UCAB Universidad Católica
ANDRÉS BELLO

Facultad de Ingeniería
Escuela de Ingeniería Informática

Diseñar, Desarrollar e Implementar los módulos de Facturación, Punto de venta, Tesorería, Lectura de Archivos y Seguridad del Sistema de Soporte a Servicios de Cobranza Electrónica SIPAQUI.

(APÉNDICES)

REALIZADO POR: Sergio De Freitas

TUTOR EMPRESARIAL: Antonio Castro

TUTOR ACADÉMICO: Gloria Tarrío

FECHA: Febrero, 2015

TABLA DE CONTENIDO

APÉNDICE 1: Diagrama E-R.....1

1.1 Diagrama E-R Versión 1.....2

1.2 Diagrama E-R Versión 2.....5

1.3 Diagrama E-R Versión 3.....8

1.4 Diagrama E-R Versión 4.....11

1.5 Diagrama E-R Versión 5.....14

1.5 Diagrama E-R Merchant.....17

APÉNDICE 2: Descripcion de Tablas de Base de Datos.....19

APÉNDICE 3: Controladores, Modelos y Vistas.....36

APÉNDICE 4: Reportes y Archivos.....80

APÉNDICE 5: Java Vs PHP.....91

A.5.1 Lenguaje Java.....92

A.5.2 Lenguaje PHP.....102

INDICE DE FIGURAS

Figura 35. E-R Versión 1 Parte 1.....3

Figura 36. E-R Versión 1 Parte 2.....4

Figura 37. E-R Versión 2 Parte 1.....6

Figura 38. E-R Versión 2 Parte 2.....7

Figura 39. E-R Versión 3 Parte 1.....9

Figura 40. E-R Versión 3 Parte 2.....10

Figura 41. E-R Versión 4 Parte 1.....12

Figura 42. E-R Versión 4 Parte 2..... 13

Figura 43. E-R Versión 5 Parte 1.....15

Figura 44. E-R Versión 5 Parte 2.....16

Figura 45. E-R Merchant.....18

Figura 46. Archivo Excel BanPlus.....81

Figura 47. Archivo Excel Banesco.....81

Figura 48. Archivo Excel Banco del Tesoro.....82

Figura 49. Correo de la Tarea TaskIntegridad.....83

Figura 50. Reporte Conciliación.....84

Figura 51. Reporte Conciliación Conciliadas.....85

Figura 52. Reporte Conciliar Parcialmente Conciliadas.....86

Figura 53. Reporte Conciliar No Conciliadas.....87

Figura 54. Reporte Proceso Cuadre I.....88

Figura 55. Reporte Proceso Cuadre II Parte 1.....89

Figura 56. Reporte Proceso Cuadre II Parte 2.....90

Figura 57. Tabla Comparativa Java Vs PHP.....105

Apéndice 1

(Diagrama E - R)

