UAEM | Universidad Autónoma del Estado de México

CENTRO UNIVERSITARIO UAEM TEXCOCO

“REGISTRO DE FICHAS DE DEPÓSITO Y APLICACIÓN DE PAGOS”

MEMORIA DE EXPERIENCIA LABORAL

QUE PARA OBTENER EL TÍTULO DE

“INGENIERO EN COMPUTACIÓN”

PRESENTA:

OMAR PACHECO RAMÍREZ

DIRECTOR

DR. JOEL AYALA DE LA VEGA

REVISORES

DR. OZIEL LUGO ESPINOSA

M. en I. S. C. IRENE AGUILAR JUAREZ

FEBRERO DEL 2015

Texcoco, México, a 21 de Enero del 2015.

M. en C. Ed. VIRIDIANA BANDA ARZATE
SUBDIRECTORA ACADEMICA
CENTRO UNIVERSITARIO UAEM TEXCOCO.

COPIA

PRESENTE:

AT'N M. EN P.P. ANTONIO INOUE CERVANTES
RESPONSABLE DEL DEPARTAMENTO DE TITULACIÓN

Con base a las revisiones efectuadas al trabajo escrito titulado "REGISTRO DE FICHAS DE DEPÓSITO Y APLICACIÓN DE PAGOS" que para obtener el título de Licenciado de Ingeniería en Computación, presenta el sustentante Omar Pacheco Ramírez, con número de cuenta 0522732 respectivamente, se concluye que cumple con los requisitos teórico – metodológicos, por lo que se le otorga voto aprobatorio para su sustentación, pudiendo continuar con la etapa de digitalización del trabajo escrito

ATENTAMENTE

M. en I. S. C. Irene Aguilar Juárez

Dr. en C. Oziel Lugo Espinosa

Dr en Ed. Joel Ayala de la Vega

c.p.p. Sustentante: Omar Pacheco Ramírez
c.p.p. Director.- Dr en Ed. Joel Ayala de la Vega
c.p.p. Titulación.- M. en P.P. Antonio Inoue Cervantes

TITULACION
RECIBIDO
Por INOUE
Texcoco, Méx., a 21 de 01 del 2015

Contenido

1. Introducción

Desde 2010, Portafolio de Negocios S.A. de C.V. SOFOM ENR cuenta con el sistema BeMoney que le permite generar y administrar la información de cada uno de los créditos que otorga.

A partir de 23 de diciembre de 2011 se publicó en el Diario Oficial de la Federación en la que se reforman, adicionan, derogan las disposiciones, en la modificación de aspectos operativos con el reporte de Operaciones Relevantes, Inusuales e internas preocupantes a Comisión Nacional Bancaria y de Valores (CNBV) por parte de las sociedades financieras de objeto múltiple no reguladas para la prevención de lavado de dinero.

El sistema BeMoney cuenta con funcionalidad para registrar de distinta forma los pagos hacia los documentos de cada crédito a acuerdo a su tabla de amortización, estos a través de pagos directos, pagos parciales, incluso agregar cargos, reprogramar y remplazar documentos, así como registrar depósitos no identificados e identificarlos posteriormente.

Sin embargo, esto no ha sido suficiente ya que la aplicación del pago de un documento en ocasiones parte de la división o dispersión de depósitos superiores qué los clientes suelen realizar para cubrir el pago de más un documento, incluso esto sucede hacia cuentas de diferentes bancos, lo que la identificación en la mayoría de estos casos se hacía manual y visual lo que era lenta, ya que el sistema no contaba con la capacidad de dejar una referencia que pudiera explotar la información de forma eficaz.

Partiendo de esa problemática, es así, que se desarrolló el módulo de “Administración de Pagos”, el cual incluye el **registro de fichas de depósito y la aplicación de pagos**, cuyo objetivo es conocer el origen de cada pago aplicado en los documentos para la identificación de las operaciones para el cumplimiento de la prevención de lavado de dinero.

Siguiendo el marco de trabajo y una metodología adecuada se aprovecharon tecnologías y herramientas informáticas en su mayoría de Microsoft, como Bases de Datos (SQL Server 2008 R2), IDE’s o entornos de desarrollo integrados (Visual Studio 2010), Frameworks

(.Net Framework 4.0) y lenguajes de programación (C# y VB .Net), los cuales permitieron que el desarrollo fuera más rápido y óptimo.

Lo anterior permitió que el módulo de Administración de Pagos se terminara en los tiempos planeados y permitiera cumplir con su objetivo principal que es emitir información confiable para la emisión de información para la prevención de lavado de dinero hacia las entidades reguladoras como la CNVB.

La creación y mejora constante de los sistemas de información a la medida según las necesidades de cada empresa es la base primordial para la emisión de información confiable para la toma de decisiones hacia el otorgamiento de nuevos créditos y la prevención de lavado de dinero.

2. Importancia del módulo de Administración de Pagos

El contar con un módulo que permita registrar la información inicial de los recursos que permiten la aplicación de los documentos de un crédito es primordial ya que permite conocer de forma más precisa los movimientos que se registran en el sistema y permite emitir y reportar información confiable.

Con ayuda del registro de las fichas de depósito y la administración de movimientos y aplicación de pagos a través de estos ha sido de gran ayuda para conocer los movimientos que se realizan entre cuentas del mismo banco e interbancarios, así como su aplicación en los documentos del crédito correspondiente.

Este módulo es utilizado por el área de operaciones y finanzas quienes se encargan de registrar y realizar cada movimiento para la aplicación de pago en los documentos, por lo que bajo permisos pueden o no visualizar y hacer uso de la funcionalidad.

Por ultimo cumple con los requerimientos para el cumplimiento de la normatividad para el reporte de información hacia la prevención de lavado de dinero.

3. Descripción del puesto de Desarrollador Sr.

El área de **Procesos** pertenece a la Dirección de Capital Humano y permite conocer la documentación para ordenar y homologar la información ocupacional y procedimientos con la que cuenta Grupo Administrador Empresarial S.A. de C.V. hacia Portafolio de Negocios S.A. de C.V.

Derivado de esto se describen las funciones que debe desempeña el Programador Analista Sr. como sigue:

Perfil y descripción del puesto Programador Analista Sr.[1]

Objetivo del Puesto

Diseñar, desarrollar, verificar, implementar y mantener en buen estado los programas informáticos de la empresa.

Los resultados se requieren para cumplir con los objetivos del puesto, sugieren tareas o funciones que deben realizarse de acuerdo con una periodicidad, y capacidades de competencia que son necesarias para tener un buen desempeño.

Tareas o Funciones:

- Determinar, en colaboración con la coordinación de sistemas, el plan de programas a desarrollar\carátula y Gantt de proyectos.
- Elaborar gráficos y diagramas para describir y determinar en qué secuencia habrá que proceder al registro y tratamiento de los datos\código documentado.
- Desarrollar e implementar programas informáticos, con base en el plan y programa establecidos/ programas en producción con base en proyecto.
- Asegurar la consistencia de los programas elaborados para eliminar errores\bitácora de pruebas.

[1]Perfil y descripción del puesto Programador Analista Sr. versión 2. Fecha de Consulta Julio 21 2014 GA15_programador_analista_Sr_2011.pdf

- Actualizar los programas de acuerdo a las nuevas normas, procedimientos de operación y requerimientos de la dirección\orden de trabajo cubierta.
- Generar e implementar los reportes del sistema\reportes listos y correctos.

4. Problemática Identificada

En los últimos años Portafolio de Negocios S.A. de C.V. SOFOM ENR ha crecido su cartera en el otorgamiento de créditos, ese crecimiento ha generado nuevos clientes y clientes recurrentes, a la par las normas regulatorias hacia las sociedades de objeto múltiple no reguladas, lo que ha provocado que la administración de cada crédito otorgado deba ser precisa y ha ido afinando sus procesos en el Sistema BeMoney donde se administra y consulta mayor parte de la información.

De esta manera, la presentación de información de manera oportuna y confiable permite tomar decisiones para la planeación y toma de decisiones. Lamentablemente aún hay funcionalidad y procesos que no permiten la validación rápida de la información.

Dentro de BeMoney no existe una manera rápida de identificar el origen real de todos los depósitos que son aplicados como pagos en los documentos de cada crédito, por lo que el proceso de validación debe hacerse manual y visual.

Lo anterior se traduce en:

- Información y referencias incompletas
- Referencias difíciles de detectar y corregir
- Acceso lento a la información
- Pérdida de tiempo en la recopilación de información
- Perdida de referencias en la captura de los datos
- Combinación de referencias de datos
- Duplicidad de trabajo
- Duplicidad de información
- Ejecución lenta de los procesos

Para agilizar el proceso de acceso a la información se pensó concentrar la información en la base de datos de BeMoney y enriquecer la funcionalidad de éste y así sustituir las herramientas con las que se concentraba la información (utilizando hojas de cálculo para

recopilar información, búsquedas manuales y criterios visuales de validación, etc.), eliminando la pérdida de tiempo en la recopilación de la información, a través de nuevas formas de trabajo más centralizadas para identificación y dispersión de los deposito que serán aplicados en el pago de los documentos , y que permita la consulta a la información en tiempo real, para la identificación de Operaciones Relevantes, Inusuales e internas preocupantes para su reporte a la CNVB, ya que esta emisión la realizan dos personas del área de cumplimiento.

Por lo anterior se buscó dar una solución a este problema realizando un análisis de la estructura de la base de datos y la funcionalidad de BeMoney, utilizando los lenguajes de programación, Frameworks e IDE en los que está desarrollado el sistema, que para el proyecto fueron clave para alcanzar los objetivos que se buscaban para la solución.

5. Detalles del puesto de Programador Analista Sr. dentro de Portafolio de Negocios S.A. de C.V. SOFOM ENR

El puesto de Programador Analista Sr. dentro de Portafolio de Negocios S.A. de C.V. SOFOM ENR está especificado en el documento de Perfil y descripción de Puesto Programador Analista Sr., y detallado en el documento llamado catálogo de competencias[2], éste último detalla la interrelación, pensamiento, comportamiento y las técnicas de soporte y el nivel de desempeño que se espera en cada una.

En el catálogo de competencia se tienen tres categorías.

1. Organizacionales (Saber hacer, saber estar)
   1.1. Organizacionales Interrelación
   1.2. Organizacionales Pensamiento
   1.3. Organizacionales Comportamiento
2. Técnicas de Soporte
3. Técnicas Administración.

ORGANIZACIONALES INTERRELACIÓN

OR01 Liderazgo

Capacidad para influir en sus colaboradores y compañeros de trabajo para el logro de los objetivos de la organización. Sabe dirigir el comportamiento, acciones y esfuerzos de sus subordinados, obteniendo de ellos resultados, en un clima laboral adecuado.

- Comprender mensajes. Comprender la información recibida en forma oral-directa o a distancia-, a través de una escucha activa.
- Coordinar las actividades de un equipo o de una persona.
- Planear e instrumentar acciones con respecto a un equipo, coordinar y promover las actividades derivadas del plan de trabajo mediante el convencimiento.

[2] Catálogo de Competencias. Fecha de Consulta Julio 26 2014. <catálogo_competencias_administración.pdf> pag.1-2.

OR02 Integración:

Capacidad para integrarse a un quipo y participar sinérgicamente con él hacia el logro de las metas comunes.

OR03 Comunicación Interpersonal (asertividad)

Capacidad para transmitir ideas y sentimientos a los de más de forma clara, oportuna y adecuada.

- Organizar la comunicación y responder en forma oportuna.
- Aclarar el propósito e importancia, enfatizar los puntos principales y seguir una secuencia lógica.
- Responder con oportunidad, adecuación y precisión a los requerimientos de información y orientación y plateados.

OR04 Manejo de Conflictos

Capacidad para manejar efectivamente a otros en una situación antagónica: utilizar estilos y métodos interpersonales apropiados para reducir la tensión o conflicto entre dos o más personas.

OR05 Negociación

Capacidad para explorar de manera efectiva alternativas y posiciones para arribar a resultados que aseguren el apoyo y la aceptación de todas las partes frente a un asunto concreto.

- Aclarar la situación. Explorar las necesidades, inquietudes, intereses y posiciones iniciales de las partes, incluyendo la propia y enfatizar las áreas de concordancia y de desacuerdo.

OR06 Organización, Planeación y Control

Capacidad para establecer rutas de acción para uno mismo y para otros a fin de asegurar que el trabajo sea completado con eficiencia.

- Programar. Ubicar unidades apropiadas de tiempo para completar el trabajo propio y de otros; evitar conflictos de programación; desarrollar líneas de tiempo.

OR07 Delegación

Capacidad para asignar responsabilidad en la toma de decisiones y/o responsabilidad por tareas a las personas apropiadas, a fin de maximizar la efectividad de la organización y los individuos.

- Compartir las responsabilidades apropiadas. Otorga la autoridad para la toma de decisiones/responsabilidades en las áreas y personas adecuadas, considerando el impacto positivo y negativo, los valores, estructuras organizacionales y el incremento del conocimiento/habilidades del individuo).
-

ORGANIZACIONALES PENSAMIENTO

OR08 Análisis y solución de problemas

Capacidad para segmentar un problema en tantas partes como sea posible, para reconocer la naturaleza de sus partes, las relaciones entre éstas y plantear una solución pertinente que lo resuelvan en forma integral.

- Plantear y desarrollar alternativa de solución.
- Describir y desarrollar las opciones de solución al problema humano, técnico u organizacional que se trate.

OR09 Toma de decisiones

Capacidad para elegir entre diversas alternativas con base en un análisis consistente que permita predecir, en lo posible, los efectos.

OR10 Pensamiento Estratégico

Capacidad para comprender las circunstancias personales, del área y de la organización, con respecto a su entorno para asegurar decisiones y rutas de acción con una visión de amplio alcance.

- Reunir y organizar información.
- Obtener información e identificar asuntos clave y relaciones que sean relevantes: datos para identificar/explicar tendencias, problemas y causas importantes;
- comparar y combinar información para identificar asuntos subyacentes.

OR11 Pensamiento Sistémico

Capacidad para comprender la estructura de un sistema y las relaciones que se establecen entre sus elementos y los subsistemas y supra sistemas que lo contienen.

- Tener conciencia. Comprender que las personas y las organizaciones somos parte de un sistema.
- Identificar nuestra posición y las de los demás y saber de los efectos de las acciones u omisiones recíprocas.
- Participar. Reconocer los alcances del sistema al que pertenecemos (persona, área, organización, entorno) y actuar en el marco de nuestra influencia.
- Diseñar. Reconocer la relación entre los elementos de un sistema y diseñar o rediseñar dichas relaciones en función de la efectividad del mismo.

ORGANIZACIONALES COMPORTAMIENTO

OR12 Empuje

Capacidad para actuar con energía y voluntad en forma proactiva y con efectividad.

- Responder con prontitud. Emprender acciones inmediatas al confrontar un problema o al adquirir conciencia de una situación.

- Mantener vigor y efectividad. Mantiene un fuerte ritmo de trabajo y realiza actividades exigentes física y mentalmente y logra sostenerlos a través del tiempo.

OR13 Autorregulación (equilibrio)

Capacidad para atender las dimensiones física, emocional, mental/aprendizaje y espiritual en la vida personal y trasladar este equilibrio a los ambientes laboral y familiar en los que se despliega.

OR14 Tolerancia a la presión

Capacidad para mantener un desempeño estable bajo presión u oposición (como presiones de tiempo o ambigüedad de puesto); manejar el estrés de una manera que sea aceptable para otros y para la organización.

OR15 Desarrollo de los demás

Capacidad para desarrollar competencias, retroalimentar y orientar a los colaboradores o colaterales con el fin de que cumplan con eficacia su trabajo.

- Explicar y demostrar. Proporcionar instrucción, modelos positivos y oportunidades de observación, a fin de ayudar a otros a desarrollar habilidades: alentar la formulación de preguntas para asegurar la comprensión.

OR16 Tenacidad

Capacidad para conservar una postura o plan de acción hasta obtener el objetivo deseado o hasta que deje de ser razonablemente alcanzable.

- Persistir en el esfuerzo. Trabajar para alcanzar la meta a pesar de barreras o dificultades;
- Trabajar de manera activa para superar obstáculos cambiando estrategias, redoblando esfuerzos, utilizando abordamientos múltiples, etc.
- Resistir. Mantener la efectividad y el entusiasmo después de una decepción o rechazo.

- La resistencia significa ser capaz de volver a ponerse en pie y seguir adelante.

OR17 Creación de alianzas (tolerancia a la diversidad)

Capacidad para desarrollar y utilizar relaciones estratégicas de trabajo entre la propia área y otras áreas, equipos, departamentos, unidades u organizaciones para el logro de metas del negocio.

- Buscar oportunidades y aclarar la situación presente.
- De manera proactiva trata de crear relaciones de trabajo efectivas y aclara situaciones con las demás personas, áreas u organizaciones.

TECNICAS DE SOPORTE

TS01 Gestión efectiva del tiempo

Capacidad para administrar el tiempo en forma eficaz para sí mismo y para otros

- Programar actividades. Identificar las unidades de trabajo en el tiempo, el espacio y la duración adecuados para completar el trabajo propio y de los otros para realizar el trabajo con eficacia.

TS02 Manejo y participación en reuniones.

Capacidad para preparar, participar facilitar y dar seguimiento a una reunión de trabajo para que ésta cumpla con los objetivos establecidos.

- Preparar la reunión de trabajo. Determinar la necesidad, el propósito, de una reunión y las condiciones idóneas para su realización (participantes, lugar, fecha, hora, soporte, material…).

TS03 Comunicación escrita

Capacidad para comunicar ideas y sentimientos de manera escritos a una persona o aun grupo.

TS04 Administración de Procesos

Capacidad para administrar uno o más procesos con el propósito de asegurar la eficacia de la organización.

- Comprender y ajustarse. Entiende un proceso y puede ajustar sus tareas a los estándares de este.

TS05 Diseño y Desarrollo de Proyectos

Capacidad para administrar proyectos con el propósito de asegurar el desarrollo de la organización.

TS06 Adaptabilidad y promoción del cambio

Capacidad para facilitar el cambio, mantener la efectividad durante el proceso y ajustarse a las nuevas condiciones.

TS07 Asunción de riesgos

Capacidad para actuar a favor de un beneficio o ventaja reconocidos con base en la comprensión y administración del riesgo que supone.

- Calcular el riesgo. Reunir información para comprender la probabilidad y los beneficios del éxito y las consecuencias del fracaso y decidir su viabilidad.

TS08 Alineación y productividad (estándares)

Capacidad para concentrarse y guiar a otros en los logros de los objetivos organizacionales y evaluar mediante estándares el avance.

TS09 Innovación (creatividad)

Capacidad para generar soluciones innovadoras en situaciones de trabajo; poner a prueba maneras diferentes y novedosas de enfrentar problemas y oportunidades de trabajo.

- Asegurar la relevancia. Identificar importantes áreas de innovación y desarrollar soluciones que aborden asuntos de trabajo significativos, inspirado en múltiples fuentes (individuos, disciplinas…).
- Pensar de manera expansiva. Combinar ideas de manera única, hacer conexiones entre ideas dispares.
- Explotar diferentes líneas de pensamiento.
- Visualizar las situaciones desde múltiples perspectivas.
- Generar múltiples abordamientos/soluciones.

TECNICAS ADMINISTRACIÓN

TA12 Manejo de lenguajes de desarrollo

Capacidad para manejar lenguajes de desarrollo de software.

- Nivel Avanzado.

TA13 Aplicaciones propias de servidores

Capacidad para manejar servidores

- Nivel Intermedio
- Nivel Avanzado

TA14 Lenguaje de diagramación y documentación de sistemas UML

Capacidad para manejar lenguajes de diagramación u documentos de sistemas UML.

- Nivel Intermedio
- Nivel Avanzado

TA15 Técnicas de Calidad Software

Capacidad para manejar técnicas de calidad de software.

- Nivel Avanzado

6. Análisis del Módulo de Administración de Pagos

6.1 Introducción al Problema

Portafolio de Negocios S.A. de C.V SOFOM ER hoy es una empresa que está clasificada como una sociedad de objeto múltiple y entidad no regulada (SOFOM ENR), la cual otorga créditos a Pequeñas y Medianas Empresas (PyMEs), es supervisada y sujeta a disposiciones por la Comisión Nacional Bancaria y de Valores (CNBV).

La CNBV en sus disposiciones legales en materia de prevención de lavado de dinero señalan que una SOFOM tiene como obligación: “reportar aquellas operaciones relevantes y operaciones inusuales que rebasen las cantidades realizadas por un monto igual o superior al equivalente en moneda nacional o 10000 dólares EUA , esto con el fin de seguir las normas para la prevención de lavado de dinero, que refieren a las Disposiciones de carácter general a que se refieren los artículos 115 de la Ley de Instituciones de Crédito en relación con el 87-D de la Ley General de Organizaciones y Actividades Auxiliares del Crédito y 95-Bis de este último ordenamiento, aplicable a las sociedades financieras de objeto múltiple, (Disposiciones de carácter general) en materia de prevención de operaciones con recursos de procedencia ilícita y financiamiento al terrorismo”.

Actualmente Portafolio de Negocios S.A. de C.V. SOFOM ENR cuenta con el sistema BeMoney, el cual tiene módulos que la captura de una cotización, la generación de un contrato hasta la administración del mismo, así como emisión de reportes que ayudan a revisar y analizar el control de las operaciones realizadas día con día.

La generación de un contrato en BeMoney requiere de la captura de datos que permitan calendarizar la forma en la que el cliente deberá liquidar su crédito, es decir una "tabla de amortización", esto con el fin llevar un control y generar un perfil transaccional de pago, el cual ayude a identificar y cumplir las disposiciones para las operaciones antes mencionadas. A continuación se listan los datos que definen a un crédito (contrato) y que permite la generación de su tabla de amortización:

- Capital
- Plazo o Número de pagos
- Periodicidad
- Pago inicial
- Pago Mensual o Mensualidad
- Intereses
- IVA de Interés
- Gastos Administrativos
- IVA Gastos Administrativos
- Divisa
- Tasa Mensual
- Tasa Anual
- % IVA
- Fecha Inicial
- Fecha de Contrato
- Fecha Primer Amortización

También tiene otras características las cuales determinan su clasificación:

- Tipo de Crédito

- Origen de Cartera(Fideicomiso)

La administración de un contrato en BeMoney se realiza a través del módulo denominado “Administración de Cartera por Cliente” que en diferentes secciones muestra la información relacionada del cliente respecto al contrato consultado. En la sección de “Documentos Pendientes de Pago” se cuenta con la funcionalidad para permitir aplicar el pago a los documentos con estatus pendiente de pago relacionados a cada periodo de su tabla de amortización. Así esta funcionalidad permite cubrir y realizar operaciones como:

Pago Directo: Permite aplicar el pago de un documento por el monto establecido al momento de la documentación en la generación de un contrato, Normalmente se utiliza para cobrar cheques.

Pago Parcial: Permite realizar un pago por una cantidad menor al documento original, este genera un nuevo documento con referencia al inicial y deja un saldo pendiente para saldarlo posteriormente, normalmente se cobran primero intereses y después capital.

Remplazo de Documento: Permite realizar como se indica el cambio de un documento por otro el cual permita la aplicación del pago del mismo, este puede ser por los siguientes tipos de documento o forma de pago:

- Cheque
- Pagaré
- Tarjeta de Crédito
- Amexco (American Express)

- Efectivo
- Transferencia Electrónica
- Bienes
- Depósito en Efectivo

Agregar Cargo: Permite agregar cargos adicionales por los siguientes conceptos:

- Cargo por Movimiento.
- Comisión por Amexco.
- Comisión por Tarjeta de Crédito.

Agregar Excedentes: Permite agregar o crear un documento que indique que el cliente pago de más.

Un documento asociado a una amortización contiene la información necesaria para saber de qué manera se le aplicara un pago según las formas descritas anteriormente, a continuación se presenta los datos que describen el detalle de un documento al cual se le aplicara una forma de pago:

- No. Documento
- Fecha de Envió a Cobro
- Tipo de Documento
- Divisa
- Concepto
- Banco PDN
- Banco Cliente
- Cuenta PDN
- Cuenta Cliente
- Total

- Capital
- Interés
- IVA Interés
- Gastos Administrativos
- IVA Gastos Administrativos
- Cargos Adicionales
- IVA Cargos Adicionales
- % IVA
- Comentarios

Los datos de un documento describen y permiten conocer la forma en que se aplicar el pago; datos como el nombre del Banco Cliente, Banco PDN o cuenta Cliente y cuenta PDN son datos que describen y representan de manera general el detalle de cómo se aplicó el pago, sin embargo, en ocasiones estas referencias suelen ser origen de depósitos incluso mayores a los que indica el monto que cubre el total del documento ya que pudo haber sido ser dispersado al pago de uno o más documentos para diferentes contratos del cliente y que además van dirigidos a las cuentas de diferentes bancos según definidas en la documentación de acuerdo al origen de cartera del contrato.

A continuación se describen algunos casos que se pueden presentar al aplicar un pago.

1. Un depósito fue realizado a una cuenta de PDN 1012311 en el banco A, esta es identificada hacia cliente 1, este ha indicado que el depósito es para cubrir el importe dos documentos, el del periodo en curso además de otro que tenía vencido en su contrato(crédito) 1 , es decir:

a. Depósito disperso entre documento 1 y documento 2 en la cuenta 1012311 de PDN del banco A del contrato 1.

2. Un depósito fue realizado a la cuenta de PDN 110110003 en el banco A, esta es identificada hacia el cliente 1, cabe mencionar que este cuenta con dos contratos (créditos) activos. El cliente indica que el deposito cubre el pago un documento por cada contrato que tiene activo, es decir:

   a. Depósito 1 disperso entre documento 1 de contrato 1 y documento 1 de Contrato 2 en la cuenta 110110003 del Banco A.

3. Un depósito fue realizado a la cuenta de PDN 00101021 en el banco B, esta es identificada hacia un cliente 2. El cliente ha indicado que el depósito solo es para cubrir una parcialidad del total del importe del documento, sin embargo días después realiza otro depósito el cual cubre el saldo pendiente del mismo contrato, es decir:

   a. Depósito 1er pago parcial documento 1 en la cuenta 00101021 del banco B, del contrato 1.
   b. Depósito 2do pago parcial documento 1 en la cuenta 00101021 del banco B, del contrato 1.

4. Un depósito fue realizado a una cuenta en determinado banco, esta es identificada hacia un cierto cliente, el cliente indica este cubre el importe de uno de los cheques que se tenían considerado para su pago en cierto periodo para determinada contrato.

a. Depósito 1 remplaza cheque de documento 1 de contrato 1.

5. Un depósito fue realizado a una cuenta en determinado banco, esta cuenta es identificada hacia un cierto cliente, partimos que el cliente tiene dos contratos y de acuerdo a lo pactado en sus documentos de contrato 1 y contrato 2 manejan diferentes cuentas destino, entonces el cliente indica que hizo el deposito a la cuenta de los documentos del contrato 1 y que este debe ser destinado para un documento del contrato 2 el cual tiene una cuenta diferente a la cual debe realizar el depósito, esto se debe a que el centro 1 Pertenece a Cartera Original PDN y el contrato 2 Pertenece a la Fideicomiso Maestro.

Los ejemplos anteriores nos presentan una descripción de algunas de las tantas formas en que los clientes de Portafolio de Negocios suelen cubrir los importes de los documentos que están por vencer en determinado periodo y que además puede aplicarse para diferentes contratos según su clasificación por tipo de crédito y origen de cartera. Analizando los casos anteriores podemos darnos cuenta que:

En los casos que la captura de los datos mencionados con anterioridad permiten identificar el pago y pareciera no tener problema, sin embargo, ante la necesidad de cubrir normas descritas anteriormente para identificar operaciones inusuales para la identificación de lavado de dinero, la funcionalidad actual no genera y no almacena esa relación entre las disposiciones de un deposito que se dispersa entre los diferentes documentos como se muestra en los ejemplos presentados. La falta de funcionalidad al intentar identificar si existe una relación entre los pagos se vuelve difícil determinar la existencia de operaciones inusuales o relevantes de parte de los clientes.

Nota1: Los Tipos de crédito pueden variar las características de cada contrato

Nota2: El Origen de la cartera (Fideicomiso) indica ciertas caracterizas para la administración del contrato entre ellas el manejo de las cuentas bancarias a las que se deben realizar los depósitos.

Nota3: Los Movimientos para realizar los traspasos entre cuentas y traspasos interbancarios se realizan atreves de un formato de solicitudes de transferencias realizado en Excel.

6.2 Objetivo

El Objetivo principal de este proyecto ha sido implementar un mecanismo que a través de fichas de depósito permite el rastreo de los depósitos recibidos para la aplicación del pago de los documentos de cada crédito, con el fin de identificar y reportar aquellas operaciones relevantes, inusuales y preocupantes hacia la prevención de lavado de dinero.

Esto permitirá además explotar información más precisa, la cual permite conocer movimientos de los cuales anteriormente no se podía detectar a través del sistema

Así mismo ha traído una centralización de información, para la identificación de movimientos interbancarios o entre cuentas, las cuales son muy comunes en la operación día a día.

6.3 Objetivos del Módulo de Administración de Pagos

Si bien el sistema BeMoney contaba con funcionalidad que permitía aplicar los pagos de los documentos y capturar información que permitía saber cómo se había realizado, está aún se encontraba incompleta según lo descrito anteriormente, y ante la necesidad de obtener información más eficaz y precisa se decidió crear el módulo de Administración de Pagos.

El objetivo principal de este módulo es generar información que permita reconocer el origen de las operaciones con las cuales han sido aplicados los pagos en los documentos de los créditos y para lograr esto se tomaron en cuenta los siguientes objetivos:

- Contar con una herramienta que permita administrar las fichas de depósito.
- Poder registrar depósitos no identificados para su futura identificación.
- Poder aplicar los pagos desde el modulo.
- Poder identificar el origen del pago de un documento mediante una ficha de depósito.
- Ahorro y tiempo para capturar los datos para la aplicación de los pagos.
- Crear un modelo relacional con las entidades de la base de datos actual del sistema.
- Utilizar las herramientas actuales de desarrollo.
- Construir el modulo bajo la arquitectura actual del sistema.

6.4 Análisis de Requerimientos

El análisis de requerimientos permite especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software. La obtención de estos nos permite

implementar técnicas que faciliten y aceleren el cumplimiento de los objetivos planteados y así dar solución a la problemática o necesidad planteada.

La identificación de entidades y la relación que existen entre ellas permiten definir las reglas de negocio que nos dan pauta para dar el tratamiento correcto y dar consistencia a los datos.

6.4.1 Entidades

Si nos referimos a un desarrollo como una solución, y que además ésta se realiza con un paradigma orientado a objetos y que utilizara un almacenamiento de información en una base de datos, como primer paso es identificar las entidades que tendrán relación con cada uno de los procesos que serán integrados.

En la solución se han detectado las siguientes entidades:

- Ficha de Depósito
- Cliente
- Contrato
- Documento
- Usuario
- Perfil

Las cuales se describen a continuación brevemente:

Ficha de Depósito

Documento que permite registrar el depósito realizado por algún cliente para ser aplicada e indique el pago de algún documento de su(s) contrato(s).

Cliente

Representa una entidad con un régimen fiscal (Persona Física, Persona Física con Actividad Empresarial, Persona Moral), que tiene una capacidad de pago.

Contrato

Acuerdo de voluntades, escrito, manifestado en común entre Portafolio de Negocios S.A. de C.V. SOFOM ENR y su Acreditado.

Documento

Representa de manera física la obligación por parte del cliente a pagar la cantidad indicada en éste, de acuerdo a un calendario de pagos, lo que indica una fecha para el cobro del mismo, este puede ser un cheque o un pagare.

Perfiles/Roles

Entidad encargada de agrupar a los usuarios para diferenciarlos y otorgarles mayores o menores privilegios dentro del sistema, para el Modulo de Administración de pagos son los siguientes.

- **Administrador**: Los usuarios con este perfil tienen los máximos privilegios del sistema, dentro del Módulo de Administración de Pagos podrá dar de alta, modificar (identificar, aplicar, devolver, realizar movimientos) y dar de baja (cancelar) las fichas de depósito, además de aplicar en un documento para indicar el pago de éste.

- **Aplicador de Pagos**: Este perfil podrá crear, identificar y cancelar las fichas de depósito, puede realizar movimientos además de que tiene acceso a la aplicación de las mismas en los documentos de los contratos para indicar el pago de éstos.

También puede generar solicitudes de transferencia, los cuales le permite realizar movimientos de transferencia entre cuentas y traspasos interbancarios

- **Capturista de fichas de depósito:** Este perfil tiene privilegios mínimos para dar de alta las fichas de depósito, y si en su caso fuesen identificados hacia algún cliente, sólo puede cancelar las fichas si éstas no han tenido algún movimiento previo.

6.4.2 Problemática

- Existe pantalla para realizar solo registro de Depósitos no Identificados
- Existe funcionalidad para aplicar pagos de forma directa, sin pasar por depósitos no Identificados.
- La aplicación de registro sólo se realiza cargando cada contrato
- Se pierde la referencia del depósito cuando se aplica a diferentes documentos
- No existe histórico de los traspasos interbancarios o transferencia entre cuentas de los depósitos aun cuando estos han sido identificados
- Los traspasos y transferencias se realizan por separado mediante procedimientos fuera del Sistema BeMoney.
- No se tiene la identificación de los movimientos de traspasos interbancarios y transferencias entre cuentas.
- No se tiene la información o dato preciso el cual se requiere para reportar operaciones, inusuales relevantes y mayores a $100,000.00 MXN. Ante la CNBV.
- La recolección de los datos que se reportan actualmente, son lentos y están basadas en la información final del pago por documento y no por depósito realizado por cliente.
- No existe un módulo centralizado que permita el registro de los depósitos no identificados, la identificación de los mismos, realizar movimientos y aplicarlos en los documentos como medio de pago.

6.5 Recopilación de necesidades

El Ing. Pablo Cesar Dorantes Alcantara, Gerente del Área de Sistemas y la Lic. Lucia Castellanos Xolocotzi, Líder de Proyecto del Área de Desarrollo se dirigieron conmigo para que me encargara del proceso de recopilación de necesidades.
Dicho proceso se llevó a cabo dentro Portafolio de Negocios e involucro a las áreas de Operaciones y Cumplimiento haciéndoles mención que habría una restructuración en la funcionalidad y procesos del sistema.

A cada una de las personas se les pidió que dieran su punto de vista y experiencia con la funcionalidad actual y así conocer la practicidad de la misma para realizar sus procesos o actividades, también para saber si la información que se extrae es la suficientes para generar sus reportes.

Esto permitió determinar los datos y los campos que se requieren para el diseño de las nuevas tablas en la base de datos y priorizar el desarrollo de la primera versión del módulo.

6.5.1 Necesidades del Sistema

Después de las diferentes sesiones donde se realizaron entrevistas al personal de las áreas de Operaciones y Cumplimiento se tomó en cuenta lo siguiente:

1) Crear catálogos que permitan identificar:
    a) Tipo de Movimientos que se le aplica a la ficha de depósito.
    b) Forma de Pago con que se registra la ficha de depósito.
2) Crear un área de trabajo que permita
    a) Capturar las fichas de depósito, estén o no identificados.

b) Identificar o asociar a algún cliente a las fichas de depósito.
c) Realizar o aplicar movimientos a las fichas de depósito.
d) Aplicar la ficha en los documentos pendientes de pago.
e) Realizar cancelaciones de los movimientos a las fichas de depósito.
f) Marcar como devuelta una ficha de depósito.
g) Integración de funcionalidad actual de la aplicación pagos para tener un módulo más completo.

3) Contar con perfiles de usuario y que puedan hacer los siguiente:
   a) Un perfil que solo pueda capturar las fichas de depósito.
   b) Un perfil que pueda identificar las fichas de depósito.
   c) Un perfil que pueda:
      i) Realizar movimientos a las fichas de depósito.
         (1) Traspaso interbancario
         (2) Transferencia entre cuentas
         (3) Recuperación de Fideicomiso
         (4) Recuperación de Saldos
      ii) Aplicar las fichas de depósito como pago en los documentos pendientes de pago.
      iii) Realizar cancelaciones de movimientos y fichas de depósito.
      iv) Marcar como devueltas las fichas de depósito.
   d) Un perfil de administrador que pueda realizar todas las funciones anteriores.

