Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

Marco de Desarrollo Estándar Basado en el Protocolo ISO-8583 para Terminales de Venta

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por el Bachiller:

Luis German Gil Messinese
V-16.474.875

Para optar por el título de Licenciado en Computación.

Tutor: Prof. Eric Gamess

Caracas, Diciembre 2014

Resumen

Título:
Marco de Desarrollo Estándar Basado en el Protocolo ISO-8583 para Terminales de Venta.

Autor:
Luis G. Gil Messinese

Tutor:
Prof. Eric Gamess

El presente Trabajo Especial de Grado consiste en la propuesta e implementación para el protocolo ISO-8583 de un marco de desarrollo que facilite la programación de aplicaciones en la amplia gama de terminales de venta electrónicos existentes.

El documento comienza con un análisis del protocolo ISO-8583 y un estudio teórico de los aspectos más relevantes que se deben tomar en cuenta para la implementación del marco de desarrollo. Durante la implementación del marco de desarrollo y la aplicación del terminal de venta se llevó a cabo bajo un proceso que abarca las fases de Análisis, Desarrollo y Pruebas.

El protocolo ISO-8583 es un estándar diseñado por la ISO (International Organization for Standardization) para el intercambio de transacciones financieras. A pesar de que ISO-8583 sea un estándar, no todas las empresas implementan el protocolo como lo describe su norma. Es de este singular conflicto que surge por parte de la empresa Corporación Novared C. A. la idea de proponer e implementar un marco de desarrollo que estandarice los mensajes de transacciones financieras que intercambian los terminales de venta y la plataforma Scolopendra.

En este trabajo se presentan las implementaciones que se llevaron a cabo en los terminales de venta y en la plataforma Scolopendra para completar el modelo de negocio planteado por la empresa GRE-5 a la empresa Corporación Novared C. A., que incluye un sistema de recargas electrónicas llevado a cabo en terminales de venta siendo este el primer prototipo que sale al ambiente de producción.

Palabras Claves: Plataforma Scolopendra, ISO-8583, Recargas Electrónicas, Terminales de Venta.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

1. Introducción

Hoy en día las empresas tratan de mejorar sus procesos de producción a través de innovaciones tecnológicas que permitan disminuir sus costos e incrementar la sus ganancias.

El mundo se ha globalizado, la competencia está en todos lados, estos nuevos desafíos han llevado a una transformación en la manera de realizar transacciones comerciales. La aparición de nuevas tecnologías ha facilitado la evolución natural de las redes de voz hacia los datos, hecho que se conoce mejor con el término de multimedia. Esto ha permitido la creación de medios más potentes y novedosos y de nuevos canales de relación entre personas o entre personas y sistemas.

En la actualidad, un proceso costoso es el de la producción de tarjetas Scratch-Card, el cual implica el pago de comisión por la producción, distribución y venta de la misma. Por tal motivo la empresa GRE-5 ha tomado la iniciativa de implementar una innovación sobre este proceso reemplazándolo por terminales de venta POS, como un método de recarga electrónica. Esto es debido a las características del POS de facilitar la portabilidad, comunicación y flexibilidad para adaptar dispositivos externos.

El objetivo del presente Trabajo Especial de Grado es la creación de un marco de desarrollo, es decir, una plantilla o una estructura que permita facilitar la construcción de aplicaciones en los diferentes modelos de terminales de venta. Se habla de la solución como un todo, en donde el problema se piensa combatir desde una perspectiva más amplia y general.

Con este fin, se ha estructurado el informe en 8 capítulos que explican los diferentes aspectos tomados en cuenta durante la creación de la aplicación. A continuación, se ofrece un breve resumen del contenido de cada uno de estos capítulos:

- Capítulo 2 (El Problema): Plantea los diferentes enfoques desde los que puede ser estudiado el problema junto al análisis de la solución; mostrando en detalle los objetivos y el alcance del presente Trabajo Especial de Grado.
- Capítulo 3 (ISO-8583): Presenta las bases teóricas que fueron estudiadas y sobre las que se fundamenta la creación de ISO-8583 estándar para transacciones financieras con mensajes originados en una tarjeta.
- Capítulo 4 (Plataforma Scolopendra): Describe el sistema al cual los terminales de venta se conectan para realizar las transacciones financieras.
- Capítulo 5 (Terminales de Venta): Presenta un análisis de las características, ventajas y desventajas de los dispositivos que fueron utilizados para realizar el Trabajo Especial de Grado.
- Capítulo 6 (Implementaciones Existentes del Protocolo ISO-8583): Presenta algunos trabajos que implementan el protocolo ISO-8583.

- Capítulo 7 (Marco de Desarrollo ISO-8583): Explica en detalle el proceso de implementación que se llevó a cabo.
- Capítulo 8 (Aplicación en el Terminal de Venta): Muestra el desarrollo que se realizó para generar una aplicación que realiza transacciones financieras con la plataforma Scolopendra.
- Capítulo 9 (Implementación en Scolopendra): Describe la fase de programación que se implementó en la plataforma Scolopendra.
- Capítulo 10 (Conclusiones): Presenta las conclusiones y recomendaciones obtenidas a partir del Trabajo Especial de Grado realizado.

2. El Problema

2.1 Planteamiento del Problema

Ante el interés de la empresa Gestora de Recargas Electrónicas GRE-5 de ofrecer una solución de recargas electrónicas, surge la necesidad de desarrollar un API (Application Programming Interface) que permita establecer la comunicación entre terminales de venta y el sistema Scolopendra utilizando el protocolo ISO-8583 [1] (estándar de comunicación para transacciones electrónicas financieras).

Debido a la amplia oferta de terminales de venta con arquitectura micro, se da una exigencia en la empresa GRE-5 para implementar un modelo que minimice los esfuerzos de desarrollo ante la incorporación de nuevos modelos y marcas de terminales.

Como parte del problema, se agregan las limitaciones del hardware que tiene el terminal, como la memoria y el repertorio de instrucciones del procesador. Los cuales constituyen una dificultad a la hora de implementar el estándar ISO-8583. Es necesario resaltar que así como es necesario desarrollar un protocolo de comunicación con el sistema Scolopendra, también es importante desarrollar una interfaz que permita a los usuarios manejar el terminal de manera sencilla e intuitiva.

2.2 Justificación del Problema

La creciente demanda que hay en los sistemas pre-pagados ha llevado a que las compañías que usan estos sistemas busquen alternativas que permitan disminuir los costos en la producción, distribución y consumo de las tarjetas Scratch-Card[1]. Una de las soluciones que ha tenido la aprobación de estas compañías, es la gestión de recargas electrónicas a través de los sistemas web, telefonía celular y terminales de venta. Cabe señalar que los terminales de venta (POS) son usados hoy en día en gran parte de los establecimientos del país, siendo esta una ventaja para las compañías con sistemas pre-pagados.

2.3 Objetivos

2.3.1 **Objetivo General**

Implementar un marco de desarrollo que regule y estandarice los procesos de desarrollo, para la inclusión de nuevas funcionalidades dentro de la solución de recargas electrónicas en la amplia gama de terminales de venta electrónicos existentes.

[1] Es una pequeña tarjeta, a menudo hecha de papel fino o de plástico para ocultar PINs en una o más áreas que contienen información que puede ser revelada por el rascado de una cubierta opaca.

2.3.2 **Objetivos Específicos**

- Obtener un amplio conocimiento del funcionamiento del protocolo de transacción financiera ISO-8583.
- Utilizar el lenguaje de programación C para implementar el protocolo de transacción financiera ISO-8583.
- Desarrollar un marco de desarrollo que permita utilizar el protocolo de transacción financiera ISO-8583 en forma estándar.
- Realizar pruebas para verificar el correcto funcionamiento.

2.4 Alcance

El marco de desarrollo debe permitir trabajar con una gran variedad de terminales de venta, de tal manera que al estandarizar sus procesos, este pueda facilitar el desarrollo de aplicaciones de usuario sobre los dispositivos, haciendo mucho más simple su implementación.

3. ISO-8583

ISO-8583 [1] es un estándar definido por la Organización Internacional de Estándares (ISO) utilizado para el intercambio de mensajes en transacciones financieras. Este estándar define un formato de mensaje y un flujo de comunicación para que diferentes sistemas puedan realizar estas transacciones.

A pesar de que ISO-8583 es un estándar común, no es utilizado en forma estricta por sistemas o redes. Sino que más bien es adaptado a las necesidades particulares de cada constructor.

El mensaje ISO-8583 tiene tres partes como se muestra en la Figura 3.1:

- Message Type Indicator (MTI): Es un indicador de tipo de mensaje.
- Bit Maps: Es un indicador que muestra que elementos del mensaje están activos.
- Data Elements: Son los campos del mensaje.

Figura 3.1: Mensaje ISO-8583

3.1 Message Type Indicator (MTI) - Indicador de Tipo de Mensaje

El campo MTI es un campo numérico que consta de cuatro dígitos para clasificar la función de alto nivel del mensaje. MTI consta de las siguientes secciones:

- Versión ISO-8583.
- Message Class - Clase del mensaje.
- Message Function - Función del mensaje.
- Message Origin - Origen del mensaje.

3.1.1 **Versión ISO-8583**

La versión del ISO-8583 especificada en el estándar, ocupa la primera posición dentro del campo MTI como se observa en la Tabla 3.1.

Posición	Significado
0xxx	Versión ISO 8583-1: 1987
1xxx	Versión ISO 8583-2: 1993
2xxx	Versión ISO 8583-3: 2003
9xxx	Para uso privado

Tabla 3.1: Versiones ISO-8583

3.1.2 **Message Class - Clase del Mensaje**

El campo de la clase del mensaje ocupa la segunda posición del MTI y se usa para especificar el propósito general del mensaje. En la Tabla 3.2 se puede ver como se clasifican los mensajes dentro del estándar ISO-8583 y cuál es el propósito de los mismos.

Posición	Significado	Propósito
x0xx	Uso reservado por la ISO	
x1xx	Mensaje de autorización	Se utiliza mayormente para transacciones con tarjetas de crédito, para verificar la disponibilidad de fondos, obtener una aprobación de una transacción, sistema de mensaje dual (DMS)[2]
x2xx	Mensaje financiero	Se utiliza preferiblemente para transacciones con tarjetas de débito, como la verificación de la disponibilidad de fondos, para aprobar e imputar directamente a la cuenta, y usa el sistema de mensaje individual (SMS)[3]
x3xx	Mensaje de manejo de archivo	Puede ser usado para cuadre de cajas con el banco, hot-card[4]
x4xx	Mensaje de reverso	Se utiliza para reversar o anular transacciones previamente autorizadas
x5xx	Mensaje de conciliación	Es utilizado para manejar la operación de cierre
x6xx	Mensaje administrativo	Se transmite información de una falla en los mensajes
x7xx	Mensaje de cobro de tarifas	
x8xx	Mensaje de manejo de red	Se utiliza para el intercambio de claves, el inicio de sesión, entre otras funciones de la red
x9xx	Uso reservado por la ISO	

Tabla 3.2: Tipo de Mensaje

[2]El sistema separa la autorización de la liquidación como dos diferentes procedimientos del software.
[3]El sistema envía una solicitud de autorización al banco emisor al mismo tiempo que constituye una confirmación financiera del pago.
[4] Las hot-cards son tarjetas que se encuentran en estado desactivado, por lo que pueden ser reactivadas o canceladas.

3.1.3 **Message Function - Función del Mensaje**

La función del mensaje ocupa la tercera posición del MTI y define cómo va a ser procesado el mensaje dentro del sistema. La Tabla 3.3 muestra el significado exacto de las funciones con las que cuenta el estándar ISO-8583.

Posición	Significado
xx0x	Requerimiento (Request)
xx1x	Respuesta del requerimiento (Request Response)
xx2x	Aviso (Advice)
xx3x	Respuesta del aviso (Advice Response)
xx8x	Reconocimiento de respuesta (Response acknowledgment)
xx9x	No reconocimiento (Negative acknowledgment)

Tabla 3.3: Función del Mensaje

3.1.4 **Message Origin - Origen del Mensaje**

El origen del mensaje provee la fuente de dónde se originó el mensaje, este campo ocupa la cuarta y última posición del MTI. En la Tabla 3.4 se especifica los valores que puede tomar este campo y su respectivo significado.

Posición	Significado
xxx0	Comprador
xxx1	Comprador repetición
xxx2	Emisor
xxx3	Emisor repetición
xxx4	Otros
xxx5	Otros repetición

Tabla 3.4: Origen del Mensaje

3.2 Bit Maps - Mapa de Bits

Dentro de un mensaje ISO-8583 existe el campo Bit Maps que indica cuáles elementos del campo de datos están activos en el mensaje. Este campo consiste en una cadena de 64 bits donde cada bit está asociado a un elemento de dato único. Si el contenido del dato está disponible, se establece en (1) el bit correspondiente, en cambio si no está disponible se establece el valor cero (0).

Un mensaje ISO-8583 puede utilizar tres Bit Maps de 64 posiciones cada uno, para ello existe un Bit Map primario el cual comprende los subcampos que corresponden a un rango de 1-64, pero estos subcampos también se puede extender con los bitmaps secundario y terciario que indican que los campos del 65-128 y 129-192 respectivamente están presentes en el mensaje.

El campo Bit Maps define varios formatos con los que puede ser transmitido el mensaje ISO-8583, entre estos se encuentra la representación como caracteres ASCII, EBCDIC, decimal codificado en binario (BCD).

3.3 Data Elements - Campos del Mensaje

Este campo se encarga de llevar la información correspondiente a la transacción, contiene ciento noventa y dos (192) subcampos, de los cuales la norma solo ha especificado los primeros ciento veintiocho (128), y estos (128) subcampos individuales tienen definidos un conjunto de tipos de datos, los cuales contienen un significado, un formato específico y su tamaño, así como también se incluyen subcampos de propósito general y algunas especialidades para sistemas o países. Además, los subcampos pueden tener un tamaño fijo o variable, para ello la norma especifica que los campos variables deben estar precedidos por un indicador del largo, de los cuales los tipos especificados por la norma son los siguientes:

- LLVAR – El LL es un indicador del tamaño del campo en el subcampo, y el VAR son los datos en el subcampo, una restricción de este campo es que el tamaño de estos tipos de subcampos están en el intervalo 01-99.
- LLLVAR – Al igual que su predecesor el LLL es un indicador del tamaño de los datos en el subcampo que está en el intervalo 001-999, y el VAR son los datos que le siguen.

A menos que se especifique lo contrario, todos los subcampos numéricos de longitud fija deben ser justificados a la derecha con ceros, los subcampos alfanuméricos de longitud fija deben justificarse a la izquierda con el carácter espacio, y campos binarios deben estar en bloques de ocho bits que se justifican a la izquierda y ceros.

Algunos subcampos son requeridos para procesar el mensaje ISO-8583, para ello la norma especifica que hay tres tipos de requerimientos para los subcampos, los cuales se muestran a continuación:

- Obligatorio: El subcampo y el contenido son necesarios para procesar este mensaje. El subcampo debe contener el texto apropiado o información numérica, como se indica.
- Opcional: El subcampo y el contenido no son obligatorios para procesar el mensaje, sino que deben ser proporcionados si está disponible.
- Condicional: Un subcampo puede ser condicional si se utiliza sólo en determinadas circunstancias. Los subcampos adicionales tienen que ser cargado en determinadas circunstancias por el remitente.

La norma ISO-8583 no especifica sobre cómo deben ser representados los subcampos. De tal manera un subcampo numérico puede ser representado como ASCII, EBCDIC, BCD.

4. Plataforma Scolopendra

La plataforma Scolopendra [7] fue diseñada por la Corporación Novared C. A. en el año 2008, como una solución para el manejo de múltiples transacciones financieras, que permite a sus clientes tener varios métodos de pago/intercambio, siendo capaz de concentrar gran cantidad de transacciones simultáneas, provenientes de múltiples canales.

Para cumplir con los objetivos planteados, Scolopendra se encuentra diseñada bajo dos pilares. En primer lugar, la estandarización de la comunicación. Scolopendra es una solución implementada para recibir peticiones que cumplen con el estándar de transacciones bancarias ISO-8583. Al apegarse a un estándar como ISO-8583, Scolopendra es un sistema que permite la fácil definición de nuevos procedimientos haciéndolo versátil y capaz de comunicarse con las plataformas tradicionales de comunicación financiera. En segundo lugar, es una arquitectura multi-tarea. Scolopendra es un sistema desarrollado bajo RUP [3], metodología para el desarrollo de aplicaciones basadas en el paradigma de orientación a objetos. Scolopendra posee una arquitectura donde un objeto central llamado balanceador, se encarga de distribuir las peticiones recibidas a través de múltiples puertos de comunicación TCP/IP, hacia múltiples procesos que interpretan, ejecutan y producen una respuesta de manera asíncrona y simultánea. Scolopendra está desarrollado bajo una arquitectura de tres capas, en primer lugar la capa de interfaz, la capa lógica, ambas capas se encuentran desarrolladas en el lenguaje de programación C++, brindando un desempeño óptimo en sistemas operativos Unix o Linux y finalmente la capa de repositorio de datos.

4.1 La capa de Interfaz

Está compuesta por todos los mecanismos de recepción y balanceo de peticiones mediante conexiones vía TCP/IP con cifrado SSL [2]. Las peticiones son recibidas por múltiples procesos encargados de escuchar cada uno por un puerto TCP/IP. Las peticiones recibidas son enviadas al balanceador de peticiones, un proceso que constantemente chequea su cola de peticiones y las reenvía basado en un algoritmo de FIFO hacia un proceso despachador de peticiones Round-robin, Round-robin con peso o Work-Stealing[5]. En esta capa existen varios parámetros configurables, como lo son, la cantidad de procesos receptores, los números de puertos a utilizar por los receptores, la cantidad máxima de conexiones aceptadas por cada proceso receptor, la posibilidad de habilitar o no el uso de SSL, el tiempo de inactividad del balanceador y el tipo de balanceo en la cola.

4.2 La capa de lógica

Constituyen todos los componentes que permiten la interpretación y ejecución de los comandos recibidos por el sistema, es aquí donde se lleva a cabo toda la

[5] Es una extensión del mecanismo de programación FIFO. El algoritmo mueve las tareas que aún no se han iniciado a los subprocesos que se están quedando sin elementos de trabajo

lógica de negocio. También existen varios parámetros configurables en esta lógica, los más importantes son: El período de inactividad de los procesos despachadores para chequear sus colas individuales de peticiones, las especificaciones del protocolo ISO-8583, los grafos de los procesos implementados por el sistema financiero (pago en línea, scratch card, billetera electrónica, F48, etc).

4.3 La capa de repositorio de datos

Está compuesto por el manejador de base de datos, así como el controlador de base de datos, que es una librería que encapsula el manejo de conexiones, así como la creación, ejecución y manejo de resultados de las consultas. Está diseñada bajo un proceso de normalización, con una base de datos modular y orientada a favorecer las consultas prioritarias y potenciando las capacidades de integridad de datos mediante restricciones del manejador MySQL

En la Figura 4.1 se puede ver el diagrama que representa la solución arquitectónica propuesta del sistema Scolopendra.