1.1 Diagrama E - R

Versión 1

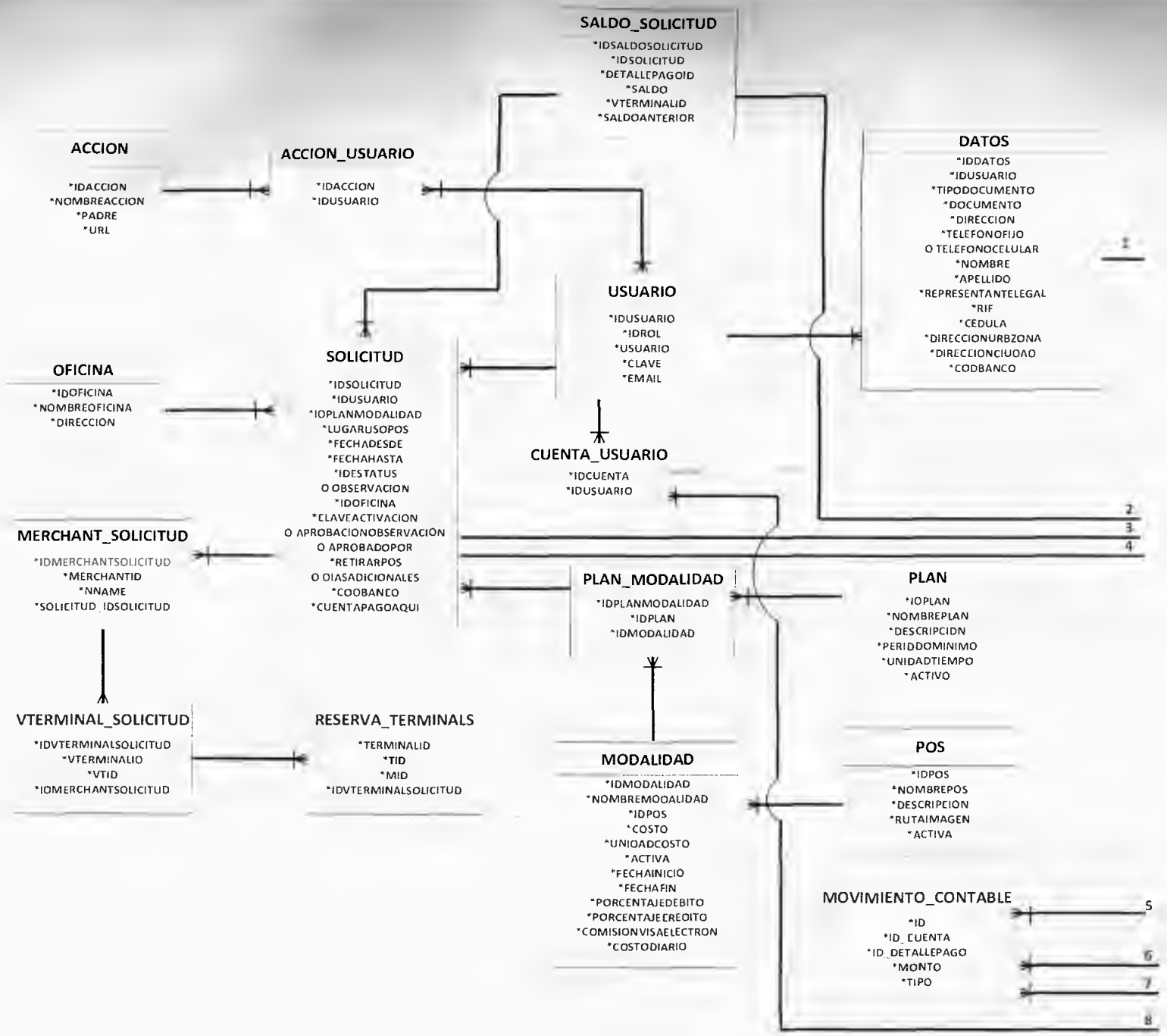

Figura 35. E-R Versión 1 Parte 1

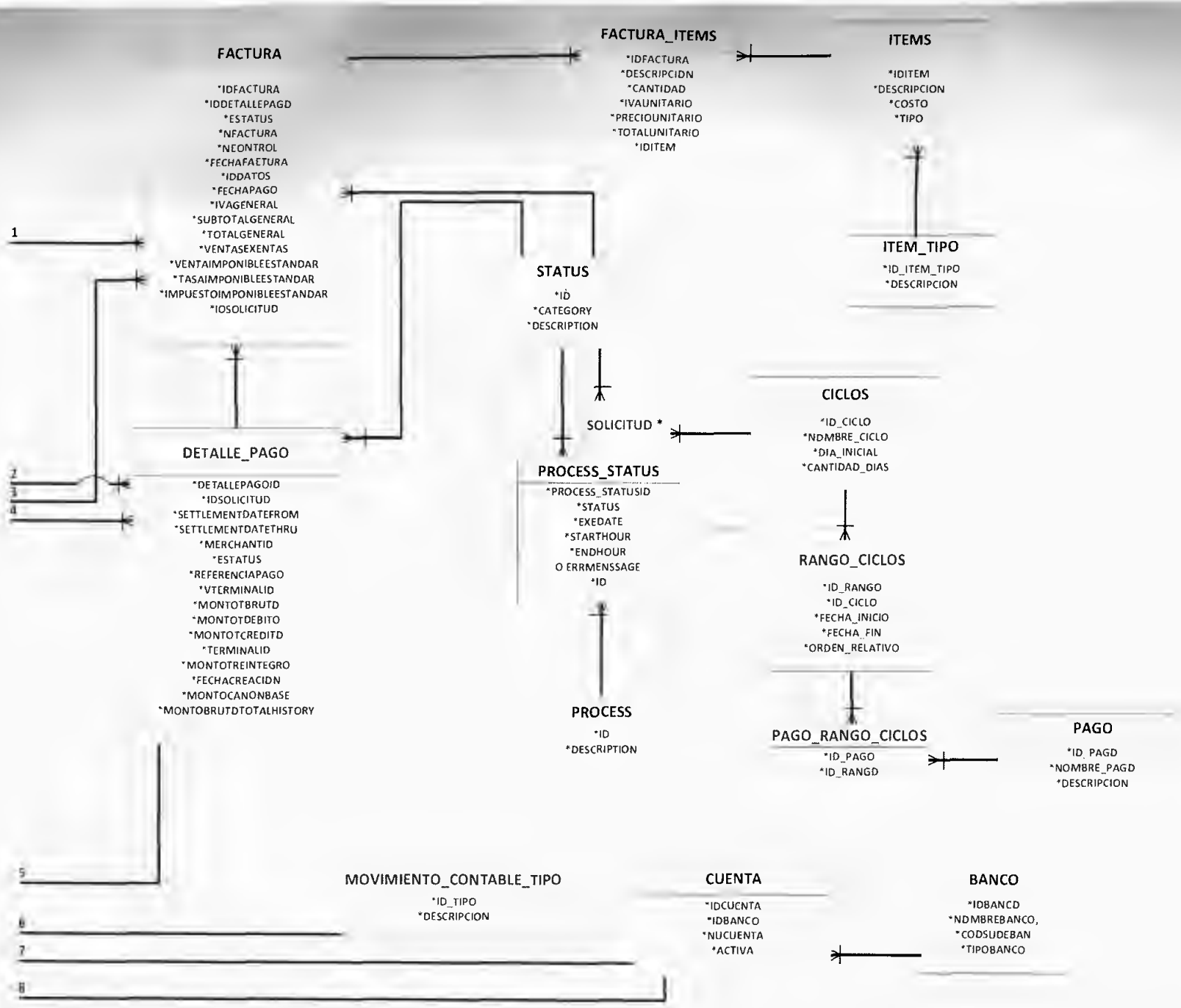

Figura 36. E-R Versión 1 Parte 2

1.2 Diagrama E - R

Versión 2

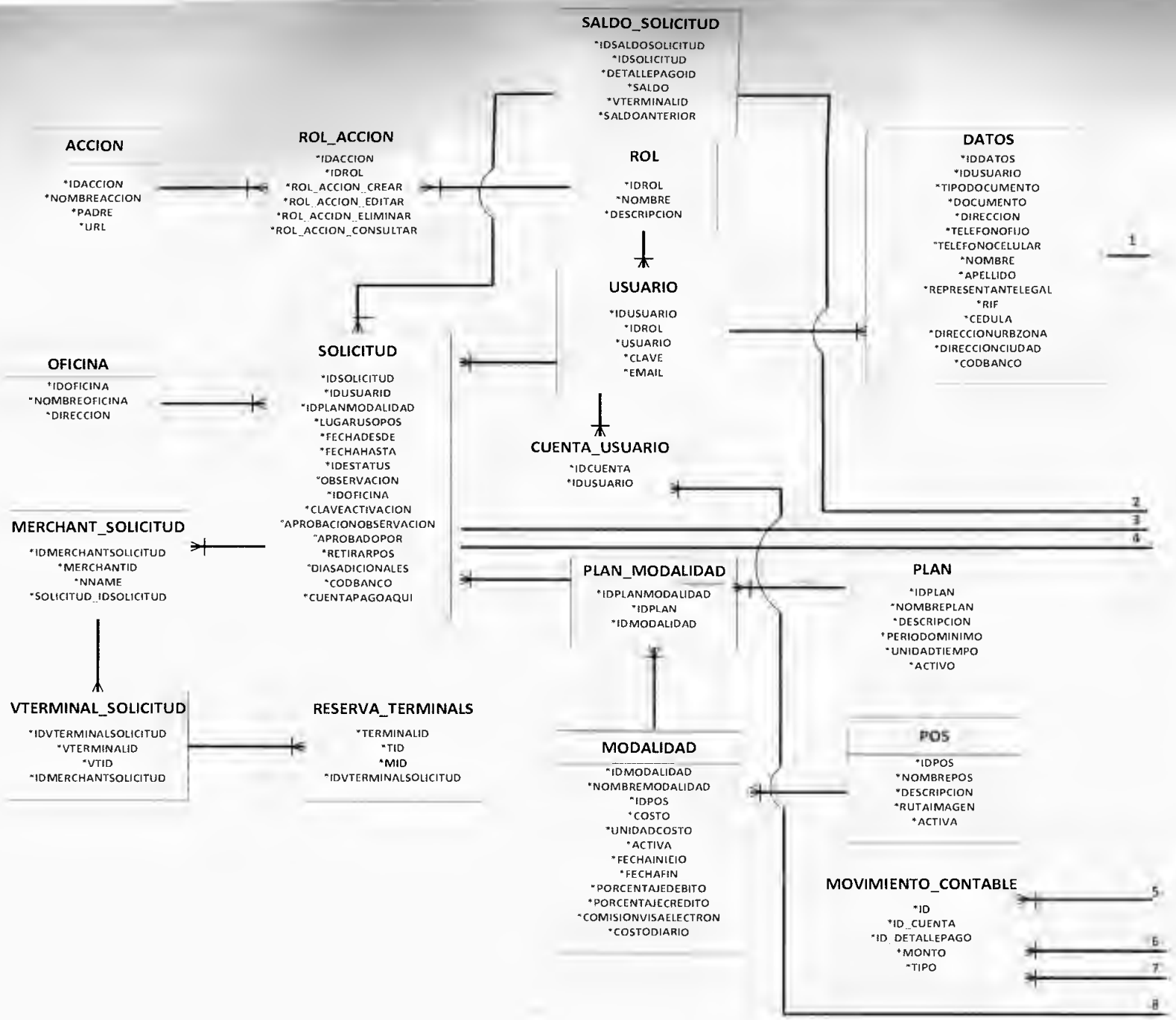

Figura 37. E-R Versión 2 Parte 1

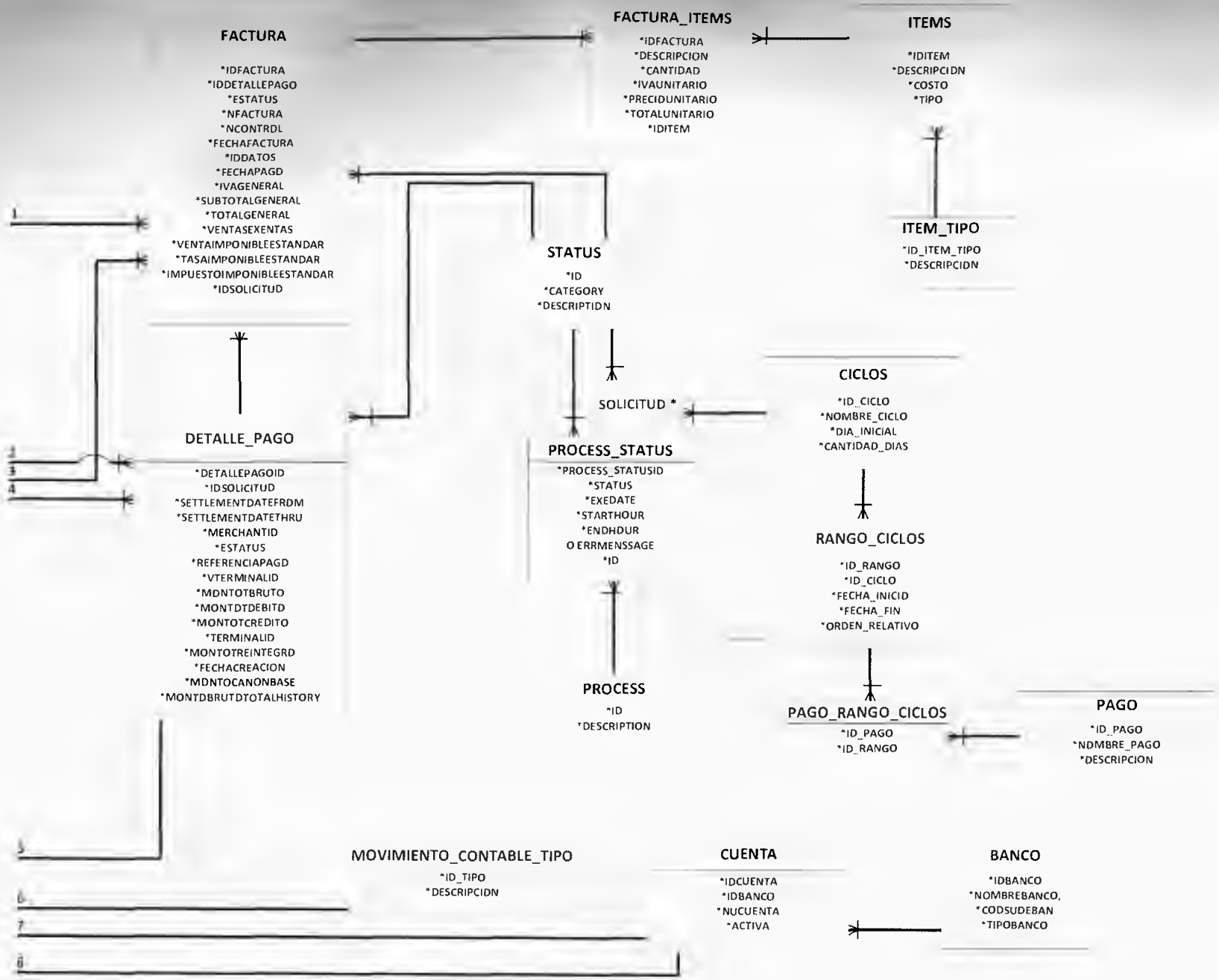

Figura 38. E-R Versión 2 Parte 2

1.3 Diagrama E - R

Versión 3

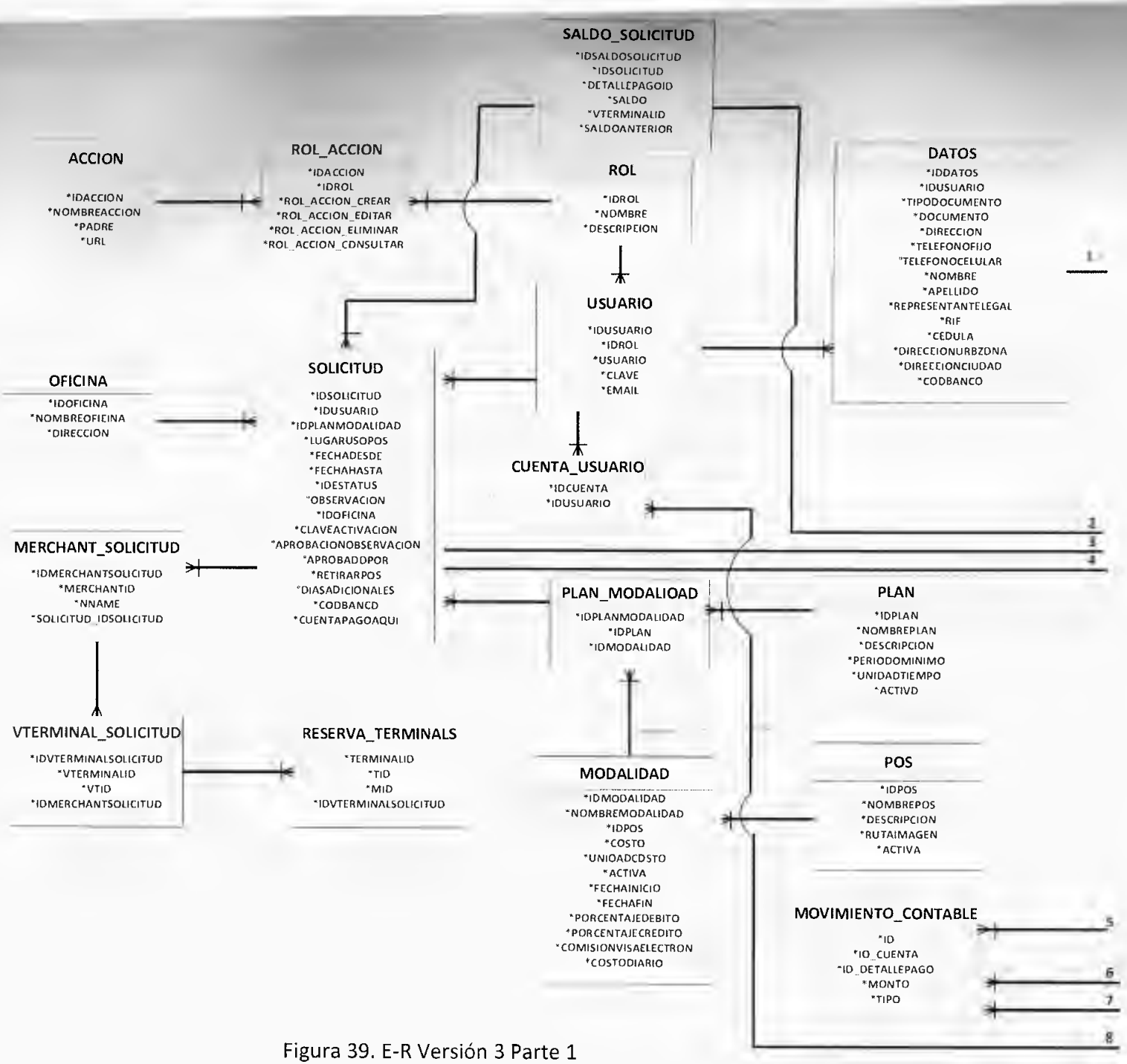

Figura 39. E-R Versión 3 Parte 1

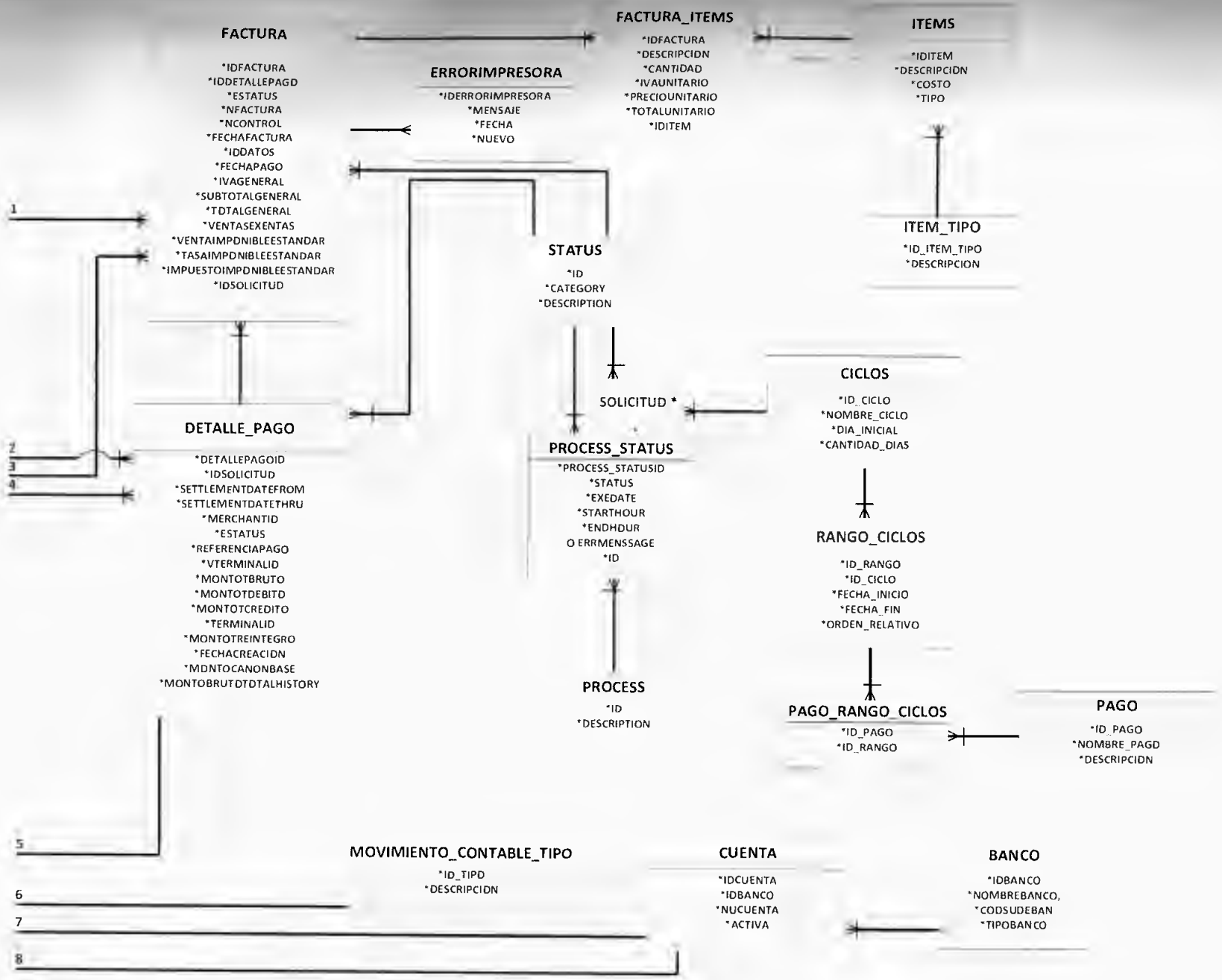

Figura 40. E-R Versión 3 Parte 2

1.4 Diagrama E - R

Versión 4

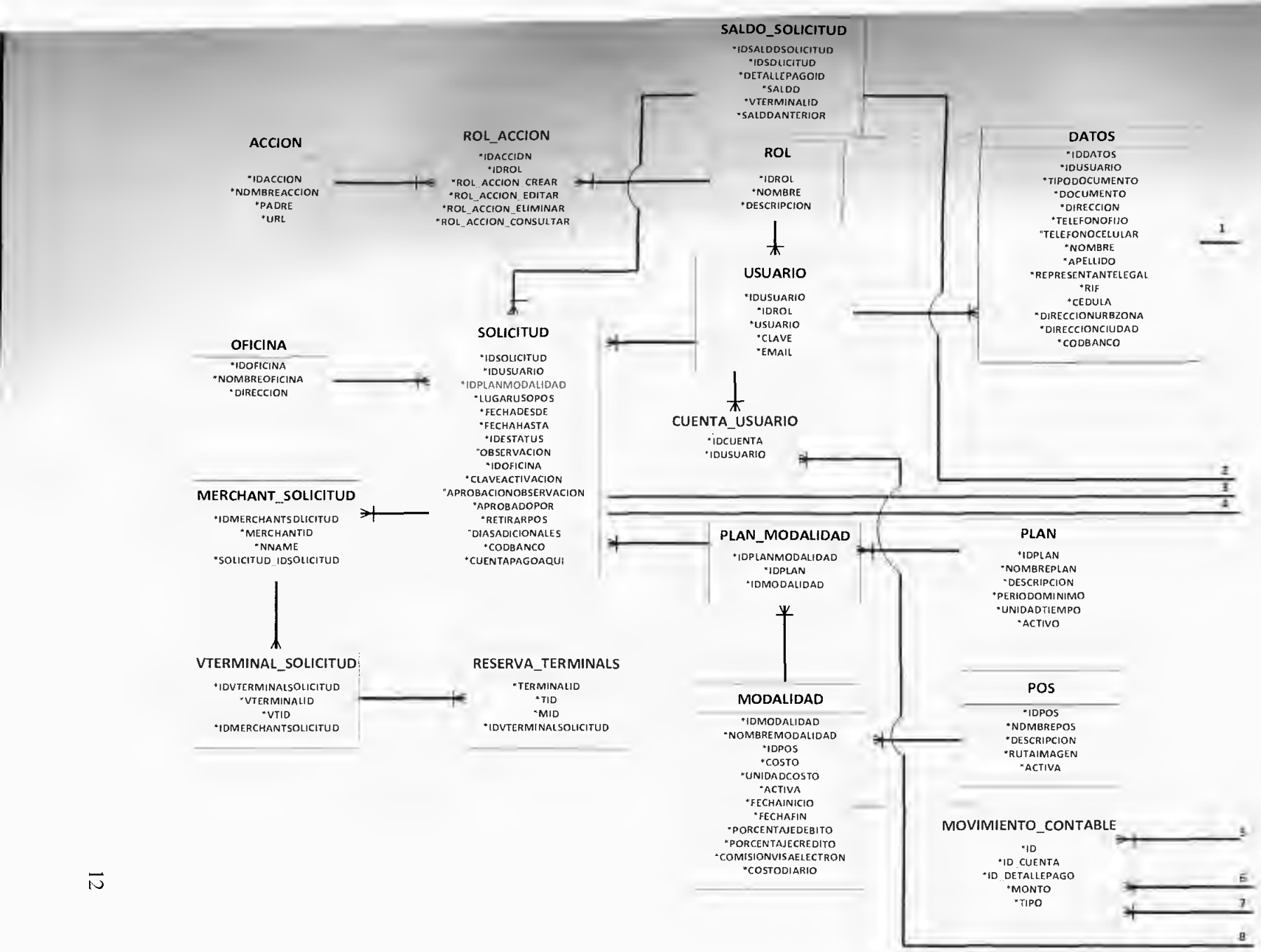

Figura 41. E-R Versión 4 Parte 1

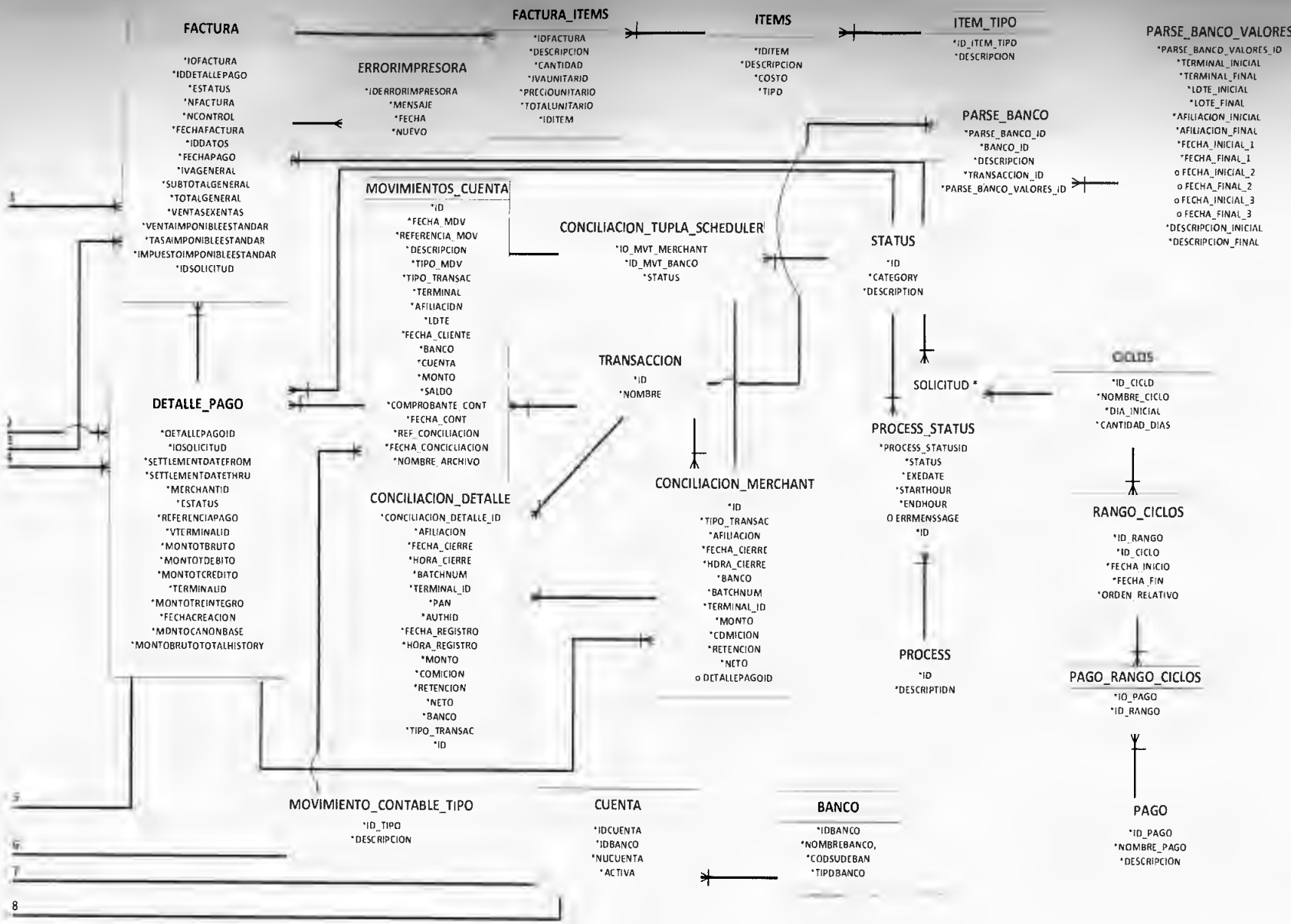

Figura 42. E-R Versión 4 Parte 2

1.5 Diagrama E - R

Versión 5

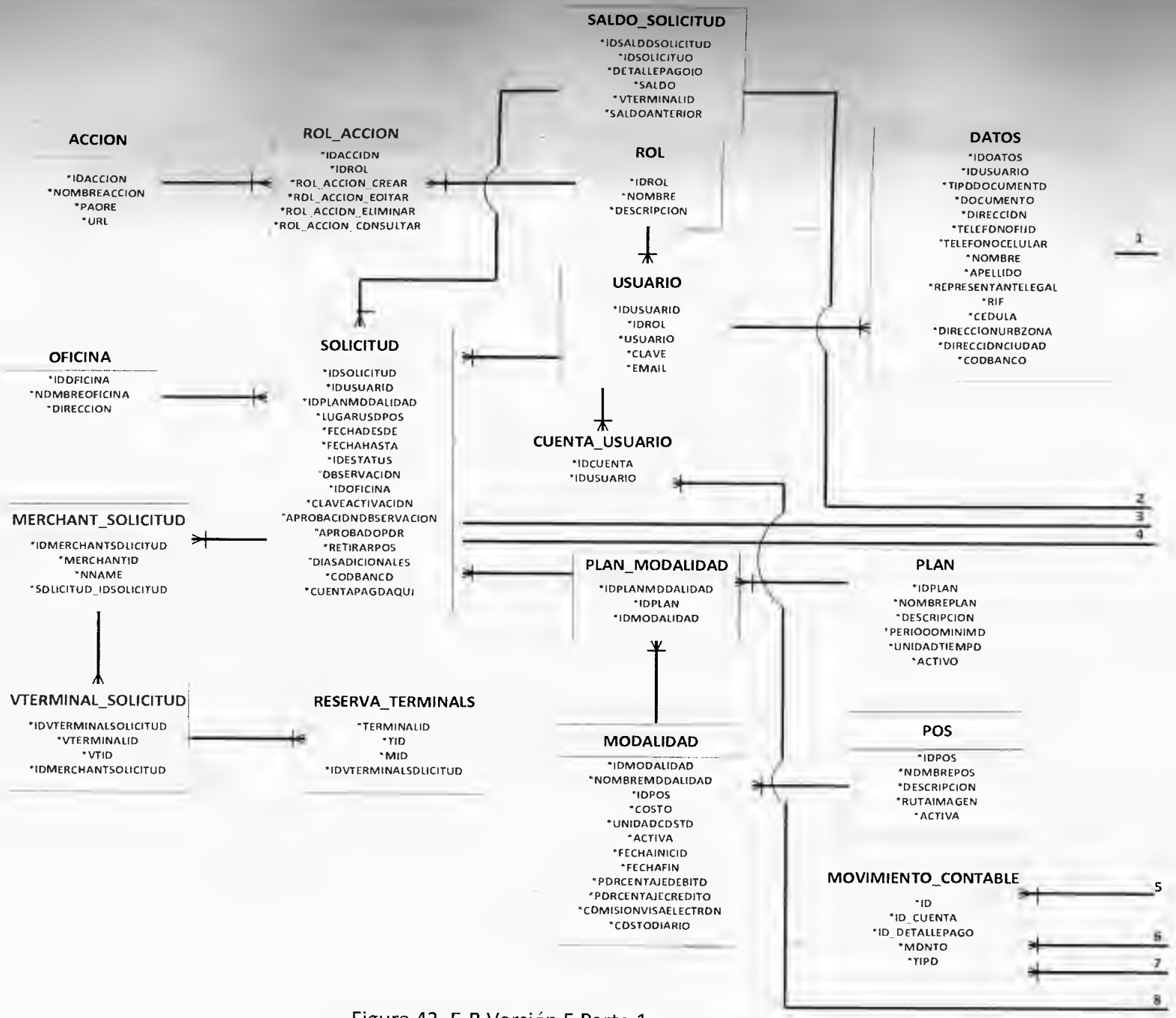

Figura 43. E-R Versión 5 Parte 1

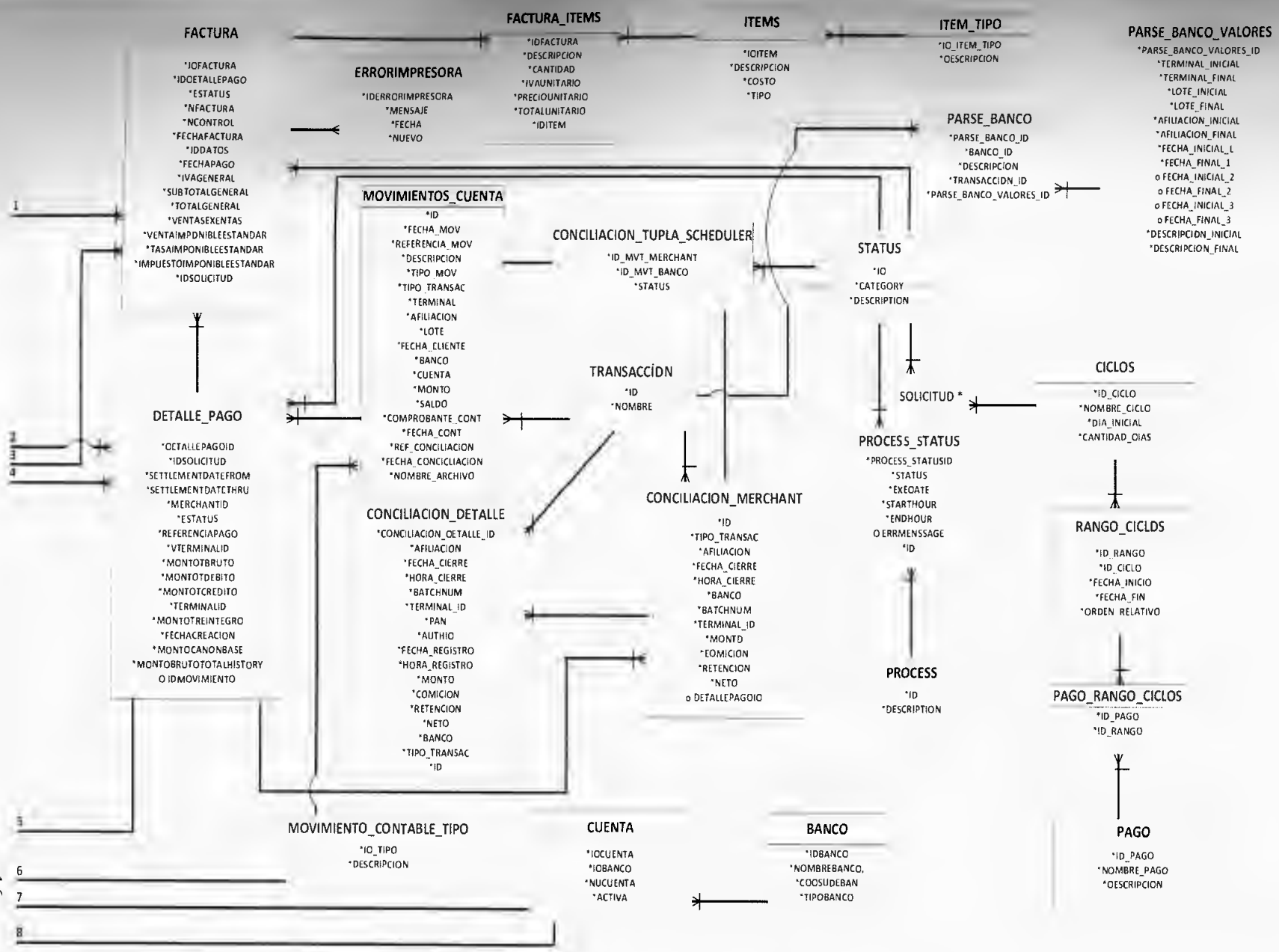

Figura 44. E-R Versión 5 Parte 2

1.6 Diagrama E - R

Merchant

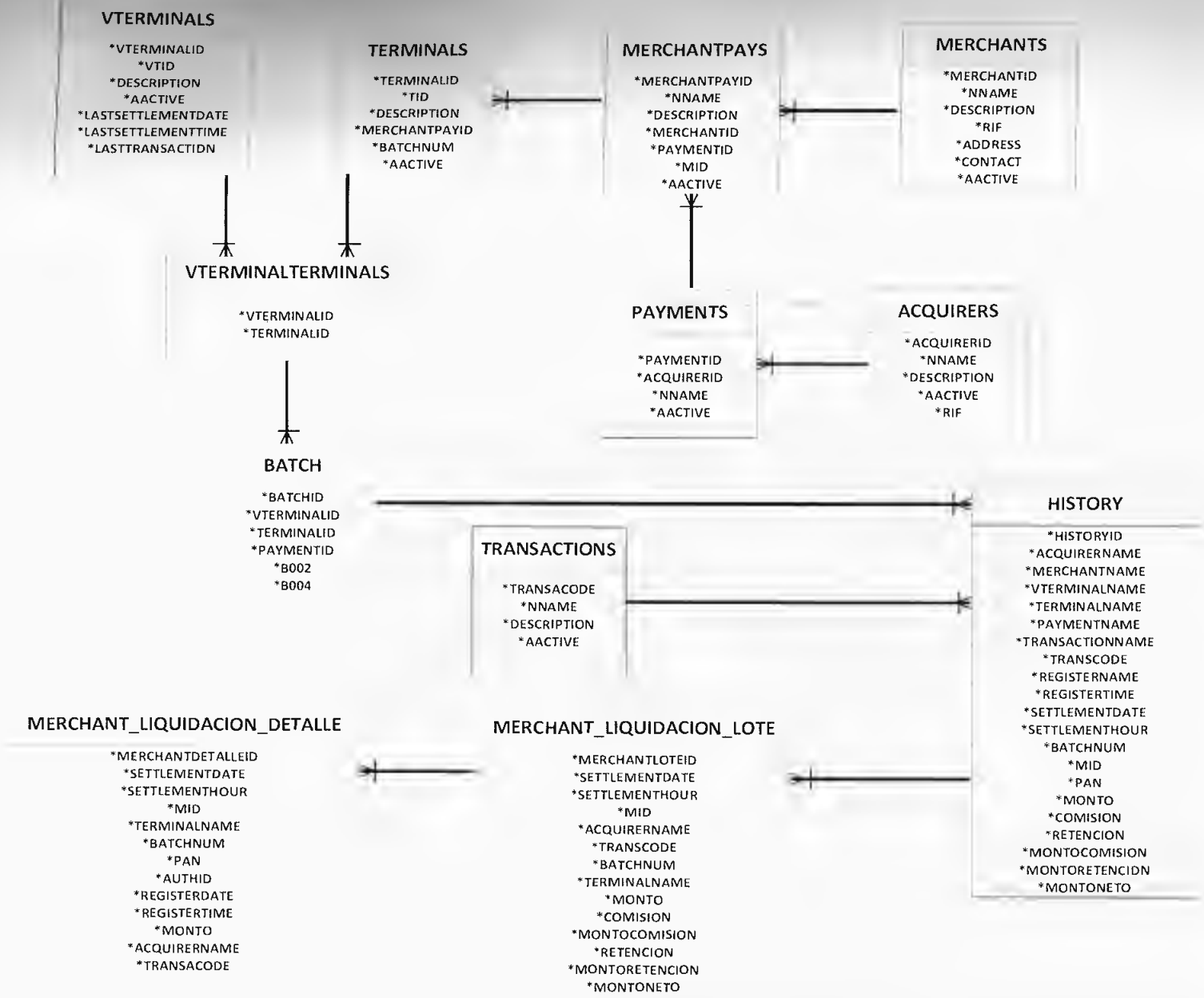

Figura 45. E-R Merchant

Apéndice 2

(Descripción de las tablas de la Base de Datos)

Tabla: Solicitud

Se utiliza para guardar todas las solicitudes de cada uno de los clientes de Pago Aquí.

idSolicitud PK int NOT NULL, clave primaria de la tabla.
idUsuario int NOT NULL, foránea de la tabla Usuario.
idPlanModalidad int NOT NULL, foránea de la tabla Plan_Modalidad.
LugarUsoPos varchar(300) NOT NULL, lugar de utilización del punto de venta.
fechaDesde datetime NOT NULL, fecha inicial de creación de la solicitud.
fechaHasta datetime NOT NULL, fecha final de validez de la solicitud.
idEstatus int NOT NULL, estatus actual de la solicitud.
observacion varchar(300), observaciones adicionales de la solicitud.
idOficina int, clave foranea de la tabla Oficina.
claveActivacion varchar(50), clave de activación de la solicitud
aprobacionObservacion varchar(500), observaciones de aprobación de la solicitud.
aprobadoPor int, usuario que aprobó la solicitud.
retirarPos datetime, fecha cuando es retirado el punto de venta.
diasAdicionales int NOT NULL, días adicionales de cobro.
codBanco nchar(4), clave foránea de la tabla banco.
cuentaPagoaqui smallint, clave foránea de la tabla cuenta_paqui.

Tabla: Oficina

Se utiliza para almacenar las distintas oficinas que posee la empresa en el país.

idOficina PK int NOT NULL, clave primaria de la tabla.
nombreOficina varchar(50), nombre de la oficina.
direccion varchar(300), dirección donde se encuentra la oficina.

Tabla: Status

Se utiliza para definir los distintos status que puedan existir para las solicitudes, detalles de pago, facturas, conciliaciones y porches status.

id PK int NOT NULL, clave primaria de la tabla.
category varchar(400), categoría a la cual hace referencia (solicitud, factura, etc.).
description varchar(400), descripción o nombre del status.

Tabla: Transaccion

Se guardan los distintos tipos de transacciones que pueden existir, importantes de definir a la hora de leer los archivos Excel de movimientos provenientes de los distintos bancos.

ID int NOT NULL, clave primaria de la tabla.
nombre nvarchar(255) NOT NULL, nombre de la transacción.

Tabla: Pos

Se guardan las características de los puntos de ventas ofrecidos actualmente por Pago Aquí.

idPos PK int NOT NULL, clave primaria de la tabla.
nombrePos varchar(50), nombre del punto de venta.
descripcion varchar(500), descripcion del punto de venta.
rutaImagen varchar(100), ruta donde se encuentra la imagen del punto de venta.
activa smallint, si se encuentra o no activo el punto de venta.

Tabla: Modalidad

Se almacenan las distintas modalidades que puede tener cada uno de los puntos de ventas.

idModalidad PK int NOT NULL, clave primaria de la tabla.
nombreModalidad varchar(100), nombre de la modalidad.
idPos int NOT NULL, clave foránea de la tabla POS.
costo float, costo de dicha modalidad.
unidadCosto varchar(20), unidad del costo de la modalidad (mensual, diario, etc.).
activa smallint, si se encuentra o no activa la modalidad.
fechaInicio datetime, fecha de inicio de la modalidad.
fechaFin datetime, fecha de fin de utilidad de la modalidad.
porcentajeDebito float, porcentaje de comisión de transacciones de débito.
porcentajeCredito float, porcentaje de comisión de transacciones de crédito.
ComisionVisaElectron float, comisión para transacciones visa electron.
CostoDiario flota, costo diario de la modalidad.

Tabla: Plan

En esta tabla se almacenan los distintos planes que ofrece Pago Aquí para sus clientes.

idPlan PK int NOT NULL, clave primaria de la tabla.
nombrePlan varchar(50) NOT NULL, nombre del plan.
descripcion varchar(1000), descripcion general del plan.
periodoMinimo int NOT NULL, periodo mínimo de utilización del plan.
unidadTiempo char(1), unidad del periodo minimo de tiempo (mensual).
activo int, se encuentra o no activado el plan.

Tabla: Plan_Modalidad

Esta tabla permite establecer la relación Plan – Modalidad.

idPlanModalidad PK int NOT NULL, clave primaria de la tabla.
idPlan int NOT NULL, clave foránea de la tabla Plan.
idModalidad int NOT NULL, clave foránea de la tabla Modalidad.

Tabla: Accion

Se guardan las distintas acciones que se pueden acceder en el sistema, como por ejemplo reportes de rol, solicitudes, conciliación, error impresora, entro otras.

idAccion PK int NOT NULL, clave primaria de la tabla.
nombreAccion varchar(50), nombre de la accion.
padre int, clave foránea de la tabla acción.
url varchar(100), URL de la acción (ruta).

Tabla: Rol

Se almacenan todos los roles que se encuentran en el sistema.

idRol PK int NOT NULL, clave primaria de la tabla.
nombre varchar(50) NOT NULL, nombre del rol.
descripcion varchar(100), descripcion del rol.

Tabla: Rol_Accion

Esta tabla permite realizar la relación entre Rol y Accion.

idAccion PK int NOT NULL, clave foránea de la tabla Acción.
idRol int NOT NULL, clave foránea de la tabla Rol.
Rol_Accion_Crear int, si se encuentra o no activado el privilegio crear.
Rol_Accion_Editar int, si se encuentra o no activado el privilegio editar.
Rol_Accion_Eliminar int, si se encuentra o no activado el privilegio eliminar.
Rol_Accion_Consultar int si se encuentra o no activado el privilegio consultar.

Tabla: Usuario

Se guarda cada uno de los usuarios que existen en el sistema.

idUsuario PK int NOT NULL, clave primaria de la tabla.
idRol int NOT NULL, clave foránea de la tabla Rol.
usuario varchar(50) NOT NULL, id del usuario.
clave varchar(50) NOT NULL, clave del usuario.
email varchar(120) NOT NULL, correo electrónico del usuario.

Tabla: Datos

En esta tabla se guarda toda la información relacionada a los clientes de Pago Aquí.

idDatos PK int NOT NULL, clave primaria de la tabla.
idUsuario int NOT NULL, clave foránea de la tabla Usuario.
tipoDocumento char(1) NOT NULL, tipo de documento del cliente (jurídico, etc.).
documento varchar(50) NOT NULL, identificación del documento (cédula, etc.)
direccion varchar(300) NOT NULL, dirección del cliente.
telefonoFijo varchar(50) NOT NULL, teléfono fijo del cliente.
telefonoCelular varchar(50), teléfono celular del cliente.
nombre varchar(100) NOT NULL, nombre del cliente.
apellido varchar(400), apellido del cliente.
representanteLegal varchar(50), nombre del representante legal.
RIF varbinary(max), RIF del cliente.
direccionUrbZona varchar(75), dirección del cliente.
direccionCiudad varchar(75), ciudad donde vive el cliente.
codBanco nchar(4), clave foránea de la tabla Banco.

Tabla: Banco

Se almacenan los distintos bancos que se utilizan en el sistema.

idBanco PK int NOT NULL, clave primaria de la tabla.
nombreBanco varchar(100), nombre del banco.
codSudeban nchar(4) NOT NULL, codifo Sudaban de cada banco.

Tabla: Cuenta

En esta tabla se guardan la información relacionada a las cuentas bancarias de cada uno de los clientes de Pago Aquí.

idCuenta PK int NOT NULL, clave primaria de la tabla.
idBanco int NOT NULL, clave foránea de la tabla Banco.
