ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"ANÁLISIS E IMPLEMENTACIÓN DE UNA ENTIDAD EXTERNA DE MENSAJES CORTOS (ESME) QUE OPERA BAJO EL PROTOCOLO DE MENSAJERÍA ESCRITA PUNTO A PUNTO (SMPP) VERSIÓN 3.4 EN LINUX PARA EL ENVÍO Y RECEPCIÓN AUTOMÁTICO DE MENSAJES ESCRITOS CORTOS (SMS'S)"

INFORME DE MATERIA DE GRADUACIÓN

PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES

Presentado por:

GIANCARLO OMAR LLERENA CHÉVEZ

ROMARIO RENEE PINDA PAUCAR

GUAYAQUIL – ECUADOR

2015

AGRADECIMIENTO

En primer lugar a Dios por darme la fuerza, protección y amor. A mis padres por el apoyo constante e incondicional cada día. Y a todos mis amigos y familiares que de una u otra manera me han brindado su apoyo y buenos deseos para superarme y cumplir con mis objetivos.

Giancarlo Llerena Chévez

En primer lugar a Dios por todo lo que me ha dado y tengo hasta ahora, mis padres las bases de esta edificación que se fue haciendo paso a paso a mis amigos que fui conociendo a lo largo de mi carrera y demás familiares que se preocuparon y estuvieron siguiendo mi progreso. Y a un amigo de mil batallas Javier Villacres.

Romario Renee Pinda Paucar

DEDICATORIA

El presente proyecto se lo dedico a mi abuela Petra Morán Cantos, a mi padre Giovanny Llerena y a mi madre Teresa Chévez quienes han estado conmigo brindándome su apoyo incondicional en todo momento y por medio de sus enseñanzas y consejos han contribuido de manera sustancial en mi desarrollo personal y profesional.

Giancarlo Llerena Chévez

Dedico este proyecto primero a Dios y luego a mi padre Salvador Pinda Pérez y a mi madre Aida Paucar Guijarro quien me ayudó en esos momentos de flaqueza, una mujer luchadora en todo sentido y con su fuerza de carácter me deja un legado que nunca lo olvidare, a mis hermanos y demás familiares que me dieron ese aliento de seguir adelante.

Romario Renee Pinda Paucar

TRIBUNAL DE SUSTENTACION

_____

Ing. José Miguel Menéndez, Msc.

PROFESOR DE LA MATERIA DE GRADUCAIÓN

_____

Ing. Jorge Rodríguez, Msc.

PROFESOR DELEGADO POR LA UNIDAD ACADÉMICA

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Informe, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Art. 12 del Reglamento de Graduación de la ESPOL)

_____

GIANCARLO OMAR LLERENA CHÉVEZ

_____

ROMARIO RENEE PINDA PAUCAR

RESUMEN

El presente proyecto fue realizado con la finalidad de explicar las características del protocolo SMPP (Protocolo De Mensajería Escrita Punto A Punto), así como también los comandos y parámetros que este necesita para su funcionalidad.

Para entender mejor este protocolo, se implementará un ESME (Unidad Externa de Mensajes Cortos) que interactuará con una SMSC (Central de Servicios de Mensajes Cortos) que también debe implementar el protocolo SMPP para que entre ellos se puedan comunicar.

El ESME responderá solicitudes de los usuarios de manera automática a través de la SMSC. Estas solicitudes son palabras claves que envían los usuarios móviles, y que posteriormente el ESME escogerá y enviará un contenido asociado a esta palabra como respuesta a esta solicitud.

ÍNDICE GENERAL

ABREVIATURAS Y SIMBOLOGÍA

CDR	Registro detallado de llamadas
CSS	Hojas de estilo en cascada
ESME	Entidad Externa para transferencia de mensajes cortos
FRAMEWORK	Marco de trabajo
HTML	Es un lenguaje de marcas utilizado para el desarrollo de páginas web
HTTP	Protocolo de transferencia de hipertexto
IP	Protocolo de Internet
JQUERY	Librerías hechas en JAVASCRIPT para facilitar el manejo de eventos y datos en el lado del cliente
LAN	Red de Área Local
MC	Centro de mensajes
MVC	Patrón de arquitectura de software Modelo-Vista-Controlador
OCTETOS	Unidad de información formada por ocho dígitos binarios
PDU	Unidades de datos del protocolo
QUERIES	Consultas que se realizan a la base de datos
RE	Entidad de Ruteo
SMPP	Protocolo de mensajería escrita punto a punto
SMS	Servicio de mensajes cortos

SMSC	Centro de servicios de mensajes cortos
SS7	Sistema de señalización número 7
TCP/IP	Modelo que posee un sistema de protocolos de red que hacen posible la comunicación entre equipos remotos
UNICODE	Estándar de codificación de caracteres con el objetivo de facilitar el tratamiento de los datos
X.25	Estándar para las redes de amplia de conmutación de paquetes

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Hoy en día gracias al avance de la tecnología móvil y la creación de diversas aplicaciones es posible transferir datos y archivos multimedia de gran tamaño, de una manera rápida y a un menor costo. Por este motivo, generalmente los usuarios prefieren usar las redes sociales para poder comunicarse y no el servicio de mensajería corta conocidos como SMS debido a la limitada cantidad de caracteres que puede contener y su alto costo; sin embargo, los mensajes cortos no dejan de ser útiles, seguros y rentables para las operadoras.

Otro uso común de los mensajes de texto cortos es el envío de datos a centros externos donde se podría obtener información de diversa índole, por ejemplo política, deportes, entretenimiento o noticias, esto depende a la organización al cual se está solicitando la información.

A estos centros externos donde procesan los mensajes cortos de los usuarios móviles se los denominan ESME conocidos en español como entidad externa de mensajes cortos que, haciendo uso del protocolo de SMPP que significa protocolo de mensajería escrita punto a punto que hacen posible la transferencia confiable y eficiente de mensajes cortos de textos.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÓN

1.1 Antecedentes

SMPP es un protocolo que no pertenece al conjunto de protocolos del sistema de señalización que en sus siglas significa SS7, ya que su implementación se la realiza fuera de la red telefónica. Es un protocolo que originalmente fue desarrollado por Lan J, Chambers con el fin de transmitir mensajes de textos cortos sin usar la red móvil. Actualmente en la operadoras celulares implementan este protocolo en la versión 3.3 y 3.4. Una de las diferencias sustanciales entre estas dos versiones es que la versión 3.4 implementa la sesión *transceiver* que

es aquella que permite recibir y enviar mensajes a través de sólo un hilo. En el capítulo dos se describirán detalladamente cuales son las sesiones que implementa este protocolo; sin embargo, es importante señalar que actualmente el protocolo SMPP se encuentra la versión 5, desarrollada por *SMS FORUM*.

1.2 Justificación

En este proyecto se presenta la oportunidad de aprender el protocolo SMPP a través de su implementación e integración con herramientas informáticas. En la actualidad no existen aplicaciones web realizadas con tecnologías de código abierto en el que se puede administrar al ESME de forma remota y los mensajes cortos que se envían y se reciben.

Usando las herramientas de código abierto se evita comprar el software y muchas veces adquirir una licencia por un determinado período. Los mensajes cortos no solamente se usan para la comunicación de un móvil a otro, sino que poseen múltiples ventajas, por ejemplo se podrían utilizar en sistemas de seguridades, entretenimiento o para solicitar alguna información promocionada por un ESME

1.3 Descripción del Proyecto

Para la transferencia de los mensajes cortos debe existir la participación de dos elementos: la Central de Servicio de Mensajes Cortos (SMSC) que llevará el papel de servidor, y el ESME para la transferencia de mensajes cortos que tendrá el papel de cliente. Se debe tener presente que para que puedan transferirse los mensajes correctamente, tanto el cliente como el servidor deben tener implementado el protocolo SMPP que es aquel que interpretará los mensajes transferidos entre ambos elementos. SMPP tiene la ventaja que puede ser implementado a partir del modelo TCP/IP y con esta facilidad se puede implementar en varias plataformas de desarrollo en las cuales se usan lenguaje de programación como PHP, JAVA, PERL, PYTON, VISUAL BASIC entre otros.

El proyecto busca tener un ESME dinámico en la cual no solo se enfoque en enviar o recibir mensajes de una manera manual, sino que además interactuará con una base de datos que permitirá enviar las respuestas por el protocolo SMPP de manera automática a partir de un requerimiento hecho por el cliente. Para ello se tendrá que iniciar una sesión SMPP ya sea como transmisor, receptor o transceptor, según sea el caso.

Para la interfaz del usuario se pretende desarrollar una aplicación web amigable y segura que permita administrar al ESME de manera remota por medio de un navegador web, esta es una ventaja ya que el usuario se independiza de la plataforma. En esta plataforma se pretende implementar las opciones básicas para la configuración del cliente SMPP además de una interfaz para la gestión de los contenidos dinámicos.

Con respecto a los contenidos, el proyecto no solo se enfatiza en enviar texto con contenidos cómicos o de entretenimiento, sino que buscará ofrecer un servicio brindando información útil. Por ejemplo, los estudiantes de la comunidad politécnica generalmente no se enteran de los cambios internos o procesos que ocurren dentro del campus, es por ese motivo que mensajes en donde se muestren los horarios de salidas de las unidades de buses o promociones de los bares quizás sean ejemplos de información que podría resultar útil al estudiante en cualquier momento.

Las herramientas que se usarán para el presente proyecto son: PHP como lenguaje de programación para el desarrollo de la aplicación web en el lado del servidor, HTML que es el lenguaje de marcas de

hipertexto utilizado para el desarrollo de aplicaciones web y JAVASCRIPT que es un lenguaje de programación que permite crear efectos dinámicos en las interfaces web que se muestran en el lado del cliente. Como gestor de base de datos se usará MYSQL que es un sistema gestor de bases de datos relacionales. Todos estos elementos estarán instalados en un servidor para que puedan ser accedidos por el usuario de la aplicación. Cada uno de estos elementos se los explicarán al detalle en el capítulo tres donde se mencionarán las herramientas que forman parte de la solución de este proyecto.

1.4 Objetivos

1.4.1 Objetivos generales

El presente proyecto tiene como objetivo general implementar un ESME usando el protocolo SMPP a través del modelo de red TCP/IP para la transferencia de mensajes cortos con contenido de entretenimiento e informativo. Con esta implementación se pretende explicar los procesos que se llevan a cabo para la transferencia de mensajes cortos de una manera ordenada y simple de tal manera que el lector pueda comprenderlo fácilmente.

1.4.2 Objetivos específicos

- Implementar una interfaz gráfica usando el lenguaje de programación PHP en el lado del servidor conjuntamente con HTML y JAVASCRIPT en el lado del cliente para la administración del ESME de forma remota a través de un explorador web.
- Implementar una base de datos en MYSQL para almacenar los CDR's (registros detallados de llamadas) generados a partir de los mensajes que recibe y emite el ESME.
- Almacenar los comandos y mensajes SMPP que se usan para transmitir los mensajes cortos entre la SMSC y el ESME en archivos de logs para que puedan ser accedidos por los usuarios que administran el ESME.

1.5 Alcance

En este proyecto se tiene como alcance crear un ESME e implementar el protocolo SMPP a través del lenguaje de programación PHP para el envío y recepción de mensajes cortos. Se describirá también cuáles son los pasos necesarios y cuáles son las aplicaciones que se necesitarán para poder desarrollar e instalar una aplicación en un servidor web y poder administrar al ESME de forma remota. Se explicarán también cuáles son los mensajes que se transmiten entre el ESME y la SMSC para que entre ellos se puedan conectar, desconectar, recibir y transmitir paquetes.

1.6 Limitaciones

Aunque existe la versión 5 del protocolo SMPP este proyecto implementa la versión 3.4. Esta versión posee una documentación amplia, existiendo una comunidad activa y siendo utilizado actualmente por las operadoras celulares.

Otra limitación que presenta este proyecto es que la aplicación se instalará dentro de una LAN, es decir una red de área local; así mismo, se comunicará con otra computadora en donde estará instalado en la misma red cuyo papel será el del SMSC, es decir el servidor SMPP. No habrá distinción de roles, en otras palabras todos los usuarios que accedan a la aplicación tendrán privilegios de administrador, es decir tendrán acceso a todas las opciones que brinda la aplicación web.

En este proyecto no se trabaja directamente con el celular ya que para que exista esta comunicación en la SMSC debe estar implementado otro protocolo diferente al de SMPP que permita comunicarse sin inconvenientes con el dispositivo móvil; sin embargo, se usará otro ESME provisional que simule el celular.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

2.1 Descripción del protocolo SMPP

SMPP es un protocolo que se usa para la transferencia de mensajes cortos desde un ESME hacia una SMSC, en esta transferencia el ESME se comporta como cliente mientras que la SMSC se comporta como servidor. Ambos elementos deben tener implementado el protocolo SMPP para que entre ellos puedan comunicarse. El protocolo SMPP es un protocolo que es implementado fuera de la red celular por lo que permite a entidades externas realizar cambios en

sus sistemas sin que afecte de manera directa a otros procesos en la red móvil. Este protocolo trabaja bajo el modelo TCP/IP que es un conjunto de protocolos de red que permiten la comunicación entre computadoras a través de la internet transfiriendo los datos de forma síncrona, es decir esperando una respuesta a una petición o también se puede trabajar de manera asíncrona en la cual cada petición o respuesta es independiente y se lo hará a través de distintos hilos.

2.2 Estructura general de la unidad de datos del protocolo SMPP

Antes de explicar el funcionamiento del protocolo SMPP para el envío y recepción de mensajes se explicará de manera breve cuáles son los parámetros y mensajes que se envían desde el cliente al servidor y viceversa para que el lector comprenda de una mejor manera el análisis de las sesiones que se levantan en este protocolo.

De ahora en adelante se mencionarán a los mensajes que se colocan en la cabecera y cuerpo de la estructura de los datos SMPP como PDU que significa unidad de datos de protocolo. Estos parámetros y PDU’s como son desarrollados por una firma anglosajona no se traducirán al español ya que así es como fueron denominados [1].

A continuación, en la Figura 2.1 se muestra la tabla donde se detalla la estructura de un dato SMPP.

SMPP PDU				
PDU Header (mandatory)				**PDU Body (Optional)**
command length	*command id*	*command status*	*sequence number*	*PDU Body*
4 octets	Length = (Command Length value - 4) octets			

Figura 2.1 Estructura de un mensaje SMPP [1]

Inmediatamente se describirán cada uno de los parámetros de cabecera:

Command_length

Define la longitud en octetos del mensaje SMPP. Esta longitud considera la cabecera, el cuerpo del mensaje e inclusive el mismo parámetro *command_length.*

Command_id

Este comando identifica el tipo de mensaje SMPP que se está transmitiendo. Por ejemplo pueden representar el comando *submit_sm*, *deliver_sm*, *deliver_sm_resp* entre otros.

Command_status

Indica el éxito o el fracaso de una solicitud SMPP. Es relevante en los mensajes SMPP de respuesta, mientras que en los mensajes de solicitud se coloca un valor NULL.

Sequence_number

Este parámetro contiene el número de secuencia y se utiliza para correlacionar la respuesta de un PDU SMPP con una solicitud. Este número puede oscilar de 0x00000001 y 0x7FFFFFFF.

2.3 Definición de los PDU's SMPP

Los mensajes que forman parte del cuerpo de esta estructura se describen a continuación:

PDU Bind

El propósito de esta operación es registrar una instancia de la SMSC con el ESME y solicitar una sesión SMPP (puede ser *transceiver*, *transmitter* o *receiver*) a través de una conexión de red para la entrega y recepción de mensajes.