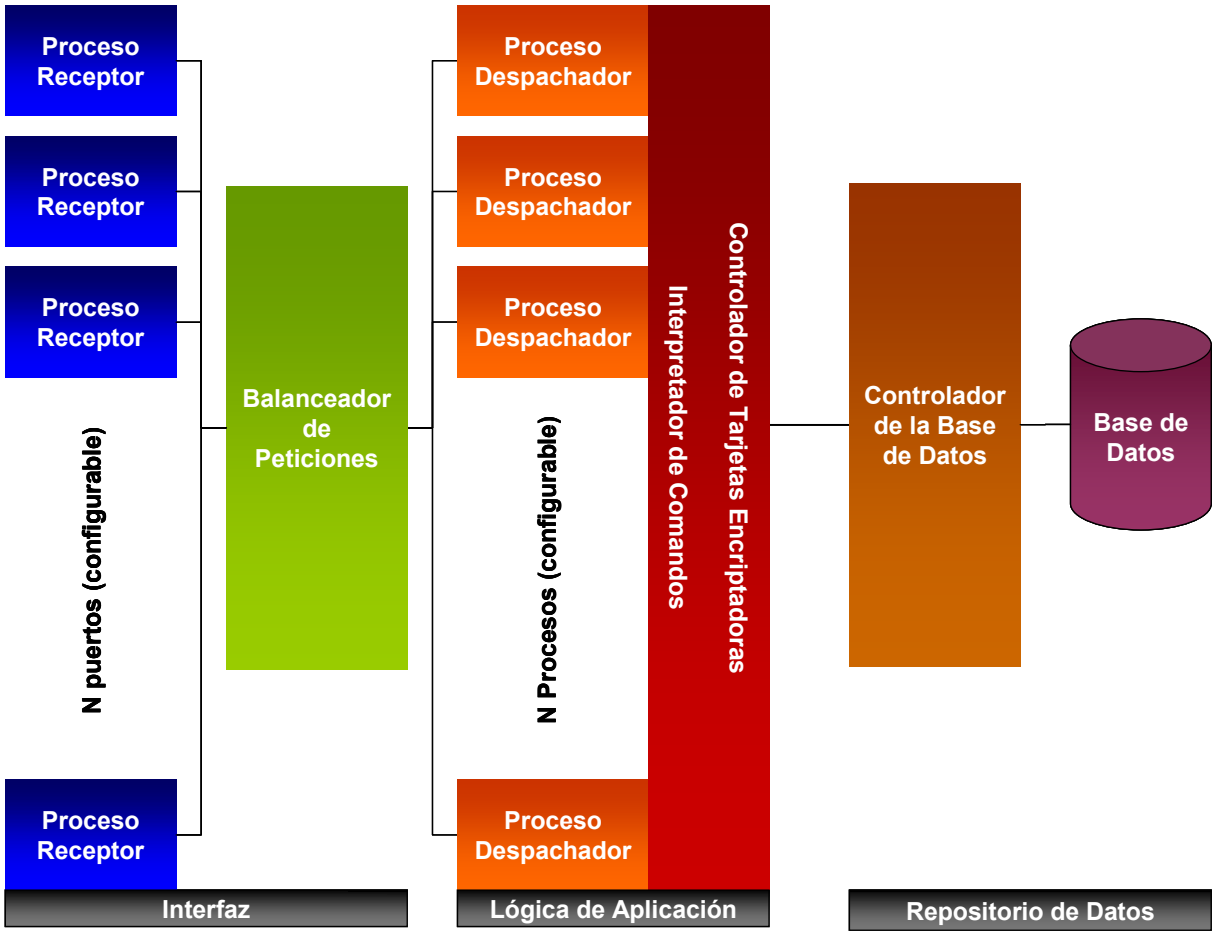

Figura 4.1: Arquitectura de Scolopendra

La modularidad de Scolopendra y su manera de interpretar los paquetes ISO-8583, recae en su totalidad en una clase denominada command_interpreter o interpretador de comandos. Esta clase analiza los paquetes y toma una decisión sobre cuál es el proceso a ejecutar basado en el contenido de tres de los campos incluidos en el paquete y que se detallan a continuación:

- Tipo de mensaje: Indica si se trata de un mensaje de aprobación, confirmación o cuadre de un proceso en específico.
- Número de proceso: Identifica el proceso que se va a ejecutar, es decir, diferencia cuál de los procesos funcionales incluidos corresponde al paquete recibido.
- Código del evento: Indica cuál de los pasos o estados dentro del proceso se está tratando de ejecutar, validar, confirmar, etc.

Basado en la conjugación de estos tres valores el sistema cuenta con un catálogo de acciones a seguir de manera que resulta sencillo entender, modificar e incluso expandir.

Para el desarrollo de una nueva funcionalidad es necesario extender el interpretador de comandos para recibir la nueva clase de peticiones, y luego desarrollar todas las consultas hacia la base de datos mediante la clase responsable. Estas mejoras son una característica de mucha importancia para la evolución del sistema y se sitúan como las actividades prioritarias del desarrollo de Scolopendra.

El sistema cuenta actualmente con el proceso de scratch cards como el principal proceso de transacciones electrónicas en producción. Ha estado operativo desde el arranque del sistema en Diciembre de 2008.

Existen varios procesos de transacciones electrónicas implementadas por Scolopendra, entre ellas destacan: (1) la billetera electrónica es una solución que permite la recolección de dinero en un punto de venta para solicitar un PIN dinámico creado al instante y entregado al cliente, (2) F48 permite al sistema generar un PIN sin recibir ningún pago, quedando registrado en una cuenta por pagar asociada al suscriptor y (3) La recarga electrónica genera un PIN asociado al monto agregado al suscriptor del proveedor de servicio de televisión por cable, telefonía celular, proveedor de servicio a Internet, entre otros.

5. Implementación en Scolopendra

En esta sección se describe el desarrollo que se llevó a cabo en la plataforma Scolopendra para cada uno de los mensajes generados por la aplicación del terminal de venta.

Para Scolopendra, cada código de proceso implica una clase que administra los eventos que éste puede efectuar en el proceso de negocio. En la Figura 5.1 se puede observar la declaración de la clase *independent_salesman_manager*, que contiene todos los procesos involucrados en el modelo de negocio del sistema de vendedores independientes (SVI) y su código de proceso es el “040000”. Esta clase, al igual que todas las clases que manejan los procesos de los modelos de negocios de Scolopendra, es una clase Singleton, esto especifica que solo puede haber una única instancia de la clase corriendo en Scolopendra.

```
44  class independent_salesman_manager : public e_card_manager{
45  private:
46      /**
47       *   Singleton pointer.
48       */
49      static independent_salesman_manager * manager;
50
51      /**
52       *   The default constructor.
53       */
54      independent_salesman_manager():e_card_manager(){
55
56      }
57
58      /**
59       *   The default constructor.
60       */
61      independent_salesman_manager(string configuration_filename):e_card_manager(configuration_filename){
62
63      }
64
65  public:
66
67      /**
68       *   This is the only way to get the only instance of independent_salesman_manager.
69       */
70      static independent_salesman_manager * get_manager();
71      /**
72       *   This is the only way to get the only instance of independent_salesman_manager.
73       */
74      static independent_salesman_manager * get_manager(string configuration_filename);
```

Figura 5.1: Código de la Clase independent_salesman_manager

Los métodos mostrados a continuación, son los procesos que involucran a la clase *independent_salesman_manager* que implementan la funcionalidad asociada al sistema de vendedores independientes:

5.1 Mensaje de Inicio de Sesión

Una vez recibido el mensaje de inicio de sesión en la plataforma Scolopendra, éste es tomado como un código de proceso “001000” y un código del evento “04”, de los cuales indica que el inicio de sesión es de un usuario proveniente del sistema de vendedores independientes:

En la Figura 5.2 se observa en la línea 374 el evento que se está procesando, en la línea 376 el método *validate_user* verifica que el nombre de usuario y la contraseña existen en la base de datos del sistema, y en la línea 380 obtiene el

saldo del usuario que está iniciando sesión y es fijado en el campo 4 del mensaje ISO-8583 enviado de respuesta al terminal.

```
374:   case 04: // Login with balance
375:     cout << "Command Interpreter - Login plus Balance\n";
376:     result_map = manager->get_manager()->validate_user( processing_code,
         event_code,user_string, log_code);
377:     result_map_reader(iso,result_map);
378:     user_company = result_map[34];
379:     user = result_map[104];
380:     string balance = manager->get_manager()->get_balance(user,
         user_company);
381:     iso->set_field(4,str_to_vector(balance.c_str()));
```

Figura 5.2: Código Fuente del Inicio de Sesión

El inicio de sesión es único para todos los modelos de negocio, por lo que este se maneja en una clase padre de la cual *independent_salesman_manager* hereda.

5.2 Mensaje de Consulta de Saldo

Cuando Scolopendra recibe un mensaje consulta de saldo del sistema de vendedores independientes (SVI), se envía al proceso “040000” con el código del evento “04”.

En la Figura 5.3 se puede ver en la línea 1094 el método *get_balance_sec_token*, quien se encarga de obtener el saldo que tiene el usuario para realizar ventas y el saldo pendiente por aprobar en el sistema. En la línea 1095 el método *result_map_reader*, se encarga de colocar en los campos 5 y 6 los saldos activos y pendientes, respectivamente, del mensaje ISO-8583 enviado de respuesta al terminal de venta.

```
1091:  case 7: //Generador de Balance
1092:    check_transaction_list = true;
1093:    cout << "Command Interpreter - Generador de Balance \n";
1094:    result_map = manager->get_manager()->get_balance_sec_token(
         processing_code, event_code,user_company,user,security_token);
1095:    result_map_reader(iso,result_map);
1096:    break:
```

Figura 5.3: Código Fuente de la Consulta de Saldo

5.3 Mensaje de Registro de Depósito

Los mensajes de registro de depósito del sistema de vendedores independientes que recibe el sistema Scolopendra con el código de proceso “040000” y el código del evento “10”

En la Figura 5.4 se ve en la línea 1117 el método *add_deposit* que se encarga de insertar nuevos registros de depósito en la base de datos con la información proveniente del terminal de venta y retorna un mensaje indicando si fue posible

insertar el depósito o el error por el cual no se pudo registrar. En la línea 1118 el método *result_map_reader* se encarga de colocar en el campo 39 del mensaje ISO-8583 si fue exitosa o no la transacción.

```
1114:  case 10: //Registro de depositos
1115:     check_transaction_list = true;
1116:     cout << "Command Interpreter - Registro de depositos \n";
1117:     result_map = manager->get_manager()->add_deposit(user_company,
          distributer, route, user, bank, amount, control, date, security_token,
.         processing_code, event_code);
1118:     result_map_reader(iso,result_map);
1119:     break;
```

Figura 5.4: Código Fuente del Registro de Depósito

El monto de un depósito registrado no es agregado al saldo de un usuario hasta que sea corroborado con el banco y aprobado por el sistema

5.4 Mensaje de Recarga Electrónica

En la recarga electrónica del sistema de vendedores independientes, el mensaje recibido por el sistema Scolopendra tiene el código de proceso “040000” y el código del evento “21” para crear la tarjeta y el evento “22” para confirmarla.

La Figura 5.5 muestra el código que ejecuta Scolopendra para realizar una recarga electrónica, en donde el método *create_e_card* es el encargado de crear un PIN asociado a la compañía de servicio del cliente y enviarle el mensaje de respuesta. En el evento de confirmación el método *confirm_e_card_topup* recibe el mensaje con los datos asociados a la creación del PIN y verifica que este coincida con el registrado en el sistema. Una vez verificado exitosamente, Scolopendra procede a realizar la recarga del PIN.

```
1248:  case 21: //Registro de depositos
1249:     check_transaction_list = true;
1250:     cout << "Command Interpreter - Venta de e Pin \n";
1251:     result_map = manager->get_manager()->create_e_card(processing_code,
          event_code, service_company, user_company, amount, user,
          security_token, pos, log_code, smart_card, serial, message_type);
1252:     result_map_reader(iso,result_map);
1253:     break;
1254:  case 22: // Confirmacion de Recarga
1255:     cout << "Command Interpreter - Confirmacion de recarga \n";
1256:     result_map = manager->get_manager()->confirm_e_card_topup (
          service_company, user_company, processing_code, event_code,
          smart_card, amount, pos, user, security_token, incoming_log_code,
          serial,"1"/*fixed to the creation status (1)*/);
1257:     result_map_reader(iso,result_map);
1258:     break;
```

Figura 5.5: Código Fuente de la Recarga Electrónica

5.5 Mensaje de Pago de Factura

El mensaje de pago de factura del sistema de vendedores independientes es recibido por Scolopendra con el código de proceso “040000” y evento “25” para crear el PIN asociado al pago y el “22” para confirmar el PIN.

La Figura 5.6, se observa en la línea 1273 el método *create_e_card_for_payment*, que es el encargado de crear un PIN asociado a la compañía de servicio a la cual el usuario va a generar un pago y enviarle el mensaje de respuesta. En el evento de confirmación el método *confirm_e_card_topup* recibe el mensaje con los datos asociados a la creación del PIN y verifica que éste coincida con el registrado en el sistema. Una vez verificado exitosamente, Scolopendra procede a validar el pago.

```
1270:  case 25: // Pagos a traves de pin electronico
1271:     check_transaction_list = true;
1272:     cout << "Command Interpreter - Pago con e Pin \n";
1273:     result_map = manager->get_manager()->create_e_card_for_payment(
          processing_code, event_code, service_company, user_company, amount,
          user, security_token, pos, log_code, smart_card, message_type);
1274:     result_map_reader(iso,result_map);
1275:     break;
```

Figura 5.6: Código Fuente del Pago de Factura

5.6 Mensaje de Cierre de POS

El mensaje de cierre de POS del sistema de vendedores independientes recibido por Scolopendra con el código de proceso “040000” y evento “43” para realizar el cierre y el evento “44” para la confirmación del cierre.

```
1296:  case 43: // Close out an e_pos
1297:     cout << "Command Interpreter - Close out e_pos \n";
1298:     result_map = manager->get_manager()->close_out_e_pos(processing_code,
          event_code, user_company, user, security_token, pos, date, total_txn,
          total_amount, transaction_list);
1299:     result_map_reader(iso,result_map);
1300:     break;
1301:  case 44: //Confirmation of e_pos closeout
1302:     cout << "Command Interpreter - Confirm close out e_pos \n";
1303:     response = manager->get_manager()->confirm_close_out_e_pos(
          processing_code,event_code,user_company,user,security_token,pos);
1304:     iso->set_field(38,str_to_vector("0"+response));
1305:     break;
```

Figura 5.7: Código Fuente del Cierre de POS

La Figura 5.7, se puede ver en la línea 1298 el método *close_out_e_pos* que se encarga de verificar y cuadrar si las transacciones que envía el terminal de venta corresponden a las registradas en el sistema. En la línea 1296 se observa el

método *confirm_close_out_e_pos*, que se encarga de marcar los PINs involucrados en el cierre como cierre exitoso, de tal manera que no se vuelvan a corroborar en cierres posteriores.

Por otra parte cada vez que se realiza alguna transacción, Scolopendra verifica si el terminal de venta envía un PIN que no haya podido ser confirmado. Esto permite que no existan transacciones sin confirmar en el sistema y que los terminales mantengan la integridad de los PINs que tiene registrado el sistema Scolopendra.

6. Terminales de Venta

En este capítulo, se presentan los terminales de venta que la empresa GRE-5 facilitó para el desarrollo del marco de trabajo.

6.1 Modelo de Terminal BD-632

BD-632[6] es un terminal portátil desarrollado por la empresa China Shenzhen Guanri Telecom-Tech para diversas aplicaciones de pago. Tiene un diseño compacto como se puede ver en la Figura 6.1. El terminal fue diseñado para la solución de aplicaciones de pago de facturas, recargas electrónicas, impresión de vales, transferencias de dinero, ventas de lotería, etc. El terminal está basado en POS inalámbricos y se combina con los sistemas de manejo de fondos. Cuenta con la comunicación inalámbrica, la transmisión de datos en tiempo real, la autenticación de identidad, la gestión de privilegios y la actualización del software.

Figura 6.1: Terminal BD-632

6.1.1 **Características**

La Tabla 6.1 muestra las características del terminal BD-632.

General	Red 2G	850/900/1800/1900 MHz
	SIM[7]	Mini SIM
Cuerpo	Dimensiones	198mm x 82mm x 60mm
	Peso neto	0.8 kg
	Teclado	21 botones
Lector tarjeta	Tarjeta inteligente	No
	Banda magnética	No
Pantalla	Visualización	128 x 64 píxeles, con retroiluminación LCD gráfico

[6] http://www.guanri.com.cn

[7] SIM (módulo de identificación de abonado) es una tarjeta inteligente desmontable usada en teléfonos móviles, módems HSPA etc.

	Fuentes	6 x 12 píxeles
Datos	GPRS	Sí
	EDGE	Sí
	USB	Sí , Mini USB
	RS - 232	Sí
Procesador	Coretex-M3	CPU 32 bits, 72 MHz
Memoria	Memoria interna	4MB
Impresora	Tipo de impresora	Impresora térmica
	Ancho de impresión	58mm
	Velocidad	50mm/s
Batería	Sí	batería de iones de litio 7.4V/1800 mAH
	Tiempo de espera	100 horas máximo.
	Duración operativa	48 horas (aprox. 250 transacciones)
Alimentación	Input	AC 100V~240V, 50 Hz~60 Hz
	Output	DC 9V/2A
Desarrollo	Lenguaje de desarrollo	Lenguaje C
	Herramienta de desarrollo	Keil, IDE Eclipse

Tabla 6.1: Características del Terminal BD-632

6.1.2 **Desarrollo**

El desarrollo de terminal BD-632 se basa en la programación en lengua C y se utiliza el compilador Keil MDK para procesadores Coretex-M3 [11]. Además este terminal viene con un conjunto de herramientas que facilitan la compilación y arranque del sistema, así como también una plantilla como guía para su programación. Por lo que el usuario debe solamente programar el sistema que desea implementar en el terminal.

6.1.3 **Ventajas y Desventajas**

Las ventajas del terminal BD-632 son:

- Aplicable para varios ambientes de negocio.
- Diseño modular para facilitar el mantenimiento.
- Provee herramientas para la compilación.
- Suministra una plantilla de guía para el desarrollo inicial.
- Capaz de almacenar datos de transacciones.
- Impresión en tiempo real.
- Tiene una interfaz amigable.

Las desventajas del terminal BD-632 son:

- Poca o nada de documentación a la hora de desarrollar.
- Para hacer pruebas hay que cargar el firmware de la aplicación a la memoria ROM.
- No cuenta con un simulador.

6.2 Modelo de Terminal Multicab POS

Multicab POS[8] es un terminal diseñado por la empresa argentina Bonus Comunicaciones (ver Figura 6.2), que permite realizar transacciones electrónicas en tiempo real contra un servidor administrador o plataforma de servicio. Funciona como una terminal esclava de dicho servidor con el cual dialoga con simples comandos para el ingreso de datos, informes a través de la pantalla e impresión de tickets o comprobante de operación.

Figura 6.2: Multicab POS

6.2.1 **Características**

La Tabla 6.2 contiene las especificaciones y características que define el diseño del Multicab POS.

General	Red 2G	850/900/1800/1900 MHz
	SIM	Mini SIM
Cuerpo	Dimensiones	230mm x 170mm x 60mm
	Peso neto	0.64 kg
	Teclado	16 botones
Lector tarjeta	Tarjeta inteligente	No
	Banda magnética	No
Pantalla	Visualización	128 x 64 píxeles, con retroiluminación LCD gráfico
	Fuentes	6 x 12 píxeles
Datos	GPRS	Sí
	EDGE	Sí
	USB	No
	WiFi	IEEE 802.11b, 2.4GHz
Procesador	No especificado	
Memoria	No especificado	

[8] http://www.bonuscom.com.ar

Impresora	Tipo de impresora	Impresora térmica
	Ancho de impresión	58mm
	Velocidad	50mm/s
Batería	No	
	Tiempo de espera	
	Duración operativa	
Alimentación	Input	AC 100V~240V, 50 Hz~60 Hz
	Output	12 VDC / 700 mA
Desarrollo	Lenguaje de desarrollo	HTML
	Herramienta de desarrollo	No

Tabla 6.2: Características Multicab POS

6.2.2 **Desarrollo**

La programación en el Multicab POS [12] consiste en un intercambio de mensajes de tipo XML, entre el terminal y un servidor. El protocolo comunicación incluye mensajes en ambas direcciones, por lo que se debe implementar un servicio al cual el terminal se debe conectar, dicho servicio establece un intercambio de mensajes los cuales permiten la carga del menú, proceso de recarga y configuraciones. Un ejemplo de la comunicación del terminal se puede ver en la Figura 6.3.

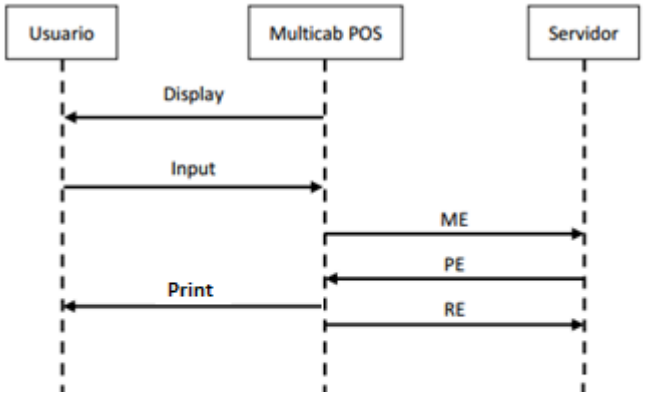

Figura 6.3: Comunicación Multicab POS a Servidor

6.2.3 **Ventajas y Desventajas**

Las Ventajas del terminal Multicab POS son:

- Fácil utilización y rápida implementación en el servidor por comandos XML.
- Impresión en tiempo real.
- Tiene una interfaz de operaciones amigables.
- Facilita los siguientes requerimientos:
  - Recargas de celulares.
  - Pago en efectivo de facturas con validación electrónica online e impresión de comprobante.

- Autorización de prestaciones online.
- Sorteos electrónicos instantáneos.