nuCuenta varchar(20) NOT NULL, número de cuenta.
activa smallint NOT NULL, si se encuetra o no activada la cuenta.

Tabla: Cuenta_Usuario

Esta tabla permite realizar la relación Cuenta – Usuario.

idCuenta int NOT NULL, clave foránea de la tabla Cuenta.
idUsuario int NOT NULL, clave foránea de la tabla Usuario.

Tabla: Movimiento_Contable_Tipo

Se almacenan los tipos de movimientos que puedan existir. Los tipo de movimiento que se encuentran actualmente son de Haber y Deber.

ID_TIPO PK int NOT NULL, clave primaria de la tabla.
DESCRIPCION nvarchar(255), descripción del tipo.

Tabla: Movimientos_Cuenta

Se guardan los movimientos obtenidos de los archivos Excel provenientes de cada banco.

ID PK int NOT NULL, clave primaria de la tabla.
FECHA_MOV date NOT NULL, fecha del movimiento.
REFERENCIA_MOV varchar(30) NOT NULL, referencia del movimiento.
DESCRIPCION varchar(255), descripcion del movimiento-
TIPO_MOV int NOT NULL, clave foranea de la tabla Movimiento_Contable_Tipo.
TIPO_TRANSAC int, clave foránea de la tabla Transacción.
TERMINAL int, Terminal del punto de venta.
AFILIACION int, afiliación del movimiento.
LOTE int, lote del movimiento.
FECHA_CLIENTE date,
CUENTA varchar(20) NOT NULL, número de cuenta del movimiento.
MONTO float NOT NULL, monto del movimiento.
SALDO float NOT NULL, saldo del movimiento.
NOMBRE_ARCHIVO varchar(1000), nombre del archivo del movimiento.

Tabla: Item_Tipo

Se almacenan los tipos de transacción de cada uno de los ítems en el sistema.

ID_ITEM_TIPO PK int NOT NULL, clave primaria de la tabla.
DESCRIPCION nvarchar(255), descripcion del tipo de item.

Tabla: Detalle_Pago

En esta tabla se almacenan el detalle de todos los pagos que se tienen por cada solicitud.

DetallePagoID PK int NOT NULL, clave primaria de la tabla.
idSolicitud int NOT NULL, clave foránea de la tabla Solicitud.
SettlementDateFrom varchar(8), fecha inicio de liquidación.
SettlementDateThru varchar(8), fecha fin de liquidacion.
MerchantID int, id del Merchant asociado al detalle pago.
Estatus varchar(2), foranea de la tabla status.
ReferenciaPago varchar(50), concatenado de las fechas de liquidación.
VterminalID int, id del vterminal asociado al detalle pago.
MontoTBruto bigint, suma de los montos de transacciones de débito y crédito.

MontoTDebito bigint, monto de transacciones de débito
MontoTCredito bigint, monto de transacciones de crédito.
TerminalID int, id del terminal asociado al detalle pago.
MontoTReintegro bigint, monto de reintegro al cliente.
FechaCreacion datetime, fecha de creación del detalle pago.
MontoCanonBase float, monto de cobro del servicio.
MontoBrutoTotalHistory bigint, historial de los montos brutos.
idMovimiento int, clave foranea de la tabla Movimientos_Cuenta.

Tabla: Saldo_Solicitud

Esta tabla permite realizar la relación de los detalle pagos y solicitudes.

idSaldoSolicitud PK bigint NOT NULL, clave primaria de la tabla.
idSolicitud int NOT NULL, clave foránea de la tabla Solicitud.
detallePagoID int NOT NULL, clave foránea de la tabla Detalle_Pago.
saldo decimal(12,2), saldo negativo del cliente.
VterminalID int, Vterminal de la solicitud.
saldoAnterior decimal(12,2), saldo negativo del ciclo anterior.

Tabla: Items

En esta tabla se guardan los ítems proporcionados en Pago Aquí. Algunos de los ítems especificados en esta tabla son Administración de Pago Electrónicos, transporte, papel térmico, entre otros.

idItem PK int NOT NULL, clave primaria de la tabla.
descripcion varchar(500) NOT NULL, descripción del item.
costo int, costo del item.
tipo int NOT NULL, clave foránea de la tabla Item_Tipo.

Tabla: Factura

Se guardan todas las facturas que se crean en Pago Aquí por cada solicitud.

idFactura PK int NOT NULL, clave primaria de la tabla.
idDetallePago int, clave foránea de la tabla Detalle_Pago.
Estatus int, clave foránea de la tabla Status.
nFactura varchar(50) NOT NULL, numero de factura.
nControl varchar(50), numero de control de la factura.

fechaFactura datetime, fecha de la factura.
idDatos int, clave foránea de la tabla Datos.
fechaPago datetime, fecha de realización de pago.
impuestoImponibleEstandar decimal(10,2), impuesto de la factura.
idSolicitud int, clave foránea de la tabla Solicitud.

Tabla: Factura_Items

Esta tabla permite realizar la relación entre Factura y Items.

idFactura PK int NOT NULL, clave primaria de la tabla.
descripcion varchar(500) NOT NULL,
cantidad int NOT NULL, cantidad de items.
ivaUnitario decimal(10,2),
precioUnitario decimal(10,2),
totalUnitario decimal(10,2),
idItem int
Tabla: Movimiento_Contable

Esta tabla permite realizar la relación entre los Detalle Pagos y las Cuentas.

ID PK int NOT NULL, clave primaria de la tabla.
ID_CUENTA int NOT NULL, clave foránea de la tabla Cuenta.
ID_DETALLEPAGO int NOT NULL, clave foránea de la tabla Detalle_Pago.
MONTO bigint NOT NULL, monto de cada movimiento contable.
TIPO tinyint NOT NULL, clave foránea de la tabla Movimiento_Contable_Tipo.

Tabla: Conciliacion_Merchant

Tabla donde se almacena la información de los lotes de transacciones obtenidos en la tarea CargaMerchantLoteDeamon.

ID PK int NOT NULL, clave primaria de la tabla.
TIPO_TRANSAC int, clave foránea de la tabla Transacción.
AFILIACION int, afiliación del movimiento.
FECHA_CIERRE date, fecha de cierre del movimiento.
HORA_CIERRE time(0), hora de cierre del movimiento.
BANCO int, clave foránea de la tabla Banco.
BATCHNUM int NOT NULL, número de lote relacionado al movimiento.
TERMINAL_ID varchar(255), terminal del movimiento.
MONTO float NOT NULL, monto del movimiento.
COMICION float NOT NULL, comisión del movimiento.
RETENCION float NOT NULL, retencion del movimiento.

NETO float NOT NULL, monto neto del movimiento.
DetallePagoID int, clave foranea de la tabla Detalle_Pago.

Tabla: Conciliacion_Detalle

Se guarda toda la información de las transacciones pertenecientes a los lotes obtenidos en la tarea CargaMerchantLoteDeamon.

CONCILIACION_DETALLE_ID PK int NOT NULL, clave primaria de la tabla.
AFILIACION int, afiliación del lote.
FECHA_CIERRE date, fecha de cierre del lote.
HORA_CIERRE time(0), hora de cierre del lote.
BATCHNUM int, número de lote.
TERMINAL_ID varchar(255), terminal del lote.
PAN varchar(48), número de tarjeta encriptada.
AUTHID varchar(6), numero de autorización del lote.
FECHA_REGISTRO date,
HORA_REGISTRO time(0),
MONTO float NOT NULL, monto del lote.
COMICION float NOT NULL, comisión del lote.
RETENCION float NOT NULL, retención del lote.
NETO float NOT NULL, monto neto del lote.
BANCO int, clave foránea de la tabla Banco.
TIPO_TRANSAC int, clave foránea de la tabla Transaccion.
ID int, clave foranea de la tabla Conciliación_Merchant.

Tabla: Conciliacion_Tupla_Scheduler

Esta tabla permite almacenar la comparación que realiza la tarea ComparacionTupla entre las tablas Movimientos_Cuenta, Conciliacion_Merchant y Status.

ID_MVT_MERCHANT int, clave foránea de la tabla Conciliacion_Merchant.
ID_MVT_BANCO int, clave foránea de la tabla Movimientos_Cuenta.
STATUS int, clave foránea de la tabla Status.

Tabla: Process

Se guardan los nombres de cada una de las tareas del Administrador de tareas Scheduler.

id varchar(50) NOT NULL, clave primaria de la tabla.
description varchar(100), descripción de la tarea.

Tabla: Process_Status

Esta tabla permite hacer la relación entre las tablas Process y Status.

process_statusid PK int NOT NULL, clave primaria de la tabla.
Status int, clave foranea de la tabla Status.
ExeDate varchar(8), fecha de ejecución de la tarea.
StartHour varchar(8), hora de inicio de la tarea.
EndHour varchar(8), hora de fin de la tarea.
ErrMenssage varchar(200), mensaje del error ocurrido durante la ejecución.
id varchar(50), clave foranea de la tabla Process.

Tabla: Parse_Banco_Valores

Se guarda toda la información relacionada a los rangos de parseo para los archivos Excel de movimientos.