7. Diseño del Módulo Administración de Pagos

7.1 Quien Autorizo el Módulo

La detección de las operaciones relevantes, inusuales que se emiten a la Comisión Nacional Bancaria y de Valores (CNBV), venía realizándose a través de un proceso en el que no se tenían los datos completos para el rastreo de los mismos en la base de datos, y

por consecuencia en el proceso y de manera visual en el sistema BeMoney, lo cual el Lic. Oliver Omar Cano Rey, Oficial de Cumplimiento, en una primera reunión con el Ing. Pablo Cesar Dorantes Alcantara, Gerente del Área de Sistemas, le expreso el problema y la necesidad que se tenía, ya que por disposición oficial y por plazo de fechas para tener una solución se necesitaba atender inmediatamente.

En una reunión posterior, que incluyó al Director de las área de Operaciones el Ing. Alejandro de la Peña Villela y el Director de Finanzas Carlos Garcia Mangin, el Ing. Pablo Cesar Dorantes Alcantara y el Lic. Oliver Omar Cano Rey, presentaron las necesidades y los puntos que afectaban a cada área en particular, lo cual llevo a tomar la decisión de crear un módulo que permitiera una administración centralizada de los pagos realizados por los clientes cada periodo.

El Lic. Carlos Garcia Mangin, Director del Área de Finanzas y siendo parte de esta el área Sistemas, es quien autorizo el desarrollo del Módulo de Administración de Pagos.

7.2 Tramites

Los trámites se realizaron de manera interna, en Portafolio de Negocios S.A. de C.V. SOFOM ENR con el Lic. Carlos Garcia Mangin Director de Finanzas y el Ing. Pablo Cesar Dorantes Alcantara Gerente del área de Sistemas. Quienes en acuerdo común dieron su visto bueno para empezar a realizar el modulo.

7.3 Requerimientos

Los requisitos se tomaron de manera interna con los usuarios del sistema en las áreas de Finanzas, Operaciones y Cumplimiento, debido a que serían los futuros usuarios del módulo, y de esta manera conocer que es lo que desean del módulo y saber de manera más acertada lo que el sistema será capaz de realizar.

Algunos usuarios ya han tenido contacto con módulos y funcionalidad de BeMoney, lo que ayudo a ser conocer las necesidades específicas de lo que se requiere.

Partiendo de las necesidades de la prevención de lavado de dinero y de que se pretende crear un módulo en el que se integran fichas de depósito para el registro de los depósitos para el cubrimiento del saldo de los documentos de los créditos de los cliente de Portafolio de Negocios S.A. de C.V. SOFOM ENR, surgieron los siguientes comentarios:

- Que la captura de los datos no sea complicada.
- Poder dar de alta una ficha de depósito sin identificar.
- Realizar movimientos entre cuentas y traspasos interbancarios
- Registrar las transferencias entre cuentas y traspasos interbancarios.
- Saber quién da de alta, asigna, aplica o cancela una ficha de depósito.
- Facilitar la aplicación de la ficha de depósito.
- Que se enlace los datos relevantes de la ficha de depósito a las formas de pago.
- Poder consultar como se aplicó el pago en los documentos.
- Generar un reporte para conocer el detalle de los pagos de un cliente atreves de las fichas de depósito.

7.4 Diseño base de datos.

El consumo de los datos a través de medios electrónicos debe ser confiable, para cumplir este objetivo es importante crear un buen diseño de base de datos para que pueda ser escalable, para que la explotación de ésta genere información que lleve a la toma de decisiones.

Las consideraciones en el diseño de las tablas para el Módulo de Administración de Pagos son las siguientes:

- La velocidad de acceso
- Tamaño de la información

- El tipo de la información
- Facilidad de acceso a la información
- Facilidad para extraer información
- Transaccionalidad

BeMoney cuenta con una base de datos que es administrada en SQL Server 2008 R2, por lo que el diseño e implementación es sobre esta plataforma.

Entre las principales características de SQL Server R2 que nos brinda para poder diseñar y administrar una base de datos son:

- Facilidad de instalación, distribución y utilización.
- Almacenamiento de datos
- Soporte de Transacciones
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Incluye también entorno grafico de administración, que permite el uso de comandos DDL (Definición de Datos) y DML (Manipulación de Datos) gráficamente.
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
- Permite administrar información de otros servidores de datos

SQL Server proporciona un conjunto de tipos de datos del sistema que define todos los tipos de datos que pueden utilizarse con SQL Server y están clasificados en:

- Numéricos Exactos
- Numéricos Aproximados
- Fecha y hora
- Cadenas de caracteres

De esta forma es como se llegó al diseño de las entidades en la base de datos que se utilizaron para el almacenamiento de los datos en el Modulo de Administración de Pagos. Como se muestra en la Figura 1.

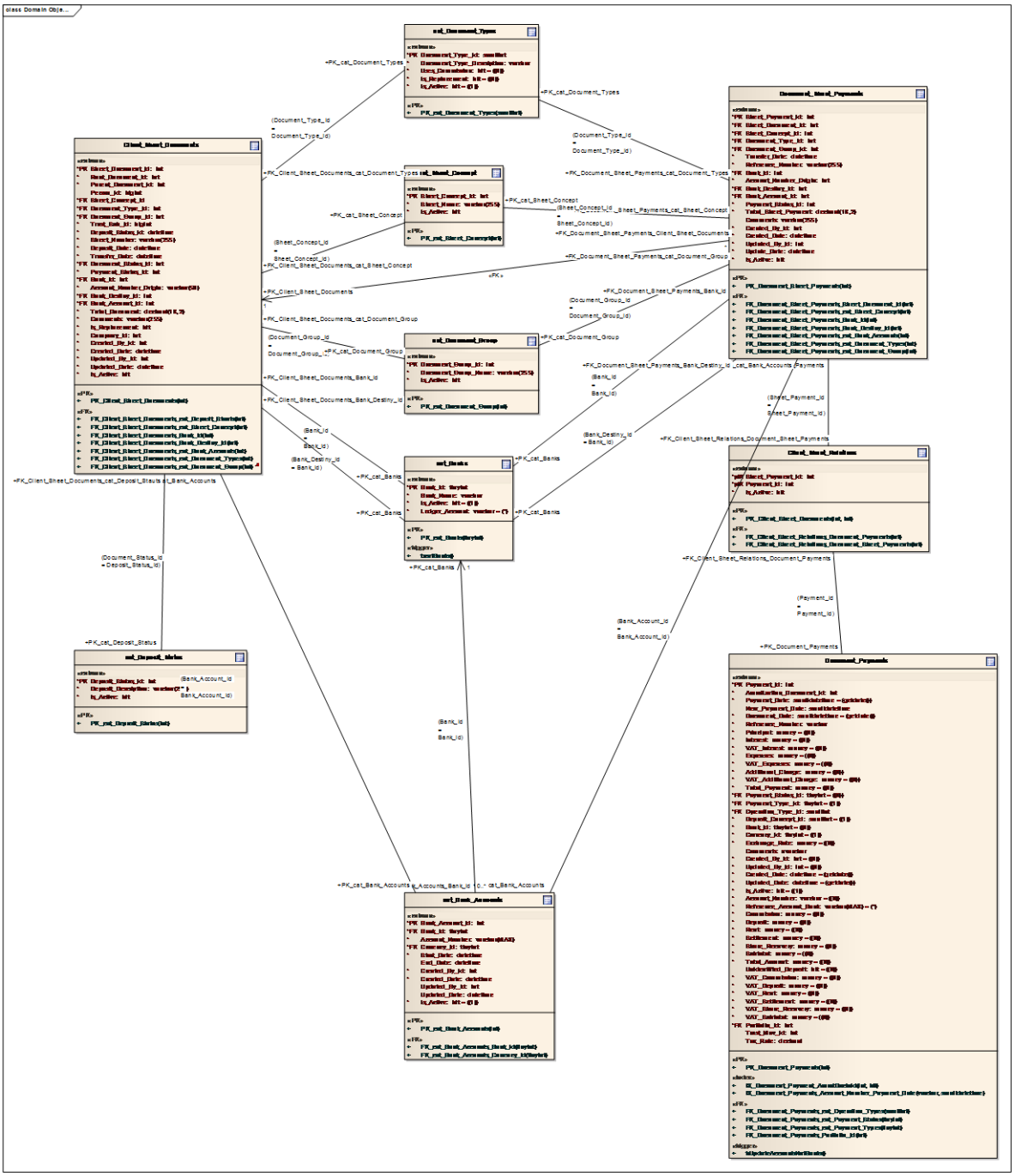

Figura 1. Entidades Modulo Administración de Pagos (Fichas de Depósito).

7.5 Lenguaje a utilizar

El uso de Visual Basic .net (VB.Net) y C# como lenguaje de programación se debió principalmente a la arquitectura de BeMoney y que además están escritos bajo .net Framewrok 4.0 de Microsoft lo que permite compatibilidad entre estos lenguajes.
Tanto VB.Net y C# son orientado a objetos, lo que nos da la posibilidad de definir un objeto y utilizar múltiples instancias de éste para implicar nuestro código.

El .Net Framework 4.0 cuenta con una extensa documentación por parte de Microsoft para la consulta de librerías, así como la implementación de las mejores prácticas, para el uso de éstas.

También existe una comunidad activa, la cual permite conocer solución a las dudas y/o problemas en los que nos podemos enfrentar en nuestro proceso de desarrollo.

El uso de un IDE (Entorno de Desarrollo Integrado por sus siglas en inglés) permite tener un mejor y más rápido proceso de desarrollo, Visual Studio 2010 es un IDE desarrollado por Microsoft que trabaja con el .NET Framework 4.0 y que integra el uso de leguajes de programación como VB.Net y C# entre otros. Además de que provee de herramientas que nos ayudan optimizar los proyectos.

7.6 Donde se encuentra ubicado el sistema

BeMoney es un sistema de escritorio que se instala en cada uno de los equipos que están autorizados, se distribuye a través de la intranet de Portafolio de Negocios S.A. de C.V. SOFOM ENR.

Requisitos Mínimos para Instalar BeMoney:

- Procesador 1Ghz.
- 512 MB
- 850 MB HDD 32bits, 2GB HDD 64Bits
- Windows XP

La Base de Datos está alojada en un equipo H.P. Proliant Generación 5 con las siguientes características:

- Dual-Core Intel® Xeon® Processor L5240 (3.00 GHz, 40Watts, 1333 FSB)
- RAM 16 GB
- Raid 5 vol. 1.6
- 2x 146GB 10K SAS Hard Drives

Sistema Operativo:

- Windows Server 2008R2 Enterprise 64 Bits
- SQL Server 2008 R2 SP1

7.7 Pruebas

Las pruebas fueron realizadas principalmente por el mi compañero el Lic. Miguel Ángel Herrera Arano, quien es el Tester quien es el encargado de realizar las pruebas a los desarrollos de la empresa.

Las principales pruebas que se realizaron fueron la usabilidad y estabilidad al capturar y editar la información, así como las validaciones y reglas de negocio que se definieron al realizar el análisis de la solución.

Se realizó un paralelo en la aplicación entre la versión anterior y la nueva para la aplicación de los pagos en producción con el fin de comparar la información arrojada fuese correcta.

7.8 Áreas participaron en estas pruebas

El área de desarrollo tiene sólo una persona realizando pruebas, lo que hace que la revisión de los datos sea más difícil de verificar, por lo que en paralelo se integraron las áreas de operaciones y finanzas quienes son las encargadas de dar de alta las fichas de depósito y la aplicación de pagos, además de que son las que conocen mejor el flujo de las transacciones que se realizan día a día.

7.9 Capacitación

La importancia de conocer el funcionamiento de un sistema para llevar una buena operación, para el Módulo de Administración de Pagos no fue la excepción, por lo que se tuvo que capacitar al personal de las áreas de Finanzas y Operaciones, con un total de cinco personas, lo que facilito que las dudas fueran más puntuales.
La capacitación se llevó a cabo dentro de la empresa en horario de trabajo en el que durante 2 horas se explicó de manera técnica la importancia del módulo y la funcionalidad, y los casos esenciales para el registro de las fichas de depósito, la asignación de un cliente a una ficha de depósito, la aplicación de transferencias entre cuentas, la aplicación de traspasos interbancarios, la cancelación y la devolución de una ficha de depósito, así como la aplicación de éstas a un documento para registrarse como un pago.

En una reunión adicional se le mostro a grandes rasgos el modulo al Oficial de cumplimiento a quien se le explico la importancia del módulo, ya que permitirá conocer de manera más completa el flujo que sigue un depósito realizado por algún cliente hasta la aplicación hacia un documento. Esta información se consume a través de un servicio que permite consultar los pagos y fichas de depósito para que posteriormente se pueda realizar un proceso de depuración en otro sistema el cual permite administrar las

operaciones inusuales, operaciones relevantes, operaciones internas preocupantes, de acuerdo a las disposiciones emitidas por la CNVB.

8. Experiencia adquirida al desarrollar el Módulo Administración de Pagos

En el desarrollo del Módulo de Administración de Pagos me he permitido trabajar con herramientas como:

- **Visual Studio 2010:** Visual Studio es un conjunto completo de herramientas de desarrollo para la generación de aplicaciones web ASP.NET, Servicios Web XML, aplicaciones de escritorio y aplicaciones móviles. Visual Basic, Visual C# y Visual C++ utilizan todos el mismo entorno de desarrollo integrado (IDE), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes. Asimismo, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones web ASP y Servicios Web XML.

  Se pude descargar en http://msdn.microsoft.com/es-mx/vstudio/bb984878.aspx

- **.Net Framework:** .NET Framework es un entorno de ejecución administrado que proporciona diversos servicios a las aplicaciones en ejecución. Consta de dos componentes principales: Common Language Runtime (CLR), que es el motor de ejecución que controla las aplicaciones en ejecución, y la biblioteca de clases de .NET Framework, que proporciona una biblioteca de código probado y reutilizable al que pueden llamar los desarrolladores desde sus propias aplicaciones. Los servicios que ofrece .NET Framework a las aplicaciones en ejecución son los siguientes:

- Administración de la memoria. En muchos lenguajes de programación, los programadores son responsables de asignar y liberar memoria y de administrar la vida útil de los objetos. En las aplicaciones de .NET Framework, el Common Language Runtime o CLR contiene y proporciona estos servicios en nombre de la aplicación.
- Sistema de tipos comunes. En los lenguajes de programación tradicionales, el compilador define los tipos básicos, lo que complica la interoperabilidad entre lenguajes. En .NET Framework, los tipos básicos los define el sistema de tipos de .NET Framework y son comunes a todos los lenguajes que tienen como destino .NET Framework.
- Biblioteca de clases extensa. En lugar de tener que escribir cantidades extensas de código para controlar operaciones comunes de programación de bajo nivel, los programadores pueden usar una biblioteca de tipos accesible en todo momento y sus miembros desde la biblioteca de clases de .NET Framework.
- Frameworks y tecnologías de desarrollo. .NET Framework incluye bibliotecas para determinadas áreas de desarrollo de aplicaciones, como ASP.NET para aplicaciones web, ADO.NET para el acceso a los datos y Windows Communication Foundation para las aplicaciones orientadas a servicios.
- Interoperabilidad de lenguajes. Los compiladores de lenguajes cuya plataforma de destino es .NET Framework emiten un código intermedio denominado Lenguaje intermedio común (CIL), que, a su vez, se compila en tiempo de ejecución a través de Common Language Runtime. Con esta característica, las rutinas escritas en un lenguaje están accesibles a otros lenguajes, y los programadores pueden centrarse en crear aplicaciones en su lenguaje o lenguajes preferidos.

- Compatibilidad de versiones. Con raras excepciones, las aplicaciones que se desarrollan con una versión determinada de .NET Framework se pueden ejecutar sin modificaciones en una versión posterior.
- Ejecución en paralelo. .NET Framework ayuda a resolver conflictos entre versiones y permite que varias versiones de Common Language Runtime coexistan en el mismo equipo. Esto significa que también pueden coexistir varias versiones de las aplicaciones, y que una aplicación se puede ejecutar en la versión de .NET Framework con la que se compiló.

Se puede descargar en http://msdn.microsoft.com/es-es/library/5a4x27ek(v=vs.110).aspx

- **C# y VB.Net:**
  - C# es un lenguaje de programación que se ha diseñado para compilar diversas aplicaciones que se ejecutan en .NET Framework. C# es simple, eficaz, con seguridad de tipos y orientado a objetos. Las numerosas innovaciones de C# permiten desarrollar aplicaciones rápidamente y mantener la expresividad y elegancia de los lenguajes de estilo de C.
  - Visual Basic .NET se ha diseñado en torno a .NET Framework, que proporciona una mejora en la seguridad, administración de la memoria, control de las versiones y compatibilidad con la implementación. .NET Framework también habilita la interoperabilidad entre los objetos creados con cualquier lenguaje de programación .NET. Esto significa que se pueden crear objetos con Visual Basic .NET que después podrán utilizarse fácilmente en otros lenguajes .NET, y que se pueden utilizar objetos de otros lenguajes .NET de la misma manera en que se utilizan los objetos creados con Visual Basic .NET.

- **LinQ:** Language-Integrated Query (LINQ) es un conjunto de características presentado en Visual Studio 2008 que agrega capacidades de consulta eficaces a la sintaxis de los lenguajes C# y Visual Basic. LINQ incorpora patrones fáciles y estándar para consultar y

actualizar datos, y la tecnología se puede ampliar para proporcionar compatibilidad prácticamente con cualquier tipo de almacén de datos. Visual Studio incluye ensamblados de proveedor LINQ que habilitan el uso de LINQ con colecciones de .NET Framework, bases de datos de SQL Server, conjuntos de datos ADO.NET y documentos XML.

- **EntityFramework 4.0:** Entity Framework es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Los arquitectos y programadores de aplicaciones orientadas a datos se han enfrentado a la necesidad de lograr dos objetivos muy diferentes. Deben modelar las entidades, las relaciones y la lógica de los problemas empresariales que resuelven, y también deben trabajar con los motores de datos que se usan para almacenar y recuperar los datos. Los datos pueden abarcar varios sistemas de almacenamiento, cada uno con sus propios protocolos; incluso las aplicaciones que funcionan con un único sistema de almacenamiento deben equilibrar los requisitos del sistema de almacenamiento con respecto a los requisitos de escribir un código de aplicación eficaz y fácil de mantener.

  Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio, como clientes y direcciones de cliente, sin tener que preocuparse por las tablas y columnas de la base de datos subyacente donde se almacenan estos datos. Con Entity Framework, los desarrolladores pueden trabajar en un nivel mayor de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Dado que Entity Framework es un componente de .NET Framework, las aplicaciones de Entity Framework se pueden ejecutar en cualquier equipo en el que esté instalado .NET Framework a partir de la versión 3.5 SP1.

- **ADO .Net:** ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para programadores de .NET Framework. ADO.NET ofrece abundancia de componentes para la creación de aplicaciones de uso compartido de datos distribuidas. Constituye una parte integral de .NET Framework y proporciona acceso a datos relacionales, XML y de aplicaciones. ADO.NET satisface diversas necesidades de desarrollo, como la creación de

clientes de base de datos front-end y objetos empresariales de nivel medio que utilizan aplicaciones, herramientas, lenguajes o exploradores de Internet.

- **SQL SERVER 2008 R2:** Microsoft® SQL Server™ es un sistema de administración y análisis de bases de datos relacionales de Microsoft para soluciones de comercio electrónico, línea de negocio y almacenamiento de datos. En esta sección, encontrará información sobre varias versiones de SQL Server. También encontrará artículos sobre bases de datos y aplicaciones de diseño de bases de datos así como ejemplos de los usos de SQL Server.

Por otro lado durante la planeación y desarrollo del Módulo de Administración de Pagos me ha permitido adquirir experiencia en la realización de análisis al recabar los requerimientos, así como la adquisición de conocimientos de negocio, y la interacción con los usuarios para conocer sus necesidades, asi como la toma de desiciones para dar una solución.

9. Referencias de Consulta

Visual Studio 2010 http://msdn.microsoft.com/es-es/library/vstudio/fx6bk1f4(v=vs.100).aspx

.Net Framwework http://msdn.microsoft.com/es-es/library/hh425099(v=vs.110).aspx

C# http://msdn.microsoft.com/es-es/library/kx37x362.aspx

VB.Net http://msdn.microsoft.com/es-es/library/cc437084(v=vs.71).aspx

LinQ http://msdn.microsoft.com/es-es/library/bb397926.aspx

Entity Framework http://msdn.microsoft.com/es-es/library/bb399567(v=vs.110).aspx

ADO .Net http://msdn.microsoft.com/es-es/library/e80y5yhx(v=vs.110).aspx

SQL SERVER 2008 r2 http://msdn.microsoft.com/es-es/library/bb545450.aspx

10. Anexos

Anexo A: Manual de Usuario

Como parte complementaria al módulo, se desarrolló un manual de usuario del módulo implementado, que se anexa a continuación, este documento es una copia fiel del original, cambiando en este las referencias a las figuras, así como la secuencia numérica del documento.

1. Acceso al Sistema BeMoney

El sistema de BeMoney es una aplicación de escritorio que se instala como terminal en el equipo del usuario, este a su vez se conecta a una base de datos ubicada en un servidor ubicado en Portafolio de Negocios S.A. de C.V. SOFOM ENR.

Al hacer clic en el icono del acceso directo de la aplicación de se presenta la siguiente pantalla como se muestra en la figura 1.

Teclear su Nombre de Usuario y Contraseña y oprimir Aceptar.

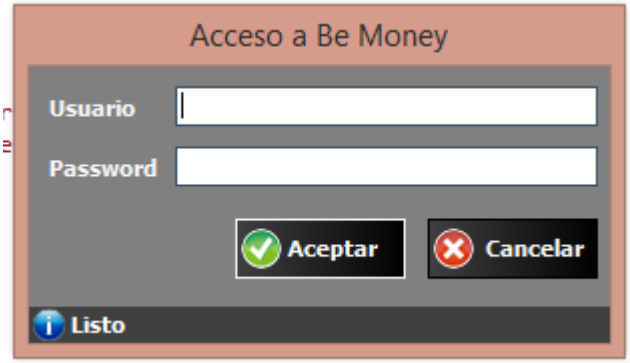

Figura 1

Una vez que el sistema verifique la información proporcionada por el usuario al sistema, se mostrara la vista principal con los menús autorizados para el usuario con el rol de Administrador tiene acceso a todos los menús del sistema cómo se puede observar en la figura 2.

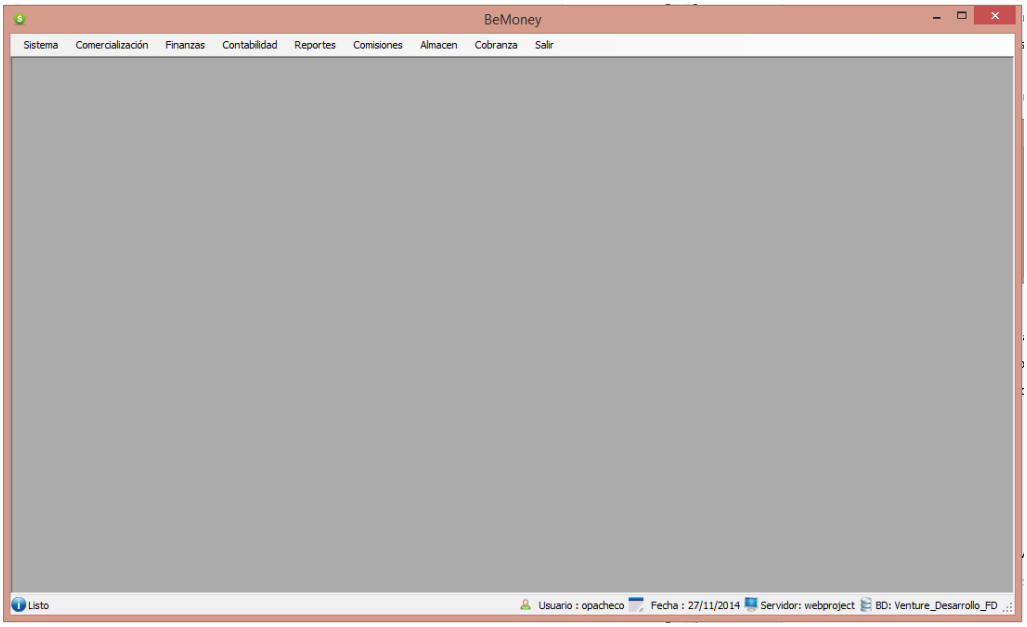

Figura 2

1. Acceso al Módulo de Administración de Pagos

Al acceder al sistema el usuario debe hacer clic en el Menú Finanzas/Administración de Pagos para que pueda acceder al área de trabajo del Módulo de Administración de Pagos, también se puede utilizar las teclas rápidas Ctrl+ O como se muestra en la Figura 3.

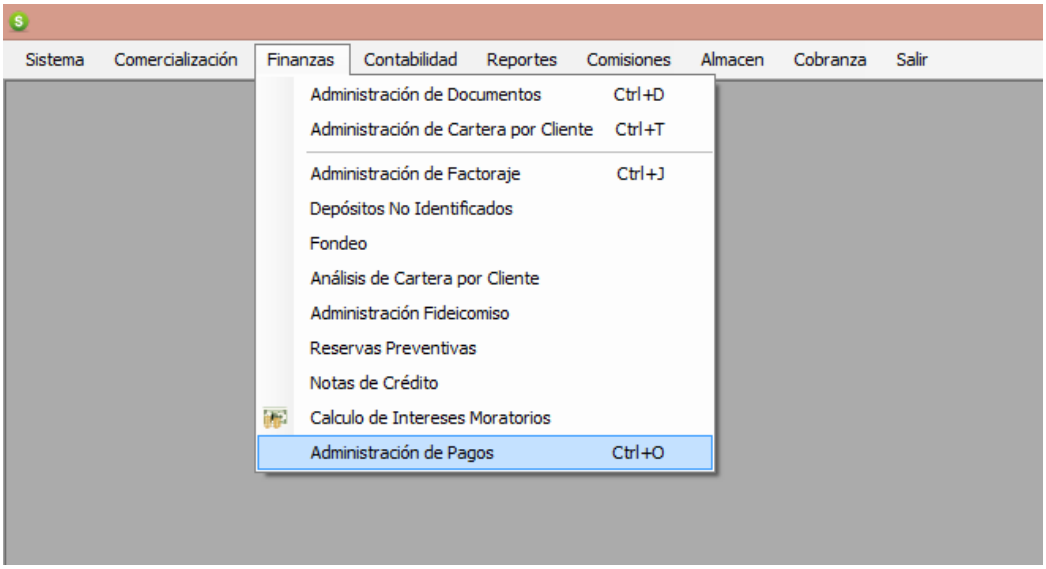

Figura 3.

Una vez cargada la pantalla de trabajo del Módulo de Administración de Pagos donde se carga por defecto el área de captura de las fichas de depósito con sus opciones como se muestra en la figura 4.

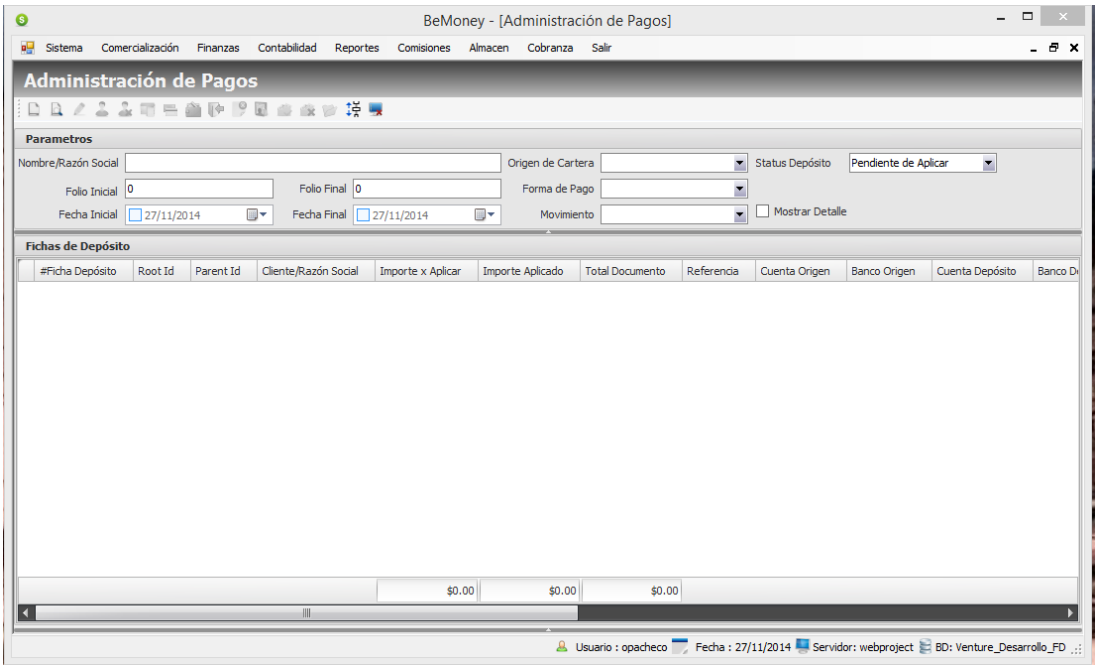

Figura 4.

Barra de Herramientas Fichas de Depósito Figura 5

Figura 5.

La barra de herramientas contiene las siguientes opciones:

- Nuevo
- Buscar
- Editar
- Asignar Cliente
- Desasignar Cliente
- Ficha Parcial
- Traspaso Interbancario

- Transferencia entre Cuentas
- Devolución
- Cancelar
- Cargar Contratos
- Cargar Comprobante
- Eliminar Comprobante
- Mostrar Comprobante
- Ocultar /Mostrar Parámetros
- Salir

La pantalla principal también cuenta con una sección de parámetros de búsqueda de las fichas de depósito como se muestran en la figura 6.

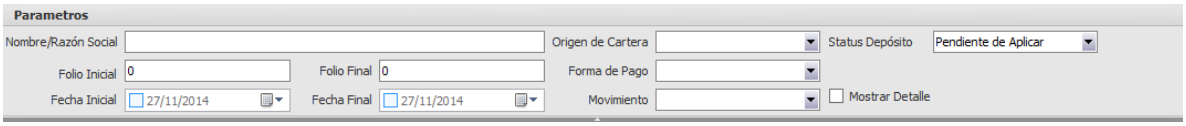

Figura 6

A continuación se describen brevemente los campos de la sección de parámetros:

Nombre Cliente/Razón Social: Indica el nombre del cliente para realizar la búsqueda de fichas de depósito.

Origen de Cartera: Filtro que permite identificar la clasificación de las fichas de depósito, tiene una relación con los contrato/créditos del cliente, además de la clasificación de las cuentas bancarias.

Status de Deposito: Permite filtrar las fichas de depósito de acuerdo a su estatus, es decir si están, pendientes de aplicar, canceladas o devueltas.

Folio Inicial y Folio Final: Permite indicar un rango de búsqueda de folios de fichas de depósito.

Fecha Inicial y Fecha Final: Permite filtrar las fichas de acuerdo a la fecha en la que el cliente realizo el depósito o pago.

Movimiento: Permite Filtrar por tipo de Movimiento de la ficha de depósito.

Mostrar Detalle: Esta opción permite habilitar la funcionalidad para que por cada registro se muestre el detalle de los movimientos que se realizaron hasta llegar a su estado actual.

A continuación se describen cada una de las opciones de la barra de herramientas:

Botón Nuevo 

Por medio de este botón accedemos a la funcionalidad para poder dar de alta una nueva ficha de depósito y carga la pantalla de alta como se muestra en la figura 7.

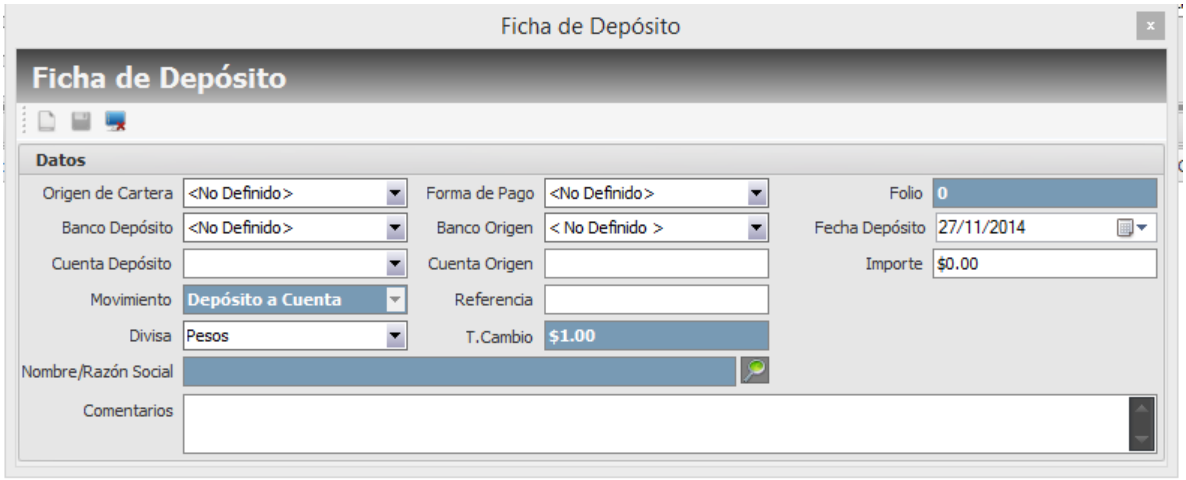

Figura 7

En la Figura 6 se puede observar los campos que el sistema requiere para dar de alta una ficha de depósito. A continuación se describen brevemente.

Origen de Cartera: En este campo se escogerá el tipo de cartera al cual se cargara ficha de depósito, este campo depende las cuentas bancarias que se utilizaran.
Banco Depósito: En este campo se selecciona el banco en el que se depositó el pago por parte del cliente.

Cuenta Depósito: En este campo se selecciona el banco donde se registró el pago por parte del cliente.

Movimiento: Este campo indica el tipo de movimiento que se está dando de alta, por defecto se dan de alta las fichas de depósito como Depósito a Cuenta.

Divisa: Indica el tipo de Moneda con el que se realizó el pago por parte del cliente, por defecto se carga Pesos.

Forma de Pago: Este campo indica la forma en la que se realizó el pago por parte del cliente.

Banco Origen: Este campo indica el banco es la procedencia del pago realizado por parte del cliente, esta opción no aplica cuando el pago fue realizado a través de un depósito en efectivo o pago en efectivo.

Cuenta Origen: Este campo indica la cuanta de procedencia del pago realizado por el cliente, al igual que el Banco origen este campo no aplica cuando el pago fue realizado por un depósito en efectivo o pago en efectivo.

Referencia: Este campo se utiliza para dejar una marca para identificar el pago por parte del usuario, normalmente es un dato que se encuentra en el comprobante de pago del cliente, un consecutivo o dato alfanumérico asignado por el usuario, o número de cheque.

Tipo de Cambio: este campo indica el tipo de cambio que se utiliza para efectuar el pago de acuerdo a la selección del campo Divisa, por defecto carga la unidad del peso que es la divisa por defecto.

Folio: Este campo indica el consecutivo de la ficha de depósito, por defecto se coloca cero indicando que se está dando de alta o se está realizando un movimiento al hacer clic en el botón guardar este se actualiza e indica el consecutivo asignado por el sistema.

Fecha de Depósito: Este campo indica la fecha en la que se realizó el depósito o pago por parte del cliente, por default se carga la fecha actual, pero es modificable por si no se registró en la fecha correspondiente.

Importe: En este campo se captura el importe del depósito o pago que realizo el cliente.

Nombre Cliente/Razón Social: En este campo indica el cliente que estará relacionado con la ficha de depósito, este campo es opcional, y se pueda captura posteriormente. A continuación se muestra como se asigna un cliente al dar de alta la ficha de depósito.

Paso 1: Hacer clic en el botón contiguo al campo de Nombre Cliente/Razón Social  el cual desplegara la pantalla de búsqueda de clientes como se muestra en la figura 8.