Las desventajas del terminal Multicab POS son:

- Poca documentación.
- No es capaz de almacenar datos de transacciones.
- No tiene batería.
- No provee protocolo de encriptación de datos.
- No es robusto.

6.3 Modelo de Terminal TPS300

TPS300[9] es un terminal desarrollado por la compañía china Telepower Communication (ver Figura 6.4), se trata de un terminal de transacciones móviles con una impresora térmica y batería. Es compatible con el procesamiento de la transferencia de dinero, pago de facturas, reserva de entradas, venta de comida en línea, loterías y recargas electrónicas.

Figura 6.4: Terminal TPS300

6.3.1 **Características**

La Tabla 6.3 contiene en detalle las características y propiedades del terminal TPS300.

General	Red 2G	850/900/1800/1900 MHz
	SIM	Mini SIM
Cuerpo	Dimensiones	205mm x 81mm x 68mm
	Peso neto	0.8 kg
	Teclado	21 botones
Lector tarjeta	Tarjeta inteligente	No
	Banda magnética	No
	Visualización	128 x 64 píxeles, con retroiluminación

[9] http://www.telpo.com

Pantalla		LCD gráfico
	Fuentes	12 x 24 píxeles
Datos	GPRS	Sí
	EDGE	Sí
	USB	Sí, Mini USB
	RS - 232	Sí
Procesador	ARM7	CPU 32 bits, 72 MHz
Memoria	Memoria RAM	4MB
	Memoria interna	64MB
Impresora	Tipo de impresora	Impresora térmica
	Ancho de impresión	58mm
	Velocidad	50mm/s
Batería	Sí	Batería de iones de litio 7.4V/1800 mAH
	Tiempo de espera	100 horas máximo
	Duración operativa	40 horas
Alimentación	Input	AC 90V~250V, 50 Hz~60 Hz
	Output	DC 12V/2A
Desarrollo	Lenguaje de desarrollo	Lenguaje C
	Herramienta de desarrollo	Compilador para procesadores ARM, IDE Microsoft Visual Studio y un simulador SafeNet.

Tabla 6.3: Características TPS300

6.3.2 **Desarrollo**

El terminal TPS300 utiliza un conjunto de herramientas que facilitan su programación, como lo es el API Microsoft Visual Studio 6.0 [8]. Con Visual Studio, se puede integrar las herramientas que proporciona Telepower Communication para el desarrollo de soluciones. Por otra parte se tiene la opción de adquirir un dispositivo hardware desarrollado por la compañía SafeNet llamado SoftDog [9] (ver Figura 6.5), y un software manejador del simulador denominado VS2008, de tal manera que este dispositivo permita la simulación casi exacta del terminal TPS300. Además el terminal también provee una plantilla de soluciones y manual de referencia de los métodos del API [10]. De esta manera se hace más simple la implementación y desarrollo de este terminal, puesto que no hace falta cargar el firmware de la aplicación a la memoria ROM del dispositivo cada vez que se necesite probar un nuevo requerimiento. A pesar del conjunto de herramientas que facilita la compañía proveedora de este terminal, se hace necesario mencionar que las herramientas solo funcionan en un sistema operativo Windows XP.

La programación está basada en el lenguaje de programación C, la plantilla está desarrollada en el mismo lenguaje, y como tal facilita un conjunto de ejemplos de los posibles modelos de negocio que se pueden ofertar, o incluso tiene ejemplos orientadas al funcionamiento de parte del hardware, como la impresora, el sonido de teclas, la iluminación de la pantalla, entre otras.

Figura 6.5: SafeNet SoftDog

En la Figura 6.6 se puede ver un ejemplo de cómo es el desarrollo para el terminal TPS300 en el sistema operativo Windows XP usando el simulador SofDog[10].

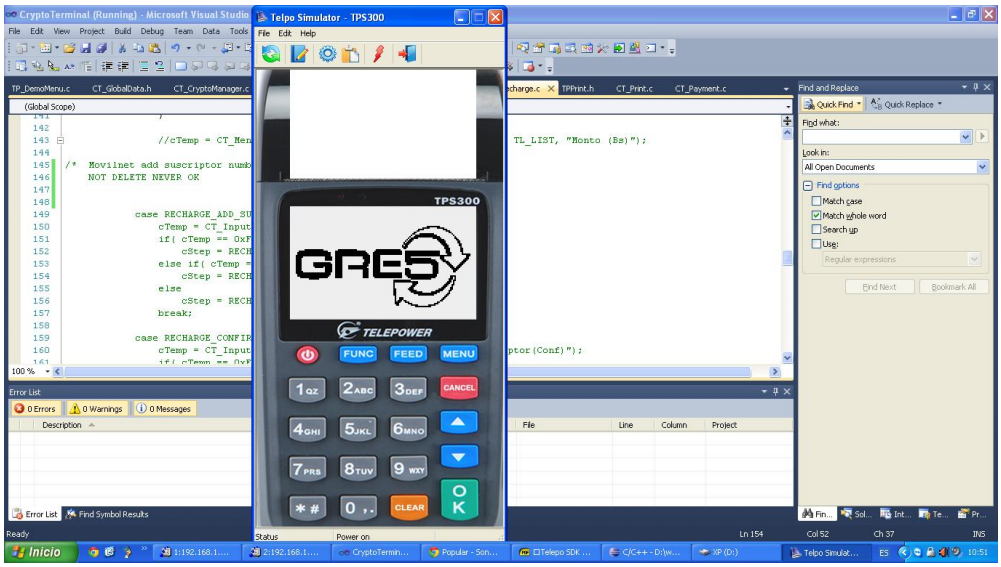

Figura 6.6: Simulación TPS300

6.3.3 **Ventajas y Desventajas**

Las ventajas del terminal TPS300 son:

- Aplicable para varios ambientes de negocio.
- Diseño modular para facilitar el mantenimiento.
- Provee herramientas para la compilación.
- Suministra una plantilla completa de todas las posibles soluciones.
- Capaz de almacenar datos de transacciones.
- Impresión en tiempo real.
- Tiene una interfaz de operaciones amigable.
- Proporciona un manual de referencias del API.
- Tiene batería.
- Tiene soporte para el desarrollo.

Las desventajas del terminal TPS300 son:

- Algunas referencias de métodos están completamente en chino.

[10] http://www.safenet-inc.com

- El hardware de simulación debe comprarse por separado.
- Solo puede ser utilizado el sistema operativo Windows XP para el desarrollo de soluciones para el terminal.

6.4 Modelo de Terminal ICT220

ICT220[11] es un terminal desarrollado por la empresa francesa Ingenico (ver Figura 6.7). Es un terminal diseñado principalmente para que pequeños comerciantes provean a sus clientes de diferentes métodos de pagos electrónicos. El ICT220 da a sus clientes comerciales la libertad de aceptar todo tipo de pagos con tarjeta electrónica sin necesidad de una caja registradora o interfaz del sistema POS. Este dispositivo es robusto y fácil de usar, también lleva las medidas de seguridad avanzadas de Ingenico. Posee un lector de tarjetas inteligentes y un lector de tarjetas de banda magnéticas.

Figura 6.7: Terminal ICT220

6.4.1 **Características**

La Tabla 6.4 muestra las características y propiedades que califican al terminal ICT220.

General	Red 2G	No
	SIM	No
Cuerpo	Dimensiones	83mm x 185mm x 63mm
	Peso neto	0.325 kg
	Teclado	19 botones
Lector tarjeta	Tarjeta inteligente	Sí
	Banda magnética	Sí
Pantalla	Visualización	128 x 64 píxeles, con retroiluminación LCD gráfico
	Fuentes	6 X 12 píxeles
Datos	GPRS	No
	EDGE	No

[11] www.ingenico.com

	USB	Sí , Mini USB
	RS - 232	Sí
Procesador	ARM9 & ARM7	450MIPS & 50MIPS
Memoria	Memoria RAM	16MB
	Memoria interna	16MB
Impresora	Tipo de impresora	Impresora térmica
	Ancho de impresión	58mm
	Velocidad	50mm/s
Batería	No	
	Tiempo de espera	
	Duración operativa	
Alimentación	Input	AC 110V, 60 Hz
	Output	DC 12V/2A
Desarrollo	Lenguaje de desarrollo	Lenguaje ANSI C
	Herramienta de desarrollo	Telium 2, IDE basado en Eclipse

Tabla 6.4: Características ICT220

6.4.2 **Desarrollo**

El desarrollo en el terminal ICT220 se basa en el lenguaje de programación ANSI C. Se utiliza un compilador ARM para generar el firmware de la aplicación que se va a cargar en el terminal. Para el desarrollo de nuevos requerimientos se utiliza Eclipse. Además existe una variedad de simuladores tanto en software como en hardware que se ajustan al IDE Eclipse, permitiendo facilitar el desarrollo de nuevos requerimientos sin tener que cargar el firmware de la aplicación a la memoria ROM del terminal cada vez que se desee hacer pruebas sobre el mismo. Asimismo la compañía Ingenico provee de una gran cantidad de soluciones de comercio electrónico en sus plantillas, de igual manera provee ejemplos del funcionamiento del hardware del terminal.

6.4.3 **Ventajas y Desventajas**

Las ventajas del terminal ICT220 son:

- Aplicable para varios ambientes de negocio.
- Diseño modular para facilitar el mantenimiento.
- Provee herramientas para la compilación.
- Suministra una plantilla completa de todas las posibles soluciones.
- Capaz de almacenar datos de transacciones.
- Impresión en tiempo real.
- Tiene una interfaz de operaciones amigables.
- Proporciona un manual de referencias del API.

Las desventajas del terminal ICT220 son:

- No tiene batería.
- No tiene conexión GPRS.

6.5 Modelo de terminal I5100

I5100[12] es un terminal desarrollado por la empresa francesa Ingenico (ver Figura 6.8). Es la solución efectiva de pago para los comerciantes con bajo o medio nivel de transacción. Diseñado para la máxima versatilidad y facilidad de uso, las I5100 incorporan una gran pantalla gráfica, una impresora ultra-silenciosa de sencilla carga, y un teclado retroiluminado para la entrada de PIN sin problemas. Posee un lector de tarjetas y un lector de tarjetas magnéticas haciendo del I5100 uno de los terminales más conveniente para paquete de pagos electrónicos.

Figura 6.8: Terminal I5100

6.5.1 **Características**

La Tabla 6.5 contiene las especificaciones y características que el terminal I5100 provee.

General	Red 2G	Opcional
	SIM	Opcional
Cuerpo	Dimensiones	210mm x 95mm x 75mm
	Peso neto	0.600 kg
	Teclado	18 botones
Lector tarjeta	Tarjeta inteligente	Sí
	Banda magnética	Sí
Pantalla	Visualización	128 x 64 píxeles, con retroiluminación LCD gráfico
	Fuentes	6 X 12 píxeles
	GPRS	Sí

[12] www.ingenico.com

Datos	EDGE	Sí
	USB	No
	RS - 232	Sí
Procesador	ARM7	29 Mhz
Memoria	Memoria RAM	4MB
	Memoria interna	8MB
Impresora	Tipo de impresora	Impresora térmica
	Ancho de impresión	58mm
	Velocidad	50mm/s
Batería	No	
	Tiempo de espera	
	Duración operativa	
Alimentación	Input	AC 100-230V 50-60 Hz
	Output	DC 12V/2A
Desarrollo	Lenguaje de desarrollo	Lenguaje ANSI C
	Herramienta de desarrollo	Telium 2, IDE basado en Eclipse

Tabla 6.5: Características I5100

6.5.2 **Desarrollo**

El desarrollo en el terminal I5100 se basa en el lenguaje de programación ANSI C, con un compilador ARM. Además, para ayudar al programador a desarrollar las soluciones de negocios, la compañía proporciona una plantilla de desarrollo inicial basada en el lenguaje ANSI C, el cual contiene ejemplos del manejo del terminal. De igual manera para hacer las pruebas mucho más fáciles al programador, Ingenico ofrece un simulador en software o hardware, para que el programador no tenga que cargar el firmware de la aplicación a la memoria ROM del terminal cada vez que desee hacer pruebas del funcionamiento de un nuevo requerimiento.

6.5.3 **Ventajas y Desventajas**

Las ventajas del terminal I5100 son:

- Aplicable para varios ambientes de negocio.
- Diseño modular para facilitar el mantenimiento.
- Provee herramientas para la compilación.
- Suministra una plantilla completa de todas las posibles soluciones.
- Capaz de almacenar datos de transacciones.
- Impresión en tiempo real.
- Tiene una interfaz de operaciones amigables.
- Proporciona un manual de referencias del API.
- Tiene conexión GPRS.
- Tiene conexión WiFi.

Las desventajas del terminal I5100 son:

- No tiene batería.

6.6 Tabla Comparativa

La Tabla 6.6 muestra una comparación de hardware, facilidad de desarrollo, costo, etc., de los diferentes terminales presentados anteriormente de manera que se pueda apreciar las ventajas y desventajas de cada modelo.

	BD-632	Multicab POS	TPS300	ICT220	I5100
Procesador	Coretex-M3	No especificado	ARM7	ARM9 & ARM7	ARM7
Memoria RAM	No especificado	No especificado	4MB	16Mb	4MB
Memoria Flash	4MB	No especificado	64Mb	16MB	8MB
Impresora	Térmica 58mm	Térmica 58mm	Térmica 58mm	Térmica 58mm	Térmica 58mm
Batería	Sí	No	Sí	No	No
GPRS	Sí	Sí	Sí	No	No
Ethernet	No	No	No	Sí	Sí
USB	Sí	No	Sí	Sí	Sí
RS - 232	Sí	No	Sí	Sí	Sí
WiFi	No	Sí	No	No	No
Lenguaje de desarrollo	Lenguaje C	XML	Lenguaje C	ANSI C	ANSI C
Soporte ISO-8583	Sí	No	Sí	Sí	Sí
Costo	$$$$	$$$$	$$$	$$$$$	$$$$

Tabla 6.6: Comparativa de Terminales

7. Implementaciones Existentes del Protocolo ISO-8583

Actualmente existen varias implementaciones del protocolo ISO-8583 que pueden ser utilizados para el desarrollo de aplicaciones que manejen este tipo de comunicaciones. A continuación, se describen las más conocidas.

7.1 jPOS

jPOS[13] es un software de código abierto, basado en el protocolo ISO-8583 estándar para transacciones financieras con mensajes originados en una tarjeta - especificaciones de los mensajes de intercambio [1]. Está desarrollado en Java y proporciona un manejador de mensajes basados en la norma ISO-8583 entre las redes de procesamiento y los sistemas internos.

Fue creado por Alejandro Revilla a finales de 1990, escrito inicialmente para reemplazar el software que se ejecutaba en Visanet. Se utiliza actualmente en más de 80 países de los cinco continentes impulsando las aplicaciones de pago de alta exigencia. jPOS y sus proyectos alojados se distribuyen bajo la Licencia Pública General Affero GNU[14]. Por lo que sólo es adecuada para los usuarios finales que tengan la intención de redistribuir sus aplicaciones basadas en jPOS o jPOS Extended Edition; o cuando el código fuente de la solución también está disponible bajo una licencia compatible AGPL. De tal manera, las empresas que deseen utilizar los jPOS comercialmente sin distribuir su código específico bajo una licencia de código abierto deben comprar una licencia comercial. Además, las empresas que desean acelerar su desarrollo basado en jPOS o que desean tener acceso al soporte pueden contratar asistencia y consultoría a los servicios de jPOS Consulting.

En su versión Java, jPOS [13] provee varios servicios que ahorran mucho trabajo al programador, entre estos se encuentra:

- Empaquetado y desempaquetado de los mensajes ISO-8583, personalización del empaquetado, para los diferentes tipos de implementaciones ISO-8583 a través de una clase Java denominada *ISOBasePackager* o un archivo XML.
- La clase Java *ISOChannel* se encarga del manejo completo del envío y recepción de mensajes ISO-8583.
- *ISOServer* es la clase Java encargada de manejar las conexiones entrantes, aceptarlas y pasar el control a una aplicación *ISOChannel* subyacente. Esta clase también maneja hilos para aceptar y procesar las transacciones.
- Manejo de una interfaz multiplexor de conexiones, permitiendo manipular más de una conexión por un socket de uso común.
- *SystemMonitor* es una clase que periódicamente registra información útil tales como el número de hilos en ejecución, el uso de memoria, etc.

[13] http://www.jpos.org
[14] http://www.gnu.org/licenses/agpl-3.0.html

A pesar de que la versión de Java de jPOS es más potente y conveniente para muchos usuarios, jPOS desarrolló una pequeña biblioteca en lenguaje C, que puede empacar y desempacar mensajes ISO-8583.

7.2 j8583

j8583[15] es una biblioteca de Java para manejar mensajes de ISO-8583. Esta biblioteca solo se encarga de empaquetar y desempaquetar los datos en una trama ISO-8583, por lo que no maneja el envío y recepción de mensajes a través de una conexión de red.

La biblioteca j8583 [14] contiene un conjunto de clases con funciones muy simple para poder manejar los mensajes ISO-8583. Las clases *MessageFactory* e *IsoMessage* son las dos clases principales que se utilizan para trabajar con mensajes ISO-8583. La clase *IsoMessage* permite crear el mensaje de manera manual, mientras que la clase *MessageFactory*, una vez configurada, puede crear mensajes de distintos tipos con algunos valores predefinidos, y también puede leer un arreglo de bytes y convertirlo en un mensaje ISO-8583. La clase *MessageFactory* puede leer un archivo XML para configurar sus plantillas de mensajes, encabezados ISO y plantillas de lectura.

7.3 Trx Framework

Trx Framwork[16] es un proyecto de código abierto en .NET que permite crear aplicaciones cliente/servidor para intercambiar mensajes financieros. Es compatible con muchos mecanismos de codificación/decodificación de los mensajes de campos fijos y variables, incluyendo ISO-8583, XML y el formato condicional. Es altamente personalizable y extensible. Trx Framework también proporciona una infraestructura de comunicación flexible de canal que soporta TCP/IP y otras tecnologías. Actualmente se distribuye bajo la Licencia Pública General Affero GNU, también se ofrece soporte comercial para la adaptación de Trx Framwork a los requerimientos del sistema que se desee implementar.

Trx Framwork fue escrito a principios de 2003 como base para la transferencia electrónica de fondos para la empresa Uruware[17]. Actualmente es utilizado por varios tipos de usuarios, que incluyen bancos, tiendas y centros comerciales, donde da soporte de crédito y débito, pago de saldo, recarga de saldo de teléfonos celulares prepago y firma de facturas electrónicas.

La primera versión de Trx Framwork fue liberado al dominio público en 2005 y está actualmente en uso por muchas organizaciones. La última versión de Trx Framework, versión 2, está disponible desde mediados de 2012. Ha sido renovado con nuevas características interesantes como SSL, empaquetado y

[15] http://j8583.sourceforge.net
[16] http://www.trxframework.net
[17] http://www.uruware.com

desempaquetado dinámico de servicios configurables con archivos XML, y manejo de seguridad a través de la ofuscación de los datos sensibles.

Los siguientes servicios que ofrece Trx Framework le permite al programador simplificar el trabajo:

- Crear y analizar los mensajes ISO-8583 o personalizar el formato del mensaje.
- Iniciar conexiones salientes a través de TCP/IP para enviar y recibir los mensajes.
- Implementar un servidor para aceptar conexiones entrantes.
- Generar un puente que transforma una conexión ISO-8583 a una llamada web service.

8. Marco de Desarrollo ISO-8583

En este capítulo se muestra la implementación del protocolo de transacciones financieras ISO-8583 [1], la cual consiste en tres fases (Análisis, Desarrollo y Pruebas). A continuación, se describe el proceso práctico que se siguió a lo largo del desarrollo del protocolo.

8.1 Análisis

Como parte del desarrollo fue necesario hacer un análisis general que permitiera recopilar los principales aspectos o requerimientos del protocolo ISO-8583, de manera de abordar el problema y poder cubrir con los objetivos definidos en el Capítulo 1.

A continuación, se muestra cómo se estructuró la lista de requerimientos del protocolo ISO-8583:

- Definir y crear una estructura de datos que permita almacenar la información proveniente del mensaje ISO-8583.
- Crear un método que permita la construcción del mensaje ISO-8583.
- Implementar métodos que permitan insertar y extraer el MTI del mensaje ISO-8583.
- Implementar métodos que permitan activar los campos de los mapas de bits.
- Implementar métodos que permitan insertar y extraer la información de los campos del mensaje.

8.2 Desarrollo del Protocolo ISO-8583

Para definir la estructura fue necesario tomar en cuenta el sistema Scolopendra, que es quien establece los campos y los tipos de mensajes que se van a utilizar en las transacciones que realizarán los diferentes terminales de ventas.

