CAPÍTULO 3 - DESARROLLO DEL PROYECTO

3.1 Digitalización automatizada de cheques en sucursales

Como se mencionó antes, el proceso mediante el cual se digitalizan las imágenes de los *cheques*, ya no se realizará de forma central, sino en las sucursales. El nuevo proceso consistió en lo siguiente:

Se optó por implementar una nueva funcionalidad en una de las aplicaciones que se utilizan actualmente (2010) en los equipos de las *cajas* de las sucursales. A esta aplicación se le denominará en lo sucesivo "*Sistema en ventanilla*", pues es una aplicación que se utiliza actualmente en las ventanillas de atención a clientes en las sucursales *del banco*. La nueva funcionalidad fue desarrollada en el lenguaje de programación "*Java*". Actualmente está concluido.

3.1.1 Sistema en ventanilla

El *Sistema en ventanilla* es una aplicación que se creó por otra empresa (distinta de *TCS*) de consultoría en tecnologías de información, para ofrecer servicios a los clientes *del banco* en las sucursales del mismo. Esta aplicación ya estaba siendo utilizada en las sucursales *del banco* cuando inició el proyecto "*Digitalización automatizada de cheques en sucursales bancarias*", pero no contaba con las funciones para digitalizar *cheques*. Al inicio del proyecto, los componentes básicos del sistema ya existían, salvo aquellos que se agregaron específicamente para la funcionalidad de digitalización de *cheques*. El desarrollo de la nueva funcionalidad se hizo sobre los componentes existentes y con la adición de otros nuevos. En términos generales, se le agregó a la aplicación las funciones para interactuar con el escáner LS100, almacenar imágenes de los *cheques* digitalizados en una base de datos y consultar qué *cheques* operados habían sido digitalizados y cuáles no.

La aplicación funciona con una arquitectura cliente-servidor. El componente de la aplicación del lado del servidor se publica en un *Servidor de aplicaciones*; en este caso se utilizó la versión 8.1 de *WebLogic*, creado por la empresa *BEA Systems*. El componente *cliente* de la aplicación se instala en cada equipo PC de las *cajas* de las sucursales *del banco*.

Una vez instalados y configurados los componentes de la aplicación, el componente *cliente* realiza peticiones al servidor, mismo que atiende las peticiones de todos los clientes mediante un *EJB*.

La siguiente figura muestra una perspectiva general del sistema:

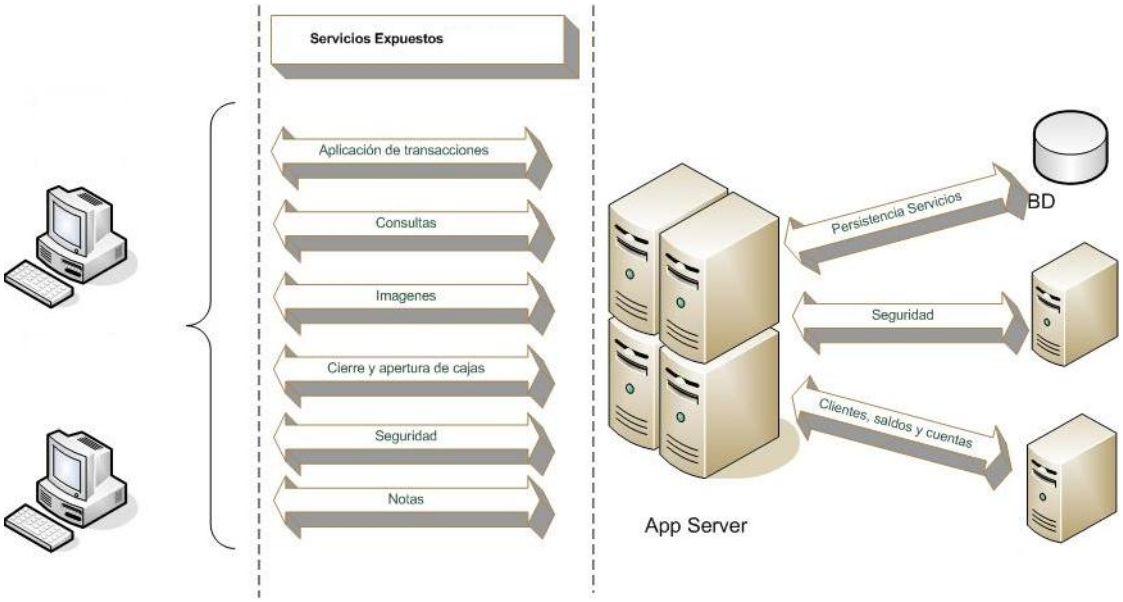

Fig. 3.1.1a: Diagrama de arquitectura.

El *Sistema en ventanilla*, como se mencionó arriba, fue creado por otra empresa. Se hizo utilizando el lenguaje de programación *Java*, cabe mencionar que utilizó los siguientes *frameworks* y tecnologías:

- *Swing*: para la creación de los componentes gráficos.
- *EJBs*: para el manejo de transaccionalidad y seguridad, entre otros.
- *Spring*: para facilitar el trabajo con *EJBs*, *inyección de dependencia* y programación orientada a aspectos.
- *Hibernate* y Spring-*JDBC-Template*: para facilitar consultas e interacción con la base de datos.

En nuestro proyecto tuvimos que minimizar el uso de *Hibernate* ya que provocaba una carga adicional al sistema que no compensaba su beneficio. Por esta razón en la mayor parte de los casos optamos por realizar las consultas a la base de datos de forma directa utilizando *SQL* que obtuviera únicamente la información requerida.

Desde la creación del *Sistema en ventanilla* se contempló la funcionalidad de *digitalización automatizada de cheques* en sucursales. Sin embargo, nunca se implementó la funcionalidad por completo ni de forma correcta, sino hasta este proyecto.

El flujo general del servicio de digitalización de *cheques* implementado en el *Sistema en ventanilla* es el siguiente:

1.0) *El usuario* ingresa un *cheque* por digitalizar en la bandeja lectora del escáner LS100.

2.0) *El usuario* selecciona, en el *Sistema en ventanilla,* la opción para digitalizar el *cheque* en el escáner LS100.

3.0) *El usuario* valida los datos capturados y mostrados por el *Sistema en ventanilla*.

3.1) Si el sistema no pudo capturar los datos de la *banda magnética* del *cheque* de forma correcta, solicitará *al usuario* que los ingrese nuevamente y de forma manual (basta con que ingrese la secuencia completa de la *banda magnética* del cheque).

3.2) Si es necesario, *el usuario* realiza la captura manual de los datos de la *banda magnética* del *cheque* operado previamente en el *Sistema en ventanilla*.

3.3) La aplicación valida que la *banda magnética* proporcionada corresponda con la de algún *cheque* operado previamente en el *Sistema en ventanilla*.

3.4) Si el sistema no encuentra ninguna operación asociada a la *banda magnética*, indicará esto *al usuario* y solicitará el re-ingreso de la *banda magnética*.

3.5) Si el sistema encuentra la operación asociada a la *banda magnética* del *cheque* capturado, mostrará los datos de dicho *cheque* y se regresa al paso 3.0.

4.0) *El usuario* selecciona la opción para aceptar la digitalización del cheque.

5.0) Termina el proceso de digitalización del cheque.

Para integrar el servicio de digitalización e imponer su uso en el *Sistema en ventanilla*, se realizaron las siguientes modificaciones a la funcionalidad original de la aplicación:

1. Se agregó la función para digitalizar *cheques*.

2. Se agregó una funcionalidad para solicitar la digitalización de un *cheque* cada vez que se realizaba una operación en la que se utilizara un cheque.

3. Se agregó una opción para consultar los *cheques* que sí se han digitalizado y aquellos que están pendientes por digitalizar, de acuerdo con las operaciones realizadas.

4. Se agregó una funcionalidad que impide el cierre de *caja* si existen *cheques* pendientes por digitalizar.

3.1.2 Transmisión y almacenamiento de imágenes de cheques digitalizados

Cuando se digitaliza un cheque, esto se realiza en la parte *cliente* de la aplicación del *Sistema en ventanilla*, la cual transmitirá los archivos relacionados a la imagen del *cheque* utilizando un *EJB*.