PDU OutBind

Esta operación generalmente es originada por la SMSC y transmitido al ESME para que este pueda establecer una sesión como receiver. Generalmente este mensaje se envía cuando la SMSC tiene mensajes pendientes para enviar al ESME.

PDU Generic_Nack

Este mensaje es una confirmación negativa a un PDU SMPP que representa un encabezado inválido del mensaje. Por ejemplo se puede originar este comando por un *command_length* inválido (representan datos corruptos ya sea muy largo o muy corto con respecto al tamaño real) o command_id desconocido.

PDU Unbind

El objetivo de este mensaje es desvincular la sesión que existe entre el ESME y la SMSC. Es decir el ESME informa a la SMSC que ya no desea ni enviar ni recibir mensajes.

PDU enquire_link

Este mensaje puede ser enviado por el ESME o por la SMSC y se lo utiliza para conocer el estado de la ruta de comunicación. La entidad

receptora de este mensaje envía un *enquire_link_resp* para indicar si está o no funcionando la conexión.

PDU submit_sm

Esta operación solo la realiza el ESME y es donde está contenido el mensaje corto de texto. La SMSC envía el correspondiente *submit_sm_resp* para indicar al ESME que recibió correctamente el mensaje.

PDU submit_multi

Esta operación se la utiliza para enviar un mensaje corto de texto a múltiples destinatarios o a una o más listas de distribución.

PDU *deliver_sm*

Este mensaje es emitido por la SMSC para que este pueda enviar mensajes al ESME. Para que la SMSC pueda enviar este mensaje, debe existir un enlace como *transceiver* o *receiver* entre ambas entidades. El ESME debe enviar un *deliver_sm_resp* para indicar que recibió correctamente el mensaje.

PDU data_sm

Es un comando que puede ser usado por el ESME o la SMSC. Es una alternativa de los comandos *submit_sm* y *deliver_sm*. La entidad que recibe este mensaje debe enviar un *data_sm_resp* a la otra parte para indicar que recibió correctamente el mensaje.

PDU Alert_notification

Este mensaje es enviado por la SMSC al ESME, e indica que un abonado habilitado tiene una entrega pendiente.

PDU cancel_sm

Este comando es emitido por el ESME y enviado a la SMSC para cancelar mensajes que aún están pendientes por entregar.

PDU query_sm

Este mensaje es emitido por el ESME para consultar el estado de un mensaje corto enviado anteriormente.

PDU replace_sm

Este mensaje es emitido por el ESME para reemplazar un mensaje corto que se envió anteriormente y que aún no ha sido entregado al usuario móvil por parte de la SMSC.

2.4 Parámetros Mandatorios

A continuación se describen los parámetros SMPP que son obligatorios para poder iniciar un enlace SMPP.

System_id

Este parámetro sirve para identificar un ESME o un SMSC durante la vinculación. Un *system_id* ESME identifica el ESME al SMSC. El *system_id* SMSC determina la SMSC al ESME.

Password

Este parámetro es usado por el SMSC en conjunto con el *system_id* para identificar también el ESME. La contraseña normalmente es expedida por el administrador del sistema SMSC.

System_type

El parámetro *system_type* se usa para categorizar el tipo de ESME que es vinculante para el SMSC.

Addr_ton, source_addr_ton, dest_addr_ton, esme_addr_ton

Estos parámetros definen el tipo de número (TON) para ser utilizados en los parámetros de dirección del ESME. Los valores TON son mostrados en la Figura 2.2 y se definen de la siguiente manera:

TON	Valor
Desconocido	0
Internacional	1
Nacional	10
Red especifica	11
Numero de abonado	100
alfanumérico	101
Abreviado	110
Todos los demás valores reservados.	

Figura 2.2 Valores que se atribuyen al parámetro TON [2]

Addr_npi, source_addr_npi, dest_addr_npi, esme_addr_npi

Estos campos definen el indicador plan numérico conocido como NPI para ser utilizado en los parámetros de dirección de los ESME. En la figura 2.3 se muestran los valores que puede tomar este parámetro.

NPI	Valor
Desconocida	0
ISDN (E163/E164)	1
Data (X.121)	11
Telex (F.69)	100
Land Mobile (E.212)	110
Nacional.	1000
Privada	1001
ERMES	1010
Internet (IP)	1110
WAP Client Id (to be defined by WAP Forum)	10010
Todos los demás valores reservados.	

Figura 2.3 Valores que se atribuyen a los NPI [2]

En la Figura 2.4, se detallan los diferentes formatos en la que puede adoptar el mensaje corto seguido de su descripción [3].

Tipo de SMS.	*Descripción y Ejemplo.*
Plano 7-bits	*Texto plano*
Binario 8-bits	*Data binario codificado hexadecimal*
Unicode(UCS2)	*UCS2 hexadecimal usado para representar caracteres que no están incluidos en el alfabeto GSM*

Figura 2.4 Tipos de SMS

El protocolo SMPP es capaz de soportar texto plano de 7-bits, Binario 8-bits y UNICODE (estándar de codificación de caracteres con el objetivo de facilitar el tratamiento de los datos).

Para la implementación del protocolo en alguna aplicación en particular y para que las operaciones puedan ser interpretadas correctamente existe una asignación de códigos para cada operación, en la siguiente Figura 2.5 se muestran las operaciones que se pueden ejecutar usando SMPP con su respectivo código.

Command ID	Value
generic_nack	0x80000000
bind_receiver	0x00000001
bind_receiver_resp	0x80000001
bind_transmitter	0x00000002
bind_transmitter_resp	0x80000002
query_sm	0x00000003
query_sm_resp	0x80000003
submit_sm	0x00000004
submit_sm_resp	0x80000004
deliver_sm	0x00000005
deliver_sm_resp	0x80000005
unbind	0x00000006
unbind_resp	0x80000006
replace_sm	0x00000007
replace_sm_resp	0x80000007
cancel_sm	0x00000008
cancel_sm_resp	0x80000008
bind_transceiver	0x00000009
bind_transceiver_resp	0x80000009
Reserved	0x0000000A 0x8000000A
outbind	0x0000000B
Reserved	0x0000000C - 0x00000014 0x8000000B - 0x80000014
enquire_link	0x00000015
enquire_link_resp	0x80000015
Reserved	0x00000016 - 0x00000020 0x80000016 - 0x80000020

Figura 2.5 Comandos SMPP con su respectivo valor

2.5 Estados de una sesión SMPP

Para que el ESME se enlace con la SMSC y entre ellas levantar una sesión para transmitir mensajes cortos de texto, el ESME debe enviar un mensaje de conexión para comenzar la transmisión. En la Figura 2.6, se muestra el mensaje en donde el ESME solicita conectarse con la SMSC para que posteriormente pueda enlazarse ya sea como *receiver*, *transmitter* o *transceiver*.

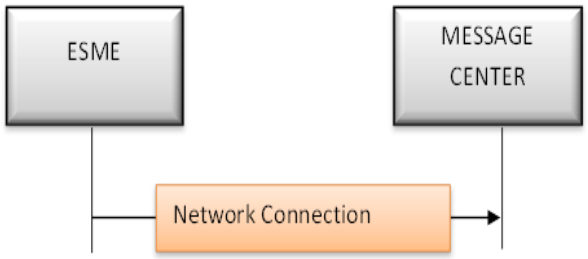


Figura 2.6 Conexión SMPP

Existen tres tipos de sesiones que se pueden establecer entre la SMSC y el ESME, pero para iniciar cualquiera de estas sesiones debe haber conectividad desde el ESME a la SMSC o viceversa. Las tres sesiones son:

- Transmitter
- Receiver
- Transceiver

Transmitter

El ESME puede iniciar una sesión como transmisor enviando un mensaje *bind_transmitter* a la SMSC. Luego de esto, la SMSC para confirmar que ha recibido este mensaje la central, envía un *bind_transmitter_resp*. En la Figura 2.7, se muestran los mensajes SMPP que se transmiten para iniciar el enlace como *transmitter*.

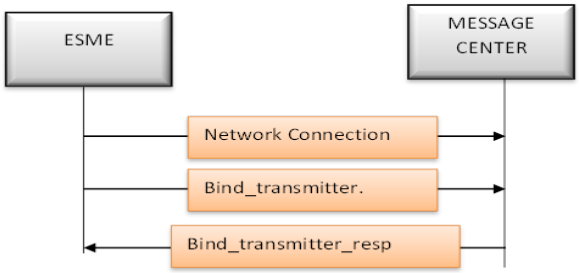


Figura 2.7 Inicio de sesión como *Transmitter*

Receiver

Para iniciar una sesión como receptor, el ESME debe enviar a la central un *bind_receiver* y esta contestar con *bind_receiver_resp* para corroborar que se ha establecido la conexión. En la Figura 2.8, se muestran los mensajes SMPP que sirven para iniciar un enlace como *receiver*.

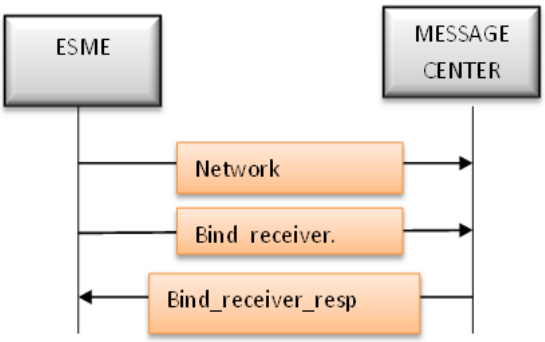


Figura 2.8 Inicio de sesión como *Receiver*

Transceiver

Se combina ambas sesiones mencionadas anteriormente, el mensaje que se envía para el inicio de sesión como transceptor es *bind_transceiver*, así mismo la central debe responder con un *bind_transceiver_resp* para corroborar de que se haya establecido esta sesión. En la Figura 2.9, se muestra los mensajes que permiten al ESME enlazarse como *transceiver*.

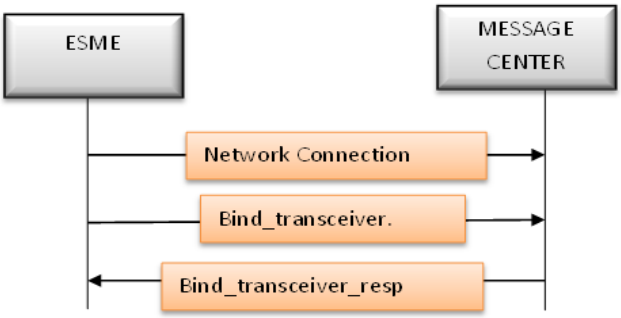


Figura 2.9 Inicio de sesión como *Transceiver*

Dentro de cada sesión entre el ESME y la SMSC (ya sea como *receiver*, *transmitter* o *transceiver*), existen varios mensajes adicionales que se envían entre ambas entidades para controlar la conexión. Estos mensajes hacen que la conexión sea eficiente y confiable, y son *connect*, *unbind*, *unbind_resp*, *closed*. En la figura 2.10 se muestra un diagrama que resumen las sesiones definidas anteriormente.

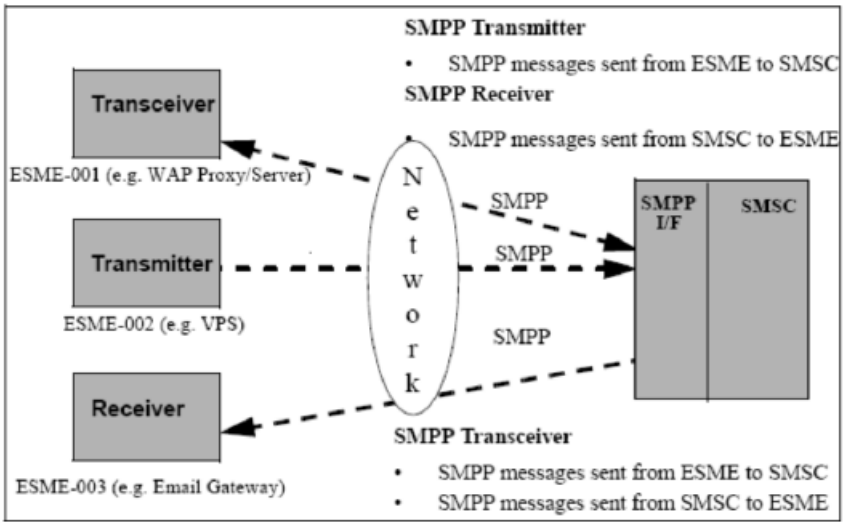


Figura 2.10 Sesiones SMPP [3]

En la Figura 2.11, se muestra el inicio de sesión de un ESME como *transmitter,* mostrando los mensajes necesarios desde el inicio hasta el fin de la conexión. Se puede observar que para iniciar esta sesión debe haber antes una conexión, después de esto se transmiten en pares los paquetes, tanto de envío como de respuesta. Al iniciar sesión

como *transmitter* sólo se envían mensajes de tipo *submit_sm* y se recibe el respectivo mensaje de acuse de recibo en este caso es *submit_sm_resp*.

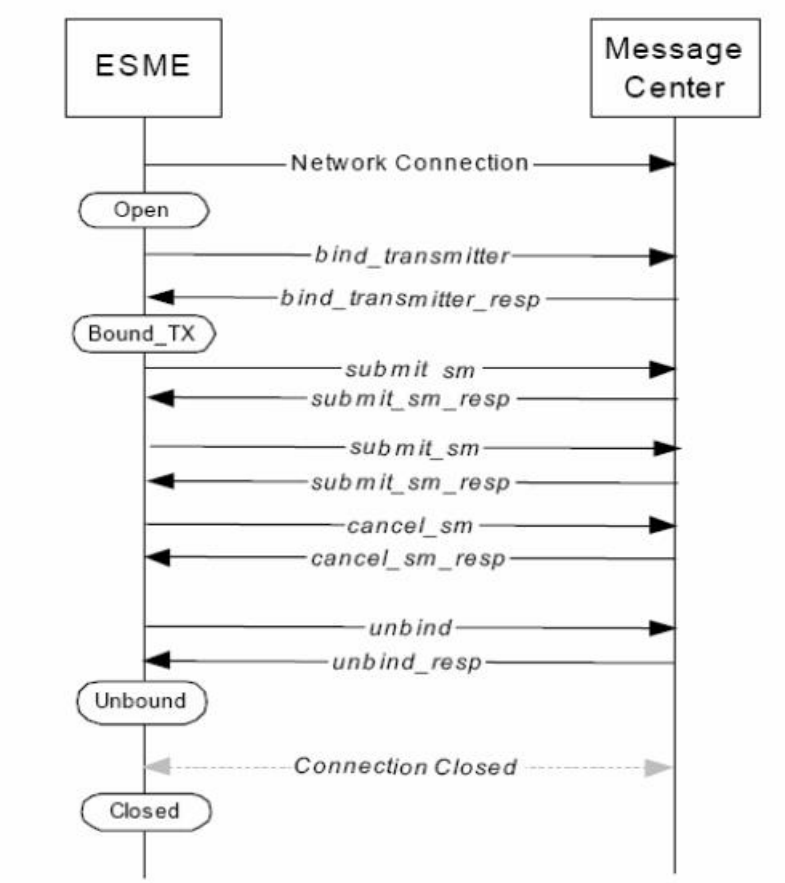


Figura 2.11 Sesión *Transmitter* [2]

En seguida en la Figura 2.12, se muestra el inicio de sesión de un ESME, en este caso como *receiver*. Al igual que el anterior se debe levantar en primer lugar una conexión antes de recibir los mensajes desde la SMSC. Después de esto, solo se puede recibir mensajes del

tipo *deliver_sm* y su respectivo acuse de recibo. En un intercambio de solicitud y respuesta SMPP entre un SMSC y ESME se puede implementar de forma sincrónica o asincrónica así, el SMSC puede enviar varias solicitudes con el parámetro *deliver_sm* a la ESME, sin esperar la sincronía de la PDU de la respuesta. El ESME siempre debería devolver las respuestas SMPP al SMSC en el mismo orden en que se recibe las solicitudes originales. No obstante, esto no es obligatorio dentro de SMPP y el SMSC debe ser capaz de dar respuestas de manejo recibido fuera de la secuencia.

