UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN

Diseño y Desarrollo de Plataforma Integrada GSM – SMPP para la administración de mensajes de texto SMS y publicación Web

Trabajo Especial de Grado presentado ante la ilustre
Universidad Central de Venezuela por los Bachilleres

Giovanni Esposito Ambrosio
Jorge Gómez Martínez

para optar al título de
Licenciado en Computación

TUTORES
Prof. Ana Romero
Prof. María Elena Villapol

Caracas, Octubre de 2008

RESUMEN

El objetivo del presente Trabajo Especial de Grado consiste en la implementación de una plataforma capaz de proveer, a empresas que prestan servicios de valor agregado mediante la mensajería corta de texto, el control de todos sus envíos, así como las respectivas respuestas de los mismos, además de ser configurable para cualquier tipo de operadora, no sólo a nivel nacional sino también internacional. Este desarrollo fue motivado por la necesidad de la empresa ALS Comunicaciones, quién presta este tipo de servicios, con el fin de ampliar su plataforma y poder brindar un mejorar servicio a sus clientes; haciendo uso de las especificaciones del protocolo estándar de clase mundial SMPP (Short Message Peer-to-Peer), la implementación fue llevada a cabo en lenguaje Java y como manejador de base de datos MySQL.

Palabras Claves

- SMPP.
- Short Message Peer to Peer.
- SMS.
- Short Messaging System.
- Mensajería de texto.

Índice General

Índice de Tablas e Imágenes

CAPÍTULO I. Introducción

La mensajería de texto a través de teléfonos celulares o dispositivos móviles no sólo brinda la posibilidad de comunicarnos con otras personas desde cualquier lugar y en cualquier momento, sino que además permite tener al alcance diversos tipos de información.

En los últimos años los volúmenes de la mensajería de texto SMS (Short Messaging System) han venido incrementándose considerablemente, lo que ha dado pie a que empresas comos ALS Comunicaciones que prestan servicios de valor agregado a través de este tipo de mensajería, se vean en la necesidad de ampliar su plataforma, y buscar nuevas maneras de tener un mayor control en la administración de sus envíos y recepción de mensajes.

A través de los distintos capítulos de este Trabajo Especial de Grado se muestra como se llevó acabo esta ampliación de la plataforma de ALS Comunicaciones, desde la propuesta, el estudio de las herramientas necesarias para lograrlo, el análisis, diseño y desarrollo del nuevo sistema.

1.1 Planteamiento del Problema

Inicialmente, la empresa ALS Comunicaciones, contaba con una aplicación del tipo *standalone*, que se ejecutaba en distintos PCs, la cual, mediante una conexión Infrarroja (IrD) establecía una comunicación entre este equipo y un celular, permitiendo así, realizar envíos de mensajes cortos de texto a través de los mismos. El principal problema de esta solución, era que frecuentemente los celulares perdían la comunicación con el PC, por diversas razones, o simplemente se apagaban puesto que las baterías o los cargadores se dañaban, generando retraso en la línea de producción.

La empresa, debido a estos inconvenientes, actualizó el proceso de envío de mensajes, y cuenta actualmente con una plataforma para llevarlo a cabo mediante un módem GSM. Esta consiste en una interfaz de comunicación, mediante una aplicación

Java, entre un computador y un dispositivo externo GSM (Global System For Mobile Communications) para la administración, envío y recepción de mensajes de texto SMS.

Esta plataforma fue implementada bajo Java, dado que por el manejo de hilos, proporcionado por el lenguaje, se puede realizar el envío y la recepción de mensajes de forma simultánea, además de múltiples facilidades para conectividad con bases de datos, así como también con dispositivos celulares.

La conexión que se establece con la base de datos, es fácil de manipular dada las bondades de las conexiones JDBC (Java Database Connectivity). La comunicación entre el dispositivo externo GSM y el computador se lleva a cabo mediante una conexión serial, utilizando el API (Application Programming Interface) de comunicación de Java y el envío de comandos AT, que permiten gestionar el envío y recepción de SMS.

Con la implementación de este sistema, la empresa, incrementó su productividad, sin embargo, recientemente la misma ha crecido de manera considerable y esta solución no le es suficiente para prestarle el servicio a todos sus clientes, puesto que necesita tener un mayor control en los estatus de los mensajes enviados, es decir, si el mensaje ha sido entregado, está pendiente, etc.

Por ello, se propone el desarrollo de un sistema que complementará la plataforma actual, basado en el protocolo SMPP (Short Message Peer to Peer), con el cual se podrán manejar volúmenes mayores de mensajes, y adicionalmente, se tendrá la posibilidad de evaluar el estado de los mensajes, para garantizar la entrega efectiva de los mismos.

1.2 Objetivos

Objetivo General: desarrollar una aplicación que realice, a través del protocolo estándar de clase mundial SMPP, así como también, mediante la plataforma GSM, el envío y recepción de mensajes de texto, e igualmente sea capaz de administrar y registrar estos eventos, para su posterior publicación en un ambiente web.

Objetivos Específicos:

- ✓ Levantar los requerimientos de ALS Comunicaciones, para establecer los lineamientos de la aplicación.
- ✓ Investigar sobre el protoco SMPP.
- ✓ Realizar el diseño lógico de la aplicación.
- ✓ Establecer las herramientas y tecnologías a utilizar.
- ✓ Implementar las funcionalidades contempladas para la aplicación.
- ✓ Integrar el nuevo sistema con la plataforma GSM existente.
- ✓ Realizar las pruebas pertinentes para el sistema.

1.3 Justificación

Con el desarrollo de esta investigación se logrará:

- ✓ Ampliar la plataforma de operaciones de ALS Comunicaciones, permitiendo así brindar un mejor servicio a los clientes.
- ✓ Se reducirán los factores adversos que afectan actualmente al proceso de envío y recepción de mensajes, trayendo como consecuencia una mayor productividad.
- ✓ Dar pie a que se incorporen nuevos servicios a la cartera actual de la empresa, tales como servicios interactivos, entre otros.
- ✓ Aportar conocimientos teórico-práctico para investigaciones que involucren el estudio del protocolo SMPP.

1.4 Distribución del Documento

Este trabajo se encuentra dividido en 7 capítulos, los cuales se describen a continuación:

Capítulo 2 – Marco Conceptual: en este capítulo se definen funciones, características y conceptos referentes a SMS y SMPP.

Capítulo 3 – Metodología y Herramientas de Desarrollo: en este capítulo se describen las tecnologías y herramientas que se utilizaron para el desarrollo de la aplicación.

Capítulo 4 – Análisis y Diseño de la Aplicación: este capítulo presenta el estudio de la plataforma utilizada anteriormente por la empresa, así como también, los diagramas involucrados en el diseño de la nueva.

Capítulo 5 – Implementación: en este capítulo se describen los detalles de la implementación del nuevo sistema, así como también las herramientas utilizadas para llevar a cabo ésta.

Capítulo 6 – Fase de Pruebas: se presentan los tipos de pruebas que se llevaron a cabo para verificar el correcto funcionamiento de la aplicación.

Capítulo 7 – Conclusiones: se muestran las contribuciones de la plataforma desarrollada así como también las recomendaciones para trabajos futuros.

CAPÍTULO II. Marco Conceptual

2.1 Sistema De Mensajería Corta (SMS: Short Messaging System)

El Sistema de Mensajería Corta (SMS: Short Messaging System) es un servicio inalámbrico aceptado globalmente que permite el envío y recepción de mensajes alfanuméricos entre dos usuarios de un sistema de telefonía móvil, además del envío y recepción de mensajes alfanuméricos entre un suscriptor móvil y sistemas externos, como es el caso del correo electrónico, servicios de *paging*.

2.1.1 Mensajes de Texto en los Sistemas Celulares

La consolidación de las redes digitales de telefonía permitió servicios de ámbito internacionales además de soportar más de una tecnología inalámbrica. Esto originó la aparición de la demanda de nuevo servicios. Para satisfacer estos requerimientos se creó el Centro de Servicios de Mensajería Corta (SMSC: Short Messaging Service Center) basado en sistemas de Redes Inteligentes (IN: Intelligent Networks) [1].

Los usuarios del servicio hacen uso del SMSC: Short Messaging System Center como un sistema de almacenamiento y envío de sus mensajes de texto. La red inalámbrica provee los mecanismos requeridos para encontrar a la estación destino, y transporta el mensaje desde el SMSC hasta la misma.

En contraste con otros sistemas de transmisión de texto, los elementos del servicio de mensajería corta están diseñados para garantizar el envío seguro de los mensajes hasta su destino. Adicionalmente, SMS soporta varios mecanismos de entrada que permiten la interconexión de diferentes fuentes de mensajes con diferentes destinos, es decir, que pertenezcan a diferentes redes móviles de distintos operadores.

Una característica distintiva de este servicio es que cualquier suscriptor móvil puede recibir o enviar un mensaje corto de texto en cualquier momento, independientemente si

una llamada o una transmisión de datos está en progreso (en algunas implementaciones, dependiendo de las capacidades de la operadora). SMS también garantiza el envío de los mensajes por la red. Si se produce alguna falla en el MSC (Mobile Switching Center) destino, el mensaje es almacenado en el SMSC (Short Messaging Service Center) hasta que el móvil destinatario esté nuevamente disponible. SMS se caracteriza por el envío de paquetes fuera de banda y transferencia de mensajes en un bajo ancho de banda, lo que resulta en una manera muy eficiente de transmitir ráfagas de paquetes cortos de datos. Con el desarrollo de la tecnología, una variedad de servicios fueron introducidos, incluyendo correo electrónico, fax, paging integrado, banca interactiva, servicios de información como revisión de stock, y aplicaciones de integración con Internet [1].

2.1.2 Elementos de Red y Arquitectura

En esta sección se presentan los elementos que conforman un Sistema de Mensajería Corta de Texto y cómo éstas se relacionan entre si para proveer la funcionalidad del sistema [2].

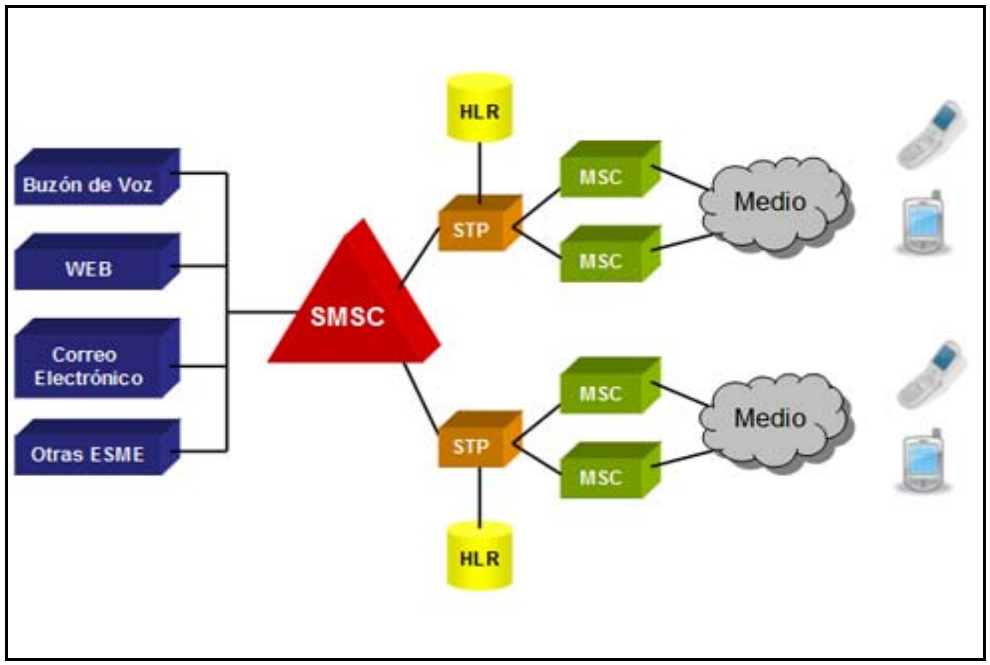

Figura 01. Arquitectura Básica de un Sistema de Mensajería Corta

2.1.2.1 Entidades Externas al Sistema (ESME: External Short Messaging Entities)

Una ESME es un elemento que puede recibir o enviar un mensaje de texto. La entidad externa (ESME: External Short Message Entity) puede estar localizada en una red fija, en otro centro de servicio, o puede ser un dispositivo móvil, tales como: buzón de voz, servicios web, correo electrónico, entre otros [1].

2.1.2.2 SMSC (Short Message System Center)

El SMSC es una combinación de hardware y software responsable de recibir, almacenar y enviar mensajes cortos de texto entre las entidades de mensajería corta, ESME y dispositivos móviles. El SMSC debe tener una alta confiabilidad, capacidad para suscriptores, y eficiencia en mensajería. Debe ser escalable para poder manejar la creciente demanda de suscriptores SMS en la red.

Otro factor a considerar es la facilidad de operación y mantenimiento de las aplicaciones, además de la flexibilidad para activar nuevos servicios y actualizaciones de software [1].

2.1.2.3 Punto de Transferencia de Señal (STP: Signal Transfer Point)

El STP es un elemento de red normalmente disponible en desarrollos de IN que permite conexiones IS-41, el cual es un estándar para identificar y autenticar usuarios además de enrutar llamadas en redes telefónicas móviles [3], sobre enlaces del Sistema de Señalización 7 (SS7), con múltiples elementos de la red [1]

2.1.2.4 Registro de Suscriptores Locales (HLR: Home Location Register)

El HLR es una base de datos usada para el permanente almacenamiento y manejo de los perfiles de usuario. El HLR provee la información de enrutamiento para el usuario indicado. Inclusive si la estación destino no está disponible cuando el mensaje ha sido

enviado. El HLR informa al SMSC cuando la estación destino es reconocida por la red como accesible y entonces se envía el mensaje [1].

2.1.2.5 Registro de Suscriptores Visitantes (VLR: Visitor Location Register)

Esta es una base de datos que contiene información temporal acerca de los suscriptores domiciliados en un HLR que se encuentran ingresando a otro HLR. Esta información es requerida por el MSC para dar servicio a los suscriptores visitantes [1].

2.1.2.6 Centro de Conmutación Móvil (MSC: Mobile Switching Center)

El MSC realiza las funciones de conmutación del sistema y controla las llamadas entre los dispositivos móviles y los sistemas de datos. El envía los mensajes cortos al suscriptor móvil específico a través de la estación base apropiada [1].

2.1.2.7 El Medio (Air Interface)

El medio está definido en todas las tecnologías inalámbricas como el rango de frecuencias utilizado para transmitir y recibir las señales de voz y datos desde el MSC hasta los dispositivos móviles [1].

2.1.2.8 Las Estaciones Bases (BS: Base Stations)

Todas las funciones relacionadas con la transmisión de las señales electromagnéticas de radio entre el MSC y los dispositivos móviles, son efectuadas por las estaciones base.

Las BS consisten en controladores y transceivers también conocidos como sitios de celda o simplemente celdas. El Base Station Controller (BSC) puede controlar a una o más estaciones base y está a cargo del manejo de sus propios recursos cuando un suscriptor se mueve de un sector de la celda a otro, sin importar si este nuevo sector es limítrofe con otras celdas [1].

2.1.2.9 El Dispositivo Móvil (MD: Mobile Device)

Es el terminal capaz de recibir y originar los mensajes de textos. Comúnmente estos dispositivos son teléfonos celulares digitales pero recientemente estas capacidades han sido dadas a otros dispositivos como PDAs y computadoras de mano; la infraestructura inalámbrica está basada en señalización SS7. SMS hace uso de la Mobile Application Part (MAP), la cual define los métodos y mecanismos de la comunicación en redes inalámbricas que emplean SS7 y sus capacidades de Transacción (TCAP: Transactional Capabilities Application Part). La capa de servicio SMS hace uso de MAP y TCAP para permitir la transferencia de mensajes cortos entre entidades iguales. Las capacidades de las terminales varían dependiendo de la tecnología inalámbrica soportada por la terminal [1].

2.1.2.10 Elementos de señalización

La MAP (Mobile Application Part) define las operaciones necesarias para soportar SMS. Los estándares Americanos e Internacionales han definido la MAP usando los servicios de SS7 TCAP. El estándar Americano es publicado por la TIA (Telecommunications Industry Association) y es referido como IS-41. El estándar internacional es definido por la ETSI: European Telecommunications Standards Institute y se denomina como GSM MAP [1].

Las siguientes operaciones básicas MAP son requeridas para proveer el servicio SMS de extremo a extremo [1]:

- ✓ Petición de Información de Enrutamiento.- Antes de intentar enviar el mensaje, el SMSC debe recibir la información de enrutamiento para determinar el MSC correspondiente al dispositivo móvil destino. Esto se realiza a través de una petición del HLR del móvil destino, lo cual se efectúa mediante el uso del mecanismo: SMSrequest y SendRoutingInfoForShortMsg en IS-41 y GSM respectivamente.
- ✓ Envío de Mensajes Punto a Punto.- Después que la dirección del MSC ha sido obtenida del HLR de la estación, la operación de envío provee un servicio confirmado de entrega. La operación trabaja en conjunto con la estación base mientras el mensaje está siendo llevado desde la MSC a la MS. Esta operación

es realizada mediante el uso de los mecanismos: short message delivery–point-to-point (SMD–PP) y forwardShortMessage mechanisms en IS–41 y GSM, respectivamente.

- ✓ Indicación de Espera.- Esta operación se realiza cuando la operación de envío falla debido a ciertas causas como cuando el móvil destino no ha sido registrado; y provee la capacidad de que el HLR notifique al SMSC cuando el móvil indicado está nuevamente disponible. Esto se realiza mediante los siguientes mecanismos: SMS_notification indicator y set_message_waiting_data en IS–41 y GSM, respectivamente.
- ✓ Alerta del Centro de Servicio.- Permite que el HLR informe al SMSC que el móvil está ya reconocido como disponible. Esto se logra mediante los mecanismo de: SMS_notification y alert_service_center en IS–41 y GSM, respectivamente.

2.1.2.11 Elementos de Servicio

El sistema SMS está compuesto de varios elementos de servicio involucrados en el envío y recepción de mensajes cortos [1]:

- ✓ Expiración de mensaje.- El SMSC almacenará y reintentará el envío de los mensajes de receptores inalcanzables hasta que la entrega se haya completado satisfactoriamente o hasta que el tiempo de expiración configurado se haya cumplido.
- ✓ Prioridad.- Este es un elemento de información proveído por un SME que indica la urgencia del mensaje y lo diferencia de los mensajes con prioridad normal.
- ✓ Intercalación de mensajes.- El SMSC almacena los mensajes por un período no mayor al tiempo de expiración (se asume que el tiempo de intercalación es menor que el tiempo de expiración asociado al mensaje), y después de que el tiempo de intercalación expira, el mensaje será enviado a un sistema alternativo como una red paging o un servidor de correo.

Además, SMS provee una etiqueta de tiempo que reporta el tiempo de llegada del mensaje al SMSC y un indicador sobre la existencia o no de más mensajes para enviar. [1]

2.1.2.12 Servicios al Suscriptor

SMS provee dos servicios básicos punto a punto [1]:

- ✓ Mensaje Corto Originado en el móvil (Mobile-originated short message MO–SM).
- ✓ Mensaje Corto Terminado en el móvil (Mobile-terminated short message MT–SM).

El mensaje corto MO es transportado desde el dispositivo móvil hasta el SMSC y puede ser enviado a otro suscriptor móvil o para suscriptores de servicios de paging o para usuarios de una red IP.

El mensaje corto MT es transportado desde el SMSC hasta el dispositivo móvil y puede ser enviado desde el SMSC por otro suscriptor móvil o por otras fuentes como óbice-mail, paging, etc.

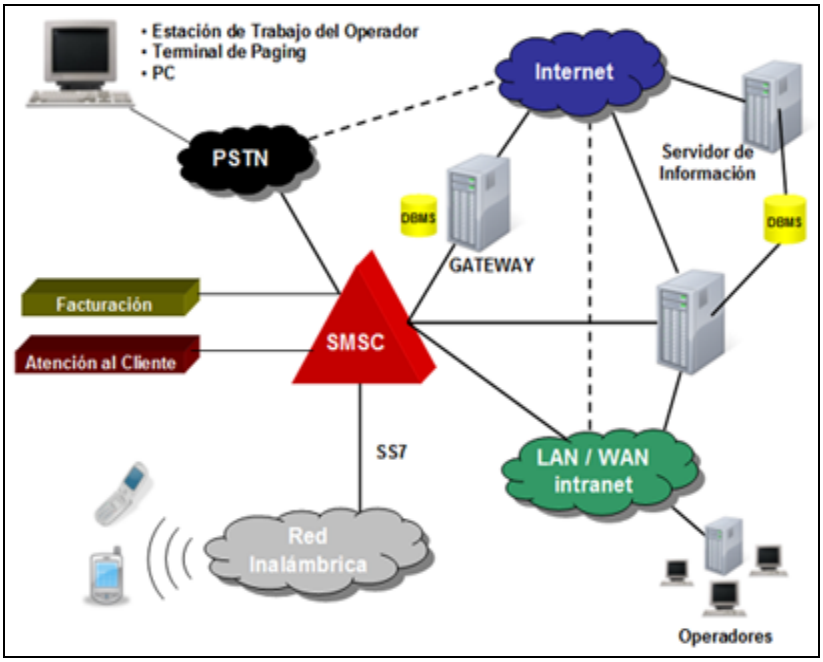

Figura 02. Infraestructura de Red

2.1.3 Protocolo SMS

El stack del protocolo SMS, está compuesto por cuatro capas: Capa de Aplicación, Capa de Transferencia, Capa de Retransmisión, Capa de Conexión.

La Capa de Aplicación es implementada en las ESME como softwares de aplicación que envían, reciben e interpretan el contenido de los mensajes (ej.: editor de mensajes, juegos, etc.). La Capa de Aplicación también se conoce como SM-AL (Short-Message-Applicaction-Layer).

En la Capa de Transferencia, el mensaje es considerado como una secuencia de octectos que contienen información tales como, longitud del mensaje, remitente del mensaje, fecha de recepción, etc. La Capa de Transferencia también es conocida como SM-TL (Short-Message-Transfer-Layer).

La Capa de Retransmisión, permite el transporte del mensaje entre varios elementos de red. Un elemento de red puede almacenar temporalmente un mensaje si el próximo elemento al cual se debe pasar el mensaje no está disponible. En la Capa de Retransmisión, el MSC lleva a cabo dos funciones, adicionales a sus capacidades de enrutamiento. La primera, llamada SMS-GMSC (SMS Gateway SMS) consiste en recibir el mensaje desde un SMSC y consultar al HLR para obtener la información de enrutamiento para hacer entrega la red destino. La segunda, llamada SMS-IWMSC (SMS InterWorking MSC), consiste en recibir un mensaje desde una red móvil y pasarlo al SMSC pertinente. La Capa de Retransmisión también se conoce como SM-RL (Short-Message-Relay-Layer).

La Capa de Conexión permite la transmisión del mensaje a nivel físico. Por este sentido, el mensaje está preparado para lidiar con canales de error de bajo nivel. La Capa de Conexión también es conocida como SM-LL (Short-Message-Link-Layer).

El stack de capas de protocolo de transporte para SMS se muestra en la Figura 03. Con fines de transporte, una aplicación mapea el contenido del mensaje y las instrucciones de envío dentro de una Unidad de Data de Protocolo de Transferencia - TPDU (Transfer

Protocol Data Unit) en la Capa de Transferencia. Una TPDU está compuesta por varios parámetros que indican el tipo de mensaje, especifican si se requiere o no un reporte de estátus, el texto correspondiente al mensaje en si, etc. Cada parámetro tiene el prefijo TP por Protocolo de Transferencia (Transfer Protocol) tales como TP-MTI, indicador de tipo de mensaje (TP-Message-Type-Indicator), TP-SRI, indicador de reporte de estátus (TP-Status-Report-Indicator), TP-UD, data de usuario (TP-User-Data), etc [5].

