
Universidad del Valle de Guatemala

Facultad de Ingeniería

# Prácticas Profesionales: Módulo de Gestión de Cuentas por Cobrar para el ERP IQ Profit del Sistema Nervioso Digital IQ Dimension

Taller de Prácticas Profesionales



Trabajo de investigación presentado por Hugo Roberto Sandoval Torselli para optar al grado de Licenciado de Ingeniería en Ciencias de la Computación

Guatemala

2006

Universidad del Valle de Guatemala

Facultad de Ingeniería

# Prácticas Profesionales: Módulo de Gestión de Cuentas por Cobrar para el ERP IQ Profit del Sistema Nervioso Digital IQ Dimension

Trabajo de investigación presentado por Hugo Roberto Sandoval Torselli para optar al grado de Licenciado de Ingeniería en Ciencias de la Computación

Guatemala

2006

Vo. Bo.:

E. _____

Ing. Luis Molina
Asesor

Tribunal:

(f) _____
Ing. Andrés Piñol

(f) _____
Ing. Luis Molina

(f) _____
MSc. María Mercedes Zaghi

Fecha de aprobación: 19 de junio del 2006

Guatemala, 21 de octubre de 2008

A: Hugo Roberto Sandoval Torselli

De: Lda. Silvia de Buratti

Ref.: Tesis

---

Por este medio le comunico que luego de cumplir con los requisitos estipulados para la elaboración de su tesis, puede iniciar los trámites para obtener su título.

Lo saluda, atentamente,

Lda. Silvia Ciudad-Real de Buratti
Col.7754

# PREFACIO

Toda empresa necesita mantener un control en la información que se maneja dentro de las distintas áreas que conforman a la misma, entiéndase contabilidad, bodega, ventas, etc. Hoy en día con la globalización y los avances en la tecnología de la informática las necesidades del cliente aumentan, deseando productos y servicios de una manera rápida manteniendo una buena calidad. Esto ocasiona que los procesos dentro de una empresa se agilicen para poder satisfacer las necesidades del cliente haciendo que sea más difícil mantener un control en todas las áreas de la empresa.

Muchas empresas han empleado las tecnologías digitales para crear sistemas que les permitan manejar la información fácil y rápidamente permitiendo así mantener un control. Sin embargo lo realizan de manera separada en cada área dentro de la empresa, es decir, cada área posee un sistema propio que maneja la información. Esto permite mantener control dentro de esa área, pero no garantiza mantener un control a nivel de todas las áreas de la empresa, ya que cada sistema puede llegar a manejar la información de manera diferente.

Con este problema surgen los sistemas de planificación de recursos de empresas (ERP) y el concepto del sistema nervioso digital que tienen como fin lograr integrar todas las áreas dentro de una empresa permitiendo así tener un control y lograr satisfacer las necesidades de los clientes.

Este trabajo muestra un informe sobre las prácticas realizadas en la empresa guatemalteca Infonet S.A. en la adaptación e implementación de un

ERP llamado IQ Profit que forma parte de un sistema nervioso digital denominado IQ Dimension.

**Agradecimientos:**

A mi familia:

Por haber estado siempre conmigo apoyándome no solo desde el inicio de mi carrera universitaria sino desde el inicio de mi vida. Sin ellos hubiera sido imposible estar donde me encuentro actualmente.

A mis amigos:

Por sus consejos que me han permitido seguir adelante y no desistir en alcanzar mis metas.

Al Ingeniero Luis Molina:

Por haberme dado la oportunidad de trabajar y brindarme su apoyo, amistad y conocimiento en la realización de este trabajo.

A la empresa Infonet:

Por haberme permitido realizar este trabajo de graduación dentro de sus instalaciones.

# INDICE GENERAL

# LISTA DE GRÁFICOS

# RESUMEN

Una empresa requiere brindar de un crédito a sus clientes para así poder atraer nuevos clientes y mantener los que ya tiene. Este trabajo tiene como finalidad el análisis y diseño del módulo para la gestión de las cuentas por cobrar que se integrará en un sistema de planificación de recursos de empresa llamado IQ Profit desarrollado por la empresa Software Intelligence Corp. El módulo se encarga de la creación y mantenimiento de los documentos de cargos (facturas y notas de débito) y sus respectivos documentos de abono (notas de crédito, anticipos, cobros, recibos, etc.,) automáticamente relacionando uno con el otro. De estar un módulo de contabilidad o bancos presente, el módulo automáticamente se encarga de generar la información de contabilidad o de bancos, dependiendo del caso. El módulo provee de un calendario para poder mantener un control de los cobros tanto a realizarse como también los que se encuentran vencidos.

# I. INTRODUCCIÓN

Actualmente muchas empresas agilizan sus procesos para cubrir las necesidades de sus clientes y poder mantenerse en la competencia del mercado. Esto causa que se sea más difícil tener control de toda la información que se maneja dentro de la empresa. Para poder remediar esto, las empresas decidieron emplear la tecnología para poder crear sistemas que logren administrar su información y así lograr controlarla. Sin embargo, realizaron sistemas para cada área dentro de la empresa, manipulando cada uno la información de manera diferente, logrando así un control, pero sólo dentro de esa área, no a nivel global de la empresa.

Debido a esto surgieron los sistemas de planificación de recursos de empresas, los cuales tienen como finalidad integrar todas las áreas dentro de una empresa en un solo sistema.

Una de estas áreas son las cuentas por cobrar, que comprenden el crédito que la empresa brinda a sus clientes. El sistema debe permitir a la empresa administrar los documentos de cargos (facturas y notas de débito) y documentos de abono (anticipos, recibos, cobros, notas de crédito, etc.,) de acuerdo a las políticas de crédito que la empresa defina. La empresa debe revisar el estado de los documentos de cargo y abono que genere a todo momento.

Este trabajo tiene como finalidad el análisis y diseño del módulo de cuentas por cobrar para un sistema de planificación de recursos llamado IQ Profit que está siendo desarrollado por la empresa Software Intelligence Corp.

El módulo cuenta con catálogos que definen la información a nivel de toda la empresa de las cuentas por cobrar. Una configuración que define información que deben emplear los procesos, como lo son información contable para crear la partida contable, números de serie, etc., toda esta información es manejada por vigencias. Y por último por procesos que administran las operaciones de las cuentas por cobrar.

## A. Metodología

El módulo fue desarrollado empleando un modelo de cascada. A continuación se describen los pasos que se realizaron:

1. Definición del Módulo. Para esto se recopilo información de libros de texto e Internet. Los puntos que se investigaron fueron los siguientes:

- Cuentas por cobrar
- Sistema de Planificación de Recursos de una Empresa
- Sistema Nervioso Digital

2. Análisis del Módulo. Descripción del problema. Definición del problema fundamental y lo que se requiere para resolverlo.

a. Descripción de la solución. Definición de la información que se necesita y la manera como la solución la administra. Para esto se emplearon diagramas de flujo.

3. Diseño del Módulo. Conforma los diseños del modelo de la base de datos y las definiciones de las clases, procedimientos e interfaz de usuario que el módulo debe emplear y como estos se deben adecuar a la arquitectura del software IQ Dimension.

4. Desarrollo e implementación. Conforma la realización del código fuente con base al análisis y diseño realizado anteriormente. Este paso también incluye la realización de pruebas, primero en un ambiente local del área de desarrollo y luego en otro ambiente con mayor similitud a uno real, en el área de pruebas, para la estabilización del módulo de cuentas por cobrar.

# II. MARCO TEÓRICO

## A. Sistema Nervioso Digital

El Sistema Nervioso Digital (SND) es una nueva opción para los negocios del siglo XXI. Es necesario que cada empresa agregue la riqueza de su información que manejan en función de sus intereses. Esto para que la información fluya a través de toda la organización y así se puede mejorar el rendimiento de la organización.

La finalidad del SND es motivar una reacción de las personas que conforman una organización para que se puedan implementar estrategias que hagan posible adaptarse al cambio y al uso del medio digital para lograr una comunicación más efectiva y directa al momento de realizar la toma de acciones. El SND se basa en un ciclo constante entre operaciones básicas de la organización, estrategias, respuestas a imprevistos y la interacción con los usuarios. <<Un SND comprende los procesos digitales que encadenan estrechamente todos los aspectos del pensamiento y acción de la empresa>> (Azócar, Ramón).

El SND permite que los procesos internos en una organización se realicen rápida y suavemente. A su vez permite a la organización responder rápidamente a la retroalimentación de sus clientes, reaccionar en su ambiente competitivo a tiempo y dar poder a los empleados con un conocimiento crítico.

La clave de todo esto está en la efectividad con que una organización puede manejar la fluidez de su información de manera digital. Toda la información (números, textos, audio, video, etc.) puede estar en formato digital. Con esto y la accesibilidad de hardware y software, las organizaciones deben redefinir la manera en como manejan sus procesos.

En la actualidad muchas empresas han comenzado realizando inversiones en información tecnológica, permitiendo obtener, archivas, compartir e interactuar con la información de manera innovadora, esto mediante el uso de intranets y redes globales como el Internet.

La idea detrás del SND es el empleo de la tecnología de manera que resalte la forma de trabajar de las personas y transforme la manera en que las organizaciones operan.

Deben existir procesos digitales dentro de las empresas que manejen y entreguen información a sus empleados, ayudándolos a responder de manera correcta y efectivamente ante los cambios y oportunidades, y hacer la toma de decisiones rápidamente para solucionar problemas en el momento que surjan.

El SND está constituido de cinco partes básicas:

1. Arquitectura de cómputo basado en Computadoras Personales (PCs). La industria de las computadoras personales es una de las más grandes hoy en día. Su competencia en el mercado ha ocasionado que se desarrollen maquinas más innovadoras, poderosas y de menor costo. Existen una variedad de tamaños y estilos a escoger.

2. Información en forma digital. Este es una de las premisas fundamentales de un SND. Toda la información debe estar en formato digital. No debe haber nada disponible en papel que no esté electrónicamente. La información cobra vida de manera digital, pudiendo ser analizada en forma creativa y encontrada, actualizada y compartida más fácil y rápidamente.

3. Correo electrónico universal. Los empleados pueden alternar el trabajo individual con el trabajo grupal. No importa la tarea específica que se realice, una aplicación de correo electrónico siempre es útil.

Que exista un correo electrónico universal ayuda a asegurar que todos los empleados estarán actualizados, sin importar la rapidez de los cambios en las condiciones del negocio.

4. Conectividad. Todos deben estar conectados entre sí. Además del correo electrónico se requiere de un sistema de redes, el cual permita conectar todos los elementos de su sistema e Internet, que permite unir los elementos con los usuarios.

El SND debe tener acceso a Internet, esto con el propósito de permitir a las compañías expandir sus sistemas existentes y aprovechar las ventajas del ambiente WEB.

5. Aplicaciones de negocios integradas. Los negocios necesitan de aplicaciones para realizar operaciones concretas, como la contabilidad, inventarios, sistemas de control de producción, bancos, etc.

Para poder implementar bien un SND, las aplicaciones del negocio deben hacer uso de los elementos explicados anteriormente (información en forma digital y correo electrónico universal). Esto permitirá que la información fluya mejor, pudiendo manejar, organizar y compartirla de una manera más sencilla.

## B. Sistemas de Planificación de Recursos de Empresas

Los Sistemas de Planificación de Recursos de Empresas o ERP según sus siglas en inglés ("Enterprise Resource Planning") son sistemas que integran todos los departamentos y funciones dentro de una empresa en un solo sistema que pueda servir a todas las necesidades particulares de la empresa.

Generalmente, cada departamento dentro de una empresa posee su propio sistema optimizado para trabajar de acuerdo a la forma en que realiza su trabajo. El objetivo del ERP es combinar estos sistemas en un solo sistema que

maneja una sola base de datos para que los diferentes departamentos puedan compartir información y comunicarse entre ellos de una manera más sencilla. Lograr esta integración puede representar una recompensa enorme para la empresa, si esta logra realizarla de manera correcta.

1. ¿Cómo puede el ERP optimizar una Empresa? Siendo el ERP un solo sistema que integra todos los departamentos de una empresa, las tareas de la empresa se realizan rápidamente. La información fluye entre los departamentos de forma automática, agilizando el tiempo en que se realizan los procesos dentro de la empresa.

Las personas en los diferentes departamentos tienen acceso a la misma información y pueden actualizarla. Cuando un departamento termina su respectivo proceso, el ERP automáticamente transmite la información generada al siguiente departamento para que éste realice su respectivo proceso. Para tener acceso al estado en que se encuentra el proceso, simplemente se ingresa al ERP y se realiza la búsqueda del proceso. Como la información es transmitida digitalmente entre los departamentos por el ERP, la velocidad con que se transmite es más veloz a comparación si se realizará de manera manual, siendo una persona quien lleve la información al departamento correspondiente.

Con un ERP, los departamentos de una empresa ya no están aislados unos de otros, ahora se encuentran entrelazados. Esto causa que los departamentos tengan acceso a mayor información y con esto, realizar mejores decisiones.

Por ejemplo, el departamento de ventas ya no solo genera órdenes, ahora posee información como el crédito de un cliente, la disponibilidad de cierto producto y la posibilidad de cuándo se pueda entregar una orden. El departamento de inventarios tiene que estar actualizando el sistema con la disponibilidad de productos, ya que de no hacerlo los demás departamentos no estarían enterados del estado actual en la disponibilidad de productos.

A las personas no les gusta cambiar y un ERP les exige que cambien la manera en que están realizando su trabajo. Es por eso que el valor de un ERP es muy difícil de determinar. El sistema en sí es menos importante, que los cambios que la empresa debe realizar en la manera en que realizan sus negocios. Si una empresa emplea un ERP para optimizar la manera en que se maneja las ordenes y manufactura, envío y facturación, la empresa verá un valor en el sistema. Si simplemente se instala el sistema y no hay ningún cambio dentro de la empresa, no se verá ningún valor, inclusive el nuevo sistema ocasionaría un menor desempeño ya que se esta cambiando el sistema que la empresa conocía por un nuevo que no se conoce.

2. ¿Qué cosas corrige un ERP dentro de una Empresa? Existen cinco razones por las que las empresas emplean un proyecto ERP:

a. Integración de la información financiera. Mientras se quiere determinar el desempeño general de una empresa, se pueden encontrar diferentes versiones. Las finanzas tienen su propia versión, las ventas tiene otra y los demás departamentos también generan sus propias versiones. El ERP crea una única versión en la que no se puede cuestionar ya que todos los departamentos emplean el mismo sistema.

b. Integración de la información de las órdenes de clientes. Un ERP se convierte en el lugar donde una orden está desde que se genera hasta el momento en que la orden es enviada desde envíos y finanzas envía la facturación. Teniendo esta información en un solo sistema, en lugar de varios sistemas que no se comunican entre ellos, las empresas pueden rastrear las órdenes fácilmente y coordinar los diferentes departamentos de manera simultánea.

c. Estandarización y agilización en los procesos de manufactura. El ERP provee de métodos estándar para automatizar algunos de los pasos en los procesos de manufactura. Estandarizando dichos procesos dentro de un solo sistema provoca reducción en tiempo e incremento en la productividad.

d. Reducción de inventario. El ERP ayuda a que los procesos de manufactura fluyan de mejor manera, mejorando la visibilidad de los procesos para cumplir una orden dentro de una empresa. Esto puede llevar a reducir inventarios de los materiales usados para realizar productos y ayudar a la empresa planear mejor las entregas hacia los clientes.

e. Estandarización de la información de Recursos Humanos. Un ERP puede arreglar que Recursos Humanos tenga métodos simples y unificados para rastrear tiempos de los empleados y comunicarse con ellos acerca de beneficios y servicios. Especialmente para empresas que cuentan de múltiples unidades de negocio.

En la carrera por resolver estos problemas, las empresas olvidan el hecho que los sistemas ERP son representaciones genéricas de la manera en que una empresa típicamente realiza negocios. Aunque muchos de los sistemas son entendibles, muchas empresas poseen su manera única de realizar ciertas cosas. Ocasionando disputas para cambiar el sistema base de los ERP para satisfacer sus necesidades únicas.

3. ¿Cuál es el costo oculto del ERP? Al momento de implementar un ERP hay ciertos costos que comúnmente no son tomados en cuenta o son sobrestimados. Entre estos costos están:

a. Capacitación. Este es uno de los costos más sobrestimados. Los costos en capacitación son altos ya que los usuarios deben aprender nuevos procesos, no solamente nuevas pantallas de interfaces. Las empresas que

desarrollan el ERP se enfocan en enseñar como utilizar el sistema, no educan a los usuarios en la manera en que la empresa donde trabajan realizan los negocios o explicar los diferentes procesos que se verán afectados por el ERP.

Con el ERP todos los departamentos emplean el mismo sistema, por lo que la información que un departamento ingrese afecta a los demás, por lo que ahora cada departamento debe tener un conocimiento más amplio sobre cómo los demás departamentos realizan sus procesos.

Invertir en la capacitación es una de las mejores cosas que se pueden hacer al momento de implementar un ERP.

b. Integración y pruebas. Realizar pruebas de conexión entre el sistema ERP y otros sistemas que la empresa posea con base a casos es otro costo que se sobrestima. Así como la capacitación, las pruebas a la integración del ERP deben ser realizadas desde una perspectiva orientada a procesos. Es recomendable que en lugar de utilizar datos ficticios se utilicen datos reales, por ejemplo, realizar una compra desde el comienzo, generando la orden, hasta el envío y facturación del mismo. Es preferible si los mismos empleados que realizan estos procesos participen en las pruebas.

c. Personalización. Es una de las cosas que más costo tiene, personalizar la base del sistema ERP. Esto ocurre cuando el sistema ERP no puede manejar uno de los procesos internos de la empresa y ésta decide arreglar el ERP para que pueda. La personalización puede afectar cada componente o módulo del ERP debido a que todos están estrechamente conectados entre ellos. Esto provoca que la actualización del ERP sea difícil ya que hay que realizar otra vez la personalización en la versión nueva. De ocurrir un error, el proveedor inicial del ERP no podrá ayudar y se tendría que contratar gente extra para mantener el sistema en orden.

d. Conversión de los datos. Cuesta dinero mover información de la empresa, como lo son registros de proveedores y clientes, información de productos, etc., del sistema antiguo al ERP.

e. Análisis de datos. Muchas veces la información del ERP debe ser combinada con información externa para poder realizar análisis. Usuarios con necesidades altas de análisis deben incluir dentro de su presupuesto el costo en el almacenamiento de datos y deben esperar hacer bastante trabajo para que todo funciones de manera correcta.

f. La implementación nunca termina. Muchas empresas tratan la implementación de un ERP como cualquier otro proyecto. Una vez que el sistema es implementado, se cree que el equipo que implemento el sistema se separa y cada quien regresará a su trabajo. Pero no es así, la gente que implementó el sistema es muy valiosa. Debido a que trabajaron muy de cerca con el ERP, ellos conocen más profundamente los procesos que la gente que realiza los procesos, por lo que es necesario que la gente que implementó el sistema permanezca para poder realizar análisis sobre el ERP.

g. Espera de recompensa. Muchas empresas esperan recibir ganancias al instante en que un sistema es implementado mientras que el grupo o empresa de desarrollo esperan un descanso. Ninguno de los dos aplica para un ERP. Muchos sistemas ERP no revelan su ganancia hasta que las empresas los emplean durante mucho tiempo y se concentran en hacer mejoras en los procesos que son afectados por el sistema. Y el grupo o empresa de desarrollo no tendrán descanso hasta que estas ganancias salgan.

h. Depresión Post-ERP. Los sistemas ERP generalmente causan problemas a las empresas cuando son instalados. Las razones más comunes para estos problemas es que todo se mira y trabaja de manera diferente a como se estaba haciendo anteriormente. Cuando la gente no puede hacer su trabajo de la

manera como lo estaban haciendo y no han aprendido bien la nueva manera, entran en pánico, y los problemas surgen.

**C. Cuentas por cobrar**

Las cuentas por cobrar conforman el crédito que las empresas brindan a sus clientes creándoles una cuenta.

Muchas empresas encuentran necesario ofrecer crédito para atraer nuevos clientes y mantener los que ya tiene. Esto genera las ventas al crédito, las cuales incluyen condiciones que estipulan el pago dado un determinado número de días. Dentro de una empresa, las cuentas por cobrar constituyen activos circulantes.

1. Políticas de crédito. Determinan la concesión y el monto de crédito que debe darse a un cliente. Dentro de las responsabilidades de una empresa no solo esta establecer estándares de crédito sino también su correcta utilización al momento de realizar decisiones de crédito.

Para tener una administración exitosa de las cuentas por cobrar, la empresa debe desarrollar fuentes adecuadas de información y métodos de análisis de crédito en sus políticas. De no tener en cuenta esto, las políticas de crédito no resultarían óptimas.

2. Estándares de crédito. Definen el criterio mínimo para que una empresa conceda crédito a sus clientes. Al realizar un análisis sobre los estándares de crédito, se deben tener en cuenta las siguientes variables fundamentales:

a. Gastos de oficina. Si se concede más crédito a los clientes, los estándares de crédito se vuelven más accesibles, esto ocasiona que los gastos de oficina aumenten, de manera contraria, si los estándares son poco accesibles los gastos de oficina disminuyen, pero la concesión de crédito es más difícil.

b. Inversión de cuentas por cobrar. El manejo de las cuentas por cobrar representa un costo para una empresa. Si la empresa posee un alto promedio de cuentas por cobrar, su manejo es más costoso. Las ventas al crédito y los cobros son dos factores que afectan a las cuentas por cobrar, ya que de existir muchas ventas al crédito, las cuales generan sus respectivos cobros, ocasionan un aumento de las cuentas por cobrar, de manera contraria, si las ventas al crédito disminuyen las cuentas por cobrar también disminuirán. Todo esto depende de la accesibilidad de los estándares de crédito establecidos.

c. Estimación de cuentas incobrables. Representa el aumento o disminución en la probabilidad o riesgo de que una cuenta se vuelva difícil de cobrar en proporción a la accesibilidad de los estándares de crédito, que son determinados acorde a los estudios sobre los clientes y su capacidad de pagar en corto y largo plazo.

d. Volumen de ventas. La accesibilidad de los estándares de crédito determina un aumento o disminución en las ventas al crédito, lo cual incide directamente con los costos e ingresos de la empresa y en la utilidad esperada.

3. Evaluación de estándares de crédito. Es necesario calcular el efecto que tengan los estándares de crédito en las utilidades marginales de las ventas y en el costo de la inversión marginal en las cuentas por cobrar para determinar la accesibilidad de los mismos.

a. Costo de la inversión marginal en cuentas por cobrar accesibles. Se calcula estableciendo una diferencia entre en costo de manejo de las cuentas por cobrar antes y después de la implantación de estándares de crédito accesibles.

Primero hay que calcular la razón financiera de promedio de cuentas por cobrar.

**Fórmula 1. Promedio de cuentas por cobrar**

| Promedio de CxC = Ventas anuales al crédito / Rotación de cuentas por cobrar |
|---|

Con esto se calcula la inversión promedio en cuentas por cobrar, siendo el porcentaje del precio de ventas que representan los costos de una empresa multiplicado por el promedio de cuentas por cobrar.

De último se obtiene el costo de la inversión marginal en cuentas por cobrar calculando la diferencia entre la inversión promedio en cuentas por cobrar con el programa presupuesto y el actual.

Este costo de la inversión marginal en cuentas por cobrar representa la cantidad de dinero adicional que una empresa debe comprometer a las cuentas por cobrar al hacer más accesibles los estándares de crédito.

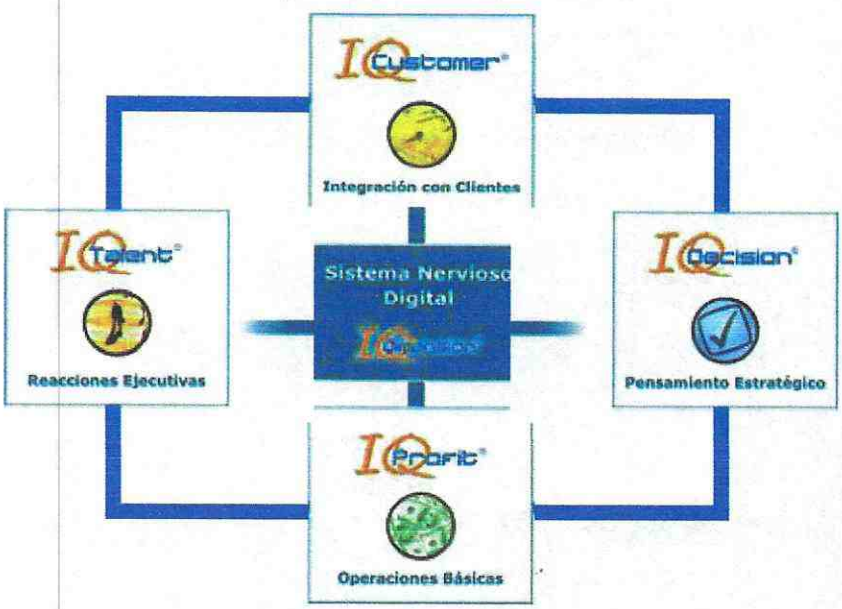
b. Toma de decisiones. Una vez comparados las utilidades marginales sobre las ventas con el costo de la inversión marginal en cuentas por cobrar, hay que decidir la accesibilidad de los estándares de crédito. Si las utilidades marginales son mayores a los costos marginales, hay que hacer más accesibles los estándares de crédito, de lo contrario no hay que hacer cambios a los estándares de crédito.

## D. IQ Dimension

Es un sistema nervioso digital que integra cuatros productos:

- IQProfit
- IQCustomer
- IQTalent
- IQDecision

Figura 1. Sistema IQ Dimension




Este producto es realizado por la empresa Software Intelligence Corp., la cual es representada principalmente por la empresa Infonet S.A., que fue fundada como empresa miembro de Grupo Tecnetron el 15 de marzo de 1996 por medio de capital privado con el objetivo de incursionar en Internet.

Infonet S.A., participa de la fundación y operación de ISP (Internet Service Provider) Infovía en Guatemala que se consolida como el ISP número uno en Guatemala y canal Platino de Microsoft. Posteriormente, Infonet adquiere las empresas Infoexpress S.A. y Novamática S.A., ambas empresas con más de 15 años de experiencia en el desarrollo y comercialización de productos de software en los sectores de telefonía, Internet y empresas de distribución y servicios.

1. IQProfit. Es una EPR que procesa y almacena las transacciones de la gestión financiera, de operaciones y comercial como la facturación, el pago a proveedores o la emisión de órdenes de compra.

IQ Profit está formado de los siguientes módulos, de acuerdo a la gestión que realizan:

Gestión financiera

- Módulo de contabilidad
- Módulo de presupuesto
- Módulo de consolidación
- Módulo de cuentas por cobrar
- Módulo de cuentas por pagar

Gestión de operaciones

- Módulo de inventario
- Módulo de compras
- Módulo de ventas

Gestión comercial

- Módulo de punto de venta

# III. ANÁLISIS

## A. Descripción del problema

Las cuentas por cobrar dentro de una empresa son las encargadas de establecer y manejar el crédito que se brinda a los clientes de la empresa. Estas cuentas son un factor muy importante en la adquisición de nuevos clientes y mantener a los ya existentes.

Dados los avances de la tecnología hoy en día en lo que refiere a la manipulación de la información, las empresas necesitan manejar su información de una manera más rápida y eficiente para poder ser competitiva ante las demás empresas. Ya que la empresa que maneje eficaz y ágilmente su información tendrá la mayor probabilidad de tener éxito sobre las demás. Las cuentas por cobrar no son la excepción, ya que si la empresa maneja correctamente su información de crédito sobre los clientes, estos estarán satisfechos y continuaran deseando los servicios y/o productos de la empresa.

Uno de los aspectos más difíciles de las cuentas por cobrar es llevar control sobre la aplicación de una política de crédito, como lo son las fechas en las cuales se tienen que realizar cobros a los clientes. Otro aspecto es arreglar cualquier problema en los cobros a los clientes o problemas con productos y/o servicios que ya han sido cobrados, con creaciones de notas de crédito, etc.

Todo esto obliga a las empresas a contar con herramientas de alta tecnología que puedan manejar su información ágil y eficazmente de una manera sencilla y fácil ante las personas que utilicen estas herramientas.

## B. Descripción de la solución

La empresa Software Intelligence está realizando un sistema nervioso digital llamado IQ Dimension para permitir a las empresas manejar, distribuir y compartir su información dentro de ella de manera rápida, empleando las herramientas más modernas hoy en día en el manejo de la información.

IQ Dimension está desarrollada sobre el .NET Framework de Microsoft en el lenguaje de programación Visual Basic .NET 2003. Este producto soporta las bases de datos SQL Server 2000 u Oracle 9x. IQ Dimension emplea programación basada en objetos para abstraer funcionalidades y características comunes dentro de sus componentes.

IQ Dimension cuenta con un ERP llamado IQ Profit el cual incluye un módulo para las cuentas por cobrar. Este módulo de cuentas por cobrar comprende lo siguiente:

1. Catálogos. Estos comprenden información general y básica que las cuentas por cobrar y otros módulos necesiten para realizar sus procesos y/o tareas.

   a. Cobradores. Define y maneja la información de los cobradores dentro de la empresa.

Los datos que conforman al catálogo de cobradores son:

- Código. determina un código de identificación para el cobrador. El código es único, por lo que dos cobradores no pueden poseer el mismo código. Este dato es obligatorio, por lo que no puede estar en blanco. Al momento de ingresar el código, ya sea en un registro nuevo o en una modificación de uno ya existente, se debe verificar que no exista ya en el sistema, de existir se debe obtener el registro entero asociado al código. Esto sirve para hacer la navegación entre

registro más sencilla. El código no puede exceder de dieciséis caracteres.

- Nombre. Define el nombre del cobrador. Es obligatorio por lo que no puede estar en blanco. No debe exceder de los ciento veintiocho caracteres.

- Descripción. Es una breve descripción del cobrador. No debe exceder de los doscientos cincuenta y seis caracteres. No puede quedar en blanco, es obligatorio.

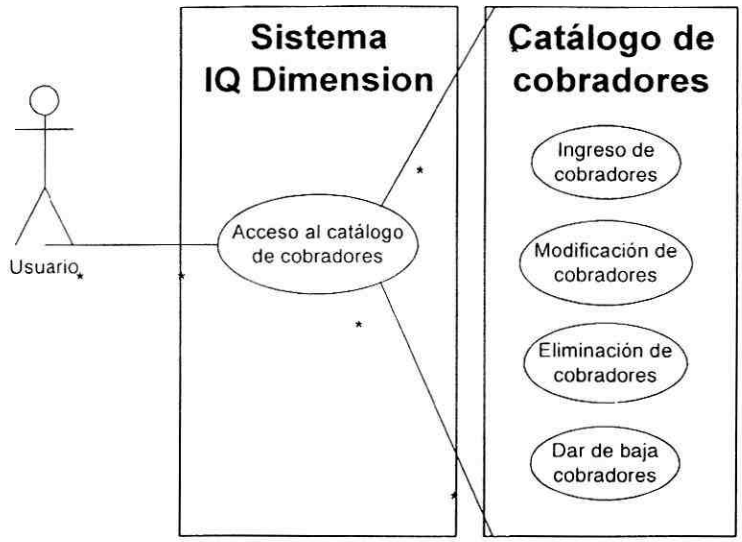
- Porcentaje de comisión. Determina el porcentaje de comisión que el cobrador recibirá. No puede ser mayor a cien ni menor a cero, debe ser de dos decimales. Este dato puede ser cero o quedar en blanco.

- Estado. Define el estado del cobrador, es un identificador que referencia a la tabla de estado del sistema. Este dato no es visible para el usuario, debe ser manejado por el sistema. Es visible al usuario el valor al que referencia en la tabla de estados del sistema.

- Fecha de baja. Representa la fecha en que el cobrador fue dado de baja. Este dato es asignado automáticamente por el sistema en el momento de dar de baja al cobrador y corresponde a la fecha actual que se realiza la operación.

- Usuario. Define el usuario que creó al cobrador. No es visible por el usuario. El sistema lo asigna automáticamente al momento de crear un nuevo registro.

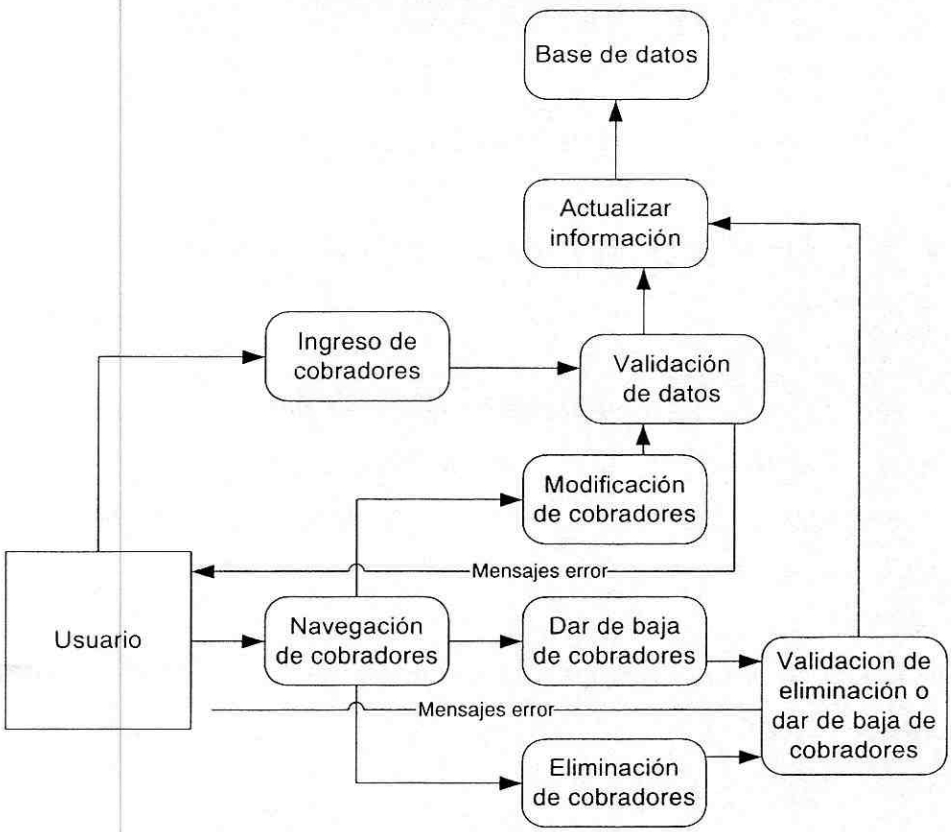
Figura 2. Caso de uso del catálogo de cobradores




Se puede ingresar un cobrador al sistema, modificarlo, eliminarlo o darlo de baja. Cuando sea el caso de la eliminación o dada de baja, se debe validar que el cobrador no sea referenciado por algún proceso en las cuentas por cobrar. Si un cobrador se encuentra en estado de baja, su información no puede ser modificada y no se puede eliminar al cobrador. Tanto la eliminación como el dar de baja deben solicitar una confirmación por parte del usuario para realizar la operación.

La navegación se realiza sobre los registros que se encuentren en estado de alta.

Figura 3. Diagrama de flujo del catálogo de cobradores




El sistema debe permitir la opción de generar un listado de los registros existentes para que el usuario pueda seleccionar uno e ir a ese registro. Así también debe permitir hacer una impresión de los registros existentes. Los datos que deben mostrar estas dos opciones son:

- Código
- Nombre
- Descripción
- Porcentaje comisión
- Estado
- Fecha baja

2. Configuración de cuentas por cobrar. Define datos específicos y/o formas de conducta al momento de utilizar las cuentas por cobrar, esto también aplica para otros módulos que utilicen las cuentas por cobrar. La configuración

está definida por agencia dentro de la empresa y por vigencias. La validación de la información también tiene que realizarse por agencia.

Los datos que conforman la configuración son:

- Fecha de inicio. Define la fecha de inicio en que tomará efecto la vigencia en la agencia.

- Fecha final. Define la fecha en que la vigencia ya no tomara efecto sobre la agencia. Este dato no es modificable por el usuario, al momento de ingresar una nueva vigencia, la fecha esta en blanco significando que es indefinida, al mismo momento el sistema debe calcular la fecha final de la vigencia anterior, siendo un día antes a la fecha inicial de la vigencia nueva. Al momento de eliminar una vigencia, si existe una vigencia anterior, la fecha final de la vigencia anterior se vuelve indefinida.

- Método de distribución. Determina el método de distribución que los procesos de cuentas por cobrar deben utilizar en sus operaciones. Este dato puede quedar en blanco.

- Serie factura. Define la serie que deben emplear las facturas de cuentas por cobrar.

- Serie de nota de crédito. Define la serie que se debe en las notas de crédito de cuentas por cobrar.

- Serie de nota de débito. Define la serie a emplear en las notas de débito de cuentas por cobrar.

- Serie recibos. Define la serie que deben emplear los recibos de cuentas por cobrar.

- Serie vales. Define la serie a utilizar en los vales de las cuentas por cobrar.

- Cuenta contable predeterminada para caja. Establece la cuenta contable que se debe emplear para la caja. Este dato puede estar en blanco.

- Cuenta puente pago a crédito. Establece la cuenta contable para utilizar en los pagos al crédito. Este dato puede quedar en blanco.

- Cuenta pago contra entrega. Establece la cuenta a utilizar en los pagos contra entregas. Este campo puede quedar en blanco.

- Devoluciones sobre ventas. Define la cuenta contable que se emplea en las devoluciones sobre ventas. Este dato puede quedar en blanco.

- Tipo tasa predeterminado. Define el tipo de tasa a emplear en las cuentas por cobrar. Este dato es obligatorio.

- Tipo partida para cargo. Define el tipo de partida contable que generan los documentos de cargos. Este dato es obligatorio.

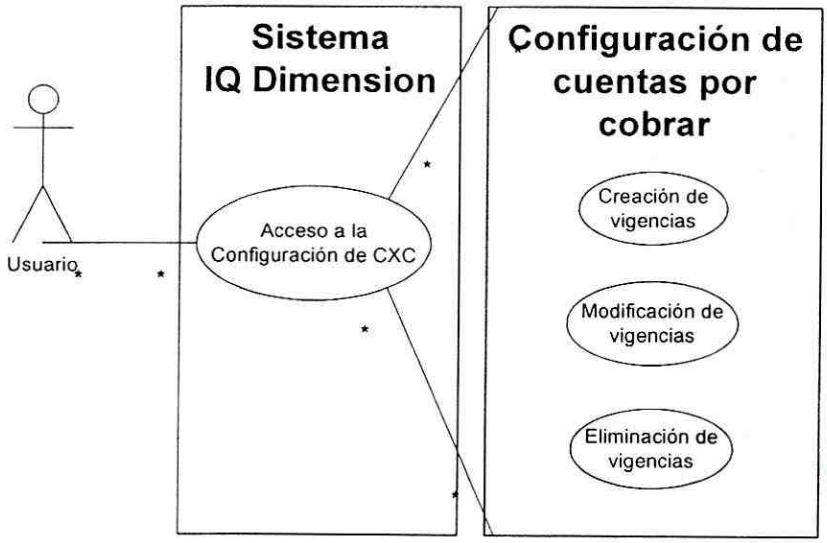
- Tipo partida para abono. Define el tipo de partida contable que generan los documentos de abonos. Este dato es obligatorio.

- Moneda. Define la moneda que manejaran las cuentas por cobrar. Este dato es obligatorio.

- Leyenda. Establece la leyenda a utilizar en reportes, en el caso que se necesite. Este dato puede quedar en blanco.

- Usuario. Indica el usuario que creo las vigencias. Este campo es obligatorio. No es visible ni modificable por el usuario, el sistema debe asignarlo automáticamente al momento que se creen vigencias.

