Especificación Técnica de la Solución

MTI-III-ETS

**Mejoras al módulo de Conciliación Bancaria, Corte de caja, Crédito y cobranza, Caja para facturación 3.3**

**Fecha de Publicación**

29/09/2017

Versión 1.0

**Documento Confidencial.**

Derechos Reservados. El contenido y formato de este documento son propiedad de Grupo Metropolitano S. A. de C. V. y es confidencial. Cualquier reproducción del todo o en parte está estrictamente prohibida sin el permiso escrito de Grupo Metropolitano, S. A. de C. V.

Este documento está sujeto a cambios. Comentarios, correcciones o preguntas deberán ser dirigidas al autor.

# Control de Cambios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versión** | **Descripción** | **Autor** | **Autoriza** | **Fecha** |
| 1.0 | Versión Inicial | Iván Trujillo López itrujillo@softgrade.mx |  | 29/09/2016 |

# Índice

[Control de Cambios 2](#_Toc496803300)

[Índice 3](#_Toc496803301)

[Propósito del Documento 5](#_Toc496803302)

[1. Identificación 5](#_Toc496803303)

[2. Descripción del Requerimiento 5](#_Toc496803304)

[3. Análisis Técnico para la Solución 6](#_Toc496803305)

[4. Diseño Técnico de la Solución 7](#_Toc496803306)

[4.1. Acciones Técnicas 7](#_Toc496803307)

[Especificaciones técnicas. 7](#_Toc496803308)

[Creación de parámetro 7](#_Toc496803309)

[Atributos 8](#_Toc496803310)

[Operaciones 8](#_Toc496803311)

[Ventana emergente de verificación carga de conciliación manual 8](#_Toc496803312)

[Validación de ingestión de archivos 11](#_Toc496803313)

[Validacion 11](#_Toc496803314)

[CuentaBancariaExisteException 12](#_Toc496803315)

[DetalleValidacion 12](#_Toc496803316)

[DocumentoReferenciaExisteException 13](#_Toc496803317)

[EncabezadoExisteException 14](#_Toc496803318)

[Exception 15](#_Toc496803319)

[FormatoExcelException 15](#_Toc496803320)

[LayoutException 16](#_Toc496803321)

[LineaVaciaException 17](#_Toc496803322)

[MontoValidoException 17](#_Toc496803323)

[ValidadorCyC 18](#_Toc496803324)

[wucCargaManualExcelCyC 23](#_Toc496803325)

[Atributos 23](#_Toc496803326)

[6.1.1 Conciliación por cantidad y referencia 25](#_Toc496803327)

[6.1.2.1 Agregar columnas a grid 28](#_Toc496803328)

[6.1.2.2 y 6.1.2.3 Inclusión de un control de filtrado de clientes y resalte de facturas 29](#_Toc496803329)

[wucBuscaClientesFacturas 29](#_Toc496803330)

[Atributos 29](#_Toc496803331)

[Operaciones 30](#_Toc496803332)

[6.1.2.4 Importación de archivo Excel 30](#_Toc496803333)

[6.1.3 Conciliación varios a uno 30](#_Toc496803334)

[6.1.3.1 Agregar columnas a grid 31](#_Toc496803335)

[6.1.3.2 y 6.1.3.3 Inclusión de un control de filtrado de clientes y resalte de facturas 31](#_Toc496803336)

[6.1.4 Conciliación Manual 32](#_Toc496803337)

[6.1.4.1 Consultar cliente o referencia 32](#_Toc496803338)

[6.1.4.2 Recuperación de pedidos y facturas saldo mayor a cero 35](#_Toc496803339)

[6.1.4.11 Carga de archivo Excel de facturas para conciliaciones manuales 42](#_Toc496803340)

[6.1.5 Generación automática de relaciones de cobranza 43](#_Toc496803341)

[Atributos 46](#_Toc496803342)

[Operaciones 47](#_Toc496803343)

[RelacionCobranza 47](#_Toc496803344)

[RelacionCobranzaDatos 48](#_Toc496803345)

[RelacionCobranzaException 50](#_Toc496803346)

[DetalleValidacion 50](#_Toc496803347)

[MovimientoCaja 51](#_Toc496803348)

[Referencia 8. Creación de tabla relación para TransBans y conciliaciones 59](#_Toc496803349)

[MovimientoCajaConciliacion 61](#_Toc496803350)

[MovimientoCajaConciliacionDatos 64](#_Toc496803351)

[TransBan 67](#_Toc496803352)

[TransBanException 69](#_Toc496803353)

[DetalleValidacion 70](#_Toc496803354)

[Transaccionalidad del proceso 73](#_Toc496803355)

[4.2. Consideraciones y Dependencias 87](#_Toc496803356)

[ Plan de Actividades 88](#_Toc496803357)

[Configuración de ambiente de desarrollo 88](#_Toc496803358)

[Instalar *runtime* Crystal Reports 88](#_Toc496803359)

[ Anexos 90](#_Toc496803360)

[.1. Tabla de Referencia de Posibles Entregables 90](#_Toc496803361)

[.2. Anexo 1 90](#_Toc496803362)

# Propósito del Documento

El objetivo de este documento es especificar las entradas, salidas y procedimientos de los que ha de consistir la funcionalidad u objeto o componente a desarrollar o modificar, con el fin de satisfacer el requerimiento del usuario.

# Identificación

|  |  |
| --- | --- |
| **N° de PEP o N° Remedy** | No aplica |
| **Tipo de Requerimiento** | *Extensión de funcionalidad.* |
| **Sistema** | Conciliación bancaria |
| **Módulo** | Conciliación |
| **Proceso** | Generación de múltiples TransBans. |
| **Descripción** | La nueva funcionalidad debe:   * Con base a los movimientos conciliados, es decir la interrelación existente entre un registro en estado de cuenta (Externo) y un documento pendiente de pago debe ser capaz de registrar ni uno, uno o múltiples TransBan (Folios de Transferencia Bancara) * Con la inclusión de esta capacidad el proceso que actualmente existe para edificios administrados (Metropoli) no debe ser afectado. * El sistema debe contar con un parámetro de tipo entero que permita establecer la cantidad de documentos relacionados a una TransBan. |

|  |  |  |
| --- | --- | --- |
| **Requerido por** |  | **Fecha**: |
| **Aprobado por** |  | **Fecha**: |

# Descripción del Requerimiento

Incorporar al sistema de conciliaciones bancarias de Grupo metropolitano los elementos necesarios para:

1. Ser capaz de generar ni uno, uno o múltiples movimientos de tipo *TransBan* asociados a un folio de conciliación bancaria.
2. Crear un parámetro que gobernará la cantidad máxima de documentos que pueden estar relacionados a una *TransBan*.
3. El parámetro deberá tener asignado un valor numérico de tipo entero, de no ser el caso la situación debe ser manejada por el sistema y reportada mediante mensaje a usuario y registro de log.
4. El valor propuesto para el parámetro de máximo de documentos es de 500, este mismo es enunciativo, más no limitativo.
5. El sistema deberá relacionar como máximo el número de documentos indicado por el parámetro del punto dos.
6. Si el máximo de documentos vinculados a una TransBan es excedido el sistema deberá sin asistencia del usuario generar una o más folios de TransBan según el escenario.
7. El código fuente de la solución se encuentra disponible en [**https://github.com/ivantrujillo79/conciliacionbancaria**](https://github.com/ivantrujillo79/conciliacionbancaria)
8. Utilice Github Desktop o Sourcetree para conectarse al repositorio
9. El código fuente a editar debe ser **sin excepción** tomado de la rama “development”

# Análisis Técnico para la Solución

De acuerdo al análisis técnico efectuado para cumplir con el requerimiento solicitado se ejecutarán las siguientes acciones:

**Creación de parámetro**

El sistema cuenta con un par de tablas para la creación y asignación de valor de parámetros, estas serán utilizadas para el objeto de este punto.

**Ventana emergente de verificación carga de conciliación manual**

Debido a que el proceso implica la carga de un archivo externo, éste deberá ser validado y a través de un formulario el usuario será informado de las incidencias o de los registros que han sido cargados a un área de staging.

**Creación de tabla relación para *TransBans* y conciliaciones**

Con la finalidad de que el sistema vincule a un determinado folio de conciliación múltiples *TransBan* se hace necesaria la creación de una tabla que permita vincular ambas entidades por medio de la “copia” de los campos que conforman la llave primaria de sus correspondientes tablas.

**Modificación del proceso de conciliación**

Una vez que la parametrización ha sido establecida y los controles de carga de datos se han ejecutado, el proceso de conciliación toma los diferentes insumos y es a partir de ellos deberá vincular dos cosas:

1. La relación entre un folio de conciliación y los folios de *TransBan*
2. La relación existente entre un folio de *TransBan* y los documentos conciliados
3. En presencia de *TransBan* con una cantidad de documentos relacionados superior al parámetro el proceso hará una segmentación de ésta en múltiples *TransBan*.

**Transaccionalidad del proceso**

El proceso de asiento de *TransBan* implica la afectación de diversas tablas principales del sistema E.R.P. SIGAMET, la afectación potencial a la consistencia de información en el mismo es un aspecto que debe evitarse, por lo mismo es que el procedimiento debe impedir dicha posibilidad a través de la implementación de transacciones breves; esto último con el fin de impedir la creación de cuellos de botella en otros procesos del negocio.

# Diseño Técnico de la Solución

## Acciones Técnicas

# Especificaciones técnicas.

## Creación de parámetro

El sistema E.R.P. SIGAMET -en lo siguiente SIGAMET- se compone de diversos módulos, de lo anterior la existencia de un catálogo que contiene los mismos, el nombre de la tabla es “Modulo”, por otra parte, cada módulo del sistema cuenta con sus parámetros, lo anterior es posible gracias a la tabla llamada “Parametro”. Ambas tablas mencionadas se esquematizan a continuación.



Fig. 1 Tablas de parametrización

El módulo de conciliaciones bancarias corresponde al id 30 y es éste el que debe ser utilizado para relacionar con el nuevo parámetro que será etiquetado como “**MaxDoctosTransBan**”.

Con el fin de obtener el valor actual de MaxDoctosTransban se deberá utilizar el método “**ValorParametro**” de la clase “SeguridadCB.Public.Parametros” que se ilustra a continuación:



Fig. 2 Clase *Parametros*

La clase *Parametros* se documenta a continuación:

# Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **lista** ArrayList  Private |  | *Predeterminado:* new ArrayList() |

Tabla 1 Atributos

# Operaciones

| **Método** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **Parametros()**  Internal |  | **DataTable** [in]listaParametros |
| **ValorParametro()** string  Public |  | **short** [in]modulo  **string** [in]nombreParametro |

Tabla 2 Operaciones

La implementación bajo ninguna circunstancia usará el procedimiento almacenado **spObtieneParametroPorModulo** de forma directa para recuperar el valor de MaxDoctosTransBan o cualquier otro parámetro que en funcionalidades futuras se requiera, esto no aplica a funcionalidades ya implementadas.

## Ventana emergente de verificación carga de conciliación manual

Como parte del requerimiento la carga de archivo es necesaria una verificación de calidad de los datos, de lo anterior debe verificarse lo siguiente:

1. Conformidad del contenido del archivo con el *layout* esperado
2. No existencia de líneas en blanco
3. Existencia y contenido del encabezado.
4. Extensión del archivo (XLS y XLSX)

En caso de que por lo menos alguna de las situaciones mencionadas se suscite el sistema deberá reportar al usuario mediante los siguientes mensajes de error.

|  |  |
| --- | --- |
| Validación | Mensaje mostrado al usuario |
| 1 | “El formato del contenido del archivo que desea cargar no cumple la especificación \n Se esperan las columnas: Documento, Cuenta y Monto” |
| 2 | “No es posible cargar el archivo ya que contiene líneas en blanco, posición: {Número de línea}” |
| 3 | “El archivo no cuenta con encabezado, por favor verifique” |
| 4 | “El archivo a cargar debe ser de formato Excel, con extensión de archivo XLS o XLSX” |

Tabla 3 Mensajes de error durante el proceso de carga

La ventana emergente de carga del archivo deberá cumplir con la siguiente apariencia:



Fig. 3 Ventana de verificación de carga

Se utilizará el control ModalPopUpExtender que ofrece asp.net el ID del control será: “**mpeCargaArchivoConciliacionManual**” para presentar y ocultar el formulario descrito en la figura número tres.

El formulario descrito en la figura tres deberá ser llamado desde la página /conciliación/DetalleConciliacion.aspx a través de un botón y una opción de menú contextual en el grid de documentos internos.

A su vez se usará un control DataGridView ofrecido por asp.net con el ID “**dgvDetalleConciliacionManual**” con el objeto de presentar detalle al usuario de los registros (documentos) que pudieron ser conciliados a través de la carga del archivo.

Una vez que el archivo elegido por el usuario se ha cargado sin incidentes el sistema deberá mostrar el siguiente mensaje de confirmación:



Fig. 4 Mensaje de confirmación de carga

Donde:

**{Número de coincidencias}**: Corresponde al valor asignado a un contador que contiene el total de registros que pudieron relacionarse con pedidos con saldo del cliente (Internos) que se obtiene a través de la propiedad “**registrosCociliados**” de la clase “**ValidadorConciliacionManual**”.

**{Número de contrato}**: Es el número del contrato asociado al pedido indicado por el usuario en el archivo importado. Existen dos casos:

1. Si el cliente buscado es de tipo padre, el mensaje deberá ser como se indica en la figura cinco.
2. Si el cliente es hijo, deberá mostrarse como se indica en la figura cuatro.