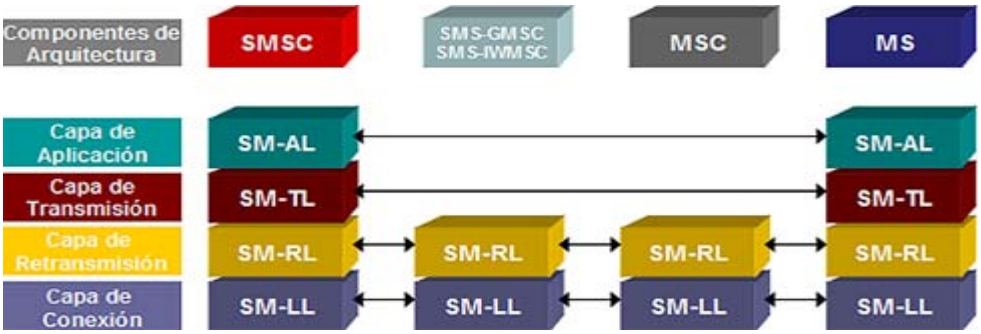

Figura 03. Stack del Protocolo SMS

2.2 Protocolo de Mensajes Cortos en Redes Punto a Punto (Short Message Peer to Peer, SMPP)

El Protocolo de Mensajes Cortos en Redes Punto a Punto (Short Message Peer to Peer, SMPP) es un protocolo estándar abierto, diseñado para proporcionar una interfaz flexible en la comunicación de datos para la transmisión de mensajes cortos, entre un centro de mensajes, como un centro de servicios de mensajes cortos (Short Message Service Center, SMSC), Servidor GSM de servicio de datos suplementarios no estructurados (GSM Unstructured Supplementary Services Data, USSD, Server), u otro tipo de centro de mensajes SMS y un sistema de aplicación, como un servidor proxy WAP, EMail Gateway u otros Gateways de mensajería.

Usando el protocolo SMPP, una aplicación SMS (ESME) puede iniciar una conexión de capa de aplicación sobre una red TCP / IP o una conexión de red X.25 y podrá enviar y recibir mensajes cortos hacia y desde el SMSC, respectivamente. La ESME también podrá consultar, cancelar o sustituir mensajes cortos usando SMPP [4].

SMPP soporta un completo conjunto de funciones de la mensajería de dos vías, tales como [4]:

- ✓ Transmitir mensajes desde una ESME a uno o múltiples destinos a través del SMSC.
- ✓ Una ESME puede recibir mensajes a través del SMSC desde otras SME (por ejemplo, estaciones móviles).
- ✓ Consultar el estado de un mensaje corto almacenado en el SMSC.
- ✓ Cancelar o sustituir un mensaje corto almacenado en el SMSC.
- ✓ Envío de un mensaje corto registrado (para que el SMSC devuelva una notificación de entrega a quien originó el mensaje).
- ✓ Programar la fecha y la hora para la transmisión de un mensaje corto.
- ✓ Seleccionar el modo del mensaje, es decir, datagrama o almacenar y reenviar.
- ✓ Establecer la prioridad de la entrega del mensaje corto.
- ✓ Definir el tipo de codificación de los datos del mensaje corto.
- ✓ Establecer el período de valides del mensaje corto.
- ✓ Asociar un tipo de servicio con cada mensaje, por ejemplo, notificación de correo de voz.

2.2.1 Descripción del Protocolo SMPP

SMPP, es un protocolo abierto de transferencia de mensajes que permite que entidades de mensajería corta (SMEs) que se encuentran fuera de la red móvil puedan conectarse con un SMSC. Las entidades no móviles que envían mensajes a, o reciben mensajes de, un SMSC son conocidas como entidades externas de mensajería corta (External Short Message Entities - ESME) [4].

El Protocolo SMPP define [4]:

- ✓ Un conjunto de operaciones para el intercambio de mensajes de texto entre una ESME y un SMSC.
- ✓ Los datos que una aplicación ESME debe intercambiar con un SMSC durante las operaciones SMPP.

Los suscriptores de una red celular con capacidad de SMS pueden recibir mensajes cortos en una estación móvil (Mobile Station - MS) de una o más ESMEs. La manera en que estos mensajes llegan a la ESME a través de una interfaz distinta de SMPP está más allá del alcance de esta investicación. Sin embargo, ejemplos de este tipo de aplicaciones ESME, incluyen [4]:

- ✓ Alertas de voz procedente de un VPS (Sistema de procesamiento de voz), indicando mensajes de voz en el buzón de correo de un cliente.
- ✓ Servicios de paginación numéricos y alfanuméricos.
- ✓ Los servicios de información. Por ejemplo, una aplicación que permite a los suscriptores móviles consultar la tasa de cambio o compartir la información de los precios de una base de datos o de la WWW y se le muestra como un mensaje corto, en su celular.
- ✓ Llamadas directamente discadas o desviadas al operador de un centro de mensajes (message-bureau), que reenvía el mensaje al SMSC, para su posterior envío.
- ✓ Una aplicación de manejo de flotas que permite que una estación central use un SMSC, para determinar la ubicación de sus vehículos de servicio y notificar al más cercano de una demanda de servicio en su área.
- ✓ Aplicaciones de telemetría. Por ejemplo, un medidor que transmite un mensaje corto al sistema de facturación de una compañía de servicios públicos, para almacenar automáticamente el uso del cliente.
- ✓ Servidor Proxy WAP. Un servidor Proxy WAP actúa como la puerta de enlace WAP para las aplicaciones de Internet inalámbricas. Un servidor Proxy WAP puede seleccionar entre un SMS ó USSD portador para enviar datagramas WDP a, y recibir datagramas WDP de una estación móvil.

2.2.1.1 Definición del Protocolo SMPP

SMPP se basa en el intercambio de peticiones y respuestas de las unidades de datos del protocolo (PDUs) entre las ESMEs y el SMSC sobre TCP/IP ó una conexión de red X.25. El protocolo SMPP define [4]:

- ✓ Un conjunto de operaciones y las unidades de datos del protocolo (PDUs) asociadas, para el intercambio de mensajes cortos entre una ESME y un SMSC.
- ✓ Los datos que una aplicación ESME puede intercambiar con un SMSC durante las operaciones SMPP.

Cada operación SMPP debe consistir en una petición PDU y una respuesta PDU. El receptor debe devolver la respuesta SMPP asociada a una petición SMPP. La única excepción a esta regla es el PDU de alerta de notificación para el cual no hay ninguna respuesta. [4]

El intercambio de mensajes entre una ESME y SMSC vía SMPP se puede categorizar bajo tres grupos distintos de transacciones, como sigue:

- ✓ Mensajes enviados desde la ESME (transmisor) al SMSC.
- ✓ Mensajes enviados del SMSC la ESME (Receptor).
- ✓ Mensajes enviados desde la ESME (Transceptor) al SMSC y de los mensajes enviados desde el SMSC a la ESME (Transceptor).

La Figura 04 ilustra las categorías anteriores que se explican en más detalle en las secciones subsecuentes.

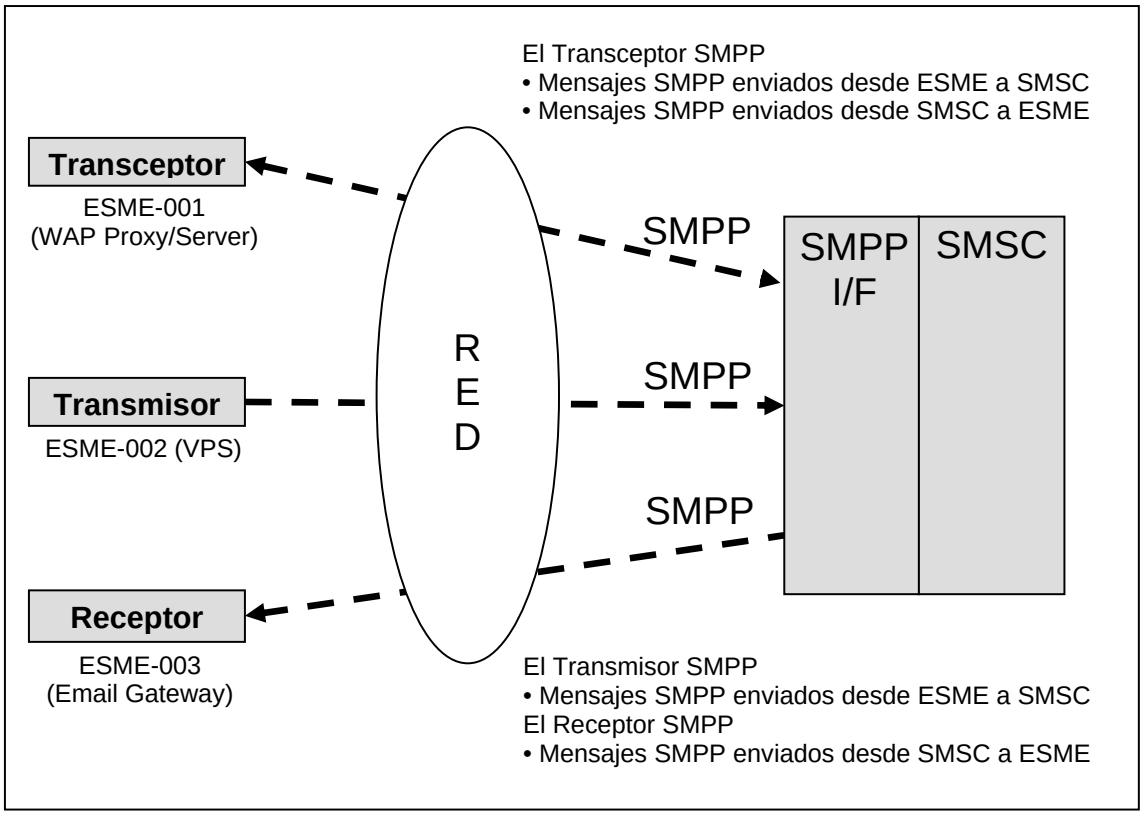

Figura 04. Interfaz SMPP entre SMSC y ESMEs

2.2.1.2 Descripción de la Sesión SMPP

Una sesión SMPP entre un SMSC y una ESME, es iniciada por la ESME, primero, estableciendo una conexión de red con el SMSC y, a continuación, emite una petición de enlace SMPP para abrir una sesión SMPP; una ESME que desea enviar y recibir mensajes, necesita establecer dos conexiones de red (TCP/IP ó X.25) y dos sesiones SMPP (Transmisor y Receptor), alternativamente, según la versión del protocolo una ESME puede establecer una sesión SMPP Transceptor sobre una sola conexión de red.

Durante una sesión SMPP, una ESME puede emitir una serie de peticiones a un SMSC y recibirá las respuestas apropiadas a cada demanda, del SMSC. Igualmente, el SMSC puede emitir las peticiones SMPP a la ESME, que debe responder de forma adecuada [4].

La sesión SMPP puede definirse en término de los siguientes estados posibles [4]:

- ✓ **OPEN (Conectado y enlace pendiente)**

Una ESME ha establecido una conexión de red al SMSC pero no ha emitido una perición de enlace todavía.

- ✓ **BOUND_TX**

Una ESME conectada solicitó enlazarse como Transmisor ESME (emitiendo un bind_transmitter PDU) y ha recibido una respuesta del SMSC autorizando la solicitud de enlace.

Una ESME enlazada como transmisor puede enviar los mensajes cortos a un SMSC para su posterior entrega a una Estación Móvil o a otro ESME. La ESME también puede reemplazar, consultar o cancela un mensaje corto previamente enviado.

- ✓ **BOUND_RX**

Una ESME conectada solicitó enlazarse como un Receptor ESME (emitiendo un bind_receiver PDU) y ha recibido una respuesta del SMSC autorizando la solicitud de enlace.

Una ESME enlazada como un receptor puede recibir los mensajes cortos de un SMSC que pueden originarse por una estación móvil, por otro ESME o por el propio SMSC (por ejemplo un SMSC, acuse de recibo).

- ✓ **BOUND_TRX**

Una ESME conectada solicitó enlazarse como un Transceptor ESME (emitiendo un bind_transceiver PDU) y ha recibido una respuesta del SMSC autorizando la solicitud del enlace. Una ESME enlazado como transceptor soporta el conjunto completo de operaciones soportado por un Transmisor ESME y un Receptor ESME.

Un Transceptor ESME puede enviar mensajes cortos a un SMSC para su posterior entrega a una Estación Móvil o a otro ESME. La ESME también puede recibir los mensajes cortos de un SMSC que puede originarse por una Estación Móvil, por otro ESME o por el propio SMSC.

- ✓ **CLOSED (Desenlazado y Desconectado)**

Una ESME se ha desenlazado y ha cerrado la conexión de red. El SMSC puede también desenlazarse de la ESME.

2.2.2 Unidad de Datos del Protocolo SMPP

La siguiente tabla muestra el conjunto de PDUs SMPP y el contexto en que cada PDU puede utilizarse [4]:

Nombre PDU SMPP	Estado de Sesión SMPP Requerido	Emitido por ESME	Emitido por SMSC
bind_transmitter	OPEN	SI	NO
bind_transmitter_resp	OPEN	NO	SI
bind_receiver	OPEN	SI	NO
bind_receiver_resp	OPEN	NO	SI
bind_transceiver	OPEN	SI	NO
bind_transceiver_resp	OPEN	NO	SI
outbind	OPEN	NO	SI
unbind	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI
unbind_resp	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI
submit_sm	BOUND_TX	SI	NO
	BOUND_TRX	SI	NO
submit_sm_resp	BOUND_TX	NO	SI
	BOUND_TRX	NO	SI
submit_sm_multi	BOUND_TX	SI	NO
	BOUND_TRX	SI	NO
submit_sm_multi_resp	BOUND_TX	NO	SI
	BOUND_TRX	NO	SI

Tabla 1. Listado resumido de PDUs SMPP

Nombre PDU SMPP	Estado de Sesión SMPP Requerido	Emitido por ESME	Emitido por SMSC
data_sm	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI
data_sm_resp	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI
deliver_sm	BOUND_RX	NO	SI
	BOUND_TRX	NO	SI
deliver_sm_resp	BOUND_RX	SI	NO
	BOUND_TRX	SI	NO
query_sm	BOUND_RX	SI	NO
	BOUND_TRX	SI	NO
query_sm_resp	BOUND_RX	NO	SI
	BOUND_TRX	NO	SI
cancel_sm	BOUND_TX	SI	NO
	BOUND_TRX	SI	NO
cancel_sm_resp	BOUND_TX	NO	SI
	BOUND_TRX	NO	SI
replace_sm	BOUND_TX	SI	NO
replace_sm_resp	BOUND_TX	NO	SI
enquire_link	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI
enquire_link_resp	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI
alert_notification	BOUND_RX	NO	SI
	BOUND_TRX	NO	SI
generic_nack	BOUND_TX	SI	SI
	BOUND_RX	SI	SI
	BOUND_TRX	SI	SI

Tabla 1 (continuación). Listado resumido de PDUs SMPP.

2.2.3 Conexiones de la Capa de Red

El transporte entre el SMSC y ESME puede basarse en una red TCP / IP o conexión de red X.25. SMPP es un protocolo de nivel de aplicación y no ofrece la funcionalidad de transporte. Por lo tanto supone, que la conexión de red subyacente facilitará la transferencia

de datos confiable punto a punto, incluido la codificación de paquetes, control de flujo, ventanas deslizantes y gestión de errores.

Así, en SMPP, la ESME y SMSC deben tratar a la conexión de red como un transporte fiable que gestiona la entrega y recepción de los PDUs SMPP.

En la siguiente figura se ilustra una aplicación genérica SMPP que sirve de interfaz entre una ESME y SMSC.

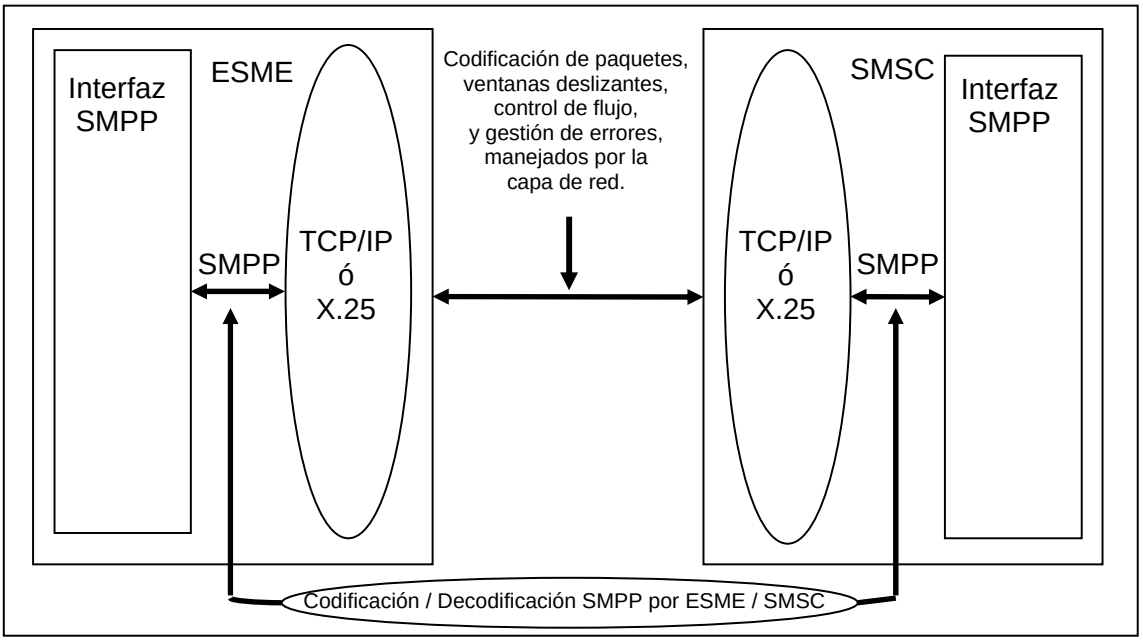

Figura 05. Modelo de la interfaz entre SMPP y ESME-SMSC.

Si es necesario, la capa de red de la entidad emisora se encargará de la fragmentación de los PDUs SMPP para su transmisión como una serie de paquetes fragmentados a través de la red de conexión. Del mismo modo, la capa de red de la entidad receptora, volverá a reunir el PDU fragmentado antes de pasar todo el PDU SMPP a la capa SMPP [4].

2.2.4 Manejo de Errores SMPP

Todas las operaciones SMPP consisten en un PDU de solicitud y un PDU de respuesta asociado, a excepción del PDU de alert_notification, para el cual no hay un PDU de respuesta.

En todos los otros casos, la entidad receptora debe retornar un PDU SMPP de respuesta asociado a un PDU SMPP de solicitud, indicando que el PDU fue recibido por el destinatario. Mientras que la entidad que genero la solicitud no recibe una respuesta, debe asumir que el PDU no ha sido recibido por la entidad destino.

En el momento que el PDU SMPP de solicitud contiene un error, la entidad receptora debe retornar una respuesta con el código de error apropiado en el campo command_status de la cabecera del PDU de respuesta. Si la entidad receptora encuentra un error en la cabecera del PDU, esta debe retornar el PDU de generic_nak a la entidad que originó el mensaje [4].

2.2.5 Formato de las Unidades de Datos del Protocolo SMPP – Visión General

El formato general de la PDU SMPP consiste de una cabecera seguida de un cuerpo, como se muestra en la siguiente tabla [4].

PDU SMPP				
Cabecera (obligatoria)				**Cuerpo (opcional)**
longitud_ comando	*id_comando*	*estatus_comando*	*numero_secuencia*	*cuerpo_pdu*
4 octetos	Longitud = (longitud de comando - 4) octetos			

Tabla 2. Formato de la SMPP – Visión General

La cabecera SMPP es obligatoria para cada PDU SMPP y siempre debe estar presente. El cuerpo es opcional y puede no estar incluido en cada PDU SMPP.

2.2.5.1 Diseño de PDU SMPP

La siguiente tabla muestra la descripción del formato de PDU SMPP [4]:

	Campo	Tamaño Octetos	Tipo	Descripción
CABECERA	*longitud_comando*	4	Entero	Define la longitud total en octetos del paquete PDU SMPP incluyendo la longitud del campo.
	id_comando	4	Entero	Identifica el PDU SMPP en particular, (***submit_sm, query_sm,*** etc.) Un identificador de comando único es destinado en cada PDU de solicitud SMPP en el rango: 0x00000000 a 0x000001FF Un indentificador único de comando también es destinado en cada PDU de respuesta SMPP en el rango: 0x80000000 a 0x800001FF
	estatus_comando	4	Entero	Identifica el éxito o fallo de un requerimiento SMPP. Es relevante únicamente en el PDU de respuesta SMPP, y debe contener solamente un valor NULL en un PDU de solicitud SMPP.
	número_secuencia	4	Entero	Este campo contiene un número de secuencia que permite ser asociado a solicitudes y respuestas con propósito de mantener un correlativo. El uso de números de secuencia para llevar un correlativo de mensajes permite a los PDU SMMP ser intercambiados asíncronamente. Asingar el *número_secuencia* es responsabilidad del que origina el PDU SMPP. El *número_secuencia* debe ser incrementado de manera monótona para cada PDU de solicitud SMPP y debe ser preservado en el PDU de respuesta SMPP asociado. El rango del *número_secuencia* es de: 0x00000001 a 0x7FFFFFFF.
CUERPO	*Parámetros Obligatorios*	Var.	Mix.	Lista de parámetros obligatorios correspondientes al PDU SMPP definido en el campo *id_comando*.
	Parámetros opcionales	Var.	Mix.	Lista de parámetros opcionales correspondientes al PDU SMPP definido en el campo *id_comando* y son incluidos como sean requeridos.

Tabla 3. Descripción del formato de PDU SMPP

2.2.5.2 Longitud PDU SMPP

El campo longitud_comando al comienzo de la cabecera de la PDU SMPP, indica el número total de octetos contenidas en dicha PDU. Este campo contiene 4 octetos enteros, transmitidos en formato Big Endian.

Para decodificar un PDU SMPP, la ESME o SMSC debe primero leer el campo longitud_comando (4 octetos) para determinar la longitud del PDU. La cantidad de data restante es entonces determinada con la resta de la longitud del campo longitud_comando (4 octetos) menos la longitud total de la PDU provista por el valor del campo longitud_comando. De esta manera, la extracción del valor de la longitud del comando de valor N indica que N-4 octetos están restantes para la PDU en cuestión [4].

2.2.5.3 Longitud del Mensaje SMPP y Longitud del Mensaje Extendido

La longitud del mensaje corto de texto (o data de usuario) es definida en el campo longitud_sm de los PDU SMPP: submit_sm, submit_multi_sm, deliver_sm and replace_sm.

La longitud máxima del mensaje a ser especificada en el campo longitud_sm es de 254 octetos. Si un ESME desea enviar un mensaje con una longitud mayor a 254 octetos, el campo longitud_sm debe estar en NULL y el parámetro opcional message_payload debe llenarse con la longitud del mensaje y la data de usuario.

La longitud del mensaje que puede ser enviado por un MS podría variar dependiendo de la red bajo la cual éste se encuentre operando [4].

2.2.6 Tipos y Definiciones de PDU SMPP

2.2.6.1 Definición de Tipos

La siguiente definición de tipo de data PDU SMPP es utilizada para definir los parámetros SMPP [4]:

Entero

Un valor sin signo con el número de octectos definidos.

Cadenas de C-Octetos

Una serie de caracteres ASCII que termina con el caracter NULL.