Figura 4. Caso de uso de la configuración de cuentas por cobrar

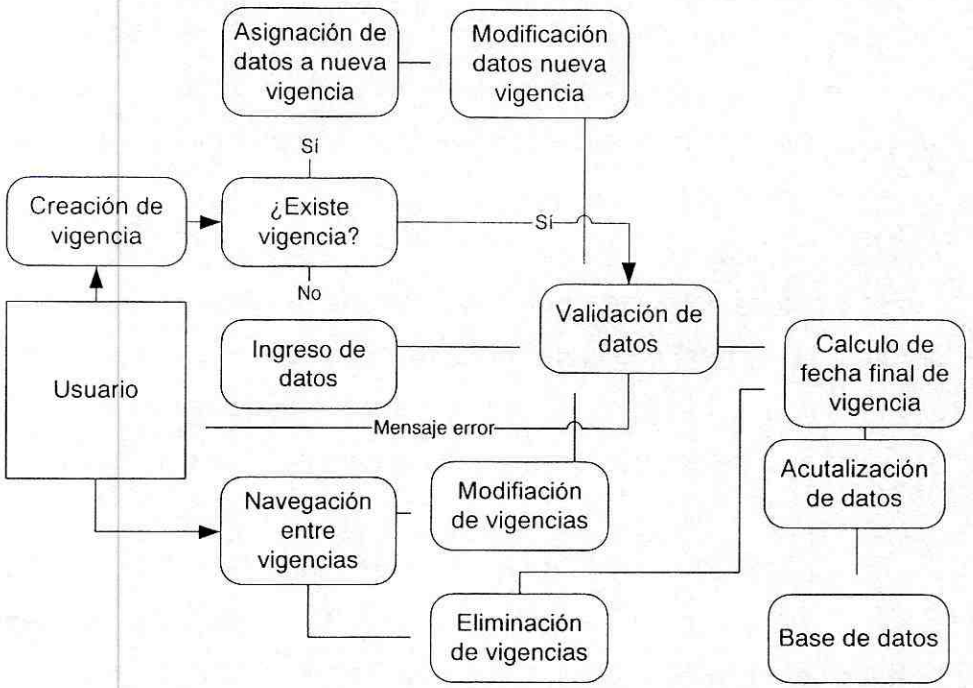



El usuario podrá crear nuevas vigencias solamente si los datos de la vigencia actual están correctos, de no estarlo no se deben permitir crear nuevas vigencias. Se debe validar que, por lo menos cada, agencia del sistema tenga creada una vigencia, de lo contrario no dejas guardar la configuración.

Al momento de crear una vigencia para una agencia, si ya existe una vigencia anterior para esa agencia, se debe copiar toda la información a la nueva agencia, esto para agilizar la creación de nuevas vigencias.

Se debe proporcionar una manera de navegar en las diferentes vigencias, tanto entre diferentes agencias, como dentro de ellas mismas.

Figura 5. Diagrama de flujo de la configuración de cuentas por cobrar




3. Procesos. La mayoría de los procesos de las cuentas por cobrar puede ser clasificada en dos grandes grupos:

- Procesos de cargos.
- Procesos de abonos.

a. Procesos de cargos. Dentro de estos procesos tenemos:

- Facturas.
- Notas de débito.

Los procesos de cargos deben manejar los siguientes datos:

- Agencia. Define la agencia en que se esta generando el cargo. Este dato no puede quedar en blanco. Al momento que se ingrese la agencia, se debe validar que para esa agencia y la fecha del cargo existe una configuración vigente en las cuentas por cobrar, de no existir no debe permitir al usuario emplear esa agencia. El sistema automáticamente, al momento de crear un nuevo registro, debe asignar la agencia actual.

- Serie. Determina la serie del cargo. Este campo puede quedar en blanco. El sistema, al momento de crear un nuevo registro, debe verificar si para la agencia y la fecha del cargo (dependiendo si es una factura o nota de débito) existe una serie definida en la configuración, de existir se debe asignar esa serie.

- Fecha. Define la fecha en que se esta realizando el cargo. Automáticamente al crear un nuevo registro se debe asignar la fecha actual del sistema. Al igual que la agencia, al momento de definir una fecha, se debe validar que exista una configuración vigente en cuentas por cobrar para la fecha y la agencia. Este dato es obligatorio.

- Número. Determina el número del cargo. El número puede ser alfa numérico. Este dato es obligatorio.

- Cliente. Define código del cliente del sistema al que se le aplica el cargo.

- Nombre razón social. El nombre al que se realiza el cargo al cliente. Si el cliente tiene definida esta información, se debe asignar automáticamente al seleccionar al cliente y no se debe permitir al usuario modificarlo, de lo contrario el usuario debe ingresarlo. Este dato es obligatorio.

- Dirección. Establece la dirección del cliente. Si el cliente tiene definida esta información, se debe asignar automáticamente al seleccionar al cliente y no se

debe permitir al usuario modificarlo, de lo contrario el usuario debe ingresarlo. Este dato es obligatorio.

- Identificación tributaria. Define el número de identificación tributaria del cliente. Si el cliente tiene definida esta información, se debe asignar automáticamente al seleccionar al cliente y no se debe permitir al usuario modificarlo, de lo contrario permitir al usuario ingresarlo. Este dato puede quedar en blanco.

- Vendedor. Define el vendedor del sistema responsable del cargo. Este campo no es obligatorio.

- Cobrador. Define el cobrador del sistema que se encargara de cobrar el cargo al cliente. Este campo no es obligatorio.

- Moneda. Determina la moneda en que se realiza el cargo. Este no lo puede editar el usuario, se debe asignar automáticamente obteniéndola del cliente.

- Tasa. Determina la tasa de conversión entre la moneda del cargo y la moneda de la empresa. Al momento en que la moneda del cargo cambie, se debe actualizar la tasa. Si la moneda del cargo y la moneda de la empresa son la misma, la tasa es uno y el usuario no puede modificarlo, De lo contrario el usuario puede modificar la tasa. Este dato es obligatorio.

- Descripción. Define una descripción del cargo. Este dato es obligatorio.

- Afecta libro. Es un dato de valor S o N y determina si el cargo aparece el reporte del libro de venta o no.

- Subtotal. Es el subtotal del cargo. Este dato se calcula automáticamente y debe ser mayor a cero.

- Subtotal base. Corresponde al subtotal del cargo en la moneda de la empresa. No es visible para el usuario y se calcula automáticamente, siendo el subtotal por la tasa. Debe estar redondeado a dos decimales.

- Impuestos. Equivale a la suma de impuestos aplicados al cargo. Este dato se calcula automáticamente.

- Impuestos base. Corresponde a los impuestos del cargo en la moneda de la empresa. No es visible para el usuario y se calcula automáticamente, siendo los impuestos por la tasa. Debe estar redondeado a dos decimales.

- Total por cobrar. Es el total del cargo. Se calcula automáticamente, siendo la suma entre el subtotal y los impuestos.

- Total por cobrar base. Corresponde al total por cobrar del cargo en la moneda de la empresa. No es visible para el usuario y se calcula automáticamente, siendo el total por cobrar por la tasa. Debe estar redondeado a dos decimales.

- Saldo por cobrar. Define el saldo pendiente a cobrar del cargo. Este dato se calcula automáticamente siendo el total por cobrar menos lo que se haya cobrado ya de la factura. Este dato debe ser mayor a cero.

- Condiciones de cobro. Define la condición de cobro del sistema que se aplica para cobrar el cargo. Este dato es obligatorio.

- Partida. Muestra datos de la partida contable generada a partir del cargo, si el módulo de contabilidad se encuentra presente, este dato sólo es para mostrarse. Se debe mostrar el año, mes y número de la partida.

- Impreso. Indica si el cargo está impreso o no.

- Fecha baja. Fecha que indica cuándo fue el cargo anulado. Se asigna automáticamente por el sistema momento de anular.

- IdPartida. Este dato no es visible para el usuario. Hace referencia a la partida contable que se generó a partir del cargo. Este campo puede estar en blanco, debido a que no está presente el módulo de contabilidad. A través de este dato se debe obtener la información de la partida, de existir.

- IdTipoCargo. Este dato no es visible para el usuario. Hace referencia a la tabla de tipos de cargo del sistema. Determina el tipo de cargo que es el registro, puede ser factura o nota de débito. Este dato es obligatorio y el sistema debe asignarlo automáticamente al momento que se realiza un nuevo registro.

- Usuario. Este dato no es visible para el usuario. Define el usuario que génerо el registro de cargo. Debe ser manejado automáticamente por el sistema.

- IdEstado. Este dato referencia a la tabla de estados del sistema, indica el estado del cargo. No es modificable por el usuario, debe manejarse por el sistema dependiendo de la operación que se realice sobre el cargo.

El sistema debe permitir la opción de generar un listado de los registros existentes para que el usuario pueda seleccionar uno e ir a ese registro. El listado debe estar filtrado de acuerdo al tipo de cargo. Los datos que debe mostrar el listado son:

- Agencia
- Serie
- Número
- Estado
- Fecha
- Código del cliente

- Nombre del cliente
- Moneda
- Tasa
- Total a cobrar

La condición de cobro del cargo debe contar con un detalle que especifica las fechas en que se deben efectuar los cobros y el monto que se debe cobrar. Se debe dar la opción de agregar y eliminar líneas al detalle. La información que comprende cada línea es:

- Descripción. Da una breve información sobre la línea del detalle de condición de cobro. Este dato es obligatorio y no puede haber más de una línea con la misma descripción.

- Porcentaje. Determina el porcentaje sobre el total del cargo que conforma la línea. Si el usuario ingresa un total de la línea, el porcentaje debe calcularse automáticamente, siendo el total de la línea dividido por el total del cargo multiplicado por cien, debe estar redondeado a seis decimales. El porcentaje no puede ser mayor a cien, y la suma del porcentaje de todas las líneas del detalle de la condición de cobro debe ser igual a cien. Este dato es obligatorio.

- Total. Define el total que conforma la línea del detalle. Este dato no puede ser mayor al total del cargo. La suma del total de todas las líneas del detalle debe ser igual al total del cargo. Si el usuario ingresa primero el porcentaje, el total debe calcularse automáticamente siendo el total del cargo por el porcentaje dividido cien, esto redondeado a dos decimales. Este dato es obligatorio.

- Fecha vencimiento. Determina la última fecha en que la línea del detalle puede ser cobrado. Este dato es obligatorio. No puede ser menor a la fecha de cobro de la línea.

- Fecha cobro. Define la fecha en que la línea del detalle de la condición debe ser cobrada. Este dato es obligatorio. No puede ser mayor a la fecha de vencimiento.

- Cobrado. Este dato puede ser S o N. No es modificable por el usuario, únicamente se muestra. Muestra si la línea del detalle de la condición fue cobrado o no. Se actualiza al momento que se realiza un abono.

- Usuario. Define el usuario que creo la línea del detalle de la condición. Este dato es obligatorio y no se muestra al usuario, debe ser manejado automáticamente por el sistema al momento de crear la línea.

Un cargo debe contar con un detalle de rubros. Se debe proveer con la opción de agregar y eliminar líneas del detalle. Cada línea del detalle debe incluir la siguiente información:

- Rubro. Rubro del sistema al que corresponde a la línea del detalle. El rubro es único, por lo que no debe haber más de una línea con el mismo rubro. Este dato es obligatorio.

- Clasificación libro. Define a que clasificación del libro de ventas del sistema corresponde la línea del detalle. Este dato es obligatorio.

- Subtotal. Define el subtotal de la línea. Este dato es ingresado por el usuario, debe ser mayor a cero. El subtotal del cargo se calcula a partir de la suma de las líneas del detalle.

- Impuestos. Determina el total de impuestos que se aplican a la línea del detalle. Este dato no es modificable por el usuario, el sistema lo calcula automáticamente. Los impuestos del cargo se calculan a partir de la suma de los impuestos de las líneas del detalle.

- Total. Define el total de la línea. Este dato no es modificable por el usuario. Se calcula automáticamente siendo la suma entre el subtotal y los impuestos de la línea. Este dato debe ser mayor a cero.

- Cuenta contable. Determina la cuenta contable del sistema que se aplica a la línea del detalle. Este dato es obligatorio.

- Subtotal base. Define el subtotal de la línea expresado en la moneda de la empresa. Este dato no es visible para el usuario, se debe calcular automáticamente, siendo el subtotal de la línea por la tasa de cargo, redondeado a dos decimales.

- Impuestos base. Define los impuestos del detalle expresados en la moneda de la empresa. Este dato no es visible para el usuario, se debe calcular automáticamente, siendo los impuestos de la línea por la tasa de cargo, redondeado a dos decimales.

- Total base. Es el total de la línea expresado en la moneda de la empresa. No es visible para el usuario. Equivale al total de la línea por la tasa del cargo, redondeado a dos decimales.

- Total recibo. Define el total que se ha aplicado a la línea en recibos (abonos). Este dato no es visible para el usuario. Se calcula al momento que se realiza el recibo sobre la línea del detalle. Esto se realiza a través de triggers en la base de datos.

- Total recibo base. Corresponde al total de recibo de la línea expresado en la moneda de la empresa. Se calcula automáticamente a través de triggers, siendo el total de recibos por la tasa del cargo.

- Total cobrado. Indica el total que se ha cobrado a la línea. Este dato no es visible al usuario. Se calcula automáticamente por el sistema al momento que se realiza un cobro (abono). Esto se realiza a través de triggers en la base de datos. El total a cobrar del cargo menos la suma del total cobrado de las líneas del detalle debe ser igual al saldo por cobrar del cargo.

- Total cobrado base. Es el total cobrado de la línea expresado en la moneda de la empresa. No es visible para el usuario. Es el total cobrado de la línea por la tasa del cargo redondeado a dos decimales. Se calcula a través de triggers en la base de datos.

- Usuario. Define el usuario del sistema que generó la línea del detalle. No es visible para el usuario. El sistema debe asignarlo automáticamente al momento de crear la línea.

Así también cada línea del detalle de rubros cuenta con un detalle de impuestos y un detalle de centros de costo. Para cada uno se proveer la opción de agregar y eliminar líneas.

Cada línea de detalle de impuestos debe llevar los siguientes datos:

- Impuesto. Define qué impuesto dentro del sistema se va aplicar al cargo. Este dato es obligatorio. No puede haber más de una línea con el mismo impuesto.

- Porcentaje. Determina qué porcentaje del subtotal de la línea de rubro se aplica para la línea. Este dato es obligatorio y su valor no puede ser mayor a cien. Si el usuario ingresa primero el monto del impuesto, el porcentaje debe calcularse automáticamente, siendo el monto del impuesto dividido el total cálculo por cien, redondeado a seis decimales.

- Total cálculo. Este dato sólo se muestra al usuario, no es modificable. Define la base sobre la cual se calcula el impuesto, corresponde al subtotal de la línea del detalle de rubro. El sistema debe asignarlo automáticamente al momento de crear la línea del detalle de impuestos.

- Total impuesto. Es el total del impuesto que se aplica a la línea del detalle de rubros. Este dato es obligatorio y debe ser mayor a cero. Si el usuario ingresa primero el porcentaje, el monto debe calcularse automáticamente, siendo el total cálculo por el porcentaje dividido cien, redondeado a dos decimales.

- Total impuesto base: Es el total del impuesto expresado en la moneda de la empresa. No es visible por el usuario, el sistema debe asignarlo automáticamente, siendo el total del impuesto multiplicado por la tasa del cargo, redondeado a dos decimales.

- Usuario. Define al usuario que creó la línea de detalle de impuestos. Este dato no se muestra al usuario. El sistema debe asignarlo automáticamente al momento que se cree la línea.

Ahora las líneas del detalle de centros de costo tienen la siguiente información:

- Centro de costo. Indica el centro de costo del sistema que describe la línea del detalle. Este dato es obligatorio. No puede haber más de una línea con el mismo centro de costo.

- Porcentaje. Indica el porcentaje sobre el subtotal de la línea del detalle de rubros que corresponde al centro de costo. Este dato es obligatorio y su valor no puede ser mayor a cien. Si el usuario ingresa primero el subtotal de la línea de centro de costo, el porcentaje debe calcularse automáticamente, siendo el subtotal de la línea dividido el subtotal de de la línea del detalle de rubros por cien,

redondeado a seis decimales. La suma del porcentaje de todas las líneas de centros de costo debe ser igual a cien.

- Subtotal. Define el monto que corresponde a la línea. Este dato es obligatorio y su valor no puede ser mayor al subtotal de la línea del detalle de rubros. Si el usuario ingresa primero el porcentaje, el subtotal debe ser calculado automáticamente, siendo el subtotal de la línea del detalle de rubros multiplicado por el porcentaje dividido por cien, redondeado a dos decimales. La suma del subtotal de todas las líneas debe ser igual al subtotal de la línea del detalle de rubros.

- Subtotal base. Es el subtotal de la línea de centro de costo expresado en la moneda de la empresa. No se muestra al usuario, el sistema debe calcularlo automáticamente, siendo el subtotal de la línea por la tasa del cargo, redondeado a dos decimales.

- Usuario. Define al usuario que creo la línea de centro de costo. No se muestra al usuario, el sistema debe asignarlo automáticamente al momento de crear la línea.

Figura 6. Caso de uso de facturas de cuentas por cobrar

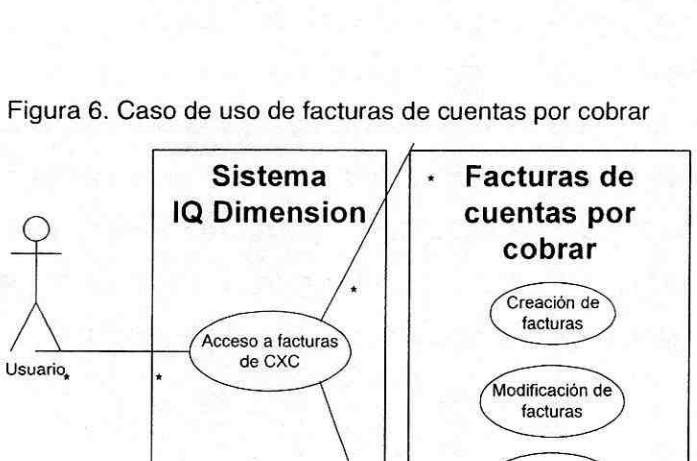
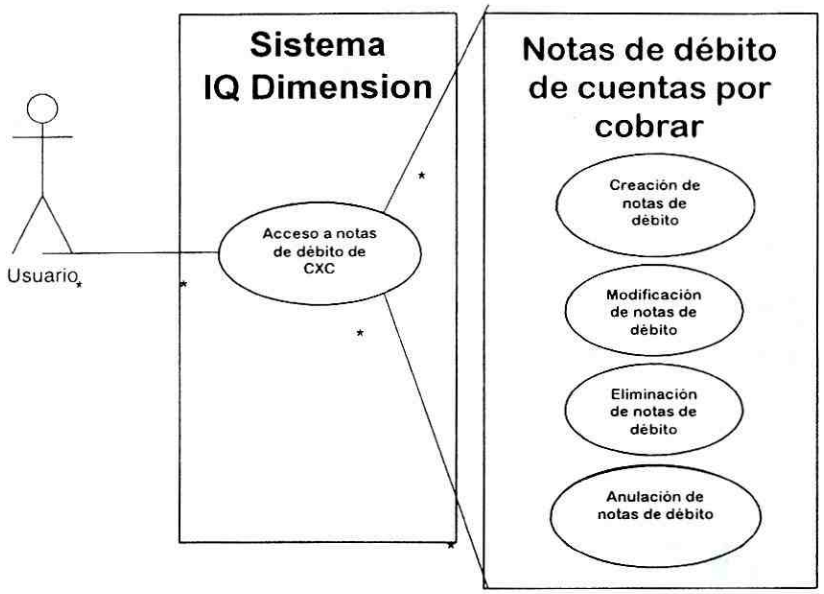


Figura 7. Caso de uso de notas de débito de cuentas por cobrar



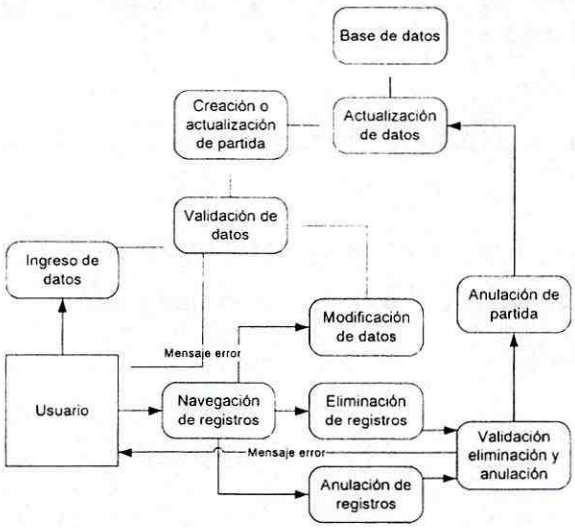

Si el módulo de contabilidad está presente en el sistema, el cargo debe generar una partida contable. La descripción de la partida corresponde al tipo del cargo (F para factura y ND para nota de débito) seguido de la serie y número del cargo, luego sigue el código del cliente. La línea del detalle de cargo de la partida es el total por cobrar del cargo y la cuenta contable por cobrar asociada el cliente. Las líneas de abono de la partida son el detalle de rubros y el detalle de impuestos del cargo de cuentas por cobrar y se emplean las cuentas contables asignadas en las líneas de detalle de rubros y las cuentas contables asociadas a los impuestos. No pueden haber líneas repetidas, por lo que si ya existe una línea con una cuenta contable, esta debe actualizarse y no crear otra línea nueva. El tipo de partida contable que genera se debe obtener de la configuración de cuentas por cobrar, obteniendo el tipo de partida para cargos para la agencia y fecha del documento de cargo.

Para modificar, anular o eliminar un cargo en cuentas por cobrar, se debe validar que, de tener una partida asociada, no esté actualizada y no se haya realizado ningún registro de abono (cobro, nota de crédito, etc.) sobre el cargo.

Al momento de anular o eliminar el usuario debe confirmar la realización de la operación. Si el cargo tiene asociado una partida, debe anular primero la partida.

La navegación de los registros debe ser de los registros que se encuentren en estado de alta. Debido a que los registros cuentan con un tipo de registro de cargo, la navegación también debe realizarse acorde a este tipo, dependiendo del cual sea el caso. Es decir, si son facturas, sólo navegar registros de tipo factura y de estado de alta, lo mismo con las notas de débito.

Figura 8. Diagrama de flujo de los procesos de cargos




b. Procesos de abono. Dentro de estos procesos tenemos:

- Anticipos
- Notas de crédito
- Notas de crédito sobre documentos cobrados

Los procesos de abonos en cuentas por cobrar deben manejar la siguiente información:

- Agencia. Define la agencia dentro de la empresa en que se realiza el registro de abono. Al momento de crear un nuevo registro, el sistema debe asignar la

agencia actual. Al momento que el usuario ingrese una agencia, se debe validar que para esa agencia y la fecha del registro de abono, exista una configuración de cuentas por cobrar vigente. Este dato es obligatorio.

- Fecha. Indica la fecha en que se realiza el registro de abono. Este dato es obligatorio. Al momento que el usuario ingrese una fecha, se debe validar que para esa fecha y agencia del registro, exista una configuración de cuentas por cobrar vigente.

- Cliente. Define el código del cliente del sistema al que se aplica el registro de abono. Este dato es obligatorio.

- Número. Indica el número del registro de abono. Este dato es obligatorio.

- Moneda. Determina la moneda que maneja el registro de abono. Este dato no es modificable por el usuario. El sistema debe asignar automáticamente la moneda del cliente.

- Tasa. Indica la tasa de conversión de la moneda del registro de abono a la moneda de la empresa. Se calcula automáticamente al momento que se asigna la moneda al abono. Si la tasa es uno, no se debe permitir modificarla, de lo contrario el usuario puede modificar la tasa. Este dato es obligatorio y debe ser mayor a cero.

- Descripción. Indica una pequeña descripción sobre el registro de abono. Este dato es obligatorio.

- Nombre o razón social. Es el nombre del cliente a que se emite el registro de abono. Este dato es obligatorio. Si el cliente tiene definida esta información, se debe asignar automáticamente y el usuario no puede modificarla, de lo contrario el usuario debe asignar esta información.

- Dirección. Corresponde a la dirección actual del cliente. Este dato es obligatorio. Si el cliente tiene definida esta información, el sistema debe asignarlo automáticamente y el usuario no puede modificarla, de lo contrario el usuario debe asignar la información.

- Identificación tributaria. Corresponde al número de identificación tributaria del cliente. Si el cliente tiene definida esta información se debe asignar automáticamente y no puede ser modificada por el usuario, de lo contrario el usuario puede ingresar esta información.

- Partida. Representa la información sobre la partida contable generada a partir del registro de abono, si esta se generó. Se debe mostrar el año, mes y número de la partida.

- Total. Determina el total por el que se realizara el abono. Este dato debe ser mayor a cero.

- Total base. Corresponde al total expresado en la moneda de la empresa. Se calcula multiplicando el total del abono por la tasa del abono, redondeado a dos decimales.

- IdAbonoTipo. Identifica a una tabla de tipos de abono del sistema, en este campo se define el tipo de proceso de abono del registro. El sistema debe asignarlo automáticamente al momento de realizar un registro nuevo. Este dato es obligatorio.

- Estado. Identifica a la tabla del estado del sistema y determina el estado del registro de abono. Este dato es obligatorio.

- Fecha baja. Indica la fecha en que el registro de abono fue anulado. El sistema debe asignarlo automáticamente al momento de anular el registro.

- Usuario. Define el usuario que realizó el registro de abono. Se debe asignar automáticamente al momento de realizar un nuevo registro.

El sistema debe permitir la opción de generar un listado de los registros existentes para que el usuario pueda seleccionar uno e ir a ese registro. El listado debe estar filtrado de acuerdo al tipo de abono. Los datos que debe mostrar el listado son:

- Agencia
- Serie documento
- Número documento
- Fecha
- Código del cliente
- Nombre del cliente
- Moneda
- Tasa
- Total

Los procesos de abonos tienen un detalle de cargos en el cual se define a cuáles de ellos se aplica el abono, dicho detalle de cargos no se guarda en la base de datos como relación, por lo que debe ser derivada lógicamente por el proceso de abono. Al momento de definirse un cliente para el abono, el sistema debe obtener todos los registros de cargos asociados al cliente que tengan un saldo por cobrar (el saldo sea mayor a cero). De cada registro de cargo se debe mostrar la siguiente información:

- Tipo de cargo. Indica el tipo de cargo del registro, factura o nota de débito. Este dato no es modificable por el usuario.

- Serie. Indica la serie del cargo. No es modificable por el usuario.

- Número. Indica el número del cargo. No es modificable por el usuario.

- Fecha. Define la fecha de realización del cargo. No es modificable por el usuario.

- Total. El total por cobrar del cargo. No es modificable por el usuario.

- Saldo por cobrar. Indica el saldo pendiente por cobrar actual del cargo. No es modificable por el usuario.

- Total abono. Indica la cantidad total a abonar al cargo. Este dato no puede ser mayor al saldo por cobrar del cargo. Si el saldo por cobrar es diferente al total por cobrar, se debe calcular automáticamente el total abono, siendo el total por cobrar menos el saldo por cobrar, redondeado a dos decimales.

- Nuevo saldo. Determina el nuevo saldo que tendrá el cargo al momento de aplicarse el abono. No es modificable por el usuario, de be asignarse automáticamente al momento de definir el total abono, siendo el saldo por cobrar menos el total abono, redondeado a dos decimales.

El abono se aplica a nivel del detalle de rubros, por lo que para cada cargo al que se asigne un total abono se debe construir un detalle de rubros del abono, el cual se crea a partir del detalle de rubros del cargo. Cada línea del detalle de rubros del abono debe mostrar la siguiente información:

- Rubro. El rubro que define a la línea del detalle de rubros del cargo. No es modificable por el usuario.

- Total. Es el total de la línea del detalle de rubros del cargo. No es modificable por el usuario.

- Abono. Es la cantidad que se abona en la línea del detalle. No puede ser mayor al total de la línea ni mayor al total abono de la línea de detalle de cargos. La suma de todas las líneas del detalle de rubros debe ser igual al total abono de la línea del detalle de cargos.

- Abono base. Corresponde al abono expresado en la moneda de la empresa, es el abono multiplicado por la tasa del abono, redondeado a dos decimales. No es visible por el usuario.

- Nuevo saldo. Corresponde al saldo al momento de aplicar el abono. No es modificable por el usuario, se calcula automáticamente al momento de definir cantidad de cobro, corresponde al total menos el cobro, redondeado a dos decimales.

- Impuesto. Corresponde al total de impuestos de la línea del detalle de rubro del cargo. Este dato no es modificable por el usuario. Corresponde al total impuesto de la línea del detalle de rubros del cargo.

- Impuesto base. Es el impuesto expresado en la moneda de la empresa. No es visible por el usuario, se calcula automáticamente, siendo el impuesto multiplicado por la tasa del abono, redondeado a dos decimales.

- No. aplicación. Define el número de aplicación anticipo asociado a la línea del detalle de rubros del cargo, de existir. Este dato no es modificable por el usuario.

- Fecha aplicación. De existir un anticipo asociado a la línea del detalle de rubros del cargo, este campo muestra la fecha en que fue realizado. No es modificable por el usuario.

- Usuario. Define al usuario que generó la línea. Este dato no es visible para el usuario, debe ser asignado por el sistema al momento de crear la línea.

- IdCargoRubro. Este caso hace referencia a la línea del detalle de rubros del cargo al que afecta la línea del detalle de rubros del abono. Se asigna automáticamente al momento de crear el detalle de rubros del abono. No es visible por el usuario.

De igual forma, cada línea del detalle de rubros de abono cuenta con un detalle de impuestos. La información que debe mostrar este detalle es:

- Impuesto. Define el impuesto del sistema que corresponde a la línea del detalle.

- Porcentaje. Define el porcentaje del impuesto que se aplica sobre el total de la línea de detalle de rubros. Debe estar redondeado a seis decimales.

- Total impuesto. Corresponde al total que aplica el impuesto. Debe estar redondeado a dos decimales.

- Total impuesto base. Corresponde al total impuesto expresado en la moneda de la empresa. Es el total impuesto multiplicado por la tasa del abono, redondeado a dos decimales. No es visible por el usuario.

- Porcentaje sustraendo. Indica el porcentaje que se sustraerá del total impuesto.

- Total sustraendo. Corresponde la cantidad que se sustraerá del total impuesto.

- Usuario. Define el usuario que creo la línea. No es visible para el usuario. Debe ser asignado por el sistema al momento de crear la línea.

Al momento que el usuario define un total de abono en el detalle de cargos, el sistema debe generar el detalle de rubros y de impuestos. Si el total de abono es igual al total por cobrar del cargo, se debe asignar a cobro el valor de total de la línea de rubro. Si las cantidades son distintas y el detalle de rubros solo cuenta con una línea, se debe asignar a cobro los que se haya definido en total abono. De contar con más de una línea, el usuario debe definir la distribución del total abono. Si el usuario modifica el total abono se debe actualizar el detalle de rubros. Si el usuario modifica el total abono y este tiene valor cero, se deben eliminar los detalles de rubros e impuestos.

Esto conforma la información global que manejan los procesos de abono, sin embargo, cada proceso administra la información de manera diferente, y a parte de esta información cuentan con más datos.

1) Notas de crédito. La información adicional que cuenta las notas de crédito es:

- Afecta libros. Este dato tiene valor S o N y define si el registro aparecerá en el reporte del libro de ventas. No puede quedar en blanco.

- Afecta inventarios. Determina de que manera la nota de crédito afecta los inventarios asociados, de tener, del cargo. Este dato puede tener tres valores. No afecta, indicando que no tiene efecto sobre los inventarios la nota de crédito. Afecta cantidad, la nota de crédito afecta la cantidad de inventarios del cargo. Y afecta total, en donde la nota de crédito tiene efecto sobre el total de cada inventario del cargo. Este dato es obligatorio.

- Serie. Define la serie de la nota de crédito. Se debe buscar si en la configuración de cuentas por cobrar, para la agencia y fecha, existe una serie para notas de crédito definida, de existir se debe asignar. La serie y el número de la nota de crédito son únicos, no puede haber más de un registro con la misma serie y

número, de existir, se debe navegar hacia ese registro y obtener toda su información. La serie puede quedar en blanco.

- Subtotal. Corresponde al monto de la nota de crédito sin haberse sumado los impuestos. Este dato debe ser mayor a cero.

- Total impuestos. Corresponde al total de impuestos que se aplican a la nota de crédito. Este dato puede ser cero.

El total del abono se calcula sumando el subtotal más el total impuestos. En las notas de crédito existe una validación extra para la agencia. Se tiene que validar que para la configuración de cuentas por cobrar de esa agencia y fecha, esté definida una cuenta para las devoluciones de ventas.

El detalle de impuestos de cada línea del detalle de rubros se calcula automáticamente a partir del detalle de impuestos del cargo asociado a la línea del detalle de rubro. Este detalle no es modificable por el usuario, por lo que todos sus datos sólo se muestran. El detalle se debe generar al momento de generar el detalle de rubros del abono.

En las notas de crédito, el detalle de rubros cuenta aparte con un detalle de inventarios. Este detalle lleva la siguiente información:

- Código del inventario. Indica el código del inventario, este dato no es modificable por el usuario.

- Descripción del inventario. Corresponde a la descripción del inventario. El usuario no puede editar este dato.

- Presentación. Corresponde a la presentación que maneja el inventario. El usuario no puede editar este dato.

- Código de la medida. Corresponde al código de la medida que maneja el inventario. El usuario no puede editar este dato.

- Descripción de la medida. Corresponde a la descripción de la medida del inventario. El usuario no puede editar este dato.

- Precio. Define el precio del inventario. El usuario no puede editar este dato.

- Cantidad despachada. Define la cantidad que será retornada por efecto de la nota de crédito. Si la nota de crédito define que afecta inventarios en cantidad, este dato es modificable por el usuario. De ser este el caso, el total debe ser calculado, siendo la cantidad despachada por el precio, redondeado a dos decimales.

- Total. Define el total afectado por la nota de crédito. El valor no puede ser mayor al abono de la línea de rubro. Si la nota de crédito afecta total, este dato es modificable por el usuario. De ser así, la cantidad despachada debe ser calculada, siendo el total divido el precio, redondeado a seis decimales. La suma del total de las líneas del detalle de inventarios debe ser igual al abono de la línea de rubro.

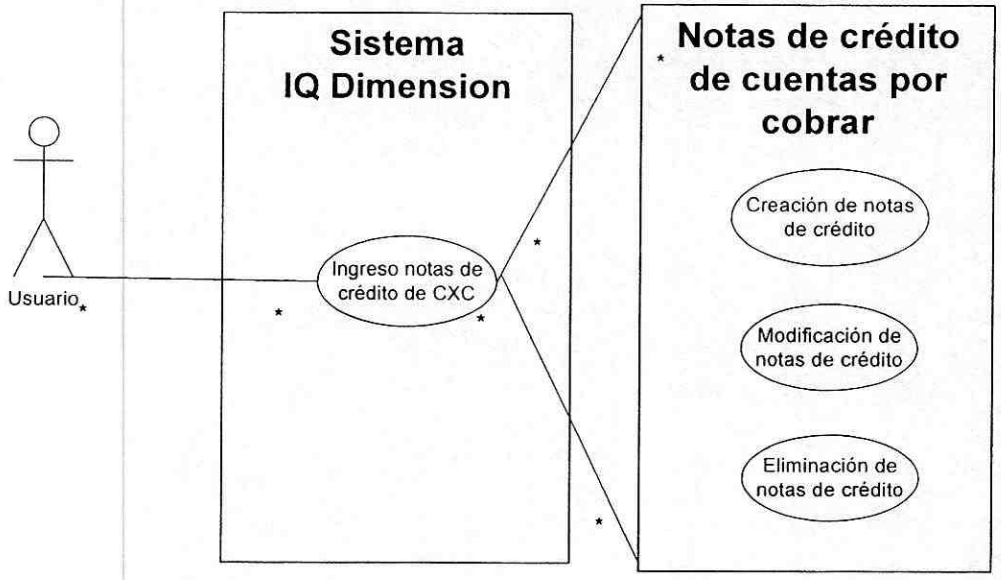
- Total impuesto. Define los impuestos que afectan a la línea de inventario. El usuario no puede editar este dato.

- Usuario. Define el usuario que creó la línea, no es visible por el usuario. Debe asignarse automáticamente al momento de crear la línea.

- IdVentaDetalle. Este campo hace referencia hacia el detalle de la venta asociado a la línea del rubro del cargo, con este dato se obtiene la información de la línea, puede que este en blanco.

El detalle de inventarios se genera obteniendo los inventarios de la venta que está asociada al cargo y rubro que referencia la línea del detalle de rubros del abono. Este detalle se genera si la nota de crédito afecta inventarios, ya sea en cantidad o total y de encontrar una venta relacionada al cargo y rubro.

Figura 9. Caso de uso de notas de crédito de cuentas por cobrar

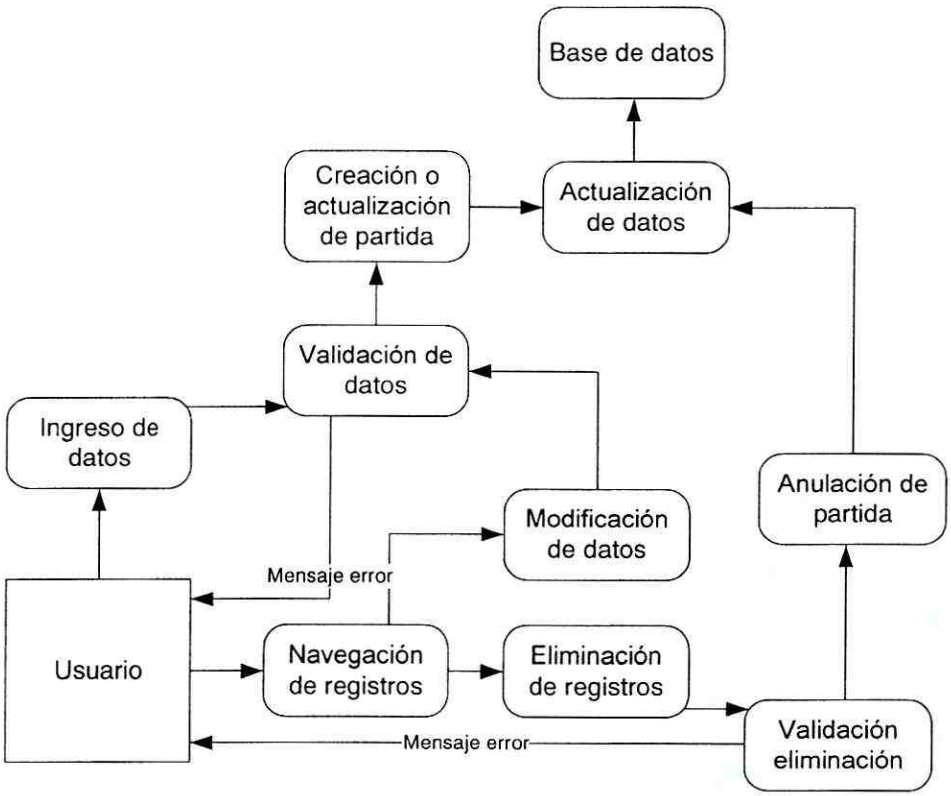



Si el módulo de contabilidad esta presente, la nota de crédito debe generar una partida contable. La descripción de la partida corresponde a la descripción de la nota de crédito, así también como la fecha. El detalle de abono lo conforma el cliente, asignando la cuenta contable por cobrar asociada a él y el total del abono, el detalle de los abonos lo conforman el detalle de rubros y de impuestos, para las líneas de rubros se emplea la cuenta contable de devoluciones definida en la configuración para la agencia y fecha de la nota de crédito y para las líneas de impuestos se emplea la cuenta contable que el impuesto tenga asociado. No puede haber líneas repetidas, por lo que si ya existe una línea con una cuenta contable, ésta debe ser actualizada en vez de crear una nueva línea. El tipo de partida contable que genera se debe obtener de la configuración de cuentas por cobrar,

obteniendo el tipo de partida para abonos para la agencia y fecha del documento de abono.