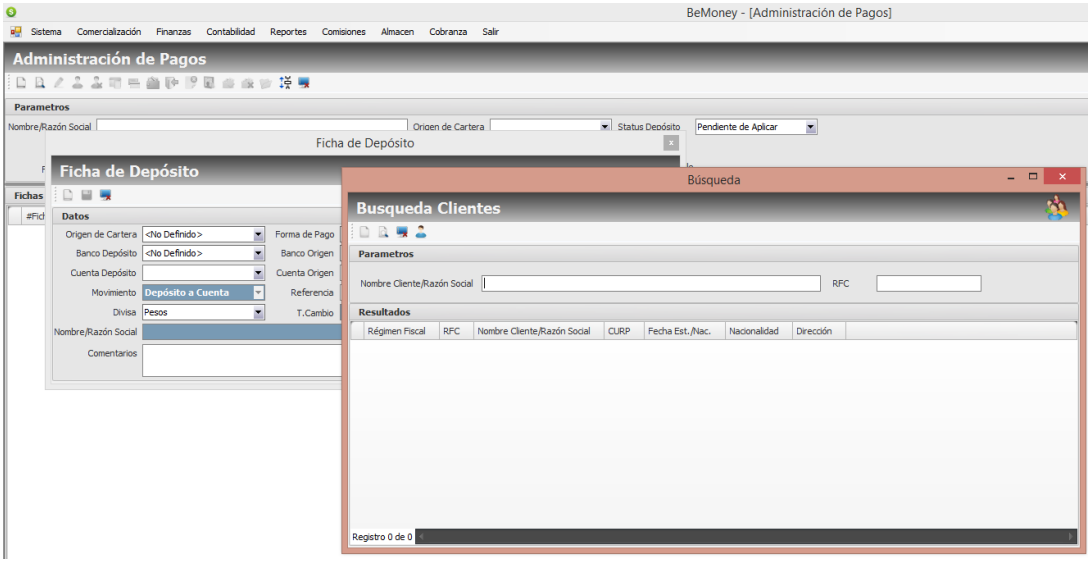

Figura 8.

Paso 2: Al escribir el nombre del cliente al que se le asignara la ficha de depósito, hacer clic en el botón buscar, el cual desplegara las coincidencias como se muestra en la figura 9.

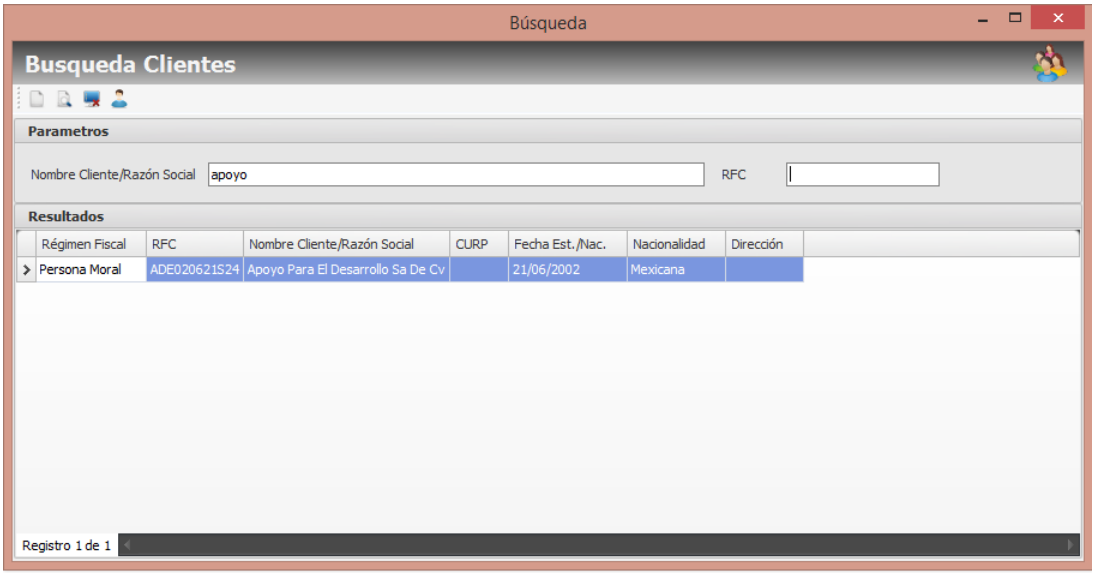

Figura 9.

Paso 3. Una vez cargados los resultados, se puede dar clic en cliente deseado y posteriormente hacer clic en el botón asignar, el cual mostrar un mensaje de

confirmación, aceptamos la confirmación y posteriormente se mostrara el cliente asignado a la ficha de depósito como se muestra en la figura 10.

Figura 10.

Comentarios: En este campo el usuario puede colocar algún comentario que pueda servir posteriormente para la administración de la ficha de depósito.

Al realizar clic en el boto guardar se almacena la ficha de depósito y actualiza el folio, la pantalla queda activa por si desea seguir realizando capturas de fichas de depósito, de lo contrario se cierra la pantalla haciendo clic en el botón Salir. La pantalla principal carga las ficha de depósito pendientes de aplicar de acuerdo a los filtros que se cargan por defecto en la pantalla principal, esta búsqueda también se puede realizar haciendo clic en el botón buscar de la pantalla de Administración de Pagos como se muestra en la figura 11.

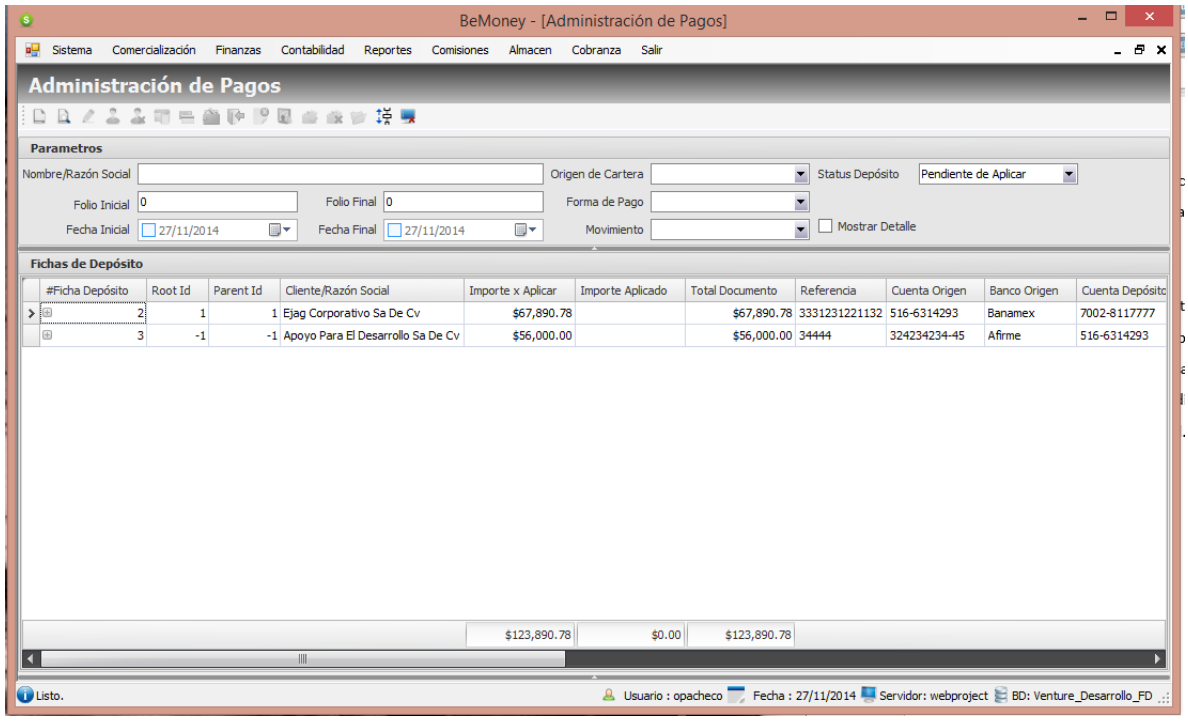

Figura 11

Botón Editar

La funcionalidad de este botón es poder editar una ficha de depósito que no tiene movimientos y se utiliza para modificar los datos de la ficha de depósito a excepción del folio como se muestra en la figura 12.

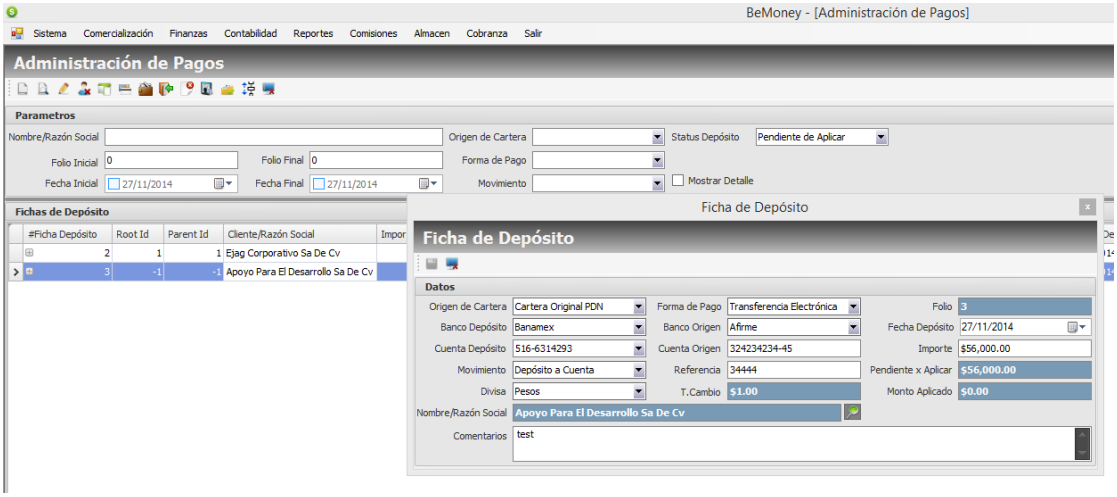

Figura 12

Botón Asignar Cliente

La funcionalidad de este botón es idéntica a la asignación de cliente en la pantalla de alta de la ficha de depósito pero se realiza directamente desde el listado de resultados de fichas de depósito como se muestra en la figura 13.

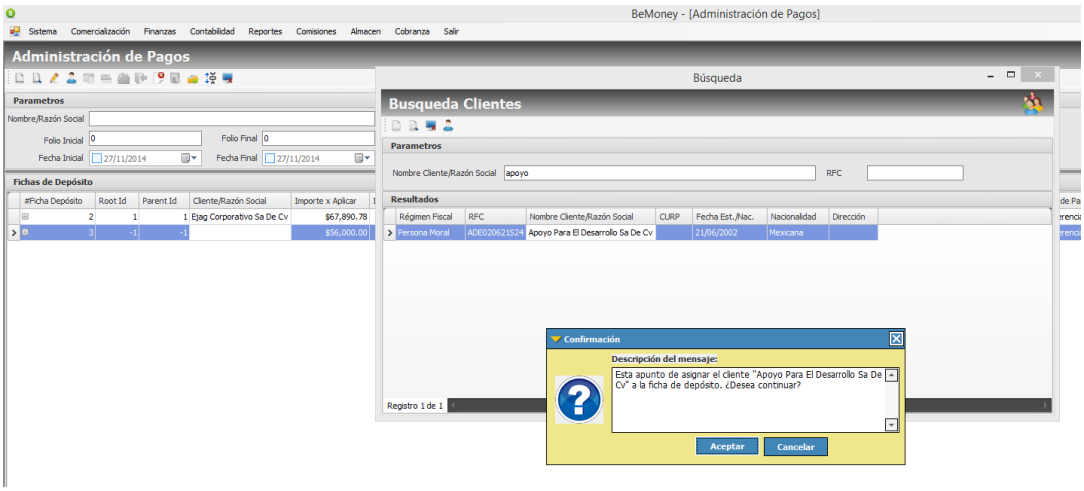

Figura 13

Botón Desasignar Cliente

Su funcionalidad permite desasignar el cliente de la ficha de depósito, pero solo puede realizarse en fichas de depósito que no tienen o hayan tenido movimientos, esta presenta un mensaje de confirmación y al aceptar la des asignación deja la ficha de depósito sin un cliente como se puede ver en la figura 14.

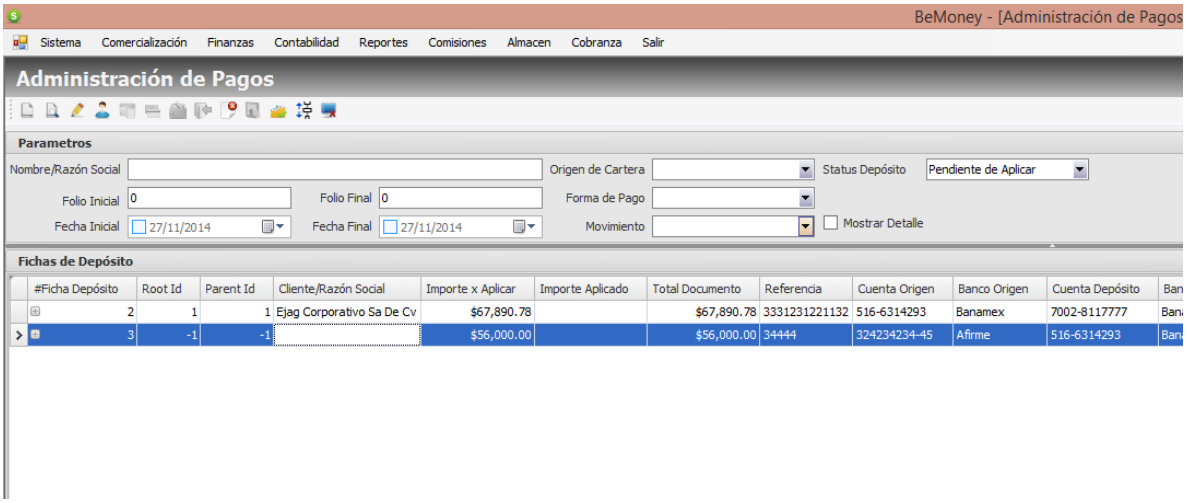

Figura 14

Botón Ficha Parcial

La funcionalidad de este botón permite mostrar una pantalla en la que se puede realizar un movimiento denominado ficha parcial el cual hace referencia al documento origen el cual es otra ficha de depósito, este toma los datos de destino del de la ficha de depósito origen y solo permite la edición del Importe, Fecha de Depósito y Referencia así como un comentario, en la figura 15 se puede observar que el movimiento queda como transferencia entre cuentas.

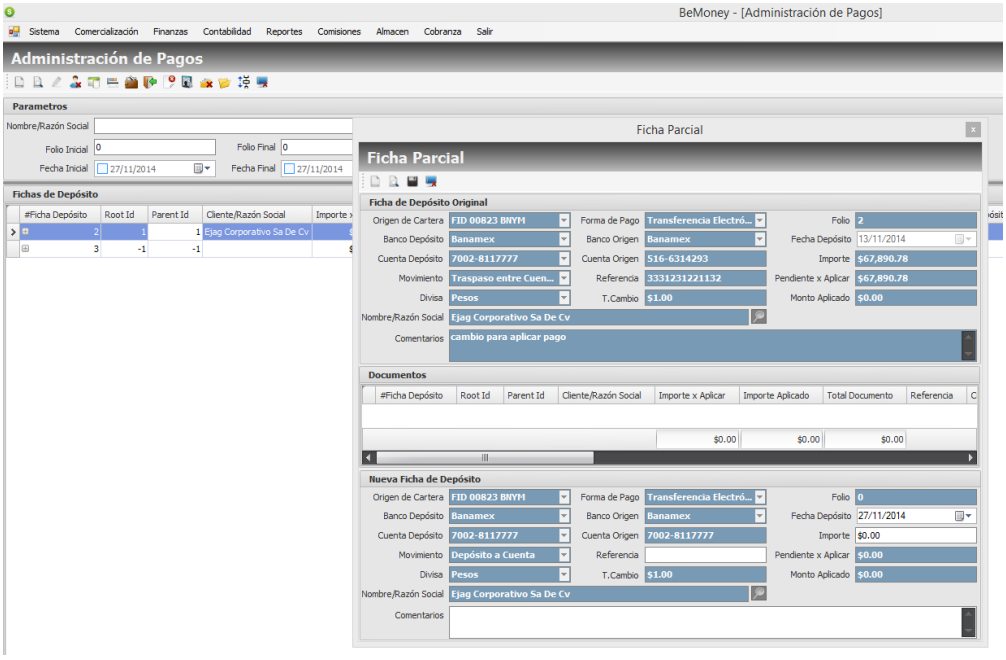

Figura 15.

Botón Traspaso Interbancario

La funcionalidad de este botón permite cargar una pantalla la cual está configurada para realizar un movimiento denominado Traspaso Interbancario, el cual permite crear una ficha de depósito a un banco distinto además de poder elegir el tipo de cuenta destino de acuerdo a la configuración del origen de cartera, los campos editables como se muestran en la figura 16 son los siguientes:

- Origen de Cartera
- Banco Depósito
- Cuenta Depósito
- Referencia
- Fecha Depósito
- Importe
- Comentarios

El traspaso interbancario toma los datos destino de la ficha de depósito origen y los coloca como datos origen de la nueva ficha de depósito, como se puede ver en la figura 16 el Movimiento queda como Traspaso Interbancario.

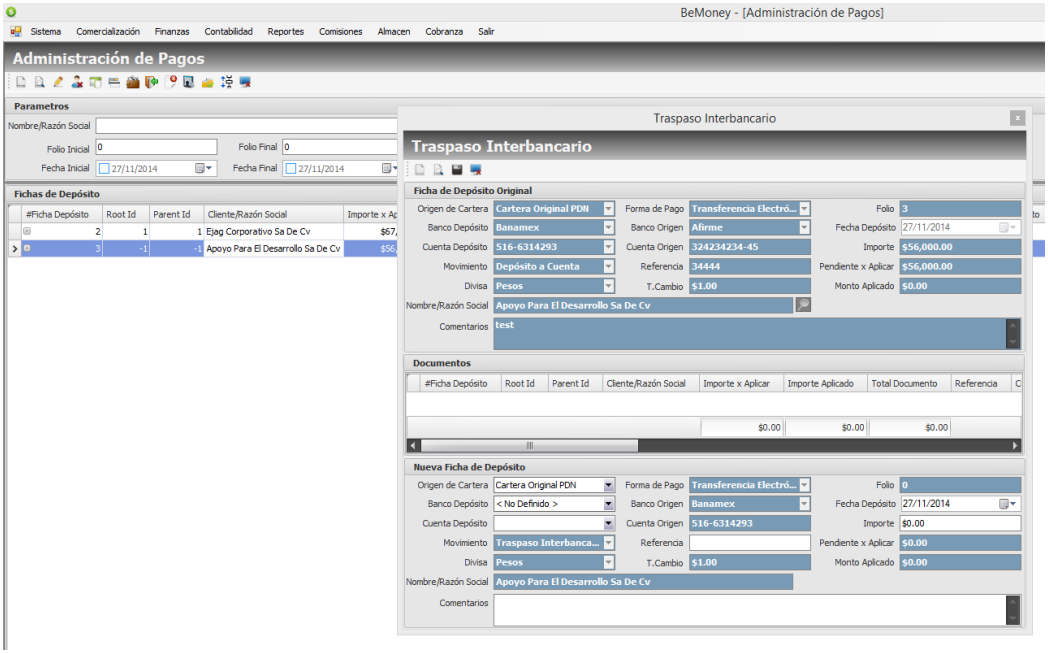

Figura 16.

Boton Transferencia entre Cuentas

La funcionalidad de este botón permite cargar una pantalla la cual está configurada para realizar un movimiento denominado Trasferencia entre Cuentas, el cual permite crear una ficha de depósito a una cuenta diferente a la de su origen pero del mismo banco y la configuración del origen de cartera, los campos editables como se muestran en la figura 17 son los siguientes:

- Origen de Cartera

- Cuenta Destino
- Referencia
- Fecha Depósito
- Importe
- Comentarios

La transferencia entre cuentas toma los datos destino de la ficha de depósito origen y los coloca como datos origen de la nueva ficha de depósito, como se puede ver en la figura 16 el Movimiento queda como Trasferencia entre cuentas.

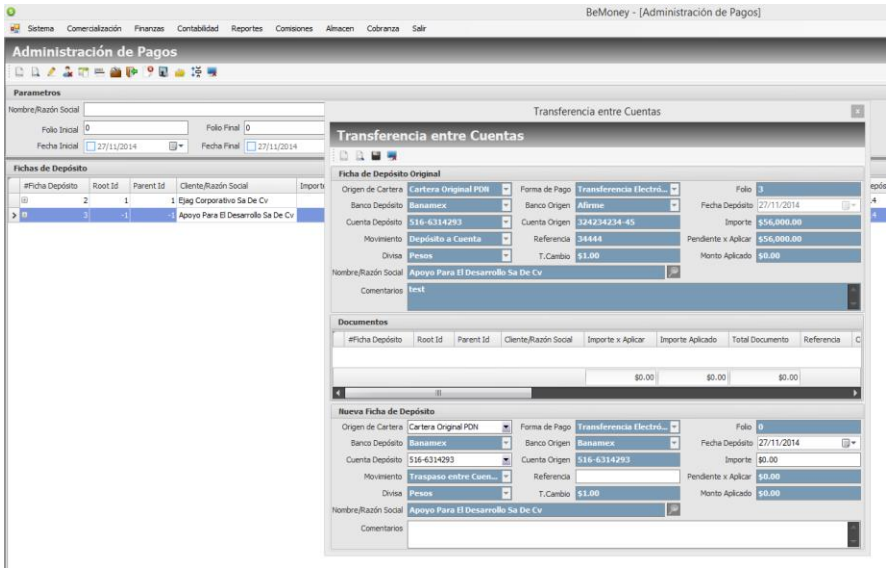

Figura 17

Botón Devolución

La funcionalidad de este botón permite al usuario marcar como devuelta una ficha de depósito, para realizar esta acción debe colocarse una descripción del motivo por el cual se marcara como devuelta como se muestra en la figura 18.

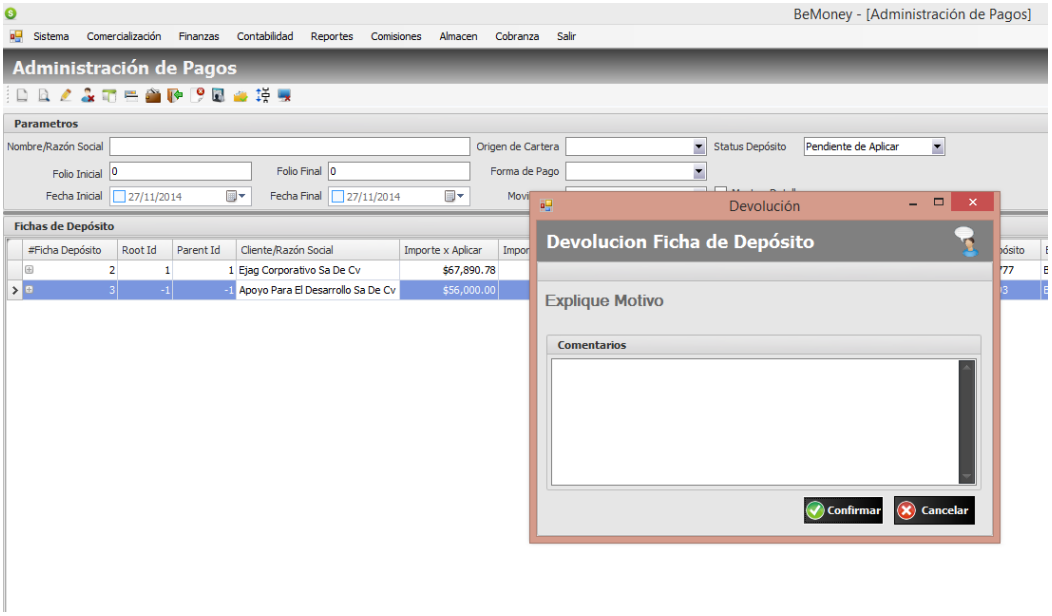

Figura 18.

Botón Cancelar

La función de este botón permite al usuario como su nombre lo indica cancelar una ficha de depósito, a diferencia de la devolución no se requiere de escribir un motivo, ya que estos pueden ser por que los datos no son correctos y ya tuvo movimientos los cuales no permiten la edición de estos, en la figura 19 se puede apreciar un mensaje de confirmación para la cancelación de una ficha de depósito.

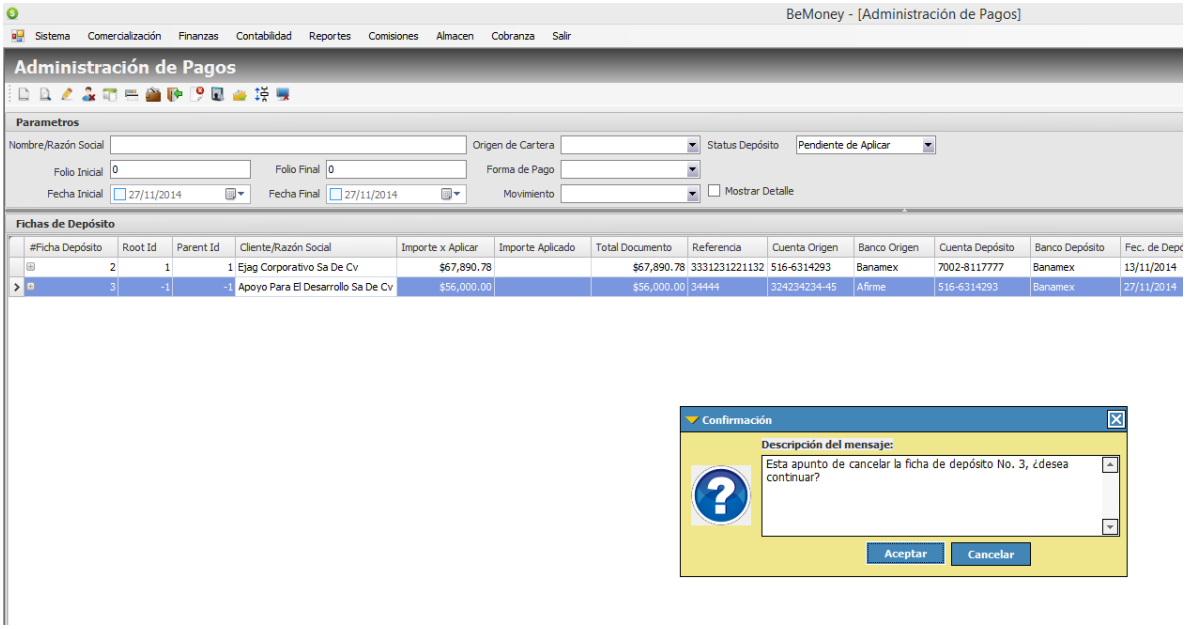

Figura 19.

Botón Cargar Contratos

La funcionalidad de este botón permite cargar los contratos del cliente asignado a la ficha de depósito, con el fin de ser aplicada como pago de alguno de sus documentos con estatus pendiente pago.

En la Figura 20 se puede apreciar la carga de dos secciones adicionales de trabajo:

- Contratos: Permite visualizar los contratos activos del cliente que está asignado a la ficha de depósito.
- Documentos: esta sección consta de 4 pestañas en las cuales carga:
    - Documentos Pendientes de Pago: son aquellos que como su nombre lo indica tiene un estatus de pago pendiente, el color amarillo indica que es un documento vencido.
    - Pagos: enlista todos los pagos que el cliente realizado al contrato o crédito

- Tabla de Amortización: muestra el detalle de las amortizaciones o periodos de pago del crédito, los documentos pendientes de pago están ligados a estas, así como como los pagos están ligados a un documento.
- Facturas: Muestra el detalle de lo facturado y pendiente por facturar, cada factura está ligada a un periodo o amortización del crédito.

Los Documentos se cargan cuando se hace clic en alguno de los contratos o créditos del cliente, en la figura 20 se puede ver un ejemplo cargado.

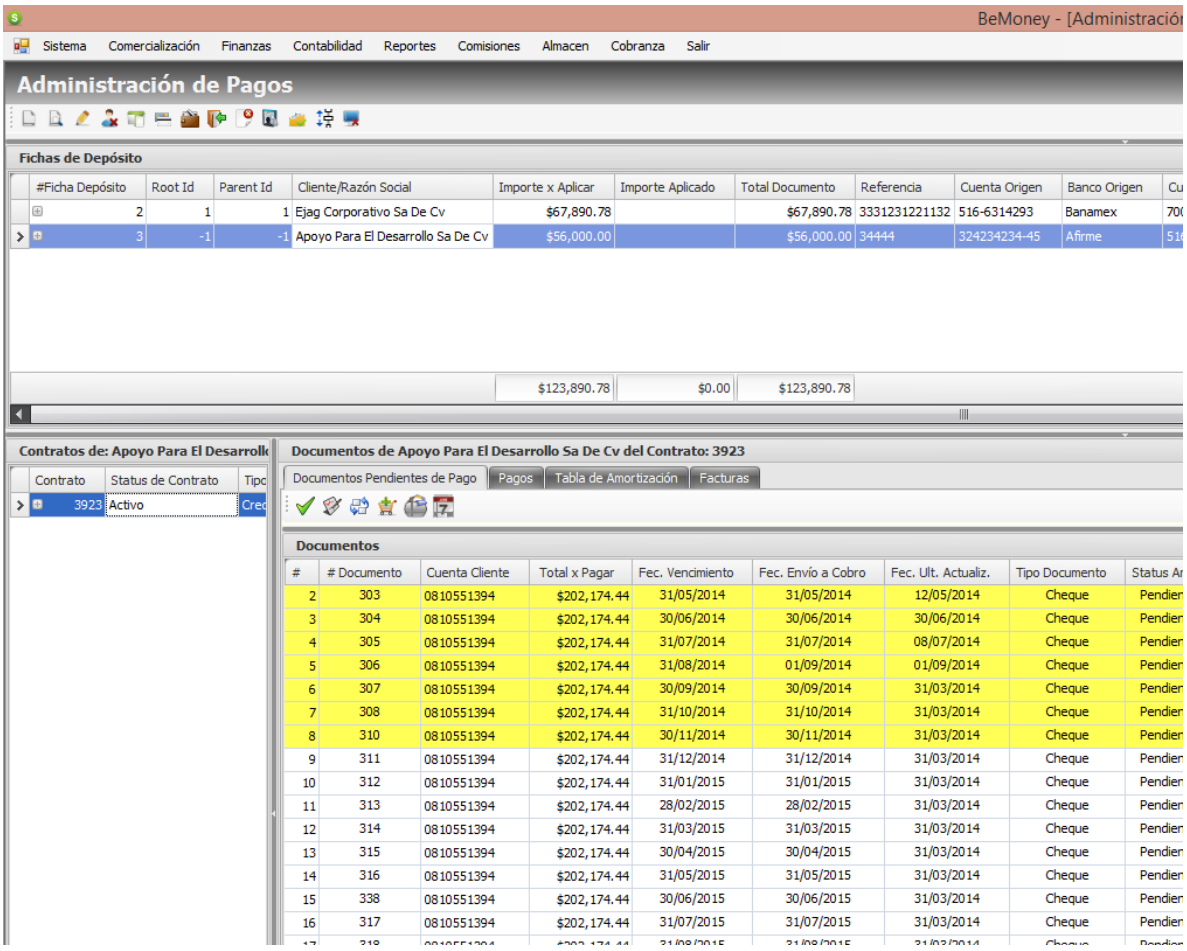

Figura 20.

Barra de Herramientas Documentos Pendientes de Pago 

En los documentos pendientes de pago se cuenta con una barra de herramientas la cual permite ejecutar las diferentes formas de pago:

- Pago Directo/Aplicar Pago
- Pago Parcial
- Remplazo de Documento
- Agregar Cargo
- Agregar Excedente
- Reprogramar Pago

Pago Directo

La funcionalidad del botón Pago Directo o Aplicar Pago carga una pantalla que carga el detalle del documento a pagar, cabe mencionar que esta opción solo aplica para pagar un cheque, por lo que para que se pueda utilizar una ficha de depósito los datos deben ser idénticos a los del documento, en la Figura 21 se puede ver un ejemplo del detalle del documento y como se muestra solo son editables los campos:

- Fecha Envió Cobro
- Fecha Pago
- Comentarios

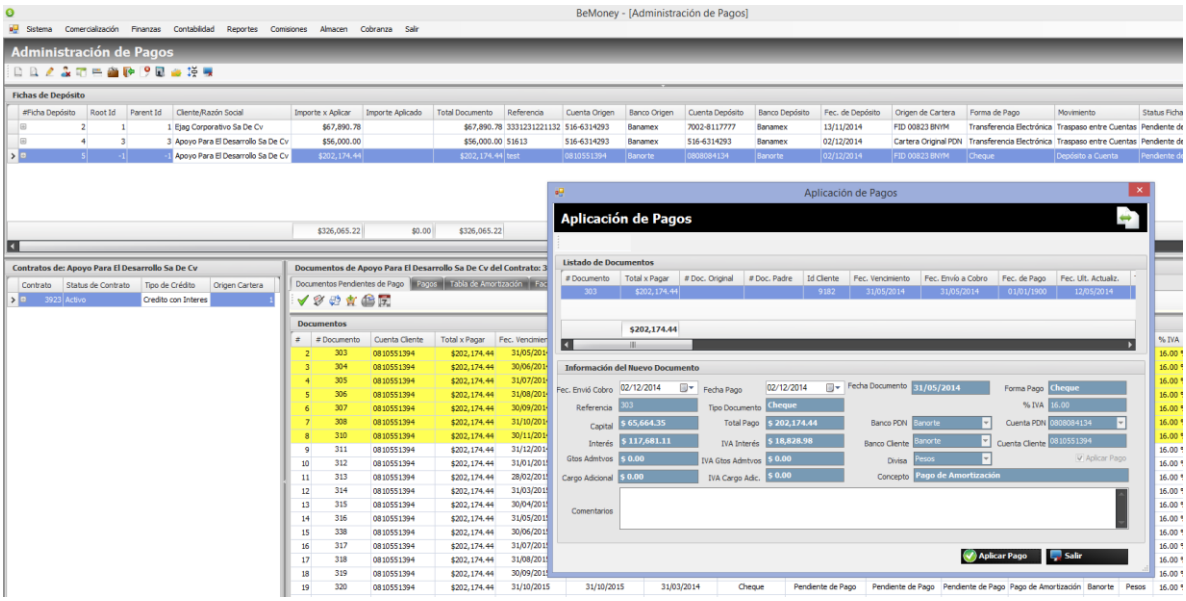

Figura 21.

Pago Parcial

La funcionalidad de este botón carga una pantalla en la cual presenta una sección en la que presenta la información del documento original, una sección con los documentos hijos, y una sección con la información del documento nuevo el cual se utilizará para aplicar un pago, en la Figura 22, se muestra que los únicos datos de edición son No. Documento, Fecha de Envió, Concepto y Comentario, en los demás datos se puede ver que trae información que proviene de la ficha de depósito que fue seleccionada para aplicar el pago parcial.

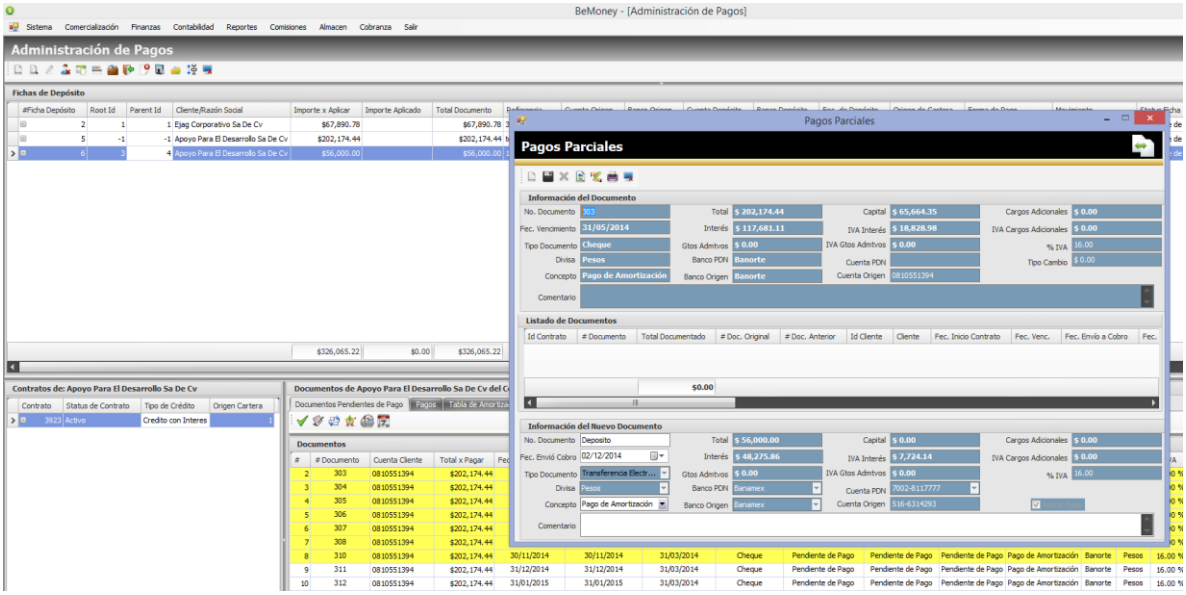

Figura 22.

