Universidad de las Ciencias Informáticas

Facultad 3

Componente para la configuración de los presupuestos del sistema XEDRO ERP

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Wendolyn Rodríguez León

Yunior Feria Chapman

Tutores:

MSc. Maylé Díaz Castro

MSc. Johanny Rivera López

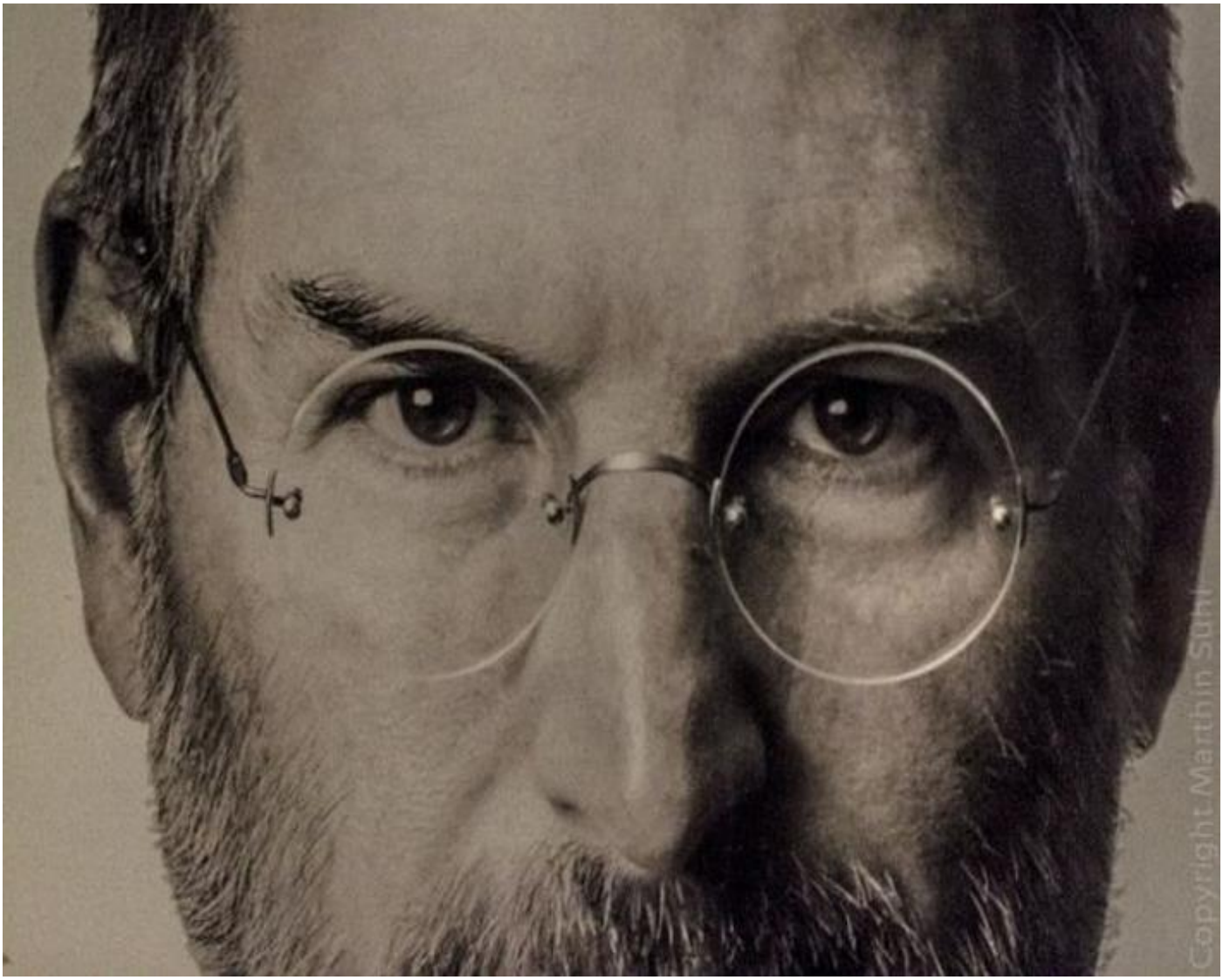
Co-Tutores:

Ing. Tamara Rodríguez Sánchez

Ing. Carlos Heriberto Cordoví García

La Habana, Cuba 2014

"Año 56 de la Revolución"



"Tu tiempo está limitado, así que no lo desaproveches viviendo la vida de algún otro. No te dejes arrastrar por los dogmas, que es lo mismo que vivir con los resultados del pensamiento de otras personas. No dejes que el ruido de las opiniones de otros ahoguen completamente tu voz interior. Y más importante, ten el valor de seguir a tu corazón y a tu intuición. Ellos, de algún modo, ya saben en lo que verdaderamente te quieres convertir. Todo lo demás es secundario."

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Wendolyn Rodríguez León	Yunior Feria Chapman
_____	_____
MSc. Maylé Díaz Castro	MSc. Yohany Rivera López

Dedicatoria

Dedico el presente trabajo de diploma únicamente a una persona especial, a mi mamá, que ha sido mi ejemplo, mi guía, mi razón de ser y te agradeceré siempre por haberme guiado mis pasos hasta convertirme en la mujer que soy hoy.

Wendolyn

A mi madre, padre, padrastro y a mi tía Joaquina por ser las personas que con mayor entrega, esfuerzo y responsabilidad han llevado mi formación personal.

Yunior

Agradecimientos

A mis padres, por brindarme su apoyo, dedicación y amor.

A mi tía Mildrey por ser como una segunda madre para mí.

A mi hermanito Kevin por ser cariñoso, un hermano ejemplar y cuidar de mami todo este tiempo que no estuve allá, a mi padrastro, a todos mis abuelos, mis primas y a mi familia en general por su constante preocupación y por ayudarme a cumplir mi sueño.

Un agradecimiento muy especial a mi novio Rembe, gracias por ayudarme en todos estos años de universidad y por tantos momentos lindos.

A mi compañero de tesis Yunior por su dedicación, por su amistad y por esos momentos en que este trabajo nos hizo compartir.

A mis amigos de toda la vida y a los que hice en la universidad.

A mis amigas Greter, Diana, Lisandra, Dayana, Maylin, entre otras, a mis compañeras de todos los apartamentos por los que he pasado, a mis compañeros de aula, a mi equipo de seminario y en especial a mi compañero de mesa Yandry por robarme los conocimientos, a todos mis compañeros de proyecto por su apoyo, a los profesores y agradecer a todos los que han contribuido con la realización de este trabajo y en mi formación profesional.

Wendolyn

A mi madre por ser la mujer más valiente y dedicada que he conocido.

A mi padre por nuestras largas conversaciones y los consejos siempre oportunos que me ha ofrecido.

A mi padrastro por asumir mi crianza como si fuese su hijo propio.

A mi tía Joaquina porque siempre ha estado como una madre más en los momentos que la he necesitado.

A mis abuelos, tíos y primos. Y de manera especial a mis hermanas Yamila, Jessica y Yeliexis así como a mis primos Alexis, Yury y Alexander.

A Victor quien ha venido ha sustituir el espacio del hermano varón que siempre quise tener. Con quien he compartido muchísimos de los momentos inolvidables que me llevo de la universidad.

A mi otro hermano de la universidad Jorge que ha pesar de ser un gordo insoportable ha estado siempre en los momentos duros de verdad a mi lado.

A mi compañera y amiga Wendy con quien tuve el placer de realizar este trabajo de diploma.

A mis amigos Eiler, Yorguy, Luis Manuel, Niurka, Rubén, Fabra, Nicolau, Julio César, Oscar y Yorjanis por los tantos momentos que compartimos juntos en esta aventura.

A todos los que hemos estado juntos en el trabajo de la juventud.

Al grupo con el que trabajé en el secretariado de la FEU.

A los profesores de la Facultad que hicieron posible que realizara este sueño.

A todos mis compañeros de brigada.

Yunior

Resumen

En la actualidad, a las organizaciones en sentido general les resulta de vital importancia contar con información actualizada en cuanto al estado de sus negocios. Esta constante necesidad ha provocado que muchas de ellas se encuentren inmersas en la informatización de sus procesos de gestión empresarial con la intención de agilizar los mismos. Contar con información actualizada sobre la gestión presupuestaria que refleje lo sucedido o lo que está sucediendo con los planes de los presupuestos de gastos, ingresos y/o inversiones es fundamental para una empresa. El presente trabajo de diploma describe el desarrollo del componente para la configuración de los presupuestos del sistema XEDRO ERP, tomando como base las normas y resoluciones emitidas por el Ministerio de Finanzas y Precios de Cuba. La propuesta de solución desarrollada, permite una vez emitido el presupuesto a la entidad, realizar el proceso de configuración del mismo. Con la utilización del componente desarrollado se contribuye a disminuir el tiempo en la gestión de los presupuestos de acuerdo a las técnicas actuales de gestión presupuestaria.

PALABRAS CLAVES: Sistema de planificación de recursos empresariales, notificación del presupuesto, desagregación del presupuesto, configuración del presupuesto.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURA

ÍNDICE DE TABLA

Introducción

En la actualidad las tecnologías de información juegan un papel importante en las estrategias de negocios, debido a que están cambiando la forma en que las empresas gestionan sus procesos. Los sistemas de información permiten a las compañías lograr ventajas competitivas de diferentes maneras (1). Hoy como nunca antes, las empresas necesitan agrupar y controlar su información, garantizando que esta sea íntegra y confiable para asegurar el cumplimiento de sus objetivos estratégicos y una mejor toma de decisiones. Una de las alternativas para garantizar lo antes mencionado son los Sistemas de Planificación de Recursos Empresariales (ERP[1]). Los ERP son soluciones informáticas cuyo objetivo es gestionar la información a través de las diferentes áreas de la empresa para agilizar tareas, mejorar los procesos de producción y reducir costos (2).

La autenticidad de un ERP está dada por sus tres características esenciales: integrables porque tratan a las diferentes áreas o departamentos de una organización como elementos unidos por la información que generan. Adaptables pues son sistemas capaces de ajustarse a cualquier empresa, independientemente del sector al que pertenezcan y modulares pues estos sistemas están formados por un número específico de módulos, independientes entre sí, pero que a la vez están comunicados, lo que permite una gran adaptabilidad a las empresas de acuerdo a su tamaño y cantidad de recursos (3).

Son sistemas estructurados para satisfacer las demandas de soluciones de gestión empresarial basados en el ofrecimiento de una solución completa que permite a las empresas evaluar, implementar y administrar con mayor facilidad su negocio. La implantación de un ERP sirve de soporte para una administración eficiente. En los últimos tiempos han alcanzado un gran auge en el mercado empresarial, pues las compañías están obligadas cada día más a ser competitivas por lo que se hace necesaria la integración de sus flujos internos de información y sus relaciones comerciales externas (4).

A través de los Organismos de la Administración Central del Estado (OACE) son muchos los esfuerzos que se realizan hoy en Cuba para lograr un reordenamiento empresarial que permita una gestión eficiente y mayor solvencia económica de las empresas cubanas. Aspirar a estos dos indicadores bajo las condiciones actuales de competencia en el mercado sin tener en cuenta las posibilidades que brinda el desarrollo de la informática es casi utópico. El Estado cubano es consciente de esta realidad y como parte del reordenamiento empresarial incluye la modernización y empleo de las nuevas tecnologías en la gestión de los recursos humanos, financieros y de otro tipo dentro de la empresa.

La Universidad de las Ciencias Informáticas (UCI) es uno de los principales centros nacionales que desarrolla y promueve el uso de las tecnologías es función del gestión empresarial. En el Centro para la

[1] Enterprise Resource Planning

Informatización de la Gestión de Entidades (CEIGE) se desarrolla el producto XEDRO ERP conocido como ERP- Cuba. Antes de XEDRO ERP se han liberado productos como el Versat Sarasola, SISTCONT y RODAS XXI, pero con la limitante de estar dirigidos a resolver problemas dentro de un dominio restringido.

Con XEDRO ERP se busca proporcionar un producto completo y adaptable a todas las esferas de la economía. Se han definido los subsistemas: Inventario, Activos fijos tangibles e intangibles, Facturación, Cobros y pagos, Caja, Banco, Estructura y composición, Capital humano, Costos y procesos, Contabilidad, Configuración y Multimoneda (3).

La configuración de los presupuestos es uno de los procesos fundamentales de las entidades cubanas. Un presupuesto es una herramienta de gestión conformada por un documento donde se cuantifican pronósticos o previsiones de diferentes conceptos de un negocio (5). Los presupuestos no están relacionados exclusivamente con los ingresos y gastos de una empresa, sino que es posible además hacer uso de estas herramientas para cuantificar pronósticos de los cobros, los pagos de las deudas, los productos que se realizarán o los materiales que necesitará la empresa para producir dichos productos (5). Realizar previsiones o pronósticos presupuestarios es fundamental para cualquier entidad ya que les permitirá planificar, coordinar y controlar las operaciones de la institución.

En las entidades cubanas el proceso de configuración de los presupuestos se realiza de forma manual o utilizando Excel, ya que los sistemas contables utilizados en la mayoría de ellas no contemplan un componente para esta actividad. Este proceso se realiza al inicio de cada año fiscal por las direcciones económicas de las entidades, quienes tienen la responsabilidad de realizar la desagregación del mismo en todas las áreas de la institución, en función de la propuesta de plan enviada a los organismos superiores para su aprobación. Sin embargo cuando se le es notificada la aprobación del plan, si este sufrió alguna modificación se debe realizar el mismo proceso, provocando duplicidad de esfuerzo, factibilidad a errores humanos, lentitud en el procesamiento de la información e insatisfacción de los especialistas de la dirección y de las áreas.

Actualmente la notificación y desagregación por áreas en las entidades se realiza en la dirección de Planificación y Estadística, teniendo en cuenta el consumo del año anterior y en conciliación con los directores de las mismas. El personal de planificación y estadística mediante un documento impreso propone las desagregaciones del presupuesto por las distintas áreas. En caso de existir errores como resultado de las revisiones con los directores de las áreas será necesario realizar nuevamente el proceso, trayendo una influencia desfavorable en el tiempo de realización del proceso de configuración del presupuesto. Al tenerse registrada la información en Excel cuando se realizan modificaciones del mismo se introducen errores al escribir los conceptos y en la asignación del monto a los mismos, errores humanos que llevan a que el proceso deba ser repetido nuevamente.

Según datos ofrecidos por la directora de la oficina de Planificación y Estadística de la Universidad de las Ciencias Informáticas, para los años 2012, 2013 y 2014 se empleó un tiempo de veinte, diecinueve y diecinueve días respectivamente para realizar el proceso de configuración del presupuesto:

Proceso de configuración de los presupuestos			
Año	Fecha de llegada del presupuesto	Fin del proceso de configuración	Cantidad de días
2012	20 de enero	16 de febrero	20
2013	21 de enero	15 de febrero	19
2014	3 febrero	27 de febrero	19

Por tanto el **problema a resolver** queda formulado de la siguiente manera: La gestión de los datos correspondientes a los presupuestos de gastos, ingresos e inversiones en las entidades cubanas se realiza de forma manual o utilizando Excel lo cual provoca lentitud en el procesamiento de la información.

Se define como **objeto de estudio:** Proceso de gestión de presupuesto económico, tomando como **campo de acción:** Sistemas para la configuración del presupuesto económico. Para dar respuesta al problema planteado se trazó el siguiente **objetivo general:** Desarrollar un componente para el sistema XEDRO ERP que permita la gestión de la configuración de los presupuestos económicos para aumentar la velocidad en el procesamiento de la información.

A partir del cual se desglosan los siguientes **objetivos específicos:**

- Realizar el estudio del estado del arte de la investigación para identificar funcionalidades y normas relacionadas con la configuración de los presupuestos.
- Realizar el análisis y diseño del componente para la configuración de los presupuestos del sistema XEDRO ERP.
- Implementar el componente para la configuración de los presupuestos del sistema XEDRO ERP, mediante las herramientas y tecnologías seleccionadas.
- Validar que la solución implementada permite aumentar la velocidad en el procesamiento de la información.

Estableciendo como **idea a defender:** Si se desarrolla un componente para el sistema XEDRO ERP que permita la gestión de la configuración de los presupuestos se podrá aumentar la velocidad en el procesamiento de la información.

Tareas a cumplir:

- Estudio y análisis de los sistemas para la gestión de los presupuestos económicos.
- Estudio y análisis de los sistemas para la configuración de los presupuestos económicos.
- Estudio y selección de la tecnología a utilizar.
- Estudio y selección de la metodología a utilizar.
- Identificación de los requisitos asociados al componente de configuración de los presupuestos del sistema XEDRO ERP.
- Descripción de los requisitos asociados al componente de configuración de los presupuestos del sistema XEDRO ERP.
- Validación de los requisitos asociados al componente de configuración de los presupuestos del sistema XEDRO ERP.
- Diseño de los artefactos del negocio asociados al componente de configuración de los presupuestos del sistema XEDRO ERP.
- Validación del diseño asociado al componente de configuración de los presupuestos del sistema XEDRO ERP.
- Implementación de las funcionalidades relacionadas a la modificación de las versiones de los presupuestos así como mantener el histórico de todas las versiones que el mismo posea.
- Implementación de las funcionalidades relacionadas con la notificación del presupuesto en las entidades.
- Implementación de las funcionalidades relacionadas a la desagregación del presupuesto por las distintas áreas de la entidad.
- Validación de los resultados obtenidos a través de pruebas de caja negra y caja blanca al sistema para comprobar su correcto funcionamiento.
- Validación de los resultados obtenidos a través pruebas de liberación interna en el centro, que valide la calidad de la implementación desarrollada.

En el desarrollo de la investigación se utilizaron **métodos científicos** tanto teóricos como empíricos.