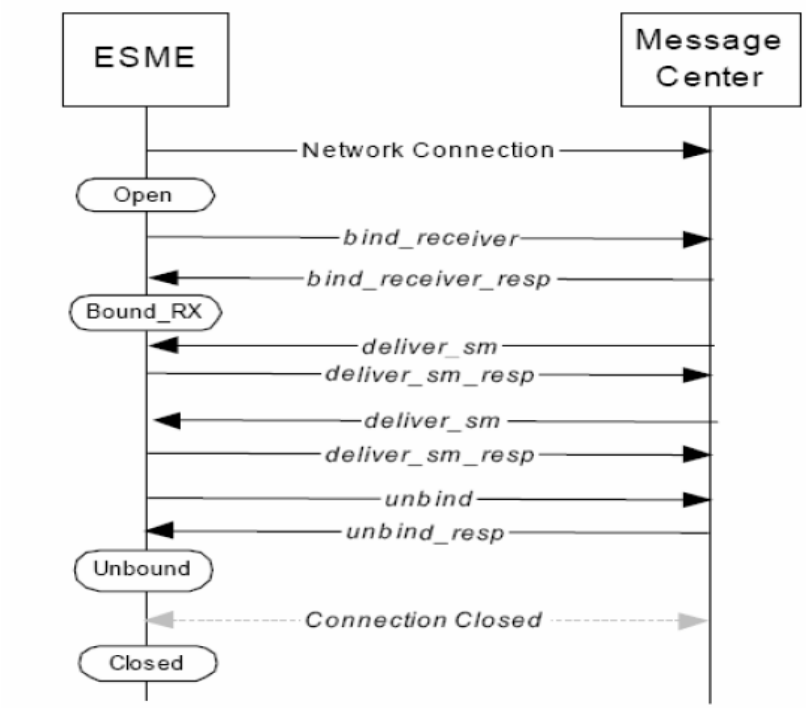


Figura 2.12 Sesión *Receiver* [2]

Por último en la figura 2.13 se muestra el inicio de sesión como *transceiver* en el cual al igual que los dos casos anteriores, debe existir un inicio de sesión. Pueden operar como una sesión de mensajes dúplex el SMSC y ESME, es decir los mensajes se intercambian en ambas direcciones, en este caso el ESME debe estar conectado como un ESME *transceiver* [6].

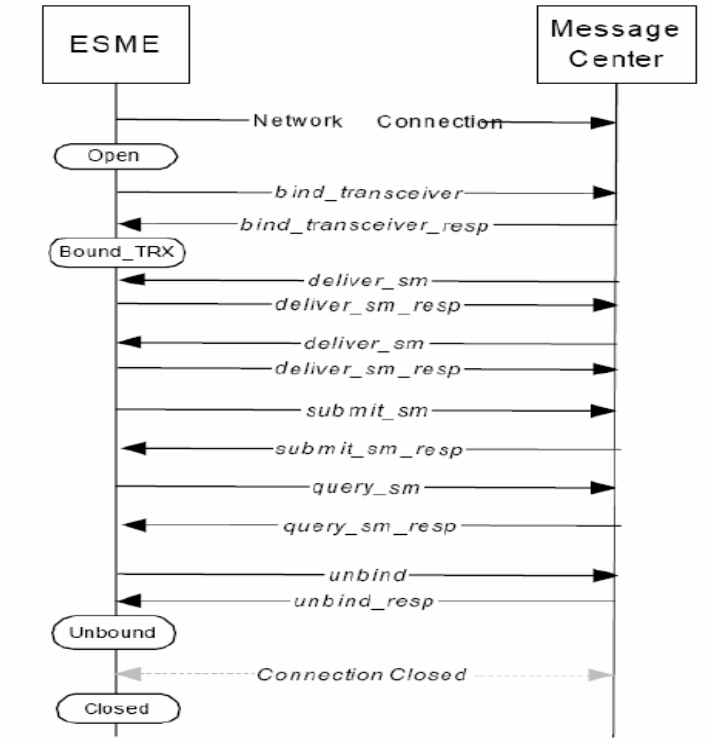


Figura 2.13 Sesión *Transceiver* [2]

2.6 Funcionamiento general del protocolo SMPP entre SMSC y ESME

En la Figura 2.14, se muestra gráficamente el proceso normal que se debe seguir para que el ESME pueda responder y procesar la respuesta a una petición que se origina de la SMSC. Como ya se mencionó, una de las limitaciones que se presentan en este proyecto es que no habrá teléfono móvil verdadero que envíe un mensaje a la SMSC sino que se utilizará otro ESME que pueda simular el dispositivo; sin embargo, se muestra a este ESME como el dispositivo celular.

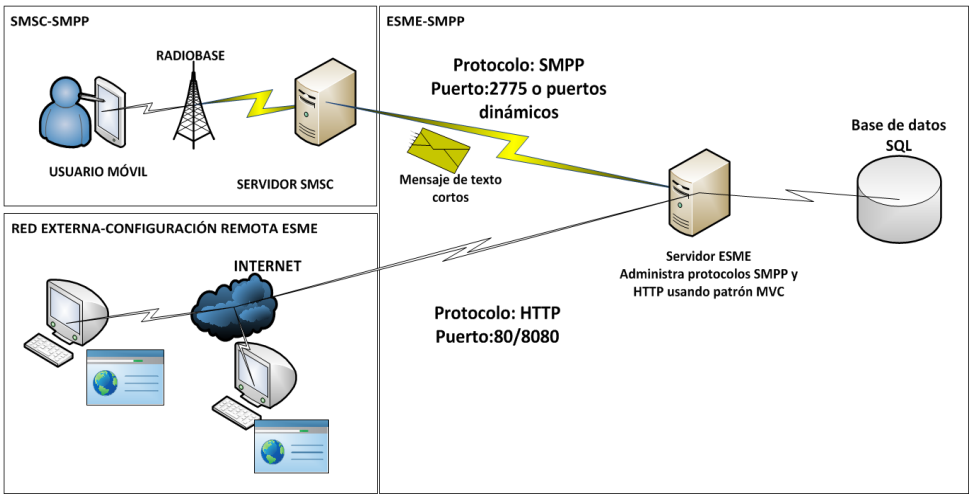


Figura 2.14 Vista general del funcionamiento entre ESME, SMSC

La petición se origina en el teléfono móvil a través de un mensaje corto que debe contener el número destino y el mensaje que contendrá el

texto que procesará el ESME. Una vez que el usuario móvil envíe su mensaje a la SMSC, esta decodifica el mensaje e identifica el ESME destino para poder direccionar el contenido. La SMSC puede contener varios ESME asociados y cada uno de ellos tiene un número público, el cual los usuarios móviles lo usan para solicitar un contenido dinámico. Una vez que la SMSC identifica el ESME destino se establece la conexión SMPP para que entre estas dos entidades se puedan comunicar. Establecida la conexión, la SMSC envía el mensaje al ESME para que este pueda decodificar el paquete SMPP e identificar el mensaje que el usuario móvil solicitó. El ESME realiza el proceso de búsqueda de contenido dinámico por medio de la aplicación en conjunto con la base de datos El mensaje dinámico obtenido se reenvía automáticamente a la SMSC para que lo reenvíe al móvil que originó esta petición. Estos procesos que se mencionaron entre el ESME y la SMSC son automáticos y no requiere la intervención directa del ser humano. El puerto asignado para la transmisión SMPP es el puerto 2775. Sin embargo, la persona que administra la red puede utilizar cualquiera de los puertos dinámicos comprendidos entre 49152 y el 65535. La interfaz gráfica sirve para poder administrar el ESME, en ella se pueden configurar parámetros iniciales como la ip de la SMSC el puerto a usar o el tipo de enlace para transmitir o recibir los mensajes. Como es una aplicación web, se

utiliza el protocolo HTTP. Este protocolo utiliza el puerto 80 o el puerto 8080.

En la Figura 2.15, se muestran los procesos que se deben seguir para que el usuario móvil pueda receptar el mensaje que solicitó a partir de una palabra clave que él mismo envió. Solamente se muestra el proceso en donde interactúa el protocolo SMPP, mas no el proceso para configurar el ESME a través de la aplicación web. Este último proceso se lo detalla en el capítulo tres en la solución de la aplicación. En esta figura las entidades se grafican con cuadros cortados en la parte superior izquierda, los cuadros simples simbolizan los procesos y los recuadros con dos líneas cruzadas en la parte superior e izquierda representan registros de información.

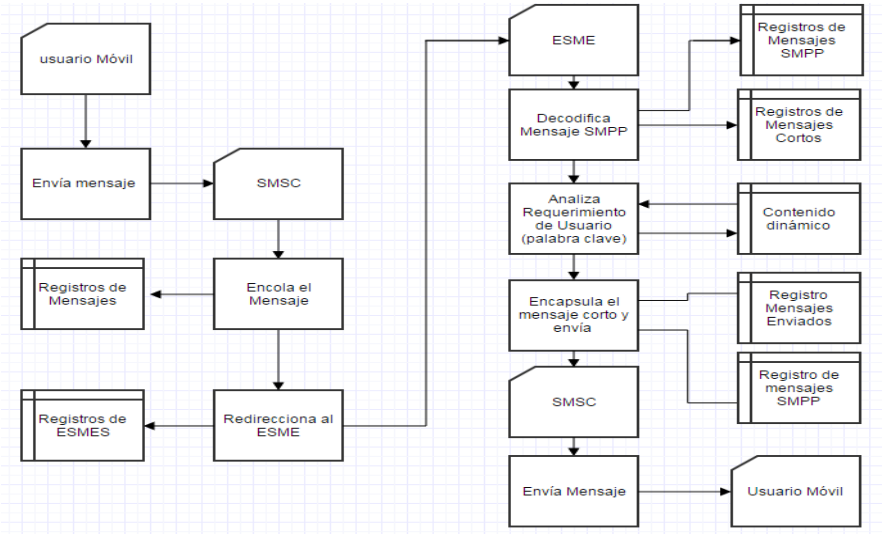


Figura 2.15 Diagrama de flujo de datos que representa el envío de mensajes desde el ESME al usuario móvil a partir de un requerimiento

CAPÍTULO 3

3. DESARROLLO DEL PROYECTO

3.1 Breve descripción de la solución

En este capítulo se explicará de manera detallada todas aquellas herramientas que se usaron para la implementación del ESME, y también aquellas aplicaciones, librerías, y patrones de diseño para el desarrollo de la aplicación web. Para tener una mejor comprensión del desarrollo de esta solución se llevará un orden específico descrito de la siguiente manera:

1. Configurando el entorno de desarrollo: En esta sección se describe de manera breve el sistema operativo que se usó y los componentes que se necesitan para acondicionar el entorno de desarrollo y posteriormente implementar la aplicación.

2. Herramientas para la creación de la aplicación y base de datos: Aquí se muestran cuáles son las herramientas que se necesitan para desarrollar la aplicación web que permitirá la administración del ESME y la base de datos.

3. Arquitectura y marcos de trabajo de la aplicación: En esta parte se detallan cuáles son los patrones que se usaron para el diseño de la aplicación web así como también una breve descripción de los marcos de trabajo que se usaron tanto del cliente como del servidor para que el desarrollo de la misma no se torne compleja.

4. Gestor de base de datos y modelo entidad relación: En esta parte se define lo que es un modelo entidad relación y qué gestor de base de datos se usó para el almacenamiento de la información.

5. Funcionamiento de la aplicación: Se describirá de manera detallada cuales son los pasos que el usuario debe seguir para el control del ESME a través de una aplicación web.

3.2 Configurando el entorno de desarrollo

Para instalar la aplicación se necesita: una computadora y un servidor web en donde se coloca en el directorio raíz el proyecto para poder acceder a ella mediante un explorador web. Como se está trabajando con herramientas código abierto, el sistema operativo que se usó es Ubuntu Server que básicamente es una distribución de Linux. En la Figura 3.1, se muestra el icono que representa a este sistema operativo.

Figura 3.1 Sistema Operativo Linux [4]

Para que la aplicación pueda ser publicada en la red local, se necesita de un servidor web. En este proyecto se usó apache server cuyo logotipo se muestra en la figura 3.2. Apache es un servidor web de código abierto diseñado específicamente para transferir datos de hipertexto, por ejemplo páginas web a través del protocolo HTTP [5].

Figura 3.2 Servidor Apache [6]

Una vez que se haya instalado el servidor web se procede a instalar las librerías de PHP (cuyo acrónimo recursivo significa *Hypertext Preprocessor* que en español significa preprocesador de hipertexto) con el objetivo de que el servidor reconozca las sentencias que pertenecen a este lenguaje de programación. El logo que representa a este lenguaje de programación se lo muestra en la figura 3.3. Como la aplicación que controlará al ESME es una aplicación web, se eligió a PHP como lenguaje de programación ya que este lenguaje trabaja en el lado del servidor [7].

Figura 3.3 Logo del lenguaje de programación PHP [8]

Para que se puedan almacenar los datos, por ejemplo los mensajes enviados o recibidos, y también los comandos y parámetros que se usan para controlar al ESME se instaló en el servidor MYSQL que es

un sistema gestor de base de datos relacional [9] y cuyo logo se lo muestra en la figura 3.4.

Figura 3.4 Logo del gestor de base de datos MYSQL [10]

3.3 Herramientas para el desarrollo de la aplicación

Para el manejo de la aplicación tanto de lado del cliente como en el lado del servidor se utilizó Eclipse como IDE para la manipulación de los códigos tanto del lado del cliente como en el lado del servidor, aunque Eclipse es originalmente diseñado para desarrolladores JAVA, tiene su versión de desarrollo para el lenguaje PHP. El logo de este IDE se lo muestra en la Figura 3.5. Se debe tener en claro que IDE significa *integrated development environment* que traducido al español significa entorno de desarrollo integrado. En este IDE existen varios componentes precargados que permiten que la codificación no sea compleja o se vea desordenada [11].

Figura 3.5 Eclipse IDE para desarrollo en PHP [12]

Para el manejo de la base de datos se usó WORKBENCH para poder realizar el modelo entidad relación que permitirá crear las tablas para almacenar los datos [13]. En la figura 3.6, se muestra el logo de esta herramienta.

Figura 3.6 IDE para administrar base de datos [14]

3.4 Arquitectura y marcos de trabajo de la aplicación

Generalmente en el desarrollo de aplicaciones web se usa la topología cliente – servidor, que básicamente es una estructura en la cual el cliente realiza peticiones a un servidor, el servidor capta esta petición procesa la información y la vuelve a enviar al cliente sin que el cliente conozca de manera detallada cuales fueron los procesos que tuvo que pasar dicho requerimiento [15]. En la Figura 3.7, se muestran equipos que interactúan entre ellos y muestran el flujo de información usando la topología cliente servidor. Cuando el servidor recibe los requerimientos del cliente, este se conecta con la base de datos en donde registrará estos requerimientos y elegirá el contenido que se va a transmitir. Cuando el servidor haya procesado la información, este lo muestra al cliente usando un lenguaje de marcación para la

elaboración de aplicaciones web más conocido como HTML que en sus siglas en inglés significa *HyperText Markup Language*. Para que el usuario pueda interpretar este lenguaje de marcas debe usar un navegador web, que son los que interpretan estas instrucciones.

