UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 7

TÍTULO: "IMPLEMENTACIÓN DE LOS COMPONENTES CONFIGURACIÓN Y NOMENCLADORES DEL SUBSISTEMA FACTURACIÓN DEL SISTEMA INTEGRAL DE GESTIÓN CEDRUX".

Autor: Reynier Santiesteban Rojas.

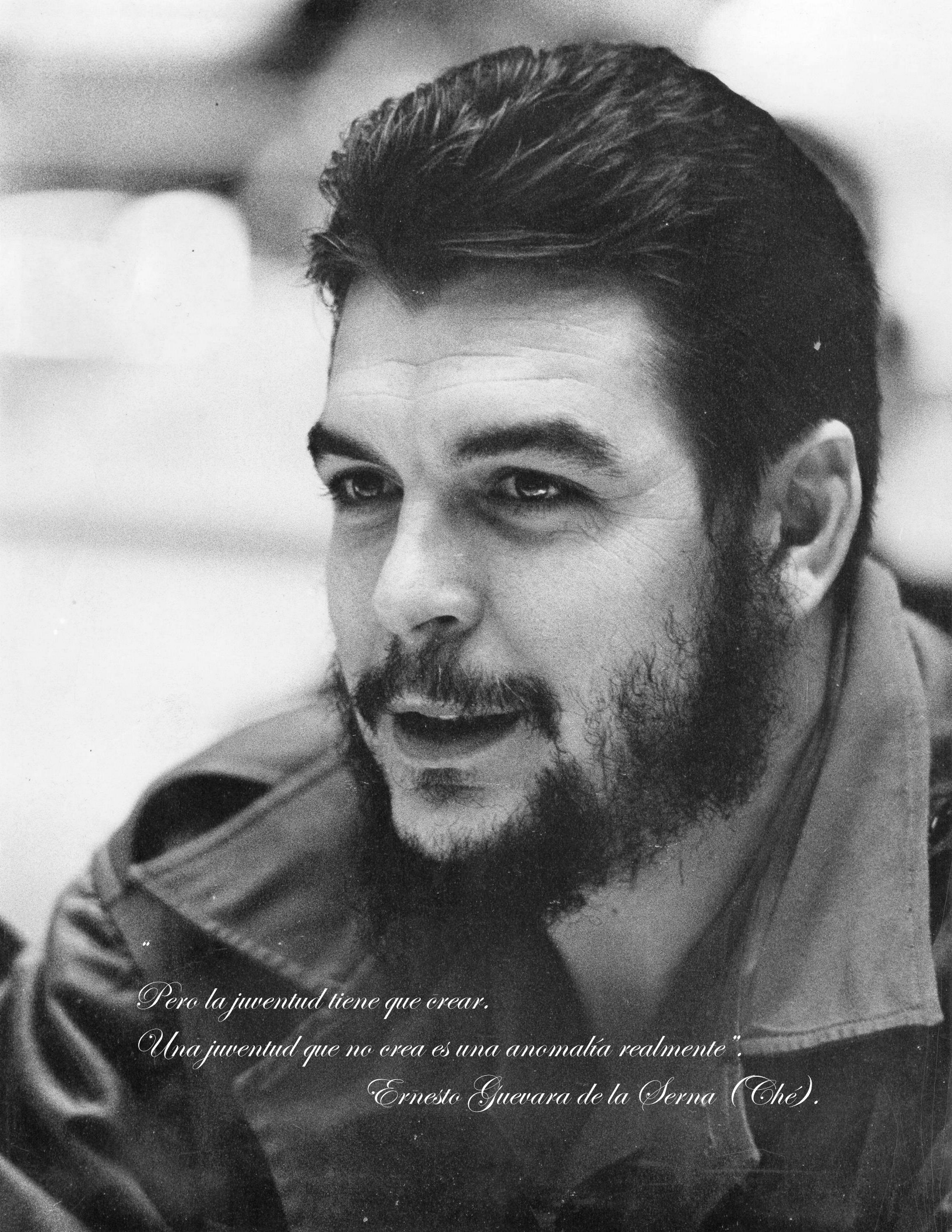
Tutor: Lic. Arismayda Dorado Risco.

Consultante: Ing. Enrique Chaviano Gómez.

Asesor: Msc. Jorge Luis León González.

Ciudad de La Habana

Junio - 2009.



"

Pero la juventud tiene que crear.

Una juventud que no crea es una anomalía realmente".

Ernesto Guevara de la Serna (Ché).

DATOS DE CONTACTO

Tutor: Lic. Arismayda Dorado Risco.

Categoría Docente: Instructor.

Correo electrónico: dorado@uci.cu

Síntesis del Tutor:

Profesora con 4 años de experiencia. Licenciada en Ciencias de la Computación. Actualmente lleva el rol de Arquitecto de Sistema del Subsistema Inventario del proyecto ERP - Cuba.

Consultante: Ing. Enrique Chaviano Gómez

Categoría Docente: Instructor.

Correo electrónico: echaviano@uci.cu

Síntesis del Asesor:

Profesor con 1 año de experiencia, graduado en la Universidad de Ciencias Informáticas, actualmente labora en una de las líneas del proyecto ERP - Cuba.

Asesor: Msc. Jorge Luis León González

Categoría Docente: Asistente.

Correo electrónico: yoshua@cfg.rimed.cu

Síntesis del Asesor:

Profesor de 5 años de experiencia, del Centro de Estudio de Software Educativo de la Universidad de las Ciencias Pedagógicas: “Conrado Benítez García”, de Cienfuegos. Actualmente labora en el proyecto de software educativo de la colección Multisaber, Mined - UCI.

AGRADECIMIENTOS

A mis padres por estar ahí con su valioso e invaluable apoyo en cada minuto de mi vida

A mi hermana que aunque ausente, desde la hermana república de Venezuela me ha mandado su amor, cariño y dedicación.

A mi gran tutora, amiga y compañera de todos los tiempos Arismayda Dorado Risco, por haberme soportado durante todo este tiempo, y cumplir con el legado de José de la Luz y Caballero de ser un evangelio vivo como lo ha demostrado ser cada día en nuestra línea logística.

A mi consultante Enrique Chaviano Gómez por su entrega y disposición.

A mi asesor Jorge Luis León González por su total entrega.

A mi novia, por darme su apoyo incondicional en todo momento.

A mí siempre hermano Carlos Daniel Fonseca Cantillo.

A mis amigos:

- Arturo Iván Morfe Zaldívar
- Yariel Llanes Góngorá
- Leshter Delgado Pérez

A mis amigas:

- Diannelly Díaz Ramírez.
- Dora Caridad Blanco Cala.
- Leskenia Acosta Londres.

A todos mis compañeros de cuarto.

De forma general, a todas aquellas personas que de una forma u otra han contribuido al desarrollo exitoso del presente trabajo de diploma.

DEDICATORIA

- A mis padres por nunca haberme fallado.
- A mi hermana por su apoyo incondicional, amor y cariño.
- A mi tutora Arismayda por su dedicación, apoyo y comprensión.
- A la Revolución y a nuestro invaluable Comandante en Jefe.

RESUMEN

La situación general de los procesos de facturación de las entidades empresariales y unidades presupuestadas en Cuba se encuentra afectada por la existencia de sistemas informáticos que no cumplen con las condiciones de las tecnologías de punta, aunque constituye una tarea de gran prioridad la informatización de cada una de sus empresas.

Velar por la seguridad de los recursos de las entidades, tener un control estricto de las salidas de cada producto y/o servicio, así como avanzar en exactitud de la información financiera y contable, son acciones que han llevado a la necesidad de mejorar los procesos de gestión en cada institución, en cuanto a sus sistemas de facturación, utilizando plataformas confiables y eficientes.

Por esta razón se decide implementar los procesos de configuración del Subsistema de Facturación del Sistema Integral de Gestión Cedrux para las entidades empresariales y unidades presupuestadas del país, en correspondencia con las nuevas concepciones de informatización.

PALABRAS CLAVES

Configuración, Facturación, Producto, Servicio.

ÍNDICE DE CONTENIDO.

INTRODUCCIÓN. 1
CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA. 5
INTRODUCCIÓN. 5
1.1. SISTEMAS UTILIZADOS PARA LA REALIZACIÓN DEL PROCESO DE FACTURACIÓN. 5
1.1.1. Principales sistemas para la realización del proceso de facturación a nivel internacional. 6
1.1.2. Principales sistemas para la realización del proceso de facturación en Cuba. .. 9
1.1.3. Valoración de sistemas estudiados. 10
1.2. LA ARQUITECTURA DEL SOFTWARE. 11
1.3. HERRAMIENTAS, TECNOLOGÍAS Y LENGUAJES. 15
CONCLUSIONES PARCIALES. 27

CAPÍTULO II. DESCRIPCIÓN Y ANALISIS DE LA SOLUCIÓN PROPUESTA. 28
INTRODUCCIÓN 28
2.1. VALORACIÓN CRÍTICA DE LOS ARTEFACTOS PROPUESTOS POR LOS ANALISTAS. 28
2.2. INFORMACIÓN QUE SE MANEJA. 31
2.3. PROCESOS OBJETO DE AUTOMATIZACIÓN. 32
2.3.1. Configuración de facturas. 32
2.3.2. Gestionar Matriz de precios. 32
2.3.3. Gestionar Matriz de tarifas. 33
2.3.4. Gestionar Nomenclador de servicios. 33
2.4. PROPUESTA DE SOLUCIÓN. 33
2.5. ESTÁNDARES DE CODIFICACIÓN. 35
2.6. DESCRIPCIÓN DE CLASES UTILIZADAS. 37
2.7. EXPLICACIÓN DE UN ALGORITMO NO TRIVIAL UTILIZADO EN LA IMPLEMENTACIÓN. 51
2.8. PROPUESTA DE IMPLEMENTACIÓN. 52

2.9. DESCRIPCIÓN DE LA IMPLEMENTACIÓN. 52
2.9.1. Integración entre componentes. 52

2.10. ESTRATEGIA PARA LA CAPTURA DE ERRORES. 56
CONCLUSIONES PARCIALES. 56

CAPÍTULO III. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA. 58
INTRODUCCIÓN. 58
3.1. Métricas. 58

3.2.1. Aplicación de las métricas: 60

3.2. PRUEBAS DE SOFTWARE. 64
3.2.1. Objetivos. 65

3.2.2. Alcance. 65

3.3. DESCRIPCIÓN DE LOS TEST DE UNIDAD. 66
3.3.1. Prueba de Caja Blanca o Estructurales. 66

3.4. APLICACIÓN DE PRUEBAS DE CAJA BLANCA. 71
CONCLUSIONES PARCIALES. 75
CONCLUSIONES GENERALES. 76
RECOMENDACIONES. 77
BIBLIOGRAFÍA. .....
ANEXOS. .....
GLOSARIO DE TÉRMINOS. .....

ÍNDICE DE FIGURAS.

Figura 1. Arquitectura Cliente Servidor. 13

Figura 2. Mejoras que provee el uso de HDS 16

Figura 3. Algoritmo correspondiente a la función devolverServicios. 51

Figura 4. Diagrama de integración de componentes (Vista externa). 54

Figura 5. Diagrama de componentes. 55

Figura 6. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos. 60

Figura 7. Representación de por ciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos. 60

Figura 8. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad. 61

Figura 9. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación. 61

Figura 10. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización de Implementación. 61

Figura 11. Representación en por ciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos. 62

Figura 12. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento. 62

Figura 13. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento. 63

Figura 14. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas. 63

Figura 15. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización 63

Figura 16. Notación de grafos de flujo para las instrucciones: Secuenciales, If, While..... 67

Figura 17. Notación de grafos de flujo para la instrucción Case..... 67

Figura 18. Componentes de los grafos de flujo..... 68

Figura 19. Representación de pruebas de Caja Blanca. 70

Figura 20. Representación del algoritmo esta relación ($idcatprecio). 71

Figura 21. Representación del flujo asociado al algoritmo esta_relacion($idcatprecio). ... 72

ÍNDICE DE TABLAS.

Tabla 1. Descripción de operaciones de la clase GestnomservicioController..... 38

Tabla 2. Descripción de las operaciones de la clase MatriztarifaController. 39

Tabla 3. Descripción de las operaciones de la clase DesglosepreciosController. 40

Tabla 4. Descripción de las operaciones de la clase GestcfgfacturaController..... 41

Tabla 5. Descripción de las operaciones de la clase Gest nomservicioModel. 43

Tabla 6. Descripción de las operaciones de la clase MatriztarifaModel..... 44

Tabla 7. Descripción de las operaciones de la clase GestcfgfacturaModel. 45

Tabla 8. Descripción de las operaciones de la clase NomServicio. 48

Tabla 9. Descripción de la métrica TOC..... 59

Tabla 10. Descripción de la métrica RC. 60

Tabla 11. Caminos básicos del flujo..... 73

INTRODUCCIÓN.

Es evidente el creciente aumento de la informatización en la sociedad a medida que transcurre el tiempo. Hoy en día se trabaja en aras de informatizar todos los procesos de la vida cotidiana y cada vez son mejores los resultados que se observan, desde la implementación de un simple software de cálculo hasta los más recientes sistemas informatizados que son capaces de realizar eficientemente el trabajo que antes al hombre le llevaba más tiempo, trabajo y con menos calidad en los resultados. Un ejemplo de esto lo vemos en nuestro diario vivir como un simple cajero automático ejerce las funciones que años antes, las solían realizar un conjunto numeroso de personas en un mayor tiempo y con mayor probabilidad de errores operacionales.

Los sistemas de software se especializan en distintos y determinados negocios. Actualmente se está desarrollando un sistema que consiste en la elaboración de un software empresarial que integra todas las áreas de organización económica contable de las entidades que se vinculan al logro de su objetivo esencial: el estricto control económico empresarial de cada actividad que se realiza en la empresa con sus activos (productos y servicios), Este sistema que se pretende desarrollar es conocido como sistema de Planeación de Recursos Empresariales (*Enterprise Resource Planning*, ERP). El mismo garantiza la centralización de la información de una empresa. Implementar un sistema ERP es un proceso largo, costoso y complejo, que requiere de gran cantidad de desarrolladores.

Por estas razones es aconsejable dividir su desarrollo en módulos que representen las distintas áreas de la empresa, de forma tal que se viabilice su proceso de desarrollo. Uno de ellos comprende la gestión de la Configuración y Nomencladores para el subsistema que abarca la emisión de documentos de salida (las facturas).

La facturación permite controlar los medios que se le han dado salida en una organización por los distintos conceptos de ventas que existan en la misma. La configuración de este importante proceso, es la encargada de crear e inicializar asignando los datos necesarios para que la empresa realice facturas eficientes a la hora de ofertar los activos a las entidades clientes, con el fin de hacer más rentable y económicas sus ventas, garantizando en gran medida el éxito económico de la organización.

La configuración del proceso planteado, en la mayoría de los casos, se realiza de forma manual, y en otros, no cumple con todos los requisitos específicos de cada entidad del país por lo que se hace necesario la gestión de un sistema que lo realice eficientemente.

Hay que destacar que la gestión de las facturas constituye una tarea importante en cualquier empresa. Un adecuado control de las mismas permite a cualquier entidad tener una adecuada organización y acceso a cada documento de salida efectuado, logrando una mayor eficacia evitando errores contables y económicos, facilitando de este modo una mayor rentabilidad económica.

La Oficina Nacional de Informatización, ha reconocido que la situación general en Cuba de las entidades de desarrollo de aplicaciones informáticas y de aquellas en explotación, que abarcan la actividad presupuestada, empresarial productiva o de servicios, esta caracterizada por:

Una considerable presencia de sistemas informáticos que están desarrollados sobre plataformas casi obsoletas, con poco o ningún criterio de seguridad y auditoría (técnico y funcionalmente).

Productos que se caracterizan por abordar solamente partes del problema de la gestión de la empresa o la unidad presupuestada y no soportan mecanismos estándares de integración con otras aplicaciones.

La mayoría fueron desarrollados para un ambiente multiusuario. Casi ninguno bajo conceptos de informática multicapa y distribuida en la red. Lo más general es desarrollos sobre arquitectura Cliente -Servidor de base de datos.

Los sistemas más potentes que actualmente están en explotación en Cuba son extranjeros y no abarcan todas las operaciones de gestión por presentar una incompleta implementación en la entidad o porque no lo soportan.

Analizando lo expuesto anteriormente se establece como **problema científico** el siguiente: *¿Cómo obtener un producto funcional a partir de los requerimientos identificados para la gestión de los procesos de Configuración del Subsistema Facturación para las entidades empresariales y unidades presupuestadas del país?*

Teniendo en cuenta el problema planteado se define como **objeto de estudio**: *Los procesos de facturación.*

Por lo que se especifica el siguiente **campo de acción**: *Los procesos de configuración para las facturas en las entidades empresariales y unidades presupuestadas del país.*

Dadas estas condiciones se plantea como **objetivo general**: *Implementar los procesos de configuración del Subsistema de Facturación para las entidades empresariales y unidades presupuestadas del país.*

Se plantean además como **objetivos específicos**:

1. Analizar los procesos de configuración, los sistemas que existen actualmente para la facturación, así como las herramientas que se utilizarán para el desarrollo de la solución.
2. Implementar el módulo de configuración del Subsistema Facturación.
3. Validar el resultado obtenido.

En la presente investigación se asume como **idea a defender**: *La implementación de los procesos de Configuración del Subsistema Facturación para las entidades empresariales y unidades presupuestadas del país contribuye a obtener un producto funcional para la gestión de dichos procesos.*

En la realización de la presente investigación se utilizaron varios métodos. Del **nivel teórico** se emplearon el **método histórico - lógico**, para conocer el desarrollo evolutivo y coherente de la metodología orientada a objetos, patrones de diseño, herramientas Case y sistemas ERP para el desarrollo de los artefactos que proponen los flujos estudiados.

Los **métodos inducción - deducción, análisis - síntesis e hipotético - deductivo** se utilizaron para resumir, sintetizar y procesar la información recopilada en el estudio realizado en la literatura cubana y extranjera.

Para representar gráficamente la información que se presenta se empleó la **modelación**.

En la determinación y solución del problema científico de investigación se utilizó del nivel empírico, la **tormenta de ideas**, al conocer los requisitos funcionales del software para los subsistemas de Configuración y Nomencladores, pertenecientes al Subsistema de Facturación.

Al determinar la factibilidad del software propuesto en la gestión de la información en el proceso de Facturación se empleó del **nivel matemático estadístico**, la **representación gráfica** y la **determinación de por ciento** para validar la solución propuesta mediante las métricas de calidad.

El **aporte práctico** de la investigación radica en la *contribución a la obtención de un eficiente proceso de facturación en las entidades empresariales y unidades presupuestadas del país* mediante la implementación de los subsistemas de Configuración y Nomencladores.

El documento cuenta con la siguiente estructura:

Capítulo I. Fundamentación Teórica.

En este capítulo se analizan los diferentes sistemas relacionados con el proceso de facturación, así como el modelo de desarrollo, las herramientas, tecnologías y lenguajes propuestos por el equipo de arquitectura del proyecto.

Capítulo II Descripción y análisis de la solución propuesta.