Remplazo de Documento

La funcionalidad de este botón carga una pantalla en la que al igual que la de Pago Parcial muestra tres secciones, y los datos editables son No. Documento, Fec. Envió a Cobro, Concepto y Comentario, como se puede ver en la figura 23 el total del nuevo documento es el mismo al del documento original distribuido con su detalle de las misma manera, también podemos ver que los datos de tipo de documento, banco origen y cuenta origen son datos obtenidos de la ficha de depósito seleccionada para realizar el remplazo de documento.

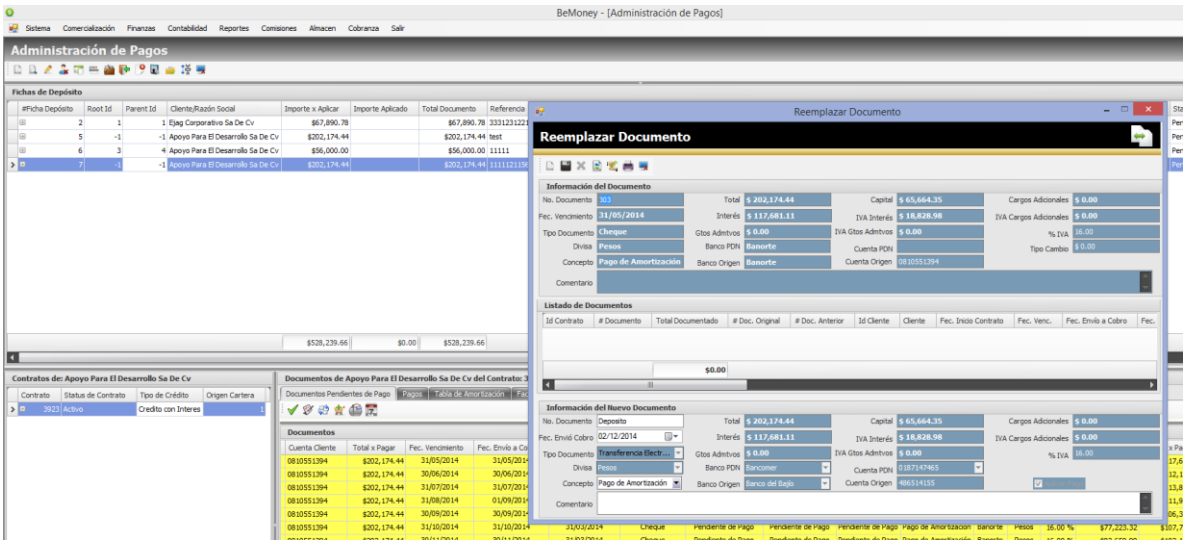

Figura 23.

Agregar Cargo

La funcionalidad de este botón carga una pantalla con tres secciones al igual que pago parcial y remplazo de documento, la diferencia es que el detalle del documento es que la distribución se realiza en los campos de Cargos Adiciones e IVA de Cargos Adicionales, los campos editables son No. Documento, Fec. Envió Cobro, el concepto puede ser seleccionado como se muestra en la figura 24, y Comentario.

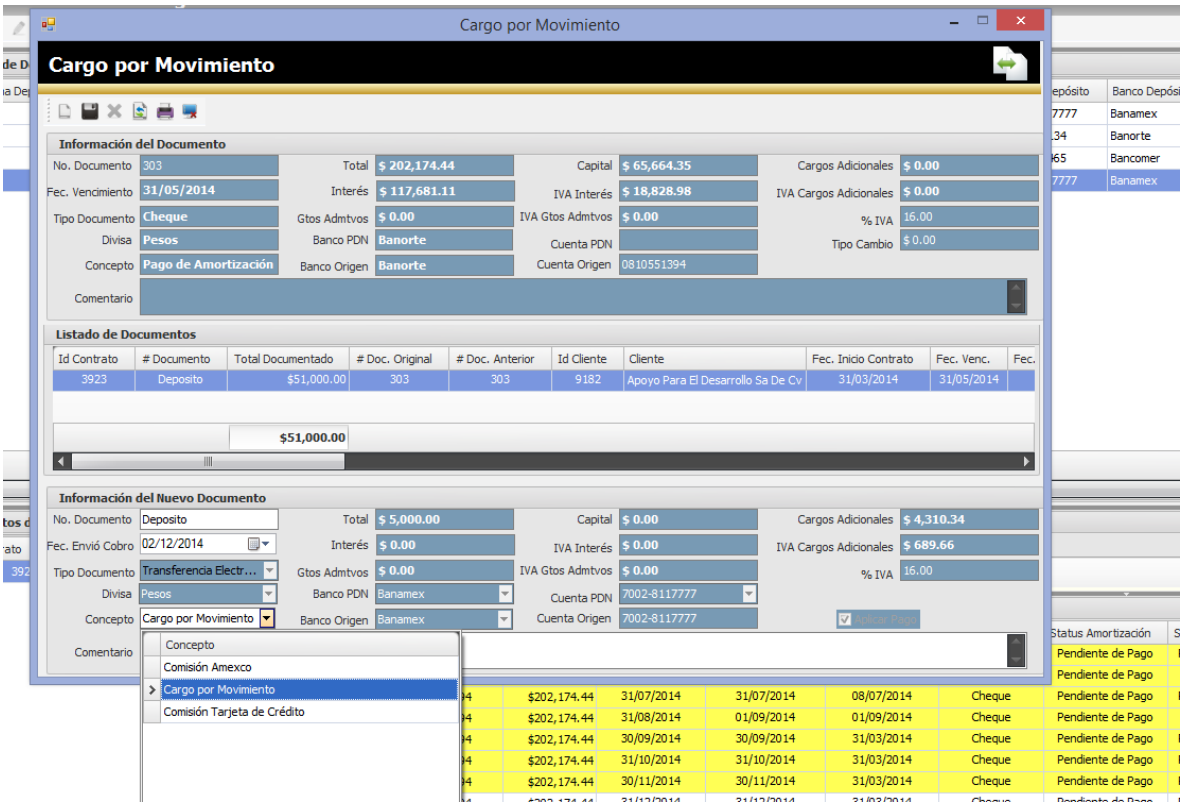

Figura 24.

Agregar Excedente

La funcionalidad de este botón permite cargar una pantalla en la que permite aplicar un pago con excedente, este caso ocurre cuando el importe de la ficha de depósito es mayor al monto del documento seleccionado como se muestra en la figura 25.

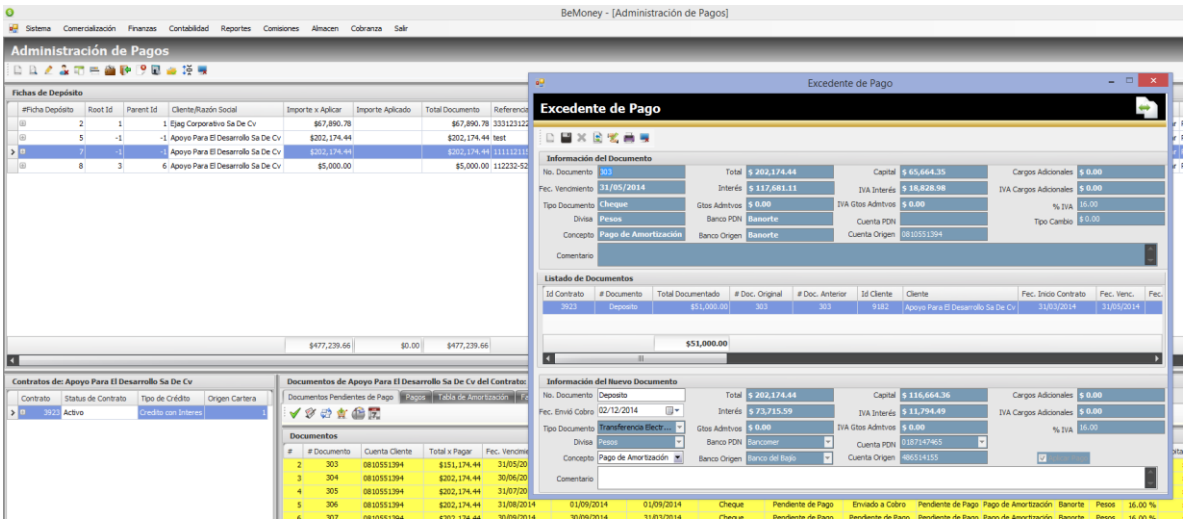

Figura 25

Reprogramar Pago

La funcionalidad de este botón carga una pantalla que permite como su nombre lo indica reprogramar el pago de un documento, es decir cambiar la fecha que será enviado a cobro, esta funcionalidad se utiliza para evitar que el documento se marque con un estatus de documento vencido. El reprogramar un pago no necesita de una ficha de depósito puesto que no se está aplicando el pago, pero, se hace mención ya que es parte de la funcionalidad de la administración de los pagos de los documentos.

En la figura 26 se muestra la pantalla de trabajo en la que solo se tiene editable los campos Nueva Fec. De Envió y Comentarios.

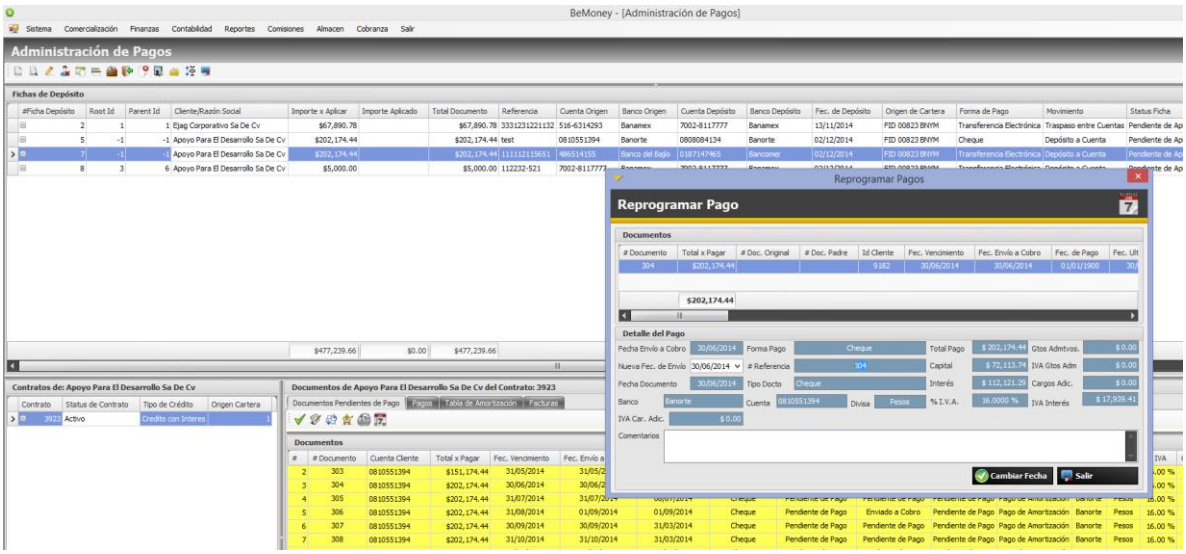

Figura 26

Pestaña de Pagos

En la pestaña de Pagos se tiene el listado de todos los pagos aplicados a los documentos de cada periodo de un Contrato/Crédito, en esta se puede ver el detalle de cómo fue aplicado como se muestra en la figura 27, cuenta con los siguientes campos que a continuación se describen brevemente:

- #
- # Documento

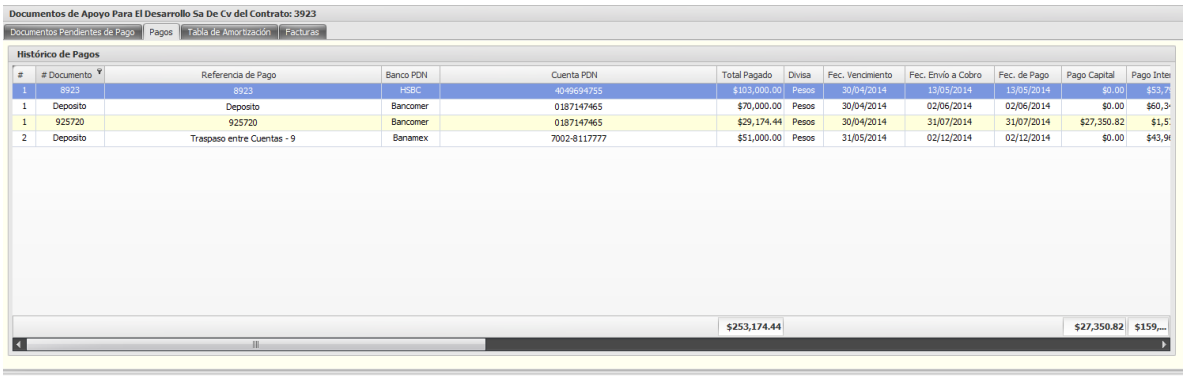

Figura 27

Si uno se posiciona en un registro automáticamente se carga el detalle del pago con referencia a la ficha de depósito que se utilizó para aplicar el pago, esto es útil para saber el flujo que se siguió para la aplicación de este como se muestra en la figura 28.

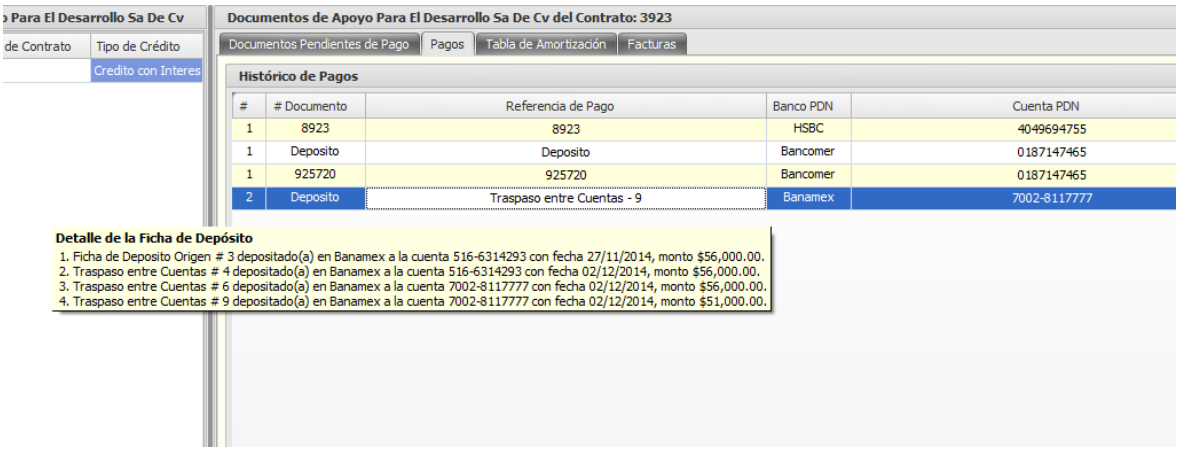

Figura 28

La pestaña de pagos también tiene funcionalidad adicional la que permite realizar:

- Reversa de Pago.
- Comentarios.

Esta se despliega al hacer clic derecho en algunos de los registros como se muestra en la figura 29.

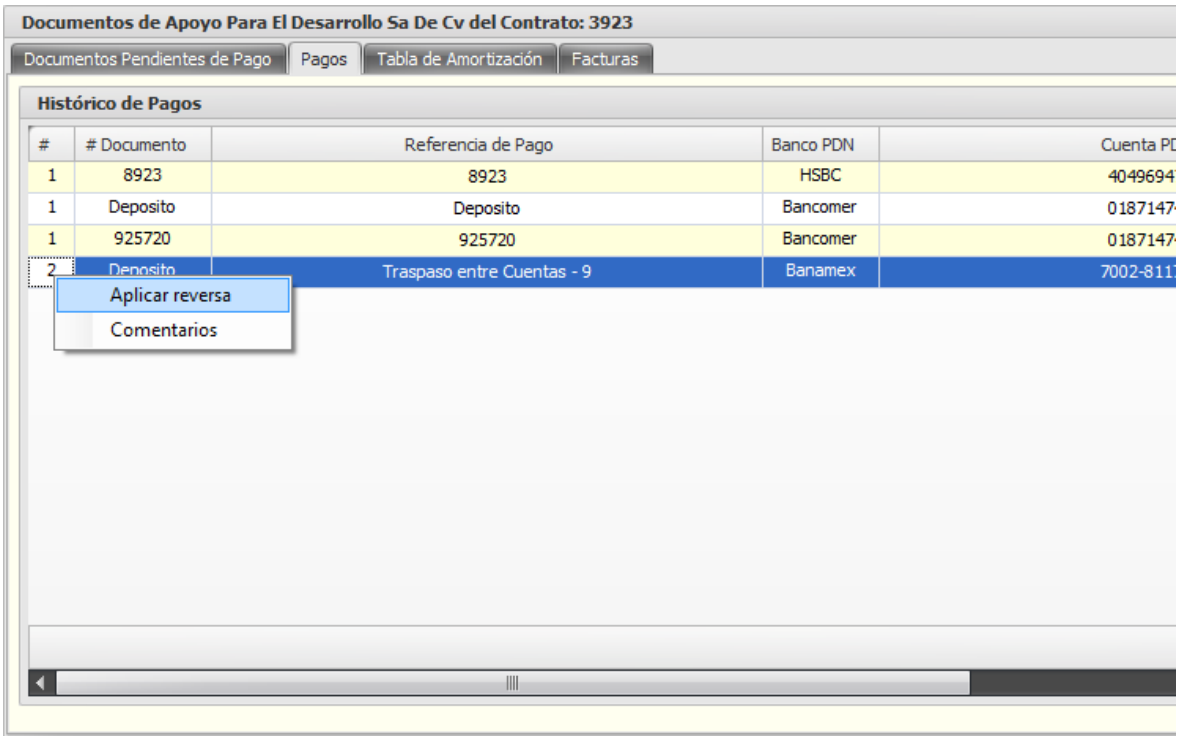

Figura 29

Reversa de Pago

Al seleccionar la opción Reversa de pago como se mostró en la figura 29 se carga la pantalla de captura de confirmación para la reversa de pago como se muestra en la figura 30, en la que se solicita un comentario o motivo por el cual se realizara la reversa del pago, una vez realiza la reversa del pago, el registro desaparece y el documento de se puede visualizar como documento pendiente de pago en la pestaña de Documentos Pendientes de Pago.

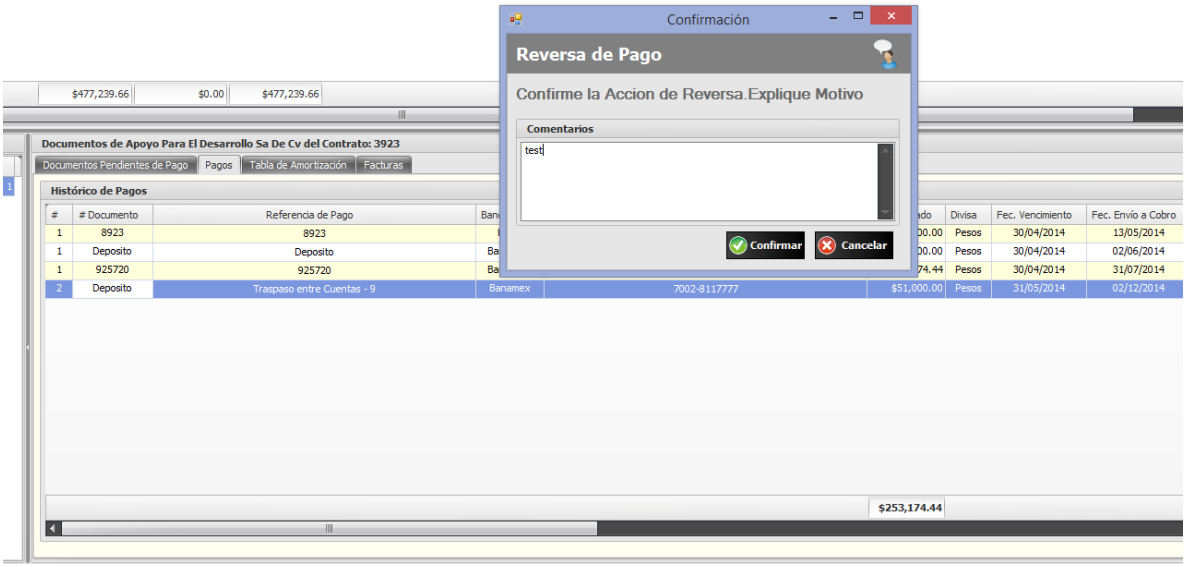
Confirmación
Reversa de Pago
Confirme la Accion de Reversa.Explique Motivo
Comentarios
test
Confirmar
Cancelar
$477,239.66
$0.00
$477,239.66
Documentos de Apoyo Para El Desarrollo Sa De Cv del Contrato: 3923
Documentos Pendientes de Pago
Pagos
Tabla de Amortización
Facturas
Histórico de Pagos
Documento
Referencia de Pago
Divisa
Fec. Vencimiento
Fec. Envío a Cobro
1 8923 8923 Pesos 30/04/2014 13/05/2014
1 Deposito Deposito Pesos 30/04/2014 02/06/2014
1 925720 925720 Pesos 30/04/2014 31/07/2014
2 Deposito Traspaso entre Cuentas - 9 Banamex 7002-8117777 $51,000.00 Pesos 31/05/2014 02/12/2014
$253,174.44

Figura 30

Anexo B Diagramas UML

Estructura Administración de Pagos (Clases) del Módulo

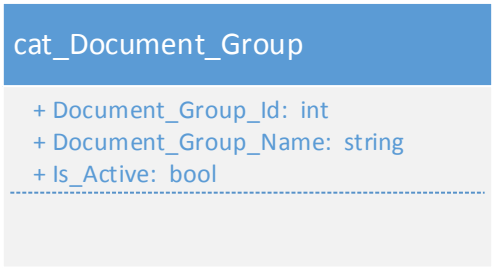

Figura 1. Clase de clasificación de documentos

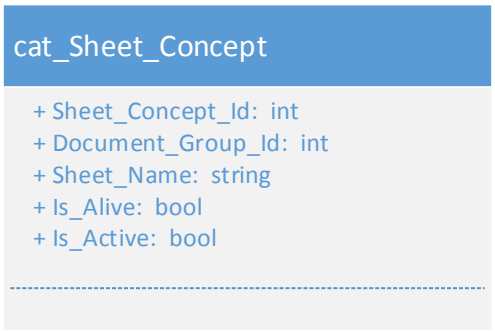

Figura 2. Clase de Movimientos de Ficha de Depósito

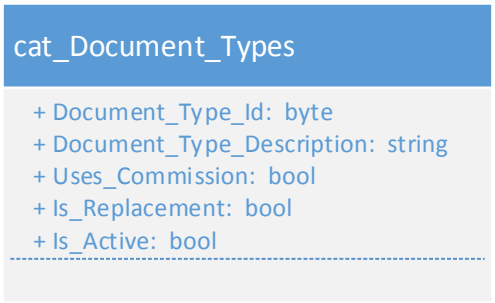

Figura 3. Clase de formas de pago

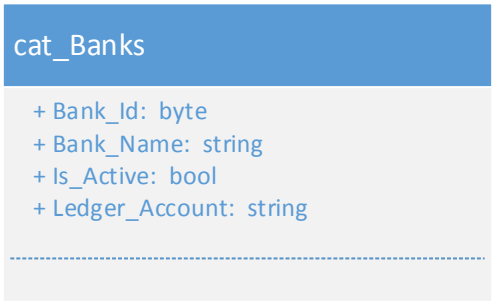

Figura 4. Clase de Bancos

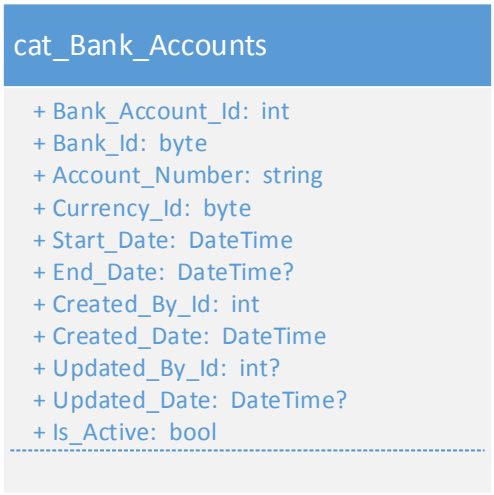

Figura 5. Clase de Cuentas Bancarias

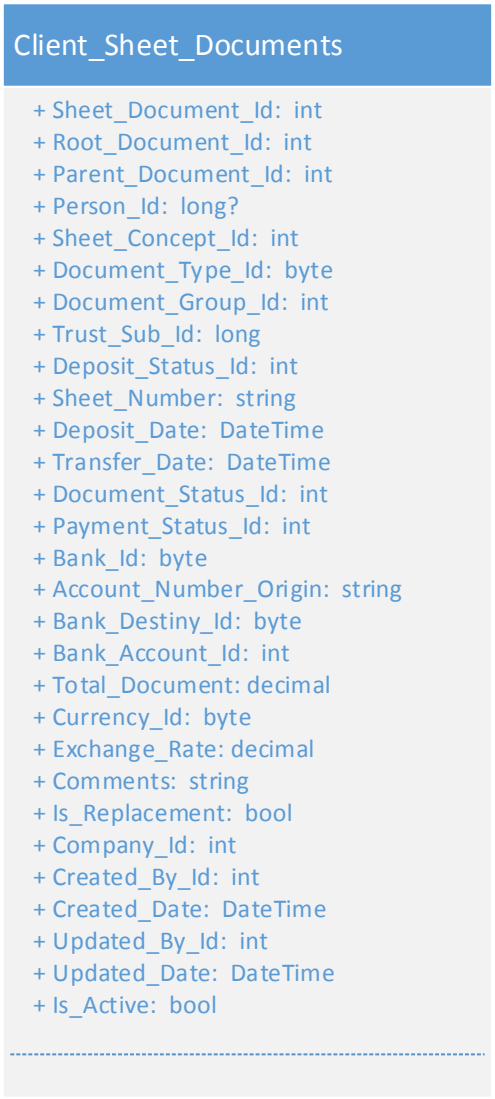

Figura 6. Clase Fichas de Depósito

Figura 7. Clase Pagos

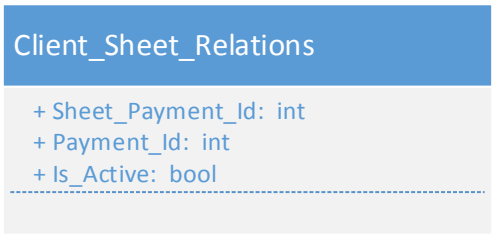

Figura 8. Clase relación Ficha de Depósito-Pagos

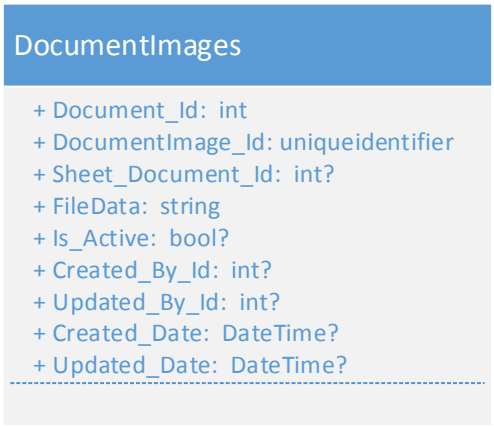

Figura 9. Clase Comprobantes

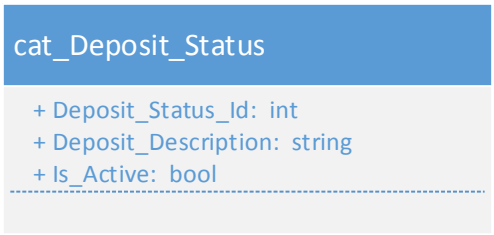

Figura 10 Clase Estatus de Depósito

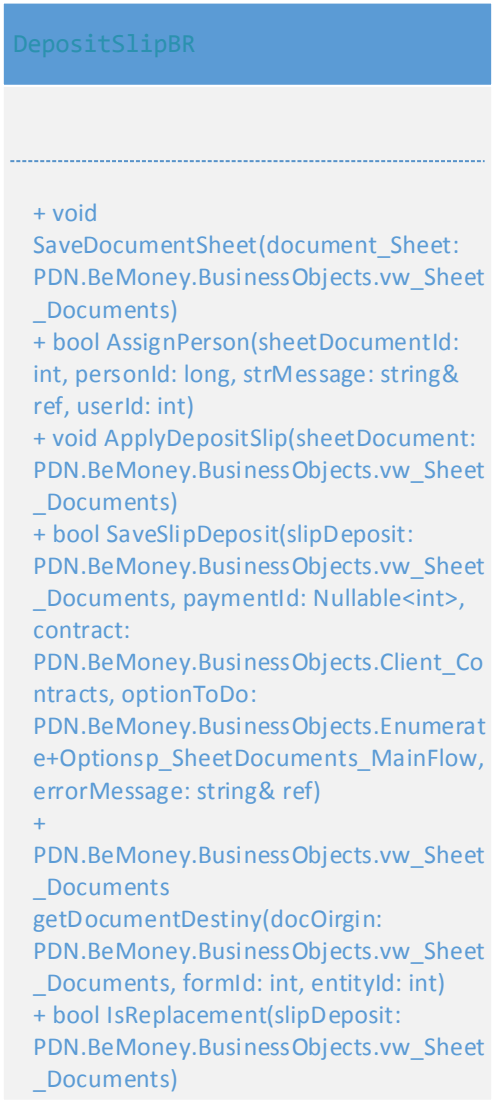

Figura 11. Clase Ficha de Depósito BR (Reglas de Negocio)

Métodos de Clase DepositSlipBR

+ void SaveDocumentSheet(document_Sheet: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents)

+ bool AssignPerson(sheetDocumentId: int, personId: long, strMessage: string ref, userId: int)

+ void ApplyDepositSlip(sheetDocument: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents)

+ bool SaveSlipDeposit(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, paymentId: Nullable<int>, contract: PDN.BeMoney.BusinessObjects.Client_Contracts, optionToDo: PDN.BeMoney.BusinessObjects.Enumerate+Optionsp_SheetDocuments_MainFlow, errorMessage: string ref)

+ PDN.BeMoney.BusinessObjects.vw_Sheet_Documents getDocumentDestiny(docOirgin: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, formId: int, entityId: int)

+ bool IsReplacement(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents)

+ bool ValidForEdit(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ bool ValidateSlipByDevolution(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ PDN.BeMoney.BusinessObjects.vw_Sheet_Documents getDepositSlipByRowItem(ldrPaymentRow: Data.DataRowView)

+ bool RewardDepositSlip(payment: PDN.BeMoney.BusinessObjects.Document_Payments, strMessege: string ref)

+ bool ApplyDevolutionSlipDeposit(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ string getDetailInfoSlipDeposit(sheetDocumentId: int)

+ string getDetailInfoSlipDeposit(sheetDocumentId: int, onlyParents: bool, orderType: PDN.BeMoney.BusinessObjects.Enumerate+OrderType)

+ bool ValidateSlipByCancel(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ bool CancelSlipDeposit(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ bool validateSelectDepositSlip(slipDepositArgs: PDN.BeMoney.BR.BREventArgs`1[[PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, PDN.BeMoney.BusinessObjects, Version=1.0.0.0, Culture=neutral, PublicKeyToken=c06b1b87d6eb6fc8]], strMessage: string ref)

+ bool ValidateForNewMovement(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, movement: PDN.BeMoney.BusinessObjects.Enumerate+SheetMovement, strMessage: string ref)

+ Collections.Generic.List<PDN.BeMoney.BusinessObjects.vw_Sheet_Documents> GetChildDocumentsByDepositSlipId(sheetDocumentId: int)

+ PDN.BeMoney.BusinessObjects.vw_Sheet_Documents GetBySlipDepositById(sheetDocumentId: int)

+ bool UnAssingClient(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ bool ValidateSlipByUnAssign(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ bool UploadSaveDocument(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, fileName: string, safeFileName: string, strMessage: string ref)

+ bool DeleteImageSlipDeposit(documentId: Nullable<int>, strMessage: string ref)

+ PDN.BeMoney.BusinessObjects.DocumentImage GetImageById(documentId: int)

+ bool ValidateNewDestinyMovement(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

+ Collections.Generic.List<PDN.BeMoney.BusinessObjects.vw_Client_Contracts> LoadActiveContractsByPersonId(personId: long)

+ Collections.Generic.List<PDN.BeMoney.BusinessObjects.vw_Person> GetPersonByContractActives(name: string, rfc: string)

+ Collections.Generic.List<PDN.BeMoney.BusinessObjects.vw_Sheet_Documents> GetBy(strWhere: string)

- Data.DataTable getDepositStructure(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, paymentId: Nullable<int>)

- Collections.Generic.List<PDN.BeMoney.BusinessObjects.vw_Cfg_Application_EntitieInPropertie> getCfg_Properties(docOirgin: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, formId: int, entitieId: int)

- Collections.Generic.List<PDN.BeMoney.BusinessObjects.vw_Sheet_Documents> getHerarchyItems(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, onlyParents: bool, orderType: PDN.BeMoney.BusinessObjects.Enumerate+OrderType)

- string validateAmountByPaymentMode(dict: Collections.Generic.Dictionary<string,Object>, slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents)

- bool ExecSaveImage(sheetDocumentId: int, fileBuffer: Byte[], userId: int, strMessage: string ref)

- void validateByNewSlipDeposit(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

- void validateNewSlipByDepositCash(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

- void validateNewSlipByElectronicTransfer(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

- void validateNewSlipByCash(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

- void validateNewSlipByCheque(slipDeposit: PDN.BeMoney.BusinessObjects.vw_Sheet_Documents, strMessage: string ref)