Fig. 5 Mensaje para cliente padre

# Validación de ingestión de archivos

Con el fin de garantizar que la información cargada al sistema de conciliación bancaria es consistente y que no introducirá errores en el proceso es preciso ejecutar las validaciones indicadas en la tabla número tres. Para cobrar el punto mencionado será necesaria la creación de un paquete “*Validacion*” en la librería de clases en el proyecto Conciliación.RunTime de la solución.

Las interfaces y clases a implementar para el paquete *Validacion* son las siguientes:

## Validacion

Tipo: Paquete

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Conciliacion.Migracion.Runtime

Detalle: *Creado el 05/10/2017*. *Última modificación el 05/10/2017*

**Validacion** - *(diagrama Lógico )*

Creado por: Desarollo\_Transforma el 05/10/2017

Última modificación: 05/10/2017

Versión: 1.0.



Fig. 6 Librería de clases *Validacion*

### CuentaBancariaExisteException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  CuentaBancariaExisteException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### DetalleValidacion

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Estructura en la que se contiene el detalle del resultado de una validación particular siendo exitosa o fallida.

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **VerificacionValida** bool  Public | Indica mediante true o false si el proceso concluyó exitosamente | *Predeterminado:* |
| **CodigoError** int  Public | En caso de error cada validador devolverá un número diferente (consultar cada clase) de cero, si el proceso es exitoso devuelve cero | *Predeterminado:* |
| **Mensaje** string  Public | Mensaje detallado que regresa el proceso de validación específico, tanto si se trata de un error o un éxito se espera un mensaje que detalle la situación | *Predeterminado:* |

### DocumentoReferenciaExisteException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  DocumentoReferenciaExisteException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### EncabezadoExisteException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  EncabezadoExisteException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### Exception

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase base Exception del .net Framework

Referencia:

https://msdn.microsoft.com/en-us/library/system.exception(v=vs.110).aspx

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  LineaVaciaException | Public  Exception |  |
| Generalización  Origen -> Destino | Public  LayoutException | Public  Exception |  |
| Generalización  Origen -> Destino | Public  FormatoExcelException | Public  Exception |  |
| Generalización  Origen -> Destino | Public  EncabezadoExisteException | Public  Exception |  |
| Generalización  Origen -> Destino | Public  MontoValidoException | Public  Exception |  |
| Generalización  Origen -> Destino | Public  DocumentoReferenciaExisteException | Public  Exception |  |
| Generalización  Origen -> Destino | Public  CuentaBancariaExisteException | Public  Exception |  |

### FormatoExcelException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  FormatoExcelException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### LayoutException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  LayoutException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### LineaVaciaException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  LineaVaciaException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### MontoValidoException

Tipo: **Clase** Exception

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase que hereda de Exception

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  MontoValidoException | Public  Exception |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

### ValidadorCyC

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase concreta que se encarga de la validación del archivo Excel de conciliación de ejecutivo de crédito y cobranza

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Dependencia  Origen -> Destino | Public  ValidadorCyC | Public  IValidadorExcel |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **RutaArchivo** string  Public | Ruta del archivo a validar | *Predeterminado:* |
| **NombreArchivo** string  Public | Nombre del archivo a validar una vez que se encuentre en el servidor | *Predeterminado:* |
| **TipoMime** string  Public | Se pasa el tipo MIME del archivo para validación de tipo de archivo | *Predeterminado:* |

Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **new()** ValidadorCyC  Public | Constructor por defecto |  |
| **new()** ValidadorCyC  Public | Constructor sobrecargado, requiere ruta y nombre de archivo a validar | string [in] RutaArchivo  string [in] NombreArchivo  string [in] TipoMIME |
| **ValidacionCompleta()** List<DetalleValidacion>  Public | Único método público para iniciar el proceso de validación. |  |
| **ValidaCuentaBancaria()** DetalleValidacion  Private | Método que verifica que todos los registros en el archivo corresponden a la cuenta bancaria asociada al folio de conciliación |  |
| **ValidaDocumentoReferencia()** DetalleValidacion  Private | Método que verifica la existencia del DocumentoReferencia proporcionado en el renglón |  |
| **ValidaMonto()** DetalleValidacion  Private | Verifica que el monto en el renglón:  -No esté en blanco  -No sea Nulo  -Sea numérico (dos puntos decimales)  -Sea mayor a cero |  |
| **ValidaEncabezado()** DetalleValidacion  Private | Valida mediante Hash MD5 que la concatenación sin espacios ni caracteres de finalización de los tres encabezados del archivo correspondan al hash MD5 de la cadena siguiente:  DocumentoCuentaMonto |  |
| **ValidaFormatoExcel()** DetalleValidacion  Private | string contentType = FileUpload1.PostedFile.ContentType  if (contentType == "application/vnd.ms-excel" )  {  }  else  {  LanzarError  }  Lista de tipos MIME  https://developer.mozilla.org/es/docs/Web/HTTP/Basics\_of\_HTTP/MIME\_types/Lista\_completa\_de\_tipos\_MIME |  |
| **ValidaLayout()** DetalleValidacion  Private | Verificar que las tres columnas y todas las filas contienen datos, que no hay más que tres columnas con datos y que se trata de sólo tres columnas |  |
| **ValidaLineaVacia()** DetalleValidacion  Private | Busca la existencia de líneas del archivo Excel en blanco y **REPORTA EL ID DE LAS LÍNEAS** |  |

IValidadorExcel

Tipo: **Interfaz**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Clase abstracta que funciona como fábrica de clases concretas que implementan determinada verificación de layout sobre archivos de formato Excel

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Dependencia  Origen -> Destino | Public  ValidadorCyC | Public  IValidadorExcel |  |

El proceso a ejecutar por el flujo de carga se describe en el diagrama de secuencia de la figura siete:



Fig. 7 Proceso de carga y validación de archivo de documentos internos a conciliar por cyc.

La secuencia de pasos es la siguiente:

***Conexiones***

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| **Secuencia**  new(string, string, string)  Origen -> Destino | Public  btnCargarArchivo | Public  objValidador |  |
| **Secuencia**  ValidacionCompleta()  Origen -> Destino | Public  btnCargarArchivo | Public  objValidador |  |
| **Secuencia**  ValidaFormatoExcel()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  ValidaEncabezado()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  ValidaLayout()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  ValidaLineaVacia()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  ValidaCuentaBancaria()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  ValidaDocumentoReferencia()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  ValidaMonto()  Origen -> Destino | Public  objValidador | Public  objValidador |  |
| **Secuencia**  Origen -> Destino | Public  objValidador | Public  DetalleConciliacion.aspx |  |

La prueba de concepto realizada debe ser integrada al sistema a través de un Web User Control, lo anterior se hará en la carpeta /ControlesUsuario/CargaManualExcelCyC (la carpeta no existe, crearla y cuidar de que se incluya en el control de versiones en el archivo del proyecto web). El control será llamado **“wucCargaManualExcelCyC”**.

Crear la clase en el proyecto *SitioConciliacion* bajo las siguientes especificaciones:



Fig. 8 Definición de Web User Control

### wucCargaManualExcelCyC

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: SitioConciliacion

Detalle: *Creado el* *08/10/2017*. *Última modificación el 08/10/2017*.

## Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **TotalRegistrosCargados** int  Public | Propiedad que regresa el total de registros cargados en sistema, si existe un error debe asignarle un valor de cero.  ésta propiedad sólo tiene getter, no tiene sentido que componentes externos modifiquen su valor. | *Predeterminado:* |
| **RegistrosCargados** int  Public | Lista de elementos de la clase PedidoExcelCyc con el contenido leído por el componente. | *Predeterminado:* |
| **DetalleProcesoDeCarga** List<DetalleValidacion>  Public | Regresa el detalle del proceso de validación y carga, con dicho resultado se despliegan mensajes de error o de éxito según sea el caso. | *Predeterminado:* |

La implementación de llamados a la librería de clases *ValidadorCyC* será la misma que se indicó en el diagrama de secuencia de la figura siete.

Una vez que la clase haya sido implementada instancie en el formulario de conciliaciones uno a varios que corresponde al archivo siguiente:

/Conciliacion/FormasConciliar/UnoAVarios.aspx

El nombre de la instancia del control será **wucCargaExcelCyC** tal y como se muestra en el siguiente diagrama:



Fig. 9 Flujo de llamados para carga de archivo Excel de CyC

Una vez que se han cargado los registros del archivo Excel debe determinarse por cada registro si se trata de un cliente padre o de una sucursal (Sucursal es sinónimo de cliente hijo) con la siguiente codificación de colores:

**Fondo Blanco:** los pedidos del contrato que se proporcionó.

**Fondo Azul claro:** identifica los pedidos del cliente padre.

**Fondo Verde claro:** los pedidos de las sucursales hermanas.

El procedimiento almacenado para determinar lo anterior es “SP que trabajará JC”.

**Reglas de negocio** para la conciliación aplicables son:

-Pedidos con saldo mayor a cero.

-Que el total del saldo de los pedidos elegidos por el usuario no exceda el monto del pago.

-El sistema no mantendrá saldos a favor del depósito, es decir, el total a cubrir cuadrará exactamente con el monto del depósito.

-La conciliación no será efectuada hasta que el usuario no lo indique a través del botón “Guardar” que se encuentra en la pantalla principal (grid de internos) de unoavarios.aspx, es decir, la carga no contempla el proceso de conciliación.

-El usuario podrá recargar múltiples veces el archivo para el mismo o diferentes clientes.

-Solamente se presentará información de archivos Excel completamente consistentes.

-Bajo ninguna circunstancia se permite tener pedidos de diferentes padres, esto incluye a las sucursales.

-Método que verifica que todos los registros en el archivo corresponden a la cuenta bancaria asociada al folio de conciliación

## 6.1.1 Conciliación por cantidad y referencia

La funcionalidad solicitada se refiere a la implementación de un nuevo tipo de conciliación llamada **“CONCILIACION DE INGRESOS POR COBRANZA A ABONAR PEDIDO”** y en lo particular para el algoritmo de Cantidad y referencia misma que en el catálogo TipoConciliacion tiene el ID 6, el flujo comentado se encuentra implementado en el formulario siguiente:

**FormasConciliar/CantidadYReferenciaConcuerdan.aspx**

Cree el nuevo tipo de conciliación, este deberá ser dado de alta en la tabla “TipoConciliacion” con el siguiente detalle:

|  |  |
| --- | --- |
| Identificador | Descripción |
| 6 | ‘CONCILIACION DE INGRESOS POR COBRANZA A ABONAR PEDIDO’ |

Modifique el procedimiento almacenado ***spCBCargarComboTipoConciliacion*** para que el formulario principal ***“Inicio.aspx”*** presente la nueva opción en el combo titulado “Tipo Conciliación” de acuerdo a lo presentado en el diagrama siguiente.



Fig. Flujo de carga de la pantalla principal del sistema

Modifique el código fuente localizado en el archivo CantidadYReferenciaConcuerdan.aspx.cs para remover las asignaciones de un valor fijo de 2 al miembro ***tipoConciliacion*** esta mala práctica deberá corregirse ya que actualmente está introduciendo errores en el proceso de conciliación. El valor que debe asignarse a la variable en todos casos proviene del query string llamado ***TipoConciliacion***. Adicionalmente modifique el alcance de la variable *tipoConciliacion* para que pase de *Public* a *Private*, para esta actividad válgase del siguiente diagrama.



Fig. Proceso de carga para Cantidad y Referencia concuerdan

Las tablas que los procedimientos almacenados de carga consultan son las siguientes:

|  |  |  |
| --- | --- | --- |
| spCBConciliacionPendienteExternoTipo –ED- |  | spCBConciliarArchivosPorReferencia-ST/sum- |
| Conciliacion |  | Conciliacion |
|  |  | ConciliacionArchivo |
| ConciliacionPedido |  | ConciliacionPedido |
| ConciliacionReferencia |  | ConciliacionReferencia |
| StatusConciliacion |  |  |
| Sucursal |  |  |
| TablaDestino |  | TablaDestino |
| TablaDestinoDetalle |  | TablaDestinoDetalle |
| TipoFuente |  | TipoFuente |
| TipoFuenteInformacion |  | TipoFuenteInformacion |

Tabla. Relación de tablas para procedimientos almacenados de carga

Modificar el procedimiento almacenado ***spCBConciliarArchivosPorReferencia*** para que reciba un parámetro adicional llamado @TipoConciliacion de tipo tinyint que permitirá recuperar pedidos de células distintas a la ocho.

## 6.1.2.1 Agregar columnas a grid

Agregue al grid de movimientos internos (grvPedidos / grvInternos) de la página **“/Conciliacion/FormasConciliar/UnoAVarios.aspx”**  las columnas siguientes:

1. Serie y folio de factura
2. Cliente

El flujo de carga de los grids mencionados es el siguiente:



Fig. Carga de registros internos en conciliación uno a varios

## 6.1.2.2 y 6.1.2.3 Inclusión de un control de filtrado de clientes y resalte de facturas

Para los depósitos de los que no se tenga algún número de referencia de cliente, el usuario deberá buscar los pedidos de algún cliente en particular y a su vez debe ser posible resaltar en el grid de internos una factura determinada.

Para lo anterior se construirá un web user control, la clase se llamará “**wucBuscaClientesFacturas”** y será creada en la carpeta **“/ControlesUsuario/BuscadorClienteFactura/”**.

La definición y descripción del control es la que se describe a continuación.

### wucBuscaClientesFacturas

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: SitioConciliacion

Detalle: *Creado el* *09/10/2017*. *Última modificación el 09/10/2017*.

## Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **NumeroClienteFiltrar** string  Private | Número del cliente o de referencia a ser filtrado  La referencia de cliente es igual al número de contrato concatenado con su dígito verificador. | *Predeterminado: Vacío* |
| **NumeroFacturaResaltar** string  Private | Serie y folio de factura (como se define en la tabla Factura de Sigamet) que será resaltado en el GridRelacionado | *Predeterminado: Vacío* |
| **GridRelacionado** GridView  Public | Control GridView sobre el cual se ejecutarán las tareas de filtrado de registro por cliente o resaltado por la concatenación de serie y factura | *Predeterminado: Vacío* |

## Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **FiltraCliente()** void  Private | Método privado en el que durante su ejecución los registros asignados en el datasource del grid asignado por medio de la propiedad **"GridRelacionado",** serán filtrados para sólo mostrar los pedidos relacionados con el cliente provisto por la propiedad **"NumeroClienteFiltrar".** | **string** [in]Cliente  **GridView** [in]GridRelacionado |
| **ResaltaFactura()** void  Private | Método que resaltará la columna Factura (efecto de marca textos) en el grid provisto por la propiedad **"GridRelacionado"** para todos los registros que cuenten con la cadena provista por el usuario a través de la propiedad **"NumeroFacturaResaltar"**. IMPORTANTE La factura se compone por una Serie consistente en una cadena de caracteres (máximo 10) y un Folio de tipo entero según se define en la tabla Factura de la base de datos del sistema Sigamet. | **string** [in]Factura  **GridView** [in]GridView |

Como paso final integre el componente en el formulario siguiente:

**“/Conciliacion/FormasConciliar/UnoAVarios.aspx”**

Para el caso del flujo de conciliación de uno a varios los componentes gridview que deben ser asignados a la propiedad *GridRelacionado* son los siguientes:

1. grvPedidos
2. grvInternos

La elección del grid a asignar al control corresponderá al que se esté presentando conforme al flujo elegido por el usuario, para obtener mayor referencia consulte el diagrama del punto 6.1.2.1.

## 6.1.2.4 Importación de archivo Excel

Integre el control de usuario **wucCargaManualExcelCyC** descrito en la sección “Ventana emergente de verificación carga de conciliación manual” de este mismo documento.

## 6.1.3 Conciliación varios a uno

## 6.1.3.1 Agregar columnas a grid

Agregue al grid de movimientos internos (grvPedidos / grvInternos) de la página **“/Conciliacion/FormasConciliar/VariosAUno.aspx”**  las columnas siguientes:

1. Serie y folio de factura
2. Cliente

El flujo de carga de los grids mencionados es el siguiente:



Fig. Carga de registros internos en conciliación Varios a Uno

## 6.1.3.2 y 6.1.3.3 Inclusión de un control de filtrado de clientes y resalte de facturas

Integre el control de usuario “***wucBuscaClientesFacturas*”** definido en el apartado 6.1.2.2 y 6.1.3.3 de este mismo documento en la página siguiente:

**/Conciliacion/FormasConciliar/VariosAUno.aspx**

Para el caso del flujo de conciliación de varios a uno los componentes gridview que deben ser asignados a la propiedad *GridRelacionado* son los siguientes:

1. grvPedidos
2. grvInternos

La elección del grid a asignar al control corresponderá al que se esté presentando conforme al flujo elegido por el usuario; para obtener mayor detalle a este respecto refiérase al diagrama en el punto 6.1.3.1.

## 6.1.4 Conciliación Manual

La conciliación manual es un proceso implementado en la página siguiente:

**ReportesConciliacion/ReporteConciliacionI.aspx**

La funcionalidad es accesible a través de la opción: Reporte -> Libro de Contabilidad; tal y como se muestra a continuación.



## 6.1.4.1 Consultar cliente o referencia

La tabla de SIGAMET que cuenta con la relación de clientes se llama **“Cliente”** en ella encontramos las siguientes columnas:

|  |  |
| --- | --- |
| Nombre de columna | Descripción |
| Cliente (int) | Identificador único del cliente, es equivalente al número de contrato. |
| DigitoVerificador (Smallint) | Dígito calculado y asignado a un número de contrato dterminado, su finalidad es asegurar que se efectúe una verificación del pago en ventanillas bancarias y portales de internet bancarios. |
| ClientePadre (int) | La tabla cliente tiene una relación unaria entre las columnas Cliente y ClientePadre, a partir de dicha relación se establece que:  Un cliente padre es aquel en el que su registro el valor de Cliente es igual a ClientePadre.  Un cliente hijo es aquel en el que su registro el valor de Cliente es distinto a ClientePadre. |

Tabla Columnas del catálogo de clientes

Fundamentado en lo anterior, en el grid de consulta de la conciliación manual (grvConciliacionCompartida) deberá implementarse la búsqueda de pedidos de cliente por dos diferentes métodos y prioridad:

1. Número de referencia (concatenación de cadena cliente y digitoverificador)
2. Número de cliente

Implemente la búsqueda por cliente o referencia en el método click del control “***btnBuscarPedido”*** que se llama “***btnBuscarPedido\_Click”*** ,para lograrlo es necesario crear un nuevo procedimiento almacenado llamado “**spCCLConsultaVwDatosClienteOReferencia**” que está basado en*“****spCCLConsultaVwDatosCliente***” y ***spCCLConsultaVwDatosClienteReferencia***.

Construya las clases Cliente, ClienteDatos y ClienteException que se muestran a continuación:

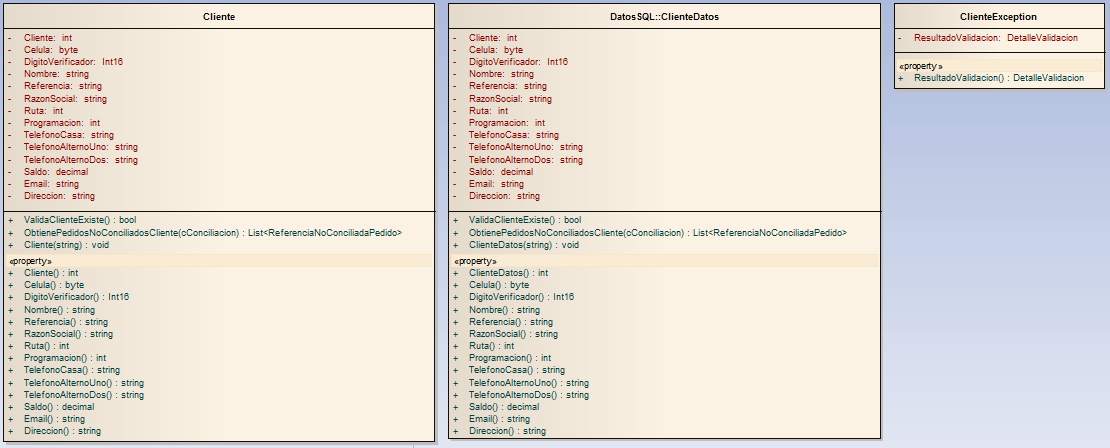


Fig. Clases Cliente, ClienteDatos y ClienteException

A continuación se incluye el código base de las clases mencionadas previamente.

|  |  |  |
| --- | --- | --- |
| Clase | Código base | Namespace |
| Cliente |  | Conciliacion.RunTime.ReglasDeNegocio |
| ClienteDatos |  | Conciliacion.RunTime.DatosSQL |
| ClienteException |  | Conciliacion.RunTime. ReglasDeNegocio |

Tabla. Código fuente base

El método constructor sobrecargado void Cliente(String) recibirá un número de referencia o de cliente y ejecutará el método ***ValidaClienteExiste*** si el parámetro devuelve un valor cliente se deberán asignar los valores de retorno a los miembros privados de la clase según correspondan (Mapeo directo entre el resultado del SP y la clase Cliente).

Implemente el método ***ValidaClienteExiste*** que ejecutará el procedimiento almacenado llamado ***spCCLConsultaVwDatosClienteOReferencia***, este recibirá el número de referencia o de cliente proporcionado y regresará un valor cierto o falso además de asignar valor a los miembros privados correspondientes dependiendo de las siguientes condiciones:

**Cierto**: Si el procedimiendo almacenado recuperó uno y sólo un valor para el cliente o referencia proporcionado

**False**: Si el procedimiendo almacenado no recuperó ni un registro para el cliente o referencia proporcionado en cuyo caso es necesario que el método arroje una excepción del tipo ***ClienteException*** con la descripción del error en la propiedad ***DetalleValidacion***.

Implemente el método ***ObtienePedidosNoConciliadosCliente*** para que ejecute al procedimiento almacenado ***spCBConciliacionBusquedaPedido,*** y el resultado se recorra registro por registro para asignar y regresar el resultado a una colección del tipo List<ReferenciaNoConciliadaPedido>.

**Criterio de aceptación:**

-La clase Cliente y ClienteDatos deberá atrapar las diferentes excepciones y transformarlas al tipo ClienteException con el debido detalle del error.

-Cuando el cliente no sea válido se deberá arrojar una excepción del tipo ClienteException informando de la situación

-Cuando el Cliente siendo válido no tenga pedidos por conciliar se deberá arrojar una excepción del tipo ClienteException informando de la situación

-Cuando el Cliente no sea válido y no tenga pedidos por conciliar se deberá arrojar una excepción del tipo ClienteException informando de la situación

-Las excepciones controlables (a través de la lógica de negocio) deberán ser llevadas hasta la capa de presentación e informadas al usuario

-Deberá proporcionar evidencia de los casos de prueba

-El código fuente deberá encontrarse en el sistema SCM (Github)

Para la integración del producto implemente el evento onKeyPress del control textbox para que reaccione al uso de la tecla *Enter*, así el usuario podrá iniciar la búsqueda mediante el botón  que se encuentra a la derecha o mediante el uso del teclado incrementando la usabilidad del producto.

## 6.1.4.2 Recuperación de pedidos y facturas saldo mayor a cero

Como primera tarea modifique el control “***btnBusquFact”*** para que su propiedad text muestre el texto “***Facturas Manuales***” tal y como se indica en el mockup siguiente:



Fig. Pantalla de búsqueda de pedidos

Adicionalmente edite la etiqueta Ped.Referencia o Pedido referencia para que en su lugar se muestre el texto “Documento”. Ahora elimine la etiqueta Pedido.

En la imagen anterior se muestra el estado inicial de la pantalla de búsqueda de pedidos para el flujo de conciliación manual, ésta pantalla tomará como criterio de restricción de los datos la referencia del cliente y del pago indicado en el formulario principal (grid del punto 6.1.4.1) que se llama “***grvConciliacionCompartida”***. Una vez que se haya determinado que se trata de un número de referencia o un número de contrato invoque el procedimiento almacenado “***spCBConciliacionBusquedaPedido”*** con el método ***Consulta\_Pedidos***; el resultado de la consulta debe mapearse como se indica a continuación.

|  |  |  |
| --- | --- | --- |
| Campo en clase receptora | Campo de SP | Status |
| Pedido | Pedido | ok |
| PedidoReferencia | Referencia | ok |
| AñoPed | AñoPed | ok |
| Celula | Celula | ok |
| Cliente | Cliente | ok |
| Nombre | Nombre | ok |
| Fsuministro | Hardcodee un valor  (Solución temporal) | Falta campo en SP |
| Total | Total | Ok |
| Concepto | Concepto | OK |
| RemisionPedido | RemisionPedido | Pendiente mapear a clase |
| FolioFacturaSerie | FolioSat | Pendiente mapear a clase |

Tabla. Mapeo de recordset a clase ReferenciaNoConciliadaPedido para pedidos

Modifique la clase ***ReferenciaNoConciliadaPedido*** para incluir las siguientes propiedades:

|  |  |
| --- | --- |
| Propiedad | Tipo de dato |
| FolioFacturaSerie | String |
| FechaFactura | Date |

Tabla. Propiedades nuevas clase ReferenciaNoConciliadaPedido

La clase receptora de los datos arrojados por el procedimiento almacenado es *“****ReferenciaNoConciliadaPedido***” una lista de dicha clase se asocia al datagridview “***grvPedidos***”. De la tabla anterior tome el resultado del procedimiento almacenado (columna Campo SP) y asígnelo al atributo correspondiente (columna Campo de la clase receptora).

Los elementos para el caso de la carga de pedidos con los que el datagridview grvPedidos debe contar para ser aceptado son los siguientes:

1. Un checkbox de selección por cada renglón
2. Columna Documento
3. Columna FSuministro
4. Columna Monto
5. Columna Remisión
6. Columna Folio factura
7. Columna Cliente
8. Columna Nombre cliente
9. Columna Concepto

Criterio de aceptación funcional:

1. Un checkbox para seleccionar todos los renglones y conciliarlos
2. Los pedidos mostrados deben pertenecer al cliente elegido por el usuario
3. Siempre deben mostrarse pedidos con saldo mayor a cero
4. El monto total de los pedidos elegidos por el usuario no debe superar, ni ser inferior que el total del depósito, es decir, el sistema no controlará saldos a favor
5. El **monto a conciliar** y el **resto** son dos elementos de la interfaz gráfica que debe actualizarse dependiendo de los pedidos elegidos por el usuario y nunca debe producir un postback, por lo tanto debe implementarse con el uso de jQuery y Javascript.
6. La ordenación de los registros se debe dar por **Cliente** y **FSuministro**
7. De acuerdo a la selección del usuario en el control **rblClienteTipo** el grid presentará pedidos del cliente elegido o de su padre, sus hermanos y él mismo, de conformidad con el siguiente diagrama.