Los archivos se envían como un arreglo de datos de tipo byte. El siguiente *código fuente* muestra una función que realiza la transformación del archivo a un arreglo de datos de este tipo:

```
public byte[ ] transformaArchivo(File file){
    byte[ ]  contenido = null;
    try {
         FileInputStream fis = new FileInputStream(file);

         int bufferSize =  file.length();
         contenido = new byte[bufferSize];

         long n = fis.read(contenido, 0, bufferSize);
         fis.close();
         fis = null;

         if( n != file.length() ){
            log.info ("Error leyendo archivo '" + file.getName()  + "'");
         }else{
            log.info ("Archivo transformado con exito, tamano : +
contenido.length);
         }
    } catch(Exception e){
        log.error("Ocurrio un error al transformar el archivo: ", e);
    }
    return contenido;
}
```

El componente *EJB* se encarga de ingresar los archivos de las imágenes a una base de datos *Oracle*. El registro contiene (entre otros datos) la fecha de inserción, los números de identificación del *cheque* y el contenido binario del archivo de la imagen. A continuación se muestra un fragmento del *código fuente* que realiza esta función en el *EJB*:

```
public void insertaImagenCheque(final ImagenCheque imagen) throws
SystemException {
    try {
        log.debug("Actualizando imagen en PREFIJO_CHEQUE_IMG: " +
imagen.getBandaMagnetica());
        String sql = "SELECT PREFIJO_CHEQUE_IMG_SEQ.nextval FROM dual";
        final int secuencia = ((Integer)jdbcTemplate.queryForObject(sql,
Integer.class)).intValue();

        StringBuffer query = new StringBuffer("INSERT INTO
PREFIJO_CHEQUE_IMG (SECUENCIA, BANDA_MAGNETICA, FECHA, AOR, IMAGEN)
VALUES(?,?,?,?,?,?,?)" );
        jdbcTemplate.update(query.toString(), new PreparedStatementSetter()
{
                                    public void setValues(PreparedStatement
prepStmt) throws SQLException {

                                        log.debug("imagen.secuencia: "+
secuencia);
                                        log.debug("imagen.getBandaMagnetica: "+
imagen.getBandaMagnetica());
                                        log.debug("imagen.fecha: "+ new
Date(imagen.getFecha().getTime()));
                                        prepStmt.setInt(1, secuencia);
                                        prepStmt.setLong(2,
Long.parseLong(quitarCaracteres(imagen.getBandaMagnetica())));
                                        prepStmt.setDate(5, new
Date(imagen.getFecha().getTime()));
                                        prepStmt.setString(6, imagen.getAor());
                                        prepStmt.setBinaryStream(7, new
ByteArrayInputStream(imagen.getImagenArreglo()),
imagen.getImagenArreglo().length);

                                        log.debug("prepStmt : " + prepStmt);
                                    }
                                });
    } catch (Exception exception) {
        log.error("Ocurrio un error al momento de insertar la imagen del
cheque con la banda: "+ imagen.getBandaCheque() + " " + exception);
        throw new SystemException("Ocurrio un error al momento de insertar
la imagen "+ exception);
    }
}
```

3.2 Trabajo simultáneo sobre el mismo código fuente de la aplicación

Para que el equipo de programadores o desarrolladores pudiera trabajar conjuntamente sobre el *código fuente* de la misma aplicación se utilizó una aplicación llamada *CVS* (por sus siglas en inglés: *Concurrent Versions System)*. *CVS* es una aplicación de tipo *software libre* que, entre otras de sus funciones, registra todos los cambios realizados en una serie de archivos por lo que permite a varios desarrolladores trabajar de forma colaborativa.

3.3 Aseguramiento de calidad y pruebas

Posterior a completar todos los cambios al *código fuente* del *Sistema en ventanilla*, se pasó a una fase en la que un área *del banco*, independiente del área de desarrollo, realizaría pruebas sobre la aplicación con el propósito de verificar y certificar que estuviera libre de defectos y que funcionara correctamente según lo especificado en los "*Casos de uso"*. El problema con esta área de *aseguramiento de calidad* (en lo sucesivo *QA*, del inglés "*Quality Assurance*"), es que pretende hacer pruebas para encontrar defectos en aplicaciones que no conoce a detalle. El personal que realiza dichas pruebas no está involucrada en el desarrollo del *código fuente* de la aplicación y, por lo tanto, no tiene conocimiento ni de la parte técnica ni del *código fuente* de la aplicación; la detección de errores o defectos de esta forma se vuelve una especie de juego de azar, en el que los defectos encontrados resultan más una coincidencia que una hipótesis acertada. Esto no sucedería si se buscaran errores después de realizar un análisis en el que se identificaran situaciones propicias o susceptibles de generar errores en la aplicación. En general, los defectos en el flujo normal de uso de la aplicación se detectan durante la fase de desarrollo, sólo los defectos provocados en casos especiales podrían pasar sin ser detectados y es por esto mismo que requieren atención especial.

Mi opinión es que la razón por la que se realizan de esta forma es porque para llevarlas a cabo de la forma en que yo propongo, el área de *QA* tendría que invertir un mayor presupuesto para contratar a gente más calificada, con más

conocimientos técnicos y sobre desarrollo de software, además de que el personal que realiza las pruebas tendría que estar más involucrado en el desarrollo del software y aunque sí podría haber un beneficio, las áreas de dirección probablemente considerarían que el costo y el esfuerzo no compensarían el beneficio obtenido.

Las pruebas realizadas son a nivel usuario. Esto significa que se busca encontrar defectos en el uso normal de la aplicación. Teóricamente también se hacen operaciones de formas no-convencionales o fuera de la norma, para verificar si el desarrollador consideró dichos casos.

En general, la realización de pruebas en *QA* resultó más un problema que un beneficio. Además del tiempo adicional requerido y del esfuerzo para generar la documentación solicitada, instalar y configurar las aplicaciones en el *ambiente* de *QA*, generó un problema significativo: la aplicación fallaba en el *ambiente* de *QA* debido a una configuración incorrecta realizada en la base de datos de dicho ambiente. Existen registros en la base de datos que relacionan componentes en la aplicación; si uno de los registros relaciona incorrectamente uno de los componentes, la interacción entre dichos componentes tendrá problemas. El problema en nuestro caso fue que cuando se detectó esto, no se sabía exactamente cuál de los varios registros podía ser la causa del conflicto, pero además, existe la política de que la gente de desarrollo no debe tener acceso a los ambientes de *QA*. Esto significa que se nos solicitó hacer la corrección de un problema que existía en una base de datos de *QA* pero sin que nosotros pudiésemos acceder a dicha base de datos. Después de varios días se nos permitió enviar consultas a la base de datos. Así fue que se encontró que existía un registro que generaba un conflicto en la aplicación. Cuando se eliminó el registro conflictivo, la aplicación funcionó correctamente. Las pruebas de *QA* no encontraron otra incidencia o anomalía. Cuando terminaron las pruebas con *QA* se prosiguió a realizar pruebas con *el usuario* de la aplicación. *El usuario* tuvo varias observaciones las cuales se describen a continuación.

3.4 Incidencias y desviaciones

En las pruebas *del usuario* se detectaron varias incidencias y errores en la funcionalidad del *Sistema en ventanilla*. Algunas son las siguientes:

3.4.1 Saturación de la red en el cierre de operación

En el diseño inicial de nuestra solución, los archivos de las imágenes de *cheques* se almacenaban en formato *JPEG* en el equipo PC de cada cajero. Al momento de hacer el *cierre de operación* de la sucursal, estos archivos se transferían digitalmente a un servidor central, específico para cada sucursal. Posteriormente se tomaban las imágenes de estos servidores y se agrupaban centralmente. El problema con este proceso fue que en ocasiones frecuentes se agotaba el ancho de banda y se saturaba la red (intranet) *del banco* por el envío simultáneo de archivos de imágenes, lo cual provocaba que el proceso de envío fuera lento o incluso que fallara la comunicación y se cancelara el proceso de envío.

Para corregir el problema de saturación de red, se optó por enviar los archivos de las imágenes de *cheques* en el momento en que cada *cheque* era digitalizado. Aunque el sistema no requiere que *el usuario* digitalice un *cheque* en el momento en que es operado, sí se recomienda hacerlo, debido a que si se intenta hacerlo después, existe el riesgo de que dado un percance (e. g.: un corte en el suministro de la energía eléctrica) se lo impida o provoque que no se haga dentro del horario establecido.

