Universidad de las Ciencias Informáticas

Facultad 3

Diseño e Implementación de los componentes Modelo, Plan, y Comentario del subsistema de Planificación del Sistema CEDRUX.

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

Autor(es): Yeliem Gutierrez Martinez
Julio Denis Ortega Lago

Tutor(es): Ing. Osvaldo Díaz Verdecia.
1er Tte. Ing. Dionisdel Ponce Santana.
Co-Tutor(es): Ing. Danelys Brito González.

La Habana, junio de 2011.
"Año 53 de la Revolución"

"Si no existe la organización, las ideas, después del primer momento de impulso, van perdiendo eficacia."

Ernesto Guevara

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo “Diseño e implementación de los componentes Modelo, Plan y Comentario del Subsistema Planificación del Sistema CEDRUX” y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Yeliem Gutierrez Martinez

_____

Firma del Autor

Julio Denis Ortega Lago

_____

Firma del Autor

Ing. Osvaldo Díaz Verdecia

_____

Firma del Tutor

1er Tte. Ing. Dionisdel Ponce Santana

_____

Firma del Tutor

Ing. Danelys Brito González.

_____

Firma del Co-Tutor

Agradecimientos

Dedicatoria

Resumen

En el presente trabajo se describe el proceso de realización de planes, modelos y comentarios, perteneciente al subsistema de Planificación del Sistema Integral de Gestión Cedrux, el cual integra y automatiza muchas de las actividades asociadas al negocio de planificación. La implantación de estos componentes tiene el objetivo de mejorar el procedimiento y de lograr una mayor organización en las entidades cubanas. Actualmente, la complejidad de los procesos de planificación es muy elevada, por lo que se realiza un estudio profundo de cada uno de los artefactos de la fase de análisis y se desarrolla un sistema que cumpla con las exigencias de los clientes. En este documento se plasman los resultados del trabajo investigativo realizado, mediante el cual se obtuvo el diseño y la implementación de los componentes propuestos. Todo esto se realiza siguiendo el modelo de desarrollo definido en la Unidad de Compatibilización, Integración y Desarrollo de Software para la Defensa (UCID) y cumpliendo con las normas y estándares que este contiene.

Palabras claves:

Componente, Comentario, Modelo, Negocio, Procesos, Plan, Planificación.

Índice

Índice

Introducción

A lo largo de la historia, el hombre siempre ha querido simplificar su modo de vida, buscando tener herramientas que le ayudarán a efectuar cálculos precisos y rápidos. Hoy en día, el desarrollo de las empresas demanda gran cantidad de información, las empresas están obligadas a tomar decisiones cada vez más precisas y con mayor rapidez.

La planificación es un proceso donde se definen las acciones a seguir, y por el que se establece el esfuerzo necesario para cumplir con los objetivos planteados en un proyecto durante un tiempo determinado.

La planificación empresarial ha constituido y constituye un poderoso instrumento para potenciar e incidir sobre el uso racional de los recursos materiales y financieros del país. Es un complejo trabajo que debe ser realizado profesionalmente en cada uno de los niveles de la empresa donde los directivos determinan el futuro, las acciones y recursos que se necesitan para realizar y alcanzar las metas, que permite fijar adecuadamente los objetivos, determinar las acciones, las estrategias, las prioridades y calcular la inversión.

Por tanto una mala planificación en cualquier entidad puede presentarse como una inversión desacertada donde se gasta más de lo que se recibe, el establecimiento de políticas de precio inadecuadas donde el producto se vende más barato que el costo real de producción, o el atraso de la producción lo cual provocaría incumplir con los contratos firmados con el cliente. Esto traería graves consecuencias como la pérdida de recursos que en la mayoría de los casos no son recuperables.

Según la Msc. Norma Sánchez Paz, en la economía cubana se realizan dos planificaciones, la empresarial y la presupuestaria. El objetivo de la planificación empresarial es conformar el plan anual de la empresa a partir de un conjunto de actividades interrelacionadas, mientras que la planificación presupuestaria se encarga de planificar y controlar el presupuesto de las Unidades Presupuestadas (UP), que son las entidades a través de las cuales se organiza la prestación de servicios y que reciben el financiamiento total del presupuesto del estado. (1)

En el mundo existen varios sistemas dedicados a la gestión de recursos de las empresas, pero estos sistemas, generalmente están desarrollados sobre tecnologías y herramientas privativas, por lo que resultaría una pérdida económica para el país, ya que sería necesario el pago de una licencia para poder hacer uso de los mismos. Además, estos programas no tienen en cuenta las particularidades de la economía cubana.

En Cuba también existen diversos sistemas para la planificación de los recursos en una entidad, pero no existe un estándar en las empresas a la hora de realizar el proceso de planificación, esto dificulta el flujo de información entre las entidades y como consecuencia, proporciona un aumento de la pérdida de los datos. Debido a que no existe un sistema integral para este proceso que además logre la gestión de los modelos, planes y comentarios, se dificulta la centralización, normalización, y control de los recursos en las entidades del país. Como consecuencia a estas insuficiencias para realizar una planificación eficaz en el país, se decidió que era necesario el desarrollo de un software adecuado para las entidades cubanas.

Por otra parte, en varias de las entidades del país, el proceso de planificación se efectúa utilizando herramientas ofimáticas de apoyo, como las hojas de cálculo de Microsoft Office (Excel), que aunque son útiles, han demostrado ser ineficientes, debido a que requieren un alto nivel operacional por parte del usuario, lo que significa que el personal debe tener conocimientos avanzados sobre las herramientas ofimáticas provocando un mayor tiempo de duración del proceso de planificación. También esta herramienta tiene capacidad limitada para almacenar información, y no garantiza la consistencia, integridad y seguridad de la misma, debido a que los datos pueden ser modificados fácilmente aunque se pueden usar protecciones, pero no son suficientemente seguras. Además, se tiene que guardar la información cada cierto tiempo, para garantizar que ante cualquier falla no se pierdan los datos que se están gestionando. Por tanto la planificación resulta sumamente engorrosa e ineficiente debido a la gran dispersión ante grandes flujos de información.

A partir del 2008 se desarrolla en la Universidad de las Ciencias Informáticas (UCI) un producto para la gestión de entidades que lleva por nombre CEDRUX[1], integrado por un conjunto de subsistemas: logística, contabilidad, Planificación, entre otros; que propiciará un ambiente empresarial que contribuya a tomar decisiones cada vez más precisas a partir del análisis de la información, siendo utilizado en cada una de las entidades empresariales y presupuestarias del país, como un sistema único capaz de informatizar los procesos de gestión en las empresas.

El subsistema de Planificación es el encargado de facilitar a las entidades, la preparación y presentación de los elementos para elaborar, ejecutar y controlar el plan y presupuesto anual. Además en este subsistema se centralizan todos los componentes que tienen lugar dentro del proceso de planificación. Algunos de los componentes básicos son:

[1] CEDRUX: CEDR proviene de Cedro, planta de un tronco grueso, estable, robusto, cualidades que se pretende cumplir con el sistema; la terminación UX viene del sistema operativo Linux, con el objetivo de destacar su carácter de código libre.

- Plan: permitirá la gestión de la actividad económica de una empresa, un sector o una región; y además se compone por modelos que establecen el destino que se les dará a los recursos financieros de una empresa.
- Modelo: permitirá la organización, en forma de plantilla, de los indicadores y columnas.
- Comentario: manejará la gestión de observaciones que se le hagan a los planes y modelos, para facilitar la comprensión de los mismos.

Con este subsistema se incluyen de manera más detallada los requisitos que se deben tener en cuenta a la hora de gestionar cada uno de los componentes, los cuales ya pasaron por una etapa de análisis pero no están diseñadas ni implementadas sus funcionalidades.

Los esfuerzos estarán encaminados en resolver el siguiente **problema de la investigación**: ¿Cómo contribuir a la centralización, normalización y control de los recursos en las entidades cubanas?

Se define como **objeto de estudio:** los procesos de planificación.

Para el desarrollo de la investigación se definió como **objetivo general de investigación**: Desarrollar el diseño e implementación de los componentes: Modelo, Plan y Comentario del subsistema Planificación del Sistema Integral de Gestión CEDRUX, que contribuyan a la centralización, normalización y control de los recursos en las entidades cubanas.

El **campo de acción** está encaminado a: los procesos de planificación empresarial y presupuestaria en el Sistema Integral de Gestión CEDRUX.

Para resolver el problema de la investigación planteado se presenta la siguiente **idea a defender**: Si se diseñan e implementan los componentes que permitan gestionar los planes, modelos y comentarios del subsistema Planificación del sistema CEDRUX, se contribuirá a la centralización, normalización y control de los recursos en las entidades cubanas.

Objetivos específicos

- Fundamentar la investigación a partir del estudio del estado del arte.
- Diseñar una solución en consecuencia con el análisis.
- Implementar los componentes del subsistema.
- Validar la funcionalidad de los componentes.

Tareas de investigación

- Describir los procesos de planificación que se desarrollan en una entidad.

- Valorar la experiencia nacional e internacional de la aplicación de los Subsistemas de Gestión de Entidades.
- Valorar las soluciones informáticas asociadas a la planificación: Versat Sarasola, Presupuesto Maestro, OpenBravo, SAGE, ERP SAP.
- Valorar los procesos y la documentación generada por los analistas.
- Realizar el diseño de clases de los componentes Plan, Modelo y Comentario.
- Validar el diseño de los componentes Plan, Modelo y Comentario mediante la aplicación de métricas para el diseño de clases.
- Implementar los componentes Plan, Modelo y Comentario.
- Validar los componentes Plan, Modelo y Comentario mediante pruebas de software, de caja blanca y caja negra.
- Formalizar los resultados obtenidos en el documento de tesis.

Para el correcto desarrollo del presente Trabajo de Diploma se emplearon métodos científicos de corte teórico.

Métodos teóricos:

- Histórico-Lógico: Se empleó para analizar la trayectoria y evolución que han tenido los diferentes sistemas para la planificación de los recursos en las empresas.
- Analítico-Sintético: Permitió el procesamiento de la información y arribar a las conclusiones prácticas y teóricas de la investigación.
- Modelación: Para la creación de abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados en la fase de diseño.

El trabajo de diploma está estructurado de la siguiente manera:

Capítulo 1: Fundamentación Teórica. Se realiza un estudio minucioso del proceso de planificación, así como el estado del arte de los sistemas existentes que automatizan dicho proceso, y las herramientas y tecnologías que se utilizan para el desarrollo del sistema.

Capítulo 2: Propuesta de solución del diseño e implementación de los componentes.

Capítulo 3: Validaciones realizadas en el diseño y la implementación.

Capítulo 1. Fundamentación Teórica

1.1 Introducción.

La planificación es un proceso gradual en el cual se definen las acciones a seguir, y por el que se establece el esfuerzo, y el tiempo necesario para cumplir con los objetivos de un proyecto.

A pesar de que en la planificación se definen todos los aspectos que se deben tener en cuenta para su correcto funcionamiento, puede que en un algún momento exista la necesidad de realizar cambios respecto a lo definido originalmente durante la ejecución, los cuales servirán de punto de partida para un nuevo análisis, y en caso de ser requerido, una nueva planificación.

La planificación se encuentra, dentro de la mayoría de las actividades de las personas, instituciones y organismos de toda índole.

Dentro de las características fundamentales que se presenta en el proceso de planificación, se puede mencionar el sentido de proceso, ya que es una actividad continua, con un reajuste permanente entre los medios, actividades, fines y procedimientos involucrados en la misma. El vínculo con el medio en que se desarrolla es otra de sus características, debido a que tiene en cuenta los diferentes factores sociales y contingentes que conforman el escenario donde se desarrolla. Planificar implica la selección de algunas soluciones entre una gama de opciones, sin embargo, más que una decisión única, la planificación es un conjunto de decisiones interrelacionadas y en progresión. Aunque este proceso no es de ejecución, siempre está dirigido hacia la realización de acciones, en las cuales se busca el logro de objetivos futuristas. El alcance de los objetivos propuestos, concretos, y definidos es una de las principales metas de la planificación, así como la búsqueda de los medios más eficientes para la realización de sus objetivos. (2)

La planificación empresarial es un proceso técnico, económico y organizativo en el que se establecen los objetivos y estrategias de la organización a corto y mediano plazo, y se definen las acciones y recursos para su cumplimiento de forma racional, constituyendo al mismo tiempo, un proceso político-ideológico que expresa la voluntad de priorizar el aporte de las empresas estatales a la sociedad cubana por encima de cualquier interés colectivo o individual y asegurar así el desarrollo de las empresas en correspondencia con los requisitos de la economía nacional.

El objetivo fundamental de la planificación empresarial, puede expresarse como la elaboración del sistema de planes económicos de la empresa, garantizando los más altos niveles de actividad, con la utilización

eficiente de la capacidad productiva y los recursos materiales, laborales y financieros disponibles, que den respuesta a las estrategias, políticas y programas de desarrollo económico y social de la nación y la empresa.

La previsión de unos resultados calculados en base a un plan de acción adoptado por la empresa según su estrategia y la forma de llevarlos a cabo es la finalidad de la planificación presupuestaria.

Este tipo de planificación es realizada tanto por la alta dirección, como por los departamentos, sus objetivos son a corto plazo, y la información manejada es principalmente interna. Además su control es presupuestario, y tiene una alta formalización.

En Cuba esta planificación debe constituir un instrumento óptimo de política económica y social, que garantiza un mejor y más racional uso de los recursos materiales, laborales y financieros que el estado pone a disposición de los sectores más sensibles.

En todas las etapas de la planificación presupuestaria, se utiliza el método de análisis económico, ya que sobre su base se determina el nivel de cumplimiento de los distintos indicadores económicos, estableciéndose comparaciones entre los datos reales y los planificados.

1.2. Sistemas Nacionales.

1.2.1. Versat Sarasola.

Es un sistema económico integrado constituido por 10 módulos o subsistemas que incluyen configuración y seguridad, contabilidad general y de gastos, costos y procesos, finanzas y caja. Además, en el proceso intervienen activos fijos, planificación y presupuestos, control de inventarios, pago de salario (nómina), facturación y generador de reportes.

Sus creadores definen al VERSAT-Sarasola como “un paquete integrado para la gestión económica financiera que permite enviar información eficaz, de forma inmediata, desde lugares apartados, a la vez que ofrece mayor organización, control y disciplina en cada gestión”. (3)

Se distingue por ser el primer sistema de contabilidad cubano certificado, según las nuevas normativas establecidas por los Ministerios de Finanzas y Precios (MFP) y de la Informática y las Comunicaciones (MIC), para este tipo de software. (3)

Características del VERSAT SARASOLA.

• Herramienta para la planificación económica, el control y el análisis de gestión.

• Diseñado para su empleo en cualquier tipo de entidad empresarial o presupuestada.

• Permite llevar el control y registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades, así como exponer el estado financiero y toda la información económica y contable en este universo.

• Se estructura en un grupo de subsistemas en los cuales se procesan y contabilizan los documentos primarios, donde se anotan los movimientos, los recursos materiales, laborales y financieros que se utilizan en una entidad.

• Se logra establecer un proceso de interacción usuario-sistema.

