Universidad de las Ciencias Informáticas
Facultad 3

Título: Implementación del componente Subsidios del Subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Mairin Rodríguez Dávila.

Tutor(es): Ing. Rodolfo Rodríguez Molinet.

Junio 2011

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

<nombre autor> <nombre tutor>

_____ _____

Firma del Autor Firma del Tutor

RESUMEN

La informatización de las entidades cubanas como parte del desarrollo tecnológico que necesita el país, precisa de un sistema que gestione todos sus procesos, trayendo consigo la utilización de herramientas computacionales para la gestión de toda esta información.

La Universidad de las Ciencias Informáticas (UCI) en este sentido se ha dado a la tarea de desarrollar un Sistema Integral de Gestión, CEDRUX. Este sistema está concebido bajo los principios de independencia tecnológica e incluye un conjunto de procesos que hasta el momento no se gestionaban en las entidades del país de forma automatizada. Con el objetivo de incorporar estos procesos, el presente trabajo de diploma comprende la Implementación del componente Subsidio, el cual forma parte del subsistema Capital Humano de CEDRUX.

La solución que se muestra se sustenta en el uso de los lenguajes de desarrollo php y Java Script, en herramientas como postgreSQL para la base de datos, el Subversion para el control de versiones, los Framework ExtJs, Zend y Doctrine así como la obtención de un conjunto de artefactos que responden a la fase de Implementación dentro del ciclo de vida del proceso de desarrollo de software. Para la obtención del componente Subsidio se desarrollaron 34 funcionalidades que responden a los requisitos identificados.

PALABRAS CLAVE

Seguridad Social, Subsidio, CEDRUX.

ÍNDICE DE CONTENIDOS

TABLAS

FIGURAS

INTRODUCCIÓN

El capital humano hace referencia a la riqueza que se puede tener en una fábrica, empresa o institución en relación con la calificación del personal que allí trabaja. En ese sentido, el término capital humano representa el valor que el número de empleados de una institución supone de acuerdo a sus estudios, conocimientos, capacidades y habilidades. El factor humano en las organizaciones ha ido adquiriendo una importancia vital, independientemente de la tecnología de que pueda disponerse, el hombre decide en el éxito o no del cumplimiento de su misión por lo que una gestión eficiente de los recursos humanos es fundamental.

La gestión del capital humano abarca un grupo de procesos que interactúan entre sí, como son: la organización del trabajo, la integración y selección, la evaluación del desempeño, la capacitación y desarrollo, la estimulación moral y material, la seguridad social, el autocontrol y la comunicación institucional. En específico el proceso de seguridad social ofrece protección al trabajador ante la enfermedad de origen común o profesional y accidente de origen común o de trabajo, invalidez y vejez y, en caso de muerte del trabajador, protege a su familia, así como, a cualquier persona no apta para trabajar que carezca de familiares en condiciones de prestarle ayuda. La seguridad social a su vez tiene procesos asociados entre ellos el subsidio por enfermedad o por accidente, el mismo se refiere a los ingresos que recibe el trabajador en sustitución del salario, cuando presenta una de las situaciones antes relacionadas y que le impiden laborar. Para la concesión del subsidio por enfermedad o accidente se requiere que el trabajador se encuentre en activo servicio al momento de enfermarse o accidentarse, y que no haya ocurrido por autoprovocación o por motivo u ocasión de cometer un delito intencional o de pretender su comisión.

Actualmente Cuba en las entidades del país realiza el control de los distintos tipos de subsidios de los trabajadores de forma manual llevándose los datos en papel, en el mejor de los casos con sistemas que no están integrados al pago de los mismos, éstos pueden ser, por licencia de maternidad, accidente laboral y enfermedad, ésta situación ocasiona la demora en la entrega de la información pues existe el riesgo de pérdida y otras situaciones relacionadas con errores humanos, pudiendo traer consigo la pérdida del pago del subsidio al trabajador. Con el fin de que esta y otras problemáticas cotidianas sean erradicadas la dirección del país, como parte del fortalecimiento de la gestión de las entidades y la informatización de la sociedad cubana, planteó la necesidad de crear un sistema de Planificación de Recursos Empresariales (ERP por sus siglas en inglés), que fuese capaz de informatizar los procesos de

gestión de las entidades a escala nacional. (1)Es así como a partir de julio de 2008 un grupo de profesionales y estudiantes de la Universidad de las Ciencias Informáticas, en conjunto con el Ministerio de Finanzas y Precios (MFP), el Ministerio de Economía y Planificación (MEP) y la participación de especialistas de otras entidades desarrolladoras de software como la Unidad de Compatibilización, Integración y Desarrollo de Software para la Defensa (UCID), asumieron la gran responsabilidad de crear el ERP cubano, integrado por varios subsistemas, por ejemplo: los subsistemas de Contabilidad, Logística, Capital Humano, Costos y Procesos, Finanzas, Estructura y Composición y Configuración. El subsistema de Capital Humano se encarga de la gestión del capital humano en el cuál se registrarían los datos del certificado médico y los subsidios de los trabajadores, brindando la información necesaria para que el sistema realice el pago a través del sistema de nómina.

Por tal razón surge el siguiente problema a resolver:

La forma actual de realizar el pago de la Seguridad Social para la Administración del Capital Humano obstaculiza generar la nómina mediante el subsistema de Capital Humano del Sistema Integral de Gestión CEDRUX.

El **objeto de estudio** Sistemas de Gestión del Capital Humano. Tomando como **campo de acción** Los procesos de la Seguridad Social. Siendo el **objetivo general de la investigación:** Implementar el componente Subsidios para su integración al subsistema de Capital Humano del Sistema Integral de Gestión CEDRUX.

Se trazaron las siguientes **tareas de investigación**:

- Documentación de la gestión de la Seguridad Social para la Administración del Capital Humano y su relación con sistemas similares a nivel nacional e internacional.
- Justificación del uso de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
- Valoración crítica del análisis y diseño.
- Estudio de la arquitectura definida por el proyecto ERP Cuba.
- Implementación de las funcionalidades del componente subsidios.
- Realización de pruebas al componente obtenido.

Idea a defender:

Si se obtiene el componente subsidio mediante su implementación entonces se podrá generar la nómina mediante el subsistema Capital Humano del Sistema Integral de Gestión Cedrux.

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

Estructura del documento:

Capítulo 1: Fundamentación teórica. Se realiza un estado del arte de los Sistemas de Gestión del Capital Humano que se utilizan a nivel nacional e internacional. Se hace una valoración de las herramientas, lenguaje de modelado y de programación a usar, así como un estudio de la arquitectura y modelo de desarrollo por el que se rige esta investigación.

Capítulo 2: Descripción y análisis de la solución propuesta. Se hará una valoración crítica del diseño propuesto por el analista, un análisis sobre componentes, módulos o implementaciones que podrían ser reutilizadas para dar solución al problema propuesto, un análisis del estándar de codificación a utilizar así como la descripción de las clases implementadas para dar solución al problema.

Capitulo 3: Validación de la solución propuesta: Se valida la solución propuesta a través de la realización de pruebas, el mismo se encuentra dividido en dos partes la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y las pruebas de caja negra que se realizaron a la interfaz del mismo.

Alcanzando como **posible resultado** Obtención del componente Subsidios integrado al subsistema Capital Humano del Sistema Integral de Gestión Cedrux que soporte los requisitos identificados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Durante el desarrollo de este capítulo se describirá cómo se lleva a cabo el proceso de subsidios a partir del estudio de sistemas ya existentes relacionados con el problema a resolver. Además se hará referencia al modelo de desarrollo para dar solución a este problema, a las herramientas y a los lenguajes de modelado y de programación a utilizar.

1.2 Conceptos básicos asociados al dominio del problema:

Subsidio: Es una prestación publica asistencial de carácter económico. Se trata de un medio para estimular el consumo o la producción, o de una ayuda que se brinda por un periodo determinado.

Un subsidio es la diferencia entre el precio real de un bien o servicio y el precio real cobrado al consumidor de estos bienes o servicios.

Los subsidios que reducen lo que paga el usuario, por debajo del costo del bien o servicio. Pueden ser:

a) Subsidios directos: El Gobierno paga directamente una parte del servicio a algunos consumidores. En el mejor de los casos este subsidio debe aparecer dentro de la factura como una rebaja al precio normal, señalando quien lo paga y cuál es la base del cálculo.

b) Subsidios cruzados (entre diferentes usuarios): En este caso la Empresa calcula su tarifa general (que cubre los costos totales) pero no cobra el mismo monto a todos los clientes. Algunos pagan más que el costo real, para permitir que otros paguen menos. No hay necesidad de que el Gobierno ponga nada del costo de este subsidio. Ya que el ingreso total de la empresa se mantiene igual. El sector en su totalidad no está siendo subsidiado; sino, algunos usuarios (quienes, se supone, son los menos necesitados) están subsidiando el consumo de otros usuarios (los supuestamente más necesitados). (2)

Seguridad social: Es el instrumento jurídico y económico que establece el Estado para abolir la necesidad y garantizar a todo ciudadano el ingreso para vivir y la salud, a través del reparto equitativo de la renta nacional y por medio de prestaciones del Seguro Social, al que contribuyen los patrones, los trabajadores y el Estado, o alguno de ellos como subsidios, pensiones y atención facultativa y de servicios sociales, que otorgan de los impuestos de las dependencias de aquel, quedando amparados contra los

riesgos profesionales y sociales, principalmente de las contingencias de la falta o insuficiencia de ganancia para el sostenimiento de él y de su familia. La seguridad social tiene como finalidad garantizar el derecho a la salud, la atención medica, la protección de los medios de subsistencia, el otorgamiento de una pensión garantizada, guarderías y prestaciones sociales para el bienestar individual y colectivo. (3)

Recurso Humano: Es el conjunto de capital humano que está bajo el control de la empresa en una relación directa de empleo, en este caso personas, para resolver una necesidad o llevar a cabo cualquier actividad en una empresa. (4)

Capital Humano: Conjunto de conocimientos, habilidades y aptitudes inherentes a los individuos que forman la organización.

Concepto de Gestión de Recursos Humanos ofrecido por varios autores:

Guillermo Eulogio Ferriol Molina Vicepresidente de la Sociedad Cubana de Derecho Laboral y Seguridad Social agrega que se entiende como la actividad que se realiza en la empresa para:

Obtener, formar, motivar, retribuir y desarrollar los recursos humanos que la organización requiere.· Diseñar e implantar la estructura, sistemas y mecanismos organizativos, que coordinen los esfuerzos de dichos recursos para que los objetivos se consignen de la forma más eficaz posible.· Crear una cultura de empresa que integre a todas las personas que la componen en una comunidad de intereses y relaciones, con unas metas y valores compartidos que den sentido, coherencia, motivación y dedicación.

Santiago Pereda Marín, autor de Técnica de Gestión en Recursos Humanos puntualiza que se refiere a todas las decisiones y acciones directivas relativas a las características de la relación entre la organización y sus empleados. (5)

1.3 Soluciones informáticas vinculadas al campo de acción.

Se investigó sobre las principales aplicaciones informáticas, similares al que se desea desarrollar, caracterizándolas y describiendo sus principales funcionalidades. A continuación, se analizan sistemas informáticos dedicados a la gestión del capital humano en las empresas tanto en ámbito internacional y nacional.

OpenERP

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

Incorpora funcionalidades de gestión de documentos, conexiones con otras aplicaciones y permite trabajar remotamente mediante una interfaz web o aplicación de escritorio multiplataforma.

Según su propia definición y orden de módulos básicos, a continuación se muestran algunos de ellos:

- Facturación, cobros y pagos.
- Contabilidad.
- Estadísticas.
- Productos.
- Recursos Humanos.
- Facturación.
- Gestión de Informes.
- Gestión Documental.

Observación: OpenERP se describe a sí mismo como el ERP de código abierto más destacado y sencillo que existe hasta el momento. Es una aplicación que gestiona de manera eficiente tanto los datos económicos como los recursos humanos en las empresas, sin embargo, el registro de los subsidios de los trabajadores no está comprendido dentro de sus funcionalidades.

ERP SAP

Software de Gestión de Recursos Empresariales desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM, su nombre se forma con las siglas en alemán: Sistemas, Aplicaciones y Procesamiento de datos. SAP está compuesto por una serie de áreas funcionales o módulos que responden de forma completa y en tiempo real a los procesos operativos de las organizaciones. Comprende cuatro soluciones independientes que brindan soporte a procesos de negocio clave a través de su sistema ERP específico: SAP ERP Finanzas, SAP Administración de Capital Humano, SAP ERP Operaciones, SAP ERP Servicios Generales y SAP ERP Planificación de la Producción (PP). Este último maneja la Planificación de Ventas y Operaciones y la Planificación Estratégica de Negocio.(7)

Observación: SAP, como aplicación informática da la posibilidad de ejecutar todos los procesos del negocio haciéndolos más ágiles; incluida la administración de capital humano y la gestión de operaciones del sistema de la contabilidad. Inconveniente fundamental: Software privativo.