Fig. Diagrama selección clientes normales y clientes padre

Los atributos de la clase ReferenciaNoConciliadaPedido se documentan a continuación:

***Atributos***

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **añoconciliacion** int  Private |  | *Predeterminado:* |
| **mesconciliacion** short  Private |  | *Predeterminado:* |
| **folioconciliacion** int  Private |  | *Predeterminado:* |
| **sucursalpedido** int  Private |  | *Predeterminado:* |
| **sucursalpedidodes** string  Private |  | *Predeterminado:* |
| **celulapedido** int  Private |  | *Predeterminado:* |
| **añopedido** int  Private |  | *Predeterminado:* |
| **pedido** int  Private |  | *Predeterminado:* |
| **remisionpedido** int  Private |  | *Predeterminado:* |
| **seriepedido** string  Private |  | *Predeterminado:* |
| **foliosat** int  Private |  | *Predeterminado:* |
| **seriesat** string  Private |  | *Predeterminado:* |
| **conceptopedido** string  Private |  | *Predeterminado:* |
| **total** decimal  Private |  | *Predeterminado:* |
| **statusmovimiento** string  Private |  | *Predeterminado:* |
| **selecciona** Boolean  Private |  | *Predeterminado:* |
| **cliente** int  Private |  | *Predeterminado:* |
| **nombre** string  Private |  | *Predeterminado:* |
| **pedidoreferencia** string  Private |  | *Predeterminado:* |
| **saldo** decimal  Private |  | *Predeterminado:* |

El proceso de carga de pedidos para un cliente determinado es como se esquematiza a continuación:



Fig. Proceso de carga de pedidos

Modifique la función GenerarTablaPedidos para que reciba el parámetro descrito en la siguiente tabla y se creen las columnas que requiera el caso.

|  |  |  |
| --- | --- | --- |
| Parámetro | Tipo | Notas |
| TipoDocumento | Int32 | Parámetro empleado para elegir la estructura de la tabla tblPedidos dependiendo de si se esté ejecutando el proceso de pedidos o el de facturas.  Los valores posibles del parámetro serán:  1=Cuando se trate de una carga de pedidos  2=Cuando se trate de una carga de facturas |

Así mismo, modifique la función LlenaGridViewPedidos para que reciba el parámetro descrito en la siguiente tabla y se creen las columnas que requiera el caso.

|  |  |  |
| --- | --- | --- |
| Parámetro | Tipo | Notas |
| TipoDocumento | Int32 | Parámetro empleado para ajustar de manera cosmética el DataGridView grvPedidos dependiendo de si se esté ejecutando el proceso de pedidos o el de facturas.  Los valores posibles del parámetro serán:  1=Cuando se trate de una carga de pedidos  2=Cuando se trate de una carga de facturas |

Tabla Modificación de funciones

Para el flujo de consulta y conciliación por facturas de un cliente determinado tenemos el siguiente diagrama de flujo que esquematiza y clarifica el funcionamiento esperado:



Fig. Proceso de búsqueda de facturas

Modifique el botón ***btnBusquFact*** para que una vez que el usuario haya hecho clic sobre él su propiedad text sea modificada para que muestre el texto “***Facturas***”. La pantalla deberá ajustarse para que sea mostrada conforme al siguiente mockup:



Fig. Pantalla de consulta de facturas

La consulta de facturas se realiza a través del botón ***“imgBuscarPedido”*** su método clic invoca a la función ***Consulta\_PedidosFactura*** que a su vez ejecuta al procedimiento almacenado ***“spCBConciliacionBusquedaPedidoPorFactura”*** y el criterio de restricción de registros será el siguiente: corresponde al número de cliente indicado por el usuario en el DataGridView llamado, así mismo los siguientes campos son parte del filtro:

1. grvConciliacionCompartida: Provee el número de cliente
2. txtFactura: Proporciona el número de factura

Adicionalmente es necesario incluir los siguientes cambios:

1. Modifique y refactorice el componente txtFechaFactura por txtFechaFacturaIni
2. Cree un componente de tipo TextBox que se llame txtFechaFacturaFin y vincule un calendario idéntico al del componente del punto tres

Para el caso de carga de facturas el grid grvPedidos deberá contener los siguientes elementos para ser aceptado:

1. Check box de selección por cada renglón
2. Columna FFactura
3. Columna Folio factura
4. Columna Cliente
5. Columna Nombre cliente
6. Columna Concepto
7. Columna Monto

El mapeo de los campos obtenidos desde el procedimiento almacenado hasta su presentación en pantalla es el siguiente.

|  |  |  |
| --- | --- | --- |
| Gid muestra | Clase | SP |
| FFactura | FechaFactura | Foperacion/Fmovimiento |
| Folio factura | FolioFacturaSerie | SerieSat |
| Pedido |
| FolioSat |
| Cliente | Cliente | Cliente |
| Nombre cliente | Nombre | Nombre |
| Concepto | Concepto | Concepto |
| Monto | Total | Monto/Total |

Tabla. Mapeo de recordset a clase ReferenciaNoConciliadaPedido para facturas

Criterio de aceptación funcional:

1. Un checkbox para seleccionar todos los renglones y conciliarlos
2. Las facturas mostradas deben pertenecer al cliente elegido por el usuario
3. Siempre deben mostrarse facturas con saldo mayor a cero
4. El monto total de las facturas elegidas por el usuario no debe superar, ni ser inferior que el total del depósito, es decir, el sistema no controlará saldos a favor
5. El **monto a conciliar** y el **resto** son dos elementos de la interfaz gráfica que debe actualizarse dependiendo de las facturas elegidas por el usuario y nunca debe producir un postback, por lo tanto debe implementarse con el uso de jQuery y Javascript.
6. La ordenación de los registros debe ser por **Cliente y Fecha de facturación**
7. El resultado final del proceso de conciliación debe arrojar el detalle de los documentos seleccionados en el grid *grvConciliacionCompartida* tal y como se ilustra en la siguiente imagen.



Fig. Ejemplo de conciliación manual entre un pago y múltiples pedidos del cliente 502554854

## Carga de archivo Excel de facturas para conciliaciones manuales

Cree un acordeón de Jquery UI en el que embeba el componente de carga de Excel y los grids de conciliación tal y como se muestra en las siguientes ilustraciones.



Fig. Áreas colapsables para Datagridview y Web user control

Integre el componente de carga de archivos de Excel para crédito y cobranza **wucCargaManualExcelCyC** del punto “Ventana emergente de verificación carga de conciliación manual” de este mismo documento. Es condición indispensable que el control regrese los registros leídos en una lista del tipo ***ReferenciaNoConciliadaPedido.***

***Considere los siguientes requerimientos de usabilidad:***

-El componente de carga estará oculto cuando la página se cargue, para ello embeba el web user control en un acordeón, utilice la librería jQuery UI (ya incluida en el proyecto) para su implementación, como referencia consulte: [**http://jqueryui.com/accordion/**](http://jqueryui.com/accordion/)

-Cuando el usuario haga clic en el botón el control de carga de Excel se mostrará

-La sección correspondiente a cualquiera de los datagridview (grvPedidos o grvFacturas) también queda embebida en una sección colapsable del acordeón

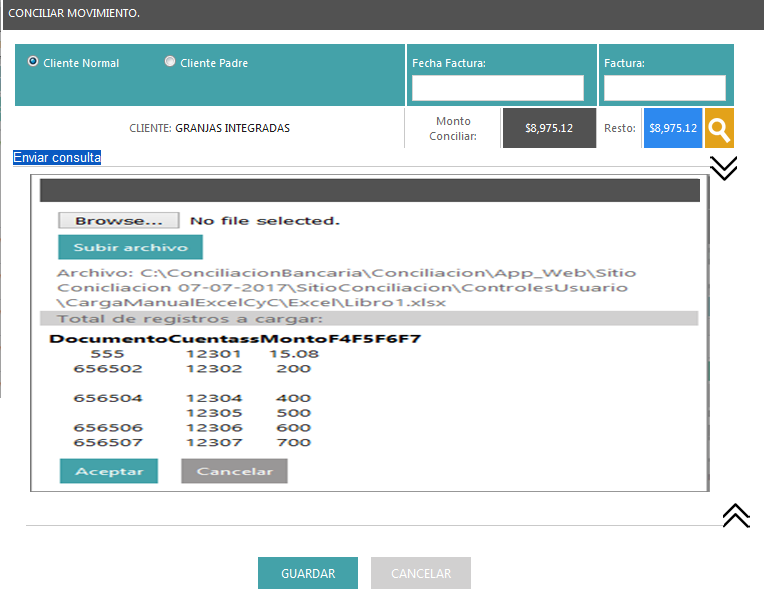


Fig. Web user control de carga de Excel visible

## 6.1.5 Generación automática de relaciones de cobranza

Para la implementación del requerimiento es necesaria la modificación de la siguiente página:

***Conciliacion/Pagos/AplicarPago.aspx***

El usuario accede a la funcionalidad mediante la búsqueda de conciliaciones en la pantalla principa del sistema, las conciliaciones deben estar en status Cerrado como se muestra a continuación (resaltado en rojo):

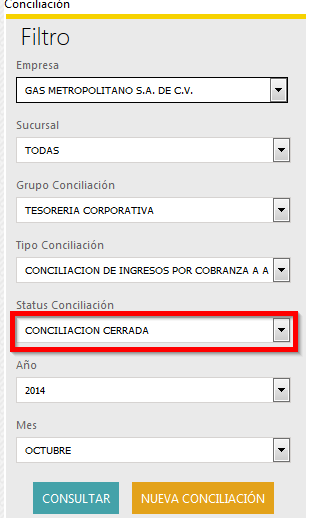


Fig. Selección de conciliaciones cerradas

Después, el usuario elige alguna de las conciliaciones con el botón derecho del mouse y elige la opción *Pagos.*

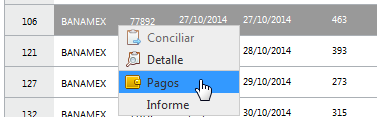


Fig. Opción de pagos

Como último paso el usuario hace clic en el botón “Aplicar Pago A Todos” que se muestra a continuación:



Fig. Botón Aplicar Pago A Todos

El objetivo del requerimiento es que el sistema genere de manera autónoma una relación de cobranza con todo el detalle que la misma requiere. El flujo se conectará a la aplicación del pago, dicha función está disponible en la página mencionada anteriormente.

Una vez que el flujo presentado en la figura siguiente haya concluido se ejecutará el proceso de generación de relaciones de cobranza.



Fig. Flujo de inicialización de la pantalla de pagos

Como primera actividad cree el parámetro “***BovedaExiste***”, relaciónelo al módulo de conciliaciones bancarias corresponde al id 30 conforme al siguiente modelo de datos.



Fig. Tablas de parametrización

Con el fin de obtener el valor actual de BovedaExiste se deberá utilizar el método “**ValorParametro**” de la clase “SeguridadCB.Public.Parametros” que se ilustra a continuación:



Fig. Clase *Parametros*

La clase *Parametros* se documenta a continuación:

# Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **lista** ArrayList  Private |  | *Predeterminado:* new ArrayList() |

Tabla Atributos clase Parametros

# Operaciones

| **Método** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **Parametros()**  Internal |  | **DataTable** [in]listaParametros |
| **ValorParametro()** string  Public |  | **short** [in]modulo  **string** [in]nombreParametro |

Tabla Operaciones clase Parametros

La implementación bajo ninguna circunstancia usará el procedimiento almacenado **spObtieneParametroPorModulo** de forma directa para recuperar el valor de BovedaExiste o cualquier otro parámetro que en funcionalidades futuras se requiera, esto no aplica a funcionalidades ya implementadas.

Después, para implementar la generación de la relación de cobranza será necesaria la creación de la clase “***RelacionCobranza”*** y “***RelacionCobranzaDatos”*** correspondientes al paquete ***“Conciliacion.Migracion.Runtime”***.

### RelacionCobranza

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: SitioConciliacion Palabras claves:

Detalle: *Creado el* *12/10/2017*. *Última modificación el 16/10/2017*.

Coordinador en la capa de negocio para la creación de relaciones de cobranza.

***Conexiones***

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| **Secuencia**  Origen -> Destino | Public  btnAplicarPagos | Public  RelacionCobranza |  |
| **Secuencia**  CreaRelacionCobranza(MovimientoCaja)  Origen -> Destino | Public  btnAplicarPagos | Public  RelacionCobranza |  |
| **Secuencia**  Origen -> Destino | Public  RelacionCobranza | Public  RelacionCobranzaDatos |  |
| **Secuencia**  CreaRelacionCobranza(MovimientoCaja)  Origen -> Destino | Public  RelacionCobranza | Public  RelacionCobranzaDatos |  |
| **Secuencia**  Origen -> Destino | Public  RelacionCobranzaDatos | Public  RelacionCobranza |  |
| **Secuencia**  Origen -> Destino | Public  RelacionCobranza | Public  btnAplicarPagos |  |

***Operaciones***

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **CreaRelacionCobranza()** void  Public | Método coordinador del proceso de generación de relación de cobranza | **MovimientoCaja** [in]MovimientoCaja  Detalle del movimiento caja (TransBan) generado por la aplicación que es el insumo para la relación de cobranza |

### RelacionCobranzaDatos

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: SitioConciliacion Palabras claves:

Detalle: *Creado el* *16/10/2017*. *Última modificación el 16/10/2017*.

Coordinador en la capa de acceso a datos para la creación de relaciones de cobranza.