PARSE_BANCO_VALORES_ID PK int NOT NULL, clave primaria de la tabla.
TERMINAL_INICIAL int, valor inicial del rango de parseo del terminal.
TERMINAL_FINAL int, valor final del rango de parseo del terminal.
LOTE_INICIAL int, valor inicial del rango de parseo del lote.
LOTE_FINAL int, valor final del rango de parseo del lote.
AFILIACION_INICIAL int, valor inicial del rango de parseo de la afiliación.
AFILIACION_FINAL int, valor final del rango de parseo de la afiliación.
FECHA_INICIAL_1 int, valor inicial del rango de parseo de fecha 1.
FECHA_FINAL_1 int, valor final del rango de parseo de fecha 1.
FECHA_INICIAL_2 int, valor inicial del rango de parseo de fecha 2.
FECHA_FINAL_2 int, valor final del rango de parseo de fecha 2.
FECHA_INICIAL_3 int, valor inicial del rango de parseo de fecha 3.
FECHA_FINAL_3 int, valor final del rango de parseo de fecha 3.
DESCRIPCION_INICIAL int, valor inicial del rango de parseo de descripción.
DESCRIPCION_FINAL int, valor final del rango de parseo de descripción.

Tabla: Parse_Banco

En esta tabla se almacena la información que permite definir qué tipo de parseo se debe realizar en los movimientos de los archivos Excel.

PARSE_BANCO_ID PK int NOT NULL, clave primaria de la tabla.
BANCO_ID int, clave foranea de la tabla Banco.
DESCRIPCION varchar(200), descripción base para identificar tipo de parseo.
TRANSACCION_ID int, clave foránea de la tabla Transaccion.

PARSE_BANCO_VALORES_ID int, clave foránea de la tabla Parse_Banco_Valores.

Tabla: Ciclo

Se guarda los ciclos que existen en el sistema que se usan para la realización de pagos a clientes.

ID_CICLO PK int NOT NULL, clave principal de la tabla Ciclo.
NOMBRE_CICLO varchar(255) NOT NULL, nombre del ciclo (lunes, martes, etc.).
DIA_INICIAL varchar(50), dia inicial del ciclo de cada mes.
CANTIDAD_DIAS int, cantidad de días que dura el ciclo.

Tabla: Rango_Ciclos

Se almacena el rango de fechas de cada ciclo.

ID_RANGO PK int NOT NULL, clave primaria de la tabla.
ID_CICLO int NOT NULL, clave foránea de la tabla Ciclo.
FECHA_INICIO date NOT NULL, fecha de inicio del ciclo.
FECHA_FIN date NOT NULL, fecha de fin del ciclo.
ORDEN_RELATIVO int NOT NULL, orden de cada ciclo durante el mes.

Tabla: Merchant_Solicitud

Esta tabla permite relacionar cada solicitud a un id de Merchant que se encuentre en la base de datos de Merchant.

idMerchantSolicitud PK int NOT NULL, clave primaria de la tabla.
MerchantId int NOT NULL, id merchant de la solicitud.
Nname varchar(50), nombre del merchant.
Solicitud_idSolicitud int NOT NULL, clave foránea de la tabla Solicitud.

Tabla: Vterminal_Solicitud

Se guardan todos los vterminals de cada uno de los Merchants asociados a una solicitud.

idVterminalSolicitud PK int NOT NULL, clave primaria de la table.
vTerminalId int, id vterminal del merchant.
VTID varchar(400),
idMerchantSolicitud int NOT NULL, clave foránea de la tabla Merchant_Solicitud.

Tabla: Reserva_Terminals

Se almacenan los terminales de cada vterminal del sistema.

terminalId PK int NOT NULL, clave primaria de la tabla.
TID varchar(400), id del terminal del vterminal.
MID varchar(400), afiliación del terminal.
idVterminalSolicitud int NOT NULL, clave foránea de la tabla Vterminal_Solicitud.

Tabla: ErrorImpresora

Se guarda los errores producidos por la impresora fiscal al realizar la impresión de facturas.

idErrorImpresora PK int NOT NULL, clave primaria de la tabla.
mensaje varchar(4000) NOT NULL, mensaje de error de la impresora fiscal.
fecha datetime NOT NULL, fecha de ocurrencia del error.
nuevo int NOT NULL, novedad del error.

Tabla: Transactions

Se guardan los tipos de transacciones que pueden existir.

TransCode PK int NOT NULL, clave primaria de la tabla.
NName varchar(40) NULL, nombre de la transacción.
Description varchar(50) NULL, descripción de la transacción.
AActive int NULL, Se encuentra activa o no.

Tabla: Vterminals

Se guardan todos los vterminals de cada uno de los puntos de venta.

VTerminalID PK int NOT NULL, clave primaria de la tabla.
VTID varchar(20) NULL, nombre del vterminal.
Description varchar(50) NULL, descripción del vterminal.
AActive int NULL, se encuentra activo o no.
LastSettlementDate varchar(8) NULL, última fecha de liquidación.
LastSettlementTime varchar](6) NULL, última hora de liquidación.
LastTransaction varchar(14) NULL, fecha de la última transacción.

Tabla: VterminalTerminals

Esta tabla permite realizar la asociación entre las tablas Vterminals y Terminals.

VTerminalID int NOT NULL, clave foránea de la table vterminals.
TerminalID int NOT NULL, clave foránea de la tabla terminals.

Tabla: Acquirers

Se guardan los bancos adquirientes del sistema.

AcquirerID PK int NOT NULL, clave primaria de la tabla.
NName varchar(40) NULL, nombre del adquiriente.
Description varchar(50) NULL, descripción del adquiriente.
AActive int NULL, se encuentra activo o no.
RIF varchar(15) NULL, rif del adquiriente.

Tabla: Batch

Se guardan las transacciones realizadas en los puntos de venta de los cuales aun no han hecho cierre.

BatchID PK int NOT NULL, clave primaria de la tabla.
VTerminalID int NULL, clave foránea de la tabla vterminals.
TerminalID int NULL, clave foránea de la tabla terminal.
PaymentID int NULL, clave foránea de la tabla payment.
b002 varchar(48) NULL, número de tarjeta encriptado.
b004 int NULL, monto de la transacción.

Tabla: MerchantPays

Se guardan los contratos de afiliacion de cada banco.

MerchantPayID PK int NOT NULL, clave primaria de la tabla.
NName varchar(30) NULL, nombre del contrato.
Description varchar(50) NULL, descripción del contrato.
MerchantID int NULL, clave foránea de la tabla merchants.
PaymentID int NULL, clave foránea de la tabla payments.
MID varchar(15) NULL, número de afiliación.
AActive int NULL, se encuentra activo o no.

Tabla: Merchants

Se guarda la informacion referente a los puntos de venta.

MerchantID PK int NOT NULL, clave primaria de la tabla.
NName varchar(30) NULL, nombre del merchant.
Description varchar(50) NULL, descripción del merchant.
RIF varchar(15) NULL, rif de la persona.
Address varchar(80) NULL, dirección de la persona.
Contact varchar(25) NULL, nombre de la persona de contacto.
AActive int NULL, se encuentra activo o no.

Tabla: Payments

Tabla que almacena los tipos de pagos que se pueden realizar.

PaymentID PK int NOT NULL, clave primaria de la tabla.
AcquirerID int NULL, foránea de la tabla acquirers.
NName varchar(30) NULL, nombre del tipo de pago.
Description varchar(50) NULL, descripción del tipo de pago.
AActive int NULL, se encuentra activo o no.

Tabla: Terminals

Se guardan todos los terminales que serán asociados a un vterminal.

TerminalID PK int NOT NULL, clave primaria de la tabla.
TID varchar(16) NULL, nombre del terminal.
Description varchar(50) NULL, descripción del terminal.
MerchantPayID int NULL, foránea de a tabla merchantpays.
BatchNum int NULL, número de lote.
AActive int NULL, se encuentra activo o no.

Tabla: History

Tabla que almacena un historial de transacciones realizadas en los puntos de venta.

HistoryID PK int NOT NULL, clave primaria de la tabla.
AcquirerName varchar(20) NULL, nombre de banco adquiriente.
MerchantName varchar(30) NULL, nombre del merchant.
VTerminalName varchar(20) NULL, nombre del vterminal.
TerminalName varchar(16) NULL, nombre del terminal.

PaymentName varchar(30) NULL, nombre del pago.
TransactionName varchar(40) NULL, nombre de la transacción.
RegisterDate varchar(8) NULL, fecha de registro.
RegisterTime varchar(6) NULL, hora de registro.
SettlementDate varchar(8) NULL, fecha de liquidación.
SettlementHour varchar(8) NULL, hora de liquidación.
BatchNum varchar(6) NULL, número de lote.
MID varchar(15) NULL, número de afiliación.
PAN varchar(48) NULL, número de tarjeta encriptada.
Monto int NULL, monto de la transacción.
AcquirerID int NULL, foránea de la tabla acquirers.
MerchantID int NULL, foránea de la tabla merchants.
VTerminalID int NULL, foránea de la tabla vterminal.
TerminalID int NULL, foránea de la tabla terminal.
PaymentID int NULL, foránea de la tabla payment.
TransCode int NULL, foránea de la tabla transaction.
Comision float NULL, porcentaje de la comisión de la transacción.
Retencion float NULL, porcentaje de la retención de la transacción.
MontoComision int NULL, monto de la comisión.
MontoRetencion int NULL, monto de la retención.
MontoNeto int NULL, monto neto de la transacción.

Tabla: Merchant_Liquidacion_Lote

Se guardan los lotes de transacciones realizadas en los puntos de venta.

MerchantLoteID PK int NOT NULL, clave primaria de la tabla.
SettlementDate varchar(8) NULL, fecha de liquidación.
SettlementTime varchar(8) NULL, hora de liquidación.
MID varchar(15) NULL, número de afiliación.
AcquirerName varchar(20) NULL, nombre de banco adquiriente.
TransCode int NULL, foránea de la tabla transaction.
BatchNum varchar(6) NULL, número de lote.
TerminalName varchar(16) NULL, nombre del terminal.
Monto int NULL, monto de la transacción.
Comision int NULL, comisión de la transacción.
MontoComision int NULL, monto de la comisión.
Retencion int NULL, retención de la transacción.
MontoRetencion int NULL, monto de la transacción.
MontoNeto int NULL, monto neto de la transacción.

Tabla: Merchant_Liquidacion_Detalle

Se guardan los detalles de cada uno de los lotes de transacciones.

MerchantDetalleID int NOT NULL, clave primaria de la tabla.
SettlementDate varchar(8) NULL, fecha de liquidación.
SettlementTime varchar(8) NULL, hora de liquidación.
MID varchar(15) NULL, número de afiliación.
TerminalName varchar(16) NULL, nombre del terminal.
BatchNum varchar(6) NULL, número del lote.
PAN varchar(48) NULL, número de tarjeta encriptado.
AuthID varchar(6) NULL, número de autorización.
RegisterDate varchar(8) NULL, fecha de registro.
RegisterTime varchar(8) NULL, hora de registro.
Monto int NULL, monto de la transacción.
AcquirerName varchar(20) NULL, nombre de banco adquiriente.
TransCode int NULL, foránea de la tabla transaction.

Apéndice 3

(Controladores, Modelos y Vistas)

- Controlador ***UsuarioDAO***

```
public RespuestaJsonBean claveActual(UsuarioBean usuario2) {
		RespuestaJsonBean bean = new RespuestaJsonBean();
		try {
			UsuarioBean usuario1 =(UsuarioBean)
ActionContext.getContext().getSession().get("usuario");
			UsuarioBean usuario = (UsuarioBean)sqlMap.queryForObject(map
+ "claveActual",usuario1);

			String clave = md5(usuario1.getUsuario() + usuario2.getClave());
			System.out.println(usuario1.getUsuario());
			System.out.println(usuario2.getClave());
			System.out.println(clave);
			System.out.println(usuario.getClave());

			if (clave.equals(usuario.getClave())){

				bean.setCodigo(2);
				bean.setMensaje("Correcto");

			}else{

				bean.setCodigo(3);
				bean.setMensaje("Incorrecto");

			}


		} catch (Exception e) {
			log.error("Error al Actualizar Datos. Error: " + e.getMessage(), e);
			bean.setCodigo(1);
			bean.setMensaje("Error al Actualizar Datos en el registro. Error: "
+ e.getMessage());
		}

		return bean;
	}
```

```
public RespuestaJsonBean actualizarClave(UsuarioBean usuarioBean) {
		RespuestaJsonBean bean = new RespuestaJsonBean();
		try {
			UsuarioBean usuario =(UsuarioBean)
ActionContext.getContext().getSession().get("usuario");
			usuarioBean.setIdUsuario(usuario.getIdUsuario());
			usuarioBean.setClave(md5(usuario.getUsuario() +
usuarioBean.getClave()));
			sqlMap.update(map + "actualizarClave", usuarioBean);
			bean.setCodigo(0);
			bean.setMensaje("Cambio de Clave Exitosa . ");
		} catch (Exception e) {
			log.error("Error al Actualizar Datos. Error: " + e.getMessage(), e);
			bean.setCodigo(1);
			bean.setMensaje("Error al Actualizar Datos en el registro. Error: "
+ e.getMessage());
		}

		return bean;
	}
```

- Modelo ***AdminUsuarioModel***

```
public RespuestaJsonBean actualizarClave(UsuarioBean usuario){
		return dao.actualizarClave(usuario);
	}

public RespuestaJsonBean claveActual(UsuarioBean usuario){
		return dao.claveActual(usuario);
	}
```

- Vista ***AdminUsuarioAction***

```
public String actualizarClave(){
		respuesta = model.actualizarClave(usr);
		return "compeltos";
	}

public String claveActual(){
		respuesta = model.claveActual(usr);
		return "compeltos";
```

```
}
```

- Controlador ***AccionRolDAO***

```
public List<AccionRolBean> buscaRol() throws Exception {
		try {
			if(log.isDebugEnabled())
			log.debug("Obteniendo todos los Roles del sistema.");
			return sqlMap.queryForList(MAP+"obtenerRol");
		} catch (SQLException e) {
			log.error("Error obteniendo lista de Roles del sistema." +
e.getMessage(), e);
			throw e;
		}

	}



public List<AccionRolBean> consultarAccionRol(int id)throws Exception {
		try {
			log.debug("Obteniendo informacion sobre acciones padre.");
			return sqlMap.queryForList(MAP+"consultarAccionRol",id);

		} catch (Exception e) {

			log.error("Obteniendo informacion sobre acciones padres. Error: "
+ e.getMessage(), e);
			throw e;

		}
	}


public AccionRolBean existeAccionRol(AccionRolBean accionrol1){
		AccionRolBean accionrol = null;
		try{
			log.debug("Obteniendo Si existe o No existe la accion padre.");
			accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"existeAccionRol",accionrol1);
		}catch(Exception e){
			log.error("No fue posible obtener si existe o no la accion padre
para la combinacion Accion / Rol: " + accionrol1.getIdaccion() + " / "+
accionrol1.getIdrol() + ". Error: " + e.getMessage(),e);
```

```
		}
		return accionrol;
	}


  public AccionRolBean obtener(int idaccion){
		AccionRolBean accionrol = null;
		try{
			log.debug("Obteniendo informacion sobre Accion.");
			accionrol=(AccionRolBean)sqlMap.queryForObject(MAP +
"obtener", idaccion);
		}catch(Exception e){
			log.error("No fue posible obtener la informacion de la accion " +
idaccion + "  Error: " + e.getMessage(),e);
		}
		return accionrol;
	}


  public List<AccionRolBean> obtenerAcciones(AccionRolBean accionrol1) throws
Exception{
		List<AccionRolBean> accionrol = null;
		try{
			log.debug("Obteniendo Acciones hijas.");
			accionrol = (List<AccionRolBean>)sqlMap.queryForList(MAP +
"obtenerAcciones",accionrol1);

		}catch(Exception e){
			log.error("No fue posible obtener Acciones hijas de la Accion: " +
accionrol1.getIdaccion() + "  Error: " + e.getMessage(),e);
			throw e;
		}
		return accionrol;
	}


  public List<AccionRolBean> rellenoAccion(AccionRolBean accionrol1) throws
Exception{
		List<AccionRolBean> accionrol = null;
		try{
			log.debug("Obteniendo relleno de acciones.");
			accionrol = (List<AccionRolBean>)sqlMap.queryForList(MAP +
"rellenoaccion",accionrol1);

		}catch(Exception e){
```

```
                        log.error("No fue posible obtener relleno de acciones de la
Accion: " + accionrol1.getIdaccion() + ". Error: " + e.getMessage(),e);
                        throw e;
                }
                return accionrol;
        }


    public List<AccionRolBean> rellenoRetirarAccion(AccionRolBean accionrol1) throws
Exception{
                List<AccionRolBean> accionrol = null;
                try{
                        log.debug("Obteniendo relleno retiro de acciones.");
                        accionrol = (List<AccionRolBean>)sqlMap.queryForList(MAP +
"rellenoretiraraccion",accionrol1);

                }catch(Exception e){
                        log.error("No fue posible obtener relleno retiro de acciones de la
Accion: " + accionrol1.getIdaccion() + "  Error: " + e.getMessage(),e);
                        throw e;
                }
                return accionrol;
        }


    public AccionRolBean obtenerAccionesHabilita(AccionRolBean accionrol1){
                AccionRolBean accionrol = null;
                AccionRolBean accionrolexiste = null;
                try{
                        log.debug("Obteniendo si se encuentra habilitada la accion.");
                        accionrolexiste = (AccionRolBean)sqlMap.queryForObject(MAP +
"obtenerAccionesHabilitaExiste",accionrol1);

                        if (Integer.parseInt(accionrolexiste.getExiste()) == 1){
                            accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"obtenerAccionesHabilita",accionrol1);
                        }
                        if (Integer.parseInt(accionrolexiste.getExiste()) == 0){
                                accionrol = null;
                        }


                }catch(Exception e){
                        log.error("No fue posible obtener si se encuentra habilitada la
accion: " + accionrol1.getIdaccion() + "  Error: " + e.getMessage(),e);
```

```
		}
		return accionrol;
	}


	public AccionRolBean obtenerAccionesHabilitaExiste(AccionRolBean accionrol1){
		AccionRolBean accionrol = null;
		try{
			log.debug("Obteniendo Si existe o No existe la accion padre.");
			accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"obtenerAccionesHabilitaExiste",accionrol1);
		}catch(Exception e){
			log.error("No fue posible obtener si existe o no la accion accion
padre para la Accion: " + accionrol1.getIdaccion() + ". Error: " + e.getMessage(),e);
		}
		return accionrol;
	}



	public RespuestaJsonBean actualizar (AccionRolBean accionrol){
		RespuestaJsonBean respuesta = new RespuestaJsonBean();
		respuesta.setMensaje("Cambios realizados exitosamente.");

		try {
			sqlMap.update(MAP + "actualizar", accionrol);
		} catch (Exception e) {
			respuesta.setCodigo(1);
			log.error("Error actualizando permisos. Error: " + e.getMessage(),
e);
			respuesta.setMensaje("Error realizando cambios");
		}
		return respuesta;
	}


	public RespuestaJsonBean insertar (AccionRolBean accionrol){
		RespuestaJsonBean respuesta = new RespuestaJsonBean();
		respuesta.setMensaje("Accion habilitada exitosamente");


		try {
			sqlMap.insert(MAP + "insertar", accionrol);
		} catch (Exception e) {
			respuesta.setCodigo(1);
```

```
                        log.error("Error insertando accion hija para ser habilitada. Error: "
+ e.getMessage(), e);
                        respuesta.setMensaje("Error insertando accion para ser
habilitada");
                }
                return respuesta;
        }


    public RespuestaJsonBean eliminar (AccionRolBean accionrol){
                RespuestaJsonBean respuesta = new RespuestaJsonBean();
                respuesta.setMensaje("Accion desactivada exitosamente");


                try {
                        sqlMap.delete(MAP + "eliminar", accionrol);
                } catch (Exception e) {
                        respuesta.setCodigo(1);
                        log.error("Error eliminando accion hija. Error: " + e.getMessage(),
e);
                        respuesta.setMensaje("Error eliminando accion");
                }
                return respuesta;
        }


    public AccionRolBean obtenerTodoAccion(AccionRolBean accionrol1){
        AccionRolBean accionrol = null;
                try{
                        log.debug("Obteniendo informacion de Accion.");
                        accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"obtenerTodoAccion",accionrol1);

                }catch(Exception e){
                        log.error("No fue posible obtener informacion de la Accion: " +
accionrol1.getIdaccion() + "  Error: " + e.getMessage(),e);
                }
                return accionrol;
        }


    public RespuestaJsonBean agregarAcc (AccionRolBean accionrol){
                RespuestaJsonBean respuesta = new RespuestaJsonBean();
                int idmaximo = 0;
                respuesta.setMensaje("Accion agregada exitosamente");
```

```
		try {
			idmaximo = (Integer) sqlMap.queryForObject(MAP +
"obtenerMaxAccion");
			int idmaximo1 = idmaximo + 1;
			accionrol.setIdAccion(idmaximo1);
			sqlMap.insert(MAP + "agregarAcc", accionrol);
			//sqlMap.insert(MAP + "actualizarAcc", accionrol);
		} catch (Exception e) {
			respuesta.setCodigo(1);
			log.error("Error agregando accion. Error: " + e.getMessage(), e);
			respuesta.setMensaje("Error agregando accion");
		}
		return respuesta;
	}


	public RespuestaJsonBean actualizarAcc (AccionRolBean accionrol){
		RespuestaJsonBean respuesta = new RespuestaJsonBean();
		//int idmaximo = 0;
		respuesta.setMensaje("Accion agregada exitosamente");

		try {
			//idmaximo = (Integer) sqlMap.queryForObject(MAP +
"obtenerMaxAccion");
			//int idmaximo1 = idmaximo + 1;
			//accionrol.setIdAccion(idmaximo1);
			//sqlMap.insert(MAP + "agregarAcc", accionrol);
			sqlMap.insert(MAP + "actualizarAcc", accionrol);
		} catch (Exception e) {
			respuesta.setCodigo(1);
			log.error("Error agregando accion. Error: " + e.getMessage(), e);
			respuesta.setMensaje("Error agregando accion");
		}
		return respuesta;
	}


	public RespuestaJsonBean actualizarNombreUrlAcc (AccionRolBean accionrol){
		RespuestaJsonBean respuesta = new RespuestaJsonBean();
		//int idmaximo = 0;
		respuesta.setMensaje("Accion editada exitosamente");

		try {
```

```
                        //idmaximo = (Integer) sqlMap.queryForObject(MAP +
"obtenerMaxAccion");
                        //int idmaximo1 = idmaximo + 1;
                        //accionrol.setIdAccion(idmaximo1);
                        //sqlMap.insert(MAP + "agregarAcc", accionrol);
                        sqlMap.insert(MAP + "actualizarNombreUrlAcc", accionrol);
                } catch (Exception e) {
                        respuesta.setCodigo(1);
                        log.error("Error agregando accion. Error: " + e.getMessage(), e);
                        respuesta.setMensaje("Error agregando accion");
                }
                return respuesta;
        }


    public RespuestaJsonBean retirarAcc (AccionRolBean accionrol){
                RespuestaJsonBean respuesta = new RespuestaJsonBean();
                respuesta.setMensaje("Accion retirada exitosamente");


                try {
                        sqlMap.delete(MAP + "retirarAcc", accionrol);
                } catch (Exception e) {
                        respuesta.setCodigo(1);
                        log.error("Error retirando accion. Error: " + e.getMessage(), e);
                        respuesta.setMensaje("Error retirando accion");
                }
                return respuesta;
        }

    public AccionRolBean obtenerPermisos(AccionRolBean accionrol1){
        AccionRolBean accionrol = null;
                try{
                        log.debug("Obteniendo permisos de Usuario."),
                        UsuarioBean usuario =(UsuarioBean)
ActionContext.getContext().getSession().get("usuario");
                        accionrol1.setUsuario(usuario.getUsuario());
                        accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"obtenerPermisos",accionrol1);

                }catch(Exception e){
                        log.error("No fue posible obtener permisos del Usuario: " +
accionrol1.getUsuario() + "  Error: " + e.getMessage(),e);
                }
                return accionrol;
```

```
        }

    public AccionRolBean obtenerNoAcceso(AccionRolBean accionrol1){
                AccionRolBean accionrol = null;
                try{
                        log.debug("Obteniendo Si tiene o No acceso a la accion.");
                        UsuarioBean usuario =(UsuarioBean)
ActionContext.getContext().getSession().get("usuario");
                        accionrol1.setUsuario(usuario.getUsuario());
                        accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"obtenerNoAcceso",accionrol1);
                }catch(Exception e){
                        log.error("No fue posible obtener la informacion de acceso para el
Usuario:  " + accionrol1.getUsuario() + "  Error: " + e.getMessage(),e);
                }
                return accionrol;
        }

    public List<AccionRolBean> rellenoPadre(AccionRolBean accionrol1) throws
Exception{
                List<AccionRolBean> accionrol = null;
                try{
                        log.debug("Obteniendo relleno de acciones padre."),
                        accionrol = (List<AccionRolBean>)sqlMap.queryForList(MAP +
"rellenopadre",accionrol1);

                }catch(Exception e){
                        log.error("No fue posible obtener relleno de acciones padre de la
accion: " + accionrol1.getIdaccion() + ". Error: " + e.getMessage(),e);
                        throw e;
                }
                return accionrol;
        }

    public RespuestaJsonBean insertarAccion (AccionRolBean accionrol){
                RespuestaJsonBean respuesta = new RespuestaJsonBean();
                int idmaximo = 0;
                respuesta.setMensaje("Accion agregada exitosamente");

                try {
                        idmaximo = (Integer) sqlMap.queryForObject(MAP +
"obtenerMaxAccion");
                        int idmaximo1 = idmaximo + 1;
                        accionrol.setIdAccion(idmaximo1);
                        sqlMap.insert(MAP + "insertarAccion", accionrol);
```

```
		} catch (Exception e) {
			respuesta.setCodigo(1);
			log.error("Error insertando accion. Error: " + e.getMessage(), e);
			respuesta.setMensaje("Error insertando accion");
		}
		return respuesta;
	}

	public AccionRolBean validarRetirarAcc(AccionRolBean accionrol1){
		AccionRolBean accionrol = null;
		try{
			log.debug("Obteniendo Si se puede o No eliminar la accion.");
			accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"validarRetirarAcc",accionrol1);
		}catch(Exception e){
			log.error("No fue posible eliminar la accion " +
accionrol1.getIdaccion() + ". Error: " + e.getMessage(),e);
		}
		return accionrol;
	}

	public List<AccionRolBean> validarRetirarAccRol(AccionRolBean accionrol1) throws
Exception{
		List<AccionRolBean> accionrol = null;
		try{
			log.debug("Obteniendo Roles con la accion:" +
accionrol1.getIdaccion() + " activada ");
			accionrol = (List<AccionRolBean>)sqlMap.queryForList(MAP +
"validarRetirarAccRol",accionrol1);

		}catch(Exception e){
			log.error("No fue posible obtener Roles con la accion " +
accionrol1.getIdaccion() + " activada. Error: " + e.getMessage(),e);
			throw e;
		}
		return accionrol;
	}

	public AccionRolBean hijaExiste(AccionRolBean accionrol1){
		AccionRolBean accionrol = null;
		try{
			log.debug("Obteniendo Si existen o No acciones hijas.");
			accionrol = (AccionRolBean)sqlMap.queryForObject(MAP +
"hijaExiste",accionrol1);
		}catch(Exception e){
```

```
                log.error("No fue posible obtener si existe o no acciones hijas para
la accion: " + accionrol1.getIdaccion() + ". Error: " + e.getMessage(),e);
            }
            return accionrol;
        }
```