SISCONT5

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

- Cobros y Pagos.
- Facturación.
- Activos Fijos Tangibles.
- Nóminas.
- Contabilidad de Costos.
- Inventarios.
- Efectivo en Caja y Bancos. (8)

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

Versat Sarasola

Es el primer sistema de contabilidad cubano certificado, en cuya evaluación participaron el Ministerio de Finanzas y Precios, consultorías internacionales y el organismo encargado de la seguridad informática. Es un sistema económico conformado por 12 módulos que permite llevar el control y registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades, pues es configurable por cada una de ellas en el momento de su instalación. Uno de sus objetivos es permitirle a los directivos, analizar, controlar y evaluar los resultados del negocio o actividad en tiempo real. (9)

Está compuesto por los siguientes módulos:

- Configuración.
- Contabilidad general.
- Control de inventarios.
- Generador de reportes.
- Control de activos fijos.
- Costos y procesos.
- Finanzas, caja y banco.

- Contratación y facturación.
- Planificación económico-productiva.
- Análisis económico empresarial.
- Paquetes de gestión.
- Nóminas de salario. (10)

Observación: Versat-Sarasola es unas de las soluciones informáticas más utilizadas actualmente en el país. En sus últimas versiones tiene en cuenta las particularidades de la economía nacional, comprende la gestión de recursos humanos sin embargo, está además soportado sobre tecnología privativa.

1.3.1 Valoración del Estado de arte:

Después de realizar un estudio a los sistemas informáticos antes mencionados donde se tuvieron en cuenta no sólo los aspectos fundamentales desde el punto de vista del software sino también del producto, se evidencia la inexistencia de un software que responda cabalmente a lo que en realidad necesita la economía cubana, presentan diferentes inconvenientes ya sea por las herramientas sobre las que están soportadas, por las licencias de dichos productos o bien porque no son capaces de cumplir con los requisitos establecidos a nivel nacional, sería entonces factible que el proyecto y el país en el proceso de lograr su soberanía tecnológica trabajaran sobre herramientas y tecnologías que no fueran propietarias, siendo una aplicación web y el trabajo con PHP lo más indicado para desarrollar un sistema de este tipo con el fin de lograr un alto grado de eficiencia, confiabilidad y rapidez en la gestión de las empresas.

1.4 Tendencias y tecnologías actuales:

La industria del software ha mostrado ser una de las áreas más dinámicas y con mayor crecimiento en los últimos años. La evolución hacia a un modelo más racional para los usuarios, con menos costes de licencia, donde se intensifique la prestación de servicios, que reduzca el tiempo de desarrollo e incremente la calidad, viene siendo lo más importante a la hora de desarrollar cualquier tipo de aplicación. A continuación se ofrecerá una valoración sobre algunas de las tendencias y tecnologías que marcan un nivel alto en el mundo del software y que bien contribuye a lo dicho anteriormente:

1.4.1 Aplicación Web:

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

- Compatibilidad Multiplataforma: Una misma versión de la aplicación puede correr sin problemas en múltiples plataformas como Windows y Linux.
- Menos requerimientos de hardware: Este tipo de aplicación no consume (o consume muy poco) espacio en disco y también es mínimo el consumo de memoria RAM en comparación con los programas instalados localmente. Tampoco es necesario disponer de computadoras con poderosos procesadores ya que la mayor parte del trabajo se realiza en el servidor donde reside la aplicación.
- Actualización: Una de las características fundamentales de las aplicaciones web es que siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones ni realizar tareas de instalación. (11)

Cuando se desarrolla una aplicación web existe una tendencia al uso de Arquitectura en capas y del patrón de diseño: Modelo-Vista-Controlador (MVC).

1.4.2 Arquitectura Cliente/Servidor:

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

- Permite integrar y compartir información entre sistemas diferentes sin necesidad de que todos tengan que utilizar el mismo sistema operativo.
- Posibilita el mantenimiento y el desarrollo rápido de aplicaciones. (12)

La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.

Se puede asociar el uso de esta arquitectura con los principios de Arquitectura Orientada a Servicios (SOA7).

1.4.3 Software libre:

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

- Independencia tecnológica: El acceso al código fuente permite el desarrollo de nuevos productos sin la necesidad de desarrollar todo el proceso partiendo de cero.
- Libertad de uso y redistribución: Las licencias de software libre existentes permiten la instalación del software tantas veces y en tantas máquinas como el usuario desee. (13)

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

1.5 Modelo de desarrollo:

Los elementos de un proceso de desarrollo de software y sus relaciones deben responder Quién debe hacer Qué, Cuándo y Cómo. Esto se logra modelando las interacciones y relaciones que suceden entre las personas (roles), las actividades que estas desarrollan y los artefactos que se crean o actualizan durante el proceso.

Quién: Las personas participantes en el proyecto de desarrollo desempeñando uno o más roles específicos.

Qué: Un artefacto es producido por un rol como resultado del desarrollo de sus actividades. Los artefactos se especifican utilizando notaciones. Las herramientas apoyan la elaboración de artefactos.

Cómo y Cuándo: Las actividades son una serie de pasos que lleva a cabo un rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos artefactos. (14)

La propuesta de modelo de desarrollo que a continuación aparece fue elaborada por el equipo de producción en colaboración con las Líneas de desarrollo del proyecto ERP-Cuba de acuerdo con las necesidades presentadas por cada una de ellas y donde se tuvieron en cuenta los principales riesgos con los que se cuentan en el proyecto. (15)

1.5.1 Características:

- **Centrado en la arquitectura:**

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

- **Orientado a componentes:**

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

- **Iterativo e incremental:**

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

- **Ágil y adaptable al cambio:**

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

1.5.2 Arquitectura

Uno de los principales elementos que debe tener el proceso de desarrollo de software es la arquitectura sobre el cual va a estar sustentado. “La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. (16)

A partir de esta definición se puede ver a la arquitectura de una aplicación como la vista conceptual de la estructura de esta y donde se establecen los fundamentos básicos para que analistas, diseñadores y programadores trabajen en una línea común y así alcanzar los objetivos del sistema de acuerdo con las necesidades del cliente.

1.5.2.1 Características de la Arquitectura Base:

La arquitectura base del Sistema CEDRUX está conformada por las diferentes vistas y estilos arquitectónicos que serán especificados a continuación:

- Arquitectura Basada en Componentes:

Uno de los enfoques en los que actualmente se trabaja es la arquitectura basada en componentes que tiene como objetivo hacer un uso correcto de software reutilizable, para la construcción de aplicaciones mediante el ensamblaje de partes ya existentes.

“Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y de requisitos, que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio” (17)

El equipo de arquitectura del proyecto ERP-Cuba dentro de la vista del sistema antes presentada incluye la vista de componente como una de sus subdivisiones; en ese caso propone que todas las funcionalidades levantadas y modeladas en las fases de negocio y requerimientos deben ser expresadas o contenidas en al menos un componente y que las distintas interacciones entre ellos originen funcionalmente la existencia de subsistemas en consecuencia a las dependencias definidas. Ejemplificado en el modelo de componentes del Subsistema Capital Humano que será presentado posteriormente.

- Modelo-Vista-Controlador (MVC):

El patrón arquitectónico MVC es utilizado para el desarrollo de aplicaciones Web con el fin de separar en tres componentes distintos la interfaz de usuario, la lógica de negocio y los datos persistentes, potenciando la flexibilidad y la adaptabilidad a futuros cambios. Por su parte:

El Modelo: Es la representación de la información que maneja la aplicación. Son los datos puros que puestos en un contexto del sistema son mostrados al usuario por medio del Controlador, proveen de información al usuario o a la aplicación misma.
La Vista: Constituye la representación del modelo en forma gráfica, disponible para la interacción con el usuario. En una aplicación web la "Vista " es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

El Controlador: Se encarga de responder a las solicitudes del usuario desde la interfaz, manejando los diferentes eventos a través de las funcionalidades necesarias y la información perteneciente al Modelo. (18)
Para el desarrollo del Sistema Integral de Gestión de Entidades CEDRUX se decidió trabajar con este patrón, evidenciándose en los framework definidos para cada una de las capas como parte del MVC de cada componente dentro de la aplicación, es decir: para la Vista: Extjs-Framework, el cual es muy utilizado en el desarrollo de aplicaciones Web con tecnología AJAX, para el Controlador: Zend-Framework quien emplea específicamente el estilo Modelo- Vista- Controlador como base de su

funcionamiento y para agilizar el acceso a datos en el Modelo se utilizó Doctrine, un potente y completo sistema ORM (Mapeo Objeto Relacional).

De forma general, según lo descrito con anterioridad, en la arquitectura del Proyecto ERP-Cuba los estilos arquitectónicos que se emplean no se pueden ver de forma independiente sino como un estilo híbrido que comprende numerosas ventajas:

- Desarrollos paralelos: en cada capa.
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo: es más sencillo cambiar un componente que modificar íntegramente una aplicación.
- Mayor flexibilidad: se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad.
- Alta escalabilidad: La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. (19)

1.6 Lenguajes de programación:

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

1.6.1 Lenguaje del lado del servidor:

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

- **Lenguaje PHP:**

PHP es un lenguaje de programación usado normalmente para la creación de páginas web dinámicas. Es conocido como una tecnología de código abierto que resulta muy útil para diseñar de forma rápida y eficaz aplicaciones web dirigidas a bases de datos. No requiere definición de tipos de variables, no es un

lenguaje de marcas. Permite la conexión a numerosas bases de datos de forma nativa tales como Postgres, MySQL, Oracle y Microsoft SQL Server, lo cual permite la creación de aplicaciones web muy robustas. PHP tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos tales como UNIX, Linux y Windows. (20)

Características:

1. Es un lenguaje multiplataforma.

2. Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL.

3. Capacidad de expandir su potencial utilizando una enorme cantidad de módulos.

4. Posee una amplia documentación en su página oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.

5. Es libre, lo que representa que una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.

6. No requiere definición de tipos de variables.

7. Tiene manejo de excepciones.

A partir de la decisión de la dirección del proyecto y del equipo de arquitectura, lo más factible para la implementación es la utilización del lenguaje PHP debido a las facilidades ya presentadas, la experiencia alcanzada sobre el mismo en el desarrollo de aplicaciones web y la posibilidad de utilizarlo libremente y comercializar los productos realizados con él. En este caso se estará haciendo uso de PHP en su versión 4.0 o superior, con los siguientes módulos o extensiones: pdo, pdo_pgsql, pgsql, soap, gd2, mbstring, mysql, mysqli, pdo_sqlite, sqlite, xsl.

1.6.2 Lenguajes del lado del cliente:

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

- **HTML:**

Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML. (21)

- **XML:**

Es el metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos. No es un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. (22)

- **JavaScript:**

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM9. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape y Mozilla Firefox. (23)

1.7 Frameworks:

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

1.7.1 Ext:

Librería construida empleando JavaScript con el fin de desarrollar aplicaciones web interactivas usando tecnologías como AJAX y DOM. Ext es muy potente ya que contiene rica colección de componentes para

el diseño de Interfaces Gráficas de Usuario (GUI's) del lado del cliente haciendo uso extensivo de AJAX. Dispone de un conjunto de componentes gráficos como (24):

1. Cuadros y áreas de texto.
2. Campos para fechas.
3. Campos numéricos.
4. Selectores estáticos y dinámicos.
5. Botones.
6. Editor HTML.
7. Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar).
8. Árbol de datos.
9. Pestañas.
10. Barra de herramientas.
11. Menús al estilo de Windows.

Se estará haciendo uso de Ext en su versión 2.2.

1.7.2 Zend:

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

1. Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos.
2. Proporcionan los componentes que forma la infraestructura del patrón MVC.
3. Proporciona una capa de acceso a base de datos, construida sobre PDO10 pero ampliándola con diferentes características.
4. Proporciona mecanismos de filtrado y validación de entradas de datos.
5. Permite convertir estructuras de datos PHP a JSON11 y viceversa, para su utilización en aplicaciones AJAX (especificado en el epígrafe 1.9.1).
6. Proporciona capacidades de búsqueda sobre documentos y contenidos. (24)

Se estará haciendo uso de Zend Framework en su versión 1.9.7.

1.7.3 Doctrine:

Doctrine es un potente y completo sistema ORM12 (en inglés Object Relational Mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Se estará haciendo uso de Doctrine en su versión 1.2.1.

1.8 Tecnologías y Herramientas de desarrollo:

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

1.8.1 Tecnologías:

- **Tecnología AJAX:**

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. (26)

1.8.2 Herramienta de desarrollo colaborativo:

- **Control de versiones:**

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto). Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema.