En primer lugar se definió una estructura de datos que facilitará el manejo, construcción y extracción de la data del mensaje ISO-8583. Para ello, y dada las limitaciones del repertorio de instrucciones que proveen los terminales de venta, se estableció el siguiente tipo de estructura denominada *iso8583* mostrado en la Figura 8.1, en la que se puede observar la declaración del MTI en la línea 9, los dos mapas de bits en las líneas 10-11 y luego la definición de los campos a ser utilizados en los mensajes de intercambio entre el sistema Scolopendra y los terminales de venta.

Algunos terminales de ventas tienen limitada la memoria RAM, por lo tanto crear una variable de 64 bytes diera un fallo de memoria, por eso se definieron dos mapas de bits con un tamaño de 8 bytes como se puede ver las líneas 10 y 11 de la Figura 8.1, para solventar esta limitante. Aprovechando que el tamaño máximo que puede tomar un mapa de bits es un múltiplo de 8 y dado que la memoria RAM es limitada, se definieron dos métodos (*setBit* y *testBit*) para manipular los dos

mapas de bits, que se encargan cambiar u obtener el valor de los indicadores de los campos que están activos en el mensaje ISO-8583.

```
 8:  typedef struct iso8583{
 9:   uint8 mti[4];
10:   uint8 bitmap[8];
11:   uint8 bitmap2[8];
12:   …
13://Campos del Mensaje ISO-8583
14:   …
```

Figura 8.1: Estructura ISO-8583

Para entender cómo funcionan los métodos *setBit* y *testBit* se puede guiar tomando en cuenta la Figura 8.1 la cual muestra cómo está definida la variable *bitmap*, en donde la variable se declara como una lista de 8 elementos de tipo entero de 1 bytes (8 bits), lo que nos da un total de 8*8=64 bits. La Figura 8.2 muestra el código fuente para activar o verificar un bit de dato en el mapa de bits.

```
 4:  void setBit(int bitPos, uint8 *bitmap){
 5:   if(!testBit(bitPos, bitmap))
 6:         bitmap[(bitPos-1) / 8] += 128 >> ((bitPos-1) % 8);
 7:  }
 8:
 9:  uint8 testBit(int bitPos, uint8 *bitmap){
10:   uint8 testNum = bitmap[(bitPos-1) / 8];
12:   return testNum & (128 >> ((bitPos-1) % 8));
13:  }
```

Figura 8.2: Código Fuente del setBit y testBit

En la figura la Figura 8.3, se puede ver un mapa de bit y la posición del bit 31.

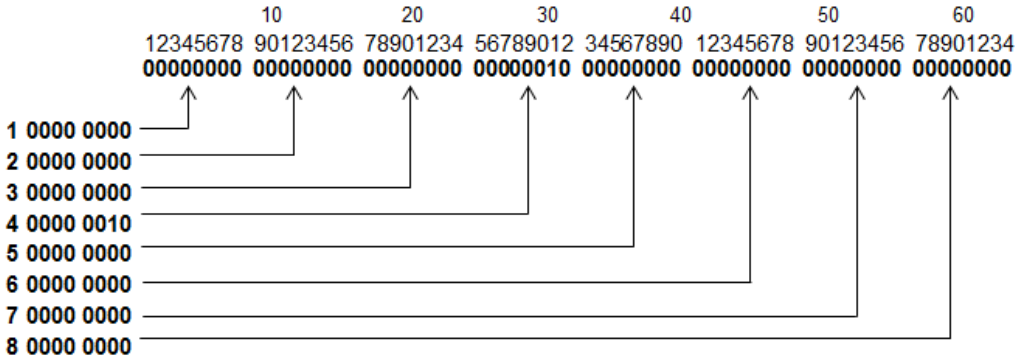

Figura 8.3: Ejemplo de Mapa de Bits

En segundo lugar se tuvo que definir los campos del mensaje, para ello solo se tomó en cuenta los campos con los que trabaja el sistema Scolopendra en las transacciones que tiene definida. Esta restricción permite utilizar eficientemente los recursos de los terminales de venta, y no desperdiciar espacio en memoria RAM y en disco declarando campos que no van a ser utilizados en ningún momento por un requerimiento de la aplicación.

En consecuencia los siguientes campos ISO-8583 2, 3, 4, 5, 7, 11, 32, 33, 34, 38, 39, 40, 44, 46, 104, 107, 108, 109, 111 son mostrados con sus respectivas especificaciones y características de formato para los mensajes de solicitud utilizados por Scolopendra en las siguientes tablas: Tabla 8.1, Tabla 8.2, Tabla 8.3, Tabla 8.4, Tabla 8.5, Tabla 8.6, Tabla 8.7, Tabla 8.8, Tabla 8.9, Tabla 8.10, Tabla 8.11, Tabla 8.12, Tabla 8.13, Tabla 8.14, Tabla 8.15, Tabla 8.16, Tabla 8.17, Tabla 8.18:

Campo 2	**Número de Contrato del Cliente**
Tamaño del campo: Tamaño de los datos:	3 bytes mínimo, 21 bytes máximo (LLVAR) 19 bytes máximo
Tipo de campo:	Numérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el número de contrato del cliente, precedido por 2 dígitos, que señalan el tamaño exacto del número de contrato. Ejemplo: Si un número de teléfono es 02121234567, entonces la data contenida en el campo seria “1102121234567”.

Tabla 8.1: Características del Campo 2

Campo 3	**Código del Proceso**
Tamaño del campo: Tamaño de los datos:	6 bytes Tamaño fijo
Tipo de campo:	Numérico, justificado con ceros a la derecha
Condición del campo:	Mandatorio
Descripción:	Este campo indica el servicio financiero solicitado. Los códigos utilizados para las recargas electrónicas son: • 001000: Inicio de sesión del sistema. • 040000: Sistema de Vendedores Independientes (SVI).

Tabla 8.2: Características del Campo 3

Campos 4	Monto
Tamaño del campo: Tamaño de los datos:	12 bytes Tamaño fijo
Tipo de campo:	Numérico, justificado con ceros a la derecha
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el monto de la transacción. Ejemplo: el monto a enviar es de 150,26 Bs, el valor a enviar es “000000015026”

Tabla 8.3: Características del Campo 4

Campos 5 y 6	Monto
Tamaño del campo: Tamaño de los datos:	12 bytes Tamaño fijo
Tipo de campo:	Numérico, justificado con ceros a la derecha
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el monto de la transacción. Ejemplo: el monto a enviar es de 500,26 Bs, el valor a enviar es “000000050026”

Figura 8.4: Características del Campo 5 y 6

Campo 7	Fecha y Hora
Tamaño del campo: Tamaño de los datos:	10 bytes Tamaño fijo
Tipo de campo:	Numérico, formato de la fecha MMDDhhmmss
Condición del campo:	Mandatorio
Descripción:	Este campo contiene la fecha y hora en la que se hizo la transmisión del mensaje. Este valor es utilizado mayormente para registrar el momento exacto en el que se realizó una recarga. Este campo debe contener un valor correcto, como lo define la siguiente cuadro: **Subcampo** **Definición** **Dígitos** **Rango** MM Mes 2 01-12 DD Día 2 01-31 hh Hora 2 00-23 mm Minuto 2 00-59 ss Segundo 2 00-59

Tabla 8.4: Características del Campo 7

Campo 11	Token de Seguridad
Tamaño del campo: Tamaño de los datos:	6 bytes Tamaño fijo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene un valor aleatorio el cual permite identificar si el usuario está conectado al sistema. El Token de seguridad se obtiene cada vez que un usuario inicia sesión en el sistema y este tiene un tiempo de duración definido por la compañía a la cual representa.

Tabla 8.5: Características del Campo 11

Campo 32	Identificador de la Compañía de Servicio
Tamaño del campo: Tamaño de los datos:	3 bytes mínimo, 13 bytes máximo (LLVAR) 11 bytes máximo
Tipo de campo:	Numérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el número que identifica a la compañía prestadora del servicio precedido por 2 dígitos que señalan el tamaño exacto del identificador de servicio. Ejemplo: Si el identificador de la compañía es 123456, entonces la data contenida en el campo seria “06123456”.

Tabla 8.6: Características del Campo 32

Campo 33	Identificador del Usuario
Tamaño del campo: Tamaño de los datos:	3 bytes mínimo, 13 bytes máximo (LLVAR) 11 bytes máximo
Tipo de campo:	Numérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el número que identifica al usuario del sistema, el cual debe estar previamente registrado. El identificador es precedido por 2 dígitos que señalan el tamaño exacto del identificador de usuario. Ejemplo: Si el identificador del usuario es 123, entonces la data contenida en el campo seria “03123”.

Tabla 8.7: Características del Campo 33

Campo 34	Identificador de la Compañía del Usuario
Tamaño del campo: Tamaño de los datos:	3 bytes mínimo, 30 bytes máximo (LLVAR) 28 bytes máximo
Tipo de campo:	Numérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el número que identifica a la compañía del usuario del sistema, el cual debe estar previamente registrado. El identificador es precedido por 2 dígitos que señalan el tamaño exacto del identificador de la compañía de usuario. Ejemplo: Si el identificador de la compañía del usuario es 123456, entonces la data contenida en el campo seria de la siguiente manera “06123456”.

Tabla 8.8: Características del Campo 34

Campo 38	Código de Registro
Tamaño del campo: Tamaño de los datos:	6 bytes Tamaño fijo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el valor que registra el sistema sobre la operación que se está efectuando. Es un valor aleatorio utilizado para validar la aprobación de la operación.

Tabla 8.9: Características del Campo 38

Campo 39	Código de Respuesta
Tamaño del campo: Tamaño de los datos:	3 bytes Tamaño fijo
Tipo de campo:	Numérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el valor de respuesta, que indica si el proceso fue ejecutado satisfactoriamente. En caso de falla, devuelve cual es el problema que impide la ejecución del proceso.

Tabla 8.10: Características del Campo 39

Campo 40	Código del Evento
Tamaño del campo: Tamaño de los datos:	3 bytes Tamaño fijo
Tipo de campo:	Numérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el número que indica el servicio financiero que se va a ejecutar.

Tabla 8.11: Características del Campo 40

Campo 44	Identificador del Terminal de Venta
Tamaño del campo: Tamaño de los datos:	3 bytes mínimo, 30 bytes máximo (LLVAR) 28 bytes máximo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el valor que identifica el terminal de venta de donde proviene la transacción. En la mayoría de los casos este identificador es el número de serial del dispositivo, en otros casos este número es asignado en la configuración del dispositivo. El identificador es precedido por 2 dígitos que señalan el tamaño exacto del identificador del terminal. Ejemplo: Si el identificador del terminal de venta es 12345678901, entonces la data contenida en el campo seria “1112345678901”.

Tabla 8.12: Características del Campo 44

Campo 46	Identificador del Banco
Tamaño del campo: Tamaño de los datos:	4 bytes mínimo, 207 bytes máximo (LLLVAR) 204 bytes máximo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el valor que identifica el banco al cual se le va a agregar el depósito. El identificador es precedido por 3 dígitos que señalan el tamaño exacto del identificador del banco. Ejemplo: Si el identificador del banco es 12, entonces la data contenida en el campo seria “00212”.

Tabla 8.13: Características del Campo 46

Campo 104	Número de Control del Recibo Bancario / Data de Inicio de Sesión
Tamaño del campo: Tamaño de los datos:	4 bytes mínimo, 100 bytes máximo (LLLVAR) 97 bytes máximo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el valor que identifica el número de control del depósito o los datos del usuario que inicia la sesión. El identificador es precedido por 3 dígitos que señalan el tamaño exacto del número de control del recibo bancario o los datos de inicio de sesión. Ejemplo: Si el identificador del voucher es 78945612379, entonces la data contenida en el campo seria “01178945612379”. Cuando envía data para el inicio de sesión este envía un xml que contiene el nombre del usuario y la contraseña **<DsUsers>** **<DtUser>** **<Login>**nombre de usuario**</Login>** **<Pwd>**XXXXX**</Pwd>** **</DtUser>** **</DsUsers>**.

Tabla 8.14: Características del Campo 104

Campo 107	Número Serial de la Tarjeta Electrónica
Tamaño del campo: Tamaño de los datos:	4 bytes mínimo, 999 bytes máximo (LLLVAR) 996 bytes máximo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el valor que identifica el serial de una tarjeta electrónica. Este valor es único dentro del sistema. El número es precedido por 3 dígitos que señalan el tamaño exacto del número serial. Ejemplo: Si el número de serial de la tarjeta electrónica es 1234567890123456, entonces la data contenida en el campo seria “0161234567890123456”.

Tabla 8.15: Características del Campo 107

<table>
<tr><th>Campo 108</th><th>Número de Identificación Personal (PIN)</th></tr>
<tr><td>Tamaño del campo:
Tamaño de los datos:</td><td>4 bytes mínimo, 999 bytes máximo (LLLVAR)
996 bytes máximo</td></tr>
<tr><td>Tipo de campo:</td><td>Alfanumérico</td></tr>
<tr><td>Condición del campo:</td><td>Mandatorio</td></tr>
<tr><td>Descripción:</td><td>Este campo contiene el valor del PIN de una tarjeta electrónica. Este valor es único dentro del sistema e identifica el monto una recarga electrónica.

En este caso el PIN puede variar tanto de tamaño como de valor dependiendo del servicio que se desee recargar, los PINs pueden tener una longitud entre 10 a 20 y pueden ser alfanuméricos o numéricos. El número es precedido por 3 dígitos que señalan el tamaño exacto del PIN.

Por ahora solo se tiene tres variaciones de PIN mostrados en el siguiente cuadro:

<table>
<tr><th>Tamaño</th><th>Tipo</th><th>Valor</th></tr>
<tr><td>20</td><td>Alfanumérico</td><td>V123 4567 8901 2345 6789</td></tr>
<tr><td>16</td><td>Numérico</td><td>1234 5678 9012 3456</td></tr>
<tr><td>10</td><td>Alfanumérico</td><td>8CHZ2S5QQI</td></tr>
</table>

Ejemplo: Si el número de serial de la tarjeta electrónica es 1234 5678 9012 3456, entonces la data contenida en el campo seria “0161234567890123456”.</td></tr>
</table>

Tabla 8.16: Características del Campo 108

Campo 109	Días de Programación
Tamaño del campo: Tamaño de los datos:	4 bytes mínimo, 999 bytes máximo (LLLVAR) 996 bytes máximo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el número de días que el cliente tiene disponible para ver televisión por cable. Sólo utilizado para los proveedores de televisión por cable. El número es precedido por 3 dígitos que señalan el tamaño exacto de los días de programación. Ejemplo: Si el número días de programación son 30, entonces la data contenida en el campo seria “00230”.

Tabla 8.17: Características del Campo 109

Campo 111	Mensaje de Respuesta
Tamaño del campo: Tamaño de los datos:	4 bytes mínimo, 999 bytes máximo (LLLVAR) 996 bytes máximo
Tipo de campo:	Alfanumérico
Condición del campo:	Mandatorio
Descripción:	Este campo contiene el mensaje de respuesta que corresponde al código de respuesta de un evento. Los valores que puede tener este campo son mayormente mensajes de referencia de un dato erróneo. El número es precedido por 3 dígitos que señalan el tamaño exacto del mensaje de respuesta. Ejemplo: Si el valor del mensaje es “Acceso Denegado - Usuario o Clave incorrectos” (46 bytes), entonces la data contenida en el campo seria “046Acceso Denegado - Usuario o Clave incorrectos”.

Tabla 8.18: Características del Campo 111

En tercer lugar se definieron dos métodos que facilitan el empaquetado y desempaquetado del mensaje ISO-8583, *isoToByteArray* y *parseIsoResponse*.

En la Figura 8.5, se puede observar como la función *isoToByteArray* recibe un dato del tipo iso8583, una cadena de caracteres y retorna el valor de tipo entero que representa el tamaño del mensaje ISO-8583. La función va recorriendo la estructura iso8583 y va empaquetando en una cadena de caracteres el MTI, los dos bit maps y los campos que fueron previamente activados.

```
205
206  uint16 isoToByteArray(iso8583 *isoIn, uint8 *output){
207      uint16 len = 0;
208
209      //set fields
210
211      output[0] = '\0';
212      strcat((char *)output,"000000");
213      memcpy(&output[6],isoIn->mti, 4);
214
215      memcpy(&output[10],isoIn->bitmap, 8);
216      memcpy(&output[18],isoIn->bitmap2, 8);
217      output[26] = '\0';
218
219      if(testBit(2,isoIn->bitmap)){
220          strcat((char *)&output[26],(char *)isoIn->field_2_len);
221          strcat((char *)&output[26],(char *)isoIn->field_2);
222      }
223
224      if(testBit(3,isoIn->bitmap)){
225          strcat((char *)&output[26],(char *)isoIn->field_3);
226      }
227
228      if(testBit(4,isoIn->bitmap)){
229          strcat((char *)&output[26],(char *)isoIn->field_4);
230      }
231
232      if(testBit(5,isoIn->bitmap)){
233          strcat((char *)&output[26],(char *)isoIn->field_5);
234      }
235
236      if(testBit(6,isoIn->bitmap)){
237          strcat((char *)&output[26],(char *)isoIn->field_6);
```

Figura 8.5: Código del Empaquetado del Mensaje ISO-8583

Por otra parte la función *parseIsoResponse* se encarga de desempaquetar los mensajes ISO-8583, como se puede ver en la Figura 8.6. La función recibe dos parámetros que corresponden a la estructura iso8583 y una cadena de caracteres, en donde la cadena contiene el mensaje recibido por los terminales de venta, y a su vez, es pasada como parámetro a esta función de manera que pueda ser recorrida para ir extrayendo la data contenida en el mensaje.

```
14
15  void parseIsoResponse(uint8 *input, iso8583 *isoOut){
16      int index = 26;
17      int fieldLen = 0;
18
19      memcpy(isoOut->mti, &input[6], 4);
20      isoOut->mti[4] = '\0';
21      memcpy(isoOut->bitmap, &input[10], 8);
22      isoOut->bitmap[8] = '\0';
23      memcpy(isoOut->bitmap2, &input[18], 8);
24      isoOut->bitmap2[8] = '\0';
25
26      if(testBit(2,isoOut->bitmap)){
27          memcpy(isoOut->field_2_len, &input[index], 2);
28          isoOut->field_2_len[2] = '\0';
29          index += 2;
30          fieldLen = atoi((char *)isoOut->field_2_len);
31          memcpy(isoOut->field_2, &input[index], fieldLen);
32          index += fieldLen;
33          isoOut->field_2[fieldLen] = '\0';
34      }
35
36      if(testBit(3,isoOut->bitmap)){
37          memcpy(isoOut->field_3, &input[index], 6);
38          isoOut->field_3[6] = '\0';
39          index += 6;
40      }
41
42      if(testBit(4,isoOut->bitmap)){
43          memcpy(isoOut->field_4, &input[index], 12);
44          isoOut->field_4[12] = '\0';
45          index += 12;
46      }
```

Figura 8.6: Código del Desempaquetado del Mensaje ISO-8583

Por otra parte y dado que los mensajes enviados por los terminales de venta están estandarizados se definió una biblioteca denominada *CryptoManager* la cual contiene un conjunto de métodos que facilitan la construcción de los mensajes ISO-8583.