Métodos Teóricos

El método **Histórico-Lógico** permitió el estudio de los principales sistemas ERP desarrollados, determinando sus principales características, ventajas y desventajas para tenerlas en cuenta en la implementación del componente de XEDRO ERP.

El método **Analítico-Sintético** permitió realizar un análisis de la teoría y la literatura relacionada con la gestión de los presupuestos de los sistemas identificados como objeto de estudio, de manera que fue

posible evaluar los módulos, componentes y/o funcionalidades. Realizando además un análisis de los elementos comunes y fundamentales de cada teoría y/o aproximación.

La **Modelación** fue otro de los métodos empleados en la gestión de los procesos a desarrollar en la implementación, y para obtener los diagramas necesarios para el entendimiento del negocio y mantenimiento del producto.

Métodos Empíricos

Observación: este método ha permitido percibir a partir de la situación real que se está investigando, cómo se desarrolla el proceso que constituye el objeto de estudio.

El método de la **medición** permitió la aplicación de métricas y estándares de calidad a la aproximación implementada (prueba de software), garantizando la calidad de la propuesta.

El presente trabajo está estructurado en tres capítulos:

Capítulo 1 Fundamentación teórica: en este capítulo se realiza un estudio de las características, ventajas e inconvenientes de los principales módulos desarrollados para la gestión de presupuestos en los sistemas ERP, los principales conceptos asociados al presupuesto. Se hace referencia al modelo de desarrollo, patrones de diseño y métricas de software. Se definen las herramientas necesarias para desarrollar la propuesta de solución.

Capítulo 2 Modelado del negocio, análisis y diseño: en este capítulo se desarrolla la modelación del proceso, obteniéndose el modelo conceptual y diagrama de proceso de negocio, se identifican, describen y validan los principales requisitos del sistema, se realiza el diagrama de clases, se construye el modelo de datos, el diagrama de secuencia y se expone la utilización de los patrones de diseño, arquitectónico y se aplican métricas para la validación del diseño propuesto.

Capítulo 3 Implementación y prueba: en este capítulo se realiza la descripción de la implementación. Se muestran los diagramas de componentes, diagrama de despliegue y un fragmento del código de la propuesta de solución. Las pruebas de caja blanca y caja negra, así como la validación de las variables de la investigación.

Capítulo 1 Fundamentación teórica

1.1 Introducción

En este capítulo se realiza un estudio sobre los conceptos fundamentales asociados a la gestión de la configuración de los presupuestos de las entidades cubanas, así como de las características, ventajas e inconvenientes de algunos componentes para la configuración de presupuestos. Se definen las herramientas para realizar un correcto análisis, diseño, implementación y prueba de la propuesta de solución.

1.2 Marco conceptual

Con la finalidad de facilitar la comprensión del presente trabajo se enuncian los conceptos fundamentales utilizados en el mismo.

ERP

Según Víctor Manuel Herrera un ERP es "*un paquete informático que cubre de forma parcial o total las áreas funcionales de la empresa*"(6).

Ricardo Montaño Badilla plantea: *"Los ERP son un tipo de software que permiten a las empresas controlar la información que se genera en cada departamento y a cada nivel de la misma"* (7).

A partir de las definiciones estudiadas y tomando en cuenta las necesidades de las entidades cubanas en manejar la información de sus distintas áreas, la que más se ajusta a la investigación es: Los ERP son un tipo de software que permiten a las empresas integrar y controlar la información de los distintos departamentos, posibilitando una mejor toma de decisiones.

Presupuesto

Según el Ministerio de Producción Peruano se llama presupuesto al *"cálculo anticipado de los ingresos y gastos de una actividad económica (personal, familiar, un negocio, una empresa, una oficina) durante un período, por lo general en forma anual"* (8).

Según (9) los objetivos de un presupuesto son:

- Planear sistemáticamente todas las actividades que la empresa debe desarrollar en dinero y volúmenes, en un período determinado.
- Controlar el manejo de ingresos y egresos de la empresa y medir los resultados cuantitativos, cualitativos.
- Fijar responsabilidades en las diferentes dependencias de la organización para lograr el cumplimiento de las metas previstas.

- Coordinar los diferentes centros de costo y relacionar las actividades de la organización.

Charles T. Horngren en su libro titulado "Contabilidad de Costos: Un enfoque General" plantea: *"Un presupuesto es la expresión cuantificada de un plan de acción propuesto por la gerencia para un período específico, y una ayuda para coordinar todo aquello que se necesita para implantar dicho plan y sirve como un proyecto a seguir por la compañía en un proyecto futuro"* (10).

Para el presente trabajo de diploma se toma como definición de presupuesto la registrada en el Decreto Ley No 192 de la Administración Financiera del Estado Cubano, la cual no solo está enfocada a las entidades, sino que maneja el presupuesto como parte del mantenimiento y desarrollo de la sociedad cubana, *"el Presupuesto es el cómputo anticipado de gastos e ingresos. Generalizando esta definición al ámbito social, se puede decir que el presupuesto es el cálculo de los recursos financieros necesarios para el mantenimiento y desarrollo de una colectividad, territorio o nación, a partir de los ingresos que sean capaces de generar y obtener, para cumplir determinados objetivos en un período de tiempo (generalmente un año)"* (11).

Configuración del presupuesto

Es el proceso que inicia cuando se le notifica a la entidad mediante un documento oficial el presupuesto asignado por tipo de moneda para el período fiscal, al llegar el presupuesto a la entidad el personal de planificación lo notifica por conceptos y desagrega por las distintas áreas de la institución teniendo en cuenta la planificación y ejecución del mismo en el año anterior (12).

1.3 Escenario actual del proceso

Según la resolución 241 del 2012 emitida por el MPF mediante el documento Metodología general para la elaboración, notificación, desagregación y programación del presupuesto del Estado, en su sección V se explica el proceso de notificación del presupuesto a sus diferentes niveles. A través del procedimiento de notificación se da a conocer a los órganos y organismos del Estado, las organizaciones y asociaciones, las organizaciones superiores de dirección empresarial y las otras entidades vinculadas al Presupuesto del Estado, así como a las provinciales, municipales, unidades presupuestadas y empresas, el presupuesto aprobado para el nuevo ejercicio fiscal, sus límites de gastos y los gastos de destino específicos (12).

Una vez que el Estado notifica el presupuesto a las entidades, los titulares del presupuesto en las distintas entidades están listos para proceder en los plazos y modelos establecidos a desagregar el presupuesto aprobado. Mediante el procedimiento de desagregación, todos los titulares de presupuestos notificados tienen la oportunidad de desagregar por divisiones, clases, partidas y elementos de gastos, el total de los gastos corrientes aprobados para la actividad presupuestada como límite de gastos. A partir del estudio realizado de la resolución 241 de 2012 del MFP, se concluye que una vez notificado el presupuesto se

procederá a su desagregación ya sea por programación mensual, área de responsabilidad y/o por períodos y áreas de responsabilidad (12).

1.4 Sistemas informáticos empresariales

Para el estudio de la gestión de presupuestos se definieron los sistemas ERP que aparecen señalados como líderes entre los sistemas de tipo open source en el portal de la empresa peruana Advance Global Consulting (13). Además de estos se incluye el estudio de los sistemas contables cubanos Versat Sarasola y Rodas XXI.

Según (13) **los tipos de ERP son:**

- Propietarios: requiere el pago previo por contar con la licencia de uso.
- Open Source: solo debes descargarlos de Internet y asumir los convenios de la comunidad de desarrolladores de ERP.
- Open Source-Propietarios: son de código abierto pero se debe pagar por la capacitación y soporte del sistema.

1.4.1 OpenERP

Es un completo sistema de gestión que engloba todos los procesos de negocio de cualquier empresa con independencia de su tamaño. Cuenta con varios módulos entre los cuales está compuesto por: facturación, cobros y pagos, contabilidad, productos, recursos humanos, control de inventario, gestión de atención a clientes y proveedores, gestión de compras, gestión de almacenes, gestión de proyectos, gestión de producción-fabricación, gestión de ventas, gestión documental y gestión de informes (14).

Para confeccionar el presupuesto anual, OpenERP usa dentro del módulo contabilidad la funcionalidad C2C_Budget (Camp to Camp). Esta crea una estructura presupuestaria a partir de la desagregación de los conceptos presupuestarios que manipula la empresa, los cuales son asociados a una cuenta financiera. Luego se define el presupuesto anual por conceptos, definiendo una fecha de inicio y fin, con vista a las revisiones presupuestarias (15).

OpenERP define líneas presupuestarias, quienes no son más que la dotación en términos económicos de las partidas de ingresos y gastos asociados a un concepto presupuestario, dentro de una versión de presupuesto, y para un período concreto (16).

1.4.2 Openbravo

Está basado en una arquitectura revolucionaria que repercute en un modo mejor de desarrollar aplicaciones de software. Los módulos que gestiona lo hacen especialmente funcional y adaptable a las necesidades de las empresas.

El módulo de contabilidad desarrolla la gestión presupuestaria, manipulando conceptos como:

- Esquema contable, contempla las cuentas que se utilizarán para la contabilización de las transacciones.
- Calendario anual y períodos, permite configurar los ejercicios y los períodos.
- Concepto contable.

En la interfaz dedicada a la gestión presupuestaria, se especifica el tipo de presupuesto y el ejercicio para el que aplica. Además de una serie de líneas que corresponden con las partidas presupuestarias(13).

1.4.3 Adempiere

Es un sistema construido sobre una estructura de flujo de trabajo (flujo de procesos a seguir para la consecución de una tarea o trabajo predeterminado) que integra un ERP y un sistema de administración de relaciones con el cliente (CRM[2]). Se basa por completo en la interrelación de las variables (diferentes áreas) y en su plena interdependencia (no es modular). El Adempiere integra o unifica las diferentes áreas de una empresa (17).

Permite ver todos los movimientos en el reporte de asientos de diario de contabilidad así como la construcción de informes financieros de un presupuesto enmarcado en un período y un ejercicio (17).

De esta manera los presupuestos se ingresan como asientos de diario, así se puede fijar todo un conjunto de resultado como presupuesto y comparar las transacciones reales contra los presupuestos usando la funcionalidad de reportes financieros (17).

1.4.4 Versat Sarasola

Es el primer sistema de contabilidad cubano certificado. Conformado por doce módulos que permiten llevar el control y registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades, siendo configurable por cada una de ellas en el momento de su instalación. Uno de los módulos con que cuenta es planificación económico-productiva (18).

[2] Customer relationship management

Dentro de los objetivos perseguidos por el MFP con el subsistema de planificación económica-productiva resalta el de integrar bajo una misma concepción contable-financiera el presupuesto y su ejecución. Mediante este también se logra una planificación adecuada a cualquier actividad económica, bajo los principios de que todos los resultados se logran por los niveles más importantes de la entidad: centros de costo, áreas de responsabilidad, productos, unidades contables y consolidados empresariales (19).

1.4.5 RODAS XXI

Es un sistema multiempresa que permite el intercambio de los comprobantes generados por cada módulo con el de contabilidad, trabaja con doble moneda, crea reportes fácilmente, permitiendo su visualización, y de forma opcional imprimirlos. Además cuenta con varios módulos que pueden emplearse en su totalidad o en cualquier combinación, estos son los de contabilidad, activos fijos, nóminas, inventario, facturación y finanzas.

El módulo de contabilidad permite tener un control detallado del presupuesto y su ejecución, generar informes económicos financieros y volver sobre períodos contables ya cerrados (20).

Análisis crítico

A partir del estudio de los sistemas seleccionados se identifica la necesidad de desarrollar un componente que permita la gestión de la configuración de los presupuestos de las entidades cubanas, debido a que los sistemas analizados no contemplan un componente dedicado a la gestión presupuestaria. Por lo que la investigación se dirigió a la obtención de funcionalidades, normas o métodos que fuesen recurrentes en las distintas formas de realizar el presupuesto en los sistemas trabajados, por lo que se identificó que el componente a desarrollar debe contemplar: un conjunto de conceptos presupuestarios por los que se desagregará el mismo, las versiones del presupuesto que no es más que un registro histórico de las modificaciones sufridas, el conjunto de áreas por las que se divide la entidad, notificación del presupuesto, desagregaciones por tipo de presupuesto, áreas y conceptos, permitir visualizaciones de las notificaciones y desagregaciones realizadas a un presupuesto así como generar reportes y modelos financieros que apoyen a la toma de decisiones.

1.5 Modelo de desarrollo de software

Para guiar el proceso de desarrollo de software, el CEIGE definió su Modelo de Desarrollo de Software versión 1.2. En el cuál se expone que los proyectos del centro transitan por dos fases: Inicio o Estudio preliminar y Desarrollo. La fase de desarrollo está constituida por disciplinas como: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación (21)

Este modelo abarca el total de acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o producto final, sin embargo, se debe adaptar a las características particulares del proyecto que puede no ejecute determinada disciplina, así como la elaboración de determinados artefactos del total definido (21).

Inicio o Estudio preliminar: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización en la cual se implantará el producto, que permita obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto (21).

Al inicio del proyecto XEDRO ERP se llevó a cabo esta fase, por lo que el presente trabajo solo hace énfasis en la fase de desarrollo.

Desarrollo: En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se define el diseño, se implementa y libera el producto. El objetivo de esta fase es obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales (21).

1.6 Herramientas y tecnologías

En el CEIGE se realizó un estudio para determinar las tecnologías más propicias para el desarrollo de la solución. Como conclusión se integraron un grupo de estas en lo que se denomina marco de trabajo Sauxe y sobre el cual se desarrolla el Sistema Integral de Gestión XEDRO ERP. Está compuesto por un conjunto de componentes reutilizables que provee la estructura genérica, con soporte para entornos multientidad, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo, alejando a los programadores de los detalles arquitectónicos. A continuación se presentan las tecnologías empleadas (22).

1.6.1 Herramientas para el modelado

Visual Paradigm 8.0

Es una herramienta de diseño de Lenguaje Unificado de Modelado (UML[3]) libre y profesional, empleado en la construcción de todos los artefactos que modelan la solución final.

UML 2.0

Es el lenguaje de modelado de software más conocido y utilizado en la actualidad. Permite visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un plano del

[3] Unified Modeling Language

sistema (modelo), incluyendo aspectos conceptuales, tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación y esquemas de bases de datos.

Notación para el Modelado de Procesos de Negocio (BPMN[4])

Es una notación gráfica estandarizada que permite el modelado de procesos de negocio, la cual fue utilizada para llevar a cabo la modelación del proceso de negocio del componente configuración de los presupuestos económicos. Esta notación fue empleada en la confección del mapa de proceso de negocio.

1.6.2 Entorno de desarrollo integrado

Netbeans 7.3

Es un entorno de desarrollo integrado (IDE[5]) de código abierto donde se llevó a cabo la implementación del sistema.

1.6.3 Gestor de base de datos

PostgreSQL 9.1

Gestor de base de datos relacional libre. Se destaca por sus grandes prestaciones y cuenta con una gran comunidad de desarrollo en Internet, su código fuente es libre y es multiplataforma.

1.6.4 Lenguaje de programación del lado del cliente

JavaScript 1.4

Es un lenguaje de programación que le permite a los desarrolladores crear acciones en sus páginas web, empleado en la capa de presentación de la solución, principalmente en el manejo de eventos realizados con el usuario.

HTML 4

Es un lenguaje de marcas de hipertexto(HTML[6]) que utiliza etiquetas para describir la forma en la que deberían aparecer el texto y los gráficos en un navegador web que, a su vez, están preparados para leer esas etiquetas y mostrar la información en un formato estándar. Utilizado en la confección de los reportes y vistas previas de la aplicación.

Firefox 28.0

[4] Business Process Modeling Notation
[5] Integrated Development Environment
[6] HyperText Markup Language

Navegador multiplataforma y de código fuente libre utilizado durante el desarrollo y para el trabajo con la solución final.

1.6.5 Lenguaje de programación del lado del servidor

PHP 5.3

Lenguaje interpretado (PHP[7]) de propósito general ampliamente usado en la programación de toda la lógica de negocio. Puede ser desplegado en la mayoría de los servidores web y en casi todas las plataformas sin costo alguno.

1.6.6 Subversion 1.6.17

Software utilizado para la gestión del repositorio y llevar a cabo el control de versiones de todos los artefactos generados durante el desarrollo.

1.6.7 Apache 2.2

Es un servidor web de software libre desarrollado por la Apache Software Foundation cuyo objetivo es servir o suministrar páginas web (en general, hipertextos) a los clientes web o navegadores que las soliciten.

1.6.8 Zend Framework 1.4

Se trata de un marco de trabajo basado en lenguaje de programación PHP utilizado para el desarrollo de la aplicación.

1.6.9 ORM Doctrine 2

Potente y completo sistema de mapeo relacional de objetos (ORM[8]) empleado para el acceso a datos.

1.6.10 ExtJs 2.2

Es una potente biblioteca o conjunto de librerías de JavaScript utilizado en el desarrollo de las interfaces de usuario.

1.7 Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño y para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias.

[7] PHP Hypertext Pre-processor
[8] Object-Relational Mapping

1.7.1 Patrón de arquitectura

Un patrón arquitectónico de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, presentando un esquema genérico y probado de su solución. A continuación se presenta el patrón arquitectónico utilizado para la realización del sistema informático.

El patrón arquitectónico modelo vista controlador (MVC) presente en el marco de trabajo seleccionado, está compuesto por tres capas:

- **Modelo:** cuenta con los datos y las reglas del negocio.
- **Vista:** muestra la información del modelo al usuario.
- **Controlador:** gestiona las entradas del usuario.

El modelo permite:

- Acceder a la capa de almacenamiento de datos.
- Define las reglas de negocio (la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.

La vista es responsable de:

- Recibir datos del modelo y mostrarlos al usuario.
- Tiene un registro de su controlador asociado.