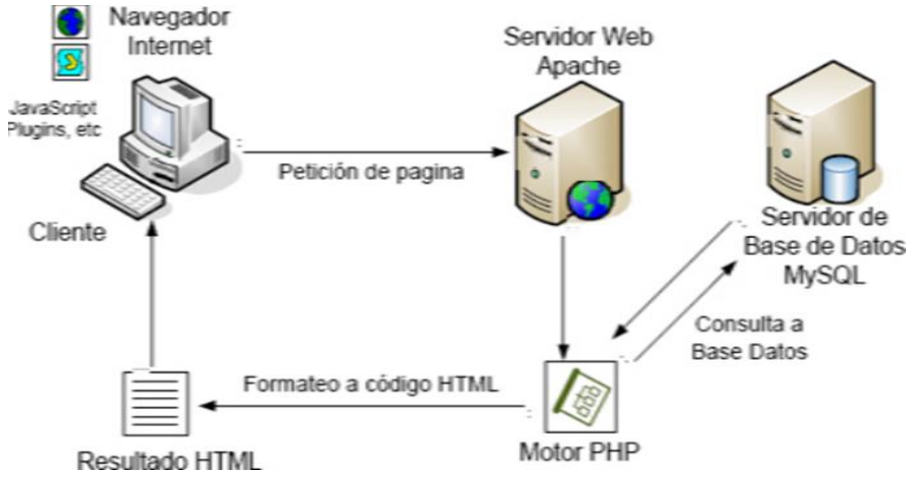


Figura 3.7 Topología Cliente-Servidor [16]

Además de HTML que es un lenguaje de marcas que se muestra en el lado del cliente, también existen lenguajes de programación en el lado del cliente para realizar una aplicación web profesional y que se usaron para el desarrollo de este proyecto como son CSS que en sus siglas en inglés significa *cascading style sheets,* y también JAVASCRIPT que es un lenguaje de programación para el desarrollo de funcionalidades en el lado del navegador (cliente). En la Figura 3.8, se muestran los logotipos de los lenguajes más utilizados para crear aplicaciones web dinámicas y profesionales.

Figura 3.8 Lenguaje de programación en el lado del Cliente [17]

CSS permite insertar un formato personalizado a todas las etiquetas HTML permitiendo desarrollar una aplicación profesional y amigable. Para el diseño de la aplicación se usó el marco de trabajo o *framework* de CODEIGNITER basado en el lenguaje de programación de PHP, el lector quizás se preguntará ¿qué es un *framework*?, pues básicamente es una plataforma de desarrollo que traducido al español significa marco de trabajo en donde existen un conjunto de librerías y complementos ya implementados que sirven para realizar los módulos necesarios para el funcionamiento eficiente de la aplicación. En la Figura 3.9, se muestra el logotipo del *framework* CODEIGNITER utilizado en este proyecto [18].

Figura 3.9 CODEIGNITER, Framework PHP [19]

CODEIGNITER se basa en un patrón de diseño como MVC que en significa modelo, vista y controlador. Este patrón de diseño de software

permite separar la lógica del negocio con la interfaz del usuario. En este patrón los controladores son aquellos que interactúan con los modelos y esta comunicación le es indiferente al usuario, quien siempre accede únicamente a la vista, Figura 3.10.

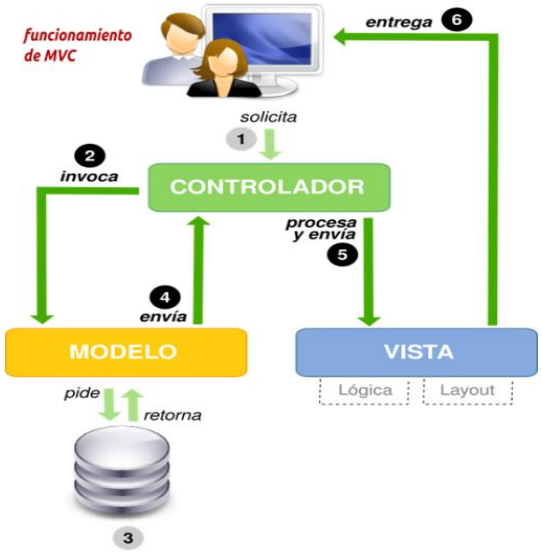


Figura 3.10 Patrón de desarrollo de software MVC [20]

Así mismo como en PHP se usó el *framework* de CODEIGNITER que se implementa en el lado del servidor, también en el lado del cliente se usó un *framework* muy potente conocido como *JQUERY* que permitirá escribir menos código JAVASCRIPT, y por medio de este *framework* se usarán las librerías *AJAX* que son librerías que permitirán realizar peticiones al servidor sin recargar la aplicación web [21]. En la Figura

3.11, se presenta el logotipo del Framework JQUERY basado en JAVASCRIPT.

Figura 3.11 *Framework* JAVASCRIPT usado en el lado del cliente [21]

3.5 Gestor de base de datos y modelo entidad relación

Con lo que respecta a la Base de datos, se usó el lenguaje SQL que significa en inglés *Structure Query Language* que se traduce al español como el lenguaje de consulta estructurado y que es soportado por la mayoría de los motores de base de datos robustos como SQL SERVER, ORACLE o MYSQL. En la Figura 3.12, se muestran algunos de los comandos SQL que se pueden ejecutar en la base de datos. Para el manejo de las consultas, actualización, agregación, y eliminación de los datos se utilizó el motor de base de datos MYSQL, que soporta y ejecuta el lenguaje SQL.

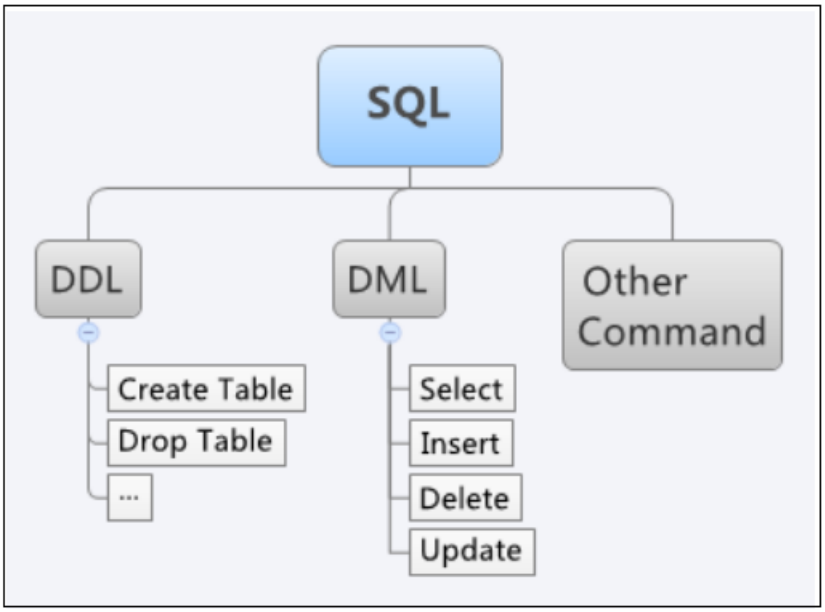


Figura 3.12 Lenguaje SQL para el manejo de la Base de datos [22]

El IDE *WORKBENCH* posee varias funcionalidades que permiten administrar una base de datos, por ejemplo se pueden manipular los datos insertando, actualizando, solicitando o eliminando registros. También se pueden realizar el diagrama Entidad-Relación (E-R) y a partir de este generar automáticamente el código *SQL* que será ejecutado para crear los objetos o tablas en la base. Un diagrama E-R en base de datos representa el modelado de los datos y las interrelaciones que tiene en el sistema.

En este diagrama se presentan las principales entidades que posee un ESME en la que se guardarán los estados que permitirán controlar de manera óptima el funcionamiento de la aplicación.

El ESME no sólo está diseñado para enviar y recibir mensajes de manera manual a través del protocolo SMPP sino que también es capaz de interpretar los mensajes que son recibidos de la SMSC y enviar de manera autónoma la respuesta concerniente a dicha solicitud. En la Figura 3.13, se muestra el diagrama E-R y es donde se guardarán los comandos SMPP, los mensajes de textos enviados, recibidos, así como también los contenidos y tipos de contenidos tanto informativos o de entretenimiento.

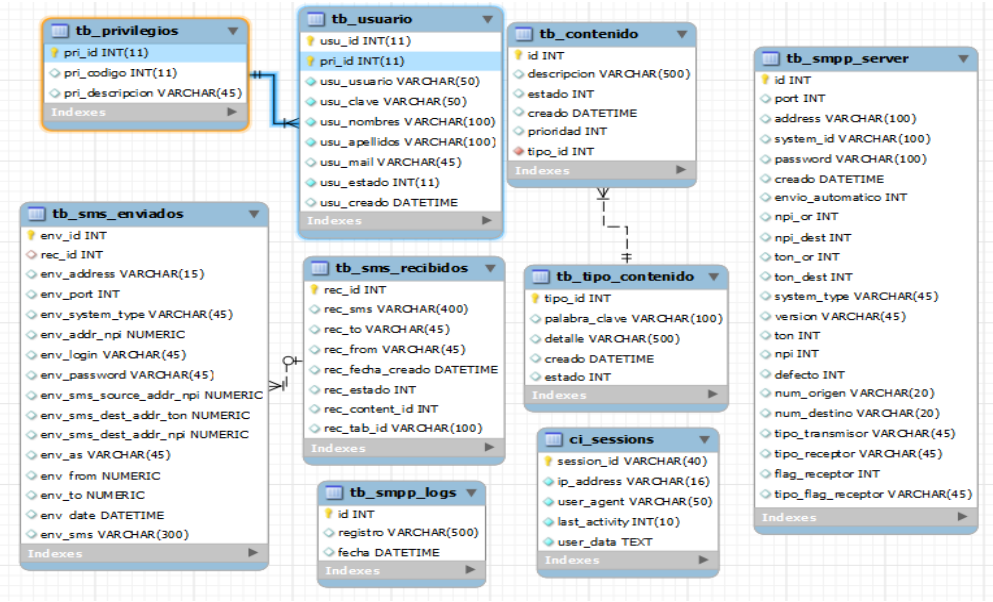


Figura 3.13 Modelo entidad relación

3.6 Funcionamiento de la aplicación

A partir de ahora se explicarán cuáles son los pasos para que el usuario pueda enviar y recibir de manera manual los mensajes hacia

y desde un dispositivo en específico y almacenarlos en una base de datos. Después se describirán los procesos que el usuario debe seguir para que el ESME pueda enviar las respuestas a aquellas solicitudes provenientes del servidor SMSC de una manera manual y automática.

En primer lugar para que el usuario pueda ingresar a la aplicación, este necesita tener instalada en su computadora un navegador web (puede ser Firefox, Chrome, Safari Internet Explorer, entre otros) que es aquel que interpreta el código HTML para que el usuario pueda comprender e interactuar con la aplicación. Después de abrir el navegador se debe digitar en la barra de direcciones la dirección en donde se encuentra la aplicación. En este caso como el proyecto se encuentra alojada en la misma computadora donde se abrió la aplicación web en la barra de dirección del explorador se digita *localhost* o la IP 127.0.0.1, en caso de que se quiera acceder desde otra computadora se debe digitar la dirección IP donde se encuentra alojado el proyecto.

A continuación en la Figura 3.14, se presenta un formulario de registro en la cual se debe digitar el usuario y contraseña para poder acceder a la aplicación. Los usuarios y las contraseñas deben estar previamente registrados en la tabla *tb_usuarios* (se puede observar esta tabla en el modelo entidad relación).

Figura 3.14 Inicio de sesión

Para que el usuario pueda enviar o recibir un mensaje se da clic a la opción que dice Opciones SMPP. A continuación, aparecerá automáticamente un formulario con parámetros por defecto que se encuentran registrados en la base de datos como el que se detalla en la Figura 3.15. Estos parámetros se encuentran registrados en la tabla *tb_parametros,* en la cual también se puede observar en el modelo entidad relación.

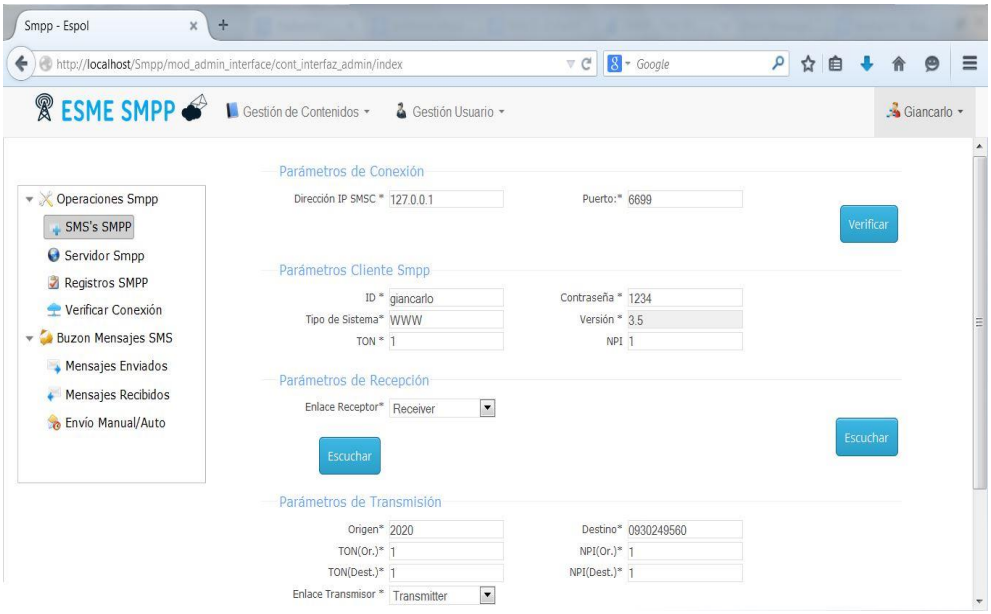


Figura 3.15 Aplicación para la administración del ESME

El siguiente paso es verificar que el servidor SMSC esté funcionando. Para verificar esto se debe observar que los campos de ip y puerto estén correctamente configurados. Una vez que los campos estén llenos, se da clic en la opción conectar.

En caso de que la conexión sea exitosa, como se muestra en la Figura 3.16, se puede proseguir con las opciones de envío y recepción de mensajes.

Figura 3.16 Conexión exitosa del ESME a la SMSC

En caso de que no se pueda establecer correctamente la conexión entre la SMSC y el ESME como se muestra en la Figura 3.17, se debe revisar cual es el problema. Uno de los problemas más comunes podría ser que las direcciones IP en ambas entidades no estén correctamente configuradas y no exista comunicación entre los equipos o de que exista un bloqueo de los puertos por parte del sistema operativo.

Figura 3.17 Conexión fallida del ESME a la SMSC

Para establecer una conexión como *receiver*, en el mismo formulario se debe dirigir a la opción recepción SMPP y configurar el tipo de conexión que se quiere establecer como receptor de mensajes, las opciones pueden ser *transceiver* o *receiver*.

Cuando se da clic a la opción Escuchar se activa un algoritmo que se seguirá ejecutando hasta cuando el servidor o el cliente finalicen la conexión. La Figura 3.18 muestra la parte del formulario en el que se configura al ESME para que pueda recibir mensajes enlazándose como *receiver*.