Los métodos definidos son los siguientes:

- GetSecTokenIso8583
- RegisterPaymentIso8583
- AddECardIso8583
- ConfirmOperationIso8583
- AddECardNoTopupIso8583
- CloseOutEposIso8583
- GetBalanceIso8583
- CloseOutConfirmEposIso8583

El método *GetSecTokenIso8583* es utilizado para generar un mensaje de inicio de sesión en el sistema Scolopendra. La función solo necesita recibir tres parámetros, entre ellos se encuentra el mensaje *iso8583*, el nombre del usuario registrado en el sistema Scolopendra y la contraseña del mismo. Por otra parte este método se encarga de colocar en el MTI el valor “0100” que indica que es un mensaje de

requerimiento, activar en los mapas de bits los campos 3, 40, 104 y finalmente asignar los valores en dichos campos.

```
440
441  void GetSecTokenIso8583(iso8583 *isoOut, const char *login, const char *password){
442      size_t len;
443      int i;
444      for(i = 0; i < 8; i++){
445          isoOut->bitmap[i]=0;
446          isoOut->bitmap2[i]=0;
447      }
448      isoOut->bitmap[8] = '\0';
449      isoOut->bitmap2[8] = '\0';
450
451      setBit(1,isoOut->bitmap);
452
453      strcpy((char *)isoOut->mti,"0100");
454
455      setBit(3,isoOut->bitmap);
456      strcpy((char *)isoOut->field_3,"001000");
457
458      setBit(40,isoOut->bitmap);
459      strcpy((char *)isoOut->field_40,"004");
460
461      setBit(104-64,isoOut->bitmap2);
462      len = 62 + strlen(login) + strlen(password);
463      sprintf((char *)isoOut->field_104_len,"%03d",len);
464      strcpy((char *)isoOut->field_104,"<DsUsers><DtUser><Login>");
465      strcat((char *)isoOut->field_104,login);
466      strcat((char *)isoOut->field_104,"</Login><Pwd>");
467      strcat((char *)isoOut->field_104,password);
468      strcat((char *)isoOut->field_104,"</Pwd></DtUser></DsUsers>");
469
```

Figura 8.7: Código del Método GetSecTokenIso8583

El método *RegisterPaymentIso8583* se utiliza para construir un mensaje que permita al sistema Scolopendra registrar un depósito. Como se puede observar en la Figura 8.8, el método recibe ocho parámetros de los cuales se encuentran el mensaje *iso8583*, los identificadores de la compañía del usuario y del usuario, el token de seguridad, la fecha en la que se realizó el depósito, el número de control del depósito y el monto del mismo. En resumen este método se encarga de colocar el MTI “0100” indicando que es un mensaje de requerimiento, activa los campos de los mapas de bits 3, 4, 7, 11, 33, 34, 40, 44, 46, 104 y asigna los datos correspondientes a cada uno de estos campos.

```
471
472
473  void RegisterPaymentIso8583(iso8583 *isoOut, const char* userCompany, const char* user, const char* securityToken, c
474  char* depositDate, const char* depositControlNumber, const char* depositAmount, const char * serialsToConfirm){
475      size_t len;
476      char lenStr[5];
477      int i;
478      for(i = 0; i < 8; i++){
479          isoOut->bitmap[i]=0;
480          isoOut->bitmap2[i]=0;
481      }
482      isoOut->bitmap[8] = '\0';
483      isoOut->bitmap2[8] = '\0';
484
485      setBit(1,isoOut->bitmap);
486
487      strcpy((char *)isoOut->mti,"0100");
488
489      setBit(3,isoOut->bitmap);
490      strcpy((char *)isoOut->field_3,"040000");
491
492      setBit(4,isoOut->bitmap);
493      strcpy((char *)isoOut->field_4,depositAmount);
494
495      setBit(7,isoOut->bitmap);
496      strcpy((char *)isoOut->field_7,depositDate);
497
498      setBit(11,isoOut->bitmap);
499      strcpy((char *)isoOut->field_11,securityToken);
```

Figura 8.8: Código del Método RegisterPaymentIso8583

El método *AddECardIso8583* es utilizado para generar un mensaje al sistema Scolopendra indicando que debe generar un PIN. En la Figura 8.9 se puede ver que este método recibe nueve parámetros en donde se encuentran el mensaje *iso8583*, el identificador de la compañía de servicio, la compañía del usuario y el usuario, el número de contrato, el monto del PIN, el identificador del terminal de venta y la hora en la que se emitió el mensaje de creación del PIN. En este método se define un MTI con el valor de "0100" indicando que es un mensaje de requerimiento, se activan los campos 3, 4, 7, 11, 32, 33, 34, 40, 44, 111 y se le asigna los datos correspondientes a los campos mencionados.

```
551
552  void AddECardIso8583(iso8583 *isoOut, const char* serviceCompany, const char* userCompany, const char* user, const cl
553  accountNumber, const char* amount, const char* posId,  const char* serviceTime, const char * serialsToConfirm){
554      size_t len;
555      char lenStr[5];
556      int i;
557      for(i = 0; i < 8; i++){
558          isoOut->bitmap[i]=0;
559          isoOut->bitmap2[i]=0;
560      }
561      isoOut->bitmap[8] = '\0';
562      isoOut->bitmap2[8] = '\0';
563
564      setBit(1,isoOut->bitmap);
565
566      strcpy((char *)isoOut->mti,"0100");
567
568      setBit(3,isoOut->bitmap);
569      strcpy((char *)isoOut->field_3,"040000");
570
571      setBit(4,isoOut->bitmap);
572      strcpy((char *)isoOut->field_4,amount);
573
574      setBit(7,isoOut->bitmap);
575      strcpy((char *)isoOut->field_7,serviceTime);
576
577      setBit(11,isoOut->bitmap);
578      strcpy((char *)isoOut->field_11,securityToken);
579
580      setBit(32,isoOut->bitmap);
581      len = strlen(serviceCompany);
582      sprintf(lenStr,"%02d",len);
583      strcpy((char *)isoOut->field_32_len,lenStr);
584      strcpy((char *)isoOut->field_32,serviceCompany);
585
```

Figura 8.9: Código del Método AddECardIso8583

El método *ConfirmOperationIso8583* es utilizado para construir un mensaje que le permita al sistema Scolopendra realizar la confirmación de la creación de un PIN. En la Figura 8.11 se puede ver que recibe diez parámetros en donde se encuentran el mensaje *iso8583*, el identificador de la compañía de servicio, la compañía del usuario y el usuario, el número de contrato, el monto del PIN, el identificador el terminal de venta, la hora en la que se emitió el mensaje de confirmación de la creación del PIN y el valor que identifica el registro de la operación de creación del PIN. En este método se define un MTI con el valor de "0100" indicando que es un mensaje de requerimiento, se activan los campos 3, 4, 7, 11, 32, 33, 34, 38, 40, 44, 111 y se le asigna los datos correspondientes a los campos mencionados.

Esta confirmación se debe a que el Internet no es confiable y las tramas pueden no llegar a su destino, originando una inconsistencia en el sistema. Para evitar dicha inconsistencia en el sistema, se determinó una forma que permitiera evitarla o en su defecto corregirla, por lo que la confirmación de un PIN forma parte del proceso de creación del PIN.

El proceso de creación de PIN está definido en 4 pasos como se puede ver en la Figura 8.10:

1. Se envía un mensaje de creación de PIN al sistema Scolopendra.
2. Scolopendra envía un mensaje de recibido y el código de registro de la transacción.
3. El terminal reenvía a Scolopendra los datos de la creación del PIN y el código de registro de la transacción.
4. Scolopendra envía un mensaje de recibido.

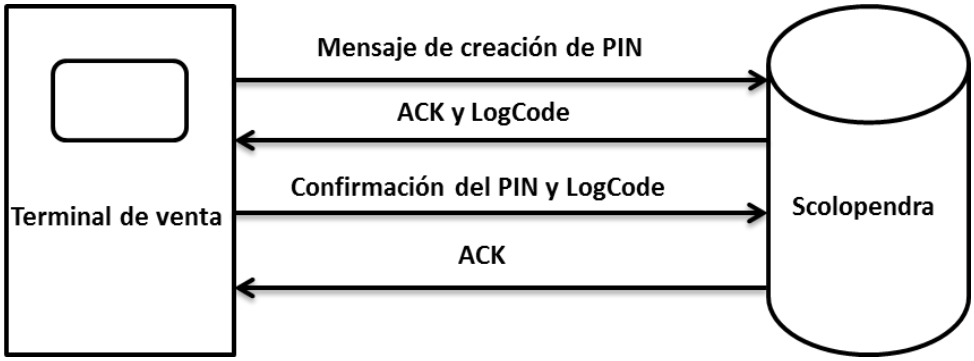

Figura 8.10: Proceso de Creación de PIN

```
367
368  void ConfirmOperationIso8583(iso8583 *isoOut, const char* serviceCompany, const char* userCompany, const char* user,
369  accountNumber, const char* amount, const char* posId,  const char* serviceTime, uint8 is_serial, const char * serial
370      size_t len;
371      char lenStr[5];
372      int i;
373      for(i = 0; i < 8; i++){
374          isoOut->bitmap[i]=0;
375          isoOut->bitmap2[i]=0;
376      }
377      isoOut->bitmap[8] = '\0';
378      isoOut->bitmap2[8] = '\0';
379
380      setBit(1,isoOut->bitmap);
381
382      strcpy((char *)isoOut->mti,"0100");
383
384      setBit(3,isoOut->bitmap);
385      strcpy((char *)isoOut->field_3,"040000");
386
387      setBit(4,isoOut->bitmap);
388      strcpy((char *)isoOut->field_4,amount);
389
390      setBit(7,isoOut->bitmap);
391      strcpy((char *)isoOut->field_7,serviceTime);
392
393      setBit(11,isoOut->bitmap);
394      strcpy((char *)isoOut->field_11,securityToken);
395
396      setBit(32,isoOut->bitmap);
397      len = strlen(serviceCompany);
398      sprintf(lenStr,"%02d",len);
399      strcpy((char *)isoOut->field_32_len,lenStr);
400      strcpy((char *)isoOut->field_32,serviceCompany);
401
402      setBit(33,isoOut->bitmap);
403      len = strlen(user);
```

Figura 8.11: Código del Método ConfirmOperationIso8583

El método *AddECardNoTopupIso8583* al igual que *AddECardIso8583* es utilizado para generar un mensaje que indica al sistema Scolopendra que debe crear un PIN. La diferencia entre ambos métodos es que generan dos tipos de PIN diferentes. *AddECardIso8583* crea un PIN numérico y *AddECardNoTopupIso8583* un pin alfanumérico. La Figura 8.12 muestra los parámetros que recibe este método, los cuales son, el mensaje *iso8583*, el identificador de la compañía de servicio, la compañía del usuario y el usuario, el número de contrato, el monto del PIN, el identificador el terminal de venta, la hora en la que se emitió el mensaje de confirmación de la creación del PIN. En este método se define un MTI con el valor de “0100” indicando que es un mensaje de requerimiento, se activan los campos 3, 4, 7, 11, 32, 33, 34, 40, 44, 111 y se le asignan los datos a los campos activados.

```
694
695   void AddECardNoTopupIso8583(iso8583 *isoOut, const char* serviceCompany, const char* userCompany, const char* user,
696   accountNumber, const char* amount, const char* posId,  const char* serviceTime, const char * serialsToConfirm){
697       size_t len;
698       char lenStr[5];
699       int i;
700       for(i = 0; i < 8; i++){
701           isoOut->bitmap[i]=0;
702           isoOut->bitmap2[i]=0;
703       }
704       isoOut->bitmap[8] = '\0';
705       isoOut->bitmap2[8] = '\0';
706
707       setBit(1,isoOut->bitmap);
708
709       strcpy((char *)isoOut->mti,"0100");
710
711       setBit(3,isoOut->bitmap);
712       strcpy((char *)isoOut->field_3,"040000");
713
714       setBit(4,isoOut->bitmap);
715       strcpy((char *)isoOut->field_4,amount);
716
717       setBit(7,isoOut->bitmap);
718       strcpy((char *)isoOut->field_7,serviceTime);
719
720       setBit(11,isoOut->bitmap);
721       strcpy((char *)isoOut->field_11,securityToken);
722
723       setBit(32,isoOut->bitmap);
724       len = strlen(serviceCompany);
725       sprintf(lenStr,"%02d",len);
726       strcpy((char *)isoOut->field_32_len,lenStr);
727       strcpy((char *)isoOut->field_32,serviceCompany);
728
729       setBit(33,isoOut->bitmap);
730       len = strlen(user);
731       sprintf(lenStr,"%02d",len);
732       strcpy((char *)isoOut->field_33_len,lenStr);
733       strcpy((char *)isoOut->field_33,user);
```

Figura 8.12: Código del Método AddECardNoTopupIso8583

El método *CloseOutEposIso8583* es utilizado para generar un mensaje al sistema Scolopendra indicando que debe realizar el cuadre del terminal de venta con el sistema. En la Figura 8.13, se observa que recibe ocho parámetros en los cuales se encuentran el mensaje *iso8583*, el identificador de la compañía del usuario y el usuario, el token de seguridad, el identificador del terminal de venta, la fecha de cuadre, el total de transacciones y el monto de dichas transacciones. En este método se define un MTI con el valor de “0100” indicando que es un mensaje de requerimiento, se activan los campos 2, 3, 4, 7, 11, 33, 34, 40, 44, 111 y se le asignan los datos de los campos activados.

```
874
875  void CloseOutEposIso8583(iso8583 *isoOut, const char* userCompany, const char* user, const char* securityToken, cons
876  const char* closeOutDate, const char* totalTxn, const char* totalAmount, const char * serialsToConfirm){
877      size_t len;
878      char lenStr[5];
879      int i;
880      for(i = 0; i < 8; i++){
881          isoOut->bitmap[i]=0;
882          isoOut->bitmap2[i]=0;
883      }
884      isoOut->bitmap[8] = '\0';
885      isoOut->bitmap2[8] = '\0';
886
887      strcpy((char *)isoOut->mti,"0100");
888
889      setBit(1,isoOut->bitmap); //bug cuando no hay bm2 zzzzz
890
891      setBit(2,isoOut->bitmap);
892      len = strlen(totalTxn);
893      sprintf(lenStr,"%02d",len);
894      strcpy((char *)isoOut->field_2_len,lenStr);
895      strcpy((char *)isoOut->field_2,totalTxn);
896
897      setBit(3,isoOut->bitmap);
898      strcpy((char *)isoOut->field_3,"040000");
899
900      setBit(4,isoOut->bitmap);
901      strcpy((char *)isoOut->field_4,totalAmount);
902
903      setBit(7,isoOut->bitmap);
904      strcpy((char *)isoOut->field_7,closeOutDate);
905
```

Figura 8.13: Código del Método CloseOutEposIso8583

El método *GetBalanceIso8583* es un método utilizado para generar un mensaje ISO-8583 que devuelva el saldo que tiene un vendedor en el sistema Scolopendra. La Figura 8.14 muestra que este método recibe cinco parámetros en los cuales se encuentran el mensaje *iso8583*, el identificador de la compañía del usuario y el usuario, el token de seguridad y el identificador del terminal de venta. En este método se define un MTI con el valor de “0100” indicando que es un mensaje de requerimiento, se activan los campos 3, 11, 33, 34, 40, 44, 111 y se le asignan los datos de los campos activados.

```
765
766  void GetBalanceIso8583(iso8583 *isoOut, const char* userCompany, const char* user, const char* securityToken, const
767      size_t len;
768      char lenStr[5];
769      int i;
770      for(i = 0; i < 8; i++){
771          isoOut->bitmap[i]=0;
772          isoOut->bitmap2[i]=0;
773      }
774      isoOut->bitmap[8] = '\0';
775      isoOut->bitmap2[8] = '\0';
776
777      strcpy((char *)isoOut->mti,"0100");
778
779      setBit(1,isoOut->bitmap); //bug cuando no hay bm2 zzzzz
780
781      setBit(3,isoOut->bitmap);
782      strcpy((char *)isoOut->field_3,"040000");
783
784      setBit(11,isoOut->bitmap);
785      strcpy((char *)isoOut->field_11,securityToken);
786
787      setBit(33,isoOut->bitmap);
788      len = strlen(user);
789      sprintf(lenStr,"%02d",len);
790      strcpy((char *)isoOut->field_33_len,lenStr);
791      strcpy((char *)isoOut->field_33,user);
792
793      setBit(34,isoOut->bitmap);
794      len = strlen(userCompany);
795      sprintf(lenStr,"%02d",len);
796      strcpy((char *)isoOut->field_34_len,lenStr);
797      strcpy((char *)isoOut->field_34,userCompany);
798
799      setBit(40,isoOut->bitmap);
800      strcpy((char *)isoOut->field_40,"004");
801
802      setBit(44,isoOut->bitmap);
```

Figura 8.14: Código del Método GetBalanceIso8583

El método *CloseOutConfirmEposIso8583* es utilizado para generar un mensaje de confirmación del cierre en el sistema Scolopendra. Este mensaje sirve como indicador de que el terminal recibió la respuesta del sistema. La Figura 8.15 muestra que este método recibe parámetros en los cuales se encuentran el mensaje *iso8583*, el identificador de la compañía del usuario y el usuario, el token de seguridad y el identificador del terminal de venta. En este método se define un MTI con el valor de “0100” indicando que es un mensaje de requerimiento, se activan los campos 3, 11, 33, 34, 40, 44, 111 y se le asignan los datos de los campos activados

```
946
947  void CloseOutConfirmEposIso8583(iso8583 * isoOut, char * userCompany, char * user, char * securityToken, char * posI
948      size_t len;
949      char lenStr[5];
950      int i;
951      for(i = 0; i < 8; i++){
952          isoOut->bitmap[i]=0;
953          isoOut->bitmap2[i]=0;
954      }
955      isoOut->bitmap[8] = '\0';
956      isoOut->bitmap2[8] = '\0';
957
958      strcpy((char *)isoOut->mti,"0100");
959
960      setBit(1,isoOut->bitmap);
961
962      setBit(3,isoOut->bitmap);
963      strcpy((char *)isoOut->field_3,"040000");
964
965      setBit(11,isoOut->bitmap);
966      strcpy((char *)isoOut->field_11,securityToken);
967
968      setBit(33,isoOut->bitmap);
969      len = strlen(user);
970      sprintf(lenStr,"%02d",len);
971      strcpy((char *)isoOut->field_33_len,lenStr);
972      strcpy((char *)isoOut->field_33,user);
973
974      setBit(34,isoOut->bitmap);
975      len = strlen(userCompany);
976      sprintf(lenStr,"%02d",len);
977      strcpy((char *)isoOut->field_34_len,lenStr);
978      strcpy((char *)isoOut->field_34,userCompany);
979
980      setBit(40,isoOut->bitmap);
981      strcpy((char *)isoOut->field_40,"044");
982
```

Figura 8.15: Código del Método CloseOutConfirmEposIso8583

Al igual que la creación de PIN, el cierre de un terminal de venta sufre el mismo problema y es que el Internet no es confiable, por tal razón existe la confirmación del cierre. El proceso de cierre consta de 4 pasos que son descritos a continuación:

1. Se envía un mensaje de cierre de terminal de venta al sistema Scolopendra.
2. Scolpendra envía un mensaje de recibido y el código de registro de la transacción.
3. El terminal efectúa un proceso de limpiado de transacciones y reenvía a Scolopendra un mensaje indicando que se realizó dicho proceso de limpiado de transacciones.
4. Scolopendra envía un mensaje de recibido.

8.3 Pruebas Realizadas al Protocolo ISO-8583

Las pruebas se desarrollaron enfocadas en el correcto funcionamiento del empaquetado y desempaquetado de la data enviada/recibida en los mensajes ISO-8583. En las pruebas a realizar se verifica que el mensaje ISO-8583 enviado por el terminal de venta, se reciba de manera correcta en el sistema Scolopendra.

En la realización de las pruebas se utilizó una estructura que tiene Scolopendra que facilita la depuración de los mensajes ISO-8583 y los nuevos requerimientos.

Los tres tipos de mensajes utilizados para las pruebas son los siguientes:

- Mensaje de inicio de sesión.
- Recarga Electrónica.
- Registrar un depósito en el sistema.