La navegación debe realizarse dentro de los registros que sean de tipo nota de crédito y que estén de estado de alta. Una nota de crédito puede ser modificada y eliminada sólo si su partida contable asociada no se encuentra actualizada por el módulo de contabilidad.

Figura 10. Diagrama de flujo de notas de crédito de cuentas por cobrar




2) Notas de crédito sobre documentos cobrados. Son similares a las notas de crédito, la diferencia es que éstas se aplican a un solo documento de cargo que ya esté cobrado, que su saldo sea igual a cero. Por lo que el detalle de cargos del abono lo conforma este documento de cargo.

Se maneja la misma información adicional de las notas de crédito más la siguiente:

- Abono a. Este campo define que tipo de movimiento va a crear la nota de crédito. Este dato puede ser, cheques bancarios, anticipo, nota de débito bancaria o caja. Este dato es obligatorio.

- IdMovimientoBancario. Hace referencia hacia al movimiento bancario, de haberse creado. Este campo puede quedar en blanco si no se genera movimiento bancario.

- Movimiento bancario. Muestra información sobre el movimiento bancario, de haberse creado. Se debe mostrar la cuenta bancaria del movimiento y el número del movimiento.

- Total anticipo. Define la cantidad que será de anticipo, de haber seleccionado que la nota de crédito genera un anticipo.

Al momento de crear un nuevo registro, el sistema debe crear una lista de clientes del sistema que tengan documentos de cargos con abonos aplicados. Se debe mostrar la siguiente información de los clientes:

- Código del cliente
- Nombre del cliente
- Identificación tributaria

El usuario debe seleccionar un cliente, al hacerlo, el sistema debe generar un listado con los documentos de cargos que tienen abonos aplicados, debe mostrar la siguiente información:

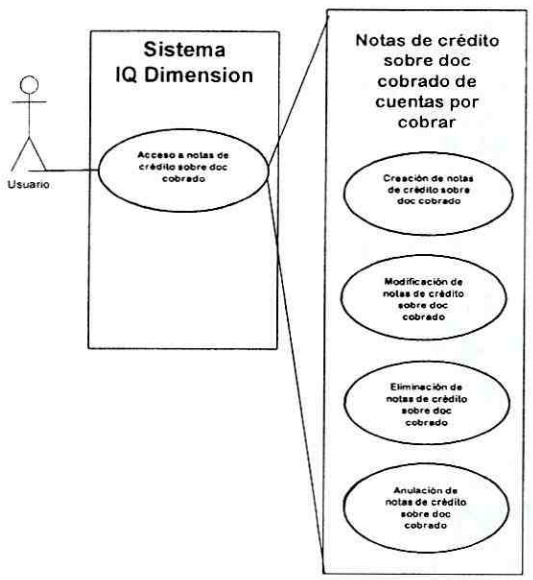
- Serie
- Número

- Fecha
- Moneda
- Total factura

El usuario debe seleccionar el documento de cargo sobre el que desea crear la nota de crédito. Utilizando esta selección el sistema debe crear el detalle de cargos de la nota de crédito. Con la información del cliente que seleccionó el usuario, se debe asignar el cliente, nombre o razón social, dirección, identificación tributaria y moneda de la nota de crédito.

El manejo del detalle de rubros, impuestos e inventarios del abono es idéntico al de notas de crédito.

Figura 11. Caso de uso de notas de crédito sobre documentos cobrados de cuentas por cobrar




Dependiendo del tipo de documento sobre el cual vaya la nota de crédito a aplicarse, el sistema debe crear una partida contable o un movimiento bancario (cheque o nota de débito).

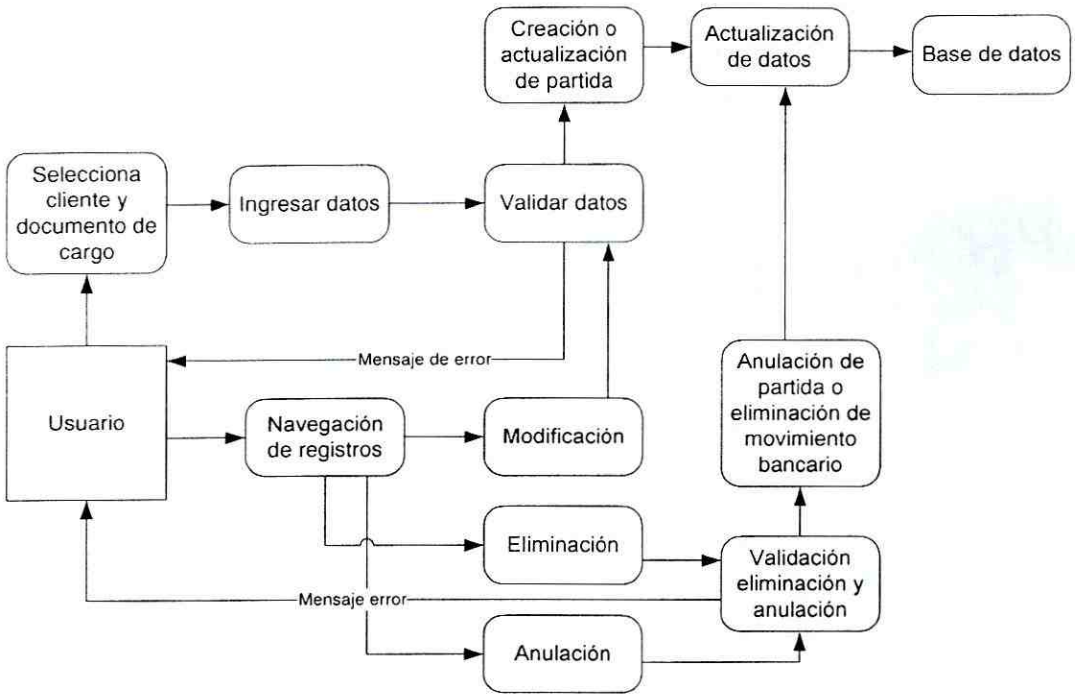
Si tiene que generar una partida contable. La descripción y fecha de la partida corresponden a la descripción y fecha de la nota de crédito. El detalle de

abonos de la partida corresponde al cliente y al total del abono, si el abono es un anticipo se utiliza la cuenta contable de anticipos asociada al cliente, si el abono es a caja se utiliza la cuenta contable de caja definida en la configuración para la agencia y fecha de la nota de crédito. El detalle de cargos de la partida lo conforman en detalle de rubros e impuestos del abono. Para las líneas del detalle de rubro se utiliza la cuenta contable de devolución de ventas definida en la configuración para la agencia y fecha de la nota de crédito. Para las líneas de impuestos se utiliza la cuenta contable asociada al impuesto. El tipo de partida contable que genera se debe obtener de la configuración de cuentas por cobrar, obteniendo el tipo de partida para abonos para la agencia y fecha del documento de abono.

Si se debe generar un movimiento bancario (cheque o nota de débito). Se envía la cuenta bancaria asociada al cliente. Si es un cheque, se envía a nombre de quién, siendo el nombre del cliente. La descripción corresponde a la descripción de la nota de crédito. El monto que corresponde al total de abono de la nota. La fecha que corresponde a la fecha de la nota de crédito.

Una nota de crédito sobre documento cobrado puede ser modificada, eliminada o anulada. Todo esto se realiza si de tener una partida generada, ésta no se encuentra actualizada. Y si tiene un movimiento bancario generado no debe estar conciliado. El usuario sólo puede modificar la información del detalle de cargos, rubros, impuestos e inventarios. La información general de la nota de crédito no puede ser modificada. Para la eliminación y anulación se debe pedir una confirmación del usuario de la operación.

Figura 12. Diagrama de flujo de notas de crédito sobre documentos cobrados de cuentas por cobrar




3) Aplicación de anticipos. Cuando se efectúa un cobro sobre un cliente, se puede dar el caso que sea un cobro directo sobre algún documento de cargo del cliente, sino que el cliente dio un anticipo a la empresa. El proceso de aplicación de anticipos se encarga de asignar el anticipo que haya dado el cliente a los documentos de cargo del mismo.

A parte de la información adicional de un abono, el proceso de aplicación de anticipo maneja la siguiente información:

- Forma de pago. Indica como fue que cliente dio el anticipo, esta información se obtiene del cobro y no el usuario no puede modificarla.

- Recibo. Muestra la información sobre el recibo del cobro, si se generó. Se debe mostrar la serie y número del recibo. El usuario no puede modificar esta información.

- Información del cobro. Muestra datos sobre el cobro. Debe mostrar la serie y número del cobro y la fecha en que se realizó. El usuario no puede modificar esta información.

- Saldo anticipo. Define cuánto del anticipo todavía se puede aplicar. Al momento de aplicar un anticipo, la cantidad aplicada debe sustraerse del saldo.

- Total aplicado. Define cuánto del anticipo fue aplicado, esta cantidad no puede ser mayor al saldo anticipo con el que se cuenta en el momento. Este dato debe ser mayor a cero.

- Total aplicado base. Es el total aplicado expresado en la moneda de la empresa, se calcula por el sistema, siendo el total anticipo multiplicado por la tasa del abono, redondeado a dos decimales.

- Total retención. Corresponde a la cantidad que se retiene al momento de aplicar el anticipo por concepto de impuestos.

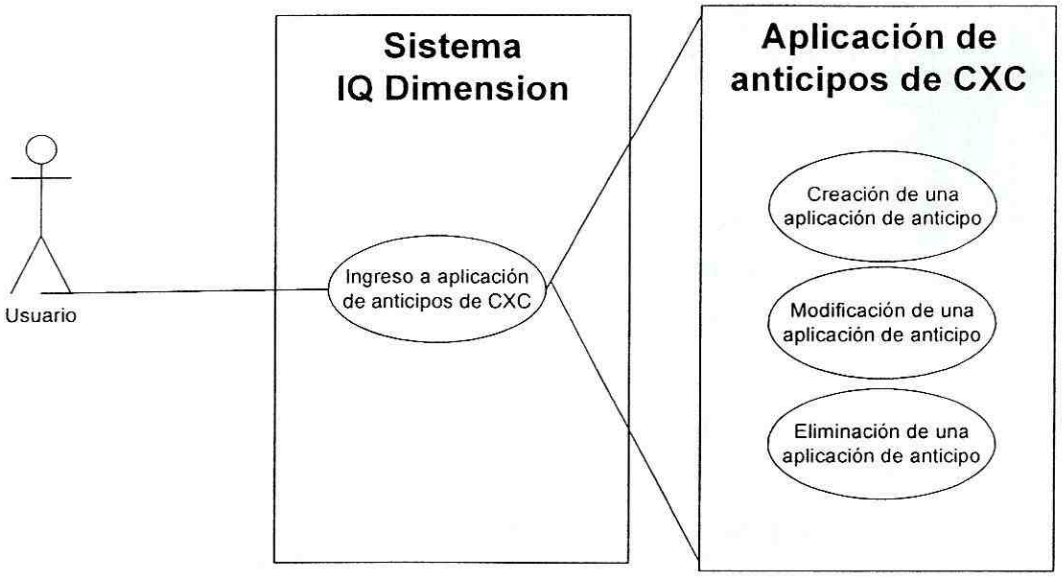
- IdAbono. Hace referencia al documento de abono cuyo anticipo esta siendo aplicado.

El número del abono no puede ser modificado por el usuario, debido a que el número es manejado automáticamente por el proceso de aplicación de anticipos.

Cuando un usuario quiera aplicar un anticipo, el sistema debe mostrar un listado de los clientes que cuentan con anticipos. El usuario debe seleccionar un cliente. Al hacerlo, el sistema debe mostrar un listado de los cobros que han recibido un anticipo, el usuario debe escoger un cobro para aplicar el anticipo. Al hacer esto, el sistema debe asignar a partir de la información de cobro, la

agencia, el cliente, forma de pago, la información de recibo, la moneda, la descripción, nombre o razón social, la dirección, la identificación tributaria y por último la información del cobro.

Figura 13. Caso de uso de aplicación anticipo de cuentas por cobrar



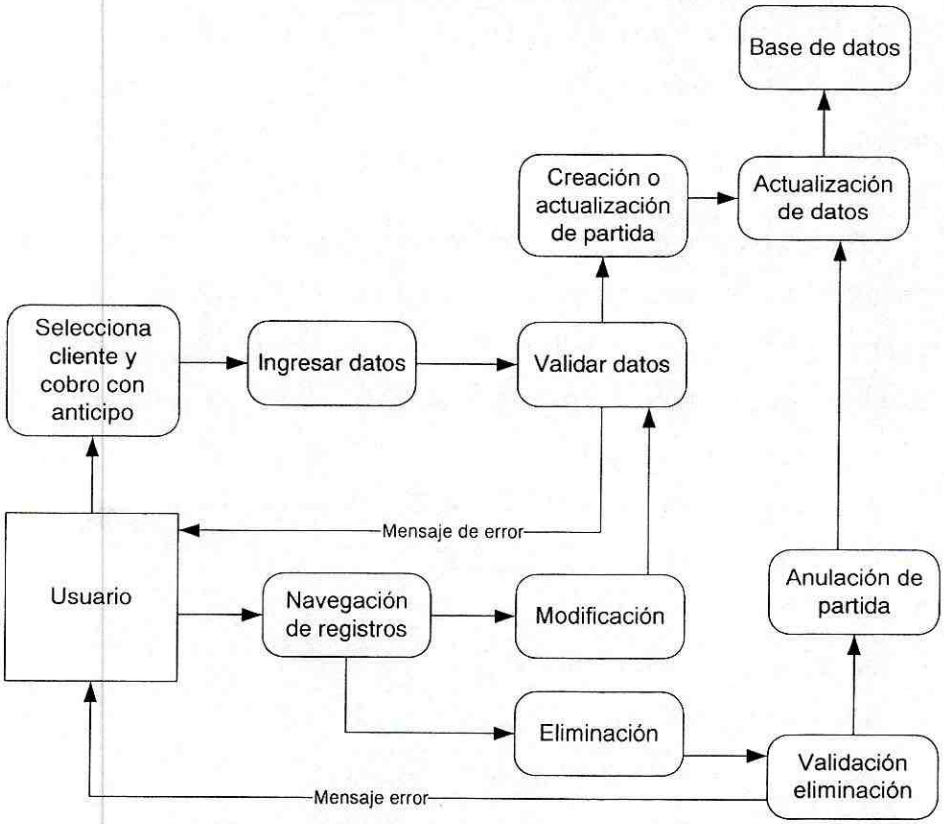

La aplicación de anticipos debe generar una partida contable. La descripción y fecha de la partida corresponden a la descripción y fecha de la aplicación anticipo. La línea de abono corresponde al cliente, asignando la cuenta contable por cobrar asociado al cliente, y el total aplicado. Los detalles de cargo lo conforma el cliente y el detalle de impuestos, para el cliente se asigna la cuenta contable de anticipo asociado al cliente y el total anticipo menos el total retenido. Para el detalle de impuestos se asigna la cuenta contable asociada al impuesto. No pueden haber líneas repetidas, si existe una línea con la cuenta contable, se actualiza, no se crea otra línea nueva. El tipo de partida contable que genera se debe obtener de la configuración de cuentas por cobrar, obteniendo el tipo de partida para abonos para la agencia y fecha del documento de abono.

El usuario puede modificar o eliminar una aplicación de anticipo. Para hacer esto hay que validar que la partida contable no esta actualizada. El usuario durante una modificación sólo puede editar la fecha, tasa y los detalles de cargos, detalle de rubros y detalle de impuestos. Durante la eliminación, el sistema debe pedir una confirmación al usuario para realizar la operación.

En las validaciones del detalle de cargos, además de validar que el abono no sea mayor al total del cargo, se tiene que validar que no sea mayor al saldo de anticipo.

En la aplicación de anticipos, el detalle de impuestos lo genera el usuario, esto comprende los impuestos que se retienen al momento de aplicar el anticipo.

Figura 14. Diagrama de flujo de aplicación de anticipos de cuentas por cobrar



c. Calendario de cobros. Este proceso permite al usuario ver los cobros que tiene que efectuar y durante cuáles fechas. Así mismo puede visualizar aquellos cobros que no han sido realizados y cuya fecha de vencimiento ya pasó.

El calendario de cobros debe ofrecer una vista mensual, semanal y diaria. Debe permitir al usuario navegar entre los diferentes meses, semanas o días, dependiendo de la vista donde este, debe poderse ir a fechas posteriores a la actual y a fechas anteriores a la actual.

Las vistas tienen que estar relacionadas entre sí, si el usuario cambia de mes, la vista semanal y diaria deben acomodarse al nuevo mes en que se encuentra. De la misma forma debe suceder cuando se cambia de fecha en las vistas semanal y diaria.

A parte de las vistas, el sistema debe proveer un listado con los cobros suya fecha de vencimiento ya pasó, esto para facilitar la localización de dichos cobros.

Para obtener la información de los cobros a efectuar, se debe revisar las condiciones de cobros de los documentos de cargos, ya que ellos definen cuánto y cuándo se debe efectuar el cobro. Se tienen que buscar las condiciones de cobros que tengan su dato de cobrado con el valor N.

Los cobros que muestre el calendario en las vistas debe incluir la siguiente información:

- Serie del documento de cargo
- Número del documento de cargo
- Monto a cobrar del documento de cargo
- Cliente a quien corresponde el documento de cargo

Para la vista diaria y la lista de cobros cuya fecha de vencimiento ya pasó además se debe mostrar:

- Fecha del cobro.
- Fecha de vencimiento del cobro.
- Descripción del documento de cargo.

Por último se debe obtener el cargo asociado al cobro pendiente, este dato no se muestra al usuario.

El proceso de calendario de cobros debe permitir al usuario refrescar la información y así actualizarla. También debe poderse imprimir un listado de los cobros pendientes de realizarse dado un rango de fechas.

Cuando el usuario haga doble clic sobre alguno de los cobros pendientes, el sistema debe mostrar en pantalla el documento de cargo completo que esta asociado al cobro pendiente.

Figura 15. Caso de uso del calendario de cobros de CXC

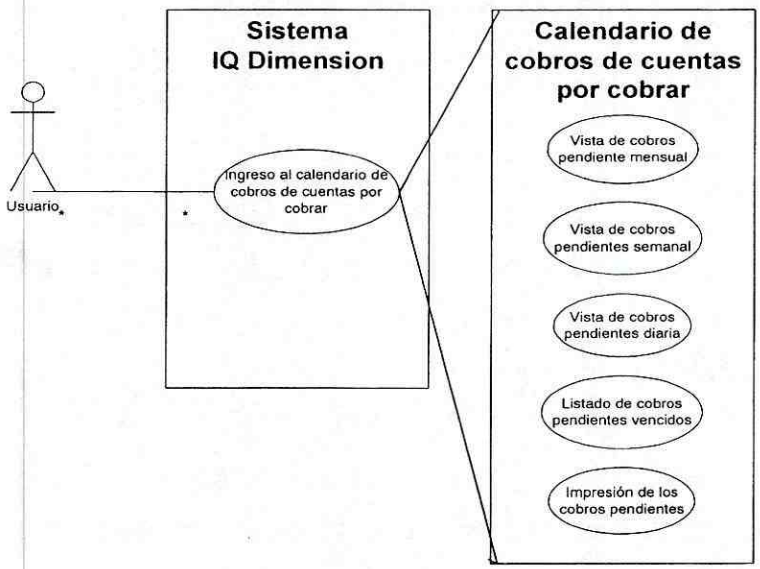
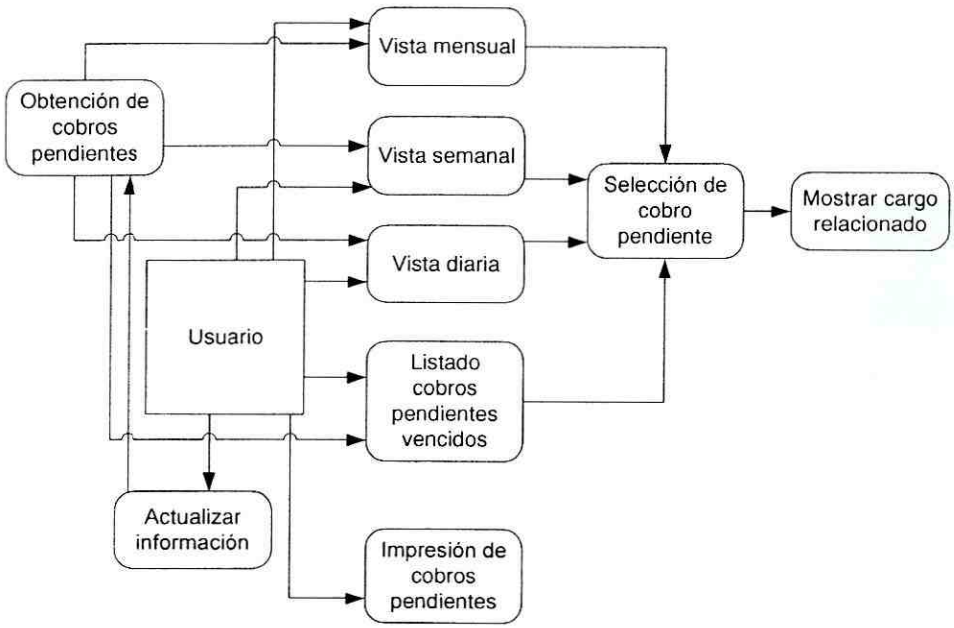


Figura 16. Diagrama de flujo del calendario de cobros de cuentas por cobrar




d. TRIGGERS. Cuando se ingresa, elimina o modifica un registro de abono, se deben actualizar las condiciones de cobros y los detalles de rubros de los documentos de cargos a los que aplica el abono.

El detalle de rubros del documento de abono debe afectar al detalle de rubros de documento de cargo que referencia. Esto debe ocurrir de la siguiente manera:

Cuando se creen las líneas del detalle de rubro del documento de abono, por cada línea, debe sumarse al total cobrado y total cobrado base de la línea del detalle de rubro del cargo que referencia la línea del abono, el abono y abono base respectivamente.

Cuando las líneas del detalle de rubro del documento de abono sean eliminadas, por cada línea, debe sustraerse al total cobrado y el total cobrado

base de la línea del detalle de rubro del cargo que referencia la línea del abono, el abono y abono base respectivamente.

Cuando se modifiquen el abono y abono base de las líneas del detalle de rubro de documento de abono, por cada línea, se resta al total cobrado y el total cobrado base de la línea del detalle de rubros del cargo que referencia la línea del abono el abono y abono base original que tenía la línea respectivamente. Luego se suma al total cobrado y al total cobrado base de la línea de cargo, el nuevo valor de abono y abono base de la línea del abono respectivamente.

Si el documento de abono es anulado, se debe manejar como si fuera una eliminación.

Además del detalle de rubro, se debe actualizar las condiciones de cobros con el total de abono.

Si es un ingreso se deben obtener las condiciones de cobros que tengan en dato de cobrado igual a N, ordenados por la fecha de cobro de manera ascendente. Por cada condición se debe:

- Si el total cobrado es igual al total de la línea. Se asigna a cobrado S y fecha de cobro igual a la fecha del abono. Se termina proceso.

- Si el total cobrado es mayor al total de la línea. Se asigna a cobrado S y fecha de cobro igual a la fecha del abono. Para las demás condiciones se maneja la diferencia entre el total cobrado y el total del cobro de la condición.

- Si el total cobrado es menor al total de la línea, se procede a la siguiente condición.

Si al final queda una diferencia de total cobrado, con esa diferencia se debe realizar una nueva condición de cobro, con total igual a la diferencia, cobrado igual a S, el usuario se pone "sistema", la fecha de cobro igual a la fecha en que se realizó el abono y descripción con "Generada automáticamente".

Si se realiza una eliminación o anulación entonces se debe obtener las condiciones de cobro que tengan el dato de cobrado igual a S, ordenados por la fecha de cobro de manera descendente. Por cada condición se debe realizar:

- Si el total cobrado es igual al total de la línea. Se asigna a cobrado N y se termina el proceso.

- Si el total cobrado es mayor al total de la línea. Se asigna a cobrado N y para las demás condiciones se maneja la diferencia entre total cobrado y el total.

# IV. DISEÑO

El módulo de cuentas por cobrar de IQ Profit fue desarrollado en la plataforma Microsoft Visual Studio .NET 2003 Enterprise Architect. Se decidió realizar el módulo en esta plataforma debido a la gran capacidad que tienen para la programación orientada a objetos, sin mencionar las herramientas que posee para la creación de interfaces gráficas y la captura de información.

Una de las grandes ventajas de Microsoft Visual Studio .NET 2003 es la colección de clases denominadas controles que se pueden utilizar. Los controles son clases que se emplean para la captura, manejo y muestra de información. Una de las mejores características de los controles son los denominados eventos de los controles. Estos son funciones que ejecutan cuando el control se comporta de cierta manera, permitiendo al desarrollador implementar su lógica de programación cuando el control se comporte de cierta manera.

## A. Arquitectura IQ

Todo módulo dentro de IQ Dimension debe estar compuesto por los siguientes proyectos:

1. Client. Este proyecto comprende todo lo referente a interfaz gráfico con el usuario. Contiene todas las formas del módulo.

2. Module. Este proyecto comprende una interfase sobre el módulo hacia otros módulos dentro de IQ Dimension. Aquí se implementan funciones las cuales pueden acceder tanto otros módulos como él mismo.

3. Proxy. Comprende las definiciones de las interfases con los que cuenta el módulo. Toda clase que implemente una interfase tiene que buscar en este proyecto la definición del mismo.

4. Server. Dentro de este proyecto se definen las siguientes clases:

a. Data Managers (DM). Esta clase se encarga de manejar las consultas y operaciones de una tabla específica en la base de datos. Se debe especificar el nombre de la tabla de la base de datos y los campos de la misma que va a manejar.

Muchas de las funcionalidad de un DM se maneja a través de herencias, por lo que no hay que definirlas cada vez que se defina una clase DM. Dentro de estas funcionalidades están las funciones de guardar, que ingresa o modifica datos a la base de datos, y la de eliminar, que elimina datos de la base de datos. Cada función recibe como parámetro una datatable con la información.

b. Group Manager (GM). Esta clase maneja la información de un catálogo, configuración o proceso. Él se encarga de realizar las operaciones crear un nuevo registro, guardar, modificar y eliminar datos a nivel general, y de ser necesario, la navegación de registros.

Un GM comprende de uno o mas DM dependiendo de la cantidad de tablas de la base de datos que se manejen. Es necesario especificar al GM cuál de todos los DM que emplea es el principal en concepto de jerarquías.

El GM es el encargado de crear una conexión con la base de datos. Se la envía a los DM para que estos empleen esta conexión al momento de realizar sus operaciones y consultas.

Mucha de la funcionalidad de un GM se maneja a través de herencia. Dentro de la funcionalidad de un GM se pueden encontrar:

- Ir A. Esta función llena un dataset con la información completa de registro dado un identificador de la base de datos. Regresa un valor bolean.

- Obtener registros por campos. Realiza la misma funcionalidad que la función Ir A, con la diferencia que recibe unas listas indicando por que campos y valores de la base de datos se desea buscar el registro. También permite enviar un identificador para indicarle que el identificador del registro que encuentre no debe ser el mismo. Regresa un valor bolean.

- Obtener listado. Llena un dataset con los datos generales de los registros existentes que se encuentren en estado de alta en la base de datos.

También se pueden definir en este proyecto los datasets que manipularán los GMs, pero estos también pueden ser definidos en otro proyecto.

5. Crystal. Corresponde a los formatos de los reportes y listados que posee el módulo.

6. Shared.Interop. Este proyecto contiene estructuras de datos que utiliza el módulo y que otros módulos pueden utilizar también. Por lo general, se definen datasets para que otros módulos también puedan utilizarlos.

7. Shared. Este proyecto comprende todas las clases que se emplean para herencia. Dentro de estas clases se pueden encontrar:

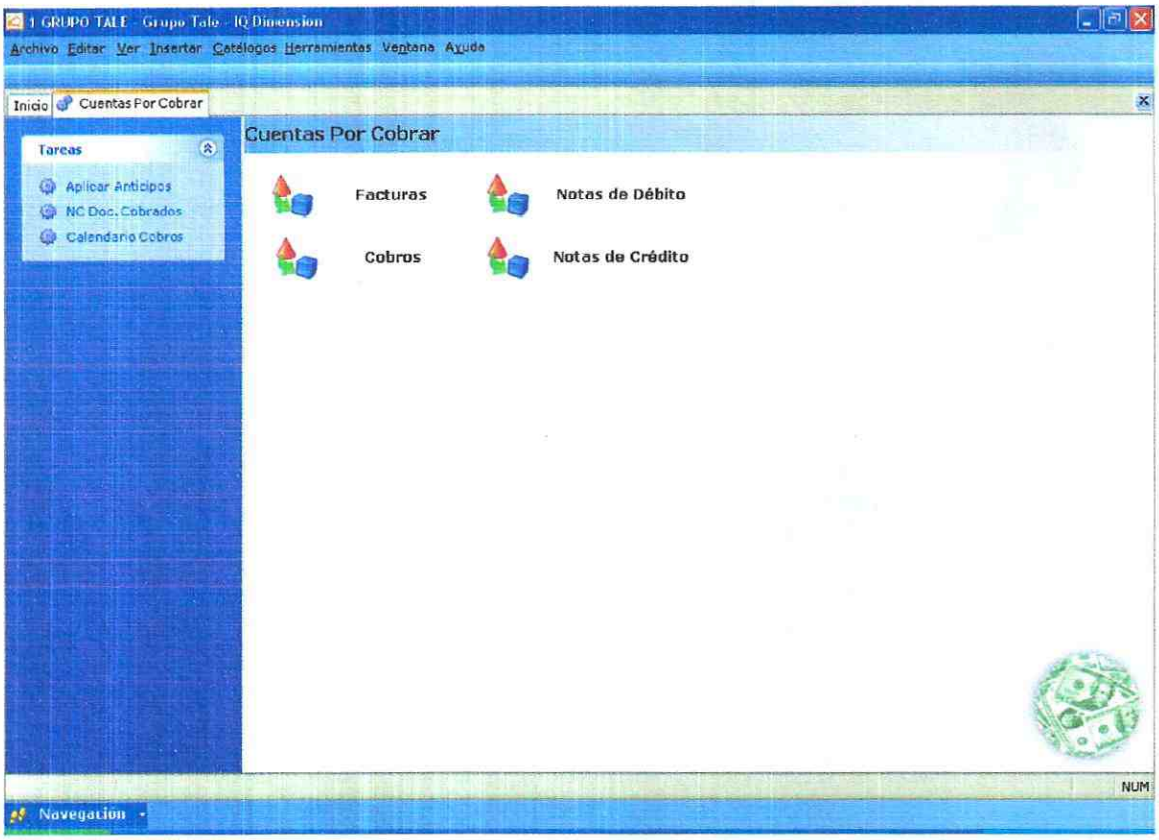
- Clases interfases de catálogos.
- Clases para interfases de procesos.
- Clases para interfases de configuraciones.
- Clases para DM
- Clases para GM

Todo esto con el fin de lograr una mejor optimización de código.

## B. Diseño del módulo cuentas por cobrar

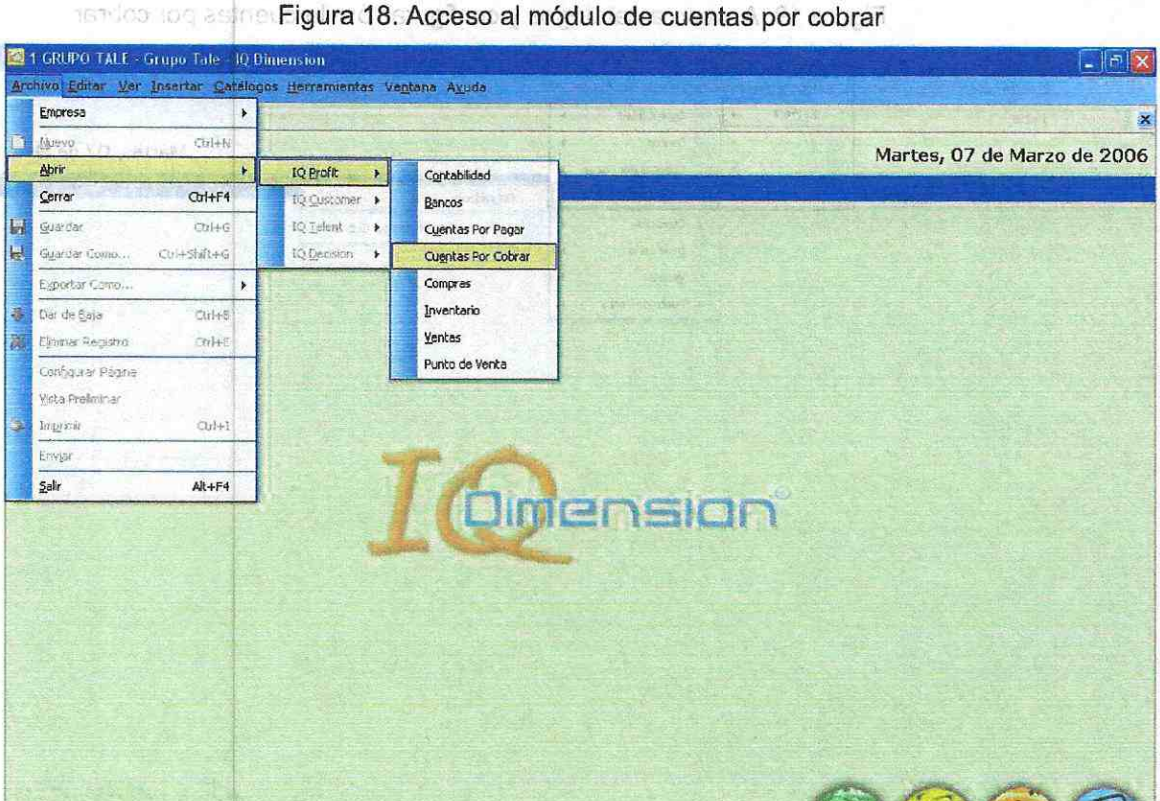
El módulo cuenta con una pantalla principal la cual brinda acceso a los procesos dentro del módulo:

Figura 17. Pantalla principal al módulo de cuentas por cobrar



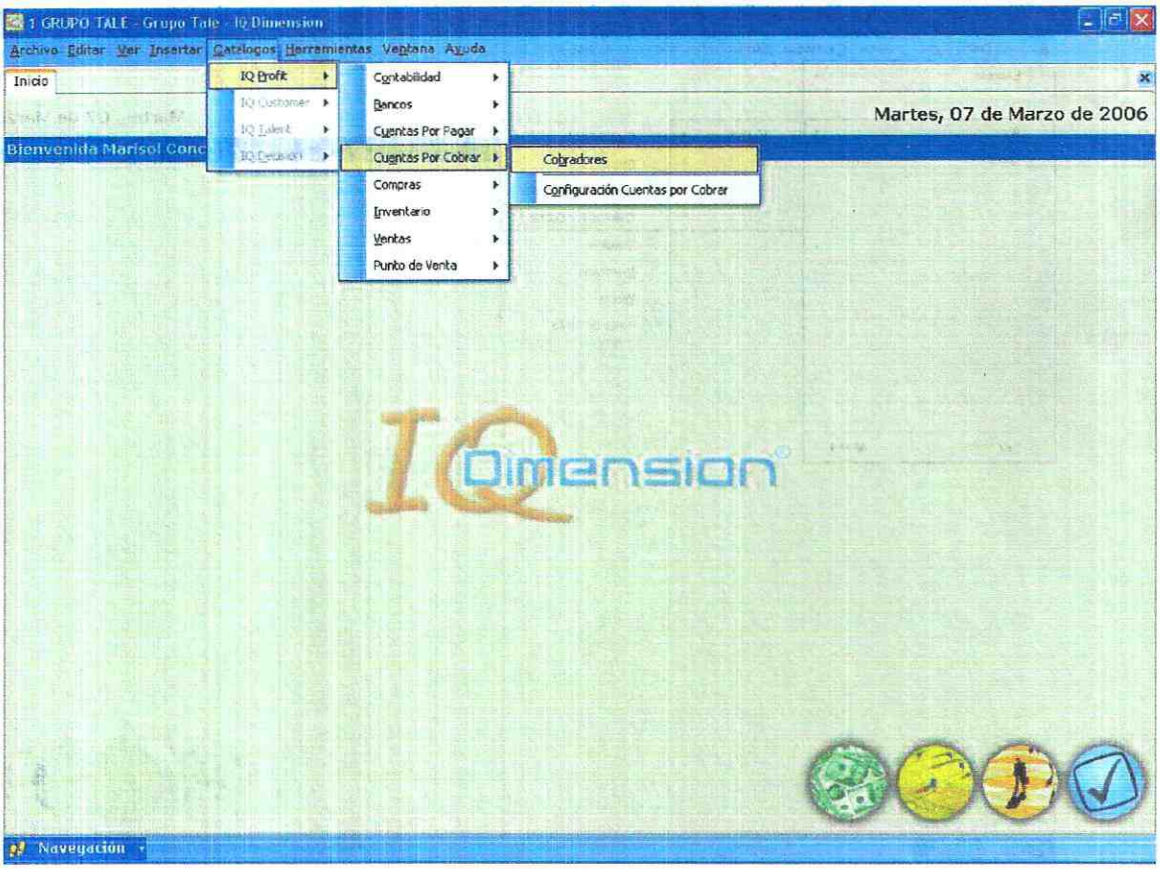

Esta pantalla se accede desde la barra de menú en el menú Archivo, la opción de abrir, IQ Profit.

Figura 18. Acceso al módulo de cuentas por cobrar




1. Catálogos. Los catálogos del módulo de cuentas por cobrar se acceden desde la barra de menú, en el menú de Catálogos, opción IQ Profit, opción Cuentas por Cobrar.

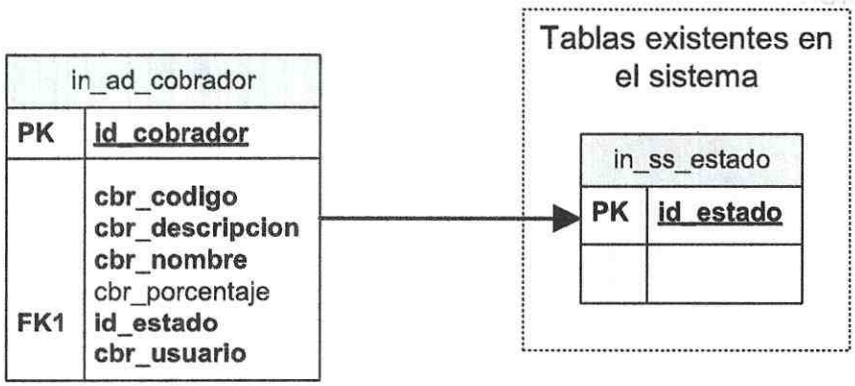
Figura 19. Acceso a catálogos y configuración de cuentas por cobrar




a. Cobradores.

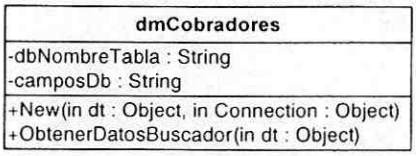
1) Diagrama de base de datos.

Figura 20. Diagrama de base de datos del catálogo de cobradores



2) Diagrama de clases. dmCobradores. Esta clase será la encargada de interactuar con la tabla in_ad_cobrador en la base de datos.

Figura 21. Diagrama de la clase dmCobradores




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

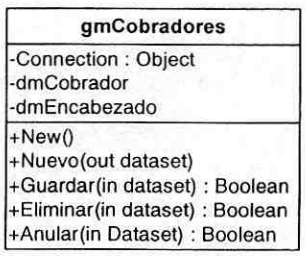
El procedimiento ObtenerDatosBuscador alimenta la datatable dt con el listado de registros existentes en la base de datos.

EsquemaCobradores. Este es el dataset que manejara la información en la aplicación.

Figura 22. Diagrama de la clase EsquemaCobradores


gmCobradores. Esta clase es la encargada de administrar los datos del catalogo de cobradores.