• Rapidez y fiabilidad, a partir de la configuración del proceso de contabilización de los documentos primarios y de las propias posibilidades de trabajo contenidas en cada subsistema. (3)

El proceso de planificación, con el uso de este sistema, comienza a partir de la definición de los codificadores (nomencladores) y la configuración del Subsistema, donde se trabaja sobre la versión oficial en el ejercicio seleccionado, capturando todas las proformas con la información de las distintas entidades vinculadas al presupuesto así como de la propia unidad contable. Luego de revisar la información confirma las proformas captadas y comprueba el reflejo de esta información en los presupuestos confeccionados y en los indicadores planificados, si se desea rectificar algún dato siempre se hará en el documento original y si ya estaba confirmado el cambio quedará automáticamente reajustado en el presupuesto. Posteriormente se realiza la validación de las restricciones impuestas a la versión tanto por normas como notificaciones y de estar todo correcto se puede realizar la exportación de la información agregada al nivel superior. (4)

La planificación con este sistema se ha concebido con una orientación a la esfera Presupuestada, sin dejar de tener posibilidades de generalización en el sector Empresarial. Predomina la terminología utilizada en el sector Presupuestado.

1.2.2. Presupuesto Maestro.

El “Presupuesto Maestro” le permite a las empresas conjugar integralmente todos los objetivos de trabajo de las distintas subdivisiones estructurales de la misma y a la vez cuantificarlos para mostrar los resultados esperados en el período previsto, todo esto sobre una concepción “Financiera” de las operaciones a realizar. Este software, puede ser utilizado por cualquier entidad de un organismo, ya que la captura, cálculo y presentación de los diferentes “Presupuestos”, está diseñada de forma general y no específicamente para un organismo.

La elaboración del Presupuesto Maestro exige de organización y coordinación de todas las operaciones a realizar en el sistema para lograr el resultado final por Unidades Contables y el consolidado a nivel de entidad, por esta razón se debe seguir una serie de instrucciones que trae consigo la explotación del mismo. Los "Codificadores" son los primeros elementos que el usuario complementará para poder efectuar los cálculos de los diferentes Presupuestos y de la calidad de su confección dependerá en gran medida el funcionamiento eficaz del resto del sistema y la confiabilidad de la información que se obtenga. Estos Codificadores son de gran importancia, ya que son la base sobre la cual se introducen todos los elementos que se requieren para poder trabajar en las capturas de los distintos presupuestos. Los "Presupuestos" se definen por productos y áreas de responsabilidad y sobre todo teniendo en cuenta el nivel de ingresos que se proyecte, es decir, se definen presupuestos asociados a la producción, los servicios, la administración y las ventas para tener diferenciados los gastos y calcular los costos de los productos con más facilidad. El sistema contempla además un grupo de reportes que resumen todo el proceso de captura de los diferentes presupuestos. Todos los reportes de los diferentes presupuestos tienen en su vista de captura la información necesaria para lograr el análisis de los resultados obtenidos en todo el proceso presupuestario, por lo que crean las condiciones para garantizar la etapa de Control del Plan. (5)

1.3. Sistemas Internacionales.

1.3.1. SAP BUSINESS ONE.

Es una aplicación en la cual se integran todas las funciones básicas de una empresa (incluye gestión financiera, ventas, gestión de atención al cliente, comercio electrónico, gestión de inventarios y operaciones). A diferencia de muchas otras pequeñas soluciones empresariales que existen actualmente en el mercado, **SAP Business One** es una única aplicación que elimina la necesidad de instalaciones separadas y la complicada integración de varios módulos.

Consta de una sola aplicación, las herramientas de personalización son de fácil utilización, y contiene más de 550 soluciones complementarias. Puede adaptarse y ampliarse de forma flexible para cumplir con las necesidades de la empresa. (6)

SAP ERP no cuenta con un módulo centralizado destinado a la planificación de todas las áreas de las empresas, en lugar de esto incluye este proceso en los módulos o áreas funcionales que lleven a cabo

una planificación de determinado recurso, variando en algunas soluciones en dependencia de las particularidades de la cada solución ofrecida.

SAP presenta los siguientes módulos que manejan el proceso de planificación de las empresas:

- **Production Planning (PP):** Planificación de la Producción engloba las distintas tareas y metodologías utilizadas en el proceso de producción. Este presenta una solución tanto para el plan de solución como para el proceso de producción. Los preparativos para la producción incluyen la adquisición, almacenamiento y transporte de materiales y productos que intervienen.
- **Materials Management (MM, Gestión de materiales):** El módulo MM tiene como función principal brindar soporte a todas las fases de gestión de materiales: planificación de necesidades y control, compras, entrada de mercaderías, gestión de inventario y verificación de facturas. Precisamente una de sus funciones es la Planificación de las Necesidades sobre Consumo (MM-CBP, por sus siglas en inglés), la cual se encarga de supervisar inventarios y crear automáticamente propuestas de pedidos para el departamento de compras y fabricación.
- **Controlling(CO):** Este además de documentar sucesos reales, la principal tarea de este módulo es la planificación. Mediante esta es posible determinar las desviaciones comparando los datos reales con los datos planeados. Esos cálculos de desviaciones permiten controlar los flujos de negocios. Dentro del área de Control de Costos del Producto (CO-PC, siglas en inglés) se encuentra Planificación de Costos del Producto, la cual se utiliza para descomponer los costes de los productos de su empresa, tales como materiales fabricados, servicios y otros bienes intangibles, todo esto con el objetivo de analizar de manera detallada los costos.

1.3.2. Sage MAS 500 ERP.

Este sistema evalúa los resultados y la eficiencia de la gestión empresarial. Realiza funciones de: registrar, clasificar, controlar, integrar los estados financieros de toda la empresa, así como las de planificación, precios, costos, finanzas y estadísticas. La solución de finanzas de Sage MAS 500 contribuye al mejoramiento del flujo de efectivo, al integrar los módulos se permite tener acceso a datos actualizados, así como aumentar la integridad de los mismos, agilizar los procesos contables y obtiene análisis instantáneos e informes financieros de alta calidad para la toma de decisiones oportuna. (7)

Este sistema contiene módulos que facilitan el proceso de planificación, como por ejemplo:

- **Planificación de Requisitos Materiales (MRP por sus siglas en inglés):** Ofrece una interfaz de planificación única, lo que proporciona que estén actualizados datos como: las estadísticas, cantidades reales, los pedidos en curso de compra, órdenes de trabajo, órdenes de transferencia, y otros elementos que afectan las decisiones de planificación. Utiliza datos históricos y las fórmulas avanzadas de reposición para ayudarle a entender el material actual y futuro, así como los requisitos de distribución y cómo van a afectar el negocio del cliente. Permite la realización de cálculos en la planificación que garantizan que se generen rápidamente órdenes de compra de materias primas, servicios subcontratados, y órdenes de transferencias para mover el material entre los lugares de depósito. Este modulo proporciona que la planificación se realice eliminando el tiempo disponible para los días de descanso, el tiempo de inactividad programado, y los días festivos. Ayuda a prever la demanda proyectada en varios períodos de inventario. Exhibe información inmediata y detallada de los productos planificados, incluyendo el balance material.
- **Planificación Avanzada & Programación:** Este módulo utiliza las reglas de planificación estándar de la industria para programar automáticamente la producción basada en avanzados algoritmos y cálculos matemáticos. Muestra una tabla con las planificaciones que permiten que el personal o los supervisores se ajusten a los pedidos de alta prioridad. Provee gran visibilidad en toda la empresa para la planificación, ya que sus clientes tienen la posibilidad de ver en línea el estado de los pedidos realizados, así como las fechas de terminación previstas. Permite el control de versiones de la planificación, así como la realización de modificaciones y ajustes a la misma. A partir de estas versiones, permite generar planificaciones específicas para cada centro y una planificación central. Incluye 9 reglas de la planificación estándar en las industrias: Las operaciones puestas en marcha, las operaciones cerradas, planificación real anterior, porcentaje crítico, holgura mínima, código de prioridad, la fecha requerida, la fecha de vencimiento, y la fecha de entrada. Cada planificación incluye fechas de inicio, y fechas tope definidas por el usuario. Se puede planificar antes o después de las fechas establecidas. Permite el trabajo con plantillas, moldes, y matrices. Establece patrones para realizar operaciones, colores para diferenciar elementos y descripciones. (7)

1.3.3. Openbravo ERP Comunity Edition.

Es una aplicación de código abierto de gestión empresarial del tipo ERP destinada a empresas de pequeño y mediano tamaño. Openbravo es una aplicación con arquitectura cliente/servidor web escrita en Java. Se ejecuta sobre Apache y Tomcat y con soporte para bases de datos PostgreSQL y Oracle.

Es una aplicación completamente web que ha sido desarrollada siguiendo el patrón Modelo Vista Controlador. La mayor parte del código se genera automáticamente por el motor denominado WAD (Wizard for Application Development), basándose en la información contenida en el diccionario del modelo de datos (Data Model Dictionary). Esta característica proporciona una mejor calidad del código al reducir drásticamente la codificación manual, al tiempo que mejora la productividad y eficiencia del desarrollo. El motor ejecuta y recompila la aplicación cada vez que el administrador modifica la configuración para adaptarla a un nuevo requisito. (8)

En el sistema de gestión económica OpenBravo ERP el proceso de planificación se encuentra disipado por algunos módulos que lo componen. Entre los módulos que implementan alguna variante de la planificación se encuentran los de:

Gestión de aprovisionamientos: Es en este módulo donde se planifican las necesidades de aprovisionamiento, por explosión de las necesidades de producción, teniendo en cuenta inventarios mínimos, plazos de entrega y pedidos en curso.

Gestión proyectos y servicios: su funcionamiento se basa en el modelado de la estructura productiva de cada organización, así como de los datos relevantes para la producción: planes de producción y productos involucrados en las mismas, entre otras funcionalidades.

Gestión Financiera y Contabilidad: la actividad del referido módulo se resume entre otras acciones en el establecimiento de los planes por defecto, la definición de los planes y ejercicios contables, así como de la gestión interanual y los presupuestos.

1.4 Valoración de los sistemas.

Después de realizar un estudio a los sistemas 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 ausencia de un software que responda cabalmente a lo que en realidad necesita la economía cubana.

Versat Sarasola, en el proceso de planificación para la elaboración del plan no cuenta con la funcionalidad de gestión de columnas, ni de comentarios, así como tampoco están implementadas sus funcionalidades para la actividad empresarial. Este se sustenta sobre los mecanismos o principios del Presupuesto Maestro, descartando otros procesos no menos importantes como es el caso de la planificación basada en objetivos. A pesar de tener la planificación entre sus prestaciones, este sistema, no trata el proceso de planificación como un módulo independiente, sino que lo integra dentro del módulo de Finanzas, Cajas y Bancos, esto impide el trato detallado de este proceso. Además, el sistema es una aplicación de escritorio, que está diseñada para pocas empresas, no para darle un alcance nacional.

Aunque brinda muchas funcionalidades, el Presupuesto Maestro no es un sistema que siga un patrón para realizar el proceso en las entidades, no permite adaptarse a la situación del momento, y la información en la planificación es muy cambiante, por lo que se necesita que sea dinámico y flexible de modo que facilite la interacción con el usuario.

El proceso de planificación que presenta SAP permite realizar algunos procesos que también se tienen en cuenta en la economía cubana como es el caso del módulo Controlling (CO) que automatiza el proceso de control de la ejecución que se realiza en la planificación, pero está dirigido a economías capitalistas, por lo que es muy difícil adaptarlo a las características de la economía cubana, sin tener elevados gastos financieros y de recursos humanos. También es un software desarrollado sobre herramientas privativas, lo cual no aportaría beneficio alguno al país, ya que además de no poder gestionar todas las empresas a nivel nacional, tendría que invertir miles de dólares para poder utilizar este sistema.

SAGE es un software bastante abarcador en cuanto a los procesos de planificación de una empresa ya que maneja elementos importantes como es el caso del control de fechas y versiones que mantiene por cada planificación que realiza, permitiendo que se manipulen los datos de la planificación de una forma más organizada por cada entidad, sin embargo, la mayoría de sus funcionalidades se encuentran en paquetes distintos, y esto dificulta el proceso de instalación de cada uno de los módulos. También es un software que no se adecua a la economía del país debido a que establece reglas propias del proceso de planificación en países capitalistas, por lo tanto no se ajusta a las indicaciones que se establecen en las entidades cubanas para realizar este proceso. Además, es un software privativo, por lo que no es lo más conveniente para la economía del país.

OpenBravo ERP a pesar de constituir un sistema integral de gestión de pequeñas y medianas empresas, una solución comercial en software libre, y de posibilitar la ejecución del proceso de planificación, aunque

subdividido por funcionalidades independientes en los módulos Gestión de aprovisionamientos, Gestión proyectos y servicios, y Gestión Financiera y Contabilidad, no garantiza de manera secuencial y centralizada la planificación de una organización. Mantiene un enfoque orientado a la planificación de proyectos, no para realizar el proceso en un ámbito económico nacional.

Estos sistemas presentan diferentes inconvenientes ya sea en la gestión de los planes, modelos y comentarios, 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.

Por lo cual, en el proceso de lograr la soberanía tecnológica se trabajará sobre herramientas y tecnologías que no son privativas, con el fin de lograr un alto grado de eficiencia, confiabilidad y rapidez en la gestión de los planes, modelos y comentarios en las empresas.

1.5 Modelo de desarrollo.

Para que el conjunto de actividades que definen al proceso no se lleven a cabo de forma caótica se ha hecho necesario incorporar una estrategia cuyo objetivo fundamental fuera ordenar dichas actividades en el desarrollo del software. Esta estrategia es lo que se suele llamar Modelo de proceso del software. (9)

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. (10)

El modelo de desarrollo que se utiliza para el avance de la aplicación es el definido en la Unidad de Compatibilización, Integración y Desarrollo de software para la Defensa (UCID), tiene las siguientes características: basado en componentes, iterativo e incremental. Se emplearán las técnicas de prototipado, en caso de ser necesarias, para los requisitos del usuario de los que no existe una visión clara por parte de estos, con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema.

Desarrollo iterativo e incremental: Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento. (10)

Desarrollo basado en componentes: Lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. (10)

1.6 Patrones de diseño.

Los patrones de diseño son soluciones simples a problemas específicos y comunes del diseño orientado a objetos, su principal objetivo es agrupar una colección de soluciones de diseño que sean válidas en distintos contextos. Es una solución a un problema de diseño no trivial que es efectiva, además facilitan el aprendizaje al programador inexperto, pudiendo establecer parejas problema-solución. Entre estos patrones se pueden encontrar:

1.6.1 Patrones GRASP

Patrones de Software para la asignación general de responsabilidad. Describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades (11). Se pueden destacar 5 patrones principales que son:

- **Experto**: Asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- **Creador:** Asigna a la clase B la responsabilidad de crear una instancia de la clase A en uno de los siguientes casos:
  B agrega los objetos A.
  B contiene objetos A.
  B registra las instancias de los objetos A.

B utiliza las instancias de los objetos A.

B utiliza específicamente los objetos A.

B tiene los datos de inicialización que serán trasmitidos a A cuando sea creado.