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

1. Mantiene versiones no sólo de archivos, sino también de directorios.
2. Mantienen versiones de los metadatos asociados a los directorios.
3. Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
4. Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
5. Soporte tanto de ficheros de texto como de binarios.
6. Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos. (27)

Se estará haciendo uso de un cliente Subversion: TortoiseSVN-1.4.5.

1.8.3 Entorno integrado de desarrollo:

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

- **Zend Studio para Eclipse:**

Es un IDE (Entorno integrado de desarrollo) para el lenguaje de PHP escrito en Java destinado a desarrolladores profesionales, propietario, compatible con las plataformas Linux, MAC y Windows. Incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Posibilita un excelente completamiento de código, coloreado en la sintaxis del código, administración avanzada de proyectos, múltiples lenguajes, incorpora el Framework de Zend, integración con subversión, integración avanzada con FTP, soporte para Web Services, PHP4 y PHP5, entre otras características están:

1. No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
2. Detección de errores de sintaxis en tiempo real.
1. Manual de PHP integrado.
2. Ofrece soporte básico para otros lenguajes web, como HTML, JavaScript y XML.

3. Acceso al paquete de plug-ins de Eclipse.
4. Mecanismo de actualización automática. (28)

Se estará haciendo uso de Zend Studio para Eclipse en su versión 3.3.

1.8.4 Servidor de Aplicaciones web:

Es un programa que permite crear un servidor http en un ordenador de una forma rápida y sencilla. Está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Es además un programa que implementa el protocolo HTTP el cual se encarga de transferir lo que llamamos hipertextos, páginas web o páginas HTML: textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.

- **Servidor web Apache:**

Es una tecnología gratuita de código fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable es decir se pueden elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. (29)

Se estará haciendo uso del Servidor web Apache en su versión 2.0 ó superior.

1.8.5 Sistema Gestor de Base de Batos:

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

- **PostgreSQL:**

PostgreSQL es un servidor de base de datos relacional, libre. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que es multiplataforma. Fue diseñado para ambientes de alto volumen. Tiene mejor soporte para vistas, procedimientos almacenados en el servidor, transacciones, almacenamiento de objetos de gran tamaño y además tiene ciertas características orientadas a objetos. Se estará haciendo uso de PostgreSQL en su versión 8.3 ó superior e inferior a 8.4

1.8.6 Navegador:

Software que permite al usuario recuperar y visualizar documentos de hipertexto desde servidores web a través de Internet.

- **Mozilla Firefox:**

Mozilla Firefox es el nuevo e innovador navegador open source. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. Es Software libre. Se trata de un práctico y ágil navegador, que está en renovación constante. Tiene la capacidad de modificarlo totalmente a gusto del usuario y según las necesidades del mismo. Algunas características de Mozilla Firefox son:

1. Navegación por tabs, esta es una de las principales características que tiene Firefox, es posible navegar por pestañas.
2. Es Software libre.
3. Trabaja de forma excelente en computadoras sin hardware muy potente, el programa está diseñado para realizar un bajo consumo de recursos. (30)

Se estará haciendo uso de Mozilla Firefox en su versión 2.0.1 ó superior.

1.9 Conclusiones del capítulo:

La Gestión de la Seguridad Social, específicamente el Subsidio es un proceso fundamental para la planificación y la organización de los recursos empresariales en las entidades cubanas; este capítulo fue esencial para valorar el estado actual del mismo a partir de soluciones informáticas tanto nacionales como internacionales, evidenciando la no existencia de un sistema informático capaz de ejecutar tales funcionalidades ni de cumplir con los requisitos establecidos a nivel nacional.

Este capítulo fue básico para la definición de un conjunto de conceptos fundamentales asociados al dominio del problema. Se explicó el modelo de desarrollo a utilizar. Se analizaron las tendencias y tecnologías actuales para el desarrollo de aplicaciones informáticas y por último se realizó una presentación de las tecnologías y herramientas conjuntamente con sus versiones, propuestas por la dirección del proyecto para el desarrollo de la solución.

CAPITULO 2: PROPUESTA DE SOLUCIÓN

2.1 Introducción

En este capítulo, se presenta la estructura de trabajo que propone Sauxe como Marco de trabajo, a partir de esta organización se explica el diseño alcanzado como propuesta para desarrollar el sistema, los principales aspectos que se han tenido en cuenta para la implementación del mismo, teniendo presente la propuesta de los analistas y diseñadores del sistema para lograr así un producto con la mayor calidad y eficiencia requerida. El objetivo principal ha sido detallar los principales procesos con los que se deben trabajar para lograr que una vez implementado el sistema, este satisfaga las necesidades del usuario final.

2.2 Estructura del Marco de Trabajo:

Cumpliendo con las decisiones arquitectónicas de la dirección del proyecto y del departamento de tecnología y mediante el empleo del Marco de Trabajo Sauxe se define una estructura contenedora donde van a estar ubicados cada uno de los subsistemas implementados dentro de la aplicación, de manera que facilite la organización y claridad durante el desarrollo, la especificación que a continuación se presenta está enfocada al componente Subsidios del Subsistema Capital Humano.

La carpeta denominada **apps** es donde se almacenan los controladores y el modelo de cada uno de los componentes correspondientes a los subsistemas. En la figura se muestra dicho contenido para el componente y el subsistema en cuestión.

Figura 1 Carpeta de aplicación correspondiente al Subsistema Capital Humano.

Común: Comprende la configuración específica para el Subsistema, contiene la carpeta **recursos** y dentro de esta una denominada **xml**. Esta última incluye a los ficheros: **ioc**, **validator**, **exception** que serán explicados a continuación.

Ioc: Es donde se publican los servicios que brinda cada uno de los componentes del Subsistema en cuanto a nombre de clases, funciones y tipo de resultado.

Validator: Chequea las precondiciones antes de ejecutar una determinada función en el servidor según el tipo de parámetros, la acción y el usuario que la realice.

Exception: Se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción en el servidor.

El componente **Subsidios** va a contener un conjunto de paquetes que serán especificados a continuación:

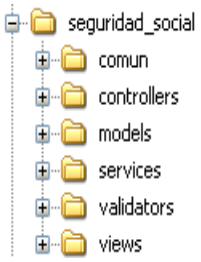

Figura 2 Carpeta de aplicación correspondiente al Subsistema Capital Humano.

En el paquete **controllers** se encontrarán las clases controladoras encargadas de gestionar las funcionalidades del sistema.
El paquete **models** estará estructurado de la siguiente forma:

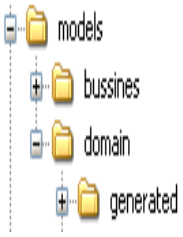

Figura 3 Paquete models correspondiente a la carpeta de aplicación.

Este paquete contiene dos paquetes los cuales a su vez agrupan clases y otros paquetes. El paquete **bussines** contendrá las clases necesarias para acceder a los datos que persisten en la base de datos.
El paquete **domain** debe contener las clases generadas por el ORM Doctrine a partir de cada una de las tablas existentes en la base de datos. Cada una de estas clases heredará de una clase generada igualmente por el Doctrine las cuales se ubican en otro paquete dentro de este llamado **generated.**
services: Este paquete incluirá todas las clases y funcionalidades de los servicios que va a ofrecer el componente. Para solicitar un servicio que está en otro dominio, accedemos a través del paquete

Services, quien analizará la solicitud e irá a la clase que tiene dicha funcionalidad y la devolverá a Services que a su vez se la entregará al dominio solicitante.

El paquete **views** contendrá los paquetes idioma y scripts, que se encargan de contener el idioma en que se va a mostrar la aplicación y las páginas clientes respectivamente.

Figura 4 Paquete view correspondiente a la carpeta de aplicación.

Al mismo nivel de la carpeta **apps** mencionada anteriormente debe existir además una carpeta que contiene las vistas de los subsistemas y componentes. Dicha carpeta se denomina **web**.

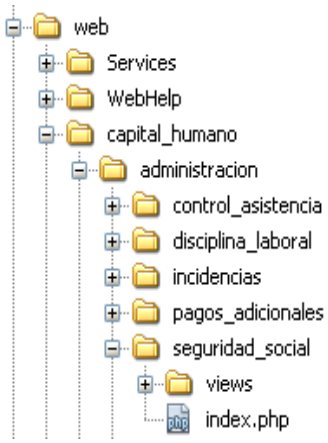

Figura 5 Carpeta de diseño correspondiente al Subsistema Capital Humano.

Index: Este fichero incluye la dirección del archivo de configuración y a través de este inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento. Su código permanece igual para todos los componentes.

Figura 6 Paquete seguridad_social correspondiente a la carpeta de diseño.

El **index** que se encuentra dentro del Subsistema y los que corresponden a cada uno de los componentes cumplen la misma función.

Figura 7 Paquete view correspondiente a la carpeta de diseño.

La carpeta **view** contendrá los **css** y los **js** que se explicarán a continuación.

El paquete **css** incluirá las clases necesarias para estructurar gráficamente el componente, separando de esta forma el estilo del contenido. El paquete **temas** que debe estar dentro de la carpeta **lib**, agrupará las clases representativas del framework ExtJs necesarias para poder utilizar sus componentes, por lo que se hace necesario incluirlas en cada una de las plantillas a utilizar.

El paquete **js** comprenderá las clases JavaScript necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios. Está compuesto por paquetes con los nombres de las funcionalidades del componente.

2.3 Valoración del diseño propuesto por el analista

Siguiendo la anterior organización de trabajo, el diseño propuesto por el analista es fácil de entender, las clases que están representadas agrupan las funcionalidades que deben ser definidas para que el sistema funcione satisfactoriamente, así como los atributos y métodos que deben tener las mismas, dando una idea clara de lo que se debe desarrollar. Unido a esto se encuentran los diagramas de interacción, que explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una, de los cuales se obtiene la información necesaria para conocer el orden de las acciones que se deben implementar.

El diseño presentado potencia la reutilización del código, la flexibilidad, la organización y el rendimiento del sistema, facilita a los programadores la adaptabilidad al mismo y de esta forma agiliza el proceso de traducción del diseño a la implementación.

Otra de las características del diseño que evidencia la claridad del trabajo es el uso de grupos de patrones de diseño como es el caso de los “GRASP” y los “GOF”. Los patrones “GRASP” se utilizan con el objetivo

de asignar responsabilidades a las diferentes clases que se definen en el diseño. Dentro de este grupo de patrones se identifican: experto, creador, alta cohesión, bajo acoplamiento y el controlador.

Experto: Dicho patrón es evidenciado en la definición de las clases de acuerdo a las funcionalidades que deben realizar a partir de la información manejada dentro del componente, por ejemplo las clases controladoras y las del modelo. Específicamente: la clase DatDiagnostico, en la configuración de los destinos de los productos, será la responsable de efectuar las operaciones que conciernen a las funciones: definir, eliminar y actualizar los diagnósticos, asumiendo toda la lógica para cada una de ellas. Sobre este mismo principio se realiza el diseño de las restantes funcionalidades.

Creador: Este patrón es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

Alta cohesión: Este patrón fue utilizado en el diseño del componente de manera general; donde se agruparon las clases en dependencia de los requerimientos: Gestionar Centro Médico, Gestionar Diagnóstico, Gestionar Tipo de Subsidio, Gestionar Doctor y Gestionar Notificación de Subsidio a los que se les debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Bajo acoplamiento: En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

Controlador: Las clases controladoras definidas GestionarCentroMedicoController, GestionarDiagnosticoController y GestionarTipoSubsidioController son un ejemplo de la aplicación de este patrón, las mismas tendrán a cargo la responsabilidad de manejar los eventos dentro del componente. Estos patrones se les aplican a las clases definidas en el diseño, distribuyendo responsabilidades entre las mismas de forma tal que no existan muchas relaciones, que no se sobrecargue de métodos a una clase en específico pudiendo acomodarlos en otras, entre otras mejoras que brinda el uso de este grupo de patrones.

Los patrones “GOF” generalmente se evidencian en clases que son creadas debido al uso de un patrón en

específico. Existe un grupo de patrones de este tipo definidos para el diseño de clases con el propósito de crear una arquitectura robusta para el sistema a desarrollar. Durante el diseño del componente se emplearon patrones GOF, específicamente:

Fachada: La aplicación de este patrón en el componente Subsidio se evidencia en la interfaz de servicios simple que se proporciona para establecer la comunicación con otros componentes dentro y fuera del Subsistema.