Figura 3.18 Inicio de sesión *Receiver*

En la Figura 3.19, se observa que el ESME está conectado con la SMSC y enlazado como *transceiver*. Con esta configuración el ESME puede receptar y enviar mensajes a la SMSC por el mismo enlace.

Figura 3.19 Inicio de sesión *Transceiver*

Independientemente del enlace de recepción, ya sea como *transceiver* o *receiver,* el ESME estará escuchando por el puerto registrado por defecto. Los mensajes que le lleguen al ESME SMPP automáticamente se guardarán en la base de datos en la tabla *tb_mensajes_recibidos*. Para poder enviar un mensaje SMPP, en el mismo formulario se debe dirigir a la opción que dice enviar SMPP. Antes de enviar un mensaje, hay que verificar que todos los campos

estén llenos para poder proceder con el envío. Los campos que se solicitan son: el número a quien va dirigido el mensaje, el número origen que en este caso es el número público del ESME y los parámetros TON y NPI (que hacen referencia al tipo de número y el indicador de plan de número) tanto del número origen y destino. Después de esto se podrá digitar el mensaje corto. Es importante recordar que la cantidad de caracteres es de 160. En caso de que la cantidad de caracteres sea mayor, automáticamente en la parte del servidor se activará un proceso en la cual el mensaje descrito se dividirá en varios mensajes y se enviarán uno tras uno al servidor SMSC. En la Figura 3.20, se muestran los campos que se solicitan para enviar un mensaje de texto corto. Una vez que todos los campos estén llenos y se haya comprobado que la conexión fue exitosa se procede a enviar el mensaje dando clic en la opción Enviar.

Figura 3.20 Envío de mensajes SMPP

La Figura 3.21, muestra un error al enviar un mensaje SMPP. En este caso uno de los problemas comunes por la falta de comunicación entre ambas entidades podrían ser la configuración errónea de las IP's, el puerto de comunicación SMPP o las credenciales erróneas que envía el ESME para que sea reconocido por la SMSC.

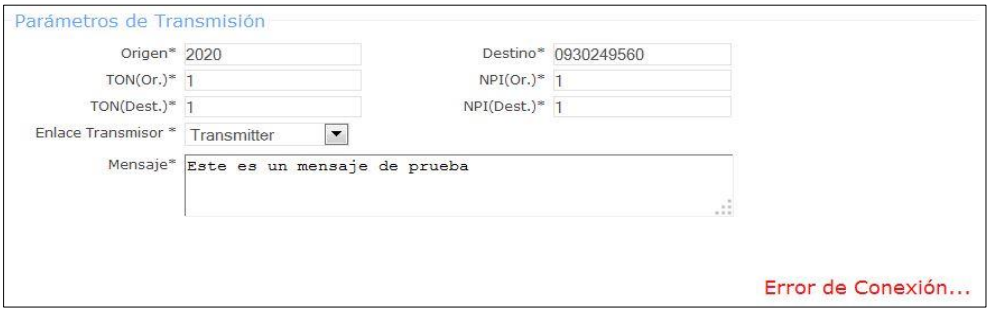


Figura 3.21 Error al enviar mensaje SMPP

Generalmente este tipo de aplicaciones están diseñadas para que trabajen de manera autónoma es decir sin la participación de un agente externo como es el ser humano, pero para efectos didácticos se agregó la opción de habilitar o deshabilitar el envío automático. Para configurar la parte de envío de mensajes automáticos, hay que dirigirse a la opción Parámetros por defecto y editar en el servidor que está seleccionado por defecto la opción Envío automático. Cada vez que se envía o transmite mensajes cortos, del lado del servidor se generan automáticamente varios procesos concernientes al protocolo SMPP. Cada uno de estos procesos se registra en una tabla con el

objetivo de llevar un seguimiento acerca de los paquetes que se transmiten desde el ESME al SMSC o viceversa. Con este archivo se podrá llevar constancia de cada uno de los paquetes que se han transmitido y la hora exacta en el que se produjeron para que en caso de que suceda alguna inconsistencia o error en la conexión se podrá conocer rápidamente hasta qué punto la aplicación se estaba ejecutando correctamente. En la Figura 3.22, se muestra un log en donde se registran los mensajes y comandos SMPP que se usan para la comunicación entre el cliente y el servidor.

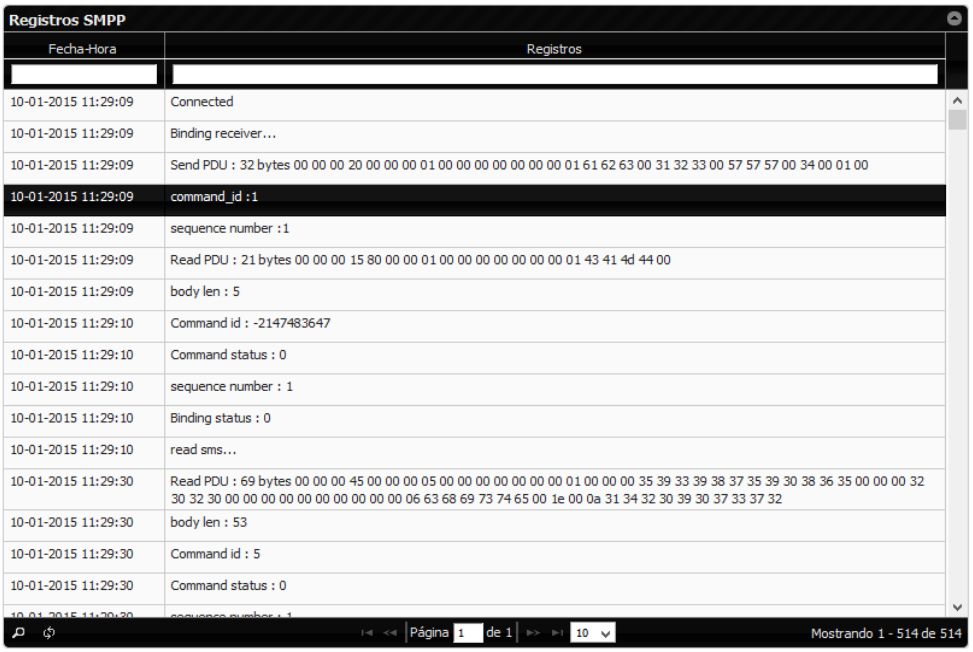


Figura 3.22 Registros de procesos SMPP

Los procesos que se ejecutan cuando se activa la opción del envío automático y también la sesión levantada para escuchar los mensajes desde la SMSC se realizan del lado del servidor, por lo tanto en caso de que el usuario de la aplicación desea cerrar sesión o cerrar el navegador no se van a ver afectados ninguno de estos procesos ya que se están ejecutando del lado del servidor y lo que se está cerrando es una vista que se encuentra del lado del cliente, en otras palabras el ESME seguirá trabajando de manera autónoma hasta que el usuario de la aplicación desee cerrar la sesión. También en la aplicación se tiene la opción de mostrar los mensajes enviados y recibidos ya que la aplicación está diseñado para que se guarden automáticamente en la base de datos en las tablas *tb_mensajes_enviado* y *tb_mensajes_recibidos* respectivamente que se muestran en el modelo E-R. A estos registros se los conocen como CDR. A continuación, en la Figura 3.23 se muestran los mensajes recibidos.

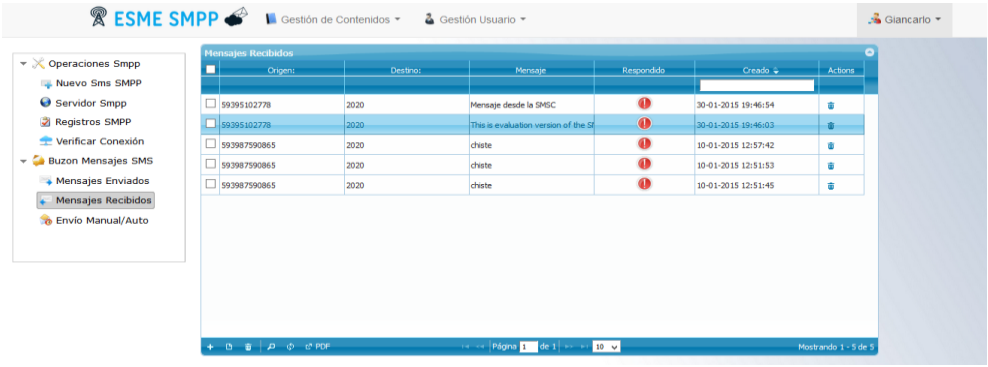


Figura 3.23 Buzón de entrada SMPP

En la Figura 3.24, se muestran los mensajes enviados por el ESME. En esta figura se detalla el mensaje que se recibió, la hora en la que llegó y los parámetros requeridos para que el envío del mensaje se realice con éxito.

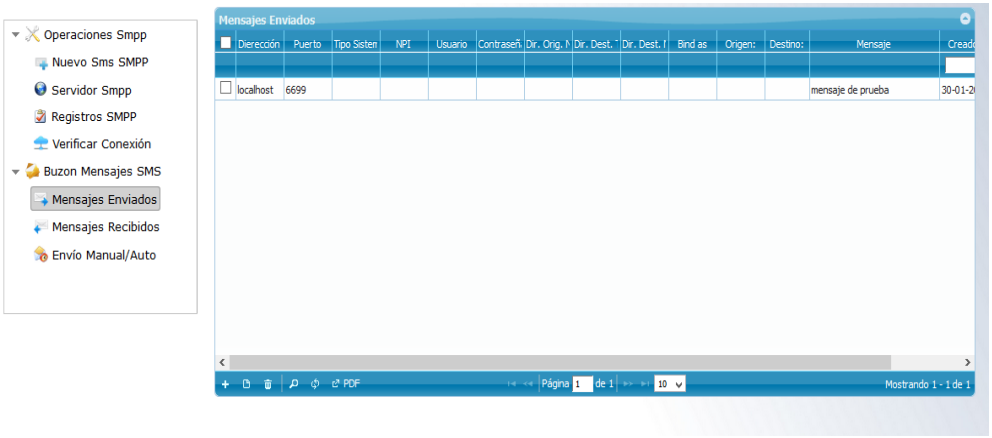


Figura 3.24 Buzón de salida SMPP

El usuario también podrá también administrar el contenido dinámico que se van a enviar a los respectivos clientes en base a un requerimiento de estos y también podrá crear tipos de contenidos con el fin de tener un ESME dinámico y no estático o monótono. Para que el usuario pueda editar los contenidos y tipos de contenidos deber dirigirse en la parte superior a la opción Contenido y elegir la opción deseada. Un campo importante en la opción en el cual se muestra los tipos de contenidos es la palabra clave. Esta palabra clave debe ser exactamente la misma palabra que los clientes enviaran desde su

teléfono móvil a la SMSC y la SMSC al ESME. Con esta palabra el ESME reconocerá que tipo de contenido se está solicitando y enviará como respuesta cualquiera de los mensajes relacionados con este tipo de contenidos aleatoriamente. En la Figura 3.25, se muestra la tabla donde se encuentran los contenidos que utilizará el ESME para enviarlos a partir de un requerimiento.

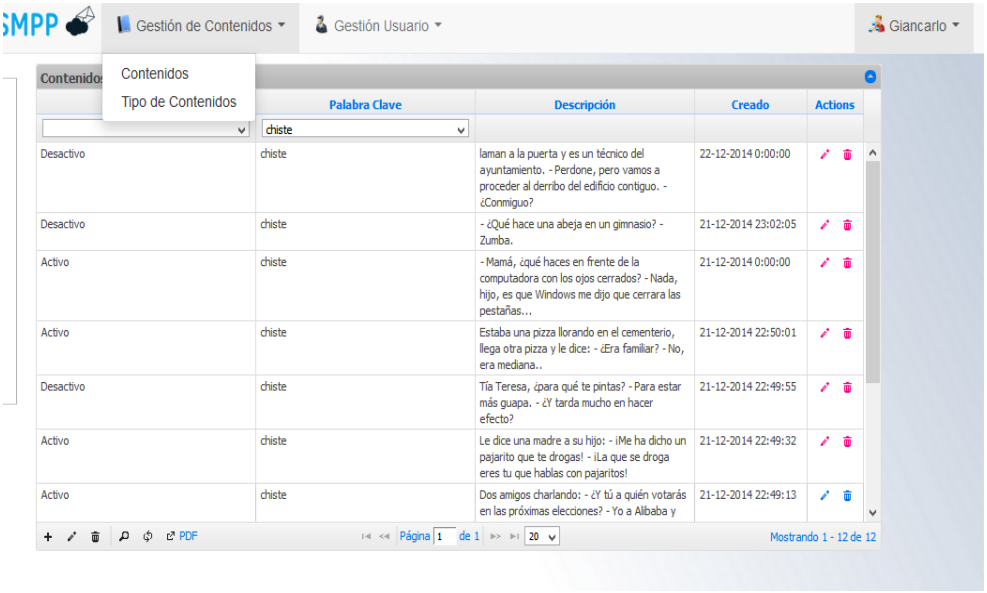


Figura 3.25 Contenidos dinámicos del ESME

Por último para llevar un control de los usuarios que podrán acceder a la aplicación se ha creado la opción de Administrar Usuarios, la tabla de configuración de usuarios se muestra en la Figura 3.26. En esta opción se podrá agregar los usuarios proporcionándoles un usuario y contraseña para que puedan acceder a la aplicación.

En la opción de privilegios de usuario se muestra la opción *root*, es decir todos aquellos usuarios que se registren tendrán acceso a todas las opciones de la aplicación.

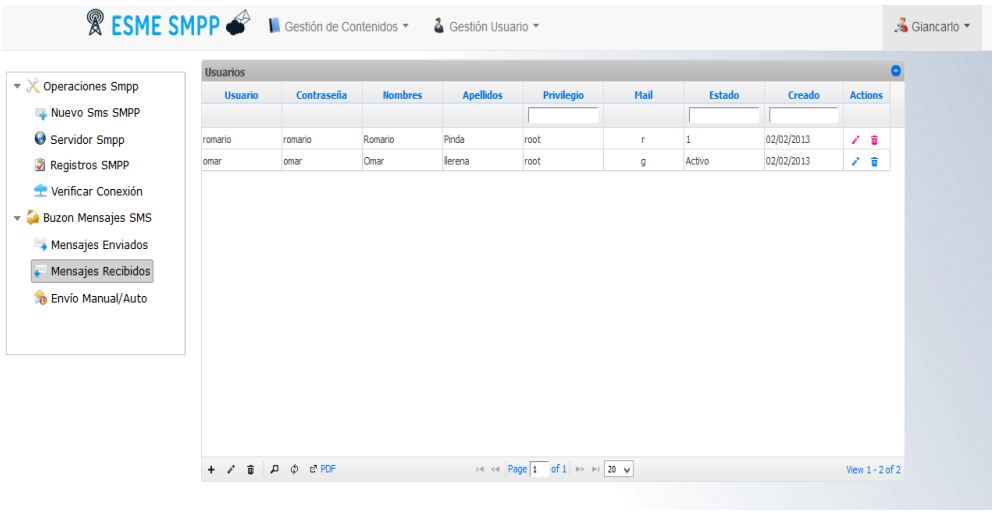


Figura 3.26 Administración de usuarios SMPP

3.7 Algoritmos relevante de la aplicación

El código que se muestra en la Figura 3.27, pertenece a la librería SMPP que se encuentra implementada en PHP [23]. Gracias a esta librería es posible enviar y recibir mensajes a través del protocolo SMPP. En la siguiente figura se muestra el código PHP que permite a la aplicación enlazarse como *Receiver* o como *Transmitter*.

```php
function bindTransmitter($login, $pass){
    if($this->state!="open")return false;
    if($this->debug){
        echo "Binding transmitter...\n\n";
    }
    $status=$this->_bind($login, $pass, 0x00000002);
    if($this->debug){
        echo "Binding status  : $status\n\n";
    }
    if($status===0)$this->state="bind_tx";
    return ($status===0);
}
```

```php
function bindReceiver($login, $pass){
    if($this->state!="open")return false;
    if($this->debug){
        echo "Binding receiver...\n\n";
    }
    $status=$this->_bind($login, $pass, 0x00000001);
    if($this->debug){
        echo "Binding status  : $status\n\n";
    }
    if($status===0)$this->state="bind_rx";
    return ($status===0);
}
```

Figura 3.27 Código PHP para iniciar sesión como *Receiver* y *Transmitter*

Si se observa cuidadosamente el código en la sentencia condicional es fácil darse cuenta que la aplicación valida que el estado esté activo, es decir solo permite enlazarse en caso de que la conexión entre ambas entidades tanto el ESME como la SMSC se dio con éxito.