3.4.2 Almacenamiento de archivos de imágenes de cheques en el equipo del usuario

Se catalogó como un incidente de seguridad el hecho de que se almacenaran los archivos de las imágenes de *cheques* en el equipo *del usuario*, puesto que esto permitía que desde el momento en que se digitalizaban hasta el momento

en que se enviaban, las imágenes permanecieran disponibles en una ruta específica del sistema de archivos del equipo, de forma que *el usuario* podía copiarlas con facilidad y se corría el riesgo de que se hiciera uso indebido de la información contenida ellas. Se consideró que la incidencia no era seria, puesto que *el usuario* también era capaz de utilizar la función de captura de pantalla integrada al sistema operativo, dado que siempre se presenta la imagen del *cheque* que se digitaliza antes de enviarla para que *el usuario* pueda verificar que fue escaneada correctamente y como recordatorio de que la imagen del *cheque* debe contener un sello *del banco*, previamente puesto por el cajero, sobre el reverso del cheque. A pesar de no ser un problema serio, se solucionó para cumplir con la solicitud hecha por el área interna de seguridad.

Para evitar que los archivos de imágenes se almacenaran en el equipo *del usuario*, se modificó la funcionalidad de la aplicación de forma que el archivo con la información de la imagen se copiara al espacio de *memoria de acceso aleatorio* ("*RAM*") y se eliminara del sistema de archivos del disco duro del equipo. Esta operación no permite ninguna pausa ya que es cuestión de milésimas de segundo el tiempo en que se almacena y elimina el archivo del disco duro del equipo. Fue necesario hacerlo de esta forma debido a que el *API* del escáner LS100 permitía generar las imágenes de *cheques* en formato *JPEG* si y sólo si se almacenan archivos en el disco duro al momento de su generación, por lo que no era posible que se almacenara la información o los datos de la imagen, obtenida por el escáner LS100*,* en la *memoria de acceso aleatorio* del equipo, sin antes guardarse también en el disco duro del mismo equipo, aunque esto fuera sólo temporalmente. Se aceptó esta opción por su facilidad de implementación y porque tuvo el visto bueno del área de seguridad que solicitó la modificación en primer lugar.

3.4.3 Instalación y configuración del escáner LS100 en sucursales bancarias

Un problema que se tuvo fue que los dispositivos utilizados para escanear los *cheques* se adquirieron y enviaron a las sucursales tiempo antes de que se utilizaran con la aplicación para digitalizar *cheques*. Por esta razón, las

sucursales tenían los escáneres LS100, sin embargo, no los utilizaban. Esto provocó que los escáneres LS100 se desconectaran, almacenaran en otro sitio, se perdieran, deterioraran, etc. Por estas razones, cuando se quisieron realizar pruebas de la aplicación y digitalización de *cheques* en todas las sucursales, se encontró que en frecuentes ocasiones no se podía realizar una prueba completa debido a que el escáner LS100 no se encontraba funcionando correctamente en el equipo. Para corregir este problema se tuvo que solicitar apoyo a la gente del área de *infraestructura* pues ellos son los responsables de que los equipos de cómputo e informática estén funcionando correctamente. Su respuesta no fue inmediata, pero eventualmente arreglaron todos los problemas, reportados hasta el momento, de instalación y configuración del escáner LS100.

3.4.4 Presentación y accesibilidad

Los otros problemas que encontró *el usuario* se resolvieron cambiando el componente de la aplicación que se seleccionaba por defecto o los mensajes que daba la aplicación.

Lo que se consideró el origen de muchos de los demás problemas fue el hecho de que *el usuario* solicitó cambios a la aplicación que no estaban contemplados en el documento original de *requerimientos*. Otro problema relacionado fue que no se tenía un documento que definiera qué cambios podían realizarse a la aplicación sin causar un retraso en el plan de trabajo y qué cambios no (también conocido como "*documento de alcance*"). Por esta razón no fue fácil poner un alto a los cambios solicitados por *el usuario*. La solución a este problema fue mostrar al usuario que mientras se solicitaran cambios frecuentes, el avance en el desarrollo de la aplicación sería muy lento. Teniendo esto en cuenta, *el usuario* limitó las solicitudes a sólo cambios indispensables para el funcionamiento correcto de la aplicación. Se trabajó así hasta que *el usuario* no tuvo ninguna solicitud de cambio.

Posteriormente se pasó a probar la aplicación directamente en las sucursales, en un *ambiente de pruebas*, pero utilizando los mismos equipos con los que los cajeros operan diariamente.

Para poder probar la funcionalidad de digitalización en los equipos de las sucursales se tuvo que preparar un ambiente especial. Para ello se realizó lo siguiente: En primer lugar, se obtuvieron respaldos muy recientes los ambientes productivos, de las bases de datos y la información pasó por un proceso especial que impide revelar datos reales de los clientes *del banco*. Después de tener los respaldos, se realizaron las modificaciones necesarias, en la base de datos, para poder utilizar las nuevas funcionalidades de las aplicaciones, esto incluyó hacer una alteración de *tablas* para agregar *columnas*, inserción de registros de datos, etc. Posteriormente, se instalaron los aplicativos de prueba en todos los equipos que participarían y, finalmente, se crearon los accesos para los usuarios de los aplicativos de prueba.

Para iniciar las pruebas, se contactó a las sucursales que participarían para verificar que sus aplicativos de pruebas estuvieran instalados. Se procedió a pedirles que realizaran un ensayo para verificar que todas las configuraciones requeridas se habían realizado correctamente.

Inicialmente fueron grupos pequeños de sucursales los que participaron. Poco a poco estos grupos se fueron incrementando en número de sucursales, teniendo como objetivo final la participación de todas las sucursales *del banco*.

Durante estas pruebas se tuvieron pocos problemas de funcionalidad específica de la aplicación. Los demás problemas que se tuvieron durante la ejecución de las pruebas en sucursales se atribuyeron directamente al uso de un ambiente especial de pruebas distinto al ambiente de *producción*. A continuación se describen algunos ejemplos de los problemas y la forma en que se corrigieron o resolvieron.

3.4.5 Incidencia de funcionalidad: duplicación de registros en consulta de imágenes

En uno de los días de pruebas, dos sucursales reportaron que para una operación en la que se depositaron cinco *cheques* (simultáneamente), dos de los *cheques* aparecían duplicados en la consulta de imágenes de *cheques* digitalizados. No se reportó problema al momento de digitalizarlos ni se detectó problema en la operación o en otra consulta. Cuando se analizó a detalle el *código fuente SQL*, que hace la consulta sobre las imágenes de *cheques* digitalizados, se detectó que la causa de la duplicidad de esos *cheques* en la consulta se debía a la falta de asociación rigurosa (también conocido como "*join*") entre los registros de las *tablas* involucradas.

Abajo se muestran *códigos fuente SQL* similares a los de las consultas originales. Primero mostraremos la consulta con duplicidad de registros de *cheques* y en seguida el *código fuente* de la consulta sin duplicados:

```
--Primera version la consulta de imagenes de cheques
SELECT bc.banda_cheque, bc.num_caja, bc.num_sucursal
FROM captura_documentos cd INNER JOIN diario di ON cd.diario_pk =
di.pk, bandas_cheques bc
WHERE 1=1
AND cd.num_sucursal = ?
AND cd.num_caja = ?
AND (cd.feacha_creacion BETWEEN ? AND ?)
AND di.cancelado = 0
AND cd.digitalizado = ?
AND cd.tipo_cheque = ?
AND cd.num_caja = di.num_caja
AND cd.num_sucursal = di.num_sucursal
AND cd.num_consecutivo = di.num_consecutivo
AND cd.num_caja = bc.num_caja
AND cd.num_sucursal = bc.num_sucursal
AND cd.num_consecutivo = bc.num_consecutivo
AND cd.monto = bc.monto
AND cd.fecha_efectiva = bc.fecha_efectiva
```

```
--Consulta de imagenes de cheques (sin duplicados)
SELECT bc.banda, bc.num_caja, bc.num_sucursal
FROM captura_documentos cd INNER JOIN diario di ON cd.diario_pk = di.pk,
bandas_cheques bc
WHERE 1=1
AND cd.num_sucursal = ?
AND cd.num_caja = ?
AND (cd.feacha_creacion BETWEEN ? AND ?)
AND di.cancelado = 0
AND cd.digitalizado = ?
AND cd.tipo_cheque = ?
AND cd.num_caja = di.num_caja
AND cd.num_sucursal = di.num_sucursal
AND cd.num_consecutivo = di.num_consecutivo
AND cd.num_caja = bc.num_caja
AND cd.num_sucursal = bc.num_sucursal
AND cd.num_consecutivo = bc.num_consecutivo
AND cd.monto = bc.monto
AND cd.fecha_efectiva = bc.fecha_efectiva
AND cd.num_cuenta = substring(bc.banda_cheque, 16, 11)
AND cd.num_routing = substring(bc.banda_cheque, 6, 4)||-
||substring(bc.banda_cheque, 10, 5)
```

Después de la corrección se pasó a realizar otras pruebas. Lamentablemente la supuesta corrección causo más problemas que los que pretendía corregir.