Figura 23. Diagrama de la clase gmCobradores



El procedimiento New es el constructor de la clase, es el encargado de inicializar los atributos.

El procedimiento Nuevo crea un nuevo registro en el dataset.

La función Guardar realiza la actualización de la información en la base de datos con la información dentro de dataset. Emplea a dmCobrador para realizar las operaciones de INSERT, UPDATE y DELETE.

La función Eliminar realiza la eliminación de datos en la base de datos que corresponde a la que tiene el dataset enviado. Emplea al dmCobrador para realizar las operaciones de DELETE.
La función Anular cambia el estado del registro definido a anulado y asigna la fecha de baja. Emplea al dmCobrador para realizar las operaciones de UPDATE.

frmCobradores. La clase tiene como fin capturar y validar la información del cobrador proveniente del usuario del sistema.

Figura 24. Diagrama de la clase frmCobradores


La función Preload se encargar de inicializar todos los atributos de la clase.

El procedimiento Nuevo inicializa datos para un registro nuevo.

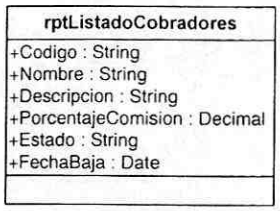
La función PreGuardar realiza las validaciones de los datos antes de proceder a actualizar el registro en la base de datos.

El procedimiento ObtenerPorCodigo busca un registro dentro del sistema que contenga el código que se recibe. Emplea la función del gmCobradores,

ObtenerRegistroPorCampos. Este procedimiento se manda a llamar cuando se ingresa un código.

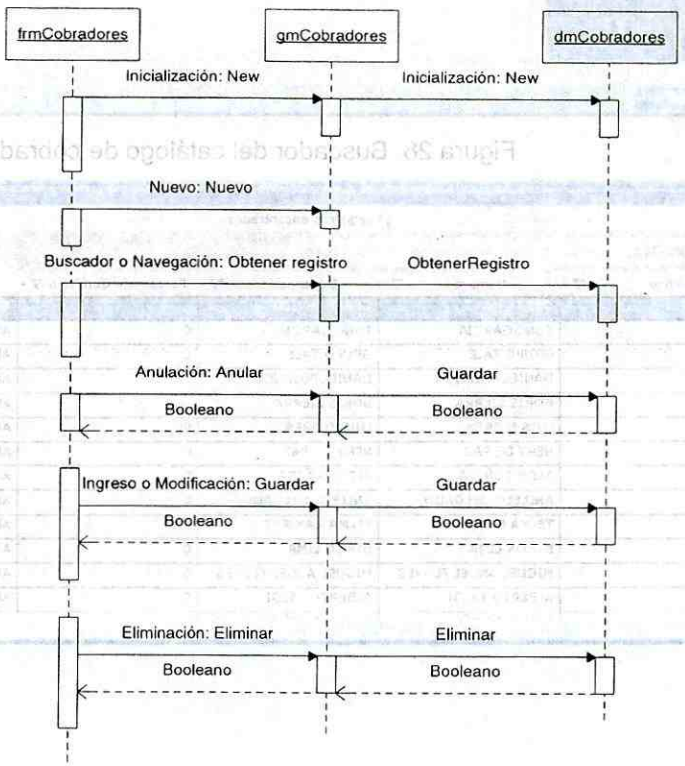
rptListadoCobradores. Esta clase se emplea para realizar la impresión del listado de cobradores existentes en el sistema.

Figura 25. Diagrama de la clase rptListadoCobradores




3) Diagrama de secuencia

Figura 26. Diagrama de secuencia del catálogo de cobradores



4) Interfaz gráfico de usuario

Figura 27. Interfaz gráfica del catálogo de cobradores

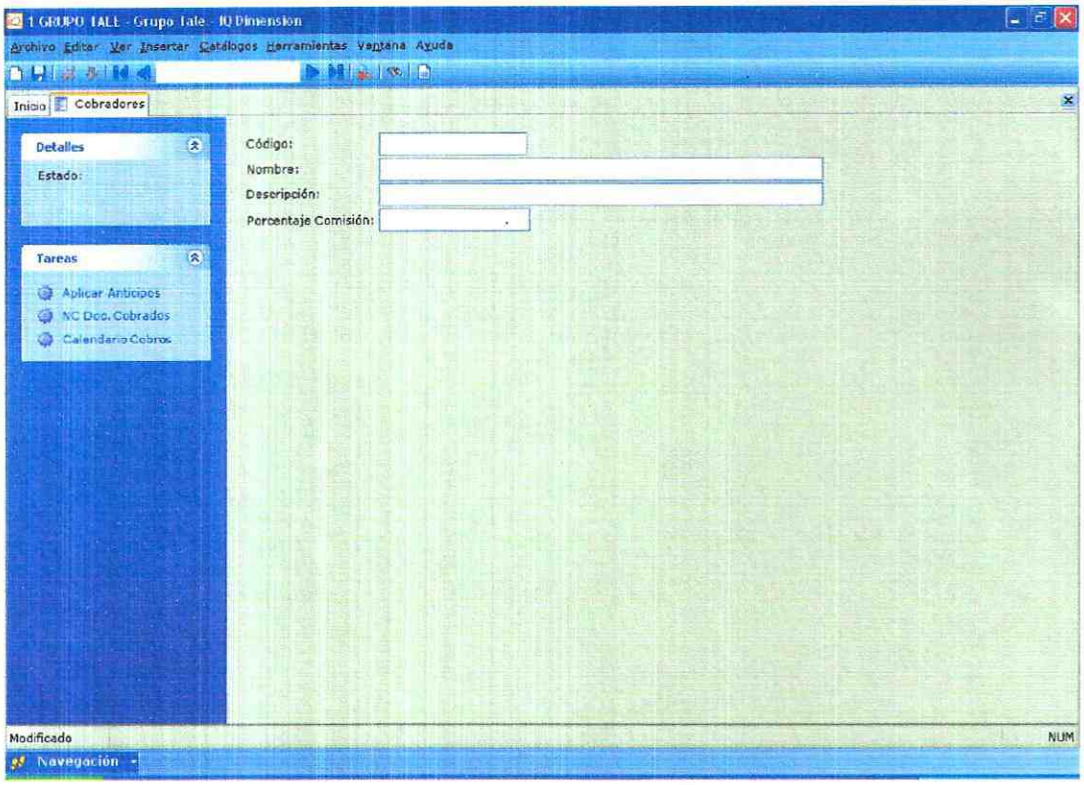
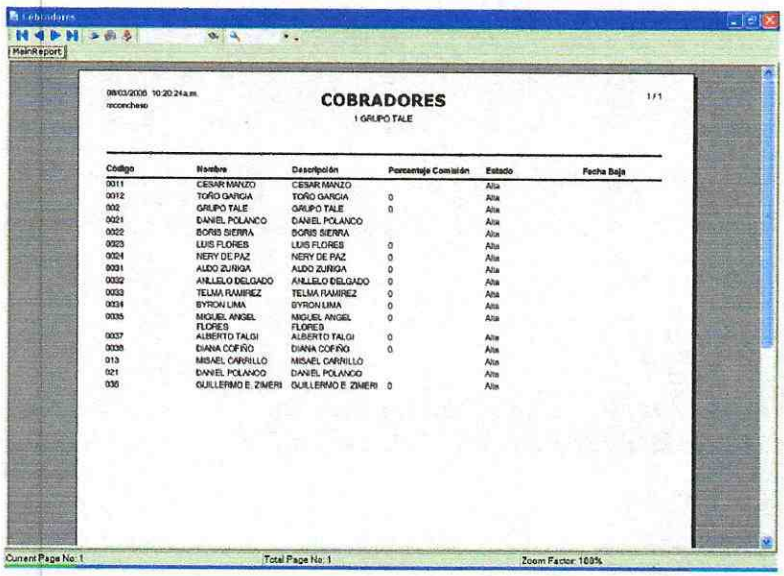



Figura 28. Buscador del catálogo de cobradores


| Código | Nombre | Descripción | Porcentaje Comisión | Estado |
|---|---|---|---|---|
| 0011 | CESAR MANZO | CESAR MANZO | | Alta |
| 0012 | TOÑO GARCIA | TOÑO GARCIA | 0 | Alta |
| 002 | GRUPO TALE | GRUPO TALE | 0 | Alta |
| 0021 | DANIEL POLANCO | DANIEL POLANCO | | Alta |
| 0022 | BORIS SIERRA | BORIS SIERRA | | Alta |
| 0023 | LUIS FLORES | LUIS FLORES | 0 | Alta |
| 0024 | NERY DE PAZ | NERY DE PAZ | 0 | Alta |
| 0031 | ALDO ZUÑIGA | ALDO ZUÑIGA | 0 | Alta |
| 0032 | ANLLELO DELGADO | ANLLELO DELGADO | 0 | Alta |
| 0033 | TELMA RAMIREZ | TELMA RAMIREZ | 0 | Alta |
| 0034 | BYRON LIMA | BYRON LIMA | 0 | Alta |
| 0035 | MIGUEL ANGEL FLORES | MIGUEL ANGEL FLORES | 0 | Alta |
| 0037 | ALBERTO TALGI | ALBERTO TALGI | 0 | Alta |

Figura 29. Impresión del catálogo de cobradores

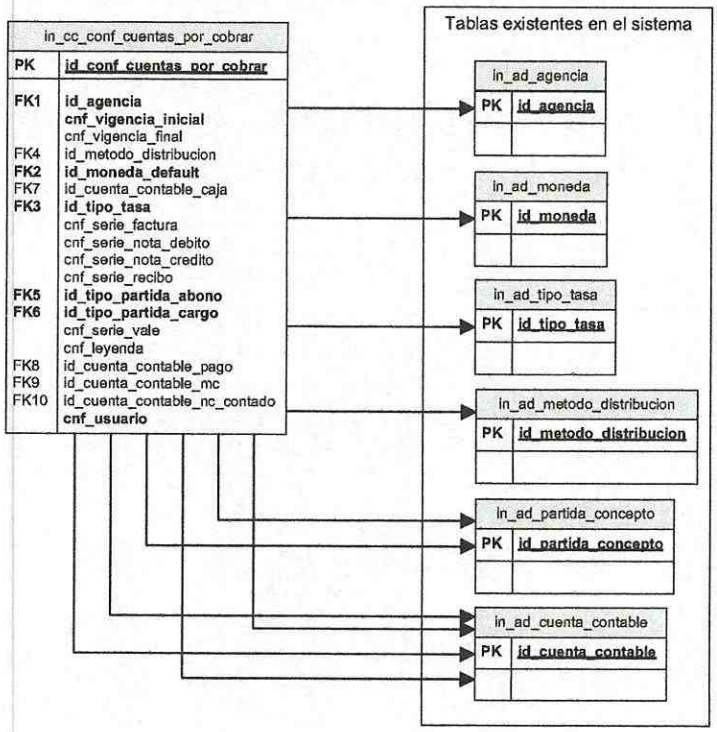



2. Configuraciones. Las configuraciones de acceden de la misma manera que los catálogos. Ver Figura 19.

a. Configuración de cuentas por cobrar

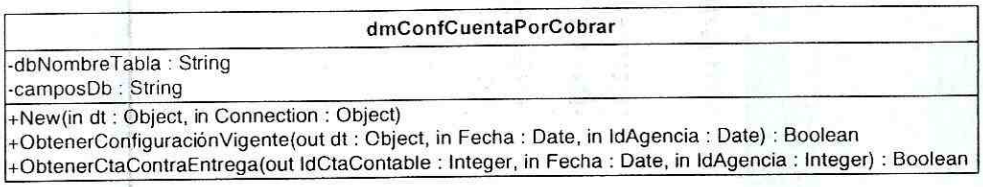
1) Diagrama de base de datos

Figura 30. Diagrama de base de datos de la configuración de cuentas por cobrar



2) Diagrama de clases. dmConfCuentaPorCobrar. Es la clase que se emplea para interactuar con la tabla in_cc_conf_cuentas_por_cobrar en la base de datos.

Figura 31. Diagrama de la clase dmConfCuentaPorCobrar




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerConfiguracionVigente llena la datatable dt con la vigencia activa para la fecha y agencia dada.

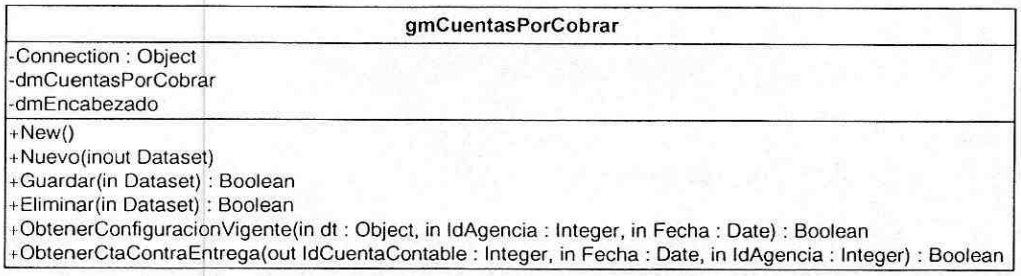
La función ObtenerCtaContraEntrega encuentra la cuenta contable para pago contra entrega definida en la configuración de cuentas por cobrar para la agencia y fecha dada.

EsquemaCuentasPorCobrar. Este es el dataset que almacena la información sobre la configuración de las cuentas por cobrar dentro de la aplicación.

Figura 32. Diagrama de la clase EsquemaCuentasPorCobrar

gmCuentasPorCobrar. Esta clase se encarga de administrar la información de la configuración de las cuentas por cobrar.

Figura 33. Diagrama de la clase gmCuentasPorCobrar




El procedimiento New es el constructor de la clase, es el encargado de inicializar los atributos.

El procedimiento Nuevo crea un nuevo registro en el dataset, inicializando campos como el usuario, fecha de ingreso, etc.

La función Guardar realiza la actualización de la información en la base de datos con la información dentro de dataset. Emplea a dmCuentasPorCobrar para realizar las operaciones de INSERT, UPDATE y DELETE.

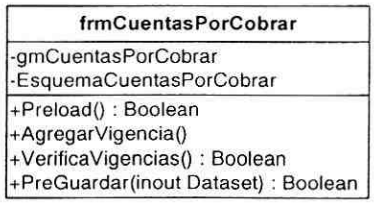
La función Eliminar realiza la eliminación de datos en la base de datos que corresponde a la que tiene el dataset enviado.

La función ObtenerConfiguracionVigente simplemente manda a llamar a la función ObtenerConfiguracionVigente del dmCuentasPorCobrar.

La función ObtenerCtaContraEntrega simplemente manda a llamar a la función ObtenerCtaContaEntrega del dmCuentasPorCobrar.

frmCuentasPorCobrar. La clase se emplea para capturar y validar la información que ingrese el usuario del sistema.

Figura 34. Diagrama de la clase frmCuentasPorCobrar




La función Preload se encargar de inicializar todos los atributos de la clase.

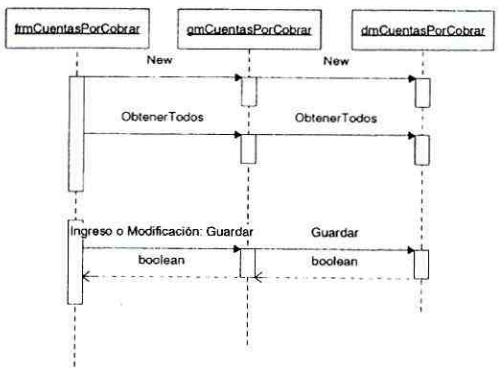
La función PreGuardar realiza las validaciones de los datos antes de proceder a actualizar el registro en la base de datos. Además calcula las vigencias finales de las vigencias que tengan una vigencia posterior a ellas.

El procedimiento AgregarVigencia crea una nueva vigencia para la agencia actual, esto se obtiene con ayuda del atributo dvAgenciaVigencia, el cual es una vista que indica con qué agencia se está trabajando. El procedimiento también copia los datos importantes de la vigencia anterior a la vigencia nueva.

La función VerificaVigencias revisa que todas las agencias tengan creada por lo menos una vigencia.

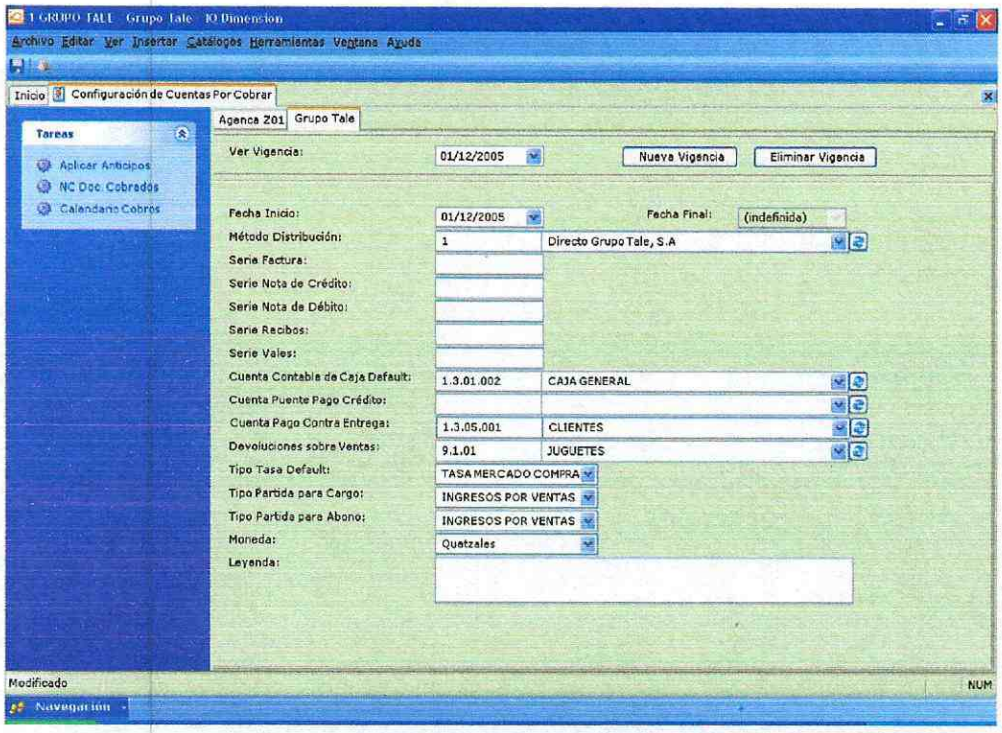
3) Diagrama de secuencia

Figura 35. Diagrama de secuencia de la configuración de cuentas por cobrar



4) Interfaz gráfica de usuario

Figura 36. Interfaz gráfico de la configuración de cuentas por cobrar

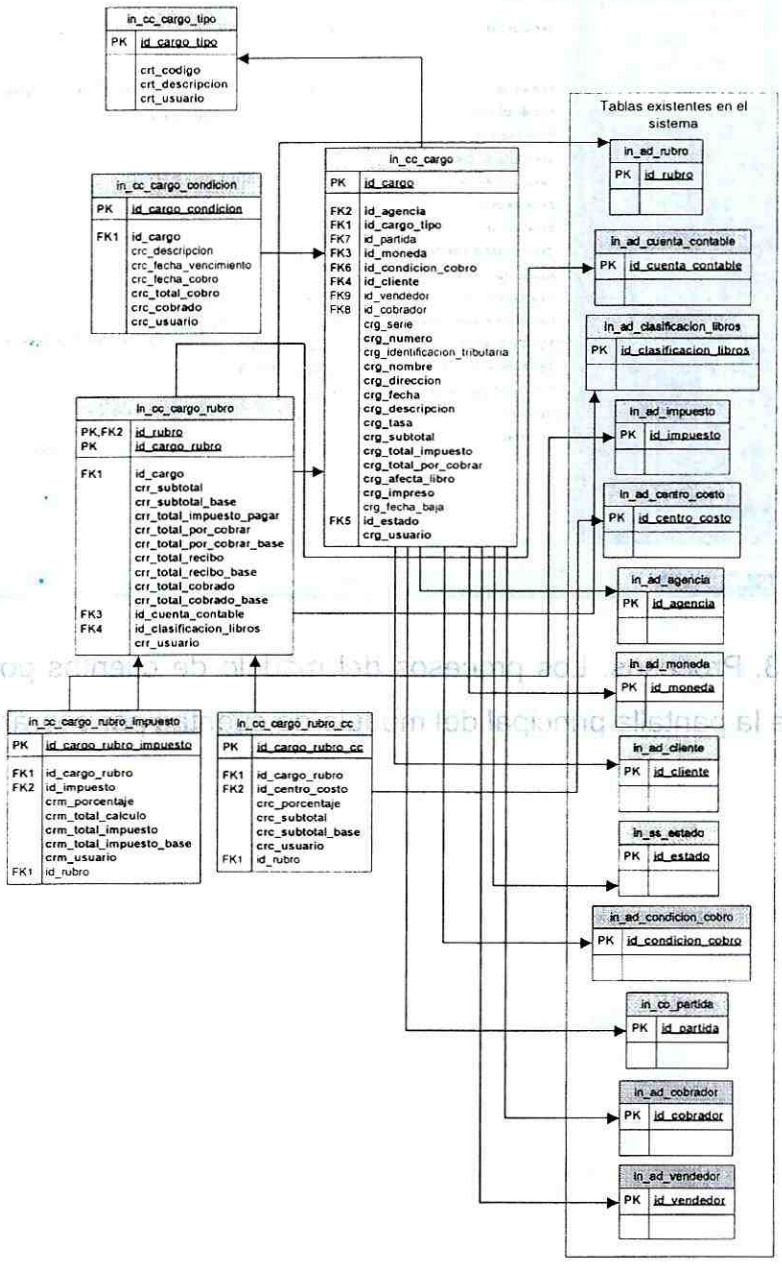



3. Procesos. Los procesos del módulo de cuentas por cobrar se acceden desde la pantalla principal del módulo de cuentas por cobrar. Ver Figura 17.

a. Facturas y notas de débito

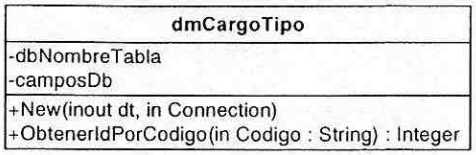
1) Diagrama de base de datos

Figura 37. Diagrama de base de datos del proceso de cargos




2) Diagrama de clases. dmTipoCargo. Es la clase que se emplea para interactuar con la tabla in_cc_cargo_tipo en la base de datos.

Figura 38. Diagrama de la clase dmCargoTipo

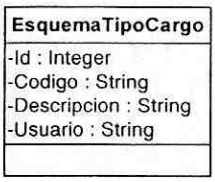



El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejara y la conexión que utilizará.

La función ObtenerIdPorCodigo obtiene el identificador del registro al que pertenece el código que recibe.

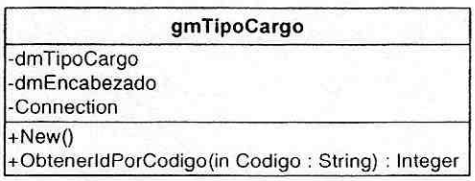
EsquemaTipoCargo. Este es el dataset que almacena la información sobre los tipos de cargos dentro de la aplicación.

Figura 39. Diagrama de la clase EsquemaTipoCargo




gmTipoCargo. Esta clase se encarga de administrar la información de los tipos de cargos.

Figura 40. Diagrama de la clase gmTIpoCargo

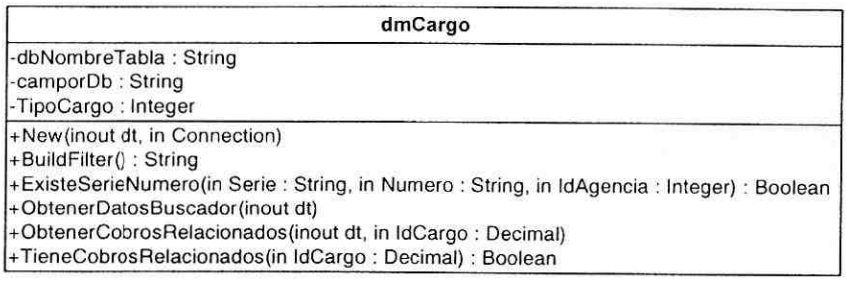



El procedimiento New es el constructor de la clase, se encarga de inicializar los atributos de la clase.

La función ObtenerIdPorCodigo simplemente manda a llamar a la función del dmTipoCargo con el mismo nombre.

dmCargo. Es la clase que se emplea para interactuar con la tabla in_cc_cargo en la base de datos.

Figura 41. Diagrama de la clase dmCargo




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función BuildFilter genera un filtro según el tipo de cargo para la navegación de registros.

La función ExisteSerieNumero indica si para la agencia recibida existe la serie y número enviado.

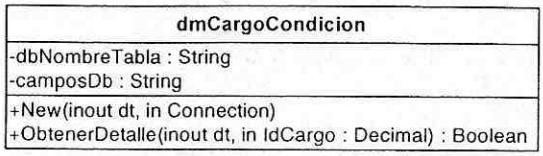
El procedimiento ObtenerDatosBuscador alimenta la datatable dt con el listado de registros existente en la base de datos.

El procedimiento ObtenerCobrosRelacionados alimenta la datatable dt con el listado de abonos relacionados al identificador del cargo recibido.

La función TieneCobrosRelacionados indica si el cargo recibido tiene algún documento de abono relacionado.

dmCargoCondicion. Es la clase que se emplea para interactuar con la tabla in_cc_cargo_condicion en la base de datos.

Figura 42. Diagrama de la clase dmCargoCondicion

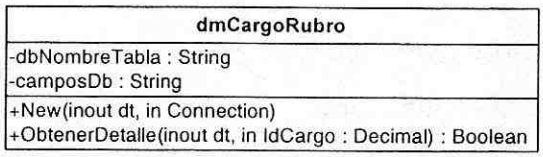



El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros asociados al identificador del cargo recibido.

dmCargoRubro. Es la clase que se emplea para interactuar con la tabla in_cc_cargo_rubro en la base de datos.

Figura 43. Diagrama de la clase dmCargoRubro

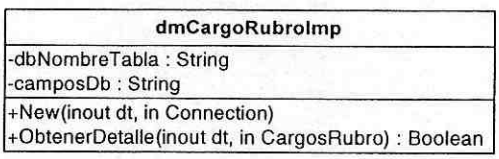



El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros asociados al identificador del cargo recibido.

dmCargoRubroImp. Es la clase que se emplea para interactuar con la tabla in_cc_cargo_rubro_impuesto en la base de datos.

Figura 44. Diagrama de la clase dmCargoRubroImp

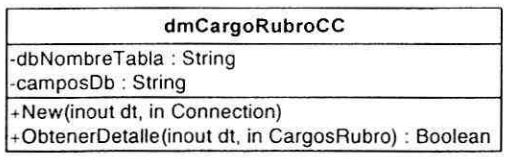



El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros asociados a los identificadores de cargo rubro contenidos en el arreglo CargosRubro.

dmCargoRubroCC. Es la clase que se emplea para interactuar con la tabla in_cc_cargo_rubro_cc en la base de datos.

Figura 45. Diagrama de la clase dmCargoRubroCC




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

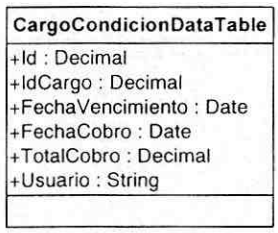
La función ObtenerDetalle alimenta la datatable dt con los registros asociados a los identificadores de cargo rubro contenidos en el arreglo CargosRubro.

CargoDataTable. Este es el datatable que almacena la información general de los cargos dentro de la aplicación.

Figura 46. Diagrama de la clase CargoDataTable

CargoCondicionDataTable. Este es el datatable que almacena la información del detalle de condiciones de cobros dentro de la aplicación.

Figura 47. Diagrama de la clase CargoCondicionDataTable




CargoRubroDataTable. Este es el datatable que almacena la información del detalle de rubros dentro de la aplicación.

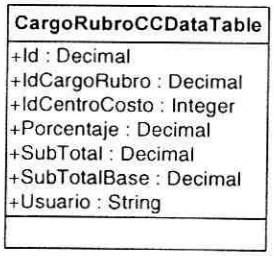
Figura 48. Diagrama de la clase CargoRubroDataTable


CargoRubroImpDataTable. Este es el datatable que almacena la información del detalle de impuestos dentro de la aplicación.

Figura 49. Diagrama de la clase CargoRubroImpDataTable

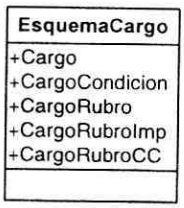
CargoRubroCCDataTable. Este es el datatable que almacena la información del detalle de centros de costo dentro de la aplicación.

Figura 50. Diagrama de la clase CargoRubroCCDataTable



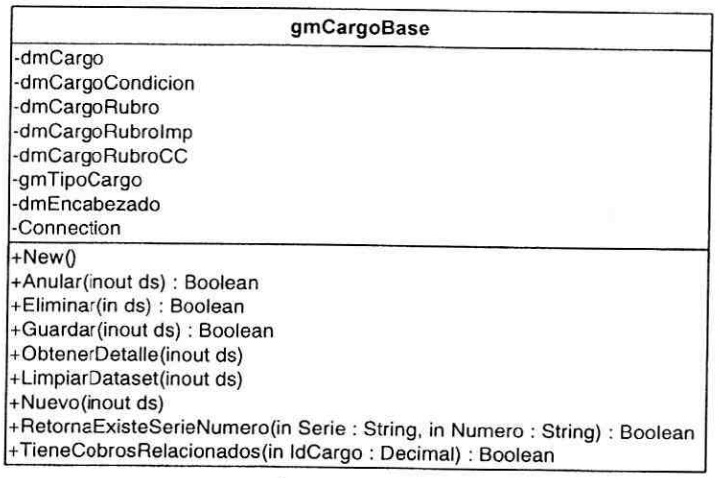

EsquemaCargo. Este es el dataset que maneja toda la información de los cargos dentro de la aplicación.

Figura 51. Diagrama de la clase EsquemaCargo




gmCargoBase. Esta clase se encarga de administrar toda la información de los cargos.

Figura 52. Diagrama de la clase gmCargoBase



El procedimiento New es el constructor de la clase, se encarga de inicializar los atributos de la clase.

La función Anular cambia el estado del registro contenido en el dataset que recibe a anulado, regresando un valor bolean indicando su éxito.

La función Eliminar elimina de la base de datos la información contenida en el dataset. Regresa un valor bolean indicando su éxito.

La función Guardar ingresa o modifica en la base de datos la información contenida en el dataset. Regresa un valor bolean para indicar su éxito.

El procedimiento ObtenerDetalle alimenta el dataset con el detalle de condiciones de cobros, rubros, impuestos y centros de costo asociados al documento de cargo.

El procedimiento LimpiarDataset elimina toda información que esta contenida en el dataset recibido.

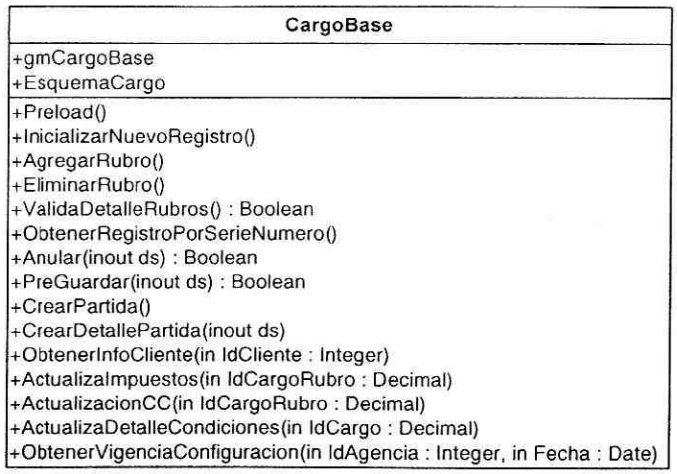
El procedimiento Nuevo crea un nuevo registro en el dataset que recibe, inicializando campos como el usuario, fecha de ingreso, etc.

La función RetornaExisteSerieNumero manda a llamar a la función del dmCargo ExisteSerieNumero.

La función TieneCobrosRelacionados manda a llamar a la función del dmCargo TieneCobrosRelacionados.

CargoBase. Esta clase es la encargada de manejar la información general y el detalle de rubros que ingrese el usuario.

Figura 53. Diagrama de la clase CargoBase




La función Preload inicializa los atributos de la clase.

El procedimiento InicializarNuevoRegistro inicializa valores de la clase al momento de crearse un nuevo registro.

El procedimiento AgregarRubro agrega una nueva línea al detalle de rubros.

El procedimiento EliminarRubro eliminar del detalle de rubros una o más líneas y respectivamente los detalles de impuestos y centros de costo asociados a las líneas.

La función ValidaDetalleRubro revisa todas las líneas del detalle de rubros validando que la información está correcta. Regresa un valor bolean indicando si todas las líneas están bien o existe alguna incorrecta.

El procedimiento ObtnerRegistroPorSerieNumero revisa si existe un registro con la serie y número recibidos y de ser así navega a ese registro. Emplea la función del gmCargoBase ObtenerRegistroPorCampos.

La función Anular marca como estado de anulado el registro contenido en el dataset que recibe. Emplea la función del gmCargoBase Anular. Regresa un valor bolean indicando el éxito de la operación.

La función PreGuardar realiza las validaciones de toda la información de cargo, así también calcula las cantidades bases del registro de cargo y la creación de la partida contable de existir el módulo de contabilidad. Regresa un valor indicando si toda la información y cálculos están bien o no.

El procedimiento CreaPartida genera la información de una partida contable empleando la contenida en el cargo.

El procedimiento CreaDetallePartida crea el detalle de cargos y abonos de la partida empleando la información del cargo.

El procedimiento ObtenerInfoCliente obtiene la información genera y contable de un cliente dado el identificador recibido y lo asigna en la información del cargo.

El procedimiento ActualizaImpuestos actualiza los totales impuestos y totales cálculo del detalle de impuestos del identificador recibido, empleando el porcentaje que posee cada línea.

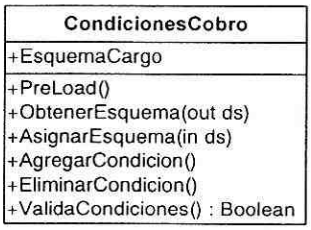
El procedimiento ActualizacionCC actualiza el subtotal del detalle de centros de costo asociado al identificador recibido empleando el porcentaje de cada línea.

El procedimiento ActualizaDetalleCondiciones actualiza el total cobro del detalle de condiciones de cobros del cargo. Emplea un porcentaje manejado lógicamente para realizar esto.

El procedimiento ObtenerVigenciaConfiguracion obtiene de la vigencia de la configuración de cuentas por cobrar al que pertenece el cargo mediante el identificador de la agencia y la fecha.

CondicionesCobro. Esta clase se encarga de manejar la información del detalle de condiciones de cobro del cargo.

Figura 54. Diagrama de la clase CondicionesCobro




El procedimiento PreLoad inicializa los atributos de la clase.

El procedimiento de ObtenerEsquema alimenta el dataset recibido con la información de las condiciones de cobros que ha manejado la clase.

El procedimiento AsignarEsquema alimenta a la clase con la condiciones de cobros ya existentes en el cargo.

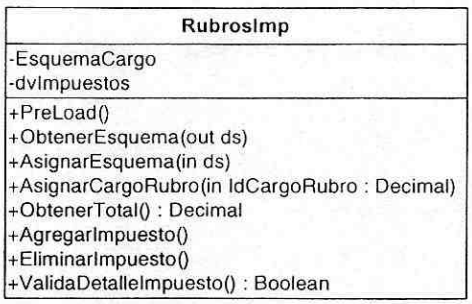
El procedimiento AgregarCondicion crea una nueva línea en las condiciones de cobro.

El procedimiento EliminarCondicion elimina de las condiciones de cobro una o más líneas.

La función ValidaCondiciones revisa todas las condiciones para que tengan la información correcta y verifica que la suma del total cobrar de todas las líneas sea igual al total por cobrar del cargo. Regresa un valor bolean indicando si tota la información esta bien o no.

RubrosImp. Esta clase maneja la información del detalle de impuestos por rubro.

Figura 55. Diagrama de la clase RubrosImp




La función PreLoad es la encargada de inicializar los atributos de la clase.

El procedimiento de ObtenerEsquema alimenta el dataset recibido con la información de los impuestos que ha manejado la clase.

El procedimiento AsignarEsquema alimenta a la clase con la información de impuestos ya existentes en el cargo.

El procedimiento AsignarCargoRubro maneja la información de impuestos a manera de sólo mostrar y manejar el detalle de impuestos asociados al identificador recibido. Para esto emplea la clase dataview dvImpuestos para crear una vista.

La función ObtenerTotal regresa la suma del total impuesto de las líneas del detalle de impuesto que está manejando la clase.

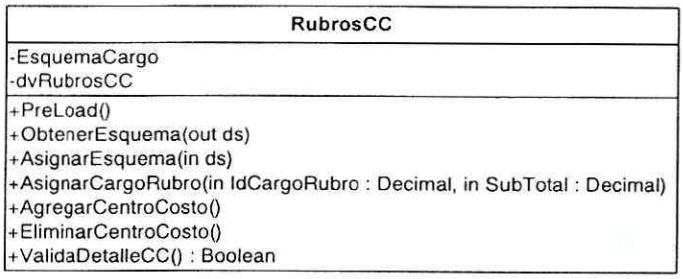
El procedimiento AgregarImpuesto agrega una nueva línea al detalle de impuesto.

El procedimiento EliminarImpuesto elimina una o más líneas del detalle de impuestos.

La función ValidaDetalleImpuesto revisa las líneas del detalle de impuestos asegurando que la información esté correcta. Regresa un valor bolean indicando si la información esta bien o no.

RubrosCC. Esta clase se encarga de manejar el detalle de centros del cargo.

Figura 56. Diagrama de la clase RubrosCC




La función PreLoad es la encargada de inicializar los atributos de la clase.

El procedimiento de ObtenerEsquema alimenta el dataset recibido con la información de los centros de costo que ha manejado la clase.

El procedimiento AsignarEsquema alimenta a la clase con la información de centros de costo ya existentes en el cargo.

El procedimiento AsignarCargoRubro maneja la información de centros de costo a manera de sólo mostrar y manejar el detalle de centros de costo asociados al identificador recibido. Para esto emplea la clase dataview dvImpuestos para crear una vista. Además define la cantidad máxima que puede cubrir el detalle de centros de costo.

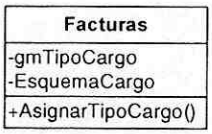
El procedimiento AgregarCentroCosto agrega una nueva línea al detalle de centros de costo.

El procedimiento EliminarCentroCosto elimina una o más líneas del detalle de centros de costo.

La función ValidaDetalleCC revisa las líneas del detalle de centros de costo asegurando que la información este correcta. Además valida que la suma del subtotal del detalle sea igual al subtotal del detalle de rubro relacionado. Regresa un valor bolean indicando si la información esta bien o no.

Facturas. Esta clase maneja los cargos de tipo factura. Hereda de la clase CargoBase.

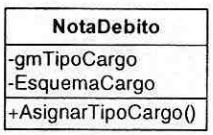
Figura 57. Diagrama de la clase Facturas




El procedimiento AsignarTipoCargo define que el documento de cargo es de tipo factura.

NotaDebito. Esta clase maneja los cargos de tipo nota de débito. Hereda de la clase CargoBase.

Figura 58. Diagrama de la clase NotaDebito




El procedimiento AsignarTipoCargo define que el documento de cargo es de tipo factura.

Cargos. Esta clase define las interfaces para que un módulo externo a cuentas por cobrar pueda generar facturas de cuentas por cobrar.

Figura 59. Diagrama de la clase Cargos

La clase define dos interfaces:

- IDataCargo. Esta interfase debe ser empleada por el módulo que desea realizar la factura, conforma la información que se pasara a las cuentas por cobrar para que con ella se genere la factura de cuentas por cobrar. Dentro de la información que se envía está la agencia, la condición de cobro, el cliente, el vendedor, el cobrador, la descripción, la serie, el número, el nombre o razón social, la dirección, la identificación tributaria, la fecha, la moneda, la tasa, el subtotal, el total, el total de impuestos y el detalle del cargo (detalle de rubros, impuestos, centros de costo y condiciones de cobro).