En este capítulo se ofrece un estudio de la solución propuesta por los analistas, a partir de la obtención de los requisitos. Se detalla uno de los algoritmos más complejos con el que cuenta la aplicación, explicando además la estructura de datos usada en el mismo. También se muestra el diagrama de componentes para un mejor entendimiento de la solución propuesta, así como la descripción de las principales clases que fueron definidas.

Capítulo III. Validación de la solución propuesta.

En este capítulo se muestran los resultados obtenidos para validar la solución propuesta, mediante la aplicación de las métricas de calidad, que les fueron aplicadas a las clases y las pruebas de caja blanca, hechas a fragmentos de código.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA.

INTRODUCCIÓN.

La rivalidad a que es sometido el ámbito empresarial internacionalmente requiere de realizar procesos y actividades de negocios que den lugar a ventajas competitivas de la entidad o compañía ante sus más fuertes rivales.

Con el desarrollo de las tecnologías de la información, entre otras técnicas de avanzadas con que hoy cuenta el mundo empresarial, las empresas necesitan de herramientas novedosas y eficaces que les proporcione un adecuado control y organización de su información, con el fin esencial de tomar mejores decisiones, en el momento necesario, con la rapidez que requieran sus procesos y estrategias de negocios. Los ERP representan una solución robusta para las entidades que buscan centralizar universalmente su información.

En todo el mundo se han desarrollado software ERP que integran todos los departamentos que gestionan la información referente en las entidades empresariales y unidades presupuestadas con subsistemas que implementan todas las funciones necesarias para el funcionamiento eficiente de la entidad, uno de estos subsistemas se encarga del Control de documentos de salida (facturas), cuyo estudio de los mismos ha servido para tener en cuenta las ventajas y defectos que presentan, en función de reunir la información necesaria para posteriormente proponer una solución que se adapte lo mejor posible a las necesidades del país.

1.1. SISTEMAS UTILIZADOS PARA LA REALIZACIÓN DEL PROCESO DE FACTURACIÓN.

El proceso de facturación consiste en “señalar los ingresos de una empresa por ventas o prestación de servicios durante un período de tiempo determinado”[1], deducidos los impuestos directos sobre las mismas (IVA, Párrafo fiscal).

La facturación constituye una de las principales vías para lograr el éxito en una empresa, pero no sólo depende en vender, sino también en cobrar la factura.

La factura es un documento que debe ser emitido con la mayor precisión y claridad posible. En ella se deben incluir todos los datos por los que se rige la ley y los que solicita

[1] http://es.mimi.hu/economia/facturacion.html

el cliente. Debe tenerse en cuenta que las dificultades en el proceso de facturación son uno de los factores que conllevan a la existencia de problemas a la hora de realizarse el cobro.

Con el amplio desarrollo que han tenido las Tecnologías de la Información y las Comunicaciones (TIC), es que en el proceso de facturación se introducen sistemas informáticos que permiten agilizar este proceso, al emitir automáticamente la factura.

Existe una variedad de aplicaciones informáticas que son utilizadas en la elaboración de facturas a nivel internacional y particularmente en Cuba.

En los siguientes apartados se exponen los principales software utilizados para la realización de facturas, así como sus características esenciales.

1.1.1. Principales sistemas para la realización del proceso de facturación a nivel internacional.

OpenBravo: Software desarrollado por la Universidad de Navarra en España, sistema de gestión empresarial integrado (ERP) líder, en software libre y entorno web, dirigido a pequeñas y medianas empresas.

Características:

1. Implementado en el lenguaje Java.
2. Aplicación completamente web que ha sido desarrollada siguiendo el modelo MVC (Model, View, Control).
3. Soporte para bases de datos PostgreSQL y Oracle.
4. Se ejecuta sobre Apache y Tomcat.

Presenta un subsistema que permite generar una factura con un importe igual al descuento resultante de aplicar el rappel a las facturas del tercero en un determinado periodo de tiempo. (1)

SAP: Software desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM. La corporación se ha desarrollado hasta convertirse en la quinta más

grande compañía mundial de software. Cuenta con el modulo Controlling (CO), que permite el control de los gastos generales, costes de producto, cuenta de resultados y centros de beneficio (Maestre, 2008).

Características:

1. Implementado en NET y WebSphere.
2. SAP también ofrece una nueva plataforma tecnológica denominada SAP NetWeaver, esta plataforma tecnológica convierte a SAP en un programa Web-enabled, lo que significa que estaría totalmente preparado para trabajar con él mediante la web.
3. Trabaja sobre el sistema operativo Windows.
4. Soporte para bases de datos Oracle.

Presenta un subsistema que permite registrar y procesar las facturas en *SAP Enterprise Buyer*. Si registra o procesa las facturas con el rol de proveedor o de prestatario de servicios, éstas deben estar autorizadas por un empleado interno responsable. (2)

Existen las opciones siguientes:

- Registro de facturas sin referencia a pedido
- Registro de facturas con referencia a pedido
- Visualización y procesamiento de facturas
- Envío de facturas mediante XML
- Recepción de factura XML

ETES (Total Enterprise Solution) Version 1.3.4.0: Concebido como un sistema integrado de gestión empresarial por lo que lleva implícito en su diseño las siguientes características:

- Integración.
- Acceso directo y remoto.
- Conectividad.
- Uso y protección de la base informativa única.
- Reglas de seguridad y autenticación.
- Concepción modular y compartimentación.
- Arquitectura de tres capas (Almacenamiento de Datos, Reglas del Negocio, Presentación).

Presenta un subsistema que permite obtener un control sobre las existencias de sus inventarios, conocer cuándo se esta vendiendo y cuándo debe pedir para no quedarse sin mercancía y SIN VENDER. Además se ahorra mucho costo financiero al no tener sobreinventario, ni haber mermado sus utilidades por productos obsoletos o por cambio de precios. (Suárez Quintana)

SEVEN-ERP: Sistema para gerenciar eficientemente las necesidades en las áreas Administrativas, Financieras, Comerciales, Manufactura y Recursos Humanos de las organizaciones.
Características:

- Tecnología cliente/servidor multinivel.
- Bases de datos relacionales.
- Intranet, Internet y procesamiento distribuido.
- Integra herramientas como Automatización de Procesos SEVEN Work Flow®, Procesamiento Electrónico de Documentos SEVEN Image®, Business Intelligence SEVEN-BI®, Comercio Electrónico SEVEN e-Commerce®, Integración SEVEN-EAI® y Administración de Relación con Clientes SEVEN-CRM®.

“Presenta un subsistema que permite la elaboración de los pedidos se pueden realizar a través de Internet, asignando permisos a cada uno de los clientes para que tengan acceso a la información necesaria para la elaboración del pedido, de esta manera se elabora la factura tomando como base el pedido elaborado por el cliente. Aumento o disminución de

la cartera a través de notas crédito o débito, manejo de anticipos con los clientes para ser cruzados con facturas. Dentro del módulo de cartera, permite el manejo de provisiones por tipo de producto, administración de intereses por mora, intereses corrientes y manejo de acuerdos de pago". (3)

1.1.2. Principales sistemas para la realización del proceso de facturación en Cuba.

Versat - Sarasola: Software que automatiza las actividades de planificación, control y análisis económico de cualquier tipo de entidad económica con el se abarca la administración, contabilidad, los medios de rotación, los activos fijos, las finanzas, cajas y costos. Esta implementado con modernas tecnologías y trabaja en red con alta seguridad. Puede acoplarse a otros sistemas para intercambiar información, es el primer sistema de contabilidad cubano certificado, en cuya evaluación participaron el Ministerio de Finanzas y Precios, consultorías internacionales y el organismo encargado de la seguridad informática. Es un sistema económico integrado.

Características:

1. Es una aplicación de escritorio.
2. Implementado en Delphi.
3. Trabaja sobre el sistema operativo Windows.
4. Soporte para bases de dato SQL Server 2000.

Presenta un subsistema con el cual se logra la facturación de todo tipo de categorías de inventarios, o sea, producciones terminadas, insumos, mercancías para la venta, además de los servicios. (4)

Actualmente lo utilizan alrededor de 200 entidades de varias provincias y en lo adelante lo introducirán más de dos mil 500 unidades presupuestadas del país, entre las que figuran organismos de la Administración Central del Estado, las direcciones municipales de finanzas, tesorerías, la ONAT y otros.

Rodas XXI Versión 3.0: Sistema integral económico administrativo creado por la empresa Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados (CITMATEL) que posibilita automatizar el funcionamiento de cualquier empresa o unidad

presupuestada. Desarrollado con los mas recientes adelantos en el diseño de interfaz de usuario. Cuenta actualmente con seis módulos: Finanzas, Contabilidad, Activos Fijos, Nóminas, Inventario y Facturación. Estos módulos pueden emplearse integrados en su totalidad, formando cualquier subconjunto entre ellos, o cada uno de forma independiente. Estos módulos del sistema están diseñados para trabajar en entorno de red e intercambiar información mediante correo electrónico o disquetes.

Presenta un subsistema que permite declarar siete tipos diferentes de clasificadores:

1. Productos.
2. Servicios.
3. Categorías.
4. Clientes.
5. Proveedores.
6. Gestor de venta.
7. Tipos de facturas.

Permite prefacturar, facturar y refacturar, se factura en doble moneda, en moneda nacional y en divisa, elabora opcionalmente componentes contables. (5)

SISCONT5: El sistema se aviene a las definiciones y conceptos del Ministerio de la Industria Básica aunque por las acciones contables financieras que permite puede ser utilizado en otras entidades nacionales.
Presenta un subsistema que permite trabajar con documentos de salida, los cuales actualizarán en forma automática el stock .El sistema permite imprimir facturas, boletas y guías de remisión estructurándolos de acuerdo a sus formatos. (6)
Puede ser explotado en régimen monousuario y multiusuario. Se define para monoentidad y multientidad, para esta última existe el control de su acceso para las entidades en un mismo equipo de cómputo como servidor.

1.1.3. Valoración de sistemas estudiados.

Los sistemas estudiados en su mayoría realizan procesos de facturación, pero no todos son libres o las licencias son poco permisivas y casi inalcanzables por el país.

La mayoría de los sistemas son aplicaciones de escritorio que requieren gran capacidad del hardware instalado para el procesamiento. Otro factor a tener en cuenta es que al ser aplicaciones de ese tipo, dependen del sistema operativo para poder funcionar.

Por las razones antes mencionadas se hace imprescindible realizar un sistema integral de gestión que realice los procesos de facturación, seleccionando las tecnologías, herramientas y lenguajes definidos por arquitectura y expuestos anteriormente, los cuales son los más adecuados a las necesidades del país y las políticas de software libre que se llevan a cabo.

Sería entonces muy factible trabajar sobre herramientas y lenguajes que no sean propietarios en función de una aplicación web para la gestión de los procesos de Configuración de la facturación, siendo el trabajo con PHP como lenguaje, Postgree SQL como gestor de Base de Datos, lo más indicado para desarrollar cualquier aplicación adaptable a las necesidades expuestas anteriormente.

1.2. LA ARQUITECTURA DEL SOFTWARE.

Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, también denominada *Arquitectura lógica*, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas. Las que a continuación se describen fueron seleccionadas por el equipo de arquitectura del proyecto.

Arquitectura Modelo - Vista - Controlador.

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de los conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente.

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y en la vista probablemente.

Arquitectura Cliente - Servidor.

Esta arquitectura consiste básicamente en un programa cliente que realiza peticiones a otro programa - servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los *sistemas multicapa* en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

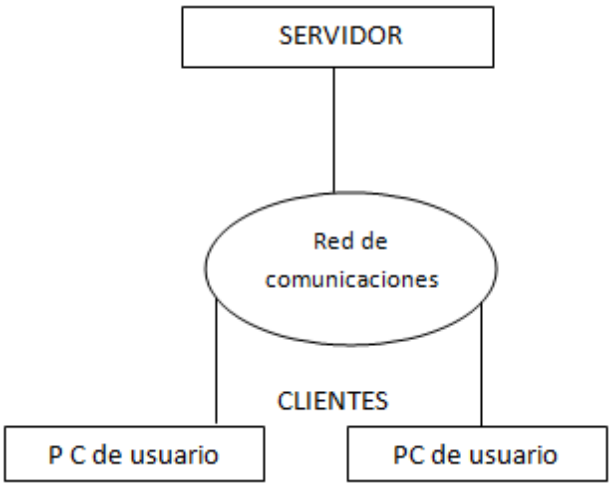


Figura 1. Arquitectura Cliente Servidor.

Ventajas

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P).

- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).

- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.

- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

Desventajas

- La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.

- El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está *caído*, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.

- El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes. Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el costo.

- El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

Arquitectura basada en capas:

Se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada. (7)

Características:

- Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.

- Las capas de una aplicación pueden residir en la misma maquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
- Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces muy bien definidas.
- Este modelo ha sido descrito como una “pirámide invertida de re-uso” donde cada capa agrega responsabilidad y abstracción a la capa directamente sobre ella.

1.3. HERRAMIENTAS, TECNOLOGÍAS Y LENGUAJES.

Se exponen las ventajas y desventajas de las herramientas definidas por el equipo de arquitectura del proyecto para el desarrollo del software.

Herramientas.

Actualmente se considera a las Herramientas de Desarrollo de Software (HDS) como herramientas basadas en computadoras que asisten el proceso de ciclo de vida de software, consolidadas en la literatura en la forma de Ingeniería de software asistida por computadora (CASE, por sus siglas en inglés). Esto es, software que se utiliza para ayudar a las actividades del proceso de software o software que es utilizado para diseñar y para implementar otro software.

Permiten automatizar acciones bien definidas, reduciendo también la carga cognitiva del ingeniero, quien requiere libertad para concentrarse en los aspectos creativos del proceso. Este soporte se traduce en mejoras a la calidad y la productividad en el diseño y desarrollo. Las HDS automatizan metodologías de software y desarrollo de sistemas y se vinculan con los diferentes conceptos involucrados en el desarrollo.

El soporte que brindan las HDS al proceso de desarrollo proporciona importantes ventajas para el equipo de trabajo de IS (Ver figura 1). Estas mejoras se sintetizan en:

a) Apoyan a las metodologías y métodos, integrando actividades y propiciando visión de continuidad entre fases metodológicas.

b) Mejoran la comunicación entre los actores involucrados, facilitándoles compartir su trabajo y desempeñarlo de forma dinámica e iterativa.

c) Establecen métodos efectivos para almacenar y utilizar los datos, lo que permite organizar y correlacionar componentes, para guardarlos a través de un repositorio.

d) Agregan eficiencia al mantenimiento, ya que los programas son construidos sobre las mismas estructuras y estándares, facilitando la adherencia a la disciplina de diseño y facilitan también la conversión automática de programas a versiones más recientes de lenguajes de programación.

e) Automatizan porciones del análisis y diseño engorrosos y propensos a error, con influencia sobre la generación de código, las pruebas y el control. Resalta la consideración de que los beneficios potenciales sólo pueden ser alcanzados si las HDS son utilizadas de forma correcta.

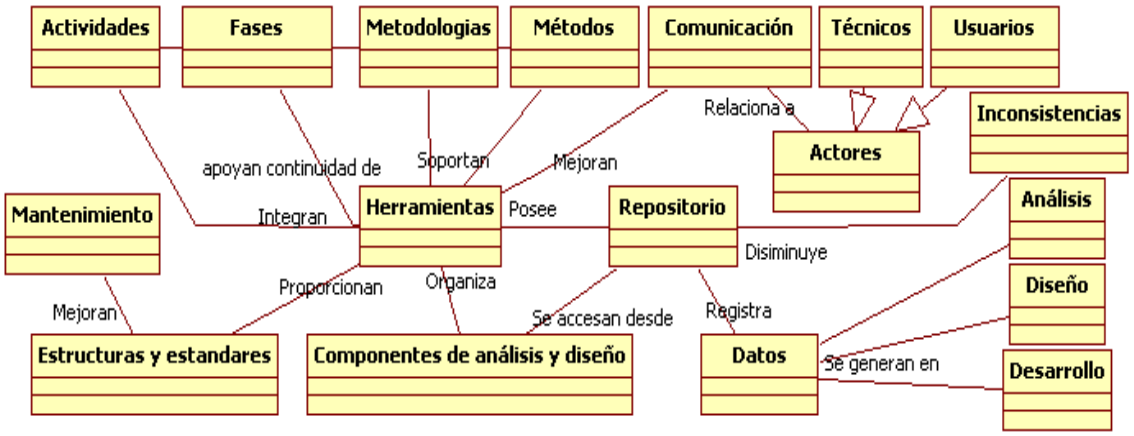


Figura 2. Mejoras que provee el uso de HDS

Servidor web con el que correrá la aplicación a elaborar. Apache http server ver 2.2.

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto

se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, *a patchy server* (un servidor "parcheado").

Presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, aunque ha sufrido un descenso en su cuota de mercado en los últimos años.

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Entorno de desarrollo integrado.

Entorno de desarrollo integrado (IDE), en inglés, *Integrated Development Environment (IDE),* es un programa compuesto por un conjunto de herramientas para un programador.

Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, puede utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se

permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante pluggins se le puede añadir soporte de lenguajes adicionales.

Zend Studio para Eclipse Ver 6.0: Combina la probada tecnología y desarrolladores de PHP de Eclipse Tools (PDT) proyecto para crear el más poderoso del mundo IDE para el desarrollo de ricas aplicaciones Web. Zend Studio para Eclipse está diseñado para profesionales que necesitan los desarrolladores de PHP para apoyar el ciclo de vida de toda la aplicación PHP, y quieren tomar ventaja de la sofisticación y la extensibilidad del marco de Eclipse y de los ecosistemas (Almada, 2008). Esta herramienta presenta entre otras las siguientes características:

- Inserción automática de paréntesis y corchetes de cierre.
- Soporte para navegación en bases de datos y ejecución de consultas SQL. Código de Cobertura.
- PHP Unit pruebas de apoyo.
- Mejora con PHP Editor avanzado de formato, las nuevas listas de tareas y problemas de vista.
- Mejora de soporte JavaScript.
- Mejora de apoyo, incluyendo HTML, Código de Folding, Drag & Drop y componentes más.
- Mejora de control de versiones con el apoyo de historia local.
- Mejora de depuración y perfilado con Sendero Cartografía.
- Mejora de apoyo con Zend marco nuevo marco del proyecto, plantillas de código, opinión y más MVC.
- Acceso al ecosistema de plug-ins de Eclipse.

- Apoyar el desarrollo de múltiples idiomas.
- Zend Studio 5.5 Herramientas de Migración.
- Mecanismo de actualización automática.

Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.

Frameworks.

Estructura de soporte definida, mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Zend Frameworks Ver 1.6: Framework de código abierto para desarrollar aplicaciones y servicios web con PHP5. ZF es una implementación que usa código 100% orientado a objetos. La estructura de los componentes de ZF es algo único; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura débilmente acoplada permite a los desarrolladores utilizar los componentes por separado. A menudo se refiere a este tipo de diseño como "use-at-will" (uso a voluntad). Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, etc., y que esta información se almacene en archivos, en memoria, en base de datos, etc.

- Simplifica la gestión de archivos de configuración.
- Proporciona los componentes que forma la infraestructura del patrón MVC.

Gestor de base de datos.

Los sistemas de gestión de base de datos (SGBD); (en inglés: DataBase Management System, abreviado DBMS) es un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

PostgreSQL Ver 8.2: Postgres SQL es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que dicha herramienta es multiplataforma. Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

Herramientas de base de datos.