El controlador permite:

- Recibir los eventos de entrada (un clic, un cambio en un campo de texto, entre otros).
- Contiene reglas de gestión de eventos.

Para el desarrollo del componente se decidió trabajar con este patrón, evidenciándose en los framework definidos para cada una de las capas como parte del MVC de cada componente dentro de la aplicación, para la Vista se utiliza el marco de trabajo Extjs y para el Controlador Zend quien emplea específicamente el estilo Modelo-Vista-Controlador como base de su funcionamiento.
Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual.

1.7.2 Patrones generales de software para asignación de responsabilidades (GRASP[9])

[9] General Responsibility Assignment Software Patterns

Estos describen un conjunto de principios para la asignación de responsabilidades. Entre los más significativos se encuentran los siguientes patrones: experto, creador, bajo acoplamiento, alta cohesión y controlador.

1.7.3 Patrones grupo de los cuatro (GoF[10])

Según su propósito se clasifican en creacionales, estructurales y de comportamiento. El marco de trabajo de Sauxe contiene algunos de estos patrones implícitamente, dándole solución a una serie de problemas recurrentes que se presentan, el mismo implementa numerosas funcionalidades para hacer un mejor uso del sistema.

1.8 Métricas

La IEEE define como métrica "*una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado*" (24).

Las métricas se aplican a todos los aspectos de calidad del software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcionalidad, documentación, métodos, proceso, usuario, entre otros. El empleo de las métricas es una buena manera de conocer lo que sucede en el desarrollo y mantenimiento del sistema, lo que hace posible controlar, predecir y probar el avance del software.

1.8.1 Métrica para los requisitos

Es necesario lograr una estabilidad en los requerimientos para el correcto funcionamiento de los demás flujos de trabajo. El objetivo de esta métrica es medir la estabilidad de los requerimientos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. Se considera que los requerimientos son estables cuando no existen adiciones o supresiones en ellos que impliquen modificaciones en las funcionalidades principales de la aplicación.

1.8.2 Métricas para el diseño

Para medir el diseño se utilizaron métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto, teniendo en cuenta que este estudio brinda un esquema sencillo de implementación y que a la vez cubre los principales atributos de calidad de software. Las métricas escogidas para la validación del diseño fueron las métricas orientada a clases: tamaño operacional de clase (TOC) y relaciones entre clases (RC).

[10] Gang of Four

1.9 Pruebas de Calidad

Las pruebas del software son un elemento crítico para garantizar de calidad del mismo y representan una revisión final de las especificaciones, del diseño y de la codificación. Pero las pruebas no pueden asegurar la ausencia de errores, solo pueden demostrar que existen defectos en el software (25).

1.9.1 Técnica de pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal o prueba de estructura, se encarga de asegurar que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada. Además, se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles, para determinar si el estado real coincide con el esperado o afirmado (25).

1.9.2 Técnica de pruebas de caja negra

La prueba de caja negra, también denominada prueba de comportamiento se refiere a las pruebas que se llevan a cabo sobre la interfaz del software, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene (25).

1.10 Conclusiones parciales

Dado que la solución ha de estar enfocada a las particularidades de las empresas cubanas, la aplicación debe realizar el proceso de notificación y desagregación del presupuesto por conceptos presupuestarios y áreas responsables.

Los módulos y funcionalidades existentes para los sistemas analizados, no permiten realizar una correcta configuración del presupuesto según las exigencias del MFP, pero si aportan un conjunto de técnicas recurrentes en sus soluciones que deben tenerse en cuenta, tales como: contemplar un conjunto de conceptos asociados a partidas o centros de costos presupuestarios, especificar el tipo de presupuesto, versión, período y ejercicio para el que aplica.

Capítulo 2 Modelado del negocio, análisis y diseño

2.1 Introducción

En este capítulo se lleva a cabo, en un primer momento la modelación del negocio, teniendo en cuenta que antes de comenzar a desarrollar un software se hace necesario comprender el negocio mediante el estudio de los procesos que en él se desarrollan, en ese sentido se realiza una descripción del mismo y se definen los procesos implicados. Posteriormente se identificaron los requisitos funcionales con el cliente mediante técnicas de tormentas de ideas y la entrevista; aplicando métricas y técnicas de validación para medir la calidad. Se reflejan los requisitos no funcionales definidos para el sistema. Se muestra cómo está concebida la arquitectura y el diseño del software mediante los artefactos generados, haciendo uso de patrones identificados por el equipo de arquitectura del sistema XEDRO ERP.

2.2 Modelado del negocio

La modelación del negocio como fase dentro del proceso de desarrollo del software tiene a cargo la comprensión de la estructura y dinámica de la organización en la cual se va a implantar el sistema. Para lograr este propósito el proceso de modelado permite obtener una visión del negocio mediante la definición, descripción y representación de los procesos. A continuación se realiza la modelación del negocio asociado al componente de configuración de los presupuestos.

2.2.1 Descripción del negocio

Para la configuración de los presupuestos en una entidad, el MFP debe aprobar y luego emitir un documento oficial con las cantidades totales de dinero por monedas, este documento es enviado a la entidad, en la cual los especialistas de la dirección de planificación y estadística realizan el proceso de notificación del presupuesto teniendo en cuenta los diferentes conceptos. Luego de la notificación del presupuesto se procederá a la desagregación del mismo por las distintas áreas responsables. Una vez notificado y desagregado el presupuesto los responsables de las distintas áreas deben reunirse y aprobar la asignación realizada por áreas, en caso de no aprobarse alguna propuesta el proceso debe repetirse nuevamente. Si la entidad es centralizada la dirección de planificación y estadística de esta será la encargada de realizar la desagregación hasta el nivel de centro de costos, en caso contrario las áreas tendrán la responsabilidad de realizarlo.

2.2.2 Diagrama de proceso de negocio

Un modelo de procesos de negocio es un conjunto de objetos gráficos, correspondientes a las actividades y controles de flujo que definen el orden de ejecución de estas. Seguidamente se muestra el diagrama de proceso de negocio correspondiente al componente de configuración de los presupuestos:

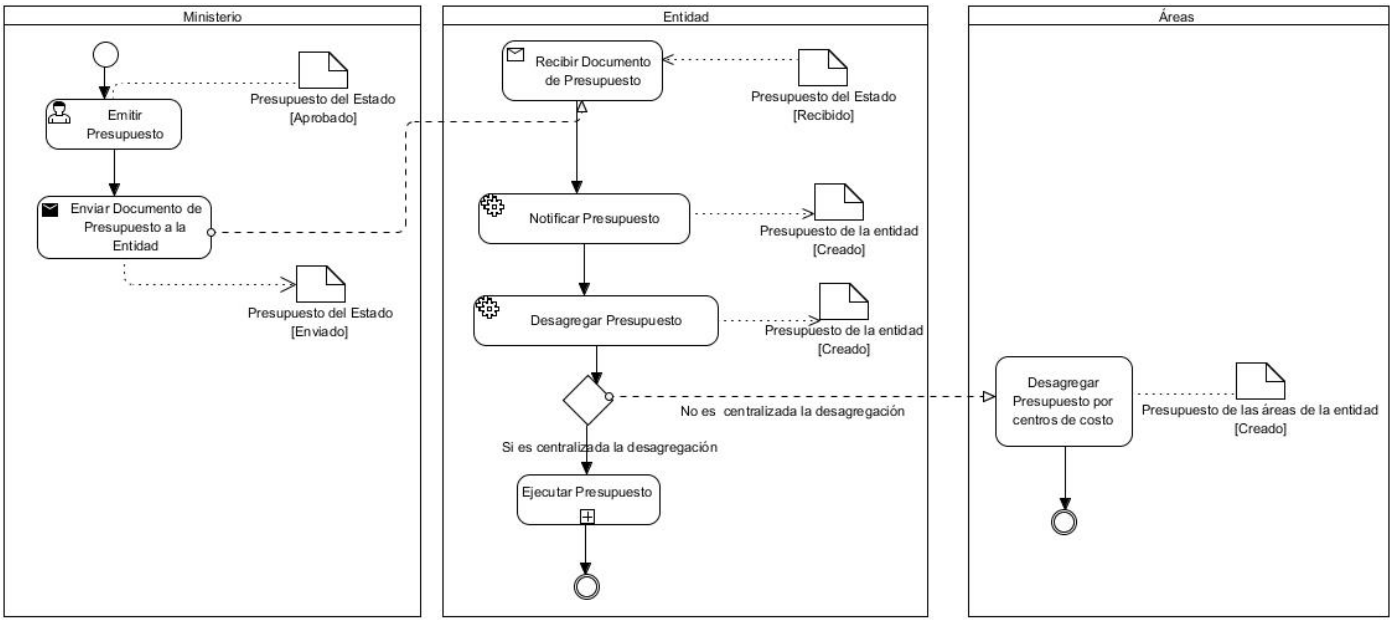


Figura 1: Diagrama de proceso de negocio del componente configuración de los presupuestos

2.2.3 Modelo conceptual

El modelo conceptual, permite dominar los principales conceptos con los que se relaciona el componente a desarrollar. Todos estos conceptos son modelados mediante tablas representando las entidades que lo conforman y sus atributos, estableciendo una representación de los mismos, asociados a la configuración del presupuesto.

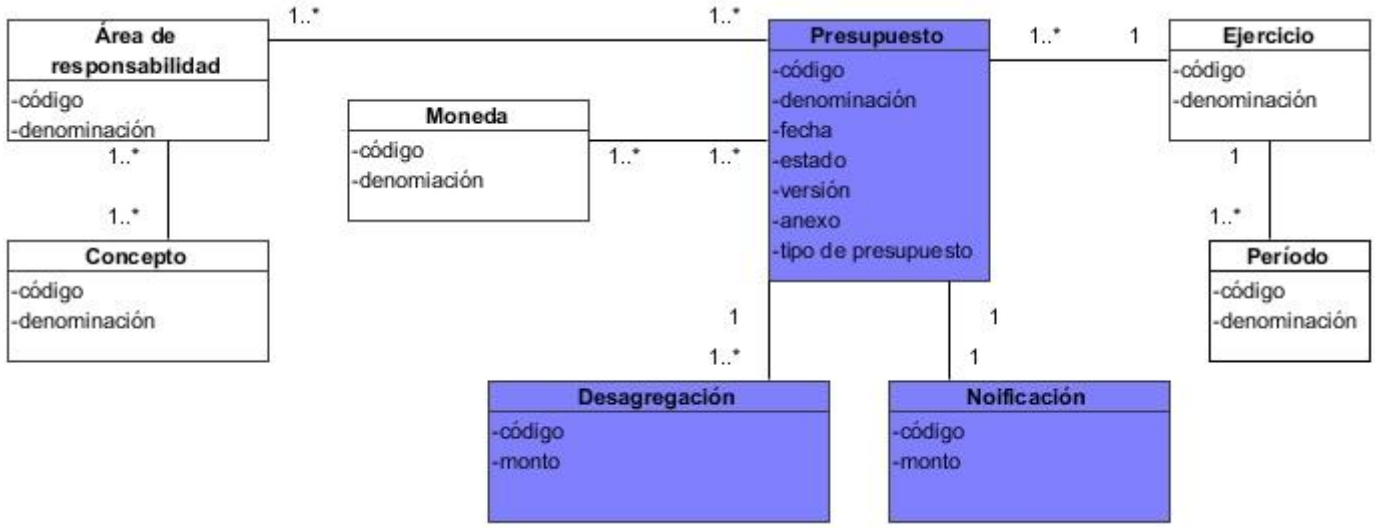


Figura 2: Diagrama de modelo conceptual del componente configuración de los presupuestos

2.3 Técnicas para la captura de requisitos funcionales

Una adecuada comprensión de los requisitos favorece al desarrollo de nuevos sistemas informáticos que cumplan con las necesidades y expectativas del cliente. Para realizar este procedimiento existen diversas técnicas que guían al analista en el proceso de comunicación con el cliente y el equipo de desarrollo.

Entrevista: es de gran utilidad para obtener información cualitativa como opiniones o descripciones de actividades, requiere seleccionar bien a los entrevistados para obtener la mayor cantidad de información en el menor tiempo posible. Es muy aceptada y permite acercarse al problema de una manera natural (26).

Tormentas de ideas: para la aplicación de esta técnica se realizaron varias reuniones entre los clientes y el equipo de desarrollo donde ambas partes brindaban sus ideas en cuanto a la propuesta de solución. Esta técnica se puede utilizar para identificar un primer conjunto de requisitos en aquellos casos donde no están muy claras las necesidades que hay que cubrir. Esta técnica arrojó un total de 26 requisitos funcionales (26).

2.4 Requisitos de software

En un proceso de desarrollo de software es fundamental la descripción de los requisitos funcionales, estos facilitan un mejor entendimiento de los procesos a desarrollar, permitiendo comprender con profundidad el problema en cuestión y facilitando una mejor identificación de las clases y funcionalidades que serán implementadas. La especificación de requisitos, es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto, debido a que son un reflejo detallado de las necesidades del cliente.

2.4.1 Requisitos funcionales del sistema

Los requisitos funcionales son lo qué debe hacer y de qué manera debe reaccionar el sistema ante las entradas y cómo debe comportarse en situaciones específicas. Estos detallan la función del producto de software, entrada, salidas, excepciones y usuarios.

Como resultado del uso de estas técnicas de obtención de requisitos se identificaron 26 requisitos funcionales y 12 requisitos no funcionales los cuales se encuentran en el epígrafe 2.2.5. A continuación se presentan los requisitos funcionales del sistema.

Requisitos funcionales
RF1: Adicionar presupuesto
RF2: Modificar presupuesto

RF3: Eliminar presupuesto
RF4: Buscar presupuesto
RF5: Búsqueda avanzada del presupuesto
RF6: Notificar monto total por moneda
RF7: Eliminar notificación del monto total por moneda
RF8: Notificar presupuesto por áreas responsables
RF9: Eliminar notificación del monto total por áreas responsables
RF10: Notificar presupuesto por conceptos
RF11: Eliminar notificación del monto total por conceptos
RF12: Desagregar presupuesto por áreas responsables
RF13: Eliminar desagregación del presupuesto por áreas responsables
RF14: Desagregar presupuesto por conceptos y períodos
RF15: Eliminar desagregación del presupuesto por conceptos y períodos
RF16: Desagregar presupuesto por áreas responsables, conceptos y período
RF17: Eliminar desagregación del presupuesto por áreas responsables, conceptos y período
RF18: Generar vista previa en notificar monto total por moneda
RF19: Generar vista previa en notificar monto total por áreas responsables
RF20: Generar vista previa en notificar monto por conceptos
RF21: Generar vista previa en desagregar presupuesto por áreas responsables
RF22: Generar vista previa en programación mensual
RF23: Generar vista previa en desagregar presupuesto por áreas responsables, conceptos y período
RF24: Generar modelo trimestral de gastos-modificaciones presupuestarias
RF25: Generar modelo desglose por partidas
RF26: Generar reportes por tipo de presupuesto y moneda

Tabla 1: Requisitos funcionales

Seguidamente se muestra la descripción de uno de los requisitos funcionales. Las restantes descripciones pueden ser consultadas en los documentos CEIGE-CNP Notificación del presupuesto por conceptos, CEIGE-CNP Notificación del presupuesto por áreas responsables, CEIGE-CNP Notificación del monto por moneda, CEIGE-CNP Desagregación del presupuesto por período, CEIGE-CNP Desagregación del

presupuesto por áreas responsables, CEIGE-CNP Desagregación del presupuesto por áreas responsables, conceptos y período y CEIGE-CNP Presupuesto.

2.4.2 Especificación de requisitos funcionales del sistema

Requisito funcional adicionar presupuesto

Precondiciones	El usuario se ha autenticado en el sistema. El usuario tiene permisos para adicionar un presupuesto.
Flujo de eventos	
Flujo básico adicionar presupuesto	
1	Se selecciona la opción de Configuración.
2	Se selecciona el botón “Adicionar”.
3	El sistema muestra una ventana para adicionar el presupuesto.
4	Se introduce la *denominación* y se seleccionan el *estado, archivo, fecha y tipo de presupuesto*.
5	Se selecciona el botón “Aceptar” y el sistema valida los datos introducidos en adicionar presupuesto. En caso de datos incorrectos ver *Flujo alterno “5a. Datos incorrectos en adicionar presupuesto”.* En caso de datos cancelados ver *Flujo alterno “5b. Datos cancelados en adicionar presupuesto”.* En caso de datos vacíos ver *Flujo alterno “5c. Datos vacíos en adicionar presupuesto”.*
6	Si los datos son correctos el sistema los registra.
7	El sistema confirma el registro de los datos.
8	Concluye el requisito.
Pos-condiciones	
1	Registrados los datos en el sistema.
2	
Flujos alternativos	
Flujo alternativo 5a. Datos incorrectos en adicionar presupuesto	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos en los campos señalados.
3	Se ejecuta el paso 6 del Flujo básico adicionar presupuesto.
Flujo alternativo 5b. Datos cancelados en adicionar presupuesto	
1	Concluye el requisito.
Flujo alternativo 5c. Datos vacíos en adicionar presupuesto	
1	El sistema señala los campos vacíos y muestra un mensaje informando: “Este campo es obligatorio”.
2	El usuario inserta los datos en los campos señalados.
3	Se ejecuta el paso 6 del Flujo básico adicionar presupuesto.
Validaciones	

	Se validan los datos según lo establecido en el modelo conceptual.
Conceptos	No aplica
Requisitos especiales	No aplica
Asuntos pendientes	No aplica

Prototipo de Interfaz.