El error fue que el formato de los valores en el campo "*cd.num_cuenta*" cambia según el tipo de cheque. Cuando el *cheque* era de otro banco, la cuenta se registraba tal cuál aparecía en la *banda magnética* del cheque. Sin embargo, en el caso de que el *cheque* fuera del mismo banco (*cheque propio*), la cuenta se almacenaba con un formato especial, en el cual se omiten ceros no-significativos. Debido a esto, después del ajuste, los *cheques* propios operados en un tipo de transacción específica, no aparecían en las consultas de imágenes pendientes o recibidas. En conclusión, cuando se arregló el problema de duplicación de *cheques*, también se provocó que algunos *cheques* fueran omitidos en el resultado de la consulta. Para arreglar esto se modificó la línea del *código fuente* anterior "*AND cd.num_cuenta = substring(bc.banda_cheque, 16, 11)*" para que sólo se utilizara en caso de que el *cheque* fuera de tipo *SBC*. Para el caso de *cheques* emitidos por el mismo banco, se validó directamente en el *código fuente Java* que no se duplicaran registros en la consulta.

El siguiente *código fuente Java*, genera la consulta *SQL* y contiene la corrección a la omisión de *cheques* en consultas y a la duplicidad de registros.

```
/**
* Metodo para consultar estado de imagenes de cheques: se filtra el
resultado por:
* fecha, caja, sucursal, tipo de cheque y estado de digitalizacion de
la imagen.
* @return List con resultados de consulta.
*/
public List obtenerEstadoImagenesCheques(ConsultaCaja consulta) {

    List cheques = null;
    StringBuffer sqlQuery = null;
    List params = new ArrayList(3);
    List types = new ArrayList(3);
    try {

        sqlQuery = new StringBuffer(" SELECT bc.banda_cheque,
cd.check_no, cd.num_cuenta, cd.num_routing, bc.num_caja,
bc.num_sucursal " + " FROM captura_documentos cd INNER JOIN diariodi
ON cd.diario_pk = di.pk, bandas_cheques bc WHERE 1=1 ");

        Sucursal sucursal = consulta.getSucursal();
        if (sucursal != null) {
            sqlQuery.append(" AND cd.num_sucursal = ? ");
            params.add( new Integer(sucursal.getNumSucursal()) );
            types.add( new Integer(Types.SMALLINT) );
        }

        Caja caja = consulta.getCaja();
        if ( caja != null) {
            sqlQuery.append(" AND cd.num_caja = ? ");
            params.add(caja.getNumCaja());
            types.add( new Integer(Types.SMALLINT) );
        }

        //FECHAS
        sqlQuery.append(" AND (cd.feacha_creacion BETWEEN ? AND ?)
");
        params.add( consulta.getFechaDesde() );
        params.add( consulta.getFechaHasta() );
        types.add( new Integer(Types.VARCHAR) );
        types.add( new Integer(Types.VARCHAR) );

        //Estado de digitalizacion de imagen
        final String statusImg;
        final String statusImgTemp = consulta.getStatus();
        if(
ImagenLlavesConstantes.IMAGEN_PENDIENTE.equals(statusImgTemp) ){
            statusImg = STATUS_DIG_IMG_PENDIENTE;
        }else if(
ImagenLlavesConstantes.IMAGEN_RECIBIDA.equals(statusImgTemp) ){
            statusImg = STATUS_DIG_IMG_RECIBIDA;
```

```
        }else if(
ImagenLlavesConstantes.STATUS_IMAGEN_ENVIADA_CECOBAN.equals(statusImg
Temp) ){
            statusImg = STATUS_DIG_IMG_CECOBAN;
        }else{
            log.warn("Estado de imagen de consulta desconocido!!");
            return new ArrayList(0); //regresa resultado vacio
        }

        //Para descartar cheques reversados
        sqlQuery.append(" AND di.cancelado = 0 " +
                        " AND cd.digitalizado = "" + statusImg + ""
");

        //Filtramos por tipo de Cheque en captura_documentos
        sqlQuery.append(" AND cd.tipo_cheque = ? ");
        types.add( new Integer(Types.CHAR) );
        if (Constantes.ETIQUETA_CHEQUES_SBC.equals(
consulta.getTipoCheque() ) ){
            params.add( CHAR_CHEQUE_TIPO_SBC );
        } else { /*if
(Constantes.ETIQUETA_CHEQUES_PROPIOS.equals(consulta.getTipoCheque()
) ){*/
            params.add( CHAR_CHEQUE_TIPO_PROPIOS );
        }

        //Agiliza JOIN captura_documentos CON diario.
        sqlQuery.append(" AND cd.num_caja = di.num_caja " +
                        " AND cd.num_sucursal = di.num_sucursal " +
                        " AND cd.num_consecutivo = di.num_consecutivo
");

        //JOIN captura documentos CON bandas cheques.
        sqlQuery.append(" AND cd.num_sucursal = bc.num_sucursal " +
                        " AND cd.num_caja = bc.num_caja " +
                        " AND cd.num_consecutivo = bc.num_consecutivo
" +
                        " AND cd.monto = bc.monto " +
                        " AND cd.fecha_efectiva = bc.fecha_efectiva "
+
                        " AND cd.num_cheque = convert(numeric,
substring(bc.banda_cheque, " +
Constantes.POSI_INI_NUM_CHEQUE_EN_BANDA + ", " +
Constantes.NUM_DIGITOS_NUM_CHEQUE_EN_BANDA + ")) " /* Join del num.
de cheque */
                        );


        //Completa el JOIN entre captura documentos y bandas cheques.
        if (Constantes.ETIQUETA_CHEQUES_SBC.equals(
consulta.getTipoCheque() ) ){
            sqlQuery.append(" AND cd.num_cuenta =
substring(bc.banda_cheque, " +
Constantes.POSI_INI_NUM_CUENTA_EN_BANDA + ", " +
Constantes.NUM_DIGITOS_NUM_CUENTA_EN_BANDA + ") "); //SOLO para SBCs
        }
```

```
        //SE DESHABILITA por causar un esfuerzo extra que
probablemente casi nunca se requerira
        //(Tendria que ser cheque con mismo monto y mismo numero de
cheque en la misma operacion
        /*
        if ( Constantes.ETIQUETA CHEQUES PROPIOS.equals(
consulta.getTipoCheque() ) ){
            sqlQuery.append(" AND right( replicate ("0",11)+
substring(cd.num cuenta,1,charindex("-",cd.num cuenta)-
1)+substring(cd.num_cuenta,charindex("-",cd.num_cuenta)+1,1),11) =
substring(bc.banda_cheque, 16, 11) ");
        }
        */

        //Copiamos tipos
        int typesSize = types.size();
        int[] typesIntArr = new int[typesSize];
        for(int i = 0; i < typesSize; i++){
            typesIntArr[i] = ((Integer)types.get(i)).intValue();
        }

        log.debug("obtenerCheques -> Query SQL:" +
sqlQuery.toString());

        List listaRes = jdbcTemplate.query(sqlQuery.toString(),
params.toArray(), typesIntArr, new ColumnMapRowMapper());

        if (listaRes != null && listaRes.size() > 0) {
            Iterator rs = listaRes.iterator();
            cheques = new ArrayList();

            /* Para revisar duplicidad de banda en el codigo... (en
vez de en query BD).*/
            final boolean isChequePropio =
Constantes.ETIQUETA CHEQUES  PROPIOS.equals( consulta.getTipoCheque()
);
            java.util.Set bandasPropioHS = new java.util.HashSet();

            iteraResultados:
            while (rs.hasNext()) {
                ResultadoConsultaImagenes resultado = new
ResultadoConsultaImagenes();
                Map registro = (Map) rs.next();

                String bandaCheque = (String)
registro.get("banda_cheque");
                String numCuenta;
                String numCheque;

                //BANDA CHEQUE
                if( bandaCheque != null && bandaCheque.length() >=
Constantes.LONGITD_MINIMA_BANDA_CHEQUE ){
                    bandaCheque = bandaCheque.replace("T", " ");
                    bandaCheque = bandaCheque.replace("O", " ");
                    bandaCheque = bandaCheque.replace("t", " ");
                    bandaCheque = bandaCheque.replace("o", " ");
                    bandaCheque = bandaCheque.replace("u", " ");
```

```
                    numCheque  = . . .
                    numCuenta  = . . .
                } else {
                    final String espacioStr = " ";
                    String cer_auten = "****"; //((String)
registro.get("cer auten")).trim();
                    String numRouting =
((String)registro.get("num_routing")).trim();
                    numCuenta =
((String)registro.get("num_cuenta")).trim();
                    numCheque =
((Integer)registro.get("num_cheque")).toString();
                    numRouting = numRouting.replaceAll("-", "");
//Quitamos guion en numRouting

                    while( numCheque.length() <
Constantes.NUM_DIGITOS_NUM_CHEQUE_EN_BANDA ){ /*Agregamos ceros no
significativos*/
                        numCheque = "0" + numCheque;
                    }

                    numCuenta = numCuenta.replaceAll("-", "");
                    while( numCuenta.length() < 11 ){
                        numCuenta = "0" + numCuenta;
                    }

                    bandaCheque = cer_auten + espacioStr + numRouting
+ espacioStr + numCuenta + espacioStr + numCheque;
                }

                /*Para revisar duplicidad de banda via codigo... (en
vez de en query BD).*/
                if( isChequePropio ){
                    if( bandasPropioHS.contains(bandaCheque) ){
                        continue iteraResultados;
                    }else{
                        bandasPropiosHS.add(bandaCheque);
                    }
                }

                //RESULTADOS
                 if (sucursal != null){

resultado.setSucursal(sucursal.getNombreSucursal());
                }else{
                    Integer num_sucursal =
(Integer)registro.get("num_sucursal");
                    if( num_sucursal != null ){
                        resultado.setSucursal(num_sucursal.toString()
);
                    }else{
                        resultado.setSucursal("TODAS");
                    }
                }

                if ( caja != null){
                    resultado.setCaja(caja.getNumCaja());
                } else {
```

```
                    Integer num_caja =
(Integer)registro.get("num_caja");
                    if( num_caja != null ){
                        resultado.setCaja( num_caja.toString() );
                    }else{
                        resultado.setCaja("TODAS");
                    }
                }

                resultado.setStatus( consulta.getStatus() );
                resultado.setCuenta( numCuenta );
                resultado.setNumCheque( numCheque );
                resultado.setNumBanda( bandaCheque );

                cheques.add(resultado);

                }//while

            }//if listaRes

            ...

        } catch (Throwable e) {
            throw new RuntimeException(e);
        }

    return cheques;
}
```