PgAdmin III: Aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.

pgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL para mayor seguridad.

SQLmanager 2007: Aplicación de alto desempeño para la administración y desarrollo de PostgreSQL Database Server. El programa trabaja con cualquier versión de PostgreSQL y soporta todas las últimas características de PostgreSQL, incluyendo espacios de tablas,

nombres de argumentos en funciones y más. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo. La versión Lite incluye las herramientas básicas de mantenimiento y administración (Traduce, 2008).

Características:

- Soporte completo para PostgreSQL hasta la versión 8.3
- Administración y navegación rápida de bases de datos.
- Administración fácil de todos los objetos PostgreSQL.
- Administración efectiva de seguridad.
- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento.
- Interfaz atractiva.

Navegadores web.

Un navegador, navegador red o navegador web (del inglés, *web browser*) es un programa que permite visualizar la información que contiene una página web (ya esté esta alojada en un servidor dentro de la World Wide Web o en uno local).

El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

Mozilla Firefox Ver 3.0: Es el nuevo e innovador navegador open source del que todo el mundo está hablando. Firefox ha sido creado por el proyecto Mozilla, un esfuerzo open source sin ánimo de lucro que incluye a miles de voluntarios alrededor del mundo. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. El apoyo organizativo del proyecto Mozilla es proporcionado por Mozilla Foundation (en los Estados Unidos de América), Mozilla Europe y Mozilla Japón (Mozilla, 2009).

¿Por qué Firefox y no Internet Explorer?

Por nombrar algunas de las posibilidades que ofrece Firefox y que no ofrece IE, están:

- Es Software libre.
- En Firefox no existen la cantidad de bugs que posee el catastrófico IE, inmediatamente se encuentra un bug en el producto es notificado al Proyecto Mozilla para que sea reparado el problema.
- Navegación por tabs: Esta es una de las principales características que tiene Firefox.
- También existen excelentes extensiones de fácil instalación, estos mejoran la usabilidad y el aspecto del navegador, cosa que no se logra en IE el cual siempre permanece con los mismos colores.

Herramientas para el control de versiones.

Una versión, revisión o edición de un producto, es el estado en el se encuentra en un momento dado en su desarrollo o modificación. Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar un mecanismo de almacenaje de los elementos que deba gestionar y un registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente brindando la posibilidad de volver o extraer un estado anterior del producto) entre otros aspectos.

Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial.

Subversion Ver 1.4.5: Software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno

un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

TortoiseSVN Ver 1.3.0: Es un cliente gratuito de código abierto para el sistema de control de versiones Subversión. Maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que puedan recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo, cómo cambiaron sus datos, y quién hizo el cambio. Esta es la razón por la que mucha gente piensa que Subversion, y los sistemas de control de versiones en general, son una especie de “máquinas del tiempo”.

Características

1. Integración con el shell de windows.
2. Puede ser usado sin un entorno de desarrollo.
3. Pequeñas imágenes decoran los íconos de los archivos mostrando qué archivos o directorios necesitan ser enviados al repositorio.
4. Disponible en 28 idiomas diferentes.
5. Maneja el mostrar la diferencia de documentos de Office tales como los creados con Microsoft Word.

Lenguajes para el modelado.

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos.

Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo.

Lenguaje Unificado de Modelado (Unified Modeling Language, UML)

UML es un lenguaje estándar para escribir planos de software. UML puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software.

Es apropiado para modelar desde sistemas de información en empresas hasta aplicaciones distribuidas basadas en la Web, e incluso para sistemas empotrados de tiempo real muy exigentes. Es un lenguaje muy expresivo, que cubre todas las vistas necesarias para desarrollar y luego desplegar tales sistemas. Aunque sea expresivo no es difícil de aprender ni de utilizar. Aprender a aplicar UML de modo ‘eficaz comienza por crear un modelo conceptual del lenguaje, lo cual requiere aprender tres elementos principales: los bloques básicos de construcción de UML, las reglas que dictan cómo pueden combinarse esos bloques y algunos mecanismos comunes que se aplican a lo largo del lenguaje.

UML es sólo un lenguaje y por tanto es tan sólo una parte de un método de desarrollo de software. Es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

Según una frase muy conocida: "El 80% de los problemas se pueden resolver usando tan solo el 20% de UML".

Lenguajes de programación web.

Desde los inicios de Internet, fueron surgiendo diferentes necesidades por los usuarios a las cuales se les dio solución a través de lenguajes de programación Web estáticos. Con el paso del tiempo, con el desarrollo tecnológico, surgieron nuevas demandas que necesitaban para su solución de nuevos lenguajes de programación.

Esto dio lugar a los lenguajes de programación para la Web dinámicos, que permitieran interactuar con los usuarios y utilizaran sistemas de Bases de Datos. A continuación se ofrecen los más actuales lenguajes de programación para la Web.

HTML: Es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de

"etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un *script* (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (HtmlCastellano, 2009).

XML: Sigla en inglés de Extensible Markup Language (lenguaje de marcas ampliable), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable (Wikilibros, 2007).

JavaScript: Es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM. Javascript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página Web y en programas más grandes, orientados a objetos mucho más complejos. Con Javascript se puede crear diferentes efectos e interactuar con nuestros usuarios. Javascript es

soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros (territoriopc, 2001).

Lenguajes del lado del servidor.

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

Se les clasifica como lenguajes del lado del servidor a los lenguajes de programación en la tecnología cliente servidor que se ejecutan del lado del servidor y de los que los cuales los usuarios solo obtienen el beneficio del procesamiento de la información. A continuación se muestran los principales lenguajes del lado del servidor:

PHP Hypertext Pre-processor (Hipertexto Pre-processor) Ver 5.2: Es software libre, puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, soporta la mayoría de servidores Web de hoy en día y ofrece soporte para unos 20 gestores de bases de datos. Su característica de ser software libre trae como consecuencia que implique menos costes y servidores más baratos que otras alternativas. Es además muy rápido, contiene una biblioteca nativa de funciones sumamente amplia e incluida, no requiere definición de tipos de variables ni manejo detallado del bajo nivel, presenta mejoras de rendimiento, es un lenguaje multiplataforma que funciona en todas las plataformas que soporten Apache.

No es un lenguaje de marcas y la meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas.PHP también soporta el uso de otros servicios que usen protocolos como IMAP, SNMP, NNTP, POP3, HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos (web, 2001).

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas: Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader), analizar código XML. Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.

CONCLUSIONES PARCIALES.

Luego de haber realizado un análisis de los sistemas para los procesos de facturación existentes en los Sistemas de Recursos Empresariales internacionales y nacionales; las ventajas y desventajas de las herramientas para el desarrollo de los subsistemas definidas por el equipo de Arquitectura, se demuestra la no existencia de sistemas de Configuración y Nomencladores para la facturación de las entidades que hayan sido implementados con las políticas de software libre a que se acoge el país y además cumplan con las condiciones necesarias que exigen las entidades empresariales y unidades presupuestadas a escala nacional, por lo que se hace necesario el desarrollo de una nueva aplicación que satisfaga las situaciones expuestas anteriormente, por lo que se está en condiciones de describir y analizar la propuesta de solución para la implementación de los subsistemas.

CAPÍTULO II. DESCRIPCIÓN Y ANALISIS DE LA SOLUCIÓN PROPUESTA.

INTRODUCCIÓN

En este capítulo se abundan varios puntos claves en cuanto al desarrollo del subsistema. Se inicia realizando un profundo análisis del levantamiento de requerimientos propuesto por los analistas del sistema, mostrándose las principales ventajas y desventajas del mismo. Se dar un esbozo de cómo está aprovechada la arquitectura y las posibilidades que nos propician los frameworks y las librerías utilizadas en la programación de la aplicación, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. También se describe y examina un algoritmo no trivial de los que son considerados como más importantes dentro del sistema, incluyendo el análisis de su complejidad y por último se hace una descripción de clases y operaciones utilizadas.

2.1. VALORACIÓN CRÍTICA DE LOS ARTEFACTOS PROPUESTOS POR LOS ANALISTAS.

En un proceso de desarrollo de software es de vital importancia la descripción de los requisitos funcionales, los cuales son brindados por el analista de sistema y facilitan una mejor comprensión de los procesos a desarrollar, dando lugar al conocimiento en profundidad el problema en cuestión, facilitando una mejor identificación de las clases y funcionalidades que serán implementadas. La obtención del diseño propuesto por los analistas, resultó de gran importancia, pues posibilitó crear una entrada apropiada y un punto de partida para las actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.

La especificación de requerimientos es la base que permite verificar si se cumplieron o no los objetivos establecidos en el proyecto pues estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema y es contra lo que Un requerimiento según la IEEE Standard Glossary of Software Engineering Terminology se puede definir como una:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 o 2

Los requerimientos se pueden clasificar en funcionales y no funcionales.

Requerimientos funcionales: Son capacidades o condiciones que el sistema debe cumplir.

Los requerimientos no funcionales: Son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable

Existen múltiples categorías para clasificar a los requerimientos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta, aunque no limitan a la definición de otros.

⇒ Requerimientos de Software

⇒ Requerimientos de Hardware

⇒ Restricciones en el diseño y la implementación

⇒ Requerimientos de apariencia o interfaz externa

⇒ Requerimientos de Usabilidad

⇒ Requerimientos de Soporte

⇒ Requerimientos Legales

⇒ Requerimientos de confiabilidad

⇒ Requerimientos de interfaz Interna

⇒ Requerimientos de Seguridad

La seguridad puede ser tratada en tres aspectos diferentes:

- Confidencialidad
- Integridad
- Disponibilidad

Los requisitos funcionales que fueron identificados por los analistas y que posibilitaron la identificación de las clases y las funcionalidades a implementar son:

Componente Configuración del Subsistema Facturación

RF 1: Gestionar formato.

RF 1.1: Adicionar configuración de un formato a una entidad.

RF 1.2: Modificar configuración de un formato en una entidad.

RF 1.3: Eliminar una configuración del formato de una entidad.

RF 2: Gestionar Matriz de Precio

RF 2.1: Insertar un precio de un producto a una entidad, teniendo en cuenta los conceptos de cuentas de ventas que son asignadas a las entidades, y la dualidad monetaria en que se oferta dicho producto.

RF 2.2: Modificar un precio de un producto a una entidad, teniendo en cuenta los conceptos de cuentas de ventas que son asignadas a las entidades, y la dualidad monetaria en que se oferta dicho producto.

RF 2.3: Eliminar un precio de un producto a una entidad, teniendo en cuenta los conceptos de cuentas de ventas que son asignadas a las entidades, y las distintas dualidades monetarias en que se oferta dicho producto.

RF 3: Gestionar Matriz de Tarifa

RF 3.1: Insertar una tarifa a un servicio teniendo en cuenta los conceptos por cuentas de la entidad y la dualidad monetaria de dicho servicio.

RF 3.2: Modificar una tarifa a un servicio teniendo en cuenta los conceptos por cuentas de la entidad y la dualidad monetaria de dicho servicio.

RF 3.3 Eliminar una tarifa de un servicio teniendo en cuenta las restricciones anteriores

RF 4: Confeccionar la ayuda del subcomponente Configuración del subsistema de facturación.

RF 5: Gestionar Nomencladores de la configuración.

RF .5.1: Gestionar Nomenclador de Conceptos.

RF 5.1.1: Insertar Concepto a una entidad.

RF 5.1.2: Modificar Concepto de una entidad.

RF 5.1.3: Eliminar concepto de una entidad.

RF 5.2: Gestionar Nomenclador de Conceptos y Clientes.

RF 5.2.1: Asignar un concepto a un cliente para una entidad determinada.

RF 5.2.2: Eliminar un concepto a un cliente (Desasociarlo)

RF 5.3: Gestionar Nomenclador de Logos y Sellos.

RF 5.3.1: Insertar Logo o Sello.

RF 5.3.2: Modificar Logo o Sello.

RF 5.3.3: Eliminar Logo o Sello.

RF 5.4: Gestionar Nomenclador de Conceptos por Cuentas

RF 5.4.1: Asignar un una cuenta a un concepto.

RF 5.4.2: Eliminar una cuenta de un concepto. (Eliminar asignación)

2.2. INFORMACIÓN QUE SE MANEJA.

Conceptos: El sistema debe permitir gestionar todos los conceptos necesarios con que trabajará en aras de realizar las ventas a cada una de las entidades clientes.

Clientes – Conceptos: El sistema permite asignar a los clientes los conceptos por los que este se regirá para efectuar las compras de los productos y servicios a la entidad. (Ver Anexo II.)

Conceptos – Productos: El sistema debe permitir asignar a cada producto de la entidad los precios de ventas de acuerdo con las estructuras de precios pertenecientes a las cuentas que utiliza la entidad cliente, y la dualidad monetaria por la que se rige la empresa.

Conceptos – Servicios: El sistema debe permitir asignar a cada servicio que se brinda en la entidad los precios de ventas de acuerdo con las estructuras de precios pertenecientes a las cuentas que utiliza la entidad cliente, y la dualidad monetaria por la que se rige la empresa.

Configuración – Entidad: El sistema debe permitir asignar a cada una de las entidades clientes el formato por el cual facturarán las mismas y actualizar cada uno de sus indicadores como la longitud del formato, entre otros.

Entidad – Cuentas: El sistema debe permitir asignar a cada una de las entidades cliente cada una de las diferentes cuentas pertenecientes a las estructuras de precios y tarifas con que operarán las mismas a la hora de solicitar un servicio.

2.3. PROCESOS OBJETO DE AUTOMATIZACIÓN.

A continuación se explican los principales procesos a automatizar por su importancia para el funcionamiento de la configuración del subsistema y el buen funcionamiento de los procesos de facturación.

2.3.1. Configuración de facturas.

La configuración de las facturas para una entidad es unos de los pasos iniciales a seguir cuando se desea tener un funcionamiento y control eficiente de este proceso. La selección adecuada de cada uno de los datos del formato y el logotipo de las facturas para la entidad es fundamental a la hora de realizar la configuración, y conlleva a una calidad óptima al realizar el proceso de facturación.

2.3.2. Gestionar Matriz de precios.

La matriz de precios es otro de los pasos vitales en la configuración de la facturación en una entidad, aquí se definen los distintos precios que tendrán cada uno de los productos atendiendo a los conceptos de ventas existentes para cada una de las entidades clientes. Este proceso realizarlo de forma manual robaría más tiempo y el riesgo al error seria mayor debido a las operaciones matemáticas que hay que realizar a la hora de llevar el control de los precios de ventas de acuerdo con los diferentes indicadores existentes en la entidad.

2.3.3. Gestionar Matriz de tarifas.

La matriz de tarifas también es otro de los procesos de gran peso en la configuración de la facturación en la entidad, en este caso se definen las tarifas de los distintos servicios que se brindan en la institución por los distintos conceptos de ventas asociados a cada entidad cliente, al igual que la matriz de precio, realizar a mano este proceso demoraría un tiempo prudencial y el riesgo a cometer errores seria grande.

2.3.4. Gestionar Nomenclador de servicios.

La gestión de los servicios es de vital importancia en la configuración de la facturación en una entidad. Estos se asignan por diferentes niveles de acuerdo a las diferentes ramas a las que perteneces, por lo que se requiere de una estructura de datos arbólica para el almacenamiento de los datos. La no automatización de este nomenclador implicaría ejecutar el proceso de una manera engorrosa de forma manual, la cual conllevaría a un mayor tiempo y esfuerzo por parte de los trabajadores y una menor calidad del trabajo con grandes probabilidades a cometer errores.

2.4. PROPUESTA DE SOLUCIÓN.

Para informatizar los procesos que rigen la configuración en el subsistema facturación el sistema debe inicialmente permitir crear la configuración con la que las entidades realizarán el proceso de facturación, o sea, seleccionar el tipo de factura que utilizarán, el logotipo de la factura y las operaciones a realizar con las mismas, luego de esto se procedería a crear cada uno de los conceptos de ventas por los que se ofertarán los productos y servicios existentes en la entidad.

Cada concepto operará con los tipo de activos, ya sea productos, servicios o activos fijos tangibles, luego se va a asociar a cada uno de los clientes los diferentes conceptos de ventas creados anteriormente, así como también se van a crear los conceptos por cuentas de cada entidad, donde se le asignan a las entidades los tipos de cuentas que van a utilizar a la hora de realizar las diferentes operaciones.

Después de haber finalizado el proceso de inicialización de la configuración, se van a llenar las matrices de precios y tarifas, las cuales utilizan los datos anteriormente

inicializados, teniendo en cuenta aquí la dualidad monetaria en que se pueden ofertar los productos y servicios en las diferentes monedas con que opere la entidad, recalcar que el sistema solo debe mostrar la información referente a la Entidad que inició sesión en el sistema, o sea, mostrar solo los productos, servicios, activos fijos y conceptos asociados solo a los clientes de dicha entidad. Y finalmente agilizar el trabajo cumpliendo con todos los requisitos funcionales del sistema expuestos en el capítulo anterior.

Análisis de posibles implementaciones, componentes o módulos ya existentes.

En los subsistemas Inventario, Común, Útiles externos y Configuración general existen implementados varios componentes los cuales brindan servicios necesarios para la el funcionamiento de la solución propuesta. A continuación se detalla una breve descripción de los mismos.

Componente Producto.

El componente producto es el encargado de gestionar los productos almacenados en la entidad, ya sea los nomenclados como los nuevos que se van a adicionar a la empresa. Este componente brinda los servicios necesarios para operar sobre los productos que a través de la matriz de precios son relacionados los mismos dándole un precio a ofertar a una entidad determinada.

Nomenclador de Clientes.

El nomenclador de clientes tiene función vital en la facturación y en la configuración, ya que es a los mismos a quienes les ofertamos los precios y tarifas de los productos, servicios y activos fijos respectivamente, este nomenclador es el encargado de gestionar los clientes asociados a cada entidad y brinda los servicios necesarios para operar sobre cada cliente, obteniendo los datos de los mismos para asignarles los distintos conceptos de ventas por los que se acogerá la entidad para realizar las ventas de productos o ofertarle servicios a través de la matrices de precios y tarifas y el nomenclador de conceptos por clientes, en este ultimo a cada cliente se le asignan diferentes conceptos

de ventas por los cuales se determinará el precio de venta del producto, servicio o activo fijo, elaborándose el documento de salida (Factura).

Componente Formato:

La integración con el componente formato tiene una vital importancia para la configuración en el proceso de facturación, el mismo brinda servicios que reportan datos pertenecientes al formato por el que se rigen las entidades, que son necesarios para configurar las facturas que utilizará la entidad a la hora de efectuar las compras.

2.5. ESTÁNDARES DE CODIFICACIÓN.

Los estándares de codificación son modelos de programación que no están enfocados a la lógica del programa, sino a su estructura y aspecto físico para facilitar la lectura, comprensión y mantenimiento del código. Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino también tiene que ver con el orden y claridad del código escrito.

Siguiendo estas ideas, se definen 3 partes principales dentro de un estándar de programación:

- Convención de nomenclatura: Cómo nombrar variables, funciones, métodos, entre otros.
- Convenciones de legibilidad de código: Cómo identificar el código.
- Convenciones de documentación: Cómo establecer comentarios, archivos de ayuda, entre otros.