***Conexiones***

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| **Secuencia**  Origen -> Destino | Public  RelacionCobranza | Public  RelacionCobranzaDatos |  |
| **Secuencia**  CreaRelacionCobranza(MovimientoCaja)  Origen -> Destino | Public  RelacionCobranza | Public  RelacionCobranzaDatos |  |
| **Secuencia**  CreaEncabezadoRelacionCobranza(MovimientoCaja)  Origen -> Destino | Public  RelacionCobranzaDatos | Public  RelacionCobranzaDatos |  |
| **Secuencia**  CreaDetalleRelacionCobranza(MovimientoCaja)  Origen -> Destino | Public  RelacionCobranzaDatos | Public  RelacionCobranzaDatos |  |
| **Secuencia**  Origen -> Destino | Public  RelacionCobranzaDatos | Public  RelacionCobranza |  |

***Operaciones***

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **CreaRelacionCobranza()** void  Public | Método coordinador del proceso de generación de relación de cobranza | **MovimientoCaja** [in]MovimientoCaja  Detalle del movimiento caja (TransBan) generado por la aplicación que es el insumo para la relación de cobranza |
| **CreaEncabezadoRelacionCobranza()** void  Private | Método encargado del proceso de generación del encabezado de la relación de cobranza, debe tomar su insumo de la propiedad "***listaCobros***" e invocará al procedimieno almacenado **spCyCCobranzaAltaModifica** | **MovimientoCaja** [in]MovimientoCaja  Detalle del movimiento caja (TransBan) generado por la aplicación que es el insumo para la relación de cobranza |
| **CreaDetalleRelacionCobranza()** void  Private | Método encargado del proceso de generación del detalle de la relación de cobranza, debe tomar su insumo de la propiedad "***listaPedidos***" e invocará al procedimieno almacenado **spCyCPedidoCobranzaAlta**  El método debe encargarse de agrupar mediante el uso de una expresión Lambda los pedidos por medio de su llave primaria miembros:  int celulapedido  int añopedido  int pedido | **MovimientoCaja** [in]MovimientoCaja  Detalle del movimiento caja (TransBan) generado por la aplicación que es el insumo para la relación de cobranza |

|  |  |
| --- | --- |
| Parámetro | Origen |
| @TipoCobranza tinyint = 1, | 10 -RELACION DE COBRANZA PARA EJECUTIVO PARA SU PLANEACION- |
| @FCobranza datetime, | GetDate/DateTime.Now |
| @UsuarioCaptura char(15), | Usuario.Usuario / Dummy = "ROPIMA" |
| @Empleado int, | Usuario.Empleado / Dummy = 9999 |
| @Total money, | MovimientoCaja.total |
| @Observaciones varchar(100), | "Relación de cobranza automática Conciliación Bancaria" |
| @Status char(10) = Null, | "CERRADO" |
| @Cobranza int = Null, | NULL |
| @SigCobranza int = Null OUTPUT, | No aplica |
| @Alta bit = 1, | 1 / default |
| @CobranzaOrigen int = null, | NULL |
| @UsuarioEntrega varchar(15) = null, | NULL |
| @StatusEntrega varchar(11) = null | NULL |

Tabla. Mapeo de propiedades (origen) a parámetro spCYCCobranzaAltaModifica

### RelacionCobranzaException

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: SitioConciliacion Palabras claves:

Detalle: *Creado el* *16/10/2017*. *Última modificación el 16/10/2017*.

Clase genérica para el control de excepciones provenientes desde el proceso de generación de relaciones de cobranza en un subcomponente determinado, su objetivo es el de proveer un alto nivel de detalle en los mensajes de error y también el de facilitar la identificación de fuentes de incidentes.

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **DetalleExcepcion** DetalleValidacion  Private |  | *Predeterminado:* |

Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **DetalleExcepcion()** DetalleValidacion  Public |  |  |

### DetalleValidacion

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 05/10/2017*.

Estructura en la que se contiene el detalle del resultado de una validación particular siendo exitosa o fallida.

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **VerificacionValida** bool  Public | Indica mediante true o false si el proceso concluyó exitosamente | *Predeterminado:* |
| **CodigoError** int  Public | En caso de error cada validador devolverá un número diferente (consultar cada clase) de cero, si el proceso es exitoso devuelve cero | *Predeterminado:* |
| **Mensaje** string  Public | Mensaje detallado que regresa el proceso de validación específico, tanto si se trata de un error o un éxito se espera un mensaje que detalle la situación | *Predeterminado:* |

### MovimientoCaja

Tipo: **Clase** EmisorMensajes

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: ReglasDeNegocio Palabras claves:

Detalle: *Creado el* *20/09/2017*. *Última modificación el 16/10/2017*.

*Conexiones*

| **Conector** | **Origen** | **Destino** | **Notas** |
| --- | --- | --- | --- |
| Generalización  Origen -> Destino | Public  MovimientoCajaDatos | Public  MovimientoCaja |  |
| Generalización  Origen -> Destino | Public  MovimientoCaja | Public  EmisorMensajes |  |
| Secuencia  Origen -> Destino | Public  btnAplicarPagos | Public  MovimientoCaja |  |
| Secuencia Guardar()  Origen -> Destino | Public  btnAplicarPagos | Public  MovimientoCaja |  |
| Secuencia  Origen -> Destino | Public  MovimientoCaja | Public  MovimientoCajaDatos |  |
| Secuencia Guardar()  Origen -> Destino | Public  MovimientoCaja | Public  MovimientoCajaDatos |  |

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **caja** short  Private |  | *Predeterminado:* |
| **foperacion** DateTime  Private |  | *Predeterminado:* |
| **consecutivo** short  Private |  | *Predeterminado:* |
| **folio** int  Private |  | *Predeterminado:* |
| **fmovimiento** DateTime  Private |  | *Predeterminado:* |
| **total** decimal  Private |  | *Predeterminado:* |
| **usuario** string  Private |  | *Predeterminado:* |
| **empleado** int  Private |  | *Predeterminado:* |
| **observaciones** string  Private |  | *Predeterminado:* |
| **saldoafavor** decimal  Private |  | *Predeterminado:* |
| **listacobros** List<Cobro>  Private |  | *Predeterminado:* new List<Cobro>() |
| **listapedidos** List<ReferenciaConciliadaPedido>  Private |  | *Predeterminado:* new List<ReferenciaConciliadaPedido>() |

Operaciones

| **Método** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **MovimientoCaja()**  Public |  | IMensajesImplementacion [in] implementadorMensajes |
| **MovimientoCaja()**  Public |  | short [in] caja  DateTime [in] foperacion  short [in] consecutivo  int [in] folio  DateTime [in] fmovimiento  decimal [in] total  string [in] usuario  int [in] empleado  string [in] observaciones  decimal [in] saldoafavor  List<Cobro> [in] listacobros  IMensajesImplementacion [in] implementadorMensajes |
| **ListaCobros()** List<Cobro>  Public |  |  |
| **ListaPedidos()** List<ReferenciaConciliadaPedido>  Public |  |  |
| **Caja()** short  Public |  |  |
| **FOperacion()** DateTime  Public |  |  |
| **Consecutivo()** short  Public |  |  |
| **Folio()** int  Public |  |  |
| **FMovimiento()** DateTime  Public |  |  |
| **Total()** decimal  Public |  |  |
| **Usuario()** string  Public |  |  |
| **Empleado()** int  Public |  |  |
| **Observaciones()** string  Public |  |  |
| **SaldoAFavor()** decimal  Public |  |  |
| **MovimientoCajaAlta()** bool  Public |  | Conexion [in] \_conexion |
| **ValidaMovimientoCaja()** bool  Public |  | Conexion [in] \_conexion |
| **AplicarCobros()** bool  Public |  | Conexion [in] \_conexion |
| **Guardar()** bool  Public |  |  |
| **CrearObjeto()** MovimientoCaja  Public |  |  |
| **CadenaConexion()** string  Public |  |  |

|  |  |
| --- | --- |
| El esqueleto para la implementación de la librería de clases puede ser tomado del siguiente archivo |  |

Implemente el flujo de creación de relaciones de cobranza a través del método click del botón llamado “***btnAplicarPagos”*** haciendo uso de las clases correspondientes y de la manera en que se ilustra a continuación:



Fig. Ejecución del proceso de creación de generación de Relaciones de cobranza

A nivel detallado los flujos de generación de pagos y de generación de relaciones de cobranza deberán integrarse del modo en que se ilustra en la siguiente figura.



Fig. Flujo para la aplicación de pago y generación de relación de cobranza

**Criterios de aceptación:**

-La relación de cobranza resultante deberá crearse con status “Relación de cobranza para ejecutivo para su planeación” (tabla TipoCobranza where TipoCobranza = 10)

-El detalle de la relación de cobranza contendrá en todos los casos el DISTINCT de todos los PedidoReferencia –Célula, AñoPedido y Pedido- asignados en la tabla Cobro y CobroPedido (TransBan)

-La relación de cobranza sólo se generará en automático si la planta tiene una bóveda (Parámetro ***BovedaExiste*** del sistema)

## Referencia 8. Creación de tabla relación para TransBans y conciliaciones

Para la implementación del requerimiento es necesaria la modificación de la siguiente página:

***Conciliacion/Pagos/AplicarPago.aspx***

El usuario accede a la funcionalidad mediante la búsqueda de conciliaciones en la pantalla principa del sistema, las conciliaciones deben estar en status Cerrado como se muestra a continuación (resaltado en rojo):

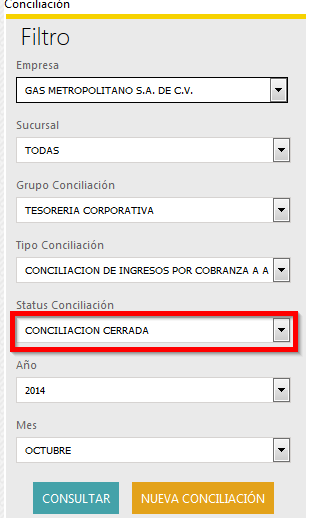


Fig. Selección de conciliaciones cerradas

Después, el usuario elige alguna de las conciliaciones con el botón derecho del mouse y elige la opción *Pagos.*

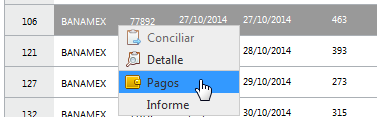


Fig. Opción de pagos

Finalmente, clic en el botón “Aplicar Pago A Todos” que se muestra a continuación:



Fig. Botón Aplicar Pago A Todos

El proceso al día de hoy sólo permite relacionar un MovimientoCaja con una Conciliación, el objetivo de este requerimiento consiste en permitir que una conciliación se componga por múltiples MovimientosCaja (TransBans).

Cree las tablas MovimientoCajaConciliacion y StatusMovimientoCajaConciliacion, el script de generación de las tablas MovimientoCajaConciliacion y StatusMovimientoCajaConciliacion es el siguiente:

IF EXISTS (SELECT \* FROM dbo.sysobjects WHERE id = object\_id('MovimientoCajaConciliacion') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)

DROP TABLE MovimientoCajaConciliacion;

CREATE TABLE MovimientoCajaConciliacion ( Caja tinyint NOT NULL,

FOperacion datetime NOT NULL,

Consecutivo tinyint NOT NULL,

Folio int NOT NULL,

CorporativoConciliacion tinyint NOT NULL,

SucursalConciliacion tinyint NOT NULL,

AñoConciliacion int NOT NULL,

MesConciliacion smallint NOT NULL,

FolioConciliacion int NOT NULL,

Status tinyint NOT NULL);

ALTER TABLE MovimientoCajaConciliacion ADD CONSTRAINT PK\_MovimientoCajaConciliacion

PRIMARY KEY CLUSTERED (Caja, FOperacion , Consecutivo, Folio, CorporativoConciliacion, SucursalConciliacion, AñoConciliacion, MesConciliacion, FolioConciliacion);

IF EXISTS (SELECT \* FROM dbo.sysobjects WHERE id = object\_id('StatusMovimientoCajaConciliacion') AND OBJECTPROPERTY(id, 'IsUserTable') = 1)

DROP TABLE StatusMovimientoCajaConciliacion;

CREATE TABLE StatusMovimientoCajaConciliacion ( Status tinyint NOT NULL,

Descripcion varchar(50) NOT NULL);

ALTER TABLE StatusMovimientoCajaConciliacion ADD CONSTRAINT PK\_StatusMovimientoCajaConciliacion

PRIMARY KEY CLUSTERED (Status);

Como referencia, la estructura de las tablas creadas por el script anterior es como se muestra a continuación:



Fig. Tabla MovimientoCajaConciliacion y catálogo de status

Crear las clases ***MovimientoCajaConciliacion*** y ***MovimientoCajaConciliacionDatos***

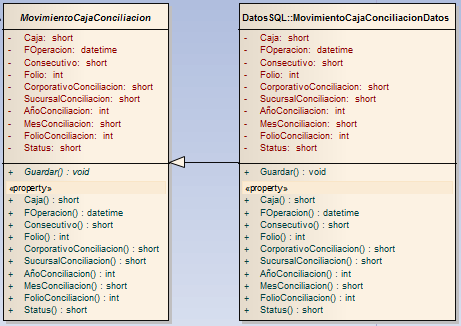


Fig. Modelo de clases

### MovimientoCajaConciliacion

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: ReglasDeNegocio Palabras claves:

Detalle: *Creado el* *18/10/2017*. *Última modificación el 18/10/2017*.

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **Caja** short  Private |  | *Predeterminado:* |
| **FOperacion** datetime  Private |  | *Predeterminado:* |
| **Consecutivo** short  Private |  | *Predeterminado:* |
| **Folio** int  Private |  | *Predeterminado:* |
| **CorporativoConciliacion** short  Private |  | *Predeterminado:* |
| **SucursalConciliacion** short  Private |  | *Predeterminado:* |
| **AñoConciliacion** int  Private |  | *Predeterminado:* |
| **MesConciliacion** short  Private |  | *Predeterminado:* |
| **FolioConciliacion** int  Private |  | *Predeterminado:* |
| **Status** short  Private |  | *Predeterminado:* |

Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **Guardar()** void  Public |  |  |
| **Caja()** short  Public |  |  |
| **FOperacion()** datetime  Public |  |  |
| **Consecutivo()** short  Public |  |  |
| **Folio()** int  Public |  |  |
| **CorporativoConciliacion()** short  Public |  |  |
| **SucursalConciliacion()** short  Public |  |  |
| **AñoConciliacion()** int  Public |  |  |
| **MesConciliacion()** short  Public |  |  |
| **FolioConciliacion()** int  Public |  |  |
| **Status()** short  Public |  |  |

### MovimientoCajaConciliacionDatos

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: DatosSQL Palabras claves:

Detalle: *Creado el* *18/10/2017*. *Última modificación el 18/10/2017*.