Como la librería implementa la versión 3.4 del protocolo SMPP, el algoritmo que permite a la aplicación enlazarse como *Transceiver* es el que se muestra en la Figura 3.28. El código es muy similar al código para enlazarse como *transmitter* o *receiver*. La ventaja es que para enviar y recibir mensajes no es necesario abrir dos conexiones de manera independiente sino que en una sola conexión se puede enviar y recibir mensajes.

```
function bindTransceiver($login, $pass)
{

    if ($this->state != "open")
        return false;
    if ($this->debug) {
        echo "Binding transceiver...\n\n";
    }
    $status = $this->_bind($login, $pass, 0x00000009);
    if ($this->debug) {
        echo "Binding status  : $status\n\n";
    }
    if ($status === 0)
        $this->state = "bind_tcx";
    return ($status === 0);
}
```

Figura 3.28 Código PHP para iniciar sesión como *Transceiver*

En la Figura 3.27 y 3.28 se puede observar que los parámetros que se pasan a la función *$this->_bind* son *$login, $pass* y un número hexadecimal que corresponden a los parámetros *system_ID*, *password* y el tipo de comando SMPP respectivamente para que el ESME se conecte con la SMSC. El número hexadecimal que se pasa como parámetro depende del tipo de enlace que se realice. Por ejemplo 0X00000009 corresponde a *bindTransceiver*, 0X0000001 a *bindReceiver* y 0X00000002 que finalmente corresponde a *bindTransmitter*.

En la figura 3.29 se muestra también el código en la que permite al ESME enviar un mensaje a la SMSC ya sea como *transmitter* o *transceiver*. Se observa en la parte inferior se muestra el valor del

comando SMPP 0X00000004 y el *pdu*. Este código representa al comando *submit_sm* y al mensaje respectivamente.

```
function sendSMS($from, $to, $message)
{
    if (strlen($from) > 20 || strlen($to) > 20)
        return false;
    if (($this->state != "bind_tx") && ($this->state != "bind_tcx"))
        return false;
    if (preg_match('/\D/', $from)) //alphanumeric sender
    {
        $this->sms_source_addr_ton = 5;
        $this->sms_source_addr_npi = 0;
    }
    $multi = $this->split_message($message);
    $multiple = (count($multi) > 1);
    if ($multiple) {
        if (!($this->sms_esm_class & 0x00000040))
            $this->sms_esm_class += 0x00000040;
    }
    reset($multi);

    while (list(, $part) = each($multi)) {
            $message = $part;
            $pdu = pack('a1cca' . (strlen($from) + 1) . 'cca' . (strlen($to) + 1) .
                'ccca1a1ccccca' . (strlen($message)), $this->sms_service_type, $this->
                sms_source_addr_ton, $this->sms_source_addr_npi, $from, //source_addr
                $this->sms_dest_addr_ton, $this->sms_dest_addr_npi, $to, //destination_
                $this->sms_esm_class, $this->sms_protocol_id, $this->sms_priority_flag,
                sms_schedule_delivery_time, $this->sms_validity_period, $this->
                sms_registered_delivery_flag, $this->sms_replace_if_present_flag, $this
                sms_data_coding, $this->sms_sm_default_msg_id, strlen($message), //sm_l
                $message //short_message
                );
            $status = $this->sendCommand(0x00000004, $pdu);
        }

        $this->message_sequence = rand(1, 255);
        $this->sms_esm_class = 0;
        return ($status === 0);
    }
```

Figura 3.29 Código PHP para enviar un mensaje corto a la SMSC

En esta aplicación es frecuente el envío y recepción de mensajes largos por el contenido que se envía, es decir en caso de que el mensaje que envía el ESME sea muy largo y sobrepase los 155 caracteres, la aplicación presenta un algoritmo en la cual puede dividir al mensaje en paquetes más pequeños con el objetivo de enviar toda la información al destinatario. En este caso, se envía los mensajes y el celular es quien agrupa los mensajes para mostrarlo como uno solo

mensaje. En la figura 3.30, se muestra el algoritmo para segmentar el mensaje de texto en caso de que sobrepase los 160 caracteres.

```php
function split_message($text)
{
    $max_len = 153;
    $res = array();
    if (strlen($text) <= 160) {
        $res[] = $text;
        return $res;
    }
    $pos = 0;
    $msg_sequence = $this->message_sequence++;
    $num_messages = ceil(strlen($text) / $max_len);
    $part_no = 1;
    while ($pos < strlen($text)) {
        $ttext = substr($text, $pos, $max_len);
        $pos += strlen($ttext);
        $udh = pack("cccccc", 5, 0, 3, $msg_sequence, $num_messages, $part_no);
        $part_no++;
        $res[] = $udh . $ttext;
    }
    return $res;
}
```

Figura 3.30 Código en PHP para dividir mensajes largos

En la figura 3.31, se muestra el directorio en la cual está instalado todo el proyecto. Todo este directorio que se muestra es la estructura establecida por el *framework* de CODEIGNITER en la que se observa la división de los módulos, en este caso CODEIGNITER trabaja con el modelo arquitectónico Modelo-Vista-Controlador, con el objetivo de no mezclar el código que se implementa en la parte del cliente con la del servidor.

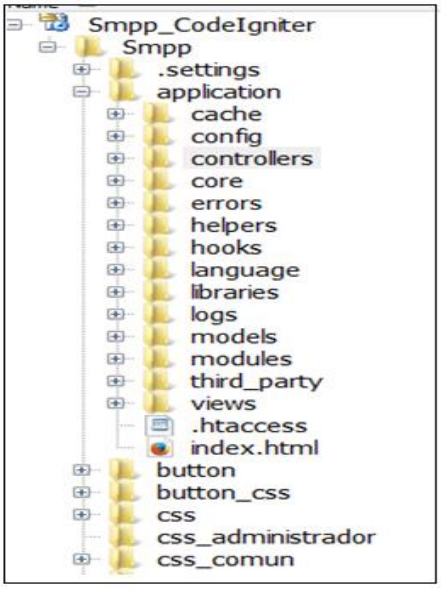


Figura 3.31 Directorio donde se presenta el patrón Modelo-Vista-Controlador

Para el registro de los datos, en la figura 3.32 se describe el algoritmo implementado en el lenguaje de base de datos SQL para el registro de los mensajes y procesos SMPP. En este caso las letras que están en rojo son las sentencias SQL que en este ejemplo generan un número aleatorio para escoger arbitrariamente un contenido dinámico para enviarlo a la SMSC

```php
function get_contenido($sms)
{

    //$table="tb_chistes";
    $sql = 'SELECT descripcion FROM ' . $table .
        ' WHERE id >= (SELECT FLOOR( MAX(id) * RAND()) FROM ' . $table .
        ' ) ORDER BY id LIMIT 1;';
    $r = $this->db->query($sql);
    if ($r->num_rows() > 0) {
        foreach ($r->result_array() as $row) {
            return $row['descripcion'];
        }
    }
    return false;
}
```

Figura 3.32 Código PHP y SQL para la búsqueda de un mensaje dinámico

Con respecto a la aplicación del lado del cliente, en la Figura 3.33 se muestra el código HTML, en este caso este código mostrado representa la interfaz en donde el usuario administra el ESME para el envío y recepción de mensajes de manera manual.

```html
<fieldset class="row1">
    <legend>Par&aacute;metros de Conexi&oacute;n
    </legend>
    <p>
        <label>Direcci&oacute;n IP SMSC *
        </label>
        <input  id="address" type="text" required="required" name="direccion"/>
        <label>Puerto:*
        </label>
        <input  id="port" type="text" required="required" name="puerto"/>

    </p>
    <p>
      <label>
        <div style="text-align:right">
            <a id="conectar"  class="a_demo_one" >Verificar</a>
            <div style="text-align: right;" id="divConexion"></div>
        </div>
      </label>
    </p>
</fieldset>
```

Figura 3.33 Código en HTML que muestra la interfaz para iniciar conexión con la SMSC

A continuación, en la figura 3.34 se muestra código en JAVASCRIPT en el cual se usan funciones implementadas por el *framework* JQUERY para que el usuario pueda interactuar con el servidor sin necesidad de refrescar la aplicación. En importante recalcar que este código puede ser visible para el cliente en caso de que habilite en el navegador la opción mostrar código fuente, pero este no podrá ver código escrito en PHP ya que es utilizado en el lado del servidor.

```
.done(function() {
    if(flag)
    {
            $.ajax({
             url:'mod_smpp_admin/cont_receiver/verificar_parametros_base',
             type:'post',
             //dataType:'jason',system_type
             //data:dataJSON,
             beforeSend:function(){

                 $('#divReceptor').html(tipo_receptor+" <img width=40 height=10
                 $("#divReceptor").fadeIn();
                 $("#btn_receptor").fadeOut(0.000001);

             },
             success:function(respuesta){
                         $("#divReceptor").fadeOut(3000);
                         $("#divReceptor").html(respuesta);
                         $("#btn_receptor").fadeIn(3000);

                    },
```

Figura 3.34 Código JAVASCRIPT para el envío de datos al servidor

Para que la aplicación se vea amistosa para el usuario y atractiva se usó también código CSS que es aquel que permite mejorar la apariencia de la aplicación web y aquellos elementos en la cual pueden ser insertados en ella como son los botones, las cajas de texto

entre otros. Este tipo de código también puede ser interpretado por el navegador al igual que el código en JAVASCRIPT.

En la Figura 3.35, se presenta código en CSS que modifica algunas propiedades de los elementos de la aplicación web, por ejemplo en el *body* que representa el cuerpo de la aplicación se está colocando un color azul oscuro (#123) mientras que a los hipervínculos con etiqueta <a> se les atribuyen el color blanco (#fff).

```css
body {
  background: #123;
  color: #333;
  font-size: 11px;
  height: auto;
  padding-bottom: 20px;
}

a {
  color: #fff;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

h1 {
  font-family: Georgia, serif;
  font-weight: normal;
  padding-top: 20px;
  text-align: center;
}
```

Figura 3.35 Código CSS para el diseño de los componentes HTML

CAPÍTULO 4

4. ANÁLISIS DE RESULTADOS

4.1 Resultados obtenidos

En la figura 4.1 se muestra una captura por *wireshark* en donde se exponen los datos que se transmiten desde el servidor al cliente y viceversa. Cabe mencionar que esta gráfica se la capturó cuando se integró el ESME desarrollado en el presente proyecto y la SMSC desarrollada en la tesis “Análisis e implementación de un servidor de Protocolo de Mensajería Escrita Punto a Punto (SMPP) versión 3.4 en Linux que interactúe con un cliente SMPP en el envío y recepción de Mensajes Cortos (SMS's) y que genere archivos de Registros de Datos

de Llamadas, CDRs". En ella se puede observar que se muestran algunos de los parámetros que se configuraron en la aplicación web y que *wireshark* los interpreta fácilmente ya que esta aplicación de red toma en cuenta la implementación del protocolo SMPP.

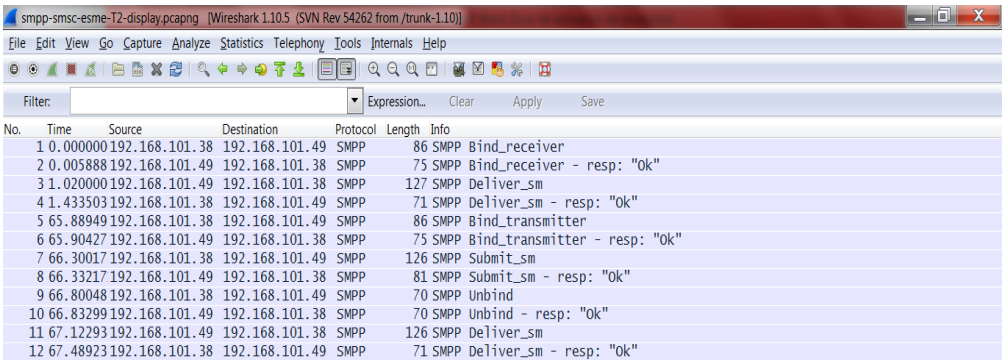


Figura 4.1 Captura en *wireshark* de un paquete SMPP

En la gráfica se presentan doce mensajes SMPP transmitidos entre el cliente y el servidor, cada uno de estos mensajes pertenecen al protocolo SMPP y se puede distinguir cuál de los dos ya sea el ESME o el SMSC han originado algún comando SMPP viendo la dirección de IP origen y destino, en este caso la dirección 192.168.101.49 es el cliente mientras que la dirección 192.168.101.38 es el servidor. El tipo de comando que ejecuta ya sea el ESME o la SMSC se lo puede conocer en la columna INFO, mientras que el tamaño del mensaje se observa en la columna *lenght*. El primer mensaje que se observa proviene del cliente es decir el ESME cuyo comando ejecutado es el *Bind_receiver,* este mensaje indica que el cliente desea conectarse

con el servidor como un receptor de mensajes cortos. Sin embargo, no es suficiente para que el ESME pueda conectarse ya que debe esperar de la SMSC un mensaje de confirmación. El siguiente mensaje proviene de la IP del servidor, en este caso la SMSC en el cual responde con un *Bind_receiver_resp OK* indicándolo al ESME que se ha establecido correctamente la comunicación y que está en la capacidad de enviar mensajes tipos *delivery* que se usan para la recepción de mensajes.

Luego de la conexión como *receiver*, se muestra que el servidor SMSC le envía a la IP cliente un *Deliver_sm*; en estos mensajes SMPP es donde se encapsula el mensaje corto que se originó en algún usuario móvil. Para que la SMSC reconozca que el cliente recibió el mensaje, el ESME debe enviar un mensaje de confirmación. En este caso, el mensaje número cuatro (que lo origina el ESME) es un *Deliver_sm_resp OK* y es aquel mensaje que esperaba la SMSC para corroborar que se ha enviado correctamente el mensaje.

El proceso que se realiza para poder enlazarse como *transmitter* es similar al proceso que se realiza para poder enlazarse como *receiver.* *S*iguiendo la secuencia de los mensajes, en el mensaje número cinco el cliente solicita al servidor conectarse como un transmisor, para ello este envía un *Bind_transmitter* al servidor y este le responde con un *Bind_transmitter_resp OK* indicando al cliente que la conexión como

transmisor ha sido correcta. A partir de este momento el cliente puede enviarle al servidor mensajes de tipo *submit*. En esta conexión el ESME no puede receptar mensajes tipo *delivery* de la SMSC ni enviarlos.