3.4.6 Incidencia de funcionalidad: imposibilidad para digitalizar cierto tipo de cheque

Durante las sesiones de capacitación a los cajeros, se detectó que el sistema no permitía digitalizar los *cheques de caja* cuando se pagaban al presentador. Estos tipos de *cheques* sí se podían digitalizar cuando se utilizaban en las demás operaciones, esto es, si se hacía una operación llamada "*depósito a cuenta*", el *cheque de caja* utilizado para el depósito sí se podía digitalizar, pero en la operación "*pago de cheque*" no era posible. Se detectó que el sistema no almacenaba las operaciones de pago de *cheques de caja* de la misma forma que lo hacía con los demás *cheques* en las otras operaciones. Para corregir esto se modificó la forma de guardar los *cheques*. Sin embargo, también se tuvo que ajustar el mecanismo de consulta de imágenes de *cheques* y la funcionalidad de digitalización. Afortunadamente, esta modificación no requirió

actualizar el componente del lado *cliente* de la aplicación. Bastó con actualizar el lado del *servidor* para que los cajeros en las pruebas vieran reflejada la corrección. La corrección se realizó en un día y su instalación fue inmediata por lo que los cajeros no tuvieron este problema en la siguiente sesión de pruebas.

3.4.7 Problema en pruebas: instalación y funcionamiento de escáner LS100

Al iniciar las pruebas en sucursales, se detectó que algunos equipos no contaban con el escáner LS100 instalado correctamente. Hubo varias razones, entre las que destacan dos: la falta de controladores en la PC y la falta del cable tipo USB o de corriente para conectar el *escáner* LS100 a la PC. Estos problemas impidieron que en dichos equipos se realizara la prueba de digitalización. Para solucionar esto se contactó a las sucursales para conocer el estado de los *escáneres* LS100. En los casos en los que se consideró viable, se les pidió a las sucursales que de forma masiva realizaran procedimientos que probaran la funcionalidad del *escáner* LS100. De esta forma se pudo recabar el estado de los dispositivos por sucursal para que finalmente se mandara personal capacitado a revisar la instalación física de los dispositivos. La instalación de los controladores en la PC se atendió con mayor velocidad debido a que se realizó de forma remota. En el caso de la falta de cables se tuvo que enviar a personal de soporte técnico para que llevara el o los cables y conectara el dispositivo a la PC de los cajeros. Eventualmente (entre dos y tres semanas) todos los escáneres LS100 funcionaron correctamente.

3.4.8 Problema en pruebas: escaneo incorrecto de cheques

Después de un día de pruebas, se detectó que entre los más de 3000 *cheques* que se habían digitalizado, 3 de ellos presentaban una imagen incorrecta. El error fue que la imagen que se tenía de cada uno de esos tres cheques presentaba una especie de “duplicación”, esto es, parecía que de alguna forma el escáner había digitalizado dos cheques uno tras otro, pero la imagen se traslapaba, por lo que se veía una parte de un *cheque* y sobre este otro *cheque* completo. Esto resultó un inconveniente puesto que la información de esta

imagen podría ser errónea, lo cual causaría complicaciones en el intercambio de imágenes con *Cecoban*. Lo que se hizo para evitar este problema fue agregar una validación en la aplicación. Esta validación se ejecuta cada vez que se pasa un *cheque* por el escáner LS100. La validación mide las dimensiones de la imagen del cheque, obteniendo la relación de la magnitud del alto sobre la del ancho del cheque. Cuando la proporción del ancho contra el alto sobrepasa un límite establecido, el sistema manda un mensaje al usuario advirtiendo que las proporciones del *cheque* sobrepasan la norma y se le solicita volver a digitalizar el cheque. De esta forma se espera que *el usuario* ponga mayor atención a la imagen mostrada por el *Sistema en ventanilla* y si es necesario vuelva a digitalizar el cheque. A la fecha se desconoce la razón por la que ocurrió este problema pues nunca se ha podido replicar el error y en las sucursales en que ocurrió sólo pasó una vez y después no volvió a suceder. Se espera que se pueda evitar en todos los casos en que habría sucedido con el mecanismo de validación que se agregó.

A continuación se muestra un fragmento de *código fuente* que contiene la funcionalidad de validación previamente descrita.

```
/**
 * Metodo que valida el tamanio de las imagenes escaneadas; revisa la
proporcion del
 * alto contra el ancho (no debe superar la "ESCALA_PERMITIDA").
 * Proposito: Evitar que se manden imagenes con cheques que posiblemente se
escanearon mal.
 * @author MAJL
 * Date: 27/Oct/2009
 */
public static void validaTamanioImagenes(byte[] imgAnverso, byte[]
imgReverso) {
    if(imgAnverso != null && imgReverso != null) {
        final double ESCALA_PERMITIDA = 2.5D; //TODO: Parametrizar
        ImageIcon tempAnverso = new ImageIcon(imgAnverso);
        int widthAnverso = tempAnverso.getImage().getWidth(null);
        int heightAnverso = tempAnverso.getImage().getHeight(null);
        ImageIcon tempReverso = new ImageIcon(imgReverso);
        int widthReverso = tempReverso.getImage().getWidth(null);
        int heightReverso = tempReverso.getImage().getHeight(null);
        double escalaAnverso = (double)widthAnverso /
(double)heightAnverso;
        double escalaReverso = (double)widthReverso /
(double)heightReverso;
        if(escalaAnverso > ESCALA_PERMITIDA || escalaReverso >
ESCALA_PERMITIDA) {
            JOptionPane.showMessageDialog(null,

recursosTexto.getString("msgImagenAnormal"),

recursosTexto.getString("titleWarning"),
                                          JOptionPane.OK_OPTION);
        }
    }
}

. . .
(donde:)
private final ResourceBundle recursosTexto;
. . .
recursosTexto = java.util.ResourceBundle.getBundle("mensajes",
Locale.getDefault());
. . .
(Archivo: mensajes_es.properties)
msgImagenAnormal=Las dimensiones de la imagen no corresponden al estandar.
Favor de validar.
```

Después de realizar la modificación se tuvo que instalar la nueva versión del componente cliente de la aplicación en todos los equipos que la utilizan. El proceso de distribución no se describirá a detalle, pero en general consiste en copiar componentes de la aplicación a varios servidores de los cuales posteriormente cada cliente se actualizará de manera automatizada al detectar que existe una nueva versión.