- **Alta cohesión:** Asigna una responsabilidad de modo que la cohesión siga siendo alta.
- **Bajo acoplamiento:** Asigna una responsabilidad para mantener bajo acoplamiento.
- **Controlador**: Asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. (12)

1.6.2 Patrones GOF

Los patrones GOF son 23 patrones que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Están clasificados en patrones creacionales, estructurales y de comportamiento. Los patrones creacionales se encargan de la creación de los objetos ayudando a que el sistema sea independiente de la creación, composición y representación de los objetos. Los patrones estructurales son los encargados de definir cómo las clases y objetos están compuestos para formar estructuras más grandes. Usan la herencia para componer interfaces u objetos en tiempo de ejecución. Los patrones de comportamiento plantean algoritmos y la asignación de responsabilidades entre objetos. Estos patrones no solo describen clases y objetos sino también describen la comunicación entre ellos. (13)

Entre ellos se encuentra el patrón Fachada, que se utiliza para proporcionar una interfaz unificada de alto nivel para un conjunto de clases en un subsistema, haciéndolo más fácil de usar. Simplifica el acceso a dicho conjunto de clases, ya que el cliente sólo se comunica con ellas a través de una única interfaz. (14)

1.7 Patrón Arquitectónico.

Los patrones arquitectónicos son los que definen la estructura de un sistema software. Se componen de subsistemas con sus responsabilidades. Tienen una serie de directivas para organizar los componentes del mismo sistema, con el objetivo de facilitar la tarea del diseño de tal sistema. (15)

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.

El equipo de arquitectura del proyecto ERP-Cuba dentro de la vista del sistema incluye la vista de componente; en ese caso propone que todas las funcionalidades levantadas y modeladas en las fases de

negocio y requisitos 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.

Por lo que en la estructura del marco de trabajo queda organizado por componentes, dentro de los cuales se aplica el patrón Modelo Vista Controlador.

El Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la interfaz de usuario y el código es el que provee de datos dinámicos a la página; el modelo que está conformado por el acceso a datos y la lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Modelo:

Esta es la representación específica de la información con la cual el sistema opera. Maneja los datos y controla sus transformaciones. Este no tiene conocimiento específico de los controladores y las vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas y notificar a las vistas cuando cambia el modelo.

Vista:

Maneja la presentación visual de los datos representados por el modelo el cual es presentado en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador:

Responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y probablemente en la vista. (16)

1.8 Lenguajes de Programación web.

1.8.1 PHP 5.2

“PHP (acrónimo de Hypertext Preprocessor) es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas Web dinámicas. Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo.” (17).

Es gratuito y multiplataforma, rápido, con una gran librería de funciones y mucha documentación. En este caso se estará haciendo uso de PHP 5.2 o superior, con los siguientes módulos o extensiones: pdo, pdo_pgsql, pgsql, soap.

Características:

- Completamente orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5). (18)

1.8.2 Javascript 1.8.2.

“Javascript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos.” (19)

Entre las acciones típicas que se pueden realizar en javascript se tienen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, javascript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se puede crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas,

etc. Además, pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. Su sintaxis es similar a la usada en Java y C, al ser un lenguaje del lado del cliente este es interpretado por el navegador, no se necesita tener instalado ningún Framework. Javascript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

Con el surgimiento de lenguajes como PHP del lado del servidor y javascript del lado del cliente, surgió Ajax en acrónimo de (Asynchronous Javascript And XML o Javascript asíncrono y XML). AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. Javascript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax. (20)
Las aplicaciones Ajax pueden enviar peticiones al servidor web para traer solamente los datos que son necesarios, generalmente utilizando SOAP[2] o algún otro dialecto de servicio web basado en XML. O sea que la ayuda más potente que ofrece AJAX es el poder hacer consultas asíncronas al servidor sin necesidad de recargar la página. (20)

1.8.3 HTML (Hypertext Markup Language)

El lenguaje de programación de marcas hipertextuales, creado por Tim Berners-Lee. Nació como un lenguaje de marcas para producir todo tipo de documentos estructurados. Actualmente es 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 (<,>). (21)

1.8.4 XML (Extensible Markup Language)

XML es un lenguaje de etiquetado extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Permite definir la gramática de lenguajes específicos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

[2] SOAP: (Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. (22)

Ventajas:

- Es extensible: después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan errores y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server. (22)

1.9 Tecnologías y herramientas.

Para el desarrollo de los módulos propuestos se utilizaron varias herramientas y tecnologías seleccionadas por el Departamento de Tecnología del Centro Integral para la Gestión de Entidades (CEIGE).

1.9.1 Herramienta de modelado: Visual Paradigm 6.0

Visual Paradigm es una herramienta CASE que ayuda a construir aplicaciones rápidamente, y con mayor calidad (23). Esta herramienta tiene características gráficas muy cómodas para los usuarios y posee numerosas ventajas:

- Genera código para Java y exportación como HTML.
- Entorno de creación de diagramas para UML.
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.

- Disponibilidad en múltiples plataformas.
- Soporta aplicaciones web.
- Fácil de instalar y actualizar.

1.9.2 Tortoise SVN 1.6.5

TortoiseSVN es un cliente para Subversion implementado de forma muy práctica: integrado en Windows. Es además software libre liberado bajo la licencia GNU GPL. Es intuitivo y fácil de usar, y proporciona un acceso más rápido y visual a todas las funciones de Subversion. Este programa también permite, mantener controladas las copias, poder recuperar antiguas versiones con facilidad, y da a conocer el historial de cambios realizados. (24)

Características

- Integración con la línea de comandos de Windows.
- Puede ser usado sin un entorno de desarrollo.
- Pequeñas imágenes decoran los íconos de los archivos mostrando qué archivos o directorios necesitan ser enviados al repositorio.
- Disponible en 28 idiomas diferentes. (24)

1.9.3 Firefox 3.6.

Se estará utilizando como explorador web Firefox de la familia Mozilla u otro que implemente el DOM 2.0 y que soporte Javascript. Este navegador ofrece una seria alternativa al extendido y "monopolizante" Internet Explorer. Se trata de un práctico y ágil navegador, que está en renovación constante una de las ventajas del código abierto, la capacidad de modificarlo totalmente a gusto del usuario y según las necesidades del mismo. Esto se consigue gracias a la multitud de "extensiones" que existen, y que cada día aparecen más, que permiten añadirle nuevas funciones de todo tipo. (25)

- DOM: El Document Object Model (Modelo de Objeto de Documento), Es una plataforma que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos HTML y XML, navegar por su estructura y, modificar, añadir y borrar tanto elementos como contenidos. Al no apoyarse en un lenguaje de programación en particular, DOM facilita el diseño de páginas web activas, proporcionando una interfaz estándar para que otro software manipule los documentos. (25)

1.9.4 Zend Studio 5

Zend Studio es un completo entorno de desarrollo integrado (IDE) para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux (26)

Características

- No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- Plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases).
- Emparejamiento (*matching*) de paréntesis y corchetes (si se sitúa el cursor sobre un paréntesis (corchete) de apertura (cierre), Zend Studio localiza el correspondiente paréntesis (corchete) de cierre (apertura)).
- Detección de errores de sintaxis en tiempo real.
- Funciones de depuración: Botón de ejecución y traza, marcadores, puntos de parada (*breakpoints*), seguimiento de variables y mensajes de error del intérprete de PHP. Permite también la depuración en servidores remotos (requiere Zend Platform).
- Instalación de barras de herramientas para Internet Explorer y Mozilla Firefox (opcional).
- Soporte para control de versiones usando CVS o Subversion.
- Cliente FTP integrado. (26)

1.9.5 Frameworks

Es una biblioteca de programación pensada como un gran ¨paquete¨ que resuelve un cierto problema con un nivel de abstracción muy grande, para el programador que lo utiliza. Puede contener o requerir varias bibliotecas para su correcto funcionamiento. (27)

1.9.5.1 Marco de Trabajo Sauxe 1.5.4.

El Marco de Trabajo Sauxe se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. Sauxe utiliza ExtJS para implementar la capa de presentación, se apoya en Zend, una extensión

de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza Doctrine. Este utiliza como patrón arquitectónico MVC. También tiene como propósito insertar la programación orientada a aspectos así como la inversión de controles. El framework define que la conexión a la base de datos será configurada en un xml almacenado en la carpeta de recursos comunes del proyecto. Es válido aclarar que este cuenta con un componente de transacciones mediante el cual serán salvados automáticamente los datos de modificaciones e inserciones. Es decir, ya no será necesaria la implementación por parte del programador de las consultas de inserción, éste solamente deberá programar la obtención de los campos de la presentación y el Sauxe será el responsable de que los mismos sean guardados en la base de datos. (28)

Algunas de las ventajas más importantes del marco de trabajo:

- Los identificadores de cada tupla de las tablas serán generados automáticamente en la base de datos como una secuencia.
- El antiguo método de try y catch para el lanzamiento de excepciones desaparece, solo basta registrar las excepciones en el manager de excepciones y el framework será capaz de realizar un tratamiento óptimo de las mismas.
- No será necesario en cada método de inserción de información a la base de datos ejecutar el método correspondiente en la clase del negocio correspondiente, el framework será capaz de que una vez tomados los datos de la presentación salvarlos en la base de datos directamente.

1.9.5.2 ExtJS 2.2.

ExtJS es una biblioteca de Javascript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, ExtJS no se preocupará de lo que pase en el servidor.

ExtJS es simplemente un framework escrito en Javascript que permite desarrollar aplicaciones web complejas de una forma cómoda sin tener que lidiar con los típicos problemas de la programación. Soporte para construir interfaces gráficas complejas y dinámicas, comunicar datos de forma asíncrona con el servidor y manejar datos de distinta índole de una manera simple. ExtJS se ha basado en otros framewroks previamente construidos y los han unificado obteniendo una mayor potencialidad. (29)

1.9.5.3 Zend Framework 1.9.7.

Zend Framework (ZF) es un framework de código abierto para desarrollar aplicaciones web y servicios web con PHP 5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. (30)

1.9.5.4 Doctrine Framework 1.2.1

Es un framework que mapea objetos relacionales (ORM por sus siglas en inglés) para PHP con una potente capa de abstracción de bases de datos (dbal por sus siglas en inglés). Uno de sus principales características es la opción de escribir las consultas de base de datos orientada a objeto en un lenguaje SQL llamada Doctrine Query Language (DQL por sus siglas en inglés). Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (31)

1.9.6 PostgreSQL 8.3.9.

Es un sistema de gestión de base de datos relacional orientada a objetos de software libre**,** entre sus principales características están:

- **Alta concurrencia**: Mediante un sistema denominado MVCC (Acceso concurrente multiversión) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. (32)
- **Amplia variedad de tipos nativos**

  PostgreSQL provee nativamente soporte para:
  - Números de precisión arbitraria.
  - Texto de largo ilimitado.
  - Figuras geométricas (con una variedad de funciones asociadas).
  - Direcciones IP (IPv4 e IPv6).
  - Bloques de direcciones estilo CIDR.
  - Direcciones MAC.
  - Arreglos. (32)

1.9.7 Servidor Web.

Un servidor web es un programa que implementa el protocolo transferencia de hipertexto (HTTP por sus siglas en inglés). Este protocolo está diseñado para transferir 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 2.0.

Es una tecnología de código fuente abierto, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable de diseño modular, trabaja con gran cantidad de lenguajes como por ejemplo: Perl, PHP y otros lenguajes de script.

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa.

Características:

- Multiplataforma.
- Apache es una tecnología gratuita de código fuente abierto.
- Apache es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor Web Apache. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que los instalemos cuando los necesitemos (33)
- Apache trabaja con Perl, PHP y otros lenguajes de script. Perl destaca en el mundo del script y Apache utiliza su parte del pastel de Perl tanto con soporte CGI como con soporte mod perl. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- Tiene una alta configurabilidad en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor. (33)

1.9.8 Aptana Studio 3.

Aptana es un entorno de desarrollo especializado en la programación de aplicaciones dinámicas para web, con especial soporte para Ajax/Javascript, que permite editar fácilmente HTML, CSS y Javascript. Está basado en el conocido entorno de desarrollo Eclipse (IDE), también de código abierto. Pero mientras que Eclipse está focalizado en el desarrollo para Java, Aptana Studio es una distribución focalizada en el desarrollo web. Además, cuenta con interesantes detalles como un depurador integrado, este último le avisará si hay errores en el código. (34)

1.9.9 Lenguaje de modelado.

1.9.9.1 UML

El Lenguaje de Modelado Unificado (Unified Modeling Language) es una especificación de notación orientada a objetos. Es utilizado con el fin de especificar y documentar un sistema de software, de un modo estándar. Incluye aspectos conceptuales tales como procesos de negocios y funciones del sistema. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto. UML implementa un lenguaje de modelado común para todos los desarrollos por lo que se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo. (35)

1.10. Métricas

Una métrica es un instrumento que cuantifica un criterio. Sus principales objetivos son: ayudar a comprender mejor la calidad del producto, estimar la efectividad del proceso, y mejorar la calidad del trabajo realizado en el nivel del proyecto. Hay varias razones para medir un producto:

1. Para indicar la calidad del producto.
2. Para evaluar la productividad de la gente que desarrolla el producto.
3. Par evaluar los beneficios en términos de productividad y de calidad, derivados del uso de nuevos métodos y herramientas de la ingeniería de software.
4. Para establecer una línea de base para la estimación
5. Para ayudar a justificar el uso de nuevas herramientas o de formación adicional.

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." (36)

Para hacer una evaluación del diseño propuesto, se tienen en cuenta tantos atributos como métricas de calidad.

Atributos de calidad que se abarcan:

- **Responsabilidad:** Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
- **Complejidad del mantenimiento**: Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
- **Complejidad de implementación:** Grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
- **Acoplamiento:** Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- **Cantidad de pruebas:** Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado. (37)

Métricas que se emplean:

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.

1.11 Conclusiones Parciales

Durante el desarrollo de este capítulo se efectuó un estudio del estado del arte, con el fin de valorar algunos de los sistemas que contienen funcionalidades para realizar la planificación de recursos, evidenciándose la no existencia de un sistema informático capaz de ejecutar tales funcionalidades, ni de

cumplir con los requisitos establecidos a nivel nacional. Se hizo también un estudio de las tendencias y tecnologías actuales. Por otra parte, se explicó el modelo de desarrollo y la arquitectura a utilizar, así como las técnicas, patrones de diseño, tecnologías y herramientas indispensables para el desarrollo de la investigación. Se caracterizaron los lenguajes de modelado y de programación a utilizar analizando sus ventajas y desventajas.

Capítulo 2- Propuesta de solución.

2.1 Introducción

La solución propuesta está sustentada sobre la base de todo lo que se expone en este capítulo, en el cual se definen las actividades más importantes que posibilitaron la implementación de la misma.

Se persigue con la misma proporcionarle al usuario la posibilidad de listar, modificar y eliminar los planes, modelos y comentarios, pertenecientes al subsistema de planificación, lo que no solo ayudará a cumplir en tiempo con su planificación, sino que también permitirá agregar indicadores a un modelo, consultar un plan, adicionar comentarios, entre otras.

Dicha aplicación debe brindar la posibilidad de mantener un control y seguimiento de determinadas funciones, indicaciones, y resoluciones emitidas por los niveles superiores de cada entidad. Con la implementación de estos componentes, se necesitará la integración con otros componentes del subsistema Planificación, como indicadores, tipo modelo y nomencladores, mejorando en eficiencia, ya que se cuenta de forma general con una alta cohesión por parte de los componentes en cuestión. Además contará con una interfaz amigable para el usuario, de manera que le sea más agradable el manejo de estos componentes.

2.2 Diseño de la solución:

2.2.1. Requisitos Funcionales

Los requisitos funcionales identificados y aprobados se agruparon en un conjunto de funcionalidades con una finalidad u objetivo específico:

Gestionar Planes:

1-Adicionar Plan

2- Modificar Plan

3- Eliminar Plan.

4- Listar Planes.