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **Caja** short  Private |  | *Predeterminado:* |
| **FOperacion** datetime  Private |  | *Predeterminado:* |
| **Consecutivo** short  Private |  | *Predeterminado:* |
| **Folio** int  Private |  | *Predeterminado:* |
| **CorporativoConciliacion** short  Private |  | *Predeterminado:* |
| **SucursalConciliacion** short  Private |  | *Predeterminado:* |
| **AñoConciliacion** int  Private |  | *Predeterminado:* |
| **MesConciliacion** short  Private |  | *Predeterminado:* |
| **FolioConciliacion** int  Private |  | *Predeterminado:* |
| **Status** short  Private |  | *Predeterminado:* |

Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **Guardar()** void  Public | Invoca al procedimiento almacenado de la base de datos SIGAMET llamado **spCBMovimientoCajaConciliacionAlta**, el mapeo entre los parámetros y los miembros de clase es directa. |  |
| **Caja()** short  Public |  |  |
| **FOperacion()** datetime  Public |  |  |
| **Consecutivo()** short  Public |  |  |
| **Folio()** int  Public |  |  |
| **CorporativoConciliacion()** short  Public |  |  |
| **SucursalConciliacion()** short  Public |  |  |
| **AñoConciliacion()** int  Public |  |  |
| **MesConciliacion()** short  Public |  |  |
| **FolioConciliacion()** int  Public |  |  |
| **Status()** short  Public |  |  |

El código fuente base de ambas clases se provee a continuación.

|  |  |
| --- | --- |
| Clase | Código base |
| MovimientoCajaConciliacion |  |
| MovimientoCajaConciliacionDatos |  |

Tabla. Código base

### TransBan

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: ReglasDeNegocio Palabras claves:

Detalle: *Creado el* *19/10/2017*. *Última modificación el 19/10/2017*.

Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **ReorganizaTransban()** List<MovimientoCaja>  Public | Método que divide una TransBan en varias (las que sean necesarias) a partir de una instancia de **MovimientoCaja**,  cuando se excede el máximo de documentos permitidos.  El método es responsable de verificar que los movimientos de un cliente se registren en un mismo folio de TransBan, en caso de que un cliente tenga más documentos respecto a lo parametrizado deberá lanzar una excepción del tipo TransBanException y detener el proceso. En este caso la interfaz de usuario mostrará el siguiente mensaje "No es posible generar una Transban con más de {valor parámetro MaxDoctos} documentos asociados, por favor modifique el el parámetro MaxDoctos del sistema". | MovimientoCaja [in] TransBan  int [in] MaxDocumentos |

El proceso de reorganización de transbans consiste en la recepción de una instancia del tipo MovimientoCaja, para ello habrá que contabilizar la cantidad de pedidos, posteriormente determinar si se excedió el máximo de documentos, si ese fuera el caso entonces se agrupan los pedidos por número de cliente en una estructura intermedia.

Posteriormente se verifica si alguno de los clientes cuenta con más pedidos que el valor del parámetro, si ese fuera el caso se lanza una excepción del tipo TransBanException, en caso contrario se crea una lista del tipo MovimientoCaja cuyas instancias consistirán en grupos de clientes y pedidos que no superen el total paramétrico.

**Flujo alternativo:**

Una TransBan es válida siempre que todos los movimientos (documentos) de un cliente estén contenidas, es decir, es incorrecto que un cliente quede seccionado en dos diferentes TransBans (refiérase a la siguiente figura). Por lo anterior, ante un cliente que tenga más documentos que los establecidos por el valor actual de MaxDocumentos se debe suspender el proceso y notificar al usuario mediante el siguiente mensaje:

No es posible generar una Transban con más de {valor parámetro} documentos asociados, por favor modifique el el parámetro MaxDoctos del sistema.

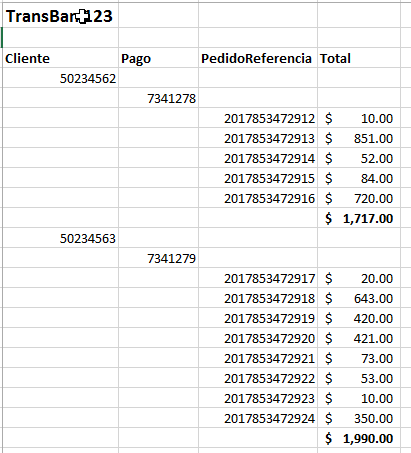


Fig. Transban válida

El proceso de reorganización de TransBans se ilustra en el siguiente diagrama de actividad:



Fig. Reorganización de TransBans

### TransBanException

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: ReglasDeNegocio Palabras claves:

Detalle: *Creado el* *19/10/2017*. *Última modificación el 19/10/2017*.

Clase que hereda de Exception y reporta errores en el proceso de gestión de TransBans

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **ResultadoValidacion** DetalleValidacion  Private | Instancia de la clase DetalleValidacion con el detalle del error | *Predeterminado:* |

Operaciones

| **Metodo** | **Notas** | **Parámetros** |
| --- | --- | --- |
| **ResultadoValidacion()** DetalleValidacion  Public | Instancia de la clase DetalleValidacion con el detalle del error |  |

### DetalleValidacion

Tipo: **Clase**

Estado: . *Versión* 1.0. *Fase* 1.0.

Paquete: Validacion Palabras claves:

Detalle: *Creado el* *05/10/2017*. *Última modificación el 16/10/2017*.

Estructura en la que se contiene el detalle del resultado de una validación particular siendo exitosa o fallida.

Atributos

| **Atributo** | **Notas** | **Restricciones y etiquetas** |
| --- | --- | --- |
| **VerificacionValida** bool  Public | Indica mediante true o false si el proceso concluyó exitosamente | *Predeterminado:* |
| **CodigoError** int  Public | En caso de error cada validador devolverá un número diferente (consultar cada clase) de cero, si el proceso es exitoso devuelve cero | Predeterminado: |
| **Mensaje** string  Public | Mensaje detallado que regresa el proceso de validación específico, tanto si se trata de un error o un éxito se espera un mensaje que detalle la situación | Predeterminado: |

|  |  |
| --- | --- |
| Clase | Código base |
| TransBan |  |
| TransBanException |  |
| DetalleValidacion |  |

Tabla. Código fuente base

El proceso de reorganización de TransBans deberá integrarse de la siguiente manera:

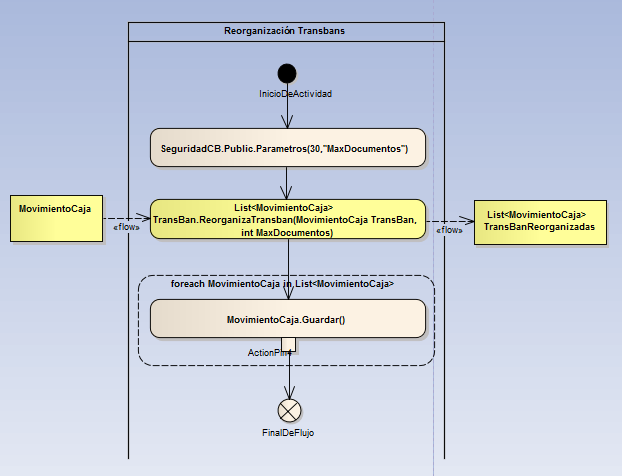


Fig. Integración del proceso de reorganización de TransBans

Por último, el proceso de relación del MovimientoCaja o TransBan con una Conciliación determinada quedará integrado como se describe en el siguiente diagrama de actividad.

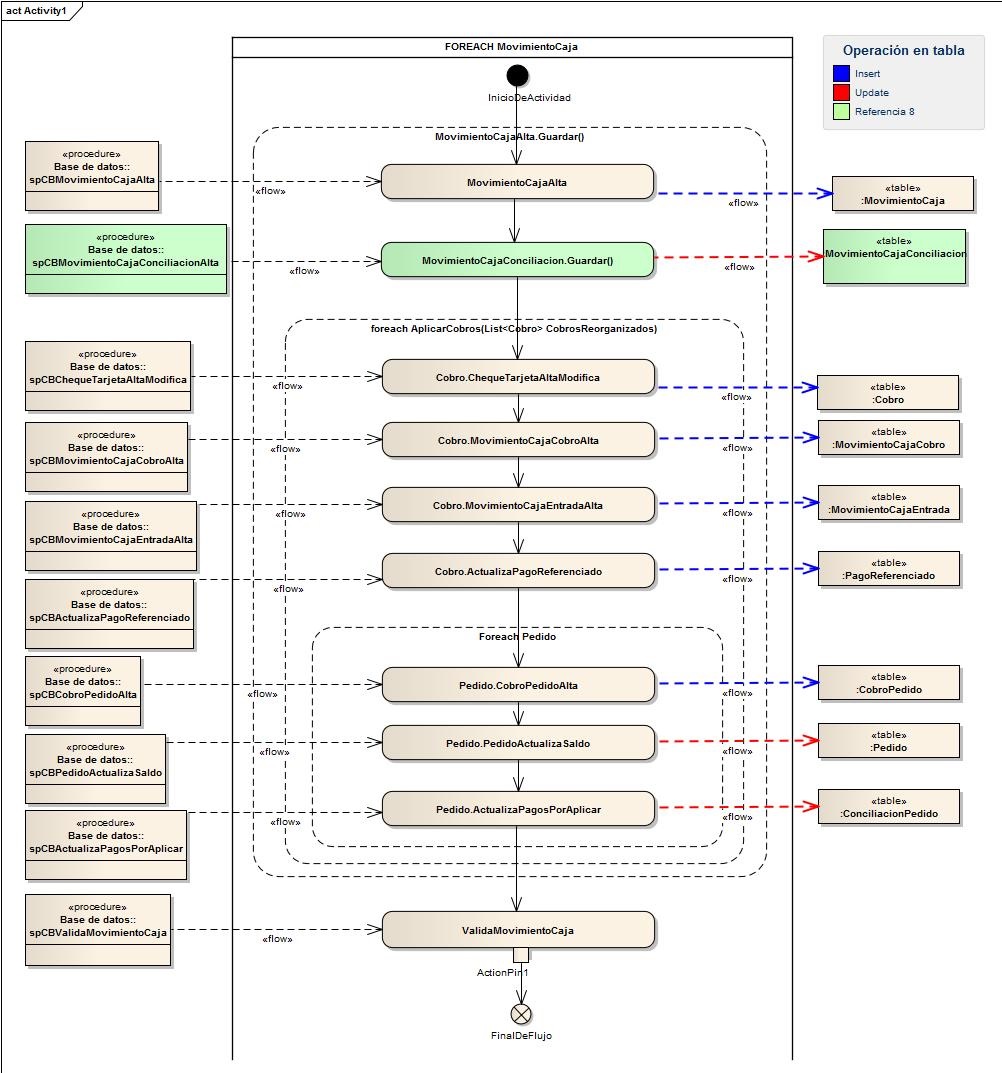


Fig. Process To-Be

**Criterios de aceptación:**

* Una conciliación podrá tener más de una TransBan asociada
* A una conciliación se le podrá generar el número de TransBans que sean necesarias aunque esta no se encuentre en status de cerrado
* Ni una TransBan podrá tener más documentos de los que el parámetro MaxDocumentos tenga en el momento de la generación
* Los docuemtnos de un cliente no deben estar dispersos en diferentes TransBans
* Si un cliente llegara a tener más documentos que el máximo establecido por el parámetro MaxDocumentos el sistema debe detener el proceso e informar al usuario con el correspondiente mensaje de error.