- Modelo ***AccionRolModel***

```
    public List<AccionRolBean> consultarAccionRol(int id) throws Exception {
            return dao.consultarAccionRol(id);
    }

    public List<AccionRolBean> recuperaRol() throws Exception {
            return dao.buscaRol();
    }

    public RespuestaJsonBean eliminar(AccionRolBean accionrol){
            return dao.eliminar(accionrol);
    }

    public RespuestaJsonBean insertar(AccionRolBean accionrol){
            return dao.insertar(accionrol);
    }

    public RespuestaJsonBean actualizar(AccionRolBean accionrol){
            return dao.actualizar(accionrol);
    }

    public AccionRolBean obtener(int item){
            return dao.obtener(item);
    }

    public AccionRolBean existeAccionRol(AccionRolBean accion){
            return dao.existeAccionRol(accion);
    }

    public List<AccionRolBean> obtenerAcciones(AccionRolBean accion) throws
Exception{
            return dao.obtenerAcciones(accion);
    }

    public List<AccionRolBean> rellenoAccion(AccionRolBean accion) throws
Exception{
            return dao.rellenoAccion(accion);
```

```
	}

	public List<AccionRolBean> rellenoRetirarAccion(AccionRolBean accion) throws
Exception{
		return dao.rellenoRetirarAccion(accion);
	}

	public AccionRolBean obtenerAccionesHabilita(AccionRolBean accion){
		return dao.obtenerAccionesHabilita(accion);
	}

	public AccionRolBean obtenerAccionesHabilitaExiste(AccionRolBean accion){
		return dao.obtenerAccionesHabilitaExiste(accion);
	}

	public AccionRolBean obtenerTodoAccion(AccionRolBean accion){
		return dao.obtenerTodoAccion(accion);
	}

	public RespuestaJsonBean agregarAcc(AccionRolBean accionrol){
		return dao.agregarAcc(accionrol);
	}

	public RespuestaJsonBean actualizarAcc(AccionRolBean accionrol){
		return dao.actualizarAcc(accionrol);
	}

	public RespuestaJsonBean actualizarNombreUrlAcc(AccionRolBean accionrol){
		return dao.actualizarNombreUrlAcc(accionrol);
	}

	public RespuestaJsonBean retirarAcc(AccionRolBean accionrol){
		return dao.retirarAcc(accionrol);
	}

	public AccionRolBean obtenerPermisos(AccionRolBean accion){
		return dao.obtenerPermisos(accion);
	}

	public AccionRolBean obtenerNoAcceso(AccionRolBean accion){
		return dao.obtenerNoAcceso(accion);
	}

	public List<AccionRolBean> rellenoPadre(AccionRolBean accion) throws
Exception{
```

```
        return dao.rellenoPadre(accion);
    }

    public RespuestaJsonBean insertarAccion(AccionRolBean accionrol){
        return dao.insertarAccion(accionrol);
    }

    public AccionRolBean validarRetirarAcc(AccionRolBean accion){
        return dao.validarRetirarAcc(accion);
    }

    public List<AccionRolBean> validarRetirarAccRol(AccionRolBean accion) throws
Exception{
        return dao.validarRetirarAccRol(accion);
    }

    public AccionRolBean hijaExiste(AccionRolBean accion){
        return dao.hijaExiste(accion);
    }
```

- Vista ***AccionRolAction***

```
    public String obtenerInfo(){
        accionrol = model.obtener(accionrol.getIdaccion());
        return INFO;
    }

    public String existeAccionRol(){
        accionrol = model.existeAccionRol(accionrol);
        return "existeaccionrol";
    }

    public String obtenerAcciones() throws Exception{
        acciones = model.obtenerAcciones(accionrol);
        return "obteneracciones";
    }

    public String obtenerAccionesHabilita(){
        accionrol = model.obtenerAccionesHabilita(accionrol);
        return "obteneraccioneshabilita";
    }

    public String obtenerAccionesHabilitaExiste(){
        accionrol = model.obtenerAccionesHabilitaExiste(accionrol);
```

```
        return "obteneraccioneshabilitaexiste";
}

public String consultarAccRol() throws Exception {

        respuesta2 = model.consultarAccionRol(getId());
        return "consultaraccrol";
}

public String rellenoAccion() throws Exception{
        respuesta2 = model.rellenoAccion(accionrol);
        return "rellenoaccion";
}

public String rellenoRetirarAccion() throws Exception{
        respuesta2 = model.rellenoRetirarAccion(accionrol);
        return "rellenoretiraraccion";
}

public String relleno() throws Exception{
        respuesta2 = model.recuperaRol();
        return "relleno";
}

public String actualizar(){
        respuesta = model.actualizar(accionrol);
        return ACTUALIZAR;
}

public String insertar(){
        respuesta = model.insertar(accionrol);
        return INSERTAR;
}

public String eliminar(){
        respuesta = model.eliminar(accionrol);
        return ELIMINAR;
}

public String obtenerTodoAccion(){
        accionrol = model.obtenerTodoAccion(accionrol);
        return "obtenertodoaccion";
}
```

```
public String agregarAcc(){
        respuesta = model.agregarAcc(accionrol);
        return "agregaracc";
}

public String actualizarAcc(){
        respuesta = model.actualizarAcc(accionrol);
        return "agregaracc";
}

public String actualizarNombreUrlAcc(){
        respuesta = model.actualizarNombreUrlAcc(accionrol);
        return "agregaracc";
}

public String retirarAcc(){
        respuesta = model.retirarAcc(accionrol);
        return "retiraracc";
}

public String obtenerPermisos(){
        accionrol = model.obtenerPermisos(accionrol);
        return "obtenerpermisos";
}

public String obtenerNoAcceso(){
        accionrol = model.obtenerNoAcceso(accionrol);
        return "obtenernoacceso";
}

public String rellenoPadre() throws Exception{
        respuesta2 = model.rellenoPadre(accionrol);
        return "relleno";
}

public String insertarAccion(){
        respuesta = model.insertarAccion(accionrol);
        return "agregaracc";
}

public String validarRetirarAcc(){
        accionrol = model.validarRetirarAcc(accionrol);
        return "obtenerpermisos";
}
```

- DAO ***FacturaDAO***

```
public List<FacturaBean> listarFacturas()throws Exception {
		try {
			logger.debug("Obteniendo facturas.");
			return sqlMap.queryForList("FacturaMap.listarFacturas");


		} catch (Exception e) {

			logger.error("Error Obteniendo facturas. Error: " +
e.getMessage(), e);
			throw e;

		}
	}


	public FacturaBean existeFactura(){
		FacturaBean existefactura = null;
		try{
			logger.debug("Obteniendo Si existen o No existen facturas.");
			existefactura =
(FacturaBean)sqlMap.queryForObject("FacturaMap.existeFactura");
		}catch(Exception e){
			logger.error("No fue posible obtener si existen o no las facturas.
Error: " + e.getMessage(),e);
		}
		return existefactura;
	}
```

- Modelo ***FacturaModel***

```
	public List<FacturaBean> listarFacturas()throws Exception
	{
		return dao.listarFacturas();
	}

	public FacturaBean existeFactura(){
		return dao.existeFactura();
	}
```

- Vista ***FacturaAction***

```
public String listarFacturas() throws Exception {
        respuesta2 = model.listarFacturas();
        return "reimprimir";
}

public String existeFacturas(){
        consultarFactura = model.existeFactura();
        return "reimprimir";
}
```

- DAO ***AdminDetallePagoDAO***

```
public MerchantBean obtenerMerchants(int id) throws Exception {
        try {
                if (log.isDebugEnabled())
                        log.debug("Obteniendo merchants con el ID [ " + id + " ].
"),
                return (MerchantBean)
sqlMap.queryForObject("detallePagoMap.obtenerMerchant", id);
        } catch (Exception e) {
                log.error("Error obteniendo merchants, con el ID [ " + id + " ]. ");
                throw e;
        }
}

public List<DetallePagoBean> consultarRechazadas() throws Exception {
        try {
                if (log.isDebugEnabled())
                        log.debug("Obteniendo detalle, de la tabla DetallePago. "),
                return
sqlMap.queryForList("detallePagoMap.obtenerConsultarRechazadas");
        } catch (Exception e) {
                log.error("Error obteniendo detalle, de la tabla DetallePago. " +
e.getMessage());
                throw e;
        }

}
```

- Modelo ***AdminDetallePagoModel***

```
public MerchantBean obtenerMerchants(int id) throws Exception {
		return dao.obtenerMerchants(id);
	}
public  List<DetallePagoBean> consultarRechazadas() throws Exception {
		return dao.consultarRechazadas();
	}
```

- Vista ***PagoAction***

```
public String obtenerMerchants() throws Exception {
		merchant = model.obtenerMerchants(id);
		return "merchant";
	}
public String consultarRechazadas() throws Exception {
		respuesta6 = model.consultarRechazadas();
		return "datosEditar2";
	}
```

- DAO ***AprobarDAO***

```
public List<TerminalBean> getTerminalLista(int retailerID, int merchantId, int
idSolicitud, int VterminalID) throws SQLException {

		try {
			List<TerminalBean> merchantPays =
sqlMapMerchant1.queryForList("M_AprobarMap.obtenerVTerminalTerminals",
VterminalID);
			return merchantPays;

		} catch (Exception e) {
			log.error("Error al obtener los merchants para el retailer " +
retailerID + "  Error: " + e.getMessage(), e);
			throw new SQLException("Error al obtener los merchants para el
retailer " + retailerID + ". Error: " + e.getMessage(), e);
		}
	}

public RespuestaJsonBean insertarTerminal(String TID,int terminalId, int VterminalID,
int idSolicitud) {
```

```
RespuestaJsonBean res = new RespuestaJsonBean();
try {

ReservaTerminalsBean reservaBean = new
ReservaTerminalsBean();

reservaBean.setTID(TID);
reservaBean.setTerminalId(terminalId);
System.out.println(terminalId);
System.out.println(reservaBean.getTerminalId());

reservaBean.setMID((String)sqlMapMerchant1.queryForObject("M_AprobarMap
.obtenerMID", reservaBean.getTerminalId()));

reservaBean.setIdVterminalSolicitud((Integer)sqlMap.queryForObject("Aprobar
Map.obtenerIdVterminalSolicitud", idSolicitud));

String existe = (String)
sqlMap.queryForObject("AprobarMap.obtenerTerminalExiste", TID);

if (existe.equals("no")){

sqlMap.update("AprobarMap.insertReserva", reservaBean);

}

res.setCodigo(0);
res.setMensaje("Terminal(es) insertados exitosamente."),

} catch (Exception e) {

log.error("Error al insertar terminal. Terminal ["+ TID +"]. Error:
"+e.getMessage(), e);
res.setCodigo(0);
res.setMensaje("Error al insertar terminal(es)."+e.getMessage());
}

return res;
}


public List<VTerminalBean> getVterminals(int merchantId) {


try {
```

```
                List<Integer> listaPagoAqui=new ArrayList<Integer>();
                listaPagoAqui=
sqlMap.queryForList("AprobarMap.getVTerminals", merchantId);
                List<VTerminalBean> respuesta=new ArrayList<VTerminalBean>();
                String Valores;
                if(!listaPagoAqui.isEmpty()){
                        Valores=String.valueOf(listaPagoAqui);
                        Valores=Valores.replace("[", "");
                        Valores=Valores.replace("]", "");
                }else{
                        Valores="0";
                        }
                log.debug("Valores  ["+listaPagoAqui+"]");
                Connection con = null;

                try {
                        String query = " select " +
                                        " VterminalID,VTID " +
                                        " From VTerminals as v,Groups as" +
                                        " g,Merchants as m " +
                                        " Where " +
                                        " v.GroupID= g.GroupID " +
                                        " and " +
                                        " m.MerchantID=g.MerchantID " +
                                        " and " +
                                        " m.MerchantID="+merchantId;
//                                      +
//                                      " and " +
//                                      " v.vTerminalId not in("+Valores+")";

                        con = getDataSourceMerchant().getConnection();
                        Statement  stm = con.createStatement();

                        if(log.isDebugEnabled())
                                log.debug(query);

                        ResultSet rs = stm.executeQuery(query);

                        while (rs.next()) {
                                VTerminalBean vt=new VTerminalBean();
                                vt.setvTerminalID(rs.getInt("vTerminalID"));
                                vt.setVTID(rs.getString("VTID"));
                                respuesta.add(vt);
                        }
```

```
                } catch (Exception e) {
                        log.error("Error al obtener Datos del VTerminal para
aprobar . Error: " + e.getMessage(), e);
                } finally {
                        if (con != null) {
                                con.close();
                        }
                }


//
        listaMerchan=sqlMapMerchant1.queryForList("M_AprobarMap.getVTerminals",
Valores);

                log.debug("Retornando Valores  ["+respuesta.size()+"]");
                return respuesta;

        } catch (Exception e) {

                log.error("Error al obtener los VTerminals del merchant [" +
merchantId + "] error: " + e.getMessage(), e);
        }

        return null;
    }
```

- Modelo ***AprobarModel***

```
public List<VTerminalBean> getVTerminals(int merchantId) {
        return dao.getVterminals(merchantId);
    }

    public RespuestaJsonBean insertarTerminal(String TID, int terminalId, int
VterminalID, int idSolicitud) {

        return dao.insertarTerminal(TID,terminalId,VterminalID,idSolicitud);
    }

public List<TerminalBean> obtenerTerminalLista(int retailerID, int merchantId, int
idSolicitud, int VterminalID) throws Exception {
        return dao.getTerminalLista(retailerID, merchantId,
idSolicitud,VterminalID);

    }
```