5- Consultar Plan.

Gestionar Modelos:

1- Adicionar Modelo.

2- Modificar Modelo

3- Eliminar Modelo.

4- Listar Modelos.

5- Adicionar indicador a un modelo.

6- Planificar indicador.

7- Eliminar indicador a modelo.

8- Cambiar estado de un modelo.

9- Consultar modelo.

Gestionar Comentarios.

1- Adicionar comentario.

2- Modificar comentario.

3- Eliminar comentario.

4- Listar comentarios.

5- Consultar comentario.

2.2.2. Valoración crítica del análisis.

Partiendo de los artefactos obtenidos durante el análisis de los procesos de planificación para los componentes Plan, Modelo y Comentario se realiza la valoración de los mismos. Se validaron los requisitos para demostrar que sus definiciones son las que el usuario final necesita, a través del artefacto Validación de requisitos, el cual fue generado por el analista principal del proyecto (Ver anexo 1). Además se firmaron las actas de aceptación por los especialistas funcionales y el líder de la línea.

Los requisitos identificados están claros y se entienden correctamente, lo que permite alcanzar retroalimentación en cuanto a si el sistema diseñado basándose en los requisitos permite al usuario realizar su trabajo de manera eficiente y efectiva. Para asegurar que los requisitos han sido establecidos sin ambigüedades o inconsistencias y que los errores encontrados durante la definición de los mismos hayan sido corregidos, se realizó una revisión a las especificaciones realizadas para ser aprobadas y no se detectaron deficiencias, omisiones, ni errores en la especificación. Se realizaron revisiones de artefactos generados en la etapa inicial de la fase de modelación de software.

Se comprobó que las necesidades de los clientes se encuentren cubiertas por los requisitos capturados y especificados, creando una puerta de entrada apropiada y un punto de partida para las actividades de diseño e implementación.

2.2.3. Diseño de la solución en términos de componentes:

El subsistema Planificación perteneciente al Sistema Integral de Gestión de Entidades CEDRUX, está destinado a la integración de los procesos automatizados de planificación con los procesos de control contable y físico de recursos; a posibilitar un ahorro de tiempo en la planificación y a mejorar el control del estado de los planes.

Se presentará el modelo de componentes propuesto para el subsistema Planificación, donde se relacionan a través de interfaces de comunicación un conjunto de componentes definidos para dar solución arquitectónica al sistema entre los cuales se encuentran: Tipo de Modelo, Balance Material, Bases para el Plan, Indicadores y los componentes a desarrollar Plan, Modelo y Comentario.

Modelo de componentes:

Un modelo de componentes representa los componentes, sus interfaces y las relaciones de los componentes con las interfaces que utilizan.

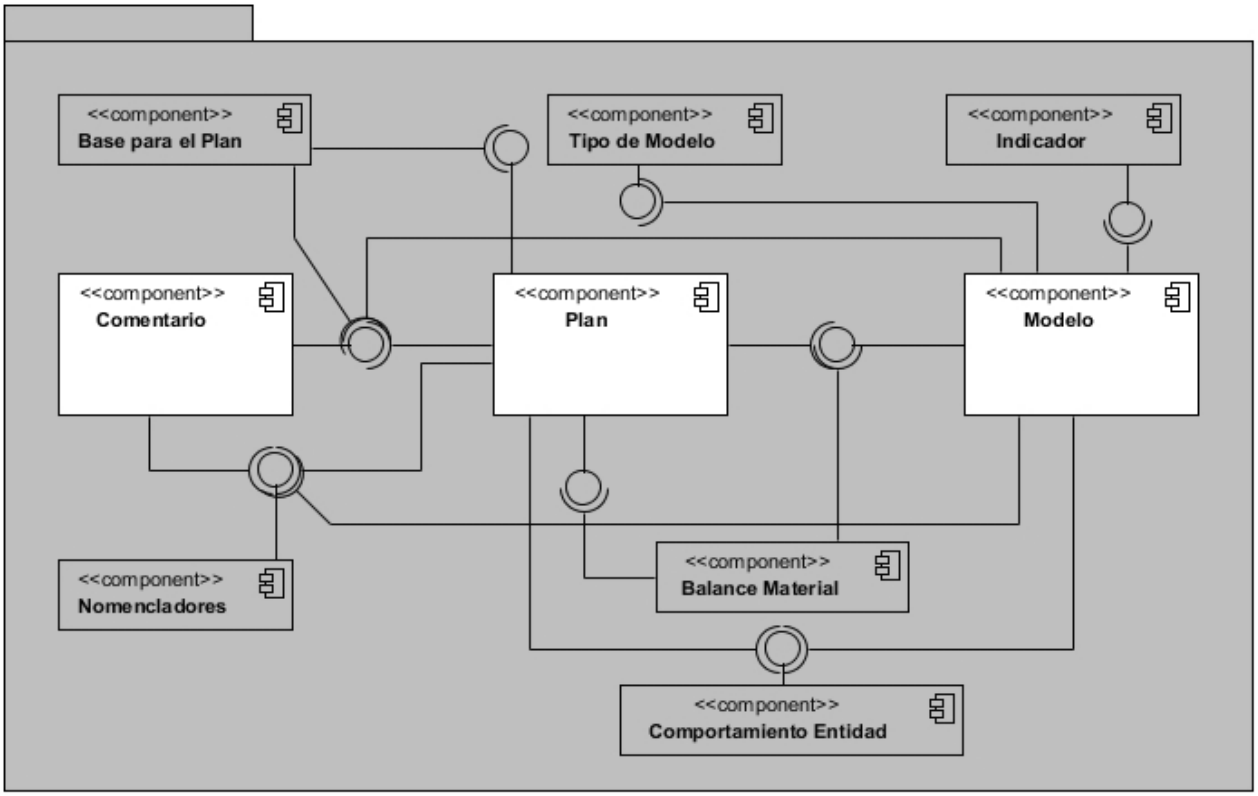

Ilustración 1- Modelo de Componentes

A continuación, se muestra una explicación más detallada de la funcionalidad de cada uno de estos componentes a partir de las funcionalidades que brindan y reciben:

Indicador:

Componente encargado de la gestión de los indicadores a planificar, donde además se definen los niveles de planificación. Necesita del componente Nomencladores para establecer la(s) vía(s) de adquisición de los indicadores, y de los subsistemas Estructura y Composición y Configuración para el manejo de entidades, formatos y unidades de medida respectivamente. Es utilizado por el componente Modelo.

Tipo de Modelo:

En este componente se especifica qué atributos (columnas) va a tener el tipo de modelo, así como la configuración de cada una de ellas. Es utilizado por el Componente Modelo, ya que a partir de la definición de un tipo de modelo en específico es que se crean los modelos.

Balance Material:

Este componente es encargado de realizar el balance material a los centros de balance y a cualquier entidad que desee hacer su balance material, así como la aprobación de la demanda por parte de los indicadores y realizar ajustes a ésta. Utiliza los componentes Plan y Modelo.

Comportamiento Entidad:

El componente Comportamiento integrado con Nomencladores, permitirá especificar qué entidad del dominio de entidades o subordinación se comportará como Centro de Balance, Órgano demandante u Órgano ejecutor. Los componentes Plan y Modelo, hacen uso de este componente.

Nomencladores:

El componente Nomencladores gestiona los nomencladores del subsistema de manera general, está representado en varias tablas entre las que se encuentran: Tipo de comentario, Tipo de documento, Ejercicio, entre otros.

Bases para el plan:

Este componente es encargado de crear actividades, asignarles recursos, niveles de actividad, asociarlas a entidades, definir y notificar reglas dado niveles de actividad, calcular la necesidad e importe por actividad, gestionar las normas, y desglosar actividades. Utiliza funcionalidades del componente Plan.

Plan:

Componente que debe ser diseñado e implementado. Para crear un plan debe existir al menos un ejercicio. Además, en el componente Bases para el plan, se deben definir con anterioridad los niveles de

actividades, las actividades, asignarles recursos, asociarlas a entidades, definir y notificar reglas dado niveles de actividad, calcular la necesidad e importe por actividad, gestionar las normas, y desglosar actividades. Un plan puede agrupar dentro otros planes, o modelos, además tener comentarios, por lo tanto utiliza los componentes Modelo, Comentario, Nomencladores, y Comportamiento Entidad, al mismo tiempo que los componentes Bases para el Plan y Balance Material hacen uso de él.

Modelo:

Componente que debe ser diseñado e implementado. Para crear un modelo, se debe definir con anterioridad al menos un tipo de modelo, donde se especifican las columnas que va a tener el mismo. En este componente es donde se realiza la planificación de los indicadores. El modelo deberá permitir además adicionarle comentarios. Permitirá establecer un formato para cada columna según el indicador, los formatos de unidades de medida, se obtendrán del componente Unidad de Medida del marco de trabajo. Por lo tanto utiliza los componentes Comentario, Tipo de modelo, Indicador, Nomenclador y Comportamiento Entidad; mientras que los componentes Plan y Balance Material hacen uso de él.

Comentario:

Componente que debe ser diseñado e implementado, para permitir la gestión de los comentarios que se les puedan agregar a los modelos y a los planes. Los comentarios podrán ser de varios tipos (Ejemplo: de alerta, notificación, informativo, de error, violación de restricción, ajuste al plan, aviso…etc.), además cuando se solicita el reajuste de una demanda, se crea un comentario donde la entidad que rechaza la demanda explica las razones del reajuste.

Integración externa.

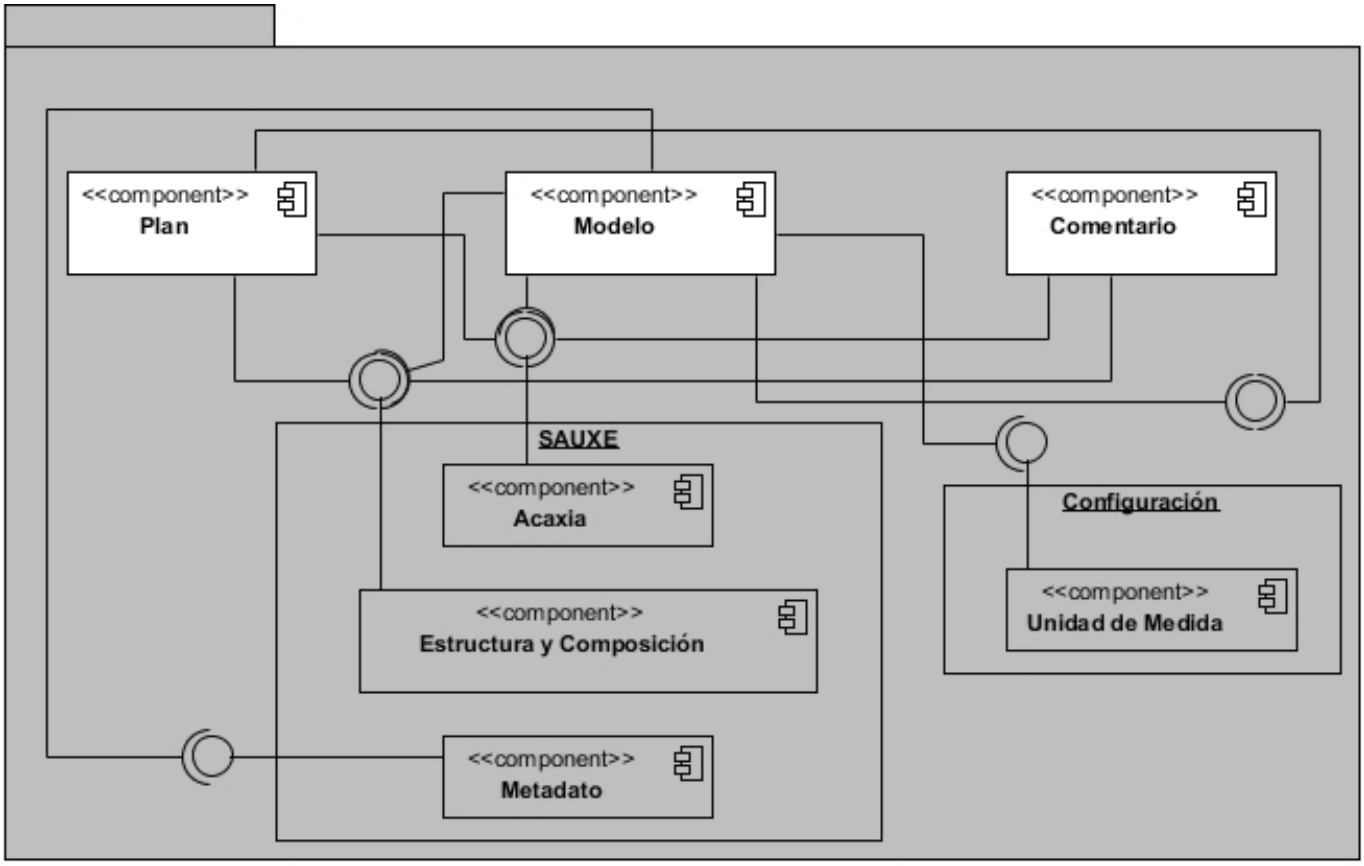

Ilustración 2- Integración externa.

A continuación, una explicación más detallada de la funcionalidad de cada uno de estos componentes a partir de los servicios que reciben:

Metadato: componente del subsistema SAUXE, que permite almacenar y recuperar dinámicamente la información de los indicadores que se planifican en los modelos.

Acaxia: componente del subsistema SAUXE encargado de mantener una administración segura y centralizada de todos los sistemas que utilicen sus servicios.

Estructura y composición: componente del subsistema SAUXE que brinda una estructura jerárquica para el manejo de las entidades que controlarán los planes y modelos.

Unidad de Medida: componente del subsistema Configuración que gestionará la información referente a las unidades de medida que se utilizan en los modelos.

2.2.4. Diagramas de Clases de Diseño

Los diagramas de clases según la clasificación UML son diagramas de estructura estática donde la representación de los requisitos se lleva a cabo a través de las clases del sistema y sus interrelaciones. Representan una abstracción del dominio de modo que es formalizado el análisis de conceptos y constituyen el pilar básico del modelado, mostrando en términos generales qué debe hacer el sistema.

Específicamente los diagramas de clases de diseño son muy útiles porque muestran a través de atributos y métodos la estructura de las clases que después serán escritas en algún lenguaje de programación (PHP en este caso).