8.3.1 **Mensaje de Inicio de Sesión**

La primera prueba realizada está basada en el mensaje de inicio de sesión, se puede ver en la Figura 8.16 que el mensaje enviado desde el terminal de venta es el siguiente **"0100¢ 00100007291541230710760100000071<DsUsers><DtUser><Login>prueba</Login><Pwd>xxx</Pwd></DtUser></DsUsers>"**, en donde el MTI es 0100 indicando que es un mensaje de requerimiento emitido por un comprador, los mapas de bits uno y dos tienen los campos 3, 40 y 104 activados, los cuales son descritos en la Tabla 8.19.

```
Request Handler - Received: 0100¢ 00100007291541230710760100000071<DsUse
rs><DtUser><Login>prueba</Login><Pwd>xxx</Pwd></DtUser></DsUsers>

[MTI]0100[BM1]10100000000000000000000000000000000001000000000000000000
00000[BM2]00000000000000000000000000000000000001000000000000000000000
0[DATA]001000004068<DsUsers><DtUser><Login>prueba</Login><Pwd>xxx</Pwd></
DtUser></DsUsers>

Command Interpreter - The received ISO is:
ISO8583 -
mti = 0100
bitmap 1:
0        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234
10100000000000000000000000000000000001000000000000000000000000000000000000
bitmap 2:
6    7         8         9         10        11        12
56789012345678901234567890123456789012345678901234567890123456789012345678
00000000000000000000000000000000000001000000000000000000000000000000000000
field [3] = 001000
field [40] = 004
field [104] = <DsUsers><DtUser><Login>prueba</Login><Pwd>123</Pwd>
</DtUser></DsUsers>
```

Figura 8.16: Prueba de Inicio de Sesión

Campo	Valor	Descripción
3	001000	Significa que el proceso invocado es el inicio de sesión
40	004	Servicio a ser procesado por el sistema Scolopendra
104	<DsUsers><DtUser><Login>prueba</Login><Pwd>XXX</Pwd></DtUser></DsUsers>	XML con el nombre del usuario y la clave

Tabla 8.19: Prueba de Inicio de Sesión

8.3.2 **Recarga Electrónica**

Esta prueba está basada en el mensaje ISO-8583 para realizar una recarga electrónica, en este caso se puede ver en la Figura 8.17 el mensaje enviado por el terminal de venta: **"0100²Å040000000000002000080114543919175503219035490139036550220790123451627014117427336300110416000000"**, en donde el MTI es 0100 indicando que es un mensaje de requerimiento emitido por un comprador, los mapas de bits uno y dos tienen los campos 3, 4, 7, 11, 32, 33, 34, 38, 40, 44 y 111 activos, los cuales son descritos en la Tabla 8.20.

```
Request Handler - Received: 0100²Å040000000000002000080114543919175503219
035490139036550220790123451627014117427336300110416000000

[MTI]0100[BM1]1011001000100000000000000000001110001010001100000000000000
00000[BM2]0000000000000000000000000000000000000000000000010000000000000000
0[DATA]040000000000002000080114543919175503219035490139036550220790123451
627014117427336300110416000000
Command Interpreter - The received ISO is:
ISO8583 -
mti = 0100
bitmap 1:
0        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234
10110010001000000000000000000011100010100011000000000000000000000
bitmap 2:
6    7         8         9         10        11        12
56789012345678901234567890123456789012345678901234567890123456789012345678
00000000000000000000000000000000000000000000000010000000000000000
field [3] = 040000
field [4] = 000000002000
field [7] = 0801145439
field [11] = 191755
field [32] = 219
field [33] = 549
field [34] = 3
field [38] = 903655
field [40] = 022
field [44] = 9012345
field [111] = 04160000000
```

Figura 8.17: Prueba de Recarga Electrónica

Campo	Valor	Descripción
3	040000	Significa que se trata de un proceso de SVI
4	000000002000	Monto de la recarga electrónica con un valor de (20,00)
7	0801145439	Fecha en la que se está enviando el mensaje (01 de agosto a las 2:54:39 PM)
11	191755	Valor del token de seguridad dado por el sistema Scolopendra
32	219	Identificador de la compañía de servicio
33	549	Identificador del usuario
34	3	Identificador de la compañía
38	903655	Contiene el código de registro
40	022	Servicio de recarga electrónica Scolopendra
44	9012345	Identificador del terminal de venta
111	0416000000	Número de contrato del cliente

Tabla 8.20: Prueba de Recarga Electrónica

8.3.3 Registrar un Depósito en el Sistema

La prueba de registro de depósito es un mensaje ISO-8583 que envía el terminal de venta al sistema Scolopendra, para agregarlo al sistema. En la Figura 8.18 se puede ver que la trama enviada es: **"0100²Á040000000000250000201407050068153003549013010079012345013001000110091234567 89"**, en donde el MTI es 0100 indicando que es un mensaje de requerimiento emitido por un comprador, los mapas de bits uno y dos tienen los campos 3, 4, 7, 11, 33, 34, 40, 44, 46 y 104 activos, los cuales son descritos en la Tabla 8.21.

Campo	Valor	Descripción
3	040000	Significa que el proceso de SVI
4	000000250000	Monto de la recarga electrónica con un valor de (2500,00)
7	0801153602	Fecha en la que se está enviando el mensaje (01 de agosto a las 3:36:02 PM)
11	681530	Valor del token de seguridad dado por el sistema Scolopendra
33	549	Identificador del usuario
34	3	Identificador de la compañía
40	010	Servicio de Registro de depósito en el sistema Scolopendra
44	9012345	Identificador del terminal de venta
46	0	Identificador del banco del depósito
104	123456789	Contiene el número de control del depósito

Tabla 8.21: Prueba de Registro de Depósito en el Sistema

```
Request Handler - Received: 0100²Á040000000000250000201407050068153003549013010079012345013001
00011009123456789

[MTI]0100[BM1]1011001000100000000000000000000011000001000111010000000000
0000000[BM2]0000000000000000000000000000000000000000100000000000000000000
00000[DATA]040000000000250000201407050068153003549013010079012345013001
00011009123456789

Command Interpreter - The received ISO is:
ISO8583 -
mti = 0100
bitmap 1:
0        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234
10110010001000000000000000000001100000100011101000000000000000000
bitmap 2:
6    7         8         9         10        11        12
56789012345678901234567890123456789012345678901234567890123456789012345678
00000000000000000000000000000000000000001000000000000000000000000000000000
field [3] = 040000
field [4] = 000000250000
field [7] = 2014070500
field [11] = 681530
field [33] = 549
field [34] = 3
field [40] = 010
field [44] = 9012345
field [46] = 0
field [104] = 123456789
```

Figura 8.18: Prueba de Registro de Depósito en el Sistema

9. Aplicación en el Terminal de Venta

En este capítulo se muestra el diseño de la solución que fue implementada sobre el marco de trabajo ISO-8583 para pagos electrónicos [5] descrito en el Capítulo 4. La solución contiene los eventos utilizados para el Sistema de Vendedores Independientes (SVI) de recargas electrónicas.

9.1 Análisis General

Antes de iniciar el desarrollo de los eventos, se llevó a cabo una fase de análisis general donde se determinó de manera global los principales requerimientos para lograr la construcción de las funcionalidades que debe cumplir la aplicación. Estos requerimientos deben cubrir los objetivos estipulados en el Capítulo 1 para lograr construir la solución a la problemática planteada.

A continuación, se muestra cómo se estructuró la lista de requerimientos:

- Iniciar sesión.
- Consulta de saldo.
- Agregar depósito.
- Recarga electrónica.
- Pago de factura.
- Última transacción.
- Cierre de POS.
- Cerrar sesión.
- Actualización.

La Figura 9.1 presenta el diagrama de casos de uso que refleja la interacción del usuario con el sistema.

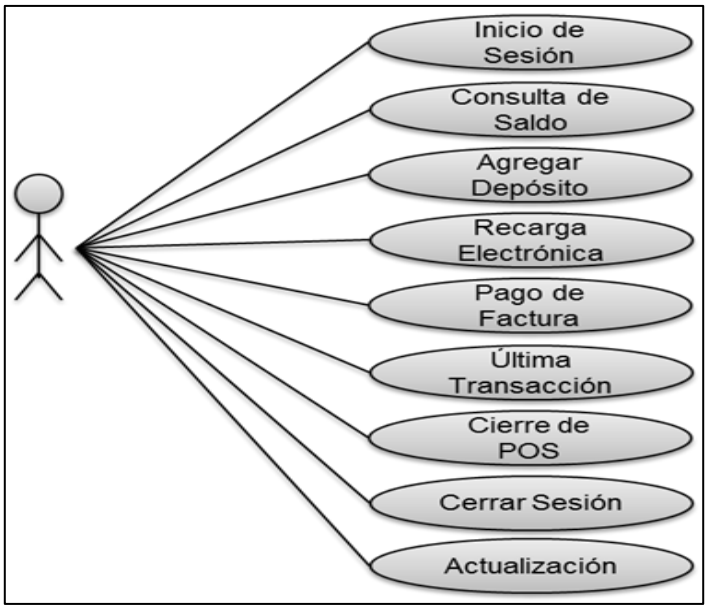

Figura 9.1: Diagrama del Caso de Uso Nivel 0

La especificación de cada caso de uso se muestra en las tablas: Tabla 9.1, Tabla 9.2, Tabla 9.3, Tabla 9.4, Tabla 9.5, Tabla 9.6, Tabla 9.7, Tabla 9.8, Tabla 9.9.

Caso de uso
1. Inicio de Sesión.
Actores
✓ Usuario.
Descripción
✓ Función que corresponde al inicio de las operaciones donde el cliente deberá ingresar el nombre del usuario y la clave de acceso. Para efectos prácticos el terminal de venta permanecerá activo por un periodo de tiempo durante el cual el cliente podrá hacer uso del mismo sin tener que ingresar nuevamente los datos.
Flujo básico
✓ El usuario accede a la opción “inicio de sesión” del menú principal del terminal de venta. ✓ Introduce su nombre de usuario del sistema y su clave. ✓ Se le muestra en pantalla un mensaje de bienvenida al usuario y el saldo disponible. ✓ Se despliega el menú de opciones del usuario.

Tabla 9.1: Especificación de Caso de Uso 1 - Inicio de Sesión

Caso de uso
2. Consulta de Saldo.
Actores
✓ Usuario.
Descripción
✓ Función mediante el cual el usuario autorizado solicita el saldo que tiene disponible para poder realizar ventas de recargas.
Flujo básico
✓ El usuario selecciona la opción “Consulta de Saldo”. ✓ Se le muestra un mensaje con dos montos, uno del saldo disponible y otro con el saldo por aprobar por GRE5.

Tabla 9.2: Especificación de Caso de Uso 2 - Consulta de Saldo

Caso de uso
3. Agregar Depósito.
Actores
✓ Usuario.
Descripción
✓ Función que sirve para aumentar el saldo de la billetera electrónica del usuario para poder realizar venta de recargas.

Flujo básico
✓ El usuario selecciona la opción “Agregar Deposito”. ✓ Selecciona el banco del depósito. ✓ Introduce el monto depositado. ✓ Introduce el número de control del depósito. ✓ Muestra un mensaje de verificación con los valores introducidos. ✓ Muestra un mensaje de éxito o error del registro. ✓ Si el registro es exitoso muestra la opción de imprimir un comprobante del depósito registrado en el sistema.

Tabla 9.3: Especificación de Caso de Uso 3 - Agregar Depósito

Caso de uso
4. Recarga Electrónica.
Actores
✓ Usuario.
Descripción
✓ Función para realizar operaciones de recarga o pagos de servicios de prepago.
Flujo básico
✓ El usuario selecciona la opción “Recarga Electrónica”. ✓ Selecciona la opción de recarga. ✓ Según el servicio se introduce el número de contrato. ✓ Introduce o selecciona un monto. ✓ Muestra un mensaje de verificación con los valores introducidos. ✓ Muestra un mensaje de éxito o error del registro. ✓ Si la recarga es exitosa, imprime un comprobante para el cliente y para el comercio.

Tabla 9.4: Especificación de Caso de Uso 4 - Recarga Electrónica

Caso de uso
5. Pago de Factura.
Actores
✓ Usuario.
Descripción
✓ Función habilitada solo para terminales de venta que requieran realizar pagos de facturas pendientes.
Flujo básico
✓ El usuario selecciona la opción “Pago de Factura”. ✓ Selecciona la opción de pago. ✓ Introduce el monto ✓ Introduce el número de factura. ✓ Muestra un mensaje de verificación con los valores introducidos. ✓ Muestra un mensaje de éxito o error del registro. ✓ Si el pago es exitoso, imprime un comprobante para el cliente y el comercio.

Tabla 9.5: Especificación de Caso de Uso 5 - Pago de Factura

Caso de uso
6. Última Transacción.
Actores
✓ Usuario.
Descripción
✓ La Función de “Última Transacción” se utiliza como un informe para que el usuario verifique cual fue la última recarga vendida con éxito.
Flujo básico
✓ El usuario selecciona la opción “Última Transacción”. ✓ Imprime el comprobante de la última transacción exitosa.

Tabla 9.6: Especificación de Caso de Uso 6 - Última Transacción

Caso de uso
7. Cierre de POS.
Actores
✓ Usuario.
Descripción
✓ Función que corresponde al cuadre y conciliación de las ventas.
Flujo básico
✓ El usuario selecciona la opción “Cierre de POS”. ✓ Muestra un mensaje de verificación del cierre. ✓ Si el cierre es exitoso, imprime un comprobante para el comercio con las ventas efectuadas.

Tabla 9.7: Especificación de Caso de Uso 7 - Cierre del Terminal de Venta

Caso de uso
8. Cierre de Sesión.
Actores
✓ Usuario.
Descripción
✓ Función que corresponde a la desconexión con el sistema.
Flujo básico
✓ El usuario selecciona la opción “Cierre de Sesión”. ✓ Se le muestra en pantalla un mensaje de sesión cerrada. ✓ Se contrae el menú dejando la opción de “Inicio de Sesión”

Tabla 9.8: Especificación de Caso de Uso 8 - Cierre de Sesión

Caso de uso
9. Actualización.
Actores
✓ Usuario.
Descripción
✓ Función que corresponde al cambio del sistema por una versión más reciente.
Flujo básico
✓ El usuario selecciona la opción “Actualización”.

Tabla 9.9: Especificación de Caso de Uso 9 - Actualización

9.2 Fase de Desarrollo

En esta fase se muestra el desarrollo que se llevó a cabo para la ejecución de los casos de usos anteriormente mencionados.

9.2.1 Inicio de Sesión

Para conseguir lo planteado en el caso de uso descrito en la Tabla 9.1, primero se definieron las interfaces que permiten obtener el nombre de usuario y la contraseña que serán enviadas a Scolopendra para iniciar sesión.

El método *CT_login* es el encargado de ejecutar los pasos que debe seguir el terminal para que un usuario del sistema Scolopendra pueda iniciar sesión y visualizar las opciones que tiene disponible en el menú. En la Figura 9.2 se observan los pasos que va recorriendo el terminal para cumplir con el inicio de sesión, en el cual el terminal a través de ventanas va pidiendo la información necesaria al usuario, como son el nombre de usuario y la contraseña para luego comunicarse con el sistema Scolopendra, a través del método *CT_GetSecurityToken*.

```
29      while(cStep!=LOGIN_EXIT)
30      {
31          switch(cStep)
32          {
33              case LOGIN_INIT:
34                  cStep=LOGIN_ENTER_USERNAME;
35                  break;
36
37              case LOGIN_ENTER_USERNAME:
38                  cTemp=CT_InputString(userName,"Usuario");
39                  if(cTemp==0xFB && strlen(userName) > 0)
40                      cStep=LOGIN_ENTER_PASSWORD;
41                  else if(cTemp==0xFF)
42                      cStep=LOGIN_EXIT;
43                  else
44                      cStep=LOGIN_FAIL;
45                  break;
46
47              case LOGIN_ENTER_PASSWORD:
48                  cTemp=CT_InputHidden(userPass,"Clave");
49                  if(cTemp==0xFB)
50                      cStep=LOGIN_START_COMM;
51                  else if(cTemp==0xFF)
52                      cStep=LOGIN_ENTER_USERNAME;
53                  else
54                      cStep=LOGIN_FAIL;
55                  break;
56
57              case LOGIN_START_COMM:
58                  result=CT_GetSecurityToken(userName,userPass);
59
60                  if(result==0x00){
61                      g_userData.sessionState = ONLINE;
62                      point_to_comma_decimal(g_userData.balance,balance);
63
```

Figura 9.2: Código del Método CT_login

CT_GetSecurityToken es un método intermedio que funciona como una interfaz que establece la comunicación entre el terminal y el sistema Scolopendra. Este método es el encargado de llamar a las funciones de construir el ISO-8583, conectarse al sistema Scolopendra y devolver cual es el siguiente paso para finalizar el inicio de sesión.

En la Figura 9.3 se muestra el flujo de pantallas que visualiza el usuario al iniciar sesión.

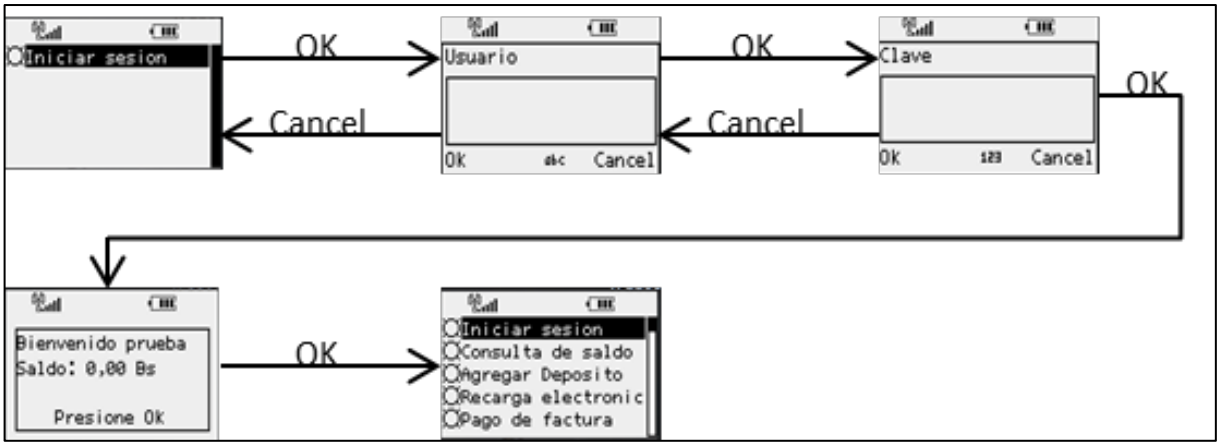

Figura 9.3: Flujo de Pantallas del Inicio de Sesión

9.2.2 **Consulta de Saldo**

El caso de uso descrito en la Tabla 9.2 referente a la consulta de saldo, está desarrollado en el método *CT_Balance*, encargado de ejecutar los pasos que debe seguir el terminal para que un usuario del sistema Scolopendra pueda consultar su saldo disponible y diferido.

En la Figura 9.4 se muestra el flujo de pantallas que visualiza el usuario al consular su saldo.

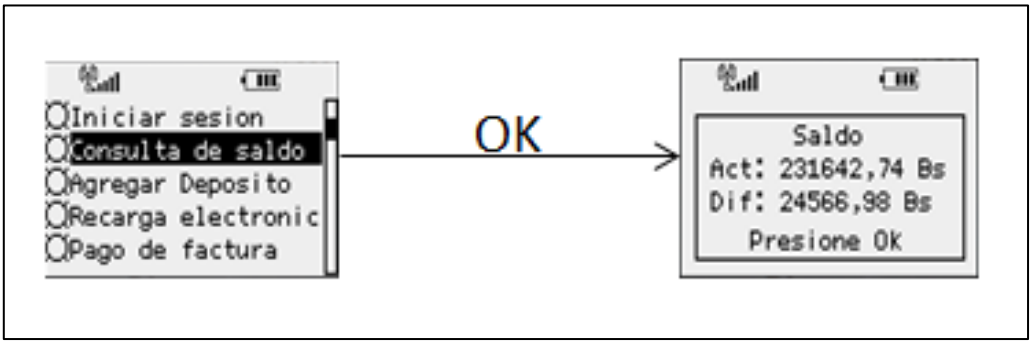

Figura 9.4: Flujo de Pantallas de la Consulta de Saldo

En la Figura 9.5 se puede ver el código del método *CT_Balance* y los pasos que va recorriendo el terminal para realizar una consulta de saldo, en donde el método sólo verifica que el usuario del sistema esté conectado y luego hace un llamado a la función *CT_GetBalance.*

La función *CT_GetBalance* actúa como una interfaz entre el terminal y el sistema Scolopendra. Esta función se encarga de llamar al constructor del mensaje ISO-8583, conectarse al sistema Scolopendra y decidir cuál es el siguiente paso que el terminal debe seguir para finalizar la consulta de saldo.

```
18  void CT_Balance(){
19
20
21      BALANCE_STEP cStep;
22      char message[128];
23      char information[INF_LENGHT];
24      uint8 cTemp;
25      char date[20], balance[20], pendingBalance[20];
26      char * system_time;
27      char time[15];
28
29
30      cStep=BALANCE_INIT;
31      while(cStep!=BALANCE_EXIT)
32      {
33          switch(cStep)
34          {
35              case BALANCE_INIT:
36                  if( test_session() )
37                      cStep=BALANCE_START_COMM;
38                  else{
39                      sprintf(message,"Debe iniciar sesion para continuar.");
40                      CT_Notify(message,0);
41                      cStep = BALANCE_FAIL;
42                  }
43                  break;
44
45              case BALANCE_START_COMM:
46                  cTemp=CT_GetBalance();
47                  if(cTemp==0x00){
48                      point_to_comma_decimal(g_userData.balance,balance);
49                      point_to_comma_decimal(g_userData.pendingBalance,pendingBalance);
50                      sprintf(message,"\t\t\t\t\t\t\tSaldo\n Act: %s Bs\n Dif: %s Bs",balance,pendingBalance);
51                      CT_Notify(message,0);
52                      cStep=BALANCE_EXIT;
53                  }else if(cTemp==0x01){
54                      g_userData.sessionState = EXPIRED;
55                      sprintf(information, "Error: %s", response_message);
56                      CT_Notify(information,0);
57                      cStep=BALANCE_FAIL;
58                  }else{
59                      sprintf(information, "Error: %s", response_message);
60                      CT_Notify(information,0);
61                      cStep=BALANCE_FAIL;
```

Figura 9.5: Código del Método CT_Balance

9.2.3 **Agregar Depósito**

La opción de agregar depósito en el menú del terminal descrita en el caso de uso de la Tabla 9.3, está implementada en el método *CT_Deposit*, el cual se encarga de ejecutar los pasos que debe seguir el terminal para que un usuario pueda registrar un depósito en el sistema Scolopendra y este sea agregado como parte de su saldo disponible.