Anexo C Código Fuente

Control General de Administración de Pagos (lenguaje Visual Basic .Net)

```
Imports DevExpress.XtraTab
Imports PDN.BeMoney.BusinessObjects

Public Class frmManageSlipPayment


    Public WithEvents _frmSearchDepositSlip As New
PDN.BeMoney.Windows.Forms.DepositSlip.frmSearchDepositSlip
    Public WithEvents _uclstContracts As New
PDN.BeMoney.Windows.Forms.DepositSlip.uclstContracts

    Private WithEvents _DocumentPendings As frmContractDocumentsData
    Private WithEvents _Payments As frmPaymentHistory

    Private WithEvents _ContractInvoices As frmContractInvoices
    Private WithEvents _ContractAmortizationTable As frmContractAmortizationTable

    Dim blnLoad As Boolean
    Dim isloadContracts As Boolean

    Private Sub InitializeControls()

        Me.sccManageSlipPayment.Collapsed = True

        If Not
Me.sccManageSlipPayment.Panel1.Controls.Contains(Me._frmSearchDepositSlip) Then

            Me._frmSearchDepositSlip.TopLevel = False
            Me.sccManageSlipPayment.Panel1.Controls.Add(Me._frmSearchDepositSlip)
            Me._frmSearchDepositSlip.lblTitle.Text = "Administración de Pagos"
            Me._frmSearchDepositSlip.gpslipDocument.Text = "Fichas de Depósito"
            Me._frmSearchDepositSlip.FormBorderStyle =
Windows.Forms.FormBorderStyle.None

            Me._frmSearchDepositSlip.Show()
            Me._frmSearchDepositSlip.tsbExit.Visible = True
            Me._frmSearchDepositSlip.Dock = DockStyle.Fill

        End If


        If Not Me.gpContracts.Controls.Contains(Me._uclstContracts) Then
            Me.gpContracts.Controls.Add(Me._uclstContracts)
            Me._uclstContracts.Dock = DockStyle.Fill
        End If
```

```
    End Sub

    Public Sub LoadMe()

        InitializeControls()
        LoadPage("XTPDOCUMENTPENDINGS")
        loadAditionalConfiguration()


    End Sub

    Public Sub initializeDocumentPendings()

        If IsNothing(_DocumentPendings) Then
            _DocumentPendings = New frmContractDocumentsData
        End If

        If Not Me.xtpDocumentPendings.Controls.Contains(Me._DocumentPendings) Then
            Me._DocumentPendings.MinimumSize = New Size(800, 250)

            LoadFormToTab(Me._DocumentPendings, xtpDocumentPendings, False)
        End If
    End Sub

    Public Sub InitalizeContractInvoices()

        If IsNothing(_ContractInvoices) Then
            __ContractInvoices = New frmContractInvoices
        End If

        If Not Me.xtpContractInvoices.Contains(Me._ContractInvoices) Then
            LoadFormToTab(Me._ContractInvoices, xtpContractInvoices, False)
        End If

    End Sub

    Public Sub InitializeAmortizationTable()
        If IsNothing(Me._ContractAmortizationTable) Then
            _ContractAmortizationTable = New frmContractAmortizationTable
        End If

        If Not Me.xtpAmortizationTable.Contains(Me._ContractAmortizationTable) Then
            LoadFormToTab(Me._ContractAmortizationTable, xtpAmortizationTable,
False)
        End If
    End Sub

    Public Sub initializePayments()
        If IsNothing(_Payments) Then
            _Payments = New frmPaymentHistory
        End If

        If Not Me.xtpPayments.Controls.Contains(Me._Payments) Then
            LoadFormToTab(Me._Payments, xtpPayments, False)
        End If
    End Sub
```

```
    Public Sub LoadFormToTab(ByVal lfrmForm As System.Windows.Forms.Form, ByVal
lxtpTab As XtraTabPage, ByVal evalueLoadForm As Boolean)
        Try

            If Not lxtpTab.Controls.Contains(lfrmForm) Then
                lfrmForm.TopLevel = False
                lxtpTab.Controls.Add(lfrmForm)
                'lfrmForm.Dock = DockStyle.Fill
            End If

            If evalueLoadForm Then
                If Not blnLoad Then
                    lfrmForm.Show()
                    lfrmForm.Visible = True

                End If
            End If
        Catch ex As Exception
            frmMessage.WindowType = frmMessage.enWindowType.Critical
            frmMessage.Message = ex.Message & "-->" & ex.ToString() + vbCrLf +
ex.StackTrace.ToString
            frmMessage.ShowDialog()
        End Try
    End Sub

    Private Sub LoadManageSlipDeposit()

        _frmSearchDepositSlip.Show()

    End Sub

    Private Sub LoadContracts()
        Throw New NotImplementedException
    End Sub

    Private Sub LoadDOcumentPendings()

        initializeDocumentPendings()


        If Not IsNothing(_uclstContracts.Item) Then

            Me.gpDocuments.Text = String.Format("Documentos de {0} del Contrato: {1}
", _uclstContracts.Item.Commercial_Name, _uclstContracts.Item.Contract_Id)
            Me._DocumentPendings.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._DocumentPendings.EXContract._Contract =
PDN.BeMoney.BR.Client_ContractsBR.Instance.GetByKeyByADO(_uclstContracts.Item.Contra
ct_Id, _uclstContracts.Item.Portfolio_Id, _uclstContracts.Item.Company_Id,
_uclstContracts.Item.Movement_Id, True)

        Else

            Me.gpDocuments.Text = "Documentos del Contrato"
            Me._DocumentPendings.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
```

```
            Me._DocumentPendings.EXContract._Contract = New
PDN.BeMoney.BusinessObjects.Client_Contracts() With {.Contract_Id = 100,
.Contract_Status_Id = 1}

        End If

        Me._DocumentPendings.Show()


        Me._DocumentPendings.spccDocumentPendings.Collapsed = True

        Me._DocumentPendings.Dock = DockStyle.Fill

    End Sub

    Private Sub LoadPayments()

        initializePayments()

        If Not IsNothing(_uclstContracts.Item) Then

            Me._Payments.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._Payments.EXContract._Contract =
PDN.BeMoney.BR.Client_ContractsBR.Instance.GetByKeyByADO(_uclstContracts.Item.Contra
ct_Id, _uclstContracts.Item.Portfolio_Id, _uclstContracts.Item.Company_Id,
_uclstContracts.Item.Movement_Id, True)

        Else

            Me._Payments.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._Payments.EXContract._Contract = New
PDN.BeMoney.BusinessObjects.Client_Contracts() With {.Contract_Id = 100,
.Contract_Status_Id = 1, .Person_Id = -1}

        End If

        Me._Payments.Show()
        Me._Payments.Dock = DockStyle.Fill

    End Sub

    Private Sub LoadContractInvoices()

        InitalizeContractInvoices()

        If Not IsNothing(_uclstContracts.Item) Then

            Me._ContractInvoices.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._ContractInvoices.EXContract._Contract =
PDN.BeMoney.BR.Client_ContractsBR.Instance.GetByKeyByADO(_uclstContracts.Item.Contra
ct_Id, _uclstContracts.Item.Portfolio_Id, _uclstContracts.Item.Company_Id,
_uclstContracts.Item.Movement_Id, True)

        Else
```

```
            Me._ContractInvoices.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._ContractInvoices.EXContract._Contract = New
PDN.BeMoney.BusinessObjects.Client_Contracts() With {.Contract_Id = 100,
.Contract_Status_Id = 1, .Person_Id = -1}

        End If

        Me._ContractInvoices.Show()
        'Me._ContractInvoices.LoadInvoices()
        Me._ContractInvoices.Dock = DockStyle.Fill

    End Sub

    Private Sub LoadAmortizationTable()

        InitializeAmortizationTable()

        If Not IsNothing(_uclstContracts.Item) Then

            Me._ContractAmortizationTable.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._ContractAmortizationTable.EXContract._Contract =
PDN.BeMoney.BR.Client_ContractsBR.Instance.GetByKeyByADO(_uclstContracts.Item.Contra
ct_Id, _uclstContracts.Item.Portfolio_Id, _uclstContracts.Item.Company_Id,
_uclstContracts.Item.Movement_Id, True)

        Else

            Me._ContractAmortizationTable.EXContract = New
PDN.BeMoney.BusinessObjects.Custom.EXContract()
            Me._ContractAmortizationTable.EXContract._Contract = New
PDN.BeMoney.BusinessObjects.Client_Contracts() With {.Contract_Id = 100,
.Contract_Status_Id = 1}

        End If

        Me._ContractAmortizationTable.Show()

        Me._ContractAmortizationTable.Dock = DockStyle.Fill

    End Sub

    Private Sub xTabDocuments_SelectedPageChanged(sender As System.Object, e As
DevExpress.XtraTab.TabPageChangedEventArgs) Handles
xTabDocuments.SelectedPageChanged

        LoadPage(e.Page.Name)

        If e.PrevPage.Name <> e.Page.Name Then
            RemovePrevPage(e.PrevPage.Name)
        End If

    End Sub

    Private Sub LoadPage(PageName As String)

        Select Case PageName.ToUpper
```

```
            Case "XTPDOCUMENTPENDINGS"

                LoadDOcumentPendings()

            Case "XTPPAYMENTS"

                LoadPayments()

            Case "XTPAMORTIZATIONTABLE"

                LoadAmortizationTable()

            Case "XTPCONTRACTINVOICES"

                LoadContractInvoices()

            Case Else

        End Select

    End Sub

    Private Sub RemovePrevPage(PrevPage As String)

        Select Case PrevPage.ToUpper
            Case "XTPDOCUMENTPENDINGS"

                RemoveFormFromTab(_DocumentPendings, xtpDocumentPendings)

            Case "XTPPAYMENTS"

                RemoveFormFromTab(_Payments, xtpPayments)

            Case "XTPAMORTIZATIONTABLE"

                RemoveFormFromTab(_ContractAmortizationTable, xtpAmortizationTable)

            Case "XTPCONTRACTINVOICES"

                RemoveFormFromTab(_ContractInvoices, xtpContractInvoices)

            Case Else

        End Select


    End Sub

    Private Sub RemoveFormFromTab(ByRef lfrmForm As System.Windows.Forms.Form, ByVal
lxtpTab As XtraTabPage)

        If Not IsNothing(lfrmForm) Then
            If lxtpTab.Controls.Contains(lfrmForm) Then
                lxtpTab.Controls.Remove(lfrmForm)
                lfrmForm = Nothing
            End If
        End If
```

```
    End Sub

    Private Sub frmManageSlipPayment_Load(sender As Object, e As System.EventArgs)
Handles Me.Load

        LoadMe()

    End Sub

    Private Sub OnSearchContracts(sender As Object, e As EventArgs) Handles
_frmSearchDepositSlip.OnRaiseSearchContracts

        Dim strWhere As String = ""
        Dim _slipDeposit As PDN.BeMoney.BusinessObjects.vw_Sheet_Documents
        _slipDeposit = DirectCast(sender,
PDN.BeMoney.BusinessObjects.vw_Sheet_Documents)


        If Not IsNothing(_slipDeposit) Then

            strWhere += String.Format("Person_Id = {0} AND Trust_Sub_Id = {1} ",
_slipDeposit.Person_Id, _slipDeposit.Trust_Sub_Id)
            gpContracts.Text = "Contratos de: " & _slipDeposit.Commercial_Name

        End If


        If Not String.IsNullOrEmpty(strWhere) Then


            Me._uclstContracts.ListItems =
PDN.BeMoney.BR.Custom.DepositSlipBR.Instance.loadActiveContractsByPersonId(_slipDepo
sit.Person_Id)



        Else

            Me._uclstContracts.ListItems.Clear()

        End If


        isloadContracts = True
        Me.sccManageSlipPayment.Collapsed = False

        Me._uclstContracts.LoadData()
        Me._uclstContracts.Item = Nothing

        LoadDocumentsByItemSelect()

        Me.SplitContainerControl1.Collapsed = False

    End Sub

    Private Sub On_LoadDocumentPendigs(sender As Object, e As EventArgs) Handles
_uclstContracts.OnRaiseLoadDocumentPendings
```

```
        Try
            Me.Cursor = Cursors.WaitCursor

            LoadDocumentsByItemSelect()

            Me.Cursor = Cursors.Default
        Catch ex As Exception
            Me.Cursor = Cursors.Default
        End Try


    End Sub


    Private Sub LoadDocumentsByItemSelect()

        If xTabDocuments.SelectedTabPage.Name = xtpDocumentPendings.Name Then
            RemovePrevPage("XTPDOCUMENTPENDINGS")
            LoadPage("XTPDOCUMENTPENDINGS")
        Else
            xTabDocuments.SelectedTabPage = xtpDocumentPendings
        End If

        'SplitContainerControl1.CollapsePanel =
DevExpress.XtraEditors.SplitCollapsePanel.Panel1
        SplitContainerControl1.Collapsed = IsNothing(Me._uclstContracts.Item)

    End Sub

    Private Sub OnRaise_CloseMe(sender As Object, e As EventArgs) Handles
_frmSearchDepositSlip.OnRaiseCloseMe

        Me.Close()

    End Sub

    Private Sub OnRaiseCollapsePanel2(sender As Object, e As EventArgs) Handles
_frmSearchDepositSlip.OnRaiseCollapseClientInfo

        isloadContracts = False
        Me.sccManageSlipPayment.Collapsed = True

    End Sub

    Private Sub On_RaiseValidateSlipDepositForDocument(sender As Object, e As
PDN.BeMoney.BR.BREventArgs(Of vw_Sheet_Documents)) Handles
_DocumentPendings.OnRaiseValidateSlipDeposit

        If IsNothing(e) Then
            e = New PDN.BeMoney.BR.BREventArgs(Of vw_Sheet_Documents)
        End If

        If Not IsNothing(_frmSearchDepositSlip.uclistDepositSlip1.Item) Then
            e.Item = _frmSearchDepositSlip.uclistDepositSlip1.Item
        End If

    End Sub
```

```
    Private Sub loadAditionalConfiguration()


    End Sub

    Private Sub sccManageSlipPayment_SplitGroupPanelCollapsing(sender As
System.Object, e As DevExpress.XtraEditors.SplitGroupPanelCollapsingEventArgs)
Handles sccManageSlipPayment.SplitGroupPanelCollapsing

        If e.Collapsing = False Then
            e.Cancel = Not isloadContracts
        End If

    End Sub

    Private Sub OnRaise_DocumentPendings_SplitGroupPanelCollapsing(sender As
System.Object, e As DevExpress.XtraEditors.SplitGroupPanelCollapsingEventArgs)
Handles _DocumentPendings.OnRaiseDocumentPendings_SplitGroupPanelCollapsing

        If e.Collapsing = False Then
            e.Cancel = True
        End If

    End Sub


    Private Sub On_RaiseUpdateSlipDepositByDocumentPendings(sender As Object, e As
EventArgs) Handles _DocumentPendings.OnRaiseUpdateSlipDeposit

        UpdateSearchByEvent()
    End Sub
    Private Sub On_RaiseUpdateSlipDepositByPayments(sender As Object, e As
EventArgs) Handles _Payments.OnRaiseUpdateSlipDeposit
        UpdateSearchByEvent()

    End Sub

    Public Sub UpdateSearchByEvent()
        Me._frmSearchDepositSlip.Search()
        isloadContracts = True
        Me.sccManageSlipPayment.Collapsed = False

    End Sub

End Class
```

Control de trabajo de Fichas de Depósito (lenguaje C#)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using PDN.BeMoney.BusinessObjects.Custom;
using PDN.BeMoney.BusinessObjects;
using PDN.BeMoney.BR;



namespace PDN.BeMoney.Windows.Forms.DepositSlip
{
    public partial class frmSearchDepositSlip : Form
    {


        public frmSearchDepositSlip()
        {
            InitializeComponent();

            initializelist();
            addEvents();

        }

        public event EventHandler<BREventArgs<vw_Sheet_Documents>>
OnRaiseSearchContracts;
        public event EventHandler OnRaiseCloseMe;
        public event EventHandler OnRaiseCollapseClientInfo;
        public event EventHandler<BREventArgs<Client_Contracts>> OnRaiseGetContract;

        private void initializelist()
        {
            if (this.uclistDepositSlip1 == null)
                this.uclistDepositSlip1 = new uclistDepositSlip();

            if (!this.splitContainerControl1.Contains(this.uclistDepositSlip1))
            {
                this.uclistDepositSlip1 = new uclistDepositSlip();
                this.uclistDepositSlip1.CRUD =
PDN.BeMoney.BusinessObjects.Enumerate.CRUD.Create;
                this.uclistDepositSlip1.Dock = System.Windows.Forms.DockStyle.Fill;
                this.uclistDepositSlip1.Load += new
System.EventHandler(this.uclistDepositSlip1_Load);
                this.gpslipDocument.Controls.Add(this.uclistDepositSlip1);
            }

        }

        private bool _bnLoad;
```

```
        private Enumerate.OptionSlipDepositForm _OptionDepositSlipForm =
Enumerate.OptionSlipDepositForm.ManageSlipDeposit;

        public Enumerate.OptionSlipDepositForm OptionDepositSlipForm
        {
            get { return _OptionDepositSlipForm; }
            set
            {
                _OptionDepositSlipForm = value;
                if (uclistDepositSlip1 != null)
                    uclistDepositSlip1._OptionForm = this.OptionDepositSlipForm;
            }
        }

        public uclistDepositSlip uclistDepositSlip1;

        #region botones de Menu
```

Alta Ficha de Depósito

```
        private void tsbNew_Click(object sender, EventArgs e)
        {
            loadNewDpositSlip();
        }

        private void loadNewDpositSlip()
        {
            if (this.uclistDepositSlip1.NewDepositSlip())
                this.Search();
        }
```

Busqueda Fichas de Depósito

```
        private void tsbSearch_Click(object sender, EventArgs e)
        {
            Search();
        }

        public void Search()
        {
            try
            {
                this.Cursor = Cursors.WaitCursor;
                SendMessageOnLoadInfo(new MessageOnLoadInfo("Realizando
búsqueda...por favor espere", Enumerate.ImageStatusMessage.Warning));

                string strQuery = this.ucParametersSlipDepositSearh1.strFilters;

                this.Cursor = Cursors.WaitCursor;


                if (!string.IsNullOrEmpty(strQuery))
                    this.uclistDepositSlip1.ListItems =
PDN.BeMoney.BR.Custom.DepositSlipBR.Instance.GetBy(strQuery);
```

```
                else
                    this.uclistDepositSlip1.ListItems = new
List<BusinessObjects.vw_Sheet_Documents>();

                //this.splitContainerControl1.Collapsed =
true;//(this.uclistDepositSlip1.ListItems.Count > 0);


                this.uclistDepositSlip1.LoadData();
                this.uclistDepositSlip1.showDetailSlipDeposit =
this.ucParametersSlipDepositSearh1.chkShowSlipDetail.Checked;
                this.On_RaiseCollapseClientInfo();

                SendMessageOnLoadInfo(new MessageOnLoadInfo("Listo.",
Enumerate.ImageStatusMessage.Info2));
                this.Cursor = Cursors.Default;

                enableActions(false);

                this.tsbEditSlipDeposit.Enabled = false;
                this.tsbAssignClient.Enabled = false;
                this.tsbLoadContracts.Enabled = false;
                this.tsbUnassignClient.Enabled = false;
                this.tsbDeleteSlipImage.Enabled = false;
                this.tsbUpLoadDocument.Enabled = false;
                this.tsbShowSlipImage.Enabled = false;
            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.WaitCursor;
                Message.frmMessage _frmMessage = new Message.frmMessage();
                _frmMessage.Message = string.Format("Ocurrio un Error al Realizar la
consulta.{0}-->{1}", ex.Message, ex.ToString());
                _frmMessage.WindowType = Message.frmMessage.enWindowType.Warning;
                _frmMessage.ShowDialog();
                SendMessageOnLoadInfo(new MessageOnLoadInfo("",
Enumerate.ImageStatusMessage.Info2));
            }
        }
```

Edicion de Ficha de Depósito

```
        private void tsbEdit_Click(object sender, EventArgs e)
        {
            loadEditDepositSlip();
        }

        private void loadEditDepositSlip()
        {
            this.uclistDepositSlip1.EditDepositSlip();
            this.Search();
        }
```

Asignación de Persona (Cliente) a Ficha de Depósito

```
        private void tsbAssignClient_Click(object sender, EventArgs e)
        {
            this.assignClient();
        }

        private void assignClient()
        {
            if ((this.uclistDepositSlip1.Item.Person_Id == null ? 0 :
this.uclistDepositSlip1.Item.Person_Id) == 0)
            {
                if (this.uclistDepositSlip1.AssignClient())
                    this.Search();
            }
        }
```

Desasignar Persona (Cliente) a Ficha de Depósito

```
        private void tsbUnassignClient_Click(object sender, EventArgs e)
        {
            this.unAssingClient();
        }

        private void unAssingClient()
        {
            if (this.uclistDepositSlip1.Item.Person_Id != null &&
this.uclistDepositSlip1.Item.Person_Id > 0)
            {
                if (this.uclistDepositSlip1.UnAssignClient())
                    this.Search();
            }
        }
```

Crear Ficha Parcial

```
        private void tsbPartialSlip_Click(object sender, EventArgs e)
        {
            this.addMov(Enumerate.SheetMovement.PartialSlip);
        }
```

Traspaso Interbancario

```
        private void tsbInterbank_Click(object sender, EventArgs e)
        {
            this.addMov(Enumerate.SheetMovement.Interbank);
        }
```

Transferencia entre Cuentas

```
        private void tsbTransfer_Click(object sender, EventArgs e)
        {
            this.addMov(Enumerate.SheetMovement.Transfer);
        }



        private void addMov(Enumerate.SheetMovement concepMovement)
        {
            if (this.uclistDepositSlip1.AddMovement(concepMovement))
                this.Search();
        }
```

Devolución de Ficha de Depósito

```
        private void tsbDevolution_Click(object sender, EventArgs e)
        {
            //this.addMov(Enumerate.SheetMovement.Devolution);
            this.devolutionSlip();
        }

        private void devolutionSlip()
        {
            if
(this.uclistDepositSlip1.DevolutionSlip(this.uclistDepositSlip1.Item))
                this.Search();
        }
```

Cancelación de la Ficha de Depósito

```
        private void tsbCancelSlipDeposit_Click(object sender, EventArgs e)
        {
            this.cancelSlipDeposit();
        }

        private void cancelSlipDeposit()
        {

            if (this.uclistDepositSlip1.CancelSlipDeposit())
            {
                this.Search();
            }
        }
```

Selección de Ficha de Depósito para ser aplicada

```
        private void tsbSelectSlipDeposit_Click(object sender, EventArgs e)
        {
            finishSelectSlipDeposit();
```

```
        }

        public void finishSelectSlipDeposit()
        {
            try
            {

                if (validateSelectDepositSlip(this.uclistDepositSlip1.Item))
                {
                    if ((new Message.frmMessage()
                    {
                        Message = string.Format(@"Esta apunto de aplicar la ficha de
depósito ""{0}"". ¿Desea continuar? ",
this.uclistDepositSlip1.Item.Sheet_Document_Id),
                        WindowType = Message.frmMessage.enWindowType.Question
                    }).ShowDialog() == System.Windows.Forms.DialogResult.Yes)
                    {
                        this.DialogResult = System.Windows.Forms.DialogResult.OK;
                        this.Close();
                    }
                    else
                    {

this.uclistDepositSlip1.EnableSelectionApperanceFocusRow(false);
                        this.tsbSelectSlipDeposit.Enabled = false;
                    }
                }
            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.WaitCursor;
                Message.frmMessage _frmMessage = new Message.frmMessage();
                _frmMessage.Message = string.Format("Ocurrio un error al realizar la
validación de la ficha de depósito. {0} --> {1}", ex.Message, ex.ToString());
                _frmMessage.WindowType = Message.frmMessage.enWindowType.Warning;
                _frmMessage.ShowDialog();
                SendMessageOnLoadInfo(new MessageOnLoadInfo("",
Enumerate.ImageStatusMessage.Info2));
            }
        }

        private void tsbLoadContracts_Click(object sender, EventArgs e)
        {
            searchContracts();
        }

        private void searchContracts()
        {

            this.splitContainerControl1.Collapsed = true;
            var e = new BREventArgs<vw_Sheet_Documents>() { Item =
uclistDepositSlip1.Item };
            if (OnRaiseSearchContracts != null)
            {
                OnRaiseSearchContracts(uclistDepositSlip1.Item, e);

                if (!e.Cancel)
                {
```

```
                    //this.ucParametersSlipDepositSearh1.txtCommercialName.Text =
e.Item.Commercial_Name;
                    //this.Search();


                    //this.uclistDepositSlip1.setSelectItem();
                    //this.OnRaiseCollapseClientInfo+=new
EventHandler(On_RaiseCollapseClientInfo);
                }
            }

        }
```

Carga de Comprobante de Ficha de Depósito

```
        private void tsbUpLoadDocument_Click(object sender, EventArgs e)
        {
            this.addSlipDepositImage();
        }

        private void addSlipDepositImage()
        {
            if (uclistDepositSlip1.UpdloadDocument())
                this.Search();

        }
```

Baja de Comprobante de Ficha de Depósito

```
        private void tsbDeleteSlipImage_Click(object sender, EventArgs e)
        {
            this.deleteSlipDepositImage();
        }

        private void deleteSlipDepositImage()
        {
            if (uclistDepositSlip1.DeleteDocumentImage())
                this.Search();
        }
```

Mostrar Compreobante de Ficha de Depósito

```
        private void tsbShowSlipImage_Click(object sender, EventArgs e)
        {
            this.showSlipImage();
        }

        private void showSlipImage()
```

```
        {
            this.uclistDepositSlip1.ShowSlipDepositImage();
        }
```

Mostrar Paramteros Adicionales

```
        private void tsbViewParameters_Click(object sender, EventArgs e)
        {

            viewAdditionalParameters();

        }

        private void viewAdditionalParameters()
        {
            this.splitContainerControl1.CollapsePanel =
DevExpress.XtraEditors.SplitCollapsePanel.Panel1;
            this.splitContainerControl1.Collapsed =
this.splitContainerControl1.Collapsed ? false : true;
        }
```

Salir de la Pantalla de Trabajo

```
        private void tsbExit_Click(object sender, EventArgs e)
        {

            this.Close();

        }

        #endregion
```

Funciones Adicionales para la funcionalidad de la Pantalla

```
        public void SendMessageOnLoadInfo(MessageOnLoadInfo _messageOnLoadInfo)
        {
            BR.Custom.SecurityBR.SendMessage(_messageOnLoadInfo);
        }

        private void addEvents()
        {
```

```
            this.uclistDepositSlip1.OnRaiseGridControleClick += new
EventHandler(uclistDepositSlip1_OnRaiseClick);
            this.uclistDepositSlip1.OnRaiseDoubleClick += new
EventHandler(uclistDepositSlip1_OnRaiseDoubleClick);
            this.ucParametersSlipDepositSearh1.chkShowSlipDetail.CheckedChanged +=
new EventHandler(chkShowSlipDetail_CheckedChanged);

            if (_OptionDepositSlipForm ==
PDN.BeMoney.BusinessObjects.Enumerate.OptionSlipDepositForm.ManageSlipDeposit)
            {

                this.uclistDepositSlip1.OnRaiseAssingClient += new
EventHandler(uclistDepositSlip1_OnRaiseAssingClient);
                this.uclistDepositSlip1.OnRaiseEditDepositSlip += new
EventHandler(uclistDepositSlip1_OnRaiseEditDepositSlip);
                this.uclistDepositSlip1.OnRaiseDeleteImage += new
EventHandler(uclistDepositSlip1_OnRaiseDeleteImage);
                this.uclistDepositSlip1.OnRaiseAddMovement += new
EventHandler(uclistDepositSlip1_OnRaiseAddMovement);
                this.uclistDepositSlip1.OnRaiseAddSlipDepositImage += new
EventHandler(uclistDepositSlip1_OnRaiseAddSlipDepositImage);
                this.uclistDepositSlip1.OnRaiseSearch += new
EventHandler(On_RaiseSearch);

            }
            else
            {

                this.uclistDepositSlip1.OnRaiseAssingClient -= new
EventHandler(uclistDepositSlip1_OnRaiseAssingClient);
                this.uclistDepositSlip1.OnRaiseEditDepositSlip -= new
EventHandler(uclistDepositSlip1_OnRaiseEditDepositSlip);
                this.uclistDepositSlip1.OnRaiseDeleteImage -= new
EventHandler(uclistDepositSlip1_OnRaiseDeleteImage);
                this.uclistDepositSlip1.OnRaiseAddMovement -= new
EventHandler(uclistDepositSlip1_OnRaiseAddMovement);
                this.uclistDepositSlip1.OnRaiseAddSlipDepositImage -= new
EventHandler(uclistDepositSlip1_OnRaiseAddSlipDepositImage);
                this.uclistDepositSlip1.OnRaiseSearch -= new
EventHandler(On_RaiseSearch);

            }

        }

        private void setPermission()
        {
            switch (this._OptionDepositSlipForm)
            {
                case Enumerate.OptionSlipDepositForm.ManageSlipDeposit:

                    this.tsbExit.Visible = false; //por default no cambiar
                    this.tsbSelectSlipDeposit.Visible = false;//por default no
cambiar

                    this.tsbNewSlipDeposit.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbNewSlipDeposit;
```

```
                    this.tsbAssignClient.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbAssignClient;
                    this.tsbUnassignClient.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbAssignClient;
                    this.tsbInterbank.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbInterbank;
                    this.tsbTransfer.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbTransfer;
                    this.tsbPartialSlip.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbPartialSlip;
                    this.tsbDevolutionSlip.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbDevolutionSlip;
                    this.tsbEditSlipDeposit.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbEditSlipDeposit;
                    this.tsbCancelSlipDeposit.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbCancelSlipDeposit;
                    this.tsbLoadContracts.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbLoadContracts;

                    this.tsbShowSlipImage.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbShowSlipImage;
                    this.tsbUpLoadDocument.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbUpLoadDocument;
                    this.tsbDeleteSlipImage.Visible =
PDN.BeMoney.BR.Custom.SecurityBR.permissionsbyuser.read_tsbDeleteSlipImage;

                    setPermissionUclList();
                    break;
                case Enumerate.OptionSlipDepositForm.OnlySearch:

                    this.tsbSelectSlipDeposit.Visible = true;
                    this.tsbExit.Visible = true;
                    this.tsbSearch.Visible = true;
                    this.tsbShowSlipImage.Visible = true;

                    this.tsbNewSlipDeposit.Visible = false;
                    this.tsbAssignClient.Visible = false;
                    this.tsbInterbank.Visible = false;
                    this.tsbTransfer.Visible = false;
                    this.tsbPartialSlip.Visible = false;
                    this.tsbDevolutionSlip.Visible = false;
                    this.tsbEditSlipDeposit.Visible = false;
                    this.tsbCancelSlipDeposit.Visible = false;
                    this.tsbLoadContracts.Visible = false;
                    this.tsbUnassignClient.Visible = false;
                    this.tsbUpLoadDocument.Visible = false;
                    this.tsbDeleteSlipImage.Visible = false;

                    break;
                default:
                    break;
            }


        }

        private void setPermissionUclList()
        {
```

```
            this.uclistDepositSlip1.AllowAssigClient = (this.tsbAssignClient.Visible
&& this.tsbAssignClient.Enabled);
            this.uclistDepositSlip1.AllowIterBank = (this.tsbInterbank.Visible &&
this.tsbInterbank.Enabled);
            this.uclistDepositSlip1.AllowTransfer = (this.tsbTransfer.Visible &&
this.tsbTransfer.Enabled);
            this.uclistDepositSlip1.AllowPartial = (this.tsbPartialSlip.Visible &&
this.tsbPartialSlip.Enabled);
            this.uclistDepositSlip1.AllowCancel = (this.tsbCancelSlipDeposit.Visible
&& this.tsbCancelSlipDeposit.Enabled);
            this.uclistDepositSlip1.AllowEdit = (this.tsbEditSlipDeposit.Visible &&
this.tsbEditSlipDeposit.Enabled);
            this.uclistDepositSlip1.AllowDevolution =
(this.tsbDevolutionSlip.Visible && this.tsbDevolutionSlip.Enabled);

            this.uclistDepositSlip1.AllowDeleteImage =
(this.tsbDeleteSlipImage.Visible && this.tsbDeleteSlipImage.Enabled);
            this.uclistDepositSlip1.AllowAddSlipDepositImage =
(this.tsbUpLoadDocument.Visible && this.tsbUpLoadDocument.Enabled);

        }

        private void loadSlipDeposit()
        {
            frmSlipDepositHeader f = new frmSlipDepositHeader();

            f.MdiParent = this.MdiParent;
            f.SheetDeposit = this.uclistDepositSlip1.Item;
            f.WindowState = FormWindowState.Maximized;

            f.Load();
            f.Show();
        }

        private bool validateSelectDepositSlip(vw_Sheet_Documents slipDeposit)
        {
            string strmessage = "";
            BREventArgs<vw_Sheet_Documents> e = new
BREventArgs<vw_Sheet_Documents>();
            BREventArgs<Client_Contracts> eC = null;


            if (this.OnRaiseGetContract != null)
                this.OnRaiseGetContract(this, eC = new
BREventArgs<Client_Contracts>());

            e.Item = slipDeposit;
            e.DictionaryParams = eC.DictionaryParams;
            e.DictionaryParams.Add("Contract", eC.Item);

            if (!BR.Custom.DepositSlipBR.Instance.validateSelectDepositSlip(e, out
strmessage))
            {
                Forms.Message.frmMessage _frmMessage = new Message.frmMessage();
                _frmMessage.Message = "No se puede aplicar la ficha de depósito por
las siguientes razones:".AddNewLine(2) + strmessage;
                _frmMessage.WindowType = Message.frmMessage.enWindowType.Warning;
```

```
                _frmMessage.ShowDialog();

                return false;
            }

            return true;
        }

        private void enableActions(bool _enable)
        {
            this.tsbSelectSlipDeposit.Enabled = _enable;
            this.tsbInterbank.Enabled = _enable;
            this.tsbTransfer.Enabled = _enable;
            this.tsbPartialSlip.Enabled = _enable;
            this.tsbEditSlipDeposit.Enabled = _enable;
            this.tsbDevolutionSlip.Enabled = _enable;
            this.tsbCancelSlipDeposit.Enabled = _enable;

            this.tsbUpLoadDocument.Enabled = _enable;
            this.tsbDeleteSlipImage.Enabled = _enable;
            this.tsbShowSlipImage.Enabled = _enable;
            this.tsbLoadContracts.Enabled = _enable;

            this.setPermissionUclList();
        }

        private void loadMenuByActiveControl()
        {

            switch (this.ActiveControl.Name.ToUpper())
            {
                case "UCLISTDEPOSITSLIP1":
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit)
                        this.uclistDepositSlip1.ShowMenuRow();
                    break;
                default:
                    break;
            }

        }

        private void loadByQuikKeys(Keys _keyCode)
        {
            switch (_keyCode)
            {
                case Keys.Q://Asignar Cliente
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbAssignClient.Visible &&
this.tsbAssignClient.Enabled)
                        this.assignClient();

                    break;
                case Keys.E://Editar
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit)
                        loadEditDepositSlip();
                    break;
```

```
                case Keys.K://Traspaso Interbancario
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbInterbank.Visible &&
this.tsbInterbank.Enabled)
                        addMov(Enumerate.SheetMovement.Interbank);
                    break;
                case Keys.W://Nueva Ficha de Depósito
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbNewSlipDeposit.Visible
&& this.tsbNewSlipDeposit.Enabled)
                        loadNewDpositSlip();
                    break;
                case Keys.P://Ficha Parcial
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbPartialSlip.Visible &&
this.tsbPartialSlip.Enabled)
                        addMov(Enumerate.SheetMovement.PartialSlip);
                    break;
                case Keys.G://Transferencia entre Cuentas
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbTransfer.Visible &&
this.tsbTransfer.Enabled)
                        addMov(Enumerate.SheetMovement.Transfer);
                    break;
                case Keys.B://Buscar
                    Search();
                    break;
                case Keys.U://Devolución
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbDevolutionSlip.Visible
&& this.tsbDevolutionSlip.Enabled)
                        addMov(Enumerate.SheetMovement.Devolution);
                    break;
                case Keys.L:// cargar Contratos
                    if (_OptionDepositSlipForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit && this.tsbLoadContracts.Visible
&& this.tsbLoadContracts.Enabled)
                        searchContracts();
                    break;
                default:
                    break;
            }
        }

        private void frmSearchDepositSlip_Load(object sender, EventArgs e)
        {

            setPermission();
            //AddEvents();
            Search();
            //solo aplica para la carga inicial del control

this.ucParametersSlipDepositSearh1.cceDepositStatus.SetEditValue(2);//Pediente de
aplicar
        }

        private void uclistDepositSlip1_Load(object sender, EventArgs e)
        {
```

```
        }

        private void uclistDepositSlip1_OnRaiseDoubleClick(object sender, EventArgs
e)
        {
            finishSelectSlipDeposit();
        }

        private void uclistDepositSlip1_OnRaiseClick(object sender, EventArgs e)
        {
            if (this.uclistDepositSlip1.Item != null)
            {
                if (this.uclistDepositSlip1.Item.Person_Id == null)
                    enableActions(false);
                else if (this.uclistDepositSlip1.Item.Person_Id == 0)
                    enableActions(false);
                else if (this.uclistDepositSlip1.Item.Pending_Amount == 0)
                    enableActions(false);
                else
                    enableActions(true);



                if (this.uclistDepositSlip1.Item.Deposit_Status_Id == 3
                    /*|| (this.uclistDepositSlip1.Item.Parent_Document_Id > 0 &&
this.uclistDepositSlip1.Item.Deposit_Status_Id == 4)
                    || ((this.uclistDepositSlip1.Item.Person_Id != null ?
this.uclistDepositSlip1.Item.Person_Id : 0) == 0 &&
this.uclistDepositSlip1.Item.Deposit_Status_Id == 4)*/
                        || this.uclistDepositSlip1.Item.Deposit_Status_Id == 4)
                {
                    enableActions(false);

                    //Desasignar Cliente
                    this.tsbUnassignClient.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbAssignClient
                                                        &&
((this.uclistDepositSlip1.Item.Person_Id == null ? 0 :
this.uclistDepositSlip1.Item.Person_Id) > 0));
                    this.tsbUnassignClient.Enabled = false;

                    //Asignar Cliente
                    this.tsbAssignClient.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbAssignClient
                                                         &&
((this.uclistDepositSlip1.Item.Person_Id == null ? 0 :
this.uclistDepositSlip1.Item.Person_Id) == 0));
                    this.tsbAssignClient.Enabled = false;

                    this.setPermissionUclList();
                    return;

                }
                else
                {
                    ///Validaciones Adicionales para funcionalidad
```

```
                    //Asignar Cliente
                    this.tsbAssignClient.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbAssignClient
                                                        &&
((this.uclistDepositSlip1.Item.Person_Id == null ? 0 :
this.uclistDepositSlip1.Item.Person_Id) == 0));
                    this.tsbAssignClient.Enabled = this.tsbAssignClient.Visible;

                    //Desasignar Cliente
                    this.tsbUnassignClient.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbAssignClient
                                                        &&
((this.uclistDepositSlip1.Item.Person_Id == null ? 0 :
this.uclistDepositSlip1.Item.Person_Id) > 0));
                    this.tsbUnassignClient.Enabled = this.tsbUnassignClient.Visible;

                    //Cargar Contratos del Cliente
                    this.tsbLoadContracts.Enabled =
((this.uclistDepositSlip1.Item.Person_Id == null ? 0 :
this.uclistDepositSlip1.Item.Person_Id) > 0);

                    //Editar Ficha de Depósito
                    this.tsbEditSlipDeposit.Enabled =
(this.uclistDepositSlip1.Item.Parent_Document_Id < 0 &&
this.uclistDepositSlip1.Item.Deposit_Status_Id == 2);