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

2.4 Artefactos de la fase de implementación

2.4.1 Modelo de datos

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

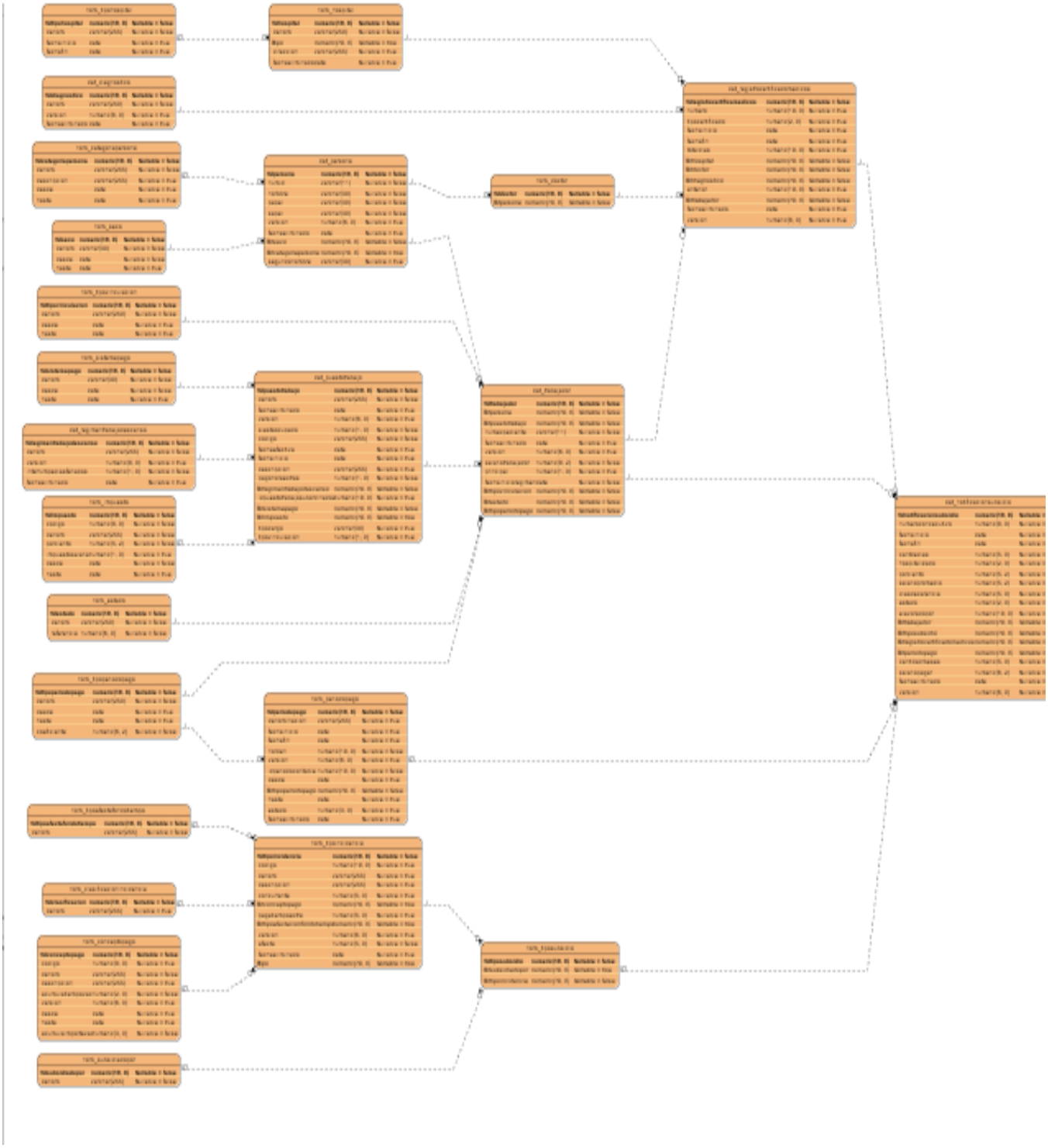

Figura 8 Modelo de datos.

2.4.2 Modelo de componentes.

Un modelo de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan. A continuación se muestra el Modelo de componente y el Diagrama de interacción del componente.

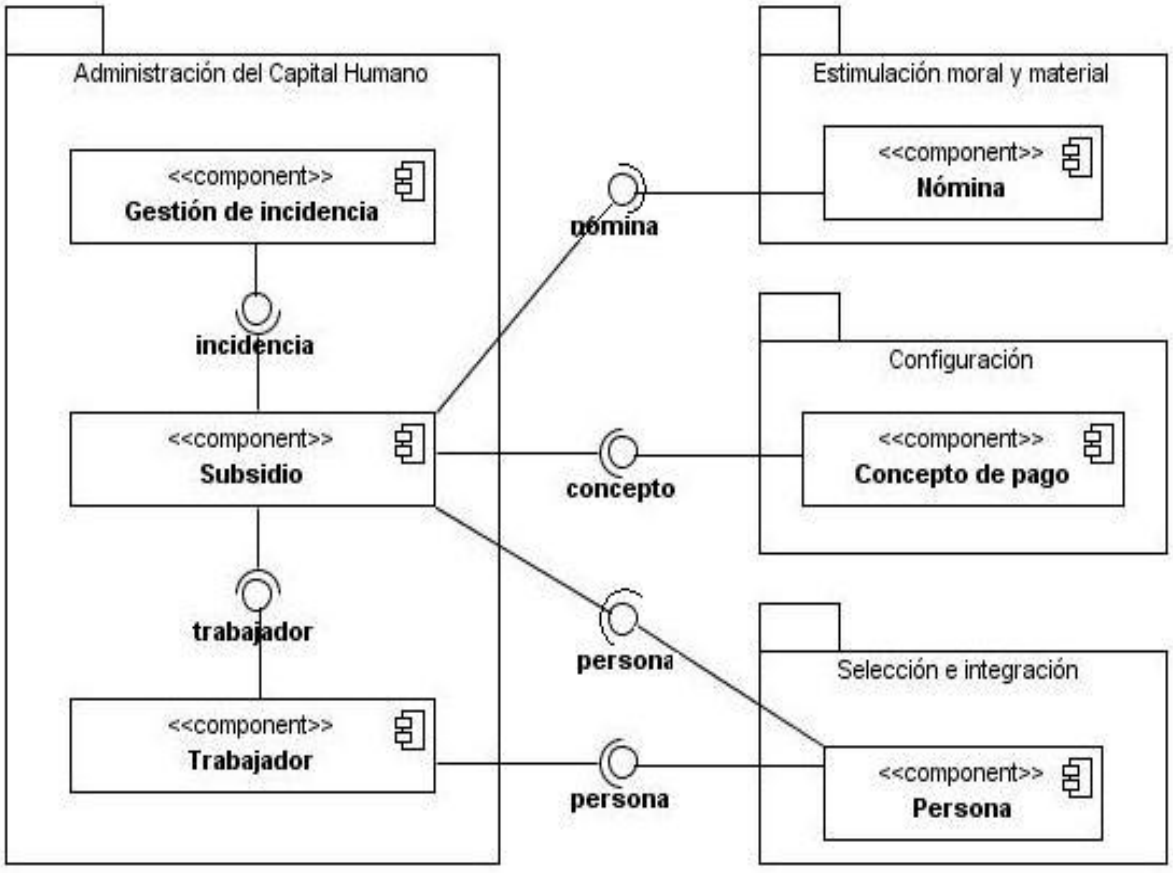

Figura 9 Modelo de componente.

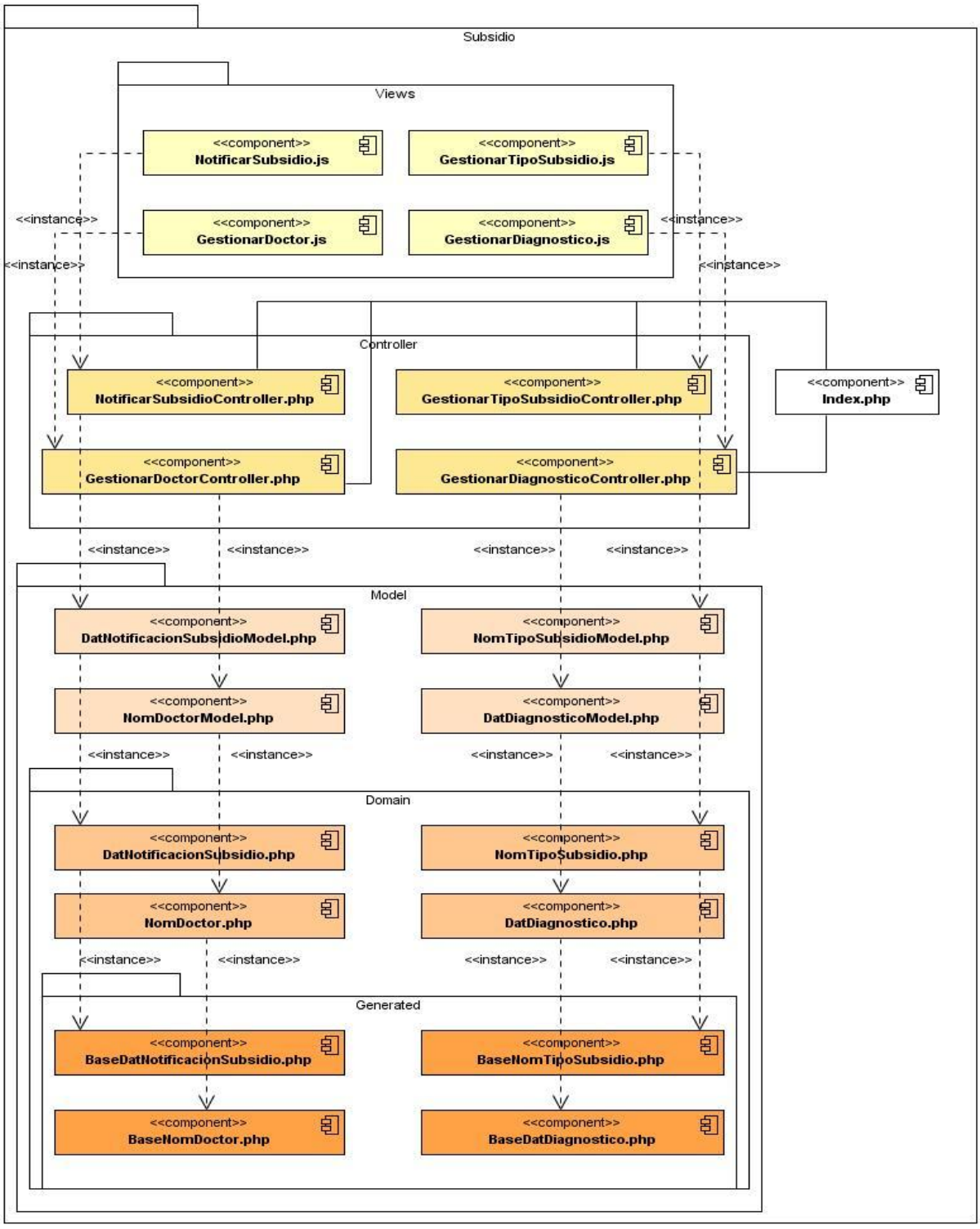

Figura 10 Diagrama de interacción del componente.

Estrategia de integración:

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 concebida sobre 4 nodos y a partir de cada uno de los elementos arquitectónicos definidos.

El primer nodo se sitúa entre la vista y el controlador, el segundo está entre el controlador y el modelo, el tercero vincula el modelo con el framework doctrine y el último se encuentra entre la base de datos y el doctrine.

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

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

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

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

La comunicación 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 mas allá de un simple llamado a un servicio, esta se basa en el empleo de un registro de datos de los subsistemas contenidos en un fichero xml mapeado por el framework para el funcionamiento del componente llamado: inversión de control (IoC). El IoC registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema, especifica respuestas deseadas a sucesos o solicitudes de datos concretas, orden necesario y el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio.

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

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

CargarSexosPersona: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente subsidios que devuelve todos los sexos de la persona.

GuardarPersona: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente subsidios guardar los datos de los doctores.

ActualizarPersona: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente subsidios actualiza los datos de los doctores.

EliminarPersona: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente subsidios elimina los datos de los doctores insertando en la tupla correspondiente al doctor a eliminar la fecha actual en la columna fecha eliminada de la persona.

BuscarCategoriaPorParam: Servicio que brinda el componente Persona del subsistema Capital Humano que le permite al componente subsidios obtener la categoría de un doctor.

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

ModificarEstadoPuestoTrab: Servicio que brinda el componente Trabajador del subsistema Capital Humano que le permite al componente subsidios modificar el estado del puesto de trabajo una vez que ha sido insertada una notificación de subsidio de un trabajador en especifico.

ObtenerConcepto: Servicio que brinda el componente Concepto de Pago del subsistema Capital Humano que le permite al componente subsidios obtener todos los conceptos de pago.

InsertarIncidencia: Servicio que brinda el componente Incidencia del subsistema Capital Humano que le permite al componente subsidios guardar un tipo de subsidio como una incidencia.

ActualizarIncidencia: Servicio que brinda el componente Incidencia del subsistema Capital Humano que le permite al componente subsidios actualizar los datos de un tipo de subsidio.