La Figura 9.6 muestra parte del código que ejecuta el terminal para realizar el registro de un depósito. En este método, a través de las ventanas que le muestra al usuario obtiene la siguiente información: el banco donde se realizó el depósito, el número de referencia o voucher, el monto depositado, la fecha en la que realizó el depósito y una confirmación con los datos suministrados al terminal.

```
63      while(cStep!=DEPOSIT_EXIT)
64      {
65          switch(cStep)
66          {
67              case DEPOSIT_INIT:
68                  if( test_session() )
69                      cStep = DEPOSIT_SELECT_BANK;
70                  else{
71                      CT_Notify("Debe iniciar sesion para continuar.",0);
72                      cStep = DEPOSIT_EXIT;
73                  }
74                  break;
75
76              case DEPOSIT_SELECT_BANK:
77                  cTemp = CT_Menu(bank_list, 10, &bank_index, TL_LIST, PNULL);
78                  if(cTemp==0x01)
79                      cStep = DEPOSIT_CONTROL_NUMBER;
80                  else if( cTemp == 0xFF )
81                      cStep = DEPOSIT_EXIT;
82                  else
83                      cStep = DEPOSIT_FAIL;
84                  break;
85
86              case DEPOSIT_CONTROL_NUMBER:
87                  cTemp=CT_InputInteger(depositControlNumber,"Numero de Deposito");
88                  if(cTemp==0xFB)
89                      cStep=DEPOSIT_ADD_AMOUNT;
90                  else if(cTemp==0xFF)
91                      cStep=DEPOSIT_SELECT_BANK;
92                  else
93                      cStep=DEPOSIT_FAIL;
94                  break;
95
96              case DEPOSIT_ADD_AMOUNT:
97                  cTemp=CT_InputDecimal(depositAmount,"Monto de Deposito");
98                  if(cTemp==0xFB)
99                      cStep=DEPOSIT_ADD_DATE;
100                 else if(cTemp==0xFF)
101                     cStep=DEPOSIT_CONTROL_NUMBER;
102                 else
103                     cStep=DEPOSIT_FAIL;
104                 break;
105
```

Figura 9.6: Código del Método CT_Deposit

La Figura 9.7 muestra el flujo de pantallas que visualiza el usuario al agregar un depósito.

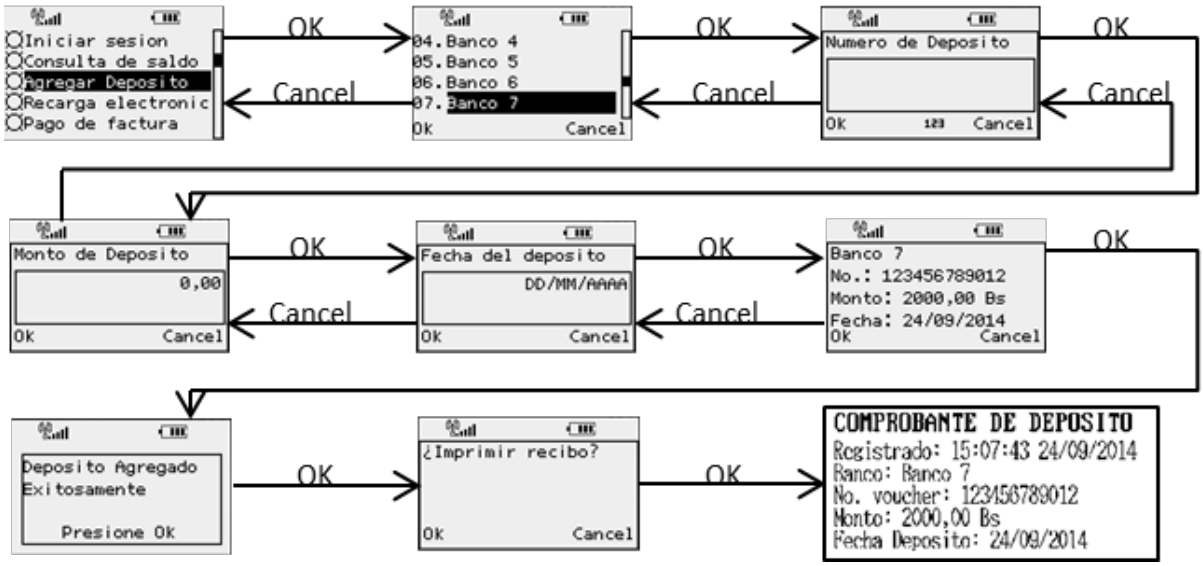

Figura 9.7: Flujo de Pantallas de Agregar Depósito

Una vez suministrada la información al terminal, este hace una conexión al sistema Scolopendra, a través del método *CT_AddDeposit*, el cual funciona como

una interfaz que establece la comunicación entre el terminal de venta y el sistema Scolopendra. El método es el encargado de llamar a las funciones que se encargan de construir el mensaje ISO-8583, conectarse al sistema Scolopendra y devolver cual es el siguiente paso para finalizar la opción de agregar depósito.

9.2.4 **Recarga Electrónica**

Como solución del caso de uso planteado en la Tabla 9.4, se implementó el método *CT_Recharge* el cual se encarga de ejecutar los pasos que debe seguir el terminal para que un usuario pueda realizar una recarga electrónica o PIN electrónico en el sistema Scolopendra.

La Figura 9.8 muestra parte del código que ejecuta el terminal para realizar una recarga electrónica o PIN electrónico. En este método se obtiene la información para la recarga electrónica o el PIN electrónico, esto varía dependiendo del servicio que desea el cliente. Para la recarga electrónica es necesario obtener a través de las ventanas que se le muestran al usuario la siguiente información del cliente: la compañía proveedora del servicio, el monto de la recarga, el número de contrato y la confirmación del número de contrato. Para la creación de un PIN electrónico se necesita la siguiente información: la compañía proveedora del servicio y el monto fijo del PIN.

```
                        CT_Notify("El numero del suscriptor no coincide",0);
                        cStep = RECHARGE_ADD_SUSC_NUMBER;
                    }
                }else if( cTemp == 0xFF )
                    cStep = RECHARGE_ADD_SUSC_NUMBER;
                else
                    cStep = RECHARGE_FAIL;
                break;

            case RECHARGE_CONFIRMATION:
                sprintf(information, "Operadora: %s\nMonto: %s Bs", company_list_recharge[company_index-1].name, amount_recha
                cTemp = CT_Confirm(information);
                if( cTemp == 0x00 )
                    cStep = RECHARGE_START_COMM;
                else
                    cStep = RECHARGE_INIT;
                break;

            case RECHARGE_START_COMM:


                cTemp = CT_AddRecharge(company_list_recharge[company_index-1].id, suscriptorNumber, amount_recharge, serialN

                if( cTemp == 0x00 )
                    cStep = RECHARGE_REGISTER;
                else
                    cStep = RECHARGE_FAIL;
                break;

            case RECHARGE_REGISTER:
                response = CT_RegisterInLastTransFile(company_list_recharge[company_index-1].id, serialNumber, pin, suscriptc

                CT_RegisterInLogFile(company_list_recharge[company_index-1].id, serialNumber, suscriptorNumber, system_time,

                if( response == 0x01 ){
                    CT_Notify("File don't exist", 0);
                }else{
                    if( response < 0 ){
                        sprintf(information, "Error: %s", response_message);
                        CT_Notify(information, 0);
                    }else{
                        CT_Notify("Operacion Registrada", 2);
                    }
```

Figura 9.8: Código del Método CT_Recharge

Luego de recibir la información del usuario, el siguiente proceso es conectarse con el sistema Scolopendra, a través del método *CT_AddRecharge*, que funciona como una interfaz que establece la comunicación entre el terminal de venta y el sistema Scolopendra, este método es el encargado de llamar a las funciones de construir el mensaje ISO-8583, conectarse al sistema Scolopendra y retornar el siguiente paso que debe ejecutar el terminal para finalizar el proceso.

En la Figura 9.9 se muestra el flujo de pantallas que visualiza el usuario al hacer una recarga electrónica.

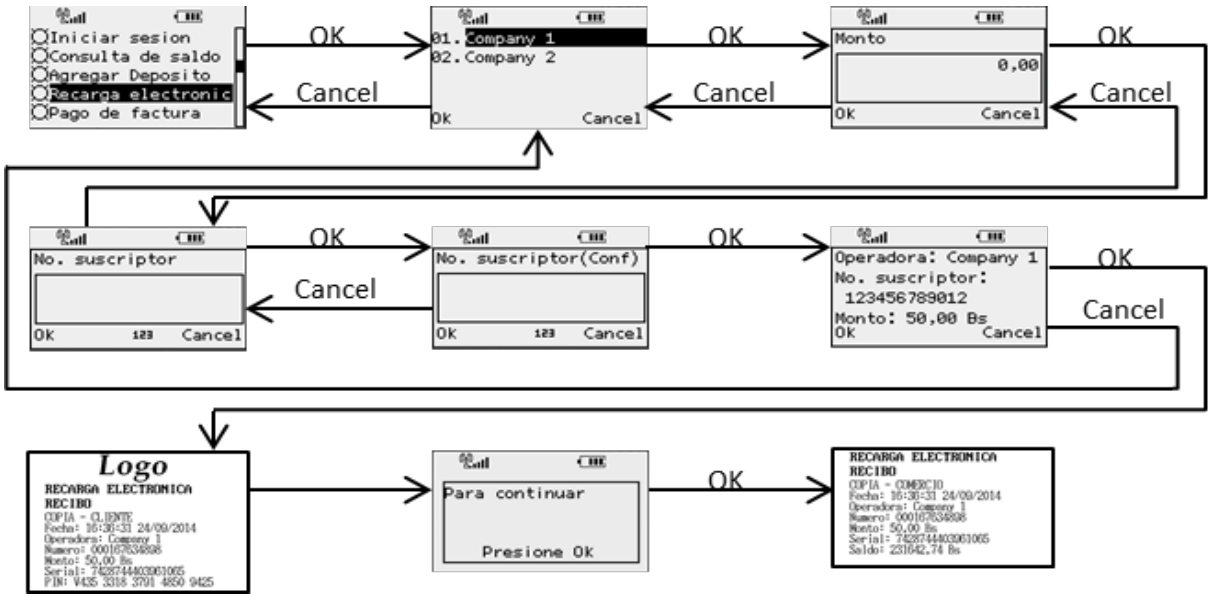

Figura 9.9: Flujo de Pantallas de la Recarga Electrónica

En la Figura 9.10 se muestra el flujo de pantallas que visualiza el usuario al hacer un PIN electrónico.

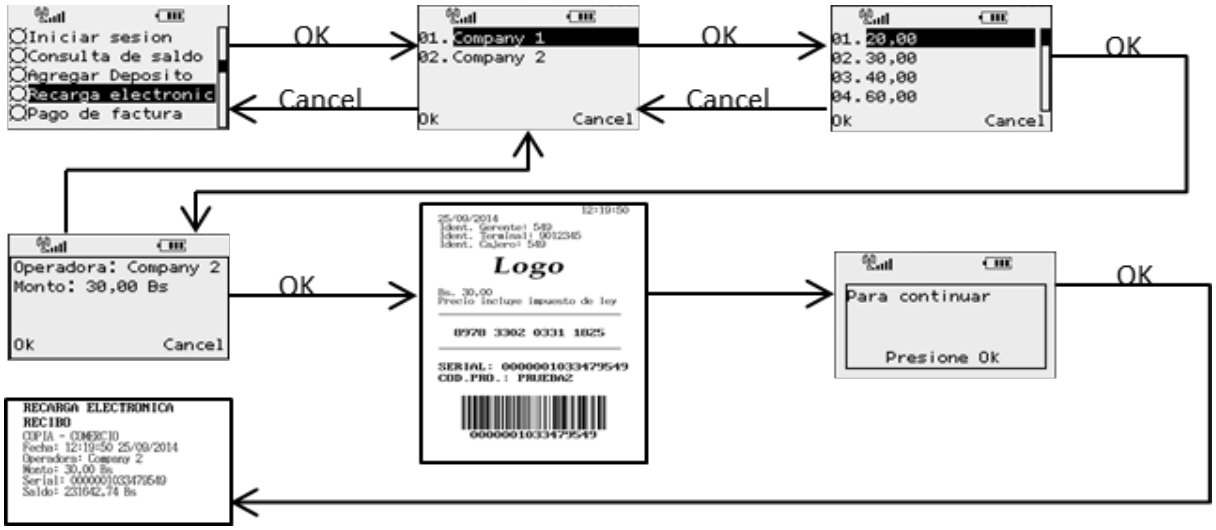

Figura 9.10: Flujo de Pantallas del PIN Electrónico

9.2.5 **Pago de Factura**

En el pago de factura descrito en el caso de uso planteado en la Tabla 9.5, se desarrolló el método *CT_Payment* que se encarga de ejecutar los pasos que debe seguir el terminal para que un usuario pueda realizar un pago de factura en el sistema Scolopendra.

La Figura 9.11 muestra parte del código que ejecuta el terminal para realizar un pago de factura. En este método, se va recopilando la información necesaria para hacer un pago de factura, a través de ventanas mostradas al usuario para que suministre la siguiente información: compañía de servicio, monto de la factura, número de control de la factura y confirmación del número de control de la factura. Una vez suministrada la información, el terminal inicia la comunicación con el sistema Scolopendra a través del método *CT_AddPayment*.

El método *CT_AddPayment* funciona como una interfaz que establece la comunicación entre el terminal de venta y el sistema Scolopendra. Para ello *CT_AddPayment* se encarga de construir el mensaje ISO-8583, conectarse al sistema Scolopendra y retornar el siguiente paso que debe ejecutar el terminal de venta para finalizar el pago de factura.

```
109
110             case PAYMENT_ADD_BILL_NUMBER:
111                 cTemp = CT_InputInteger(billNumber, "No. Factura");
112                 if( cTemp == 0xFB )
113                     cStep = PAYMENT_CONFIRM_BILL_NUMBER;
114                 else if( cTemp == 0xFF )
115                     cStep = PAYMENT_ADD_AMOUNT;
116                 else
117                     cStep = PAYMENT_FAIL;
118                 break;
119
120             case PAYMENT_CONFIRM_BILL_NUMBER:
121                 cTemp = CT_InputInteger(billNumberConfirm, "No. Factura(Conf)");
122                 if( cTemp == 0xFB ){
123                     if ( strcmp(billNumber, billNumberConfirm) == 0 )
124                         cStep = PAYMENT_CONFIRMATION;
125                     else{
126                         CT_Notify("Los numeros de facturas no coinciden",0);
127                         cStep = PAYMENT_ADD_BILL_NUMBER;
128                     }
129                 }else if( cTemp == 0xFF )
130                     cStep = PAYMENT_ADD_BILL_NUMBER;
131                 else
132                     cStep = PAYMENT_FAIL;
133                 break;
134
135             case PAYMENT_CONFIRMATION:
136                 sprintf(information, "Compania: %s\nNo. Factura: %s\nMonto: %s Bs", company_list_payment[company_index-1].nam
137                 cTemp = CT_Confirm(information);
138                 if( cTemp == 0x00 )
139                     cStep = PAYMENT_START_COMM;
140                 else
141                     cStep = PAYMENT_INIT;
142                 break;
143
144             case PAYMENT_START_COMM:
145                 cTemp = CT_AddPayment(company_list_payment[company_index-1].id, billNumber, paymentAmount, serialNumber, pin,
146
147                 if( cTemp == 0x00 )
148                     cStep = PAYMENT_REGISTER;
149                 else
150                     cStep = PAYMENT_FAIL;
151                 break;
```

Figura 9.11: Código del Método CT_AddPayment

En la Figura 9.12 se muestra el flujo de pantallas que visualiza el usuario al hacer un pago de factura.

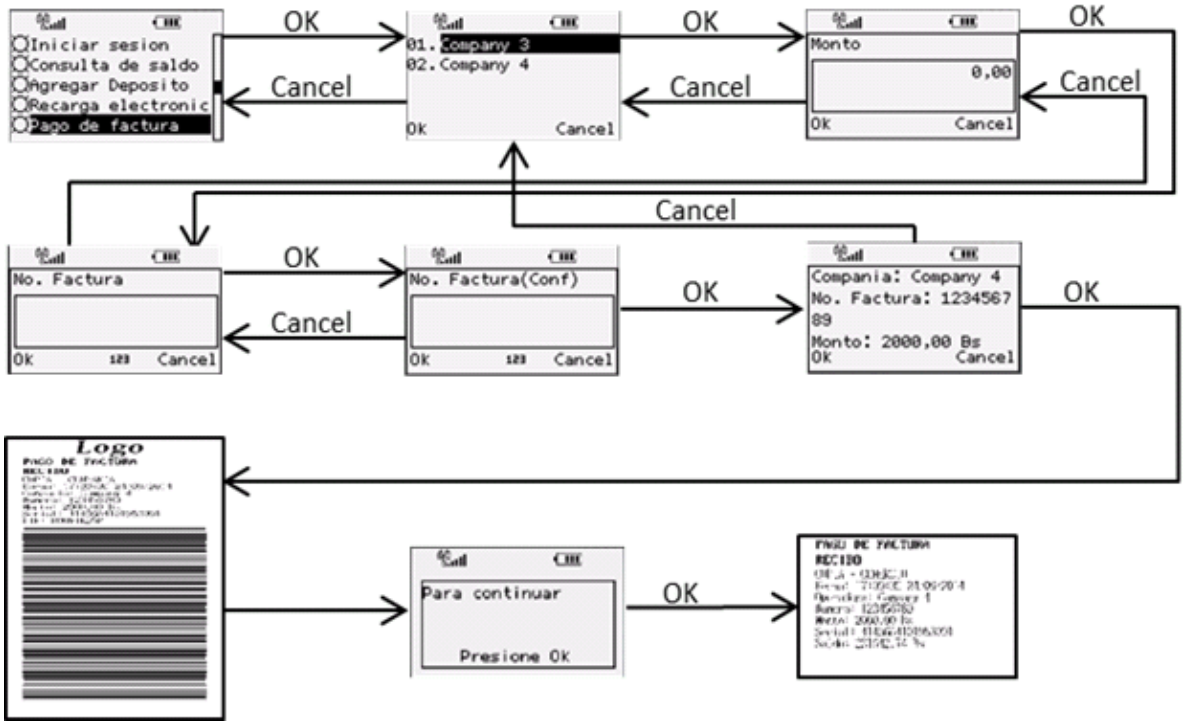

Figura 9.12: Flujo de Pantalla del Pago de Factura

9.2.6 **Última Transacción**

Como solución para el caso de uso planteado en la Tabla 9.6, se implementó el método *CT_PrintLastTransaction* encargada de ejecutar los pasos que debe seguir el terminal para que un usuario pueda imprimir la última transacción.

La Figura 9.14 muestra el código que sigue el terminal para imprimir la última transacción. Este método verifica que exista la última transacción, si existe entonces busca la información ahí almacenada y la imprime a través del método *CT_PrintReceiptClientAndCompany*. En caso contrario, verifica las transacciones realizadas por el usuario y busca la última para luego imprimirla a través del método *CT_PrintReceiptClientAndCompany*.

El método *CT_PrintReceiptClientAndCompany* es el encargado de verificar si ocurrió un fallo en la impresora y cuales tickets faltaron por imprimirse, si no hubo fallo en la impresora, imprime solo la copia del comercio, de esta manera se prevé que un usuario no pueda imprimir el ticket del cliente luego de ser previamente impreso.

En la Figura 9.13 se muestra el flujo de pantallas que visualiza el usuario al hacer la última transacción.