                    //Cancelar Ficha de Depósito
                    this.tsbCancelSlipDeposit.Enabled =
!(this.uclistDepositSlip1.Item.Deposit_Status_Id == 3 ||
this.uclistDepositSlip1.Item.Deposit_Status_Id == 4);


                }


                //Cargar Imagen de Ficha de Depósito
                this.tsbUpLoadDocument.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbUpLoadDocument
                                                    &&
((this.uclistDepositSlip1.Item.Document_Id == null ? 0 :
this.uclistDepositSlip1.Item.Document_Id) == 0));
                this.tsbUpLoadDocument.Enabled = this.tsbUpLoadDocument.Visible;


                //Eliminar Imagen de Ficha de Depósito
                this.tsbDeleteSlipImage.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbDeleteSlipImage
                                                    &&
((this.uclistDepositSlip1.Item.Document_Id == null ? 0 :
this.uclistDepositSlip1.Item.Document_Id) > 0));
                this.tsbDeleteSlipImage.Enabled = this.tsbDeleteSlipImage.Visible;


                //Mostrar Imagen de Ficha de Depósito
                this.tsbShowSlipImage.Visible =
(BR.Custom.SecurityBR.permissionsbyuser.read_tsbShowSlipImage
```

```
                                                &&
((this.uclistDepositSlip1.Item.Document_Id == null ? 0 :
this.uclistDepositSlip1.Item.Document_Id) > 0));
                this.tsbShowSlipImage.Enabled = this.tsbShowSlipImage.Visible;


            }

            this.setPermissionUclList();
        }

        private void uclistDepositSlip1_OnRaiseAssingClient(object sender, EventArgs
e)
        {
            assignClient();
        }

        private void uclistDepositSlip1_OnRaiseEditDepositSlip(object sender,
EventArgs e)
        {
            loadEditDepositSlip();
        }

        private void frmSearchDepositSlip_KeyDown(object sender,
System.Windows.Forms.KeyEventArgs e)
        {

            if (e.Control)
            {
                loadByQuikKeys(e.KeyCode);
            }
            else
                if (e.KeyCode == Keys.Apps)
                {
                    loadMenuByActiveControl();
                }

        }

        public void uclistDepositSlip1_OnRaiseAddMovement(object sender, EventArgs
e)
        {
            this.addMov((Enumerate.SheetMovement)sender);
        }

        public void uclistDepositSlip1_OnRaiseDeleteImage(object sender, EventArgs
e)
        {
            this.deleteSlipDepositImage();
        }

        public void uclistDepositSlip1_OnRaiseAddSlipDepositImage(object sender,
EventArgs e)
        {
            this.addSlipDepositImage();

        }
```

```
        private void On_RaiseSearch(object sender, EventArgs e)
        {

            var d = sender as vw_Sheet_Documents;

            if (d != null)
            {

//ucParametersSlipDepositSearh1.cceTrust_Sub.setStrigValueMembers(d.Trust_Sub_Id.ToS
tring());

                if (this.uclistDepositSlip1.ListItems.Count == 0)
                {
                    ucParametersSlipDepositSearh1.dtStartDepositDate.Text =
d.Deposit_Date.ToString("dd/MM/yyyy");
                    ucParametersSlipDepositSearh1.dtEndDepositDate.Text =
d.Deposit_Date.ToString("dd/MM/yyyy");
                }
                //if (d.Person_Id != null && d.Person_Id > 0)
                //{
                //    ucParametersSlipDepositSearh1.txtCommercialName.Text =
d.Commercial_Name;
                //}
                //else
                //{
                //
ucParametersSlipDepositSearh1.cceSheetConcept.setStrigValueMembers(d.Sheet_Concept_I
d.ToString());
                //
ucParametersSlipDepositSearh1.ccePaymentForm.setStrigValueMembers(d.Document_Type_Id
.ToString());
                //}

                this.Search();
            }
        }

        private void On_RaiseCollapseClientInfo()
        {

            if (this.OnRaiseCollapseClientInfo != null)
            {
                this.OnRaiseCollapseClientInfo(this, new EventArgs());


            }
        }

        public void OnRaise_SendMessageOnLoadInfo(object sender, EventArgs e)
        {
            SendMessageOnLoadInfo(sender as MessageOnLoadInfo);
        }

        private void chkShowSlipDetail_CheckedChanged(object sender, EventArgs e)
        {
            this.uclistDepositSlip1.showDetailSlipDeposit =
this.ucParametersSlipDepositSearh1.chkShowSlipDetail.Checked;
        }
```

```
        private void frmSearchDepositSlip_FormClosed(object sender,
FormClosedEventArgs e)
        {

            if (this.OnRaiseCloseMe != null)
                this.OnRaiseCloseMe(this, new EventArgs());
        }









    }

}
```

Control de Listado de Fichas de Depósito

Configuracion General de Control

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using DevExpress.Utils;
using DevExpress.Utils.Menu;
using DevExpress.XtraGrid;
using DevExpress.XtraGrid.Views.Grid;
using PDN.BeMoney.BR;
using PDN.BeMoney.BusinessObjects;
using PDN.BeMoney.Windows.Forms.Message;
using PDN.BeMoney.Windows.Forms.Peoples;

namespace PDN.BeMoney.Windows.Forms.DepositSlip
{

    public partial class uclistDepositSlip : ucControlBase
```

```
{
    public uclistDepositSlip()
    {
        InitializeComponent();
    }


    #region Permisos

    /// <summary>
    /// Permiso para realizar Transferencia entre Cuentas
    /// </summary>
    public bool AllowTransfer = true;

    /// <summary>
    /// Permiso para realizar Traspasos interbancarios
    /// </summary>
    public bool AllowIterBank = true;

    /// <summary>
    /// permiso para realizar fichas parciales
    /// </summary>
    public bool AllowPartial = true;

    /// <summary>
    /// permiso para realizar la edicion de una ficha de depósito
    /// </summary>
    public bool AllowEdit = true;

    /// <summary>
    /// permisio para desasignar un cliente
    /// </summary>
    public bool AllowAssigClient = true;

    /// <summary>
    /// permiso para realizar una devolución
    /// </summary>
    public bool AllowDevolution = true;

    /// <summary>
    /// permiso para realizar una cancelación
    /// </summary>
    public bool AllowCancel = true;

    /// <summary>
    /// permiso para dar de baja un comprobante
    /// </summary>
    public bool AllowDeleteImage = true;

    /// <summary>
    /// permiso para dar de alta un comprobante
    /// </summary>
    public bool AllowAddSlipDepositImage = true;

    /// <summary>
    /// permiso para crear un documento de recuperación
    /// </summary>
    public bool AllowDocumentDestiny = true;
```

```
        #endregion

        #region Variables privadas

        private DevExpress.XtraEditors.Repository.RepositoryItemMemoEdit
repositoryItemRichTextEdit1 = null;

        /// <summary>
        /// Control que muestra el detalle del flujo de una ficha de depósito
        /// </summary>
        private ToolTipController tcDetailSlipDeposit = null;

        /// <summary>
        /// indica si ya se cargo la configuracion de las columnas del grid de
documentos de recuperacion
        /// </summary>
        private bool _isLoaded_gvDocuments_Ref;

        /// <summary>
        /// indica el listado de fichas de deposito
        /// </summary>
        private List<vw_Sheet_Documents> _listItems = new
List<vw_Sheet_Documents>();

        /// <summary>
        /// indica la ficha de deposito selccionada o en foco
        /// </summary>
        private vw_Sheet_Documents _item = null;

        /// <summary>
        /// indica si se mostrar el detalle del flujo de una ficha depósito
        /// </summary>
        private bool _showDetailSlipDeposit = true;

        /// <summary>
        /// indica las columnas del control
        /// </summary>
        private List<DevExpress.XtraGrid.Columns.GridColumn> _Columns = null;

        /// <summary>
        /// indica si se ya esta cargada la configuración de las columnas
        /// </summary>
        private bool _isLoaded = false;

        /// <summary>
        /// indica las columnas para las fichas de deposito de referencia para un
documento de recupeación
        /// </summary>
        private List<DevExpress.XtraGrid.Columns.GridColumn>
_Columns_gvDocuments_Ref = null;

        #endregion

        #region Propiedades publicas

        /// <summary>
        /// indica si se mostrar el detalle del flujo de una ficha depósito
```

```
        /// </summary>
        public bool showDetailSlipDeposit
        {
            get { return _showDetailSlipDeposit; }
            set { _showDetailSlipDeposit = value; }
        }

        /// <summary>
        /// evento que se dispara al hcae clic en el grid
        /// </summary>
        public event EventHandler OnRaiseGridControleClick;

        /// <summary>
        /// evento auxiliar de doble clic en un registro
        /// </summary>
        public event EventHandler OnRaiseDoubleClick;

        /// <summary>
        /// evento auxiliar para la asignación de cliente a la ficha de depósito
        /// </summary>
        public event EventHandler OnRaiseAssingClient;

        /// <summary>
        /// evento auxiliar para la creacion de un movimiento a una fi9cha de
depósito
        /// </summary>
        public event EventHandler OnRaiseAddMovement;

        /// <summary>
        /// evento auxiliar para la edición de una ficha de depósito
        /// </summary>
        public event EventHandler OnRaiseEditDepositSlip;

        /// <summary>
        /// evento auxiliar para dar de alta una ficha de depósito
        /// </summary>
        public event EventHandler OnRaiseNewDepositSlip;

        /// <summary>
        /// evento auxiliar para la devolución de una ficha de depósito
        /// </summary>
        public event EventHandler OnRaiseDevolution;

        /// <summary>
        /// evento auxiliar para la búsqueda de fichas de depósito
        /// </summary>
        public event EventHandler OnRaiseSearch;

        /// <summary>
        /// evento auxiliar para baja de una comprobante de uan ficha de depósito
        /// </summary>
        public event EventHandler OnRaiseDeleteImage;

        /// <summary>
        /// evento auxiliar para la carga de un comprobante de una ficha de depósito
        /// </summary>
        public event EventHandler OnRaiseAddSlipDepositImage;
```

```
        /// <summary>
        /// indica la forma en que se esta trabajando en el control
        /// </summary>
        public Enumerate.OptionSlipDepositForm _OptionForm =
Enumerate.OptionSlipDepositForm.ManageSlipDeposit;

        /// <summary>
        /// Listado de fichas de depósito
        /// </summary>
        public List<vw_Sheet_Documents> ListItems
        {
            get { return _listItems; }
            set { _listItems = value; }
        }

        /// <summary>
        /// ficha de depósito activa
        /// </summary>
        public vw_Sheet_Documents Item
        {
            get
            {
                GetSelectRow();
                return _item;
            }
            set { _item = value; }
        }

        /// <summary>
        /// Columnas del control
        /// </summary>
        public List<DevExpress.XtraGrid.Columns.GridColumn> Columns
        {
            get
            {
                if (_Columns == null)
                    _Columns = new List<DevExpress.XtraGrid.Columns.GridColumn>();
                return _Columns;
            }
            set
            {
                _Columns = value;
            }
        }

        /// <summary>
        /// Columnas de fichas de deposito de un documento de recuperación
        /// </summary>
        public List<DevExpress.XtraGrid.Columns.GridColumn> Columns_gvDocuments_Ref
        {
            get
            {
                if (_Columns_gvDocuments_Ref == null)
                    _Columns_gvDocuments_Ref = new
List<DevExpress.XtraGrid.Columns.GridColumn>();
                return _Columns_gvDocuments_Ref;
            }
            set
```

```
        {
            _Columns_gvDocuments_Ref = value;
        }
    }


    #endregion

    #region Eventos de acceso interno

    /// <summary>
    /// evento de clic sobre el grid
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void gridControl1_Click(object sender, EventArgs e)
    {

        EnableSelectionApperanceFocusRow(true);

        GetSelectRow();

        setSelectedRow();

        setToolTipInfoDetail();

    }

    /// <summary>
    /// doble clic en el grid
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void gridControl1_MouseDoubleClick(object sender, MouseEventArgs e)
    {
        switch (e.Button)
        {
            case MouseButtons.Left:
                setDoubleClick();
                break;
            case MouseButtons.Middle:
                break;
            case MouseButtons.None:
                break;
            case MouseButtons.Right:
                break;
            case MouseButtons.XButton1:
                break;
            case MouseButtons.XButton2:
                break;
            default:
                break;
        }
    }

    /// <summary>
    /// carga de menu en el grid
    /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void gridView1_ShowGridMenu(object sender,
DevExpress.XtraGrid.Views.Grid.GridMenuEventArgs e)
        {


            if (_OptionForm == Enumerate.OptionSlipDepositForm.ManageSlipDeposit)
            {
                if (e.MenuType == GridMenuType.Row)
                {

                    e.Menu.Items.Clear();

                    if (this._item != null)
                    {
                        if (this.AllowAssigClient)
                            if (this._item.Person_Id == null &&
this._item.Destiny_Document_Id < 1)
                                e.Menu.Items.Add(new DXMenuItem("Asignar Cliente",
new EventHandler(OnRaise_AssignClient)));

                        if (this.AllowEdit && this._item.Destiny_Document_Id < 1)
                            e.Menu.Items.Add(new DXMenuItem("Editar", new
EventHandler(OnRaise_Edit)));

                        if (this.AllowAddSlipDepositImage)
                            e.Menu.Items.Add(new DXMenuItem("Agregar Comprobante",
new EventHandler(OnRaise_AddSlipDepositImage)));

                        if (this.AllowDeleteImage)
                            e.Menu.Items.Add(new DXMenuItem("Eliminar Comprobante",
new EventHandler(OnRaise_DeleteImage)));

                        DXSubMenuItem menuAddMovement = new
DXSubMenuItem("Movimientos");

                        if (this.AllowPartial && this._item.Destiny_Document_Id < 1)
                            menuAddMovement.Items.Add(new DXMenuItem("Ficha
Parcial", new EventHandler(OnRaise_PartialSlip)));

                        if (this.AllowIterBank)
                            menuAddMovement.Items.Add(new DXMenuItem("Traspaso
Interbancario", new EventHandler(OnRaise_InterBank)));

                        if (this.AllowTransfer)
                            menuAddMovement.Items.Add(new DXMenuItem("Transferencia
entre Cuentas", new EventHandler(OnRaise_Transfer)));

                        if (this.AllowDevolution && this._item.Destiny_Document_Id <
1)
                            menuAddMovement.Items.Add(new DXMenuItem("Devolución",
new EventHandler(OnRaise_DevolutionConfirm)));


                        if (this.AllowDocumentDestiny &&
this.validSelectDocumentsToDestinyRecovery())
```

```
                            menuAddMovement.Items.Add(new DXMenuItem("Realizar
Recuperación", new EventHandler(OnRaise_Recovery)));


                        if (menuAddMovement.Items != null &&
menuAddMovement.Items.Count > 0)
                            e.Menu.Items.Add(menuAddMovement);



                    }
                }
            }

        }

        /// <summary>
        /// cambio de foco de un registro en el grid
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void gridView1_FocusedRowChanged(object sender,
DevExpress.XtraGrid.Views.Base.FocusedRowChangedEventArgs e)
        {

            setSelectedRow();
        }

        /// <summary>
        /// cambio de la selección de un registro en el grid
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void gridView1_SelectionChanged(object sender,
DevExpress.Data.SelectionChangedEventArgs e)
        {
            if (this.gridView1.SelectedRowsCount == 1)
                setToolTipInfoDetail();
        }

        /// <summary>
        /// indica la carga de pantalla para una devolucion como un movimiento
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void OnRaise_Devolution(object sender, EventArgs e)
        {
            if (OnRaiseAddMovement != null)
                OnRaiseAddMovement(Enumerate.SheetMovement.Devolution, new
EventArgs());
        }

        /// <summary>
        /// perdida del foco del listado de fichas deposito
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void uclistDepositSlip_Leave(object sender, EventArgs e)
```

```
        {
            clearToolTipInfoDetail();
        }



        #endregion

        #region Eventos de acceso publico

        public void OnRaise_Recovery(object sender, EventArgs e)
        {

            NewRecovery();

        }

        public void OnRaise_AssignClient(Object sender, EventArgs e)
        {

            if (OnRaiseAssingClient != null)
                OnRaiseAssingClient(this, new EventArgs());
        }

        public void OnRaise_PartialSlip(object sender, EventArgs e)
        {

            if (OnRaiseAddMovement != null)
                OnRaiseAddMovement(Enumerate.SheetMovement.PartialSlip, new
EventArgs());

        }

        public void OnRaise_InterBank(object sender, EventArgs e)
        {

            if (OnRaiseAddMovement != null)
                OnRaiseAddMovement(Enumerate.SheetMovement.Interbank, new
EventArgs());

        }

        public void OnRaise_Transfer(object sender, EventArgs e)
        {

            if (OnRaiseAddMovement != null)
                OnRaiseAddMovement(Enumerate.SheetMovement.Transfer, new
EventArgs());

        }

        public void OnRaise_DevolutionConfirm(object sender, EventArgs e)
        {
            //if (OnRaiseDevolution != null)
            //    this.OnRaiseDevolution(this, new EventArgs());

            //if (OnRaiseAddMovement != null)
```

```
            //    OnRaiseAddMovement(Enumerate.SheetMovement.Devolution, new
EventArgs());
            if (this.Item != null)
                DevolutionSlip(this.Item);
        }

        public void OnRaise_Edit(Object sender, EventArgs e)
        {

            //this.EditDepositSlip();
            if (this.OnRaiseEditDepositSlip != null)
                this.OnRaiseEditDepositSlip(this, new EventArgs());

        }

        public void OnRaise_DeleteImage(object sender, EventArgs e)
        {

            if (this.OnRaiseDeleteImage != null)
                this.OnRaiseDeleteImage(this, new EventArgs());

        }

        public void OnRaise_AddSlipDepositImage(object sender, EventArgs e)
        {

            if (this.OnRaiseAddSlipDepositImage != null)
                this.OnRaiseAddSlipDepositImage(this, new EventArgs());

        }

        public void OnRaise_Search(object sender, EventArgs e)
        {
            if (this.OnRaiseSearch != null)
                this.OnRaiseSearch(sender, new EventArgs());
        }

        /// <summary>
        ///  metodo de confirmacion para asignacion de una ficha de depósito
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        public void OnRaise_ConfirmAssignPerson(object sender,
BREventArgs<vw_Person> e)
        {
            if (e.Item != null)
            {
                if ((new Message.frmMessage()
                {
                    Message = string.Format(@"Esta apunto de asignar el cliente
""{0}"" a la ficha de depósito. ¿Desea continuar? ",
                                e.Item.Commercial_Name),
                    WindowType = Message.frmMessage.enWindowType.Question
                }).ShowDialog() == DialogResult.Yes)
                {
                    e.Cancel = false;
                }
                else
```

```
                {
                    e.Cancel = true;
                }
            }
        }

        #endregion

        #region Metodos funcionalidad interna del control

        /// <summary>
        /// Oculta columnas en el grid de fichas de depósito
        /// </summary>
        private void hideColumns()
        {
            this.gridView1.Columns["Deposit_Status_Id"].Visible = false;
        }

        /// <summary>
        /// oculta columnas de las fichas de deposito para un documento de
recuperación
        /// </summary>
        private void hideColumnsgvDocuments_Ref()
        {
            this.gvDocuments_Ref.Columns["Deposit_Status_Id"].Visible = false;
        }

        /// <summary>
        /// Crea la instancia de una columna con su configuraciónn
        /// </summary>
        /// <param name="FieldName">nombre del campo </param>
        /// <param name="Caption">descripción de la columna</param>
        /// <returns></returns>
        private DevExpress.XtraGrid.Columns.GridColumn newColumn(string FieldName,
string Caption)
        {
            DevExpress.XtraGrid.Columns.GridColumn column = new
DevExpress.XtraGrid.Columns.GridColumn();
            column.FieldName = FieldName;
            column.Caption = Caption;
            column.DisplayFormat.FormatType = DevExpress.Utils.FormatType.Custom;
            column.DisplayFormat.FormatString = "{0:c2}";
            column.Visible = true;
            return column;
        }

        //private void setRepositoryItemsgvDocuments_Ref()
        //{
        //    if (this.gvDocuments_Ref.Columns["Comments"] != null)
        //    {
        //        this.gvDocuments_Ref.Columns["Comments"].MaxWidth = 500;
        //        this.gvDocuments_Ref.Columns["Comments"].Width = 300;
        //        this.gvDocuments_Ref.Columns["Comments"].ColumnEdit = null;
        //        this.gvDocuments_Ref.Columns["Comments"].RealColumnEdit.AutoHeight
= true;
        //        this.repositoryItemRichTextEdit1 = new
DevExpress.XtraEditors.Repository.RepositoryItemMemoEdit();
```

```
        //        this.gvDocuments_Ref.Columns["Comments"].ColumnEdit =
this.repositoryItemRichTextEdit1;
        //    }
        //}

        /// <summary>
        /// metodo que da formato al grid las fichas de depósito
        /// </summary>
        private void setFormat()
        {

            // this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Checked", Caption = "Seleccionar"});
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Sheet_Document_Id", Caption = "#Ficha Depósito" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Root_Document_Id", Caption = "Root Id" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Parent_Document_Id", Caption = "Parent Id" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Commercial_Name", Caption = "Cliente/Razón Social" });
            this.Columns.Add(newColumn("Pending_Amount", "Importe x Aplicar"));
            this.Columns.Add(newColumn("Applied_Amount", "Importe Aplicado"));
            this.Columns.Add(newColumn("Total_Document", "Total Documento"));
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Sheet_Number", Caption = "Referencia" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Account_Number_Origin", Caption = "Cuenta Origen" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Bank_Name", Caption = "Banco Origen" }); ;
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Account_Number", Caption = "Cuenta Depósito" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Bank_Destiny_Name", Caption = "Banco Depósito" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Deposit_Date", Caption = "Fec. de Depósito" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Trust_Description", Caption = "Origen de Cartera" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Document_Type_Description", Caption = "Forma de Pago" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Sheet_Name", Caption = "Movimiento" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Deposit_Description", Caption = "Status Ficha" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Payment_Status_Description", Caption = "Status Pago" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Document_Status_Description", Caption = "Status Documento" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Currency_Name", Caption = "Divisa" });
            this.Columns.Add(newColumn("Exchange_Rate", "Tipo de Cambio"));
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Comments", Caption = "Comentarios" });
            this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Deposit_Status_Id", Caption = "Deposit_Status_Id" });
```

```
            this._isLoaded = true;
        }

        /// <summary>
        /// metodo que da formato al grid de las fichas de deposito de un documento
de recuperación
        /// </summary>
        private void setFormat_gvDocuments_Ref()
        {

            // this.Columns.Add(new DevExpress.XtraGrid.Columns.GridColumn() {
FieldName = "Checked", Caption = "Seleccionar"});
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Sheet_Document_Id", Caption
= "#Ficha Depósito" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Root_Document_Id", Caption =
"Root Id" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Parent_Document_Id", Caption
= "Parent Id" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Commercial_Name", Caption =
"Cliente/Razón Social" });
            this.Columns_gvDocuments_Ref.Add(newColumn("Pending_Amount", "Importe x
Aplicar"));
            this.Columns_gvDocuments_Ref.Add(newColumn("Applied_Amount", "Importe
Aplicado"));
            this.Columns_gvDocuments_Ref.Add(newColumn("Total_Document", "Total
Documento"));
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Sheet_Number", Caption =
"Referencia" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Account_Number_Origin",
Caption = "Cuenta Origen" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Bank_Name", Caption = "Banco
Origen" }); ;
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Account_Number", Caption =
"Cuenta Depósito" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Bank_Destiny_Name", Caption
= "Banco Depósito" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Deposit_Date", Caption =
"Fec. de Depósito" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Trust_Description", Caption
= "Origen de Cartera" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Document_Type_Description",
Caption = "Forma de Pago" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Sheet_Name", Caption =
"Movimiento" });
```

```
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Deposit_Description",
Caption = "Status Ficha" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Payment_Status_Description",
Caption = "Status Pago" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName =
"Document_Status_Description", Caption = "Status Documento" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Currency_Name", Caption =
"Divisa" });
            this.Columns_gvDocuments_Ref.Add(newColumn("Exchange_Rate", "Tipo de
Cambio"));
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Comments", Caption =
"Comentarios" });
            this.Columns_gvDocuments_Ref.Add(new
DevExpress.XtraGrid.Columns.GridColumn() { FieldName = "Deposit_Status_Id", Caption
= "Deposit_Status_Id" });



            this._isLoaded_gvDocuments_Ref = true;
        }

        /// <summary>
        /// asigna un control adicional a la columna de comentarios
        /// </summary>
        private void setRepositoryItems()
        {
            if (this.gridView1.Columns["Comments"] != null)
            {
                this.gridView1.Columns["Comments"].MaxWidth = 500;
                this.gridView1.Columns["Comments"].Width = 300;
                this.gridView1.Columns["Comments"].ColumnEdit = null;
                this.gridView1.Columns["Comments"].RealColumnEdit.AutoHeight = true;
                this.repositoryItemRichTextEdit1 = new
DevExpress.XtraEditors.Repository.RepositoryItemMemoEdit();
                this.gridView1.Columns["Comments"].ColumnEdit =
this.repositoryItemRichTextEdit1;
            }
        }

        /// <summary>
        /// configura el grid con formato de colores segun el estatus de la ficha de
deposito
        /// </summary>
        private void setColorConditions()
        {
            StyleFormatCondition condition = null;

            //condition = new StyleFormatCondition();
            //condition.Appearance.BackColor = Color.LightBlue;
            ////condition1.Appearance.ForeColor = Color;
            ////condition1.Appearance.BackColor2 = Color.SeaShell;
            //condition.Appearance.Options.UseBackColor = true;
            //condition.Appearance.Options.UseForeColor = true;
```

```
            //condition.Condition = FormatConditionEnum.Expression;
            //condition.Expression = "Parent_Document_Id = -1";//Depósito Origen

            condition = new StyleFormatCondition();
            condition.Appearance.BackColor = Color.Red;
            condition.Appearance.ForeColor = Color.White;
            //condition.Appearance.BackColor2 = Color.SeaShell;
            condition.Appearance.Options.UseBackColor = true;
            condition.Appearance.Options.UseForeColor = true;
            condition.Condition = FormatConditionEnum.Equal;
            condition.Condition = FormatConditionEnum.Expression;
            condition.Expression = "Deposit_Status_Id = 4";//Cancelado

            this.gridView1.FormatConditions.Add(condition);


            condition = new StyleFormatCondition();
            condition.Appearance.BackColor = Color.Gray;
            condition.Appearance.ForeColor = Color.White;
            //condition1.Appearance.BackColor2 = Color.SeaShell;
            condition.Appearance.Options.UseBackColor = true;
            condition.Appearance.Options.UseForeColor = true;
            condition.Condition = FormatConditionEnum.Expression;
            condition.Expression = "Deposit_Status_Id = 3";//Devuelto

            this.gridView1.FormatConditions.Add(condition);


        }

        /// <summary>
        /// Asigna las columnas que tendran totales
        /// </summary>
        private void setFooter()
        {
            this.gridView1.OptionsView.ShowFooter = true;
            this.gridView1.Columns["Pending_Amount"].SummaryItem.SummaryType =
DevExpress.Data.SummaryItemType.Sum;
            this.gridView1.Columns["Pending_Amount"].SummaryItem.DisplayFormat =
"{0:c2}";

            this.gridView1.Columns["Applied_Amount"].SummaryItem.SummaryType =
DevExpress.Data.SummaryItemType.Sum;
            this.gridView1.Columns["Applied_Amount"].SummaryItem.DisplayFormat =
"{0:c2}";

            this.gridView1.Columns["Total_Document"].SummaryItem.SummaryType =
DevExpress.Data.SummaryItemType.Sum;
            this.gridView1.Columns["Total_Document"].SummaryItem.DisplayFormat =
"{0:c2}";


        }

        /// <summary>
        /// asigna las columnas que tendran totales cuando se agrupen las fichas de
depósito
        /// </summary>
```

```
        private void setFooterGroup()
        {
            // forcing the group footer to be always visible
            gridView1.GroupFooterShowMode = GroupFooterShowMode.VisibleAlways;
            // create and setup the first summary item
            GridGroupSummaryItem item = new GridGroupSummaryItem();
            item.FieldName = "Sheet_Document_Id";
            item.SummaryType = DevExpress.Data.SummaryItemType.Count;
            // adding the first summary item to the collection
            gridView1.GroupSummary.Add(item);
            // create and setup second summary item
            GridGroupSummaryItem item1;
            item1 = new GridGroupSummaryItem();
            item1.FieldName = "Pending_Amount";
            item1.SummaryType = DevExpress.Data.SummaryItemType.Sum;
            item1.DisplayFormat = "Total Pendiente: {0:c2}";
            item1.ShowInGroupColumnFooter =
this.gridView1.Columns["Pending_Amount"];
            gridView1.GroupSummary.Add(item1);

            // create and setup second summary item
            GridGroupSummaryItem item2;
            item2 = new GridGroupSummaryItem();
            item2.FieldName = "Applied_Amount";
            item2.SummaryType = DevExpress.Data.SummaryItemType.Sum;
            item2.DisplayFormat = "Total Aplicado: {0:c2}";
            item2.ShowInGroupColumnFooter =
this.gridView1.Columns["Applied_Amount"];
            gridView1.GroupSummary.Add(item2);

            // create and setup second summary item
            GridGroupSummaryItem item3;
            item3 = new GridGroupSummaryItem();
            item3.FieldName = "Total_Document";
            item3.SummaryType = DevExpress.Data.SummaryItemType.Sum;
            item3.DisplayFormat = "Total Ficha: {0:c2}";
            item3.ShowInGroupColumnFooter =
this.gridView1.Columns["Total_Document"];
            gridView1.GroupSummary.Add(item3);

        }

        /// <summary>
        /// oculta la columna Checked
        /// </summary>
        private void showCheckedColumn()
        {
            //solo para aplicar pagos
            if (this.gridView1.Columns["Checked"] != null && this._OptionForm ==
Enumerate.OptionSlipDepositForm.ManageSlipDeposit)
            {

                if (BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrapply
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrpartial
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrreplace
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctraddit
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrexced
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrpayment
```

```
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrreverse)
                {
                    this.gridView1.Columns["Checked"].Visible = true;
                    this.gridView1.Columns["Checked"].OptionsColumn.AllowEdit =
true;
                }
                else
                {
                    this.gridView1.Columns["Checked"].Visible = false;
                    this.gridView1.Columns["Checked"].OptionsColumn.AllowEdit =
false;
                }
            }
            else
            {
                this.gridView1.Columns["Checked"].Visible = false;
                this.gridView1.Columns["Checked"].OptionsColumn.AllowEdit = false;
            }
        }

        /// <summary>
        /// muestra la columna Checked para la funcionalidad de generacion de
Documento de Recuperación
        /// </summary>
        private void showCheckedColumngvDocuments_Ref()
        {
            //solo para aplicar pagos
            if (this.gvDocuments_Ref.Columns["Checked"] != null && this._OptionForm
== Enumerate.OptionSlipDepositForm.ManageSlipDeposit)
            {

                if (BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrapply
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrpartial
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrreplace
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctraddit
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrexced
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrpayment
                    || BR.Custom.SecurityBR.permissionsbyuser.read_btnmngctrreverse)
                {
                    this.gvDocuments_Ref.Columns["Checked"].Visible = true;
                    this.gvDocuments_Ref.Columns["Checked"].OptionsColumn.AllowEdit
= true;
                }
                else
                {
                    this.gvDocuments_Ref.Columns["Checked"].Visible = false;
                    this.gvDocuments_Ref.Columns["Checked"].OptionsColumn.AllowEdit
= false;
                }
            }
            else
            {
                this.gvDocuments_Ref.Columns["Checked"].Visible = false;
                this.gvDocuments_Ref.Columns["Checked"].OptionsColumn.AllowEdit =
false;
            }
        }
```

```
        /// <summary>
        /// asigna el formato de las columnas de las fichas de depósito
        /// </summary>
        private void bindFormat()
        {
            DevExpress.XtraGrid.Columns.GridColumn column = null;
            List<DevExpress.XtraGrid.Columns.GridColumn> wrapColumns = new
List<DevExpress.XtraGrid.Columns.GridColumn>();
            int index = 0;
            foreach (var columnSpecification in this.Columns)
            {
                columnSpecification.Name = columnSpecification.FieldName;
                columnSpecification.VisibleIndex = index;
                columnSpecification.BestFit();
                columnSpecification.OptionsColumn.AllowEdit = false;
                wrapColumns.Add(columnSpecification);

                index++;
            }

            this.gridView1.Columns.AddRange(wrapColumns.ToArray());

            //this.gridView1.ScrollStyle =
DevExpress.XtraGrid.Views.Grid.ScrollStyleFlags.LiveHorzScroll;
            this.gridView1.HorzScrollVisibility =
DevExpress.XtraGrid.Views.Base.ScrollVisibility.Auto;
        }

        /// <summary>
        /// asigna el formato de las columnas de las fichas de depósito para un
documento  de recuperación
        /// </summary>
        private void bindFormatgvDocuments_Ref()
        {
            DevExpress.XtraGrid.Columns.GridColumn column = null;
            List<DevExpress.XtraGrid.Columns.GridColumn> wrapColumns = new
List<DevExpress.XtraGrid.Columns.GridColumn>();
            int index = 0;
            foreach (var _columnSpecification in this.Columns_gvDocuments_Ref)
            {
                _columnSpecification.Name = "gvDocuments_Ref_" +
_columnSpecification.FieldName;
                _columnSpecification.VisibleIndex = index;
                _columnSpecification.BestFit();
                _columnSpecification.OptionsColumn.AllowEdit = false;
                wrapColumns.Add(_columnSpecification);

                index++;
            }

            this.gvDocuments_Ref.Columns.AddRange(wrapColumns.ToArray());

            //this.gridView1.ScrollStyle =
DevExpress.XtraGrid.Views.Grid.ScrollStyleFlags.LiveHorzScroll;
            this.gvDocuments_Ref.HorzScrollVisibility =
DevExpress.XtraGrid.Views.Base.ScrollVisibility.Auto;
        }
```

```
        /// <summary>
        /// limpia el control que muestra el detalle de los movimientos activos de
una ficha de depósito
        /// </summary>
        private void clearToolTipInfoDetail()
        {
            if (tcDetailSlipDeposit != null)
            {
                tcDetailSlipDeposit.HideHint();
            }
        }

        /// <summary>
        /// ejecuta funcionalidad para el doble clic sobre una ficha de depósito
        /// </summary>
        private void setDoubleClick()
        {

            GetSelectRow();

            if (this.Item != null)
            {
                if (OnRaiseDoubleClick != null && this._OptionForm ==
Enumerate.OptionSlipDepositForm.OnlySearch)
                {
                    OnRaiseDoubleClick(this.Item, new EventArgs());
                    return;
                }

                // ShowDepositSlip();

                // this.EditDepositSlip();
            }
        }

        /// <summary>
        /// asigna el valor al control que muestra el detalle del flujo de una ficha
depósito si esta tiene movimientos activos
        /// </summary>
        private void setToolTipInfoDetail()
        {
            string strInfo = "";

            clearToolTipInfoDetail();

            if (this.showDetailSlipDeposit)
            {

                if (this.Item != null && this.Item.Parent_Document_Id > 0)
                {
                    strInfo = getDetailInfoSlipDeposit();

                    if (!string.IsNullOrEmpty(strInfo))
                    {
                        if (tcDetailSlipDeposit == null)
                            tcDetailSlipDeposit = new ToolTipController();
```

```
                        tcDetailSlipDeposit.HideHint();
                        tcDetailSlipDeposit.ToolTipType = ToolTipType.Default;

                        tcDetailSlipDeposit.ResetAutoPopupDelay();
                        tcDetailSlipDeposit.AutoPopDelay = 10000;
                        tcDetailSlipDeposit.ShowHint(strInfo, "Detalle del
Movimiento", ToolTipLocation.BottomCenter);
                        tcDetailSlipDeposit.Active = true;

                    }
                }
            }
        }

        /// <summary>
        ///  asigna la ficha de deposito activa y la envia en cun evento para su
evaluacion en el control padre
        /// </summary>
        private void setSelectedRow()
        {
            if (this.gridView1.SelectedRowsCount == 1)
            {
                if (this.Item != null)
                {
                    if (this.OnRaiseGridControleClick != null)
                        this.OnRaiseGridControleClick(this.Item, new EventArgs());
                }
            }
            else if (this.gridView1.SelectedRowsCount > 1)
            {
                // crear evento para enviar  items y validar botones de acuerdo a
estos

            }

        }

        /// <summary>
        /// carga la ficha de deposito seleccionada
        /// </summary>
        private void showDepositSlip()
        {
            if (this.Item != null)
            {
                frmDepositSlip f = new frmDepositSlip();


                f.ucDepositSlip1.Document_Sheet = this.Item;
                f.ucDepositSlip1.CRUD = Enumerate.CRUD.Update;
                f.tsbSave.Enabled = false;

                this.Cursor = Cursors.WaitCursor;
                f.ShowDialog();

                this.Cursor = Cursors.Default;
            }
        }
```

```
        #endregion



        #region Metodos publicos/ Acceso funcionalidad de los Movimientos y carga de
configuración del control



        /// <summary>
        ///
        /// </summary>
        /// <param name="strmessage"></param>
        /// <param name="imageStatusMessage"></param>
        public void SendMessageOnLoadInfo(string strmessage,
Enumerate.ImageStatusMessage imageStatusMessage)
        {
            BR.Custom.SecurityBR.SendMessage(strmessage, imageStatusMessage);
        }

        /// <summary>
        /// carga la configuracion del control
        /// </summary>
        public void LoadData()
        {
            if (!this._isLoaded)
            {
                this.gridView1.OptionsBehavior.Editable = true;

                setFormat();
                bindFormat();
                setRepositoryItems();
                setFooter();
                setFooterGroup();
                hideColumns();
                setColorConditions();

                setFormat_gvDocuments_Ref();
                bindFormatgvDocuments_Ref();
                hideColumnsgvDocuments_Ref();


            }

            this.gridControl1.DataSource = _listItems;
            _item = null;
            this.gridView1.BestFitColumns();
            this.gridView1.OptionsView.ColumnAutoWidth = false;
            this.gridView1.OptionsSelection.EnableAppearanceFocusedRow = false;
            this.gridView1.OptionsView.ShowGroupPanel = false;// (this._OptionForm
== Enumerate.OptionSlipDepositForm.ManageSlipDeposit);
```

```
            this.gvDocuments_Ref.BestFitColumns();
            this.gvDocuments_Ref.OptionsView.ColumnAutoWidth = false;
            this.gvDocuments_Ref.OptionsSelection.EnableAppearanceFocusedRow =
false;
            this.gvDocuments_Ref.OptionsView.ShowGroupPanel = false;//
(this._OptionForm == Enumerate.OptionSlipDepositForm.ManageSlipDeposit);


            //showCheckedColumn();

        }

        /// <summary>
        /// Obtien el registro seleccionado
        /// </summary>
        public void GetSelectRow()
        {

            if (this.gridControl1.FocusedView != null)
            {

                var gv = this.gridControl1.FocusedView as GridView;

                switch (this.gridControl1.FocusedView.Name)
                {
                    case "gridView1":
                        this._item =
this.gridControl1.FocusedView.GetRow((this.gridControl1.FocusedView as
GridView).FocusedRowHandle) as vw_Sheet_Documents;
                        break;
                    case "gvDocuments_Ref":
                        this._item =
this.gridControl1.FocusedView.GetRow((this.gridControl1.FocusedView as
GridView).FocusedRowHandle) as vw_Sheet_Documents;
                        break;

                    default:
                        this._item = null;
                        break;

                }
            }
            else
            {
                this._item = null;
            }

        }

        /// <summary>
        /// Muestra el Menu al hcaer clic derecho sobre una ficha de deposito
        /// </summary>
        public void ShowMenuRow()
        {
            GridMenuEventArgs e;
            gridView1_ShowGridMenu(this, e = new GridMenuEventArgs(new
DevExpress.XtraGrid.Menu.GridViewMenu(this.gridView1)
```

```
                , new DevExpress.XtraGrid.Views.Grid.ViewInfo.GridHitInfo()
                {
                    Column = this.gridView1.FocusedColumn,
                    RowHandle = this.gridView1.FocusedRowHandle
                }));

        }

        /// <summary>
        /// deshabilita la apariencia de registros selccionados
        /// </summary>
        /// <param name="p"></param>
        public void EnableSelectionApperanceFocusRow(bool p)
        {
            this.gridView1.OptionsSelection.EnableAppearanceFocusedRow = p;
        }
```