EliminarIncidencia: Servicio que brinda el componente Incidencia del subsistema Capital Humano que le permite al componente subsidios eliminar los datos de un tipo de subsidio.

PeriodosDePago: Servicio que brinda el Periodo de Pago del subsistema Capital Humano que le permite al componente subsidios cargar todos los periodos de pagos que se encuentran vigentes.

ObtenerIncidenciasSubsidio: Servicio que brinda el componente Incidencias del subsistema Capital Humano que le permite al componente subsidios obtener las incidencias que son a su vez tipos de subsidios.

DameEstructura: Servicio del subsistema Metadatos (Estructura y Composición) se utiliza para obtener los datos de la estructura de la que se está elaborando.

GetUsersProfile: Servicio que brinda el subsistema Seguridad que le permite al componente subsidios obtener los datos del perfil del usuario que se encuentra autenticado en el sistema.

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

SubsidiosDeTrabPorPeriodo: Servicio que brinda el componente Subsidios que le permite al componente Nómina obtener las notificaciones de subsidio que han sido confirmadas de un trabajador en un período de pago dado.

CambiarEstadoNotificaciones: Servicio que brinda el componente Subsidios que le permite al componente Nómina cambiar el estado de la notificación a estado pagada una vez que la nómina ha sido aprobada.

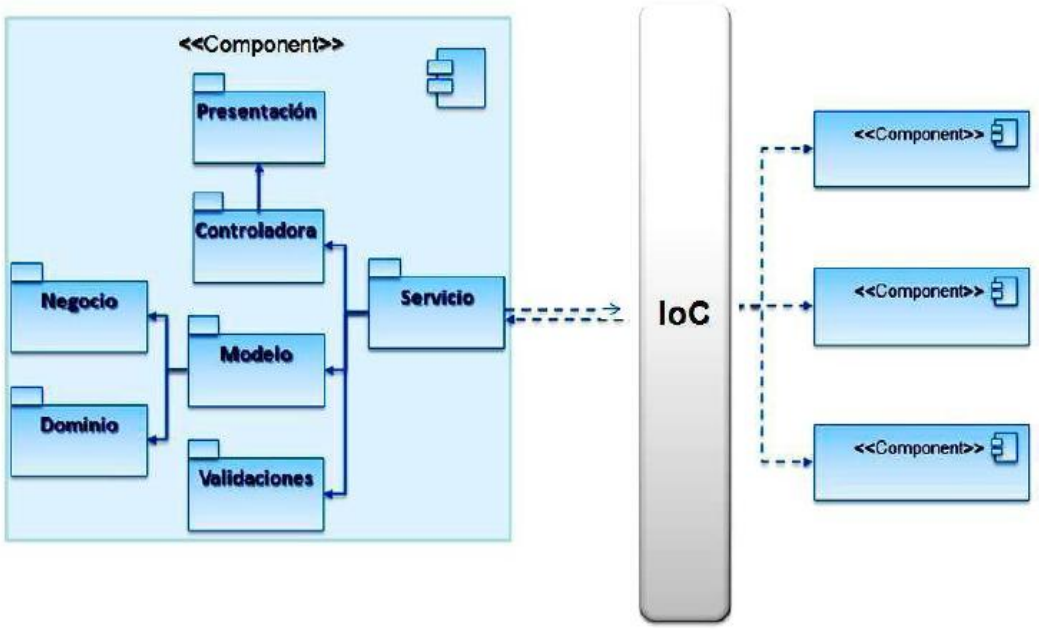

Figura 11 Integración entre componentes.

Descripción general del funcionamiento del componente:

El usuario realiza una petición en el navegador, la cual genera un evento mediante un formulario JS, que envía una notificación a la clase controladora mediante AJAX. El controlador entonces es el encargado de decidir quién va a ser el responsable de obtener los datos de la petición, una clase Bussines si la petición realizada inserta modifica o elimina (CRUD) o una clase Domain si lo que se desea es consultar, esta última es mapeada por el ORM Doctrine y se conecta mediante PDO (Doctrine Record) a la DBO (Capa interna de doctrine basada en PDO nativo) y la DBO es la que se conecta directamente a la base de datos. La clase controladora es la encargada de la lógica propia del negocio, cálculos y procesamientos.

2.4.3 Diagrama de despliegue

El Diagrama de Despliegue se utiliza para capturar los elementos de configuración del procesamiento y las conexiones entre esos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos, muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos

conectados por enlaces de comunicación. Un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento.

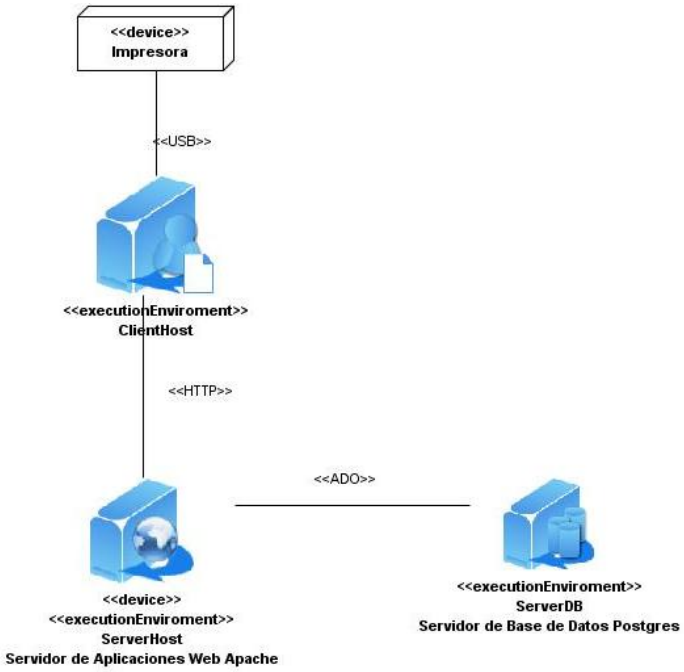

Figura 12 Diagrama de Despliegue.

2.5 Requisitos del componente subsidios

Los requisitos funcionales descritos por los analistas del Subsistema facilitaron el entendimiento de los procesos a implementar permitiendo una buena comprensión del problema y posibilitando la identificación de las clases y las funcionalidades a desarrollar. Se determinaron un total de 34 requisitos funcionales, identificados en 6 agrupaciones según el objetivo al que respondían.

Gestionar tipos de subsidio

- Adicionar tipo de subsidio.
- Modificar tipo de subsidio.
- Eliminar tipo de subsidio.
- Listar tipo de subsidio.
- Imprimir tipo de subsidio.

Gestionar centro médico

- Adicionar centro médico.
- Modificar centro médico.
- Eliminar centro médico.
- Listar centro médico.
- Imprimir centro médico.

Gestionar doctor

- Adicionar doctor.
- Modificar doctor.
- Eliminar doctor.
- Listar doctor.
- Imprimir doctor.

Gestionar diagnóstico

- Adicionar diagnóstico.
- Modificar diagnóstico.
- Eliminar diagnóstico.
- Listar diagnóstico.
- Imprimir diagnóstico.

Registrar certificado médico

- Adicionar certificado médico.
- Modificar certificado médico.
- Eliminar certificado médico.
- Listar certificados médicos.
- Consultar certificado médico.
- Imprimir certificado médico.

Gestionar notificación de subsidio

- Adicionar notificación subsidio.
- Modificar notificación subsidio.
- Eliminar notificación subsidio.
- Listar notificaciones subsidio.
- Consultar notificación subsidio.
- Calcular importe a pagar.

- Imprimir notificación subsidio.
- Confirmar notificación de subsidio.

2.6 Interfaz de usuario

Gestionar notificación de subsidio

Listar notificación de subsidio

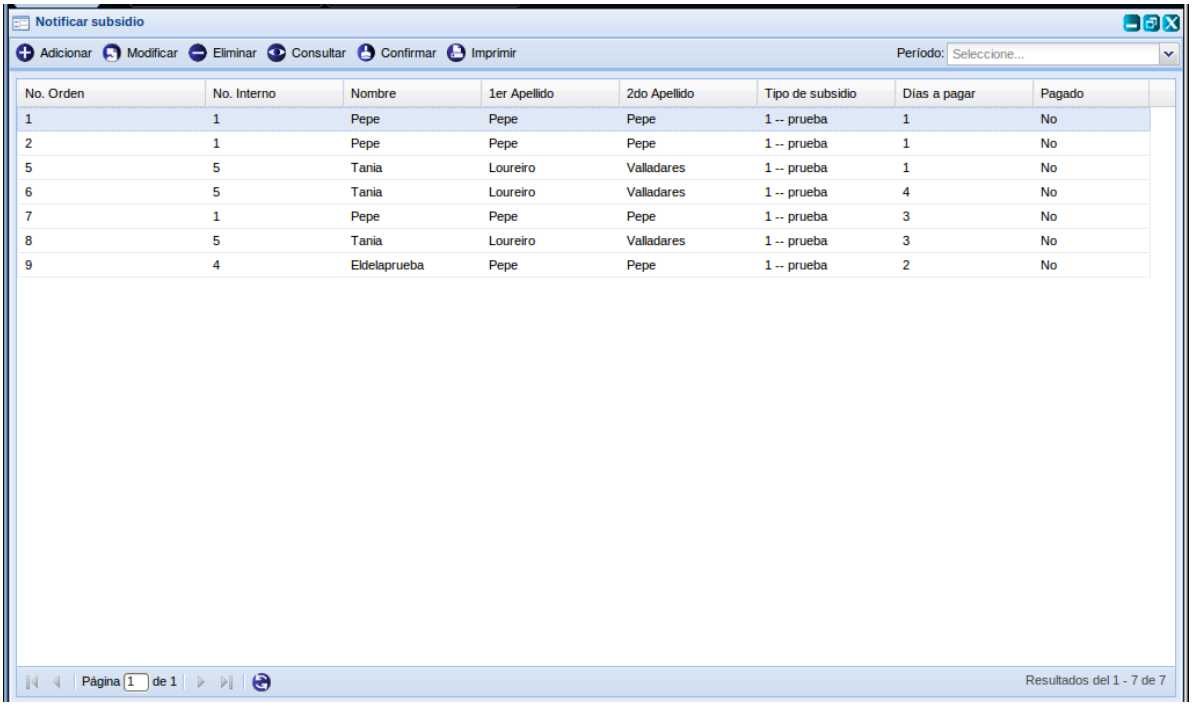

Figura 13 Listar notificación de subsidio

Adicionar notificación de subsidio

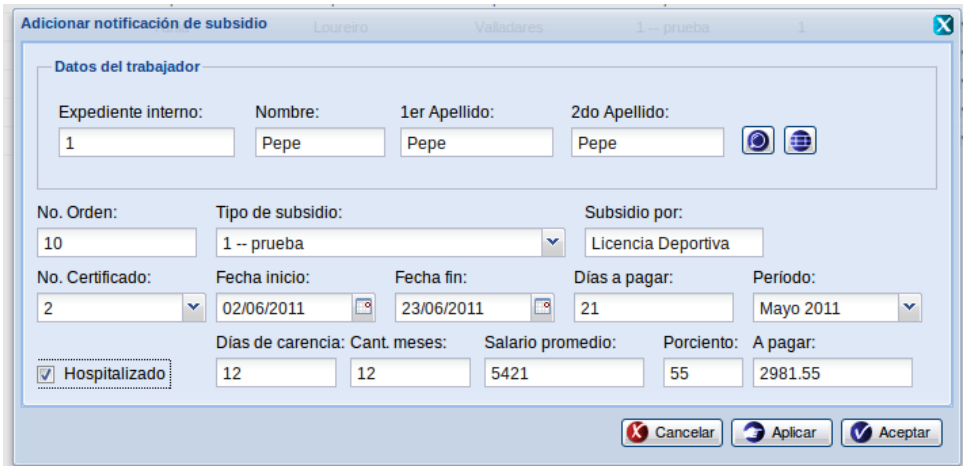

Figura 14 Adicionar notificación de subsidio

Modificar notificación de subsidio

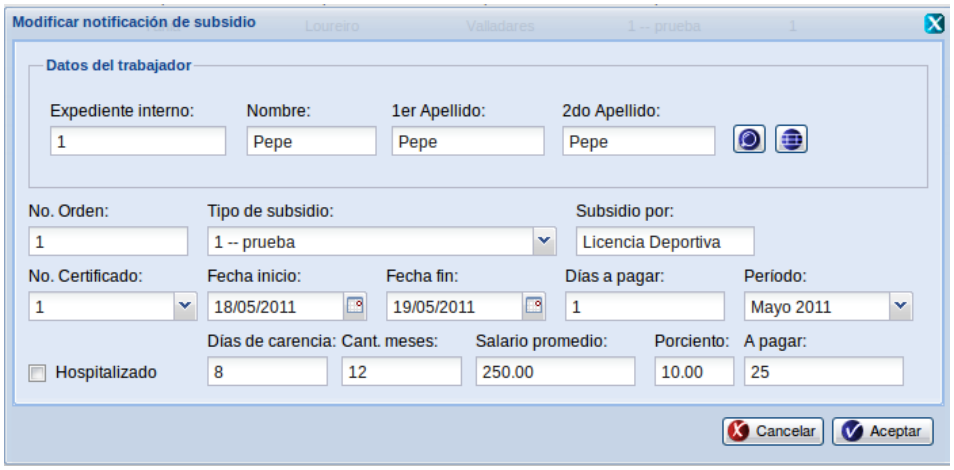

Figura 15 Modificar notificación de subsidio.

