la base de datos.
PrenominaPorIdPeriodo($idperiodopago)	Devuelve las prenóminas que pertenezcan a un período de pago específico.

Tabla 13: Descripción de la clase Incidencia Domain

Nombre: DatRegistroincidencia	
Tipo de clase: acceso a datos(domain)	
Para cada responsabilidad	
Nombre.	**Descripción.**
Insertar($obj)	Encargada de insertar una incidencia en la base de datos.
Modificar($obj)	Encargada de modificar una incidencia en la base de datos.
Buscar($idregistroincidencia)	Encargada de buscar una incidencia dado un id.
eliminarIncidencia($idincidencia)	Encargada de eliminar una incidencia dado un id.
registoIncidencia($paramet,$limite = 13,$inicio = 0)	Encargada de registrar una incidencia a un trabajador.

ANEXO 3: Caso de prueba de Caja Negra para validar el requisito funcional Gestionar Tipo de Incidencia.

Tabla 14: Escenarios de prueba.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
1: Adicionar tipos de incidencias.	El sistema debe permitir adicionar los tipos de incidencias.	EP 1.1: Adicionar tipos de incidencias introduciendo datos válidos.	Se introducen los datos del tipo de incidencia correctamente. Se presiona el botón **Aceptar**. Se muestra un mensaje de información. Se presiona el botón **Aceptar**. Se cierra la interfaz.
		EP 1.2: Adicionar tipos de incidencias introduciendo datos válidos presionando el botón **Aplicar**.	Se introducen los datos del tipo de incidencia correctamente. Se presiona el botón **Aplicar**. Se muestra un mensaje de información. Se presiona el botón **Aceptar**. La interfaz permanece abierta.
		EP 1.3: Adicionar tipos de incidencias introduciendo datos inválidos.	Se introducen los datos inválidos del tipo de incidencia. Se presiona el botón **Aceptar**. Se muestra un mensaje informando del error.
		EP 1.4: Adicionar tipos de incidencias dejando campos vacíos.	Se introducen los datos dejando algún campo en blanco. Se presiona el botón **Aceptar**. Se muestra un mensaje informando del error.
		EP 1.5: Cancelar.	Se introducen o no los datos del tipo de incidencia. Se presiona el botón **Cancelar**.

Tabla 15: Descripción de Variables para el caso de prueba

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Descripción
1	Código	Campo de texto.	Números.	Letras y caracteres especiales.	Vacío.	El código aparece vacío.
2	Denominación	Campo de texto.	Letras y números.	Caracteres especiales.	Vacío.	La denominación aparece vacía.
3	Descripción	Campo de texto.	Letras y números.	Caracteres especiales.	Vacío.	La descripción aparece vacía.
4	Clasificación	Campo de selección (No editable).	Letras.	Números, caracteres especiales.	Vacío.	La clasificación aparece vacía.
5	Forma en que afecta fondo de tiempo	Campo de selección (No editable).	Letras.	Números, caracteres especiales.	Vacío.	Se selecciona una forma en que afecta fondo de tiempo.
6	Tipo de concepto	Campo de selección (No editable).	Letras y números.	Caracteres especiales.	Vacío.	Se selecciona un tipo de concepto de pago.

Tabla 16: Juegos de Datos a probar

Id del escenario	Escenario	Código	Denominación	Descripción	Clasificación	Incide salario	Tipo de concepto	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar tipos de incidencias introduciendo datos válidos. Presionando el botón **Aceptar**.	V (1).	V(Horas extras)	V(Horas trabajadas fuera de horario laboral)	V(Horas extras)	V(suma)	V(Salario)	El sistema adiciona el tipo de incidencia y muestra el mensaje de información: "Se ha adicionado la incidencia satisfactoriamente.". El sistema cierra la interfaz.	El sistema muestra el mensaje de información: "Se ha adicionado la incidencia satisfactoriamente.".
EP 1.2	Adicionar tipos de incidencias introduciendo datos válidos presionando el botón **Aplicar**.	V (1).	V(Horas extras)	V(Horas trabajadas fuera de horario laboral)	V(Horas extras)	V(suma)	V(Salario)	El sistema adiciona el tipo de incidencia y muestra el mensaje de información: "Se ha adicionado la incidencia satisfactoriamente.". El sistema mantiene la interfaz abierta.	El sistema muestra el mensaje de información: "Se ha adicionado la incidencia satisfactoriamente.".
EP 1.3	Adicionar tipos de incidenc	I (asd)	V(Horas extras)	V(Horas trabajadas fuera de	V(Horas extras)	V(suma)	V(Salario)	El sistema no permite la inserción de letras en este	El sistema subraya el campo en rojo mostrando el mensaje:

	ias introduciendo datos inválidos.			horario laboral)				campo. El sistema subraya el campo en rojo mostrando el mensaje: "Faltan campos por llenar.". El sistema mantiene la interfaz abierta.	"Faltan campos por llenar.".
		I(%&^)	I(%&^)	V(Horas trabajadas fuera de horario laboral)	V(Horas extras)	V(suma)	V(Salario)		El sistema subraya el campo en rojo mostrando el mensaje: "Faltan campos por llenar.".
		I(%&^)	I(%&^)	I(%&^)	V(Horas extras)	V(suma)	V(Salario)		El sistema subraya el campo en rojo mostrando el mensaje: "Faltan campos por llenar.".
		I(%&^)	I(%&^)	I(%&^)	I(%&^)	I(%&^)	I(%&^)		El sistema subraya el campo en rojo mostrando el mensaje: "Faltan campos por llenar.".
EP 1.4	Adicionar tipos de incidencias dejando campos vacíos.	I(Vacío)	V(Horas extras)	V(Horas trabajadas fuera de horario laboral)	V(vacío)	V(suma)	V(Salario)	El sistema subraya el campo en rojo mostrando el mensaje: "Este campo es obligatorio.". El sistema	El sistema subraya el campo en rojo mostrando el mensaje: "Este campo es obligatorio.".

								mantiene la interfaz abierta.	
		V (14).	I(Vacío)	V(Horas trabajadas fuera de horario laboral)	V(Horas extras)	V(vacío)	V(Salario)		El sistema subraya el campo en rojo mostrando el mensaje: “Este campo es obligatorio”.
EP 1.5	Cancelar.	NA	NA	NA	NA	NA	NA	El sistema cierra la interfaz sin realizar ninguna operación.	NA