3.4.9 Problema en pruebas: ambiente de pruebas no actualizado

El siguiente problema se refiere a las operaciones que se realizan en *producción*, pero que no se llevan a cabo en el ambiente de pruebas. Por ejemplo: un cliente llega a una sucursal y solicita crear una nueva cuenta. Para abrirla realiza un depósito con un *cheque* a dicha cuenta. Pero cuando se trata de hacer la operación del depósito del *cheque* a la cuenta nueva en el ambiente de pruebas, *el usuario* recibirá un mensaje de error indicando que la operación no se puede realizar debido a que la cuenta destino no existe. En este caso *el usuario* debe contactar al área de sistemas e informar sobre este hecho. El área de sistemas procederá a crear la cuenta indicada por *el usuario* para habilitar su uso en el ambiente de pruebas. Este tipo de problema también puede ocurrir si los saldos de alguna cuenta cambian durante el día de la prueba, si se activa una chequera o se desactiva una cuenta, etc. La corrección de estos casos requiere que la sucursal informe a sistemas para que según sea el caso, se modifique la cuenta o la chequera en el *ambiente de pruebas* para quedar igual al *ambiente de producción* y así poder operar el *cheque* involucrado.

3.4.10 Problema en pruebas: inserción de registro con identificador duplicado

Para seguir el esquema utilizado por otras aplicaciones existentes, se decidió que las imágenes de los *cheques* digitalizados se almacenarían en una base de datos tipo *Oracle*. Durante las pruebas se detectó que por algún motivo desconocido, el *identificador único*, o *llave primaria,* de la *tabla* que contiene los registros de las imágenes de *cheques*, obtenida por el método que realiza la inserción de las imágenes, coincidía con el identificador de otro registro de una

imagen. Se piensa que esto sucedía por actividades de terceros en la base de datos, aunque nunca se pudo comprobar, pero con la solución empleada no se volvió a presentar el problema.

Para solucionar este problema, se creó un método que se ejecuta al iniciar la aplicación, del lado del *servidor*. Este método verifica que, en la base de datos, la *secuencia* de *Oracle* no genere un valor inferior al valor del *identificador único* más elevado de los registros existentes en la *tabla* de imágenes de *cheques* digitalizados.

El siguiente *código fuente* muestra el método creado para dar una solución veloz al problema:

```
/**
  * Metodo que valida que los valores devueltos por la secuencia:
  * PREFIJO_CHEQUE_IMG_SEQ sean validos.
  */
private void verificaSecuenciaEnImagenes() {

    String sql = "SELECT PREFIJO_CHEQUE_IMG_SEQ.nextval FROM dual";
    Long valorSequence = (Long) jdbcTemplate.queryForObject(sql,
Long.class);
    sql = "SELECT MAX(secuencia)+1 secuencia FROM PREFIJO_CHEQUE_IMG";
    Long valorMaxSecTabla = (Long) jdbcTemplate.queryForObject(sql,
Long.class);

    if ((valorSequence != null && valorMaxSecTabla != null) &&
valorSequence.longValue() < valorMaxSecTabla.longValue()) {
        log.error("Error en secuencia " + PREFIJO_CHEQUE_IMG_SEQ + " de
Oracle, el valor de la secuencia es MENOR al max(secuencia) de
PREFIJO_CHEQUE_IMG.");
        log.error("valorMaxSecTabla=" + valorMaxSecTabla + ",
valorSequence=" + valorSequence + ".");
        sql = " SELECT PREFIJO_CHEQUE_IMG_SEQ.nextval FROM dual";
        while(valorSequence != null && valorSequence.longValue() <
valorMaxSecTabla.longValue()) {
            valorSequence = (Long) jdbcTemplate.queryForObject(sql,
Long.class);
        }
        log.error("Termino de actualizar/avanzar valor de secuencia.");
    }
}
```

3.4.11 Problema en pruebas: uso incorrecto del ambiente de pruebas

Un problema que se presentó fue que al tener el *ambiente* de pruebas operando como un espejo del de *producción*, un cajero erróneamente realizó un depósito en efectivo en el ambiente de *pruebas* en lugar de hacerlo en el ambiente de *producción*. De este depósito se generó un comprobante que se entregó al cliente. Posteriormente el cliente informó al banco que no veía reflejada la cantidad que había depositado en su cuenta. Para evitar esta situación en el futuro, se procedió a realizar algunos cambios adicionales al ambiente de pruebas pero que no afectaran lo relacionado con la digitalización de los *cheques*. Por ejemplo, se procedió a modificar la funcionalidad de la aplicación para que al imprimir los recibos de depósitos y operaciones en el ambiente de pruebas, dichos recibos indicaran que eran de prueba y que no se vieran iguales a los del *ambiente productivo*.

3.4.12 Problemas en pruebas: ambientes de pruebas no exclusivos

Varios de los problemas, cuya causa no pudo rastrearse directamente como una falla en la aplicación, se atribuyeron específicamente al uso del ambiente de pruebas, debido a que dicho ambiente tiene diferencias con el ambiente de producción, que aunque sean mínimas, pueden provocar problemas e inconsistencias.

Los ambientes en los que se hicieron las pruebas no eran de uso exclusivo. Una operación realizada sobre la base de datos por un tercero podía generar conflictos en las pruebas. Frecuentemente, se realizaban cambios a la configuración que, sin saberse, podían afectar el funcionamiento de una u otra aplicación que trabajaba con el mismo *ambiente*. Para solucionar esto, se procuró tener comunicación constante con todas las personas que trabajaban en el mismo *ambiente*. Se evitó toda modificación a las configuraciones globales de los ambientes, así como la ejecución de tareas especiales, sin previo aviso, para que en caso de presentarse un error o inconsistencia, se considerara que podía deberse a un cambio hecho por otra persona.

3.4.13 Problema en pruebas: falta de capacitación de los cajeros

Un problema que se presentó durante las pruebas fue que algunos cajeros no estaban completamente informados sobre la nueva funcionalidad de la aplicación. A pesar de que se realizaron sesiones de capacitación presenciales con la gran mayoría de los cajeros en el área metropolitana, no fue posible hacerlo con los cajeros en otros estados de la república, por lo que se procedió a enviarles una guía y manual de cómo hacerlo. Sin embargo, algunos cajeros no obtuvieron de dicha guía todo el conocimiento que requerían. Para resolver este problema se procedió a establecer comunicación directa con ellos, vía telefónica o por correo electrónico, para aclarar los puntos en los que tuvieran dudas.

3.4.14 Problemas no-técnicos

Uno de los problemas no-técnicos fue explicar y dar a entender al usuario que hizo el *requerimiento* y a los directivos *del banco* involucrados, la razón de los problemas en las pruebas. Se hicieron analogías de los sistemas con proyectos anteriores, se explicaron los motivos y razones de todos los problemas y a pesar de ello al usuario le costó mucho trabajo entender y creer lo que nosotros le decíamos. La actitud del usuario fue "*hasta no ver, no creer*", por lo que se tuvo que hacer un gran esfuerzo para que los problemas en las pruebas fueran mínimos. A pesar de que algunas sucursales operaron sin un sólo problema desde las primeras pruebas, el usuario no estuvo satisfecho con las pruebas hasta que los problemas fueron mínimos (en promedio se tenía un problema en una sucursal por cada treinta sucursales). Hasta que finalmente el usuario se involucró más de cerca con las pruebas y vio por sí mismo, y no por lo que reportaban las sucursales, que los problemas no eran por fallas en la funcionalidad desarrollada en la aplicación; fue hasta entonces el usuario dio su visto bueno de las pruebas.