Los estándares de codificación desarrollan una mejor integración entre las líneas de producción y establecen patrones que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumenta su mantenibilidad a lo largo del tiempo.

Nomenclatura según el tipo de clases

1. Las clases controladoras después del nombre de dicha clase se le añade la palabra:”Controller”.

Ejemplo: GestionarMatrizPrecioController

2. Clases del modelo

   Business (Negocio)

   Las clases que se encuentran dentro de la carpeta Business después del nombre llevan la palabra: “Model”.

   Ejemplo: GestionarCertificacionLogotipoModel

3. Clases del dominio(domain)

   Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos.

   Ejemplo: NomConcepto

4. Generated (Dominio Base)

   Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: “Base” y seguido el nombre de la tabla en la Base de Datos.

   Ejemplo: BaseNomConcepto

5. Nomenclatura de las funciones

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

Ejemplo: insertarPrecio

- En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:”Action”

Ejemplo: insertarPrecioAction

Notación CamelCasing

La letra inicial del identificador no debe estar en mayúscula. Esta notación se utilizó para el nombre de las funciones y el nombre de los atributos. A continuación ejemplos:

insertarPrecio: Este nombre de método esta compuesto por 2 palabras, la primera todo en minúsculas y la segunda iniciando con letra mayúscula.

devuelveMonedas: Este nombre del atributo esta compuesto por 2 palabras, la primera todo en minúsculas y la segunda iniciando con letra mayúscula.

2.6. DESCRIPCIÓN DE CLASES UTILIZADAS.

A continuación se exponen las principales clases de los componentes a implementar, explicándose sus principales funcionalidades.

Clases controladoras.

Las clases controladoras tienen la responsabilidad de ejecutar una lógica específica en función de las acciones que se realizan en la aplicación. Se encargan de capturar los eventos, que se setéan en las vistas (views) y permiten la comunicación con las clases del negocio. (Modelo y Dominio)

Nombre: GestnomservicioController	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
$model	GestnomservicioModel
Para cada responsabilidad:	
Nombre:	**Descripción:**
traerpartesformatoAction	Devuelve las partes de un formato dado.
longitudFormatoAction	Devuelve la longitud de un formato.
obtenerCfgAction	Devuelve el formato para los servicios de una entidad dada.

longitudNivelAction($nivel)	Devuelve la longitud del nivel de un formato.
obtenerEspecialidadAction	Devuelve la especialidad de una estructura dada.
cargarnomencladorAction	Carga todos los servicios del nomenclador.
cargarformservicioAction	Carga los datos de un servicio dado su id.
cargaservarbolAction	Carga el árbol del nomenclador de servicio.
adicionarnomservtreeAction	Adiciona un nivel al árbol de servicio.
adicionarnomservgridAction	Adiciona un nuevo servicio.
modificarservtreeAction	Modifica un nivel al árbol de servicios.
modificarservgridAction	Modifica un servicio.
eliminarservtreeAction	Modifica un servicio.

Tabla 1. Descripción de operaciones de la clase GestnomservicioController

Nombre: MatriztarifaController	
Tipo de clase: Controladora	
Atributo: $model	**Tipo:** MatriztarifaModel
Para cada responsabilidad:	
Nombre: devuelveServiciosAction()	Devuelve los servicios para el formato establecido para la

	entidad cliente
devuelveMonedasAction()	Devuelve las monedas que aun no han sido asignadas al producto en cuestión.
devuelveTodasmonedasAction	Devuelve todas las monedas con las que opera la entidad cliente.
cargarGPDualidadtarifaAction	Devuelve las columnas del Grid Dinámico para la dualidad monetaria, atendiendo a los tipos de cuentas a utilizar en cada entidad.
cargarGPtarifaAction	Devuelve el desglose de tarifas para un servicio seleccionado.
insertarTarifaAction	Inserta o Modifica una Tarifa existente
eliminarTarifaAction	Elimina una tarifa de un servicio junto con todas sus dualidades monetarias.

Tabla 2. Descripción de las operaciones de la clase MatriztarifaController.

Nombre: DesglosepreciosController	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
$idestructura	Numérico.
$arrayidcuentas	Array.

Para cada responsabilidad:	
Nombre: cargarcuentasAction()	Carga las cuentas existentes en el nomenclador de cuentas, teniendo en cuenta la entidad logueada y el subsistema a trabajar.
cargarcategoriaprecioAction()	Carga las categorías de precios existentes de acuerdo con la entidad y la operación que se realiza.
cargarprecioentidadAction()	Devuelve las estructuras de precio con que trabaja la entidad de acuerdo a un tipo de operación.
definirprecioentidadAction()	Inserta una estructura de precio al nomenclador con la cual la entidad trabajará.
modificarprecioentidadAction()	Modifica un precio existente en la base de datos, de acuerdo a la entidad a que pertenece.
eliminarprecioentidadAction()	Elimina un precio existente de una entidad.
activaprecioentidadAction()	Activa un precio que anteriormente haya sido desactivado no haber podido eliminarlo completamente por estar vinculado a algún producto o servicio.

Tabla 3. Descripción de las operaciones de la clase DesglosepreciosController.

Nombre: GestcfgfacturaController	
Tipo de clase: Controladora	
Atributo:	**Tipo:**
$model	GestcfgfacturaModel
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargarcomboformatoAction	Devuelve todos los formatos posibles que la entidad logueada en el sistema puede utilizar.
cargargridoperacionesAction	Devuelve los tipos de operaciones con que opera la entidad logueada en el sistema..
cargarconfiguracionfacturaAction	Devuelve todos los elementos pertenecientes a la configuración asociados a la entidad actual.
salvarcfgalmacenAction	Guarda los datos referentes a la configuración de la entidad que fueron definidos en la vista.
eliminarcfgalmacenAction()	Elimina la configuración que tiene la entidad logueada en el sistema.
cargarcfgAction()	Devuelve la configuración actual de la entidad logueada en el sistema.

Tabla 4. Descripción de las operaciones de la clase GestcfgfacturaController.

Clases auxiliares.

Son las clases que almacenan poca o casi ninguna información del estado por si mismas, pero participan en la ejecución de tareas complejas. Van a responsabilizarse con la lógica específica del negocio, cálculos y también de las peticiones echas por la clases controladoras.

Nombre: GestnomservicioModel	
Tipo de clase: Controladora Auxiliar	
Para cada responsabilidad:	
Nombre:	**Descripción:**
adicionarServicio($idpadre, $objserv)	Inserta un servicio dado el nivel padre.
devolverTodosServicios()	Devuelve todos los servicios del nomenclador
modificarServicio($objServicio, $codserv, $codanterior, $codmod, $tipo)	Modifica un servicio dados los parámetros que se exijen
eliminarServicio($idserv)	Elimina el servicio de que lleva como id el $idserv pasado por parámetro
buscarServicios($codserv, $desc, $start, $limit)	Busca el o los servicios existentes en el nomenclador atendiendo a los criterios de búsquedas que se relacionan en los parámetros
buscarServicioDadoId($idservicio)	Devuelve los datos del servicio almacenado en el nomenclador y que lleva por 'idservicio' el

	parámetro $idservicio
obtenerServiciosNomenclador($start, $limit, $idprod, $idestructura, $idespecialidad, $codarbol, $longform, $codigo, $nropieza, $descripcion)	Devuelve los servicios existentes en el nomenclador de acuerdo con las exigencias descritas en los parámetros.
devolverArbolNomServicios($idnodo, $nivel, $longitud)	Devuelve el árbol de servicios del nomenclador de acuerdo al nodo de 'idnodo', 'nivel', y longitud pasadas por parámetros respectivamente
devolverNomServicios($longform, $codigo, $descripcion)	Devuelve los servicios del nomenclador dados que cumplan con los parámetros pasados.

Tabla 5. Descripción de las operaciones de la clase Gest nomservicioModel.

Nombre: PrecioentidadModel	
Tipo de clase: Controladora Auxiliar	
Atributo:	**Tipo:**
$idestructura	Numérico
Para cada responsabilidad:	
Nombre:	**Descripción:**
InsertarPrecioE	Inserte una estructura de precio a la entidad logueada.
modificarPrecioE($precio)	Modifica un precio de existente de la entidad logueada.

SePuedeEliminar($llave)	Es una función lógica que devuelve true si el precio de id igual a $llave no está siendo utilizado por ningún otro componente a los que se integra.
eliminarPrecioE($idprecioentidad)	Elimina el precio de $idprecioentidad asignado a la entidad logueada.
activaPrecioE($idprecioentidad, $value)	Activa el precio de id igual a $idprecioentidad que no pudo ser eliminado por estarse utilizando en el momento en que se hizo la eliminación, por lo que fue desactivado.

Tabla 6. Descripción de las operaciones de la clase MatriztarifaModel

Nombre: GestcfgfacturaModel	
Tipo de clase: Controladora Auxiliar	
Atributo:	**Tipo:**
$idestructura	Numérico
Para cada responsabilidad:	
Nombre:	**Descripción:**
modificarCfgAlmacen($objcfg)	Modifica la Configuración de los almacenes pasado por parámetros.
eliminarcfgalmacen($idestructura)	Elimina la configuración que tendrán las facturas de los

	almacenes de la entidad de $identidad pasado por parámetro.
obtenerCfgAlmacen($idestructura)	Devuelve los datos referentes a la configuración de las facturas de los almacenes pertenecientes a la entidad de $identidad pasado por el parámetro $idestructura.
activaPrecioE($idprecioentidad, $value)	Activa el precio de id igual a $idprecioentidad que no pudo ser eliminado por estarse utilizando en el momento en que se hizo la eliminación, por lo que fue desactivado.

Tabla 7. Descripción de las operaciones de la clase GestcfgfacturaModel.

Clases Entidad.

Modelan información que poseen una larga vida y por lo general representan a conceptos. Otra forma de nombrarlas es “clases dominios”, pues suelen tratar con abstracciones de entidades del mundo real. También van modelar la información del sistema y el comportamiento asociado a la misma.

Nombre: NomServicio	
Tipo de clase: Entidad	
Para cada responsabilidad:	
Nombre:	**Descripción:**
getCantNomServ($idserv, $idespecialidad, $codarbol, $longform,	Devuelve la cantidad de servicios

$codigo = '', $descripcion = '')	almacenados en la base de datos y que cumplen con las condiciones pasadas por parámetros
devolverNomServicio($idespecialidad,$logform,$idestructura= '', $codigo = '', $descripcion = '')	Devuelve los servicios que cumplan con los parámetros pasados.
obtenerServicioNomenclador($idprod, $idestructura='', $idespecialidad='', $codarbol, $longform, $codigo = '', $descripcion = '')	Devuelve los servicios que cumplan con los parámetros pasados.
devolverNomServDadoIdServ($idserv, $codigo='', $descripcion='')	Devuelve los servicios que cumplan con los parámetros pasados.
getDatosServicio($codigoesp,$codigocup, $codigo = '', $descripcion = '')	Devuelve los datos de un servicio que cumple con los parámetros pasados.
devolverArbolNomServPlan($idestructura='', $idespecialidad, $nivel, $longitud)	Devuelve el árbol de servicios del nomenclador perteneciente a la estructura de id pasada por

	parametro y correspondiente a la especialidad pasada también por de este modo.
obtenerTodosHijos($idnodo)	Obtiene los hijos del nodo perteneciente al árbol de servicios que lleva por id el pasado por parámetro.
devolverNomServicioDadoIdServ($idserv)	Devuelve el servicio de idservicio pasado por parámetro
devolverArbolNomencladorServicio($arrFiltroLftRgt,$nivel,$longitud)	Delvuelve el árbol de servicios del nomenclador dados los parámetros que se muestran en el método.
obtenerNomServ($codserv)	Obtiene el servicio de código $codserv
existeServMod($idnodo, $idestructura, $cod, $nivel)	Devuelve true si existe un servicio que cumpla con los parámetros pasados, devuelve false en caso contrario.

buscar($idserv) $codarbol, $longform, $codigo = '', $descripcion = '')	Obtiene los datos del servicio de idservicio $idserv en caso de que exista en la BD.

Tabla 8. Descripción de las operaciones de la clase NomServicio.

Nombre: DatMatriztarifa	
Tipo de clase: Entidad	
Responsabilidades	
buscar($idconcepto)	Busca en la tabla DatMatriztarifa los datos referentes a las tarifas de idconcepto igual el pasado por parámetro.
buscarS($idserv, $idconcepto)	Busca los datos referentes a la tarifa de idservicio = $idserv e idconcepto = $idconcepto
buscarT($idserv, $idconcepto, $idest)	Devuelve la tarifa que tiene por idservicio $idserv, idconcetpo $idconcepto e idestructura - $idestructura
getTodos()	Devuelve todas las tarifas almacenadas en la tabla dat_tarifa
eliminarTarifa($idmt)	Elimina la tarifa de idmatriztarifa igual a $idmt almacenada en la tabla datmatriztarifa

getPorLimite($limite = 10,$inicio = 0)	Devuelve as tarifas almacenadas en la tabla dat_tarifa con limte = $limit e inicio = $start

Tabla 9. Descripción de las operaciones de la clase DatMatriztarifa

Nombre: DatTarifa	
Tipo de clase: Entidad	
Buscar($idtarifa)	Busca un indicador de tarifa de la tabla dat_tarifa de idtarifa igual a $idtarifa
buscar_Dato($iddualidadtarifa, $idprecioentidad)	Busca los datos referentes al indicador de tarifa almacenado en la tabla dat_tarifa con idmatriztarifadualmonetaria = $iddualidadtarifa e idprecioentidad = $idprecioentidad
buscar_Concepto_Operacion($idprecioentidad)	Busca el concepto de operación que le fue asignado al indicador de idprecioentidad igual a $idprecioentidad.
eliminar_Campos($iddualidadtarifa)	Elimina todas las tuplas almacenadas en la tabla dat_tarifa referentes a los indicadores de idmatriztarifadualidadmonetaria = $iddualidadtarifa
getPorLimite($limite = 10,$inicio = 0)	Devuelve as tarifas almacenadas en la tabla dat_tarifa con limte =

	$limit e inicio = $start

Tabla 10. Descripción de las operaciones de la clase DatTarifa

Nombre: DatPrecioentidad	
Tipo de clase: Entidad	
buscarDescripcion($idestructura, $operacion)	Busca la distintas descripciones de los indicadores de las cuentas que utiliza la entidad de id $idestructura para la operación $operación
buscarIDPrecioEntidad($idestructura,$idcatprecio)	Busca el idprecioentidad correspondiente al indicador de idestructura $idestructura y operación $operacion
buscarDatosCampos($idestructura, $op)	Devuelve los datos correspondientes a los indicadores pertenecientes a la estructura de idestructura $idestructura y operación = $op
buscarCP($idcategoriaprecio,$idestructura)	Devuelve el idcategoriaprecio, idprecioentidad, idcuenta del indicador de idcategoriaprecio = $idcategoriaprecio y perteneciente a la entidad de idestructura = $idestructura
getPorLimite($limite = 10,$inicio = 0)	Devuelve los datos de los indicadores almacenados en la tabla dat_tarifa con limte = $limit

	e inicio = $start

Tabla 11. Descripción de las operaciones de la clase DatPrecioentidad.

2.7. EXPLICACIÓN DE UN ALGORITMO NO TRIVIAL UTILIZADO EN LA IMPLEMENTACIÓN.

A continuación se presenta uno de los algoritmos no triviales y con mayor grado de complejidad utilizado en la implementación del componente Nomenclador de Servicios, perteneciente al grupo de Nomencladores del subsistema, como se explicó en epígrafes anteriores, para el componente mencionado se hizo necesario el uso de la estructura de dato no lineal "ÁRBOL", y como es lógico, cuando se trabaja con estructuras de datos de este tipo, se hace imprescindible el uso de la recursividad para lograr que el funcionamiento sea eficiente y más óptimo.

El algoritmo correspondiente a la función devolverServicios, que se presenta en la figura 3, es una muestra de lo explicado anteriormente, el cual tiene una cuota de complejidad de O(N LOG N) para el peor de los casos.

```
public function devolverServicios( $start, $limit, $idserv, $codarbol, $longform, $codigo, $descripcion ){

    if ($codigo || $descripcion)
        $servicios = NomServicio::devolverServicios( $idserv, $codarbol, $longform, $codigo, $descripcion );
     else
        $servicios = NomServicio::devolverServicios( $idserv, $codarbol, $longform );

    $cantNomServ = count($servicios);
    $resNomServ = array_slice($servicios,$start,$limit);

    $servicioArr = array ();
    $nro = $start + 1;
    if ($servicios) {
        foreach ( $servicios as $index => $servicio ) {
            $servicioArr [$index] ['nro'] = $nro++;
            $servicioArr [$index] ['idserv'] = $servicio ['idserv'];
            $servicioArr [$index] ['codigoservicio'] = $servicio ['codigoservicio'];
            $servicioArr [$index] ['descripcion'] = $servicio ['descripcion'];
            $servicioArr [$index] ['um'] = $servicio ['um'];
            $servicioArr [$index] ['nivel'] = $servicio ['nivel'];
        }
    }
    $result = array ('cantfilas' => $cantNomServ, 'datos' => $servicioArr );
    return $result;
}
```

Figura 3. Algoritmo correspondiente a la función devolverServicios.

2.8. PROPUESTA DE IMPLEMENTACIÓN.

Luego de analizar lo anteriormente expuesto, se plantea que es factible implementar una aplicación Web, basada en el uso del estilo arquitectónico Modelo – Vista – Controlador, para la creación de la misma se utilizará el lenguaje de programación PHP y como soporte para los datos el Sistema Gestor de Bases de Datos PostgreSQL y para realizar las consultas necesarias en el dominio, se propone la utilización del Framework PHP DOCTRINE.

Se propone utilizar además, para hacer menos engorroso el trabajo, el Zend Framework, desarrollado en PHP bajo el estilo arquitectónico MVC, muy utilizado en la actualidad. Dicha aplicación, automatiza los procesos que integran la configuración y los nomencladores que intervienen en el proceso de facturación, teniendo en cuenta los procesos de facturación en las entidades cubanas y la documentación asociada a su desarrollo servirá de base para futuras implementaciones, mantenimientos o mejoras al software desarrollado con este mismo propósito.

2.9. DESCRIPCIÓN DE LA IMPLEMENTACIÓN.

A continuación se dará una breve explicación acerca de cómo se realiza la integración en los subsistemas a implementas y se exponen los diagramas de integración interna y externa respectivamente.

2.9.1. Integración entre componentes.

La aplicación va a estar definida por tres capas: capa de Presentación (view), Negocio (controller) y Acceso a Datos (models), esta arquitectura posibilita un trabajo seguro, rápido y eficiente. La integración vertical o llamada arquitectura en 3 capas consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura.

Consta de cuatro nodos de integración, el que encontramos entre la vista y el controlador, el que está entre la el controlador y el modelo, el que vincula el modelo con el framework doctrine y el que se encuentra entre el doctrine y la base de datos. Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa.

La comunicación entre diferentes módulos y componentes se realiza mediantes llamadas a la inversión de control (IOC). La IOC especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Cada componente tiene su registro de los datos de los módulos en un fichero xml que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IOC y registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. La base de datos es accedida de forma directa mediante controladoras y los componentes rehusados son integrados mediantes interfaces sencillas, garantizando así una total integración de las capas en el sistema.