En el mensaje número seis, el cliente le envía al servidor un mensaje *Submit_sm*, en este mensaje es donde el cliente le envía al servidor el mensaje corto que extrajo de la base de datos como respuesta a un requerimiento que recibió de la SMSC en un mensaje *delivery_sm*.

En el mensaje número ocho, la SMSC indica al ESME que su mensaje enviado fue recibido correctamente a través de un *Submit_sm_resp* OK al ESME. Una vez que el proceso de envío de mensajes corto haya finalizado, el ESME envía un *Unbind* a la SMSC para indicar que desea desconectarse como *transmitter*. Sin embargo, el enlace como *receiver* sigue estando activo. Para que el servidor proceda con la desconexión la SMSC le envía al servidor un *Unbind_resp* OK indicando que puede proceder a desconectarse. Cabe recalcar que la conexión como receptor todavía está activa puesto que no se ha solicitado una desconexión por parte del ESME a la SMSC como receptor, es decir todavía el ESME puede recibir mensajes de la SMSC.

En la figura 4.2 se muestra la captura de un paquete SMPP en la que el cliente solicita a la SMSC enlazarse como receptor, específicamente

es el mensaje número uno de la captura de la figura 4.1. En este mensaje se presenta la información del protocolo SMPP y como se mencionó anteriormente que SMPP trabaja bajo el modelo TCP/IP también posee encabezado de la capa dos y tres en la que se indica la Mac de origen, la Mac de destino por la capa dos y la dirección IP origen y destino en la capa tres; sin embargo, este proyecto se enfoca específicamente de la información SMPP que se envía de la capa de aplicación para la transmisión de los mensajes cortos.

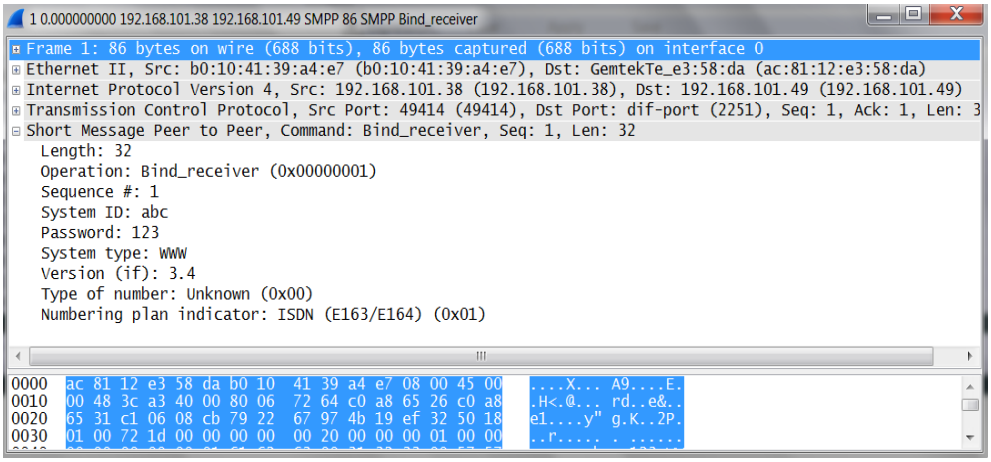


Figura 4.2 Paquete SMPP, enlace *bind_receiver*

En la franja que dice *Short message peer to peer, Command: Bind_receiver* se muestra el campo *Lenght* en la cual indica la información en bytes que representa el tamaño de la información SMPP que se está enviando, en este caso es de 32 bytes. Después de esto se presenta el tipo de mensaje SMPP que se ha enviado, como ya se mencionó en el capítulo dos, existen un conjunto de mensajes

SMPP con una tarea específica que sirven para la comunicación eficiente entre el cliente y el servidor.

El campo *Operation* es aquel comando SMPP que indica el tipo de operación que se está realizando, en este caso es *Bind_receiver* con su código de operación respectivo, en este caso es 0X0000001 que es aquel que está asociado con este comando. Esta lista de códigos con sus respectivos valores se la puede revisar en la Figura 2.5.

Como número de secuencia en este caso está asignado el número uno. Este es asignado por la aplicación para dar seguimiento a los mensajes SMPP tanto de solicitud como de respuesta, para entenderlo mejor un *bind_receiver* tendrá un número de secuencia uno y así mismo un *bind_receiver_resp* tendrá asociado el mismo número de secuencia indicando que es la respuesta a la solicitud. Después un *deliver_sm* tendrá una número de secuencia dos y así mismo un *deliver_sm_resp* tendrá un el mismo número de secuencia dos y así con cada mensaje SMPP en una conexión dada. Los números de secuencia van en pares y se reinician con cada enlace establecido, es decir si se realiza una conexión *bind_transmitter* aun estando abierta una conexión *receiver,* la aplicación asignará un número de secuencia uno así sea que esté activa otra conexión. Los siguientes elementos son el *System_id* y el *Password_id,* estos dos elementos son

configurados por el usuario y representan el usuario y la contraseña para que el ESME pueda ser identificado por la SMSC y se pueda establecer correctamente el enlace. En caso de que estos parámetros no sean correctos, la SMSC puede enviar un mensaje de desconexión con el cliente.

El *System_type*, es un parámetro que es opcional e indica el tipo de ESME que se está enlazando con la SMSC. Este parámetro puede ser VMS que significa *voice mail system* que en español significa sistema de correo de voz u OTA *over the air activation system* que en español significa sistema de activación al aire [1]. De ahí se especifica la versión SMPP que se está implementando, en este caso se envía la versión 3.4. Una de las utilidades que se presentan al enviar este parámetro es indicarle a la SMSC si puede utilizar o no comandos ya implementados para la transmisión de los mensajes.

Después se encuentra el parámetro *Type of number* que es aquel que se conoce como TON. Este parámetro puede tomar varios valores dependiendo del tipo de valor que el usuario haya colocado en esta sección. En este caso, el valor que se le asignó es el número cero que indica tipo de número desconocido. Se puede asignar el número uno que representa un número internacional o también el número dos que

representa un número nacional. El siguiente parámetro es el *Numbering plan indicator* el cual es conocido como NPI.

La figura 4.3 hace referencia a un paquete *submit_sm* enviado por el ESME. Para que el ESME pueda enviar este mensaje el ESME se debe conectar como *transceiver* o *receiver*, una vez enlazado está en la capacidad de enviar este tipo de paquetes en donde se pasa el mensaje corto que será visible por los usuarios en su dispositivo móvil.

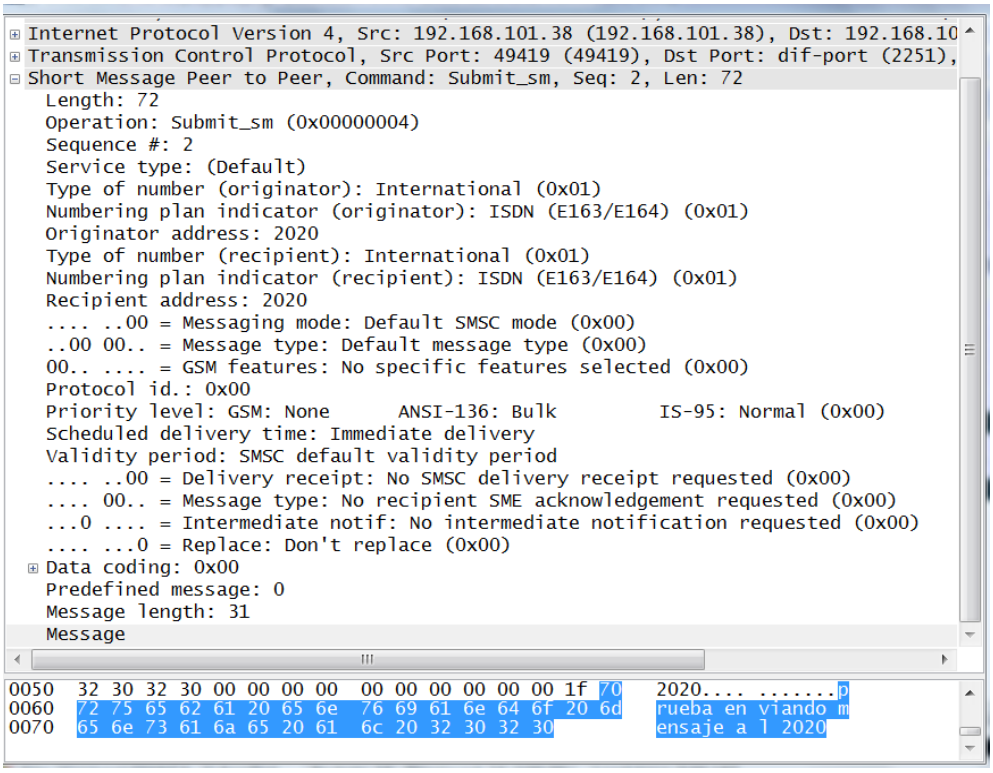


Figura 4.3 Mensaje SMPP, comando *submit_sm*

En la Figura 4.3, en primer lugar se muestra el tamaño del paquete SMPP, en este caso es de 72 bytes, este tamaño representa el mensaje que se desea transmitir y la cabecera y comandos que contiene este paquete. En seguida se muestra la operación SMPP, en este caso contiene el código 0X00000004 que representa al *submit_sm* y es el mensaje SMPP que indica al ESME que allí es donde se encuentra el mensaje corto. Después se muestra el número de secuencia, que posteriormente es utilizado para que la SMSC pueda reenviar una respuesta al ESME, es importante destacar que este número se incrementa cada vez que se envíen mensajes SMPP en una sola conexión, es decir si existen una sesión como *transmitter* y otra como *receiver*, tendrán secuencias diferentes. El siguiente parámetro corresponde a *service_type*, este parámetro indica que tipo de servicio es el que está usando el ESME para conocimiento de la SMSC y poder ejecutar alguna operación en específico, por ejemplo se puede configurar como WAP que significa protocolo de aplicación inalámbrica, VMA alerta de voz por mail entre otros. Este parámetro es opcional es decir se puede colocar un valor NULL como es el caso de este proyecto. Sin embargo, el *system_type* que se podría colocar es CMT que significa mensajería celular ya que esta es la temática del proyecto, enviar mensajes de texto cortos.

Posteriormente se muestra el TON que significa el tipo de número. En este caso en la aplicación se ha enviado el número 1 y en *wireshark* se muestra como 0X01 que representa que el número origen es tipo internacional. Se coloca también el NPI que es aquel que indica el tipo de red en la que se encuentran operando ambas entidades, generalmente en aplicaciones móviles se configura al NPI con el número 1, es decir como una red ISDN. Estos parámetros son los más importantes. Inmediatamente se muestran algunos mensajes SMPP como *replace_sm* que son propios del protocolo pero que no se implementan en este proyecto. Al final de la Figura 4.3 es donde se muestra el mensaje que el ESME envía al cliente, en la parte inferior se muestra el mensaje en código hexadecimal y en caracteres traducidos al código ASCII.

La Figura 4.4 capturada por *wireshark* representa la confirmación de un mensaje que envió el ESME a la SMSC, en otras palabras el mensaje *submit_sm_resp* es enviado por la SMSC en respuesta a un mensaje *submit_sm* que envío el ESME, cabe recalcar que la entidad externa de mensajes cortos se tuvo que haber conectado ya sea como *transmitter* o *transceiver*

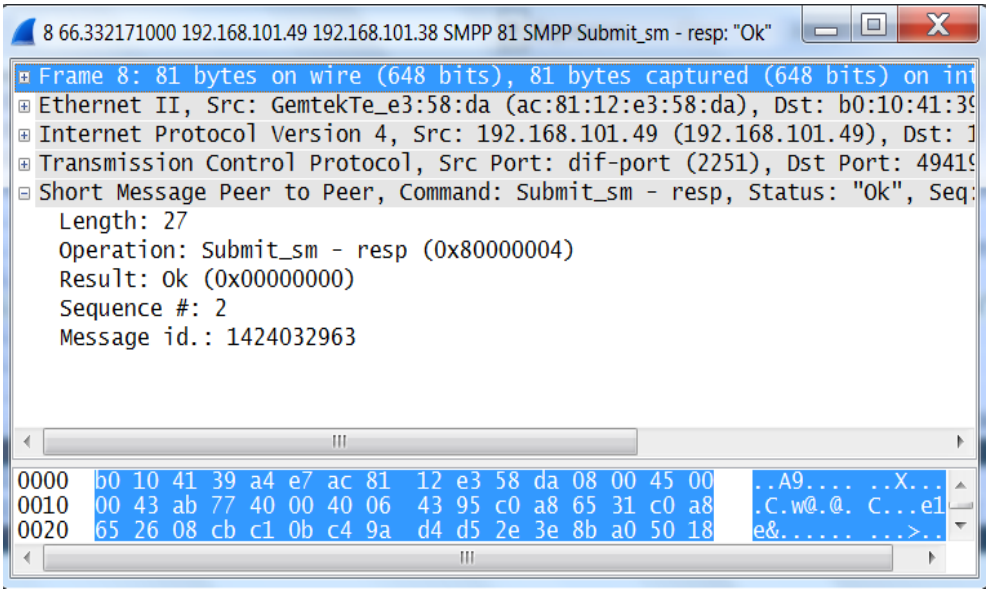
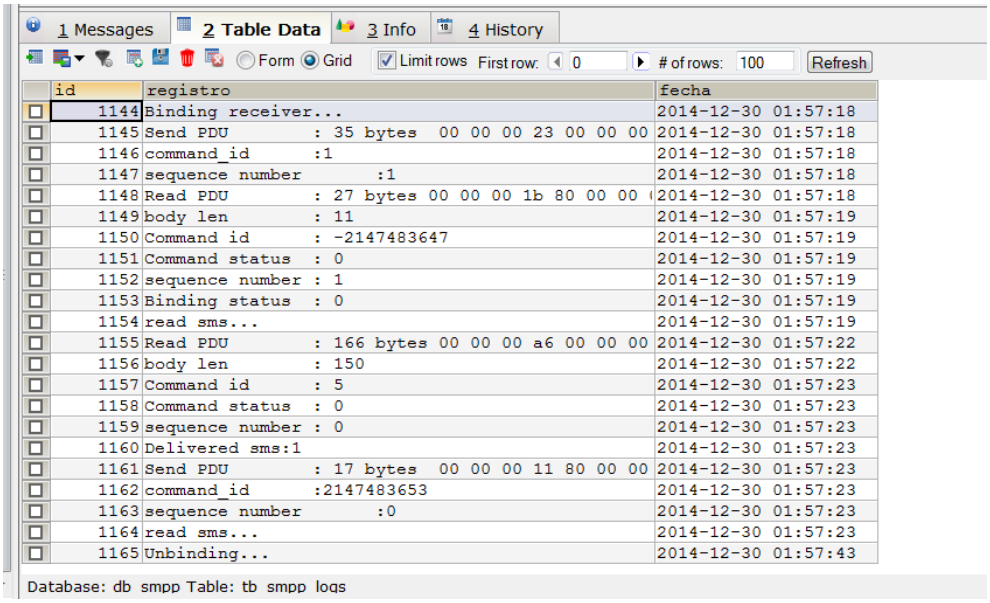


Figura 4.4 Mensaje SMPP, comando *submit_sm_resp*

En este caso la longitud del mensaje es más corta que los mensajes *submit_sm* o *deliver_sm*, apenas contiene 27 bytes en la cual solamente lleva un mensaje de confirmación y no un mensaje de texto corto. Después de la longitud se muestra que el tipo de operación SMPP que se empleó fue *submit_sm_resp* con su *comand_id* 0X80000004. Inmediatamente se muestra el resultado que envía la SMSC, en este caso es el 0X00000000 que representa ok y cuya interpretación es que el servidor recibió el mensaje del cliente sin inconvenientes. El número de secuencia que se muestra es el número dos, y este número debe ser igual al número de secuencia del mensaje *submit_sm* que envío el ESME. El objetivo de este número de secuencia es tener una referencia y asegurar que existe una respuesta

a cada mensaje enviado. Otra aplicación de este número de secuencia es que asegura que las entidades tanto cliente como servidor puedan interactuar de forma asíncrona. Y por último se muestra un *message_id* este parámetro es enviado por la SMSC como respuesta a los mensajes *submit_sm* y *deliver_sm* y puede ser usado por el ESME para ejecutar alguna operación.