- ICargo. esta interfase es implementada por cuentas por cobrar y define las operaciones que un módulo externo puede realizar con las facturas de cuentas por cobrar, como lo es obtener la información de una factura dado un identificador, anular una factura, ingresar o modificar una factura y eliminar una factura

CargosDocumentImport. Esta clase es la que implementa la interfase de ICargo en las cuentas por cobrar.

Figura 60. Diagrama de la clase CargosDocumentImport


El procedimiento New es el constructor de la clase. Inicializa sus atributos.

El procedimiento ImportandoCargo valida la información contenida en IDataCargo y con ella construye la información de la factura de cuentas por cobrar.

El procedimiento ImportandoDetalleCargo construye el detalle de rubros, impuestos, centros de costo y condiciones de cobro a partir de la información en IDataCargo. Este procedimiento es llamado por ImportandoCargo.

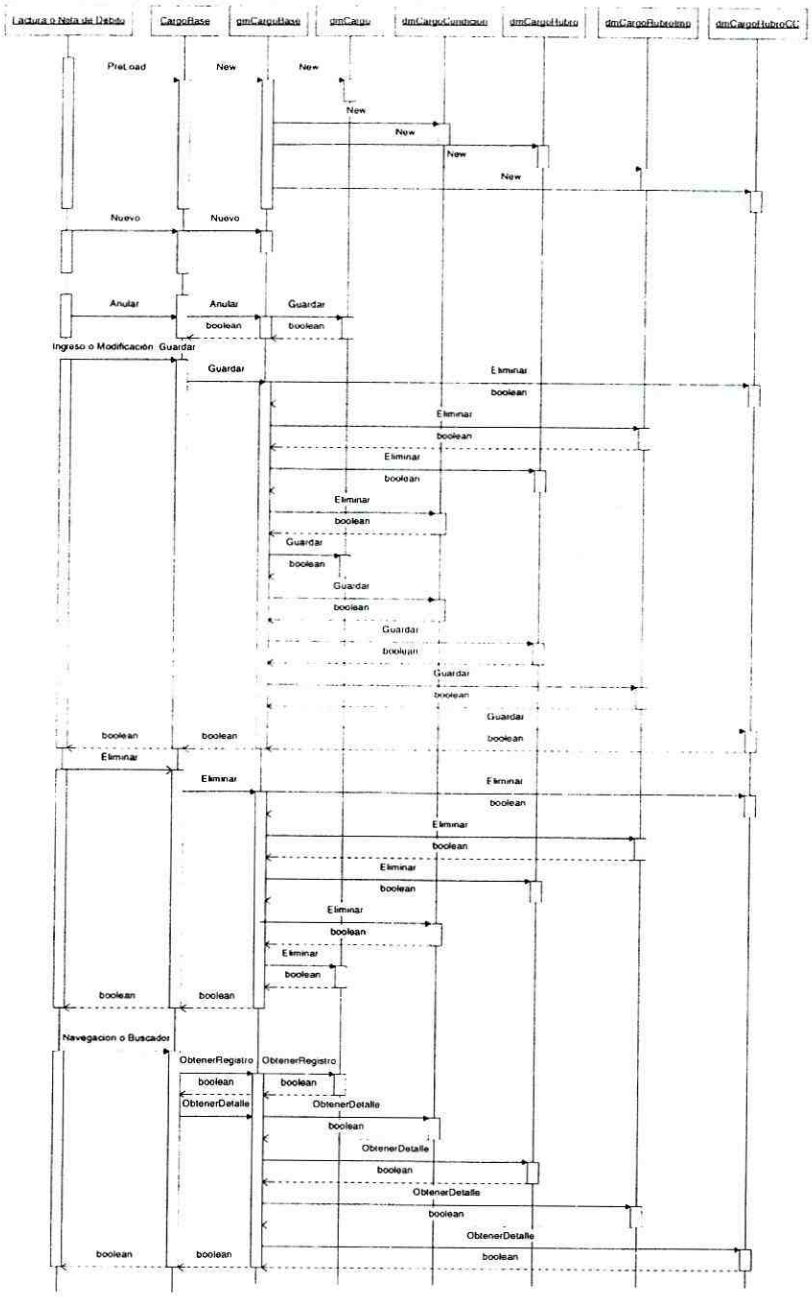
El procedimiento guardar se encarga de ingresar o modificar una factura de cuentas por cobrar a la base de datos. Cuando se realiza una modificación primero se debe llamar a IrACargo para después actualizar la información de la factura con la contenida en IDataCargo.

El procedimiento anular anula la factura de cuentas por cobrar asociada al identificador recibido.

El procedimiento eliminar elimina de la base de datos la factura de cuentas por cobrar asociada al identificador recibido.

3) Diagrama de secuencia

Figura 61. Diagrama de secuencia de facturas y notas de débito



4) Interfaz gráfica de usuario

Figura 62. Interfaz gráfico para la información general de los cargos

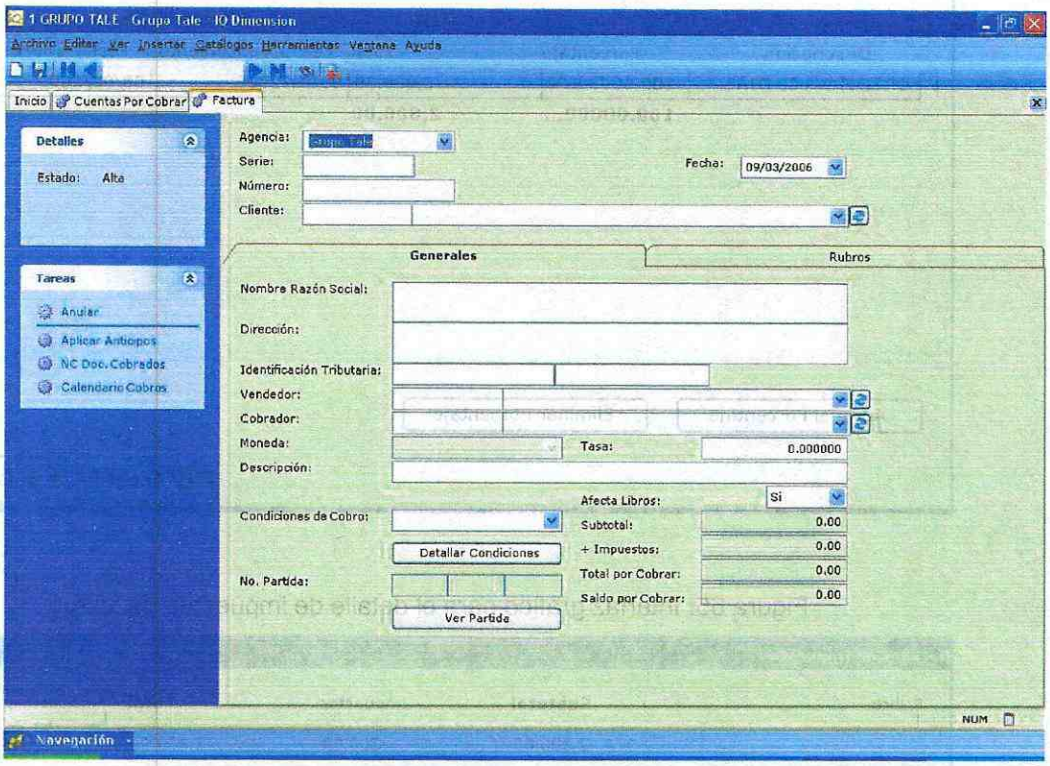



Figura 63. Interfaz gráfico para el detalle de rubros de los cargos

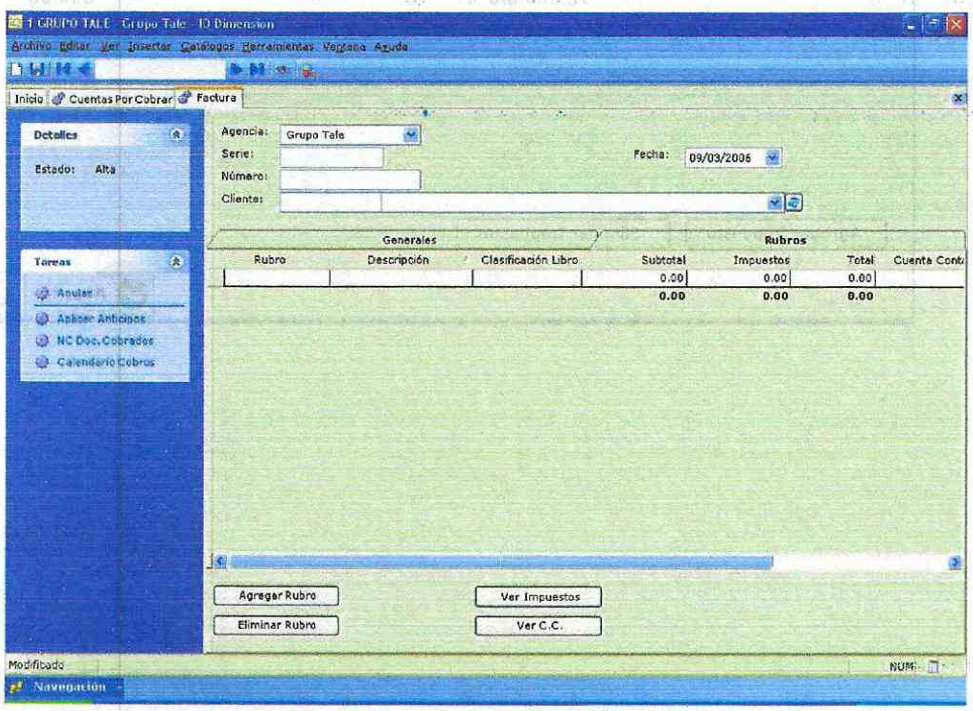


Figura 64. Interfaz gráfico para las condiciones de cobro del cargo

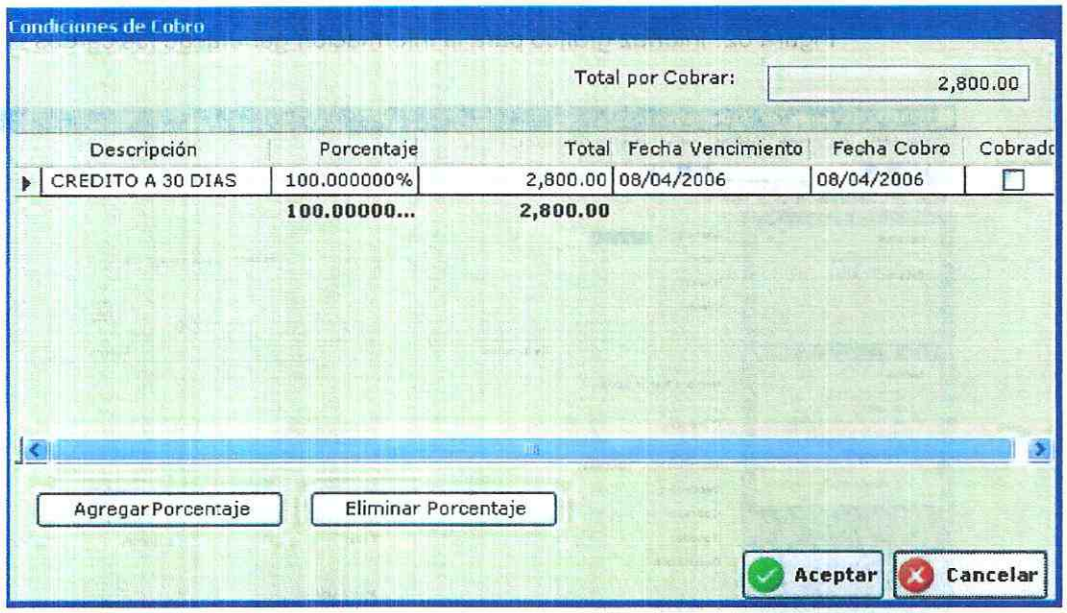



Figura 65. Interfaz gráfico para el detalle de impuestos del cargo.

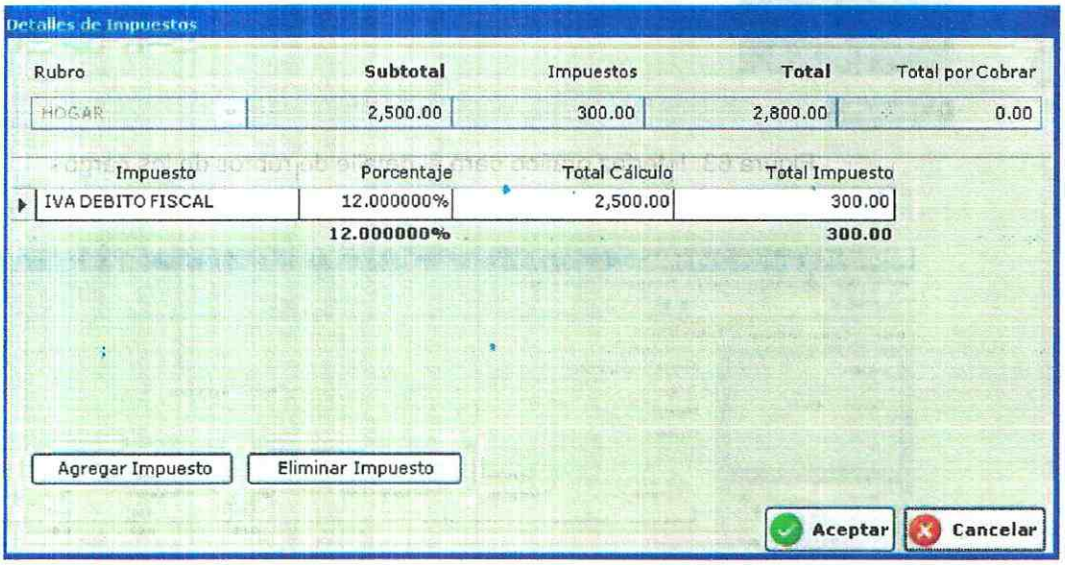


Figura 66. Interfaz gráfico para el detalle de centros de costo del cargo.

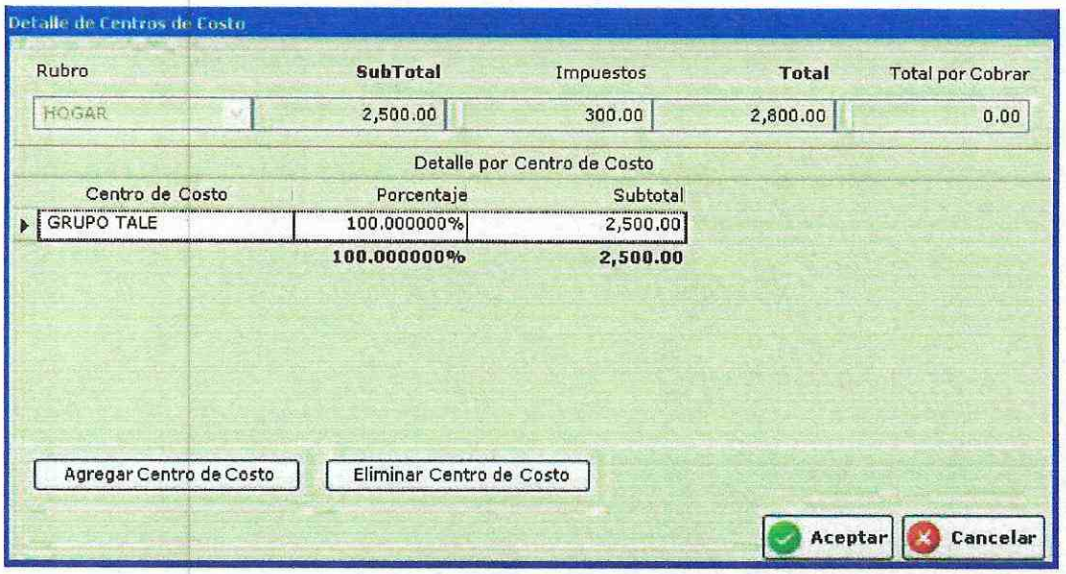
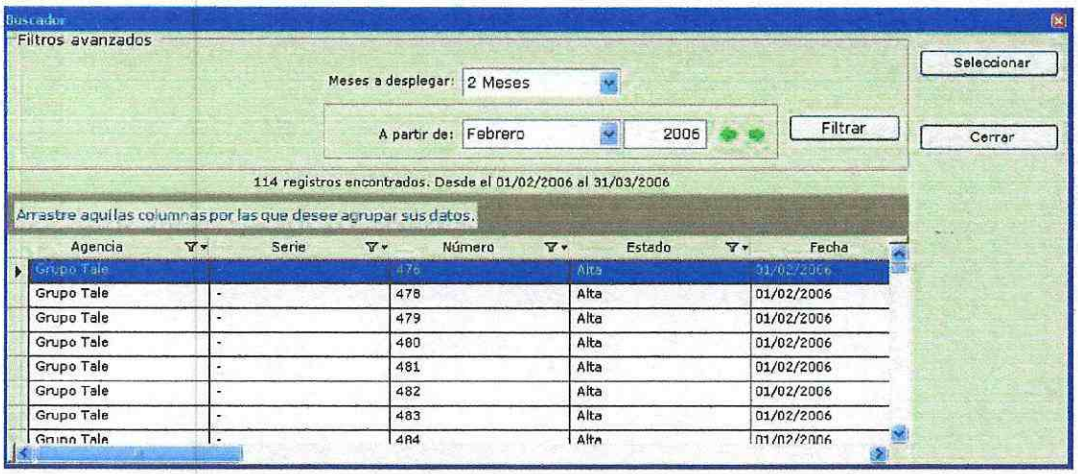


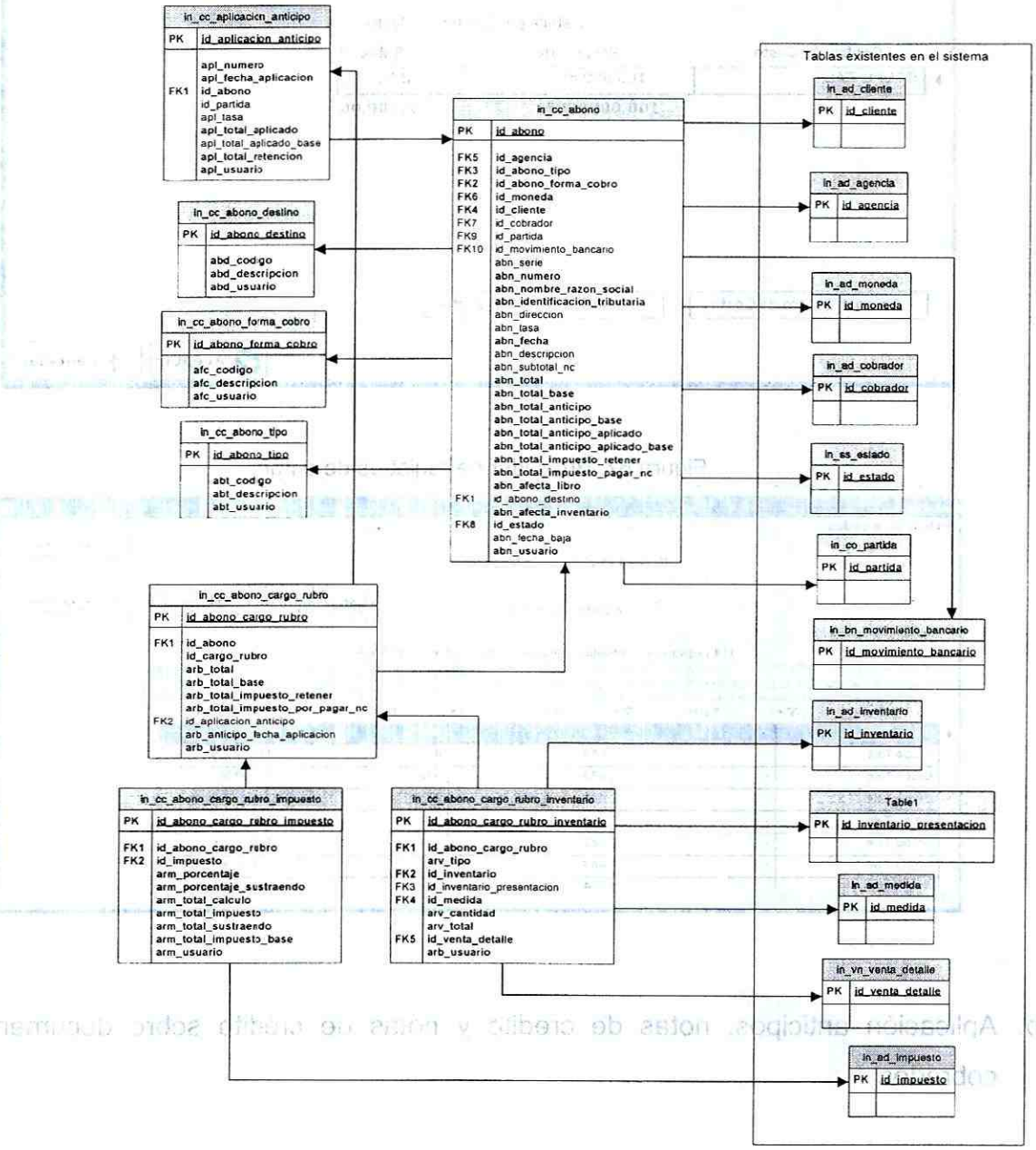

Figura 67. Buscador de registros de cargos




b. Aplicación anticipos, notas de crédito y notas de crédito sobre documentos cobrados

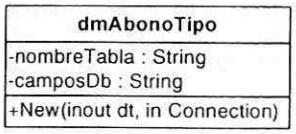
## 1) Diagrama de base de datos

Figura 68. Diagrama de la base de datos de abonos



2) Diagrama de Clases. dmAbonoTipo. Es la clase que se emplea para interactuar con la tabla in_cc_abono_tipo en la base de datos.

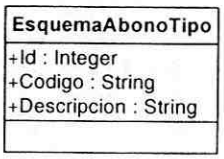
Figura 69. Diagrama de la clase dmAbonoTipo




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejara y la conexión que utilizará.

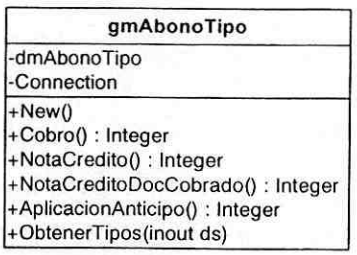
EsquemaAbonoTipo. Este es el dataset que maneja la información de los tipos de abonos dentro de la aplicación.

Figura 70. Diagrama de la clase EsquemaAbonoTipo




gmAbonoTipo. Esta clase es la que administra la información de los tipos de abonos.

Figura 71. Diagrama de la clase gmAbonoTipo




El procedimiento New es el constructor de la clase. Inicializa los atributos.

El procedimiento llamado Cobro devuelve el identificador asociado al tipo de abono cobros.

El procedimiento NotaCredito devuelve el identificador asociado al tipo de abono notas de crédito.

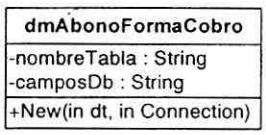
El procedimiento NotaCreditoDocCobrado devuelve el identificador asociado al tipo de abono notas de crédito sobre documento cobrado.

El procedimiento AplicacinAnticipo devuelve el identificador asociado al tipo de abono aplicación de anticipos

El procedimiento ObtenerTipo alimenta el dataset ds con todos los tipos de abonos existentes en el sistema.

dmAbonoFormaCobro. Es la clase que se emplea para interactuar con la tabla in_cc_abono_forma_cobro en la base de datos.

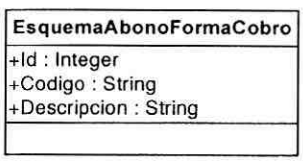
Figura 72. Diagrama de la clase dmAbonoFormaCobro




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

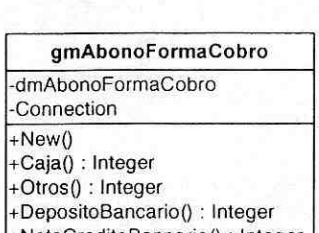
EsquemaAbonoFormaCobro. Este es el dataset que maneja la información de las formas de cobro del abono dentro de la aplicación.

Figura 73. Diagrama de la clase EsquemaAbonoFormaCobro




gmAbonoFormaCobro. Esta clase es la que administra la información de las formas de cobro de abonos.

Figura 74. Diagrama de la clase gmAbonoFormaCobro




El procedimiento New es el constructor de la clase. Inicializa los atributos.

El procedimiento llamado Caja devuelve el identificador asociado a la forma de cobro con nombre de "Caja".

El procedimiento llamado Otros devuelve el identificador asociado a la forma de cobro con nombre de "Otros".

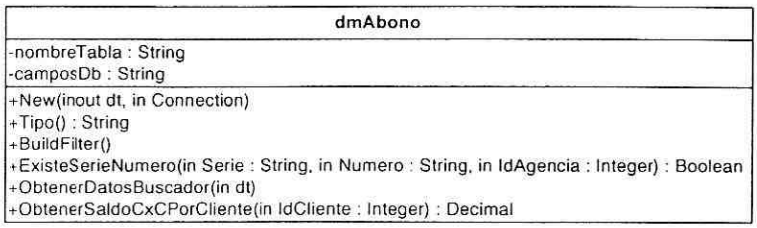
El procedimiento DepositoBancario devuelve el identificador asociado a la forma de cobro con nombre de "Deposito Bancario".

El procedimiento NotaCreditoBancario devuelve el identificador asociado a la forma de cobro con nombre de "Nota de Crédito Bancario".

El procedimiento ObtenerFormaCobro alimenta el dataset ds con todas las formas de cobro de abonos existentes en el sistema.

dmAbono. Es la clase que se emplea para interactuar con la tabla in_cc_abono en la base de datos.

Figura 75. Diagrama de la clase dmAbono




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

El procedimiento Tipo regresa el tipo de documento de abono que la clase está manejando.

El procedimiento BuildFilter construye un filtro para manejar solamente los registros del tipo de abono que está manejando la clase.

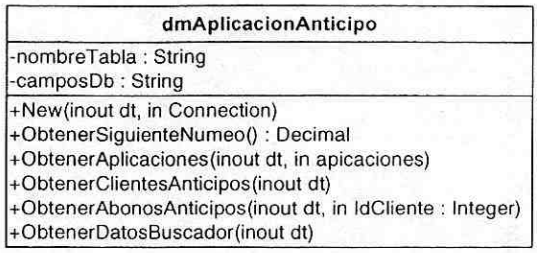
La función ExisteSerieNumero indica si para la agencia, serie y el número que se reciben ya existen en el sistema.

El procedimiento ObtenerDatosBuscador alimenta la datatable dt con los registros existentes en el sistema del tipo de abono que está manejando la clase.

La función ObtenerSaldoCxCPorCliente devuelve el saldo de los documentos de cargo de cuentas por cobrar asociado al cliente que se recibe.

dmAplicacionAnticipo. Es la clase que se emplea para interactuar con la tabla in_cc_aplicacion_anticipo en la base de datos.

Figura 76. Diagrama de la clase dmAplicacionAnticipo




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerSiguienteNumero devuelve el número que toca asignar a la aplicación del anticipo.

El procedimiento ObtenerAplicaciones alimenta la datatable dt con los registros de aplicación anticipos cuyos identificadores se encuentren en el arreglo de identificadores aplicaciones.

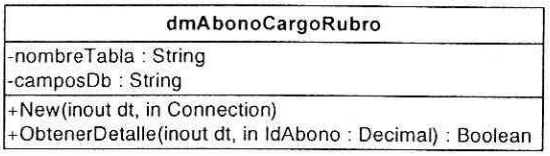
El procedimiento ObtenerClienteAnticipos alimenta la datatable dt con los clientes que poseen un anticipo para poder aplicar.

El procedimiento ObtenerAbonosAnticipos devuelve los abonos que tienen un saldo de anticipo por aplicar del cliente que se recibe.

El procedimiento ObtenerDatosBuscador alimenta la datatable dt con los registros existentes en el sistema de aplicaciones de anticipos.

dmAbonoCargoRubro. Es la clase que se emplea para interactuar con la tabla in_cc_abono_cargo_rubro en la base de datos.

Figura 77. Diagrama de la clase dmAbonoCargoRubro

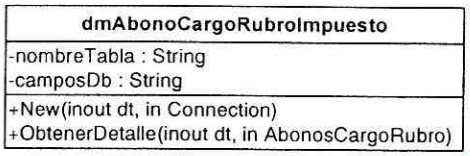



El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros que estén asociados al identificador del abono recibido.

dmAbonoCargoRubroImpuesto. Es la clase que se emplea para interactuar con la tabla in_cc_abono_cargo_rubro_impuesto en la base de datos.

Figura 78. Diagrama de la clase dmAbonoCargoRubroImpuesto

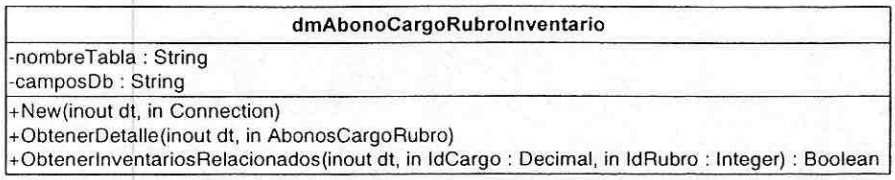



El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros que estén asociados a los identificadores de abono cargo rubro contenidos en el arreglo AbonosCargoRubro.

dmAbonoCargoRubroInventario. Es la clase que se emplea para interactuar con la tabla in_cc_abono_cargo_rubro_inventario en la base de datos.

Figura 79. Diagrama de la clase dmAbonoCargoRubroInventario



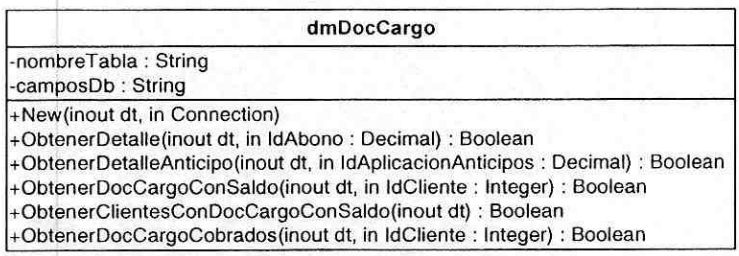

El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros que estén asociados a los identificadores de abono cargo rubro contenidos en el arreglo AbonosCargoRubro.

La función ObtenerInventariosRelacionados alimenta la datatable dt con los inventarios asociados al cargo y rubro recibidos.

dmDocCargo. Esta clase se encarga de administrar lógicamente los documentos de cargo asociados a un documento de abono.

Figura 80. Diagrama de la clase dmDocCargo




El procedimiento New corresponde al constructor de la clase. Se le envía la tabla del dataset que la clase manejará y la conexión que utilizará.

La función ObtenerDetalle alimenta la datatable dt con los registros que estén asociados al identificador del abono recibido.

La función ObtenerDetalleAnticipo alimenta la datatable dt con los registros que estén asociados al identificador de la aplicación anticipo recibido.

La función ObtenerDocCargocConSaldo alimenta la datatable dt con los documentos de cargo del cliente recibido que aún tengan un saldo pendiente por cobrar.

La función ObtenerClientesConDocCargoConSaldo alimenta la datatable dt con los clientes que tengan documentos de cargos con un saldo pendiente por cobrar.

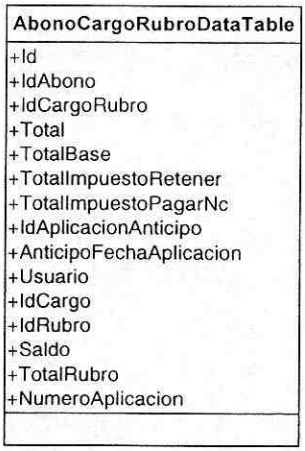
La función ObtenerDocCargoCobrados alimenta la datatable dt con los documentos de cargos del cliente recibido que no tengan un saldo pendiente por cobrar.

AbonoDataTable. Este es el datatable que almacena la información general de los abonos dentro de la aplicación.

Figura 81. Diagrama de la clase AbonoDataTable

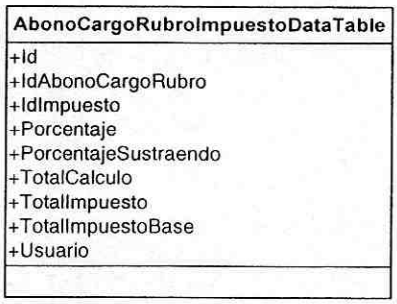
AbonoCargoRubroDataTable. Este es el datatable que almacena la información de los rubros de los cargos que afecta el abono dentro de la aplicación.

Figura 82. Diagrama de la clase AbonoCargoRubroDataTable




AbonoCargoRubroImpuestoDataTable. Este es el datatable que almacena la información del detalle de impuestos por rubro del abono dentro de la aplicación.

Figura 83. Diagrama de la clase AbonoCargoRubroImpuestoDataTable

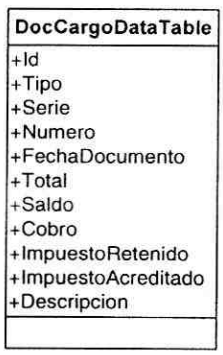



AbonoCargoRubroInventarioDataTable. Este es el datatable que almacena la información del detalle de inventarios por rubro del abono dentro de la aplicación.

Figura 84. Diagrama de la clase AbonoCargoRubroInventarioDataTable


DocCargoDataTable. Este es el datatable que almacena la información de los cargos asociados al documento de abono dentro de la aplicación. Esta información es generada de manera lógica por la aplicación.

Figura 85. Diagrama de la clase DocCargoDataTable

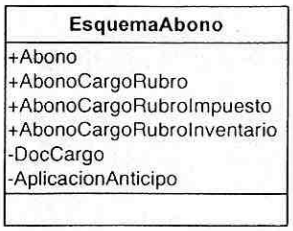



AplicacionAnticipoDataTable. Este es el datatable que almacena la información de la aplicación de anticipos dentro de la aplicación.

Figura 86. Diagrama de la clase AplicacionAnticipoDataTable

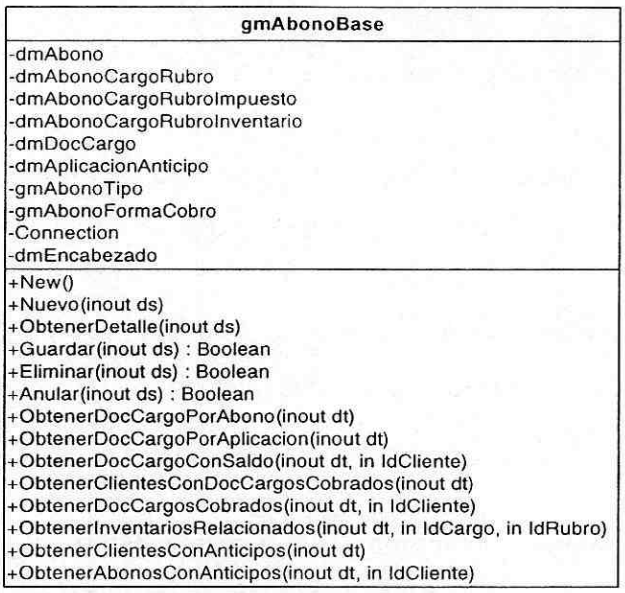
EsquemaAbono. Este es el dataset que maneja toda la información de los abonos dentro de la aplicación.

Figura 87. Diagrama de la clase EsquemaAbono




gmAbonoBase. Esta clase se encarga de administrar toda la información de los abonos.

Figura 88. Diagrama de la clase gmAbonoBase




El procedimiento New es el constructor de la clase, se encarga de inicializar los atributos de la clase.

La función anular cambia el estado del registro contenido en el dataset que recibe a anulado, regresando un valor bolean indicando su éxito.

La función eliminar elimina de la base de datos la información contenida en el dataset. Regresa un valor bolean indicando su éxito.

La función guardar ingresa o modifica en la base de datos la información contenida en el dataset. Regresa un valor bolean para indicar su éxito.

El procedimiento ObtenerDetalle alimenta el dataset con el detalle de rubros, impuestos, inventarios y documentos de cargos asociados al documento de abono.

El procedimiento nuevo crea un nuevo registro en el dataset que recibe.

El procedimiento ObtenerDocCargoPorAbono alimenta la datatable dt con los documentos de cargos asociados al abono recibido.

El procedimiento ObtenerDocCargoPorAplicacion alimenta la datatable con los documentos de cargos asociados a la aplicación de anticipo recibido.

El procedimiento ObtenerDocCargoConSaldo alimenta la datatable dt con los documentos de cargos asociados al cliente recibido que tengan un saldo pendiente por cobrar.

El procedimiento ObtenerClientesConDocCargosCobrados alimenta la datatable dt con los clientes que tienen asociados documentos de cargos que no tienen saldo pendiente de cobrar.

El procedimiento ObtenerDocCargosCobrados alimenta la datatable dt con los documentos de cargos asociados al cliente recibido que no tienen un saldo pendiente de cobrar.

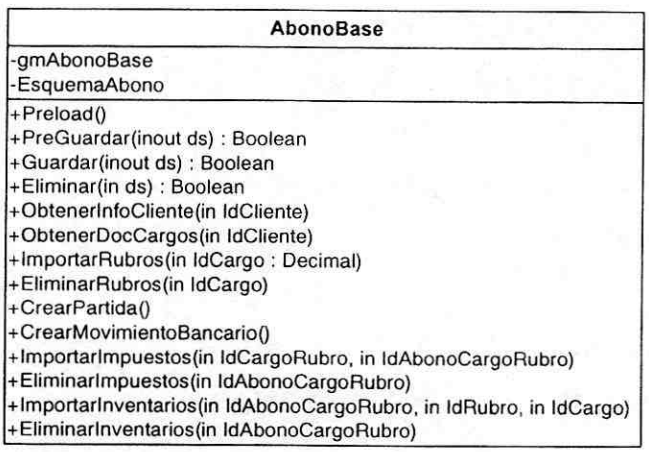
El procedimiento ObtenerInventariosRelacionados alimenta la datatable dt con los inventarios asociados al rubro y documento de cargo recibidos.

El procedimiento ObtenerClienteConAnticipos alimenta la datatable dt con los clientes que poseen un saldo de anticipo pendiente de aplicar.

El procedimiento ObtenerAbonosConAnticipos obtiene los anticipos asociados al cliente recibido que se pueden aplicar.

AbonoBase. Esta clase es la encargada de manejar la información general y los documentos de cargos del abono ingresados por el usuario.

Figura 89. Diagrama de la clase AbonoBase




La función Preload inicializa los atributos de la clase.

La función PreGuardar valida toda la información del abono y realiza los cálculos de montos bases del abono. Regresa un valor bolean indicando si todo esta bien o no.

La función guardar se encarga de guardar el registro en el sistema, empleando al gmAbonoBase, y dependiendo del caso, generar una partida contable o un movimiento bancario. Regresa un valor bolean indicando su éxito.

La función eliminar valida si un registro de abono puede ser eliminado y de serlo emplea al gmAbonoBase para eliminar el registro del sistema.

El procedimiento ObtenerInfoCliente obtiene la información general así como la contable de un cliente.

El procedimiento ObtenerDocCargos se encarga de obtener los documentos de cargos asociados a un cliente.

El procedimiento ImportarRubros se encarga de generar el detalle de rubros del abono para un determinado cargo.

El procedimiento EliminarRubros elimina el detalle de rubros del abono para un determinado cargo.

El procedimiento CrearPartida genera la información general de la partida contable a partir de la información del abono.

El procedimiento CrearMovimientoBancario genera toda la información del movimiento bancario en base a la información del abono.

El procedimiento ImportarImpuestos genera el detalle de impuestos para una línea del detalle de rubros específico del abono a partir del detalle de rubro del cargo asociado.