A continuación se muestran los diagramas de integración interna y externa el cual muestra la integración del sistema a implementar con los diferentes componentes ya existentes del sistema.

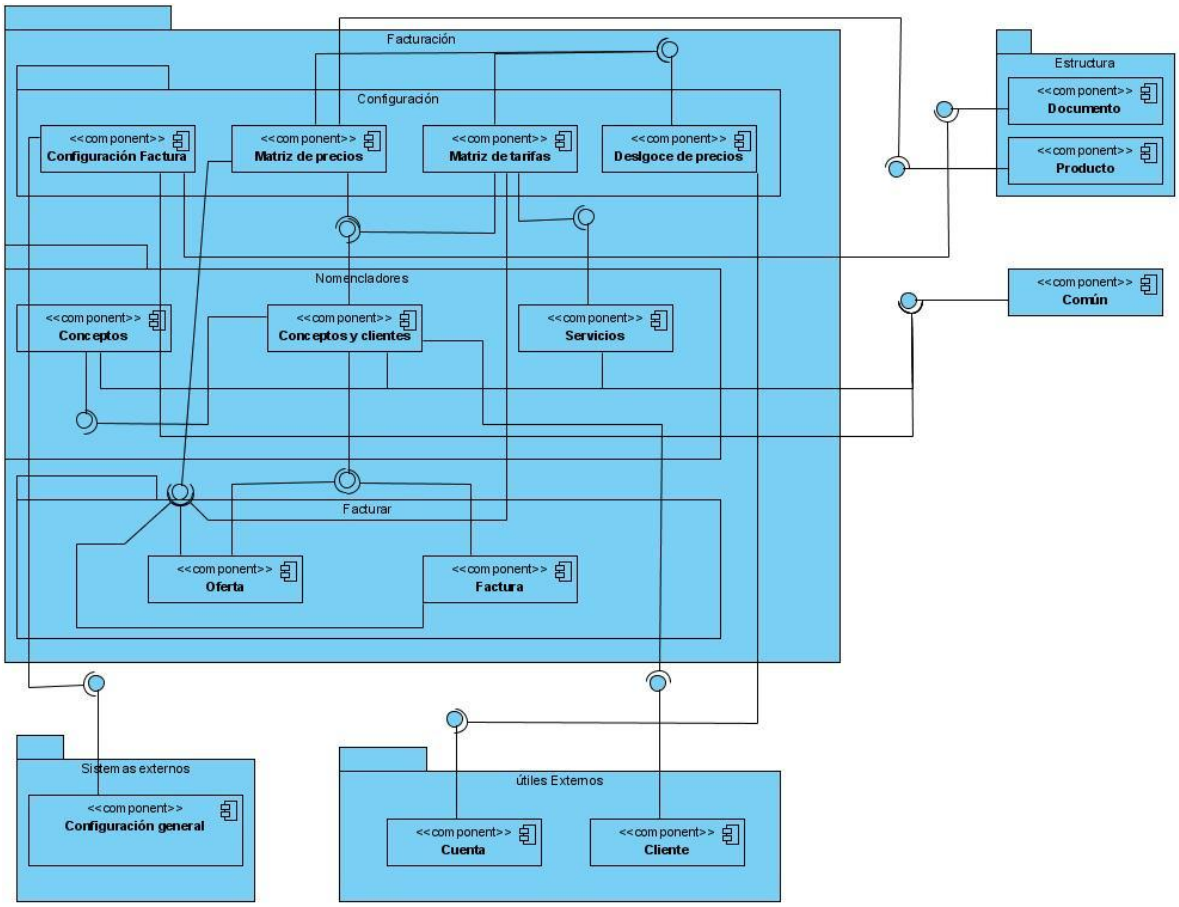


Figura 4. Diagrama de integración de componentes (Vista externa).

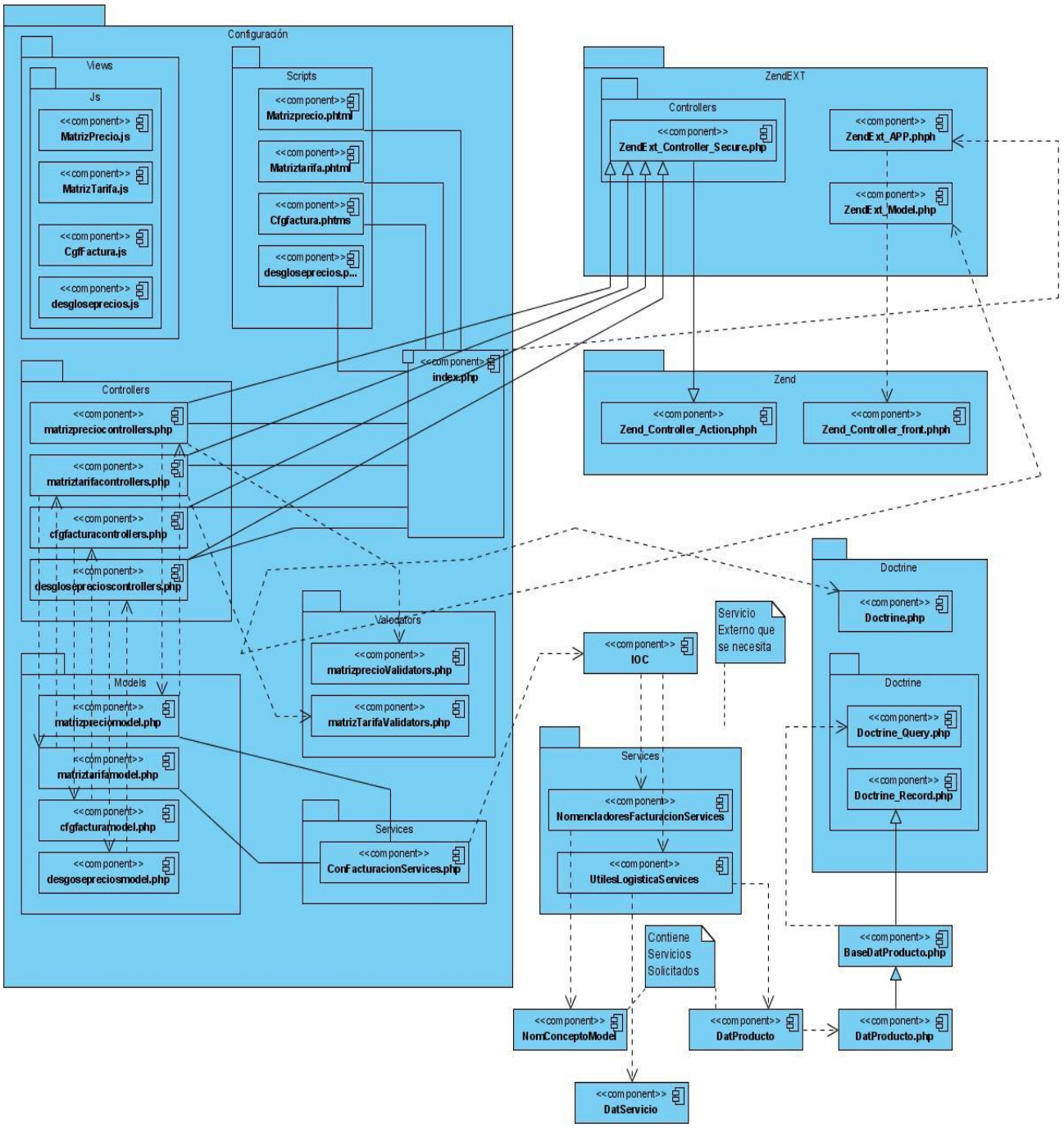


Figura 5. Diagrama de componentes.

2.10. ESTRATEGIA PARA LA CAPTURA DE ERRORES.

Como todo sistema, para garantizar su correcto funcionamiento se debe tener en cuenta un tratamiento de errores (excepciones), hacer captura de las excepciones (anexo III) que son lanzadas por el sistema para advertir al usuario de que una acción se ha ejecutado de forma incorrecta, por ejemplo, la introducción de un dato no válido, y que el mismo debe ser cambiado, el sistema captura todas aquellas excepciones que son lanzadas y se le da un tratamiento para que el sistema no interrumpa su funcionamiento.

Por ejemplo, se conoce inicialmente que a la hora efectuar una división entre dos números, el divisor debe ser distinto de 0. Si se viola esta condición, el sistema debe ser capaz de advertir al usuario acerca de este dato erróneo, en caso contrario, esta excepción es lanzada por el sistema.

De esta manera se da solución a las problemáticas de los lanzamientos de excepciones que se dan en el sistema según las peticiones del usuario, utilizando los try - catch, funciones que posibilitan hacer capturas de errores, capturando los tipos de errores lanzados y mostrando de manera visible al usuario mensajes de confirmación, el cual le dará una idea de qué aspecto no está funcionando bien y cómo solucionarlo.

Mediante la interfaz Web se impedirá que el usuario asuma un papel activo en la introducción de la información, para esto se contará con cuadros de opción, menú de selección lo cual facilitará la entrada de datos. La información que requiera ser adicionada por el usuario se validará mediante funciones que garanticen que sea válida y que el cuadro de texto no esté vacío si es obligatorio llenarlo.

Si hay un error en la información le saldrá al usuario un mensaje en pantalla indicándole el error, al oprimir Aceptar el mensaje desaparece y el usuario podrá seguir introduciendo los datos en el formulario. También se validarán las opciones correspondientes a la extracción o modificación de datos del servidor de base datos.

Si se desea eliminar algún elemento de la BD se preguntará al usuario si está seguro de realizar dicha acción, al igual que cuando desee modificar alguna información, antes de actualizarla se le preguntará si desea realizarla o no. Así se logra que se realicen las operaciones que se desean y que se rectifique al cometer un error.

CONCLUSIONES PARCIALES.

Con la realización de este capítulo, se obtiene que los resultados de los requerimientos propuestos por los analistas del sistema aportaron una gran vigencia a la implementación

propuesta por permitir una adecuada comprensión de las necesidades del cliente, lo que va a dar lugar a coleccionar toda la información necesaria para la realización de una implementación que cubra todas las funcionalidades que se necesitan para la satisfacción de las partes involucradas en el negocio.

Se implementaron las interfaces a partir de los prototipos propuestos y la información que se maneja en el sistema. Además se obtuvo la implementación del negocio conforme a los requerimientos especificados, así también como las validaciones, excepciones, y servicios incluidos dentro de las funcionalidades de los Subsistemas de Configuración y Nomencladores del Subsistema facturación.

CAPÍTULO III. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

INTRODUCCIÓN.

En la implementación de diferentes sistemas informáticos juega un papel esencial el uso de las técnicas de evaluación dinámica o pruebas. Para lograr esto se deben llevar a cabo estrategias a la hora de evaluar dinámicamente el software, pues para la evaluación de este se debe comenzar desde los componentes más simples y pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Es por esta razón que las técnicas utilizadas son las Pruebas Unitarias. Es una forma de probar el correcto funcionamiento de un módulo de código, sirve para asegurar que cada uno de los módulos funcione correctamente por separado.

La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuando se han realizado las suficientes pruebas para arribar a una determinación. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar. En este capítulo se realiza la validación a la solución propuesta en el capítulo anterior, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas para dar respuesta a los distintos requisitos planteados por el cliente.

3.1. Métricas.

Métrica es el término que describe muchos y muy variados casos de medición. Siendo una métrica una medida estadística (no cuantitativa como en otras disciplinas ejemplo física) que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcional, documentación, métodos, proceso, usuario, entre otros.

Para iniciar los elementos de medición, para el caso en cuestión se desarrollan tres diferentes tipos de medición, métricas técnicas, métricas Bang y métricas de punto de función.

Métricas Técnicas. Estas se presentan en el libro de Ingeniería del software de Pressman página 58. Estas métricas se derivan de una relación empírica según las

medidas contables del dominio de información del software y de evaluaciones de complejidad. Ejemplo,

Número de entradas usuario – cada una de las entradas de datos.

Número de salidas usuario – cada una de las salidas de datos.

Número de peticiones usuario – cada generación de un evento.

Número de archivos – cada tabla, archivo,…

Número de interfaces externas – son interfaces, discos, copias de seguridad, transmisiones de datos.

Tamaño operacional de clase (TOC):

Está dado por el número de métodos asignados una clase.

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

Tabla 9. Descripción de la métrica TOC.

Relaciones entre clases (RC):

Está dado por el número de relaciones de uso de una clase con otras.

Atributo que afecta	**Modo en que lo afecta**
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad del mantenimiento	Un aumento del RC implica un aumento de la

	complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 10. Descripción de la métrica RC.

3.2.1. Aplicación de las métricas:

Resultados del instrumento de evaluación de la métrica Tamaño Operacional de clase (TOC):

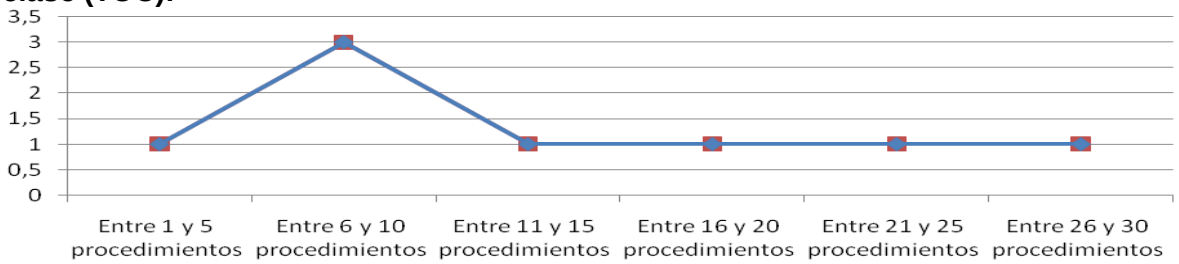


Figura 6. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

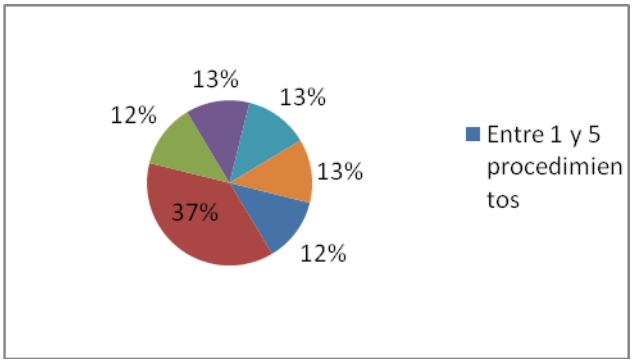


Figura 7. Representación de por ciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

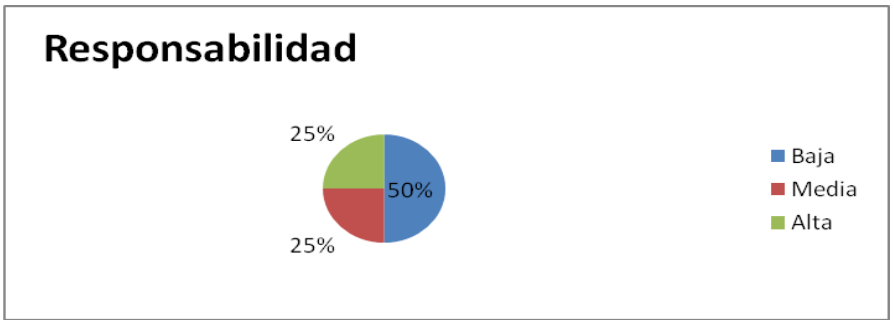


Figura 8. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

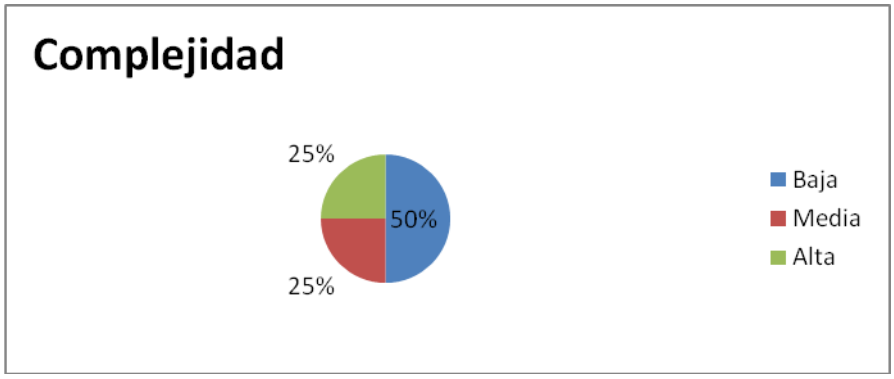


Figura 9. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

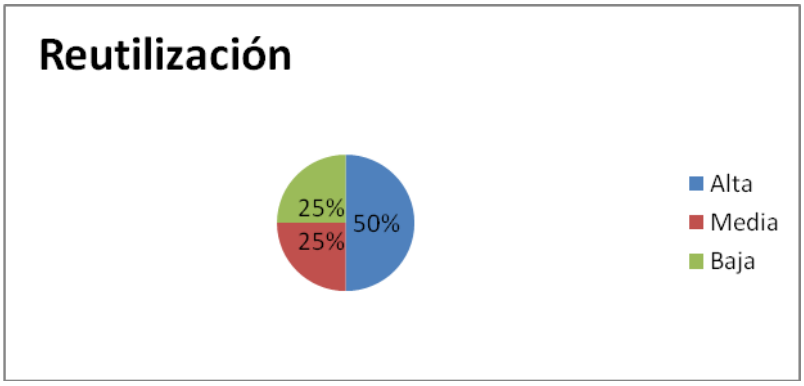


Figura 10. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización de Implementación.

Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño de los subsistemas de Configuración y Nomencladores del subsistema facturación tienen una calidad regular teniendo en cuenta que el 50 % de las clases incluidas en estos subsistemas posee igual cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 50% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

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

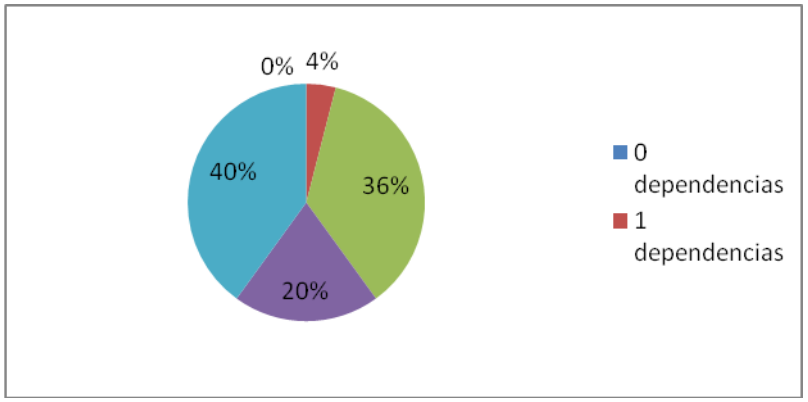


Figura 11. Representación en por ciento de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

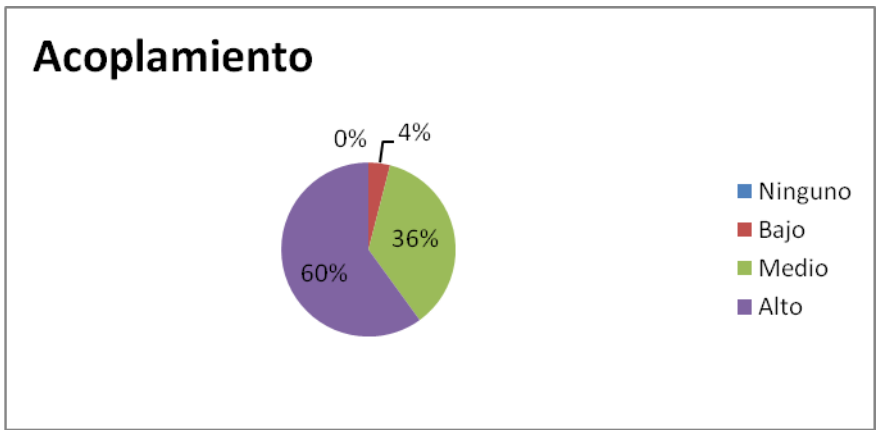


Figura 12. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

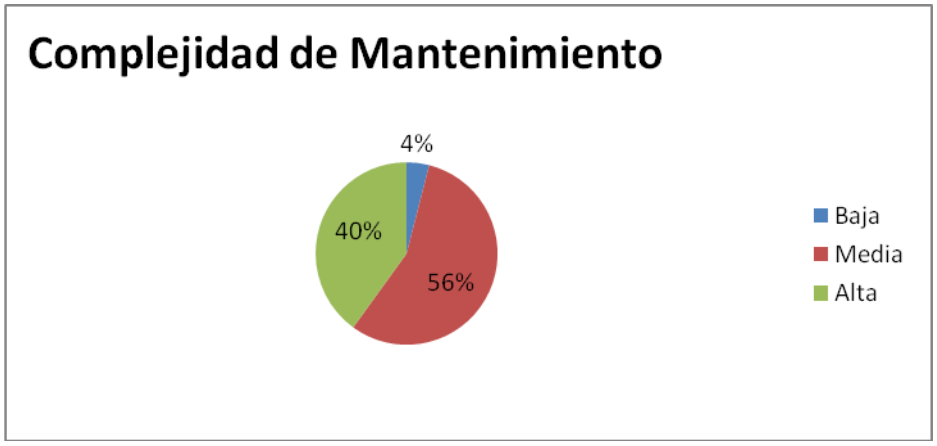


Figura 13. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

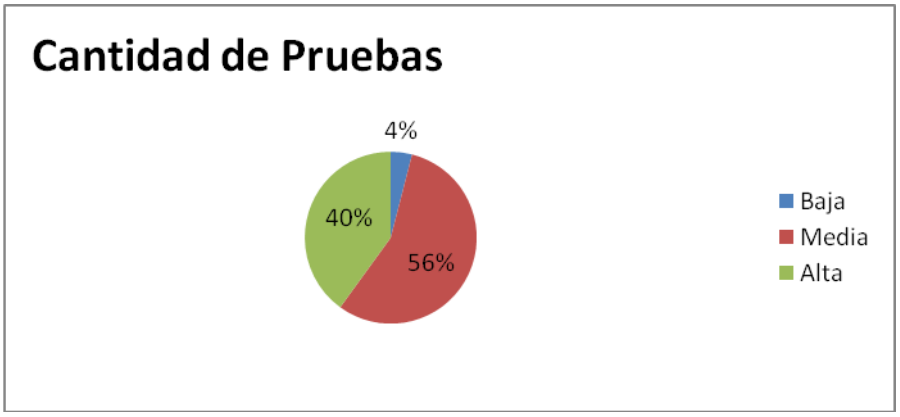


Figura 14. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

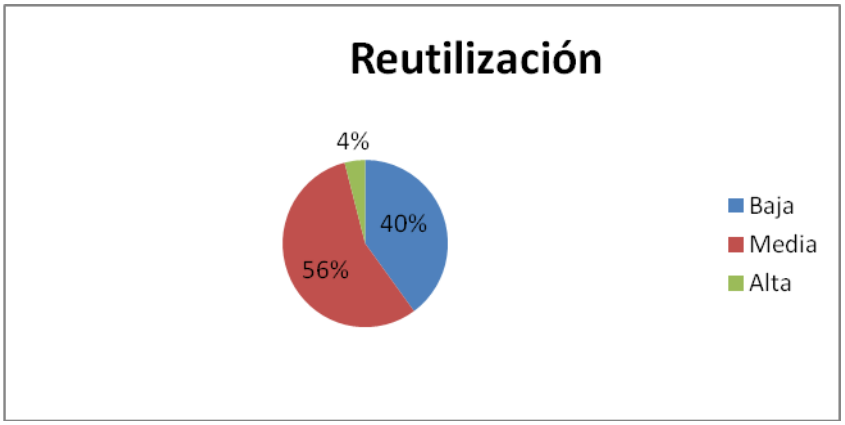


Figura 15. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño de los subsistemas de Configuración y Nomencladores de los procesos de Facturación tiene una calidad aceptable teniendo en cuenta que el 80 % de las clases incluidas en estos subsistemas posee menos de 3 dependencias de otras clases.

Además todas las clases poseen acoplamiento con otras y el 96% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 96 % de las clases.

3.2. PRUEBAS DE SOFTWARE.

La IEEE define las pruebas como una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, se observan o almacenan los resultados realizándose una evaluación de algún aspecto del sistema o componente. Las pruebas son sets de uno o más casos de pruebas.

Un proceso de pruebas son conceptos de pruebas, estrategias, técnicas y medidas que deben integrarse en un determinado proceso controlado y dirigido por personas, apoyando a las actividades de prueba y proporcionando una guía para los equipos de pruebas, desde la planificación de prueba, para probar la salida de evaluación, de tal forma que proporcione garantía de que los objetivos de las pruebas se cumplan de manera rentable.

Tomando como base los conceptos anteriormente citados se puede llegar a la conclusión de que las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, donde se valida la calidad y funcionalidad del producto mediante el análisis y evaluación de los resultados de dicha ejecución con el fin de detectar errores, que impidan que dicho producto responda con las necesidades especificadas previamente.

Un proceso de prueba es el establecimiento de los objetivos de las pruebas, el diseño de los casos de pruebas a aplicar, la codificación de los casos de pruebas, la ejecución de los mismos y el análisis de los resultados de dicha ejecución.

3.2.1. Objetivos.

El objetivo esencial en la etapa de pruebas es garantizar la calidad del producto desarrollado. Esto implica:

- ⇒ Verificar la interacción de componentes.
- ⇒ Verificar la integración adecuada de los componentes.
- ⇒ Verificar que todos los requisitos se han implementado correctamente.
- ⇒ Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.
- ⇒ Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo."

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que debe ocurrir durante todo el ciclo de vida: probar la funcionalidad de los primeros prototipos; probar la estabilidad, cobertura y rendimiento de la arquitectura; probar el producto final, entre otras. Lo que conduce al principal beneficio de la prueba: proporcionar feedback mientras hay todavía tiempo y recursos para hacer algo.

3.2.2. Alcance.

Se lleva a cabo la prueba y se evalúan los resultados obtenidos frente a los resultados esperados. Si se descubren datos erróneos da lugar a que existe un error y hay que corregirlo aquí comienza el proceso de depuración de errores. Basado en las estructuras de control del diseño procedimental para generar los casos de prueba que:

- ✓ Garanticen que se recorran por lo menos una vez todos los caminos independientes de cada módulo.
- ✓ Se ejecuten todas las decisiones lógicas en su parte verdadera y en su parte falsa.
- ✓ Se recorran todos los bucles.
- ✓ Se utilizan las estructuras de datos internas para garantizar su validez.
- ✓ Se invierte tiempo y esfuerzo en los detalles de control debido a que:

- ✓ Los errores suelen estar en situaciones fuera de las normales.
- ✓ A menudo caminos que se piensa que tienen pocas posibilidades de recorrerse, son recorridos regularmente.
- ✓ Los errores tipográficos son aleatorios. Puede que no sean detectados por los procesadores de la sintaxis del lenguaje particular y estar presentes en cualquier camino lógico.

3.3. DESCRIPCIÓN DE LOS TEST DE UNIDAD.

Las pruebas de unidad es la manera de comprobar el funcionamiento correcto de determinado módulo de código, ayuda a independizar el módulo, significa esto que se pueden probar los módulos independientemente uno de otros.

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

Los “test de unidades” son orientados en la mayoría de los casos a las pruebas de “caja blanca” aunque para realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no se comportan correctamente al ser introducidos en la aplicación, todas las demás pruebas no tienen sentido.

Estas pruebas son las denominadas pruebas de caja blanca y son aplicadas a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que estos funcionen como se espera.

3.3.1. Prueba de Caja Blanca o Estructurales.

La prueba del camino básico es una técnica de prueba de caja blanca propuesta por Tom McCabe en 1976. Esta permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución.

Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

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

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

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

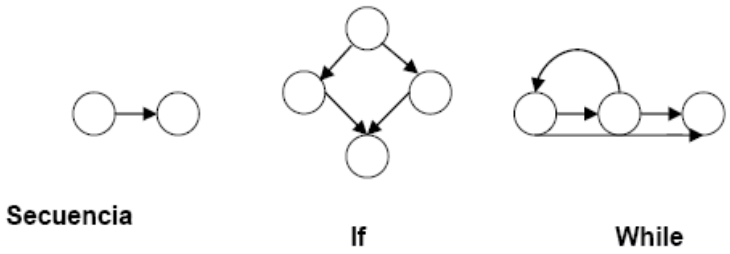


Figura 16. Notación de grafos de flujo para las instrucciones: Secuenciales, If, While.

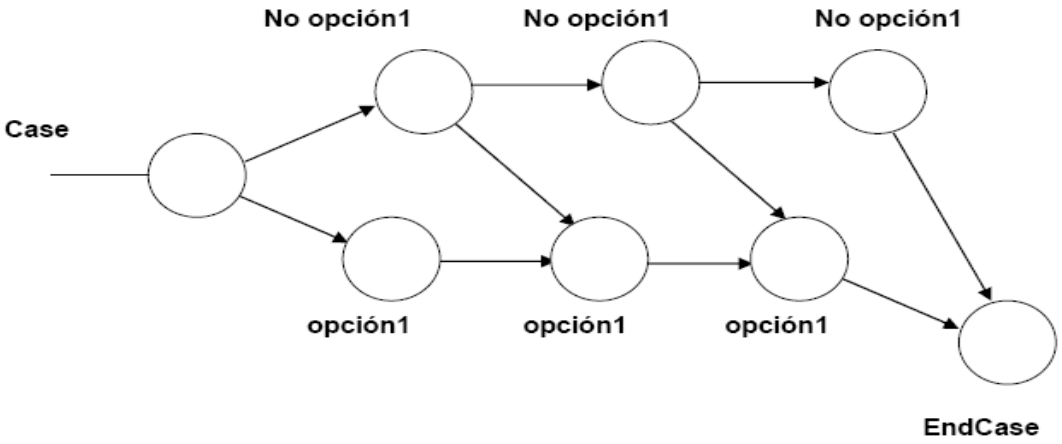


Figura 17. Notación de grafos de flujo para la instrucción Case.

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

Componentes del grafo de flujo

- Nodo: Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.
- Aristas: Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.
- Regiones: Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

En la figura 18 se observa la representación un grafo de flujo, en el cual aparecen sus componentes.

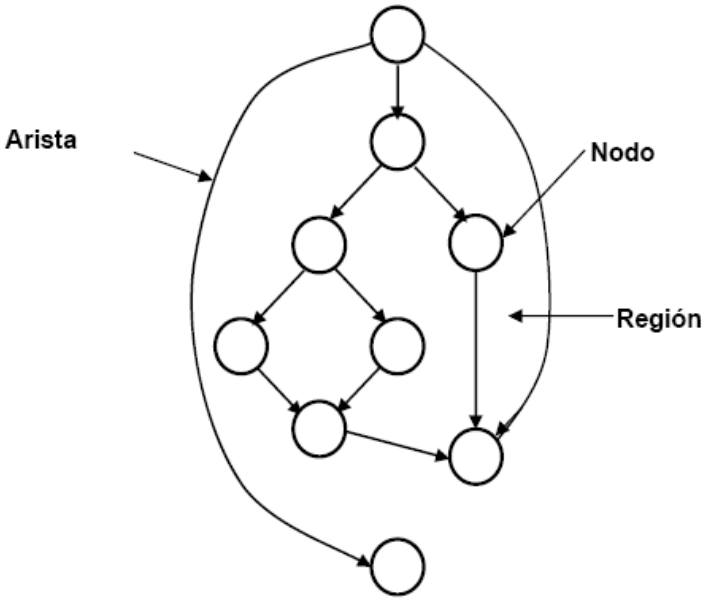


Figura 18. Componentes de los grafos de flujo.

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

La complejidad ciclomática es una métrica de software extremadamente útil, proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente.

Si se diseñan pruebas que fuesen el recorrido de esos caminos, se garantiza que se ejecute al menos una vez cada sentencia del programa y que cada condición se ejecute en sus variantes verdadera y falsa. Se debe tener en cuenta que de un mismo diseño procedimental se pueden derivar varios conjuntos básicos.

Formas fundamentales de calcular la complejidad:

Atendiendo al número de regiones del grafo de flujo:

El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

1. La complejidad ciclomática, V(G), se define como:
   $V(G) = A - N + 2$
   Donde: A es el número de aristas del grafo y N es el número de nodos.

2. La complejidad ciclomática, V(G), también se define como:
   $V(G) = P + 1$

   Donde: P es el número de nodos predicados contenidos en el grafo G

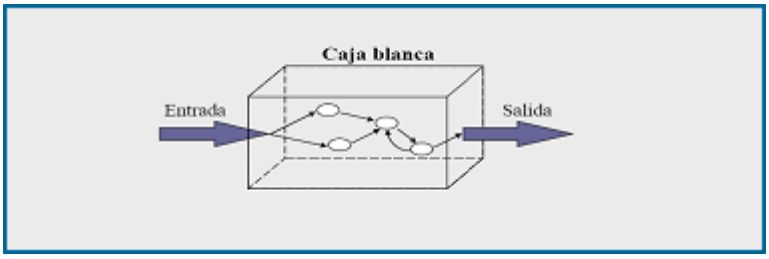


Figura 19. Representación de pruebas de Caja Blanca.

Mediante las pruebas de la caja blanca se puede obtener casos de prueba que:

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

Algunas de las técnicas de prueba de Caja Blanca podemos citar:

1. **Prueba de Condición:** Método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.
2. **Prueba de Flujo de Datos:** Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.
3. **Prueba de Bucles:** Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.
4. **Prueba del Camino Básico:** Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico.

   La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática.

Aplicando el diseño de casos de pruebas al software en cuestión se puede conseguir una prueba más completa, descubriendo y corrigiendo el número mayor de errores antes de que comiencen las “pruebas del cliente”. Para probar se necesita una organización y planificación de las pruebas que se van a realizar.

La forma más práctica de aplicar los tipos de pruebas y la que se propone es la siguiente:

- Medir la cobertura de caja blanca que se ha logrado con las fases previas y añadir más pruebas de Caja Blanca hasta lograr la cobertura deseada.
- Diseñar casos de prueba para examinar la lógica del programa para garantizar que se ejerciten todos los caminos independientes de cada módulo, todas las decisiones lógicas, y que se ejecutan todos los bucles y las estructuras de datos internas.

3.4. APLICACIÓN DE PRUEBAS DE CAJA BLANCA.

```
function esta_relacion($idcatprecio)(1)
  {
    $relaciones = DatRelacionprecio :: Gettodos();(1)
    foreach ($relaciones as $index => $relacion )(2)
    {
      if($relacion['idcategoriaprecio'] == $idcatprecio)(3)
        return true;(3)
    }

    return false;(4)
  } (5)
```

Figura 20. Representación del algoritmo esta relación ($idcatprecio).

Seguidamente se construye el grafo de flujo asociado al código anterior.

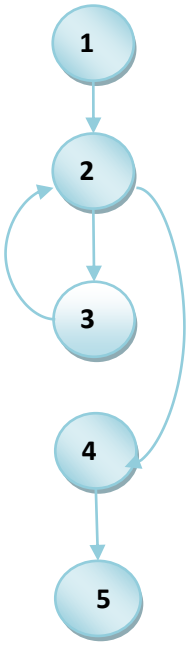


Figura 21. Representación del flujo asociado al algoritmo esta_relacion($idcatprecio).

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

Para efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo, cantidad total de nodos para la siguiente fórmula:

V (G) = (A – N) + 2

V (G) = (5 – 5) + 2

V (G) = 2

Siendo:

A: la cantidad total de aristas

N: la cantidad total de nodos.

Se puede calcular también de esta otra manera:

V (G) = P + 1

V (G) = 1 + 1

V (G) = 2

Para:

P: Cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = R

V (G) = 2

R: Cantidad total de regiones, para cada fórmula V (G) representa el valor del cálculo de la complejidad ciclomática.

Se ha obtenido el mismo resultado luego de haber calculado la complejidad ciclomática mediante las tres fórmulas por lo que se puede concluir que la complejidad ciclomática del código que implementa el método esta_relacion($idcatprecio). Es igual a 2 unidades, esto da lugar a que existan 2 caminos básicos por los que el flujo puede transitar, la complejidad ciclomática va a representar el límite del mínimo número total de casos de pruebas para el procedimiento tratado.

A continuación se representan los caminos básicos por los que puede recorrer el flujo. Se subrayan las representaciones y los elementos de cada camino que los hacen independientes a los demás.

Número	Camino básico
1	1 – 2 – 4 – 5
2	1 – 2 – 3 – 2 – 4 – 5

Tabla 11. Caminos básicos del flujo.

Ahora se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.

Para la exitosa realización de los mismos es necesario cumplir con las siguientes exigencias:

- ✓ Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento, o no se entre algún dato erróneo.
- ✓ Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- ✓ Entrada: Se muestran los parámetros que entran al procedimiento.
- ✓ Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico 1:

Camino 1: [1 – 2 – 4 – 5]

Descripción:

Se ejecuta cuando no existan relaciones registradas en la Base de datos para el idcategoriaprecio pasado por parámetro

Condición de ejecución:

- No existirá ninguna relación registrada en la base de datos para el idcategoria precio pasado por parámetro.

Resultados esperados:

Se retornará el dato booleano false.

Caso de prueba para el camino básico 2

Camino 2: [1 – 2 – 3 – 2 – 4 – 5]

Condición de ejecución:

En la Base de datos existirá registrado al menos 1 relación de precios para el idcategoria de precios pasado por parámetro.

Resultados esperados:

Se espera que se devuelva un valor booleano de verdadero (true)

Posterior a aplicar los distintos casos de pruebas, se verifica que el flujo de trabajo de la función está correcto cumpliendo en mismo con las condiciones necesarias que se habían planteado.

CONCLUSIONES PARCIALES.

Con el desarrollo del presente capítulo para la comprensión del mismo se analizaron diferentes conceptos como el de las pruebas de software, sus objetivos, diferentes alcances y aplicaciones. Se realizó también una breve descripción de los test de unidad y las pruebas de caja blanca, haciendo énfasis en las pruebas del camino básico y el cálculo de la complejidad ciclomática.