Los procesos que se realizan para el envío de los mensajes cortos se guardan en la base de datos. Estos procesos implementan mensajes y comandos del propio protocolo SMPP registrando además la fecha y hora en la cual se efectúa el proceso. Como en el capítulo 3 se mencionó, los mensajes que se reciben y que se transmiten desde y hacia el ESME se guardan en la base de datos. En la Figura 4.5, se muestra una tabla en la base de datos donde se guardan los registros SMPP con cada uno de los códigos.

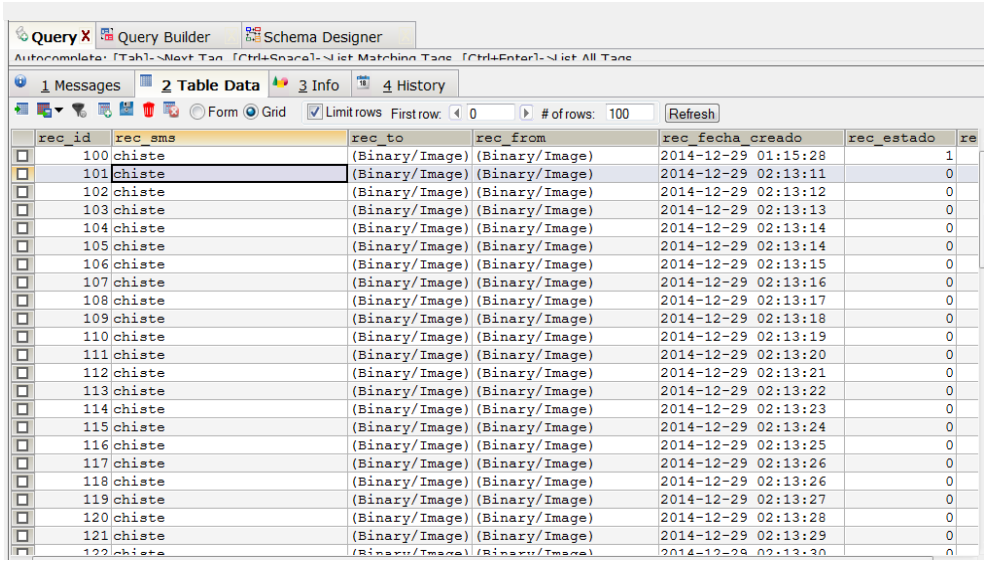


id	registro	fecha
1144	Binding receiver...	2014-12-30 01:57:18
1145	Send PDU : 35 bytes 00 00 00 23 00 00 00	2014-12-30 01:57:18
1146	command_id :1	2014-12-30 01:57:18
1147	sequence number :1	2014-12-30 01:57:18
1148	Read PDU : 27 bytes 00 00 00 1b 80 00 00	2014-12-30 01:57:18
1149	body len : 11	2014-12-30 01:57:19
1150	Command id : -2147483647	2014-12-30 01:57:19
1151	Command status : 0	2014-12-30 01:57:19
1152	sequence number : 1	2014-12-30 01:57:19
1153	Binding status : 0	2014-12-30 01:57:19
1154	read sms...	2014-12-30 01:57:19
1155	Read PDU : 166 bytes 00 00 00 a6 00 00 00	2014-12-30 01:57:22
1156	body len : 150	2014-12-30 01:57:22
1157	Command id : 5	2014-12-30 01:57:23
1158	Command status : 0	2014-12-30 01:57:23
1159	sequence number : 0	2014-12-30 01:57:23
1160	Delivered sms:1	2014-12-30 01:57:23
1161	Send PDU : 17 bytes 00 00 00 11 80 00 00	2014-12-30 01:57:23
1162	command_id :2147483653	2014-12-30 01:57:23
1163	sequence number :0	2014-12-30 01:57:23
1164	read sms...	2014-12-30 01:57:23
1165	Unbinding...	2014-12-30 01:57:43

Figura 4.5 Registros de procesos SMPP realizados desde la aplicación

En algunos registros se muestra códigos hexadecimales que representan los mensajes que se envían y se transmiten entre la SMSC y el ESME. Otro de los parámetros que se guardan es el número de secuencia y la longitud de los mensajes.

En la Figura 4.6, se muestra los mensajes recibidos, en este caso solo se muestra la palabra chiste que es una de las palabras claves que está asociado a un conjunto de mensajes para que la aplicación pueda seleccionar de manera aleatoria y responder a dicha solicitud.



rec_id	rec_sms	rec_to	rec_from	rec_fecha_creado	rec_estado
100	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 01:15:28	1
101	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:11	0
102	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:12	0
103	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:13	0
104	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:14	0
105	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:14	0
106	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:15	0
107	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:16	0
108	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:17	0
109	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:18	0
110	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:19	0
111	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:20	0
112	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:21	0
113	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:22	0
114	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:23	0
115	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:24	0
116	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:25	0
117	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:26	0
118	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:26	0
119	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:27	0
120	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:28	0
121	chiste	(Binary/Image)	(Binary/Image)	2014-12-29 02:13:29	0

Figura 4.6 Contenido de mensajes entrantes

Además de la palabra clave también se almacena el número origen y destino que tenía configurado el mensaje SMPP, asimismo se registran la fecha y hora en la cual se registró el mensaje. Cada uno de estos mensajes tienen un parámetro adicional que indica si el mensajes fue respondido o no, básicamente aquellos parámetros que contengan cero es porque no son respondidos mientras que los que tienen el número uno si lo están.

4.2 Interpretación de resultados

En las Figuras 4.3 y 4.4 donde se muestran las capturas de los datos en *wireshark* se puede demostrar que esta aplicación no se trata de una simulación, sino que claramente se puede observar que se ha

desarrollado una aplicación que implementa el protocolo SMPP para la transmisión de los datos entre el ESME y la SMSC.

Wireshark, tiene la capacidad de interpretar los mensajes SMPP para presentar el contenido que se está enviando, los parámetros que se pasan a través de la aplicación y de los comandos en texto claro y fácil de entender a quien está usando esta aplicación.

Se observa en las Figuras 4.5 y 4.6 los registros de la base de datos en la que se tiene constancia de los mensajes de texto cortos que se transmiten y también aquellos registros que representan todos los procesos que se llevan a cabo para el envío de mensajes cortos. La aplicación está diseñada para guardar los mensajes que se envían y se transmiten en texto claro específicamente los mensajes que el cliente envía o transmite y además existe un respaldo en la base de datos en donde se guardan los mensajes en código hexadecimal y el código de los comandos y parámetros SMPP. En caso de que exista algún problema con la aplicación por ejemplo alguna desconexión con estos registros guardados en la base de datos, se da apertura a realizar una auditoría y verificar que procesos se ejecutaron antes de que existiera el problema.

Los mensajes que han sido respondidos por la aplicación se marcan como correctos con una imagen de visto de color verde mientras que

los mensajes que no se han respondido, la aplicación los marca con un ícono rojo de no respondidos para que el administrador de la aplicación tome una decisión de responder o no estos mensajes.

En caso de que existan mensajes que se debieron responder automáticamente a través de la aplicación y no lo están, el administrador de la aplicación debe verificar que en los parámetros de inicialización esté configurado correctamente el envío de mensajes automático. En caso de que se haya corregido esta opción el siguiente paso es ejecutar el algoritmo que procesa los mensajes no respondidos cada cinco segundos a través del botón envío automático que se encuentra en la parte inferior izquierda de la pantalla inicial. Con esta configuración, estos mensajes se deberían responder automáticamente; en caso de que no se dé esa opción es conveniente revisar la conexión con el servidor.

Se verificó que los mensajes que el cliente solicitaba al servidor estaban realmente asociados con la palabra clave que se enviaba en el mensaje corto. Asimismo, en caso de que el cliente envíe un mensaje con la misma palabra clave, la aplicación envió un mensaje diferente al que se respondió inicialmente, ya que internamente hay un algoritmo que busca los mensajes de forma aleatoria. El administrador puede crear a través de la aplicación tipos de contenidos y los contenidos asociados a estos tipos por medio de una palabra clave sin

la necesidad de que exista alguna configuración directa por sentencias SQL en la base de datos.

Cabe recordar que para enviar mensajes se debe iniciar una sesión como *transmitter* o una sesión como *transceiver*, así también para que el ESME pueda recibir mensajes puede enlazarse como *receiver* o como *transceiver*, cualquiera de las dos opciones puede haber sido escogida teniendo la precaución de que el servidor también pueda interpretar estos tipos de enlaces. Si el servidor no puede conectarse como un *transceiver* es probable que el servidor use una versión menor a la 3.4, entonces el cliente no podrá enviar ni recibir mensajes por este enlace.

En caso de que exista algún tipo de desconexión, el administrador podrá saberlo inmediatamente ya que en la sección donde se activó el *receiver* el ícono de carga se desactivará automáticamente dando la opción de volver a activarlo; sin embargo, se puede comprobar el estado de la conexión en la opción verificar conexión.

CONCLUSIONES Y RECOMENDACIONES

Gracias al desarrollo del presente proyecto se pudo cumplir con los objetivos propuestos y llegar a las siguientes conclusiones:

1. En el desarrollo del proyecto se utilizaron varias herramientas y conceptos tanto de informática como de telecomunicaciones. Gracias a esta integración, se pudo implementar una aplicación profesional, capaz de cubrir varias necesidades en el campo de las telecomunicaciones.

2. El uso de herramientas de código abierto para la implementación del ESME sirvió para no preocuparse por el pago de licencias tanto en el sistema operativo, el motor de base de datos y de los IDE's.

3. Conforme se estuvo desarrollando el proyecto se pudo constatar que la velocidad de comunicación entre la SMSC y el ESME, así como también el enlace para la administración remota de este último, depende del ancho de

banda del enlace que poseen. En caso de implementar el ESME en redes públicas, se debe solicitar al proveedor de internet un ancho de banda que permita una conexión con una velocidad aceptable.

4. La versión 3.4 del protocolo SMPP fue suficiente para implementar un ESME robusto y eficiente, que tiene la capacidad de optimizar las conexiones con la SMSC gracias a la implementación de la sesión *transceiver*.

5. Se pudo percatar que el lenguaje SQL es compatible con el lenguaje de programación PHP. SQL permitió manejar los registros de la base de datos, y gracias a esta compatibilidad se pudo enviar los datos desde la base a la aplicación y viceversa fácilmente.

A continuación se describen las siguientes recomendaciones:

1. Como recomendación, antes de implementar cualquier solución es importante conocer e investigar cuáles son las tendencias actuales para el desarrollo de aplicaciones óptimas, para que los usuarios vean en esta solución una herramienta amigable y entretenida.

2. En caso de implementar el proyecto en un entorno corporativo y profesional, es recomendable instalar la aplicación web en un servidor robusto de alta capacidad, para brindar un mejor servicio y acceder a esta desde cualquier dispositivo conectado al internet.

3. Antes de comenzar a implementar cualquier protocolo de comunicación (no solamente el protocolo SMPP), es importante dominar cuales son los procesos y pasos que se deben seguir ya que, si no se tiene en claro estos procesos es posible que se termine desarrollando una aplicación que tenga un comportamiento no deseado.

4. Así mismo antes de implementar un protocolo de comunicación es importante estudiar cuales son las versiones y especificaciones más recientes del mismo y si es posible implementar la aplicación tomando en cuenta la última versión o una versión estable. Actualmente el protocolo SMPP se encuentra en la versión 5.0, sin embargo se implementó la versión 3.4 que sigue siendo una versión muy usada y completamente estable.

BIBLIOGRAFÍA

[1] SMPP Developers Forum, http://www.smstrade.de/pdf/smpp.pdf, fecha de consulta noviembre del 2014.

[2] A.Olmos, http://es.scribd.com/doc/77539272/Smpp-Intro#scribd, fecha consulta noviembre 2014.

[3] Luz Marina Santos Jaimes, http://dialnet.unirioja.es/descarga/articulo/4272055.pdf, fecha consulta noviembre 2014.

[4] Tecno Fanático, http://www.tecnofanatico.com/no-todo-es-windows-y-mac-os-x-las-distribuciones-de-linux-mas-interesantes-del-2014/, fecha de consulta noeviembre 2014.

[5] T.A.S Foundation, http://www.apache.org/, fecha de consulta noviembre 2014.

[6] Genbeta, http://www.genbeta.com/herramientas/apache-http-server-2-4-ya-disponible-con-mejoras-generales-de-rendimiento, fecha de consulta noviembre 2014.

[7] T.P.Group, http://php.net/downloads.php, fecha de consulta noviembre 2014.

[8] SolucionesPm, http://www.solucionespm.com/los-mejores-frameworks-de-php/, fecha de consulta noviembre 2014.

[9] Oracle, http://www.mysql.com/downloads/, fecha de consulta noviembre 2014.

[10] Genbeta, http://www.solucionespm.com/los-mejores-frameworks-de-php/, fecha de consulta noviembre 2014.

[11] T.E Foundation, https://eclipse.org/downloads/, fecha de consulta noviembre 2014.

[12] E. Foundation, https://eclipse.org/downloads/, fecha de consulta noviembre 2014.

[13] Oracle, http://www.mysql.com/products/workbench/, fecha de consulta noviembre 2014.

[14] Oracle Corporation, http://www.mysql.com/products/workbench/, fecha de consulta noviembre 2014.

[15] J.A.R Arteaga, https://sites.google.com/site/betoresendiz27/home/instalaciondeunservidorlamplinux-apache-mysql-php, fecha de consulta noviembre 2014.

[16] Intercom, http://www.mailxmail.com/curso-php-mysql-aplicaciones-web-1/programacion-cliente-servidor, fecha de consulta noviembre 2014.

[17] T. Lea, http://www.tonylea.com/2011/intro-to-phonegap-com-build-native-mobile-apps-using-htmlcssjavascript/, fecha de consulta noviembre 2014

[18] Ellislab, http://www.codeigniter.com/download, fecha de consulta noviembre 2014.

[19] E. Martínez, http://www.tutorialesenvideo.net/codeigniter/, fecha de consulta noviembre 2014.

[20] E. Bahit, http://www.monografias.com/trabajos89/poo-y-mvc-php/poo-y-mvc-php2.shtml, fecha de consulta noviembre 2014.

[21] T. J. Foundation, http://jquery.com/, fecha de consulta noviembre 2014.

[22] T. Administrator, http://toyhouse.cc/profiles/blogs/learning-notes-of-introduction-to-database-sql, fecha de consulta noviembre 2014.

[23] Paladin, http://es.sourceforge.jp/projects/sfnet_php-smppv3-4/downloads/smpp_transceiver.php/, fecha de consulta noviembre 2014.