- Vista ***AprobarAction***

```
public String vterminals() throws Exception {
		vterminals = model.getVTerminals(merchantId);
		return "vterminals";
	}

public String obtenerTerminalesLista() throws Exception {
		terminalLista = model.obtenerTerminalLista(retailerID,
merchantId,idSolicitud,VterminalID);
		return "vterminals";
	}

public String insertarTerminal(){
		respuesta =
model.insertarTerminal(TID,terminalId,VterminalID,idSolicitud);
		return "rechazar";

	}
```

- DAO ***ReporteConciliarDAO***

```
public List<ReporteConciliarBean> obtenerReporteConciliar(ReporteConciliarBean
reporteconciliar) throws Exception {
		try {
			if(log.isDebugEnabled())
			log.debug("Obteniendo todos los registros del Reporte
Conciliar.");

			List<ReporteConciliarBean> resultado = null;

			reporteconciliar.setFecha(reporteconciliar.getFecha().substring(0,
4)+"-"+reporteconciliar.getFecha().substring(4, 6)+"-
"+reporteconciliar.getFecha().substring(6, 8));

			System.out.println(reporteconciliar.getFecha());

			resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliar",reporteconciliar);

			return resultado;
		} catch (SQLException e) {
```

```
                        log.error("Error obteniendo lista de Reporte Conciliar." +
e.getMessage(), e);
                        throw e;
                }

        }

public List<ReporteConciliarBean>
obtenerReporteConciliarSummaryMovimiento(ReporteConciliarBean reporteconciliar)
throws Exception {
                try {
                        if(log.isDebugEnabled())
                        log.debug("Obteniendo todos los registros del Reporte Conciliar
Summary Movimiento.");

                        List<ReporteConciliarBean> resultado = null;

                        resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarSummary");

                        for (ListIterator<ReporteConciliarBean>
its=resultado.listIterator();its.hasNext();) {

                                ReporteConciliarBean reporteconciliar1 = its.next();


reporteconciliar1.setFecha(reporteconciliar.getFecha().substring(0, 4)+"
"+reporteconciliar.getFecha().substring(4, 6)+"-
"+reporteconciliar.getFecha().substring(6, 8));

                                System.out.println(reporteconciliar1.getFecha());


reporteconciliar1.setCountconciliada((Integer)sqlMap.queryForObject(MAP+"obtenerRe
porteConciliarSummaryMovimientoCOUNTCONCILIADA",reporteconciliar1));

reporteconciliar1.setCountparcialconciliada((Integer)sqlMap.queryForObject(MAP+"obt
enerReporteConciliarSummaryMovimientoCOUNTPARCIALCONCILIADA",reporteconcilia
r1));

reporteconciliar1.setMontoconciliada((Float)sqlMap.queryForObject(MAP+"obtenerRep
orteConciliarSummaryMovimientoMONTOCONCILIADA",reporteconciliar1));

reporteconciliar1.setMontoparcialconciliadabanco((Float)sqlMap.queryForObject(MAP+
```

```
"obtenerReporteConciliarSummaryMovimientoMONTOPARCIALCONCILIADABANCO",re
porteconciliar1));

reporteconciliar1.setMontoparcialconciliadamerchant((Float)sqlMap.queryForObject(M
AP+"obtenerReporteConciliarSummaryMovimientoMONTOPARCIALCONCILIADAMERCH
ANT",reporteconciliar1));

                                int noconciliadabanco =
(Integer)sqlMap.queryForObject(MAP+"obtenerReporteConciliarSummaryMovimientoC
OUNTNOCONCILIADABANCO",reporteconciliar1);

                                int montonoconciliadabanco =
(Integer)sqlMap.queryForObject(MAP+"obtenerReporteConciliarSummaryMovimiento
MONTONOCONCILIADABANCO",reporteconciliar1);


reporteconciliar1.setMontonoconciliada(montonoconciliadabanco);

reporteconciliar1.setCountnoconciliada(noconciliadabanco);

                        }



                        return resultado;
                } catch (SQLException e) {
                        log.error("Error obteniendo lista de Reporte Conciliar Summary
Movimiento." + e.getMessage(), e);
                        throw e;
                }

        }


public List<ReporteConciliarBean>
obtenerReporteConciliarSummaryCierre(ReporteConciliarBean reporteconciliar) throws
Exception {
                try {
                        if(log.isDebugEnabled())
                        log.debug("Obteniendo todos los registros del Reporte Conciliar
Summary Cierre."),

                        List<ReporteConciliarBean> resultado = null;

                        resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarSummary");
```

```
for (ListIterator<ReporteConciliarBean>
its=resultado.listIterator();its.hasNext();) {

ReporteConciliarBean reporteconciliar1 = its.next();


reporteconciliar1.setFecha(reporteconciliar.getFecha().substring(0, 4)+"-
"+reporteconciliar.getFecha().substring(4, 6)+"-
"+reporteconciliar.getFecha().substring(6, 8));

System.out.println(reporteconciliar1.getFecha());


reporteconciliar1.setCountconciliada((Integer)sqlMap.queryForObject(MAP+"obtenerRe
porteConciliarSummaryCierreCOUNTCONCILIADA",reporteconciliar1));

reporteconciliar1.setCountparcialconciliada((Integer)sqlMap.queryForObject(MAP+"obt
enerReporteConciliarSummaryCierreCOUNTPARCIALCONCILIADA",reporteconciliar1));

reporteconciliar1.setMontoconciliada((Float)sqlMap.queryForObject(MAP+"obtenerRep
orteConciliarSummaryCierreMONTOCONCILIADA",reporteconciliar1));

reporteconciliar1.setMontoparcialconciliadabanco((Float)sqlMap.queryForObject(MAP+
"obtenerReporteConciliarSummaryCierreMONTOPARCIALCONCILIADABANCO",reportec
onciliar1));

reporteconciliar1.setMontoparcialconciliadamerchant((Float)sqlMap.queryForObject(M
AP+"obtenerReporteConciliarSummaryCierreMONTOPARCIALCONCILIADAMERCHANT",r
eporteconciliar1));

//reporteconciliar.setSumamonto((Integer)sqlMap.queryForObject(MAP+"obtenerRepo
rteConciliarSummarySUMAMONTO",reporteconciliar));

int noconciliadamerchant =
(Integer)sqlMap.queryForObject(MAP+"obtenerReporteConciliarSummaryCierreCOUNT
NOCONCILIADAMERCHANT",reporteconciliar1);

int montonoconciliadamerchant =
(Integer)sqlMap.queryForObject(MAP+"obtenerReporteConciliarSummaryCierreMONTO
NOCONCILIADAMERCHANT",reporteconciliar1);


reporteconciliar1.setMontonoconciliada(montonoconciliadamerchant);
```

```
reporteconciliar1.setCountnoconciliada(noconciliadamerchant);

                }

                return resultado;
        } catch (SQLException e) {
                log.error("Error obteniendo lista de Reporte Conciliar Summary
Cierre." + e.getMessage(), e);
                throw e;
        }

    }


public ReporteConciliarBean
obtenerReporteConciliarSummaryData(ReporteConciliarBean reporteconciliar) throws
Exception {
        try {
                if(log.isDebugEnabled())
                log.debug("Obteniendo todos los registros del Reporte Conciliar
Summary Data."),

                ReporteConciliarBean resultado = new ReporteConciliarBean();


    resultado.setCountconciliada((Integer)sqlMap.queryForObject(MAP+"obtenerRe
porteConciliarSummaryCOUNTCONCILIADA",reporteconciliar));

    //resultado.setSumamonto((Integer)sqlMap.queryForObject(MAP+"obtenerRep
orteConciliarSummarySUMACOUNT",reporteconciliar));

    resultado.setCountparcialconciliada((Integer)sqlMap.queryForObject(MAP+"obt
enerReporteConciliarSummaryCOUNTPARCIALCONCILIADA",reporteconciliar));

                System.out.println(resultado.getCountconciliada() + "  " +
resultado.getCountparcialconciliada() + "   " + resultado.getSumamonto());

                return resultado;
        } catch (SQLException e) {
                log.error("Error obteniendo lista de Reporte Conciliar Summary
Data." + e.getMessage(), e);
                throw e;
        }
```

```
}

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleConciliadaMovimiento(ReporteConciliarBean
reporteconciliar) throws Exception {
        try {
                if(log.isDebugEnabled())
                log.debug("Obteniendo todos los registros del Reporte Detalle
Conciliada Movimiento."),

                List<ReporteConciliarBean> resultado = null;

                resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarDetalleConciliadaMovimiento",rep
orteconciliar);

                return resultado;
        } catch (SQLException e) {
                log.error("Error obteniendo lista de Reporte Detalle Conciliar
Movimiento." + e.getMessage(), e);
                throw e;
        }

}


public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleParcialConciliadaMovimiento(ReporteConciliarBean
reporteconciliar) throws Exception {
        try {
                if(log.isDebugEnabled())
                log.debug("Obteniendo todos los registros del Reporte Detalle
Parcial Conciliada Movimiento.");

                List<ReporteConciliarBean> resultado = null;

                resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarDetalleParcialConciliadaMovimient
o",reporteconciliar);

                return resultado;
        } catch (SQLException e) {
                log.error("Error obteniendo lista de Reporte Detalle Parcial
Conciliar Movimiento." + e.getMessage(), e);
```

```
                        throw e;
                }

        }


public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleNoConciliadaMovimiento(ReporteConciliarBean
reporteconciliar) throws Exception {
                try {
                        if(log.isDebugEnabled())
                        log.debug("Obteniendo todos los registros del Reporte Detalle No
Conciliada Movimiento.");

                        List<ReporteConciliarBean> resultado = null;

                        resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarDetalleNoConciliadaMovimiento",r
eporteconciliar);

                        return resultado;
                } catch (SQLException e) {
                        log.error("Error obteniendo lista de Reporte Detalle No Conciliar
Movimiento." + e.getMessage(), e);
                        throw e;
                }

        }


public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleConciliadaCierre(ReporteConciliarBean
reporteconciliar) throws Exception {
                try {
                        if(log.isDebugEnabled())
                        log.debug("Obteniendo todos los registros del Reporte Detalle
Conciliada Cierre.");

                        List<ReporteConciliarBean> resultado = null;

                        resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarDetalleConciliadaCierre",reporteco
nciliar);

                        return resultado;
```

```
		} catch (SQLException e) {
			log.error("Error obteniendo lista de Reporte Detalle Conciliar
Cierre." + e.getMessage(), e);
			throw e;
		}

	}


public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleParcialConciliadaCierre(ReporteConciliarBean
reporteconciliar) throws Exception {
		try {
			if(log.isDebugEnabled())
			log.debug("Obteniendo todos los registros del Reporte Detalle
Parcial Conciliada Cierre.");

			List<ReporteConciliarBean> resultado = null;

			resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarDetalleParcialConciliadaCierre",rep
orteconciliar);

			return resultado;
		} catch (SQLException e) {
			log.error("Error obteniendo lista de Reporte Detalle Parcial
Conciliar Cierre." + e.getMessage(), e);
			throw e;
		}

	}


public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleNoConciliadaCierre(ReporteConciliarBean
reporteconciliar) throws Exception {
		try {
			if(log.isDebugEnabled())
			log.debug("Obteniendo todos los registros del Reporte Detalle No
Conciliada Cierre.");

			List<ReporteConciliarBean> resultado = null;
```

```
                resultado =
sqlMap.queryForList(MAP+"obtenerReporteConciliarDetalleNoConciliadaCierre",reporte
conciliar);

                return resultado;
        } catch (SQLException e) {
                log.error("Error obteniendo lista de Reporte Detalle No Conciliar
Cierre." + e.getMessage(), e);
                throw e;
        }

}
```

- Modelo ***ReporteConciliarModel***

```
public List<ReporteConciliarBean> obtenerReporteConciliar(ReporteConciliarBean
reporteconciliar) throws Exception {
        return dao.obtenerReporteConciliar(reporteconciliar);
}

public List<ReporteConciliarBean>
obtenerReporteConciliarSummaryMovimiento(ReporteConciliarBean reporteconciliar)
throws Exception {
        return
dao.obtenerReporteConciliarSummaryMovimiento(reporteconciliar);
}

public List<ReporteConciliarBean>
obtenerReporteConciliarSummaryCierre(ReporteConciliarBean reporteconciliar) throws
Exception {
        return dao.obtenerReporteConciliarSummaryCierre(reporteconciliar);
}

public ReporteConciliarBean
obtenerReporteConciliarSummaryData(ReporteConciliarBean reporteconciliar) throws
Exception {
        return dao.obtenerReporteConciliarSummaryData(reporteconciliar);
}

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleConciliadaMovimiento(ReporteConciliarBean
reporteconciliar) throws Exception {
        return
dao.obtenerReporteConciliarDetalleConciliadaMovimiento(reporteconciliar);
```

```
        }

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleParcialConciliadaMovimiento(ReporteConciliarBean
reporteconciliar) throws Exception {
                return
dao.obtenerReporteConciliarDetalleParcialConciliadaMovimiento(reporteconciliar);
        }

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleNoConciliadaMovimiento(ReporteConciliarBean
reporteconciliar) throws Exception {
                return
dao.obtenerReporteConciliarDetalleNoConciliadaMovimiento(reporteconciliar);
        }

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleConciliadaCierre(ReporteConciliarBean
reporteconciliar) throws Exception {
                return
dao.obtenerReporteConciliarDetalleConciliadaCierre(reporteconciliar);
        }

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleParcialConciliadaCierre(ReporteConciliarBean
reporteconciliar) throws Exception {
                return
dao.obtenerReporteConciliarDetalleParcialConciliadaCierre(reporteconciliar);
        )

public List<ReporteConciliarBean>
obtenerReporteConciliarDetalleNoConciliadaCierre(ReporteConciliarBean
reporteconciliar) throws Exception {
                return
dao.obtenerReporteConciliarDetalleNoConciliadaCierre(reporteconciliar);
        }
```

- Vista ***ReporteConciliarAction***

```
public String obtenerReporteConciliar() throws Exception{

                respuesta2 = model.obtenerReporteConciliar(reporteconciliar);
                return "relleno";
```

```
        }

public String obtenerReporteConciliarSummaryMovimiento() throws Exception{
                respuesta2 =
model.obtenerReporteConciliarSummaryMovimiento(reporteconciliar);
                return "relleno",

        }

public String obtenerReporteConciliarSummaryCierre() throws Exception{
                respuesta2 =
model.obtenerReporteConciliarSummaryCierre(reporteconciliar);
                return "relleno",

        }

public String obtenerReporteConciliarSummaryData() throws Exception{
                reporteconciliar =
model.obtenerReporteConciliarSummaryData(reporteconciliar);
                return "relleno";

        }

public String obtenerReporteConciliarDetalleConciliadaMovimiento() throws
Exception{
                respuesta2 =
model.obtenerReporteConciliarDetalleConciliadaMovimiento(reporteconciliar);
                return "relleno";

        }

public String obtenerReporteConciliarDetalleParcialConciliadaMovimiento() throws
Exception{
                respuesta2 =
model.obtenerReporteConciliarDetalleParcialConciliadaMovimiento(reporteconciliar);
                return "relleno";

        }

public String obtenerReporteConciliarDetalleNoConciliadaMovimiento() throws
Exception{
                respuesta2 =
model.obtenerReporteConciliarDetalleNoConciliadaMovimiento(reporteconciliar);
                return "relleno";
```

```
}

public String obtenerReporteConciliarDetalleConciliadaCierre() throws Exception{
		respuesta2 =
model.obtenerReporteConciliarDetalleConciliadaCierre(reporteconciliar);
		return "relleno",

}

public String obtenerReporteConciliarDetalleParcialConciliadaCierre() throws
Exception{
		respuesta2 =
model.obtenerReporteConciliarDetalleParcialConciliadaCierre(reporteconciliar);
		return "relleno";

}

public String obtenerReporteConciliarDetalleNoConciliadaCierre() throws Exception{
		respuesta2 =
model.obtenerReporteConciliarDetalleNoConciliadaCierre(reporteconciliar);
		return "relleno",

}
```

- DAO ***MovimientosCuentaDAO***

```
public MovimientosCuentaBean obtenerMovimientoBanco(int id) {
		MovimientosCuentaBean mvtBanco = null;
		try{
			mvtBanco =
(MovimientosCuentaBean)sqlMap.queryForObject(MAP +
"obtenerMvtEdoCuentaBancoConId", id);
		}catch (Exception e){
			log.error("Error obteniendo el movimiento del estado de cuenta.
Error: " + e.getMessage());
		}
		return mvtBanco;
}
```

- Modelo ***MovimientosCuentaModel***

```
public MovimientosCuentaBean obtenerMovimientoBanco(int id){
		return dao.obtenerMovimientoBanco(id);
```

}

- Vista ***MovimientosCuentaAction***

```
public String consultarReporteMovimientoRechazo() throws Exception {
		respuesta = (MovimientosCuentaBean)
model.obtenerMovimientoBanco(getId());
		return "conciliar";
	}
```