Consultar notificación de subsidio

Figura 16 Consultar notificación de subsidio.

2.7 Estándares de código:

Para la implementación de los 34 requisitos identificados se aplicó un estándar de código basado en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa.

Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito. Partiendo de lo dicho anteriormente, se definen 3 partes principales dentro de un estándar de programación:

- Convención de nomenclatura: Es la forma de nombrar las variables, funciones y métodos.
- Convenciones de legibilidad de código: Es la forma de organizar el código.
- Convenciones de documentación: Es la manera de establecer comentarios y la ayuda.

1. Nomenclatura de las clases:

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**, la cual define que los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas, iniciando cada palabra con letra mayúscula y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: NomTipoHospital. En este caso el nombre de clase está compuesto por 3 palabras iniciadas cada una con letra mayúscula.

1.1 Nomenclatura según el tipo de clases:

Clases controladoras: Las clases controladoras después del nombre llevan la palabra:"Controller". Ejemplo: GestionarDoctorController

Clases de los modelos:

Business (Negocio): Las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model". Ejemplo: NomTipoSubsidioModel.

Domain (Dominio): Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la base de datos. Ejemplo: DatDiagnostico.

Generated (Dominio base): Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: "Base" y seguido el nombre de la tabla en la base de datos.

Ejemplo: BaseDatDiagnostico.

2. Nomenclatura de las funcionalidades y atributos:

El nombre a emplear para las funciones y los atributos se escribe con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará notación **CamelCasing** que es similar a la antes mencionada: **PascalCasing** con la excepción de la primera letra.

Ejemplo de método: obtenerFormato (). El nombre de método está compuesto por 2 palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

Las principales funcionalidades de las clases controladoras se les pone el nombre y seguida la palabra:"Action" Ejemplo: obtenerFormatoAction ().

Ejemplo de atributo: tipoSubsidio. El nombre del atributo está compuesto por 2 palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

3. Nomenclatura de los comentarios:

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. En caso de ser una función complicada se debe comentar para lograr una mejor comprensión del código.

2.8 Estructura de datos a utilizar:

Se define como estructura de datos en programación, a la forma de organizar un conjunto de datos elementales con el fin de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema. Existen diversas estructuras que en sí poseen ventajas y desventajas según la simplicidad y eficiencia para la realización de cada operación. Sin embargo a la hora de elegir la estructura de datos más conveniente es preciso evaluar factores básicos como la frecuencia y el orden en que se realiza cada operación sobre los datos.

Durante la realización de una sintaxis de código en PHP; lenguaje definido para la implementación de los diferentes componentes de la aplicación debido a las facilidades que brinda; aparecen variables que tienen información similar y que se procesan de forma semejante, concretamente, se está hablando de los arrays como estructura de datos. Un array o como también se le conoce: arreglo, es un conjunto de variables (elementos) agrupadas bajo un único nombre en distintas casillas. En dicho lenguaje de programación existen dos tipos de arreglos: los de índices numéricos y los de índices asociativos, ambos poseen una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, para contar el número de elementos que comprende y movernos por dentro de él hacia delante o atrás. A partir de lo antes descrito y por decisión del equipo de arquitectura esta vendría siendo la estructura de datos a utilizar de forma general para el desarrollo en todos los subsistemas.

2.9 Descripción de las clases y funcionalidades del componente.

Clases Controladoras:

Las clases controladoras coordinan las actividades de los objetos que implementan las funcionalidades, definen el flujo de control y las transacciones entre los objetos.

Tabla 1: Descripción de la clase GestionarTipoSubsidioController.

Nombre: GestionarTipoSubsidioController	
Tipo de clase: Controladora	
Atributo	**Tipo**
tipoSubsidio	Objetos de tipo NomTipoSubsidioModel.
subsidiadoPor	Objetos de tipo NomSubsidiadoPorModel.

notificacionSubsidio	Objetos de tipo DatNotificacionSubsidioModel
Para cada responsabilidad:	
Nombre:	**Descripción:**
gestionarTipoSubsidioAction()	La función permite establecer la comunicación entre la vista (ExtJs) y la clase controladora Gestionar Tipo Subsidio**.**
obtenerTipoSubsidioAction()	La función permite cargar los datos de tipo de subsidio cargando a su vez las incidencias que son tipos de subsidios y los conceptos de pago correspondiente a cada uno en un rango.
getSubsidiadoPorAction()	La función permite cargar los diferentes motivos de subsidios por el cual va a hacer subsidiado el trabajador ejemplo (enfermedad, licencia de maternidad, licencia deportiva y pensión).
adicionarTipoSubsidioAction()	La función adiciona un tipo de subsidio como una incidencia y luego se adiciona como un subsidio.
cargarConceptoPagoAction()	La función carga todos los conceptos de pago del modulo Concepto de Pago.
eliminarSubsidioAction()	La función le establece una fecha de eliminación en la tabla incidencias en la base de datos concibiéndose como eliminado el subsidio.
modificarTipoSubsidioAction()	La función transforma los datos referentes a un tipo de subsidio.
imprimirAction()	La función imprime los tipos de subsidios.
obtenerFormatoAction()	La función obtiene un formato de impresión que se estableció previamente.

Clases Model: Estas clases manejan la información persistente que poseen una larga vida, conceptos y sucesos que ocurren en el mundo real.

Tabla 2: Descripción de la clase DatDiagnosticoModel.

Nombre: DatDiagnosticoModel	
Tipo de clase: Model	
Atributo:	**Tipo:**
diagnostico	Objetos de tipo DatDiagnostico
Para cada responsabilidad:	

Nombre:	**Descripción:**
insertar($DatDiagnostico)	La función obtiene por parámetros un objeto y hace un llamado al método insertar de la clase DatDiagnostico del dominio.
actualizar($object)	La función obtiene por parámetros un objeto y hace un llamado al método actualizar de la clase DatDiagnostico del dominio.
cantidad()	La función hace un llamado al método cantidad de la clase DatDiagnostico del dominio.
eliminar($iddiagnostico)	La función obtiene por parámetros un idDiagnostico y hace un llamado al método eliminar de la clase DatDiagnostico del dominio.
obtenerDiagnosticos($limit, $start)	La función obtiene por parámetros un rango y hace un llamado al método getPorLimite clase DatDiagnostico del dominio.
buscarSiExiste($obj)	La función obtiene por parámetros un objeto y hace un llamado al método buscarExistente clase DatDiagnostico del dominio.
getDiagnosticos()	La función hace un llamado al método getTodos de la clase DatDiagnostico del dominio.
imprimir()	La función obtiene del perfil de la estructura el código y la denominación que se está trabajando en ese momento.
getUser($user = null)	La función obtiene todos los datos del perfil del usuario que esta autenticado actualmente.
obtenerFormato()	Obtiene todos los formatos del generador de reporte del sistema CEDRUX.

Tabla 3: Descripción de la clase DatDiagnostico.

Nombre: DatDiagnostico	
Tipo de clase: Domain	
Para cada responsabilidad:	
Nombre:	**Descripción:**

buscar($iddiagnostico)	La función permite buscar un diagnostico por su identificador.
getTodos()	La función devuelve todos los diagnósticos que existen en la base de datos.
getPorLimite($limite,$inicio)	La función devuelve todos los diagnósticos de acuerdo a un rango.
insertar($datdiagnostico)	La función obtiene como parámetro un objeto insertándolo en la base de datos.
actualizar($object)	La función obtiene como parámetro un objeto actualizándolo en la base de datos.
eliminar($iddiagnostico)	La función obtiene por parámetros un identificador de diagnostico y hace un llamado al método eliminar de la clase DatDiagnostico del dominio.
cantidad()	La función devuelve la cantidad de diagnósticos q existen en la base de datos.
buscarExistente($params)	La función devuelve dado una denominación si el diagnostico se encuentra en la base de datos.

El resto de las descripciones se pueden consultar en él: ver Anexo 2.

2.10 Conclusiones del capítulo:

A partir de los requisitos identificados por los Analistas y luego de un análisis realizado a los mismos en conjunto con el diseño propuesto se logró la realización del presente capítulo. Siendo el punto de partida para lograr la implementación del componente Subsidio, con el fin de responder a las funcionalidades claves de su gestión.

CAPITULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

1.1 Introducción:

La calidad de un producto de software; conjunto de cualidades que lo caracterizan y que determinan su utilidad y existencia; se ha convertido en un elemento estratégico de las grandes organizaciones debido a su fuerte impacto en la competitividad de las empresas.

Durante el proceso de desarrollo de software las posibilidades de errores son múltiples, estas pueden aparecer desde la misma especificación de requisitos donde se define lo que el sistema debe hacer. Como elementos críticos aparecen entonces la prueba y la validación de los resultados, estas en lugar de efectuarse una vez desarrollado el software, se llevan a cabo en cada una de las etapas de desarrollo para detectar a tiempo las imperfecciones e irregularidades y proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.

3.2 Validación del diseño propuesto:

La aplicación de métricas al diseño y la implementación de un producto de software constituyen un elemento fundamental a la hora de evaluar la calidad del mismo. “Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes.” (31)

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

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

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

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

Acoplamiento: Dependencia o interconexión de una clase o estructura de clase respecto a otras.

Cantidad de pruebas: Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

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

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

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

- **Resultados del instrumento de evaluación de la métrica Tamaño operacional de clase (TOC).**

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

Figura 17 Resultados de la Tamaño operacional de clase (TOC). definidos.

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos:

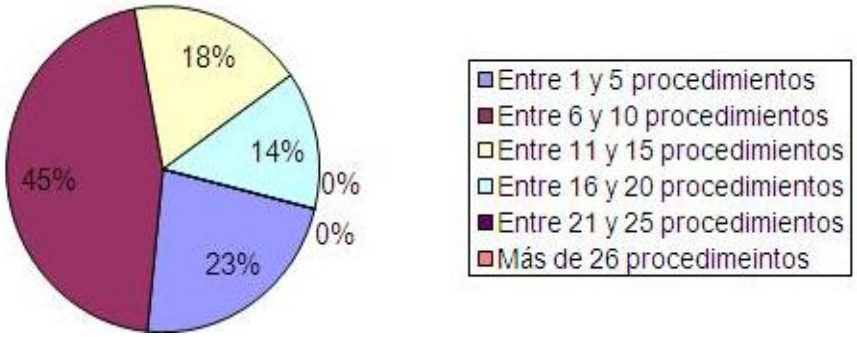

Figura 18 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad:

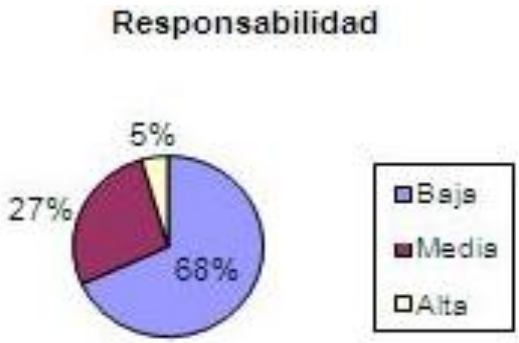

Figura 19 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación:

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

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización:

Figura 21 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para el componente Subsidio está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (86%) posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel satisfactorio en el 68% de las clases; de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

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

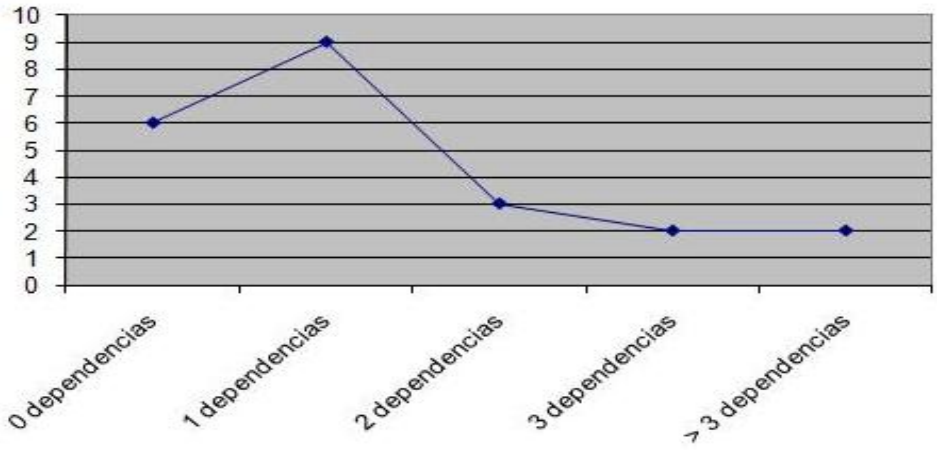

Figura 22 Evaluación de la métrica Relaciones entre Clases (RC).