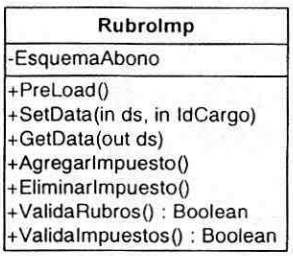
El procedimiento EliminarImpuestos elimina el detalle de impuestos asociado a un detalle de rubro del abono específico.

El procedimiento ImportarInventarios genera el detalle de inventarios para un detalle de rubro del abono determinado a partir del rubro y cargo asociado.

El procedimiento EliminarInventarios elimina el detalle de inventarios para un detalle de rubros del abono determinado.

RubroImp. Esta clase se encarga de manejar el detalle de rubros e impuestos del abono.

Figura 90. Diagrama de la clase RubroImp




El procedimiento PreLoad inicializa los atributos de la clase.

El procedimiento SetData actualiza la información del detalle de rubros, impuestos e inventarios (en el caso de ser nota de crédito o nota de crédito sobre documento cobrado) del abono para un cargo determinado.

El procedimiento GetData retorna toda la información de rubros, impuestos e inventarios (en el caso de ser nota de crédito o nota de crédito sobre documento cobrado) que administro en un momento dado.

El procedimiento AgregaImpuesto agrega una línea al detalle de impuestos. Este procedimiento solamente es usado en la aplicación de anticipos en donde el detalle de impuestos es de impuestos a retener.

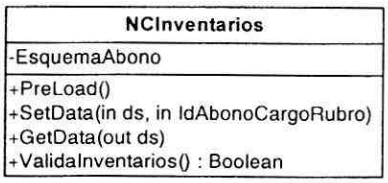
El procedimiento EliminarImpuesto elimina una o más líneas de detalle de impuestos. Este procedimiento solamente es usado en la aplicación de anticipos en donde el detalle de impuestos es de impuestos a retener.

La función ValidaRubros verifica que toda la información del detalle de rubros este correcta.

La función ValidaImpuestos verifica que toda la información del detalle de impuestos este correcta.

NCInventarios. Esta clase administra el detalle de inventarios para un determinado detalle de rubros del abono.

Figura 91. Diagrama de la clase NCInventarios




El procedimiento PreLoad inicializa los atributos de la clase.

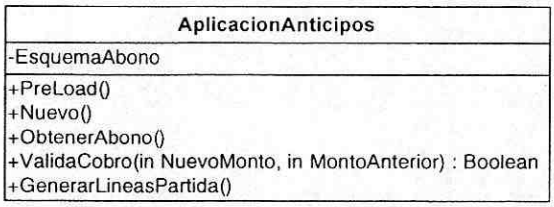
El procedimiento SetData actualiza la información del detalle de inventarios para un determinado detalle de rubro del abono.

El procedimiento GetData devuelve la información del detalle de inventarios que ha manejado hasta el momento.

La función ValidaInventarios verifica que al información del detalle de inventarios este correcta.

AplicacionAnticipos. Esta clase maneja los procesos de abono de tipo aplicación de anticipos. Hereda de AbonoBase. También asigna el tipo de abono al que corresponde al proceso.

Figura 92. Diagrama de la clase AplicacionAnticipos




El procedimiento PreLoad inicializa los atributos de la clase.

El procedimiento nuevo realiza el proceso de pedir al usuario a qué cliente desea aplicar un anticipo y qué anticipo desea aplicar.

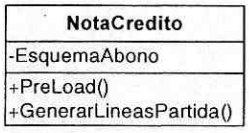
El procedimiento ObtenerAbono obtiene el anticipo que se desea aplicar y alimenta el EsquemaAbono con esa información.

La función ValidaCobro verifica que el cobro que se quiere realizar sobre un documento de cargo sea válido acorde al saldo por cobrar del cargo y al saldo del anticipo.

El procedimiento de GeneraLineasPartida crea el detalle de cargos y abono de la partida contable.

NotaCredito. Esta clase maneja los procesos de abono de tipo notas de crédito. Hereda de AbonoBase.

Figura 93. Diagrama de la clase NotaCredito

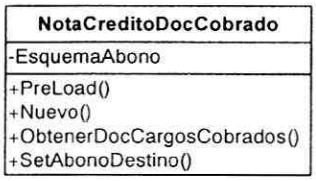


El procedimiento PreLoad inicializa los atributos de la clase. También asigna el tipo de abono que corresponde al proceso.

El procedimiento GenerarLineasPartida genera el detalle de cargos y abono de la partida contable.

NotaCreditoDocCobrado. Esta clase maneja los procesos de abono de tipo nota de crédito sobre documento cobrado. Hereda de NotaCredito.

Figura 94. Diagrama de la clase NotaCreditoDocCobrado




El procedimiento PreLoad inicializa los atributos de la clase y asigna el tipo de abono que corresponde al proceso.

El procedimiento nuevo realiza el proceso de pedir al usuario a qué cliente desea realizar la nota de crédito y luego sobre qué documento de cargo desea realizarlo.

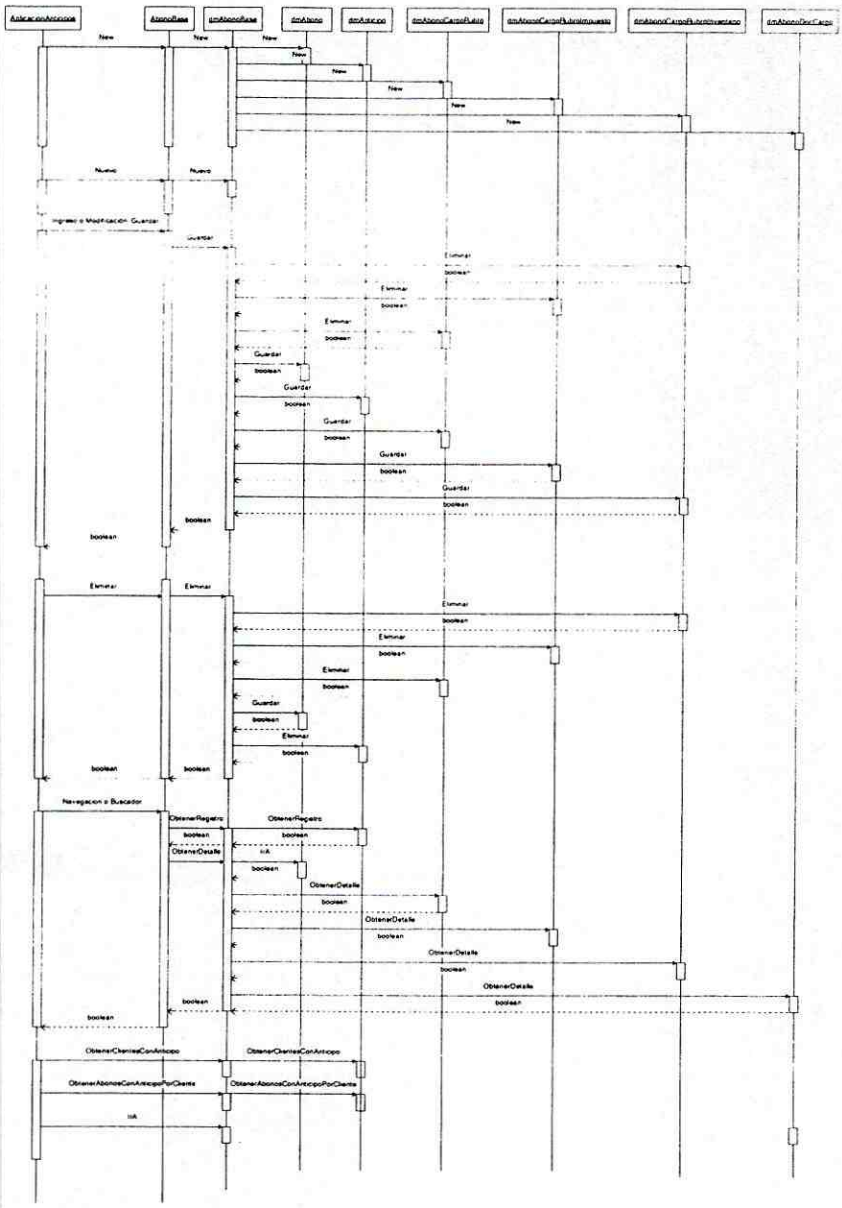
El procedimiento ObtenerDocCargosCobrados alimenta al EsquemaAbono con la información del documento de cargo sobre el cual se realizará la nota de crédito.

El procedimiento SetAbonoDestino es el que define si el proceso realizará una partida contable o un movimiento bancario.

3) Diagrama de secuencias

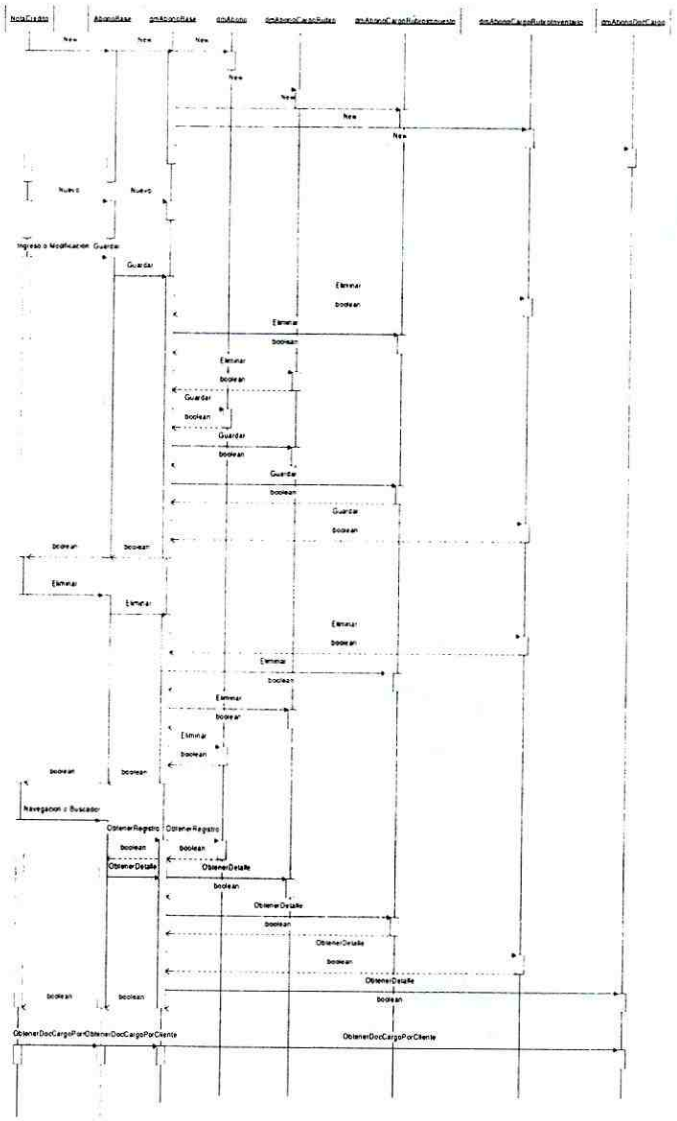
Aplicación anticipos:

Figura 95. Diagrama de secuencia del proceso de aplicación de anticipos



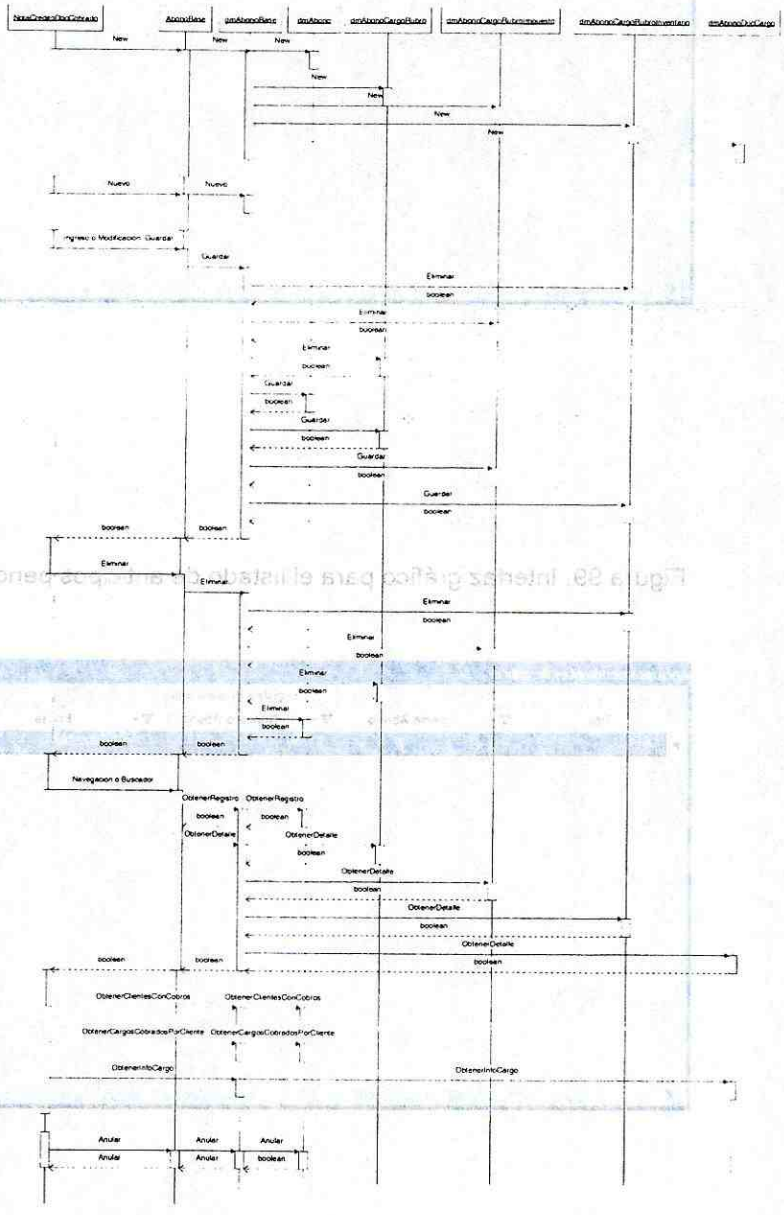
Nota de crédito:

Figura 96. Diagrama de secuencia del proceso de notas de crédito



Nota de crédito sobre documento cobrado:

Figura 97. Diagrama de secuencia del proceso de nota de crédito sobre documento cobrado



## 4) Interfaz gráfica de usuario

Figura 98. Interfaz gráfico para el listado de clientes con anticipos pendientes de aplicar

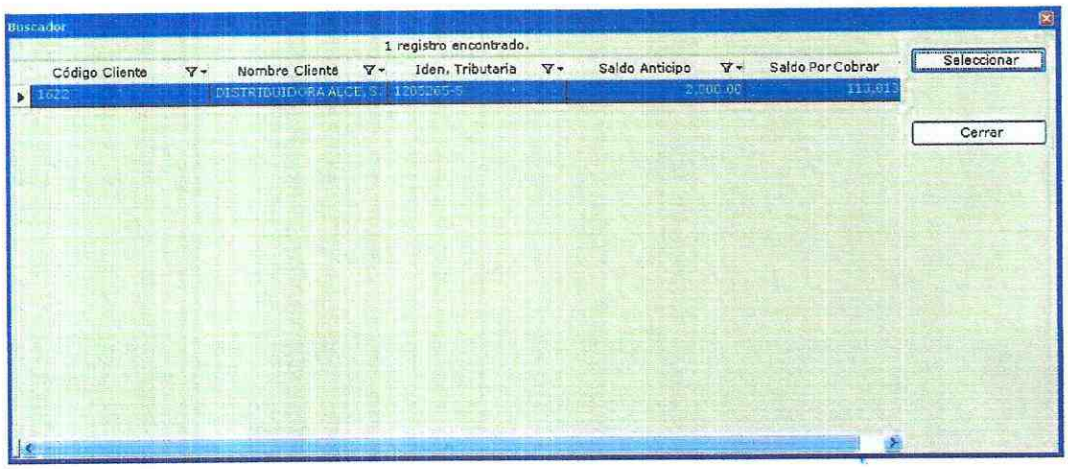



Figura 99. Interfaz gráfico para el listado de anticipos pendientes de aplicar de un cliente

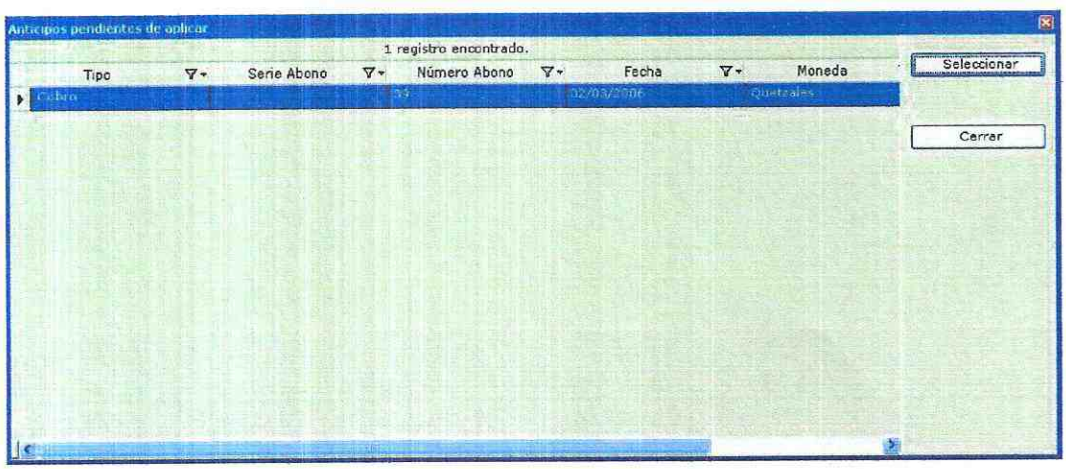


Figura 100. Interfaz gráfico de la información general de la aplicación de anticipos

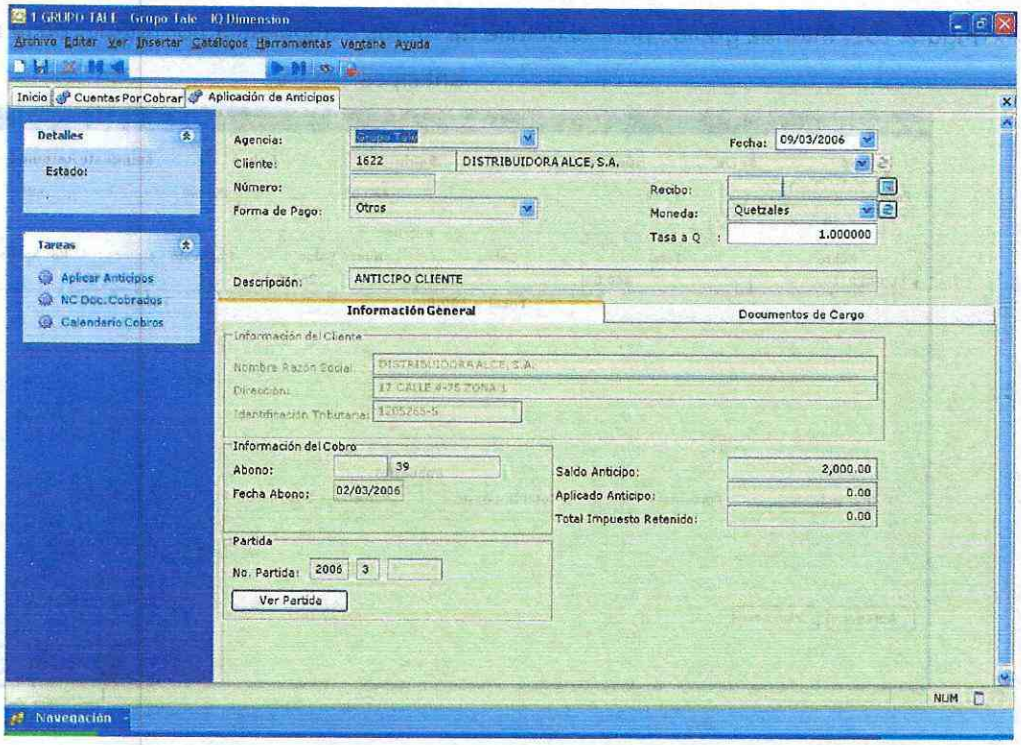



Figura 101. Interfaz gráfico del detalle de cargos en la aplicación de anticipos

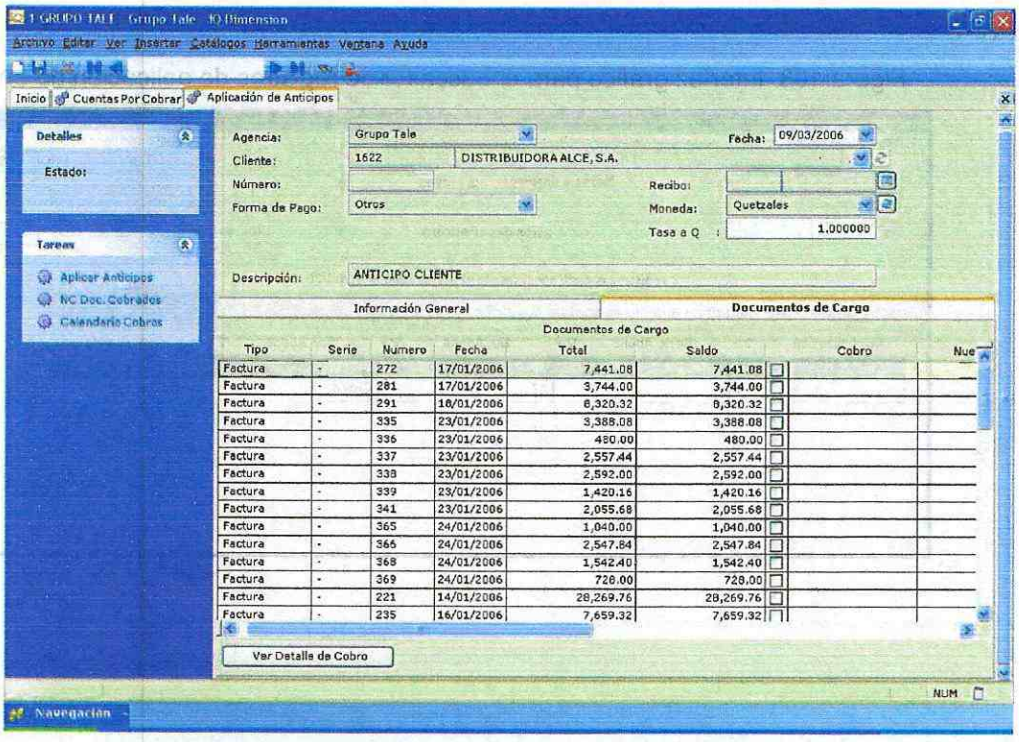


Figura 102. Interfaz gráfico para el detalle de rubros e impuestos por retener de la aplicación de anticipos

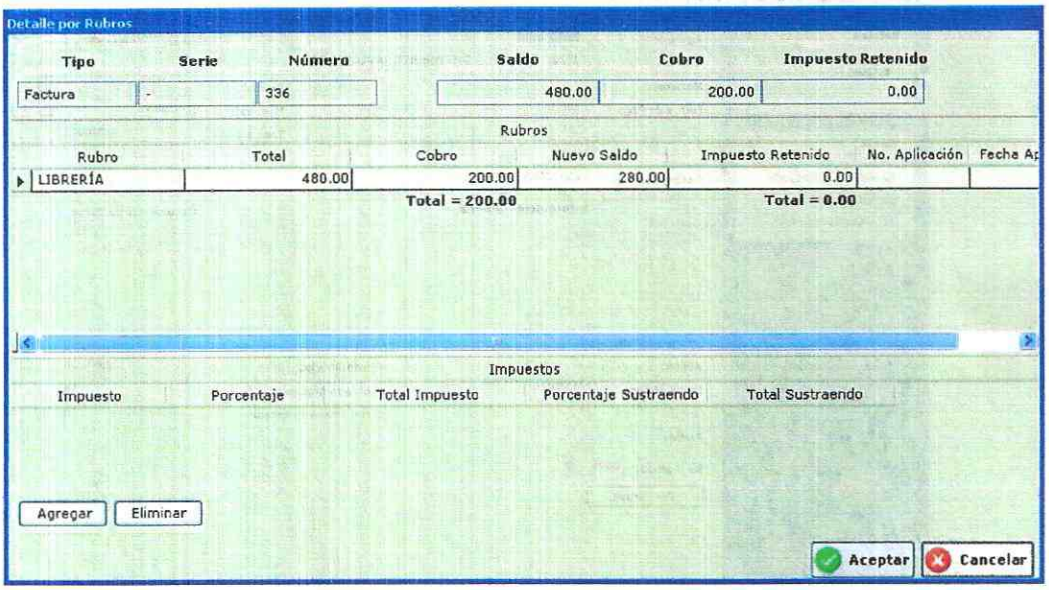



Figura 103. Interfaz gráfico para el buscador de registros de aplicación de anticipos.

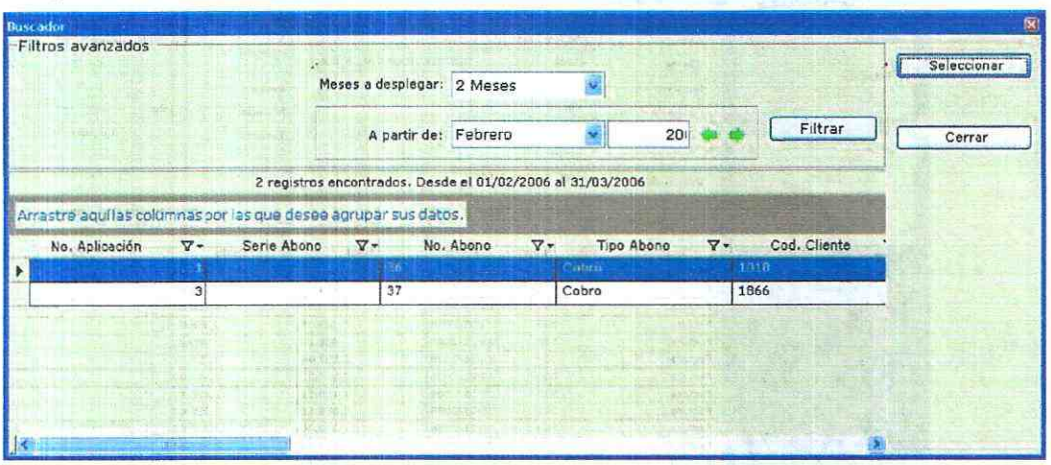


Figura 104. Interfaz gráfico para la información general de las notas de crédito.

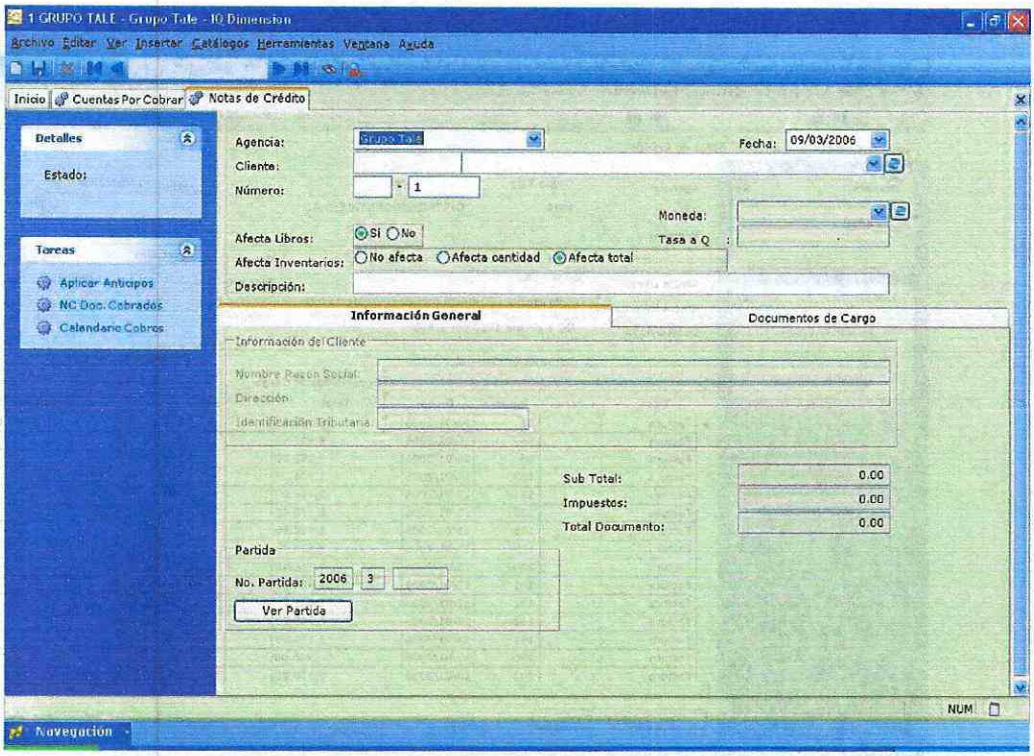



Figura 105. Interfaz gráfico para el detalle de cargos de las notas de crédito

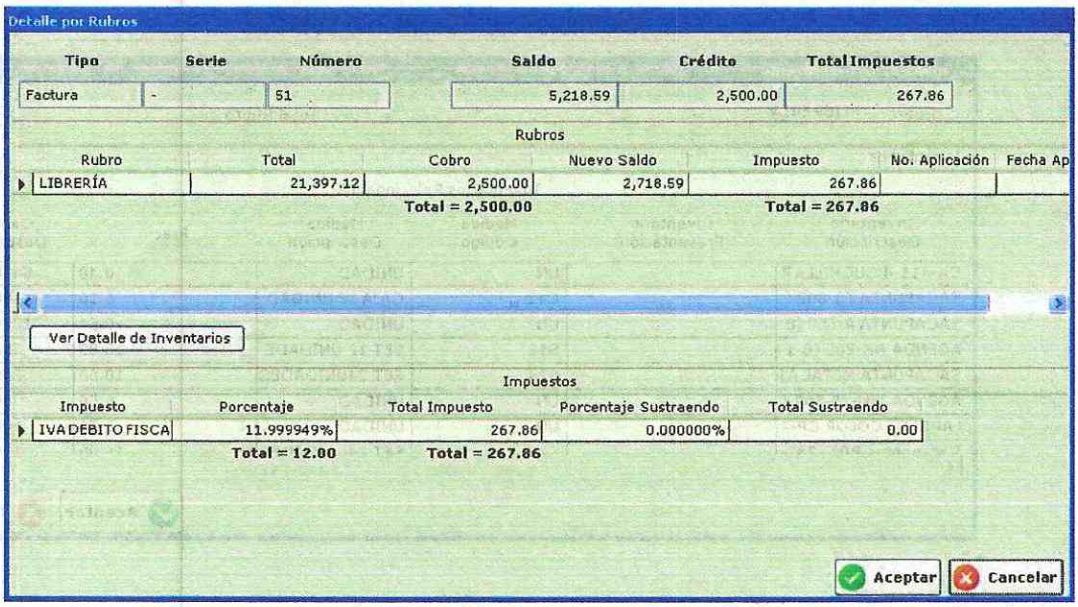


Figura 106. Interfaz gráfico para el detalle de rubros e impuestos de las notas de crédito y notas de crédito sobre documento cobrado.

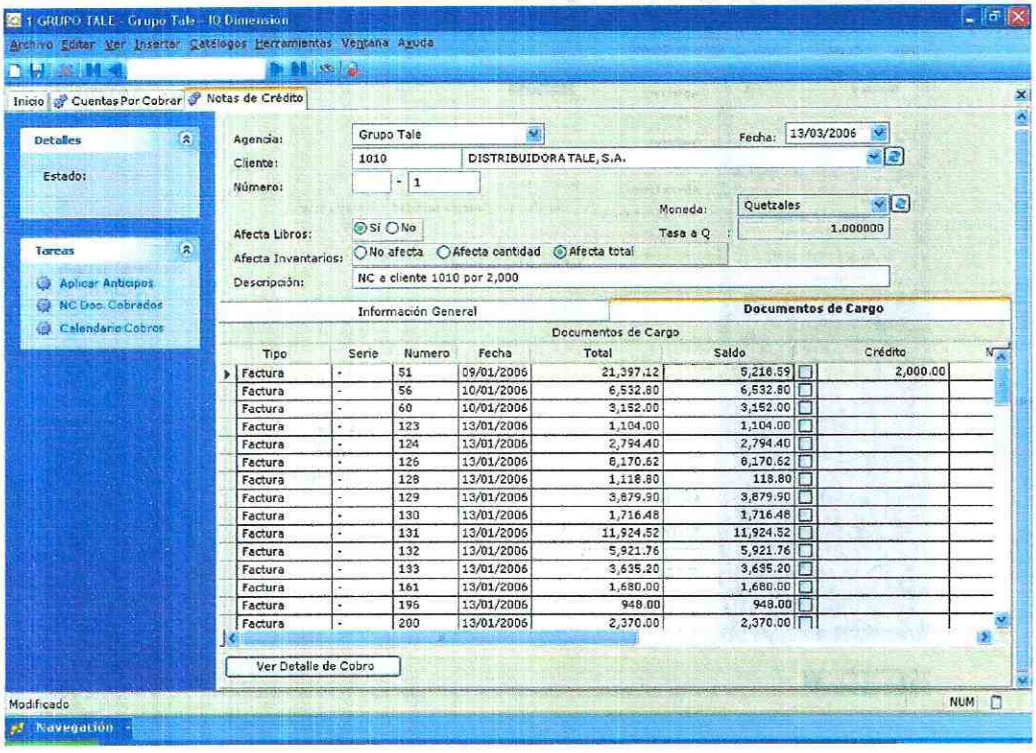



Figura 107. Interfaz gráfico para el detalle de inventarios de las notas de crédito y notas de crédito sobre documento cobrado

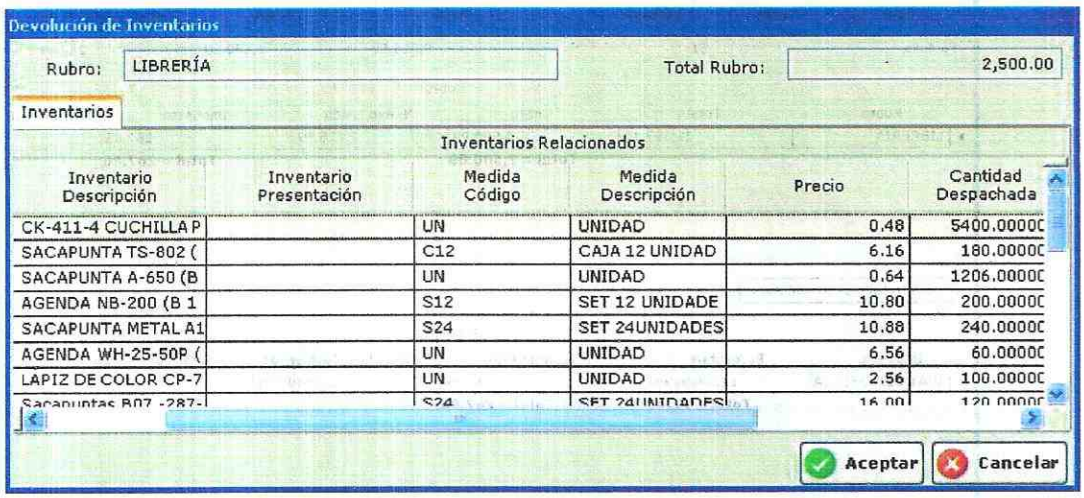


Figura 108. Interfaz gráfico para el buscador de registros de notas de crédito.

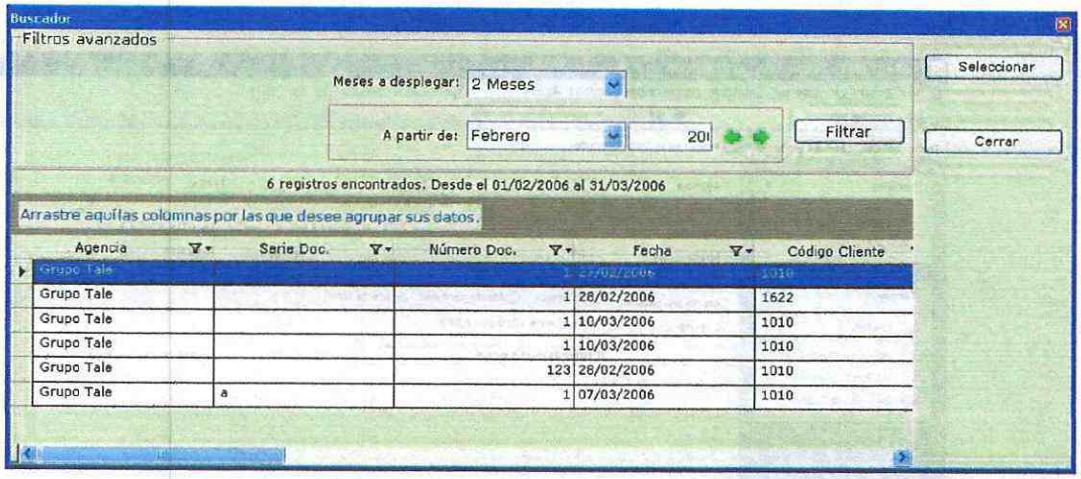



Figura 109. Interfaz gráfico para el listado de clientes con documentos de cargos cobrados.


Figura 110. Interfaz gráfico para el listado de documentos de cargo cobrados de un cliente.