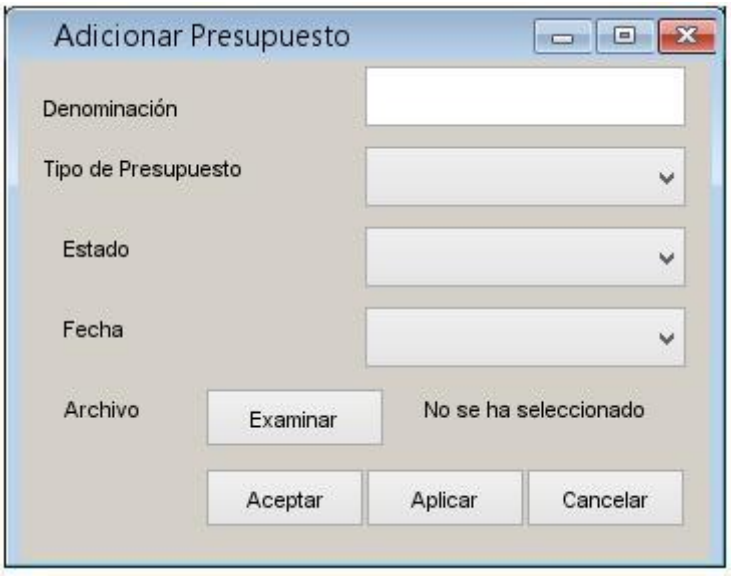


Figura 3: Prototipo de interfaz del requisito adicionar presupuesto

2.4.3 Requisitos no funcionales del sistema

Debido a que la solución propuesta forma parte del proyecto XEDRO ERP y su equipo de analistas en la fase inicial definió los requisitos no funcionales del sistema, el componente a desarrollar debe cumplir con los siguientes requisitos no funcionales:

Requisitos no funcionales
Usabilidad
RNF 1. Todos los mensajes de error del componente deben de tener una descripción textual del error y debe permitir además intentar de nuevo.
RNF 2. Las etiquetas de cada funcionalidad y los campos de cada interfaz tendrán títulos asociados a su función de negocio.
RNF 3. El idioma de todas las interfaces de la aplicación será el español.
RNF 4. Los flujos de navegación para la gestión de cualquier concepto del negocio no excederán de las tres interfaces.

Confiabilidad
RNF 5. El componente debe validar automáticamente la información contenida en los formularios de ingreso. En el proceso de validación de la información, se deben tener en cuenta aspectos tales como obligatoriedad de campos, longitud de caracteres permitida por campo, manejo de tipos de datos, el sistema no permitirá la entrada de datos incorrectos.
RNF 6. Ninguna información que se haya ingresado en el sistema y se haya asociado a alguna operación será eliminada físicamente de la base de datos.
Interfaz
RNF 7. Interfaz de usuario. El sistema tiene que ofrecer una interfaz sencilla, permitiendo un balance adecuado entre funcionalidad y simplicidad.
Seguridad
RNF 8. Autenticación. El sistema concederá acceso a cada usuario autenticado solo a las funciones que le estén permitidas, de acuerdo a la configuración del sistema.
RNF 9. Autorización. Los permisos serán limitados según los roles definido en el sistema, el usuario no debe poder modificar sus permisos. Los permisos de los usuarios deben estar en correspondencia con los niveles de acceso a la información.
Software
RNF 10. Servidor Apache 2.2 y PHP 5.3.
RNF 11. Gestor de base de datos PostgreSQL 9.1.
Mantenibilidad
RNF 12. El sistema debe estar en capacidad de permitir en el futuro su fácil mantenimiento con respecto a los posibles errores que se puedan presentar durante la operación del sistema.

Tabla 2: Requisitos no funcionales

2.5 Técnicas para la validación de requisitos

La validación de requisitos tiene como misión demostrar que la definición de estos concreta realmente el sistema que el cliente desea. Esta es una actividad fundamental, pues un levantamiento de requisitos con errores que no se detecten a tiempo y que conduzcan a resultados inesperados, provocan costos excesivos y gran pérdida de tiempo.

Para la validación de los requisitos del presente trabajo se aplicaron dos técnicas:

Construcción de prototipos: estos son la versión inicial de un sistema de software que se utiliza para demostrar los conceptos, probar las opciones de diseño y entender mejor el problema y su solución, permiten revelar errores u omisiones en los requisitos propuestos.

Casos de prueba: estos permiten definir las entradas, las salidas del software y acciones del usuario para que se pueda completar lo que expresa el requisito. Además como los requisitos deben ser posibles de probar, se le debe de asociar un caso de prueba por cada uno.

Aplicación de la métrica estabilidad en los requisitos

El objetivo de la aplicación de esta métrica es medir la estabilidad de los requisitos para asegurar su adecuación antes de pasar al próximo flujo de trabajo. **La estabilidad de los requisitos** se calcula según (26) como:

$$ETR = \left[\frac{(RT - RM)}{RT}\right] * 100$$

Siendo:

ETR: valor de la estabilidad de los requisitos

RT: total de requisitos funcionales

RM: total de requisitos modificados

Para una primera iteración se identificaron un total de 26 requisitos (RT), dentro de los cuales 13 resultaron modificados (RM), arrojando los siguientes datos:

$$ETR = \left[\frac{26 - 13}{26}\right] * 100 = 50\%$$

Para una segunda iteración del total de RF, solo se modificó 1 de ellos, dando como resultado:

$$ETR = \left[\frac{26 - 1}{26}\right] * 100 = 96\%$$

Luego de aplicar esta métrica, se obtuvo en la primera iteración como resultado, que los requisitos no fueron lo suficientemente estables. Luego de perfeccionado todos los señalamientos se realizaron nuevamente las validaciones alcanzándose un resultado favorable.

Figura 4: Métrica estabilidad de requisitos

2.6 Arquitectura del sistema

Para el desarrollo del componente se utiliza el marco de trabajo Sauxe el cual responde a un estilo arquitectónico en capas, basándose en una distribución jerárquica de las responsabilidades para proporcionar una división efectiva de los problemas a resolver, de forma tal que cada capa contiene la funcionalidad relacionada solo con las tareas de esa capa, las capas inferiores no tienen dependencias de las capas superiores y la comunicación entre ellas está basada en una abstracción que les proporciona un bajo acoplamiento. Según (27) esta arquitectura está compuesta básicamente por cinco capas:

- Capa de Presentación: en esta capa se emplea las facilidades que brinda el marco de trabajo ExtJs para la construcción de interfaces amigables a la vista de los usuarios.
- Capa de Control o Negocio: en esta capa se emplea el patrón de arquitectura MVC.
- Capa de Acceso a Datos: en esta capa estará presente el ORM Doctrine, como marco de trabajo de persistencia para la comunicación con el servidor de datos mediante la extensión Objetos de Datos de PHP (PDO[11]).
- Capa de Datos: en esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de ficheros de configuración de la arquitectura tecnológica.
- Capa de Servicio: en esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí.

[11] PHP Data Object

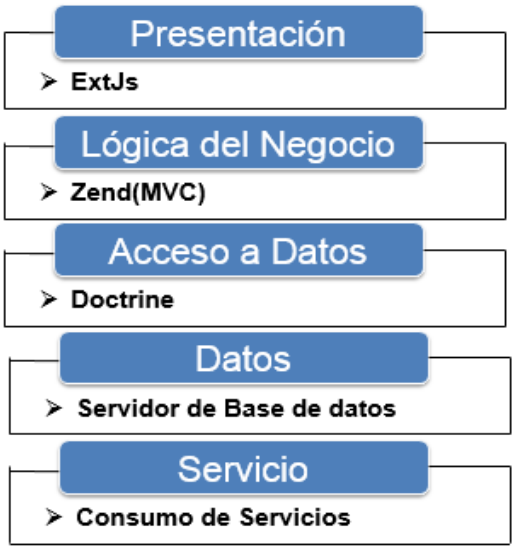


Figura 5: Vista de las capas de la arquitectura

2.7 Diseño

En la fase de diseño el objetivo es utilizar la información recolectada en la etapa de análisis para modelar el sistema de manera que cumpla con los requisitos sirviendo como punto de partida para la implementación. El diseño del sistema es una base fundamental en la elaboración del software, el cual permite que se produzcan los diversos modelos para la solución que se pondrá en desarrollo.

2.8 Patrones de diseño

GoF

El marco de trabajo Sauxe contiene algunos de estos patrones implícitamente, dándole solución a una serie de problemas recurrentes que se presentan, el mismo implementa numerosas funcionalidades para hacer un mejor uso de sistema.

- **Estructurales:**

Decorador: el patrón decorador responde a la necesidad de añadir dinámicamente funcionalidad a un objeto. En ocasiones se desea adicionar responsabilidades a un objeto, pero no a toda la clase, estas se pueden adicionar por medio de los mecanismos de herencia, pero este mecanismo no es flexible porque la responsabilidad es adicionada estáticamente. La solución flexible es la de rodear el objeto con otro objeto que es el que adiciona la nueva responsabilidad. ZF implementa el patrón decorador en su clase Zend_View, la cual es la encargada de asignarle responsabilidades a objetos de manera dinámica y configurarlos con nuevos atributos.

Fachada: este patrón proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar (28).
En el proyecto XEDRO ERP los servicios están basados en el patrón fachada, donde la relación que existe entre las clases controladoras y estos permiten acceder a métodos que se encuentran en otros componentes. Entre las ventajas que brinda el patrón fachada se encuentra que el usuario que lo va a consumir no necesita tener conocimientos del negocio interno del componente.

GRASP

Entre los patrones utilizados para el diseño del sistema se encuentran los siguientes:

- **Experto:** el patrón fue usado con el objetivo de darle a las clases la responsabilidad necesaria siempre que contaran con la información para cumplirla (29). Logrando así un mejor comportamiento entre las clases y haciendo que estas fuesen fáciles de comprender y mantener.

  Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Este patrón es ampliamente usado en las clases del dominio, por ejemplo, en la clase DatPresupuesto, es la responsable de manejar la información y efectuar las operaciones que conciernen a su función (insertar y modificar el presupuesto), asumiendo toda la lógica para cada una de ellas.

- **Creador:** el patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Su propósito es encontrar un creador que se conecte con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (29). En la solución propuesta este patrón fue ampliamente usado para hacer instancias de las clases del dominio desde la clase controlador. A continuación se muestra un fragmento de código que lo evidencia.

```
class PresupuestoController extends ZendExt_Controller_Secure {
    private $idestructura;
    private $nomenclador;

    function init (){
        $this->model [] = new DatPresupuestoModel();//[0]
        $this->model [] = new DatNotificacionModel();//[1]
        $this->model [] = new DatDesagregarcentroModel();//[2]
        $this->model [] = new DatProgramacionModel();//[3]
        $this->model [] = new DatPresupuestacionModel();//[4]
        $this->model [] = new DatPresupuestacioncentroModel();//[5]
        $this->model [] = new DatPeriodocentroModel();//[6]

        parent::init();
        $this->nomenclador = new ZendExt_Nomencladores_ADT();
        $this->estructura = $this->global->Estructura->idestructura;
    }
```

Figura 6: Fragmento de código donde se evidencia el patrón creador

- **Alta cohesión:** en la perspectiva del diseño orientado a objetos, es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme (29). Cada clase del modelo implementa sus propios métodos y ayuda a resolver alguna tarea en otra clase que dependa de esta.
- **Bajo acoplamiento:** el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras clases (29). Este patrón tiene como idea, tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre ellas.
- **Controlador:** es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación (29). A continuación se muestra un fragmento de código que lo evidencia.

```
function presupuestosAction() {
    $aux = $this->model[0]->obtenerPresupuestosModel(
                $this->getRequest()->getParam('start'),
                $this->getRequest()->getParam('limit'),
                $this->getRequest()->getParam('criterio'),
                $this->getRequest()->getParam('idestado'),
                $this->getRequest()->getParam('idtipopresupuesto'),
                $this->getRequest()->getParam('idversion'),
                $this->getRequest()->getParam('idejercicio')
        );
```

Figura 7: Fragmento de código donde se evidencia el patrón controlador

2.9 Diagrama de clases del diseño

Estos representan las clases que serán utilizadas dentro del sistema y las relaciones que existen entre ellas. Siendo útil para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y/o de convencimiento.

El siguiente diagrama recoge el requisito funcional:

- Gestionar Presupuesto

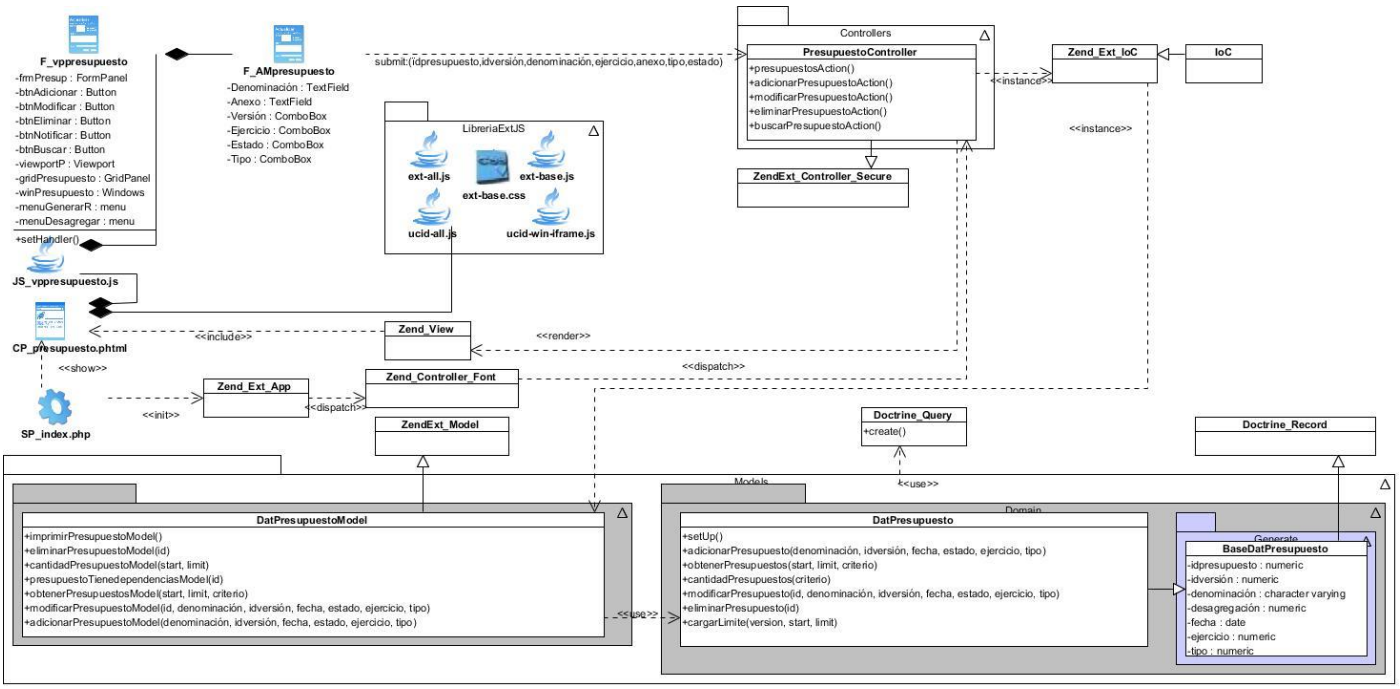


Figura 8: Diagrama de clases

Elementos representados	Descripción
Librería ExtJS	Contiene los componentes generados a través de la librería de JavaScript ExtJS necesarios para visualizar el contenido.
SP_Index.php	Página servidora que construye a las páginas clientes e inicializa las acciones en la aplicación instanciando la clase ZendExt_App para establecer la comunicación con el controlador y la lógica de negocio.
ZendExt_App	Clase del marco de trabajo Zend que maneja y usa los controladores frontales Zend_Controler_Front para acceder a la clase controladora de las acciones de la página cliente que la inicializó.
Zend_Controller_Front	Clase controladora frontal que maneja las peticiones de inicialización de una página cliente al controlador.

PresupuestoController	Clase controladora con las funciones de actualización y manejo de eventos de las páginas clientes y el modelo. La clase controladora utiliza la clase Zend_View para dibujar el contenido de la página cliente CP_presupuesto.phtml.
Zend_View	Incluye las páginas clientes y construye o actualiza la interfaz de las mismas con la información del controlador.
CP_presupuesto.phtml	Página cliente que debe ser mostrada o actualizada como respuesta del sistema.
DatPresupuestoModel	Clase del negocio encargada de manejar los datos persistentes dentro del componente. La clase definida en el diseño hereda de la clase ZendExt_Model, ya que esta incluye las principales funciones para el manejo de los datos.
ZendExt_Model	Modelo gestor del negocio que permite entre otras funcionalidades iniciar las conexiones a la base de datos.
Paquete Domain	Contiene las clases que representan las tablas de la base de datos, incluyen todas las funcionalidades referentes al acceso a los datos mediante el lenguaje (DQL[12]) propio de Doctrine.

Tabla 3: Descripción del diseño de clases asociadas al componente configuración de los presupuestos

En el documento CIG-CFM-N-MTTO-i3801.doc se representa la estructura del modelo de diseño para el diagrama de clases.

2.10 Diagrama de secuencia

Un diagrama de secuencia es una representación de cómo los eventos causan un flujo entre uno y otro como una función del tiempo. Representa clases claves y eventos que permiten que el comportamiento fluya de clase a clase (30).