Edición de una Ficha Depósito

```
        /// <summary>
        /// carga la pantalla de edicion de una ficha de depósito
        /// </summary>
        public void EditDepositSlip()
        {
            if (this.Item != null)
            {
                if (validForEdit(this.Item))
                {
                    frmDepositSlip f = new frmDepositSlip();
                    int formId = 0;
                    formId = getFormMovement(this.Item.Sheet_Concept_Id);

                    f.ucDepositSlip1.cfgFormControls =
BR.vw_Cfg_Form_ControlsBR.Instance.GetByADO(string.Format("Form_Id = {0} ",
formId));
                    f.ucDepositSlip1.Document_Sheet = this.Item;
                    f.ucDepositSlip1.CRUD = Enumerate.CRUD.Update;

                    //f.OnRaiseSearch += new EventHandler(On_RaiseSearch);
                    f.tsbSave.Enabled = true;
                    f.tsbNew.Visible = false;

                    this.Cursor = Cursors.WaitCursor;

                    f.ShowDialog();

                    this.Cursor = Cursors.Default;
                }
            }
        }
```

Alta de una Ficha de Depósito

```
        /// <summary>
        /// carga la pantalla para dar de alta una nueva ficha de depósito
        /// </summary>
        /// <returns></returns>
        public bool NewDepositSlip()
        {

            frmDepositSlip f = new frmDepositSlip();

            vw_Cfg_Form_Concept cfg_Form =
BR.vw_Cfg_Form_ConceptsBR.Instance.GetByADO(" Form_Concept_Id =
4").FirstOrDefault();

            if (cfg_Form != null)
            {
                f.cfgForm = cfg_Form;
                f.ucDepositSlip1.CRUD = Enumerate.CRUD.Create;
                f.OnRaiseSearch += new EventHandler(OnRaise_Search);
                f.ucDepositSlip1.cfgForm = cfg_Form;
                f.ucDepositSlip1.cfgFormControls =
BR.vw_Cfg_Form_ControlsBR.Instance.GetByADO(string.Format("Form_Id = {0} ",
cfg_Form.Form_Id));
                f.ucDepositSlip1.Document_Sheet =
BR.Custom.DepositSlipBR.Instance.getDocumentDestiny(new vw_Sheet_Documents() {
Sheet_Concept_Id = 1 }, cfg_Form.Form_Id, 1);

                f.ucDepositSlip1.isNewOrigin = true;
                f.tsbSave.Enabled = false;

                this.Cursor = Cursors.WaitCursor;

                f.ShowDialog();

                this.Cursor = Cursors.Default;

                return f.listSlipSave.Count > 0;
            }
            else
            {
                Message.frmMessage _frmMessage = new Message.frmMessage();
                _frmMessage.WindowType = Message.frmMessage.enWindowType.Warning;
                _frmMessage.Message = "No se encontro la configuración de la forma";
                _frmMessage.ShowDialog();

                return false;
            }

            return true;
        }
```

Asingación de Persona (Cliente) a una Ficha de Depósito

```
        /// <summary>
        /// ejecuta la funcionalidad para la asignacion de un cliente a la ficha de
depóito seleccionada
        /// </summary>
        /// <returns></returns>
        public bool AssignClient()
        {
            bool isAssign = false;
            frmSearchPerson _frmseachPerson = new frmSearchPerson();
            _frmseachPerson.searchClient = true;
            _frmseachPerson.ShowsIsPDNClientColumn = false;
            _frmseachPerson.SearchByContractsActives = true;
            _frmseachPerson.OnRaiseConfirmAssignPerson += new
EventHandler<BREventArgs<vw_Person>>(OnRaise_ConfirmAssignPerson);

            if (_frmseachPerson.ShowDialog() == DialogResult.Yes)
            {

                this.Cursor = Cursors.WaitCursor;

                this.Item.Person_Id = _frmseachPerson.ItemSelected.Person_Id;
                string strmessage = "";

                if
(BR.Custom.DepositSlipBR.Instance.AssignPerson(this.Item.Sheet_Document_Id,
_frmseachPerson.ItemSelected.Person_Id, out  strmessage,
BR.Custom.SecurityBR.UsuarioBemoney.IdUser))
                {
                    this.Cursor = Cursors.Default;
                    isAssign = true;
                }
                else
                {
                    this.Cursor = Cursors.Default;

                    Message.frmMessage _frmMessage = new Message.frmMessage();
                    _frmMessage.WindowType =
Message.frmMessage.enWindowType.Warning;
                    _frmMessage.Message = strmessage;
                    _frmMessage.ShowDialog();

                }

            }

            return isAssign;
        }
```

Movimiento a una Ficha de Depósito (Ficha Parcial, Traspaso Interbancariom, Transferencia entre Cuentas)

```
        /// <summary>
        /// Metodo que carga la funcionalidad para aplicar un movimiento a la ficha
de deposito deleccionada
        /// </summary>
        /// <param name="concepMovement"></param>
        /// <returns></returns>
        public bool AddMovement(Enumerate.SheetMovement concepMovement)
        {
            bool isAdd = true;
            try
            {
                if (validNewMovementInSlipDespoit(concepMovement))
                {
                    vw_Cfg_Form_Concept ItemForm =
PDN.BeMoney.BR.vw_Cfg_Form_ConceptsBR.Instance.GetByADO(string.Format(@"
Form_Concept_Id IN({0})", (int)concepMovement)).FirstOrDefault();

                    if (ItemForm != null)
                    {
                        this.Cursor = Cursors.WaitCursor;

                        var cfg =
BR.vw_Cfg_Form_ControlsBR.Instance.GetByADO(string.Format("Form_Id = {0} ",
ItemForm.Form_Id));

                        frmDepositSlipDocument f = new frmDepositSlipDocument();


                        //se configura Forma
                        f.cfgForm = ItemForm;
                        f.ucDepositSlip2.cfgFormControls = cfg;

                        //se configura Documento Origen
                        f.ucDepositSlip1.Document_Sheet = this.Item;
                        //se cargan los documentos relacionados con el documento
origen
                        f.LoadDocumentOfDocumentOrigin();

                        //se configura el documento destino
                        f.LoadDepositSlipDestiny();

                        //Se carga la forma
                        f.ShowDialog();

                        isAdd = f.listSlipSave.Count > 0;

                        this.Cursor = Cursors.Default;
                    }
                    else
                    {
                        this.Cursor = Cursors.Default;
                        Message.frmMessage _frmMessage = new Message.frmMessage();
                        _frmMessage.WindowType =
Message.frmMessage.enWindowType.Warning;
```

```
                        _frmMessage.Message = "Ocurrio un error al tratar de obtener
la configuración de la Forma";
                        _frmMessage.ShowDialog();
                        isAdd = false;
                    }

                }
                else
                {
                    isAdd = false;
                }
            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.Default;
                Message.frmMessage _frmMessage = new Message.frmMessage();
                _frmMessage.WindowType = Message.frmMessage.enWindowType.Critical;
                _frmMessage.Message = ex.ToString();
                _frmMessage.ShowDialog();
                isAdd = false;
            }

            return isAdd;
        }

        /// <summary>
        /// Carga la pantalla para la confirmación de la devolución de una ficha de
depósito
        /// </summary>
        /// <param name="slipDeposit"></param>
        /// <returns></returns>
        public bool DevolutionSlip(vw_Sheet_Documents slipDeposit)
        {

            bool isRefund = true;
            string strmessage = "";

            try
            {

                if (slipDeposit != null)
                {

                    if
(BR.Custom.DepositSlipBR.Instance.ValidateSlipByDevolution(slipDeposit, out
strmessage))
                    {
                        frmDevolutionSlip _frmDevolutionSlip = new
frmDevolutionSlip();
                        if (_frmDevolutionSlip.ShowDialog() == DialogResult.OK)
                        {
                            this.Cursor = Cursors.WaitCursor;

                            slipDeposit.Comments =
_frmDevolutionSlip.meComments.Text;
```

```
                            if
(BR.Custom.DepositSlipBR.Instance.ApplyDevolutionSlipDeposit(slipDeposit, out
strmessage))
                            {
                                this.Cursor = Cursors.Default;

                                Message.frmMessage _frmMessage = new
Message.frmMessage();
                                _frmMessage.WindowType =
Message.frmMessage.enWindowType.Info;
                                _frmMessage.Message = @"La devolución de la ficha de
depósito se realizo con exito";
                                _frmMessage.ShowDialog();
                            }
                            else
                            {
                                throw new Exception(strmessage);
                            }

                        }
                        else
                        {
                            isRefund = false;

                        }

                    }
                    else
                    {
                        this.Cursor = Cursors.Default;
                        isRefund = false;
                        Message.frmMessage _frmMessage = new Message.frmMessage();
                        _frmMessage.Message = @"No se puede devolver la ficha de
depósito. " + strmessage;
                        _frmMessage.WindowType =
Message.frmMessage.enWindowType.Warning;
                        _frmMessage.ShowDialog();
                    }

                }
            }
            catch (Exception ex)
            {
                isRefund = false;
                this.Cursor = Cursors.Default;
                Message.frmMessage _frmMessage = new Message.frmMessage();
                _frmMessage.Message = string.Format(@"Ha ocurrido un Error. {0} -->
{1}", ex.Message, ex.ToString());
                _frmMessage.WindowType = Message.frmMessage.enWindowType.Critical;
                _frmMessage.ShowDialog();
            }
            finally
            {
                this.Cursor = Cursors.Default;
            }

            return isRefund;
        }
```

```
        /// <summary>
        /// Carga la pantalla para la confirmación de la concelación de una ficha
depósito
        /// </summary>
        /// <returns></returns>
        public bool CancelSlipDeposit()
        {

            bool isCanceled = false;

            try
            {

                if (this.validateByCancel())
                {
                    if ((new Message.frmMessage()
                    {
                        Message = string.Format(@"Esta apunto de cancelar la ficha
de depósito No. {0}, ¿desea continuar?", this.Item.Sheet_Document_Id),
                        WindowType = Message.frmMessage.enWindowType.Question
                    }).ShowDialog() == DialogResult.Yes)
                    {
                        string strMessage = "";
                        if
(BR.Custom.DepositSlipBR.Instance.CancelSlipDeposit(this.Item, out strMessage))
                        {
                            frmMessage _frmMessage = new frmMessage();
                            _frmMessage.WindowType = frmMessage.enWindowType.Info;
                            _frmMessage.Message = String.Format(@"La ficha de
depósito No. {0} fue cancelada con exito.", this.Item.Sheet_Document_Id);
                            _frmMessage.ShowDialog();
                            isCanceled = true;
                        }
                        else
                        {
                            frmMessage _frmMessage = new frmMessage();
                            _frmMessage.WindowType =
frmMessage.enWindowType.Critical;
                            _frmMessage.Message = string.Format(@"Ocurrio un Error
al intentar cancelar la ficha de depósito. {0}", strMessage);
                            _frmMessage.ShowDialog();
                            isCanceled = false;
                        }
                    }
                }

            }
            catch (Exception ex)
            {
                frmMessage _frmMessage = new frmMessage();
                _frmMessage.WindowType = frmMessage.enWindowType.Critical;
                _frmMessage.Message = string.Format("Ocurrio un Error al intentar
cancelar la ficha de depósito. {0} --> {1}", ex.Message, ex.ToString());
                _frmMessage.ShowDialog();
                isCanceled = false;
            }
            finally
```

```
            {

            }
            return isCanceled;
        }
```

Desasingnación de la Persona (Cliente) a una Ficha de Depósito

```
        /// <summary>
        /// carga la pantalla de confirmación para la desasignación de la persona en
una ficha de depósito
        /// </summary>
        /// <returns></returns>
        public bool UnAssignClient()
        {
            bool isUnassign = false;
            string strMessage = "";

            if (this.Item.Person_Id != null && this.Item.Person_Id > 0)
            {

                if (this.validateByUnAssign())
                {
                    if (BR.Custom.DepositSlipBR.Instance.UnAssingClient(this.Item,
out strMessage))
                    {
                        isUnassign = true;
                    }
                    else
                    {
                        frmMessage _frmMessage = new frmMessage();
                        _frmMessage.WindowType = frmMessage.enWindowType.Warning;
                        _frmMessage.Message = string.Format(@"Ocurrio un Error al
intentar desasignar el cliente. {0}", strMessage);
                        _frmMessage.ShowDialog();

                        isUnassign = false;
                    }
                }
                else
                {
                    isUnassign = false;
                }
            }

            return isUnassign;
        }
```

Carga de Comprobante de una Ficha Depósito

```
        /// <summary>
        /// carga la panatalla para la alta de un comprobante de ficha de depósito
        /// </summary>
        /// <returns></returns>
        public bool UpdloadDocument()
        {
            frmMessage _frmMessage = null;
            bool isUpload = false;
            string strmessage = "";

            if (this.Item != null)
            {

                OpenFileDialog openD = new OpenFileDialog();

                if (openD.ShowDialog() == DialogResult.OK)
                {
                    SendMessageOnLoadInfo("Almacendando Archivo, por favor
espere...", Enumerate.ImageStatusMessage.Warning);
                    if
(PDN.BeMoney.BR.Custom.DepositSlipBR.Instance.UploadSaveDocument(this.Item,
openD.FileName, openD.SafeFileName, out strmessage))
                    {
                        isUpload = true;
                        SendMessageOnLoadInfo("Listo.",
Enumerate.ImageStatusMessage.Info2);
                        _frmMessage = new frmMessage();
                        _frmMessage.WindowType = frmMessage.enWindowType.Info;
                        _frmMessage.Message = "El archivo se cargo con exito";
                        _frmMessage.ShowDialog();

                    }
                    else
                    {
                        isUpload = false;
                        SendMessageOnLoadInfo("Error al cargar el archivo.",
Enumerate.ImageStatusMessage._Stop);
                        _frmMessage = new frmMessage();
                        _frmMessage.WindowType = frmMessage.enWindowType.Critical;
                        _frmMessage.Message = "Error al cargar el archivo:
".AddNewLine(2) + strmessage;
                        _frmMessage.ShowDialog();
                    }
                    SendMessageOnLoadInfo("", Enumerate.ImageStatusMessage.None);
                }
            }

            return isUpload;
        }
```

Baja de un Comprobante de una Ficha de Depósito

```
        /// <summary>
        /// da de baja el comprbante de una ficha de depósito
        /// </summary>
        /// <returns></returns>
        public bool DeleteDocumentImage()
        {
            bool isDelete = false;
            string strMessage = "";

            try
            {
                if (this.Item != null && (this.Item.Document_Id != null ?
this.Item.Document_Id : 0) > 0)
                {

                    if ((new frmMessage()
                    {
                        WindowType = frmMessage.enWindowType.Optionl,
                        Message = string.Format("Esta apunto de eliminar el archivo
relacionado a la ficha de depósito {0}, ¿desea continuar?",
this.Item.Sheet_Document_Id)
                    }).ShowDialog() == DialogResult.Yes)
                    {
                        if
(BR.Custom.DepositSlipBR.Instance.DeleteImageSlipDeposit(this.Item.Document_Id, out
strMessage))
                        {
                            isDelete = true;
                            SendMessageOnLoadInfo("Listo.",
Enumerate.ImageStatusMessage.Info2);
                        }
                        else
                        {
                            SendMessageOnLoadInfo("Error al intentar eliminar el
archivo.", Enumerate.ImageStatusMessage.Warning);
                            frmMessage _frmMessage = new frmMessage();
                            _frmMessage.WindowType =
frmMessage.enWindowType.Critical;
                            _frmMessage.Message = "Error al intentar eliminar el
archivo:".AddNewLine(2) + strMessage;
                            _frmMessage.ShowDialog();

                        }
                        SendMessageOnLoadInfo("",
Enumerate.ImageStatusMessage.None);
                    }

                }

            }
            catch (Exception ex)
            {
                frmMessage _frmMessage = new frmMessage();
                _frmMessage.WindowType = frmMessage.enWindowType.Critical;
```

```
                _frmMessage.Message = "Error al intentar eliminar el
archivo:".AddNewLine(2) + ex.Message;
                _frmMessage.ShowDialog();

            }
            finally
            {

            }

            return isDelete;

        }
```

Mostrar comprobante de una Ficha de Depósito

```
        /// <summary>
        /// carla la pantalla con el comprobante de una ficha de depósito
        /// </summary>
        public void ShowSlipDepositImage()
        {

            string strMessage = "";

            try
            {
                if (this.Item.Document_Id != null)
                {

                    var _image =
PDN.BeMoney.BR.Custom.DepositSlipBR.Instance.GetImageById((int)this.Item.Document_Id
);

                    Utilities.frmViewImage fImage = new Utilities.frmViewImage();

                    fImage.ImageBuffer = _image.FileData;
                    fImage.Text = this.Item.Sheet_Name + " No. " +
this.Item.Sheet_Document_Id;
                    //fImage.LoadMe();
                    fImage.ShowDialog();

                }

            }
            catch (Exception ex)
            {

                frmMessage _frmMessage = new frmMessage();
                _frmMessage.WindowType = frmMessage.enWindowType.Critical;
                _frmMessage.Message = string.Format(@"Error al intentar cargar la
imagen de la ficha de deposito No. {0}. {1} --> {2}", this.Item.Sheet_Document_Id,
ex.Message, ex.ToString());
                _frmMessage.ShowDialog();
            }
```

```
        }
```

Aplicación de Permisos

```
        /// <summary>
        /// Aplica los permisos asignados
        /// </summary>
        public void SetPermission()
        {
            switch (this._OptionForm)
            {
                case Enumerate.OptionSlipDepositForm.ManageSlipDeposit:
                    //this.AllowAssigClient = true;
                    //this.AllowEdit = true;
                    //this.AllowIterBank = true;
                    //this.AllowPartial = true;
                    //this.AllowTransfer = true;
                    //this.AllowDevolution = true;
                    break;
                case Enumerate.OptionSlipDepositForm.OnlySearch:
                    this.AllowAssigClient = false;
                    this.AllowEdit = false;
                    this.AllowIterBank = false;
                    this.AllowPartial = false;
                    this.AllowTransfer = false;
                    this.AllowDevolution = false;
                    this.AllowCancel = false;
                    this.AllowAddSlipDepositImage = false;
                    this.AllowDeleteImage = false;

                    break;
                default:
                    break;
            }


        }

        /// <summary>
        /// carga la pantalla para la genreación de un documento de recuperación
        /// </summary>
        public void NewRecovery()
        {


            frmDocumentSlipDestiny _frmDocumentSlipDestiny = new
frmDocumentSlipDestiny();

            _frmDocumentSlipDestiny._uclistDepositSlip.ListItems =
this.getSelectedSlip();
            _frmDocumentSlipDestiny._uclistDepositSlip.showDetailSlipDeposit =
false;
            _frmDocumentSlipDestiny._uclistDepositSlip.LoadData();
```

```
            _frmDocumentSlipDestiny.ShowDialog();

        }

        #endregion
```

Validaciones para los Movimientos

```
        #region Validaciones y funciones internas para los Movimientos

        /// <summary>
        /// valida si se puede realizar un movimiento a la ficha de depósito
seleccionada
        /// </summary>
        /// <param name="_movement"></param>
        /// <returns></returns>
        private bool validNewMovementInSlipDespoit(Enumerate.SheetMovement
_movement)
        {
            string strMessage = "";
            string strMov = "";

            var mov =
BR.vw_Cfg_Form_ConceptsBR.Instance.GetByADO(string.Format("Form_Concept_Id = {0}",
(int)_movement)).FirstOrDefault();

            strMov = mov != null ? mov.Sheet_Name : "";

            if (this.Item != null)
            {
                if (!BR.Custom.DepositSlipBR.Instance.ValidateForNewMovement(Item,
_movement, out strMessage))
                {
                    Message.frmMessage _frmMessage = new Message.frmMessage();
                    _frmMessage.Message = string.Format(@"No se puede aplicar el
movimiento de {0} a un(a) {1} por las siguientes razones:", strMov,
this.Item.Document_Group_Name).AddNewLine(2) + strMessage;
                    _frmMessage.WindowType = Message.frmMessage.enWindowType.Info;
                    _frmMessage.ShowDialog();
                    return false;
                }
            }
            else
            {
                return false;
            }

            return true;
        }

        /// <summary>
        /// Valida la ficha de depósito para la desasignación de la persona
        /// </summary>
        /// <returns></returns>
        private bool validateByUnAssign()
```

```
        {
            string strMessage = "";
            bool isValidByUnAssing = true;

            if (BR.Custom.DepositSlipBR.Instance.ValidateSlipByUnAssign(this.Item,
out strMessage))
            {

                if ((new frmMessage()
                {
                    Message = string.Format(@"Esta apunto de Desasignar el Cliente
{0} a la ficha de depósito No. {1}, ¿desea continuar?", this.Item.Commercial_Name,
this.Item.Sheet_Document_Id),
                    WindowType = frmMessage.enWindowType.Optionl
                }).ShowDialog() == DialogResult.Yes)
                {
                    isValidByUnAssing = true;
                }
                else
                {
                    isValidByUnAssing = false;
                }


            }
            else
            {
                frmMessage _frmMessage = new frmMessage();
                _frmMessage.WindowType = frmMessage.enWindowType.Warning;
                _frmMessage.Message = string.Format("No se puede Desasignar el
Cliente {0} a la ficha de depósito No. {1} por las siguientes razones:",
this.Item.Commercial_Name, this.Item.Sheet_Document_Id).AddNewLine(2) + strMessage;
                _frmMessage.ShowDialog();

                isValidByUnAssing = false;
            }

            return isValidByUnAssing;
        }

        /// <summary>
        /// Valida la ficha de deposito para su cancelación
        /// </summary>
        /// <returns></returns>
        private bool validateByCancel()
        {
            string strMessage = "";
            bool isValidByCancel = true;
            if (!BR.Custom.DepositSlipBR.Instance.ValidateSlipByCancel(this.Item,
out strMessage))
            {
                frmMessage _frmMessage = new frmMessage();
                _frmMessage.WindowType = frmMessage.enWindowType.Warning;
                _frmMessage.Message = "No se puede cancelar la ficha de depósito por
las siguientes razones:".AddNewLine(2) + strMessage;
                _frmMessage.ShowDialog();

                isValidByCancel = false;
```

```
            }

            return isValidByCancel;
        }

        /// <summary>
        /// Valida la ficha de deposito para su recuperación
        /// </summary>
        /// <returns></returns>
        private bool validSelectDocumentsToDestinyRecovery()
        {
            if (this.gridView1.SelectedRowsCount > 0)
            {

                var dd = new List<vw_Sheet_Documents>();

                dd = this.getSelectedSlip();

                var ff = dd.Where(p => p.Person_Id == null).ToList();

                return (ff.Count == dd.Count);
            }
            else
            {
                return false;
            }
        }

        /// <summary>
        /// Valida la ficha de depósito para poder ser edición
        /// </summary>
        /// <param name="slipdeposit"></param>
        /// <returns></returns>
        private bool validForEdit(vw_Sheet_Documents slipdeposit)
        {
            string strmessage = "";

            if (!BR.Custom.DepositSlipBR.Instance.ValidForEdit(slipdeposit, out
strmessage))
            {
                Message.frmMessage _frmMessage = new Message.frmMessage();

                _frmMessage.WindowType = Message.frmMessage.enWindowType.Warning;
                _frmMessage.Message = "La ficha de depósito no puede ser editada por
las siguientes razones:".AddNewLine(2) + strmessage;
                _frmMessage.ShowDialog();
                return false;
            }

            return true;
        }

        /// <summary>
        /// obtiene el la form aque se va a utilizar de acuerdo al movimiento a
realizar
        /// </summary>
        /// <param name="sheetConceptId"></param>
        /// <returns></returns>
```

```
        private int getFormMovement(int sheetConceptId)
        {
            int formId = 0;
            switch ((Enumerate.SheetConcept)sheetConceptId)
            {
                case Enumerate.SheetConcept.Undefined:
                case Enumerate.SheetConcept.Deposit_Account:
                case Enumerate.SheetConcept.Charge_Account:
                    formId = 1;
                    break;
                case Enumerate.SheetConcept.Transfer:
                    formId = 3;
                    break;
                case Enumerate.SheetConcept.Interbank:
                    formId = 2;
                    break;
                case Enumerate.SheetConcept.Devolution:
                    formId = 5;
                    break;
                default:
                    break;
            }
            return formId;
        }

        /// <summary>
        /// obtiene descripcion del detalle de los movimientos acivos de una ficha
de depósito
        /// </summary>
        /// <returns></returns>
        private string getDetailInfoSlipDeposit()
        {
            return
BR.Custom.DepositSlipBR.Instance.getDetailInfoSlipDeposit(this.Item.Sheet_Document_I
d);
        }

        /// <summary>
        /// obtiene las fichas de depósito seleccionadas en el grid
        /// </summary>
        /// <returns></returns>
        private List<vw_Sheet_Documents> getSelectedSlip()
        {
            var dd = new List<vw_Sheet_Documents>();

            foreach (var row in this.gridView1.GetSelectedRows())
                dd.Add((this.gridView1.GetRow(row) as vw_Sheet_Documents));

            return dd;
        }



        #endregion

    }

}
```

Control para la Alta y Edicion de una Ficha de Depósito

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using PDN.BeMoney.BusinessObjects;
using PDN.BeMoney.BR;

namespace PDN.BeMoney.Windows.Forms.DepositSlip
{
    public partial class frmDepositSlip : Form
    {
        public frmDepositSlip()
        {
            InitializeComponent();

            this.ucDepositSlip1.OnRaisePendingChange += new
EventHandler(ucDepositSlip1_OnRaisePendinChange);

        }
        private vw_Cfg_Form_Concept _cfgForm;

        public vw_Cfg_Form_Concept cfgForm
        {
            get { return _cfgForm; }
            set
            {
                _cfgForm = value;
                if (_cfgForm != null)
                {
                    this.Text = _cfgForm.Description_Form;
                    this.lblTitle.Text = _cfgForm.Description_Form;
                    this.Name = _cfgForm.Form_Name;
                }
            }
        }

        public  List<int> listSlipSave = new List<int>();

        public event EventHandler OnRaiseSearch;

        private void tsbSave_Click(object sender, EventArgs e)
        {
            if (this.ucDepositSlip1.SaveDocumentSheet())
            {
```

```
this.listSlipSave.Add(this.ucDepositSlip1.Document_Sheet.Sheet_Document_Id);

                if (OnRaiseSearch != null)
                    OnRaiseSearch(this.ucDepositSlip1.Document_Sheet, new
EventArgs());

                if (this.ucDepositSlip1.CRUD == Enumerate.CRUD.Create)
                    setNew();
            }
            else
            {
                this.setPendingChanges(true);
            }
        }

        private void tsbExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }




        public void ucDepositSlip1_OnRaisePendinChange (object sender, EventArgs e)
        {
            this.setPendingChanges(Convert.ToBoolean(sender));
        }

        private void setPendingChanges(bool _pendingChange)
        {
            this.tsbSave.Enabled = _pendingChange;
        }

        private void tsbNew_Click(object sender, EventArgs e)
        {
            setNew();
        }

        private void setNew()
        {
            this.Cursor = Cursors.WaitCursor;

            vw_Sheet_Documents dant = null;

            if (this.ucDepositSlip1.Document_Sheet != null &&
(this.ucDepositSlip1.Document_Sheet.Sheet_Document_Id > 0))
                dant = this.ucDepositSlip1.Document_Sheet;


            this.ucDepositSlip1.Document_Sheet =
BR.Custom.DepositSlipBR.Instance.getDocumentDestiny(new vw_Sheet_Documents() {
Sheet_Concept_Id = 1 }, (int)(Enumerate.SheetMovement.SlipDeposit), 1);

            if (dant != null && dant.Sheet_Document_Id > 0)
            {
                this.ucDepositSlip1.Document_Sheet.Trust_Sub_Id = dant.Trust_Sub_Id;
```

```
                this.ucDepositSlip1.Document_Sheet.Bank_Destiny_Id =
dant.Bank_Destiny_Id;
                this.ucDepositSlip1.Document_Sheet.Bank_Account_Id =
dant.Bank_Account_Id;
                this.ucDepositSlip1.Document_Sheet.Currency_Id = dant.Currency_Id;
                this.ucDepositSlip1.Document_Sheet.Deposit_Date = DateTime.Today;
            }


            this.ucDepositSlip1.LoadDocument();
            this.setPendingChanges(false);
            this.Cursor = Cursors.Default;
        }

        private void frmDepositSlip_FormClosing(object sender, FormClosingEventArgs
e)
        {


        }



    }
}
```

Control para Agregar un Movimiento a una Ficha de Depósito

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using PDN.BeMoney.BusinessObjects;

namespace PDN.BeMoney.Windows.Forms.DepositSlip
{
    public partial class frmDepositSlipDocument : Form
    {
        public frmDepositSlipDocument()
        {
            InitializeComponent();

            this.uclistDepositSlip1._OptionForm =
Enumerate.OptionSlipDepositForm.OnlySearch;
        }


        public List<int> listSlipSave = new List<int>();
        private vw_Cfg_Form_Concept _cfgForm;
```

```
        public vw_Cfg_Form_Concept cfgForm
        {
            get { return _cfgForm; }
            set
            {
                _cfgForm = value;
                if (_cfgForm != null)
                {
                    this.Text = _cfgForm.Description_Form;
                    this.lblTitle.Text = _cfgForm.Description_Form;
                    this.Name = _cfgForm.Form_Name;
                }
            }
        }



        private void tsbSave_Click(object sender, EventArgs e)
        {
            saveDocumentSheet();
        }

        private void saveDocumentSheet()
        {
            vw_Sheet_Documents doc = null;

            if (this.ucDepositSlip2.SaveDocumentSheet())
            {

this.listSlipSave.Add(ucDepositSlip2.Document_Sheet.Sheet_Document_Id);
                RefreshDocumentOrigin();

                LoadDepositSlipDestiny();
                this.ucDepositSlip2.LoadDocument();
            }

        }

        public  void LoadDepositSlipDestiny()
        {

            //se configura Nuevo Documento
            this.ucDepositSlip2.CRUD = Enumerate.CRUD.Create;
            this.ucDepositSlip2.cfgForm = this._cfgForm;

            this.ucDepositSlip2.Document_Sheet =
BR.Custom.DepositSlipBR.Instance.getDocumentDestiny(this.ucDepositSlip1.Document_She
et, this.cfgForm.Form_Concept_Id , 1);

            this.ucDepositSlip2.Document_Sheet.Root_Document_Id =
(this.ucDepositSlip1.Document_Sheet.Root_Document_Id == -1 ||
this.ucDepositSlip1.Document_Sheet.Root_Document_Id == 0) ?
this.ucDepositSlip1.Document_Sheet.Sheet_Document_Id :
this.ucDepositSlip1.Document_Sheet.Root_Document_Id;
            this.ucDepositSlip2.Document_Sheet.Parent_Document_Id =
this.ucDepositSlip1.Document_Sheet.Sheet_Document_Id;
```

```
            this.ucDepositSlip2.Document_Sheet.Person_Id =
this.ucDepositSlip1.Document_Sheet.Person_Id;


        }

        private void RefreshDocumentOrigin()
        {

            //se actualiza el documento
            this.ucDepositSlip1.CRUD = Enumerate.CRUD.Read;
            //this.ucDepositSlip1.Document_Sheet =
BR.vw_Sheet_DocumentsBR.Instance.GetByADO(string.Format("Sheet_Document_Id = {0}",
this.ucDepositSlip1.Document_Sheet.Sheet_Document_Id)).FirstOrDefault();
            this.ucDepositSlip1.Document_Sheet =
BR.Custom.DepositSlipBR.Instance.GetBySlipDepositById(this.ucDepositSlip1.Document_S
heet.Sheet_Document_Id);
            this.ucDepositSlip1.LoadDocument();

            LoadDocumentOfDocumentOrigin();
        }

        public  void LoadDocumentOfDocumentOrigin()
        {
            //se acitualiza los documentos
            this.uclistDepositSlip1.ListItems =
BR.Custom.DepositSlipBR.Instance.GetChildDocumentsByDepositSlipId(this.ucDepositSlip
1.Document_Sheet.Sheet_Document_Id);
            this.uclistDepositSlip1.showDetailSlipDeposit = false;
            this.uclistDepositSlip1.LoadData();
        }

        private void tsbExit_Click(object sender, EventArgs e)
        {
            this.Close();
        }



    }

}
```

Control Base para una Ficha de Depósito

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using PDN.BeMoney.Windows.Forms.Peoples;
using PDN.BeMoney.BR.Kit;
using PDN.BeMoney.BusinessObjects;
using PDN.BeMoney.BR.Custom;
using PDN.BeMoney.BR;
using PDN.BeMoney.Windows.Forms.Message;


namespace PDN.BeMoney.Windows.Forms.DepositSlip
{
    public partial class ucDepositSlip : ucControlBase
    {
        public ucDepositSlip()
        {
            InitializeComponent();
            applyReadOnlyApperance();