Cadenas de C-Octetos (Decimal)

Una serie de caracteres ASCII, donde cada caracter representa un digito decimal (0-9) y termina con el caracter NULL.

Cadenas de C-Octetos (Hex)

Una serie de caracteres ASCII, donde cada caracter representa un digito decimal (0-F) y termina con el caracter NULL.

Cadena de Octetos

Series de octetos, no necesariamente terminan con el carácter NULL.

2.2.6.2 Notación de Tamaños los Campos de Parametros SMPP

El siguiente estilo de notación es usado comúnmente. Nótese en la siguiente tabla que algunos tipos de cadenas SMMP son opcionales, mientras que otros son obligatorios .

Tamaño de Octetos	Tipo	Descripción del tipo de cadena especificada
4	Entero	Campo de enteros de tamaño fijo. En este ejemplo el tamaño del entero es de 32 bits (4 octetos)
Variable, Max 16	Cadena C-Octeto	Esta cadena es de longitud variable, entre 1 y 15 caracteres ASCII, seguida por un octeto que contiene el caracter NULL. Una cadena vacía se codifica como un solo octeto con el caracter NULL (0x00)
Fijo, 1 ó 17	Cadena C-Octeto	La cadena tiene dos posibles longitudes, 1 octeto que contiene el caracter NULL o un número fijo de caracteres que termina con el caracter NULL (en este ejemplo, 16 caracteres, más el caracter NULL)
Variable, 0 – 254	Cadena de Octetos	Campo de cadena de octetos con tamaño variable. En este ejemplo el tamaño de la cadena de octetos puede variar entre 0 y 254 caracteres.

Tabla 4. Notación de Cadena C-Octeto

2.2.6.3 Parámetros Opcionales

Los Parámetros Opcionales son campos, que pueden o no, estar presentes en un mensaje. Proveen la posibilidad de incorporar nuevos parámetros en un futuro, así como cuando se definan nuevas versiones del protocolo SMPP.

Los Parámetros Opcionales siempre deben aparecer al final de una PDU, en la sección "Parámetros Opcionales" de la PDU SMPP. Sin embargo, pueden ser incluido en el orden que convenga dentro de dicha sección.

Para una PDU SMPP en particular, la ESME o SMSC puede incluir algunos, todos o ninguno de los parámetros definidos, dependiendo de la aplicación en particular. Por ejemplo un sistema de paging podría sólo incluir el "número a contactar" como parámetro opcional en una operación submit_sm [4].

2.2.6.3.1 Formato de los Parámetros Opcionales

Todos los parámetros opcionales tienen el siguiente formato generar TLV (Tag-Etiqueta, Length-Longitud, Value-Valor) [4].

Nombre Parámetro	Tamaño	Tipo	Descripción
Etiqueta	2	Entero	Es usado como identificador único del parámetro opcional en cuestión. Este campo siempre tiene 2 octetos de longitud
Longitud	2	Entero	Indica la longitud del campo Valor en octetos. No incluye la longitud de los campos Etiqueta y Longitud Este campo siempre tiene 2 octetos de longitud
Valor	Variable	Variable	Contiene la data actual del parámetro opcional en cuestión

Tabla 4. Parámetros Opcionales de los PDU

2.2.7 Definición de los PDU SMPP

2.2.7.1 Operación "BIND"

El propósito de la operación Bind es el de registrar la instancia de una ESME en un SMSC y solicitar una sesión SMPP sobre la red en cuestión para poder enviar y recibir

mensajes. Por lo que la operación Bind puede verse como una solicitud de autenticación por parte del SMSC hacia la ESME con el fin de establecer una conexión.

Como se mencionó anteriormente, una ESME puede conectarse al SMSC bien sea como un Transmisor (denominado, Transmisor ESME), como Receptor (denominado, Receptor ESME), o como Transceptor (denominado, Transceptor ESME). Hay tres tipos de PDU bind SMPP, para soportar los distintos modos de operación, denominados bind_transmitter, bind_transceiver and bind_receive. El campo id_comando, especifica cual tipo se está usando.

Una ESME puede conectarse como Transmisor ESME y Receptor ESME usando operaciones bind_transmitter y bind_receiver por separado (habiendo establecido primero dos conexiones por separado a la red). Por otra parte, una ESME puede también conectarse como Transceptor habiendo establecido una única conexión a la red. Si un SMSC no soporta las operacioes bind_transmitter y bind_receiver, éste debe entonces retornar un mensaje de error “ID de Comando Inválido”, y la ESME debe reintentar hacer la conexión haciendo uso de la operación bind_transceiver. Similarmente si un SMSC no soporta el comando bind_transceiver debe retornar un mensaje “ID de Comando Inválido” y la ESME debe reintentar hacer la conexión usando las operacioes bind_transmitter y bind_receive [4].

Transmisor ESME

Una ESME conectado como Transmisor está autorizado para enviar mensajes cortos al SMSC y recibir las correspondientes respuestas SMPP del SMSC.

Receptor ESME

Una ESME conectado como Receptor está autorizado para recibir mensajes cortos del SMSC y retornar las correspondientes respuestas SMPP al SMSC.

Transceptor ESME

Una ESME conectado como Transceptor está autorizado para enviar mensajes cortos al SMSC y recibir las correspondientes respuestas del SMSC sobre una única sesión SMPP.

2.2.7.2 Operación "UNBIND"

El propósito de la operación SMPP Unbind es el de eliminar una instancia de una ESME del SMSC y notificar a éste que la ESME en cuestión no desea utilizar más la conexión en curso para el envío o recepción de mensajes. Por lo que la operación Unbind puede verse como una forma de solicitud de desconexión para cerrar la sesión SMPP [4].

2.2.7.3 PDU "GENERIC_NACK"

Se trata de un acuse de recibo negativo de un PDU SMPP enviado con una cabecera de mensaje inválida. Una respuesta generic_nack es retornada en los siguientes casos [4]:

- ✓ longitud_comando inválida

Si la entidad SMPP que recibe, detecta en la decodificación del PDU SMPP una longitud de comando inválida (bien sea muy corta o muy larga), esta debe asumir que se trata de data corrupta. En tales casos se devuelve un generic_nack al remitente del mensaje.

- ✓ id_comando desconocido

Si se recibe un ID de comando desconocido también se debe se devolver un generic_nack al remitente del mensaje.

2.2.7.4 Operación "SUBMIT_SM"

Esta operación es usada por una ESME para enviar un mensaje corto al SMSC para que sea transmitido a la entidad SME especificada. La PDU submit_sm no soporta el mensaje de modo de transacción [4].

2.2.7.5 Operación “DELIVER_SM”

La operación deliver_sm es usada por el SMSC para enviar un mensaje a una ESME. Usando este comando el SMSC puede enrutar el mensaje corto para que sea entregado a la ESME [4].

2.2.7.6 Operación “QUERY_SM”

Este comando es usado por la ESME para buscar el estátus de un mensaje corto que ya fue enviado [4].

2.2.7.7 Operación “CANCEL_SM”

La ESME hace uso de este comando para cancelar uno o más mensajes que ya han sido enviados pero que aún están pendientes para ser entregados. El comando puede especificar un mensaje en particular a cancelar, o todos los mensajes pertenecientes a una fuente, destino y tipo_servicio en específico [4].

2.2.7.8 Operación “ENQUIRE_LINK”

Este mensaje puede ser enviado por la ESME o por el SMSC y es utilizado para proveer una validación de la ruta de comunicación entre la ESME y el SMSC. La parte que recibe esta solicitud debe responder con un enquire_link_resp¸ con el fin de verificar que el nivel de conexión de la aplicación entre el SMSC y la ESME se encuentra en funcionamiento. La ESME puede también responder enviando cualquier primitiva SMPP válida [4].

2.2.7.9 Operación “ALERT_NOTIFICATION”

Este mensaje es enviado por el SMSC a la ESME, cuando ha detectado que un suscriptor móvil en particular está ahora disponible y una bandera de entrega pendiente ha sido levantada para dicho suscriptor por una operación data_sm previa [4].

CAPÍTULO III. Metodología y Herramientas de Desarrollo

3.1 Metodología

La metodología propuesta para el desarrollo de la aplicación es cíclica y consiste en iteraciones cortas de 4 pasos, que se relacionan entre si, esta se basa en el ciclo Shewhart [6]. En la Figura 06 se puede ver cómo está constituida la misma.

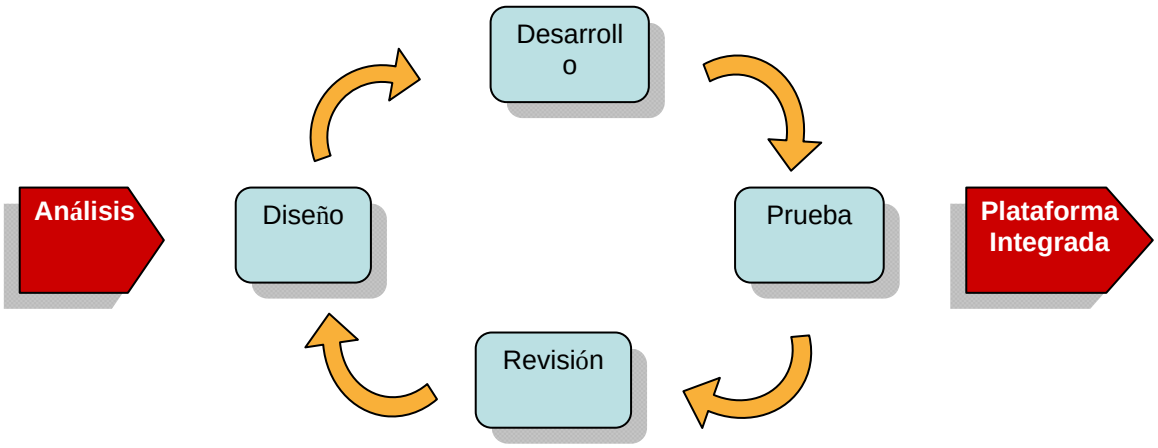

Figura 06. Metodología de Investigación

A continuación se presentan los detalles de la metodología.

1. **Diseño**: Partiendo del análisis realizado se elabora un diseño, que luego será utilizado como lineamiento en la etapa de desarrollo.

2. **Desarrollo**: Basado en el diseño se implementa la plataforma para ser probada en la siguiente etapa.

3. **Prueba**: Se realizan pruebas sobre la plataforma implementada buscando fallas y deficiencias.

4. **Revisión**: Según los resultados de las pruebas, se llevan cabo los cambios pertinentes al diseño.

Una vez que sea llevada a cabo la revisión por parte de los supervisores de la empresa y no se encuentren fallas, entonces, se documenta la implementación final y se procede con la puesta en producción de la nueva plataforma.

3.2 Herramientas de Desarrollo

3.2.1 Tecnología Java

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java y un conjunto de herramientas de desarrollo. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución, y un conjunto de librerías estándar que ofrecen funcionalidad común [7].

La plataforma es así llamada la Plataforma Java (antes conocida como Plataforma Java 2), e incluye [7]:

- ✓ Plataforma Java, Edición Estándar (Java Platform, Standard Edition), o Java SE (antes J2SE).
- ✓ Plataforma Java, Edición Empresa (Java Platform, Enterprise Edition), o Java EE (antes J2EE).
- ✓ Plataforma Java, Edición Micro (Java Platform, Micro Edition), o Java ME (antes J2ME).

Desde 2006, la versión actual de la Plataforma Java se conoce tanto por versión 1.5 o versión 5, aunque todas las denominaciones se refieren a la misma versión. Sin embargo, se prefiere el término versión 5. Una visión general de la multitud de tecnologías que componen la Plataforma Java puede encontrarse en la página de documentación del JDK.

La Plataforma Java se compone de un amplio abanico de tecnologías, cada una de las cuales ofrece una parte del complejo de desarrollo o del entorno de ejecución en tiempo real. Por ejemplo, los usuarios finales suelen interactuar con la máquina virtual de Java y el

conjunto estándar de bibliotecas. Además, las aplicaciones Java pueden usarse de forma variada, como por ejemplo ser incrustadas en una página Web. Para el desarrollo de aplicaciones, se utiliza un conjunto de herramientas conocidas como JDK (Java Development Kit, o herramientas de desarrollo para Java) [7].

3.2.1.1 Java Runtime Environment

Un programa destinado a la Plataforma Java necesita dos componentes en el sistema donde se va a ejecutar: una máquina virtual de Java (JVM), y un conjunto de librerías para proporcionar los servicios que pueda necesitar la aplicación. La JVM que proporciona Sun Microsystems, junto con su implementación de las librerías estándar, se conocen como Java Runtime Environment (JRE) o Entorno en tiempo de ejecución para Java. El JRE es lo mínimo que debe contener un sistema para poder ejecutar una aplicación Java sobre el mismo. Para el desarrollo de programas se ofrece un paquete de utilidades y herramientas conocido como JSDK (Java Software Development Kit) [7].

3.2.1.2 Máquina Virtual de Java

El corazón de la Plataforma Java es el concepto común de un procesador "virtual" que ejecuta programas escritos en el lenguaje de programación Java. En concreto, ejecuta el código resultante de la compilación del código fuente, conocido como bytecode. Este "procesador" es la máquina virtual de Java o JVM (Java Virtual Machine), que se encarga de traducir (interpretar o compilar en tiempo de ejecución) el bytecode en instrucciones nativas de la plataforma destino. Esto permite que una misma aplicación Java pueda ser ejecutada en una gran variedad de sistemas con arquitecturas distintas, siempre que con una implementación adecuada de la JVM. Este hecho es lo que ha dado lugar a la famosa frase: "write once, run anywhere" (escribir una vez, ejecutar en cualquier parte). La condición es que desde la versión 1.2 de JRE, la implementación de la máquina virtual de Sun incluye un compilador JIT (Just In Time). De esta forma, en vez de la tradicional interpretación del código bytecode, que da lugar a una ejecución lenta de las aplicaciones, el JIT convierte el bytecode a código nativo de la plataforma destino. Esta segunda compilación del código penaliza en cuanto a tiempo, pero el código nativo resultante se ejecuta de forma más eficaz

y rápida que si fuera interpretado. Otras técnicas de compilación dinámica del código durante el tiempo de ejecución permiten optimizar más aún el código.

Sin embargo, no puede decirse que el resultado de la compilación de Java pueda compilar el código con un máximo de eficiencia, y aprovechar los beneficios en cuanto a velocidad de código máquina nativo. Aunque los compiladores cada vez son más avanzados, no todas las librerías de Java tienen asociado un código máquina equivalente que aprovechar. Por ejemplo, la librería “reflect”, que permite a los programadores de Java explorar instrucciones que sólo están disponibles en tiempo de ejecución, está pobremente representado por código máquina.

Java no fue la primera plataforma basada en el concepto de una máquina virtual, aunque es la que de más amplia difusión ha gozado. El empleo de máquinas virtuales se había centrado principalmente en el uso de emuladores para ayudar al desarrollo de hardware en construcción o sistemas operativos, pero la JVM fue diseñada para ser implementada completamente en software, y al mismo tiempo hacer que fuera portable a todo tipo de hardware [7].

3.2.1.3 Librerías de Java

En la mayoría de los sistemas operativos actuales, se ofrece una cantidad de código para simplificar la tarea de programación. Este código toma la forma, normalmente, de un conjunto de librerías dinámicas que las aplicaciones pueden llamar cuando lo necesiten. Pero la Plataforma Java está pensada para ser independiente del sistema operativo subyacente, por lo que las aplicaciones no pueden apoyarse en funciones dependientes de cada sistema en concreto. Lo que hace la Plataforma Java, es ofrecer un conjunto de librerías estándar, que contiene mucha de las funciones reutilizables disponibles en los sistemas operativos actuales.

Las librerías de Java tienen tres propósitos dentro de la Plataforma Java. Al igual que otras librarías estándar, ofrecen al programador un conjunto bien definido de funciones para realizar tareas comunes, como manejar listas de elementos u operar de forma

sofisticada sobre cadenas de caracteres. Además, las librerías proporcionan una interfaz abstracta para tareas que son altamente dependientes del hardware de la plataforma destino y de su sistema operativo. Tareas tales como manejo de las funciones de red o acceso a archivos, suelen depender fuertemente de la funcionalidad nativa de la plataforma destino. En el caso concreto anterior, las librerías java.net y java.io implementan el código nativo internamente, y ofrecen una interfaz estándar para que aplicaciones Java puedan ejecutar tales funciones. Finalmente, no todas las plataformas soportan todas las funciones que una aplicación Java espera. En estos casos, las librerías bien pueden emular esas funciones usando lo que esté disponible, o bien ofrecer un mecanismo para comprobar si una funcionalidad concreta está presente [7].

3.2.2 Sistema Manejador de Base de Datos – MySQL

MySQL es un SMBD (Sistema Manejador de Bases de Datos), con las características de ser multi-hilos y multi-usuarios, esta basado en el lenguaje estándar SQL. Su principal característica es la de ser un SMBD Open Source o Código Abierto, permitiendo su uso y adaptación a las necesidades de los usuarios, así mismo MySQL emplea la modalidad de Licencia Pública General (General Public License – GPL).

MySQL fue desarrollado por la compañía MySQL AB, y ha tenido un crecimiento impresionante en los últimos años, se estima que existen más de seis millones de instalaciones de MySQL en todo el mundo [8].

3.2.2.1 Características

- ✓ **Rendimiento**: MySQL permite obtener un buen rendimiento del Hardware, ya que posee la característica de ser multi-hilo, lo cual optimiza los tiempos de respuesta, y permite aprovechar los diferentes procesadores que se encuentren disponibles. Así mismo implemente un mecanismo llamado caché de consulta, mediante el cual se almacenan las últimas consultas ejecutadas recientemente y así en caso de ser requeridas por el usuario, las mismas no deben ser resueltas nuevamente, simplemente se lista el resultado de la consulta con los datos almacenados en dicha caché [8].

- ✓ **ACID**: MySQL brinda soporta los principios básicos que deben existir en todo SMBDR (Sistema Manejador de Bases de Datos Relacionales) para que sea considerado como tal. Es por ello que MySQL cumple con el principio ACID [8].

- ✓ **Multi-Usuario**: MySQL es capaz de soportar que varios usuarios se conecten al mismo tiempo y que puedan manipular y administrar las distintas bases de datos existentes y administrar al SMBDR como tal, en el mismo instante de tiempo [8].

- ✓ **Desarrollo**: MySQL está desarrollado en lenguaje C y C++ [8].

- ✓ **Plataformas**: MySQL es capaz de trabajar bajo las siguientes plataformas: AIX 4x 5x, Amiga, BSDI, Digital Unix 4x, FreeBSD 2x 3x 4x, HP-UX 10.20 11x, Linux 2x, Mac OS, NetBSD, Novell NetWare 6.0, OpenBSD 2.5, OS/2, SCO OpenServer, SCO UnixWare 7.1.x, SGI Irix 6.x, Solaris 2.5, SunOS 4.x, Tru64 Unix y Windows 9x, Me, NT, 2000, XP, 2003 [8].

- ✓ **Integridad:** MySQL tiene como característica el manejo de integridad referencial y de transacciones, gracias a las tablas InnoDB, que permite el manejo de las transacciones mediante la sentencia “BEGIN WORK” y finaliza con un “COMMIT” o “ROLLBACK”, o puede culminar en modo “AUTOCOMMIT”. Dichas tablas deben ser creadas para cada tabla que sea creada por el usuario, mediante la sentencia “Type=InnoDB”, la cual se coloca al final de la sentencia “CREATE TABLE” [8].

- ✓ **Seguridad**: MySQL implementa un protocolo que se encarga de emplear diferentes algoritmos para cifrar los datos que viajan a través de una red pública (ejemplo: Internet), el cual tiene por nombre Capa de Seguridad de Sockets (Secure Sockets Layer – SSL). Este protocolo trabajaría en el cliente y el servidor MySQL [8].

3.2.2.2 Ventajas

✓ **Licenciamiento**: MySQL brinda la facilidad de poder emplear y adaptar el código fuente a las necesidades de los usuarios, sin verse en la obligación de adquirir una licencia para su uso. Sin embargo, si se emplea MySQL para obtener beneficios comerciales, si es necesario la adquisición de una licencia [8].

✓ **Soporte**: MySQL posee una amplia gama de manuales y documentos, mediante los cuales se le brindan a los usuarios todo el soporte necesario para solventar cualquier inconveniente que se surja con el SMBDR o para guiarlos en dudas o interrogantes que se presenten [8].

✓ **Conectividad**: MySQL provee un número amplio de Interfaces de Programación de Aplicaciones (Application Programming Interfase – API), mediante las cuales permite la recepción de servicios y la entrega de respuestas a dichos servicios, del SMBDR con otras aplicaciones. Dentro de estos lenguajes de programación se encuentran: C, C++, C#, Eiffel, Smalltalk, Java, Lisp, Perl, PHP, Python, Ruby, Tcl, etc [8].

✓ **Tablas InnoDB**: Mediante el uso de estas tablas, se obtienen ventajas tales como [8]:

- Recuperación automática ante fallas: MySQL en caso de presentarse una falla, es capaz de culminar automáticamente todas aquellas transacciones que se encuentran incompletas, es decir, hacer “COMMIT” de las mismas.
- Integridad Referencial: Esto permite la creación de claves foráneas que permitan relacionar un campo de una tabla con otro campo en otra tabla.
- “SELECTs” sin bloqueo: Las tablas InnoDB emplean una técnica conocida como Multi-Versioning, la cual se encarga de evitar el bloqueo en consultas simples, echas mediante al sentencia “SELECT”, ya que las mismas sólo implican lecturas y no se están realizando cambios en la tabla.

- ✓ **Rapidez**: MySQL es un SMBDR que se caracteriza por ser muy rápido al momento de resolver las consultas solicitadas por los usuarios, lo cual le brinda una ventaja significativa al momento de ser probado para su posible implantación, y dicha ventaja se aprecia mejor en sistemas donde se implemente tecnología Web, ya que el tiempo es un factor determinante en la usabilidad de dichos sistemas [8].

3.2.2.3 Desventajas

MySQL es un SMBD que carece de una característica primordial como lo es el poder manejar la integridad referencial de los datos, siendo esta una debilidad muy importante en comparación con otros SMBD, en particular con aquellos que son código abierto, para poder contar con dicha característica, es necesario la implementación de las tablas InnoDB.

Otras de las desventajas más importantes de este SMBD es que no posee soporte para el manejo de transacciones.

MySQL por ser un SMBD de código abierto no brinda un adecuado soporte técnico a las necesidades o interrogantes de los usuarios, siendo en la mayoría de los casos resueltas por programadores independientes y no por personal de la organización.

Al igual que otros SMBD que se encuentran bajo el mismo esquema de licenciamiento, MySQL no escapa a los inconvenientes de la escasez de documentación, que brinde soporte a los usuarios [8].

CAPÍTULO IV. Análisis y Diseño de la Aplicación

Para poder lograr con los objetivos planteados en este trabajo, fue importante realizar un análisis de las plataformas existentes para el momento, con el fin de determinar aspectos importantes, los cuales sirvieron como base para el desarrollo de la nueva plataforma integrada GSM-SMPP. Adicionalmente se analizó el sistema de información Web, a través del cual los clientes de la empresa suministran la data para el envío de mensajes SMS.

4.1 Plataforma GSM (Aplicaciones Mono-Hilo)

La plataforma GSM ha evolucionado paulatinamente, la primera versión, fue desarrollada bajo Visual Basic 6.0, y utilizaba MS Access como base de datos, su estructura se muestra en la siguiente figura.