Figura 111. Interfaz gráfico de la información general de las notas de crédito sobre documento cobrado

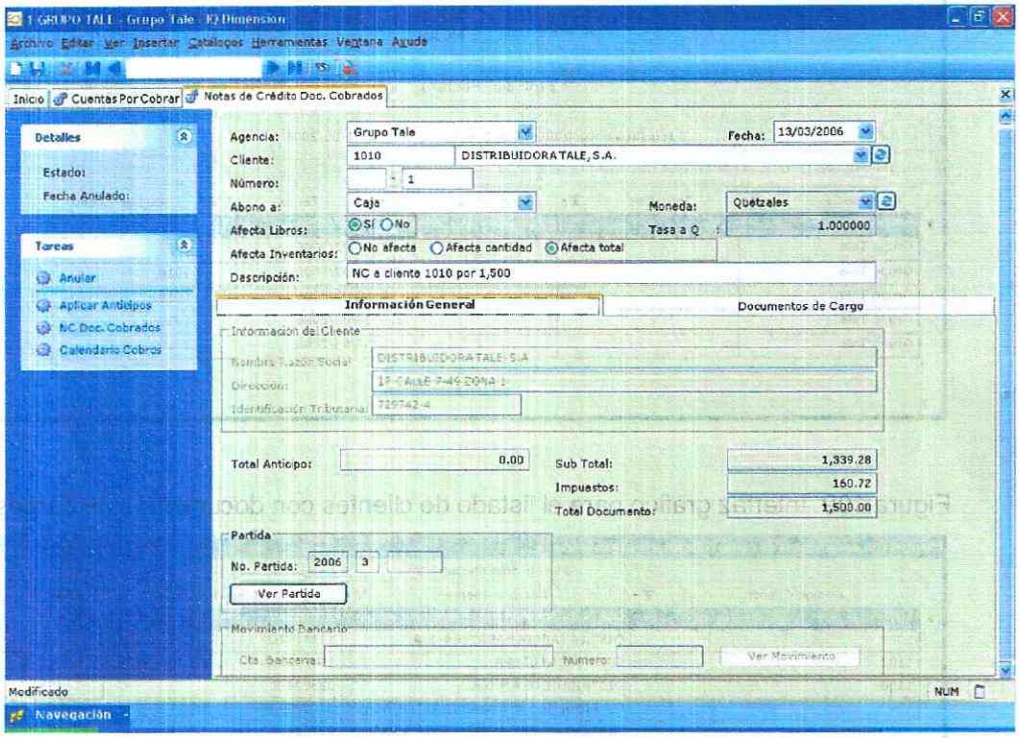



Figura 112. Interfaz gráfico del detalle de documentos de cargo de las notas de crédito sobre documento cobrado

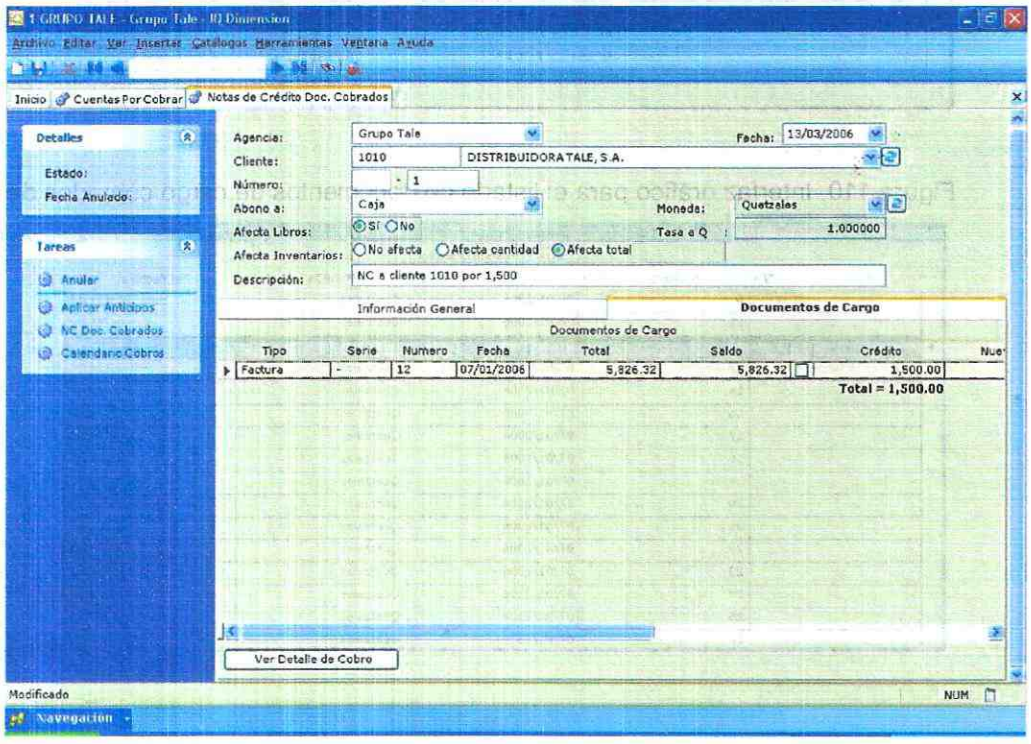
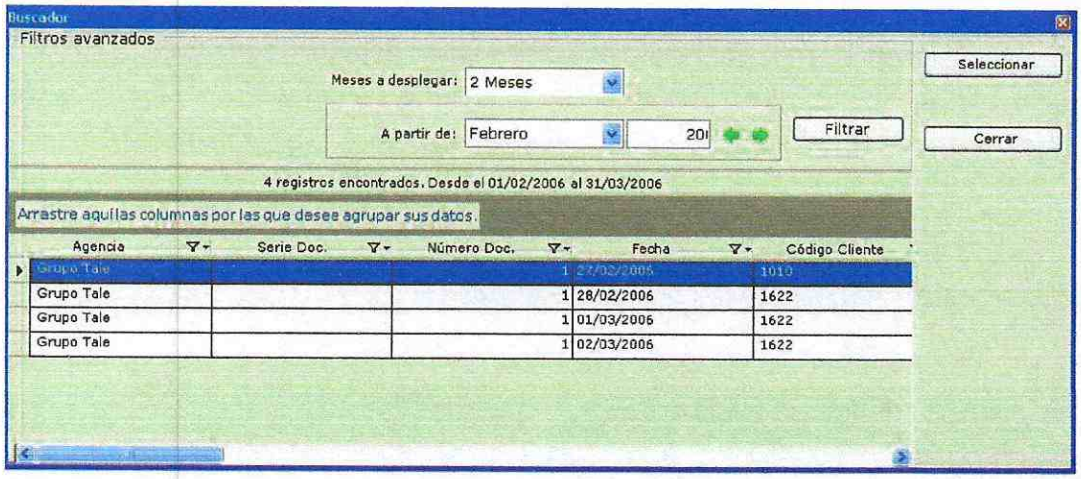


Figura 113. Interfaz gráfico del buscador de registros de notas de crédito sobre documento cobrado




c. Calendario de cobros

1) Diagrama de base de datos. Este proceso no cuenta con tablas en la base de datos. El proceso obtiene la información lógicamente empleando las tablas en la base de datos de los cargos de cuentas por cobrar.

2) Diagrama de clases. dmCalendarioCobros. Esta clase es la encargada de obtener la información para el calendario a través de los registros de cargos de cuentas por cobrar.

Figura 114. Diagrama de la clase dmCalendarioCobros

| **dmCalendarioCobros** |
|---|
| -nombreTabla<br>-camposDb |
| +New(in dt, in Connection)<br>+LlenarCalendario(inout dt, in Fecha)<br>+LlenarCobrosVencidos(in dt, in Fecha)<br>+ObtenerRegistrosImpresion(inout dt, in FechaInicio, in FechaFin) |

El procedimiento New es el constructor de la clase.

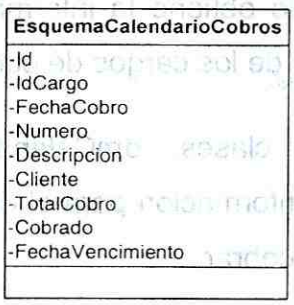
El procedimiento LlenarCalendario alimenta la datatable dt con las condiciones de cobros que están pendientes de cobrar comprendidos en un periodo de dos meses partiendo de la fecha que se recibe para atrás.

El procedimiento LlenarCobrosVencidos alimenta la datatable dt con las condiciones cuyas fechas de vencimiento ya pasaron y siguen pendientes de cobrar en un periodo de dos meses partiendo de la fecha que se recibe para atrás.

El procedimiento ObtenerRegistrosImpresion alimenta la datatable dt con las condiciones de cobro que estén pendientes de cobrarse comprendidas en el rango de fechas recibido.

EsquemaCalendarioCobros. Este es el dataset encargado de manejar la información del calendario dentro de la aplicación.

Figura 115. Diagrama de la clase EsquemaCalendarioCobros

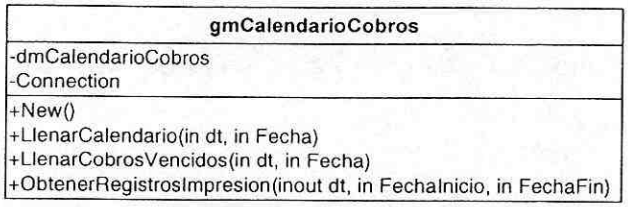



rptCalendarioCobros. Esta clase maneja la información para la impresión de los cobros pendientes a realizarse.

Figura 116. Diagrama de la clase rptCalendarioCobros

gmCalendarioCobros. Esta clase administra la información del calendario de cobros.

Figura 117. Diagrama de la clase gmCalendarioCobros




El procedimiento New es el constructor de la clase e inicializa los atributos.

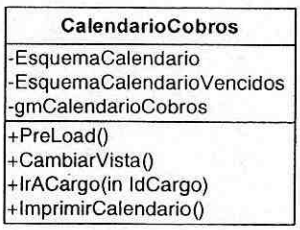
El procedimiento LlenarCalendario alimenta la datatable dt obteniendo los cobros a realizarse utilizando el dmCalendarioCobros para obtener la información.

El procedimiento LlenarCobrosVencidos alimenta la datatable dt con los cobros cuya fecha de vencimiento expiró y no han sido cobrados empleando al dmCalendarioCobros para obtener la información.

El procedimiento ObtenerRegistrosImpresion utiliza al dmCalendarioCobros para obtener la información para la impresión de las condiciones de cobro pendientes de cobrar comprendidos en el rango de fecha recibido.

CalendarioCobros. Esta clase maneja los cobros pendientes de realizar mostrando la información mensual, semanal y diariamente.

Figura 118. Diagrama de la clase CalendarioCobros



El procedimiento PreLoad inicializa los atributos de la clase y obtiene la información de las condiciones pendientes de cobrar a partir de la fecha actual del sistema.

El procedimiento CambiarVista se encarga de manejar el cambio entre las vistas mensual, semanal y diaria.

El procedimiento IrACargo muestra en pantalla la información completa del documento de cargo recibido.

El procedimiento ImprimirCalendario realiza el proceso de imprimir las condiciones pendientes de cobrar entre un rango de fechas.

ImpresionCalendario. Esta clase maneja el proceso de imprimir las condiciones de cobros pendientes de cobrar dentro de un rango de fechas.

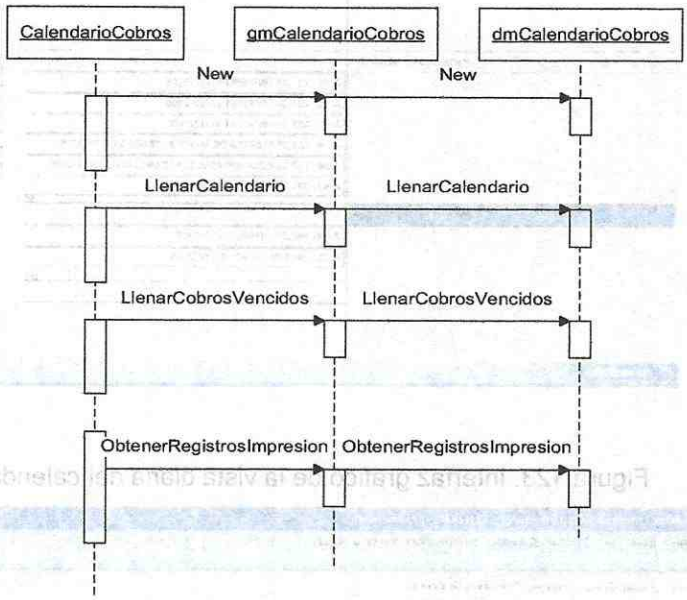
Figura 119. Diagrama de la clase ImpresionCalendario


La función Imprimir pide al usuario un rango de fechas y con eso muestra las condiciones de cobros pendientes de cobrar comprendidas dentro de ese rango.

La función ValidaFecha valida que las fechas de inicio y de fin dadas por el usuario sean correcta. Es decir, que se hayan ingresado las dos fechas y que la fecha de inicio no sea mayor a la fecha de fin.

## 3) Diagrama de secuencias

Figura 120. Diagrama de secuencia del calendario de cobros




## 4) Interfaz gráfico de usuario

Figura 121. Interfaz gráfico de la vista mensual del calendario de cobros.

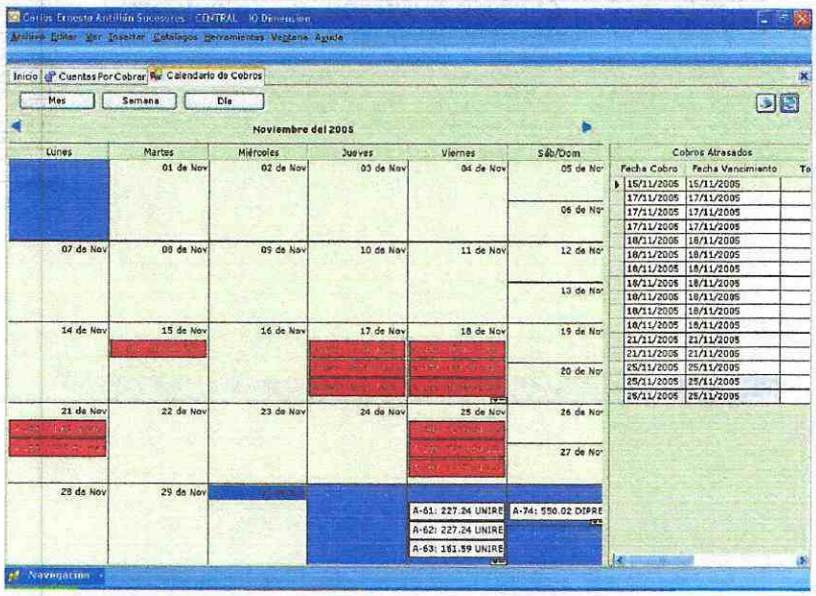


Figura 122. Interfaz gráfico de la vista semanal del calendario de cobros.

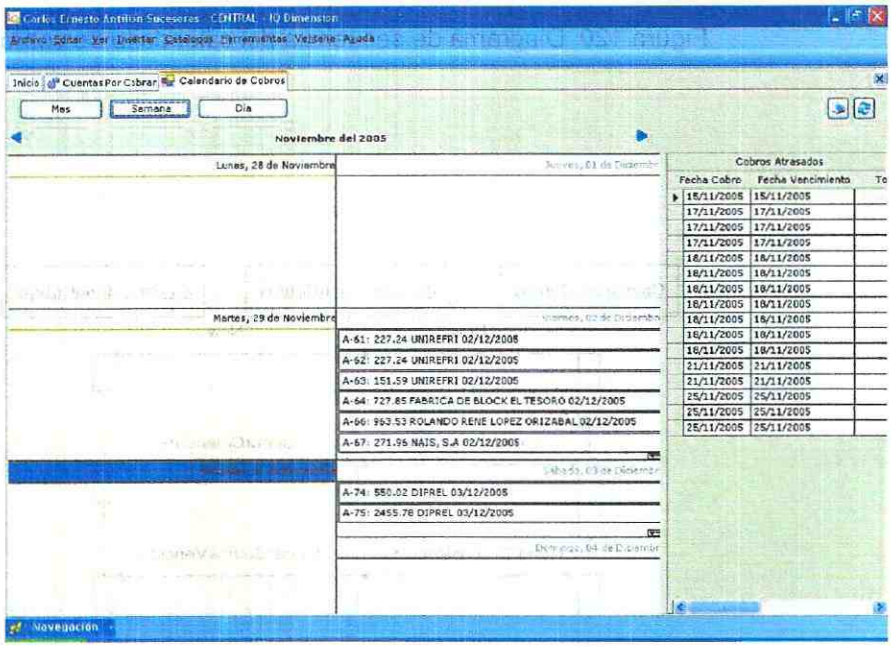



Figura 123. Interfaz gráfico de la vista diaria del calendario de cobros.

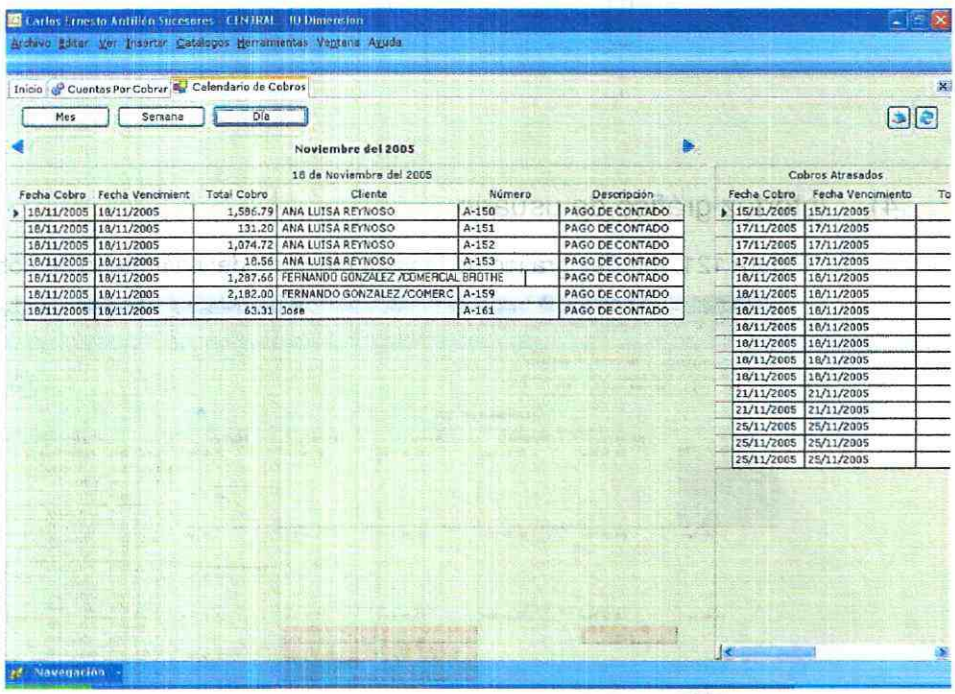
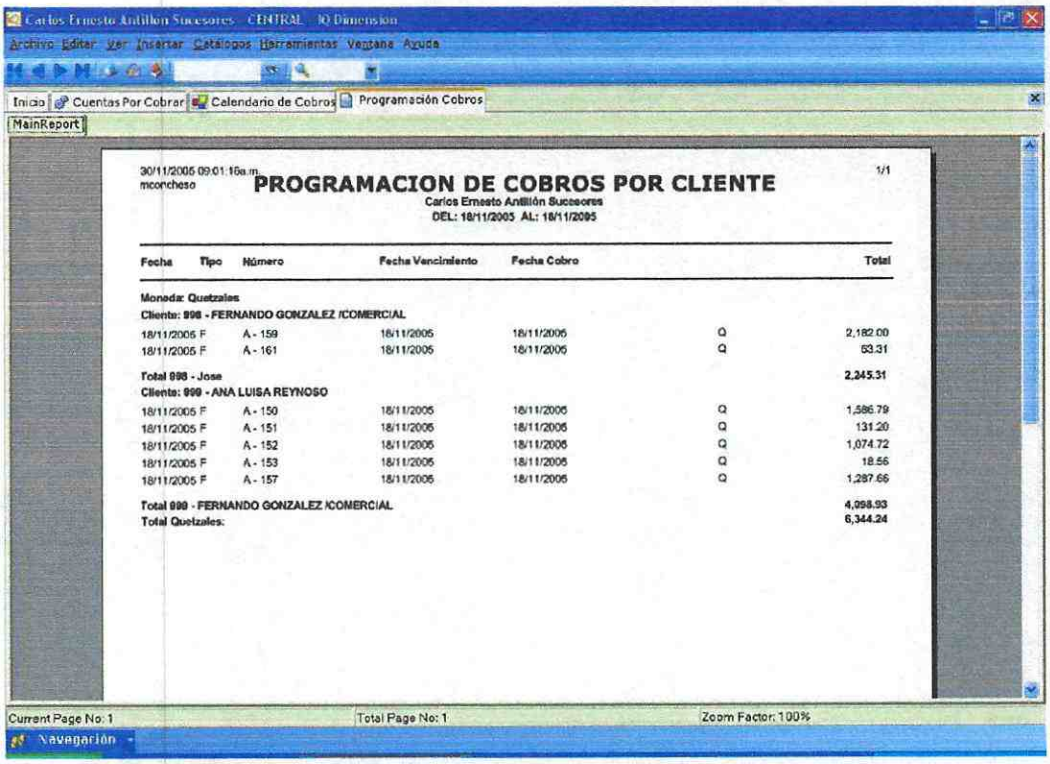


Figura 124. Interfaz gráfico para el rango de fechas de la impresión del calendario de cobros.

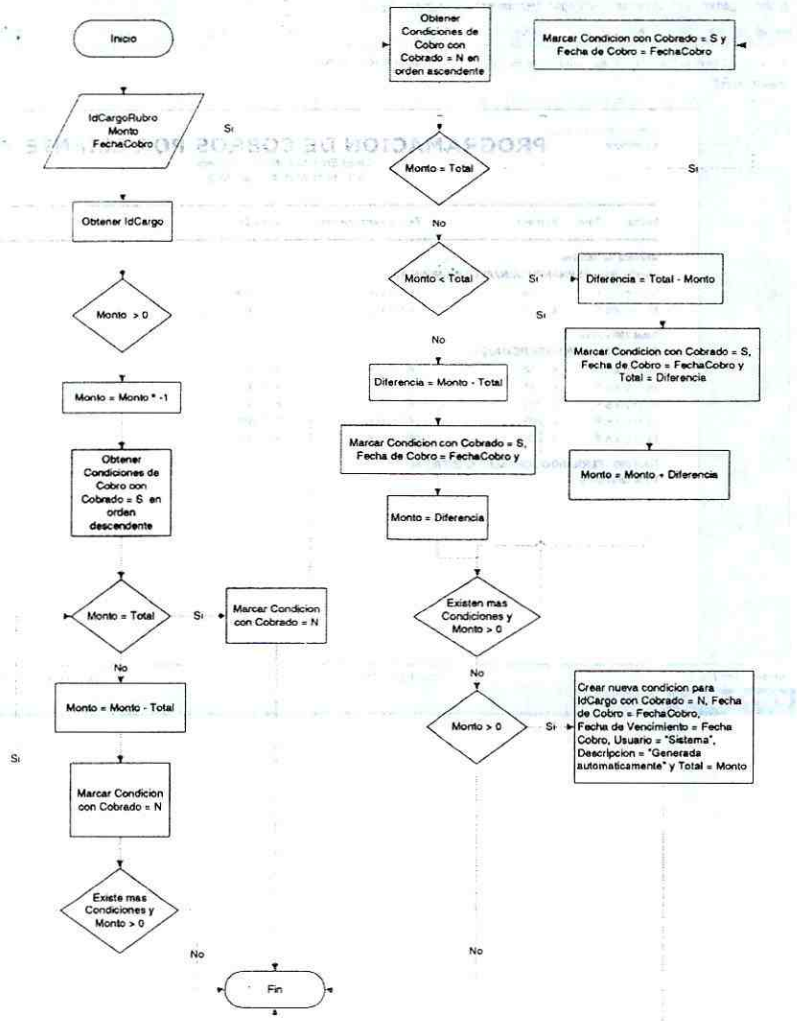

Figura 125. Interfaz gráfico de la impresión del calendario de cobros.



4. Operaciones en la base de datos. Para reflejar en los documentos de cargos cuando se realiza, modifica, anula o elimina un documento de abono se emplearon procedimientos y triggers en la base de datos. Esto con el fin de eliminar complejidad en el código fuente y lograr un mejor desempeño en el sistema.

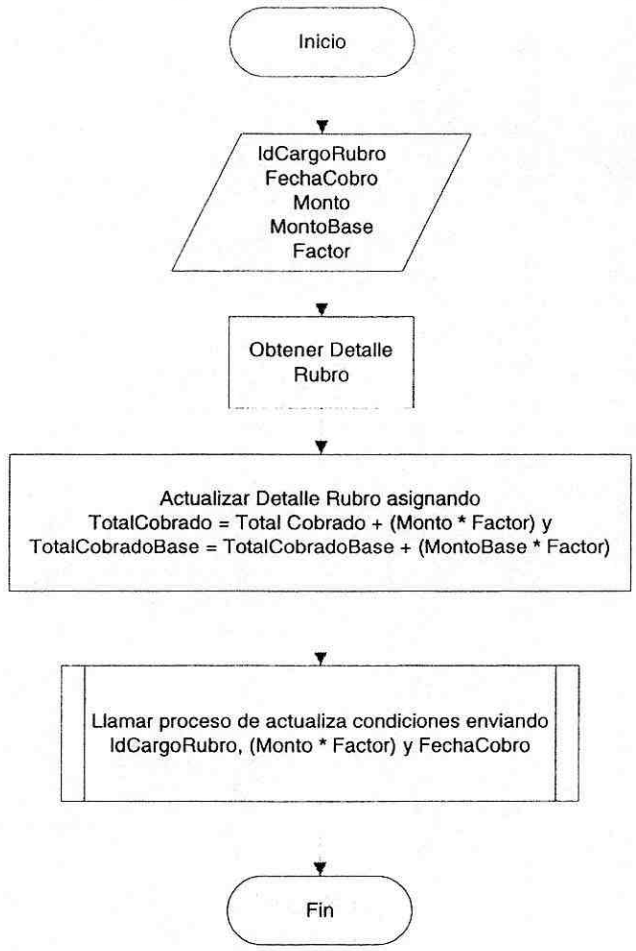
a. Procedimiento de actualización de condiciones de cobro. Este procedimiento se encarga de actualizar las condiciones de cobros para un documento de cargo determinando.

Figura 126. Diagrama de flujo de la actualización de condiciones de cobro



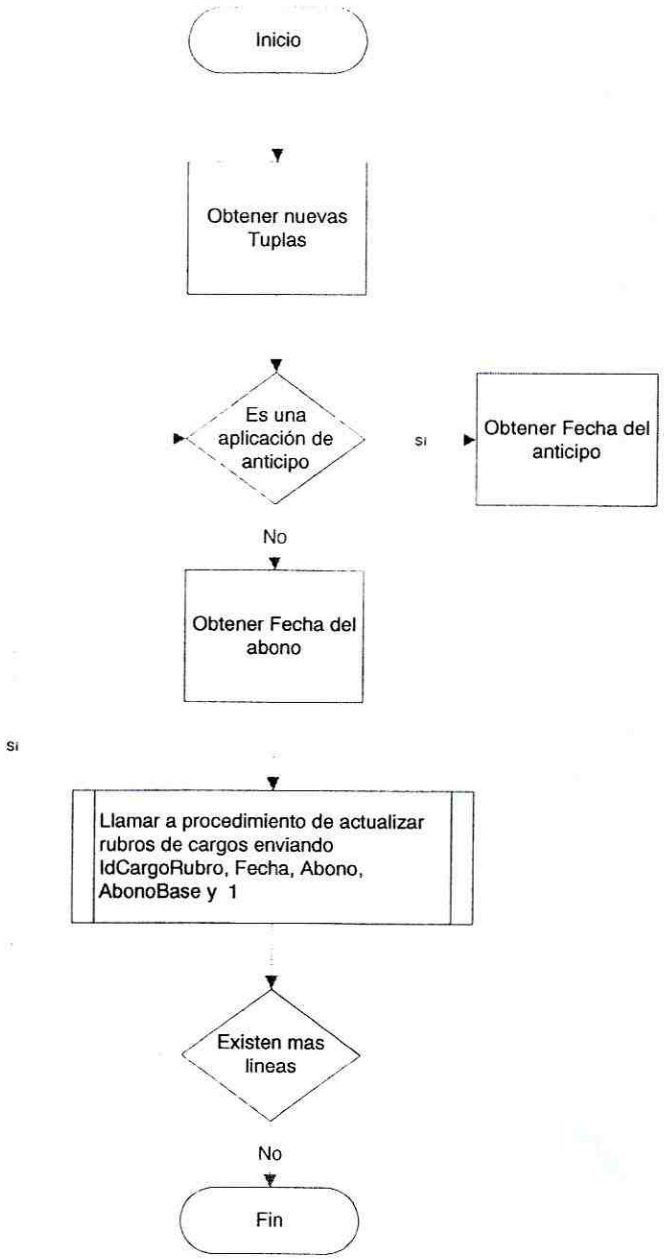
b. Procedimiento de actualización de rubros de un cargo. Este procedimiento actualiza un determinado detalle de rubro de un documento de cargo. A su vez, manda a llamar al procedimiento de actualización de condiciones de cobro.

Figura 127. Diagrama de flujo de la actualización del detalle de rubros de un documento de cargo



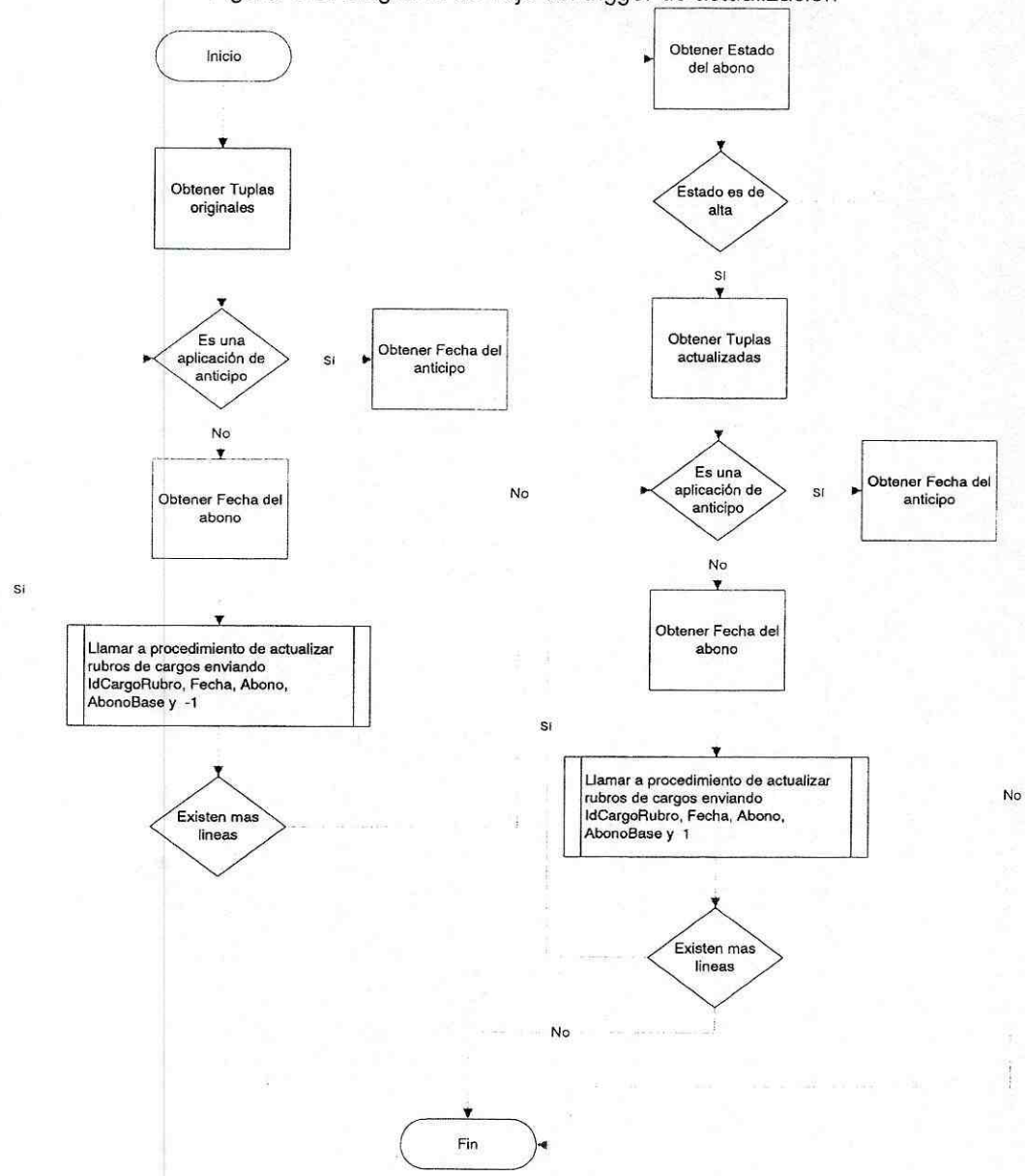

c. Trigger para inserción de rubros en documentos de abono. Este trigger se ejecuta al momento que se realiza la operación de INSERT en la tabla de in_cc_abono_cargo_rubro. Manda a llamar al procedimiento de actualización de rubros de un cargo para asignar el monto del abono.

Figura 128. Diagrama de Flujo del Trigger de inserción



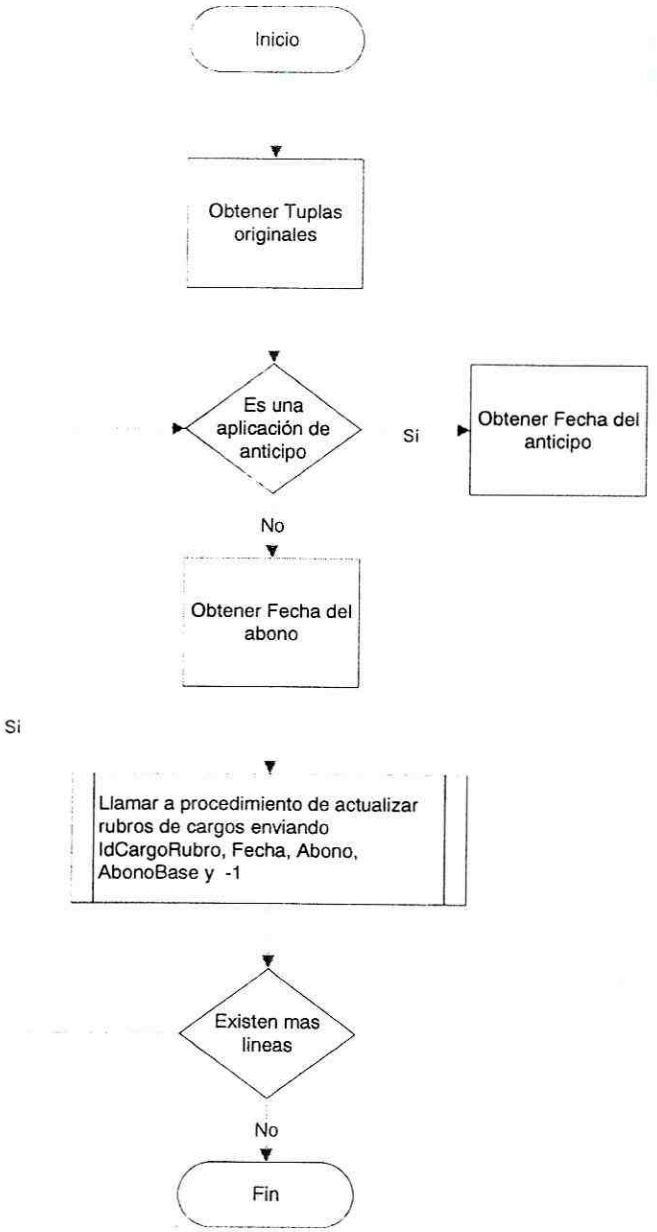
d. Trigger para actualización de rubros en documentos de abonos. Este trigger se ejecuta al momento que se realiza la operación de UPDATE en la tabla de in_cc_abono_cargo_rubro. Primero manda a llamar al procedimiento de actualización de rubros de un cargo para eliminar el monto original y luego lo vuelve llamar para asignar el nuevo monto.

Figura 129. Diagrama de flujo del trigger de actualización



e. Trigger para eliminación de rubros en documentos de abonos. Este trigger se ejecuta al momento que se realiza la operación de DELETE en la tabla de in_cc_abono_cargo_rubro. Manda a llamar al procedimiento de actualización de rubros de un cargo para eliminar el monto del abono.

Figura 130. Diagrama de flujo del trigger de eliminación



# V. DISCUSIÓN

El módulo de cuentas por cobrar es una herramienta dentro de IQ Profit que permite ingresar y administrar la información de las cuentas por cobrar de una empresa de manera sencilla y rápida. Sin embargo, el módulo no maneja las cuentas por cobrar de la empresa, esta tarea todavía se encuentra en las manos de los usuarios del sistema quienes son los que ingresan la información y toman las decisiones. El éxito de las cuentas por cobrar sigue dependiendo de la empresa, en la información que definen (las políticas de crédito, clientes, etc.).

El desarrollo de un ERP o un componente es interesante ya que se maneja mucha información dinámica que afecta el funcionamiento de los procesos, por ejemplo, en este caso, las configuraciones. Un ERP debe contemplar toda la funcionalidad de una empresa, sin embargo no todas las empresas laboran o se rigen de la misma manera. Es por esto que existe la posibilidad que no se definan todos los comportamientos e información dentro de un ERP para que este funcione correctamente en cualquier ambiente.

Para las pruebas del módulo de cuentas por cobrar, se creó un ambiente de pruebas similar a como sería en un ambiente real. Con esto se prueba todo el comportamiento del módulo y también se prueba el manejo de errores del sistema. Sin embargo esto no es suficiente, ya que el ambiente está muy controlado, los que realizan las pruebas son los desarrolladores que conocen la estructura de la base de datos y tienen un control mejor sobre el sistema. Un aspecto muy importante que ha ayudado al buen desarrollo del módulo de cuentas por cobrar y de IQ Profit a nivel general es que actualmente está siendo utilizado por empresas guatemaltecas, con esto se posee ya un ambiente real que permite estabilizar mejor el sistema.

Dentro de los problemas más grandes que se han encontrado en el sistema a nivel global es en el manejo de errores. Durante el análisis y desarrollo se definen ciertos manejos de errores de acuerdo a conductas del usuario. Sin embargo se ha encontrado que dentro de los usuarios finales del sistema existen conductas muy distintas, es especial en la manera en que ellos manejan la información. Si bien para un usuario el sistema funciona sin problemas, para otro usuario puede estar generando muchos errores debido a la manera como los maneja. Inclusive se han originado errores que no fueron contemplados en el análisis y desarrollo ocasionando que el sistema finalice abruptamente. Para remediar un poco esto, se empleó un manejo para errores desconocidos en donde se lleva una bitácora, no evitando el error pero, al menos, evitando que el sistema falle totalmente y brindando una herramienta a los desarrolladores para encontrar una solución más pronta al problema.

Otro problema que se ha encontrado en el ambiente real es el redondeo en las cantidades decimales. Mucha de la información dentro del módulo de cuentas por cobrar está a manera de detalles y también maneja porcentajes y distribución, sin mencionar tasas de conversión. Esto causa cálculos matemáticos con muchos decimales, por lo que si no se manejan bien estos decimales puede ocasionar descuadres dentro de la información. Es por esto que hay que mantener un control constante en el redondeo al momento de realizar cálculos matemáticos.

Otro aspecto muy importante en el desarrollo de un ERP es el manejo de multiusuarios. Un ERP se espera que será utilizado por más de un usuario al mismo tiempo, por lo que se ha tenido mucho control en lo que son los procesos que interactúan con la base de datos, cuidando que se realicen de manera transaccional y que no tomen demasiado tiempo para así no causar problemas a los demás usuarios. En el caso de las cuentas por cobrar, se ha tenido cuidado con el empleo de los triggers y procedimientos, ya que si bien eliminan complejidad en el código fuente, aumentan la complejidad en la base de datos y

las operaciones dentro de ella tardan más tiempo en realizarse. Por lo que se trató no crear muchos triggers y procedimientos en la base de datos.

Realizar un proyecto de esta naturaleza en una empresa real, para un mercado real, ha permitido aprender mucho no sólo de la parte técnica y metodológica de un ERP, sino también de su parte social, viendo la reacción de los usuarios ante el sistema y cómo funciona en un ambiente real.

# VI. CONCLUSIONES

El módulo de cuentas por cobrar de IQ Profit puede ser utilizada por cualquier tipo de empresa, aunque su contexto de ERP está orientado a empresas principalmente de distribución, esto debido a que el módulo puede ser configurable cubriendo así las necesidades que la empresa tenga. Y no sólo a nivel de la empresa, sino también a nivel de agencia, ya que por factores como lo es la localización geográfica, las agencias entre si pueden requerir operar de manera diferente las cuentas por cobrar.

Un aspecto que cabe destacar es que el módulo de cuentas por cobrar brinda una herramienta para manejar la información de las cuentas por cobrar, no garantiza el éxito de las mismas en la empresa por sí solo. El éxito que tengan las cuentas por cobrar para la empresa sigue dependiendo de la información que define la empresa así como del manejo y fiabilidad de la misma.

El uso de triggers y procedimientos en la base de datos ha disminuido la complejidad en el código del sistema pero ha aumentado la complejidad de la base datos, esto debido a que se extiende la atomicidad de una operación y de ocurrir un error la depuración del mismo ya no se limita solamente a la aplicación sino también a la base de datos.

Una ventaja muy grande en la estabilización del módulo de cuentas por cobrar fue el hecho de emplear una arquitectura de capas en el desarrollo. Esto permitió aislar componentes del módulo haciendo que el cambio en alguno de ellos sea rápido ya que solo afecta a ese componente y no a los demás.

Haber tenido un ambiente local de pruebas y un ambiente real de implementación fue una gran ayuda para estabilizar y agilizar el desarrollo del módulo de cuentas por cobrar.

# VII. RECOMENDACIONES

Al realizar aplicaciones de este tipo, en donde no es para una empresa específica, sino para cualquiera, es recomendable que desde el análisis y diseño se deje la posibilidad poder realizar cambios en la aplicación. Como por ejemplo, agregar nuevos datos a los procesos y no realizar demasiado estático la realización de los procesos. Además de esto se tiene que tener en mente que muchas empresas van a querer realizar cosas específicas, realizar procesos "a su manera", con esto se recomienda permitir que a la aplicación se puedan agregar procesos específicos pero sin afectar el núcleo de la aplicación, esto porque no todas las empresas van a requerir realizar este proceso específico y una de las cosas que no se desea es tener que manejar diferentes versiones de la aplicación.