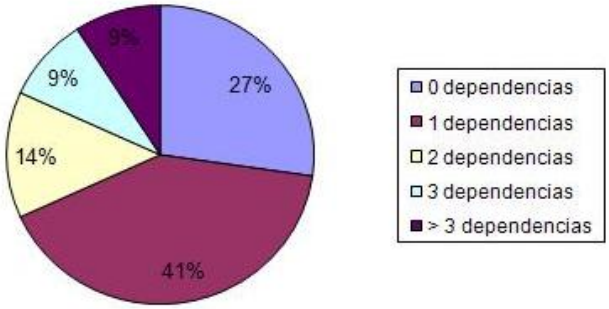

Figura 23 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

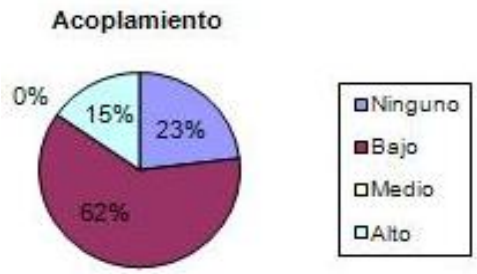

Figura 24 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento:

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

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas:

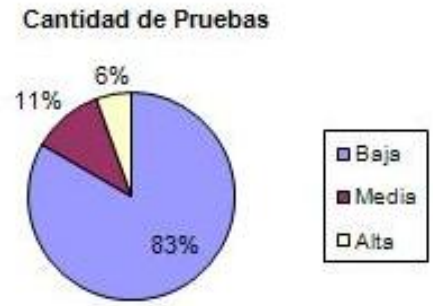

Figura 26 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización:

Figura 27 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para el componente Subsidio está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (86%) poseen menos de 3 dependencias respecto a otras.

Los atributos de calidad se encuentran en un nivel satisfactorio; en el 62% de las clases el grado de dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento y la Cantidad de Pruebas es de un 83% y la Reutilización se comporta favorablemente para un 83% de las clases.

3.3 Pruebas de Software:

Al conjunto de técnicas experimentales para la búsqueda de fallos en los programas, que determinan en cierto grado la calidad de un producto, se les denomina pruebas de software. La competencia mundial en el ámbito informático exige productos de calidad, de esta forma se muestra la necesidad de contar con pruebas de este tipo desde las primeras etapas de concepción de un proyecto.

"La prueba no puede asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software." (32)

- **Niveles de Prueba:**

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

Pruebas de Unidad:

Prueba individual a las unidades separadas de un sistema de software.

Pruebas de Integración:

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

Pruebas del Sistema:

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

Pruebas de Aceptación:

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

- **Métodos de Prueba:**

Enfoque sistemático, independiente del nivel en que se enmarque la prueba, que ayuda a encontrar buenos conjuntos de casos de prueba para detectar diferentes tipos de errores. (33)

Dos enfoques alternativos:

Caja Negra:

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

Caja Blanca:

Se comprueban los componentes internos.

3.3.1 Objetivo:

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

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

Con la idea de diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. (34)

3.3.2 Alcance:

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

3.4 Prueba de Software que será aplicada al componente:

3.4.1 Descripción general de las pruebas para el nivel de Unidad:

Se le denomina Prueba de Unidad al proceso de separar y probar el correcto funcionamiento de las partes individuales de un programa para asegurar que cada uno de estos se ejecuta correctamente por separado.

Antes de iniciar cualquier otra prueba es preciso probar el flujo de datos de la interfaz del módulo. Si los datos no fluyen correctamente, todas las demás pruebas no tienen sentido.

3.4.1.1 Descripción y aplicación de la Prueba de Caja Blanca o Estructural:

- **Descripción:**

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

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

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

Prueba del Camino Básico:

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

Prueba de Condición:

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

Prueba de Flujo de Datos:

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

Prueba de Bucles:

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

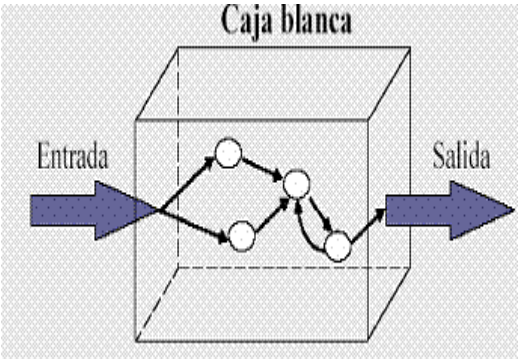

Figura 28 Caja Blanca.

- **Aplicación:**

Según la descripción de la prueba de caja blanca presentada con anterioridad y a partir de la necesidad de crear un producto de alta calidad es preciso valorar qué tan certera ha sido la implementación del componente Subsidio y para ello es necesario aplicar una de las técnicas que esta comprende, en este caso la del camino básico.

Para ello es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática. Se debe comenzar por un análisis del código, posteriormente son enumeradas cada una de las instrucciones, se construye el grafo de flujo asociado y según de las fórmulas pertinentes se calcula dicha complejidad:

A continuación se analizan y enumeran las sentencias de código de uno de los procedimientos contenidos en la clase GestionardiagnosticoController, específicamente: adicionarDiagnostico, este algoritmo se encarga de insertar los diferentes diagnósticos que se quieren registrar en la aplicación. La funcionalidad es invocada para contribuir al pago del subsidio.

```
function adicionarDiagnosticoAction() {
    $obj->denom = $this->_request->getPost ('tfDenom');1
            $aux = $this->diagnostico->BuscarSiExiste($obj);2
            if($aux){3
             echo( '{"codMsg":0}');4
            }5
            else{6
    $result = $this->diagnostico->Insertar($obj);7
    if($result){8
            echo ( '{"codMsg":1}');9
            }10
            else{11
            echo( '{"codMsg":2}');12
            }13
        }14
}15
```

Figura 29 Sentencia de código.

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

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

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

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

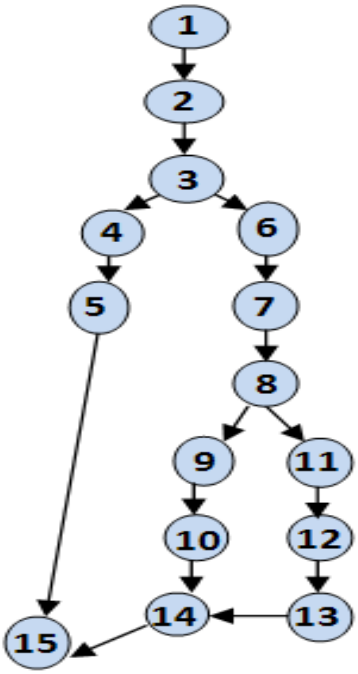

Figura 30 Grafo de flujo asociado al código.

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres vías o fórmulas de manera tal que quede justificado el resultado, siendo el mismo en cada caso:

1. V (G) = (A – N) + 2

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

V (G) = (16 -15) + 2

V (G) = 3.

2. V (G) = P + 1

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

V (G) = 2 + 1

V (G) = 3.

3. V (G) = R

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

V (G) = 3.

El cálculo efectuado mediante las fórmulas antes presentadas muestran una complejidad ciclomática de valor 3, de manera que existen tres posibles caminos por donde el flujo puede circular, este valor representa el número mínimo de casos de pruebas para el procedimiento tratado.

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

Camino básico # 1: 1-2-3-4-5-15

Camino básico # 2: 1-2-3-6-7-8-9-10-14-15

Camino básico # 3: 1-2-3-6-7-8-11-12-13-14-15

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

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

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

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

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

Caso de prueba para el camino básico # 1.

Descripción: El dato de entrada coincide con la denominación del diagnóstico: tfDenom.

Condición de ejecución: La tfDenom será igual a Fractura de pierna.

Entrada: tfDenom: Fractura de pierna.

Resultados esperados: Se espera que el diagnóstico sea insertado satisfactoriamente.

No se pudo adicionar el diagnóstico porque ya fue registrado en el sistema.

Caso de prueba para el camino básico # 2.

Descripción: El dato de entrada coincide con la denominación del diagnóstico: tfDenom.

Condición de ejecución: La tfDenom será igual a Fractura de pierna.

Entrada: tfDenom: Fractura de pierna.

Resultados esperados: Se espera que el diagnóstico sea insertado satisfactoriamente.

El diagnóstico fue adicionado satisfactoriamente en el sistema.

Caso de prueba para el camino básico # 3.

Descripción: El dato de entrada coincide con la denominación del diagnóstico: tfDenom.
Condición de ejecución: La tfDenom será igual a Fractura de pierna.
Entrada: tfDenom: Fractura de pierna.
Resultados esperados: Se espera que el diagnóstico sea insertado satisfactoriamente.
No se pudo adicionar el diagnóstico porque existen campos con valores incorrectos.

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

3.4.1.2 Descripción y aplicación de la Prueba de Caja Negra o Funcional:

- **Descripción:**

A este tipo de prueba también se le conoce como Prueba de Caja Opaca o Inducida por los Datos. Se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación, esta prueba se limita a brindar solo datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo internamente, es decir, solo trabaja sobre su interfaz externa.
En esencia permite encontrar:
Funciones incorrectas o ausentes.
Errores de interfaz.
Errores en estructuras de datos o en accesos a las bases de datos externas.
Errores de rendimiento.
Errores de inicialización y terminación.
Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación:
Partición de Equivalencia:
Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
Análisis de Valores Límites:
Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
Grafos de Causa-Efecto:
Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

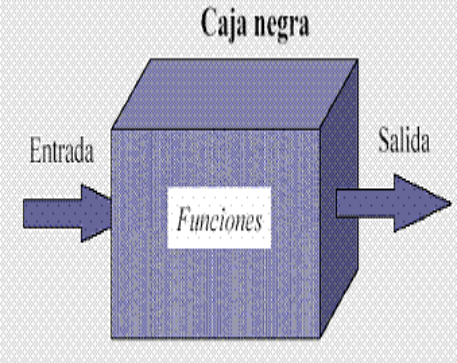

Figura 31 Caja Negra.

- **Aplicación:**

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 Subsidios.

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. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real del dato.

Luego de aplicados los métodos de prueba por el equipo de desarrollo al procedimiento adicionarDiagnostico y al requisito funcional Gestionar subsidio antes presentados, es válido señalar que los resultados obtenidos hasta el momento han sido 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).

3.5 Conclusiones del capítulo:

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

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

CONCLUSIONES GENERALES.

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

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

Esta propuesta exhibe valor técnico, donde se destaca la incorporación de principios por los que se mide la factibilidad de un diseño de software, ejemplo: la utilización de patrones que posibilita la reutilización, garantizando la sostenibilidad y mantenimiento del sistema.

La solución propuesta es novedosa, su importancia radica en la realización de los procesos referentes a la gestión del subsidio dentro de un único componente.

RECOMENDACIONES

Al concluir el trabajo de diploma**,** considerando cumplidos los objetivos trazados en el mismo, se recomienda:

- Continuar realizando pruebas de calidad al componente.
- Optimizar los algoritmos para un mejor funcionamiento interno del componente.
- Profundizar en temas referentes al Subsidio para detectar posibles debilidades en el componente y agregar mejoras al mismo.

REFERENCIAS BIBLIOGRÁFICAS

1. Documento Visión2. http://definicion.de/subsidio/

2. Definicion.de [Online] 2011. http://definicion.de/subsidio/

3. http://www.mitecnologico.com/Main/ConceptoDeSeguridadSocial

4. Trabajo de diploma Diagnóstico del Clima Laboral en la Empresa "Torrefactora de Café Regil

5. EVOLUCIÓN, CONCEPTOS Y DIFERENTES PERSPECTIVAS VISTAS EN LA REALIDAD CUBANAhttp://www.gestiopolis.com/dirgp/rec/otros.htm.

6. ERP Openbravo. [Online] [Cited: enero 12, 2011.] http://www.openbravo.com/es/.

7. El ERP de SAP: R/3. [Online] [Cited: enero 12, 2011.] http://www.tuobra.unam.mx/publicadas/040702105342-El.html.

8. SISCONT. Software Contable Financiero. [Online] [Cited: enero 13, 2011.] http://www.siscont.com/DESCRIPTIVO%20SISCONT.pdf.