[12] Data Query Language

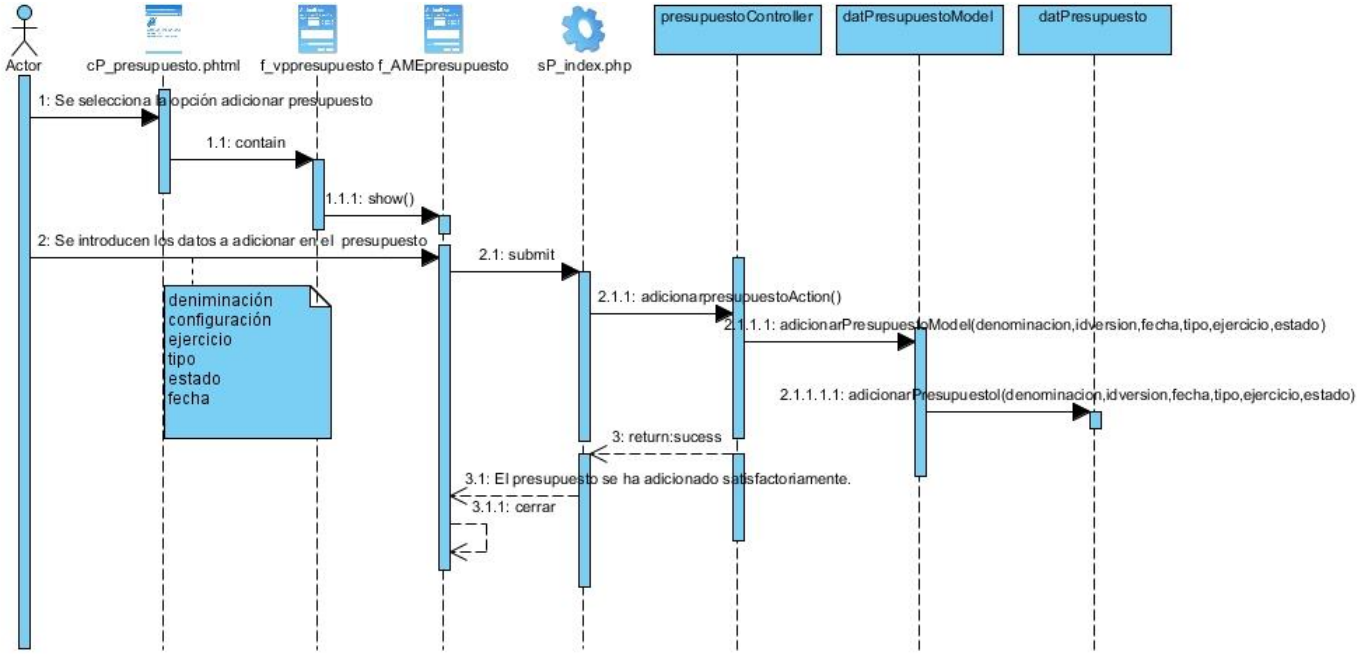


Figura 9: Diagrama de secuencia del requisito adicionar presupuesto

2.11 Modelo de datos

Para la construcción del diseño del modelo de datos se utilizó el patrón llave subrogada creando una llave primaria única para cada entidad en vez de usar un atributo identificador en el contexto dado. Sin utilizar el citado patrón hubiese sido necesario un diseño cuidadoso, ya que un error pudiera conllevar a inconsistencias en la base de datos y consecuentemente, a producir información errónea en consultas.

Para la validación del modelo de datos se estudiaron las métricas: número de atributos, grado de referenciabilidad (RD[13]), profundidad de árbol relacional (DRT[14]) y cociente de normalidad (NR[15]), definidas en (31). Asumiéndose para la validación del modelo propuesto la métrica NR quien se define como el número de tablas en tercera forma normal o superior dividido entre el número de tablas en el esquema.

$$NR = \frac{NT3NF}{NTS} \quad NR = \frac{10}{10} = 1$$

Siendo:

NT3NF: el número de tablas en tercera forma normal o superior.

NTS: el número de tablas en el esquema (32).

[13] Referential Degree
[14] Depth Referential Tree
[15] Normality ratio

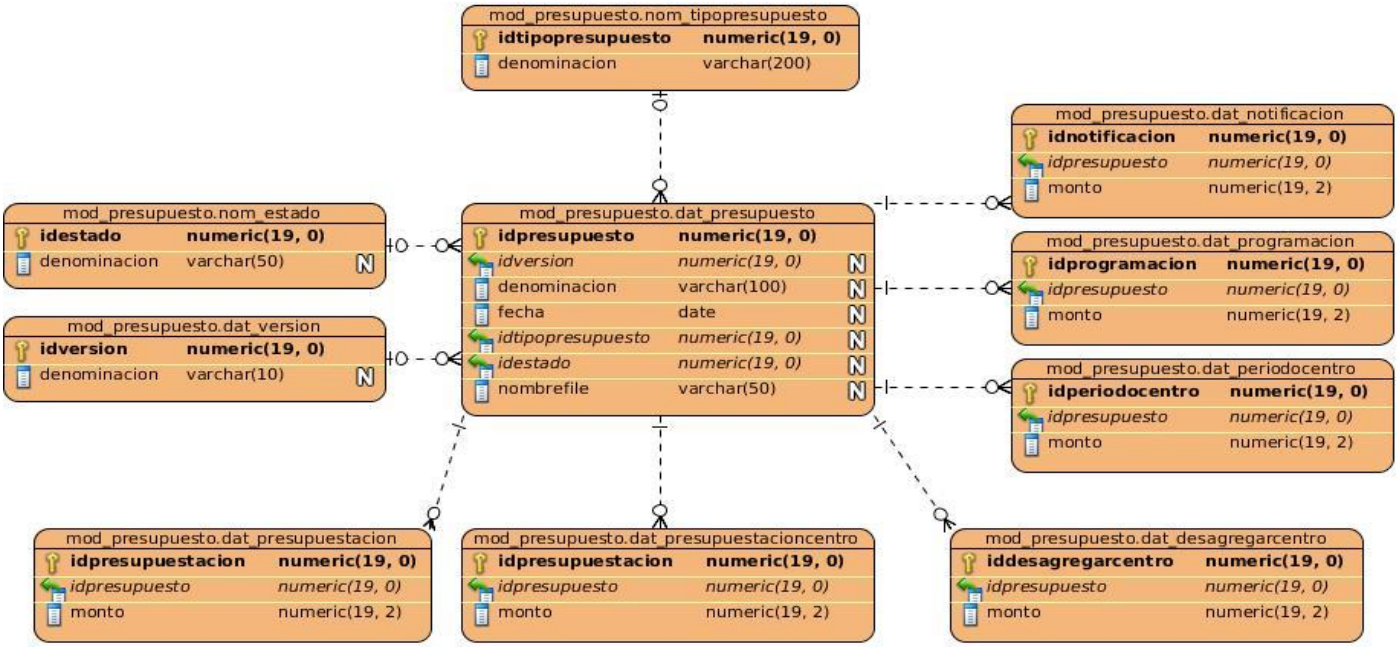


Figura 10: Modelo de datos

2.12 Validación del proceso de diseño

Métrica tamaño operacional de clase (TOC)

Para aplicar la métrica TOC se tendrá en cuenta un conjunto de atributos de calidad como los que se muestran a continuación:

Atributos de calidad	Definición
Responsabilidad	Consiste en asignarle una responsabilidad a una clase en un marco de modelado de dominio
Complejidad de implementación	Hace compleja las pruebas del sistema
Reutilización	Consiste en el grado de reutilización presente en una clase

Tabla 4: Atributos de calidad de la métrica TOC

El tamaño operacional de una clase se puede determinar a partir del número de operaciones que posee, el resultado es tomado como umbral que luego se compara de acuerdo a los valores que se muestran a continuación.

Clasificación	Valores de los umbrales
Pequeño	Umbral<=20

Medio	20<Umbral<=30
Grande	Umbral >30

Tabla 5: Clasificación de las clases según el umbral

Si existen valores grandes de TOC, estos estarán demostrando que una clase puede tener demasiada responsabilidad, lo cual reduce la reutilización de la clase. El próximo paso es calcular el promedio correspondiente a los umbrales, una vez obtenidos se procede a calcular la afectación de los parámetros establecidos.

Atributos de calidad	Clasificación	Criterio
Responsabilidad	Baja Media Alta	Umbral<= Promedio Promedio < Umbral < = 2* promedio Umbral > 2* promedio
Complejidad de implementación	Baja Media Alta	Umbral <= Promedio Promedio < Umbral < = 2* promedio Umbral > 2* promedio
Reutilización	Baja Media Alta	Umbral > 2* promedio Promedio < Umbral < = 2* promedio Umbral <= Promedio

Tabla 6: Rango de valores para medir la afectación de los atributos de calidad de la métrica TOC

A continuación se muestra un ejemplo del resultado de la aplicación de esta métrica al diseño.

Responsabilidad	Cantidad de clases	Promedio
Baja	48	72,72727273
Media	15	22,72727273
Alta	3	4,545454545

Tabla 7: Responsabilidad

Figura 11: Representación de la evaluación de la métrica TOC para el atributo responsabilidad

Complejidad	Cantidad de clases	Promedio
Baja	48	72,72727273
Media	15	22,72727273
Alta	3	4,545454545

Tabla 8: Complejidad

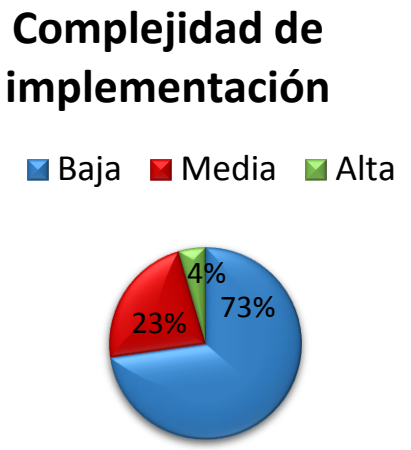


Figura 12: Representación de la evaluación de la métrica TOC para el atributo complejidad

Reutilización	Cantidad de clases	Promedio
Alta	48	72,72727273
Media	15	22,72727273
Baja	3	4,545454545

Tabla 9: Reutilización

Figura 13: Representación de la evaluación de la métrica TOC para el atributo reutilización

Después de realizar la aplicación de la métrica TOC se llega a la conclusión de que el diseño propuesto cumple con lo requerido inicialmente ya que el mismo arrojó como resultado un baja responsabilidad y complejidad y una alta reutilización, permitiendo la eliminación de clases repetidas, teniendo en cuenta que más de la mitad de las clases poseen baja dependencia respecto a otras, evidenciando que el sistema no tendrá una compleja implementación.

Métrica relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otra. A continuación se muestra una serie de tablas encaminadas a un mejor entendimiento de la utilización de esta métrica.

Atributos de calidad	Definición
Acoplamiento	Consiste en el grado de dependencia o interconexión de una clase con otras.
Complejidad de 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.
Reutilización	Consiste en el grado de reutilización presente en una clase, dentro de un diseño de software.
Cantidad de pruebas	Consiste en el número o grado de esfuerzo necesario para realizar las pruebas.

Tabla 10: Atributos de calidad de la métrica RC

Para determinar el grado de afectación de los atributos de calidad que mide la métrica RC es necesario determinar la cantidad de relaciones de uso (CRU) que posee cada una de las clases a medir. Una vez determinada la CRU, se procede a calcular el promedio de las mismas y teniendo ambos valores según los criterios expuestos a continuación.

Atributos de calidad	Clasificación	Criterio
Acoplamiento	Ninguna	CRU = 0
	Baja	CRU = 1
	Media	CRU = 2
	Alta	CRU > 2
	Baja	CRU <= Promedio
Complejidad de mantenimiento	Media	Promedio < CRU < = 2* promedio
	Alta	CRU > 2* promedio
Reutilización	Baja	CRU > 2* promedio
	Media	Promedio < CRU < = 2* promedio
	Alta	CRU <= Promedio
Cantidad de pruebas	Baja	CRC <= Promedio
	Media	Promedio<= CRC <2*Promedio
	Alta	CRC >= 2*Promedio

Tabla 11: Rango de valores para medir la afectación de los atributos de calidad de la métrica RC

A continuación se muestra un ejemplo del resultado de la aplicación de esta métrica al diseño.

Criterio	Categoría	Cantidad de clases	Promedio
0 dependencia	Muy Bueno	21	31,81818182
1 dependencia	Bueno	21	31,81818182
2 dependencias	Regular	16	24,24242424
3 dependencias	Malo	5	7,575757576
Más de 3 dependencias	Muy Malo	3	4,545454545
Total		45	100

Tabla 12: Cantidad de dependencias por clasificación

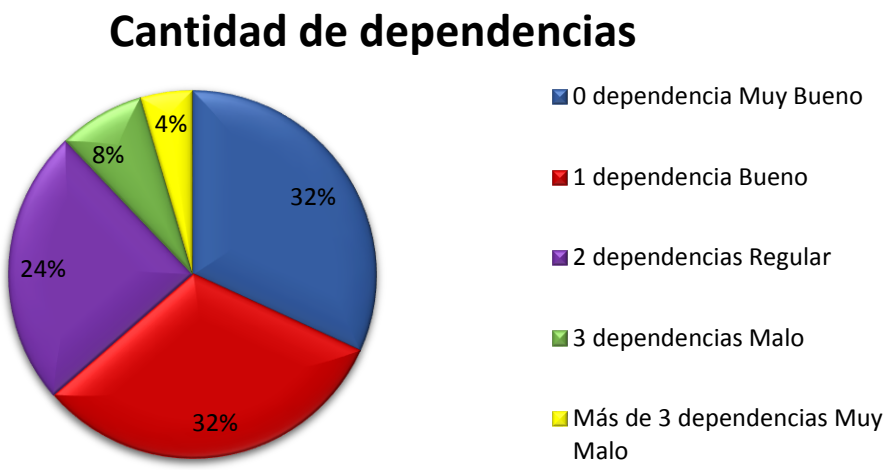


Figura 14: Comportamiento de los valores de dependencia

- **Acoplamiento**

Acoplamiento	Cantidad de clases	Promedio
Ninguno	21	31,81818182
Bajo	21	31,81818182
Medio	16	24,24242424
Alto	8	12,12121212

Tabla 13: Acoplamiento

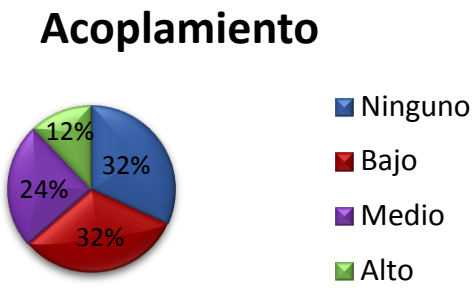


Figura 15: Representación de la evaluación de la métrica RC en el atributo Acoplamiento

- **Complejidad de mantenimiento**

Complejidad de Mantenimiento	Cantidad de clases	Promedio
Baja	42	63,63636364
Media	16	24,24242424
Alta	8	12,12121212

Tabla 14: Complejidad de mantenimiento

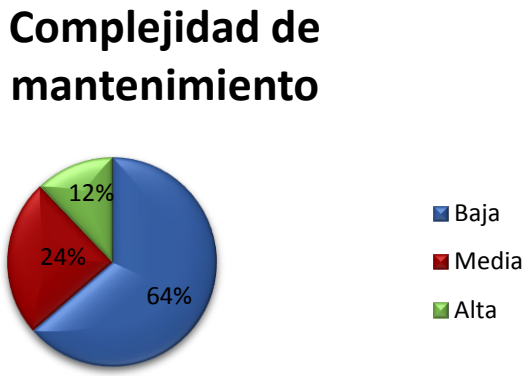


Figura 16: Representación de la evaluación de la métrica RC en el atributo complejidad de mantenimiento

- **Cantidad de pruebas**

Cantidad de Pruebas	Cantidad de clases	Promedio
Baja	42	63,63636364
Media	16	24,24242424
Alta	8	12,12121212

Tabla 15: Cantidad de pruebas

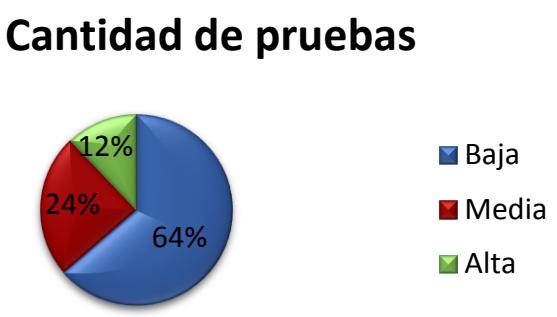


Figura 17: Representación de la evaluación de la métrica RC en el atributo cantidad de pruebas

- **Reutilización**

Reutilización	Cantidad de clases	Promedio
Baja	8	12,12121212
Media	16	24,24242424
Alta	42	63,63636364

Tabla 16: Reutilización

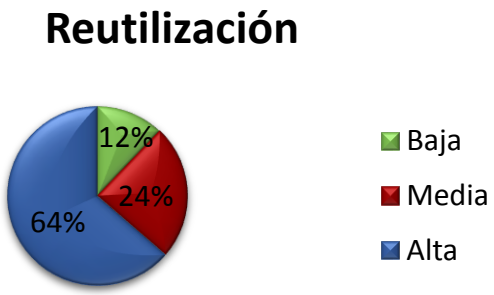


Figura 18: Representación de la evaluación de la métrica RC en el atributo reutilización

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que el diseño propuesto para el componente configuración de los presupuestos se encuentra dentro de los niveles de calidad requeridos. Los atributos de calidad fueron evaluados satisfactoriamente confirmando que las clases están diseñadas correctamente ya que presentan una elevada reutilización, un bajo acoplamiento, poca complejidad de mantenimiento y poca cantidad de pruebas, que implica que se necesitará menos esfuerzo a la hora de realizar pruebas unitarias a estas clases.

2.13 Conclusiones parciales

A partir de la arquitectura del sistema y la realización de los diagramas de clases y secuencia se obtuvo una propuesta para realizar la implementación del componente para la configuración de los presupuestos, de manera que cumpla con los patrones de diseño y los estándares de codificación definidos en el proyecto.