3.5 Contingencias

La aplicación se ha diseñado contemplando casos de fallas con el fin de poder continuar con la operación del negocio sin interrupción. En el caso de que un escáner LS100 deje de funcionar o que el equipo PC *del usuario* no funcione, basta con que se utilice otro escáner LS100 en la misma sucursal para digitalizar los *cheques* que quedaron pendientes por digitalizar en la *caja* donde se operaron originalmente. Si la sucursal completa tuviera algún problema y ningún equipo PC pudiera operar, los *cheques* tendrían que ser trasladados a otra sucursal y en ésta podrían ser digitalizados de la misma forma que en la sucursal donde se contemplaba hacerlo previamente.

Los demás casos de prevención de fallas se omiten por confidencialidad. Sólo se dirá que existen sistemas que trabajan como espejos del *ambiente productivo*.

3.6 Liberación a producción

En Enero de 2010, se tuvieron que realizar modificaciones a la aplicación *Sistema en ventanilla*, para cumplir con los lineamientos establecidos por el *SAT* en las reformas fiscales. La prioridad de este requerimiento se clasifica en lo que se conoce como “regulatorio”, por lo que fue requisito que se realizara y *liberara* para que *el banco* pudiera seguir operando, sin importar la situación de los otros cambios que se realizaban sobre el *Sistema en ventanilla*. Por esta razón, se tuvo que hacer el cambio y *liberarlo* a pesar de que el cambio para la *digitalización de cheques* no estaba validado por completo. Para esto, se modificó el *código fuente* de la aplicación (*Sistema en ventanilla)* de forma que se deshabilitaron las nuevas funcionalidades correspondientes a la *digitalización de cheques* y sobre esta versión se aplicaron las modificaciones. Se realizaron pruebas de la nueva versión en el área interna de *aseguramiento de calidad* o *QA del banco* y posteriormente se realizaron pruebas con *el usuario* de la aplicación. Las pruebas no tuvieron incidencias y se pasó a

realizar la instalación en producción. El día de la instalación, se realizaron pruebas básicas en una sucursal, previo a la apertura de las demás. Las pruebas funcionaron correctamente por lo que se preservó la versión. Las sucursales trabajaron correctamente durante el día sin reportar ningún problema, incidencia o desviación. A las ocho p.m. de la noche del día en que se hizo el cambio en producción, se recibió una llamada de la gente trabajando en el área central, reportando que se tenían *cheques* en físico, los cuáles no estaban registrados en el sistema como operados. Fueron aproximadamente 20 *cheques* en esta situación. Dichos *cheques* tuvieron que ser presentados en la *Cámara de compensación electrónica nacional de cheques* mediante otro esquema que implica la captura manual de los documentos.

Al día siguiente se revisó la programación y se detectó que por motivos de las pruebas de digitalización, había un cambio en el *código fuente* que se había instalado en el ambiente de *producción* que provocaba que la aplicación pudiera, en ciertas condiciones, omitir el registro de un *cheque* tipo *SBC* en una tabla de la base de datos. Este cambio no se tenía en la versión de la aplicación que previamente se utilizaba en el ambiente de *producción*. Para solucionar este problema, se creó un *código fuente SQL* que tomaba la información de los *cheques SBC* operados y realizaba la misma funcionalidad que debió haber hecho la aplicación. Afortunadamente la información de los *cheques* estaba disponible de forma distribuida en varias tablas de la base de datos, de lo contrario habría podido ser imposible solucionar el problema sin tener que regresar a la versión de la aplicación previamente instalada. El problema con regresar a la versión de la aplicación previamente instalada es que esta no tenía las modificaciones requeridas para aceptar pagos de *impuestos federales* de acuerdo con las reformas fiscales de 2010.

El siguiente fragmento de *código fuente SQL* fue utilizado para solucionar el problema previamente descrito:

```
SELECT holdeo.num_cuenta,
       holdeo.tipo_cuenta,
       substring(tran_liqd.no_banda, 6,2) tipo_docto,
       substring(tran_liqd.no_banda, 6,9) routing_no,
       substring(tran_liqd.no_banda, 16, 11) num_cuenta,
       substring(tran_liqd.no_banda, 28, 7) num_cheque,
       holdeo.monto AS monto,
       num_sucursal=convert(smallint, substring(tran_global.id_caja,1,4)),
       num_caja=convert(smallint, substring(tran_global.id_caja,5,2)),
       tran_global.num_tran_global sequence_no,
       0 estatus,
       . . .

       substring(tran_liqd.no_banda, 1, 4) cer_auten,
       null err_clas,
       null archivo,
       tran_liq.moneda_id AS crncy_id,
       tran_liqd.no_banda num_banda,
       (CASE WHEN holdeo.cantidad <= 10000 THEN 1 ELSE 0 END) AS
truncamiento

FROM holdeo_monto holdeo, transaccion_documento tran_doc, transaccion_global
tran_global, transaccion_liquida tran_liq, transaccion_individual tran_ind .
. .
WHERE tran_global.id_estatus = Aplicada
. . .

AND    tran_liq.id_medio = 5
AND    holdeo.hold_id = tran_liqd.hold_id
AND    tran_liqd.id_liquida_tran = tran_liq.id_liquida_tran
AND    tran_ind.id_tran_individual = tran_liq.id_tran_individual
AND    tran_global.id_tran_global = tran_ind.id_tran_global
AND    tran_global.fecha_efectiva = current_date()
```

Se aceptó el uso de valores no parametrizados, o dicho coloquialmente *"jarcodeados"* (del inglés "*hardcoded*"), debido a que dicho *código fuente* presenta una solución temporal que se utilizó sólo en tres ocasiones. Simultáneamente se modificó el *código fuente* de la aplicación *Sistema en ventanilla* para volver a realizar los registros en la base de datos de los *cheques* tipo *SBC* en todos los casos. La versión modificada del *Sistema en ventanilla* se

instaló un par de días después y el problema quedó solucionado de manera definitiva.

Esta falla se atribuyó: En primer lugar, a descuidos en el manejo del *código fuente*, puesto que este cambio no debió haberse incluido en esta versión de la aplicación, en segundo lugar, la falta de pruebas integrales o de regresión. Probando únicamente con la primera aplicación no se notaba ningún problema. El problema se observaba hasta que se trataba de hacer una operación en otra aplicación. El *Sistema en ventanilla* parecía funcionar bien, sin embargo, la aplicación que generaba los archivos para la *Cámara de compensación electrónica nacional de cheques* tenía problemas al generar estos archivos, debido a que el *Sistema en ventanilla* no almacenaba la información de los *cheques* en la base de datos como se esperaba que lo hiciera. El problema se solucionó de forma temporal con el *código fuente* mostrado previamente y un par de días después, se instaló la versión del *Sistema en ventanilla* con la corrección al problema y todo volvió a funcionar con normalidad.

Posterior a la liberación a producción que se debió a las reformas fiscales, se realizó la liberación de la nueva funcionalidad de la digitalización. Esto fue cuando se terminaron las pruebas de digitalización en las sucursales y se consideró que los usuarios estaban adecuadamente capacitados para operar la nueva funcionalidad. Teniendo en ese entonces mayor seguridad de que no ocurrirán fallas por algún motivo, se procedió a quitar la aplicación de pruebas y se instaló la nueva funcionalidad de digitalización de cheques en el ambiente productivo para emplearse en la operación diaria.

Una vez puesta la nueva funcionalidad en el *ambiente* productivo, primero se realizó una fase piloto en la que se empezaron a digitalizar todos los *cheques* en todas las sucursales, como parte de la operación normal, pero se siguió empleando la mensajería, de forma que en el área central se podía comparar que lo que se enviaría a *Cecoban* empleando el proceso tradicional, coincidiera exactamente con lo que se enviaría mediante el nuevo proceso de digitalización

de *cheques*. Los días que coincidieron los archivos para *Cecoban,* generados mediante los dos procesos, se enviaron los archivos obtenidos por medio del nuevo proceso.

Durante la fase piloto se detectaron algunos problemas, sin embargo estos se corrigieron sin tener que dejar de utilizar el nuevo proceso. Entre uno y dos meses después, se dejó por completo de utilizar el proceso tradicional y se trabajó únicamente con el nuevo proceso de digitalización de *cheques*.

La liberación a producción se realizó siguiendo el proceso establecido por *el banco* para todos los cambios a producción. Se realizó un '*control de cambios*', se revisó el cambio a detalle por las áreas involucradas y posteriormente, en una fecha previamente establecida, se realizaron los cambios para poder *liberar* la funcionalidad a *producción*.