        }


        public vw_Cfg_Form_Concept cfgForm { get; set; }

        public event EventHandler OnRaisePendingChange;

        public event EventHandler OnRaiseChangeSheetConcept;

        private List<vw_Cfg_Form_Control> _cfgformControls;

        public List<vw_Cfg_Form_Control> cfgFormControls
        {
            get { return _cfgformControls; }
            set { _cfgformControls = value; }
        }

        private bool changeTypeConcept = false;

        public  bool isNewOrigin= false ;

        public Enumerate.SheetConcept ConceptDocument =
Enumerate.SheetConcept.Undefined;

        private Enumerate.SheetConcept Prev_Concept;

        private vw_Sheet_Documents _document_Sheet;

        public vw_Sheet_Documents Document_Sheet
```

```
        {
            get { return _document_Sheet; }
            set { _document_Sheet = value; }
        }

        public string Title
        {
            get
            {
                return this.gpslipDocument.Text;
            }
            set
            {
                this.gpslipDocument.Text = value;
            }
        }

        private void applyReadOnlyApperance()
        {
            if (_crud == Enumerate.CRUD.Read)
            {
                this.gleAccountDestiny.CRUD = _crud;
                this.gleBankDestiny.CRUD = _crud;
                this.gleBankOrigin.CRUD = _crud;
                this.gleDepositConcept.CRUD = _crud;
                this.glePaymentForm.CRUD = _crud;
                this.txtAccountOrigin.CRUD = _crud;
                this.txtBankReference.CRUD = _crud;
                this.txtComments.CRUD = _crud;
                this.txtImport.CRUD = _crud;
                this.gleTrustSub.CRUD = _crud;
                this.dtDepositDate.Enabled = false;
                this.sbSearchPerson.Enabled = false;
                this.gleCurrency.CRUD = _crud;
                this.txtExchangeRate.CRUD = _crud;
            }
        }

        private List<vw_Document_Type_FD> getPaymentForm()
        {
            var pf = BR.vw_Document_Type_FDsBR.Instance.GetByADO("Is_Active = 1");

            if (!pf.Exists(p => p.Document_Type_Id == (short)10))
            {
                pf.Add(new vw_Document_Type_FD() { Document_Type_Id = 10, Is_Active
= true, Document_Type_Description = "<No Definido>", Is_Replacement = false,
Uses_Commission = false });
            }

            return pf;
        }

        private void loadCombos()
        {
            try
            {

                this.Cursor = Cursors.WaitCursor;
```

```
                SendMessageLoadCatalogue("Formas de Pago");
                this.glePaymentForm.Properties.DataSource = getPaymentForm();

                SendMessageLoadCatalogue("Origen de Cartera");
                this.gleTrustSub.Properties.DataSource = getTrustSub();

                SendMessageLoadCatalogue("Bancos Origen");
                this.gleBankOrigin.Properties.DataSource =
BR.cat_BanksBR.Instance.GetByADO("Is_Active = 1 ").OrderBy(p =>
p.Bank_Name).ToList();

                SendMessageLoadCatalogue("Bancos Destino");
                this.gleBankDestiny.Properties.DataSource = this._crud ==
Enumerate.CRUD.Read ? (new BR.Custom.vw_bank_account_extendBR()).GetBanks("") : (new
BR.Custom.vw_bank_account_extendBR()).GetBanks("Bank_Id= 0  OR  Trust_Sub_Id = 0
").OrderBy(p => p.Bank_Name).ToList();

                SendMessageLoadCatalogue("Cuentas Bancarias");
                this.gleAccountDestiny.Properties.DataSource = this._crud ==
Enumerate.CRUD.Read ? BR.vw_Bank_AccountsBR.Instance.GetAllByADO() :
BR.vw_Bank_AccountsBR.Instance.GetByADO(" Bank_Id = 0  OR  Trust_Sub_Id = 0 ");

                SendMessageLoadCatalogue("Conceptos");
                this.gleDepositConcept.Properties.DataSource =
BR.cat_Sheet_ConceptsBR.Instance.GetByADO("Is_Active = 1 AND Is_Alive = 1 AND
Document_Group_Id = 1");

                SendMessageLoadCatalogue("Divisa");
                this.gleCurrency.Properties.DataSource =
BR.cat_CurrenciesBR.Instance.GetByADO("Is_Active = 1");


                this.SendMessageOnLoadInfo("Listo.",
Enumerate.ImageStatusMessage.Info2);
                this.Cursor = Cursors.Default;

            }
            catch (Exception ex)
            {
                this.Cursor = Cursors.Default;
                this.SendMessageOnLoadInfo("Ocurrio un error en la carga de los
catálogos.", Enumerate.ImageStatusMessage._Stop);
                Message.frmMessage _frmMessage = new frmMessage();
                _frmMessage.Message = ex.Message + "-->" + ex.ToString();
                _frmMessage.WindowType = frmMessage.enWindowType.Warning;
                _frmMessage.ShowDialog();

                this.SendMessageOnLoadInfo("Listo.",
Enumerate.ImageStatusMessage.Info2);
            }
        }

        private List<cat_Trust_Sub> getTrustSub()
        {
            List<cat_Trust_Sub> d = new List<cat_Trust_Sub>();
            d = BR.cat_Trust_SubsBR.Instance.GetByADO("Is_Active = 1");
```

```
            d.Add(new cat_Trust_Sub() { Trust_Description = "<No Definido>",
Trust_Sub_Id = -1 });

            return d;
        }

        private void loadDefaultValues()
        {
            enableEventsCombos(false);

            this.gleBankOrigin.EditValue = 0;
            this.gleAccountDestiny.EditValue = 0;
            this.gleBankDestiny.EditValue = 0;
            this.gleCurrency.EditValue = 1;
            this.gleDepositConcept.EditValue = 1;
            this.glePaymentForm.EditValue = 10;
            this.gleTrustSub.EditValue = -1;
            this.txtExchangeRate.Text = string.Format("{0:c2}", 1);
            this.txtExchangeRate.CRUD = Enumerate.CRUD.Read;

            this.glePaymentForm.Focus();

            enableEventsCombos(true);
        }

        private void enableEventsCombos(bool isEnabled)
        {
            if (isEnabled)
            {
                this.gleBankDestiny.EditValueChanged += new
EventHandler(gleBankDestiny_EditValueChanged);
                this.glePaymentForm.EditValueChanged += new
EventHandler(glePaymentForm_EditValueChanged);
                this.gleTrustSub.EditValueChanged += new
EventHandler(gleTrustSub_EditValueChanged);
                this.txtAppliedAmout.EditValueChanged += new
EventHandler(txtAppliedAmout_EditValueChanged);
                this.gleCurrency.EditValueChanged += new
EventHandler(gleCurrency_EditValueChanged);
                this.gleDepositConcept.EditValueChanged+= new
EventHandler(gleDepositConcept_EditValueChanged);
                this.gleDepositConcept.EditValueChanging+=new
DevExpress.XtraEditors.Controls.ChangingEventHandler(gleDepositConcept_EditValueChan
ging);
                this.gleAccountDestiny.EditValueChanged+=new
EventHandler(gleAccountDestiny_EditValueChanged);
            }
            else
            {
                this.gleBankDestiny.EditValueChanged -= new
EventHandler(gleBankDestiny_EditValueChanged);
                this.glePaymentForm.EditValueChanged -= new
EventHandler(glePaymentForm_EditValueChanged);
                this.gleTrustSub.EditValueChanged -= new
EventHandler(gleTrustSub_EditValueChanged);
                this.txtAppliedAmout.EditValueChanged -= new
EventHandler(txtAppliedAmout_EditValueChanged);
```

```
                this.gleCurrency.EditValueChanged -= new
EventHandler(gleCurrency_EditValueChanged);
                this.gleDepositConcept.EditValueChanged -= new
EventHandler(gleDepositConcept_EditValueChanged);
                this.gleDepositConcept.EditValueChanging -= new
DevExpress.XtraEditors.Controls.ChangingEventHandler(gleDepositConcept_EditValueChan
ging);
                this.gleAccountDestiny.EditValueChanged -= new
EventHandler(gleAccountDestiny_EditValueChanged);
            }
        }

        private void loadAccountByBankDestiny()
        {
            if ((gleBankDestiny.GetSelectedDataRow() as BanksName) != null)
            {
                this.gleAccountDestiny.Properties.DataSource = (new
BR.Custom.vw_bank_account_extendBR()).GetByGroupByAccount(string.Format("Bank_Id =
{0} AND Trust_Sub_Id = {1} "

, (gleBankDestiny.GetSelectedDataRow() as BanksName).Bank_Id

, (gleBankDestiny.GetSelectedDataRow() as BanksName).Trust_Sub_Id));

                var acc = (this.gleAccountDestiny.Properties.DataSource as
List<vw_Bank_Accounts>);

                if (acc.Count == 0)
                    acc.Add(new vw_Bank_Accounts() { Account_Number = "", Bank_Id =
0, Trust_Sub_Id = -1, Is_Default = true, Currency_Id = 1 });

                this.gleAccountDestiny.EditValue = acc.FirstOrDefault(p =>
p.Is_Default) != null ? acc.FirstOrDefault(p => p.Is_Default).Bank_Account_Id :
acc.FirstOrDefault().Bank_Account_Id;

            }
            SetPendingChange(true);
        }

        private void enableControlsByPaymentForm()
        {
            var paymentForm = (glePaymentForm.GetSelectedDataRow() as
vw_Document_Type_FD);

            if ( paymentForm  != null)
            {
                if (paymentForm .Document_Type_Id == 11 || paymentForm
.Document_Type_Id == 5)
                {
                    this.txtAccountOrigin.CRUD = Enumerate.CRUD.Read;
                    this.txtBankReference.CRUD = Enumerate.CRUD.Read;
                    this.gleBankOrigin.CRUD = Enumerate.CRUD.Read;
                }
                else
                {
                    this.txtAccountOrigin.CRUD = Enumerate.CRUD.Create;
                    this.txtBankReference.CRUD = Enumerate.CRUD.Create;
                    this.gleBankOrigin.CRUD = Enumerate.CRUD.Create;
```

```
                }
            }

        }

        private void loadBanksDestinyByTrustSub()
        {

            if ((gleTrustSub.GetSelectedDataRow() as cat_Trust_Sub) != null)
            {
                string strParamAux = "";

                strParamAux = _document_Sheet.Sheet_Concept_Id ==
(int)Enumerate.SheetConcept.Interbank ? string.Format(" AND Bank_Id NOT IN({0}) ",
_document_Sheet.Bank_Id) : "";

                this.gleBankDestiny.Properties.DataSource =
BR.Custom.vw_bank_account_extendBR.Instance.GetBanks(string.Format(" Trust_Sub_Id =
{0} {1}", (gleTrustSub.GetSelectedDataRow() as cat_Trust_Sub).Trust_Sub_Id,
strParamAux)).OrderBy(p => p.Bank_Name).ToList();


                if ((this.gleBankDestiny.Properties.DataSource as
List<BanksName>).Count == 0)
                    (this.gleBankDestiny.Properties.DataSource as
List<BanksName>).Add(new BanksName() { Account_Number = "", Bank_Id = 0, Bank_Name =
"<No Definido>", Trust_Sub_Id = -1 });

                switch (this.ConceptDocument)
                {
                    case Enumerate.SheetConcept.Undefined:
                    case Enumerate.SheetConcept.Interbank:
                    case Enumerate.SheetConcept.Devolution:
                        gleBankDestiny.EditValue = 0;
                        break;
                    case Enumerate.SheetConcept.Deposit_Account:
                    case Enumerate.SheetConcept.Charge_Account:
                    case Enumerate.SheetConcept.Transfer:
                        gleBankDestiny.EditValue = this.gleBankOrigin.EditValue;
                        break;
                    default:
                        gleBankDestiny.EditValue = 0;
                        break;
                }


                this.loadAccountByBankDestiny();
            }

            SetPendingChange(true);
        }

        private void AssignClient()
        {
            frmSearchPerson _frmseachPerson = new frmSearchPerson();

            _frmseachPerson.searchClient = true;
            _frmseachPerson.ShowsIsPDNClientColumn = false;
```

```
            _frmseachPerson.SearchByContractsActives = true;
            _frmseachPerson.OnRaiseConfirmAssignPerson += new
EventHandler<BREventArgs<vw_Person>>(OnRaise_ConfirmAssignPerson);
            _frmseachPerson.ShowDialog();
            if (_frmseachPerson.ShowDialog() == DialogResult.Yes )
            {

                this.txtCommercialName.Text =
_frmseachPerson.ItemSelected.Commercial_Name;
                this._document_Sheet.Person_Id =
_frmseachPerson.ItemSelected.Person_Id;
                this._document_Sheet.Commercial_Name =
_frmseachPerson.ItemSelected.Commercial_Name;

                this.SetPendingChange(true);
            }


        }
        public void OnRaise_ConfirmAssignPerson(object sender,
BREventArgs<vw_Person> e)
        {
            if (e.Item != null)
            {
                if ((new Message.frmMessage()
                {
                    Message = string.Format(@"Esta apunto de asignar el cliente
""{0}"" a la ficha de depósito. ¿Desea continuar? ",
                                e.Item.Commercial_Name),
                    WindowType = Message.frmMessage.enWindowType.Question
                }).ShowDialog() == DialogResult.Yes)
                {
                    e.Cancel = false;
                }
                else
                {
                    e.Cancel = true;
                }
            }
        }

        private void loadOriginalDocument()
        {

            enableEventsCombos(false);

            this.glePaymentForm.EditValue = this._document_Sheet.Document_Type_Id;
            enableControlsByPaymentForm();

            this.gleTrustSub.EditValue = this._document_Sheet.Trust_Sub_Id;
            loadBanksDestinyByTrustSub();

            this.gleBankDestiny.EditValue = this._document_Sheet.Bank_Destiny_Id;
            loadAccountByBankDestiny();

            this.gleAccountDestiny.EditValue = this._document_Sheet.Bank_Account_Id;

            this.gleBankOrigin.EditValue = this._document_Sheet.Bank_Id;
```

```
            this.gleDepositConcept.EditValue =
this._document_Sheet.Sheet_Concept_Id;

            this.gleCurrency.EditValue = this._document_Sheet.Currency_Id;

            this.txtCommercialName.Text = this._document_Sheet.Person_Id > 0 ?
this._document_Sheet.Commercial_Name : "";

            this.txtFolio.Text = this._document_Sheet.Sheet_Document_Id.ToString();

            this.txtImport.Text = String.Format("{0:c2}",
this._document_Sheet.Total_Document);
            this.txtAppliedAmout.Text = String.Format("{0:c2}",
this._document_Sheet.Applied_Amount);
            this.txtPendingApplyAmount.Text = String.Format("{0:c2}",
this._document_Sheet.Pending_Amount);
            this.txtExchangeRate.Text = String.Format("{0:c2}",
this._document_Sheet.Exchange_Rate);

            this.txtAccountOrigin.Text = _document_Sheet.Account_Number_Origin;

            this.txtBankReference.Text = _document_Sheet.Sheet_Number;

            this.txtComments.Text = _document_Sheet.Comments;

            this.dtDepositDate.Text = !(_document_Sheet.Deposit_Date >=
dtDepositDate.MaxDate || _document_Sheet.Deposit_Date <= dtDepositDate.MinDate) ?
_document_Sheet.Deposit_Date.ToString("dd/MM/yyyy") : "01/01/1900";



            enableEventsCombos(true);



        }

        public void LoadDocument()
        {

            if (this._document_Sheet == null)
                this._document_Sheet = new vw_Sheet_Documents();

            this.ConceptDocument =
(Enumerate.SheetConcept)this._document_Sheet.Sheet_Concept_Id;
            switch (ConceptDocument)
            {
                case Enumerate.SheetConcept.Undefined:
                    if (this.CRUD != Enumerate.CRUD.Create)
                        loadOriginalDocument();
                    break;
                case Enumerate.SheetConcept.Deposit_Account:
                case Enumerate.SheetConcept.Charge_Account:
                case Enumerate.SheetConcept.Transfer:
                case Enumerate.SheetConcept.Interbank:
                    this._loadDocument();
                    break;
```

```
                case Enumerate.SheetConcept.Devolution:
                    break;
                default:
                    break;
            }

            //this.setReadOnly();

            if (this.CRUD == Enumerate.CRUD.Update)
            {
                this.setReadOnlyByEdit();

            }
            else
            {
                this.setReadOnlyByConfig();

                SetVisibilityLabels();

            }

        }

        private void SetVisibilityLabels()
        {


            this.lblAccountDestiny.Visible = this.gleAccountDestiny.Visible;
            this.lblAccountOrigin.Visible = this.txtAccountOrigin.Visible;
            this.lblAppliedAmount.Visible = this.txtAppliedAmout.Visible;
            this.lblBankOrigin.Visible  = this.gleBankOrigin.Visible;
            this.lblBankReference.Visible = this.txtBankReference.Visible;
            this.lblComments.Visible = this.txtComments.Visible;
            this.lblConcept.Visible = this.gleDepositConcept.Visible;
            this.lblCurrency.Visible = this.gleCurrency.Visible;
            this.lblDepositDate.Visible  = this.dtDepositDate.Visible;
            this.lblExchangeRate.Visible  = this.txtExchangeRate.Visible;
            this.lblFolio.Visible  = this.txtFolio.Visible;
            this.lblImport.Visible = this.txtImport.Visible;
            this.lblPaymentForm.Visible = this.glePaymentForm.Visible;
            this.lblPendingApplyAmount.Visible  =
this.txtPendingApplyAmount.Visible;
            this.lbltrustSub.Visible = this.gleTrustSub.Visible;



        }

        private void setReadOnlyByEdit()
        {
            this.txtFolio.CRUD = Enumerate.CRUD.Read;
            this.txtCommercialName.CRUD = Enumerate.CRUD.Read;
            this.sbSearchPerson.Enabled = true;

        }

        public void SetPendingChange(bool _pendingChange)
        {
```

```
            if (this.OnRaisePendingChange != null)
                this.OnRaisePendingChange(_pendingChange, new EventArgs());

        }

        private void SendMessageLoadCatalogue(string catalogueName)
        {
            SendMessageOnLoadInfo(string.Format("Cargando Catálogo de {0}, por favor
espere...", catalogueName), Enumerate.ImageStatusMessage.Warning);
        }

        public void SendMessageOnLoadInfo(string strMessage,
Enumerate.ImageStatusMessage imageloStatusMessage)
        {
            BR.Custom.SecurityBR.SendMessage(strMessage, imageloStatusMessage);
        }

        public bool SaveDocumentSheet()
        {
            string strMessage = "";
            bool isSave = true ;
            if (!(this._crud == Enumerate.CRUD.Read))
            {
                try
                {
                    this.SetValuesInDocument();
                    if (this.validateSlipDeposit())
                    {
                        this.ConceptDocument =
(Enumerate.SheetConcept)_document_Sheet.Sheet_Concept_Id;

                        if
(!BR.Custom.DepositSlipBR.Instance.SaveSlipDeposit(this._document_Sheet, null, null,
Enumerate.Optionsp_SheetDocuments_MainFlow.DepositSlip, out strMessage))
                            throw new Exception(strMessage);

                        this.txtFolio.Text =
_document_Sheet.Sheet_Document_Id.ToString();

                        if (this._crud == Enumerate.CRUD.Create)
                        {
                            strMessage = String.Format("Se Creo la Ficha de Depósito
{0}", _document_Sheet.Sheet_Document_Id);
                            //this._crud = Enumerate.CRUD.Update;
                        }
                        else
                        {
                            strMessage = String.Format("Se ha Actualizado la Ficha
de Depósito {0}", _document_Sheet.Sheet_Document_Id);
                        }

                        frmMessage frmmessage = new frmMessage();
                        frmmessage.Message = strMessage;
                        frmmessage.WindowType = frmMessage.enWindowType.Info;
                        frmmessage.ShowDialog();

                    }
```

```
                    else
                    {
                        isSave = false;
                    }
                }
                catch (Exception ex)
                {
                    strMessage = String.Format("Ocurrio un Error al Intentar Guardar
la Ficha de Depósito. {0} --> {1}", ex.Message, ex.ToString());
                    frmMessage frmmessage = new frmMessage();
                    frmmessage.Message = strMessage;
                    frmmessage.WindowType = frmMessage.enWindowType.Critical;
                    frmmessage.ShowDialog();
                    isSave = false;
                }
                finally
                {

                }

            }

            return isSave;

        }

        public void SetValuesInDocument()
        {

            if (this._document_Sheet == null)
                this._document_Sheet = new vw_Sheet_Documents();


            this._document_Sheet.Sheet_Document_Id =
this.txtFolio.StrValue.GetValue<int>();
            this._document_Sheet.Sheet_Concept_Id =
(this.gleDepositConcept.GetSelectedDataRow() as cat_Sheet_Concept) != null ?
(this.gleDepositConcept.GetSelectedDataRow() as cat_Sheet_Concept).Sheet_Concept_Id
: 0;
            this._document_Sheet.Document_Type_Id =
(this.glePaymentForm.GetSelectedDataRow() as vw_Document_Type_FD) != null ?
(this.glePaymentForm.GetSelectedDataRow() as vw_Document_Type_FD).Document_Type_Id :
(short)11;
            this._document_Sheet.Trust_Sub_Id =
(this.gleTrustSub.GetSelectedDataRow() as cat_Trust_Sub) != null ?
(this.gleTrustSub.GetSelectedDataRow() as cat_Trust_Sub).Trust_Sub_Id : -1;
            this._document_Sheet.Sheet_Number = this.txtBankReference.Text;
            this._document_Sheet.Deposit_Date =
this.dtDepositDate.Text.GetValue<DateTime>();
            this._document_Sheet.Transfer_Date =
this.dtDepositDate.Text.GetValue<DateTime>();
            this._document_Sheet.Bank_Id = (this.gleBankOrigin.GetSelectedDataRow()
as cat_Banks) != null ? (this.gleBankOrigin.GetSelectedDataRow() as
cat_Banks).Bank_Id : (byte)0;
            this._document_Sheet.Account_Number_Origin = this.txtAccountOrigin.Text;
            this._document_Sheet.Bank_Destiny_Id =
(this.gleBankDestiny.GetSelectedDataRow() as BanksName) != null ?
(byte)(this.gleBankDestiny.GetSelectedDataRow() as BanksName).Bank_Id : (byte)0;
```

```
            this._document_Sheet.Bank_Account_Id =
(this.gleAccountDestiny.GetSelectedDataRow() as vw_Bank_Accounts) != null ?
(this.gleAccountDestiny.GetSelectedDataRow() as vw_Bank_Accounts).Bank_Account_Id :
0;
            this._document_Sheet.Total_Document =
this.txtImport.StrValue.GetValue<decimal>();
            this._document_Sheet.Currency_Id =
(this.gleCurrency.GetSelectedDataRow() as cat_Currencies) != null ?
(this.gleCurrency.GetSelectedDataRow() as cat_Currencies).Currency_Id : (byte)0;
            this._document_Sheet.Exchange_Rate =
this.txtExchangeRate.StrValue.GetValue<decimal>();
            this._document_Sheet.Comments = this.txtComments.Text;


        }

        private void _loadDocument()
        {
            enableEventsCombos(false);

            this.glePaymentForm.EditValue = this._document_Sheet.Document_Type_Id;
            this.gleBankOrigin.EditValue = this._document_Sheet.Bank_Id;
            this.txtAccountOrigin.Text = this._document_Sheet.Account_Number_Origin;
            this.txtBankReference.Text = this._document_Sheet.Sheet_Number;
            this.gleTrustSub.EditValue = this._document_Sheet.Trust_Sub_Id;
            this.loadBanksDestinyByTrustSub();
            this.gleBankDestiny.EditValue = this._document_Sheet.Bank_Destiny_Id;
            this.loadAccountByBankDestiny();
            this.gleAccountDestiny.EditValue = this._document_Sheet.Bank_Account_Id;
            this.gleCurrency.EditValue = this._document_Sheet.Currency_Id;
            this.txtExchangeRate.Text = this._document_Sheet.Currency_Id == 2 ?
string.Format("{0:c2}", this._document_Sheet.Exchange_Rate) :
string.Format("{0:c2}", 1);

            this.gleDepositConcept.EditValue =
this._document_Sheet.Sheet_Concept_Id;
            this.ConceptDocument =
(Enumerate.SheetConcept)this._document_Sheet.Sheet_Concept_Id;

            this.txtFolio.Text = this._document_Sheet.Sheet_Document_Id.ToString();
            this.txtImport.Text = string.Format("{0:c2}",
this._document_Sheet.Pending_Amount );
            this.txtAppliedAmout .Text =string.Format("{0:c2}",
this._document_Sheet.Applied_Amount );
            this.txtPendingApplyAmount.Text = string.Format("{0:c2}",
this._document_Sheet.Pending_Amount);
            this.txtCommercialName.Text = (this._document_Sheet.Person_Id != null &&
this._document_Sheet.Person_Id > 0) ? this._document_Sheet.Commercial_Name : "";
            this.txtCommercialName.ToolTip = this.txtCommercialName.Text;
            this.txtComments.Text = ( this._document_Sheet.Sheet_Document_Id > 0) ?
this._document_Sheet.Comments : "";

            this.dtDepositDate.Value = ( this._document_Sheet.Sheet_Document_Id > 0)
? _document_Sheet.Deposit_Date : DateTime.Today;

            this.chkGenerateMovementRequest.Checked = true;
```

```
            enableEventsCombos(true);
        }

        private void setReadOnlyByConfig()
        {

            if (this._cfgformControls  != null)
            {
                foreach (var item in this._cfgformControls)
                {
                    var crtl =this.Controls.Find(item.Control_Name,
true).FirstOrDefault ()  ;

                    if (crtl != null)
                    {
                        switch (crtl .GetType().ToString () )
                        {
                            case "PDN.BeMoney.Windows.Forms.Peoples.PDNLookUpEdit":
                                (crtl as PDNLookUpEdit ).CRUD = item.Is_Editable ?
Enumerate.CRUD.Create : Enumerate.CRUD.Read;
                                (crtl as PDNLookUpEdit ).Visible = item.Is_Visible;
                                (crtl as PDNLookUpEdit).ToolTipTitle  =
item.Description_Label;
                                break ;
                            case "PDN.BeMoney.Windows.Forms.Peoples.PDNTextBox":
                                (crtl as PDNTextBox).CRUD = item.Is_Editable ?
Enumerate.CRUD.Create : Enumerate.CRUD.Read;
                                (crtl as PDNTextBox).Visible = item.Is_Visible;
                                (crtl as PDNTextBox).ToolTipTitle =
item.Description_Label;
                                break;
                            case "PDN.BeMoney.Windows.Forms.Peoples.PDNMemoEdit":
                                (crtl as PDNMemoEdit).CRUD = item.Is_Editable ?
Enumerate.CRUD.Create : Enumerate.CRUD.Read;
                                (crtl as PDNMemoEdit).Visible = item.Is_Visible;
                                (crtl as PDNMemoEdit).ToolTipTitle =
item.Description_Label;
                                break;

                            case "PDN.BeMoney.Windows.UserControls.sbAdd":
                                (crtl as UserControls.sbAdd).Enabled  =
item.Is_Editable ;
                                (crtl as UserControls.sbAdd).Visible =
item.Is_Visible;

                                break;
                            case "PDN.BeMoney.Windows.UserControls.sbCancel":
                                (crtl as UserControls.sbCancel).Enabled =
item.Is_Editable;
                                (crtl as UserControls.sbCancel).Visible =
item.Is_Visible;
                                break;
                            case "PDN.BeMoney.Windows.UserControls.sbApply" :
                                (crtl as UserControls.sbApply).Enabled =
item.Is_Editable;
                                (crtl as UserControls.sbApply).Visible =
item.Is_Visible;
                                break;
```

```
                            case "DevExpress.XtraEditors.SimpleButton":
                                (crtl as
DevExpress.XtraEditors.SimpleButton).Enabled = item.Is_Editable;
                                (crtl as
DevExpress.XtraEditors.SimpleButton).Visible = item.Is_Visible;
                                (crtl as
DevExpress.XtraEditors.SimpleButton).ToolTipTitle = item.Description_Label;
                                break;
                            case "DevExpress.XtraEditors.LabelControl":
                                (crtl as
DevExpress.XtraEditors.LabelControl).Visible = item.Is_Visible;
                                (crtl as DevExpress.XtraEditors.LabelControl).Text =
item.Description_Label ;
                                (crtl as
DevExpress.XtraEditors.LabelControl).ToolTipTitle = item.Description_Label;
                                break;
                            case "System.Windows.Forms.CheckBox":
                                (crtl as System.Windows.Forms.CheckBox).Visible =
item.Is_Visible;
                                (crtl as System.Windows.Forms.CheckBox).Text =
item.Description_Label;
                                break;
                            default:
                                break;
                        }

                    }
                }
            }

        }

        private void readOnlyNewDeposit()
        {
            this.gleDepositConcept.CRUD = Enumerate.CRUD.Read;
        }

        private void getNextFolio()
        {
            this.txtFolio.Text = (BR.Custom.DepositSlipBR.Instance.getLastFolio() +
1).ToString();
        }

        private bool validateSlipDeposit()
        {
            string strMessage = "";
            bool isValid = true;


            if (this.validateSlipDepositControls())
            {
                if
(!PDN.BeMoney.BR.Custom.DepositSlipBR.Instance.ValidateNewDestinyMovement(this.Docum
ent_Sheet, out strMessage))
                {
                    isValid = false;
                    Message.frmMessage _frmMessage = new frmMessage();
                    _frmMessage.WindowType = frmMessage.enWindowType.Warning;
```

```
                    _frmMessage.Message = strMessage;
                    _frmMessage.ShowDialog();

                }
            }
            else
            {
                isValid = false;
            }

            return isValid;
        }

        private bool validateSlipDepositControls()
        {
            string strMessage = "";

            foreach (var item in this._cfgformControls )
            {
                var crtl = this.Controls.Find(item.Control_Name,
true).FirstOrDefault();

                if (crtl != null)
                {
                    switch (crtl.GetType().ToString())
                    {
                        case "PDN.BeMoney.Windows.Forms.Peoples.PDNLookUpEdit":
                            if (item.Is_Required && item.Is_Visible)
                            {
                                if ((crtl as PDNLookUpEdit).GetSelectedDataRow() ==
null)
                                    strMessage += string.Format(" EL campo {0} es
requerido.", item.Description_Label) + '\n';
                                else if (string.IsNullOrEmpty((crtl as
PDNLookUpEdit).Text))
                                    strMessage += string.Format(" EL campo {0} es
requerido.", item.Description_Label) + '\n';
                                else if (!string.IsNullOrEmpty((crtl as
PDNLookUpEdit).Text) && (crtl as PDNLookUpEdit).Text.ToUpper().Contains("DEFINIDO"))
                                    strMessage += string.Format(" EL campo {0} es
requerido.", item.Description_Label) + '\n';

                            }
                            break;
                        case "PDN.BeMoney.Windows.Forms.Peoples.PDNTextBox":

                            if (item.Is_Required && string.IsNullOrEmpty((crtl as
PDNTextBox).Text))
                            {
                                strMessage += string.Format(" EL campo {0} es
requerido.", item.Description_Label) + '\n';
                                break;
                            }

                            if (item.Is_Required && (crtl as
PDNTextBox).StringFormat == Enumerate.StringFormat.Currency)
                            {
```

```
                                if ((crtl as
PDNTextBox).StrValue.GetValue<decimal>() == 0)
                                {
                                    strMessage += string.Format(" EL campo {0} es
requerido.", item.Description_Label) + '\n';
                                    break;
                                }
                            }
                            break;
                        case "PDN.BeMoney.Windows.Forms.Peoples.PDNMemoEdit":
                            if (item.Is_Required && string.IsNullOrEmpty((crtl as
PDNMemoEdit).Text))
                                strMessage += string.Format(" EL campo {0} es
requerido.", item.Description_Label) + '\n';
                            break;

                        default:
                            break;
                    }

                }
            }


            if (!string.IsNullOrEmpty(strMessage))
            {

                Message.frmMessage _frmMessage = new frmMessage();
                _frmMessage.WindowType = frmMessage.enWindowType.Warning;
                _frmMessage.Message = strMessage;
                _frmMessage.ShowDialog();

                return false;
            }



            return true;
        }

        private void calculatePending()
        {

            var a = this.txtAppliedAmout.StrValue.GetValue<decimal>();
            var b = this.txtImport.StrValue.GetValue<decimal>();
            var c = b - a;

            this.txtPendingApplyAmount.Text = string.Format("{0:c2}", c);

        }

        private void loadCurrencyByAccount()
        {
            var acc = (gleAccountDestiny.GetSelectedDataRow() as vw_Bank_Accounts);

            if (acc != null && acc.Currency_Id > 0)
            {
                gleCurrency.EditValue = acc.Currency_Id;
```

```
            }
            else
            {
                gleCurrency.EditValue = 1;
            }

            setExchangeRate();
        }

        private void setExchangeRate()
        {
            if ((gleCurrency.GetSelectedDataRow() as cat_Currencies).Currency_Id ==
2)
            {
                var _exchangeRate = GlobalExchangeRate.Exchange_Rate_Oficial_Daily;

                if (_exchangeRate == 0)
                {
                    frmMessage frmmessage = new frmMessage();
                    frmmessage.Message = @"El Tipo de cambio esta desactualizado. \n
Consulte con el Administrador de Tipo de Cambio para actualizarlo ";
                    frmmessage.WindowType = frmMessage.enWindowType.Info;
                    frmmessage.ShowDialog();
                    //this.txtExchangeRate.Focus();
                }
                this.txtExchangeRate.Text = string.Format("{0:c2}", _exchangeRate);
                this.txtExchangeRate.CRUD = Enumerate.CRUD.Create;
            }
            else
            {
                this.txtExchangeRate.Text = string.Format("{0:c2}", 1);
                this.txtExchangeRate.CRUD = Enumerate.CRUD.Read;
            }
        }

        private void gleBankDestiny_EditValueChanged(object sender, EventArgs e)
        {
            if (!(this._crud == Enumerate.CRUD.Read))
            {
                loadAccountByBankDestiny();
            }
            SetPendingChange(true);
        }

        private void glePaymentForm_EditValueChanged(object sender, EventArgs e)
        {
            if (!(this._crud == Enumerate.CRUD.Read))
            {
                this.enableControlsByPaymentForm();
            }
            SetPendingChange(true);
        }

        private void gleTrustSub_EditValueChanged(object sender, EventArgs e)
        {
            if (!(this._crud == Enumerate.CRUD.Read))
            {
                loadBanksDestinyByTrustSub();
```

```
            }
            SetPendingChange(true);
        }

        private void sbSearch_Click(object sender, EventArgs e)
        {
            AssignClient();
        }

        private void ucDepositSlip_Load(object sender, EventArgs e)
        {

            //Carga catalogos
            loadCombos();
            //Carga valores por defecto
            loadDefaultValues();
            //Carga Informacion de la Ficha de Depósito
            LoadDocument();
            //Aplica configuracion de Solo lectura a controles
            applyReadOnlyApperance();
            //Indica que no hay cambios pendientes
            SetPendingChange(false);
        }

        private void txtAppliedAmout_EditValueChanged(object sender, EventArgs e)
        {
            //calculatePending();
        }

        private void txtImport_EditValueChanged(object sender, EventArgs e)
        {
            calculatePending();
            SetPendingChange(true);
        }

        private void gleCurrency_EditValueChanged(object sender, EventArgs e)
        {
            setExchangeRate();
            SetPendingChange(true);
        }

        private void gleDepositConcept_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void gleDepositConcept_EditValueChanging(object sender,
DevExpress.XtraEditors.Controls.ChangingEventArgs e)
        {
            if (e.OldValue != null)
                this.Prev_Concept = (Enumerate.SheetConcept)e.OldValue;

            if (e.NewValue != null)
                this.ConceptDocument = (Enumerate.SheetConcept)e.NewValue;
        }

        private void gleAccountDestiny_EditValueChanged(object sender, EventArgs e)
        {
```

```
            gleCurrency.EditValueChanged -= new
EventHandler(gleCurrency_EditValueChanged);

            loadCurrencyByAccount();
            SetPendingChange(true);

            gleCurrency.EditValueChanged += new
EventHandler(gleCurrency_EditValueChanged);
        }

        private void gleBankOrigin_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void txtAccountOrigin_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void txtBankReference_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void txtExchangeRate_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void dtDepositDate_ValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void txtCommercialName_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }

        private void txtComments_EditValueChanged(object sender, EventArgs e)
        {
            SetPendingChange(true);
        }


    }

}
```