Al aplicar estas pruebas de unidad a un fragmento de código de la implementación, del cual se obtuvo una complejidad ciclomática de 2 unidades, dando lugar a 2 caminos básicos, comprobándose que eran los únicos por los que siempre transitaba el flujo, se diseñaron 2 casos de pruebas de los cuales se obtuvo los resultados esperados.

Por último, se elaboraron instrumentos inspirados en las métricas para la calidad del diseño como Tamaño Operacional de Clase (TOC) y las Relaciones entre Clases (RC), evaluándose varios factores claves como la reutilización, complejidad de implementación, entre otros, de los cuales se obtuvo para la métrica TOC una calidad regular, y para la RC, una calidad aceptable.

Los resultados de las pruebas realizadas al software y la aplicación de las métricas en ambos casos descritos anteriormente dan lugar a que la implementación realizada presenta una calidad aceptable, siendo positiva la validación de la solución propuesta.

CONCLUSIONES GENERALES.

La realización de la presente investigación permite plantear las siguientes conclusiones:

1. El estudio realizado sobre los procesos de Configuración y Nomencladores para la facturación tiene gran importancia para el progreso económico de las entidades empresariales y unidades presupuestadas del país. Sin embargo, los sistemas informáticos existentes no cumplen con las expectativas actuales, pues no responden a las políticas de software libre. La complejidad con que se realizan estos procesos muestra la necesidad de crear aplicaciones informáticas que los optimicen.
2. A partir del análisis teórico realizado sobre los subsistemas de facturación, pertenecientes a los sistemas ERP y las particularidades tecnológicas del país, se implementaron subsistemas de Configuración y Nomencladores que gestionan los procesos de facturación, que serán utilizados en las entidades empresariales y unidades presupuestadas del país.
3. La validez de la implementación de los subsistemas de Configuración y Nomencladores para los procesos de facturación en el sistema Cedrux se realizó utilizando pruebas de software (caja blanca); pruebas unitarias del camino básico, de acuerdo a la complejidad ciclomática del fragmento del código analizado y métricas aplicadas al diseño de las clases: métricas de Tamaño Operacional de Clases (TOC) y de Relaciones entre Clases (RC), lo que mostró evidencias positivas de calidad.

RECOMENDACIONES.

Con la culminación del presente trabajo se recomienda:

- Solucionar las no conformidades de las pruebas pilotos con el objetivo de refinar la solución.
- Implementar futuras funcionalidades que puedan ser propuestas por los analistas.

BIBLIOGRAFÍA.

1. Dos Ideas.com. [En línea] [Citado el: 29 de Junio de 2009.] http://www.dosideas.com/metodologias/320-como-entender-la-complejidad-ciclomatica.html.

2. **Suárez Quintana, Lic Arnulfo A.** *SISTEMAS INTEGRALES DE FACTURACION.*

3. **Pressman, Roger S.** *Ingeniería del Software, un enfoque práctico.*

4. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* Madrid : Pearson Educación S.A, 2000. 84-7829-036-2.

5. **ICM.** *SISTEMA DE GESTION COMERCIAL ISIS GEST EMPRESARIAL.* Miraflores : s.n.

6. **CITMATEL.** Rodas XXI. *Un Sistema Económico Integral Administrativo.* [En línea] [Citado el: 15 de Mayo de 2009.] http://www.rodasxxi.cu/.

7. Portal Especializado de Software y Servicios Relacionados para el Sector Empresarial. [En línea] 2009 de Mayo de 2000. [Citado el: 30 de Mayo de 2009.] http://www.catalogodesoftware.com/producto.aspx?pid=239.

8. Openbravo.com. [En línea] 12 de Marzo de 2009. [Citado el: 20 de Mayo de 2009.] http://wiki.openbravo.com/wiki/ManualRappels/es#Generar_factura_de_un_rappel.

9. help.sap.com. *help.sap.com.* [En línea] [Citado el: 20 de Mayo de 2009.] http://help.sap.com/saphelp_srm40/helpdata/es/66/81b56c553b5840994ea041a5becd63/content.htm.

10. El eco del Contador. *Contabilidad.* [En línea] Octubre de 2008. [Citado el: 25 de Mayo de 2009.] http://elecodelcontador.com/blog/?m=200810.

11. Libros Web. [En línea] http://www.librosweb.es/jobeet/capitulo4/la_arquitectura_mvc.html.

12. **Peláez, Juan.** Blog de Juan Peláez en Geeks.ms. [En línea] [Citado el: 27 de Mayo de 2009.] http://geeks.ms/blogs/jkpelaez/archive/2009/05/29/arquitectura-basada-en-capas.aspx.

13. El Villaclareño. [En línea] [Citado el: 15 de Mayo de 2009.] http://www.forum.villaclara.cu/.

14. **ESTERELLAS, MARLENE.** Hacia el buen desarrollo empresarial. [En línea] [Citado el: 15 de Mayo de 2009.] http://www.cubahora.cu/index.php?tpl=principal/ver-noticias/ver-not_soc.tpl.html&newsid_obj_id=1011998.

15. **B., Juan.** Openbravo. *sistema de gestión de sofware libre español.* [En línea] PrymeCrunch, 7 de Marzo de 2008. [Citado el: 1 de Junio de 2009.] http://pymecrunch.com/openbravo-sistema-de-gestion-de-sofware-libre-espanol.

16. Qué es SAP/3? [En línea] [Citado el: 20 de Mayo de 2009.] http://espanol.geocities.com/emoly188/que_es_sap_r3.htm.

17. **Productos CONDOR.** Productos Aplicativos . [En línea] [Citado el: 21 de Mayo de 2009.] http://www.open-sol.com/spa/productos/productos.htm.

18. Consejo Superior de Administración Electrónica. [En línea] [Citado el: 20 de Mayo de 2009.] http://www.csi.map.es/csi/silice/Global71.html.

19. Guia breve de Tecnologias XML. [En línea] 09 de Enero de 2008. [Citado el: 20 de Mayo de 2009.] http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML.

20. **Bravo Montero, Joaquin.** XML En Castellano. [En línea] [Citado el: 5 de Mayo de 2009.] http://www.programacion.net/html/xml/principal.htm.

21. JavaScript - MDC. [En línea] 6 de Agosto de 2008. [Citado el: 20 de Mayo de 2009.] https://developer.mozilla.org/es/JavaScript.

22. PHP: Una Explicación Sencilla. [En línea] [Citado el: 12 de Junio de 2009.] http://cl.php.net/tut.php.

23. Desarrollo Web.com. [En línea] [Citado el: 31 de Mayo de 2009.] http://www.desarrolloweb.com/php/.

24. Cómo conseguir el mejor IDE de desarrollo. [En línea] 17 de Agosto de 2008. [Citado el: 21 de Mayo de 2009.] http://www.indalcasa.com/programacion/como-conseguir-el-mejor-ide-de-desarrollo-para-la-web/.

25. Espacio Linux.COM. [En línea] [Citado el: 21 de Mayo de 2009.] http://www.espaciolinux.com/portal/.

26. **Artola, Luis.** Comparativa entre Zend Studio y Eclipse PDT. [En línea] [Citado el: 21 de Mayo de 2009.] http://www.programania.net/php/comparativa-entre-zend-studio-y-eclipse-pdt/.

27. Maestros de la Web. *Los Frameworks PHP agilizan tu trabajo.* [En línea] 31 de Julio de 2007. [Citado el: 11 de Mayo de 2009.] http://www.maestrosdelweb.com/editorial/los-frameworks-de-php-agilizan-tu-trabajo/.

28. WordPress.com. *Sobre Zend Framework.* [En línea] [Citado el: 22 de Mayo de 2009.] http://es.wordpress.com/tag/zend-framework/2/.

29. **Lockhart, Thomas.** Tutorial de PostgreSQL. [En línea] [Citado el: 21 de Mayo de 2009.] http://sdi.bcn.cl/desarrollo/doctos/PostgreSQL%20-%20Tutorial.pdf.

30. pgAdmin III - Guia Ubuntu. [En línea] 10 de Marzo de 2008. [Citado el: 13 de Mayo de 2009.] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

31. TodoExpertos.com. [En línea] 26 de Enero de 2006. [Citado el: 22 de Mayo de 2009.] http://www.todoexpertos.com/categorias/tecnologia-e-internet/bases-de-datos/respuestas/1198624/manual-de-pg-admin-iii.

32. EMS SQL Management Studio TM. [En línea] [Citado el: 21 de Mayo de 2009.] http://sqlmanager.net/.

33. **Mozilla Europe y Mozilla Foundation.** Navegador Web Firefox. [En línea] Mozilla Europe y Mozilla Foundation. [Citado el: 21 de Mayo de 2009.] http://www.mozilla-europe.org/es/firefox/.

34. **Fac de Ing Lufi.** Introducción a los Sistemas de Control de Versiones. [En línea] [Citado el: 23 de Mayo de 2009.] http://www.lug.fi.uba.ar/documentos/scms/.

35. **Collins-Sussman, Ben y Fitzpatrick, Brian W.** Control de versiones con Subversion. [En línea] [Citado el: 23 de Mayo de 2009.] http://svnbook.red-bean.com/index.es.html.

36. Tortoise SVN - Guia de Instalación. [En línea] [Citado el: 15 de Mayo de 2009.] http://tortoisesvn.net/docs/release/TortoiseSVN_es/tsvn-introduction.html#tsvn-intro-about.

37. **Geocites.com.** Especificación y Manejo de los requerimientos de Software. [En línea] [Citado el: 11 de Mayo de 2009.] http://us.geocities.com/txmetsb/req-mgm-1.htm.

38. ARQUITECTURA Modelo Vista Controlador. [En línea] [Citado el: 24 de Mayo de 2009.] http://www.ulpgc.es/otros/tutoriales/java/Apendice/arq_mvc.html.

39. Definición de Camel. *Indanga.* [En línea] [Citado el: 5 de Mayo de 2009.] http://www.indaganda.com/definir/camel.

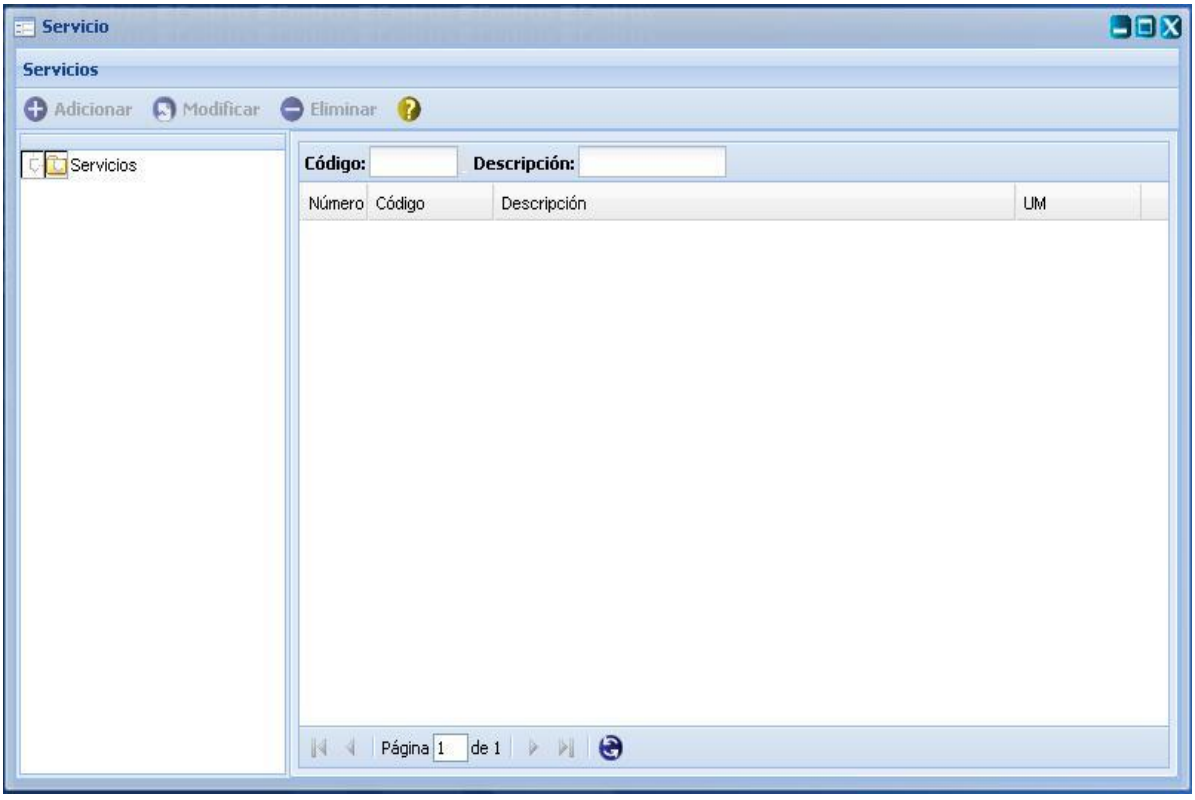
40. **Collado, Manuel.** Pruebas de software. [En línea] Marzo de 2003. [Citado el: 27 de Mayo de 2009.] http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt#290,14,Actividades%20de%20prueba%20de%20software.

41. **Hernández Sampieri, Roberto, Fernández Collado, Carlos y Baptista Lucio, Pilar.** *Metodología de la investigación.* México : Mc Graw Hil, 2003. 970-10-3632-8.

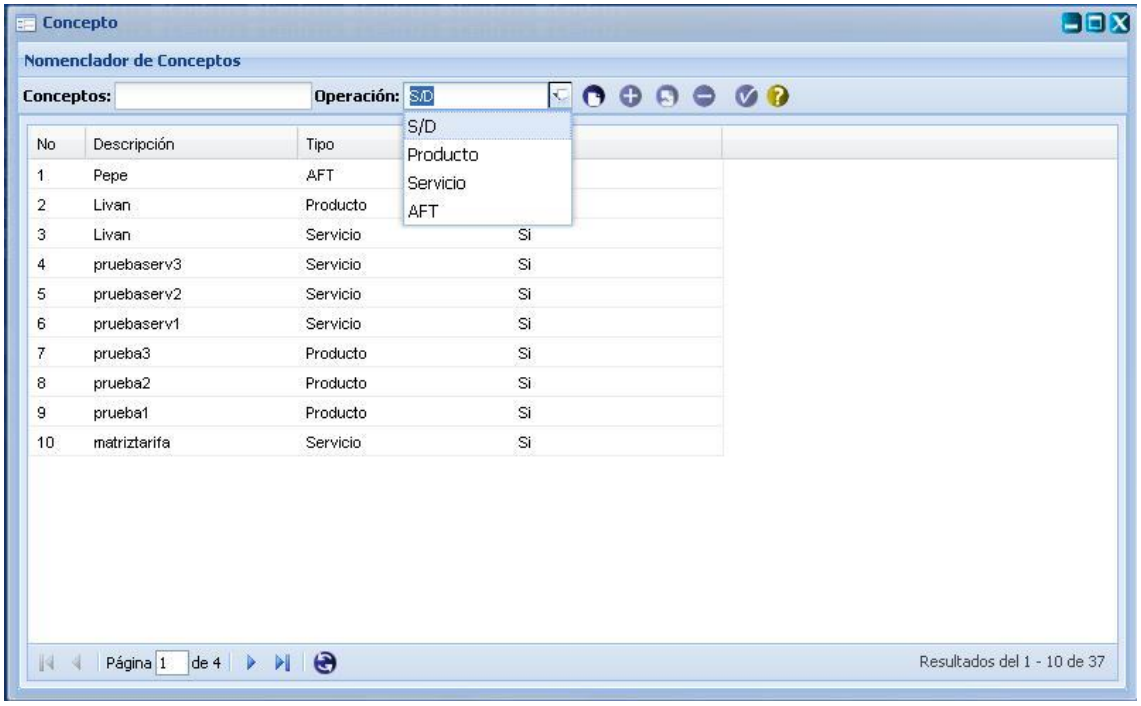
ANEXOS.

ANEXO I. INTERFACES CREADAS DEL COMPONENTE NOMENCLADORES.

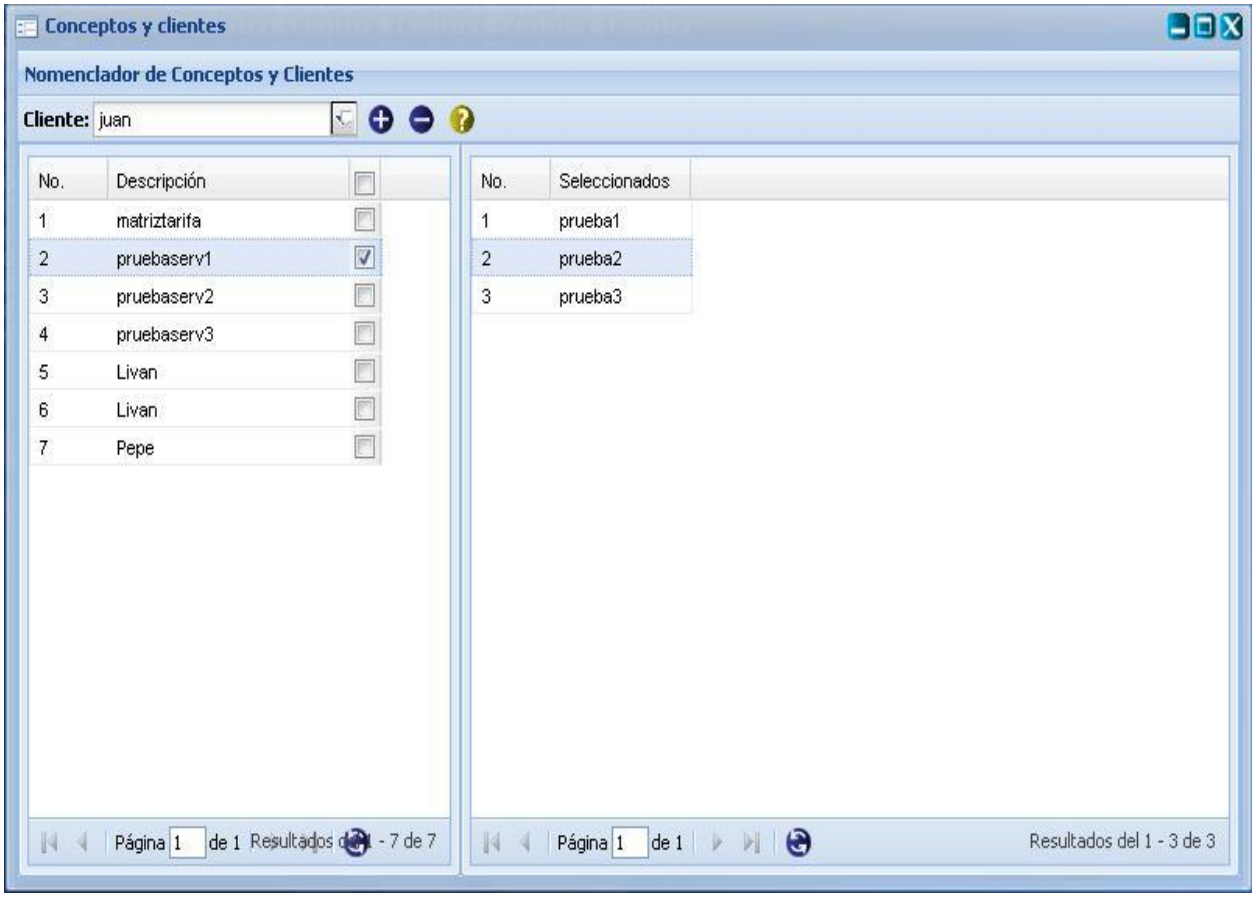
Nomenclador de Servicios:



Nomenclador de Conceptos.