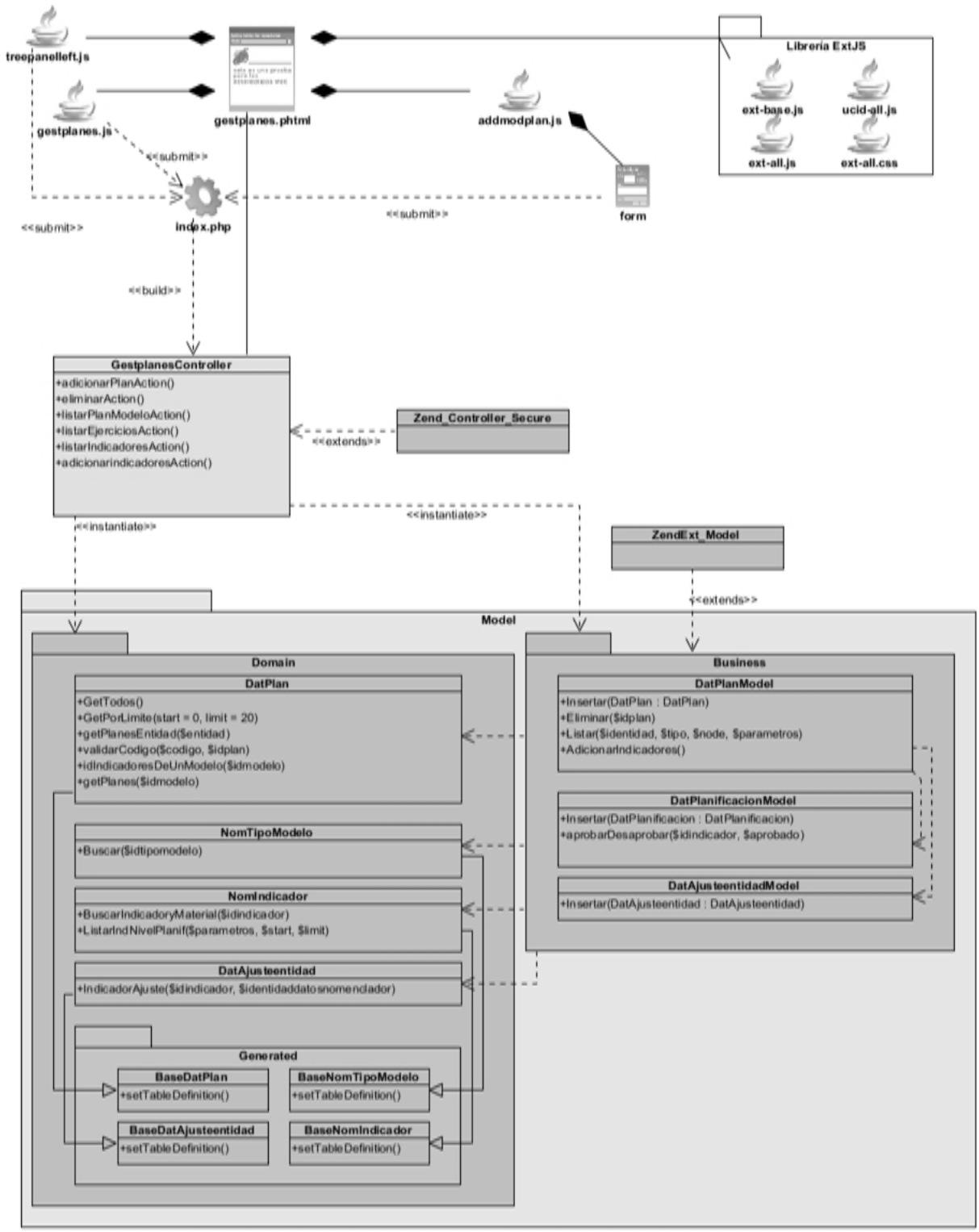
treepanelleft.js
gestplanes.phtml
Librería ExtJS
ext-base.js
ucid-gll.js
ext-all.js
ext-all.css
gestplanes.js
addmodplan.js
«submit»
«submit»
«submit»
index.php
form
«build»
GestplanesController
+adicionarPlanAction()
+eliminarAction()
+listarPlanModeloAction()
+listarEjerciciosAction()
+listarIndicadoresAction()
+adicionarindicadoresAction()
Zend_Controller_Secure
«extends»
«instantiate»
«instantiate»
ZendExt_Model
«extends»
Model
Domain
DatPlan
+GetTodos()
+GetPorLimite(start = 0, limit = 20)
+getPlanesEntidad($entidad)
+validarCodigo($codigo, $idplan)
+idIndicadoresDeUnModelo($idmodelo)
+getPlanes($idmodelo)
NomTipoModelo
+Buscar($idtipomodelo)
NomIndicador
+BuscarIndicadoryMaterial($idindicador)
+ListarIndNivelPlanif($parametros, $start, $limit)
DatAjusteentidad
+IndicadorAjuste($idindicador, $identidaddatosnomenclador)
Generated
BaseDatPlan
+setTableDefinition()
BaseNomTipoModelo
+setTableDefinition()
BaseDatAjusteentidad
+setTableDefinition()
BaseNomIndicador
+setTableDefinition()
Business
DatPlanModel
+Insertar(DatPlan : DatPlan)
+Eliminar($idplan)
+Listar($identidad, $tipo, $node, $parametros)
+AdicionarIndicadores()
DatPlanificacionModel
+Insertar(DatPlanificacion : DatPlanificacion)
+aprobarDesaprobar($idindicador, $aprobado)
DatAjusteentidadModel
+Insertar(DatAjusteentidad : DatAjusteentidad)

Ilustración 3- Diagrama de clases del diseño del componente Plan.

Tabla 1- Descripción del diseño de clases del componente Plan.

Clases	Descripción
Librería Extjs	Contiene los componentes generados a través de la librería Javascript Extjs.
gestplanes.phtml	Página encargada de visualizar, a través de los js que debe incluir, la información relacionada con la gestión de los planes, modelos y comentarios.
gestplanes.js	Encargada de generar de forma dinámica a través del DOM y utilizando la librería Extjs, los componentes que manejen la información en la vista necesaria para lograr la gestión de los planes. Debe enviar y recibir los datos de la controladora utilizando tecnología AJAX.
addmodplan.js	Encargada de generar a través del DOM y utilizando la librería Extjs, los componentes necesarios para adicionar o modificar los datos de un plan.
treepanelleft.js	Debe generar de forma dinámica a través del DOM y utilizando la librería Extjs el componente a través del cual se muestra el listado de planes y modelos, en forma de árbol.
GestplanesController	Clase controladora responsable de gestionar los planes, donde se deben ejecutar las diferentes funcionalidades, según las peticiones del usuario.
ZendExt_Controller_Secure	Encargada de gestionar acciones personalizadas y está integrada a la seguridad.
Paquete Model	Encargado de manejar los datos persistentes dentro del componente. Contiene el Bussines y el Domain.
ZendExt_Model	Modelo gestor de negocio que permite entre otras funcionalidades iniciar la conexión a la base de datos.
Index	Constituye el único punto de acceso a la aplicación,

	conjuntamente con la clase Zend_Loader y Zend_Controller_Front se encarga del funcionamiento de la aplicación, atención a solicitudes y respuestas.

Diagrama de clases del diseño del componente Modelo (Ver anexo 2).
Descripción del diseño de clases del componente Modelo (Ver anexo 3).
Diagrama de clases del diseño del componente Comentario (Ver anexo 4).
Descripción del diseño de clases del componente Comentario (Ver anexo 5).

2.2.5. Diagramas de secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. (37)

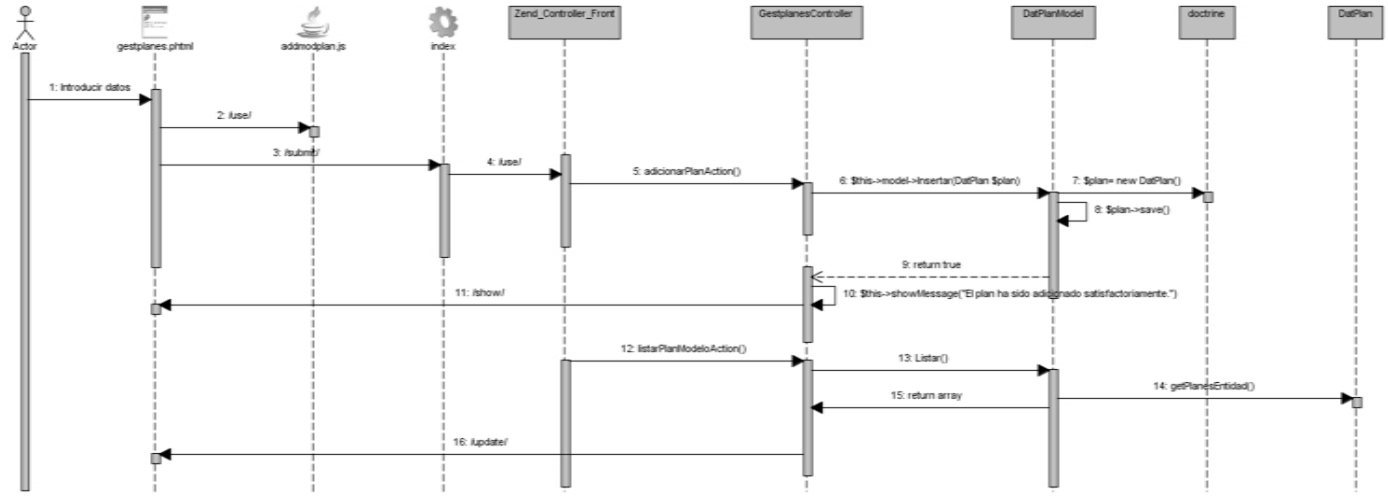

Ilustración 4- Diagrama de Secuencia Insertar Plan.

Se realizó un diagrama de secuencia para cada requisito funcional. (Ver Anexos 6 y 7.)

2.2.6. Modelo de datos:

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. (38)
El modelo propuesto consta de 15 tablas, para su elaboración se tuvo en cuenta la reducción a la mínima expresión de los campos nulos. La tabla crítica es Dat_Planificación, ya que en ella es donde más

inserciones y recuperaciones se va a realizar, debido a que es donde se va a almacenar la relación entre los modelos creados en las entidades y los indicadores planificados en estos.

Teniendo en cuenta el requisito funcional Gestionar Planes se crea la tabla dat_plan, en la misma se guardan los id de cada uno de los planes, así como su denominación, código, versión y descripción. Además se almacena en ella el identificador de la entidad que crea el plan, el id padre y el orden izquierdo y orden derecho, ya que es un componente arbóreo. Tiene relación con la tabla dat_modelo que contiene los datos de los modelos asociados al plan, y con la tabla dat_planactividad, en la que se registra la actividad para la cual se realiza el plan.

Según el requisito funcional Gestionar Modelos, se especificaron las siguientes tablas:

En primer lugar, la tabla dat_modelo contiene los atributos fundamentales de un modelo, está relacionada con la tabla nom_tipomodelo que contiene los datos de tipos de modelos, que contienen las columnas del modelo.

En la tabla dat_entidadmodelo se almacena la relación de varias entidades con varios modelos, y a través de la misma se tiene relación con la tabla dat_planificacion, que es la encargada de registrar la relación de indicadores por cada modelo. La tabla dat_planificacion a su vez provee una relación con la tabla nom_ejercicioplanificacion que contiene todos los ejercicios para los cuales se crean los planes y modelos.

También, a partir de la relación con la tabla nom_indicador, se tiene acceso a la tabla intermedia dat_indicadorcentrobalance que contiene la información de los indicadores que abastecen los centros de balance.

Teniendo en cuenta el requisito funcional Gestionar Comentario, se especificaron las siguientes tablas:

dat_comentario contiene los datos de los comentarios que se le puede agregar a los modelos y a los planes, para ello, tiene un idelemento que contiene el id del plan o del modelo al que se le va a agregar el comentario.

La tabla nom_tipocomentario guarda la clasificación a partir de la cual, se crean los comentarios, a partir de la relación que ésta guarda con la tabla dat_comentario.

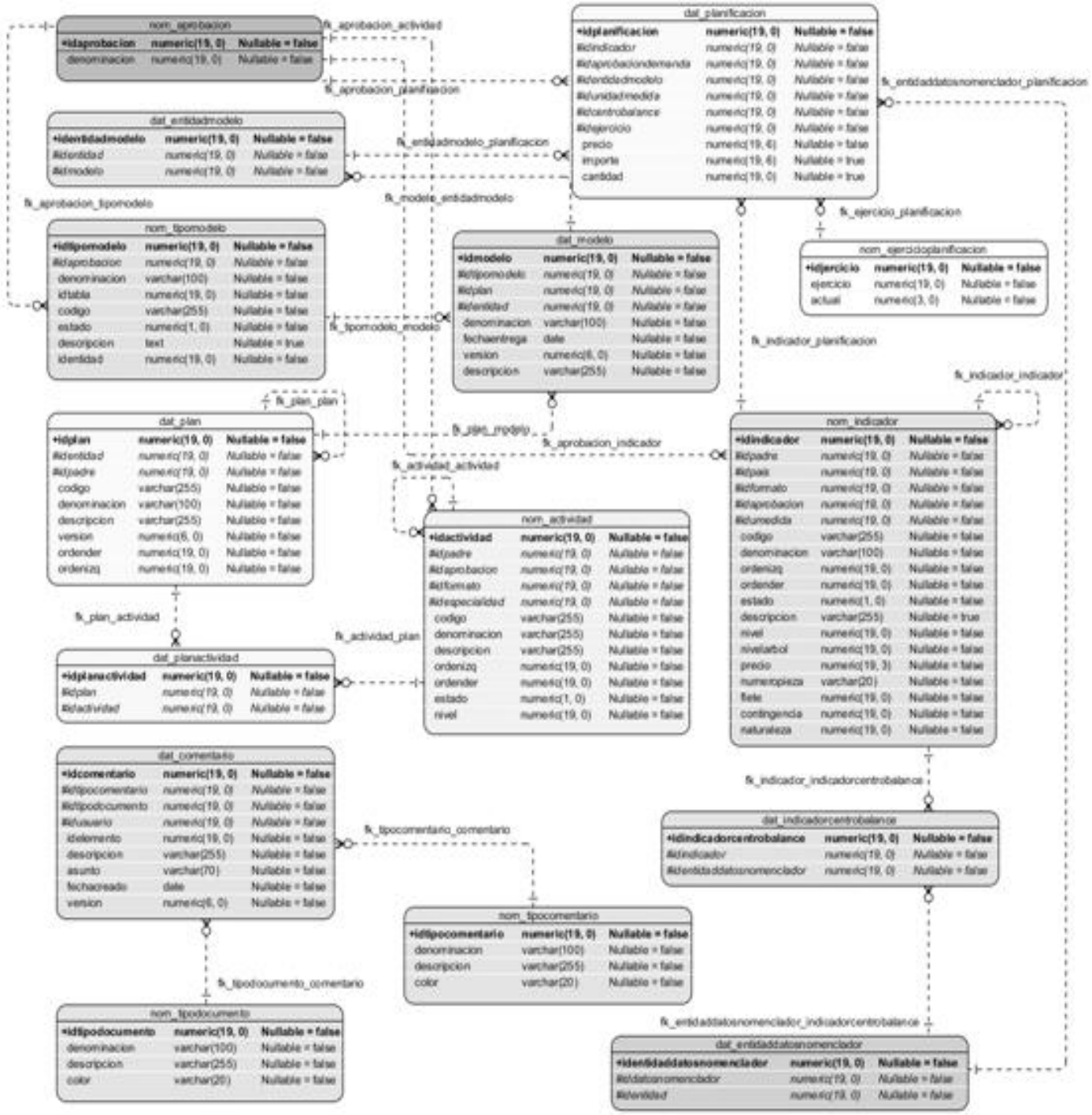

Ilustración 5- Modelo de datos

2.2.7. Patrones de diseño empleados:

El diseño fue elaborado siguiendo patrones, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. En este caso se emplearon los patrones GRASP (en inglés General Responsability Assignment Software Patterns), los que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Los patrones GRASP que se utilizaron son los siguientes:

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, como por ejemplo las clases controladoras y las del modelo. Específicamente: la clase DatPlanModel, en la gestión de los planes, será la responsable de efectuar las operaciones que conciernen a las funciones: Insertar, Eliminar, Listar, Buscar, entre otras, 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 de los componentes de manera general; donde se agruparon las clases en dependencia de los requisitos (Gestionar planes, Gestionar modelos y Gestionar comentarios) 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: GestplanesController, GestmodeloController, GestcomentarioController son un ejemplo de la aplicación de este patrón, las mismas tendrán a cargo la responsabilidad de manejar los eventos dentro del componente.