Figura 9.13: Flujo de Pantallas de la Última Transacción

```
503  void CT_PrintLastTransaction(){
504      SALE_INFO sale_info;
505      SALE_INFO_FULL sale_info_full;
506      int32 cTemp;
507      char amountString[15];
508      char companyName[10];
509
510      double amount;
511
512      if( TP_FExist( "LAST" ) ){
513          cTemp = CT_LastTransaction(&sale_info_full);
514          if( cTemp == 0x00 ){
515              amount = ISO8583DecimalTodecimalString(sale_info_full.isoAmount);
516              point_to_comma_decimal(amount, amountString);
517
518              if( sale_info_full.type == 0 ){
519                  get_name_from_company_recharge(sale_info_full.serviceCompany, companyName);
520              }else{
521                  get_name_from_company_payment(sale_info_full.serviceCompany, companyName);
522              }
523
524              cTemp = CT_PrintReceiptClientAndCompany(sale_info_full.type, companyName, sale_info_full.serviceCompany, sale_inf
525          }
526      }else{
527          if( TP_FExist( "LOG" ) ){
528              cTemp = CT_LastTransactionInLog(&sale_info);
529              if( cTemp > 0x00 ){
530                  amount = ISO8583DecimalTodecimalString(sale_info.isoAmount);
531                  point_to_comma_decimal(amount, amountString);
532
533                  if( sale_info.type == 0 ){
534                      get_name_from_company_recharge(sale_info.serviceCompany, companyName);
535                  }else{
536                      get_name_from_company_payment(sale_info.serviceCompany, companyName);
537                  }
538
539                  cTemp = CT_PrintReceiptCompany(sale_info.type, companyName, sale_info.serviceCompany, sale_info.subscriber, a
540              }else{
541                  CT_Notify("No hay transacciones para imprimir", 0);
542              }
543          }else{
544              CT_Notify("No hay transacciones para imprimir", 0);
545          }
546      }
```

Figura 9.14: Código del Método CT_PrintLastTransaction

9.2.7 **Cierre de POS**

La solución implementada del caso de uso descrito en la Tabla 9.7 está desarrollada en el método *CT_ClosePOS*, el cual se encarga de ejecutar los pasos que debe seguir el terminal para que un usuario pueda hacer un cierre de POS.

La Figura 9.15 muestra el código que sigue el terminal de venta con los pasos para ejecutar un cierre de POS. Este método se encarga de sumar todas las transacciones y montos de las ventas registrados en el terminal de venta, para

luego hacer un cuadre con el sistema Scolopendra. Una vez obtenida la información el terminal inicia la comunicación con el sistema Scolopendra a través del método *CT_CloseOutPOS*.

El método *CT_CloseOutPOS* funciona como una interfaz que establece la comunicación entre el terminal de venta y el sistema Scolopendra. Para ello *CT_CloseOutPOS* se encarga de construir el mensaje ISO-8583, conectarse al sistema Scolopendra y retornar el siguiente paso que debe ejecutar el terminal de venta para finalizar el cierre del terminal.

```
35              case CLOSEOUT_INIT:
36                  if( test_session() )
37                      cStep = CLOSEOUT_CONFIRM;
38                  else{
39                      CT_Notify("Debe iniciar sesion para continuar.",0);
40                      cStep = CLOSEOUT_EXIT;
41                  }
42                  break;
43
44              case CLOSEOUT_CONFIRM:
45                  cTemp = CT_Confirm("¿Seguro que desea cerrar el Punto de Venta?");
46                  if( cTemp == 0x00 ){
47                      cStep = CLOSEOUT_START_COMM;
48                  }else{
49                      cStep = CLOSEOUT_EXIT;
50                  }
51                  break;
52
53              case CLOSEOUT_START_COMM:
54                  cTemp = CT_CloseOutPOS(total_txn_out, total_amount_out, system_time, total_txn_amount);
55                  if( cTemp == 0x00 ){
56                      cStep = CLOSEOUT_CONFIRM_COMM;
57                  }else{
58                      cStep = CLOSEOUT_FAIL;
59                  }
60                  break;
61
62              case CLOSEOUT_CONFIRM_COMM:
63                  CT_CloseOutConfirm();
64
65                  cStep = CLOSEOUT_SUCCESS;
66                  break;
67
68              case CLOSEOUT_SUCCESS:
69                  amount = ISO8583DecimalTodecimalString(total_amount_out);
70                  total = atoi(total_txn_out);
71                  sprintf(information, "Cierre Exitoso\nMonto: %0.2f Bs\nNo. Txn: %d", amount, total);
72
73                  CT_Notify(information, 0);
74
```

Figura 9.15: Código del Método CT_ClosePOS

En la Figura 9.16 se muestra el flujo de pantallas que visualiza el usuario al hacer la última transacción.

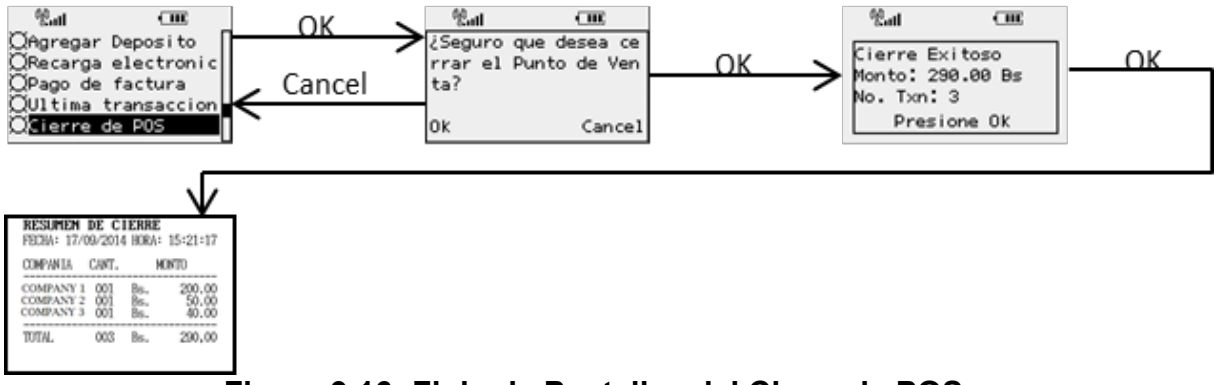

Figura 9.16: Flujo de Pantallas del Cierre de POS

9.2.8 **Cierre de Sesión**

El cierre de sesión planteado en el caso de uso descrito en la Tabla 9.8 está implementada en el método *CT_Logout*, que se encarga de borrar la información generada por el inicio de sesión, como el token de seguridad, el estado de la sesion, etc.

La Figura 9.17 se muestra el flujo de pantallas que visualiza el usuario al hacer la última transacción.

Figura 9.17: Flujo de Pantallas del Cierre de Sesión

9.2.9 **Actualización**

El caso de uso planteado en la Tabla 9.9 está implementado sobre una plantilla de programación dada por la compañía Telepower Communication y adaptada a los requerimientos de nuestro software, que permite instalar dentro del terminal de venta una versión mejorada del firmware que tiene instalado. Esta función solo está disponible para el terminal de venta TPS300 y el usuario debe estar previamente autorizado para hacer la actualización.

9.3 Seguridad

La seguridad es un aspecto importante que se debe considerar cuando se trata de terminales de venta o comercio electrónico [6] por el hecho de manejar información confidencial.

9.3.1 **Seguridad de la Sesión**

Scolopendra utiliza como seguridad un token de sesión para validar que el usuario se encuentra activo en el sistema. Esto permite controlar el tiempo de vida que puede tener una sesión activa en Scolopendra. Este tiempo puede variar según el tipo de servicio al cual pertenezca el usuario. De esta manera, Scolopendra valida que un usuario puede realizar transacciones, además cada usuario tiene un nivel de seguridad, el cual indica cuales son las transacciones que tiene permitidas en el sistema.

9.3.2 **Criptografía**

El robo de información es un tipo de ataque que efectúan personas no autorizadas para intentar acceder a la información confidencial relacionada a las transacciones a través de la red por donde viajan las mismas. Para evitar este ataque un mecanismo muy utilizado es la criptografía [4].

Como técnica de seguridad en los mensajes ISO-8583 que son enviados y recibidos entre el terminal de venta y el sistema Scolopendra, se utiliza el protocolo de seguridad 128-bit Secure Sockets Layer (SSL) 3.0 [2] que facilita la autenticación y privacidad de la información en internet mediante el uso de la criptografía. Con esto se asegura que las transacciones que se realizan a través de la red vayan por una conexión encriptada en donde se garantiza la protección de la información.

10. Conclusiones

El número de transacciones electrónicas que se realizan en el país está en vías de crecimiento y con el pasar del tiempo se ha popularizado, cada día es más común poseer dinero electrónico.

En tal sentido las recargas electrónicas no son la excepción, siendo esta una alternativa a las tarjetas Scratch-Card, y como una opción para que los sistemas pre-pagados pueda ofrecer soluciones más accesibles a sus clientes.

La solución llevada a cabo en este Trabajo Especial de Grado, va orientado a la anterior premisa de brindar una solución más accesible a los clientes de los sistemas pre-pagados utilizando terminales de venta.

Para su implementación fue utilizado los lenguajes de programación C y C++ que permitió cubrir los objetivos planteados tanto en los terminales de venta como en la plataforma Scolopendra.

La aplicación implementada en el terminal de venta modelo TPS300 fue realizada utilizando Visual Studio 2008 como entorno de desarrollo y un dispositivo hardware denominado SoftDog que permitieron ir desarrollando sin tener que cargar el firmware de la aplicación a la memoria ROM del terminal de venta para realizar pruebas a manera de facilitar el proceso de depuración. Por otra parte las librerías y plantillas que fueron utilizadas en el desarrollo llevado a cabo en la aplicación del terminal de venta fueron suministradas por las compañías proveedoras de los terminales de venta, en la cual dan al desarrollador un abanico de funciones sencillas y útiles para facilitar la programación de nuevas funcionalidades en el terminal de venta.

Es importante resaltar que las plantillas desarrolladas para establecer la comunicación entre los terminales de venta y la plataforma Scolopendra, están totalmente ajustadas a los requerimientos que utiliza Scolopendra para realizar las transacciones electrónicas.

El objetivo principal del Trabajo Especial de Grado fue realizar una plantilla para facilitar la implementación de aplicaciones en los diferentes modelos de terminales de venta, desarrollar los eventos de las transacciones electrónicas en la plataforma Scolopendra y poner en práctica una aplicación en el terminal de venta TPS300, en este sentido se cumplió con todos los objetivos planteados.

11. Anexo

Subcampo	Nombre del tipo de elemento	Tamaño	Tipo
1	Bitmap secundario	Fijo, 8 bytes	Binario
2	Número de cuenta primaria	LLVAR, 21 bytes	Numérico
3	Código del proceso	Fijo, 6 bytes	Numérico
4	Monto, Transacción	Fijo, 12 bytes	Numérico
5	Monto, Conciliación	Fijo, 12 bytes	Numérico
6	Monto, factura del tarjetahabiente	Fijo, 12 bytes	Numérico
7	Fecha y hora, transmisión	Fijo, 10 bytes	Numérico
8	Monto, cuota de factura del tarjetahabiente	Fijo, 8 bytes	Numérico
9	Tasa de conversión de conciliación	Fijo, 8 bytes	Numérico
10	Tasa de conversión de facturación del tarjetahabiente	Fijo, 8 bytes	Numérico
11	Número de rastreo de auditoria	Fijo, 6 bytes	Numérico
12	Fecha y hora, transacción local	Fijo, 12 bytes	Numérico
13	Fecha efectiva	Fijo, 4 bytes	Numérico
14	Fecha de expiración	Fijo, 4 bytes	Numérico
15	Fecha de liquidación	Fijo, 6 bytes	Numérico
16	Fecha de conversión	Fijo, 4 bytes	Numérico
17	Fecha de captura	Fijo, 4 bytes	Numérico
18	Tipo de comerciante	Fijo, 4 bytes	Numérico
19	Código del país, entidad adquirente	Fijo, 3 bytes	Numérico
20	Código del país, cuenta principal	Fijo, 3 bytes	Numérico
21	Código del país, entidad facilitadora	Fijo, 3 bytes	Numérico
22	Código del punto de servicio	Fijo, 12 bytes	alfanumérico
23	Número de secuencia de la tarjeta	Fijo, 3 bytes	Numérico
24	Código de función	Fijo, 3 bytes	Numérico
25	Código de la razón del mensaje	Fijo, 4 bytes	Numérico
26	Código de negocio	Fijo, 4 bytes	Numérico
27	Longitud del código de aprobación	Fijo, 1 bytes	Numérico
28	Fecha de conciliación	Fijo, 6 bytes	Numérico
29	Indicador de conciliación	Fijo, 3 bytes	Numérico
30	Monto original	Fijo, 24 bytes	Numérico
31	Datos de referencia del adquisidor	LLVAR, 50 bytes	Alfanumérico, y especiales
32	Código de identificación del adquisidor	LLVAR, 13 bytes	Numérico
33	Reenvío del código de identificación del adquisidor	LLVAR, 13 bytes	Numérico
34	Número de cuenta principal, extendido	LLVAR, 30 bytes	Numérico
35	Data de rastreo 2	LLVAR, 39 bytes	Alfanumérico, y especiales

36	Data de rastreo 3	LLVAR, 107 bytes	Numérico, y especiales
37	Recuperación de número de referencia	Fijo, 12 bytes	Alfanumérico, y especiales
38	Código de aprobación	Fijo, 6 bytes	Alfanumérico, y especiales
39	Código de acción	Fijo, 3 bytes	Numérico
40	Código de servicio	Fijo, 3 bytes	Numérico
41	identificador del terminal del contratista de la tarjeta	Fijo, 8 bytes	Alfanumérico, y especiales
42	Código que identifica el contratista de la tarjeta	Fijo, 15 bytes	Alfanumérico, y especiales
43	Nombre/ ubicación del contratista de la tarjeta	LLVAR, 101 bytes	Alfanumérico, y especiales
44	Datos adicionales de la respuesta	LLVAR, 27 bytes	Alfanumérico, y especiales
45	Data de rastreo 3	LLVAR, 78 bytes	Alfanumérico, y especiales
46	Montos, honorarios	LLLVAR, 207 bytes	Alfanumérico
47	Datos adicionales – nacional	LLLVAR, 290 bytes	Alfanumérico, y especiales
48	Datos adicionales – privado	LLLVAR, 43 bytes	Alfanumérico, y especiales
49	Código de la moneda, transacción	Fijo, 3 bytes	Numérico
50	Código de la moneda, conciliación	Fijo, 3 bytes	Alfabético o numérico
51	Código de la moneda, facturación del titular de la tarjeta	Fijo, 3 bytes	Alfabético o numérico
52	Dato del (PIN) número de identificación personal	Fijo, 8 bytes	Binario
53	Información de control relacionado a seguridad	LLVAR, 10 bytes	Alfanumérico
54	Montos adicionales	LLLVAR, 123 bytes	Alfanumérico, y especiales
55	Datos relacionados al sistema de circuito integrado de tarjetas	LLLVAR, 259 bytes	Alfanumérico, y especiales, BDC o binario
56	Elementos originales	LLVAR, 37 bytes	Numérico
57	Ciclo de vida del código de autorización	Fijo, 3 bytes	Numérico
58	Código de identificación del agente autorizador	LLVAR, 13 bytes	Numérico
59	Datos de transporte	LLLVAR, 1002 bytes	Alfanumérico, y especiales
60	Datos de uso nacional	LLLVAR, 303 bytes	Alfanumérico, y especiales

61	Datos de uso nacional	LLLVAR, 103 bytes	Alfanumérico, y especiales
62	Datos de uso privado	LLVAR, 63 bytes	Alfanumérico, y especiales o binario
63	Datos de uso privado	LLLVAR, 208 bytes	Alfanumérico, y especiales
64	Código de autenticación del mensaje	Fijo, 8 bytes	Binario
65	Uso reservado para ISO	Fijo, 8 bytes	Binario
66	Montos, honorarios	LLLVAR, 204 bytes	Alfanumérico, y especiales
67	Extendido de los datos del pago	Fijo, 2 bytes	Numérico
68	Código del país, institución receptora	Fijo, 3 bytes	Numérico
69	Código del país, institución liquidadora	Fijo, 3 bytes	Numérico
70	Código del país, agente autorizador	Fijo, 3 bytes	Numérico
71	Número del mensaje	Fijo, 8 bytes	Numérico
72	Registro de datos	LLLVAR, 999 bytes	Alfanumérico, y especiales
73	Fecha, acción	Fijo, 6 bytes	Numérico
74	Créditos, número	Fijo, 10 bytes	Numérico
75	Créditos, número reservado	Fijo, 10 bytes	Numérico
76	Débitos, número	Fijo, 10 bytes	Numérico
77	Débitos, número reservado	Fijo, 10 bytes	Numérico
78	Transferencia, número	Fijo, 10 bytes	Numérico
79	Transferencia, número reservado	Fijo, 10 bytes	Numérico
80	Consultas, número	Fijo, 10 bytes	Numérico
81	Autorizaciones, número	Fijo, 10 bytes	Numérico
82	Consultas, número reservado	Fijo, 10 bytes	Numérico
83	Pagos, número	Fijo, 10 bytes	Numérico
84	Pagos, número reservado	Fijo, 10 bytes	Numérico
85	Cuota de pago	Fijo, 10 bytes	Numérico
86	Créditos, número	Fijo, 16 bytes	Numérico
87	Créditos, número reservado	Fijo, 16 bytes	Numérico
88	Débitos, número	Fijo, 16 bytes	Numérico
89	Débitos, número reservado	Fijo, 16 bytes	Numérico
90	Autorizaciones	Fijo, 10 bytes	Numérico
91	Código de país, destino de transacción	Fijo, 3 bytes	Numérico
92	Código de país, originador de transacción	Fijo, 3 bytes	Numérico
93	Código de identificación, destino de transacción	Fijo, 11 bytes	Numérico

94	Código de identificación, originador de transacción	Fijo, 11 bytes	Numérico
95	Datos de referencia del emisor de la tarjeta	LLVAR, 99 bytes	Alfanumérico, y especiales
96	Clave de gestión de datos	LLLVAR, 999 bytes	Binario
97	Monto, liquidación neta	Fijo, 16 bytes	Numérico
98	Beneficiario	Fijo, 25 bytes	Alfanumérico, y especiales
99	Identificador de la institución liquidadora	LLVAR, 11 bytes	Alfanumérico
100	Identificador de la institución receptora	LLVAR, 11 bytes	Numérica
101	Nombre del archivo	LLVAR, 17 bytes	Alfanumérico, y especiales
102	Identificador de cuenta 1	LLVAR, 28 bytes	Alfanumérico, y especiales
103	Identificador de cuenta 2	LLVAR, 28 bytes	Alfanumérico, y especiales
104	Descripción de la transacción	LLLVAR, 100 bytes	Alfanumérico, y especiales
105	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
106	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
107	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
108	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
109	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
110	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
111	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
112	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
113	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
114	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
115	Uso reservado para ISO	LLLVAR, 999 bytes	Alfanumérico, y especiales
116	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico, y especiales
117	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico,

			y especiales
118	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico, y especiales
119	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico, y especiales
120	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico, y especiales
121	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico, y especiales
122	Reservado para uso nacional	LLLVAR, 999 bytes	Alfanumérico, y especiales
123	Reservado para uso privado	LLLVAR, 999 bytes	Alfanumérico, y especiales
124	Reservado para uso privado	LLLVAR, 999 bytes	Alfanumérico, y especiales
125	Reservado para uso privado	LLLVAR, 999 bytes	Alfanumérico, y especiales
126	Reservado para uso privado	LLLVAR, 999 bytes	Alfanumérico, y especiales
127	Reservado para uso privado	LLLVAR, 999 bytes	Alfanumérico, y especiales
128	Código de autenticación del mensaje	Fijo, 8 bytes	Binario

Tabla 11.1: Definición de los Campos del Mensaje

Referencias Bibliográficas

[1] ISO/IEC 8583, *Financial Transaction Card Originated Messages - Interchange Message Specifications*, 1995.

[2] E. Rescorla, *SSL and TLS: Designing and Building Secure Systems*, 2000.

[3] A. Shuja, J. Krebs, *IBM Rational Unified Process Reference and Certification Guide: Solution Designer*, 2007.

[4] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C,* Wiley, 1996.

[5] C. Radu, *Implementing Electronic Card Payment Systems*, Artech House, 2002.

[6] M. Hashem Sherif, *Protocols for Secure Electronic Commerce*, Second Edition, CRC Press, 2003.

[7] Corporacion Novared, Plan de Sistemas Scolopendra, 2008.

[8] Telepower Communication Co., SDK Development Guide, 2013.

[9] Telepower Communication Co., Simulator User Guide, 2013.

[10] Telepower Communication Co., Software Development Kit Reference Manual, 2013.

[11] Shenzhen Guanri Telecom-Tech Co., System Introduction Guide, 2013.

[12] Bonus Comunicaciones S.R.L., Protocolo de Comunicaciones Multicab POS, Abril 2012.

[13] JPOS.org, jPOS Programmer's Guide, Version 3, November 2007.

[14] J8583, Guía del Usuario, Noviembre 2013.