Con la elaboración del diseño del sistema se sentaron las bases para poder realizar la implementación y validación de la solución; tanto del diseño como de las funcionalidades.

CAPÍTULO 3 Implementación y prueba

3.1 Introducción

Teniendo en cuenta el capítulo anterior, es necesario definir cómo se desarrollará el sistema. Se da paso así al presente capítulo en el que se realiza la implementación y prueba del sistema, formado por los diagramas de componentes y de despliegue. Además se comprueba la calidad de los resultados, mediante los artefactos elaborados durante el flujo de trabajo de pruebas.

3.3 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes, entre los que se encuentran datos, archivos, ejecutables y código.

3.3.1 Diagrama de componentes

Representa cómo un sistema de software es dividido en componentes, representando las dependencias entre estos. Este tipo de diagrama contiene además interfaces y sus relaciones, pudiendo contener también paquetes que se utilizan para agrupar elementos del modelo.

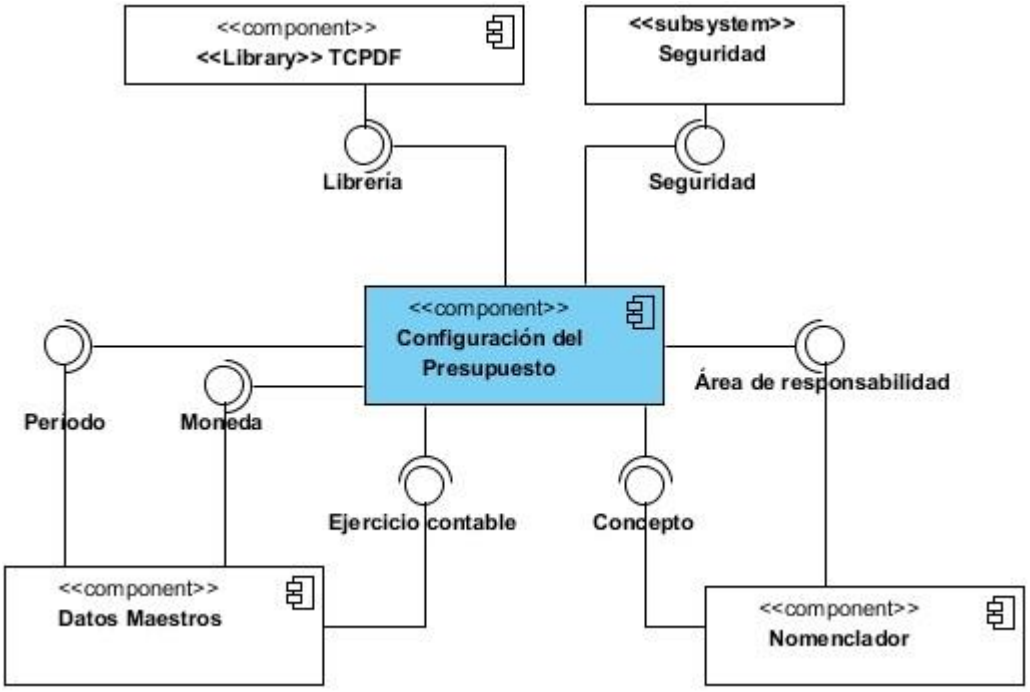


Figura 19: Diagrama de componentes

3.3.2 Diagrama de despliegue

Es un diagrama de objetos que representa la distribución física del sistema. El diagrama de despliegue es empleado para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

A continuación se presenta el diagrama de despliegue de la solución, el mismo estará compuesto por una (PC[16]) cliente conectada a través del protocolo de transferencia de hipertexto (HTTP[17]) a un servidor web que se comunica con el servidor de bases de datos mediante la extensión PDO, además de una impresora que se conecta mediante (USB[18]) o haciendo uso del protocolo de control de transmisión y el protocolo de internet (TCP/IP[19]) a la PC cliente.

[16] Personal Computer
[17] Hypertext Transfer Protocol
[18] Universal Serial Bus
[19] Transmission Control Protocol and Internet Protocol

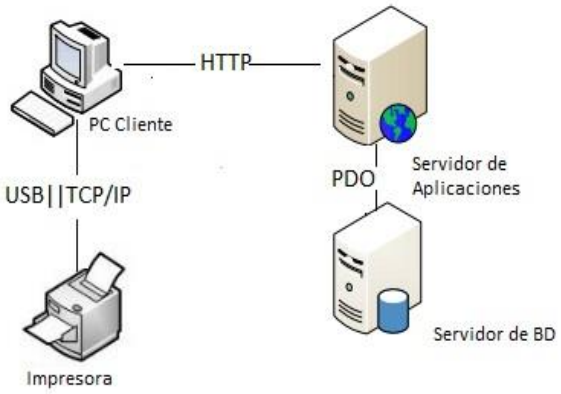


Figura 20: Diagrama de despliegue

3.4 Estándares de codificación

Los estándares de codificación son una serie de convenciones que deben seguir los desarrolladores manteniendo buenas prácticas definidas por la ingeniería de software, para obtener un código fácil de comprender y de alta calidad. Posibilita que un equipo de programadores mantenga un código legible sobre el que se efectuarán luego revisiones; con el objetivo de regular la calidad de la implementación estableciendo un estándar de desarrollo común por el cual se rige la programación del sistema.

3.4.1 PascalCasing

PascalCasing establece que los nombres de los identificadores, las variables, métodos y clases están compuestos por una o más palabras juntas, iniciando cada palabra con letra mayúscula y el resto en minúscula.

Nomenclatura de las clases según su tipo

Todas las clases están nombradas siguiendo el estándar PascalCasing, nombrándolas de acuerdo con el propósito y la función que realizan.

- **Controllers:** clases controladoras del dominio, su identificador está estructurado por el nombre de la misma seguido por la palabra “Controller”. Ejemplo: PresupuestoController.
- **Bussines:** clases que gestionan la lógica del negocio, su identificador está estructurado por el prefijo “Dat” y la función que realizan seguido de la palabra “Model”. Ejemplo: DatPresupuestoModel.

- **Domain:** clases que gestionan el acceso a la base de datos a través de consultas, su identificador está estructurado por el nombre de la tabla a la que acceden, pueden incluir el prefijo “Dat” o “Nom”. Ejemplo: DatPresupuesto.
- **Generated:** clases base del dominio encargadas de realizar el mapeo de modelo de objetos al modelo relacional, su identificador está estructurado por el prefijo “Base”, seguido de “Dat” o “Nom”. Ejemplo: BaseDatPresupuesto.

3.4.2 CamelCasing

CamelCasing es similar a PascalCasing con diferencia en la letra inicial del identificador que no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

Nomenclatura de las funciones

Los identificativos de las funciones o métodos se escriben con la primera palabra en minúscula de acuerdo con la función que realizan. Ejemplo: obtenerPresupuestos. Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: adicionarPresupuestoAction.

Nomenclatura de los atributos

El identificativo de los atributos se escribe de acuerdo con su objetivo, con la primera letra en minúscula. Ejemplo: idpresupuesto.

Nomenclatura de las variables

Las variables serán nombradas comenzando en minúscula, en caso de que el nombre de estas sea compuesto, se utilizará la notación CamelCasing, o sea, seguido del primer nombre, comenzará el segundo en mayúscula. En la capa de la vista, las variables que contengan componentes que formen parte de la interfaz, se les colocará delante un sufijo que indique el tipo de componente que contiene. Ejemplo: btnModificar, frmPresupuesto, gridNotificar.

3.5 Servicios utilizados

La comunicación que se establece entre diferentes componentes y subsistemas de la aplicación se basa en el funcionamiento del componente llamado IoC. El IoC es donde están publicados los métodos que brindan cada módulo del proyecto XEDRO ERP. La utilización de este se evidencia en el fichero .xml con igual nombre (ioc.xml), brinda servicios de los componentes del

sistema y especifica respuestas deseadas a sucesos o solicitudes de datos concretos, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Componente	Servicios	Descripción
Datos Maestros	buscarMoneda	Mediante este servicio se obtiene la moneda para luego utilizarlo en la notificación y desagregación del presupuesto.
Datos Maestros	obtenerEjercicio	Mediante este servicio se obtiene todos los ejercicios contables para utilizarlos en la funcionalidad adicionar y versionar presupuesto.
Nomenclador	obtenerElementoGasto	Mediante este servicio se obtiene todos los conceptos para luego utilizarlos en la desagregación del presupuesto.
Nomenclador	obtenerCentroCosto	Mediante este servicio se obtiene todas las áreas responsables para luego utilizarlo en la desagregación del presupuesto.
Datos Maestros	obtenerPeriodo	Mediante este servicio se obtiene el período, que es utilizado para realizar la notificación y desagregación del presupuesto.

Tabla 17: Servicios que consume el componente configuración de los presupuestos

3.6 Pruebas de software

3.6.1 Prueba de caja blanca

La técnica de caja blanca empleada en la solución fue la del camino básico, para utilizarla se hace necesario el cálculo de la complejidad ciclomática del algoritmo que vaya a ser analizado, para lo cual se deben enumerar las sentencias de código y a partir de ahí elaborar el grafo de flujo de esta funcionalidad.

La **complejidad ciclomática:** es la métrica de software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Con esta métrica se determina la cantidad de caminos independientes de cada una de las funcionalidades del programa. También provee el límite superior de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (33).

La complejidad ciclomática $V(G)$ de un grafo de flujo G se calcula de varias formas, tres de ellas son:

- El número de regiones del grafo de flujo.
- $V(G) = A - N + 2$, donde A es el número de aristas y N el número de nodos.
- $V(G) = P + 1$, donde P es el número de nodos predicado (*aquellos de los cuales parten dos o más aristas*) contenidos en el grafo de flujo G.

Aplicación de la prueba

A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método obtenerProgramacionAction, y se creará el grafo de flujo para calcular la complejidad ciclomática del mismo.

```
function obtenerProgramacionAction() {
        $programacion = $this->model[3]->obtenerProgramacionModel(
                    $this->getRequest()->getParam('start'),
                    $this->getRequest()->getParam('limit'),
                    $this->getRequest()->getParam('idelementogastopatrimonial'),
                    $this->getRequest()->getParam('idpresupuesto')
                );
            $idestructura = $this->global->Estructura->idestructura;
            $monedas = $this->integrator->parametros->BuscarMoneda(0,$idestructura);
            $periodos = $this->integrator->parametros->ObtenerPeriodos($this->getRequest()->getParam('idejercicio'));
            $result = array();
            foreach ($programacion as $p) {
                $codigoiso;
                $nombre;
                foreach ($monedas as $m) {
                    if ($p['idnommonedaentidad'] == $m->idnommonedaentidad) {
                            $codigoiso = $m->codigoiso;
                     }
                }foreach ($periodos as $per) {
                  if ($p['idperiodo'] == $per->idperiodo) {
                    $nombre = $per->nombre;
                  }
              }$result[] = array(
                        'idprogramacion' => $p['idprogramacion'],
                        'nombre' => $nombre,
                        'monto' => $p['monto'],
                        'codigoiso' => $codigoiso
                    );
            }echo json_encode(array(
            'data' => $result,
            'cant' => $this->model[3]->cantidadProgramacionModel(
                        $this->getRequest()->getParam('idelementogastopatrimonial'),
                        $this->getRequest()->getParam('idpresupuesto')
            )));
    }
```

Figura 21: Código fuente del método obtenerProgramacionAction

A continuación se muestra el grafo de flujo asociado al código anterior:

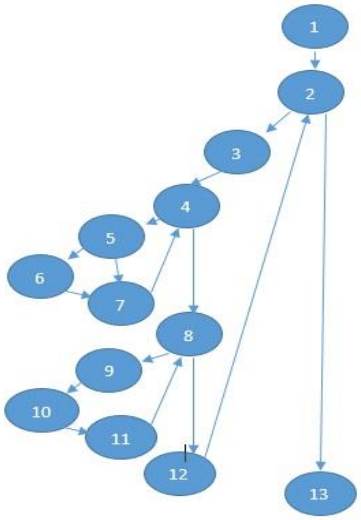


Figura 22: Grafo de flujo del método obtenerProgramacionAction

Luego se calcula la complejidad ciclomática, mediante los métodos mencionados anteriormente:

$$V(G) = (A - N) + 2, \qquad V(G) = (16 - 13) + 2, \qquad V(G) = 5$$

Siendo A la cantidad total de aristas del grafo y N la cantidad de nodos.

$$\mathrm{V\,(G)} = \mathrm{P} + 1, \ \mathrm{V\,(G)} = (4 + 1), \ \mathrm{V\,(G)} = 5$$

Siendo P la cantidad de nodos predicado.

$$V(G) = R, \ V(G) = 5$$

Siendo R la cantidad de regiones que posee el grafo.

Cada una de las fórmulas $V(G)$ representa el valor del cálculo de la complejidad ciclomática. A partir de los resultados obtenidos en cada una de ellas, se determina que la complejidad ciclomática del código analizado es 6, que a su vez es el número de caminos posibles a circular el flujo y el límite superior de casos de prueba que se le pueden aplicar a dicho código. A continuación se muestran los caminos básicos por donde puede circular el flujo:

Número	Caminos básicos
1	1-2-13
2	1-2-3-4-8-9-10-11-8-12-2-13
3	1-2-3-4-5-7-4-8-12-2-13
4	1-2-3-4-5-6-7-4-8-12-2-13
5	1-2-3-4-8-12-2-13

Tabla 18: Caminos básicos del flujo

Cada camino independiente es un caso de prueba a realizar, de forma tal que los datos introducidos provoquen que se visiten las sentencias vinculadas a cada nodo del camino; a continuación se procede a ejecutar los casos de prueba.

Caso de prueba para el camino básico # 1

Camino: 1-2-13

Descripción	A partir del idelementogastopatrimonial y el idpresupuesto se obtienen todas las programaciones.
Condición de ejecución	Se debe tener el identificador del elemento del gasto (“idelementogastopatrimonial”). Se debe tener el id de presupuesto (“idpresupuesto”).
Entrada	$this->getRequest()->getParam(90000000021), $this->getRequest()->getParam(17)
Resultados esperados	La estructura no tiene programaciones asociadas, por lo que no devuelve nada.
Resultados	Cantidad 0, datos array (). Este resultado demuestra que no existen programaciones asociadas a esta estructura.
Salida	0

Tabla 19: Caso de prueba del camino básico #1

Caso de prueba para el camino básico # 2

Camino: 1-2-3-4-8-9-10-11-8-12-2-13	
Descripción	A partir del idelementogastopatrimonial y el idpresupuesto se obtienen todas las programaciones.
Condición de ejecución	Se debe tener el identificador del elemento del gasto (“idelementogastopatrimonial”). Se debe tener el id de presupuesto (“idpresupuesto”).
Entrada	$this->getRequest()->getParam(90000000020), $this->getRequest()->getParam(201)
Resultados esperados	Devuelve todas las programaciones cuyo idelementogastopatrimonial y el idpresupuesto coincidan con los parámetros enviados.

Resultados	Cantidad 2, datos array 1 [idelementogastopatrimonial, monto, idmoneda, idperiodo]; array 2 [idelementogastopatrimonial, monto, idmoneda, idperiodo]. Este resultado demuestra que existen programaciones asociadas a esta estructura.
Salida	2

Tabla 20: Caso de prueba del camino básico #2

Caso de prueba para el camino básico # 3

Camino: 1-2-3-4-5-7-4-8-12-2-13	
Descripción	A partir del idelementogastopatrimonial y el idpresupuesto se obtienen todas las programaciones.
Condición de ejecución	Se debe tener el identificador del elemento del gasto (“idelementogastopatrimonial”). Se debe tener el id de presupuesto (“idpresupuesto”).
Entrada	$this->getRequest()->getParam(90000000022), $this->getRequest()->getParam(18)
Resultados esperados	Devuelve todos los valores de las programaciones excepto el período y la moneda.
Resultados	Cantidad 1, datos array [idelementogastopatrimonial, monto, idmoneda, idperiodo]. Este resultado demuestra que existen programaciones asociadas a esta estructura.
Salida	1

Tabla 21: Caso de prueba del camino básico #3

Caso de prueba para el camino básico # 4

Camino: 1-2-3-4-5-6-7-4-8-12-2-13

Descripción	A partir del idelementogastopatrimonial y el idpresupuesto se obtienen todas las programaciones.
Condición de ejecución	Se debe tener el identificador del elemento del gasto (“idelementogastopatrimonial”). Se debe tener el id de presupuesto (“idpresupuesto”).
Entrada	$this->getRequest()->getParam(90000000023), $this->getRequest()->getParam(19)
Resultados esperados	Devuelve todos los valores de las programaciones excepto el período.
Resultados	Cantidad 1, datos array [idelementogastopatrimonial, monto, idmoneda, idperiodo]. Este resultado demuestra que existen programaciones asociadas a esta estructura.
Salida	1

Tabla 22: Caso de prueba del camino básico #4

Caso de prueba para el camino básico # 5

Camino: 1-2-3-4-8-12-2-13	
Descripción	A partir del idelementogastopatrimonial y el idpresupuesto se obtiene todas las programaciones.
Condición de ejecución	Se debe tener el identificador del elemento del gasto (“idelementogastopatrimonial”). Se debe tener el id de presupuesto (“idpresupuesto”).
Entrada	$this->getRequest()->getParam(90000000024), $this->getRequest()->getParam(20)
Resultados	Devuelve todos los valores de las programaciones excepto el período y la

esperados	moneda.
Resultados	Cantidad 1, datos array 1 [idelementogastopatrimonial, monto, idmoneda, idperiodo]. Este resultado demuestra que existen programaciones asociadas a esta estructura.
Salida	1

Tabla 23: Caso de prueba del camino básico #5

3.6.2 Prueba de caja negra

Las pruebas de caja negra incluyen varias técnicas como:

Técnica de la partición de equivalencia: esta divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Esta permite examinar los valores válidos e inválidos de las entradas existentes en el software (33).