Durante el diseño del componente se emplearon patrones GOF, específicamente:

Fachada: La aplicación de este patrón en los componentes Plan, Modelo y Comentario se evidencia en la interfaz de servicios simple que se proporciona para establecer la comunicación con otros componentes dentro y fuera del subsistema.

2.3 Implementación de los componentes:

2.3.1 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.

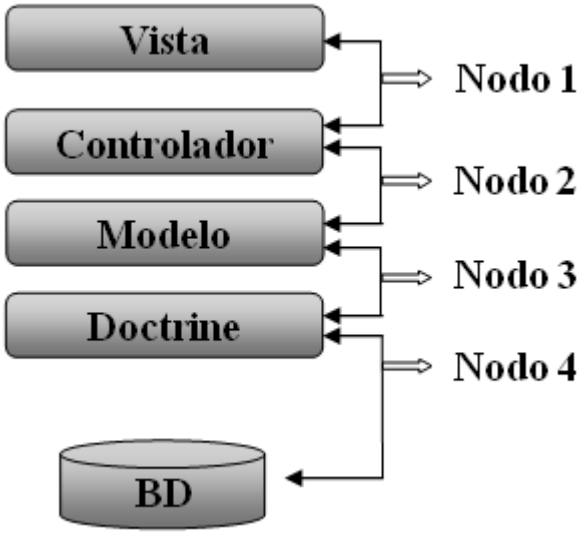

Ilustración 6- Nodos involucrados en la integración.

La comunicación dentro de un mismo componente se ejecuta de forma directa, sin embargo, la que se establece entre los diferentes 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).

Inversión de Control (IoC) es un patrón de diseño pensado para permitir un menor acoplamiento entre componentes de una aplicación y fomentar así la reutilización de los mismos.

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, en el orden necesario y ejecutando el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio. (39)

En la integración interna no se utilizan servicios debido a que se implementó el subsistema de Planificación como si fuese un solo componente. Esta decisión fue tomada por las afectaciones que se presentaron en el rendimiento del subsistema, causado por la cantidad de servicios internos que se implementaron, ya que los componentes estaban muy interrelacionados entre sí.

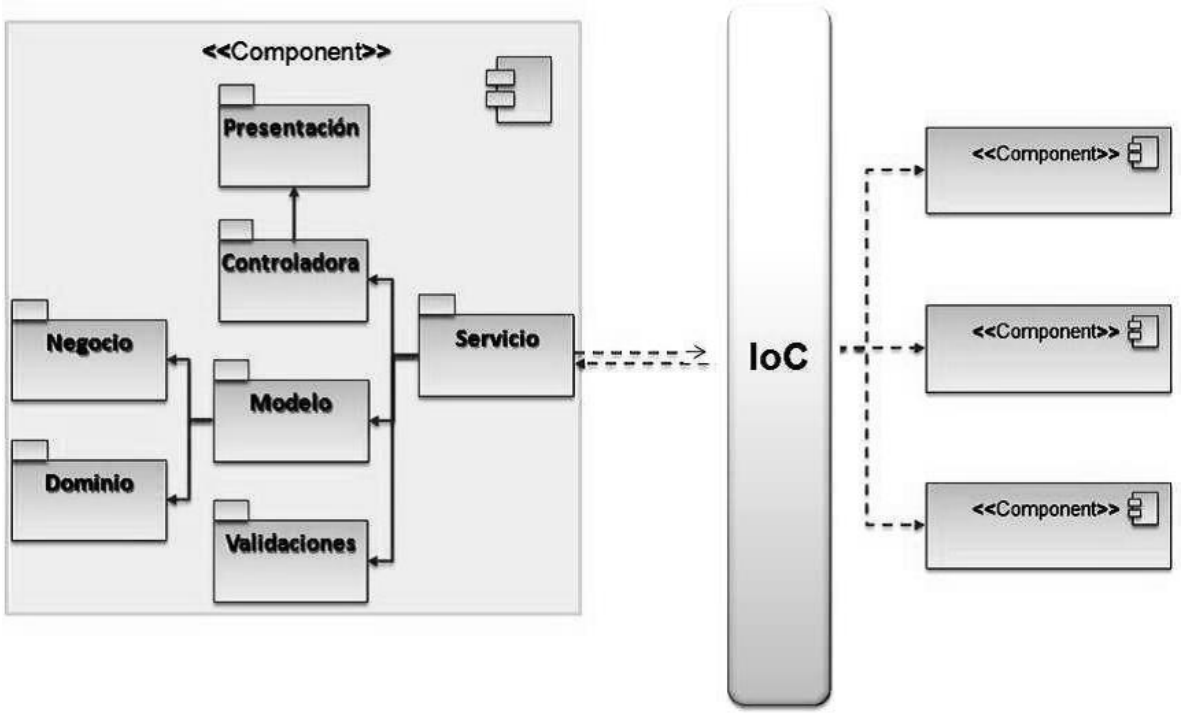

Ilustración 7- Integración entre componentes.

Servicios que recibe el componente como parte de la integración:

+ **obtenerInfoTabla ()**: Servicio que brinda el subsistema Metadato, que le permite al componente Modelo, obtener toda la información de la tabla donde se almacena el tipo de modelo dado.

+ **insertarEnMetadatos ()**: Servicio que brinda el subsistema Metadato, que le permite al componente Modelo, para insertar un nuevo indicador planificado en el esquema de mod_planificacionmeta.

+ **obtenerConfigGridPanel ()**: Servicio que brinda el subsistema Metadato, que devuelve la configuración del componente de interfaz GridPanel, donde se muestran los indicadores planificados en el modelo.

+ **ObtenerUnidadMedidaPorId ()**: Servicio que brinda el componente Unidad de medida del subsistema Configuración, que devuelve las abreviaturas de la unidad de medida dada.

2.3.2 Estándares de código:

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código (40), así como la reutilización a lo largo del proceso de desarrollo de un software. 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.

2.3.2.1 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: **GestionarModelo**. En este caso el nombre de clase está compuesto por 2 palabras iniciadas cada una con letra mayúscula.

- **Nomenclatura según el tipo de clases.**
  - **Clases controladoras**

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

Ejemplo: GestPlanController

- **Clases de los modelos**

Business (Negocio)

Las clases que se encuentran dentro de Business después del nombre llevan la palabra: "Model". Ejemplo: DatModeloModel

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: DatModelo

Generated (Dominio bases)

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: BaseDatModelo

- **Validators** (Clases validadoras).

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

- **Nomenclatura de las funciones 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: insertarModelo. 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: insertarModeloAction ().
Ejemplo de atributo: unidadMedida. El nombre del atributo está compuesto por 2 palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

2.3.3 Descripción de clases por componente y tipo.

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

Clases Controladoras

Las clases controladoras gestionan todo el flujo de datos y operaciones entre la vista y demás clases del negocio.

Clases Modelo

Actúan como intermediarias entre las clases controladoras y las clases entidades. Contienen toda la lógica del negocio, complementan las funcionalidades de las clases controladoras y realizan las funciones de insertar, modificar y eliminar datos.

Clases Entidad

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

2.3.3.1 Componente Plan.

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

Tabla 2- Descripción de la clase GestplanesController

Nombre: GestplanesController
Tipo de clase: Controladora.
Para cada responsabilidad:

Nombre:	Descripción:
init()	Inicializa el controlador.
gestplanesAction()	Renderiza la vista.
adicionarPlanAction()	Adiciona o modifica un plan.
eliminarAction()	Elimina un plan, y los modelos que están dentro de él.
listarEjerciciosAction()	Obtiene la lista de ejercicios configurados en la entidad.
adicionarindicadoresAction()	Adiciona varios indicadores (ya existentes) a un modelo.
listarPlanModeloAction()	Devuelve el árbol de planes y modelos.
getgridinterfacesAction()	Devuelve la interfaz de las columnas que tiene el modelo, según el tipo de modelo.

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

Tabla 3- Descripción de la clase DatPlanModel

Nombre: DatPlanModel	
Tipo de clase: Modelo	
Para cada responsabilidad:	
Nombre:	**Descripción:**
Insertar($idplan, $codigo, $denominacion, $descripcion, $idejercicio, $node, $version, $identidad)	Inserta o modifica un plan.
Eliminar($idplan)	Elimina un plan y el modelo que contiene.
Listar($identidad, $tipo, $node, $parametros)	Lista los planes y los modelos, si el tipo es 0, devuelve los planes de la entidad pasada por parámetro, sino devuelve los modelos. Contiene además un arreglo de parámetros por el cual se va a filtrar la información en dependencia de su contenido.
AdicionarIndicadores($idindicador, $parametros, $idmodelo,	Adiciona indicadores a un modelo seleccionado.

$identidadmodelo, $idejercicio)	

Clases Entidad:

Tabla 4- Descripción de la clase DatPlan

Nombre: DatPlan	
Tipo de clase: Entidad	
Para cada responsabilidad:	
Nombre:	**Descripción:**
GetTodos()	Consulta que devuelve todos los planes.
GetPorLimite($limite = 0, $inicio = 20)	Consulta que devuelve todos los planes por un límite de 20.
getPlanesEntidad($entidad)	Devuelve los planes de la entidad pasada por parámetro.
validarCodigo($codigo, $idplan)	Valida que el código de los planes, no se repitan.
BuscarPlanes($parametros)	Devuelve la lista de los planes, filtrada según los parámetros.
getPlanes($idmodelo)	Devuelve el plan del modelo pasado por parámetro.

Descripción de las clases del componente Modelo (Ver anexo 8).

Descripción de las clases del componente Comentario (Ver anexo 9).

2.4 Conclusiones parciales

Los artefactos generados como resultado de la especificación de los requisitos correspondientes a los componentes Plan, Modelo y Comentario, fueron la base para la realización del presente capítulo a partir de los cuales fue posible llevar a cabo el diseño y la implementación de los componentes, con el fin de responder a las funcionalidades clave en la gestión de los planes, modelos y comentarios. Se elaboraron los artefactos Modelo Entidad-Relación y los Diagramas de Clases de Diseño, así como las descripciones de las clases teniendo en cuenta el marco de trabajo del proyecto y los patrones de diseño. Se definió además la estrategia de integración del componente con el sistema.

Capítulo 3: Validación de la solución propuesta.

3.1 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. A continuación se evalúa el grado con que se le dio cumplimiento a las necesidades del cliente o usuario, para ello se definen inicialmente los tipos y métodos de prueba, y posteriormente se aplica a un requisito y procedimiento específico. Se valida además el diseño propuesto en el capítulo anterior para el desarrollo de los componentes Plan, Modelo y Comentario a través de las métricas de calidad: Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC) teniendo en cuenta las clases y operaciones definidas.

3.2 Validación de la propuesta de diseño.

Métricas empleadas:

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

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

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

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

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

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

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Las métricas escogidas como instrumento para evaluar la calidad del diseño descrito en el capítulo anterior y su relación con los atributos de calidad son las siguientes:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 5- Atributos de calidad evaluados por la métrica TOC.

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

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 6- Criterios de evaluación para la métrica TOC.

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

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 7- Atributos de calidad evaluados por la métrica RC.

Atributo de Calidad.	Modo en que lo afecta.
Responsabilidad.	Un aumento del RC implica un aumento de la responsabilidad asignada a la clase.
Complejidad del mantenimiento.	Un aumento del RC implica un aumento en la complejidad del mantenimiento de la clase.
Reutilización.	Un aumento del RC implica una disminución del grado de reutilización de la clase.
Cantidad de pruebas.	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Tabla 8- Criterios de evaluación de la métrica RC.

Atributo.	Categoría.	Criterio.
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2
Complejidad de mantenimiento.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	>2*Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	<=Promedio
Cantidad de pruebas.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio

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

Tabla 9- Instrumento de evaluación de la métrica TOC.

No	Comp.	Clase	Cant. de Proc.	Resp.	Comp. Implem.	Reutilización
1	Plan	GestplanesController	8	Media	Media	Media
2	Plan	DatPlanModel	5	Baja	Baja	Alta
3	Plan	DatPlan	7	Media	Media	Media
4	Modelo	GestmodeloController	8	Media	Media	Media
5	Modelo	DatModeloModel	5	Baja	Baja	Alta
6	Modelo	DatModelo	6	Media	Media	Media
7	Comentario	GestcomentarioController	4	Baja	Baja	Alta
8	Comentario	DatComentarioModel	3	Baja	Baja	Alta
9	Comentario	DatComentario	5	Baja	Baja	Alta

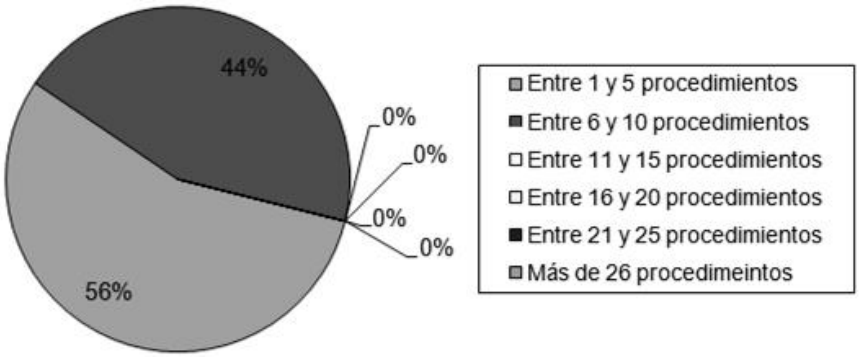

Ilustración 8- Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en cada uno de los atributos (Ver anexo 10).

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto para los componentes está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (56%) posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel medio satisfactorio en el 56% 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)**

Tabla 10- Instrumento de evaluación de la métrica RC.

No	Comp.	Clase	Cant. R. Uso	Acoplam.	Comp. Mant.	Reutiliz.	Cant. Pruebas
1	Plan	GestplanesController	0	Ninguno	Baja	Alta	Baja
2	Plan	DatPlanModel	8	Alto	Media	Media	Media
3	Plan	DatPlan	9	Alto	Alta	Baja	Alta
4	Modelo	GestmodeloController	0	Ninguno	Baja	Alta	Baja
5	Modelo	DatModeloModel	5	Alto	Media	Media	Media
6	Modelo	DatModelo	13	Alto	Alta	Baja	Alta
7	Comentario	GestcomentarioController	0	Ninguno	Baja	Alta	Baja
8	Comentario	DatComentarioModel	3	Alto	Baja	Alta	Baja
9	Comentario	DatComentario	2	Medio	Baja	Alta	Baja

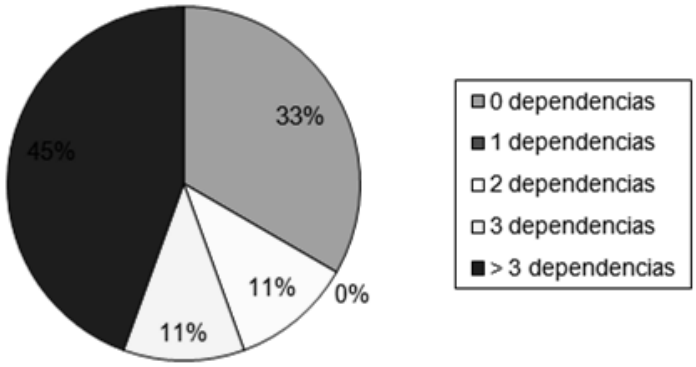

Ilustración 9- Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica RC en cada uno de los atributos (Ver anexo 11).