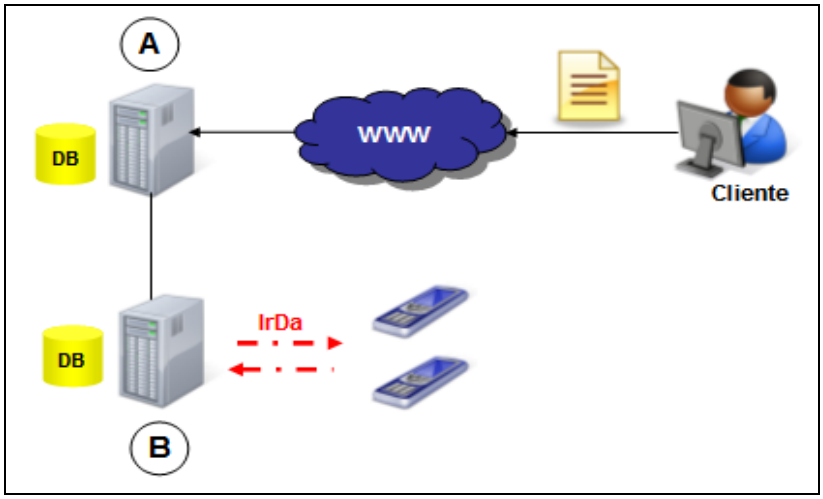

Figura 07. Plataforma GSM mono-hilo

Los clientes a través del portal web hacían la carga de archivos que contenía los datos requeridos para enviar los mensajes, estos archivos quedaban almacenados en el directorio definido para el cliente en el servidor web (A). Luego, una aplicación ejecutada en una estación de trabajo local (B), invocaba un web service en (A) que se encargaba del procesamiento del archivo para generar con este, uno nuevo estandarizado, el cual servía para surtir a la base de datos local de (B), inmediatamente se publicaba en la web el listado de los mensajes a enviar, con estátus “En proceso”. Luego, los mensajes almacenados en (B) eran transmitidos a celulares NOKIA 8210, mediante una interfaz IrDa para que se

efectuaran los envíos. De manera similar, las respuesta recibidas por los celulares, eran transmitidas a la estación (B), donde eran almacenados en su base de datos local. Adicionalmente, la aplicación ejecutada en (B), contaba con dos procesos, los cuales, eran los encargados de insertar las respuestas recibidas y actualizar los registros correspondientes a las transmisiones, en la base de datos de (A).

Esta plataforma presentaba la desventaja de contar con aplicaciones mono hilo, lo que generaba retrasos considerables en las distintas tareas que se llevaban a cabo a lo largo del proceso, por ejemplo, los archivos nuevos no eran cargados en la base de datos de (B), hasta tanto ésta no finalizara de procesar un lote de envíos, desperdiciando un tiempo valioso.

4.2 Plataforma GSM (Aplicaciones Multi-Hilo)

Buscando optimizar su plataforma, la empresa llevó a cabo un nuevo desarrollo, el cual, introdujo el uso de nuevas tecnologías: se suplantaron las aplicaciones existentes con nuevas, multi-hilos, implementadas en Java, se migró la base de datos de MS Access al sistema manejador de base datos MySQL, se suplantó el uso de celulares con interfaz IrDa por módems GSM conectados vía serial. Así mismo, realizaron la migración de las aplicaciones de ASP a PHP.

En la siguiente imagen se presenta la estructura de la nueva plataforma:

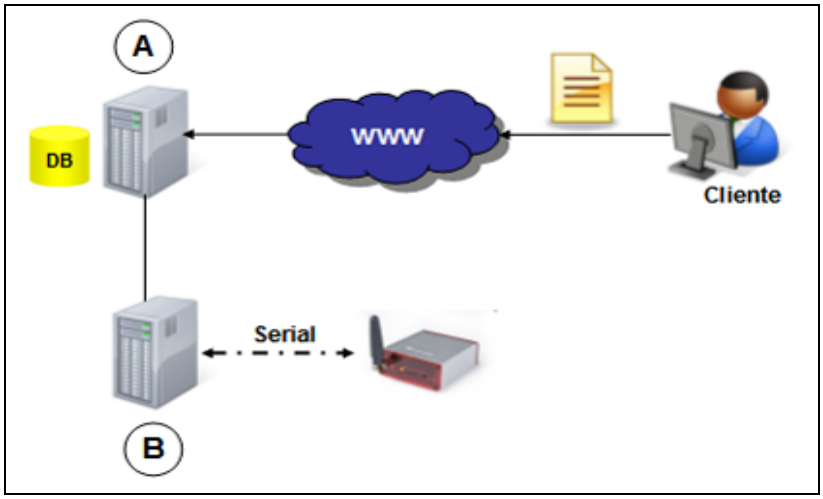

Figura 08. Plataforma GSM multi-hilo

Al igual que en la versión anterior, los clientes suben los archivos a su directorio a través de la página web, un proceso verifica constantemente la existencia de un nuevo archivo, para llevar a cabo el vaciado de esta data en las tablas de la BD, de forma tal que se encuentre disponible para la publicación web y para el nuevo proceso de envío de SMS, que lleva a cabo una estación de trabajo local (B), de la siguiente manera: un hilo consulta la base de datos en (A) por envíos pendientes e inmediatamente procede a enviarlos mediante un módem GSM conectado vía puerto serial con la estación (B); otro hilo se encarga de consultar el módem en búsqueda de mensajes recibidos, para insertar dichas respuestas en la base de datos de (A).

A pesar de que este enfoque mejora considerablemente la eficiencia de la plataforma anterior, ambas carecen de una característica muy importante y que es, en definitiva, la razón principal que motiva el desarrollo de la nueva plataforma basada en SMPP, y es que ninguna es capaz de conocer el estatus de los mensajes enviados.

4.3 Plataforma SMPP

Luego de varias reuniones junto a los directivos de la empresa ALS Comunicaciones, se concretó cual sería la estructura de la nueva plataforma (Figura 09) y cuales serían los requerimientos que la plataforma SMPP debería cumplir, estos son:

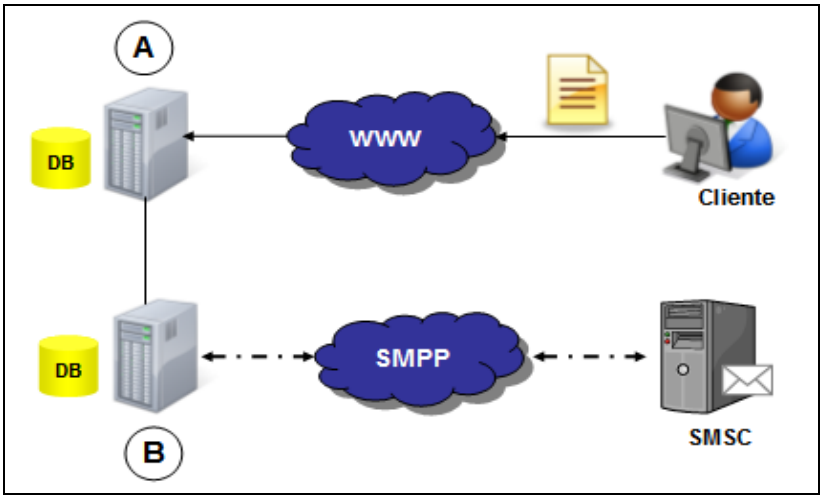

Figura 09. Plataforma SMPP

- ✓ Estar desarrollada íntegramente en Java, gracias a las características propias del lenguaje como lo son: conectividad con diversos tipos de manejadores de base de datos, conexión mediante sockets, escalabilidad, multiplataforma y multi-hilo, así como también por la gran cantidad de documentación disponible sobre el mismo.
- ✓ El manejador de base de datos debe ser MySQL dado que es el que se ha venido utilizando desde la plataforma multi-hilo y ha brindado buen rendimiento.
- ✓ La aplicación debe ser configurable, en tal sentido que pueda ser independiente de la operadora que presta el servicio de mensajería corta de texto a través de SMPP.
- ✓ Los parámetros de configuración deben estar definidos en una base de datos, dada la flexibilidad y eficiencia que ésta ofrece frente al manejo de archivos.
- ✓ Debe permitir realizar consultas sobre el estatus de los mensajes de texto enviados al SMSC, para poder tener un mayor control de los envíos realizados.
- ✓ Se debe mantener el proceso de carga de mensajes por lotes que se ha manejado a lo largo de la evolución de la plataforma, con el fin de que los cambios sean los más transparente posible para los clientes de la empresa.
- ✓ Se realizarán réplicas locales (B) de los registros de transmisiones de la base de datos web (A), según una lógica de negocio establecida para procesar los envíos.
- ✓ La carga de los mensajes de respuesta serán almacenados de forma local (B) y no automáticamente sobre la Web (A), para controles de auditoría.

Una vez realizado el análisis y levantado todos los requerimientos para la nueva plataforma se procedió a realizar el diseño de la aplicación mediante diagramas de: Casos de Uso, Modelos de Datos, de Clase y de Secuencia.

4.3.1 Diagramas de Casos de Uso

A continuación se presentan los diagramas de Casos de Uso del sistema.

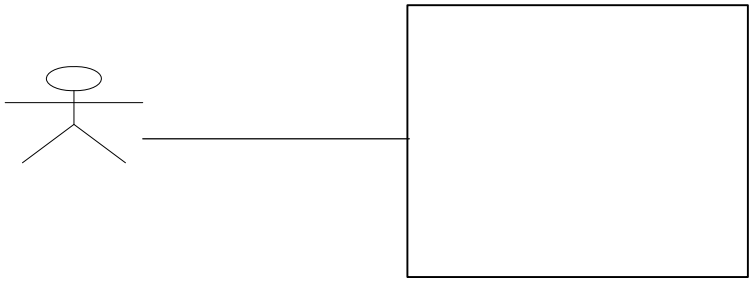

Figura 10. Diagrama de Caso de Uso, Nivel 0

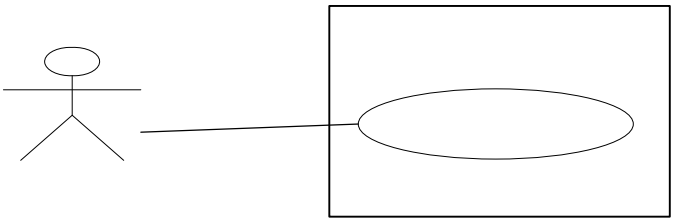

Figura 11. Diagrama de Caso de Uso, Nivel 1

Descripción de los Casos de Uso:

- Caso de uso (Conectar BD): El operador introduce los datos necesarios para establecer las conexiones a las bases de datos que sustentan la aplicación.

Pre-Condición: El operador debe conocer la dirección IP de los servidores, así como los nombres de usuario y las contraseñas, para cada uno.

Post-Condición: Se establecen las conexiones a las bases de datos local y remota.

Operador

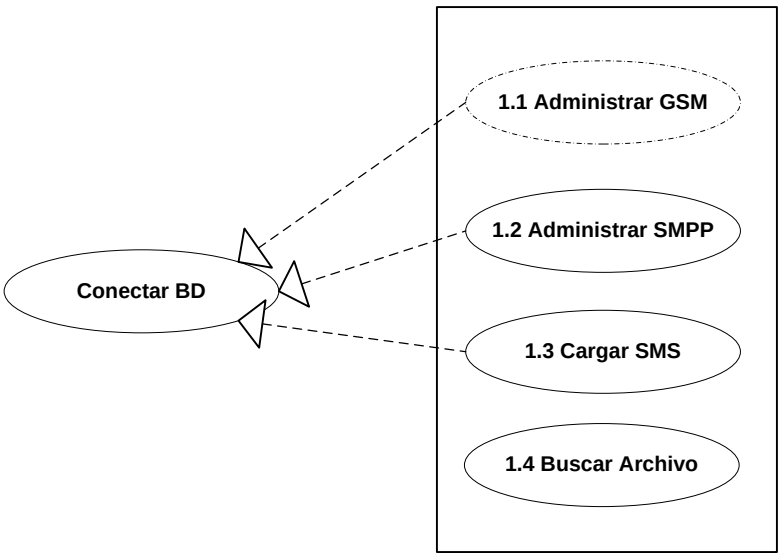

Figura 12. Diagrama de Caso de Uso, Nivel 2

- Caso de uso (Administrar SMPP): Permite ingresar al módulo administrativo de mensajería de texto SMPP.

Pre-Condición: Deben estar establecidas las conexiones a los servidores de Bases de Datos.

Post-Condición: El operador ingresa al módulo de administración de mensajería SMPP.

- Caso de uso (Cargar SMS): Se realiza la carga de los mensajes de texto a enviar, que se encuentran en la base de datos web.

Pre-Condición: Deben estar establecidas las conexiones a los servidores de Bases de Datos, y existir transmisiones pendientes para un cliente especificado.

Post-Condición: Se realiza la carga de los mensajes en la tabla que alimentará la aplicación.

- Caso de uso (Buscar Archivo): Invoca la rutina que permite cargar los archivos enviados por los clientes a la Base de Datos Web.

Pre-Condición: Debe existir un archivo en el directorio Web del cliente seleccionado.

Post-Condición: Se realiza la carga de los registros incluidos en el archivo, en la Base de Datos Web del cliente.

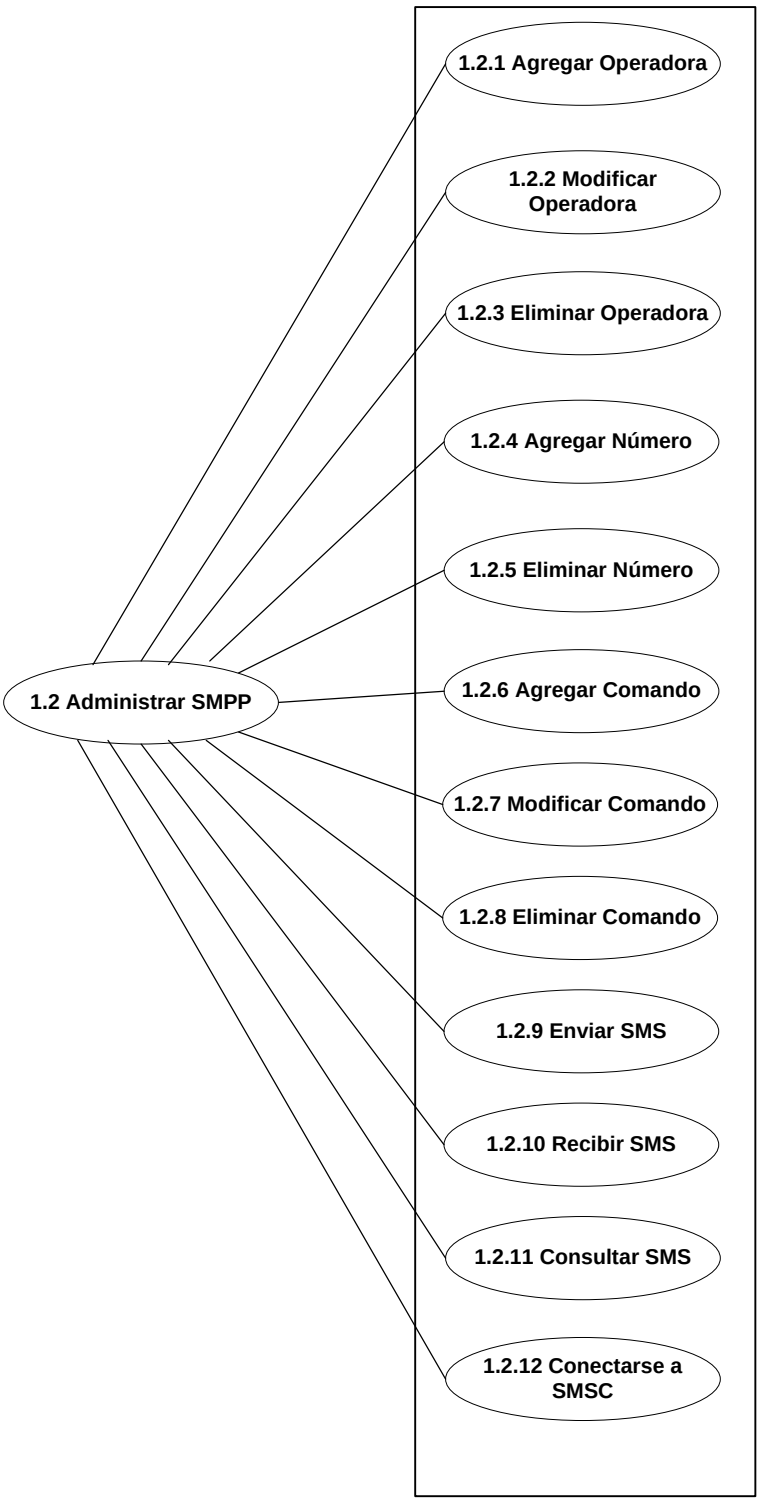

Figura 13. Diagrama de Caso de Uso, Nivel 3

- Caso de uso (Agregar Operadora): El usuario introduce una nueva operadora, especificando todos los valores requeridos para configurar la conexión con la misma.

Pre-Condición: El usuario debe conocer cuales son los valores de configuración para la conexión.

Post-Condición: La operadora es creada en la base de datos y se encuentra disponible para lograr la conexión.

- Caso de uso (Modificar Operadora): El usuario selecciona una operadora ya definida y modifica los datos de conexión.

Pre-Condición: El usuario debe conocer cuál es el nombre de la operadora, el número, y el o los valores a modificar.

Post-Condición: Se actualiza el valor de configuración modificado en la base de datos y se encuentra disponible para lograr la conexión.

- Caso de uso (Eliminar Operadora): El usuario selecciona una operadora, para ser eliminada de la base de datos.

Pre-Condición: El operador debe conocer cuál es el nombre de la operadora a eliminar.

Post-Condición: La operadora es eliminada de la base de datos.

- Caso de uso (Agregar Comando): El operador introduce un nuevo comando, indicando el parámetro específico que será utilizado para una operadora en particular.

Pre-Condición: El operador debe conocer cuál es el nombre del comando, el valor de su parámetro y la operadora a la cual se le será asignado.

Post-Condición: El comando es creado en la base de datos y se encuentra disponible para ser utilizado en la configuración de una operadora.

- Caso de uso (Modificar Comando): El operador selecciona un comando ya definido para una operadora en particular y modifica la información del parámetro.

Pre-Condición: El operador debe conocer cuál es el nombre de la operadora, el comando, y el valor del parámetro a modificar.

Post-Condición: Se actualiza el comando modificado en la base de datos y se encuentra disponible para ser utilizado en la configuración de una operadora.

- Caso de uso (Eliminar Comando): El operador selecciona una operadora, luego uno de los comandos que ésta tenga definido y por último, el parámetro a eliminar.

Pre-Condición: El operador debe conocer cuál es el nombre de la operadora, el comando, y el valor del parámetro a eliminar.

Post-Condición: El comando es eliminado de la base de datos.

- Caso de uso (Enviar Mensajes): El operador selecciona una operadora, y activa el proceso de envío de los mensajes previamente cargados hacia el SMSC.

Pre-Condición: El operador debe conocer cuál es el nombre de la operadora, los mensajes deben estar previamente cargados en la base de datos.

Post-Condición: Los mensajes son enviados al SMSC.

- Caso de uso (Recibir Mensajes): El operador selecciona una operadora, y activa el proceso de recepción de mensajes desde el SMSC.

Pre-Condición: El operador debe conocer cuál es el nombre de la operadora.

Post-Condición: Los mensajes son recibidos desde el SMSC.

- Caso de uso (Consultar Mensajes): El operador introduce un número celular y obtiene el estatus del último mensaje enviado a ese número.

Pre-Condición: El operador debe conocer cuál es el número a consultar.

Post-Condición: Se muestra el estatus del mensaje.

- Caso de uso (Conectarse a SMSC): El operador selecciona la operadora y el número a utilizar para establecer la conexión con el SMSC.

Pre-Condición: El operador debe haber seleccionado la operadora y el número a utilizar.

Post-Condición: Se establece la conexión con el SMSC.

4.3.2 Modelo de Datos

A continuación se presenta el Modelo de Datos del sistema.

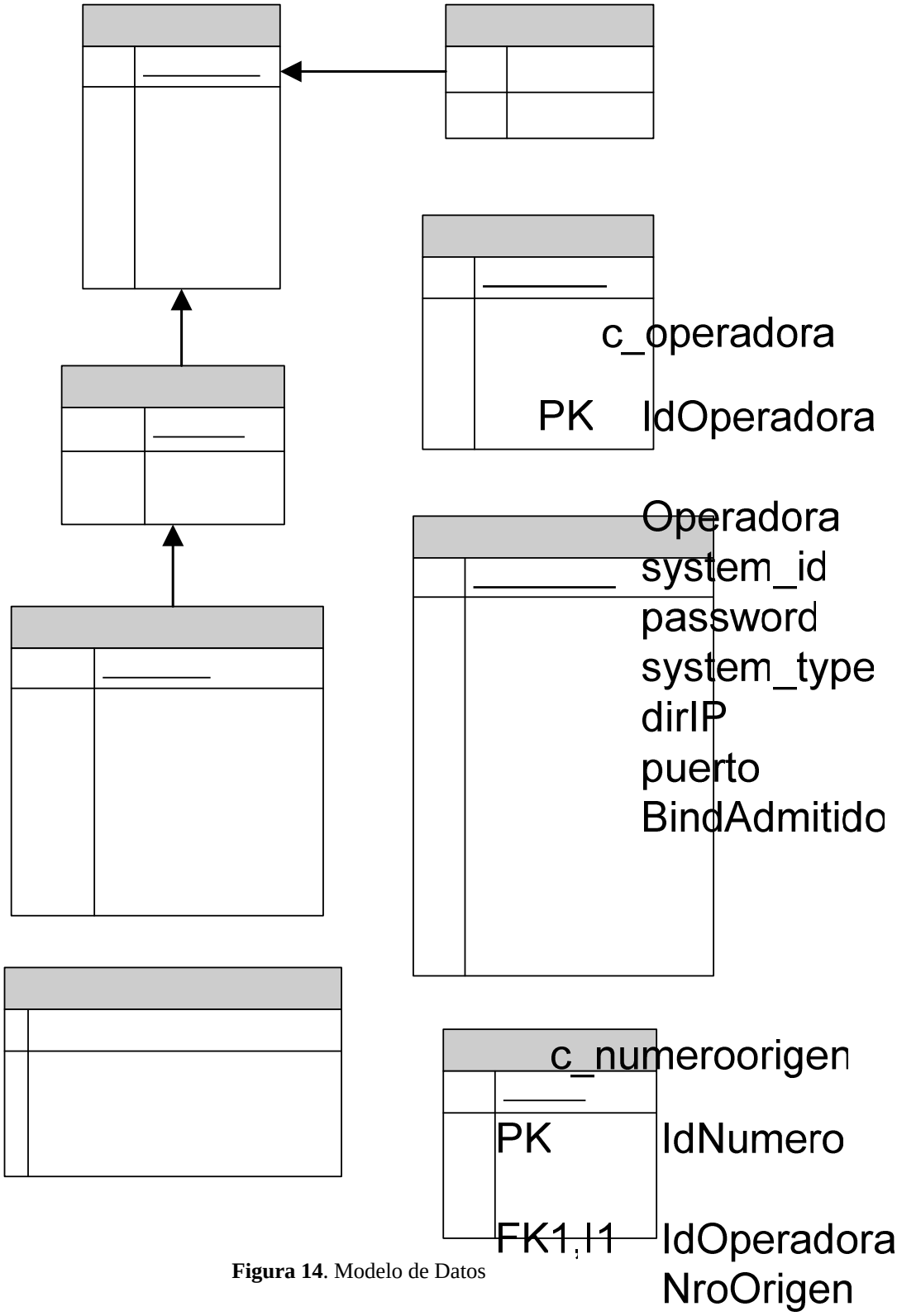

Figura 14. Modelo de Datos

- La entidad **c_operadora**, mostrada en la Figura 15, es aquella en donde se almacena toda la información necesaria para configurar los parámetros de conexión de las operadora.