## Transaccionalidad del proceso

Durante el flujo de aplicación de pagos, intervienen cinco tablas clave de SIGAMET, estas se enlistan a continuación:

1. Cobro
2. CobroPedido
3. MovimientoCajaCobro
4. Movimiento Caja
5. Pedido

Es fundamental considerar que durante la aplicación del pago es posible que el proceso falle y de lugar a inconsistencias de datos, por lo anterior es fundamental



**Referencia 5. Pago parcial**

**Actividad 1**

Crear las clases **DispersorPago**, **PagoPropuesto** y **DispersorPagoException** (considerar que la clase **ValidacionDetalle** ya existe en el namespace Conciliacion.Migracion.RunTime) en el namespace Conciliacion.RunTime tal y como se ilustra a continuación:

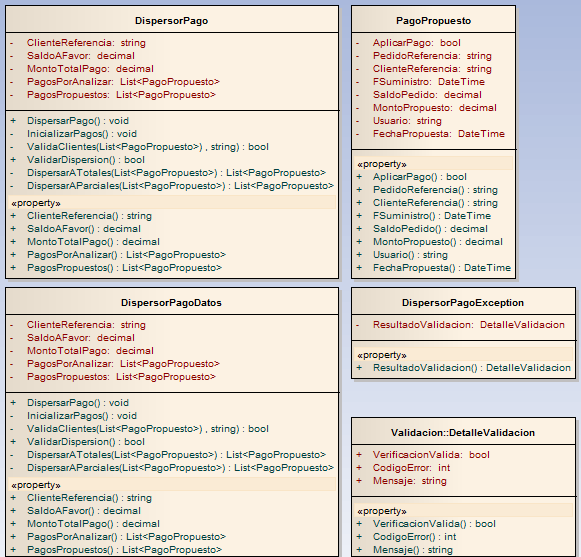


Fig. Modelo de clases para el pago parcial

El código base de las clases mencionadas es el siguiente:

|  |  |
| --- | --- |
| Clase | Código base |
| DispersorPago |  |
| DispersorPagoDatos |  |
| PagoPropuesto |  |
| DispersorPagoException |  |

Tabla. Código base para el pago parcial

El propósito de la clase ***DispersorPago*** es el de proponer a partir de un monto dado como pago (propiedad ***MontoTotalPago***) la dispersión del mismo entre una lista de documentos con saldo diferente de cero (propiedad ***PedidosPorPagar***). De lo anterior, a una instancia de esta clase contará con las siguientes propiedades:

Propiedades

| **Propiedad** | **Notas** |
| --- | --- |
| **ClienteReferencia** string  Private | Éstevalor será utilizado para verificar que el Cliente de cada uno de los documentos por analizar respeta la relación Cliente Padre – Cliente Hijo –Cliente Hermano, de esto se encargará el método **ValidaClientes**. |
| **SaldoAFavor** decimal  Private | Monto remanente de MontoTotalPago tras la dispersión de los pagos entre los PedidosPorPagar |
| **MontoTotalPago** decimal  Private | Monto del pago que la clase dispersará entre los PedidosPorPagar |
| **PedidosPorPagar** List<PagoPropuesto>  Private | Insumo de documentos propuestos para pago con un monto determinado |
| **PagosPropuestos** List<PagoPropuesto>  Private | Listado de documentos con una propuesta de dispersión tras la ejecución del método DispersarPago |

Tabla. Propiedades clase DispersorPago

Métodos

| **Método** | **Notas** |
| --- | --- |
| **DispersarPago()** void  Public | Método encargado de tomar los PedidosPorPagar, realizar la dispersión del pago y asignar el resultado a la propiedad PagosPropuestos. La propuesta considera el monto del pago, la fecha de suministro y saldo de los pedido tal y como se describe en la sección “Reglas de dispersión de pagos”. |
| **ValidaClientes(**List<PagoPropuesto>, string) void  Private | Método que verifica que la relación cliente padre, hijo y hermano se cumpla.Ésta rutina se ejecuta implícitamente como parte de la dispersión de pagos y se le pasan los pedidos a validar y el número de referencia de cliente proporcionada por el usuario. |
| **ValidarDispersion()** bool  Public | En caso de que el usuario realice una propuesta de dispersión usaremos este método para verificar que es correcta considerando que la diferencia entre el monto del pago y la sumatoria del pago propuesto por el usuario.  **Devuelve True si:**  Monto de pago - ∑Pago propuesto >= 0  **Devuelve False si:**  Monto de pago - ∑Pago propuesto < 0 |

Tabla. Métodos de la clase DispersorPago

**Reglas de disperisón de pagos**

El método DispersarPago tomará como entrada a PedidosPorPagar y ejecutará la rutina ValidaClientes, si el resultado es verdadero procede a dispersar por dos métodos.

**Pago completo**

Ordenar los pedidos de PedidosPorPagar por fecha de suministro ascendentemente y el saldo del pedido descendentemente, el resultado asignarlo a una lista de paso de la cual consultar el primer elemento y verificar que el Monto del pago sea suficiente para cubrirlo, de ser el caso:

**Procedimiento A**

1. Actualizar SaldoAFavor = MontoTotalPago;
2. Asignar a TRUE la propiedad ***AplicarPago*** de la instancia saldada.
3. Asignar en la propiedad ***MontoPropuesto*** el monto a saldar (igual al valor de la propiedad SaldoPedido)
4. Verificar que la diferencia entre el valor de MontoPropuesto y de SaldoPedido sea igual a cero.
5. Actualice la propiedad saldo a favor de la siguiente manera:

***SaldoAFavor*** -= MontoPropuesto;

En caso de que el monto del pago no sea suficiente para saldar el pedido:

**Procedimiento B**

1. No ejecute acción y pase al siguiente elemento repitiendo el procedimiento A.

Los procedimientos A y B se repiten hasta recorrer todos los elementos de la colección.

**Pago Parcial**

Para una dispersión dada podría suceder que tras aplicar todos los pagos totales aún existan documentos con saldo diferente a cero y saldo a favor mayor a cero, en dichos casos el método deberá:

1. Validar que ***SaldoAFavor*** sea mayor a cero, si esto es cierto:
2. Tomar el resultante del ***PagoCompleto*** y filtrar los elementos obteniendo todas las instancias que el valor de su propiedad ***AplicarPago*** sea *FALSE*
3. Si el paso anterior tiene instancias ordénelas por fecha de suministro de forma descendente y asigne el resultado a una variable de paso.
4. Tomar el primer elemento de la colección y comprobar que el **SaldoAFavor** es menor a **SaldoPedido.**
5. Salde el pedido.
6. Asigne TRUE a la propiedad AplicarPago .
7. Actualice el saldo a favor.
8. Continúe con el resto de los elementos de la colección.

El proceso descrito se ilustra en la figura siguiente.

**Criterios de aceptación**

-El dispersor deberá validar correctamente que referencia del cliente proporcionado y los pedidos por analizar respeten la relación padre, hijo, hermano.

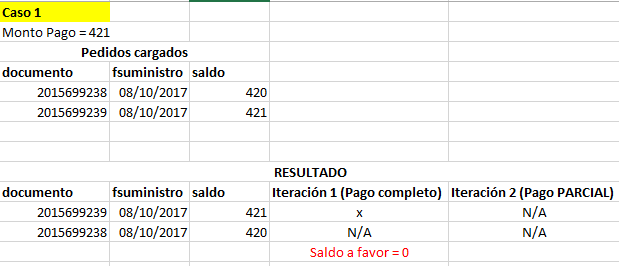
-En caso de pagos que exceden el total de los pagos propuestos el saldo a favor debe ser reflejado correctamente a través de la propiedad destinada a ese fin

-Los pagos propuestos debe observar la fecha de suministro de la forma esperada

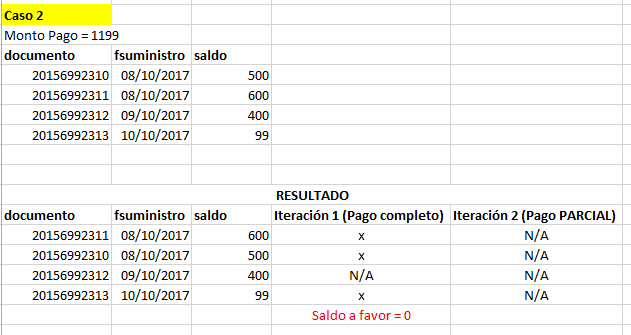
-Todos los métodos deben controlar las condiciones de error e informar a la capa superior mediante instancias de DispersorPagoException con el mensaje de error y valores de retorno de la clase DetalleValidación de forma consistente.

-Las excepciones deben generar todo el detalle de error en una sola ejecución y proporcionar el detalle en una lista de DetalleValidacion

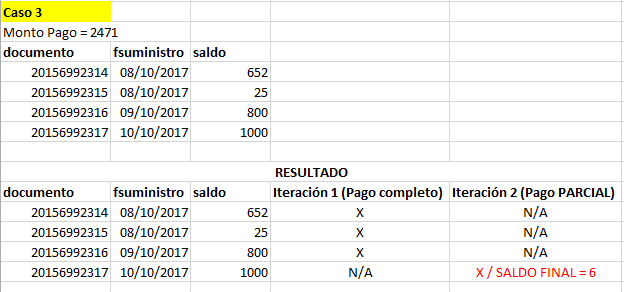
**Pruebas unitarias:**

****

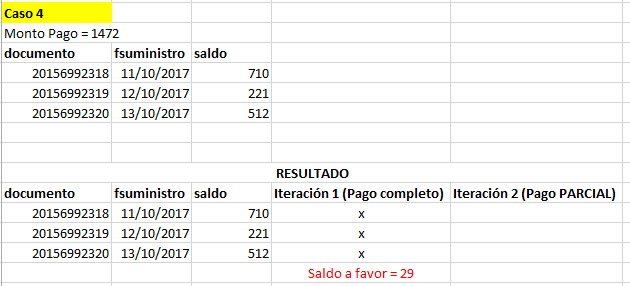
Caso de prueba 1

****

Caso de prueba 2



Caso de prueba 3



Caso de prueba 4



Fig. Diagrama de secuencia implementación de dispersiones de pago

**Método ValidaClientes**

El método tomará el valor de la propiedad ***PedidosPorPagar*** y buscará el distinct de la propiedad ***ClienteReferencia*** usando una List<string> ListaClientes auxiliar a la cual se le agregará como nuevo elemento el valor de la propiedad ***DispersorPago.ClienteReferencia*** y ejecutará el procedimiento almacenado ***spCBFamiliaresCliente*** al que se le proporcione como valor de parámetro DispersorPago.ClienteReferencia. El resultado será comparado y si en ListaClientes se encuentra un elemento no existente en el resultado del stored procedure se lanza una excepción de tipo DispersorPagoException con el **código de error 150** y mensaje:

“Los clientes {Listado de clientes erroneos} no tienen parentesco con el cliente {DispersorPago.ClienteReferencia}, revise los pedidos: {Listado de PedidoReferencia erroneos} ”

**Criterios de aceptación**

-En caso de que todos los clientes de los pedidos proporcionados tengan parentesco con el método devolverá TRUE.

-Si existe al menos un pedido con cliente sin parentesco el método devolverá FALSE.

-Si existe al menos un pedido con cliente sin parentesco el método lanzará una instancia de DispersorPagoException.

-Si un pedido no tiene cliente el método lanzará una instancia de DispersorPagoException.

-Si el cliente de prueba no existe el método lanzará una instancia de DispersorPagoException.

-Si alguno de los clientes no exite el método lanzará una instancia de DispersorPagoException.

**Método ValidarDispersion**

En caso de que el usuario realice una propuesta de dispersión usaremos este método para verificar que es correcta considerando que la diferencia entre el monto del pago y la sumatoria del pago propuesto por el usuario.

El método recorre uno a uno los elementos de la propiedad ***PedidosPorPagar*** y recupera los elementos con la propiedad ***AplicarPago*** igual a *TRUE,* el resultado se asigna a una lista de paso y se calcula la sumatoria de la propiedad ***MontoPropuesto*** aplicando la siguiente regla:

**Devuelve True si:**

Monto de pago - ∑Pago propuesto == 0

**Devuelve False si:**

Monto de pago - ∑Pago propuesto != 0

**Integración del dispersor de pagos**

La dispersión de pago y pagos parciales es un proceso que se integrará con el proceso de carga de archivos Excel. Es decir, el sistema deberá ser capaz de proponer automáticamente una opción de dipersión para el monto del pago a procesar y los pedidos cargados a través de un archivo. Debido a lo anterior existen dos escenarios de operación que se específican en la tabla siguiente:

|  |  |  |
| --- | --- | --- |
| Escenario | Uno a Varios | Manual |
| El pago cuenta con el dato de referencia de cliente | X | X |
| El pago no cuenta con el dato de referencia de cliente | X |  |

Tabla. Escenarios de integración

Para integrar la funcionalidad es necesario lo siguiente:

1. Crear la propiedad ***ClienteReferencia*** en ***wucCargaManualExcelCyC*** del tipo string y que cuando sea asignada refresque el textbox ***txtClienteReferencia***
2. Crear una etiqueta ***txtClienteReferencia*** en ***wucCargaManualExcelCyC*** que mostrará el valor asignado a la propiedad ***ClienteReferencia***
3. Crear una etiqueta ***txtSaldoAFavor*** en ***wucCargaManualExcelCyC***
4. Crear la propiedad ***MontoPago*** de tipo decimal en ***wucCargaManualExcelCyC***
5. Crear la etiqueta ***txtMontoPago*** en ***wucCargaManualExcelCyC***
6. Agregar las columnas ***AplicarPago*** de tipo checkbox, ***saldo*** tipo moneda y ***montoapagar*** de tipo moneda que se mostrarán en ***grvDetalleConciliacionManual***.