9. **Romero, Lorely Moya.** Funcionalidades y principales opciones del módulo de planificación del software integrado Versat Sarasola. [Online] [Cited: enero 10, 2011.] http://www.eumed.net/cursecon/ecolat/cu/2009/lmr2.h.

10. Integrado VerSat Sarasola. [Online] [Cited: enero 16, 2011.] http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc.

11. **Alvarez, Sara.** *Desarrollo Web.* [Online] [Cited: enero 15, 2011.] http://www.desarrolloweb.com/articulos/2477.php.

12. desarrolloweb.com. . [Online] 2007. [Cited: enero 15, 2011.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

13. GNU Operating System. [Online] 2009. [Cited: 2 febrero, 2011.] http://www.gnu.org/philosophy/free-sw.html.

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

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

16. 2010**.** Desarrollo web. [En línea] 2010. [Citado el: 02 de 15 de 2011.] http://www.desarrolloweb.com/articulos/392.php.

17. Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2007. Desarrollo de Software Basado en Componentes. Málaga, Spain : s.n., 2007.

18. 2009. debug_mode=on. [En línea] 2009. [Citado el: 10 de 03 de 2010.] http://es.debugmodeon.com/articulo/el-patron-mvc.

19. Ing. Joisel Pérez Pérez, Ing.Enrique Chaviano Gómez, Ing. Donel Vázquez Zambrano. 2009. Diseño de solución informática para la gestión y control de los costos en las entidades. Ciudad de La Habana : s.n., 2009

20. Hypertext Preprocessor. [Online] [Cited: febrero 7, 2011.] http://php.net/index.php.

21. Lenguaje HTML. [Online] [Cited: enero 23, 2011.] http://www.desarrolloweb.com/articulos/711.php.

22. Extensible Markup Language (XML). [Online] [Cited: enero 23, 2011.] http://www.w3.org/XML/

23. territoriopc. [Online] [Cited: febrero 5, 2011.] http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php

24. ZEND FRAMEWORK. [Online] [Cited: febrero 10, 2011.] http://framework.zend.com/manual/en/

25. Doctrine. . [Online] [Cited: enero 25, 2011.] http://www.doctrine-project.org.

26. AbartiaTeam. [Online] 2006. [Cited: febrero 8, 2011.] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

27. Subversion. [Online] 2010. [Cited: enero 26, 2011.] 01 de 2010.] http://subversion.apache.org.

28. *Zend Studio - The Professional PHP IDE – Zend. .* [Online] 2010. [Cited: febrero 9, 2011.] http://www.zend.com/products/studio.

29. *Documentación del Servidor HTTP Apache 2.0.* [Online] 2009. [Cited: febrero 11, 2011.] http://httpd.apache.org/docs/2.0/es/.

30. mozilla europe. [Online] [Cited: enero 26, 2011.] http://www.mozilla-europe.org/es/firefox/.

31. **Aylienn Aquino Leiva, Yasser Linares Domínguez. 2009.** Implementación del módulo de Contabilidad General del Sistema Integral de Gestión Cedrux. Cuba: s.n., 2009.

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

33. **Ramírez, Jaime.** Unidad de Programación. Métodos de Prueba del Software.

34. **Pérez Hernández, Yorji. 2008.** Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. Cuba: s.n., 2008.

BIBLIOGRAFÍA

Rolando Alfredo Hernández, Sayda Coello González. *El paradigma cuantitativo de la investigación científica.*

Mañas, J. A. 1994. Pruebas de programas. Recuperado. [En línea] 1994. [Citado el: 7 de 04 de 2010.] http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s22.

Pélaez, Juan. 2001. [En línea] Octubre de 2001. [Citado el: 15 de 02 de 2010.] http://geeks.ms/blogs/jkpelaez/archive/2010/04/18/arquitectura-basada-en-componentes.aspx.

2010.], [En línea] . [Citado el: 4 de Febrero de. 1998. Firefox web browser | Faster, more secure, & customizable. [En línea] 1998. [Citado el: 4 de 02 de 2010.] http://www.mozilla.com/en-US/firefox/firefox.html.

2010. Pruebas de Software. [En línea] 2010. [Citado el: 6 de 04 de 2010.] http://lsi.ugr.es/~ig1/docis/pruso.pdf.

2001. José Enrique González Cornejo. *Arquitectura en Capas ~ DNA Un camino hacia los procesos distribuidos.* [En línea] 2001. [Citado el: 15 de 02 de 2010.] http://www.arquitectura_tres_capas.com.

2009. Businesscol.com. [En línea] 2009. [Citado el: 13 de 01 de 2010.] http://www.businesscol.com/productos/glosarios/economico/glossary.php?word=CENTRO%20DE%20COSTOS.

Miguel P. Cabrera González, Guillermo Obregón Rodríguez, Margarita Cárdenas Negrin, Luis Mario Carralero Silva. XV FORUM DE CIENCIA Y TECNICA. SISTEMA ECONOMICO INTEGRADO

Sara Alvarez. 2006. Desarrollo Web. [En línea] 2006. [Citado el: 15 de 02 de 2010.] http://www.desarrolloweb.com/articulos/2477.php

2006. SOA. [En línea] 03 de 2006. [Citado el: 20 de 02 de 2010.] http://arquitecturaorientadaaservicios.blogspot.com/2006/03/pero-qu-es-realmente-soa.html.

2009. GNU Operating System. [En línea] 2009. [Citado el: 16 de 02 de 2010.] http://www.gnu.org/philosophy/free-sw.html. **Yanet Vega, L. S. 2009.** Definición del ciclo de vida del proyecto. Habana : s.n., 2009.

2010. Desarrollo web. [En línea] 2010. [Citado el: 02 de 15 de 2010.] http://www.desarrolloweb.com/articulos/392.php.

Lidia Fuentes, José M. Troya y Antonio Vallecillo. 2007. Desarrollo de Software Basado en Componentes. Málaga, Spain : s.n., 2007.

2009. debug_mode=on. [En línea] 2009. [Citado el: 10 de 03 de 2010.] http://es.debugmodeon.com/articulo/el-patron-mvc.

Prieto, Félix. 2009. Patrones de diseño. 2009.
2010. Hypertext Preprocessor. [En línea] 2010. [Citado el: 01 de 15 de 2010.] http://php.net/index.php.

Mini Diccionario Informático. [En línea] [Citado el: 17 de abril de 2011.] http://www.carlospes.com/minidiccionario/especificacion_requisitos.php.

software, Departamento de ingeniería de. Conferencias de ingeniería de requisitos.

Estándar IEEE para la metodología de métrica en la calidad del software.

Cunha, Gabriela Elisa da. Métricas de software.

2010. DISEÑO DE BASES DE DATOS RELACIONALES. [En línea] 2010. http://usuarios.multimania.es/cursosgbd/UD4.htm.

2010. PostgreSQL: The world's most advanced open source database Postgresql. [En línea] 2010. [Citado el: 02 de 02 de 2010.] http://www.postgresql.org.

Documento Visión2. http://definicion.de/subsidio/

Definicion.de [Online] 2011. http://definicion.de/subsidio/

Trabajo de diploma Diagnóstico del Clima Laboral en la Empresa “Torrefactora de Café Regil

EVOLUCIÓN, CONCEPTOS Y DIFERENTES PERSPECTIVAS VISTAS EN LA REALIDAD CUBANAhttp://www.gestiopolis.com/dirgp/rec/otros.htm.

Romero, Lorely Moya. Funcionalidades y principales opciones del módulo de planificación del software integrado Versat Sarasola. [Online] [Cited: enero 10, 2011.] http://www.eumed.net/cursecon/ecolat/cu/2009/lmr2.h.

Integrado VerSat Sarasola. [Online] [Cited: enero 16, 2011.] http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc.

GLOSARIO DE TÉRMINO.

1. ERP: Es un sistema compuesto por un conjunto de módulos funcionales estándar y que son susceptibles de ser adaptados a las necesidades de cada empresa.

2. Sistema gestión: Un sistema de gestión es una estructura probada para la gestión y mejora continua de las políticas, los procedimientos y procesos de la organización.

3. Sistema informático: Conjunto de elementos que hacen posible el tratamiento automático de la información.

4.Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente

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

6. 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

7. 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.

8. Componente: Un componente es una clase de uso específico, que puede ser configurada o utilizada de forma visual desde el entorno de desarrollo.

9. 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.

ANEXO

Anexo 2 Descripción de clases.

Anexo 2.1

Tabla 4 Descripción de la clase GestionarDoctorController

Nombre: GestionarDoctorController	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
doctor	Objetos de tipo NomDoctorModel
certificados	Objetos de tipo DatRegistroCertificadoMedicosModel
Para cada responsabilidad:	
Nombre:	**Descripción:**
gestionarDoctorAction()	La función permite establecer la comunicación entre la vista (ExtJs) y la clase controladora Gestionar Doctor.
cargarComboSexoAction()	La función carga en el combobox los sexos que se encuentran en la base datos.
cargarDatosDoctorAction()	La función carga todos los datos de un doctor.
adicionarDoctorAction()	La función adiciona un doctor en la tabla persona de la base datos y luego se adiciona como un doctor.
modificarDoctorAction()	La función transforma los datos referentes a un doctor en la tabla persona.
eliminarDoctorAction()	La función le establece una fecha de eliminación en la tabla persona en la base de datos concibiéndose como eliminado el doctor.
imprimirAction()	La función imprime los tipos de doctores.
obtenerFormatoAction()	La función obtiene un formato de impresión que se estableció previamente.

Anexo 2.2

Tabla 5 Descripción de la clase NomHospitalModel

Nombre: NomHospitalModel

Tipo de clase: Model	
Atributo:	**Tipo:**
hospital	Objetos de tipo NomHospital
tipoCentro	Objetos de tipo NomTipoHospitalModel
Para cada responsabilidad:	
Nombre:	**Descripción:**
insertar($obj)	La función obtiene por parámetros un objeto y hace un llamado al método insertar de la clase NomHospital del dominio.
getPorLimite($params,$limite,$inicio)	La función obtiene por parámetros un parámetro y un rango además hace un llamado al método getPorLimite de la clase NomHospital del dominio.
actualizar($obj)	La función obtiene por parámetros un objeto y hace un llamado al método actualizar de la clase NomHospital del dominio.
eliminar($obj)	La función obtiene por parámetros un objeto y hace un llamado al método eliminar de la clase NomHospital del dominio.
getTodos()	La función hace un llamado al método getTodos de la clase NomHospital del dominio.
cargarTipoCentro()	La función hace un llamado al método getTodos de la clase NomTipoHospital del dominio.
buscarSiExiste($params)	La función obtiene por parámetros un parámetro y hace un llamado al método buscarExistente de la clase NomHospital del dominio.
cantidad()	La función hace un llamado al método cantidad de la clase NomHospital del dominio.
imprimir()	La función obtiene del perfil de la estructura el código y la denominación que se está trabajando en ese momento.
getUser($user = null)	La función obtiene todos los datos del perfil del usuario que esta autenticado actualmente.
obtenerFormato()	Obtiene todos los formatos del generador de reporte del

	sistema CEDRUX.

Anexo 2.3

Tabla 6 Descripción de la clase NomSubsidiadoPorModel

Nombre: NomSubsidiadoPorModel	
Tipo de clase: Model	
Atributo:	**Tipo:**
subsidiadoPor	
Para cada responsabilidad:	
Nombre:	**Descripción:**
getSubsidiadoPor()	La función hace un llamado al método getSubsidiadoPor de la clase NomSubsidiadoPor del dominio.
buscarSubsidioPor($idsubsidiadopor)	La función obtiene por parámetros un identificador de subsidio y hace un llamado al método buscar de la clase NomSubsidiadoPor del dominio.

Anexo 2.4

Tabla 7 Descripción de la clase NomSubsidiadoPor

Nombre: NomSubsidiadoPor.	
Tipo de clase: Domain	
Para cada responsabilidad:	
Nombre:	**Descripción:**
buscar($idsubsidiadopor)	La función permite dado un identificador de subsidiado por permite buscar el nombre de ese subsidiado en la base de datos.
getTodos()	La función devuelve todos los nombres de subsidiados que existen en la base de datos.
getPorLimite($limite = 20,$inicio = 0)	La función devuelve todos los nombres de subsidiados de acuerdo a un rango.
getSubsidiadoPor()	La función devuelve todos los nombres de subsidiados.

Anexo 2.5

Tabla 8 Descripción de la clase NomTipoHospital

Nombre: NomTipoHospital	
Tipo de clase: Domain	
Para cada responsabilidad:	
Nombre:	**Descripción:**
buscar($idhospital)	La función permite dado un identificador de hospital permite buscar el nombre de ese tipo de hospital en la base de datos.
getTodos()	La función devuelve todos los nombres de tipo de hospital que existen en la base de datos.
getPorLimite($limite = 20,$inicio = 0)	La función devuelve todos los nombres de tipo de hospital de acuerdo a un rango.