3.7 Dispositivo para la digitalización de cheques

Los *cheques* se digitalizan utilizando un escáner modelo “LS100” fabricado por la empresa CTS.

Fig. 3.7a: Escáner de cheques: CTS LS100.

El escáner LS100 es capaz de leer la *banda magnética* de los *cheques* mediante un proceso de OCR. Esto facilita el ingreso y captura de la banda magnética de cada *cheque* operado en el *Sistema en ventanilla*. Adicionalmente el escáner LS100 es capaz de capturar la imagen del frente y del reverso de cada *cheque* procesado por el escáner. El *API* del escáner ofrece la posibilidad de almacenar las imágenes de los *cheques* procesados en varios formatos digitales (por ejemplo: *BMP*, *TIFF* y *JPEG*).

3.7.1 Interacción de la aplicación con el escáner LS100.

Para que el *Sistema en ventanilla* pueda interactuar con el escáner de *cheques* LS100, se utilizó un *framework* de *Java* llamado *JNI*.

A continuación se muestra un breve fragmento de *código fuente* de demostración en *Java* que ejecuta algunas de las funciones disponibles en el *API* del escáner LS100.

```
public class LS100{
. . .
  public static char SUSPENSIVE_MODE = "S";
  public static short NO_FRONT_STAMP = 0;
  public static short READ_CODELINE_MICR = 1;
. . .
  public static void Load()
  {
    System.loadLibrary("Symetry.LS100");
  }
  public static native int Open(char paramChar);
  public static native int Close(char paramChar);
  public static native long ReadImage(char paramChar1, short
paramShort, char paramChar2, long paramLong);
  public static native int SaveJPEG(char paramChar, long paramLong,
int paramInt, String paramString);
. . .
}

. . .
public class DemoJniLS100
{
  public static void main(String[ ] args)
  {
    LS100.Load();

    int lResult = LS100.Open(LS100.SUSPENSIVE_MODE);

    if (lResult >= 0)
    {
      int lNrDoc = LS100.DocHandle(LS100.SUSPENSIVE_MODE,
                                   LS100.NO_FRONT_STAMP,
                                   LS100.NO_PRINT_VALIDAT|E,
                                   LS100.READ_CODELINE_MICR,
                                   LS100.SIDE_ALL_IMAGE,
                                   LS100.SCAN_MODE_16GR200,
                                   LS100.AUTOFEED,
                                   LS100.SORTER_BAY1,
                                   LS100.WAIT_NO,
                                   LS100.BEEP);
```

```
      if (lNrDoc > 0)
      {
        long lImage = LS100.ReadImage(LS100.SUSPENSIVE_MODE,
LS100.CLEAR_ALL_BLACK, LS100.SIDE_FRONT_IMAGE, lNrDoc);

        if (lImage > 0L)
        {
          lResult = LS100.SaveJPEG(LS100.SUSPENSIVE_MODE, lImage,
127, "C:\\IMAGEN_CHEQUE_FRENTE.jpg");
          if (lResult < 0)
          {
            System.out.print("ERROR-SaveJPEG: ");
            System.out.println(lResult);
          }
          lResult = LS100.FreeImage(lImage);
        }
        else if (lResult < 0)
        {
          System.out.print("ERROR-ReadImage: ");
          System.out.println(lResult);
        }

        lImage = LS100.ReadImage(LS100.SUSPENSIVE_MODE,
LS100.CLEAR_ALL_BLACK, LS100.SIDE_BACK_IMAGE, lNrDoc);

        if (lImage > 0L)
        {
          lResult = LS100.SaveJPEG(LS100.SUSPENSIVE_MODE, lImage,
127, "C:\\IMAGEN_CHEQUE_REVERSO.jpg");
          if (lResult < 0)
          {
            System.out.print("ERROR-SaveJPEG: ");
            System.out.println(lResult); }
          lResult = LS100.FreeImage(lImage);
        }
        else if (lResult < 0)
        {
          System.out.print("ERROR-ReadImage: ");
          System.out.println(lResult);
        }

        System.out.print("CODELINE: ");

System.out.println(LS100.ReadCodeline(LS100.SUSPENSIVE_MODE));
        while (true)
        {
          System.out.println(
LS100.ReadBadgeWithTimeout(LS100.SUSPENSIVE_MODE,
LS100.FORMAT_IATA_ABA, 20000L) );
        }
      }

      System.out.print("ERROR-DocHandle: ");
      if (lNrDoc == 0)
      {
        System.out.println("NO HAY DOCUMENTO");
      }
      else
```

```
        {
          System.out.println(lNrDoc);
        }
        lResult = LS100.Close(LS100.SUSPENSIVE_MODE);
      }
      else
      {
        System.out.print("ERROR-Open: ");
        System.out.println(lResult);
      }
    }
}
```

El *código fuente Java* anterior incluye una llamada a un método para almacenar dos archivos de imagen en formato *JPEG* del frente y del reverso de un *cheque* e imprimir a la *consola de sistema* la banda magnética del *cheque* procesado en ese momento por el escáner LS100. En caso de ocurrir algún error, se imprimirá un aviso sobre el error en la *consola del sistema*.

3.8 Recepción de imágenes y envío de archivos a Cecoban

Para la generación de los archivos que se envían a *Cecoban*, se propuso una nueva aplicación para ser utilizada de forma central, la cual se encargaría de recuperar todas las imágenes digitalizadas en las sucursales y almacenadas en una base de datos. La aplicación también se encargaría de procesar las imágenes de los *cheques* para transformarlas en el formato utilizado en la "*Cámara de intercambio de imágenes*" de *Cecoban*. Cabe mencionar que a la fecha (2010) únicamente se requiere enviar imágenes de *cheques* cuyo valor supera el monto de diez mil pesos mexicanos o mil dólares estadounidenses. Sin embargo, todos los *cheques* son digitalizados, sin discriminar. Esta aplicación, también enviaría la información requerida por la "*Cámara de Compensación Electrónica Nacional de Cheques*". La aplicación se desarrolló por otra empresa con más experiencia trabajando con *Cecoban*.

3.9 Obtención de imágenes originales

Las imágenes de *cheques* obtenidas mediante el escáner LS100 no tienen colores, sino que son en tonos de grises. Por otro lado, las imágenes que se envían a *Cecoban* son puramente en blanco y negro, sin tonalidades de grises. Esto hace que las imágenes enviadas a *Cecoban* sean de menor calidad que las generadas originalmente y que a veces existan datos ilegibles en las imágenes enviadas.

Debido a que la aplicación empleada para la generación de archivos para *Cecoban* no contaba con una función para obtener las imágenes originales (con tonos de grises), se creó una tercera aplicación, muy sencilla, cuya única función consistía en buscar las imágenes originales de los *cheques* en la base de datos e importar el registro a un archivo que se almacenara en el equipo donde se ejecuta la aplicación.

Esta aplicación se creó con el fin de poder obtener las imágenes originales de los *cheques*, esto es, las imágenes en escala de grises que genera el escáner LS100. Por motivos de aclaraciones se puede requerir la imagen original o incluso una nueva digitalización del mismo cheque, con mejor calidad.

A continuación se muestra una imagen de la interfaz gráfica de usuario de dicha aplicación:

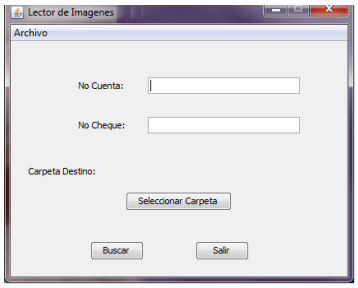

Fig. 3.9a: Interfaz gráfica de usuario de la tercera aplicación

La aplicación permite al usuario hacer búsquedas por *cheques* según el número de cuenta y/o número de cheque. Las imágenes se pueden visualizar directamente en la aplicación. Al realizar la búsqueda se muestra una tabla con los resultados obtenidos o en el caso de existir un único resultado, se procede directamente a mostrar el *cheque* una nueva ventana que únicamente muestra la imagen. La aplicación también cuenta con la opción de almacenar las imágenes localmente en el sistema de archivos, aunque esto podría resultar en un incidente de seguridad, por lo que se permite el uso de la aplicación solamente en equipos monitoreados. Posiblemente se agregaría la opción de realizar búsquedas también por fecha de digitalización.

Eventualmente se suspendió el uso de esta aplicación, debido a que se integró su función a la aplicación utilizada centralmente para generar los archivos para *Cecoban*.