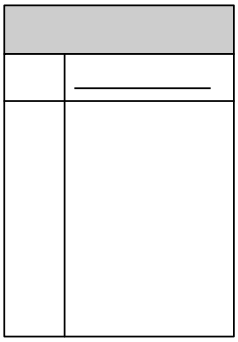

Figura 15. Entidad c_operadora

- La entidad **c_operadoracodigo**, mostrada en la Figura 16, es aquella en donde se almacenan los códigos asociados para cada operadora.(ej.: 0424-xxxxxxx)

c_operadoracodigo	
FK1	**IdOperadora**
	Codigo

Figura 16. Entidad c_operadoracodigo

- La entidad **c_numeroorigen**, mostrada en la Figura 17, almacena los números contratados por la empresa a las operadoras.

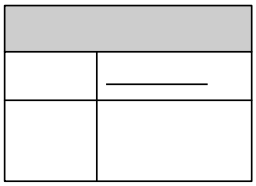

Figura 17. Entidad c_numeroorigen

- La entidad **c_cliente**, mostrada en la Figura 18, almacena los datos de conexión a las bases de datos de cada cliente de la empresa.

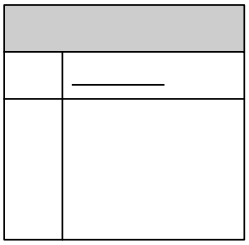

Figura 18. Entidad c_operadora

- La entidad **c_valorparametros**, mostrada en la Figura 19, almacena los valores de los parámetros por cada comando para una operadora en específico.

c_valorparametros	
PK	IdComando
FK1,I1	IdOperadora IdNumero ComandoSMPP Parametro DescripcionParametro Valor Tipo Longitud

Figura 19. Entidad c_valorparametros

- Como muestra la Figura 20, la entidad **t_transmisión** almacena toda la información correspondiente a cada uno de los mensajes por enviar.

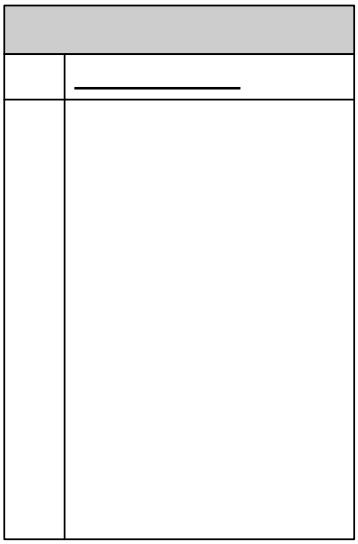

Figura 20. Entidad t_transmision

- Como muestra la Figura 21, la entidad **t_respuesta** almacena toda la información correspondiente a cada uno de los mensajes de respuesta recibidos.

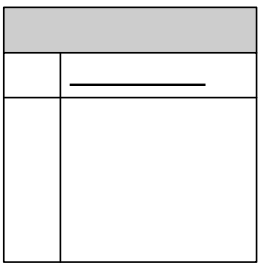

Figura 21. Entidad t_respuesta

4.3.3 Diagrama de Clases

El siguiente diagrama corresponde al diagrama de clases diseñado para implementar el sistema.

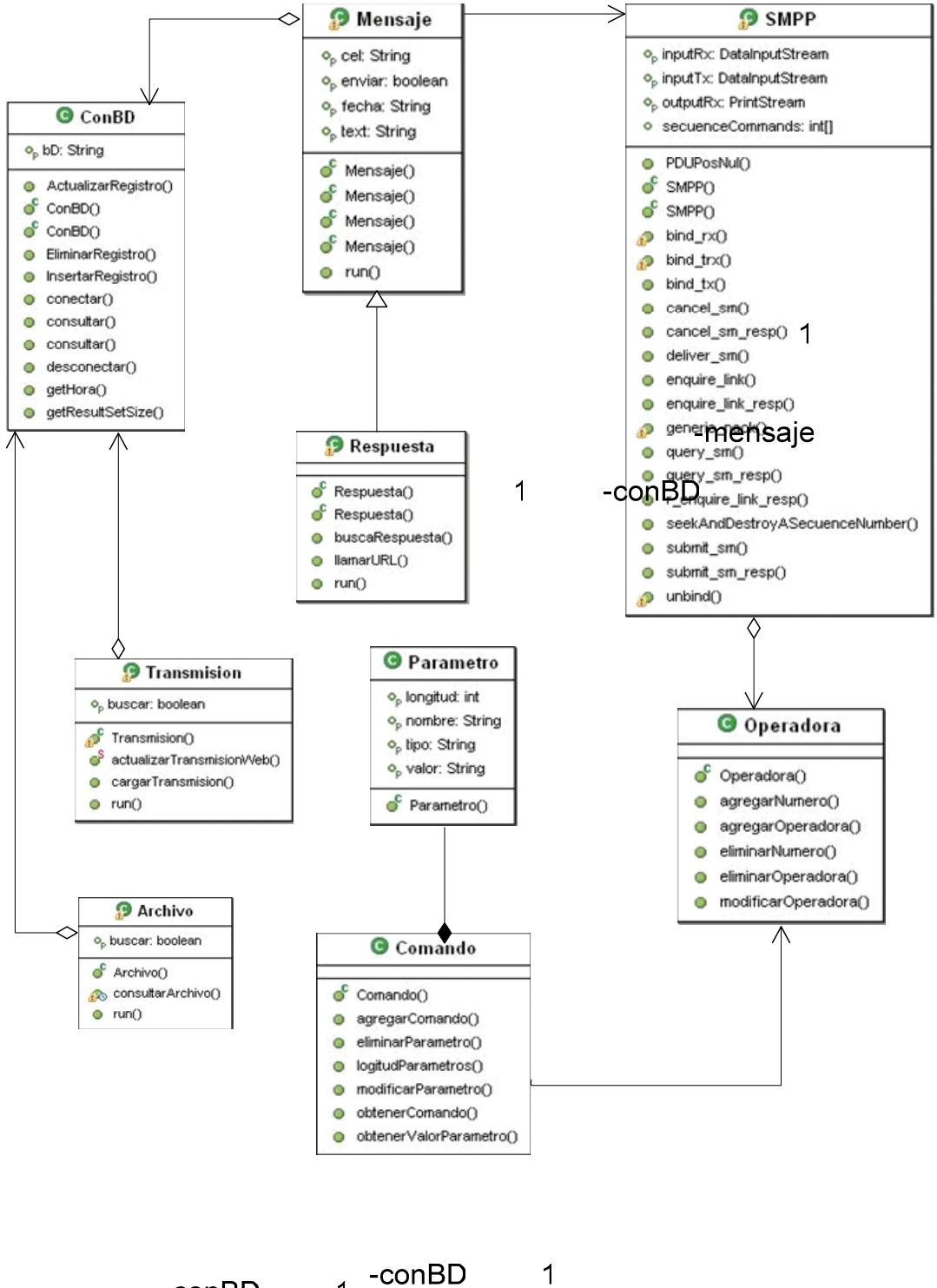

Figura 22. Diagrama de Clases

4.3.4 Diagramas de Secuencia

A continuación se presentan los diagramas de secuencia correspondientes a cada una de las funcionalidades del sistema, donde se denota la interacción entre los componentes que lo conforman:

Diagrama de secuencia donde se establece la conexión con las bases de datos:

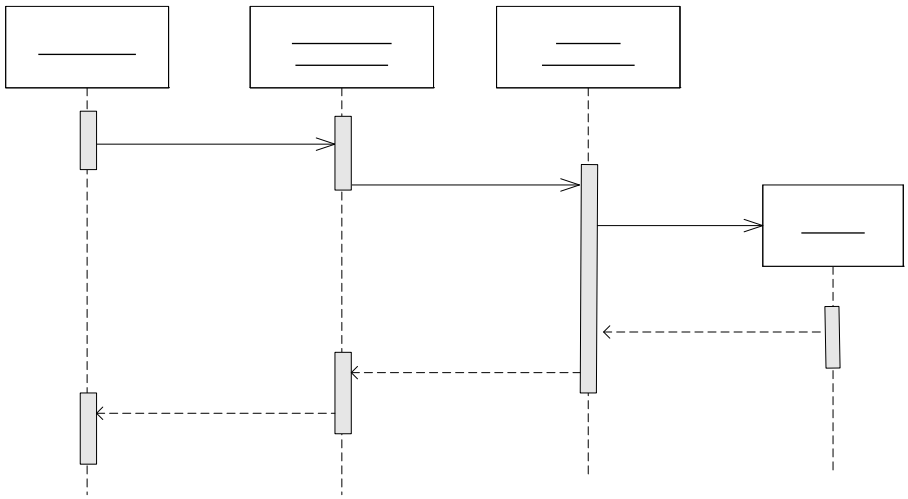

Figura 23. Diagrama de secuencia para establecer la conexión con la base de datos

Diagrama de secuencia donde se procede con la carga de los envíos pendientes de la base de datos Web a la base de datos local:

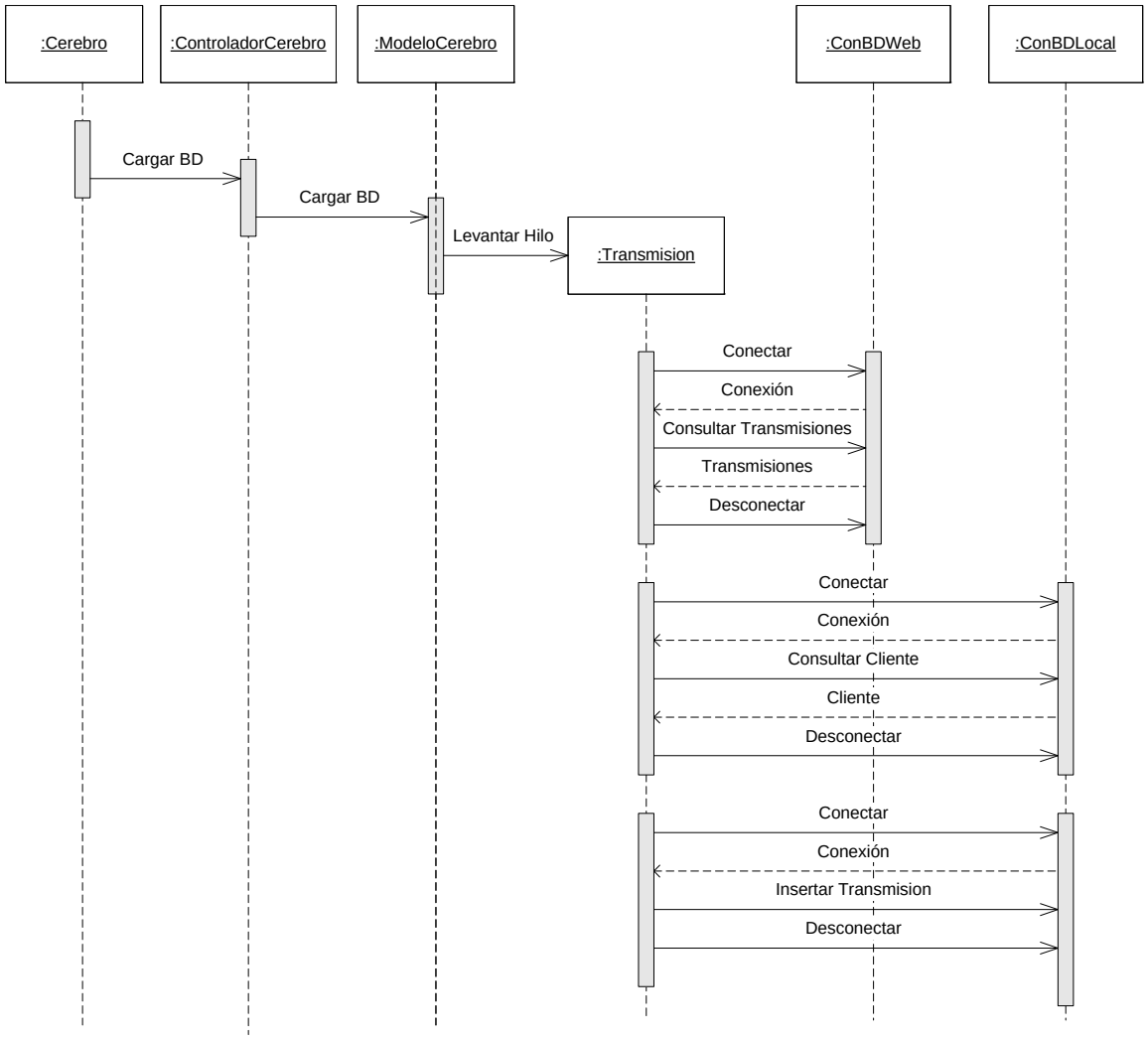

Figura 24. Diagrama de secuencia para cargar transmisiones a la base de datos local

Diagrama de secuencia donde se invoca un servicio web para la búsqueda de nuevos archivos en el directorio del cliente:

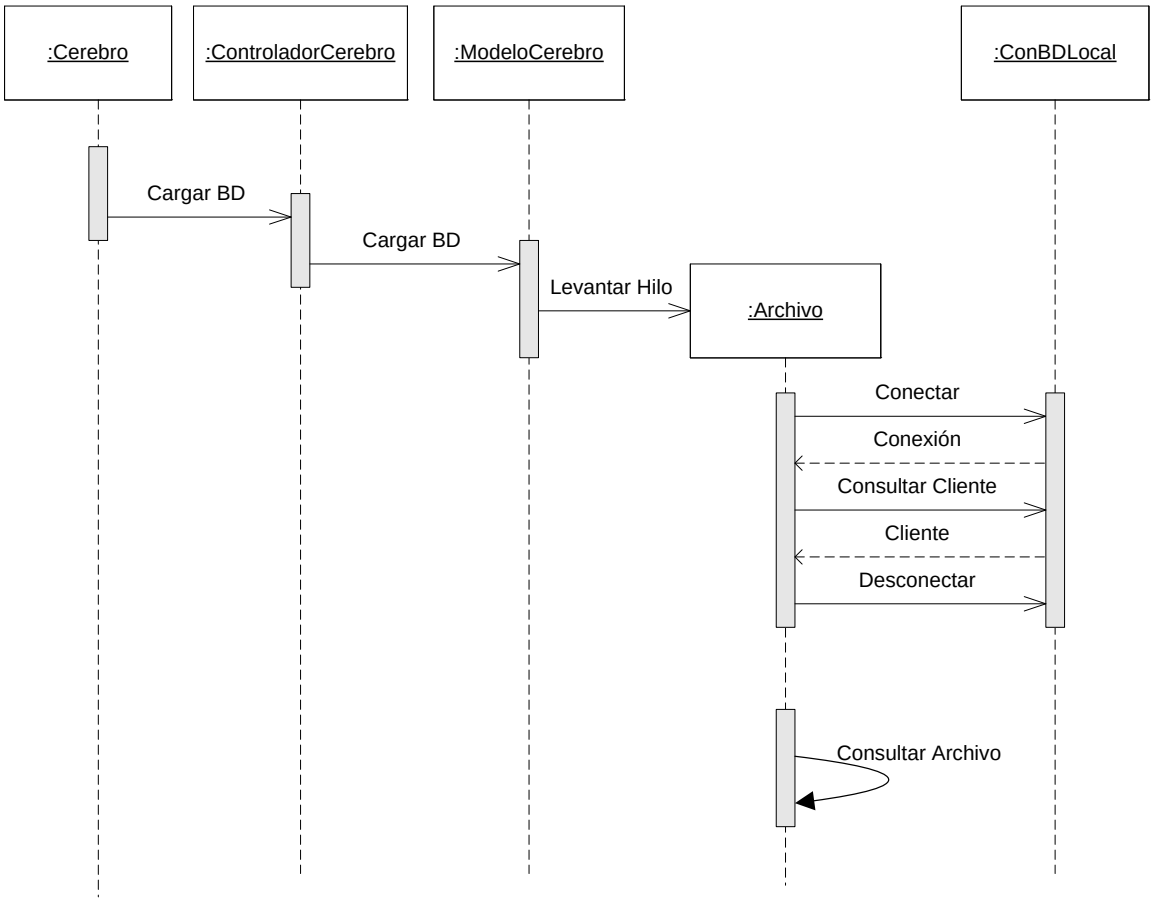

Figura 25. Diagrama de secuencia para buscar por archivos nuevos

Diagrama de secuencia donde se crea una nueva operadora en el sistema:

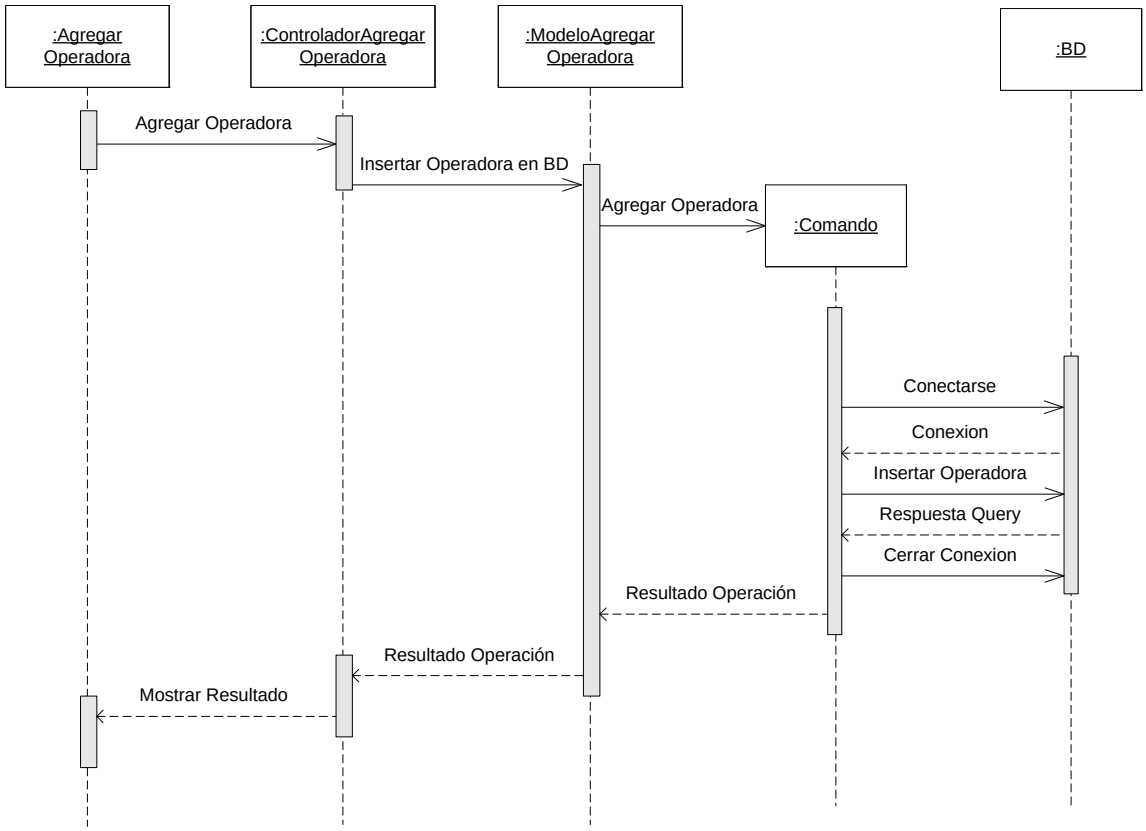

Figura 26. Diagrama de secuencia para crear una nueva operadora en el sistema

Diagrama de secuencia donde se modifican los valores de configuración de una operadora ya existente en el sistema:

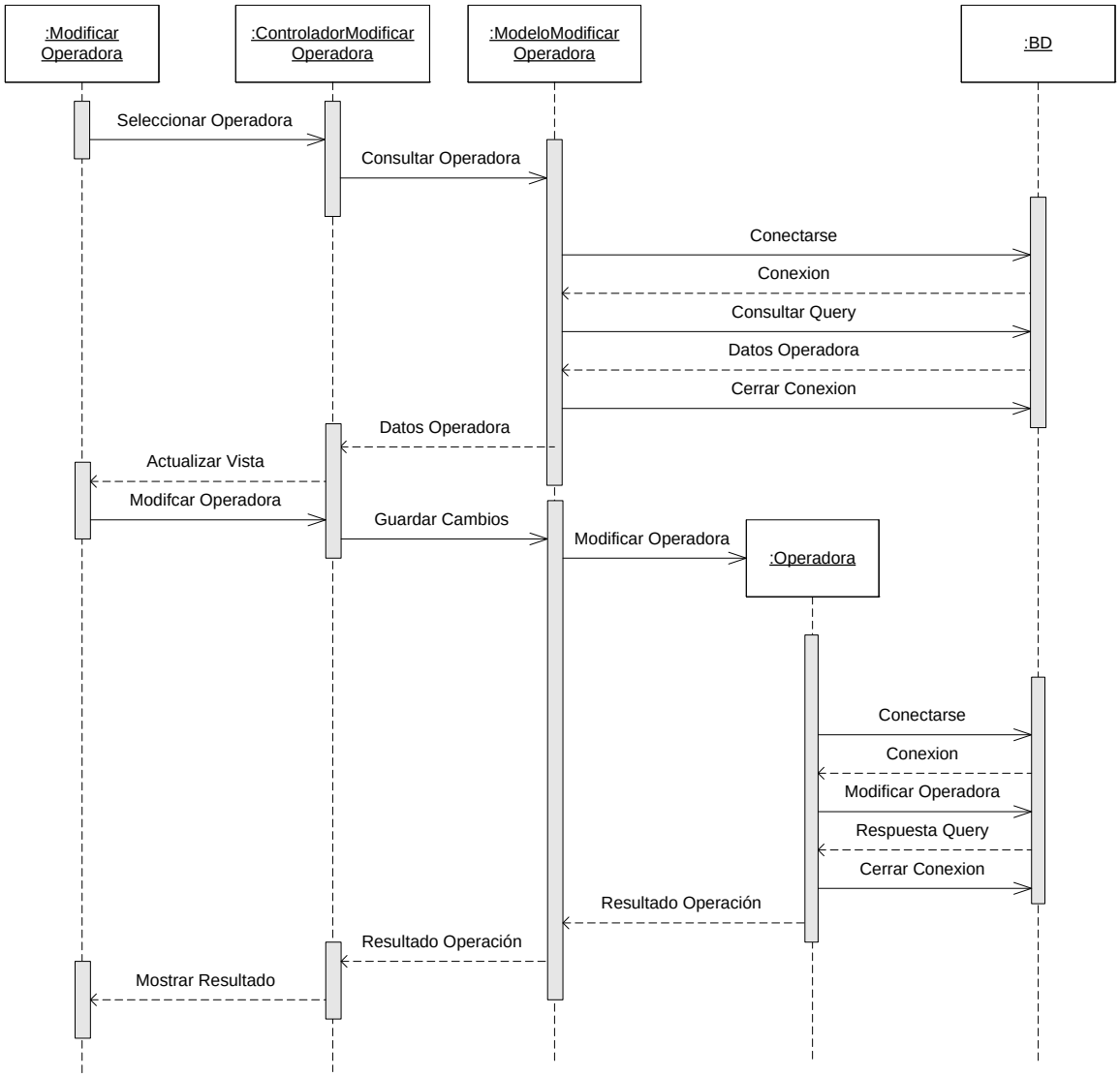

Figura 27. Diagrama de secuencia para la modificación de los valores de una operadora

Diagrama de secuencia donde se elimina una operadora configurada en el sistema:

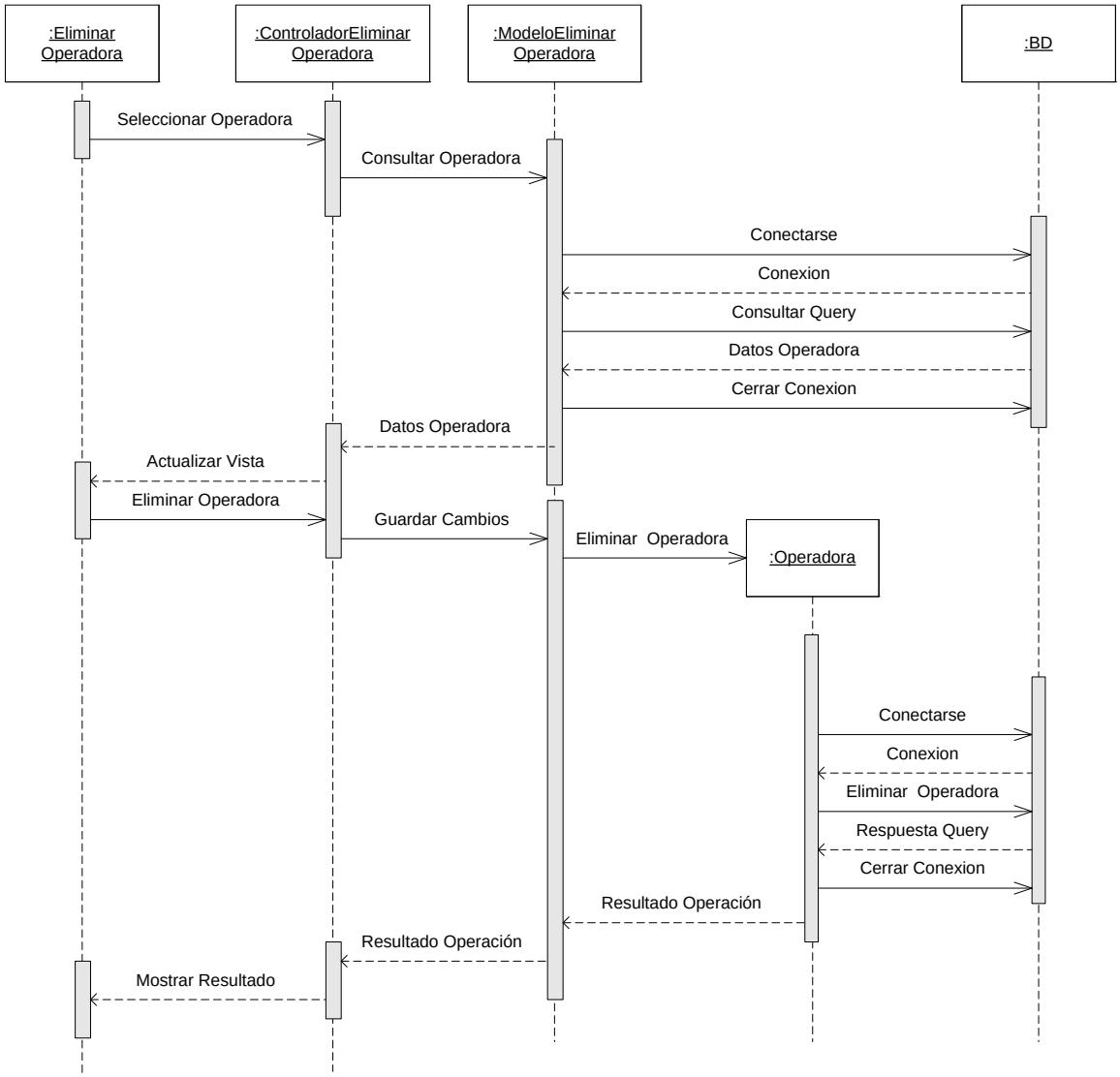

Figura 28. Diagrama de secuencia para eliminar una operadora

Diagrama de secuencia donde se agrega un nuevo número a una operadora existente en el sistema:

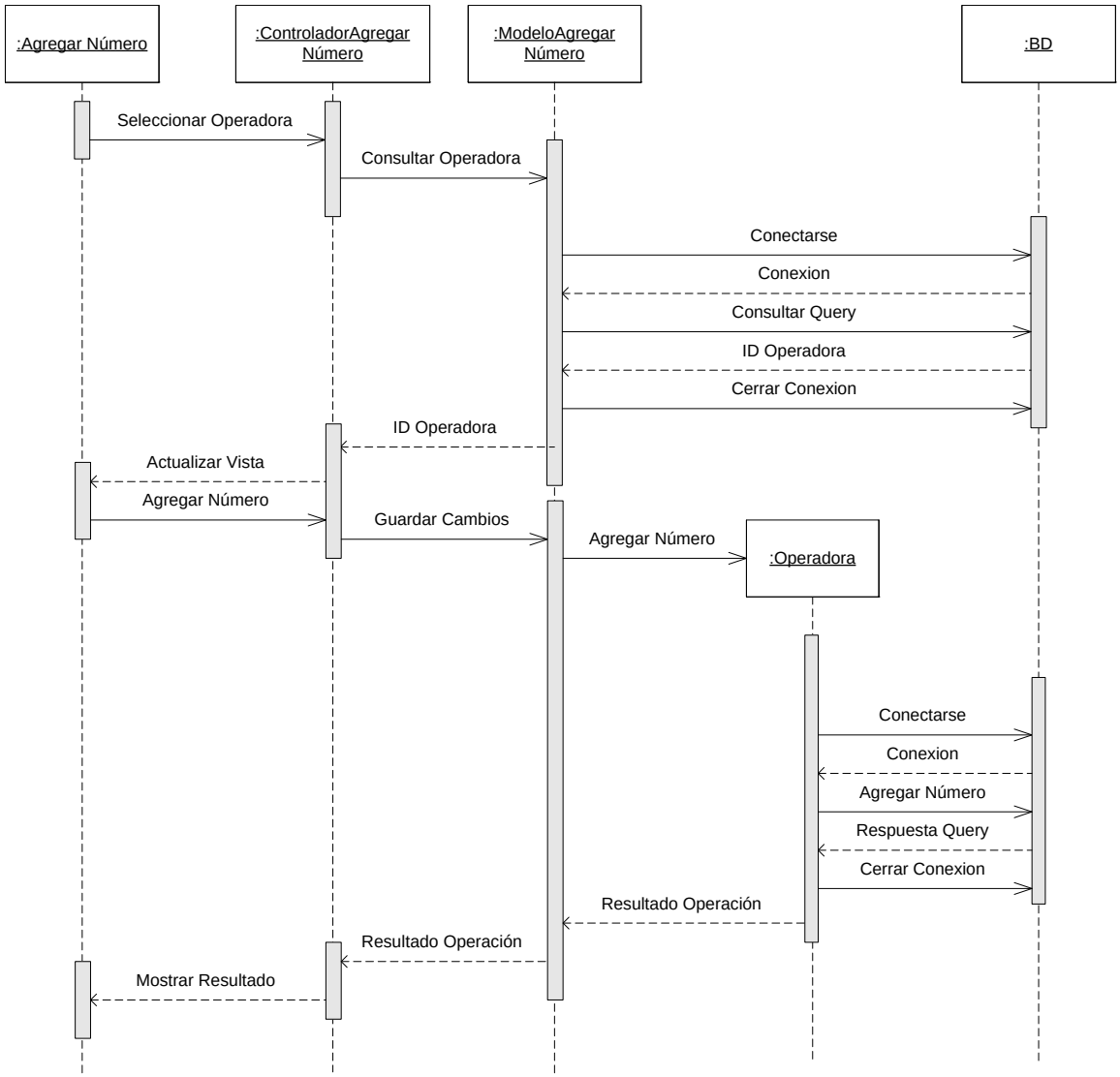

Figura 29. Diagrama de secuencia para agregar un nuevo numero a una operadora

Diagrama de secuencia donde se elimina un número de una operadora existente en el sistema:

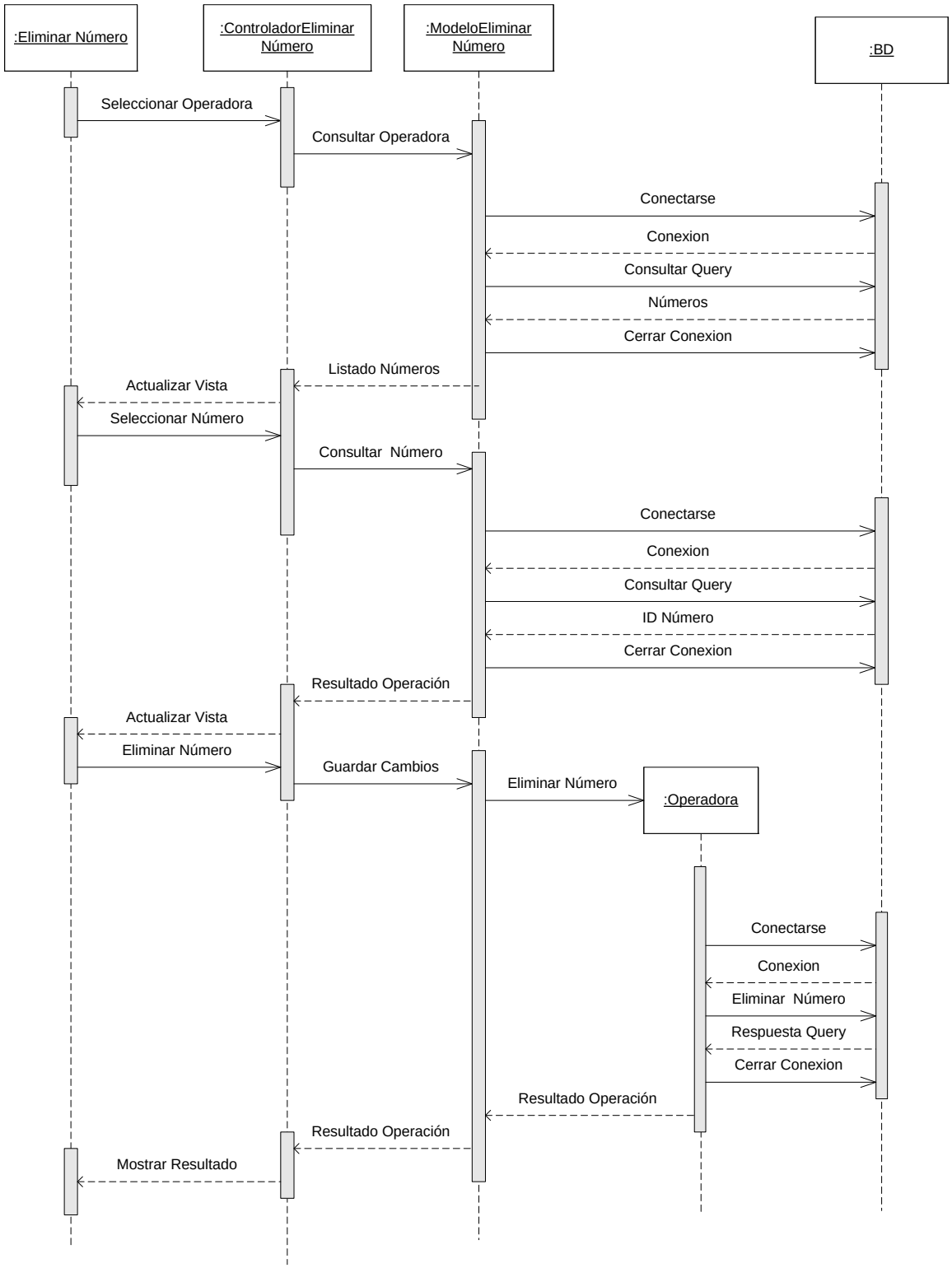

Figura 30. Diagrama de secuencia para eliminar un numero de una operadora

Diagrama de secuencia donde se configura un nuevo comando para una operadora existente en el sistema:

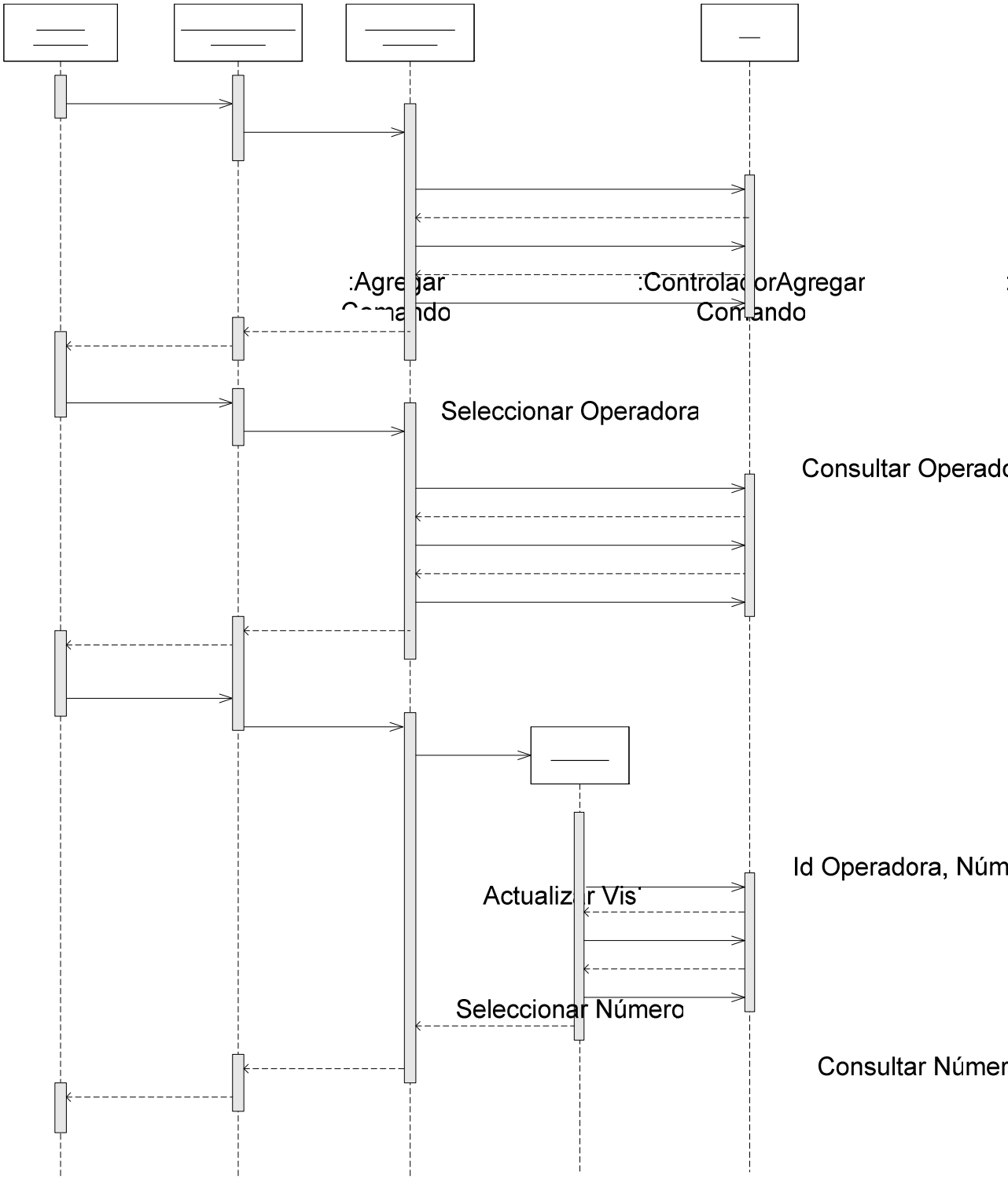

Figura 31. Diagrama de secuencia para configurar un nuevo comando

Diagrama de secuencia donde se modifican los parámetros de configuración de un comando:

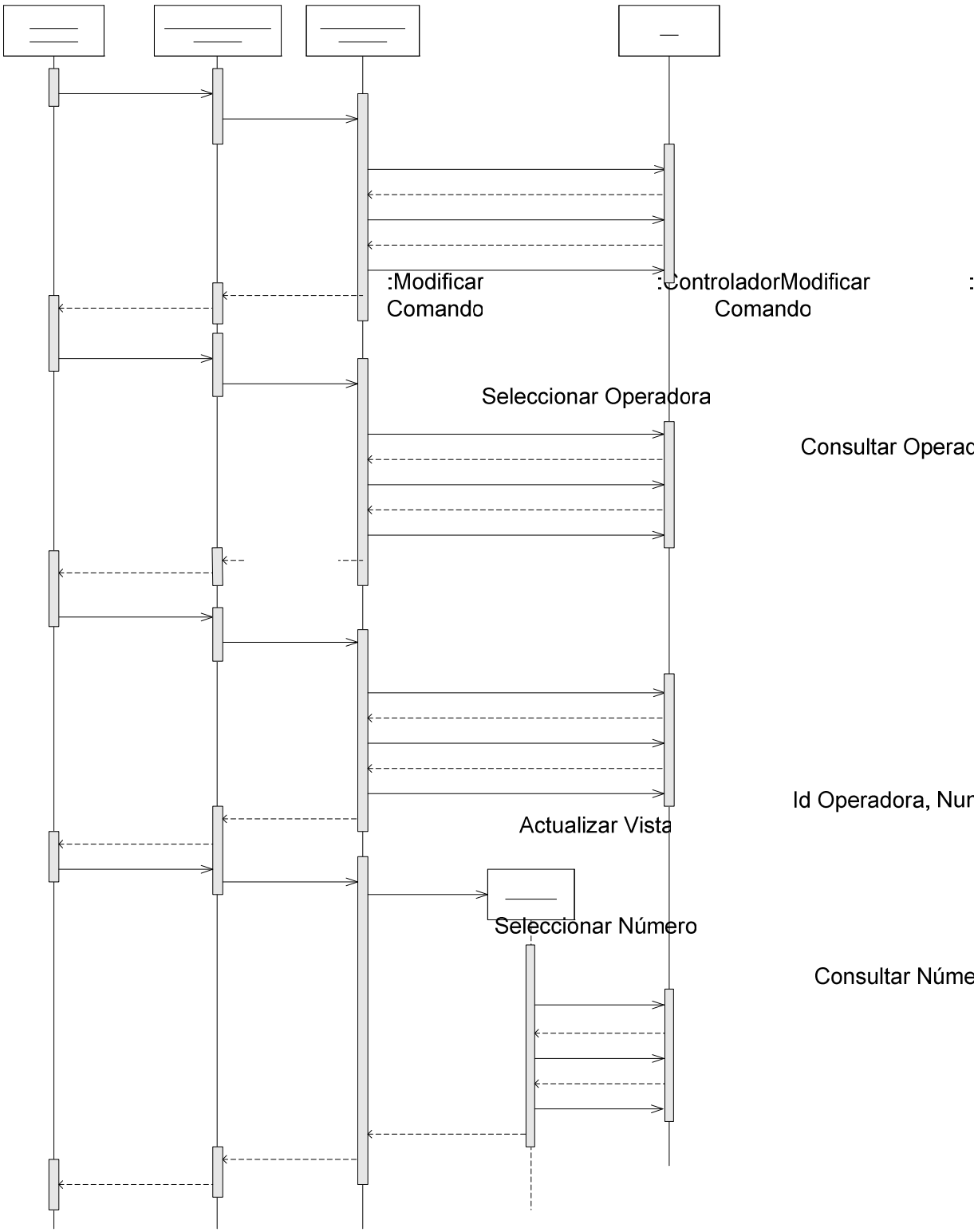

Figura 32. Diagrama de secuencia para modificar los comandos

Diagrama de secuencia donde se elimina la configuración de un comando existente de una operadora:

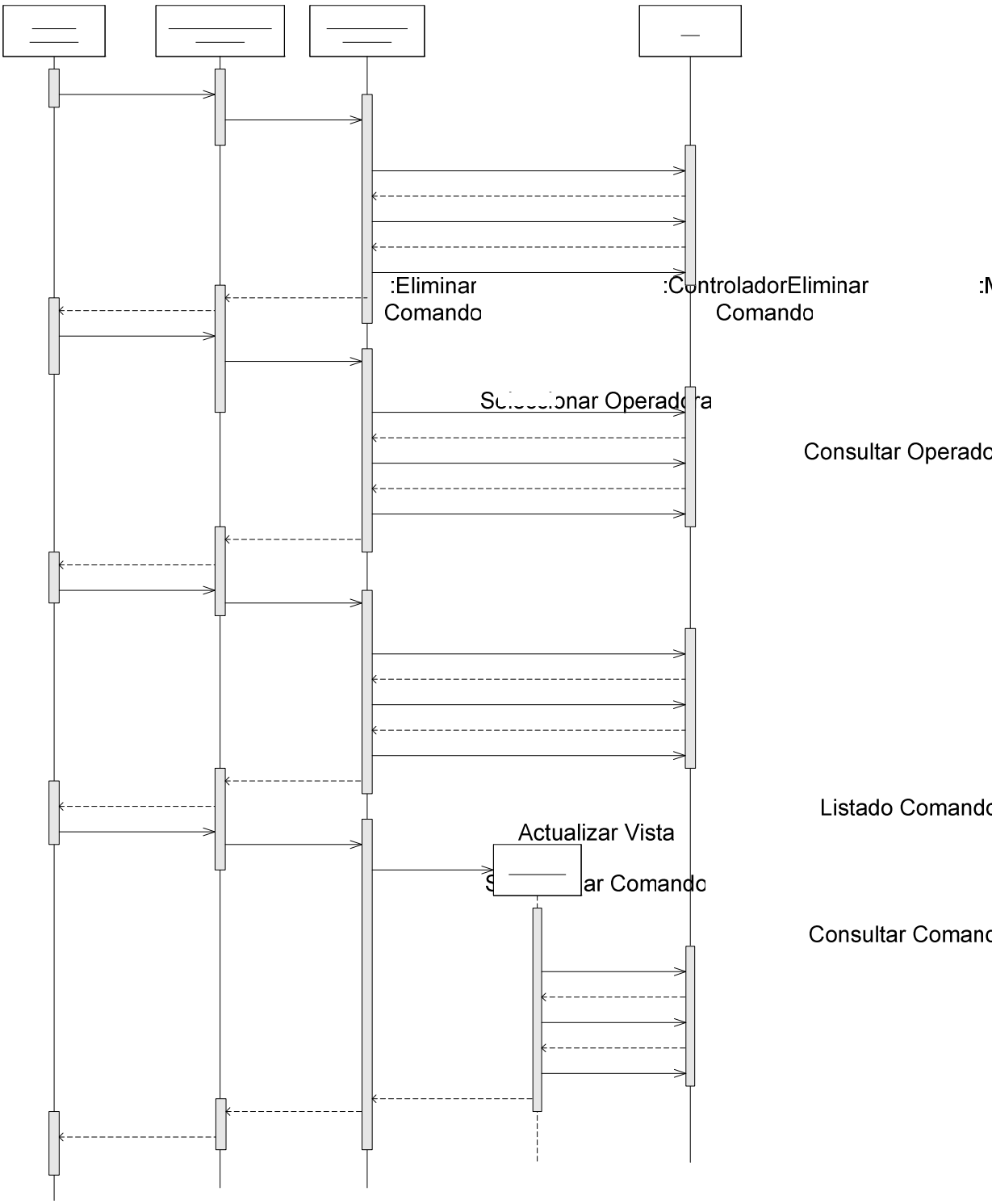

Figura 33. Diagrama de secuencia para eliminar un comando configurado

Diagrama de secuencia donde se activa el envío de mensajes SMS:

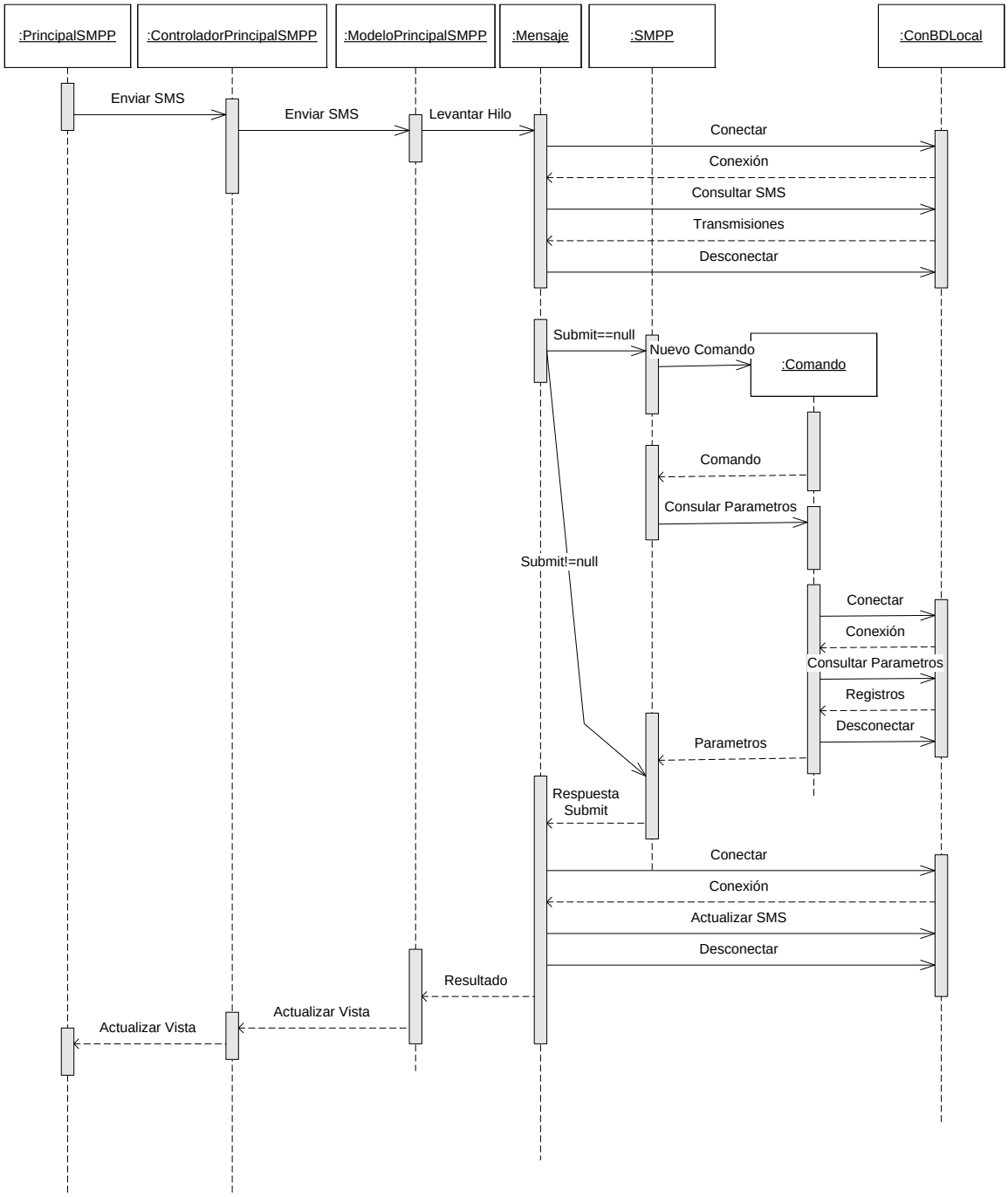

Figura 34. Diagrama de secuencia para el envío de mensajes

Diagrama de secuencia donde se realiza la carga de respuestas para su publicación WEB:

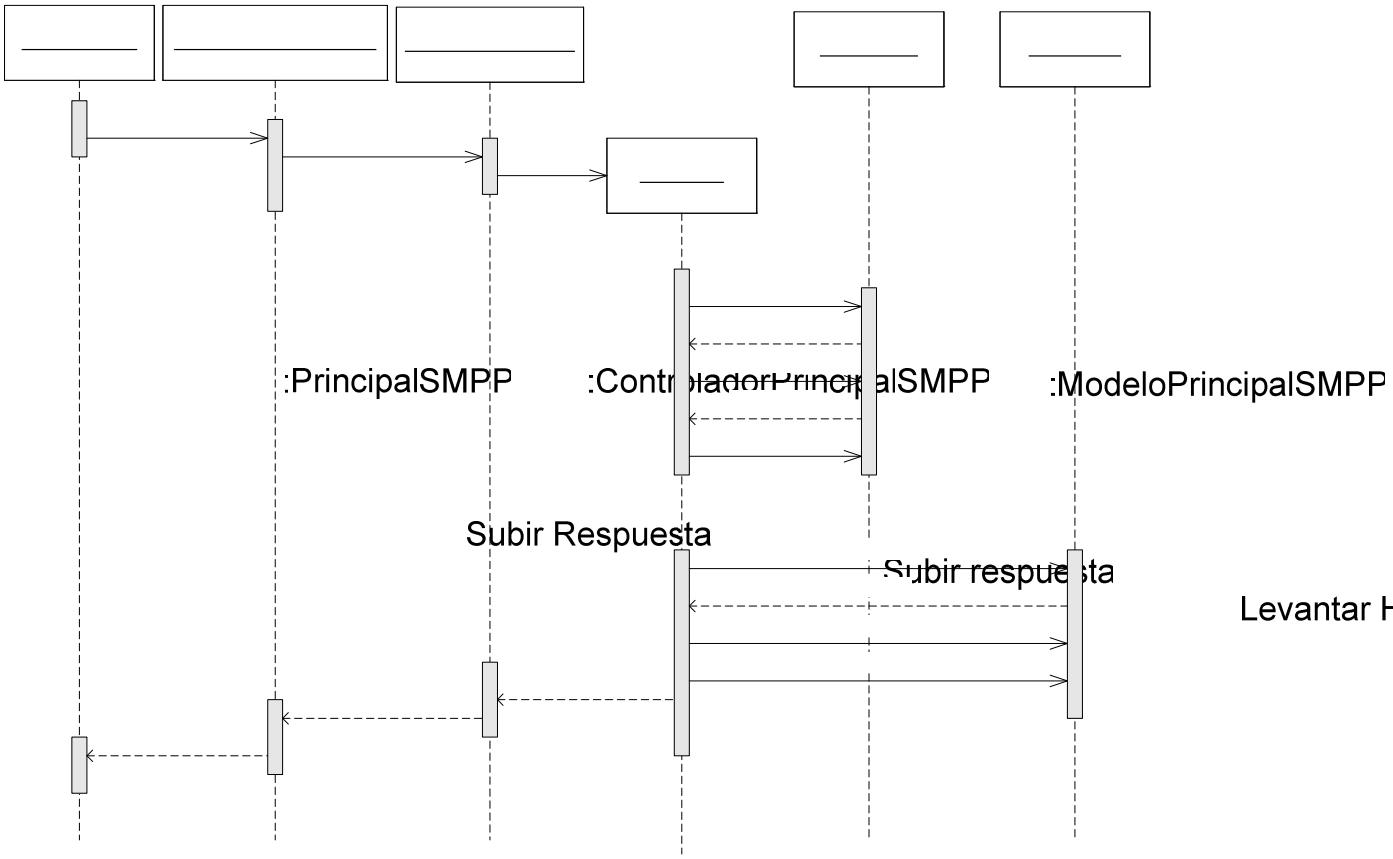

Figura 35. Diagrama de secuencia para publicar las respuestas en la web

Diagrama de secuencia donde se realiza la consulta del estatus de un SMS en particular:

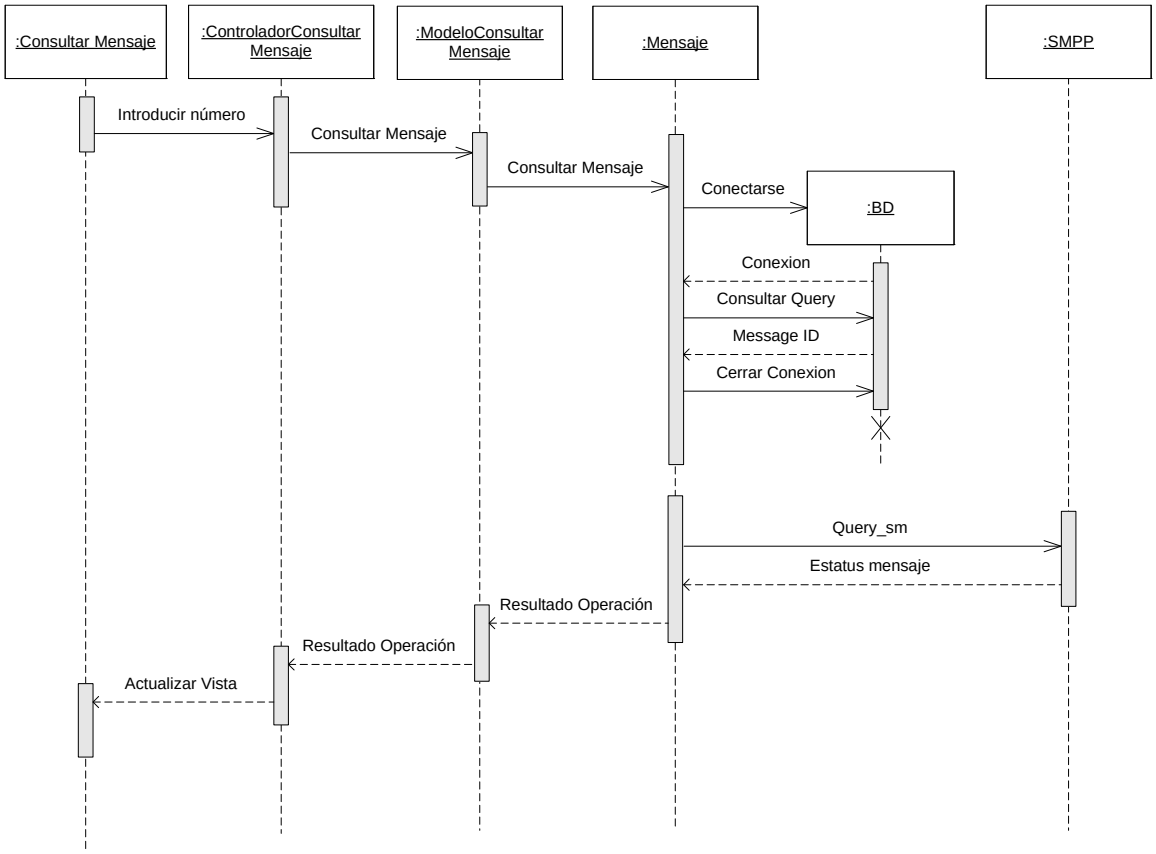

Figura 36. Diagrama de secuencia para consultar el estatus de un SMS

CAPÍTULO V. Implementación

A lo largo de este capítulo, se presentará, cómo el uso de ciertas herramientas que ayudaron al desarrollo del sistema y a la verificación del correcto funcionamiento del mismo, adicionalmente, se mostrará un análisis de la clase clave del sistema, como lo es, la clase SMPP así como también las diferentes interfaces a través de las cuales el usuario interactúa con el sistema.

5.1 Descripción de las herramientas utilizadas:

5.1.1 WIRESHARK

Antes conocido como Ethereal, es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica para educación. Cuenta con todas las características estándar de un analizador de protocolos. Posee una interfaz gráfica y muchas opciones de organización y filtrado de información. Así, permite ver todo el tráfico que pasa a través de una red (usualmente una red Ethernet, aunque es compatible con algunas otras).

Permite examinar datos de una red viva o de un archivo de captura salvado en disco. Se puede analizar la información capturada, a través de los detalles y sumarios por cada paquete. Wireshark incluye un completo lenguaje para filtrar lo que queremos ver y la habilidad de mostrar el flujo reconstruido de una sesión de TCP.

Wireshark es software libre, y se ejecuta sobre la mayoría de sistemas operativos Unix y compatibles, incluyendo Linux, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X, así como en Microsoft Windows [9].

Entre las características destacan [9]:

- ✓ Licencia GPL .
- ✓ Trabaja tanto en modo promiscuo como en modo no promiscuo.

- ✓ Puede capturar datos de la red o leer datos almacenados en un archivo (de una captura previa).
- ✓ Basado en la librería pcap.
- ✓ Tiene una interfaz flexible.
- ✓ Capacidades de filtrado.
- ✓ Admite el formato estándar de archivos tcpdump.
- ✓ Reconstrucción de sesiones TCP.
- ✓ Se ejecuta en más de 20 plataformas.
- ✓ Es compatible con más de 480 protocolos.
- ✓ Puede leer archivos de captura de más de 20 productos.

Con esta herramienta, se pudo llevar a cabo un análisis de cada uno de los paquetes, tanto enviados como recibidos, por la clase SMPP para corroborar que cumplieran con las especificaciones dictadas por el estándar del protocolo [9].

5.1.2 ECLIPSE

Eclipse es una plataforma de desarrollo código abierto basada en Java. Es un desarrollo de IBM cuyo código fuente fue puesto a disposición de los usuarios. En sí mismo Eclipse es un marco y un conjunto de servicios para construir un entorno de desarrollo a partir de componentes conectados (*plug-in*). Hay *plug-ins* para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en C/C++, COBOL, etc [10].

Entre las características principales tenemos [10]:

- ✓ Editor visual de sintaxis coloreada.
- ✓ Compilación en tiempo real.
- ✓ Modifica e inspecciona valores de variables.
- ✓ Notificación de errores mediante uso de ventanas secundarias.
- ✓ Depuración de código.
- ✓ Soportado por varios Sistemas Operativos.
- ✓ Extensible a través de *plug-ins.*

Gracias a dichas características, se facilitó, considerablemente, el proceso de desarrollo de cada una de las clases que conforman el sistema. Adicionalmente, basados en los plug-ins disponibles para Eclipse, se realizó el diseño de cada una de las interfaces de una manera mucho más eficiente [10].

5.2 Análisis de la Implementación de la Clase SMPP:

A continuación se presentan algunos detalles de la clase SMPP, donde se encuentran implementados los comandos necesarios para satisfacer los requerimientos de la empresa, según las especificaciones del protocolo estudiado.

Entre las variables, es necesario mencionar las del tipo ClaseSocket, a través de las cuales se establecen las conexiones, para el inicio de sesión ante el SMSC, y los flujos de entrada y salida de cada una de estas, para permitir la lectura y/o escritura de los distintos PDU de solicitud y respuesta.

```
public class ClaseSocket {
    private Socket socket = null;
    protected PrintStream output = null;
    protected DataInputStream input = null;
    private String mensaje = null;
    private String messageid = new String();

    //Constructor
    public ClaseSocket() {
          super();
    }

    //Metodo para establecer la conexion
    public void crearSocket(String ip, int puerto) {
          try {
                socket = new Socket(ip, puerto);
                mensaje = "Connected with server " +
                            socket.getInetAddress() + ":" +
                            socket.getPort();
                input =
                      new
                DataInputStream(socket.getInputStream());
                output =
                      new
                PrintStream(socket.getOutputStream(),true);

          }
                ....
```

También es importante destacar los objetos tipo Comando, donde el nombre dado a la variable coincide con cada uno de los comandos SMPP desarrollados en los métodos de esta clase y que almacenan la información necesaria para cada parámetro en un arreglo del tipo Parámetro, de modo tal que en cada posición se pueda encontrar atributos como nombre, valor, tipo y longitud, además de los métodos y funciones para recuperar y obtener la data de los parámetros.

```
public void obtenerComando(int idOperadora, String nroOrigen,
                                String comando){
          ResultSet rs = null;
          int idNumero, i;
          String nombre, valor, tipo;
          int lenght;


          conexionBDLocal.conectar();
          rs =
          conexionBDLocal.consultar("C_NumeroOrigen","IdOperadora = "
          + idOperadora + " AND NroOrigen = '" + nroOrigen + "'");
          try {
                if(rs.next()){
                      idNumero = rs.getInt("IdNumero");
                      rs =
           conexionBDLocal.consultar("C_ValorParametros","IdNumero =
                " + idNumero + " AND ComandoSMPP = '" + comando +
                "'");
                      parametros =
                       new Parametro[conexionBDLocal.getResultSetSize(rs)];
                      i = 0;
                      while(rs.next()){
                            nombre = rs.getString("Parametro");
                            valor = rs.getString("Valor");
                            tipo = rs.getString("Tipo");
                            lenght = rs.getInt("Longitud");
                            parametros[i] = new Parametro(nombre,
                                        valor, tipo, lenght);
                            i++;
                      }
                ....
```

```
   ....


   public int logitudParametros(){
            int longitud = 8; // command_lenght + secuence_number

            for(int i = 0; i < parametros.length; i++){

      if(parametros[i].getTipo().equalsIgnoreCase("int")){
                        longitud = longitud +
                                    parametros[i].getLongitud();
                  }
                  if(parametros[i].getTipo().equalsIgnoreCase("c-
octet")){
                        if(parametros[i].getValor() != null){
                              longitud = longitud +
                                    parametros[i].getValor().length()
+
                                    1;
                        }else{
                              longitud++;
                        }
                  }

      if(parametros[i].getTipo().equalsIgnoreCase("octet")){
                        if(parametros[i].getValor() != null){
                              longitud = longitud +

                              parametros[i].getValor().length();
                        }
                  }
            }
            return longitud;
      }

      public String obtenerValorParametro(String parametro) {
            String valor = null;

            for(int i = 0; i < parametros.length; i++){

      if(parametros[i].getNombre().equalsIgnoreCase(parametro)){
                        valor = parametros[i].getValor();
                  }
            }

            return valor;
      }
```

Adicionalmente se tiene un arreglo de enteros denominado *secuenceNumbers*, en donde se almacena el número de secuencia correspondiente al PDU de solicitud, enviado desde la ESME hasta el SMSC, de modo que se pueda controlar el número de solicitudes sin respuesta asociada.

Otra variable a destacar es *pdu_buffer*, ya que es la estructura donde se almacenan los datos correspondientes al PDU a enviar.

La clase cuenta con 2 métodos constructores que son llamados según la restricción dada por parte de la operadora, que especifica el tipo de enlace permitido por el SMSC.

```
   // Firma del Constructor usado para establecer conexiones Tx y Rx
      public SMPP(ClaseSocket socketTx, ClaseSocket socketRx, int
idOperadora, String nroOrigen, int numeroDeComandos, ConBD con,
String ip, int puerto,ModeloPrincipalSMPP principalSMPP) {
            super();
      }
   ....
```

```
   ...


      // Firma del Constructor usado para establecer conexion TRx
      public SMPP(ClaseSocket socketTRx, int idOperadora, String
nroOrigen, int numeroDeComandos, ConBD con, String ip, int
puerto,ModeloPrincipalSMPP principalSMPP){
            super();
      }
```

Por otra parte, se cuenta con funciones y métodos que resuelven problemas específicos y comunes a la mayoría de los comandos implementados. Estos son los siguientes:

- ✓ *añadirNroSecuencia*: Método encargado de ubicar una posición disponible en el arreglo *secuenceNumbers* y colocar en esa posición el valor actual de la variable *secuenceNumber*.

```
private void añadirNroSecuencia() {
      for(int i = 0; i < secuenceCommands.length; i++){
            if(secuenceCommands[i] == 0){
                  secuenceCommands[i] = secuenceNumber;
                  break;
            }
      }
}
```

✓ *seekAndDestroySecuenceNumber*: Método, que toma como parámetro el número de secuencia recibido en un PDU, correspondiente a la respuesta de una solicitud emitida por la ESME, a través del cual se recorre el arreglo *secuenceNumbers* en búsqueda de este valor y es sustituido por aquel que determina que la posición se encuentra disponible.

```
public void seekAndDestroyASecuenceNumber(int j){
      for(int i = 0; i < secuenceCommands.length; i++){
            if(secuenceCommands[i] == j){
            secuenceCommands[i] = 0;
      }
}
```

✓ *PDUPosNul*: Función cuyos parámetros son un valor de inicio y la estructura que contiene el PDU recibido por parte del SMSC, encargada de determinar y retornar la posición del próximo caracter nulo de la estructura a partir de la posición dada como inicio.

```
public int PDUPosNul(byte[] pdu, int ini){
      boolean bandera = false;
      int pos = ini;
      if(pos < pdu.length - 4){
            while ((pos < pdu.length) && (bandera == false)){
                  if (pdu[pos] != bytenulo){
                        pos = pos + 1;
                  }else{
                        bandera = true;
                  }
            }
            return pos;
      }else{
            return 0;
      }
}
```

Para cada uno de los métodos que corresponden a las operaciones del protocolo SMPP, se hace uso de funciones implementadas en la clase Comando, a través de las cuales es posible recuperar la información referente a cada comando desde la base de datos, y calcular la longitud del PDU a enviar al SMSC, para proceder al armado del PDU según las especificaciones del protocolo. En el caso de los comandos bind_tx, bind_rx y bind_trx la lectura de las respuestas generadas por el SMSC se hacen de forma inmediata, justo después de cada solicitud. Por otra parte, si se recibe un paquete correspondiente a deliver_sm la respuesta se envía inmediatamente. Para el resto de los comandos, se implementó la clase HiloRecepcionTx e HiloRecepcionRx, mediante la cual se mantiene una lectura constante del socket en búsqueda de PDU de solicitud o respuesta, generados por el SMSC.

Para realizar las pruebas correspondientes al envío y recepción de los PDU se utilizó la herramienta Wireshark, mediante la cual se capturaban los paquetes enviados y recibidos, para comparar con la información colocada y leída por la plataforma en los paquetes y verificar que estos sean correctos. En la Figura 37, se muestra la captura de un *Submit_Sm* donde se aprecia el texto contenido en el mensaje SMS “Mensaje de Prueba”.

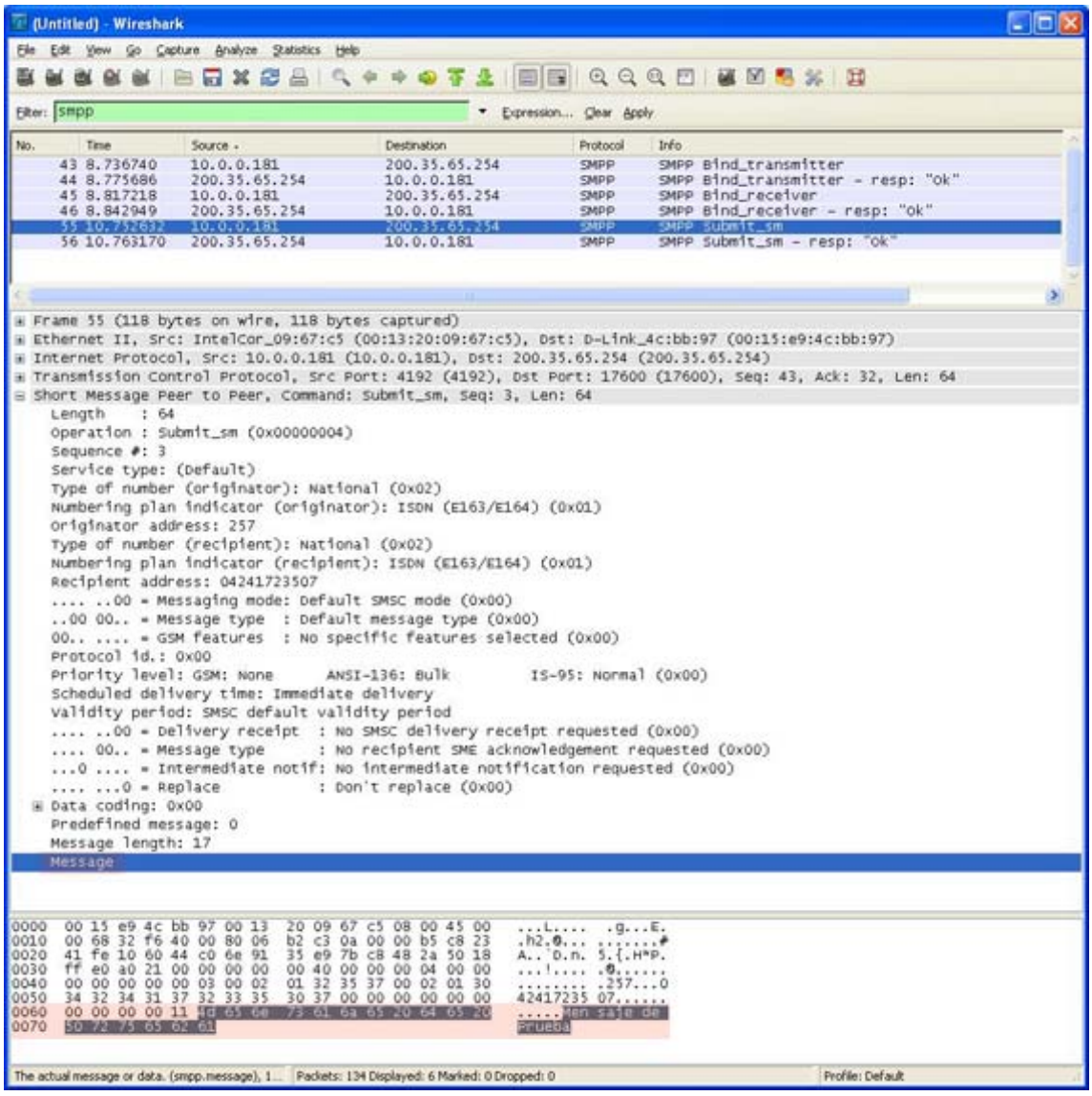

Figura 37. Captura del comando Submit_sm con la herramienta Wireshark

5.3 Desarrollo de interfaces:

Las imágenes que se muestran a continuación (Figuras 35-40) muestran las diferentes vistas que se proporcionan para el manejo de la Plataforma de Envío y Recepción SMPP.