Técnica del análisis de valores límites: la experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límites son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica indica los casos de prueba que ejerciten los valores límites (33).

Para la realización de las pruebas de caja negra se empleó la técnica partición de equivalencia. A continuación se muestra un ejemplo del caso de prueba del requisito adicionar presupuesto.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1. Adicionar presupuesto.	Debe permitir adicionar un presupuesto.	EP 1.1: Adicionar un presupuesto introduciendo datos válidos.	– Se introducen los datos de la funcionalidad correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información.
		EP 1.2: Adicionar un presupuesto introduciendo datos que no cumplan con el formato.	– Se introducen los datos que no cumplen con el formato de algún campo de la funcionalidad. – El sistema no permitirá que el usuario inserte la primera letra en minúscula. El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que este valor es incorrecto, al llenarse este campo se presiona el botón **Aceptar** – Se muestra un mensaje de información.
		EP 1.3: Adicionar un presupuesto dejando campos vacíos.	– Se introducen los datos dejando algún campo necesario en blanco. – El sistema señala en rojo el campo vacío, al pasarle por encima muestra un mensaje informando que este campo es obligatorio, al llenarse este campo se presiona el botón **Aceptar** y si aún hay campos necesarios sin datos se repite el proceso.

Figura 23: Diseño de caso de prueba del requisito adicionar presupuesto

Escenario	Descripción	Denominación	Versión	Fecha	Estado	Tipo	Ejercicio	Anexo	Respuesta del sistema
EC 1.1 Flujo básico Adicionar Presupuesto	Se muestra un formulario con los datos para insertar del presupuesto	V Presupuesto 3	V 1.1	V 12/05/2014	V Activo	V Ingreso	V 2012	V Archivo jpg	El sistema adiciona la funcionalidad y muestra un mensaje de información "Se ha adicionado el presupuesto satisfactoriamente".
		N/A vacío	N/A vacío	N/A vacío	N/A vacío	N/A vacío	N/A vacío	N/A vacío	El sistema informa que no hay conexión con el componente.
		V Presupuesto 1	I vacío	V 12/05/2014	V Inactivo	V Inversiones	V 2013	V Archivo pdf	El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que "Este campo es obligatorio".
		I presupuesto9	V 1.2	V 12/05/2014	V Inactivo	V Gasto	V 2013	V Archivo jpg	El sistema no permitirá que el usuario inserte la primera letra en minúscula. El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que "Este valor es incorrecto". Luego de haberlos corregidos se presiona el botón Aceptar.
		V Presupuesto4	V 1.3	V 09/05/2014	V Inactivo	V Gasto	I vacío	V Archivo pdf	El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que "Este campo es obligatorio".
		V Presupuesto4	V 1.2	I vacío	V Inactivo	V Gasto	V 2013	V Archivo jpg	El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que "Este campo es obligatorio".
		V Presupuesto4	V 1.3	V 09/05/2014	V Activo	I vacío	V 2011	V Archivo jpg	El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que "Este campo es obligatorio".
		V Presupuesto4	V 1.3	V 09/05/2014	V Activo	V Gasto	V 2011	N/A vacío	El sistema adiciona la funcionalidad y muestra un mensaje de información "Se ha adicionado satisfactoriamente".
		V Presupuesto4	V 1.3	V 09/05/2014	I vacío	V Gasto	V 2011	V Archivo pdf	El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que "Este campo es obligatorio".

Figura 24: Juegos de datos del requisito adicionar presupuesto

Con la aplicación de las pruebas de caja negra se validó las funcionalidades implementadas en el componente configuración de los presupuestos. Estas se realizaron en tres iteraciones, donde finalmente se comprobó en la tercera iteración que el componente estaba libre de no conformidades culminando así la fase de pruebas internas. En las mismas se detectaron errores mayormente de faltas de ortografías en los distintos tipos de mensajes, que no coincidían con los descritos en los casos de prueba, así como errores en las validaciones.

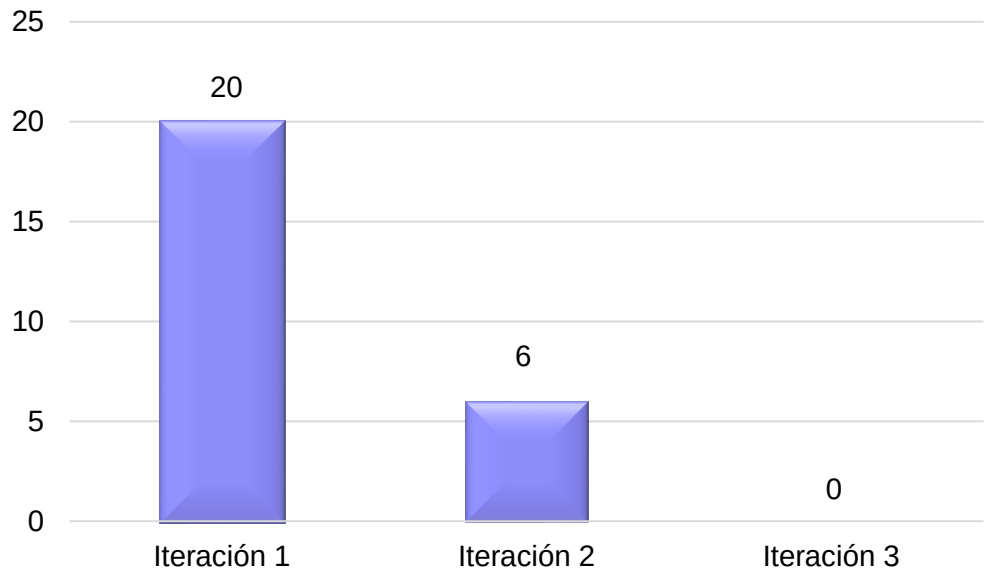


Figura 25: Resultado de la prueba de caja negra

3.6.3 Validación de la propuesta de solución

Para la investigación se plantea como idea a defender: “Si se desarrolla un componente para el sistema XEDRO ERP que permita la gestión de la configuración de los presupuestos se podrá aumentar la velocidad en el procesamiento de la información”. A continuación se evalúa la variable **velocidad en el procesamiento de la información** con el uso del componente para la configuración de los presupuestos del sistema XEDRO ERP.

Velocidad en el procesamiento de la información

Es la velocidad a la que se gestionan las distintas actividades del proceso de configuración del presupuesto. La maximización de esta variable reducirá el tiempo empleado para la gestión de estas actividades y de esta manera se haría más corta la espera del proceso de ejecución del presupuesto, como se muestra en la siguiente tabla comparativa entre la velocidad de procesamiento de la información utilizando el sistema y antes del mismo. Tomando como caso de estudio la configuración del presupuesto desarrollada en el 2014 por la dirección de planificación y estadística de la UCI donde las actividades de notificación y desagregación se realizan a través de Excel, una de las especialistas fue desarrollando el proceso en Excel y al unísono otra de ellas fue ejecutándolo en el sistema. Obteniéndose mediante la aplicación de los métodos científicos observación y medición los siguientes resultados:

Actividades del proceso de configuración del presupuesto				
No	**Sin el sistema**	**Con el sistema**	**Antes**	**Después**
1	Adicionar presupuesto	Adicionar presupuesto	20 min	1 min
2	Modificar presupuesto	Modificar presupuesto	2 días	15 min
3	Buscar presupuesto	Buscar presupuesto	1 hora	15 seg
4	Notificar presupuesto	Notificar presupuesto	2 Días	2 horas
5	Desagregar presupuesto por áreas responsables, conceptos y período	Desagregar presupuesto por áreas responsables, conceptos y período	8 días	4 horas
6	Desagregar por conceptos y período	Desagregar por conceptos y período	6 días	3 horas
7	Generar reportes	Generar reportes	4 horas	10 min
8	Generar modelos	Generar modelos	1 día	1 hora
	Total		19 días, 5 horas y 20 minutos	10 horas, 26 minutos y 15 segundos

Tabla 24: Validación de la variable

3.7 Conclusiones parciales

Al aplicar las métricas al diseño se verificó la complejidad, responsabilidad, acoplamiento y reutilización que tendrán las clases dentro de la aplicación. Quedó demostrado que a partir del diseño propuesto, la implementación de las clases sería un proceso sencillo puesto que todos los valores obtenidos se encontraron dentro del rango de error permisible para las métricas aplicadas.

Con las pruebas de caja blanca y caja negra se comprobó que todas las funcionalidades especificadas y aprobadas por el cliente fueron implementadas de forma correcta.

La comparación antes realizada, tomando como caso de estudio la configuración del presupuesto desarrollada en el 2014 por la dirección de planificación y estadística de la UCI, evidencia que la incorporación del sistema en el trabajo de los especialistas de las entidades cubanas permitirá un aumento considerable de la velocidad en el procesamiento de la

información y por consiguiente una disminución notable en la ejecución del proceso de configuración del presupuesto.

CONCLUSIONES

Con la realización del presente trabajo se desarrolló el componente para la configuración de los presupuestos para el sistema XEDRO ERP, contribuyendo de esta manera a la gestión de los presupuestos de las entidades cubanas. Es por ello que al finalizar la presente investigación se puede afirmar que:

- El estudio de los conceptos fundamentales y los componentes para la configuración de los presupuestos definidos como objeto de estudio, permitió determinar los elementos necesarios para unidos con las exigencias del MFP conformar un conjunto de funcionalidades esenciales que debía cumplir la aplicación.
- El diseño de la solución permitió la obtención de un modelo en el que se visualiza la manera en que debe ser implementado el sistema y de esta forma facilitar el cumplimiento de las funcionalidades del mismo.
- A partir de esta implementación se obtuvo una aplicación funcional capaz de satisfacer las necesidades del cliente y resolver el problema por el se decidió comenzar el desarrollo del sistema.
- Con la validación de la solución se pudo comprobar y demostrar de forma cuantitativa la calidad de los artefactos obtenidos durante el desarrollo del sistema.

RECOMENDACIONES

Luego de llevar a cabo el cumplimiento y desarrollo de los objetivos propuestos se ofrecen las siguientes recomendaciones:

- Generar reportes gráficos que permitan un mejor análisis y comprensión de los presupuestos económicos.
- Proseguir con la investigación e identificar nuevos requisitos para actualizar el sistema desarrollado.
- Realizar el despliegue del componente propuesto como parte del Sistema Integral de Gestión XEDRO ERP.
- Integrar el componente para la configuración de los presupuestos del sistema XEDRO ERP al módulo de notificaciones.

BIBLIOGRAFÍA

1. **MARLIS MURILLO NAVARRO, ILSE SAINZ MONDRAGON, LUIS HIRAM SALCEDO MEDINA.** [En línea] 05 de Noviembre de 2009. [Citado el: 5 de Mayo de 2014.] http://www.buenastareas.com/ensayos/De-Que-Manera-Beneficia-La-Implementacion/159392.html.

2. **Carbonero, Miguel Ángel.** sitio de la Universidad de Carlos III en Madrid,Implantación de un ERP. [En línea] 8 de agosto de 2011.

3. **Sánchez, Lilian Cid Escalona Ismel Sarduy.** *Análisis y diseño del subsistema Multimoneda del ERP-Cuba. .* Mayo,2009.

4. **Sistema de Administración Empresarial.** ***Sistema de Administración Empresarial.*** **[En línea] [Citado el: 02 de 05 de 2014.] http://cursos.aiu.edu/Informatica%20II/PDF/Tema%205.pdf.**

5. Archive, Arturo Komiya. CreceNegocios. ***CreceNegocios.*** **[En línea] [Citado el: 10 de 03 de 2014.] http://www.crecenegocios.com/los-presupuestos-de-una-empresa/.**

6. Herrera, Víctor Manuel. sitio GestioPolis. [En línea] junio de 2001.

7. Badilla, Ricardo Montaño. sitio GestioPolis. [En línea] 10 de febrero de 2010.

8. Herramientas Financieras, Herramientas. Ministerio de Producción Peruana. [En línea]

9. ***Véliz, Aydil Orama. 5 de octubre de 2012.***

10. Horngren, Charles T. Contabilidad de Costos. 2006.

11. Ministerio de Finanzas y Precios, Precios y. Ley No 192 de la Administración Financiera del Estado Cubano. abril de 1999 .

12. MInisterio de Finanzas y Precios, Precios. Resolución 241. 2012.

13. ERP. PERU-ERP.COM. [En línea] ADVANCE GLOBAL CONSULTING S.A.C., 2010-2011.

14. OpenERP . abartek. [En línea] 2010.

15. Romero, Pedro Manuel Baeza. Scribd. [En línea] 25 de Mayo de 2012.

16. Izquierdo, Susana. Abartia Team. Gestión presupuestaria en OpenERP. [En línea] 2011.

17. Adempire.com. [En línea] 21 de noviembre de 2011.

18. Romero, Lorely Moya. eumet. Funcionalidades y Principales Opciones del Modulo de Planificación del Software Integrado VERSAT Sarasola. [En línea] 2009.

19. —. eumet. Funcionalidades y Principales Opciones del Modulo de Planificación del Software Integrado VERSAT Sarasola. [En línea] 2009.

20. Rodas XXI . rodasxxi. Sistema Integral Ecónomico Administrativo. [En línea]

21. CEIGE, Centro. CEIGE-Modelo de Desarrollo de Software v1.2.

22. SOLUCIÓN INFORMÁTICA PARA GESTIONAR NÓMINAS MEDIANTE XEDRO ERP. William González Obregón, Arnolis Salgueiro Arzuaga, Yarenis Echemendía González, Giselle Almeida González. La Habana : s.n.

23. Pressman, Roger. Ingeniería de Software. Un enfoque práctico. 6ta Edicción : s.n., 2005.

24. —. Ingeniería del Software.Un enfoque práctico.Cap_14_Tecnicas_de_Prueba_Parte_1. s.l. : 6ta edición, 2005.

25. Raghavan, Sridhar, Zelesnik, Gregory y Ford, Gary A. Lecture Notes on Requirements Elicitation. Pittsburgh (E.E.U.U.) : Institute (Carnegie Mellon University), 1994.

26. Liliam Celia Beyris Soulary, Ludisley La Torre Hernández,Mariela Cepero Núñez,Annelis ,Irina Marrero Borges, Yailién Hernández Alba,Katia Onelia Carralero y Amado Espinosa Hidalgo. Proceso de medición y análisis para el polo de hardware y automática. s.l. : Grupo Editorial Ediciones Futuro ,Serie Científica de la Universidad de las Ciencias Informáticas (SC-UCI) , 2010.

27. Baryolo, Oiner Gómez. SOLUCIÓN INFORMÁTICA DE AUTORIZACIÓN EN ENTORNOS MULTIENTIDAD Y MULTISISTEMA. La Habana : s.n., 2010 : s.n.

28. larman, craig. UML y Patrones,Introducción al análisis y diseño orientado a objetos,07_Parte_VII_Fase_de_Diseno_2_. Mexico : Prentice hall, 1999.

29. Larman, Craig. 04_Parte_IV_Fase_del_Diseno_1. UML y Patrones.Introducción al análisis y diseño orientado a objetos. 1999.

30. Pressman, Roger. Ingeniería del Software.Un enfoque práctico.Cap_08_Modelado_del_Analisis_Parte_3. 6ta Edicción : s.n., 2005.

31. Gutierrez, korina Saldaval. MÉTRICAS PARA LA EVALUACIÓN DE LA COMPLEJIDAD DE BASES DE DATOS RELACIONALES . [En línea] 30 de octubre de 2012.

32. Coral Calero, Mario Piattini,Macario Polo,Farnacisco Ruiz. Métricas para la evaluación de la complejidad de las bases datos relacionales. Departamento de informática de la universidad de Castilla-La Mancha. España : s.n., 2000.