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto para los componentes Plan, Modelo y Comentario está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (55%) poseen menos de 3 dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 57% de las clases el grado de dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento, la

Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 62%, 62% y 56% de las clases respectivamente.

Matriz de inferencia de indicadores de calidad.

La matriz de inferencia de indicadores de calidad permite conocer si el resultado obtenido de la relación entre los atributos y las métricas asociadas a ellos para los componentes, es positivo o negativo. Es una representación estructurada de las métricas y los atributos utilizados para evaluar la calidad de la solución propuesta. Aplicando una escala numérica a estos resultados (donde 1 corresponde a positivo, 0 a negativo y un signo de menos (-) si este es nulo) realizando un cálculo donde se promedia la suma de los valores obtenidos de un atributo por cada métrica y dividiendo el resultado por la cantidad de métricas evaluadas, se pueden clasificar los atributos en buenos, regulares y malos.

Tabla 11- Resultados de la evaluación de la relación atributo/métrica.

No.	Atributos	TOC	RC	Promedio
1	Responsabilidad	1	(-)	1
2	Complejidad Implementación	1	(-)	1
3	Reutilización	1	1	1
4	Acoplamiento	(-)	1	1
5	Complejidad Mantenimiento	(-)	1	1
6	Cantidad de Pruebas	(-)	1	1

Rango de valores para la evaluación de la relación atributo/métrica (Ver anexo 12).

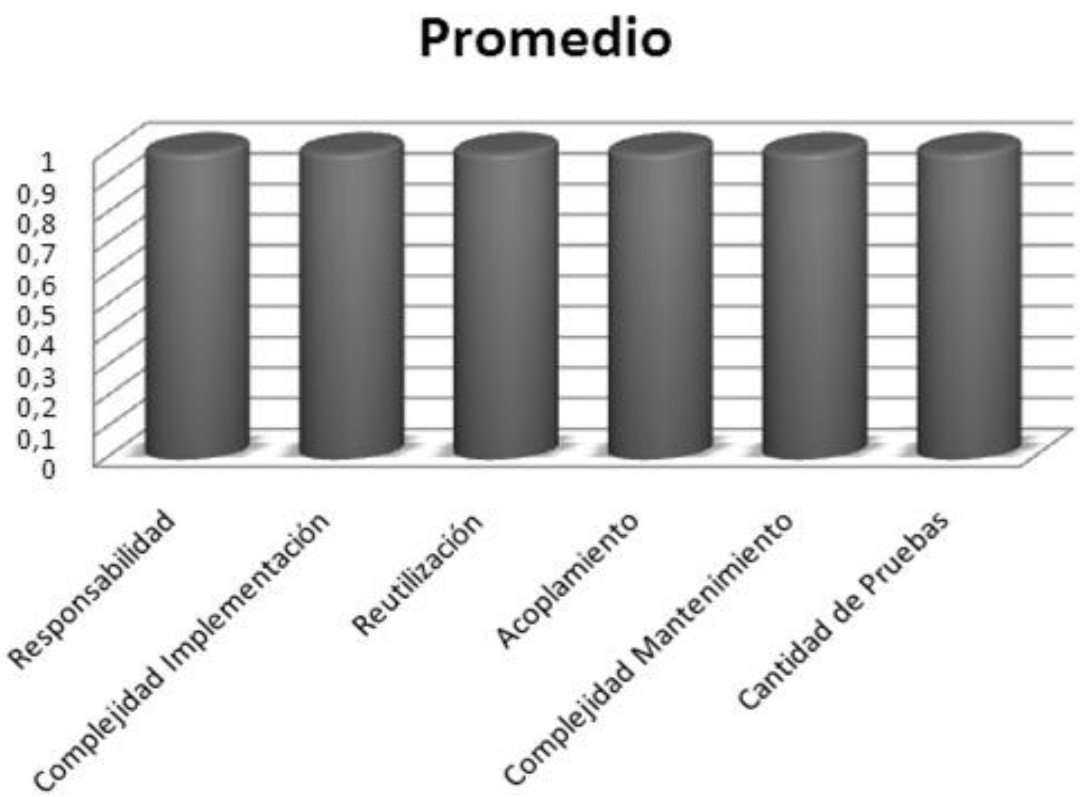

Ilustración 10- Resultados obtenidos de la evaluación de los atributos de calidad.

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 fase de pruebas es una de las más valiosas del ciclo de vida de un software y en ese sentido, deben evaluarse todos los artefactos generados, lo que incluye especificaciones de requisitos, diagramas de diversos tipos, el código fuente y el resto de productos que forman parte de la aplicación, ejemplo: la base de datos.

"La prueba no puede asegurar la ausencia de errores; sólo puede demostrar que existen defectos en el software." (41)

- **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. (42)

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 Pruebas de Caja Blanca o Estructurales.

Se denomina cajas blancas a un tipo de prueba de software que se realiza sobre las funciones internas de un módulo. Así como las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo, las de caja blanca están dirigidas a las funciones internas. Las pruebas de caja blanca se llevan a cabo en primer lugar, sobre un módulo concreto, para luego realizar las de caja negra sobre varios subsistemas. Algunas técnicas de prueba de caja blanca son:

Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de Flujo de Datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

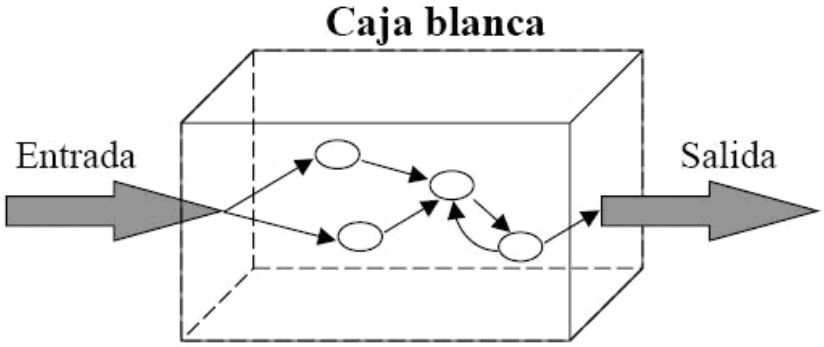

Ilustración 11- Pruebas de caja blanca

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del camino básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Para realizar la prueba es necesario realizar primeramente el análisis de complejidad del algoritmo sobre el que se va a realizar la prueba, con el propósito de calcular los valores de la complejidad ciclomática.

- **Aplicación:**

Según 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 de los componentes Plan, Modelo y Comentario; 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 DatPlanModel, específicamente: Insertar, este por su parte inserta o modifica un plan. La funcionalidad es invocada desde la clase GestplanesController.

Fragmento de código del algoritmo Insertar (Ver anexo 13).

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.

Grafo de flujo asociado al algoritmo Insertar (Ver anexo 14).

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) = (10-9) + 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 3 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 – 6 – 7 – 8 – 9.

Camino básico #2: 1 – 2 – 4 – 5 – 6 – 7 – 8 – 9.

Camino básico #3: 1 – 2 – 3 – 6 – 7 – 9.

Se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo (Ver anexo 14). 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

Camino básico #1: 1 – 2 – 3 – 6 – 7 – 8 – 9.

Descripción:

Los datos de entrada cumplirán con los siguientes requisitos:

Los parámetros $idplan, $codigo, $denominacion, $descripcion, $idejercicio, $node, $version, $identidad no están vacíos, contienen información para que el plan se modifique.

Entrada:

$idplan=100006; $codigo=5698; $denominacion=Plan Económico 2011; $descripcion=Plan económico anual del año 2011; $idejercicio=1; $node=1000008; $version=1; $identidad=10000065.

Resultados esperados del camino básico 1:

Se espera que se modifique un plan hijo.

Resultados del caso de prueba: Se modifica correctamente el plan hijo.

Caso de prueba para el camino básico 2

Camino básico #2: 1 – 2 – 4 – 5 – 6 – 7 – 8 – 9.

Los datos de entrada cumplirán con los siguientes requisitos:

Los parámetros $codigo, $denominacion, $descripcion, $idejercicio, $node, $version, $identidad no están vacíos, contienen información para que el plan se inserte. El parámetro $idplan está vacío.

Entrada:

$idplan=null; $codigo=5698; $denominacion=Plan Económico 2011; $descripcion=Plan económico anual del año 2011; $idejercicio=1; $node=1000008; $version=1; $identidad=10000065.

Resultados esperados del camino básico 2:

Se espera que inserte un nuevo plan hijo.

Resultados del caso de prueba: Se inserta correctamente el plan hijo.

Caso de prueba para el camino básico 3

Camino básico #3: 1 – 2 – 3 – 5 – 6 – 7 – 9 – 12.

Los datos de entrada cumplirán con los siguientes requisitos:

Los parámetros $idplan, $codigo, $denominacion, $descripcion, $idejercicio, $node, $version, $identidad no están vacíos, contienen información para que el plan se modifique.

Entrada:

$idplan=1000009; $codigo=5698; $denominacion=Plan Económico 2011; $descripcion=Plan económico anual del año 2011; $idejercicio=1; $node=0; $version=1; $identidad=10000065.

Resultados esperados del camino básico 3:

Se espera que se modifique el plan padre.

Resultados del caso de prueba: Se modifica correctamente el plan padre.

Resultados esperados de la prueba:

Luego de haberse aplicado pruebas de caja blanca, en específico las pruebas del camino básico, seleccionando diferentes caminos a través del cálculo de la complejidad ciclomática a partir de un segmento de código, se ha arribado a la conclusión de que los resultados obtenidos fueron aceptables ya que se pudo comprobar que el flujo de trabajo de la función está correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.3.2 Pruebas 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.

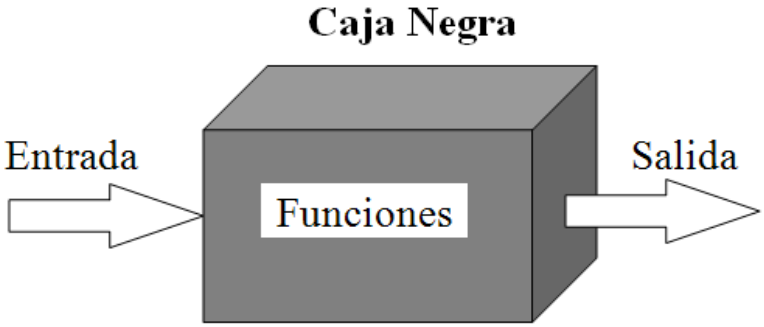

Ilustración 12- Caja Negra

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 Adicionar Plan. 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 de dato.

Caso de prueba de caja negra para validar el requisito funcional Adicionar Plan (Ver anexo 15).

Descripción de variables para el caso de prueba (Ver anexo 16).

Juegos de datos a probar (Ver anexo 17).

Luego de aplicados los métodos de prueba por el equipo de desarrollo al procedimiento Listar y al requisito funcional Adicionar Plan 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).

Para certificar la revisión por parte del cliente de los artefactos generados en el diseño e implementación de los componentes Plan, Modelo y Comentario, se firmó el acta de aceptación por el especialista funcional, el líder de la línea, y el arquitecto (Ver anexo 18).

3.4 Conclusiones Parciales

La calidad del componente desarrollado fue el elemento clave del capítulo que recién concluye. En ese sentido se efectuaron pruebas de software a 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. Los componentes Plan, Modelo y Comentario desde el punto de vista funcional cumplen con los requisitos capturados y especificados en las primeras etapas de desarrollo a partir de las expectativas del cliente.

Conclusiones Generales

Con la realización y culminación de este trabajo se arribaron a las siguientes conclusiones:

- Se efectuó un estudio del estado del arte, con el fin de valorar algunos de los sistemas de planificación actuales, donde queda evidenciada la ausencia de un sistema informático capaz de cumplir con los requisitos establecidos a nivel nacional.

- A partir de los artefactos obtenidos durante la fase de análisis, se realizó un estudio y valoración de los mismos, lo que permitió la realización del diseño.

- A partir de los resultados del estudio del análisis y del diseño propuesto se realizó la implementación de los componentes Plan, Modelo y Comentario.

- Finalmente, se aplicaron métricas al diseño realizado y se aplicaron pruebas al software implementado, las cuales demostraron el cumplimiento de diferentes atributos de calidad a través de resultados favorables, lo que evidencia que los componentes cumplen satisfactoriamente con los requisitos definidos por los clientes.

Recomendaciones

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

- Continuar realizando pruebas de calidad a los componentes Plan, Modelo y Comentario.
- Optimizar los algoritmos para un mejor funcionamiento interno de los componentes.
- Realizar el despliegue de la aplicación en varias entidades para comprobar si cumple desde el punto de vista tecnológico con las expectativas del cliente.

Referencias Bibliográficas

1. **Paz, Norma Sánchez.** *Fundamentos y Metodos Generales de Planificacion.* 2006.

2. **Gómez Veites, Alvaro. y Suarez Rey, Carlos.** *Sistemas de información. Herramientas prácticas para la gestión empresarial.* s.l. : RA-MA, 2006.

3. *El VERSAT-Sarasola: Sistema cubano de Gestión Contable-Financiero.* **Sosa Porteiro, MSc. Marisel y Cobo Morales, Lic. Pedro H.** s.l. : Consultoría Informacional. Casa Consultora DISAIC, 2009.

4. **Márquez Ruiz, Ing. Yosvany, y otros.** *Manual de Usuario V-S Complementos.* Santa Clara : s.n., 2008.

5. **Puentes Hernández, Alianet, Pérez Rodríguez, Maiby y Rodríguez Proenza, Raúl.** *Módulo para la elaboración del Anteproyecto de Planificación del Sistema Cedrux.* La Habana : s.n., 2009.

6. **SAP.** www.sap.com. *www.sap.com.* [En línea] 2010. [Citado el: 29 de 09 de 2010.] www.sap.com/sme/solutions/businessone/index.epx..

7. **SAGEMAS.** www.sagemas.com. *www.sagemas.com.* [En línea] © 2011 Sage Software, Inc. and its affiliated entities. All rights reserved., 2010. [Citado el: 29 de 09 de 2010.] http://www.sagemas.com/products/sagemas500/.

8. **OpenBravo.** openbravo.com. [En línea] 2009. [Citado el: 26 de 09 de 2010.] http://openbravo.com.

9. **EVA.UCI.CU.** Entorno Virtual de Aprendizaje. *EVA.* [En línea] 17 de 9 de 2010. [Citado el: 10 de 02 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=33617.

10. **UCID.** *Proceso de Desarrollo y Gestión de Proyectos de Software.* 2009.

11. **Visconti, Marcello y Astudillo, Hernán.** *Fundamento de Ingeniería de Software.* 2004.

12. *Patrones de Software.* **Cortés, Gloria y Casallas, Rubby.** s.l. : Universidad de Los Andes., 2009.

13. **Larman, G.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.

14. **Lopez., E. Teniente, Ramón., A. Olivé y Gómez Seoane, C.** *Diseño de sistemas de software en UML.* Barcelona : Edicions UPC, 2003.

15. **PBWORKS.** isg3.pbworks.com. *isg3.pbworks.com.* [En línea] 2008. http://isg3.pbworks.com/Patrones-Arquitect%C3%B3nicos..