**Con número de referencia de cliente disponible**

Cuando el pago cuenta con el número de referencia del cliente, el control de carga de archivos Excel (***wucCargaManualExcelCyC***) se comportará de la siguiente forma:

1. Asignar la propiedad ***ClienteReferencia*** de la instancia de ***wucCargaManualExcelCyC***.
2. Mostrar el valor de la propiedad ***ClienteReferencia*** en ***txtClienteReferencia.Text.***
3. Asignar la propiedad ***wucCargaManualExcelCyC.MontoPago*** con el monto total del pago elegido por el usuario.
4. Mostrar el valor de la propiedad ***MontoPago*** en ***txtMontoPago*** en formato de moneda.
5. Permitir que el usuario cargue el archivo Excel y las validaciones de carga se ejecuten.
   1. Si el paso anterior no arroja errores ir a seis.
   2. Si el paso arroja errores regresar a cinco.
6. Recorrer los pedidos cargados y crear instancias de ***PagoPropuesto*** (usando su constructor) mediante el uso del procedimento almacenado ***spCBConsultaPedidoReferencia*** y acumular en una lista del mismo tipo, propuesta: ***List<PagoPropuesto> PedidosPorPagar = new List<PagoPropuesto>();***.
7. Crear una instancia de ***DispersorPago.***.
8. Asigne el valor de la propiedad ***DispersorPago.MontoPago*** con el valor de ***wucCargaManualExcelCyC.MontoPago.***
9. Asigne el resultado del paso seis a la propiedad ***DispersorPago.PedidosPorPagar***.
10. Ejecute el método ***DispersorPago.DispersarPago***.
    1. En caso de error se deberá interceptar una instancia de ***DispersorPagoException***, recuperar el detalle del error y presentar en pantalla los detalles del fallo. Después destruya la instancia de ***DispersorPago*** y finalice el flujo, en caso contrario pase a once.
11. Recupere en ***wucCargaManualExcelCyC*** el valor de la propiedad ***DispersorPago*** .***PagosPropuestos*** que son del tipo List<PagoPropuesto>.
12. Convertir la lista del punto anterior a un ***System.Data.DataTable*** con un mapeo directo a las propiedades de la clase ***PagoPropuesto***, es decir:

|  |  |
| --- | --- |
| Columna | Tipo |
| AplicarPago | Bool |
| PedidoReferencia | String |
| ClienteReferencia | String |
| Fsuministro | DateTime |
| SaldoPedido | Decimal |
| MontoPropuesto | Decimal |
| Usuario | String |
| FechaPropuesta | DateTime |

Tabla. Mapeo a DataTable

1. Recuperar el valor de la propiedad ***DispersorPago.SaldoAFavor*** y mostrar el resultado en ***txtSaldoAFavor*** aplicando formato de moneda.
2. Asignar a Null la propiedad DataSource del gridview.
3. Asignar la DataTable a la propiedad DataSource de ***grvDetalleConciliacionManual***.
4. Ejecurtar ***grvDetalleConciliacionManual***.DataBind();.

**Sin número de referencia de cliente disponible**

1. Mostrar en el control ***txtClienteReferencia*** el texto“El pago no tiene referencia” resaltado en color naranja, sugerencia: ***if(this.ClienteReferencia == “”)***.
2. Asignar la propiedad ***wucCargaManualExcelCyC.MontoPago*** con el monto total del pago elegido por el usuario.
3. Mostrar el valor de la propiedad ***MontoPago*** en ***txtMontoPago*** en formato de moneda.
4. Permitir que el usuario cargue el archivo Excel y las validaciones de carga se ejecuten.
   1. Si el paso anterior no arroja errores pasar a cinco.
   2. Si el paso arroja errores regresar a cuatro.
5. Recorrer los pedidos cargados y crear instancias de ***PagoPropuesto*** (usando su constructor) mediante el uso del procedimento almacenado ***spCBConsultaPedidoReferencia*** y acumular en una lista del mismo tipo, propuesta: ***List<PagoPropuesto> PedidosPorPagar = new List<PagoPropuesto>();***.
6. Obtener un distinct de la propiedad ***ClienteReferencia*** a partir de la lista generada en el paso cinco.
7. Crear una instancia de la clase ***DispersorPago.***
8. Asignar a la propiedad ***DispersorPago.ClienteReferencia*** con el primer valor encontrado en el paso seis.
9. Asignar a la propiedad ***DispersorPago.PedidosPorPagar*** con el resultado del paso cinco.
10. Asigne el valor de la propiedad ***DispersorPago.MontoPago*** con el valor de ***wucCargaManualExcelCyC.MontoPago.***
11. Ejecute el método ***DispersorPago.DispersarPago***.
    1. En caso de error se deberá interceptar una instancia de DispersorPagoException, recuperar el detalle del error y presentar en pantalla los detalles del fallo. Después destruya la instancia de ***DispersorPago*** y finalice el flujo, en caso contrario pase a doce.
12. Recupere en ***wucCargaManualExcelCyC*** el valor de la propiedad ***PagosPropuestos*** que son del tipo List<PagoPropuesto>.
13. Convertir la lista del punto anterior a un ***System.Data.DataTable*** con un mapeo directo a las propiedades de la clase ***PagoPropuesto***, es decir:

|  |  |
| --- | --- |
| Columna | Tipo |
| AplicarPago | Bool |
| PedidoReferencia | String |
| ClienteReferencia | String |
| Fsuministro | DateTime |
| SaldoPedido | Decimal |
| MontoPropuesto | Decimal |
| Usuario | String |
| FechaPropuesta | DateTime |

Tabla. Mapeo a DataTable

1. Recuperar el valor de la propiedad ***DispersorPago.SaldoAFavor*** y mostrar el resultado en ***txtSaldoAFavor*** aplicando formato de moneda.
2. Asignar a Null la propiedad DataSource del gridview.
3. Asignar la DataTable a la propiedad DataSource de ***grvDetalleConciliacionManual***.
4. Ejecurtar ***grvDetalleConciliacionManual***.DataBind();.

***Criterios de aceptación***

-La funcionalidad deberá ser integrada en uno a varios y conciliación manual

-Las reglas de validación de archivos Excel deben cumplirse

-El resultado de la propuesta de pagos debe ser reflejado en el gridview mediante el valor correcto en las columnas AplicarPago y MontoPropuesto.

-El saldo a favor debe actualizarse correctamente con cada cambio que el usuario realice

**Validación de propuesta de usuario**

En virtud de que el sistema permitirá al usuario seleccionar pedidos que no son parte de la propuesta hecha por el sistema. En ese caso, cuando se detecte el evento onclick en el gridview y se haya seleccionado o desseleccionado un checkbox se correrá la siguiente secuencia:

1. Se crea una instancia de ***DispersorPago.***
2. Se asigna la propiedad ***ClienteReferencia*** de estar disponible, en caso contrario ignorar este punto.
3. Se asigna la propiedad ***MontoTotalPago***
4. Se asigna el listado de ***PedidosPorPagar***
5. Se ejecuta el método ***ValidarDispersion***
   1. En caso de error se presenta el mensaje de error

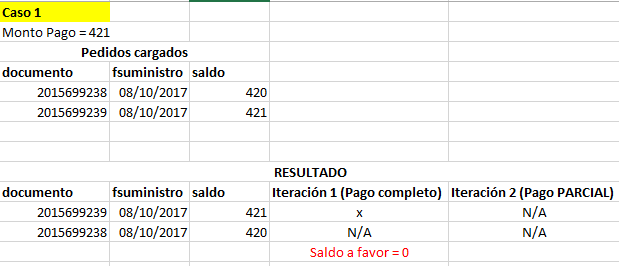
**Criterios de aceptación**

-Es fundamental que el usuario no cause una situación con saldo a favor

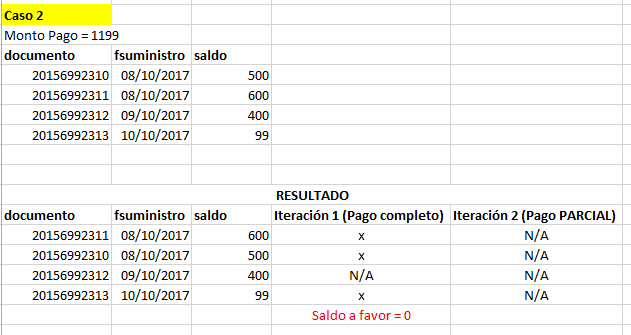
-Los mensajes de error deberán presentar el detalle al usuario

-Utilice los siguientes casos de prueba

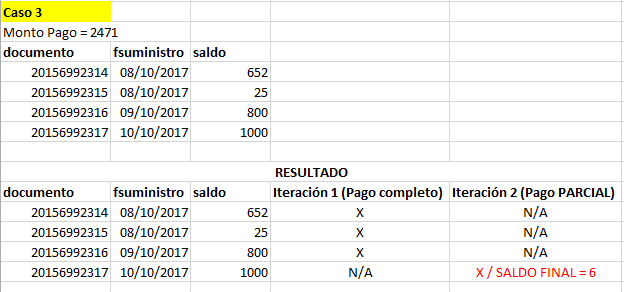
**Pruebas de integración**

****

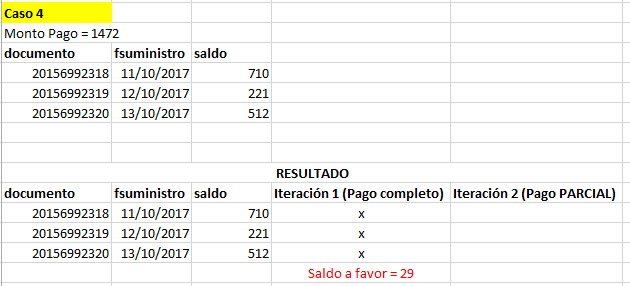
Caso de integración 1

****

Caso de integración 2



Caso de integración 3



Caso de integración 4

## Consideraciones y Dependencias

* Configuración de ambiente de desarrollo
* Configuración de ambiente de aseguramiento de la calidad (QA)
* Configuración de ambiente de UAT
* Actualización de base de datos hasta la versión más reciente proporcionada por Transforma
* Pendientes los casos de prueba y criterios de aceptación

# Plan de Actividades

|  |  |  |
| --- | --- | --- |
| Actividad | Duración | Nombre de recurso |
| **Tiempo total** | **días** |  |
| **Documentación** | **días** |  |
| MTI-III-ETP (Especificación Técnicas de Programa) | 24 hrs | Iván Trujillo |
| MTI-III-ETS (Especificación Técnica de la Solución) | 24 hrs | Iván Trujillo |
| Configuración de ambiente de desarrollo |  |  |
| Instalación de sistema operativo | 2 hrs | Ricardo Rojas |
| Instalación de SQL Server | 2 hrs | Ricardo Rojas |
| Proporcionar respaldos | 2 hrs | Iván Trujillo |
| Restaurar bases de datos Sigamet y SigametFinanciero | 4 hrs | Ricardo Rojas |
| Ejecución de script de actualización | 0 hrs | Ricardo Rojas |
| Configurar servidor vinculado | .5 hrs | Ricardo Rojas |
| Instalar .net framework Ver. 4 | .5 hrs | Ricardo Rojas |
| Instalar *runtime* Crystal Reports | .5 hrs | Ricardo Rojas |
| **Configuración de ambiente de UAT** |  |  |
| Clonado de máquina virtual | 2 hrs | Rubén Morera |
| Instalación de sistema operativo | 2 hrs | Rubén Morera |
| Instalación de SQL Server | 2 hrs | Rubén Morera |
| Proporcionar respaldos | 2 hrs | Rubén Morera |
| Restaurar bases de datos Sigamet y SigametFinanciero | 4 hrs | Rubén Morera |
| Ejecución de script de actualización | 0 hrs | Rubén Morera |
| Configurar servidor vinculado | .5 hrs | Rubén Morera |
| Instalar .net framework Ver. 4 | .5 hrs | Rubén Morera |
| Instalar *runtime* Crystal Reports | .5 hrs | Rubén Morera |
| **Desarrollo** |  |  |
| **Creación de parámetro** |  |  |
| Ejecución de script para creación de parámetro y verificación de recuperación en .net | .75 hrs | Rubén Morera |
| **Ventana emergente de verificación carga de conciliación manual** |  |  |
| Búsqueda de documentos referencia por cliente padre e hijo | 3 hrs | Ricardo Rojas |
| Filtrado por factura | 1 hrs | Ricardo Rojas |
| Creación de formulario (layout) e integración con formulario principal | 16 hrs | Rubén Morera |
| Creación de clase de validación de carga | 16 hrs | Jorge |
| Función de carga de archivo de documentos | 4 hrs | Rubén Morera |
| Rutina de conciliación (vinculación del documento referencia) | 24 hrs | Transforma |
| Integración del entregable | 1 hrs | Rubén Morera |
| Integración del entregable | 1 hrs | Ricardo Rojas |
| Prueba de integración | 1 hrs | Iván Trujillo |
| **6.1.3 Conciliación varios a uno** |  |  |
| Implementar relación por referencia a un pedido (por cliente) Contrato + Dígito verificador |  |  |
| Filtro de fecha de suministro |  |  |
| Agregar al grid de Internos, Serie, Folio factura y número de cliente |  |  |
| Agregar búsqueda de pedido por folio factura |  |  |
| Agregar búsqueda por cliente |  |  |
| Quitar la columna de Ped en Internos |  |  |
| Modificar columna Pedido referencia por Documento en Archivos internos |  |  |
| Integración del entregable | 1 hrs | Rubén Morera |
| Integración del entregable | 1 hrs | Ricardo Rojas |
| Prueba de integración | 1 hrs | Iván Trujillo |
| **6.1.4 Conciliación manual** |  |  |
| Modificación a la vista del conciliador manual |  |  |
| Implementar formulario de búsqueda de créditos |  |  |
| Implementar búsqueda del cliente |  |  |
| Implementar búsqueda caso no referenciado |  |  |
| Implementar búsqueda de pedidos por rango de fecha de suministro |  |  |
| Implementar filtrado por folio de factura |  |  |
| Integrar la importación de archivos (Función reutilizada) |  |  |
| Implementar formulario de búsqueda de facturas manuales |  |  |
| Creación de procedimientos almacenados |  | Transforma |
| **6.1.5 Generación automática de relaciones de cobranza** |  |  |
| Análisis y diseño |  |  |
| Eliminación de controles implementados para evitar múltiples *TRANSBAN* |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Pruebas unitarias |  |  |
| Integración del producto |  |  |
| Pruebas funcionales |  |  |
|  |  |  |
|  |  |  |
| **Pruebas integrales del producto** |  |  |
| **Integración del producto** |  |  |
| **Liberación del producto** |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Anexos

## Tabla de Referencia de Posibles Entregables

|  |  |  |
| --- | --- | --- |
| **X** | **ID** | **Nombre** |
|  | 1 | SolucionConciliacion (paquete de liberación) |
|  | 2 | Manual de usuario |
|  | 3 | Manual técnico |
|  | 4 |  |

## Anexo 1

No aplica