33. Roger, Pressman. Un enfoque práctico. 2005.

Anexo #1: Diagrama de Clases

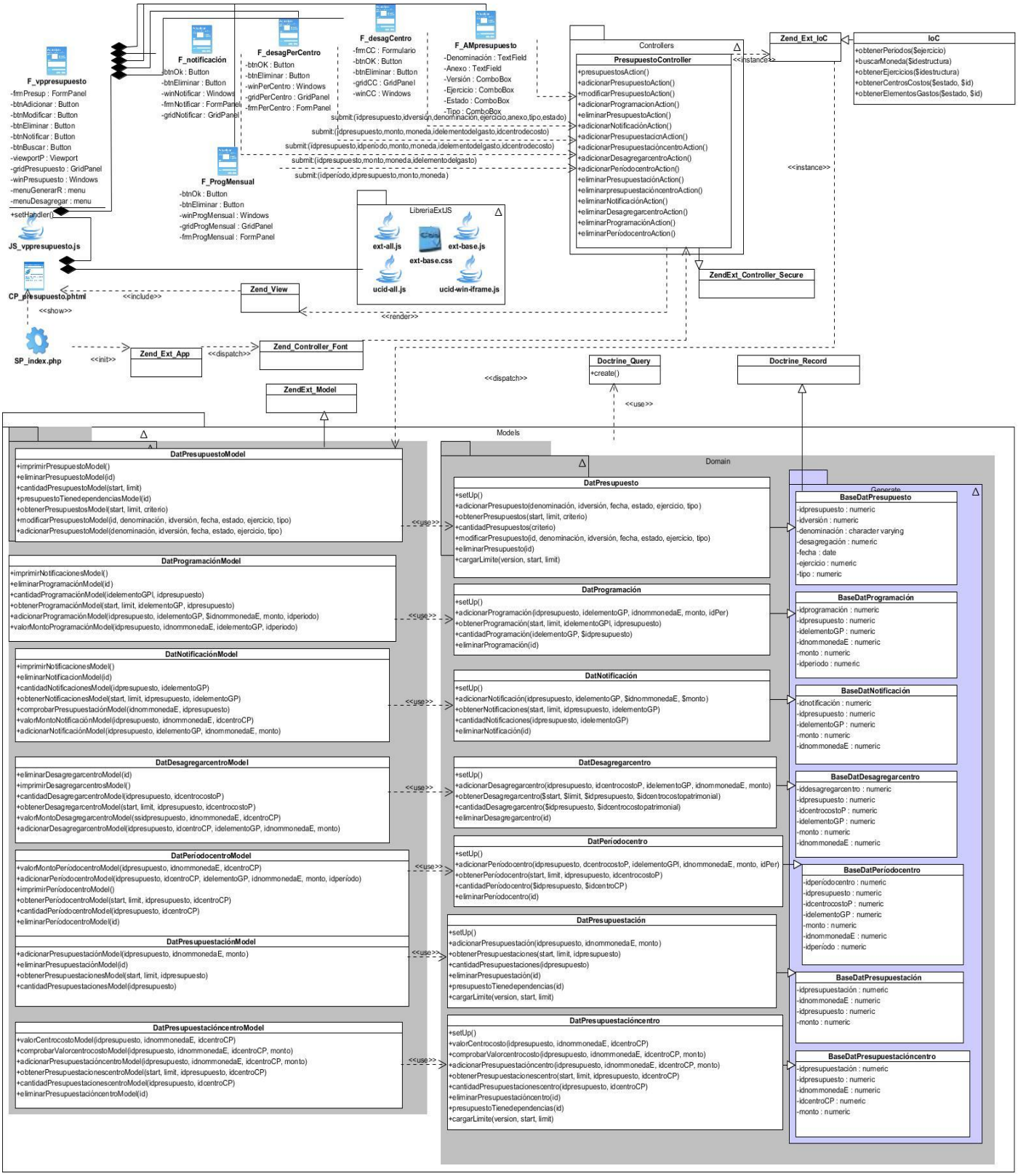


Figura 26: Diagrama de clases

Anexo # 2: Caso de Prueba para el requisito notificar presupuesto por áreas responsables

	Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1.	Notificar presupuesto por áreas de responsabilidad.	Debe permitir notificar un presupuesto por áreas de responsabilidad.	EP 1.1: Notificar un presupuesto por áreas de responsabilidad introduciendo datos válidos.	– Se introducen los datos de la funcionalidad correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Cancelar**.
			EP 1.2: Notificar un presupuesto por áreas de responsabilidad introduciendo datos que no cumplan con el formato.	– Se introducen los datos que no cumplen con el formato de algún campo de la funcionalidad. – El sistema no permitirá que el usuario inserte letras en el campo monto. El sistema señala en rojo el campo, al pasarle por encima muestra un mensaje informando que este valor es incorrecto, al llenarse este campo correctamente se presiona el botón **Aceptar** – Se muestra un mensaje de información. – Se presiona el botón **Cancelar**.
			EP 1.3: Notificar un presupuesto por áreas de responsabilidad dejando campos vacíos.	– Se introducen los datos dejando algún campo necesario en blanco. – El sistema señala en rojo el campo vacío, al pasarle por encima muestra un mensaje informando que este campo es obligatorio, al llenarse este campo se presiona el botón **Aceptar** y si aún hay campos necesarios sin datos se repite el proceso. – Se presiona el botón **Cancelar**.

Figura 27: Caso de prueba del requisito notificar por áreas responsables

Anexo # 3: Prototipo de interfaz del requisito notificar monto total por moneda

Figura 28: Prototipo de interfaz del requisito notificar monto total por moneda

Anexo # 4: Prototipo de interfaz del requisito notificar presupuesto por áreas responsables

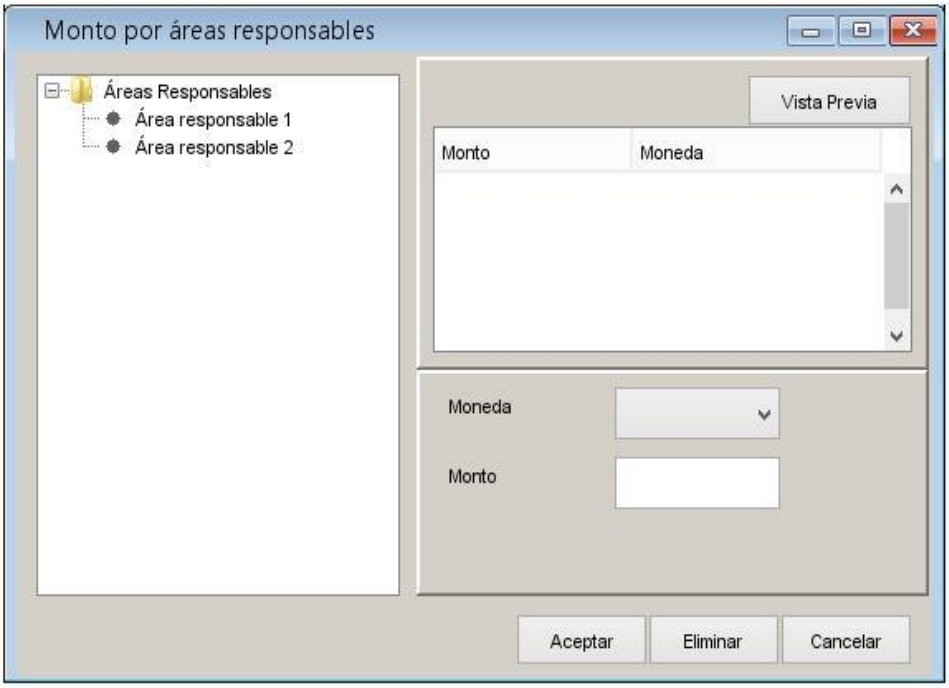


Figura 29: Prototipo de interfaz del requisito notificar presupuesto por áreas responsables

Anexo # 5: Prototipo de interfaz del requisito notificar presupuesto por conceptos

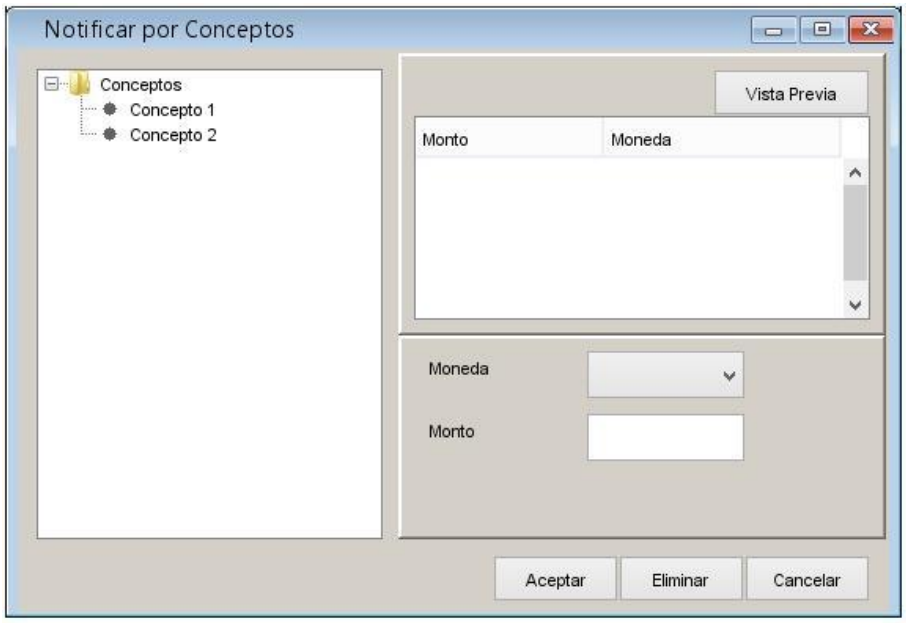


Figura 30: Prototipo de interfaz del requisito notificar presupuesto por conceptos

Anexo # 6: Prototipo de interfaz del requisito desagregar presupuesto por áreas responsables

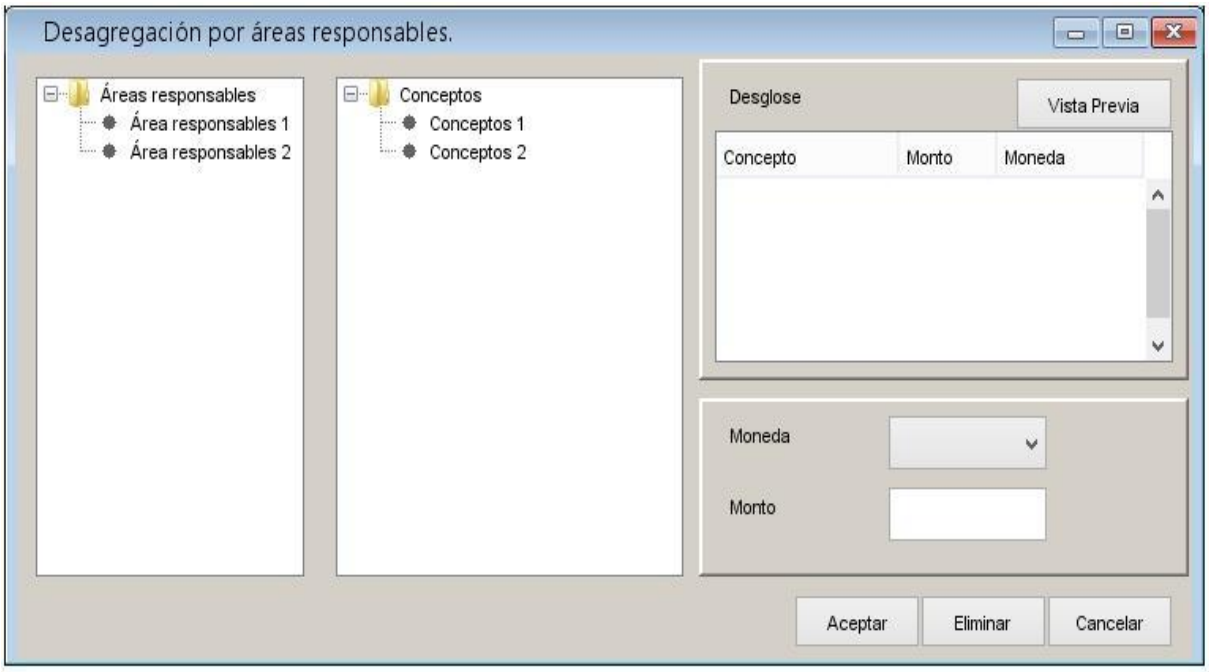


Figura 31: Prototipo de interfaz del requisito desagregar presupuesto por áreas de responsabilidad

Anexo # 7: Prototipo de interfaz del requisito desagregar presupuesto por conceptos y período

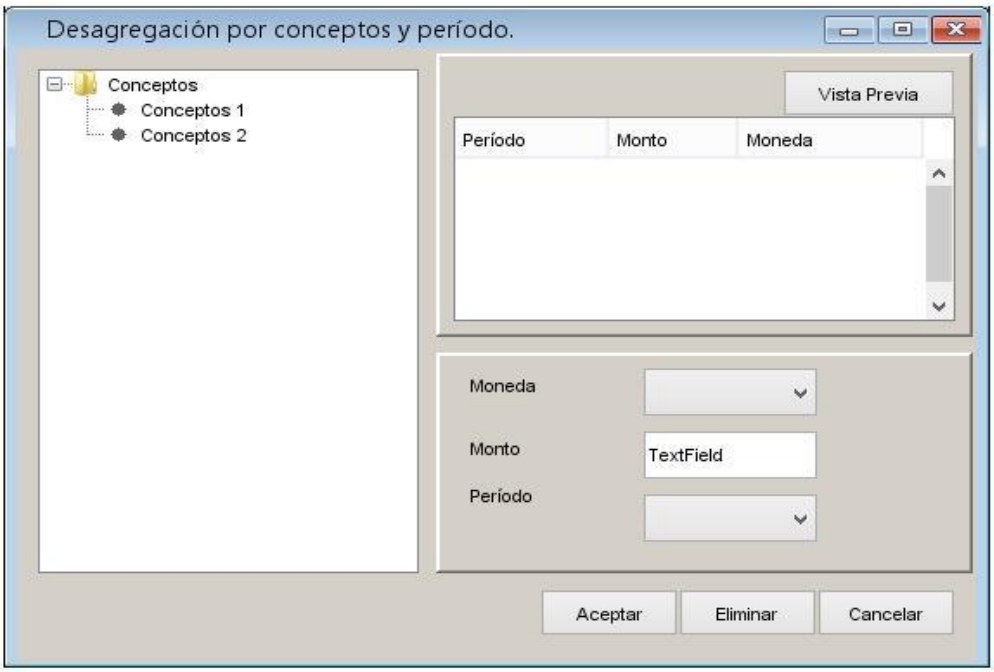
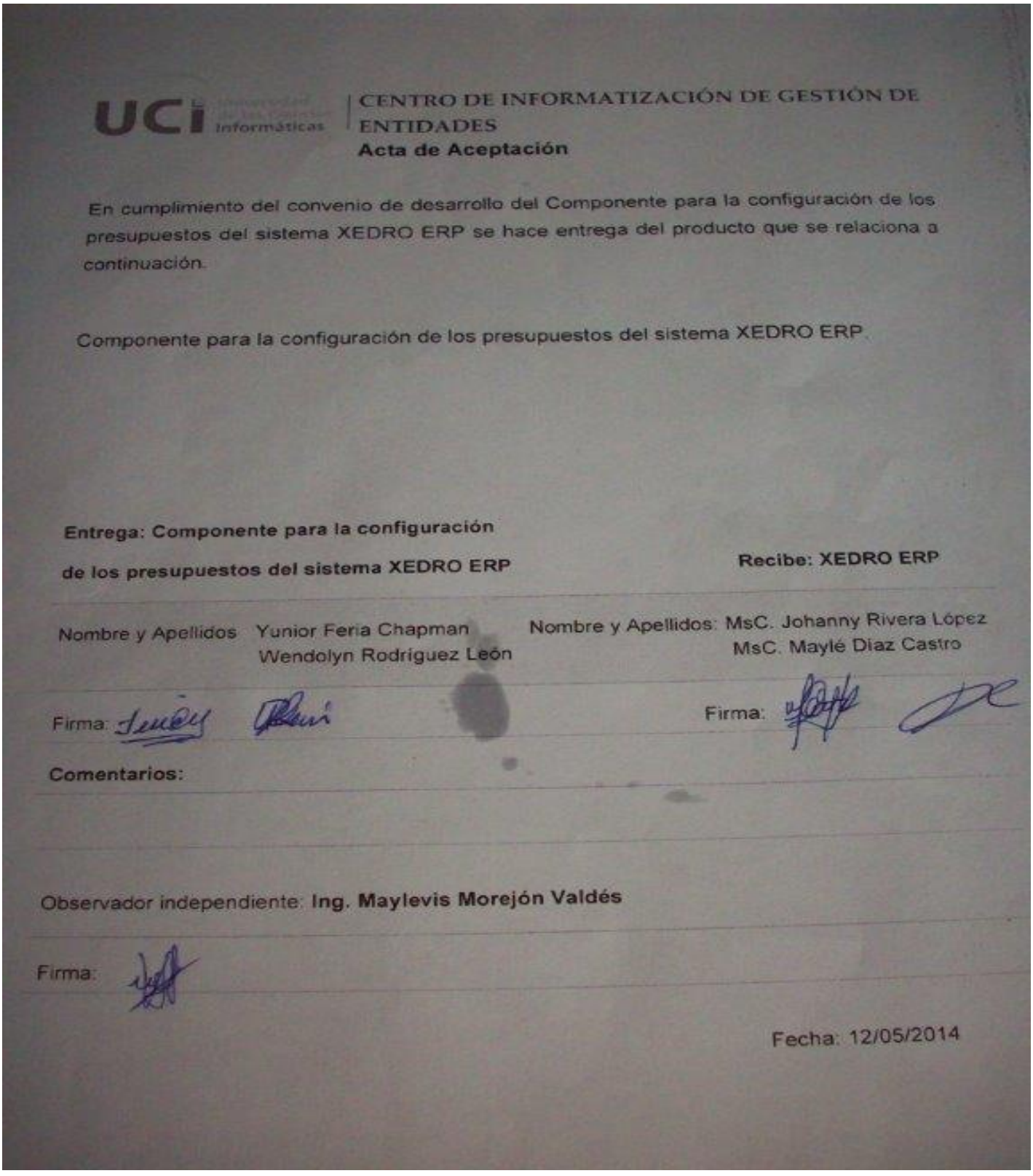


Figura 32: Prototipo de interfaz del requisito desagregar presupuesto por conceptos y período

Anexo # 8: Prototipo de interfaz del requisito desagregar presupuesto por áreas responsables, conceptos y período

Figura 33: Prototipo de interfaz del requisito desagregar presupuesto por áreas de responsabilidad, conceptos y período

Anexo # 9: Acta de aceptación del producto terminado



UCI Universidad de las Ciencias Informáticas

CENTRO DE INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

Acta de Aceptación

En cumplimiento del convenio de desarrollo del Componente para la configuración de los presupuestos del sistema XEDRO ERP se hace entrega del producto que se relaciona a continuación.

Componente para la configuración de los presupuestos del sistema XEDRO ERP

Entrega: Componente para la configuración de los presupuestos del sistema XEDRO ERP

Recibe: XEDRO ERP

Nombre y Apellidos: Yunior Feria Chapman, Wendolyn Rodríguez León

Nombre y Apellidos: MsC. Johanny Rivera López, MsC. Maylé Díaz Castro

Firma:

Firma:

Comentarios:

Observador independiente: **Ing. Maylevis Morejón Valdés**

Firma:

Fecha: 12/05/2014

Figura 34: Acta de aceptación del producto terminado