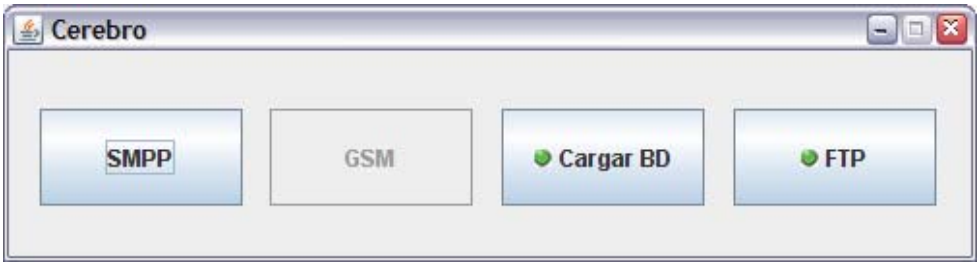

Figura 38. Ventana de selección de operaciones

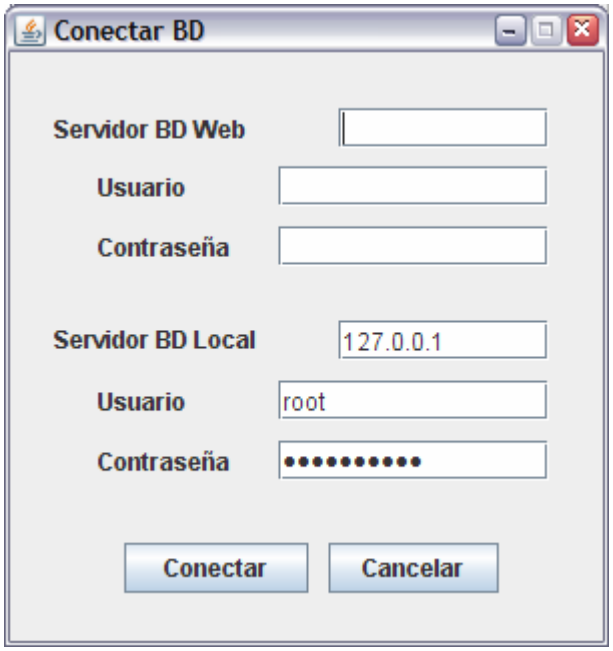

Figura 39. Ventana de conexión al servido de Base de Datos

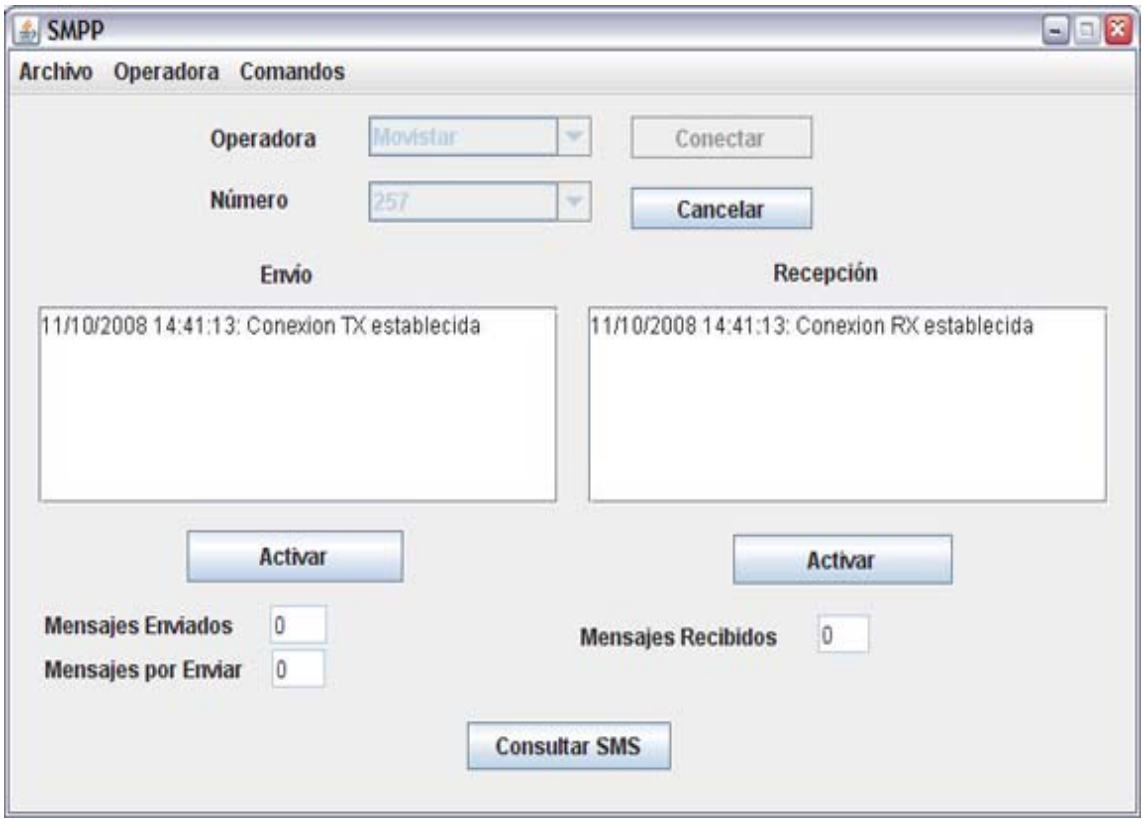

Figura 40. Ventana principal de la aplicación

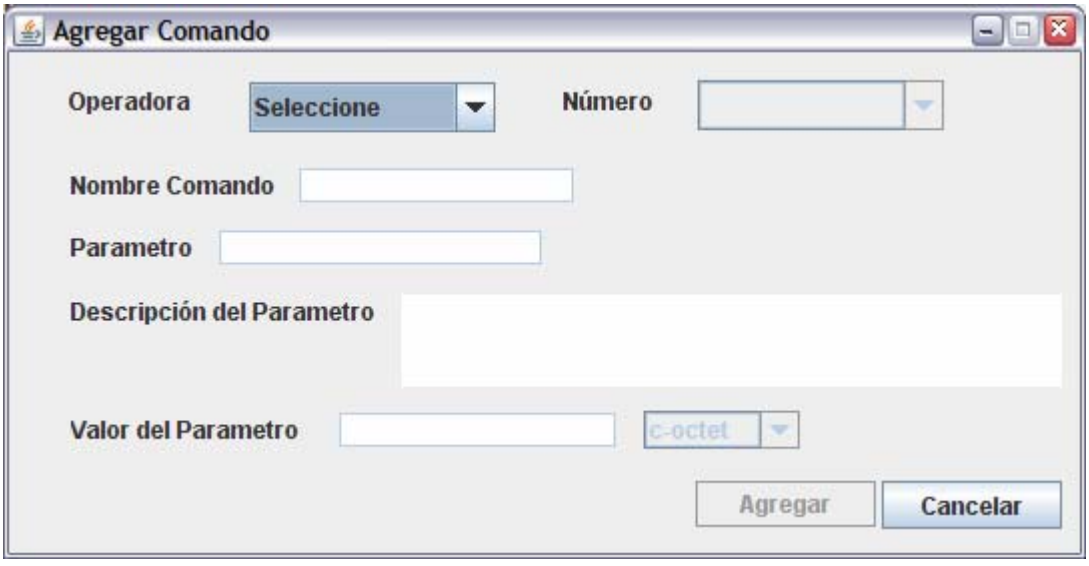

Figura 41. Ventana que permite agregar un nuevo comando

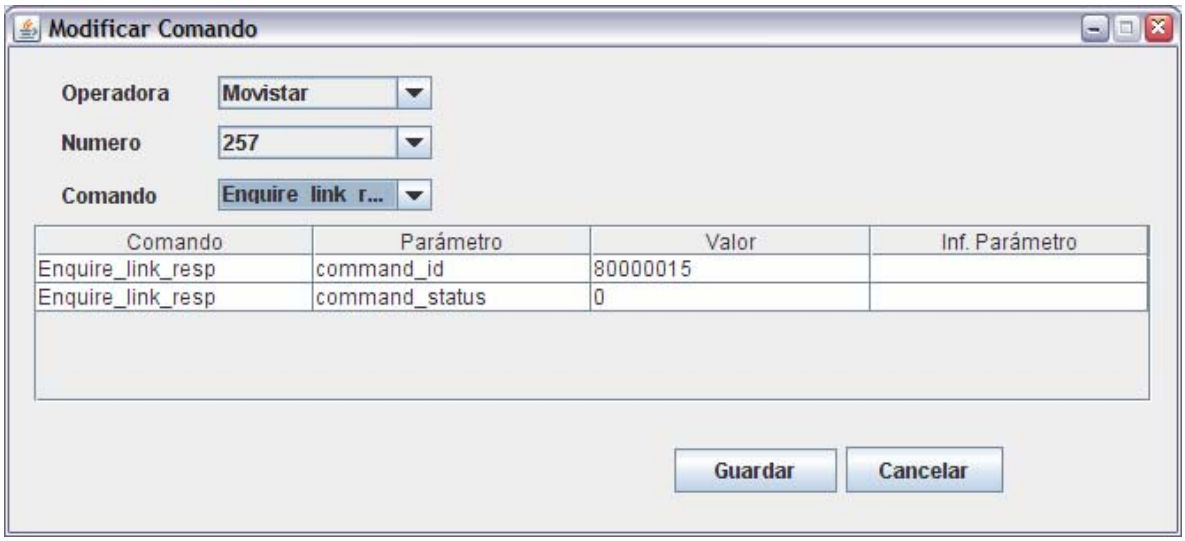

Figura 42. Ventana para modificar parámetros de comandos

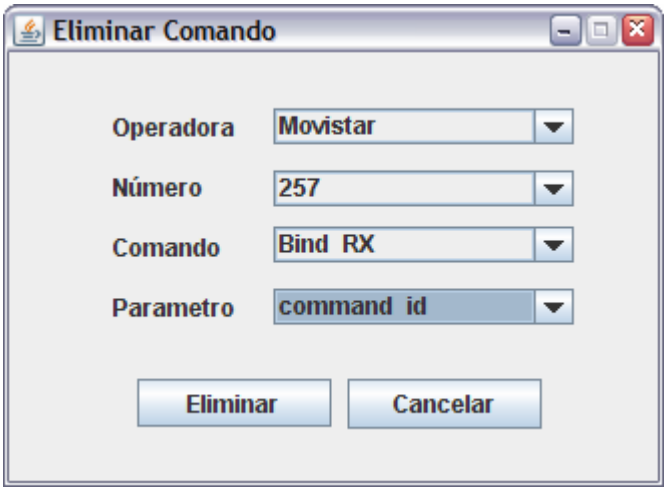

Figura 43. Ventana de Eliminación de comando/parámetros

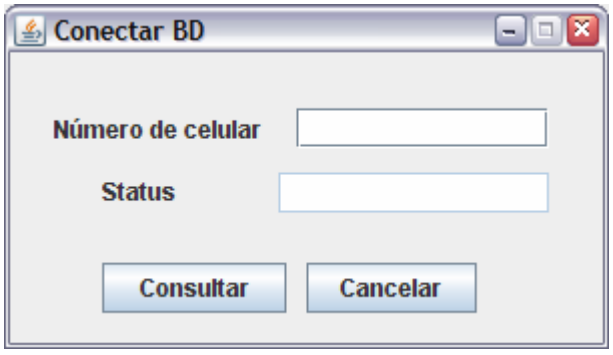

Figura 44. Ventana que permite consultar el estatus de un mensaje

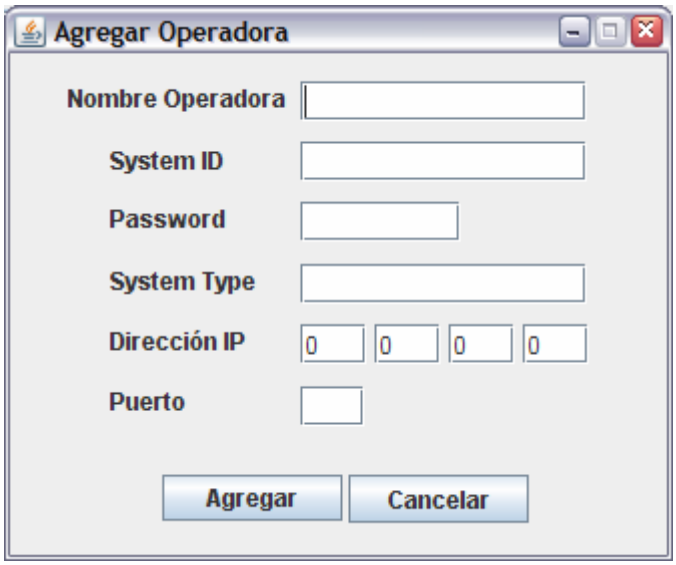

Figura 45. Ventana que permite crear una nueva operadora

Figura 46. Ventana que permite modificar una operadora

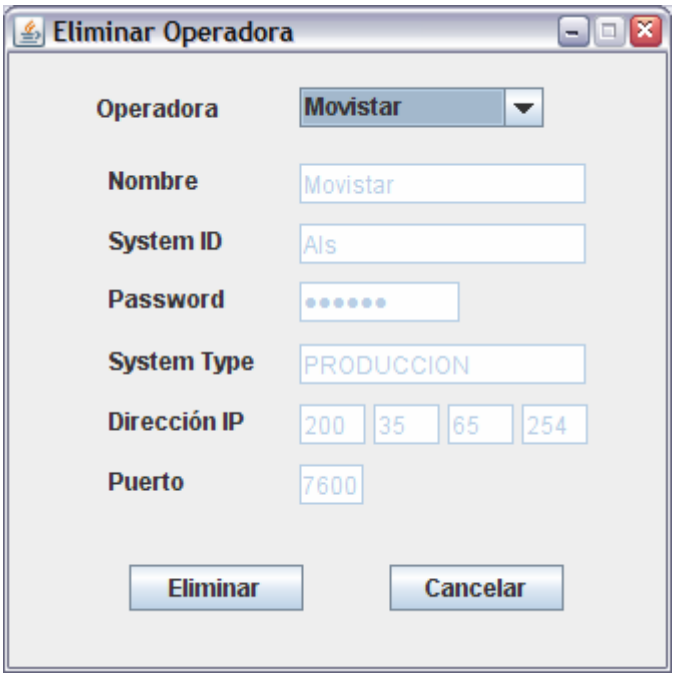

Figura 47. Ventana que permite eliminar una operadora

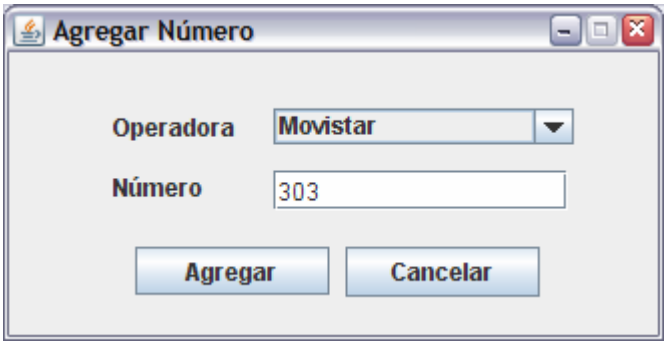

Figura 48. Ventana que permite agregar un número a una operadora

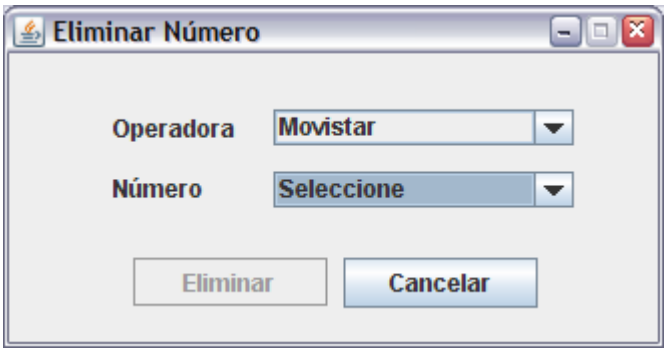

Figura 49. Ventana que permite eliminar un número a una operadora

CAPÍTULO VI. Fase de Pruebas

Para el desarrollo de esta plataforma, se ejecutaron pruebas en cada una de las iteraciones, con el fin de constatar la correctitud de la implementación y, a través de las cuales, se pudieron identificar fallas que fueron solventadas en la iteración siguiente del ciclo.

Entre los tipos de prueba que se llevaron a cabo, tenemos:

- ✓ **Pruebas de Interfaces:** para la aplicación, se adoptó como meta la elaboración de interfaces usables que contemplan los requerimientos y lineamientos de ALS Comunicaciones, para lo que se estudió y evaluó cada una de estas bajo los aspectos de claridad, facilidad de uso y consistencia, de modo de reducir el tiempo de aprendizaje de la nueva herramienta. Se les solicitó a los usuarios que estarían interactuando con el sistema, que luego de utilizar éste, dieran su opinión sobre la experiencia para así poder verificar el correcto funcionamiento de la interfaz.

- ✓ **Pruebas Unitarias:** cada una de las secciones de código desarrollado en la fase de implementación, se elaboraron bajo ciertas especificaciones, por lo cual es fundamental que realicen las acciones requeridas de forma correcta, como por ejemplo la inserción de los parámetros correspondientes a un comando, asociado a una operadora; o la ejecución de cada uno de los comandos implementados del protocolo. Se validó la correcta implementación de los comandos, mediante la captura de paquetes de red, haciendo uso de la herramienta Wireshark.

- ✓ **Pruebas de Consistencia de Datos:** puesto que los datos almacenados en la BD son fundamentales para el buen funcionamiento de la plataforma, se llevaron a cabo pruebas para constatar que la data obtenida a través de las interfaces era almacenada correctamente en las tablas correspondientes, al igual que la información mostrada en las interfaces era justo la que se encuentra almacenada y que la data recuperada y actualizada para el envío y recepción, respectivamente, de mensajería de texto, se realizara de forma correcta. Se constato que no se almacenaran caracteres especiales que están contenidos en los parámetros

que conforman los comandos SMPP, por ejemplo en el caso de un *deliver_sm*, el texto de contenido dentro del parámetro *short_message*, contiene al final de la cadena, caracteres que pondrían en riesgo el buen funcionamiento de la aplicación.

- ✓ **Prueba de Integración:** finalmente con la prueba de integración se permite evaluar la herramienta completamente, de modo de poder verificar que todas las partes del sistema funcionan en conjunto. Se verificó que los módulos desarrollados interactuaban de forma correcta, insertando datos a través de las herramientas de configuración de la aplicación, para realizar posteriormente envíos y recepción tanto de comandos SMPP como de mensajes cortos de texto. Así mismo se verifico la modificación y eliminación de operadoras creadas por el usuario, comandos y parámetros existentes, cumpliendo con restricciones especificas.

CAPÍTULO VII. Conclusiones

Con el desarrollo de este sistema, se obtuvieron conocimientos en el protocolo estándar, Short Message Peer to Peer (SMPP), el cual permite que entidades de envío de SMSs que subyacen fuera de la red móvil (ESME - External Short Message Entities) puedan interconectarse con los elementos internos como lo es el SMSC.

La aplicación se desarrolló de manera modular, dando la posibilidad que ésta sea fácilmente escalable, ya que la empresa podría solicitar que se le agregasen nuevos módulos o funcionalidades. Este tipo de desarrollo también evita que cambios en una funcionalidad afecte a otra que ya se encuentre estable.

Al ser el sistema configurable para cualquier operadora, permite no solamente abarcar un mercado local sino que también puede extenderse a nivel mundial.

Finalmente, se puede decir que se cumplió con el objetivo general, el cual fue desarrollar un sistema capaz de realizar envío y recepción de mensajería corta de texto a través del protocolo SMPP; lo que permite que la capacidad de la empresa se vea incrementada, con la posibilidad de captar nuevos clientes y crear nuevos y mejores servicios a ofrecer.

Algunas propuestas a considerar para posibles trabajos futuros, son las siguientes:

- ✓ Implementar una utilidad que permita realizar el envío de un único mensaje desde la interfaz de la aplicación.
- ✓ Agregar una opción que permita cancelar los mensajes pendientes que se encuentra en el SMSC.
- ✓ Adaptar la aplicación para que esta no sea *standalone* y pueda ser administrada de manera remota.

En un principio se tenía como objetivo especifico, integrar este desarrollo con la plataforma GSM ya existente, sin embargo, dicha integración requería de ciertas

consideraciones, las cuales se encontraban fuera del alcance de este trabajo, y limitaciones de tiempo, no permitieron completar esta tarea, por lo que la misma queda planteada por la empresa para un trabajo futuro.

Referencias

[1] ADC Telecommunications, Wireless Short Message Service Tutorial, (1999). Pág. 1-7

[2] International Engineering Consortium. Wireless Short Message Service (SMS). En: Web ProForums. (2007).
http://www.iec.org/online/tutorials/wire_sms/topic07.html

[3] Phone Scoop.IS-41 definition. En: Glossary (2008)
http://www.phonescoop.com/glossary/term.php?gid=174

[4] SMPP Developers Forum. Short Message Peer to Peer Protocol Specification v3.4. (1999). Pág. 8-107

[5] WILEY, John & Sons. Mobile Messaging Technologies and Services SMS, EMS and MMS. (2003). Pág. 44-46

[6] Project Planning and Implementing Tools
http://www.asq.org/learn-about-quality/project-planning-tools/overview/pdsa-cycle.html

[7] JAVA
http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_Java

[8] MySQL
http://dev.mysql.com/doc/refman/5.0/es/index.html

[9] Wireshark
http://es.wikipedia.org/wiki/Wireshark

[10] Eclipse (2.1) y Java, 2004. Dept. Informática, Universitat de València
http://www.uv.es/~jgutierr/MySQL_Java/TutorialEclipse.pdf