- DAO ***ProcesoCuadreDAO***

```
public List<ProcesoCuadreBean> buscaDetalles(ProcesoCuadreBean cuadre) throws
Exception {
		try {
			if(log.isDebugEnabled())
			log.debug("Obteniendo todos los Detalles.");

			List<ProcesoCuadreBean> resultado = null;

			DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
			Date date = new Date();

			cuadre.setCiclo_inicio((String) sqlMap.queryForObject(MAP +
"obtenerCicloInicio", dateFormat.format(date)));
			cuadre.setCiclo_fin((String) sqlMap.queryForObject(MAP +
"obtenerCicloFin", dateFormat.format(date)));

			cuadre.setCiclo_inicio(cuadre.getCiclo_inicio().replace("-", ""));
			cuadre.setCiclo_fin(cuadre.getCiclo_fin().replace("-", ""));

			cuadre.setCiclo_inicio("20130712");
			cuadre.setCiclo_fin("20130716");

			System.out.println(cuadre.getCiclo_inicio() +" - "+
cuadre.getCiclo_fin());

			MerchantBean merchant = (MerchantBean)
sqlMap.queryForObject("UsuarioMap." + "obtenermerchantPorSolicitud",
cuadre.getIdsolicitud());
```

```
VTerminalBean vterminal = (VTerminalBean)
sqlMap.queryForObject(MAP + "obtenerVterminalSolicitud", cuadre.getIdsolicitud());

cuadre.setVterminalid(vterminal.getvTerminalID());

cuadre.setMerchantid(merchant.getMerchantID());

if (cuadre.getEstado() == 1){
        resultado =
sqlMap.queryForList(MAP+"obtenerDetallesAprobada",cuadre);
}
if (cuadre.getEstado() == 2){
        resultado =
sqlMap.queryForList(MAP+"obtenerDetallesCancelada",cuadre);
}
if (cuadre.getEstado() == 3){
        resultado =
sqlMap.queryForList(MAP+"obtenerDetallesTodas",cuadre);
}

return resultado;
} catch (SQLException e) {
        log.error("Error obteniendo lista de Detalles." + e.getMessage(),
e);
        throw e;
}

}


public List<ProcesoCuadreBean> buscaDetallesBatch(ProcesoCuadreBean cuadre)
throws Exception {
        try {
                if(log.isDebugEnabled())
                log.debug("Obteniendo todos los Detalles Batch.");

                List<ProcesoCuadreBean> resultado = null;

                DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
                Date date = new Date();

                cuadre.setCiclo_inicio((String) sqlMap.queryForObject(MAP +
"obtenerCicloInicio", dateFormat.format(date)));
                cuadre.setCiclo_fin((String) sqlMap.queryForObject(MAP +
"obtenerCicloFin", dateFormat.format(date)));
```

```
                        cuadre.setCiclo_inicio(cuadre.getCiclo_inicio().replace("-", ""));
                        cuadre.setCiclo_fin(cuadre.getCiclo_fin().replace("-", ""));

                        cuadre.setCiclo_inicio("20130712");
                        cuadre.setCiclo_fin("20130716");

                        System.out.println(cuadre.getCiclo_inicio() +" - "+
cuadre.getCiclo_fin());

                        MerchantBean merchant = (MerchantBean)
sqlMap.queryForObject("UsuarioMap." + "obtenermerchantPorSolicitud",
cuadre.getIdsolicitud());

                        VTerminalBean vterminal = (VTerminalBean)
sqlMap.queryForObject(MAP + "obtenerVterminalSolicitud", cuadre.getIdsolicitud());

                        cuadre.setVterminalid(vterminal.getvTerminalID());

                        cuadre.setMerchantid(merchant.getMerchantID());

                        if (cuadre.getEstado() == 1){
                                resultado =
sqlMap.queryForList(MAP+"obtenerDetallesBatchAprobada",cuadre);
                        }
                        if (cuadre.getEstado() == 2){
                                resultado =
sqlMap.queryForList(MAP+"obtenerDetallesBatchCancelada",cuadre);
                        }
                        if (cuadre.getEstado() == 3){
                                resultado =
sqlMap.queryForList(MAP+"obtenerDetallesBatchTodas",cuadre);
                        }

                        return resultado;
                } catch (SQLException e) {
                        log.error("Error obteniendo lista de Detalles Batch." +
e.getMessage(), e);
                        throw e;
                }

        }


        @SuppressWarnings("unused")
```

```
	public ProcesoCuadreBean buscaDetallesSuma(ProcesoCuadreBean cuadre)
throws Exception {
		try {
			if(log.isDebugEnabled())
			log.debug("Obteniendo todos los Detalles Batch.");

			String sumaString = "";
			List<ProcesoCuadreBean> resultado;
			List<ProcesoCuadreBean> resultado1;
			ProcesoCuadreBean cuadreTotal = new ProcesoCuadreBean();;
			float suma = 0;
			DateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
			Date date = new Date();

			cuadre.setCiclo_inicio((String) sqlMap.queryForObject(MAP +
"obtenerCicloInicio", dateFormat.format(date)));
			cuadre.setCiclo_fin((String) sqlMap.queryForObject(MAP +
"obtenerCicloFin", dateFormat.format(date)));

			cuadre.setCiclo_inicio(cuadre.getCiclo_inicio().replace("-", ""));
			cuadre.setCiclo_fin(cuadre.getCiclo_fin().replace("-", ""));

			cuadre.setCiclo_inicio("20130712");
			cuadre.setCiclo_fin("20130716");

			MerchantBean merchant = (MerchantBean)
sqlMap.queryForObject("UsuarioMap." + "obtenermerchantPorSolicitud",
cuadre.getIdsolicitud());

			VTerminalBean vterminal = (VTerminalBean)
sqlMap.queryForObject(MAP + "obtenerVterminalSolicitud", cuadre.getIdsolicitud());

			cuadre.setVterminalid(vterminal.getvTerminalID());

			cuadre.setMerchantid(merchant.getMerchantID());

			if (cuadre.getEstado() == 1){
				resultado =
sqlMap.queryForList(MAP+"obtenerDetallesBatchAprobada",cuadre);
				resultado1 =
sqlMap.queryForList(MAP+"obtenerDetallesAprobada",cuadre);

				for (ListIterator<ProcesoCuadreBean>
its=resultado.listIterator();its.hasNext();) {
```

```
                                        ProcesoCuadreBean procesocuadre = its.next();

                                        suma =
Float.parseFloat(procesocuadre.getMonto()) + suma;

                                }

                                for (ListIterator<ProcesoCuadreBean>
its=resultado1.listIterator();its.hasNext();) {

                                        ProcesoCuadreBean procesocuadre1 = its.next();

                                        suma =
Float.parseFloat(procesocuadre1.getMonto()) + suma;

                                }
                        }
                        if (cuadre.getEstado() == 2){
                                resultado =
sqlMap.queryForList(MAP+"obtenerDetallesBatchCancelada",cuadre);
                                resultado1 =
sqlMap.queryForList(MAP+"obtenerDetallesCancelada",cuadre);

                                for (ListIterator<ProcesoCuadreBean>
its=resultado.listIterator();its.hasNext();) {

                                        ProcesoCuadreBean procesocuadre = its.next();

                                        //suma =
Float.parseFloat(procesocuadre.getMonto()) + suma;

                                        suma = 0;
                                }

                                for (ListIterator<ProcesoCuadreBean>
its=resultado1.listIterator();its.hasNext();) {

                                        ProcesoCuadreBean procesocuadre1 = its.next();

                                        //suma =
Float.parseFloat(procesocuadre1.getMonto()) + suma;

                                        suma = 0;
                                }
                        }
```

```
                    if (cuadre.getEstado() == 3){
                        resultado =
sqlMap.queryForList(MAP+"obtenerDetallesBatchTodas",cuadre);
                        resultado1 =
sqlMap.queryForList(MAP+"obtenerDetallesTodas",cuadre);

                        for (ListIterator<ProcesoCuadreBean>
its=resultado.listIterator();its.hasNext();) {

                            ProcesoCuadreBean procesocuadre = its.next();

                            if ((procesocuadre.getRespcode().equals("00")) &&
(procesocuadre.getReversar() == 0) && (procesocuadre.getAnulada() == 0)){

                            suma =
Float.parseFloat(procesocuadre.getMonto()) + suma;

                            }

                        }

                        for (ListIterator<ProcesoCuadreBean>
its=resultado1.listIterator();its.hasNext();) {

                            ProcesoCuadreBean procesocuadre1 = its.next();

                            if ((procesocuadre1.getRespcode().equals("00"))
&& (procesocuadre1.getReversar() == 0) && (procesocuadre1.getAnulada() == 0)){

                            suma =
Float.parseFloat(procesocuadre1.getMonto()) + suma;

                            }

                        }
                    }

                    sumaString = String.format("%.2f", suma);

                    cuadreTotal.setMonto(sumaString);


                    return cuadreTotal;
                } catch (SQLException e) {
```

```
                    log.error("Error obteniendo lista de Detalles Batch." +
e.getMessage(), e);
                    throw e;
            }

    }
```

- Modelo ***ProcesoCuadreModel***

```
public List<ProcesoCuadreBean> buscarDetalles(ProcesoCuadreBean cuadre) throws
Exception {
            return dao.buscaDetalles(cuadre);
    }

public List<ProcesoCuadreBean> buscarDetallesBatch(ProcesoCuadreBean cuadre)
throws Exception {
            return dao.buscaDetallesBatch(cuadre);
    }

public ProcesoCuadreBean buscarDetallesSuma(ProcesoCuadreBean cuadre) throws
Exception {
            return dao.buscaDetallesSuma(cuadre);
    }
```

- Vista ***ProcesoCuadreAction***

```
public String buscarDetalles() throws Exception{

            respuesta2 = model.buscarDetalles(procesocuadre);

            for (ListIterator<ProcesoCuadreBean>
its=respuesta2.listIterator();its.hasNext();) {

                    ProcesoCuadreBean procesocuadre = its.next();

                    String pan = procesocuadre.getTarjeta();
                    pan = util.cipher3Des(pan);
                    pan = MyString.ajustaAstRight(pan.substring(0, 6), pan.length() -
4)+ pan.substring(pan.length() - 4, pan.length());
                    procesocuadre.setTarjeta(pan);

            }
```

```
            return "relleno";

        }

public String buscarDetallesBatch() throws Exception{

            respuesta2 = model.buscarDetallesBatch(procesocuadre);

            for (ListIterator<ProcesoCuadreBean>
its=respuesta2.listIterator();its.hasNext();) {

                ProcesoCuadreBean procesocuadre = its.next();

                String pan = procesocuadre.getTarjeta();
                pan = util.cipher3Des(pan);
                pan = MyString.ajustaAstRight(pan.substring(0, 6), pan.length()
4)+ pan.substring(pan.length() - 4, pan.length());
                procesocuadre.setTarjeta(pan);

            }

            return "relleno";

        }


public String buscarDetallesSuma() throws Exception{

            procesocuadre = model.buscarDetallesSuma(procesocuadre);
            return "relleno";

        }
```

Apéndice 4

(Reportes y Archivos)

- Ejemplo de archivo Excel para el banco BanPlus

Fecha	Referencia	Descripción	Débito	Crédito	Saldo
26/09/2014	22900042509	Liquidación Ventas TDC		311,81	390.777,15
26/09/2014	31900012509	Liquidación Ventas TDC		140,45	390.465,34
26/09/2014	21100062509	Liquidación Ventas TDC		468,18	390.324,89
26/09/2014	2681451001	Recargo alto valor	100		389.856,71
26/09/2014	2681451001	Comisión convenio	35		389.956,71
26/09/2014	2681451001	Pago a proveedor otros Bancos	1.915.416,92		389.991,71
25/09/2014	230100052409	Liquidación Ventas TDD		3.241,88	2.305.408,63
25/09/2014	228600172409	Liquidación Ventas TDD		841,1	2.302.166,75
25/09/2014	227400202409	Liquidación Ventas TDD		897,75	2.301.325,65

Figura 46. Archivo Excel BanPlus. (Banco BanPlus)

- Ejemplo de archivo Excel para el banco Banesco

Fecha	Ref	Descripción	Monto	Balance
31/01/2014	152	TDC 0075100124 L.000152 20140130 000152	332,42	488.851,37
31/01/2014	161	TDC 0075100128 L.000161 20140130 000161	234,09	489.085,46
31/01/2014	177	TDC 0075100145 L.000177 20140130 000177	8.652,46	497.737,92
31/01/2014	221	TDC 0075100146 L.000221 20140130 000221	504,89	498.242,81
31/01/2014	379	TDB CAPIT. 0075100101 L.000379 30687	99,75	498.342,56
31/01/2014	319	TDB CAPIT. 0075100104 L.000319 30687	5.077,27	503.419,83
31/01/2014	410	TDB CAPIT. 0075100105 L.000410 30687	1.348,61	504.768,44
31/01/2014	260	TDB CAPIT. 0075100106 L.000260 30687	325,18	505.093,62

Figura 47. Archivo Excel Banesco. (Banco Banesco)

- Ejemplo de archivo Excel para el banco Banco del Tesoro

Fecha	Descripción	Referencia	Cargo	Abono
02/01/2014	LQ TDC 77924299 334 0064	200515630	0	764,67
02/01/2014	LQ TDC 77924299 289 0067	200515631	0	280,61
02/01/2014	LQ TDC 77924299 315 0142	200515632	0	935,36
02/01/2014	LQ TDC 77924299 208 0158	200515633	0	662,23
02/01/2014	LQ ELE 77924299 277 0010	300093973	0	221,89
02/01/2014	LQ ELE 77924299 313 0013	300093974	0	100,5
02/01/2014	LQ ELE 77924299 333 0039	300093975	0	746,25
02/01/2014	LQ TDD 77924299 319 0003	101636433	0	1.243,75
02/01/2014	LQ TDD 77924299 308 0024	101636434	0	1.791,00
02/01/2014	LQ TDD 77924299 327 0027	101636435	0	666,65
02/01/2014	LQ TDD 77924299 261 0044	101636436	0	995

Figura 48. Archivo Excel Banco del Tesoro. (Banco del Tesoro)

- Correo Tarea TaskIntegridad

Notificacion automática del Scheduler de Pago Aqui Operacion

Se produjo un Evento con las siguiente información asociada:

Resumen de Transacciones
- 2037 Solicitudes que poseen data correctamente en merchant.

- 4 Solicitudes que no poseen data en merchant.

- 5 Registros de merchant que no poseen data en solicitud.

Existen 5 Registros de merchant que no poseen data en solicitud.

Merchant	Vterminal	Terminal
Pago Aqui 100111	PAQUIX1001	75100111
Pago Aqui 100148	PAQUIP100148-0000007	75100148
Pago Aqui 100149	PAQUIP100149-0000008	75100149
Pago Aqui 100150	PAQUIP100150-0000009	75100150
Pago Aqui 100177	PAQUI100177	75100177

Existen 4 Solicitudes que no poseen data en merchant.

Solicitud	Merchant	Vterminal	Terminal	Estatus
1	Paqui - Jenny Mendoza (D)	PAQUIX100115	75100115	Cerrada
6	Paqui - Miguel Valdepenas (D)	PAQUIX1006	75100116	Cerrada
8	Paqui - Max Agroalimentaria	PAQUI108	67201138	Cerrada
13	Paqui - Max Agroalimentaria	PAQUI108	A75100108	Cerrada

Por favor, no responda este correo. Esto es una Notificación Automática.

Notificador Automáticos de Eventos.

Figura 49. Correo de la Tarea TaskIntegridad. (Elaboración Propia)

- Reporte de Conciliacion

Reporte Conciliacion

Reporte Conciliacion:

- Banco: BANESCO

Afiliacion	Conciliadas	Monto Conciliadas	Parcial Conciliadas	Monto Banco	Monto Merchant	NO Conciliadas	Monto NO Conciliadas	Monto Disponible
1173367	16	30059 BsF.	0	0 BsF.	0 BsF.	0	0 BsF.	30059 BsF.
1175050	2	507.5 BsF.	5	22756.8 BsF.	23196 BsF.	0	0 BsF.	23293.3 BsF.
1176219	0	0 BsF.	0	0 BsF.	0 BsF.	0	0 BsF.	0 BsF.

- Banco: BANPLUS

Afiliacion	Conciliadas	Monto Conciliadas	Parcial Conciliadas	Monto Banco	Monto Merchant	NO Conciliadas	Monto NO Conciliadas	Monto Disponible
77104968	0	0 BsF.	0	0 BsF.	0 BsF.	0	0 BsF.	0 BsF.

- Banco: BANCO DEL TESORO

Afiliacion	Conciliadas	Monto Conciliadas	Parcial Conciliadas	Monto Banco	Monto Merchant	NO Conciliadas	Monto NO Conciliadas	Monto Disponible
77924299	62	64038.64 BsF.	5	2996.57 BsF.	8809 BsF.	64	63450 BsF.	67035.21 BsF.
79363502	0	0 BsF.	0	0 BsF.	0 BsF.	0	0 BsF.	0 BsF.

Cerrar

Figura 50. Reporte Conciliación. (Elaboración Propia)

- Reporte de Conciliacion (Conciliadas)

Reporte Detalle Conciliacion

Reporte Detalle Conciliacion (Conciliadas):

Fecha	Tipo Transaccion	Afiliacion	Terminal	Lote	Banco	Status	Monto Banco	Monto Merchant
24 / 05 / 2013	LIQUIDACION POS TDD	1175050	75100142	100	BANESCO	Conciliado	255.5	255.5
24 / 05 / 2013	LIQUIDACION POS TDC	1175050	75100143	160	BANESCO	Conciliado	252	252

Total

Figura 51. Reporte Conciliación Conciliadas. (Elaboración Propia)

- Reporte de Conciliación (Parcialmente Conciliadas)

Reporte Detalle Conciliacion

Reporte Detalle Conciliacion (Parcialmente Conciliadas):

Fecha	Tipo Transaccion	Afiliacion	Terminal	Lote	Banco	Status	Monto Banco	Monto Merchant
24 / 05 / 2013	LIQUIDACION POS TDD	1175050	75100141	92	BANESCO	Parcialmente Conciliado	1182.7	1189
24 / 05 / 2013	LIQUIDACION POS ELE	1175050	75100141	92	BANESCO	Parcialmente Conciliado	6507.2	6640
24 / 05 / 2013	LIQUIDACION POS TDC	1175050	75100146	100	BANESCO	Parcialmente Conciliado	3421.99	3660
24 / 05 / 2013	LIQUIDACION POS TDD	1175050	75100146	100	BANESCO	Parcialmente Conciliado	9056.71	9105
24 / 05 / 2013	LIQUIDACION POS TDD	1175050	75100143	160	BANESCO	Parcialmente Conciliado	2588.2	2602

Cerrar

Figura 52. Reporte Conciliar Parcialmente Conciliadas. (Elaboración Propia)

- Reporte de Conciliación (No Conciliadas)

Reporte Detalle Conciliacion

Reporte Detalle Conciliacion (No Conciliadas):

Fecha	Tipo Transaccion	Afiliacion	Terminal	Lote	Banco	Status	Monto Banco	Monto Merchant
24 / 05 / 2013	LIQUIDACION POS TDC	77924299	305	1	BANCO DEL TESORO	No Conciliado Banco	0	93.37
24 / 05 / 2013	LIQUIDACION POS TDC	77924299	289	1	BANCO DEL TESORO	No Conciliado Banco	0	1307.18
24 / 05 / 2013	LIQUIDACION POS ELE	77924299	355	1	BANCO DEL TESORO	No Conciliado Banco	0	99.5
24 / 05 / 2013	LIQUIDACION POS TDC	77924299	251	2	BANCO DEL TESORO	No Conciliado Banco	0	0.47
24 / 05 / 2013	LIQUIDACION POS TDD	77924299	337	2	BANCO DEL TESORO	No Conciliado Banco	0	9.95
24 / 05 / 2013	LIQUIDACION POS TDD	77924299	366	3	BANCO DEL TESORO	No Conciliado Banco	0	119.4

Cerrar

Figura 53. Reporte conciliar No Conciliadas. (Elaboración Propia)

- Reporte de Proceso de Cuadre I

Proceso Cuadre 1

Proceso De Cuadre:

Lote No 1

Fecha	Tarjeta	Transaccion	Adquiriente	Lote	Autorización	Estado	Monto	
10 / 05 / 2013 - 17:31:28	501878********4637	Void	BancoTesoro	1	009306	Aprobada	50	
10 / 05 / 2013 - 17:30:31	501878********4637	Debit Sale EMV	BancoTesoro	1	009306	Cancelada	50	
10 / 05 / 2013 - 17:28:31	422045******3019	Void	BancoTesoro	1	229776	Aprobada	100	
10 / 05 / 2013 - 17:26:37	422045******3019	Credit Sale EMV	BancoTesoro	1	229776	Cancelada	100	
10 / 05 / 2013 - 17:26:04	422045******3019	Credit Sale EMV	BancoTesoro	1	-	Cancelada	100	Monto TOTAL: 150.00

Suma Monto Total: 150,00

Cerrar

Figura 54. Reporte Proceso de Cuadre I. (Elaboración Propia)

- Reporte Proceso de Cuadre II Parte 1

Proceso Cuadre 1

Proceso De Cuadre:

- Lote No 18

Fecha	Tarjeta	Transaccion	Adquiriente	Lote	Autorizacion	Estado	Monto	
14 / 07 / 2013 - 12:23:26	621984******1013	Debit Sale	BancoTesoro	18	230661	Aceptada	407.00	
14 / 07 / 2013 - 12:20:49	589524********2852	Debit Sale	BancoTesoro	18	995583	Aceptada	105.00	
14 / 07 / 2013 - 12:18:09	589524********2852	Debit Sale	BancoTesoro	18	993278	Aceptada	160.00	
14 / 07 / 2013 - 12:09:46	589941******8202	Debit Sale	BancoTesoro	18	991972	Aceptada	325.00	
14 / 07 / 2013 - 11:34:54	601288******3002	Debit Sale	BancoTesoro	18	153207	Aceptada	655.00	
14 / 07 / 2013 - 10:16:04	601288******2183	Debit Sale	BancoTesoro	18	34317	Aceptada	124.00	
14 / 07 / 2013 - 10:08:31	501878*******2725	Debit Sale	BancoTesoro	18	46073	Aceptada	434.00	
14 / 07 / 2013 - 09:57:03	589524********5668	Debit Sale	BancoTesoro	18	873657	Aceptada	298.00	
14 / 07 / 2013 - 09:40:19	589524********5668	Debit Sale	BancoTesoro	18	862008	Aceptada	240.00	
13 / 07 / 2013 - 13:09:01	639489********3087	Debit Sale	BancoTesoro	18	124633	Aceptada	70.00	
13 / 07 / 2013 - 11:29:17	501878*******7485	Debit Sale	BancoTesoro	18	89809	Aceptada	112.00	
13 / 07 / 2013 - 09:05:51	501878*******4570	Debit Sale	BancoTesoro	18	57798	Aceptada	110.00	
13 / 07 / 2013 - 08:52:23	601288******3432	Debit Sale	BancoTesoro	18	870537	Aceptada	222.22	Monto TOTAL: 3262.22

- Lote No 17

Cerrar

Figura 55. Reporte Proceso de Cuadre II Parte 1. (Elaboración Propia)

- Reporte Proceso de Cuadre II Parte 2

Proceso Cuadre 1

13 / 07 / 2013 - 09:05:51	501878*******4570	Debit Sale	BancoTesoro	18	57798	Aceptada	110.00	
13 / 07 / 2013 - 08:52:23	601288******3432	Debit Sale	BancoTesoro	18	870537	Aceptada	222.22	Monto TOTAL: 3262.22

Lote No 17

Fecha	Tarjeta	Transaccion	Adquiriente	Lote	Autorizacion	Estado	Monto	
12 / 07 / 2013 - 11:46:15	601288******3247	Debit Sale	BancoTesoro	17	164460	Aceptada	309.00	
12 / 07 / 2013 - 09:37:23	601288******5907	Debit Sale	BancoTesoro	17	955548	Aceptada	118.00	
12 / 07 / 2013 - 09:29:21	639489********7599	Debit Sale	BancoTesoro	17	330718	Aceptada	280.00	
12 / 07 / 2013 - 09:23:20	639489********7599	Debit Sale	BancoTesoro	17	270562	Aceptada	150.00	
12 / 07 / 2013 - 08:48:59	501878*******5236	Debit Sale	BancoTesoro	17	46170	Aceptada	110.00	
12 / 07 / 2013 - 08:37:32	601288******8975	Debit Sale	BancoTesoro	17	855647	Aceptada	192.00	
12 / 07 / 2013 - 08:19:51	601288******8975	Debit Sale	BancoTesoro	17	838196	Aceptada	137.00	
12 / 07 / 2013 - 08:10:31	501878*******5977	Debit Sale	BancoTesoro	17	31154	Aceptada	100.00	
12 / 07 / 2013 - 07:54:13	501878*******1801	Debit Sale	BancoTesoro	17	93021	Aceptada	115.00	
12 / 07 / 2013 - 06:50:36	501878*******1697	Debit Sale	BancoTesoro	17	65108	Aceptada	400.00	Monto TOTAL: 1911.00

Suma Monto Total: 5173,22

Cerrar

Figura 56. Reporte Proceso de Cuadre II Parte 2. (Elaboración Propia)

Apéndice 5

(Java Vs PHP)

A5.1 Lenguaje Java

En un primer nivel, Java es un lenguaje de programación de propósito general, orientado a objetos, que fue introducido por Sun Microsystems en 1995, y diseñado en principio para el ambiente distribuido de Internet. Pero lo que hace de Java un concepto diferente es que, en un segundo nivel, es también un entorno para la ejecución de programas, englobado en la llamada máquina virtual de Java. Este entorno es un software que permite que las aplicaciones escritas en Java se ejecuten en cualquier ordenador, independientemente del sistema operativo y de la configuración de hardware utilizados.