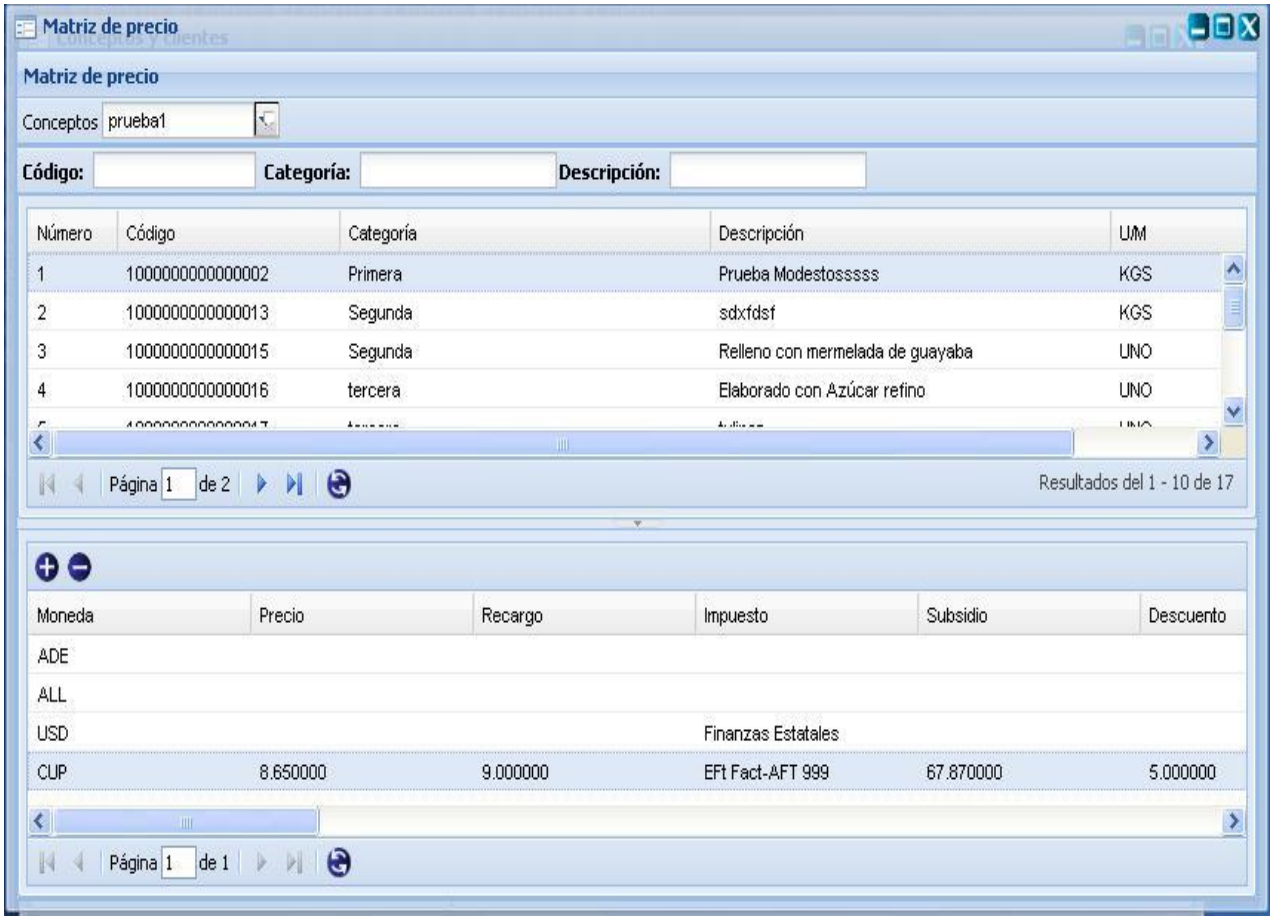


Nomenclador de conceptos y clientes.

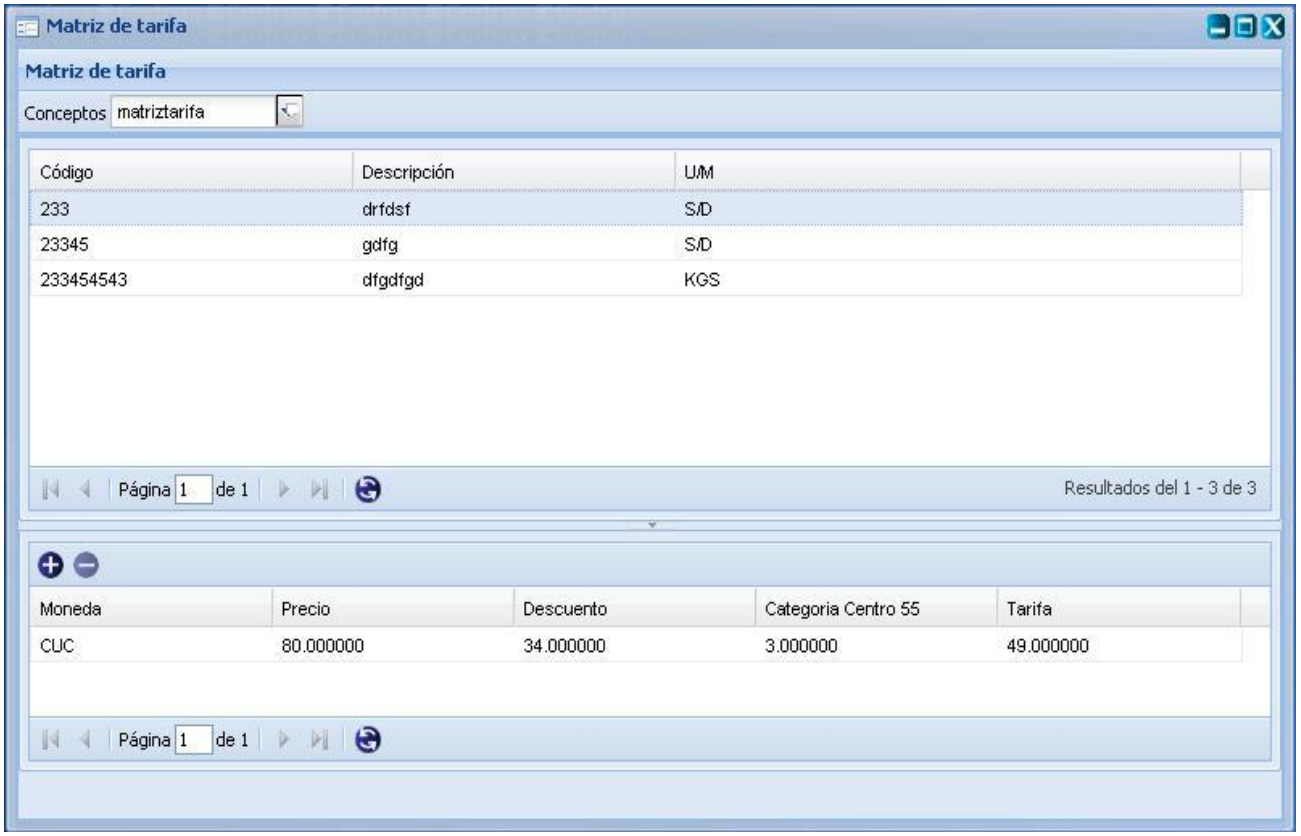


ANEXO II. INTERFACES CREADAS DEL COMPONENTE CONFIGURACIÓN.

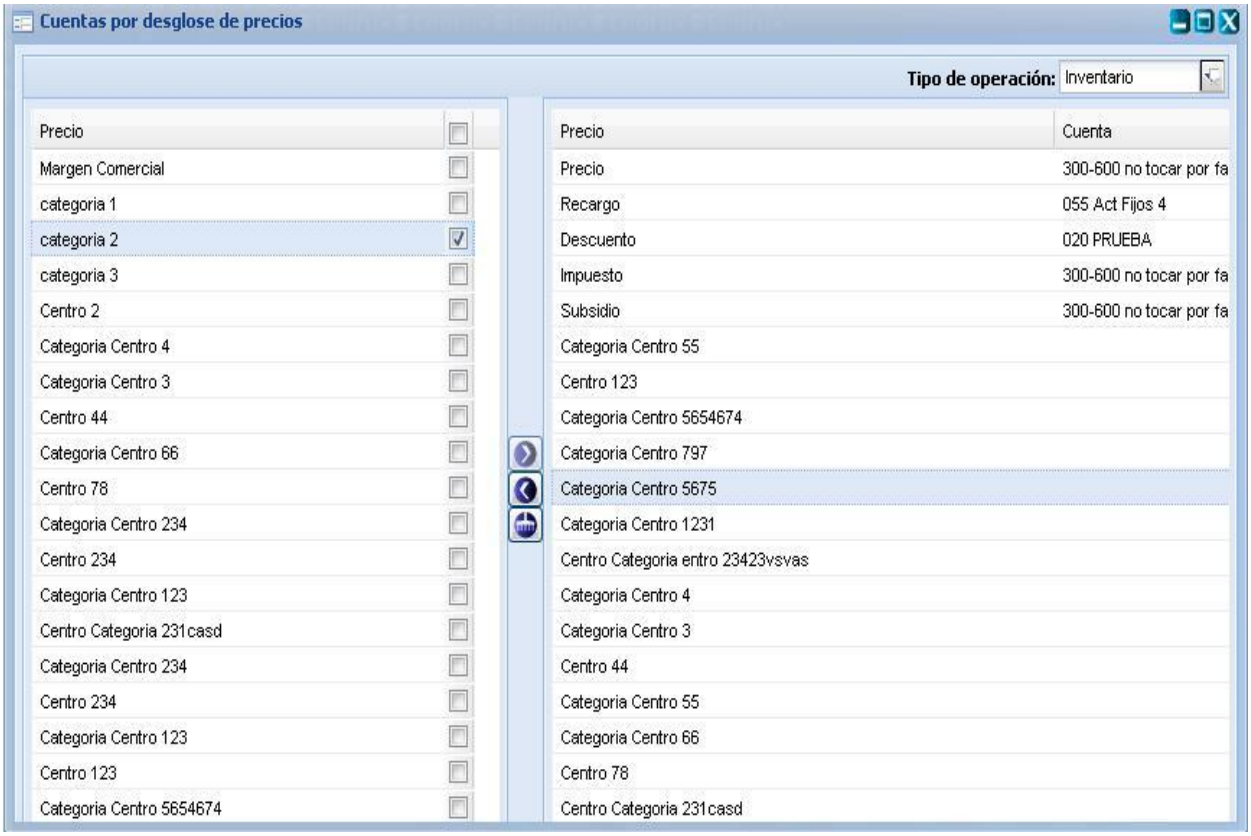
Matriz de Precios



Matriz de tarifas.



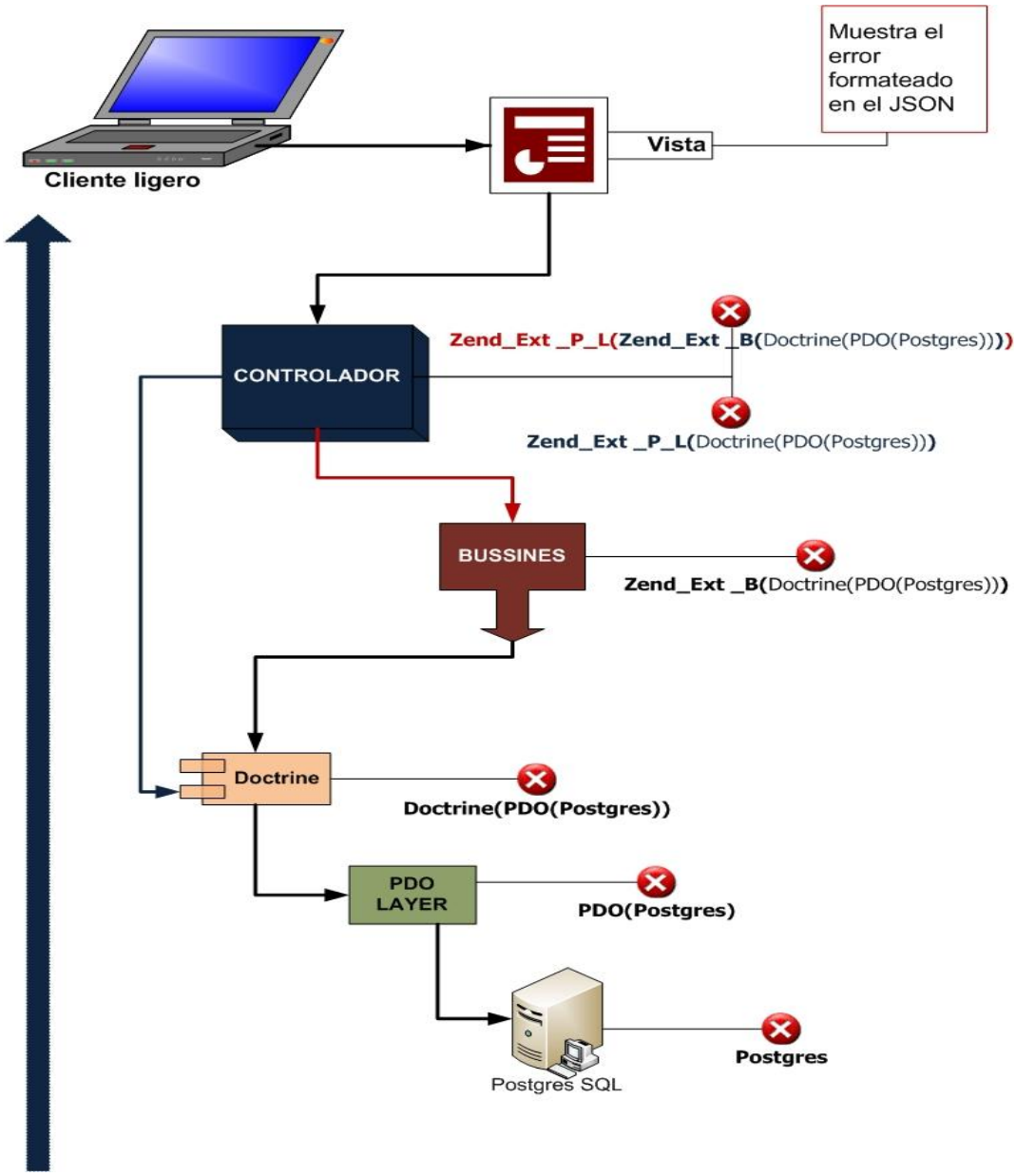
Cuentas por desglose de precios.



ANEXO III. ESTRATEGIA PARA LA CAPTURA DE ERRORES.

Se crearon cuatro (4) tipos de excepciones

- **B** Excepción ciega, posee la función de contener la información del error capturado. Se usa para encapsular las excepciones lanzadas por el sistema operativo u otro componente de software utilizado como doctrine por ejemplo.
- **L** Excepción LOG, permite dejar registro del error y sus características más técnicas, para luego ser gestionadas por el equipo de desarrollo. En ocasiones contiene varias Excepciones B, que fueron siendo anidadas en las capas que colaboran.
- **P** Excepción Presentación, se utiliza para mostrar determinado mensaje de error al cliente, en un lenguaje natural y entendible. No posee características técnicas del código. Puede poseer excepciones anidadas de tipo B o L.
- **PL** Excepción Presentación Log, se utiliza para mostrar determinado mensaje de error al cliente, en un lenguaje natural y entendible, así como registrar el mensaje en algún mecanismo de persistencia para luego ser gestionado por el equipo de desarrollo. Puede poseer excepciones anidadas de tipo B o L.



Tratamiento de excepciones en la arquitectura.

Los elementos de la arquitectura colaboran según su responsabilidad, cada elemento presenta excepciones características a su nivel de abstracción. Las excepciones internas son anidadas por otras de mayor nivel, para luego ser registrado finalmente según las características de cada excepción anidada en la colaboración.

GLOSARIO DE TÉRMINOS.

Activos: Conjunto de bienes tangibles o intangibles que posee una empresa, aquellos bienes que tienen una alta probabilidad de generar un beneficio económico en un futuro y se pueda gozar de los beneficios económicos que otorga.

Activos Fijos: Son aquellos activos que no varían durante el ciclo de explotación de la empresa (o el año fiscal). Por ejemplo, el edificio donde una fabrica monta sus productos es un activo fijo porque permanece en la empresa durante todo el proceso de fabricación y venta de los productos.

Algoritmo: Es un conjunto finito de instrucciones o pasos lógicos que sirven que brindan una solución a una tarea u problema a resolver.

Conceptos de venta: Estructura que se le asigna a clientes, por los que se regirá el mismo para realizar la compra de bienes que oferte la entidad, en el precio y estructuras determinadas para dicho concepto.

Configuración: Inicialización de las partes que componen una cosa, (en este caso el componente Facturación) y le dan su peculiar forma y propiedades para su posterior funcionamiento.

Componente: Es la unidad de construcción elemental del diseño físico. Las características de los mismos son las siguientes:

- Se define de acuerdo como interactúe con otros.
- Encapsula sus funciones y datos.
- Es reutilizable a través de las aplicaciones.
- Se puede observar como una caja negra.
- Puede contener otros componentes.

Empresa: Unidad económico-social, con fines de lucro, en la que el capital, el trabajo y la dirección se coordinan para realizar una producción socialmente útil, de acuerdo con las exigencias del bien común.

Factura: La factura necia o factura de compra es un documento que refleja la entrega de un producto o la provisión de un servicio, junto a la fecha de devengo, además de indicar

la cantidad a pagar como contraprestación. En la factura se encuentran los datos del expedidor y del destinatario, el detalle de los productos y servicios suministrados, los precios unitarios, los precios totales, los descuentos y los impuestos.

Framework: Conjunto de APIs y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

Hardware: Responde a todas las partes físicas y tangibles de una computadora, sus componentes eléctricos, electrónicos, electromecánicos y mecánicos, sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

Implementación: Proceso por en el cual se escribe, se prueba, se depura y se mantiene el código fuente de un programa informático.

Logística: Conjunto de medios y métodos necesarios para llevar a cabo la organización y el control de una empresa, unidad o servicio.

Módulo: Parte de un sistema, que se instala y funciona por separado, integrándose con otros módulos con los que intercambia información.

Presupuesto: Previsión de gastos e ingresos para un determinado lapso, por lo general un año. Permite a las empresas, los gobiernos, las organizaciones privadas y las familias establecer prioridades y evaluar la consecución de sus objetivos.

Producto: Es cualquier objeto que puede ser ofrecido a un mercado que pueda satisfacer un deseo o una necesidad.

Servicio: Conjunto de actividades que buscan responder a las necesidades de un cliente, puede definirse también como el equivalente no material de un bien.

Software: Equipamiento lógico o soporte lógico de un computador digital, comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (Hardware).

Unidad presupuestada: Unidad que se le asignan presupuestos con el fin de logro, mantenimiento y progreso de un objetivo o meta a alcanzar.