Como se mencionó anteriormente, durante la estabilización del módulo de cuentas por cobrar se encontraron muchos problemas debido al redondeo de cantidades. Esto no sólo puede suceder en las cuentas por cobrar ni solo en IQ Profit, el redondeo de cantidades afecta cualquier aplicación que maneje cantidades decimales y realice cálculos matemáticos. Por eso es bueno desde el análisis y diseño definir bien el manejo de redondeo de decimales, para que este sea uniforme en toda la aplicación y así no surjan problemas por redondeo.

Aplicaciones de este tipo, en donde se maneja una gran cantidad de información, estamos hablando de miles de registros y sólo en una tabla de la base de datos, y realiza registros a partir de otros, requieren de una herramienta capaz de soportar esto. Se recomienda utilizar plataformas de desarrollo como Microsoft Visual Studio .NET o de la misma generación, los cuales ya proveen de muchas herramientas tanto para el manejo de información del interfaz de usuario, como la información de la base de datos. También se recomienda usar una base de datos como SQL Server 2000 u Oracle, ya que proveen de mucha funcionalidad para el manejo interno e integridad de los datos.

Como se ha mencionado anteriormente, poder contar con un ambiente real para la estabilización es una de las mejores herramientas que se puede tener. Ya que en un ambiente controlado es difícil lograr definir todos los casos de uso que puedan existir y mucho menos es difícil definir las diferentes interacciones del usuario con el sistema que pueda haber.

# VIII. BIBLIOGRAFÍA


1. Software Intelligence Corp. 2005. *Sistema Nervioso Digital IQ Dimension.* http://iqdimension.com/es/productos.cfm

2. Software Intelligence Corp. 2005. *IQ Profit.* http://iqdimension.com/es/productos/iqprofit.cfm

3. Infonet S.A. 2005. *Reseña Corporativa y Perfil de la Empresa.* http://www.infonet.com.gt/contenido/compania.cfm

4. Quintanar García, Liliana. *Procedimientos en las Cuentas por Cobrar y las Cuentas por Pagar.* http://www.gestiopolis.com/canales/financiera/articulos/36/cxc.htm

5. Marston, Tony. 2005. *A Development Infrastructure for PHP.* http://www.tonymarston.net/php-mysql/infrastructure.html

6. Loshin, David. 2003. *Business Intelligence: The Savvy Manager's Guide.* Morgan Kaufmann. 200 pags.

7. El Colegio de Sonora. *Los Negocios en la Era Digital.* http://lanic.utexas.edu/project/etext/colson/20/20_R4.pdf

8. Bill; Hemingway, Collins Gates. 1999. *Business the Speed of Thought: Using a Digital Nervous System.* Time Warner on Demand. 420 pags.

9. Microsoft Corporation. 2006. *.NET Framework Developer's Guide: Overview of the .NET Framework.* http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpovrintroductiontonetframeworksdk.asp

10. Microsoft Corporation. 2006. *Product Overview for Visual Studio .NET 2003 Enterprise Architect.* http://msdn.microsoft.com/vstudio/previous/2003/overview/eaoverview.aspx

11. Microsoft Corporation. 2006. *Product Overview for Visual Studio .NET 2003.* http://msdn.microsoft.com/vstudio/previous/2003/overview/default.aspx

12. Microsoft Corporation. 2001-2002. *2389B: Programming with Microsoft ADO .NET.* 356 pags.

13. Booch, Grady; Rumbaugh, Jim y Jacobson, Ivar. *UML, el Lenguaje Unificado de Modelado.* http://elvex.ugr.es/decsai/java/pdf/3E-UML.pdf

14. Microsoft Corporation. 2002. *SQL Server Books Online: Microsoft SQL Server 2000.* http://www.microsoft.com/downloads/details.aspx?familyid=a6f79cb1-a420-445f-8a4b-bd77a7da194b&displaylang=en

15. Pressman, Roger. 1993. *Ingeniería del Software, Un enfoque Práctico*. Editorial McGraw-Hill, México. 824 págs.

16. Jensen, Randall y Tonéis Charles. 1979. *Software Engineering*. Editorial Prentice-Hall, Englewood Cliffs, New Jersey. 580 págs.

17. Davis, Stan y Meyer, Christopher. 2000. *Blur: The Speed of Change in the Connected Economy*. Editorial Warner Bk. ISBN: 0446675334

18. Turban, Efraim & Iroson, Jack E. 1998. *Decision Suport Systems and Intelligent Systems*. Editorial Prentice Hall. Quinta Edición.

19. Wiederhold, Gio. 1985. *Diseño de bases de datos*. 2ª Edición. Editorial McGraw-Hill. México.

20. Sommerville, Ian. 2002. *Ingeniería de Software*. Sexta Edición. Addisson-Wesley Iberoamericana. Argentina.

# IX. APÉNDICE

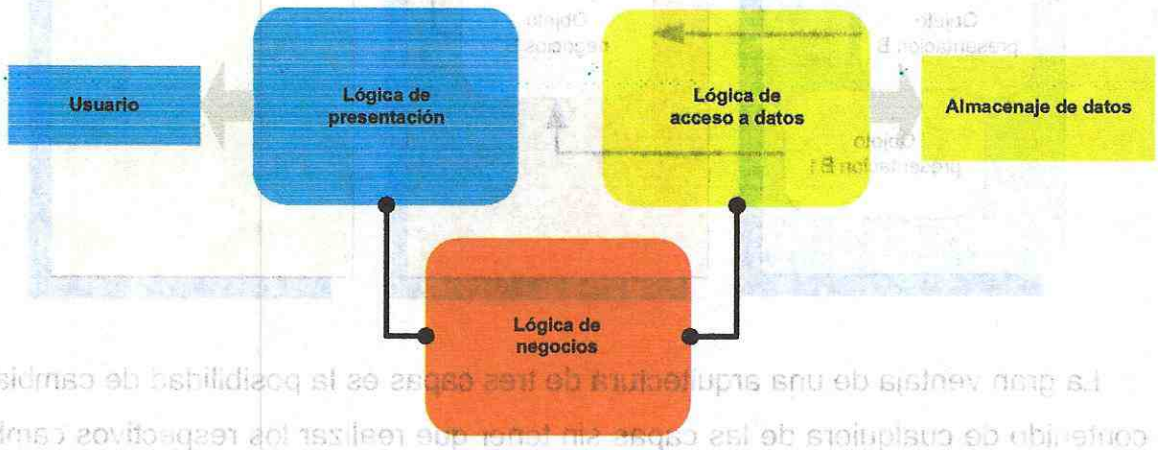
## A. Arquitectura de tres capas

La mayoría de software desarrollado a partir de lenguajes de 3G puede ser subdividido en las siguientes áreas:

- Lógica de presentación. Incluye interfaz con el usuario, despliegue de datos al usuario, aceptar ingresos de datos por parte del usuario, etc.
- Lógica de negocios. Son las reglas de negocios, maneja la validación de datos y conductas específicas para realizar tareas.
- Lógica de acceso a datos. Es la comunicación hacia donde se encuentran almacenados los datos (base de datos, archivos XML, archivos de texto, etc.). Es el encargado del manejo directo de los datos.

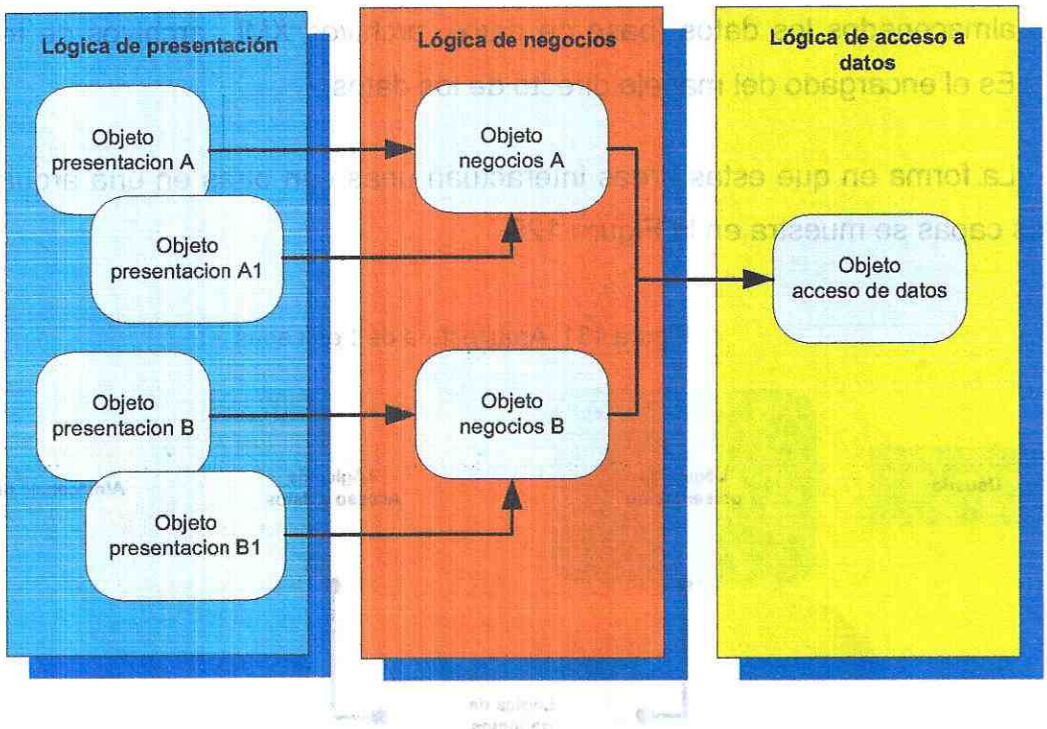
La forma en que estas áreas interactúan unas con otras en una arquitectura de tres capas se muestra en la Figura 129.

Figura 131. Arquitectura de tres capas



Cuando se implementa esta arquitectura de tres capas los resultados son evidentes, ya que muchas tareas y lógica de desarrollo se pueden compartir en lugar de ser duplicados. Múltiples componentes en la lógica de presentación pueden compartir los mismos componentes de lógica de negocios, y muchos componentes en la lógica de negocios pueden compartir los mismos componentes en la lógica de acceso a datos. También existe la posibilidad de que un solo componente en la lógica de presentación tenga acceso a múltiples componentes en la lógica de negocios y un componente de la lógica de negocios tenga acceso a otros componentes de la lógica de negocios. Un ejemplo de esto se muestra en la figura 130.

Figura 132. Funcionamiento de la arquitectura de tres capas




La gran ventaja de una arquitectura de tres capas es la posibilidad de cambiar el contenido de cualquiera de las capas sin tener que realizar los respectivos cambios en las demás capas.

Por ejemplo:

- Si se quiere cambiar de una base de datos a otra, simplemente hay que cambiar los componentes en la lógica de acceso a datos.

- Si se desea cambiar el interfaz del usuario o el ambiente del interfaz, por ejemplo, de aplicación a WEB, simplemente hay que cambiar los componentes de la lógica de presentación.

## B. .NET Framework

Es un componente integral del paradigma Microsoft .NET, el cual soporta la construcción y ejecución de las aplicaciones y servicios WEB XML de siguiente generación, es decir la plataforma sobre la cual se ejecuta el código intermedio producido por los múltiples lenguajes compatibles con Microsoft .NET. El .NET Framework fue desarrollado para los siguientes objetivos:

- Proveer un ambiente consistente para la programación orientada a objetos, ya sea para código almacenado y ejecutado localmente, distribuida a través del Internet o remotamente.

- Proveer de un ambiente de ejecución de código que minimice los conflictos por versiones y lanzamientos de software.

- Proveer de un ambiente de ejecución de código que promueva la ejecución segura de código incluyendo código creado por terceras personas.

- Hacer que el desarrollador tenga una experiencia consistente a través de varios tipos de aplicaciones, como las aplicaciones basadas en Windows y las aplicaciones basadas en WEB.

El .NET Framework está conformado principalmente por dos componentes:

1. Ejecución de Lenguaje Común (Common Language Runtime). El CLR, según sus siglas en inglés, es el encargado del manejo de memoria, ejecución de hilos (threads), ejecución de código, verificación de código seguro, compilación y otros servicios del sistema.

A cada componente y/o código que administra el CLR se le asigna un grado de confianza, el cual ayuda a restringir lo que puede o no puede hacer dicho componente y/o código. Por ejemplo, algunos podrán realizar manejo de archivos, operaciones de acceso a registros, etc., mientras otros no.

El CLR asegura una robustez en el código implementando una infraestructura de validación estricta de tipos y código llamada Common Type System (CTS).

El CLR está diseñado para aumentar desempeño. Aunque el CLR provee de servicios estándares de ejecución, los componentes y/o códigos nunca son interpretados. Una novedad llamada compilación just-in-time (JIT) permite a todo componente y/o código ser ejecutado en el lenguaje nativo del sistema en donde se está ejecutando.

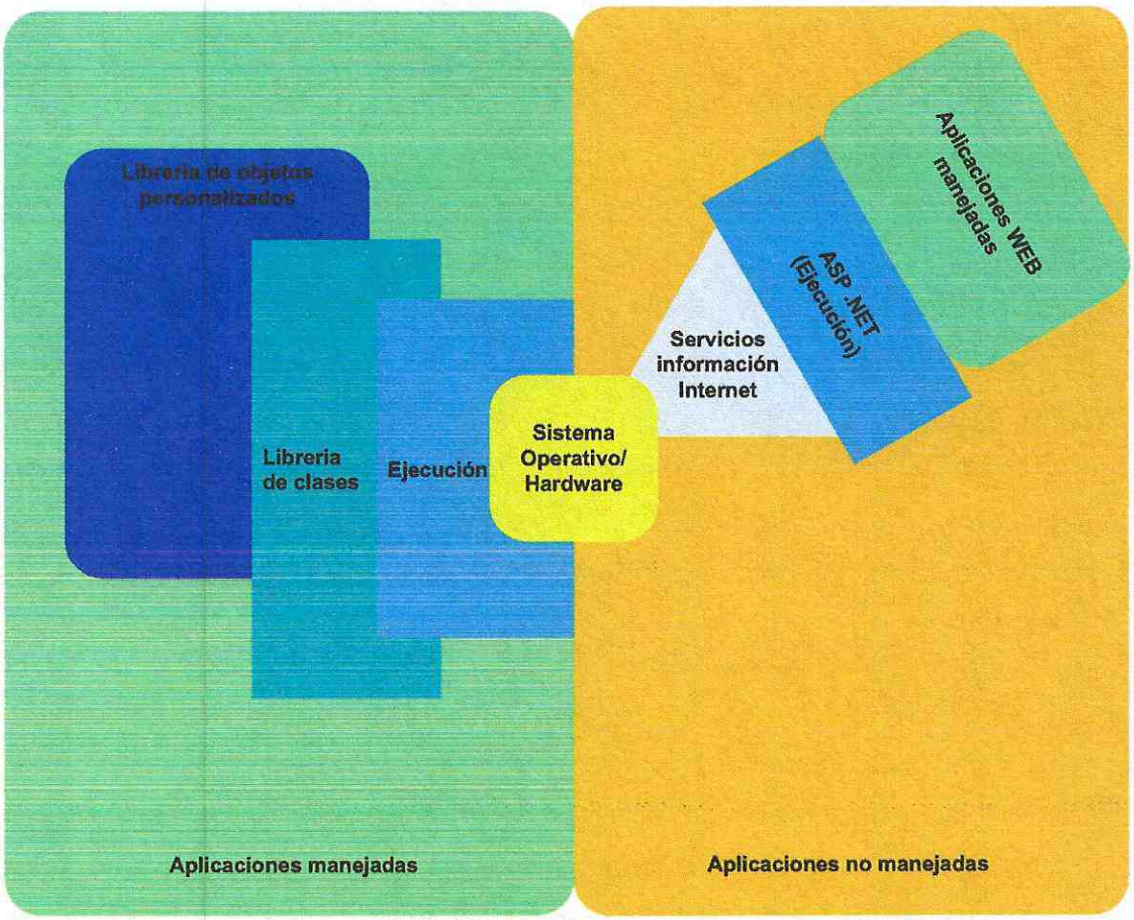
2. Librería de clases. Es una colección de tipos que se integran fuertemente al CLR. La librería de clases está orientada a objetos, permitiendo que el código del usuario derive funcionalidades para los tipos.

Los tipos en la librería de clases permiten lograr una variedad de tareas de programación comunes, como manejo de cadena de caracteres, colecciones de datos, conectividad a bases de datos, acceso a archivos, etc. Además brinda soporte para una variedad de ambientes especializados de desarrollo, como lo son:

- Aplicaciones de Consola.
- Aplicaciones de Microsoft Windows.
- Aplicaciones de ASP .NET.
- Servicios WEB XML.
- Servicios de Microsoft Windows.

La figura 131 muestra las relaciones entre el CLR y la librería de clases con las aplicaciones de usuario y el sistema en general.

Figura 133. Relación entre CLR y librería de clases con las aplicaciones de usuario y el sistema en general




## C. Visual Studio .NET 2003 Enterprise Architect (VSEA).

Visual Studio .NET 2003 es un conjunto completo de herramientas de desarrollo para la construcción de aplicaciones WEB ASP.NET, servicios XML WEB, aplicaciones de escritorio y aplicaciones móviles. Visual Studio .NET comprende los siguientes lenguajes de programación, Visual Basic, Visual C++, Visual C# y Visual J#, todos ellos usan el mismo ambiente integrado de desarrollo lo que permite que

se pueda compartir herramientas y funciones en la creación de soluciones con varios lenguajes. VSEA adicionalmente incluye herramientas para diseñar, especificar y comunicar arquitecturas de aplicaciones, desarrollar mejores prácticas y para la funcionalidad de las aplicaciones.

Con VSEA es posible:

**1.** Modelar visualmente aplicaciones, base de datos y procesos de negocios. Se puede emplear el conjunto completo de capacidad para modelar de VSEA para crear y comunicar arquitecturas de aplicaciones, requerimientos de negocios, diseño de base de datos y procesos de negocios. Arquitecturas que usan Visual C++, Visual Basic o Visual C# pueden emplear modelos hechos con el lenguaje unificado de modelado (UML) para especificar funcionalidad y arquitectura de aplicaciones, reduciendo el tiempo de desarrollo generando directamente clases, funciones y métodos, y documentar código existente mediante proyectos de reingeniería inversa. VSEA permite a los desarrolladores crear diseños de arquitecturas y modelos que pueden compartir con el resto de su equipo de desarrollo.

**2.** Construir guías para mejores prácticas. Se pueden crear proyectos Templates Empresariales y Templates de descripción de lenguaje para ayudar a los desarrolladores cumplir con los retos de la proliferación rápida de innovaciones y tecnologías, y aumentar la colaboración en el equipo de desarrollo. Los arquitectos pueden usar los proyectos Empresariales para crear puntos de partida en la creación de aplicaciones, especificando una estructura inicial de la aplicación, así también diseños de documentación y modelos.

Usando los proyectos de descripción de lenguajes se puede especificar las limitantes para el uso de los componentes a través de la aplicación, por ejemplo, configurando propiedades y rangos para que los desarrolladores puedan fácilmente escoger implementaciones correctas.

**3.** Construir aplicaciones distribuidas en una plataforma escalable y dependiente. El .NET Framework está diseñado para simplificar el desarrollo de aplicaciones en un ambiente altamente distribuido. Esto lo logra mediante la integración de:

- Estándares públicos del Internet, como el XML, SOAP, UDDI y WML.
- Mejoras en los servicios WEB, como la seguridad basada en mensajes.
- Una arquitectura altamente escalable.
- Desarrollo de aplicaciones en el lenguaje preferido.
- Transacciones automáticas fáciles de usar, manejo de memoria automática e implementación fácil.

Visual Studio .NET 2003 provee una arquitectura abierta y amplia que permite a terceros crear lenguajes, herramientas y componentes que pueden integrase en el ambiente, otorgando a los desarrolladores una amplia gama de opciones para cumplir con sus requerimientos de desarrollo.

## D. ADO .NET

Es un conjunto de clases para trabajar datos. El desarrollo de aplicaciones ha evolucionado, las nuevas aplicaciones ya no son muy integradas debido a que se basan en aplicaciones WEB. Muchas aplicaciones emplean XML para codificar datos para ser transmitidos a través de conexiones de red.

Las ventajas del ADO .NET son:

- Interoperabilidad. ADO .NET emplea XML como el formato para transmitir datos desde la fuente de datos a una copia en memoria local.
- Mantenimiento. Cuando un gran número de usuarios trabajan dentro de una aplicación, los recursos pueden verse limitados. Mediante el uso de aplicaciones de n-Capas, el desarrollador puede distribuir la lógica de la aplicación a través de capas adicionales. La arquitectura del ADO .NET emplea caches en memoria

local para hacer copias de datos, haciendo fácil agregar capas para compartir información.

- Programación. El modelo de programación del ADO .NET usa tipos de datos estrictos. Esto permite que el código sea más conciso y fácil de escribir, ya que Visual Studio .NET provee herramientas para completar código.
- Desempeño. ADO .NET ayuda a evitar conversiones costosas de datos debido a que usa tipos de datos estrictos.
- Escalabilidad. El modelo de programación del ADO .NET motiva a los desarrolladores a conservar recursos del sistema para aplicaciones que se ejecutan en un ambiente WEB. Debido a que los datos se encuentran en caches de memoria local, no hay necesidad de mantener seguros en la base de datos por periodos muy largos.

**1.** Componentes del ADO .NET. El .NET Framework divide la funcionalidad en nombres lógicos, y ADO .NET no es la excepción. ADO .NET es implementado principalmente por la jerarquía del nombre lógico System.Data, que reside físicamente en el archivo System.Data.dll.

Los componentes de System.Data son:

a. System.Data. Núcleo del ADO .NET. Incluye clases que conforman partes de la arquitectura desconectada del ADO .NET.

b. System.Data.Common. Emplea clases e interfaces que son heredadas e implementadas por los .NET data providers.

c. System.Data.SqlClient. Conforma el .NET data provider de base de datos SQL.

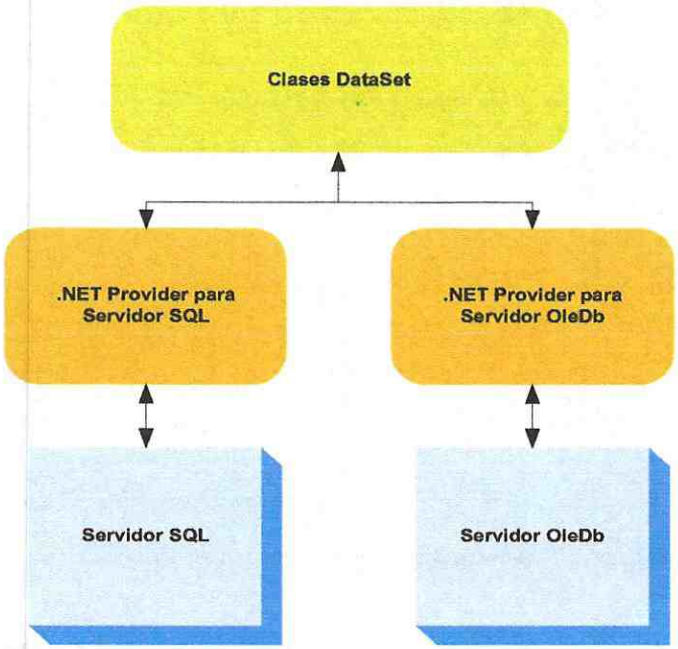
d. System.Data.OleDb. Conforma el .NET data provider de base de datos OleDb.

e. System.Data.SqlTypes. Son clases y estructuras para tipos de datos nativos de la base de datos SQL. Es una alternativa más segura y fácil que otros tipos de datos.

f. System.Xml. Clases, interfaces y enumeraciones que proveen soporte estándar para procesar XML.

**2.** Modelo de un objeto ADO .NET

Figura 134. Modelo de un Objeto ADO .NET




Como se muestra en la figura 132, un objeto ADO .NET está compuesto principalmente de dos partes:

a. Clases DataSet. Son clases que permiten almacenar y manejar datos en un ambiente desconectado. El DataSet es independiente de la fuente de datos, por lo que puede ser utilizado por cualquier aplicación, sin importan el origen de los datos de la aplicación.

b. Clases .NET Data Providers. Son clases que especifican la fuente de los datos. Por lo tanto, los .NET Data Providers deben ser construidos especialmente para la fuente de datos y funcionan solamente para esa fuente de datos. Los .NET Data Providers permiten conectarse a la fuente de datos, obtener datos de él y realizar actualizaciones de datos.

Las clases principales de un .NET Data Provider son:

a. XxxDataAdapter. Utiliza las clases de Connection, Command y DataReader para implícitamente popular un DataSet y actualizar la fuente de datos con los cambios que se hayan hecho en el DataSet.

b. XxxConnection. Establece una conexión con una fuente de datos específica.

c. XxxCommand. Ejecuta un comando de la fuente de datos.

d. XxxDataReader. En una clases para sólo lectura de datos. Estos datos solo pueden ser leídos en un solo sentido, hacia delante.

Un objeto ADO .NET incluye las siguientes clases .NET Data Providers:

- *.NET Data Provider para base de datos SQL.*
- .NET Data Provider para base de datos OleDb.

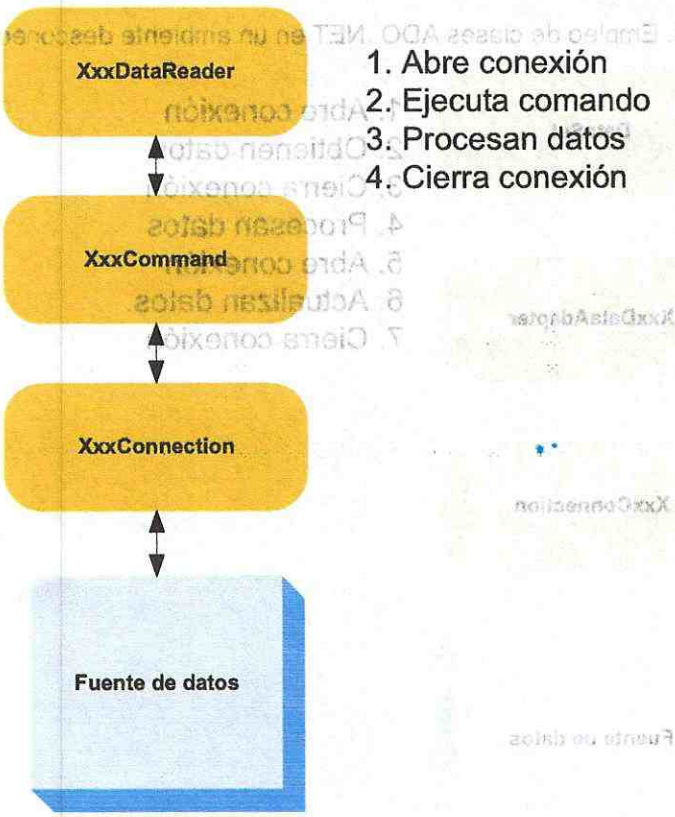
**3.** Empleando clases ADO .NET en un ambiente conectado. En un ambiente conectado, toda operación que se realice con la fuente de datos se realiza dentro de una conexión. Es decir, la conexión con la fuente de datos se mantiene activa

durante todo el proceso que se realice. Con esto los recursos del sistema son agarrados por la fuente de datos hasta que la conexión haya terminado.

El procedimiento para realizar una tarea en un ambiente conectado es el siguiente:

- Se abre la conexión con la fuente de datos.
- Se ejecuta el comando.
- Se procesan los datos obtenidos por el comando.
- Se cierra la conexión.

Figura 135. Empleo de clases ADO .NET en un ambiente conectado



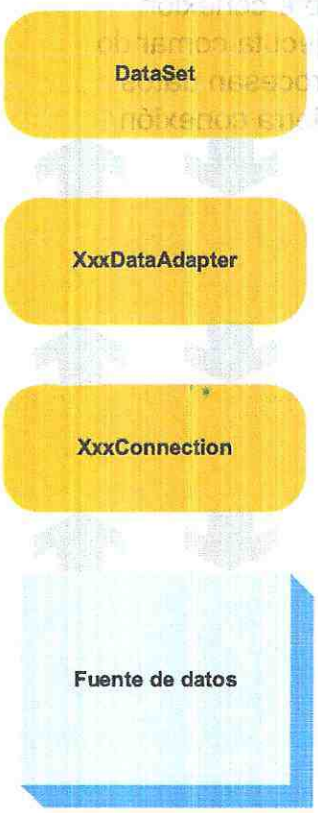

**4.** Empleo de clases ADO .NET en un ambiente desconectado. En un ambiente desconectado la conexión con la fuente de datos únicamente se establece al momento de obtener datos y cuando se actualicen los datos, el procesamiento de

los datos se realiza sin una conexión. Con esto los recursos no son agarrados por la fuente de datos mientras los datos son procesados

El procedimiento en un ambiente desconectado para realizar una tarea es:

- Se abre la conexión.
- Se obtiene la información necesaria.
- Se cierra la conexión.
- Se procesan los datos obtenidos.
- Se abre la conexión.
- Se actualizan los datos.
- Se cierra la conexión.

Figura 136. Empleo de clases ADO .NET en un ambiente desconectado



## E. Lenguaje para Consultas Estructuradas 2000 (SQL 2000)

El Lenguaje para Consultas Estructuradas (SQL) es un lenguaje utilizado para insertar, obtener, modificar y eliminar datos en una base de datos relacional. SQL también tiene funciones para definir y administrar objetos en una base de datos. SQL en el lenguaje mejor soportado por las bases de datos relacionales y es el sujeto para la publicación de estándares por la Organización Internacional de Estándares (ISO) y el Instituto Nacional Americano de Estándares (ANSI).

SQL 2000 emplea una versión del lenguaje SQL llamada Transact-SQ, el cual contiene comandos para administrar instancias de la base de datos SQL, crear y manejar todos los objetos en una instancia de la base de datos SQL.

SQL 2000 incluye muchas herramientas nuevas que lo hacen excelente plataforma de base de datos para procesamiento transaccionales en línea a gran escala, hacer almacenes de datos y aplicaciones e-commerce.

**1.** Desencadenadores (Triggers). Es una clase especial de procedimientos almacenados definidos para ejecutarse automáticamente cuando una declaración UPDATE, INSERT, o DELETE es ejecutada en una tabla o vista. Los Triggers son herramientas poderosas que se pueden usar para fortalecer reglas de negocios automáticamente cuando los datos son modificados.

Las tablas pueden tener múltiples Triggers. La declaración CREATE TRIGGER puede ser definida con el FOR UPDATE, FOR INSERT o FOR DELETE para asignar un Trigger a una acción específica de modificación de datos.

El formato general de un Trigger es:

CREATE TRIGGER <nombre_trigger>
ON <nombre_tabla>
ACCION (FOR INSERT, FOR UPDATE o FOR DELETE)
AS
EXEC <declaraciones>

Los Triggers contienen declaraciones Transact-SQL. Los Triggers regresan un conjunto de resultados generados por cualquier declaración SELECT realizada dentro del Trigger. Se puede utilizar la cláusula FOR para especificar cuando se va a ejecutar el Trigger. Puede haber dos opciones:

a. AFTER. El Trigger se ejecuta después de la declaración a la que esta asignada. Si la declaración falla por cualquier error, el Trigger no es ejecutado. Los Triggers AFTER no pueden emplearse en vistas, únicamente en tablas. Se pueden declarar múltiples Triggers AFTER para cada acción (INSERT, UPDATE Y DELETE). Si se tienen múltiples Triggers AFTER para una tabla, se puede usar la propiedad *sp_settriggerorder* para definir cuál Trigger se ejecutará primero y cuál de último, los de en medio se ejecutan en un orden no definido y no puede ser controlado por el usuario dicho orden.

Los Triggers AFTER son los predeterminados en la base de datos SQL 2000.

b. INSTEAD OF. El Trigger se ejecuta en lugar de la declaración a la que esta asignada. Los Triggers INSTEAD OF pueden emplearse tanto en tablas como vistas. Únicamente se puede asignar un Trigger INSTEAD OF para cada acción (INSERT, UPDATE y DELETE). Los Triggers INSTEAD OF pueden ser empleador para un mejor chequeo de integridad de los datos a manejar.

Entre las desventajas de emplear los Triggers en una base de datos están:

- Aumento en la complejidad de la Base de Datos. Esto causa que la configuración de la base de datos sea más compleja.

- Las reglas de negocios se esconden. Las reglas de negocios no se podrán ver al momento de analizar modelos de datos o analizando la aplicación.
- Los Triggers pueden deshabilitarse y no ejecutarse cuando deberían.
- Los Trigger pudieran afectar el desempeño del sistema.

## F. Lenguaje Unificado de Modelado (UML)

<<Es lenguaje UML es un estándar diseñado para visualizar, especificar, construir y documentar software orientado a objetos>> (Berzal, 2005:1)

El UML se utiliza para establecer un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos. El UML toma la estructura estática y el comportamiento de un sistema y lo modela como un conjunto de objetos.

El UML se emplea para:

- Incluir los conceptos necesarios para utilizar un proceso.

- Poder modelar cualquier tipo de sistema de una manera simple.

- Manejar todos los conceptos de un sistema y los mecanismos de la ingeniería de software de una manera expresiva.

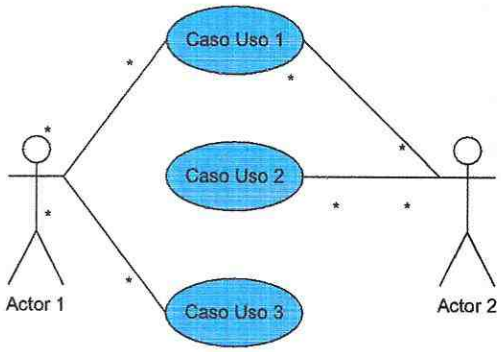
**1.** Diagramas UML. Un modelo es una representación de un sistema real de manera abstracta. El desarrollo de software debe ofrecer un conjunto de modelos que permitan que permitan expresar el sistema desde cualquier perspectiva.

Un diagrama es una representación gráfica de un conjunto de elementos de modelado, generalmente expresada como un grafo con arcos y vértices. El diagrama es una muestra de los elementos semánticos del modelo.

UML ofrece ocho tipos de diagramas para modelar distintos sistemas, estos son:

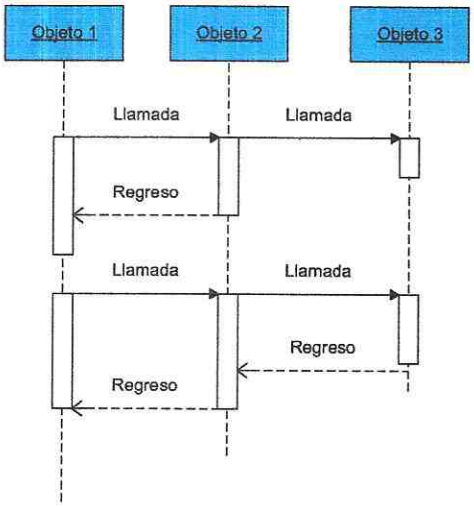
a. Diagramas de caso de uso. Se emplean para modelar procesos, los cuales representan la funcionalidad de un sistema. Este tipo de diagrama cuenta de elementos como lo son los actores y los casos de uso para lograr esto.

Figura 137. Diagrama de caso de uso



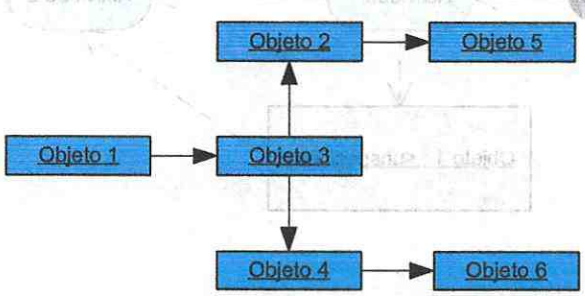

b. Diagramas de secuencias. Se emplean para modelar el paso de mensajes entre objetos dentro de un periodo de tiempo específico.

Figura 138. Diagrama de secuencias



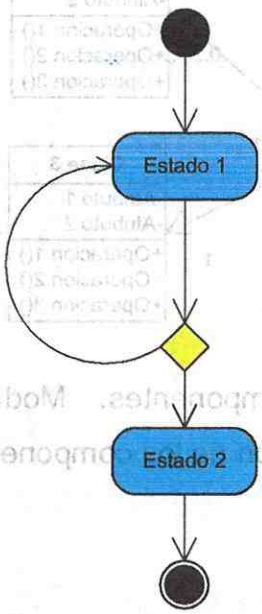
c. Diagramas de colaboración. Modelan la interacción entre los objetos durante un evento del sistema.

Figura 139. Diagrama de colaboración



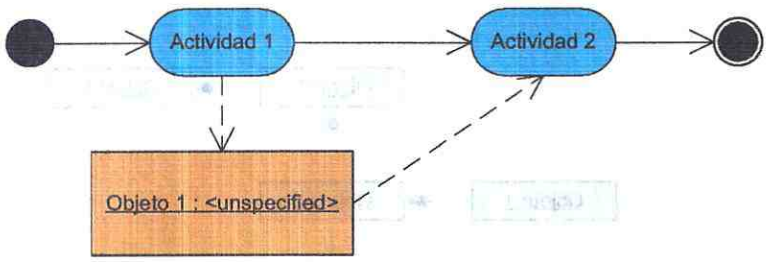

d. Diagramas de estado. Modela el comportamiento del sistema en respuesta a un estímulo externo.

Figura 140. Diagrama de estado



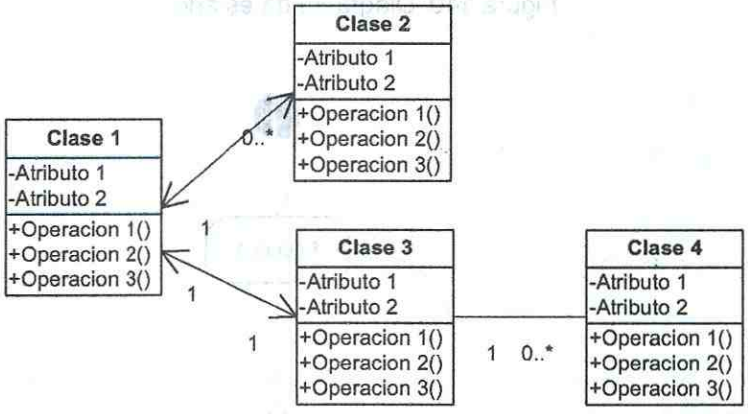
e. Diagrama de actividad. Es una clase especial de diagrama de estado en donde se modela el flujo de control de una actividad a otra.

Figura 141. Diagrama de actividad



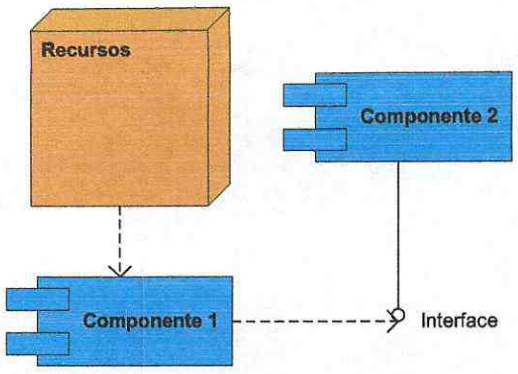

f. Diagrama de clases. Muestra la estructura estática del modelo, esto es, los elementos que existen, mostrando su estructura interna y como interaccionan entre ellos.

Figura 142. Diagrama de clases




g. Diagramas de componentes. Modela los componentes del sistema, describiendo la organización de los componentes físicos de software.

Figura 143. Diagrama de componentes




h. Diagrama de implementación. Modelan la distribución del sistema, describiendo elementos físicos como los recursos, componentes y conexiones.

Figura 144. Diagrama de implementación