16. **Martín, Sergio, y otros.** Boletín N°8 Rama de Estudiantes de IEEE-UNED. [En línea] 2007. http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin_8_Octubre_2007.pdf..

17. **Heurtel, Olivier.** *PHP y MySQL Domine el desarrollo de un sitio Web dinamico e interactivo.* s.l. : Ediciones ENI, 2009.

18. **Ruiz, Marcelo.** *Programacion Web Avanzada.* s.l. : MP Ediciones Corp, 2002.

19. **Eguíluz, Javier.** *Introducción a Javascript.* s.l. : Autoedición, 2009.

20. **Pérez, Javier Eguíluz.** *Introduccion a Ajax.* Mexico, DF. : Alfaomega., 2008.

21. **Musciano, Chuck y Kennedy, Bill.** *Html la guía completa.* s.l. : McGraw-hill Interamericana Editores, S.A de C.V, 1999.

22. **Lecomte, Sébastien y Boulanger, Thierry.** *XML practico bases esenciales.* s.l. : Editoral española: Angel Belinchon Calleja, 2009.

23. **Castañeiro Rodríguez, Dainerys y Gonzalez, Jublar.** *Sistema para el control de turnos en las instituciones hospitalarias.* 2009.

24. **TortoiseSVN.** TortoiseSVN. [En línea] 2009. [Citado el: 26 de 09 de 2010.] tortoisesvn.softonic.com/.

25. **Mozilla Corporation.** Mozilla Firefox. *Mozilla Firefox.* [En línea] © Mozilla Europe y Mozilla Foundation -, 2010. [Citado el: 26 de 11 de 2010.] http://www.mozilla.com/es-ES/firefox/.

26. **Technologies., Zend.** ZendStudio. [En línea] Zend Technologies Ltd., 2010. [Citado el: 29 de 09 de 2010.] www.zend.com/es/products/studio/.

27. **Fernández, Marcelo Fidel.** *Técnicas comunes de ataques a equipos con sistema operativo UNIX o derivados. .* Buenos Aires : Creative Commons, 2008.

28. **UCID.** *Políticas y Estándares del Rol Arquitecto de Datos.* 2010.

29. **ExtJs.** extjs.com. *extjs.com.* [En línea] 2008. [Citado el: 31 de 10 de 2010.] http://extjs.com/products/license.php.

30. **Allen, Rob, Lo, Nick y Brown, Steven.** *Zend Framework in Action.* s.l. : Copyright 2009 Manning Publications, 2009.

31. **Wage, Jonathan H.** www.doctrine-project.org. *www.doctrine-project.org.* [En línea] 2009. [Citado el: 11 de 30 de 2010.] http://www.doctrine-project.org/projects/orm.

32. **PostgreSQL.** postgresql.org. *postgresql.org.* [En línea] Copyright © 1996 – 2011 PostgreSQL Global Development Group, Actualizado 2010. [Citado el: 31 de 1 de 2011.] http://www.postgresql.org.

33. **CiberAula.** Linux.CiberAula. [En línea] © 2010 - Todos los derechos reservados Ciberaula, 2010. [Citado el: 31 de 10 de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

34. **Softonic.** aptana.softonic.com. *aptana.softonic.com.* [En línea] SOFTONIC, 2007. http://aptana.softonic.com/.

35. **Associates, O'Reilly &.** *UML en Resumen: Una Rápida Referencia de Escritorio.* 1998.

36. **Jimenez, Monica Constanza Suarez.** *Metricas en arquitectura.* Mexico, DC. : Paseo de la reforma, 2009.

37. **Rivera., Mairelys Fernández González. Osley Zorrilla.** *Diseño e implementación del componente Ajuste al Costo del Sistema Integral de Gestión de Entidades CEDRUX.* 2010.

38. **Definicion.de.** Definicion.de. [En línea] Copyright © 2008-2011 - Definición.de, 2008. http://definicion.de/modelo-de-datos/..

39. **Ruiz Durán, Javier y González Marcaida, Eyonys.** *Diseño e implementación de una herramienta para la gestión de servicios web sobre aplicaciones PHP.* 2010.

40. **Arquitectura., Proyecto ERP Cuba. Línea de.** *Normas y estándares de Codificación del ERP.* 2008.

41. **Pressman., R.** *Ingeniería de Software. Un enfoque práctico.* 1998.

Bibliografía

1. **Paz, Norma Sánchez.** *Fundamentos y Metodos Generales de Planificacion.* 2006.

2. **Gómez Veites, Alvaro. y Suarez Rey, Carlos.** *Sistemas de información. Herramientas prácticas para la gestión empresarial.* s.l. : RA-MA, 2006.

3. *El VERSAT-Sarasola: Sistema cubano de Gestión Contable-Financiero.* **Sosa Porteiro, MSc. Marisel y Cobo Morales, Lic. Pedro H.** s.l. : Consultoría Informacional. Casa Consultora DISAIC, 2009.

4. **Márquez Ruiz, Ing. Yosvany, y otros.** *Manual de Usuario V-S Complementos.* Santa Clara : s.n., 2008.

5. **Puentes Hernández, Alianet, Pérez Rodríguez, Maiby y Rodríguez Proenza, Raúl.** *Módulo para la elaboración del Anteproyecto de Planificación del Sistema Cedrux.* La Habana : s.n., 2009.

6. **SAP.** www.sap.com. *www.sap.com.* [En línea] 2010. [Citado el: 29 de 09 de 2010.] www.sap.com/sme/solutions/businessone/index.epx..

7. **SAGEMAS.** www.sagemas.com. *www.sagemas.com.* [En línea] © 2011 Sage Software, Inc. and its affiliated entities. All rights reserved., 2010. [Citado el: 29 de 09 de 2010.] http://www.sagemas.com/products/sagemas500/.

8. **OpenBravo.** openbravo.com. [En línea] 2009. [Citado el: 26 de 09 de 2010.] http://openbravo.com.

9. **EVA.UCI.CU.** Entorno Virtual de Aprendizaje. *EVA.* [En línea] 17 de 9 de 2010. [Citado el: 10 de 02 de 2011.] http://eva.uci.cu/mod/resource/view.php?id=33617.

10. **UCID.** *Proceso de Desarrollo y Gestión de Proyectos de Software.* 2009.

11. **Visconti, Marcello y Astudillo, Hernán.** *Fundamento de Ingeniería de Software.* 2004.

12. *Patrones de Software.* **Cortés, Gloria y Casallas, Rubby.** s.l. : Universidad de Los Andes., 2009.

13. **Larman, G.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 1999.

14. **Lopez., E. Teniente, Ramón., A. Olivé y Gómez Seoane, C.** *Diseño de sistemas de software en UML.* Barcelona : Edicions UPC, 2003.

15. **PBWORKS.** isg3.pbworks.com. *isg3.pbworks.com.* [En línea] 2008. http://isg3.pbworks.com/Patrones-Arquitect%C3%B3nicos..

16. **Martín, Sergio, y otros.** Boletín N°8 Rama de Estudiantes de IEEE-UNED. [En línea] 2007. http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin_8_Octubre_2007.pdf..

17. **Heurtel, Olivier.** *PHP y MySQL Domine el desarrollo de un sitio Web dinamico e interactivo.* s.l. : Ediciones ENI, 2009.

18. **Ruiz, Marcelo.** *Programacion Web Avanzada.* s.l. : MP Ediciones Corp, 2002.

19. **Eguíluz, Javier.** *Introducción a Javascript.* s.l. : Autoedición, 2009.

20. **Pérez, Javier Eguíluz.** *Introduccion a Ajax.* Mexico, DF. : Alfaomega., 2008.

21. **Musciano, Chuck y Kennedy, Bill.** *Html la guía completa.* s.l. : McGraw-hill Interamericana Editores, S.A de C.V, 1999.

22. **Lecomte, Sébastien y Boulanger, Thierry.** *XML practico bases esenciales.* s.l. : Editoral española: Angel Belinchon Calleja, 2009.

23. **Castañeiro Rodríguez, Dainerys y Gonzalez, Jublar.** *Sistema para el control de turnos en las instituciones hospitalarias.* 2009.

24. **TortoiseSVN.** TortoiseSVN. [En línea] 2009. [Citado el: 26 de 09 de 2010.] tortoisesvn.softonic.com/.

25. **Mozilla Corporation.** Mozilla Firefox. *Mozilla Firefox.* [En línea] © Mozilla Europe y Mozilla Foundation -, 2010. [Citado el: 26 de 11 de 2010.] http://www.mozilla.com/es-ES/firefox/.

26. **Technologies., Zend.** ZendStudio. [En línea] Zend Technologies Ltd., 2010. [Citado el: 29 de 09 de 2010.] www.zend.com/es/products/studio/.

27. **Fernández, Marcelo Fidel.** *Técnicas comunes de ataques a equipos con sistema operativo UNIX o derivados. .* Buenos Aires : Creative Commons, 2008.

28. **UCID.** *Políticas y Estándares del Rol Arquitecto de Datos.* 2010.

29. **ExtJs.** extjs.com. *extjs.com.* [En línea] 2008. [Citado el: 31 de 10 de 2010.] http://extjs.com/products/license.php.

30. **Allen, Rob, Lo, Nick y Brown, Steven.** *Zend Framework in Action.* s.l. : Copyright 2009 Manning Publications, 2009.

31. **Wage, Jonathan H.** www.doctrine-project.org. *www.doctrine-project.org.* [En línea] 2009. [Citado el: 11 de 30 de 2010.] http://www.doctrine-project.org/projects/orm.

32. **PostgreSQL.** postgresql.org. *postgresql.org.* [En línea] Copyright © 1996 – 2011 PostgreSQL Global Development Group, Actualizado 2010. [Citado el: 31 de 1 de 2011.] http://www.postgresql.org.

33. **CiberAula.** Linux.CiberAula. [En línea] © 2010 - Todos los derechos reservados Ciberaula, 2010. [Citado el: 31 de 10 de 2010.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

34. **Softonic.** aptana.softonic.com. *aptana.softonic.com.* [En línea] SOFTONIC, 2007. http://aptana.softonic.com/.

35. **Associates, O'Reilly &.** *UML en Resumen: Una Rápida Referencia de Escritorio.* 1998.

36. **Jimenez, Monica Constanza Suarez.** *Metricas en arquitectura.* Mexico, DC. : Paseo de la reforma, 2009.

37. **Rivera., Mairelys Fernández González. Osley Zorrilla.** *Diseño e implementación del componente Ajuste al Costo del Sistema Integral de Gestión de Entidades CEDRUX.* 2010.

38. **Definicion.de.** Definicion.de. [En línea] Copyright © 2008-2011 - Definición.de, 2008. http://definicion.de/modelo-de-datos/..

39. **Ruiz Durán, Javier y González Marcaida, Eyonys.** *Diseño e implementación de una herramienta para la gestión de servicios web sobre aplicaciones PHP.* 2010.

40. **Arquitectura., Proyecto ERP Cuba. Línea de.** *Normas y estándares de Codificación del ERP.* 2008.

41. **Pressman., R.** *Ingeniería de Software. Un enfoque práctico.* 1998.

42. **Ramirez., J.** *Unidad de Programación. Métodos de Prueba del Software. .*

43. **Web, Maestros del.** www.maestrosdelweb.com. *www.maestrosdelweb.com.* [En línea] 2009. [Citado el: 29 de 09 de 2010.] http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/.

44. www.definanzas.com. [En línea] 02 de 06 de 2008. [Citado el: 26 de 09 de 2010.] http://definanzas.com/2008/06/02/erp-sistemas-de-gestion-empresarial/..

45. *Vistos desde otros horizontes. Revista de Software Libre ATIX.* 2009.

46. *Tecnología para la Gestión de Procesos de Negocio.* **Ruiz, Francisco.** 2006.

47. **Systems., Sparx.** Sparx Systems.Tutorial UML 2. *Diagrama de Secuencia.* [En línea] 2007. [Citado el: 25 de 1 de 2011.] http://www.sparxsystems.com.ar/resources/tutorial/uml2_sequencediagram.html..

48. **UCID.** *Requerimientos y versiones de las tecnologías y herramientas del marco de trabajo Sauxe 1.5.4 Beta.* 2010.

49. **Planificación, Ministerio de Economía y.** *Plan 2011. Instrucciones para su Desagregación y Ejecución. .* 2010.

50. **tecnología, Subdirección de.** *Nueva Estructura del Marco de Trabajo.* 2009.

51. **Mehdi Achour. Friedhelm Betz.Antony Dovgal, etc...** *Manual de PHP.* 2008.

52. *Información Tecnológica.* 5, 2000, Vol. 11.

53. **Planificación, Ministerio de Economía y.** *Indicaciones Metodológicas Complementarias para la Elaboración del Plan 2011.* 2010.

54. **Ramos, Anay Carrillo.** *Enseñanza de la Metodología RUP de Ingeniería del Software.*

55. **Vesterinen, Konsta.** *Doctrine Manual.* 2008.

56. Diseño y Programación Orientado a Objetos. 1998.

57. **Marcaida., Javier Ruiz Durán. Eyonys González.** *Diseño e implementación de una herramienta para la gestión de servicios web sobre aplicaciones PHP.* 2010.

58. **Alvariño., Ing. Danelys Brito González. Ing. Dionisdel Ponce Santana. Ing. Victor Luis Borroto.** *COMPONENTE PARA LA GENERACIÓN DE CONCEPTOS Y ATRIBUTOS DINÁMICOS.* 2010.

59. **Entidades., Centro de Informatización de Gestión de.** *Arquitectura de Software.* 2010.

60. **Maza., Miguel Angel Sanchez.** *JavaScript.* s.l. : Innovacion y cualificación SSL., 2001.

61. **Microsoft.** MSDN. *MSDN.* [En línea] © 2011 Microsoft. Reservados todos los derechos., 2011. [Citado el: 25 de 03 de 2011.] http://msdn.microsoft.com/es-es/library/aa292164(v=vs.71).aspx.

Glosario de términos

1. Algoritmo: Cualquier procedimiento computacional bien definido mediante un conjunto de reglas que da solución a instancias de un problema (entrada) produciendo un valor o conjunto de valores (salida).
2. Aplicación web: En la ingeniería software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, Javascript, Java, etc.) en la que se confía la ejecución al navegador.
3. Código abierto (en inglés Open Source): Término con el que se conoce al software distribuido y desarrollado libremente.
4. Componente: Un componente es una clase de uso específico, que puede ser configurada o utilizada de forma visual desde el entorno de desarrollo.
5. Entidad: Empresa, unidad presupuestada u otro tipo de organización similar con una gestión económica, financiera, organizativa, técnica, productiva, comercial, laboral y contractual, con autonomía controlada, en cumplimiento de lo establecido por el Gobierno.
6. Framework: Es una estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
7. Herramientas: Es un objeto elaborado a fin de facilitar la realización de una tarea mecánica que requiere de una aplicación correcta de energía.
8. Talón: es un término que se utiliza para denominar las plantillas o modelos que se confeccionan para la captación de la información primaria en la confección de los planes, una lista de filas cuyos datos deben ser introducidos por el usuario.
9. Proforma: es el término utilizado para identificar los talones ya con los datos vertidos en cada fila, es decir los modelos ya completados por las distintas unidades.
10. Indicador: es un término fundamental en la explotación de posibilidades tales como la planificación de ingresos por servicios y validación del cumplimiento de normas.
11. Normas: es un término que se utiliza para agrupar uno o varios parámetros normables con el fin de restringir el gasto en determinada actividad.