A5.1.1 Descripción y cualidades del lenguaje Java

A5.1.1.1 La máquina virtual de Java

A5.1.1.1.1 Lenguajes compilados e interpretados

Un lenguaje de programación es un conjunto de instrucciones, y una serie de esas instrucciones se dice que forma el código fuente de un programa. Para que un ordenador entienda un programa hay que traducir el código fuente a código máquina; según el lenguaje de programación empleado, esto se puede hacer de dos maneras:

El código fuente se traduce a código máquina completamente antes de la ejecución del programa, mediante un proceso llamado compilación, realizado por un programa llamado compilador. Se dice que el lenguaje de programación es un lenguaje compilado.

La traducción se realiza en el mismo momento de la ejecución, mediante un proceso llamado interpretación, en el que un programa llamado intérprete lee las líneas del código fuente y las va traduciendo y ejecutando una a una. El lenguaje se dice entonces que es interpretado.

Lógicamente, un programa compilado se ejecuta más deprisa que uno interpretado, pues el compilador ha producido previamente todo el código máquina necesario, y normalmente ha llevado a cabo un proceso de optimización. Por otra parte, es mucho más fácil escribir un intérprete que un compilador, lo que permite disponer de intérpretes en una amplia variedad de entornos, incluidas máquinas de reducidas prestaciones. Esto hace que los lenguajes interpretados estén presentes en más plataformas.

En el caso de Java, se trataba de conseguir tanto la eficiencia de los lenguajes compilados como la universalidad de los lenguajes interpretados. Desde el punto de vista de las prestaciones, los requisitos exigidos del lenguaje hacían pensar en la utilización de un compilador: pero esto implicaba la pérdida de universalidad de las aplicaciones, pues la generación de código máquina para un ordenador particular hace necesario adaptar y recompilar las aplicaciones para cada plataforma existente, y para todas las futuras. De modo que se integraron las dos filosofías:

A5.1.1.1.2 Compilación e interpretación.

Se decidió que los programas escritos en Java debían sufrir una compilación como paso previo a su ejecución. Sin embargo, el resultado de la compilación no es un conjunto de instrucciones comprensibles por un cierto microprocesador, sino una representación binaria genérica, llamada *byte codes*. Se diseñó una máquina ficticia, con un juego de instrucciones asociado, que entendiera los byte codes, como lo haría un microprocesador real, y se la llamó la máquina virtual de Java (abreviadamente: VM, por *virtual machine*), pues no existe ningún microprocesador que utilice el juego de instrucciones mencionado, sino que se trata de una pieza de software.

La función del compilador de Java es por tanto producir byte codes, o código para la VM. Así se consigue: generar un código binario compacto y eficiente; aislarlo de cualquier máquina particular; y definir perfectamente el comportamiento del lenguaje. En efecto, la VM está diseñada para ajustarse exactamente a la sintaxis y a la semántica de Java. Algunas de las características de la VM están relacionadas directamente con la estructura de datos en Java, lo que permite fijar ciertos parámetros para la representación de los mismos. (Esos parámetros, sin embargo, varían entre los distintos microprocesadores reales, lo que hace que con lenguajes de programación tradicionales haya que cambiar el código fuente cada vez que se quiere ejecutar un programa en diferentes máquinas.)

Para ejecutar Java en ordenadores reales, la VM se implementa en cada plataforma como software, para que funcione en tanto que intérprete de los byte codes: en vez de

ir leyendo cada línea del código fuente, como haría un intérprete tradicional, la VM va examinando cada uno de los byte codes compilados previamente, y los ejecuta en el ordenador donde reside el intérprete. Así, el código Java compilado (los byte codes) puede ejecutarse en cualquier plataforma que disponga de un intérprete de Java, pues las características de la máquina virtual son siempre las mismas.

A5.1.1.1.3 Compilador just-in-time en la VM de Java

La interpretación de los byte codes en un ordenador no puede resultar tan rápida como la ejecución de un programa escrito en un lenguaje compilado para ese ordenador. Para aliviar en buena medida la mayor lentitud de Java con respecto a los lenguajes compilados, se han ideado dos soluciones. La primera es la adopción de la técnica de compilación *just-in-time* por la VM, cuando ésta va interpretando los byte codes, va almacenando el código máquina resultante, de modo que cuando el flujo de ejecución del programa obliga a volver sobre una parte que ya ha sido procesada, no es necesario volver a interpretar los byte codes, sino que basta con reutilizar la secuencia de código memorizada. Esto da resultados en trozos de código que utilicen a fondo la capacidad computacional del microprocesador, como es el caso de los cálculos de todo tipo. La segunda solución es la fabricación de chips específicos para Java, esto es, microprocesadores cuyas características y juegos de instrucciones coinciden con los de la VM de Java.

A5.1.1.2 Cualidades principales de Java

A5.1.1.2.1 Universalidad

Aun cuando un programa interpretado no es en principio tan rápido como un programa equivalente compilado, los beneficios de Java son sin embargo mucho mejores que las de cualquier lenguaje interpretado. Este hecho ha propiciado que se hayan escrito intérpretes de pequeño tamaño adaptados a prácticamente cualquier plataforma, desde mainframes y ordenadores personales (con cualquier sistema operativo: Windows, Macintosh OS, Unix,...) hasta dispositivos electrónicos de bajo coste. Además, la universalidad de los byte codes hacen de Java el lenguaje idóneo para desarrollar aplicaciones para Internet. De hecho, la mayor parte de los navegadores (Netscape Navigator, Internet Explorer, HotJava) integran máquinas virtuales, y por tanto intérpretes de Java, lo que hace posible acceder automáticamente a los applets presentes en las páginas web. De nuevo la sencillez de Java hace que esta integración no reduzca en absoluto las prestaciones de los navegadores, permitiendo además la ejecución rápida y simultánea de gran cantidad de applets.

También se suele hacer referencia a la universalidad de Java con términos equivalentes como transportabilidad, o independencia de plataforma, pues para ejecutar un programa basta compilarlo una sola vez: a partir de entonces, se puede hacer correr en cualquier máquina que tenga implementado un intérprete de Java.

Además, las bibliotecas estándar de funciones y métodos de Java (definidas en su API, *Application Programming Interface*) facilitan la programación de multitud

de acciones complejas (desarrollo de interfaces gráficas, multimedia, multitarea, interacción con bases de datos, entre otros). Ningún otro lenguaje (ni compilado ni interpretado) dispone como Java de una cantidad tan grande de funciones accesibles en cualquier plataforma sin necesidad de cambiar el código fuente.

A5.1.1.2.2 Sencillez

Java es un lenguaje de gran facilidad de aprendizaje, pues en su concepción se eliminaron todos aquellos elementos que no se consideraron absolutamente necesarios. Por ejemplo, en comparación con otros lenguajes como C ó C++, es notable la ausencia de punteros, o lo que es lo mismo: es imposible hacer referencia de forma explícita a una posición de memoria; esto permite ahorrar gran cantidad de tiempo a los programadores, dado que el comportamiento imprevisto de los punteros es una de las principales fuentes de errores en la ejecución de un programa. Por otra parte, el código escrito en Java es por lo general mucho más legible que el escrito en C ó C++.

Por otro lado, Java dispone de un mecanismo conocido como "recogida de basura", el cual (usando la capacidad multitarea de Java) permite que, durante la ejecución de un programa, los objetos que ya no se utilizan se eliminen automáticamente de la memoria. Dicho mecanismo facilita enormemente el diseño de un programa y optimiza los recursos de la máquina que se esté usando para la ejecución del mismo (con los lenguajes tradicionales, la eliminación de objetos y la consiguiente optimización de recursos debe planificarse cuidadosamente al idear el programa).

A5.1.1.2.3 Orientación a objetos

Aunque C++ es también, como Java, un lenguaje orientado a objetos, la diferencia fundamental es que Java lo es desde su concepción, mientras que C++ se diseñó como un lenguaje compatible con C (que no es orientado a objetos): de este modo, un programa escrito en C++ puede ignorar la mayoría de las características de orientación a objetos de C++ y compilarse sin problemas en un compilador de C++. Sin embargo, un programador no puede obviar la orientación a objetos cuando escribe un programa en Java, y esto hace que las aplicaciones escritas en Java tengan interesantes ventajas, que veremos más adelante.

Para entender qué es la orientación a objetos, recordemos que los lenguajes tradicionales no orientados a objetos, como Pascal ó C, estaban pensados para trabajar de forma secuencial y basaban su funcionamiento en el concepto de procedimiento o función. La tarea principal del programa se dividía en funciones o tareas más pequeñas, a menudo muy interrelacionadas, por lo que era difícil modificar una función sin tener que revisar el resto del programa: por tanto, también era difícil reutilizar o actualizar los programas ya escritos.

En el caso de los lenguajes orientados a objetos, el concepto clave no es el de función, sino el de objeto. Un objeto es un elemento de programación, auto contenido y reutilizable, y que podríamos definir como la representación en un programa de un concepto, representación que está formada por un conjunto de variables (los datos) y un conjunto de métodos (o instrucciones para manejar los datos).

La "encapsulación" de variables y métodos en un objeto tiene claras ventajas:

- Cada objeto puede ser modificado y mantenido por separado.
- Se pueden mantener en un objeto métodos y variables que no son accesibles desde fuera de él, lo que evita multitud de posibilidades de error en el momento de confeccionar un programa. Esta característica se llama "ocultamiento de la información".
- Es posible reutilizar porciones de programa ya escritas: en el ejemplo anterior, un programador puede reutilizar el objeto "código de barras" para escribir software para diversos tipos de transacciones, como préstamo, devolución, reserva, etc. sin necesidad de reescribir cada vez los comandos para calcular el dígito de control, dado que el programa se encargará de hacerlo cada vez que se use el objeto.
- El estándar JavaBeans de Java proporciona un entorno visual de programación que facilita y simplifica la programación de objetos. De hecho, es posible escribir applets sencillos en Java sin necesidad de escribir una sola línea de código.

A5.1.1.2.4 Seguridad

En general, se considera que un lenguaje es más seguro cuanto menor es la posibilidad de que errores en la programación, o diseños malintencionados de programas (virus), causen daños en el sistema. La extrema seguridad de Java se establece a tres niveles:

- Nivel de seguridad dado por las características del lenguaje, tales como la ausencia de punteros (que evita cualquier error de asignación de memoria) o el

"ocultamiento de la información" propio de la programación orientada a objetos, por recordar dos ejemplos ya mencionados.

- Nivel de seguridad dado por el diseño de la VM. La VM de Java posee un verificador de los byte codes, que antes de ejecutarlos se analiza su formato comprobando que no existen punteros en ellos, que se accede a los recursos del sistema a través de objetos de Java, entre otros. Otro elemento constitutivo de la VM es el cargador de clases. Una clase es una categoría de objetos utilizados en un programa; cuando se ejecuta un programa en Java, éste llama a determinadas clases a través del cargador de clases. Estas clases pueden provenir de tres lugares distintos, en donde residen en forma de ficheros: del ordenador local, de la red de área local a la que pueda estar conectado el ordenador cliente, o de Internet. En función de la procedencia de las clases, se efectúan una serie de comprobaciones diferentes y el gestor de seguridad de la VM prohíbe los accesos peligrosos.
- Nivel de seguridad dado por la API de Java. El conjunto de métodos y clases que estamos obligados a utilizar cuando programamos en Java para acceder a los recursos del sistema, está definido por la API, y constituye la última barrera defensiva. El diseño de dichos métodos y clases hace que éstos realicen múltiples verificaciones cuando son invocados, de modo que se dificultan los errores (voluntarios o involuntarios).

A5.1.1.2.5 Adaptación a redes (y en particular a Internet)

Ya hemos dicho que Java irrumpió en el mercado para potenciar la interactividad en Internet, y también de los applets, pequeños programas en Java que se cargan junto con una página web desde un servidor, y que son ejecutados (por la VM del navegador del cliente) como una parte de la página web. Además de las ventajas que supone su ejecución local, los applets disponen de una significativa riqueza de recursos y son capaces de realizar tareas muy complejas a pesar de su reducido tamaño. Una de las explicaciones de esta sorprendente capacidad es el hecho de que los applets se sirven del propio código del navegador en cuya VM se ejecutan, utilizándolo para tareas tales como presentación gráfica o comunicaciones. Sin embargo, el acceso a las funciones del navegador es totalmente automático y transparente para el programador, que debe limitarse a invocar ciertas funciones de la API de Java; estas invocaciones, interpretadas por el navegador, dan origen a acciones muy complejas. Esta observación es muy importante cuando se discute del rendimiento de Java, pues todas estas acciones se realizan en la máquina que está ejecutando el applet, y la rapidez de ejecución de las mismas no depende de que Java sea un lenguaje semiinterpretado (o semicompilado).

Entre las tareas básicas más comunes que suelen realizar los applets, se encuentran: visualizar animaciones en la ventana del navegador; reproducir sonidos; establecer comunicaciones con el servidor del que procede el applet (como por ejemplo, para cargar desde él ficheros de cualquier naturaleza); crear interfaces gráficas con los elementos habituales de los entornos de ventanas (como Macintosh o Windows): menús desplegables, botones, áreas de texto, barras de desplazamiento, etc.; pedir

datos al usuario para procesarlos (gestionando eventos como pulsaciones de teclas o acciones con el ratón).

A5.2 Lenguaje PHP

PHP (siglas que originalmente significaban Personal Home Page) fue primero escrito por Rasmus Lerdorf como un simple conjunto de scripts de Perl para guiar a los usuarios en sus páginas. Luego para satisfacer inquietudes del mismo tipo por parte de otras entidades, lo reescribe, pero esta vez como un lenguaje de script agregándole entre otras características soporte para formularios. Al ver como la popularidad del lenguaje aumenta, un grupo de desarrolladores crea para él un API, convirtiéndose así en el PHP3. Fue en ese momento cuando el parser de scripts PHP es completamente reescrito (el Zend Engine) dando vida al PHP4 mucho más rápido, tal y como lo conocemos en la actualidad. PHP actualmente significa Hypertext Preprocessor.

A5.2.1 Descripción y cualidades del lenguaje PHP

A5.2.1.1 Velocidad

No solo la velocidad de ejecución, la cual es importante, sino además no crear demoras en la máquina. Por esta razón no debe requerir demasiados recursos de sistema. PHP se integra muy bien junto a otro software, especialmente bajo ambientes Unix, cuando se configura como módulo de Apache, está listo para ser utilizado.

A5.2.1.2 Estabilidad

La velocidad no sirve de mucho si el sistema se cae cada cierta cantidad de ejecuciones. Ninguna aplicación es 100% libre de bugs, pero teniendo de respaldo una increíble comunidad de programadores y usuarios es mucho más difícil para lo bugs sobrevivir. PHP utiliza su propio sistema de administración de recursos y dispone de un sofisticado método de manejo de variables, conformando un sistema robusto y estable.

A5.2.1.3 Seguridad

Un sistema debe poseer protecciones contra ataques. PHP provee diferentes niveles de seguridad, estos pueden ser configurados desde el archivo .ini.

A5.2.1.4 Simplicidad

Se les debe permitir a los programadores generar código productivamente en el menor tiempo posible. Usuarios con experiencia en C y C++ podrán utilizar PHP rápidamente.

A5.2.2 Ventajas adicionales de PHP

- PHP corre en la mayoría de las plataformas utilizando el mismo código fuente, pudiendo ser compilado y ejecutado en algo así como 25 plataformas, incluyendo diferentes versiones de Unix, Windows y Macs. Como en todos los sistemas se utiliza el mismo código base, los scripts pueden ser ejecutados de manera independiente al OS.

- La sintaxis de PHP es similar a la del C, por esto cualquiera con experiencia en lenguajes del estilo C podrá entender rápidamente PHP. Entre los lenguajes del tipo C incluimos al Java y Javascript, de hecho mucha de la funcionalidad del PHP provienen de C en funciones como fread() o srtlen()..
- Muchas interfaces distintas para cada tipo de servidor. PHP actualmente se puede ejecutar bajo Apache, IIS, AOLServer, Roxen yTHTTPD. Otra alternativa es configurarlo como módulo CGI.
- Puede interactuar con muchos motores de bases de datos tales como MySQL, MS SQL, Oracle, Informix, PostgreSQL, y otros muchos. Siempre podrás disponer de ODBC para situaciones que lo requieran.
- Una gran variedad de módulos cuando un programador PHP necesita una interfase para una librería en particular, fácilmente podrá crear una API para esta. Algunas de las que ya vienen implementadas permiten manejo de gráficos, archivos PDF, Flash, Cybercash, calendarios, XML, IMAP, POP, etc.
- Rapidez. PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.
- PHP es Open Source, lo cual significa que el usuario no depende de una compañía específica para arreglar cosas que no funcionan, además no se está obligado a pagar actualizaciones anuales para tener una versión que funcione.

A5.2.2 Desventajas de PHP

- El manejo de errores no es tan sofisticado como Cold Fusion o ASP.
- No existe IDE o Debugger. Una IDE puede no ser importante para la mayoría de los programadores y un debugger ha sido prometido por Zend Tech para un futuro muy cercano.

A5.3 Cuadro Comparativo Lenguaje Java vs. PHP

Tomando en cuenta todo el análisis y características explicadas anteriormente para ambos lenguajes, se realiza el siguiente cuadro comparativo:

Características	Java	PHP
Costo	Javano es Open Source, dando como resultado que el costo de licencia de Java sea mayor a PHP.	PHP es Open Source, por lo tanto la licencia de dicho lenguaje es gratuita.
Programación OO	Siempre ha tenido un paradigma Orientado a Objetos.	Php 5 adapta su paradigma a OO igual que Java. Versiones anteriores Php 4.x son orientadas a la programación de scripts.

Declaración de variables	Las variables se deben declarar con un tipo de datos específico antes del uso. Siendo conveniente para el desarrollo de aplicaciones grandes.	Se declara la variable cuando se crea. Su tipo se implica del valor asignado. Una variable puede cambiar su tipo si se asigna un nuevo valor. Las variables que cambian sus tipos basados en su valor son peligrosas en aplicaciones grandes.
Debugging	Todos los editores de Java permiten realizar debug en los proyectos. Esta es una característica importante a la hora de realizar trazas y creación de logs de forma eficaz.	La opción de debug que posee Php son normalmente declaraciones "print". Dando como resultado un proceso más complicado a la hora de crear trazas y logs.
Manejo de Excepciones	Posee una dirección estructurada con la construcción de try / catch / finally para un manejo de errores más amplio y especifico.	Se puede realizar captura de errores pero no tan efectiva como Java, ya que se pueden ignorar algunos errores.

Seguridad	Tiene un nivel de seguridad bastante alto, como se explicó anteriormente con respecto a la ausencia de punteros, API de java, entre otros.	Tiene soporte de ayuda, pero muy limitada comparada con el lenguaje Java.

Figura 57. Tabla Comparativa Java Vs PHP. (Elaboración Propia)